

bliss
Internet-Draft
Intended status: Standards Track
Expires: August 15, 2013

D. Worley
Ariadne Internet Services, Inc.
M. Huelsemann
R. Jesske
Deutsche Telekom
D. Alexeitsev
TeleFLASH
February 11, 2013

Call Completion for Session Initiation Protocol (SIP)
draft-ietf-bliss-call-completion-19

Abstract

The call completion feature defined in this specification allows the caller of a failed call to be notified when the callee becomes available to receive a call.

For the realization of a basic solution without queuing, this document references the usage of the dialog event package (RFC 4235) that is described as 'automatic redial' in the SIP Service Examples (RFC 5359).

For the realization of a more comprehensive solution with queuing, this document introduces an architecture for implementing these features in the Session Initiation Protocol where "call completion" implementations associated with the caller's and callee's endpoints cooperate to place the caller's request for call completion into a queue at the callee's endpoint, and when a caller's request is ready to be serviced, re-attempt of the original, failed call is made.

The architecture is designed to interoperate well with existing call-completion solutions in other networks.

Status of this Memo

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

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

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

material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 15, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	5
2. Requirements terminology	5
3. Terminology	5
4. Solution	7
4.1. Call-completion architecture	8
4.2. Call-completion procedures	9
4.3. Automatic redial as a fallback	12
4.4. Differences from SS7	12
5. Call-completion queue model	13
6. Caller's agent behavior	14
6.1. Receiving the CC possible indication	14
6.2. Subscribing to CC	14
6.3. Receiving a CC recall notification	16
6.4. Initiating a CC call	16
6.5. Suspending CC	16
6.6. Resuming CC	17
7. Callee's monitor behavior	17
7.1. Sending the CC possible indication	17
7.2. Receiving a CC subscription	18
7.3. Sending a CC notification	19
7.4. Receiving a CC call	20
7.5. Receiving a CC suspension	20
7.6. Receiving a CC resumption	21
8. Examples	21
9. Call-completion event package	26
9.1. Event package name	26
9.2. Event package parameters	26
9.3. SUBSCRIBE bodies	26
9.4. Subscribe duration	27
9.5. NOTIFY bodies	27
9.6. Subscriber generation of SUBSCRIBE requests	28
9.7. Notifier processing of SUBSCRIBE requests	28
9.8. Notifier generation of NOTIFY requests	28
9.9. Subscriber processing of NOTIFY requests	29
9.10. Handling of forked requests	29
9.11. Rate of notifications	29
9.12. State agents	30
10. Call-completion information format	30
10.1. Call-completion status	30
10.2. Call-completion service-retention indication	30
10.3. Call-completion URI	31
11. Security considerations	31
12. IANA considerations	32
12.1. SIP event package registration for call-completion	33
12.2. MIME registration for application/call-completion	33
12.3. SIP/SIPS URI parameter 'm'	34

12.4. Call-Completion purpose parameter value	34
12.5. 'm' header parameter for Call-Info	35
13. Acknowledgements	35
14. References	35
14.1. Normative References	35
14.2. Informative References	36
Appendix A. Example Caller's Agent	37
Appendix B. Example Callee's Monitor	37
Authors' Addresses	38

1. Introduction

The call completion (CC) feature allows the caller of a failed call to have the call completed without having to make a new call attempt while guessing when the callee becomes available. When the caller requests the use of the CC feature, the callee will be monitored for its availability. When the callee becomes available the callee will be given a certain timeframe for initiating a call. If the callee does not initiate a new call within this timeframe, then the caller will be recalled. When the caller accepts the CC recall then a CC call to the callee will automatically start. If several callers have requested the CC feature on the same callee, they will be recalled in a predefined order, which is usually the order in which they have requested the CC feature.

This draft defines the following CC features:

Call Completion on Busy Subscriber (CCBS): The callee is busy. The caller is recalled after the callee is not busy any longer.

Call Completion on No Reply (CCNR): The callee does not answer the call. The caller is recalled after the callee has completed a new call.

Call Completion on Not Logged-in (CCNL): The callee is not registered. The caller is recalled after the callee has registered again.

2. Requirements terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This document uses terms from [RFC3261].

3. Terminology

For the purpose of this service, we provide the following terminology:

Callee: a destination of the original call, and a target of the CC call.

Caller: The initiator of the original call and the CC request. The user on whose behalf the CC call is made.

Callee's monitor: a logical component which implements the call-completion queue for destination user(s)/UA(s), and performs the associated tasks, including sending CC recall events, analogous to the destination local exchange's role in SS7 CC.

Caller's agent: a logical component which makes CC requests and responds to CC recall events on behalf of originating user(s)/UA(s), analogous to the originating local exchange's role in SS7 CC.

CC, or call completion: a service which allows a caller who failed to reach a desired callee to be notified when the callee becomes available to receive a call.

CC activation: the indication by the caller to the caller's agent that the caller desires CC for a failed original call; this implies an indication transmitted from the caller's agent to the callee's monitor of the desire for CC processing.

CCBS, or Call Completion on Busy Subscriber: a CC service when the initial failure was that the destination UA was busy.

CCNR, or Call Completion on No Reply: a CC service when the initial failure was that the destination UA did not answer.

CCNL, or Call Completion on Not Logged-in: a CC service when the initial failure was that the destination UA was not registered.

CC call: a call from the caller to the callee, triggered by the CC service when it has determined that the callee is available.

CC indicator: an indication in the CC call INVITE used to prioritize the call at the destination.

CC possible indication: the data in responses to the INVITE of the original call which indicate that CC is available for the call.

CC recall: the action of the callee's monitor selecting a particular CC request for initiation of a CC call, resulting in an indication from the caller's agent to the caller that it is now possible to initiate a CC call.

CC recall events: event notifications of event package "call-completion", sent by the callee's monitor to the caller's agent to inform it of the status of its CC request.

CC recall timer: maximum time the callee's monitor will wait for the caller's response to a CC recall.

CC request: the entry in the callee's monitor queue representing the caller's request for CC processing, that is, the caller's call-completion subscription.

CC service duration timer: maximum time a CC request may remain active within the network.

CC queue: a buffer at the callee's monitor which stores incoming calls which are target for call completion. Note: This buffer may or may not be organized as a queue. The use of the term "queue" is by analogy with SS7 usage.

CCE, or call-completion entity: the representation of a CC request, or equivalently, an existing call-completion subscription within a callee's monitor's queue

Failed call: a call which does not reach a desired callee, from the caller's point of view. Note that a failed call may be successful from the SIP point of view; e.g., if the call reached the callee's voicemail, but the caller desired to speak to the callee in person, the INVITE receives a 200 response, but the caller considers the call to have failed.

Notifier: the user agent that generates NOTIFY requests for the purpose of notifying subscribers of the callee's availability; for the CC service this is task of the callee's monitor.

Original call: the initial call which failed to reach a desired destination.

Retain option: a characteristic of the CC service; if supported, CC calls which again encounter a busy callee will not be queued again, but the position of the caller's entry in the queue is retained. Note that SIP CC always operates with the retain option active; a failed CC call does not cause the CC request to lose its position in the queue.

Subscriber: the user agent that receives NOTIFY requests with information of the callee's availability; for the CC service this is task of the caller's agent.

Suspended CC request: a CC request which is temporarily not to be selected for CC recall.

4. Solution

4.1. Call-completion architecture

The call-completion architecture augments each caller's UA (or UAC) wishing to use the call-completion features with a "call-completion agent" (also written as "caller's agent").

It augments each callee's UA (or UAS) wishing to be the target of the call-completion features with a "call-completion monitor" (also written as "callee's monitor").

The caller's agent and callee's monitor functions can be integrated into the respective UAs, be independent end-systems, or be provided by centralized application servers. The two functions, though associated with the two UAs (caller and callee), also may be provided as services by the endpoints' home proxies or by other network elements. Though it is expected that a UA that implements call completion will have both functions so that it can participate in call completion as both caller and callee, the two functions are independent of each other.

A caller's agent may service more than one UA as a collective group if a caller or population of users will be shared between the UAs, and especially if the UAs share an AOR.

The caller's agent monitors calls made from the caller's UA(s) in order to determine their destinations and (potentially) their final response statuses, and the Call-Info header fields of provisional and final responses for to invoke the call completion feature.

A callee's monitor may service more than one UA as a collective group if a callee or population of users will be shared between the UAs, and especially if the UAs share an AOR. The callee's monitor may supply the callee's UAS(s) with Call-Info header field values for provisional and final responses.

The callee's monitor also instantiates a presence server used to monitor caller's availability for CC recall.

The callees using the UA(s) may be able to indicate to the callee's monitor when they wish to receive CC calls.

In order to allow flexibility and innovation, most of the interaction between the caller's agent, the caller-user(s) and the caller's UA(s) is out of the scope of this document. Similarly, most of the interaction between the callee's monitor, the callee(s) and the callee's UA(s) is out of the scope of this document, as is also the policy by which the callee's monitor arbitrates between multiple call-completion requests.

The caller's agent must be capable of performing a number of functions relative to the UA(s). The method by which it does so is outside the scope of this document, but an example method is described in Appendix A. The callee's monitor must be capable of performing a number of functions relative to the UA(s). The method by which it does so is outside the scope of this document, but an example method is described in Appendix B.

As a proof of concept, simple caller's agents and callee's monitors can be devised that interact with users and UAs entirely through standard SIP mechanisms [RFC6665], [RFC4235] and [RFC3515], as described in the Appendixes.

The callers using the UA(s) can indicate to the caller's agent when they wish to avail themselves of CC for a recently-made call which the callers determined to unsuccessful. The caller's agent monitors the status of the caller's UA(s) to determine when they are available to be used for a CC recall. The caller's agent can communicate to the caller's UA(s) that a CC recall is in progress and to inquire if the relevant caller is available for the CC recall.

The callee's monitor may utilize several methods to monitor the status of the callee's UA(s) and/or their users for availability to receive a CC call. This can be achieved through monitoring calls made to the callee's UA(s) to determine their caller's and their final response' statuses. And in a system with rich presence information, the presence information may directly provide this status. In a more restricted system, this determination can depend on the mode of the CC call in question, which is provided by the URI 'm' parameter. E.g., a UA is considered available for CCBS ("m=BS") when it is not busy, but a UA is considered available for CCNR ("m=NR") when it becomes not busy after being busy with an established call.

The callee's monitor maintains information about the set of INVITES received by the callee's UA(s) considered unsuccessful by the caller. In practice, the callee's monitor may remove knowledge about an incoming dialog from its set if local policy at the callee's monitor establishes that the dialog is no longer eligible for CC activations.

4.2. Call-completion procedures

The caller's UA sends an INVITE to a request URI. One or more forks of this request reach one or more of the callee's UAs. If the call-completion feature is available, the callee's monitor (note there can be a monitor for each of the callee's UAs) inserts a Call-Info header field with its URI and with "purpose=call-completion" in appropriate non-100 provisional or final response to the initial INVITE and

forwards them to the caller. The provisional response SHOULD be sent reliably, if the INVITE contained a Supported header field with the option tag 100rel. On receipt of a non-100 provisional or a final response with the indication that the call-completion feature is available, the calling user can invoke the CC feature.

The caller indicates to the caller's agent that he wishes to invoke call-completion services on the recent call. Note that from the SIP point of view, the INVITE may have been successful, but from the user's point of view, the call may have been unsuccessful. E.g., the call may have connected to the callee's voicemail, which would return a 200 status to the INVITE but from the caller's point of view is "no reply".

In order to receive information necessary for the caller to complete the call at the callee, the caller's agent subscribes to the call-completion event package at the callee's monitor.

The possibility of the caller to complete the call at the callee is also known as the call-completion state (cc-state) of the caller. The cc-states comprehend the values 'queued' and 'ready' (for call-completion).

In order to receive information from all destinations where the callee will be reachable, the caller's agent sends a SUBSCRIBE request for the call-completion event package to the original destination URI of the call and to all known callee's monitor URIs (which are provided by Call-Info header fields in provisional and final responses to the INVITE). The callee's monitor uses the subscription as an indication that caller is interested in using the CC feature with regard to the specified callee. The callee's monitor keeps a list or queue of the caller's agent's subscriptions, representing the requests from the caller's agent to the callee's monitors for call-completion services. These subscriptions are created, refreshed, and terminated according to the procedures of [RFC6665].

Upon receiving a SUBSCRIBE request from the caller's agent, the callee's monitor instantiates a presence state for the caller's UA which can be modified by the caller's UA to indicate its availability for CC call. The status at the presence upon instantiation is "open".

When the callee's monitor determines the callee and/or callee's UA available for a CC call, it selects a caller to execute the CC call and sends a call-completion event update (''cc-state: ready'') via a NOTIFY request to the selected caller's agent's subscription, telling it to begin the CC call to the callee's UA.

When the caller's agent receives this update, it initiates a CC recall by calling the caller's UA, and then starts the CC call to the callee's UA, using 3rd party call control procedures in accordance with [RFC3725]. The caller's agent can also check by other means whether the caller is available to initiate the CC call to the callee's UA. If the caller is available, the caller's agent directs the caller's UA to initiate the CC call to the callee's UA.

The caller's agent marks the CC call as such by adding a specific SIP URI parameter to the Request-URI, so it can be given precedence by the callee's monitor in reaching the callee's UA.

If the caller is not available on the receipt of the "ready for recall" notification, the caller's agent suspends the CC request at the callee's monitor by sending a PUBLISH request containing presence information to the callee's monitor's presence server, informing about the presence status 'closed'. Once the caller becomes available for a CC call again, the caller's agent resumes the CC request by sending another PUBLISH request to the callee's monitor, informing about the presence status 'open'.

On the receipt of the suspension request, the callee's monitor performs the monitoring for the next non-suspended CC request in the queue. On the receipt of the resume from the previously suspended caller's agent that was at the top of the queue, the callee's monitor performs the callee monitoring for this caller's agent.

When the CC call fails there are two possible options: the CC feature has to be activated again by caller's agent subscribing to callee's monitor, or CC remains activated and the original CC request retains its position in the queue if retain option is supported.

The retain option (see section 3) determines the callee's monitor's behavior when a CC call fails. If the retain option is supported, CC remains activated and the original CC request retains its position in the queue. Otherwise the CC feature is deactivated, and the caller's agent would have to subscribe again to reactivate it.

A monitor that supports the retain option provides the cc-service-retention header in its call-completion events. A caller's agent that also supports the retain option uses the presence of this header to know not to generate a new CC request after a failed CC call.

Monitors not supporting the retain option do not provide the cc-service-retention header. A failed CC call causes the CC request to be deleted from the queue, and these monitors will terminate the corresponding caller's agent's subscription to inform that agent that its CC request is no longer in the queue. A caller's agent that does

not support the retain option can also terminate its subscription when a CC call fails, so it is possible that both the caller's agent and the monitor may be signalling the termination of the subscription concurrently. This is a normal SIP Events [RFC6665] scenario. After the subscription is terminated, the caller's agent may create a new subscription (as described in section 6.2) to reactivate the CC feature for the original call.

4.3. Automatic redial as a fallback

Automatic redial is a simple end-to-end design. An automatic redial scenario is described in [RFC5359], section 2.17. This solution is based on the usage of the dialog event package. When the callee is busy when the call arrives, the caller subscribes to the callee's call state. The callee's UA sends a notification when the callee's call state changes. This means the caller is also notified when the callee's call state changes to 'terminated'. The caller is alerted, then the caller's UA starts a call establishment to the callee again. If several callers have subscribed to a busy callee's call state, they will be notified at the same time that the call state has changed to 'terminated'. The problem of this solution is, that it might happen that several recalls are started at the same time. This means it is a heuristic approach with no guarantee of success.

There is no interaction between call completion and automatic redial, as there is a difference in the behavior of the callee's monitor and the caller when using the dialog event package for receiving dialog information or for aggregating a call completion state.

4.4. Differences from SS7

SIP call completion differs in some ways from the CCBS and CCNR features of SS7 (which is used in the PSTN). For ease of understanding, we enumerate some of the differences here.

As there is no equivalent to the forking mechanism in SS7, in the PSTN calls can be clearly differentiated between successful or unsuccessful. Due to the complex forking situations that are possible in SIP, a call may "fail" from the point of view of the user and yet have a "success" response from SIP's point of view. (This can happen even in simple situations: e.g., a call to a busy user that fails over to his voicemail receives a SIP success response, even though the caller may consider it "busy subscriber".) Thus, the caller must be able to invoke call completion even when the original call appeared to succeed. To support this, the caller's agent must record successful calls as well as unsuccessful calls.

In SIP, only the caller's UA or service system on the originating

side and the callee's UA or service system on the terminating side need to support call completion for call completion to work successfully between the UAs. Intermediate SIP systems (proxies or B2BUAs) do not need to implement call completion; they only need to be transparent to the usual range of SIP messages. In the PSTN also intermediate nodes like media gateway controllers have to implement the call completion service.

5. Call-completion queue model

The callee's monitor manages CC for a single URI. This URI is likely to be a published AOR, or more likely "non-voicemail AOR", but it may be as narrowly scoped as a single UA's contact URI. The callee's monitor manages a dynamic set of call-completion entities (called "CCEs") which represent CC requests, or equivalently, the existing incoming call-completion subscriptions. This set is also called a queue, because a queue data structure often aids in implementing the callee's monitor's policies for selecting CCEs for CC recall.

Each CCE has an availability state, determined through caller's presence status at the callee's monitor. A presence status of "open" represents CCE's availability state of 'available' and a presence status of "closed" represents CCE's availability state of 'unavailable'.

Each CCE has a recall state which is visible via subscriptions. The recall state is either "queued" or "ready".

Each CCE carries the From URI of the SUBSCRIBE request that caused its creation.

CC subscriptions arrive at the callee's monitor by addressing the URIs the callee's monitor returns in Call-Info header fields. The request URI of the SUBSCRIBE request determines the queue to which the resulting CCE is added. The resulting subscription reports the status of the queue. The base event data is the status of all the CCEs in the queue, but the data returned by each subscription is filtered to report only the status of that subscription's CCE. (Further standardization may define means for obtaining more comprehensive information about a queue.)

When a CCE is created, it is given the availability state "available" and recall state "queued".

When the callee's monitor receives PIDF bodies [RFC3863] via PUBLISH requests [RFC3903], these PUBLISH requests are expected to be sent by subscribers to indirectly suspend and resume their CC requests by

modifying its CCE availability state. A CCE is identified by the request-URI (if it was taken from a call-completion event notification which identifies the CCE) or the From URI of the request (matching the From URI recorded in the CCE). Receipt of a PUBLISH with 'status' of 'open' sets the availability state of the CCE to 'available' (resume); 'status' of 'closed' sets the availability state of the CCE to 'not-available' (suspend).

A CC request is eligible for recall only when its CCE's availability state is "available" and the "m" value of the CCE also indicates an available state. The callee's monitor MUST NOT select for recall any CC requests that fail to meet those criteria. Within that constraint, the callee's monitor's selections are determined by its local policy. Often, a callee's monitor will choose the acceptable CCE that has been in the queue the longest. When the callee's monitor has selected a CCE for recall, it changes the CCE's recall state from 'queued' to 'ready', which triggers a notification on the CCE's subscription.

If a selected subscriber then suspends its request by sending a PUBLISH with the presence status 'closed', the CCE becomes not-available, and the callee's monitor changes the CCE's recall state to 'queued'. This may cause another CCE (e.g., that has been in the queue for less time) to be selected for recall.

The caller's presence status at the callee's monitor is terminated when the caller completes its CC call or when the caller's agent's subscription at the callee's monitor is terminated.

6. Caller's agent behavior

6.1. Receiving the CC possible indication

The caller's agent MUST record the From URI and SHOULD record the final request status that the caller's UA received along with the contents of Call-Info header fields of provisional and final responses.

Note that receiving a CC possible indication also depends on the aggregation of final responses by proxies, in case of 4xx responses some 4xx responses are more likely to be sent to the caller.

6.2. Subscribing to CC

For CC activation the caller's agent MUST send a SUBSCRIBE to all known callee's monitor URIs. A callee's monitor URI may be provided in the Call-Info header field in provisional and final responses to

the INVITE sent back by the callee's monitor(s). Additionally, the caller's agent SHOULD include the original request-URI that it sent the original INVITE to, in its set of callee's monitor URIs, when it is unclear if the call has forked to additional callees whose responses the caller has not seen. A SUBSCRIBE to the original request-URI alone is used in cases where the caller's agent has not received or does not remember any callee's monitor URI. The caller's agent SHOULD add an 'm' parameter to these URIs in order to indicate the desired call-completion proceeding at the callee's monitor. The 'm' parameter SHOULD have the value of the 'm' parameter received in the Call-Info header field of the responses to the original INVITE.

To minimize redundant subscriptions, these SUBSCRIBES SHOULD be presented as forks of the same transaction as defined by section 8.2.2.2 of [RFC3261], if the caller's agent is capable of doing so.

The agent MUST NOT maintain more than one CC request for a single caller and directed to a single original destination URI. If a caller requests CC a second time for the same destination URI, the agent MUST consolidate the new request with the existing CC request by either reusing the existing CC subscriptions or terminating and then recreating them. For this purpose, equality of callers is determined by comparing caller's AORs and equality of destination URIs is determined by comparing them per [RFC3261] section 19.1.4.

When generating these SUBSCRIBES, the From URI MUST be the caller's AOR. The To URI SHOULD be the destination URI of the original call (if the agent knows that and can insert it into the To header), and otherwise MUST be the request-URI of the SUBSCRIBE.

The SUBSCRIBE SHOULD have header fields to optimize its routing. In particular, it SHOULD contain "Request-Disposition: parallel", and an Accept-Contact header field to eliminate callee UAs that are not acceptable to the caller.

The caller's agent MUST be prepared to receive multiple responses for multiple forks of the SUBSCRIBE and to have multiple subscriptions established. The caller's agent must also be prepared to have the SUBSCRIBE fail, in which case, CC cannot be invoked for this original call.

If the caller's agent no longer wants to initiate the CC call (e.g., because the caller has deactivated CC), the caller's agent terminates the subscription in accordance with [RFC6665] or suspends the subscription(s) as specified in subclause 6.5.

6.3. Receiving a CC recall notification

When receiving a NOTIFY with the cc-state set to 'ready', the caller's agent SHOULD suspend all other subscriptions to CC, by following the step in section 6.5, in order to prevent any other CC requests from this caller to receive CC recalls. The caller's agent starts the CC recall to the caller by confirming that the caller would be able to initiate a CC call, e.g. by calling the caller's UA(s).

6.4. Initiating a CC call

If the caller is available for the CC call and willing to initiate the CC call, the caller's agent causes the caller's UA to generate a new INVITE towards the callee. The caller's UA MAY add a 'm' URI parameter with the value of the 'm' parameter received in Call-Info header in the response to original INVITE, in order to specify his preferences in CC processing and to prioritize the CC call. The INVITE SHOULD be addressed to the URI specified in the cc-URI of the NOTIFY, or if not available it SHOULD use the URI in the Call-Info header field of the response to the original INVITE, or if not available it MAY use the request-URI of the original INVITE, if this URI was recorded. Note that the latter choice may not provide ideal routing, but in simple cases it is likely to reach the desired callee/callee's monitor.

6.5. Suspending CC

If the caller is not available for the CC recall, the CC request SHALL be suspended by the caller's agent until the caller becomes available again, or if the conditions relevant to the caller's agent's local policy for suspensions have changed. To suspend the CC request, the caller's agent SHALL publish the caller's presence state by sending a PUBLISH request to each callee's monitor where the presence server for CC resides in accordance with the procedures described in [RFC3903], giving the PIDF state 'closed' for the caller's identity as presentity. The PUBLISH request SHOULD contain an Expires header field with a value that corresponds to the current value of the remaining CC subscription duration.

Each PUBLISH SHOULD be sent to the CC URI as received in the NOTIFY, or within the corresponding SUBSCRIBE dialog, or if that is not possible, to the corresponding callee's monitor URI received in the Call-Info header field of the NOTIFY, or if one is not available, the Contact address of the subscription.

6.6. Resuming CC

When the caller is no longer busy, or if the conditions relevant to the caller's agent's suspension policy have changed, then the CC request SHALL be resumed by the caller's agent. To resume a CC request, the caller's agent SHALL publish the Caller's presence state by sending a PUBLISH request to each callee's monitor a PUBLISH request in accordance with the procedures described in [RFC3903] , informing about the PIDF state 'open' but otherwise be constructed as same as the suspend PUBLISH request. These PUBLISH requests are sent to presence server that are instantiated at a CC monitor.

In the case where the caller's agent has sent several CC suspension requests to different callee's monitors and the caller becomes available again, as determined by the caller's agent's local policy about resumption the caller's agent MAY send a PUBLISH to resume a CC request to each callee's monitor for which there is a suspended CC request. Note that the caller's agent's policy about resumption may prescribe a manual resumption and thus a suspended CC request should not be automatically resumed.

7. Callee's monitor behavior

7.1. Sending the CC possible indication

The callee's monitor MUST record the From URI and MAY record the final request status(es) returned by the callee's UA(s).

If the callee's monitor wants to enable the caller to make use of the CC service, it MUST insert a Call-Info header field with "purpose=call-completion" in the final response message (e.g. in a 486 to enable call-completion due to busy subscriber) and at least one non-100 provisional response message (e.g. in a 180 due to no response) to the initial INVITE when forwarding it to the caller. The non-100 provisional response message SHOULD be sent reliably if the INVITE contained a Supported header field with the option tag 100rel. The Call-Info header field values defined in this specification positively indicates that CC is available for the failed fork of the call.

The callee's monitor SHOULD insert a URI in the Call-Info header field where the caller's agent should subscribe for call-completion. Ideally, it is a globally-routable URI [RFC5627] for the callee's monitor. In practice, it may be the callee's AOR, and the SUBSCRIBE will be routed to the callee's monitor only because it specifies "Event: call-completion".

In order to enable call-completion, the Call-Info header field MUST be set up according to the following scheme:

Call-Info:monitor-URI;purpose=call-completion;m=XX

The 'm' parameter defines the "mode" of call completion. The "m=NR" parameter indicates that it failed due to lack of response, the "m=BS" parameter indicates that it failed due to busy subscriber, and the "m=NL" parameter indicates that it failed due to non registered subscriber (no devices are registered for the AoR contacted). The 'm' parameter is useful for PSTN interworking and assessing presence information in the callee's monitor. It is possible that other values will be defined in future. It is also allowed to omit the 'm' parameter entirely. Implementations MUST accept CC operations in which the 'm' parameter is missing or has an unknown value, and execute them at its best in their environment (which is likely to be a degraded service, especially when interoperating with SS7).

7.2. Receiving a CC subscription

The callee's monitor MUST be prepared to receive SUBSCRIBES for the call-completion event package directed to the URIs of UA(s) that it is servicing and any URIs that the callee's monitor provides in Call-Info header fields. The SUBSCRIBES MUST be processed in accordance with the procedures defined in [RFC6665].

The callee's monitor(s) that receive the SUBSCRIBE establish subscriptions. These subscriptions represent the caller's agent's request for call-completion services.

If the monitor receives two or more SUBSCRIBES that have the same Call-Id header field value and the monitor considers the request-URIs of the received SUBSCRIBES to request the status of the same set of UAs, then they are redundant forks of one SUBSCRIBE request, and the monitor SHOULD reject all but one of the requests with 482 (Merged Request) responses.

The monitor MAY determine that an incoming CC SUBSCRIBE is a duplicate of an existing CC subscription if: (1) the Call-Id header field values are different, (2) the From URIs (i.e., the caller's AORs) are the same (per [RFC3261] section 19.1.4), (3) the To URIs (which should be the request-URI of the original call) have the same user and hostpart components, and (4) the monitor considers the request-URIs of the received SUBSCRIBES to request the status of the same set of UAs.

If the monitor determines that a new subscription is a duplicate of an existing subscription, it MAY terminate the existing subscription

in accordance with the procedures defined in [RFC6665]. In any case, it MUST establish the new subscription.

The callee's monitor may apply restrictions as to which caller's agents may subscribe.

The continuation of the caller's agent's subscription indicates to the callee's monitor that the caller's agent is prepared to initiate the CC call if it is selected for the 'ready' state. If the callee's monitor becomes aware of a subscription which cannot be selected for a CC recall, it SHOULD terminate the subscription in accordance with [RFC6665].

7.3. Sending a CC notification

The call-completion event package returns various information to the caller's agent, but the vital datum it contains is the cc-state of the caller's agent's CC request in the CC queue, which in the beginning is 'queued'. When the cc-state of the agent's request changes, the callee's monitor MUST send a NOTIFY for a call-completion event to the caller's agent. The notification SHOULD also contain a URI which can be used for suspension requests. Ideally, it is a globally-routable URI [RFC5627] for the callee's monitor. In practice, it may be the callee's AOR, and the SUBSCRIBE will be routed to the callee's monitor only because it specifies "Event: call-completion".

The call-completion event package provides limited information about the callee's monitor's policy. In particular, like in the PSTN, the "cc-service-retention" datum gives an indication of the "service retention" attribute, which indicates whether the CC request can be continued to a later time if the CC call fails due to the callee's UA(s) being busy. If the callee's monitor supports the service-retention option, the callee's monitor SHOULD include the cc-service-retention parameter.

The callee's monitor has a policy regarding when and how it selects CC requests for the recall. This policy may take into account the type of the requests (e. g. CCNR vs. CCBS), the state of the callee's UA(s), the order in which the CC requests arrived, the length of time the CC requests have been active, and any previous attempts of CC activations for the same original call. Usually the callee's monitor will choose only one CC request for the recall at a time, but if the callee's UA(s) can support multiple calls, it may choose more than one. Usually the callee's monitor will choose the oldest active request.

When the callee's monitor changes the state datum for the chosen

subscription from "queued" to "ready", the callee's monitor MUST send a NOTIFY for the caller's agent's subscription with the cc-state set to 'ready' (recall notification). The NOTIFY SHOULD also contain in the cc-URI a URI to be used in the CC call. In practice, this may be the AOR of the callee.

Upon sending the recall notification the callee's monitor MUST start a recall timer. It is RECOMMENDED to use a value between 10 and 20 seconds, which corresponds to the recommendation for the call completion services in ETSI [ETS300.356-18] and ITU-T [ITU-T.Q.733].

7.4. Receiving a CC call

The callee's UA(s) and the callee's monitor may give the CC call precedence over non-CC calls by evaluating the presence of the 'm' URI parameter and the From header of the INVITE request. The callee's monitor supervises the receiving of the CC call. Upon arrival of the CC call the recall timer MUST be stopped. If the CC call does not arrive at the callee's UA(s) before the expiry of the recall timer, the callee's monitor SHOULD stop processing the recall and change the value of the cc-state datum to "queued" if it supports the retain option and terminate the subscription along with the queue if it doesn't support the retain option. Similarly, if the CC call is not accepted, the callee's monitor will stop the CC recall processing. Depending on its policy, the same original call may be selected again for a CC recall at a later time. If the CC call succeeds, the callee's monitor MUST terminate the relevant subscription in accordance with [RFC6665], and MUST remove any associated presence event state used for suspend and resume for the caller of the CC call.

Once the CC call has been terminated, successfully or unsuccessfully, the callee's monitor's policy MAY select another CC request for a recall according to the callee's monitor's policy. Note that according to the callee's monitor's policy several recalls may be processed at the same time.

7.5. Receiving a CC suspension

The monitor may receive PUBLISH requests to suspend CC requests from caller's agent as described in section 6.5. The PUBLISH requests may be received via the URI it manages, any URI that it inserts into a Call-Info header, any contact URI it uses as a notifier for "call-completion" events, or any URI it returns as the "URI" line of the call-completion event packages.

The receipt of the PUBLISH request initiates a presence event state for the caller's identity at the presence server functionality of the

callee's monitor as specified in [RFC3903] , together with a logical presence server if this has not been done before for another call.

Note: The presence server may initiate a presence event state for the caller's identity at the receipt of SUBSCRIBE request as well, dependent on the implementation.

The monitor SHOULD identify the addressed CCE by the request-URI of the PUBLISH request, or if that is not possible, by the From URI.

If the processing of a CC request results in suspending that CC request by receiving a PUBLISH request from caller's agent as described in section 6.5, the callee's monitor MUST stop the recall timer and MUST ensure that the request is set to a 'queued' state, and then the callee's monitor MUST attempt to process another CC request in the queue according to the callee's monitor's local policy.

7.6. Receiving a CC resumption

When a CC request becomes resumed by receiving a PUBLISH request from caller's agent as described in section 6.6, the presence event state for the caller's identity at the presence server functionality of the CC monitor MUST be modified as described in [RFC3903]. If the callee is not busy and there is no entry in the CC queue which is currently being processed, the callee's monitor MUST process the queue as described in section 7.3 above.

8. Examples

A basic call flow, with only the most significant messages of a call-completion activation and invocation shown, is as follows (please note this is an example and there may be variations in the failure responses):

Caller sip:123@a.com	Callee sip:456@b.com
INVITE sip:456@b.com From: sip:123@a.com	[original call]
----->	
487 Call-Info:<sip:456@z.b.com>;purpose=call-completion;m=NR	
<-----	
SUBSCRIBE sip:456@z.b.com;m=NR From: sip:123@a.com Contact: sip:123@y.a.com Request-Disposition: parallel Call-Id: abcd-efgh Event: call-completion	[initial SUBSCRIBE]
----->	
200	
<-----	
NOTIFY sip:123@y.a.com Body: status: queued	[initial NOTIFY]
<-----	
SUBSCRIBE sip:456@b.com;m=NR From: sip:foo@example.com Request-Disposition: parallel Call-Id: abcd-efgh Event: call-completion	[another init. SUB.]
----->	
482	[duplicate SUB. rej.]
<-----	
NOTIFY sip:123@y.a.com Body: status: ready URI: sip:recall@z.b.com	[CC invoked]
<-----	
INVITE sip:recall@z.b.com;m=NR From: sip:foo@example.com	[CC call]
----->	
NOTIFY sip:123@y.a.com Expires = 0	[CC terminated]
<-----	

The original call is an ordinary INVITE. It fails due to no-response (ring-no-answer). In this case, the callee's governing proxy generates a 487 response because the proxy canceled the INVITE to the UA when it rang too long without an answer. The 487 response carries a Call-Info header field with "purpose=call-completion". The Call-Info header field positively indicates that CC is available for this failed fork of the call. The "m=NR" parameter indicates that it failed due to no-response, which is useful for PSTN interworking and assessing presence information in the callee's monitor.

The URI in the Call-Info header field (<sip:456@z.b.com>) is where the caller's agent should subscribe for call-completion processing. Ideally, it is a globally-routable URI for the callee's monitor. In practice, it may be the callee's AOR, and the SUBSCRIBE will be routed to the callee's monitor only because it specifies "Event: call-completion".

CC is activated by sending a SUBSCRIBE to all known callee's monitor URIs. These can be provided by the Call-Info header field in the response to the INVITE.

Additionally, the caller's agent needs to include the original request-URI in its set of callee's monitor URIs, because the call may have forked to additional callees whose responses the caller has not seen. (A SUBSCRIBE to the request-URI alone is used in cases where the caller's agent has not received or cannot remember any callee's monitor URI.)

The caller's agent adds to these URIs an 'm' parameter (if possible). In this case, the caller's agent forks the SUBSCRIBE to two destinations as defined by section 8.2.2.2 of [RFC3261], with appropriate Request-Disposition. The first SUBSCRIBE is to the URI from Call-Info.

The second SUBSCRIBE is to the original request-URI, and reaches the same callee's monitor. Because it has the same Call-Id as the SUBSCRIBE that has already reached the callee's monitor, the callee's monitor rejects it with a 482, thus avoiding redundant subscriptions.

The initial NOTIFY for the successful SUBSCRIBE has "state: queued" in its body. Eventually, this caller is selected for CC, and is informed of this via a NOTIFY containing "state: ready". This NOTIFY carries a URI to which the INVITE for CC call should be sent. In practice, this may be the AOR of the callee.

The caller generates a new INVITE to the URI specified in the NOTIFY, or if there was no such URI or if the caller's agent cannot remember it, it may use the original request-URI. The caller adds the 'm'

parameters (if possible), to specify CC processing.

Finally the subscription for the CC request is terminated by the callee's monitor.

Another flow, with only the most significant messages of call-completion suspension and resumption shown, is as follows:

Caller sip:123@a.com	Callee sip:456@b.com
NOTIFY sip:123@y.a.com Body: status: ready URI: sip:recall@z.b.com	[CC notification, caller not available for CC recall]
<-----	
200	
----->	
PUBLISH sip:456@z.b.com From: sip:123@a.com Contact: sip:123@y.a.com Event: presence Content-Type: 'app/pidf' Body: status=closed	[non-availability for recall is published]
----->	
200	
<-----	
	[caller becomes available again]
PUBLISH sip:456@z.b.com From: sip:123@a.com Contact: sip:123@y.a.com Event: presence Content-Type: 'app/pidf' Body: status=open	[availability for recall is published]
----->	
200	
<-----	

The caller is selected for CC, and is informed of this via a NOTIFY request containing "state: ready". At this time, the caller is not available for the CC recall.

For updating his presence event state at the presence server functionality at the callee, the caller generates a PUBLISH request to the CC URI as received in the NOTIFY, or within the corresponding SUBSCRIBE dialog, or if that is not possible, to the corresponding callee's monitor URI received in the Call-Info header field of the NOTIFY, or if one is not available, the Contact address of the subscription, informing about the PIDF state 'closed'.

When the caller is again available for the CC recall, the caller updates his presence event state at the presence server functionality at the callee by generating a PUBLISH request informing about the PIDF state 'open', but otherwise constructed as same as the suspend PUBLISH request.

9. Call-completion event package

This section specifies the call-completion event package, in accordance with section 4.4 of [RFC6665]. The call-completion event package has the media type "application/call-completion".

Note that if the callee has a caller-queuing facility, the callee's monitor may want to treat the call-completion queue as part of the queuing facility, and include in the event package information regarding the state of the queue. How this information is conveyed is left for further standardization.

9.1. Event package name

The SIP Events specification requires package definitions to specify the name of their package or template-package. The name of this package is "call-completion". This value appears in the Event and Allow-events header fields.

9.2. Event package parameters

No package-specific Event header field parameters are defined for this event package.

9.3. SUBSCRIBE bodies

[RFC6665] requires package definitions to define the usage, if any, of bodies in SUBSCRIBE requests.

The SUBSCRIBE request MAY contain an Accept header field. If no such header field is present, it has a default value of "application/call-completion". If the header field is present, it MUST include "application/call-completion".

A SUBSCRIBE request for a call-completion package MAY contain a body. This body defines a filter to be applied to the subscription. Filter documents are not specified in this document, and may be the subject of future standardization activity.

A SUBSCRIBE request requests call-completion information regarding calls recently made from the same caller to the callee UA(s) serviced

by the notifier. Calls are defined to be "from the same caller" if the URI-part of the From header field value in the INVITE is the same as the URI-part of the From header field value in the SUBSCRIBE.

9.4. Subscribe duration

[RFC6665] requires package definitions to define a default value for subscription durations, and to discuss reasonable choices for durations when they are explicitly specified.

If a SUBSCRIBE does not explicitly request a duration, the default requested duration is 3600 seconds, as that is the highest service duration timer value recommended for the call completion services in ETSI [ETSI300.356-18] and ITU-T [ITU-T.Q.733]. As because of the subscription duration no explicit timer is needed, and the subscription duration can be seen as an equivalent to the SS7 service duration timer, this specification refers to the subscription duration also as the service duration timer. It is RECOMMENDED that subscribers request, and that notifiers grant, a subscription time of at least 3600 seconds.

If a notifier can determine that, according to its policy, after a certain duration the requested subscription can not any more proceed to "ready" state, it SHOULD reduce the granted subscription time to that duration. If a notifier can determine that, according to its policy, the requested subscription can never proceed to "ready" state, it should refuse the subscription.

9.5. NOTIFY bodies

[RFC6665] requires package definitions to describe the allowed set of body types in NOTIFY requests, and to specify the default value to be used when there is no Accept header field in the SUBSCRIBE request. A NOTIFY for a call-completion package MUST contain a body that describes the call-completion states.

As described in [RFC6665], the NOTIFY message will contain bodies that describe the state of the subscribed resource. This body is in a format listed in the Accept header field of the SUBSCRIBE, or in a package-specific default format if the Accept header field was omitted from the SUBSCRIBE.

In this event package, the body of the notification contains a call-completion document. All subscribers and notifiers MUST support the "application/call-completion" data format described in section 10. The SUBSCRIBE request MAY contain an Accept header field. If no such header field is present, it has a default value of "application/call-completion". If the header field is present, it MUST include

"application/call-completion". Of course, the notifications generated by the server MUST be in one of the formats specified in the Accept header field in the SUBSCRIBE request.

9.6. Subscriber generation of SUBSCRIBE requests

Subscribers MUST generate SUBSCRIBE requests when they want to subscribe to the call-completion event package at the terminating side in order to receive call-completion notifications. The generation of SUBSCRIBE requests can imply the usage of a call-completion service specific timer as described in section 9.4.

9.7. Notifier processing of SUBSCRIBE requests

Upon receiving a subscription refresh, the notifier MUST set the "expires" parameter of the Subscription-State header field to a value not higher than the current remaining duration of the subscription regardless of the value received in the Expires header field (if present) of the subscription refresh.

If a subscription is not successful because the call-completion queue has reached the maximum allowed number of entries (short term denial), the notifier MUST send a 480 Temporarily Unavailable response to the subscriber, possibly with a Retry-after header field in accordance with the notifier's policy. If a subscription is not successful because a condition has occurred that prevents and will continue to prevent the call-completion service (long term denial), the notifier MUST send a 403 Forbidden response to the subscriber.

A notifier MAY receive multiple forks of the same SUBSCRIBE, as defined by section 8.2.2.2 of [RFC3261]. In such a case, the notifier MUST reject all but one of the SUBSCRIBES with a 482 Merged Request response unless some other failure response applies.

The call-completion information can be sensitive. Therefore, all subscriptions SHOULD be handled with consideration of the security considerations discussed in section 11, in particular for verifying the identity of the subscriber.

9.8. Notifier generation of NOTIFY requests

Notifiers MUST generate NOTIFY requests when the CC request's state changes to 'queued' or to 'ready (for call-completion)'. A NOTIFY that is sent with non-zero expiration MUST contain the "cc-state" parameter. The parameter's value MUST be "queued" if the call-completion request represented by the subscription is not at this time selected by the callee's monitor for CC recall, and the parameter's value MUST be "ready" if the request is at this time

selected by the callee's monitor for CC recall.

A NOTIFY sent with a zero expiration (e.g., as a confirmation of a request to unsubscribe) MAY contain the "cc-state" parameter.

When the callee's monitor withdraws the selection of the request for the CC recall (e.g., because the caller's agent has not initiated the CC recall INVITE before the CC recall timer expires, or because the agent has suspended the request from being considered for CC recall), the notifier MUST send a NOTIFY to the subscription of the selected request. This NOTIFY MUST contain the "cc-state" parameter set to "queued".

If the call-completion subscription was successful and the retain option is supported at the callee, the NOTIFY MUST contain the "cc-service-retention" parameter.

9.9. Subscriber processing of NOTIFY requests

When receiving a NOTIFY requests with the cc-state set to 'ready', subscribers SHOULD suspend all other CC subscriptions for the original call at other notifiers. The receipt of a NOTIFY request with the cc-state set to 'ready' by the subscriber will also cause an interaction with the instances at the subscribers side that are responsible for starting the CC recall.

9.10. Handling of forked requests

Forked requests are expected to be common for the call-completion event type. The subscriber MUST be prepared to process NOTIFY requests from multiple notifiers and to coordinate its processing of the information obtained from them in accordance with the procedures in this document.

9.11. Rate of notifications

The call completion service typically involves a single notification per notifier and per subscription that notifies about the change to 'ready (for call-completion)', but MAY involve several notifications about the change to the 'ready' state, separated by a call completion call that failed due to a busy callee. Typically, notifications will be separated by at least tens of seconds. Notifiers SHOULD NOT generate more than three notifications for one subscription in any ten-second interval. Since it is important to avoid useless recalls, a notifier SHOULD send state changes to "queued" from "ready" promptly. Thus, a notifier SHOULD NOT send a state change to "ready" as the third notification in a ten-second interval, as that would make it impossible to promptly send a further state change to

"queued".

9.12. State agents

State agents have no defined role in the handling of the call-completion package.

10. Call-completion information format

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) as described in [RFC5234]. The formal syntax for the application/call-completion MIME type is described below. In general, the call-completion body is to be interpreted in the same way as SIP headers: (1) the names of the lines are case-insensitive, (2) the lines can be continued over line boundaries if the succeeding lines start with horizontal white space, and (3) lines with unknown names are to be ignored. The header lines defined in this document can occur at most once in any given call-completion document.

call-completion = 1*(cc-header CRLF)

cc-header = cc-state / cc-service-retention / cc-URI / extension-header

The above rules whose names start with "cc-" are described below. Other rules are described in [RFC3261].

10.1. Call-completion status

The cc-state line indicates the CC status of a particular user with an entry in a CC queue. It MUST be present unless the expiration time indicated in the NOTIFY is zero.

cc-state = "cc-state" HCOLON ("queued" / "ready")

The value "queued" indicates that a subscriber's entry in the call-completion queue is not selected for CC recall. The value "ready" indicates that a user's entry in the call-completion queue has been selected for CC recall.

10.2. Call-completion service-retention indication

The service-retention line indicates the support of the retain option. The retain option, if supported at the callee, will maintain the entry in the CC queue, if a CC call has failed due to callee busy condition. If present, this parameter indicates that the retain option is supported, otherwise it is not supported. This parameter

MAY be present in NOTIFY requests.

cc-service-retention = "cc-service-retention" HCOLON "true"

10.3. Call-completion URI

The cc-URI line provides a URI (possibly in the form of a name-addr) which the agent SHOULD use as the request-URI of the CC recall INVITE and the suspend/resume PUBLISH. It SHOULD be provided in all NOTIFYs. The URI SHOULD be globally routable and SHOULD uniquely identify the CCE in question. The syntax provides for generic-params in the value, but this document defines no such parameters. Parameters that are not understood by the subscriber MUST be retained with the URI.

cc-URI = "cc-URI" HCOLON addr-spec

11. Security considerations

The CC facility allows the caller's agent to determine some status information regarding the callee. This information intrinsically diminishes the privacy of the callee; in order to protect sufficiently the privacy of the callee, the overall amount of disclosure must be limited, and the amount of disclosure to any single caller must be limited.

Of course, if a caller is not permitted to call the callee, that caller should not be allowed to establish a CC subscription. Callers that are particularly sensitive about their privacy may reject all CC subscriptions. But in the ordinary case, the optimal protection is to permit any caller to subscribe, but prevent any caller from subscribing for too long, or too often, or in a pattern that does not reveal to the callee (through CC calls) that the subscriptions are taking place.

In legitimate use, CC event subscriptions will be made in stereotyped ways that limit the disclosure of status information:

1. When a subscriber is selected for CC, a call should arrive promptly for the callee, or the subscription should be terminated. This expectation may be violated by a race condition between selection of the subscription for CC and the caller becoming unavailable, but it should be rare that a single subscription will exhibit the race condition more than once.
2. Subscriptions should not remain suspended for longer than the expected duration of a call (a call by the caller to a third

party).

3. Subscriptions should be initiated only shortly after failed incoming calls.
4. Most of the time, a callee should have no queued subscriptions.

Violations of these expectations should be detected by the callee's monitor and reported as possible attempts at privacy violation.

The CC facility may enhance the effectiveness of Spam over Internet Telephony (SPIT) by the following technique: The caller makes calls to a group of callees. The caller then requests CC for the calls that do not connect to the callees. The CC calls resulting are probably more likely to reach the callees than original calls to a further group of targets.

In order to prevent Denial of Service (DoS) attacks and manipulations of the call-completion queue by suspending other call-completion entries than the own, a mechanism to correlate the identity of the original caller and the generator of the SUBSCRIBE and PUBLISH request is needed. The RECOMMENDED mechanism to authenticate the identity of the originator of call-completion relevant requests is the SIP Identity mechanism [RFC4474]. Alternatively, CC agents and monitors within an administrative domain or federation of domains MAY use the mechanism described in [RFC3325] to authenticate their identity with a P-Asserted-Identity header field.

Furthermore, the use of presence server functionality to suspend or resume SHOULD be limited to a caller which has an active queue in the callee's monitor. This can be achieved first by monitoring and logging incoming call to the callee and the destination where CC indication was sent, then to ensure subscription to the call completion package is permitted only within a short timeframe after the initial call failed and to only accept PUBLISH request to the presence server functionality if there is an active queue for the caller in question.

Note that regarding the authentication/authorization/billing logic subject to operator policy CC calls or subscriptions do not differ from other basic calls or event subscriptions.

12. IANA considerations

12.1. SIP event package registration for call-completion

This specification registers an event package, based on the registration procedures defined in [RFC6665]. The followings is the information required for such a registration:

Package Name: call-completion

Is this registration for a Template-Package: No.

Published Document: RFC XXXX (Note for RFC Editor: Please fill in XXXX with the RFC number of this specification).

Person and e-mail to contact for further information: Martin Huelsemann, martin.huelsemann@telekom.de

12.2. MIME registration for application/call-completion

MIME media type name: application

MIME subtype name: call-completion

Required parameters: none.

Optional parameters: none.

Encoding considerations: Consists of lines of UTF-8-encoded characters, ended with CR-LF

Security considerations: There are no security considerations internal to the media type. Its typical usage involves the security considerations described in RFC XXXX

(Note for RFC Editor: Please fill in XXXX with the RFC number of this specification).

Interoperability considerations: See RFC XXXX (Note for RFC Editor: Please fill in XXXX with the RFC number of this specification).

Published specification: RFC XXXX (Note for RFC Editor: Please fill in XXXX with the RFC number of this specification)

Applications that use this media type: the implementations of the call-completion features of the Session Initiation Protocol

Additional information:

Magic number(s): none

File extension(s): not expected to be stored in files

Macintosh file type code(s): not expected to be stored in files

Person & email address to contact for further information: Martin Huelsemann, martin.huelsemann@telekom.de

Intended usage: LIMITED USE

Restrictions on usage: none

Author/Change controller: the IETF

12.3. SIP/SIPS URI parameter 'm'

This specification defines one new SIP/SIPS URI parameter 'm' as per the registry created by [RFC3969]. It is used to identify that an INVITE request is a CC call, or to further identify that a SUBSCRIBE request is for the call-completion event package. The parameter may have a value that describes the type of the call-completion operation, as described in this specification.

Name of the Parameter: m

Predefined Values : yes

RFC Reference : [RFC XXXX]

(Note for RFC Editor: Please fill in XXXX with the RFC number of this specification)

12.4. Call-Completion purpose parameter value

This specification adds a new predefined value "call-completion" for the "purpose" header field parameter of the Call-Info header field. This modifies the registry header field parameters and parameter values by adding this RFC as a reference to the line for header field "Call-Info" and parameter name "purpose":

Header Field: Call-Info

Parameter Name: purpose

Predefined Values: yes

Reference: [RFC3261].[RFC5367][[RFC XXXX]]

(Note for RFC Editor: Please fill in XXXX with the RFC number of this specification)

specification)

12.5. 'm' header parameter for Call-Info

This specification extends [RFC3261] to add a new header field parameter 'm' to the Call-Info header field. This adds a row to the registry header field parameters and parameter values:

Header Field: Call-Info

Parameter Name: m

Predefined Values: yes

Reference: [RFC XXXX]

This RFC predefines the values 'BS', 'NR' and 'NL' .

(Note for RFC Editor: Please fill in XXXX with the RFC number of this specification)

13. Acknowledgements

Thanks to Paul Kyzivat, John Elwell, Keith Drage, Andrew Hutton, Thomas Stach, Dennis Luebbbers and Christer Holmberg who provided helpful comments, feedback and suggestions.

14. References

14.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3515] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.
- [RFC3863] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., and J. Peterson, "Presence Information Data Format (PIDF)", RFC 3863, August 2004.

- [RFC3903] Niemi, A., "Session Initiation Protocol (SIP) Extension for Event State Publication", RFC 3903, October 2004.
- [RFC3969] Camarillo, G., "The Internet Assigned Number Authority (IANA) Uniform Resource Identifier (URI) Parameter Registry for the Session Initiation Protocol (SIP)", BCP 99, RFC 3969, December 2004.
- [RFC4235] Rosenberg, J., Schulzrinne, H., and R. Mahy, "An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP)", RFC 4235, November 2005.
- [RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, August 2006.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5367] Camarillo, G., Roach, A., and O. Levin, "Subscriptions to Request-Contained Resource Lists in the Session Initiation Protocol (SIP)", RFC 5367, October 2008.
- [RFC5627] Rosenberg, J., "Obtaining and Using Globally Routable User Agent URIs (GRUUs) in the Session Initiation Protocol (SIP)", RFC 5627, October 2009.
- [RFC6665] Roach, A., "SIP-Specific Event Notification", RFC 6665, July 2012.

14.2. Informative References

- [ETS300.356-18]
"Completion of Calls to Busy Subscriber (CCBS) supplementary service", February 1995.
- [ITU-T.Q.733]
"DESCRIPTION FOR CALL COMPLETION SUPPLEMENTARY SERVICES USING SS No. 7", February 1995.
- [RFC3325] Jennings, C., Peterson, J., and M. Watson, "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks", RFC 3325, November 2002.
- [RFC3725] Rosenberg, J., Peterson, J., Schulzrinne, H., and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)",

BCP 85, RFC 3725, April 2004.

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

Appendix A. Example Caller's Agent

This section outlines how an autonomous caller's agent can operate using only standard SIP features to interact with the caller's UA. This example is suitable only as a learning aid, as its performance is poor.

The agent monitors calls made from the UA(s) by subscribing to the dialog event package of the UA(s).

The UA(s) or their proxy routes calls made with either of two special dial sequences to the agent, which interprets the INVITES as indications to make a CC request with BS or NR or NL mode for the latest call made by the UA.

The agent monitors the status of the UA(s) for availability to be used for a CC call by examining the dialog events.

The agent indicates to the UA(s) that CC recall is in progress by making call to the UA(s). If the UA is answered, the agent assumes that the caller is available and plays pre-recorded audio to indicate that CC recall is in progress.

After playing the pre-recorded audio, the caller's agent uses REFER to order the UA to make the CC call to the callee.

Appendix B. Example Callee's Monitor

This section outlines how an autonomous callee's monitor can operate using only standard SIP features to interact with the callee's UA. This example is suitable only as a learning aid, as its performance is poor.

The callee's monitor monitors calls made to the UA(s) by subscribing to the UA(s) dialog events. This enables it to determine their Call-Id's and their final response statuses.

The proxy for the UA(s) routes to the callee's monitor any SUBSCRIBES for the call-completion event package directed to the URIs serviced by the UA(s).

The callee's monitor monitors the status of the UA(s) to determine when they are in a suitable state to receive a CC call by watching the busy/not-busy status of the UA(s): e.g. a UA is available for CCBS when it is not busy, but a UA is available for CCNR when it becomes not busy after being busy with an established call.

Authors' Addresses

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

Phone: +1 781 647 9199
Email: worley@ariadne.com
URI: <http://>

Martin Huelsemann
Deutsche Telekom
Heinrich-Hertz-Strasse 3-7
Darmstadt, 64307
Germany

Phone: +4961515812765
Email: martin.huelsemann@telekom.de
URI: <http://www.telekom.de>

Roland Jesske
Deutsche Telekom
Heinrich-Hertz-Strasse 3-7
Darmstadt, 64307
Germany

Phone: +4961515812766
Email: r.jesske@telekom.de
URI: <http://www.telekom.de>

Denis Alexeitsev
TeleFLASH
Mainzer Landstrasse 47
Frankfurt 60329
Germany

Phone: +49-69-257-378-230
Email: alexeitsev@teleflash.com
URI: <http://www.teleflash.com>

BLISS
Internet-Draft
Updates: 3261, 4235 (if approved)
Intended status: Standards Track
Expires: July 20, 2013

A. Johnston, Ed.
Avaya
M. Soroushnejad
V. Venkataramanan
Sylantro Systems Corp
January 16, 2013

Shared Appearances of a Session Initiation Protocol (SIP) Address of
Record (AOR)
draft-ietf-bliss-shared-appearances-15

Abstract

This document describes the requirements and implementation of a group telephony feature commonly known as Bridged Line Appearance (BLA) or Multiple Line Appearance (MLA), or Shared Call/Line Appearance (SCA). When implemented using the Session Initiation Protocol (SIP), it is referred to as shared appearances of an Address of Record (AOR) since SIP does not have the concept of lines. This feature is commonly offered in IP Centrex services and IP-PBX offerings and is likely to be implemented on SIP IP telephones and SIP feature servers used in a business environment. This feature allows several user agents (UAs) to share a common AOR, learn about calls placed and received by other UAs in the group, and pick up or join calls within the group. This document discusses use cases, lists requirements and defines extensions to implement this feature. This specification updates RFC3261 and RFC4235.

Status of this Memo

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

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

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

This Internet-Draft will expire on July 20, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	5
2. Conventions used in this document	6
3. Usage Scenarios	6
3.1. Executive/Assistant Arrangement	6
3.2. Call Group	7
3.3. Single Line Extension	7
3.4. Changing UAs	7
4. Requirements and Implementation	7
4.1. Requirements	7
4.2. Implementation	9
5. Normative Description	11
5.1. Elements	12
5.2. Shared Appearance Dialog Package Extensions	12
5.2.1. The <appearance> element	12
5.2.2. The <exclusive> element	13
5.2.3. The <joined-dialog> element	13
5.2.4. The <replaced-dialog> element	14
5.3. Shared Appearance User Agents	14
5.3.1. Appearance Numbers and Call Context	17
5.3.2. Appearance Numbers and Call Control	18
5.3.3. Appearance Numbers and Transfer	18
5.4. Appearance Agent	19
6. XML Schema Definition	21
7. Alert-Info Appearance Parameter Definition	23
8. User Interface Considerations	24
8.1. Appearance Number Rendering	24
8.1.1. Single Appearance UAs	24
8.1.2. Dual Appearance UAs	24
8.1.3. Shared Appearance UAs with Fixed Appearance Number	25
8.1.4. Shared Appearance UAs with Variable Appearance Number	25
8.1.5. Example User Interface Issues	25
8.2. Call State Rendering	26
9. Interoperability with non-Shared Appearance UAs	26
9.1. Appearance Assignment	26
9.2. Appearance Release	27
9.3. UAs Supporting Dialog Events but Not Shared Appearance	27
10. Provisioning Considerations	27
11. Example Message Flows	28
11.1. Registration and Subscription	28
11.2. Appearance Selection for Incoming Call	32
11.3. Outgoing Call without Appearance Seizure	35
11.4. Outgoing Call with Appearance Seizure	38
11.5. Outgoing Call without using an Appearance Number	42
11.6. Appearance Release	44
11.7. Appearance Pickup	45

11.8.	Calls between UAs within the Group	50
11.9.	Consultation Hold with Appearances	52
11.10.	Joining or Bridging an Appearance	55
11.11.	Appearance Allocation - Loss of Appearance	58
11.12.	Appearance Seizure Contention Race Condition	59
11.13.	Appearance Agent Subscription to UAs	60
11.14.	Appearance Pickup Race Condition Failure	62
11.15.	Appearance Seizure Incoming/Outgoing Contention Race Condition	65
12.	Security Considerations	66
13.	IANA Considerations	66
13.1.	SIP Event Header Field Parameter: shared	66
13.2.	SIP Alert-Info Header Field Parameter: appearance	67
13.3.	URN Sub-Namespace Registration: sa-dialog-info	68
13.4.	XML Schema Registration	68
14.	Acknowledgements	69
15.	References	69
15.1.	Normative References	69
15.2.	Informative References	70
	Authors' Addresses	71

1. Introduction

The feature and functionality requirements for SIP user agents (UAs) supporting business telephony applications differ greatly from basic SIP user agents, both in terms of services and end user experience. In addition to basic SIP support [RFC3261], many of the services in a business environment require the support for SIP extensions such as REFER [RFC3515], SUBSCRIBE/NOTIFY primitives [I-D.ietf-sipcore-rfc3265bis] PUBLISH [RFC3903], the SIP Replaces [RFC3891], and Join [RFC3911] header fields, etc. Many of the popular business services have been documented in the SIP Service Examples [RFC5359].

This specification details a method for implementing a group telephony feature known variously in telephony as Bridged Line Appearance (BLA) or Multiple Line Appearances (MLA), one of the more popular advanced features expected of SIP IP telephony devices in a business environment. Other names for this feature include Shared Call/Line Appearance (SCA), Shared Call Status and Multiple Call Appearance (MCA). A variant of this feature is known as Single Line Extension.

This document looks at how this feature can be implemented using standard SIP [RFC3261] in conjunction with SIP events [I-D.ietf-sipcore-rfc3265bis] and publication [RFC3903] (carrying the SIP dialog state event package [RFC4235]) for exchanging status among user agents.

In traditional telephony, the line is physical. A common scenario in telephony is for a number of business telephones to share a single or a small number of lines. The sharing or appearance of these lines between a number of phones is what gives this feature its name. A common scenario in SIP is for a number of business telephones to share a single or a small number of Address of Record (AOR) URIs.

In addition, an AOR can have multiple appearances on a single UA in terms of the user interface. The appearance number relates to the user interface for the telephone - typically each appearance of an AOR has a visual display (lamp that can change color or blink or a screen icon) and a button (used to select the appearance) where each appearance number is associated with a different dialog to/from the AOR. The telephony concept of line appearance is still relevant to SIP due to the user interface considerations. It is important to keep the appearance number construct because:

1. Human users are used to the concept and will expect it in replacement systems (e.g. an overhead page announcement says "Joe pickup line 3").

2. It is a useful structure for user interface representation.

The purpose of the appearance number is to identify active calls to facilitate sharing between users (e.g. passing a call from one user to another). If a telephone has enough buttons/lamps, the appearance number could be the positional sequence number of the button. If not, it may still be desirable to present the call state, but the appearance number should be displayed so that users know which call, for example, is on hold on which key.

In this document, except for the usage scenarios in the next section, we will use the term "appearance" rather than "line appearance" since SIP does not have the concept of lines. Note that this does not mean that a conventional telephony user interface (lamps and buttons) must be used - implementations may use another metaphor as long as the appearance number is readily apparent to the user. Each AOR has a separate appearance numbering space. As a result, a given UA user interface may have multiple occurrences of the same appearance number, but they will be for different AORs.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119] and indicate requirement levels for compliant mechanisms.

3. Usage Scenarios

The following examples are common applications of the Shared Appearances feature and are mentioned here as informative use cases. All these example usages can be supported by the Shared Appearances feature described in this document. The main differences relate to the user interface considerations of the device.

3.1. Executive/Assistant Arrangement

The appearances on the executive's UA also appear on the assistant's UA. The assistant may answer incoming calls to the executive and then place the call on hold for the executive to pick up. The assistant can always see the state of all calls on the executive's UA.

3.2. Call Group

Users with similar business needs or tasks can be assigned to specific groups and share an AOR. For example, an IT department staff of five might answer a help line which has three appearances on each phone in the IT work area. A call answered on one phone can be put on hold and picked up on another phone. A shout or an IM to another staff member can result in them taking over a call on a particular appearance. Another phone can request to be added/joined/bridged to an existing appearance resulting in a conference call.

3.3. Single Line Extension

In this scenario, incoming calls are offered to a group of UAs. When one answers, the other UAs are informed. If another UA in the group seizes the line (i.e. goes off hook), it is immediately bridged or joined in with the call. This mimics the way residential telephone extensions usually operate.

3.4. Changing UAs

A user is on a call on one UA and wishes to change devices and continue the call on another UA. They place the call on hold, note the appearance number of the call, then walk to another UA. They are able to identify the same appearance number on the other UA, pickup the call, and continue the conversation.

4. Requirements and Implementation

The next section details the requirements and discusses the implementation of the shared appearances of an AOR feature.

4.1. Requirements

The basic requirements of the shared appearance feature can be summarized as follows:

REQ-1 Incoming calls to the AOR must be offered to a group of UAs and can be answered by any of them.

REQ-2 Each UA in the group must be able to learn the call status of the others in the group for the purpose of rendering this information to the user.

REQ-3 A UA must be able to join (also called bridge or conference together) or pick up (take) an active call of another UA in the group in a secure way.

REQ-4 The mechanism should require the minimal amount of configuration. UAs registering against the group AOR should be able to participate in the appearance group without manual configuration of group members.

REQ-5 The mechanism must scale for large numbers of appearances and large numbers of UAs without introducing excessive messaging traffic.

REQ-6 Each call or session (incoming or outgoing) should be assigned a common "appearance" number from a managed pool administered for the AOR group. Once the session has terminated, the appearance number is released back into the pool and can be reused by another incoming or outgoing session.

REQ-7 Each UA in the group must be able to learn the status of all appearances of the group.

REQ-8 There must be mechanisms to resolve appearance contention among the UAs in the group. Contention in this context means an appearance number being associated with multiple dialogs that are not mixed or otherwise associated.

REQ-9 The mechanism must allow all UAs receiving an incoming session request to utilize the same appearance number at the time of alerting.

REQ-10 The mechanism must have a way of reconstructing appearance state after an outage that does not result in excessive traffic and processing.

REQ-11 The mechanism must have backwards compatibility such that a UA which is unaware of the feature can still register against the group AOR and make and receive calls.

REQ-12 The mechanism must not allow UAs outside the group to select, seize or manipulate appearance numbers.

REQ-13 For privacy reasons, there must be a mechanism so that appearance information is not leaked outside the group of UAs. (e.g. "So who do you have on line 1?")

REQ-14 The mechanism must support a way for UAs to request exclusivity on a line appearance. Exclusivity means that the UA requesting it desires to have a private conversation with the external party and other UAs must not be allowed to join or take the call. Exclusivity may be requested at the start of an incoming or outgoing session or during the session. An exclusivity request may be accepted or rejected by the entity providing the shared appearance

service. Therefore, the mechanism must provide a way of communicating the result back to the requester UA.

REQ-15 The mechanism should support a way for a UA to seize a particular appearance number for outgoing requests prior to sending the actual request. This is often called seizure.

REQ-16 The mechanism should support a way for a UA to seize a particular appearance number and also send the request at the same time. This is needed when an automatic ringdown feature (a telephone configured to immediately dial a phone number when it goes off hook) is combined with shared appearances - in this case, seizing the line is integrated with dialing.

4.2. Implementation

This section non-normatively discusses the implementation of the shared appearance feature. The normative description is in Section 5. Many of the requirements for this service can be met using standard SIP mechanisms such as:

- A SIP Forking Proxy and Registrar/Location Service meets REQ-1.
- The SIP Dialog Package meets REQ-2.
- The combination of the SIP Replaces and Join header fields meets REQ-3.
- The use of a State Agent for the Dialog Package meets REQ-4 and REQ-5.

REQ-6 suggests the need for an entity which manages the appearance resource. Just as conferencing systems commonly have a single point of control, known as a focus, a Shared Appearance group has a single point of control of the appearance shared resource. This is defined as an Appearance Agent for a group. While an Appearance Agent can be part of a centralized server, it could also be co-resident in a member User Agent that has taken on this functionality for a group. The Appearance Agent knows or is able to determine the dialog state of all members of the group.

While the appearance resource could be managed co-operatively by a group of UAs without any central control, this is outside the scope of this draft. It is also possible that the Appearance Agent logic could be distributed in all UAs in the group. For example, rules that govern assigning appearance numbers for incoming requests (e.g. lowest available appearance number) and rules for contention handling (e.g. when two UAs request the use of the same appearance number,

hash dialog identifiers and compare with the lowest hash winning) would need to be defined and implemented.

To best meet REQ-9, the appearance number for an incoming INVITE needs to be contained in the INVITE, in addition to being delivered in the dialog package NOTIFY. Otherwise, if the NOTIFY is delayed or lost, a UA in the group might receive an incoming INVITE but might not know which appearance number to render during alerting.

This specification defines an extension parameter, which is normatively defined in Section 7, for the Alert-Info header field in RFC 3261 to carry the appearance number:

Alert-Info: <urn:alert:service:normal>;appearance=1

The following list describes the operation of the shared appearance feature.

1. A UA is configured with the AOR of a shared appearance group. It registers against the AOR, then attempts a dialog state subscription to the AOR. If the subscription fails, loops back to itself, or returns an error, it knows there is no State Agent, and hence no Appearance Agent and this feature is not implemented.
2. If the subscription receives a 200 OK, the UA knows there is a State Agent and that the feature is implemented. The UA then follows the steps in this list.
3. Information learned about the dialog state of other UAs in the group is rendered to the user.
4. Incoming calls are forked to all UAs in the group, and any may answer. UAs receive the appearance number to use in rendering the incoming call in a NOTIFY from the Appearance Agent and in the INVITE itself. The UA will also receive a notification if the call is answered by another UA in the group so this information can be rendered to the user.
5. For outgoing calls, the operation depends on the implementation. If the user seizes a particular appearance number for the call, the UA publishes the trying state dialog information with the desired appearance number and waits for a 2xx response before sending the INVITE.
6. For outgoing calls, if the user does not seize a particular appearance or does not care, the INVITE can be sent immediately, and the appearance number learned as the call progresses from a notification from the Appearance Agent.
7. For outgoing calls, if the user does not want an appearance number assigned (such as during a consultation call or if a UA is fetching 'service media' such as music on hold [I-D.worley-service-example]), the UA also publishes prior to

sending the INVITE but does not include an appearance number in the publication.

8. Established calls within the group may be joined (bridged) or taken (picked) by another UA. Information in the dialog package notifications can be used to construct Join or Replaces header fields. Since the same appearance number is used for these types of operations, this information is published prior to sending the INVITE Join or INVITE Replaces.
9. The Appearance Agent may not have direct access to the complete dialog state of some or all of the UAs in the group. If this is the case, the Appearance Agent will subscribe to the dialog state of individual UAs in the group to obtain this information. In any case, the Appearance Agent will send normal notifications (via the subscriptions established by the UAs in step 1) every time the aggregate dialog state of the AOR changes, including when calls are placed, answered, placed on and off hold, and hung up.

5. Normative Description

This section normatively describes the shared appearance feature extensions. The following definitions are used throughout this document:

Appearance number: An appearance number is a positive integer associated with one or more dialogs of an AOR. Appearance numbers are managed by an Appearance Agent and displayed and rendered to the user by UAs that support this specification. When an appearance number is assigned or requested, generally the assigned number is the smallest positive integer that is not currently assigned as an appearance number to a dialog for this AOR. This specification does not define an upper limit on appearance numbers; however, using appearance numbers that are not easily represented using common integer representations is likely to cause failures.

Seizing: An appearance can be reserved prior to a call being placed by seizing the appearance. An appearance can be seized by communicating an artificial state of "trying" prior to actually initiating a dialog (i.e. sending the INVITE), in order to appear as if it was already initiating a dialog.

Selecting(or Not-Seizing): An appearance is merely selected (i.e., not seized) if there is no such communication of artificial state of "trying" prior to initiating a dialog: i.e., the state is communicated when the dialog is actually initiated. The appearance number is learned after the INVITE is sent.

5.1. Elements

A complete system to implement this feature consists of:

1. User Agents that support publications, subscriptions, and notifications for the SIP dialog event package, and the shared appearance dialog package extensions and behavior.
2. An Appearance Agent consisting of a State Agent for the dialog event package that implements an Event State Compositor (ESC) and the shared appearance dialog package extensions and behavior. The Appearance Agent also has logic for assigning and releasing appearance numbers, and resolving appearance number contention.
3. A forking proxy server that can communicate with the State Agent
4. A registrar that supports the registration event package.

The behavior of these elements is described normatively in the following sections after the definitions of the dialog package extensions.

5.2. Shared Appearance Dialog Package Extensions

This specification defines four new elements as extensions to the SIP Dialog Event package [RFC4235]. The schema is defined in Section 6. The elements are <appearance>, <exclusive>, <joined-dialog>, and <replaced-dialog> which are sub-elements of the <dialog> element.

5.2.1. The <appearance> element

The <appearance> element, a child of the <dialog> element, is used to convey the appearance number of the dialog described by the parent <dialog> element. When sent by a UA in a PUBLISH with parent <dialog> with state attribute "trying" to the Appearance Agent, the UA is requesting assignment of the given appearance number to the current or future dialog with the given dialog identifiers. When an <appearance> element is sent by the Appearance Agent in a NOTIFY, it indicates that the appearance number has been assigned to the specified dialog.

Note that a <dialog-info> element describes the contained dialogs from the point of view of the UA (named by the "entity" attribute), regardless of whether the containing request is sent by the UA or the Appearance Agent. In particular, if the UA sent a request within the described dialog, the To header field URI would match the <remote> <identity> value and the to-tag parameter would match the remote-tag attribute. Similarly, the From header field URI would match the <local> <identity> value and the from-tag parameter would match the local-tag attribute.

5.2.2. The <exclusive> element

The <exclusive> element, a child of the <dialog> element, is a boolean, which when true, indicates that the UA is not willing to accept an INVITE with a Join or Replaces header field targeted to the dialog described by the <dialog> element that is the parent of the <exclusive> element. For example, some shared appearance systems only allow call pickup when the call is on hold. In this case, the <exclusive> element should be set to "false" when the call is held and "true" when the call is not held, rather than having the "exclusive" value implied by the hold state.

It is important to note that this element is a hint. In order to prevent another UA from taking or joining a call, a UA can, in addition to setting the <exclusive> tag, not report full dialog information to the Appearance Agent. Not having the full dialog information (Call-ID, remote-tag, and local-tag) prevents another UA from constructing a Join or Replaces header field. Although a UA may set exclusive to true, the UA must still be ready to reject an INVITE Join relating to this dialog. If these dialog identifiers have already been shared with the Appearance Agent, the UA could send an INVITE Replaces to change them and then not report the new ones to the Appearance Agent.

If the proxy knows which dialogs are marked exclusive, the proxy MAY enforce this exclusivity by rejecting INVITE Join and INVITE Replaces requests containing those dialog identifiers with a 403 Forbidden response.

Note that exclusivity has nothing to do with appearance number selection or seizing - instead, it is about call control operations that can be performed on a dialog.

If the <exclusive> element is not present, it is assumed to be false.

5.2.3. The <joined-dialog> element

The <joined-dialog> element, a child of the <dialog> element, is used to convey dialog identifiers of any other dialogs which are joined (mixed or bridged) with the dialog. Only the UA which is the common endpoint of the mixed dialogs (and thus controlling the mixing operation) should include this element in publications to the Appearance Agent. Note that this element should still be used even when the Join header field was not used to join the dialogs. For example, two separate dialogs on a UA could be joined without any SIP call control operations. Joined dialogs will share the same appearance number.

If the <joined-dialog> element is not present, it is assumed that the dialog is not joined or to be joined to any other dialog.

5.2.4. The <replaced-dialog> element

The <replaced-dialog> element, a child of the <dialog> element, is used to convey dialog identifiers of any other dialogs which will be or have been replaced with this dialog. For example, a UA in the group picking up a call on another UA by sending an INVITE with Replaces would include this element for the replacing dialog. Replaced dialogs will share the same appearance number.

If the <replaced-dialog> element is not present, it is assumed that the dialog has not replaced or is not to replace to any other dialog.

5.3. Shared Appearance User Agents

User Agents that support the Shared Appearance feature use the dialog state package [RFC4235] with the shared appearance extensions and the 'shared' Event header field parameter defined in Section 13.

User Agents use the dialog package extensions in Section 5.2 along with SUBSCRIBE and NOTIFY [I-D.ietf-sipcore-rfc3265bis] and PUBLISH [RFC3903]. SUBSCRIBE, NOTIFY, and PUBLISH requests for the dialog event package include the 'shared' Event header field parameter as required by this specification.

The presence of the 'shared' Event header field parameter tells the Appearance Agent that the UA supports this specification.

Upon initialization, the UA MUST subscribe to the dialog event package of the AOR and refresh the subscription per the SIP Events Framework. If the SUBSCRIBE request fails, then no Appearance Agent may be present and this feature is not active for this AOR. The UA MAY periodically retry the subscription to see if conditions have changed at intervals no shorter than 4 hours.

Four hours was chosen to limit the subscription test to 6 per day per UA. Increasing this interval would reduce this failure traffic but take longer for a newly activated Appearance Agent to be discovered.

UAs can also use the presence of the 'shared' Event header field parameter in NOTIFYS to discover the presence of an Appearance Agent for the AOR.

User Agents which implement the shared appearances feature and call pickup, joining and bridging MUST support sending an INVITE with

Replaces [RFC3891] or Join [RFC3911]. The User Agent Client needs to include the to-tag and from-tag information in the Replaces or Join header so that the correct dialog will be matched by the User Agent Server per the rules in RFC 3891 and RFC 3911.

All User Agents which implement the shared appearances feature and support INVITE MUST support receiving an INVITE with a Replaces [RFC3891] or a Join [RFC3911] header field.

When publishing or notifying dialog package information, a UA includes the largest set of dialog identification available at the time of publication, with the exception that a UA may omit information if it wishes to prevent other UAs from joining or picking up a call. Dialog identification includes local and remote target URIs, call-id, to-tag, and from-tag. While this dialog identification information is optional in [RFC4235], it is essential in the shared appearance feature, allowing call control operations. When placing calls on hold, use the "+sip.rendering=no" feature tag to indicate this in dialog package notifications. Using the full SDP session description instead forces the endpoint to do a lot of extra parsing, unnecessarily complicating the code and inviting errors.

The accurate rendering of the idle/active/alerting/hold state of other UAs in the group is an important part of the shared appearance feature.

A UA that does not need to seize a particular appearance number (or doesn't care) would just send an INVITE as normal to place an outbound call.

If the call is an emergency call, a UA MUST never wait for a confirmed seizure before sending an INVITE. Instead, the emergency call MUST proceed without waiting for the PUBLISH transaction.

If a UA requires a particular appearance number, the a UA MUST send a dialog package PUBLISH request and wait for a 2xx response before sending the INVITE. This is required in the following situations:

1. When the user seizes a particular appearance number for an outgoing call (e.g. seizing the appearance and going "off-hook", if the UA's user interface uses this metaphor).
2. When the user has requested that an appearance number not be used for an outgoing call (i.e. during a consultation call, a 'service media' call such as for music on hold [I-D.worley-service-example] or for a call not considered part of the shared appearance group).

3. When the user has selected to join (or bridge) an existing call.
4. When the user has selected to replace (or take) an existing call.

Note that when a UA seizes an appearance prior to establishment of a dialog (#1 and #2 in above list), not all dialog information will be available. In particular, when a UA publishes an attempt to seize an appearance prior to knowing the destination URI, minimal or no dialog information may be available. For example, in some cases, only the local target URI for the call will be known and no dialog information. If the From tag and Call-ID were not present in the initial PUBLISH, a new PUBLISH MUST be sent as soon as this information is available.

The first publication will cause the Appearance Agent to reserve the appearance number for this UA. If the publication does not have any dialog identifiers (e.g. Call-ID, or local tag) the Appearance Agent cannot assign the appearance number to a particular dialog of the UA until the second publication which will contain some dialog identifiers.

This publication state is refreshed as described in [RFC3903] during the early dialog state or the Appearance Agent may reassign the appearance number. Once the dialog has transitioned to the confirmed state, no publication refreshes are necessary.

This specification assumes that the Appearance Agent has other means besides UA publication to learn about the state of UA dialogs. In this specification, PUBLISH is used to indicate desired and intended appearance number operations. Once a dialog transitions from early to confirmed, this role is over, and hence no publication refreshes are needed.

Appearance numbers are a shorthand label for active and pending dialogs related to an AOR. Many of the features and services built using this extension rely on the correct rendering of this information to the human user. In addition, the group nature of the feature means that the rendering must be similar between different vendors and different models. Failure to do so will greatly reduce the value and usefulness of these protocol extensions. In a correctly designed user interface for this feature, the appearances number for each active and pending dialog is explicitly (i.e. by appearance number) or implicitly (using a user interface metaphor that makes the numbering and ordering clear to the user) rendered to the user. The far end identity of each dialog (e.g. the remote party identity) is not a useful replacement for the appearance number. The state of each appearance is also be rendered (idle, active, busy, joined, etc.). UAs can tell that a set of dialogs are joined (bridged or mixed) together by the presence of one or more <joined-

dialog> elements containing other SIP dialog identifiers. Appearance numbers of dialogs can be learned by dialog package notifications containing the <appearance> element from the Appearance Agent or from the 'appearance' Alert-Info parameter in an incoming INVITE. Should they conflict, the dialog package notification takes precedence.

A user may select an appearance number but then abandon placing a call (go back on hook). In this case, the UA frees up the appearance number by removing the event state with a PUBLISH as described in [RFC3903]. A failure to do this will require extra unnecessary operations by the Appearance Agent, and also tie up appearance numbers which could otherwise be used by other UAs in the appearance group.

A UA SHOULD register against the AOR only if it is likely the UA will be answering incoming calls. If the UA is mainly going to be monitoring the status of the shared appearance group calls and picking or joining calls, the UA SHOULD only subscribe to the AOR and not register against the AOR. If a monitoring UA registers rather than just subscribing generates large amounts of unnecessary network traffic.

All subscribed UAs will receive dialog package NOTIFYs of trying state for incoming INVITES.

A UA MUST NOT insert an 'appearance' parameter into an Alert-Info header field in an INVITE or other request.

The Appearance Agent is solely responsible for doing this.

5.3.1. Appearance Numbers and Call Context

There are cases where two separate dialogs at a UA are not mixed but share the same 'context'. That is, they relate to each other and should not be treated the same as any other two dialogs within the group. One example of this is a 'consultation call' where a user puts an existing dialog on hold, then calls another user, before switching back to the original dialog. Another case, described below, occurs during transfer operations, where for a transient period, a UA is involved in dialogs with two other UAs, but the dialogs are related, and should not be treated as independent dialogs. These cases are best handled by not assigning an appearance number to a newly-created dialog when it shares a context with an existing dialog. But if the pre-existing dialog is terminated, its appearance number should be reassigned to the newly-created dialog.

A UA wanting to place a call but not have an appearance number assigned sends a PUBLISH before sending the INVITE. The PUBLISH does

not have an 'appearance' element present, but does have the 'shared' Event header field parameter present. If the Appearance Agent policy does not allow calls without an assigned appearance number, a 400 (Bad Request) response is sent by the Appearance Agent and the UA will republish either selecting/seizing an appearance number or send the INVITE without publishing, in which case the Appearance Agent will assign one.

Note that if an Appearance Agent rejects calls without an appearance number, certain operations such as consultation calls, transfer, and music on hold may be negatively impacted.

5.3.2. Appearance Numbers and Call Control

When an INVITE is generated to attempt to bridge or take a call (i.e. contains Join or Replaces with a dialog identifier of another dialog in the shared appearance group), the UA MUST first send a PUBLISH to the Appearance Agent. This PUBLISH will contain:

1. The appearance number of the joined or replaced call in the <appearance> element
2. If the dialog is being joined, the <joined-dialog> element will contain the dialog information from the Join header field
3. If the dialog is being replaced, the <replaced-dialog> element will contain the dialog information from the Replaces header field

Note that this information is provided to the Appearance Agent so that it can provide proper appearance assignment behavior. If the INVITE Join or Replaces was sent without publishing first, the Appearance Agent might assign a new appearance number to this INVITE, which would be a mistake. With Join, the publication has the <joined-dialog> element to prevent the Appearance Agent from generating a 400 (Bad Request) response due to the reuse of an appearance number. For Replaces, the purpose of the <replaced-dialog> is to prevent a race condition where the BYE could cause the appearance number to be released when it should stay with the replacing dialog.

5.3.3. Appearance Numbers and Transfer

During a transfer operation, it is important that the appearance number not change during the operation. Consider the example of Alice, a member of an appearance group, who is talking to Carol, who is outside the appearance group. Carol transfers Alice to David, who is also outside the appearance group. For example, if Alice is using appearance 3 for the session with Carol, the resulting session with David should also use appearance number 3. Otherwise, an appearance

number change can cause a "jump" on the UI and confusion to the user. There are two possible scenarios using the terminology of RFC 5589: Alice is the transferee in any type of transfer (receives the REFER) or the transfer target in an attended transfer (receives the INVITE with Replaces).

If Alice is the transferee, the triggered INVITE from the REFER is treated as a consultation call. Alice SHOULD publish requesting that the Appearance Agent not assign an appearance number for this INVITE. When the transfer completes, Alice SHOULD publish again to move the appearance number from the dialog with Carol to the dialog with David. If a PUBLISH is sent to move the appearance number, the publication MUST be sent prior to sending the BYE to Carol to avoid a race condition where the Appearance Agent reassigns the appearance number after seeing the BYE.

If Alice is the target, the incoming INVITE will contain a Replaces header field. As a result, the Appearance Agent will have reused the appearance number of the dialog with Carol, and this appearance number will continue to be used after the dialog with Carol has been terminated.

5.4. Appearance Agent

An Appearance Agent defined in this specification MUST implement a dialog package state agent for the UAs registered against the AOR. The Appearance Agent MUST support the appearance dialog package extensions defined in Section 5.2 and use the 'shared' Event header field parameter. The Appearance Agent MUST support publications and subscriptions for this event package.

The Appearance Agent MUST have a way of discovering the state of all dialogs associated with the AOR. If this information is not available from a call stateful proxy or Back-to-Back User Agent (B2BUA), the Appearance Agent can use the registration event package [RFC3680] to learn of UAs associated with the AOR and subscribe to their dialog event state. An Appearance Agent can also subscribe to a UAs dialog event state in order to reconstruct state. As a result, the registrar MUST support the registration event package. Dialog package notifications are recommended by RFC 4235 to "only contain information on the dialogs whose state or participation information has changed." This specification extends RFC 4235 as follows. The Appearance Agent SHOULD send dialog event state notifications whenever the following events happen to UAs in the AOR group:

1. A call is received, placed, answered, or terminated.

2. A call is placed on or off hold.
3. A call is joined or replaced.
4. An appearance number is reserved or released.

The Appearance Agent MUST allocate an appearance number for all incoming calls and send immediate notifications to the UAs subscribed to the shared group AOR. A new appearance number is allocated except for an incoming INVITE with a Join or Replaces header field. For this case, the appearance number should match the appearance number of the dialog being joined or replaced. If the INVITE Replaces or Join comes from outside the appearance group, the Appearance Agent will include a <joined-dialog> or <replaced-dialog> element in the NOTIFY containing the dialog information from the Replaces or Joined header field.

The Appearance Agent MUST be able to communicate with the forking proxy to learn about incoming calls and also to pass the appearance number to the proxy, or otherwise ensure the Alert-Info header field is included in the INVITE with the appropriate appearance number.

Note that UAs need to be able to handle incoming INVITEs without an appearance number assigned. This could be caused by a failure of the Appearance Agent or other error condition. Although the proper rendering of the INVITE may not be possible, this is better than ignoring or failing the INVITE.

An Appearance Agent SHOULD assign an appearance number to an outgoing dialog if a PUBLISH has not been received selecting/seizing a particular appearance number.

Note that if the appearance group has appearance-unaware UAs making calls, the Appearance Agent will still allocate appearance numbers for INVITEs sent by those UAs.

An Appearance Agent receiving a PUBLISH with an appearance number checks to make sure the publication is valid. An appearance number can be assigned to only one dialog unless there is a <joined-dialog> or <replaced-dialog> element indicating that the dialog will be/has been replaced or joined. A 400 (Bad Request) response is returned if the chosen appearance number is invalid, and an immediate NOTIFY SHOULD be sent to the UA containing full dialog event state.

An Appearance Agent receiving a PUBLISH without an appearance number but with the 'shared' Event header field parameter present interprets this as a request by the UA to not assign an appearance number. If the Appearance Agent policy does not allow this, a 400 (Bad Request) response is returned. If policy does allow this, a 200 OK response is returned and no appearance number is allocated. An Appearance

Agent does not have to share this dialog information (i.e. send a NOTIFY) with other UAs in the group as the information will not be rendered by the other UAs.

The Appearance Agent allocates an appearance number to a dialog from the time the appearance is requested via a PUBLISH or from the receipt of an INVITE, to the time when the last dialog associated with the appearance is terminated, including all dialogs which are joined or replaced. During the early dialog state, the Appearance Agent controls the rate of dialog state publication using the Expires header field in 200 OK responses to PUBLISH requests. An interval of 3 minutes is RECOMMENDED. After the dialog associated with the publication has been confirmed, the expiration of the publication state has no effect on the appearance allocation. If the publication contains no dialog state information, the Appearance Agent MUST reserve the appearance number for the UA but can not assign the appearance to any particular dialog of the UA. When the publication state is updated with any dialog information, the appearance number can then be assigned to the particular dialog. A UA which has been allocated an appearance number using a PUBLISH MAY free up the appearance number by removing the event state with a PUBLISH as described in [RFC3903].

If an INVITE is sent by a member of the group to the shared AOR (i.e. they call their own AOR), the Appearance Agent MUST assign two appearance numbers. The first appearance number will be the one selected or assigned to the outgoing INVITE. The second appearance number will be another one assigned by the Appearance Agent for the INVITE as it is forked back to the members of the group.

The is to preserve a common behavior in legacy systems.

If an INVITE is sent by a member of the group using the shared AOR or sent to the shared AOR and no appearance number is available, the proxy MAY reject the INVITE with a 403 Forbidden response code.

Appearance numbers are only used for dialogs in which one or more UAs associated with the group AOR is a participant. If an incoming INVITE to the group AOR is forwarded to another AOR, the appearance number is immediately freed up and can be assigned to another dialog.

6. XML Schema Definition

The 'appearance', 'joined-dialog', 'replaced-dialog', and 'exclusive' elements are defined within a new XML namespace URI. This namespace is "urn:ietf:params:xml:ns:sa-dialog-info". The schema for these elements is:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="urn:ietf:params:xml:ns:sa-dialog-info"
  xmlns="urn:ietf:params:xml:ns:sa-dialog-info"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">

  <xs:element name="joined-dialog" minOccurs="0"
    maxOccurs="unbounded">
    <xs:complexType>
      <xs:attribute name="call-id" type="xs:string"
        use="mandatory"/>
      <xs:attribute name="local-tag" type="xs:string"
        use="mandatory"/>
      <xs:attribute name="remote-tag" type="xs:string"
        use="mandatory"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="replaced-dialog" minOccurs="0"
    maxOccurs="unbounded">
    <xs:complexType>
      <xs:attribute name="call-id" type="xs:string"
        use="mandatory"/>
      <xs:attribute name="local-tag" type="xs:string"
        use="mandatory"/>
      <xs:attribute name="remote-tag" type="xs:string"
        use="mandatory"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="appearance" minOccurs="0" maxOccurs="1">
    <xs:simpleType type="xs:integer">
    </xs:simpleType>
  </xs:element>

  <xs:element name="exclusive" minOccurs="0" maxOccurs="1">
    <xs:simpleType type="xs:boolean">
    </xs:simpleType>
  </xs:element>
</xs:schema>
```

7. Alert-Info Appearance Parameter Definition

This specification extends RFC 3261 [RFC3261] to add an 'appearance' parameter to the Alert-Info header field, and to also allow proxies to modify or delete the Alert-Info header field.

The changes to RFC 3261 ABNF [RFC5234] are:

```
alert-param      = LAQUOT absoluteURI RAQUOT *( SEMI
                  (generic-param / appearance-param) )
appearance-param = "appearance" EQUAL 1*DIGIT
```

A proxy inserting an 'appearance' Alert-Info parameter follows normal Alert-Info policies. To indicate the appearance number for this dialog, the proxy adds the Alert-Info header field with the 'appearance' parameter to the INVITE. If an Alert-Info is already present, the proxy adds the 'appearance' parameter to the Alert-Info header field. If an appearance number parameter is already present (associated with another AOR or by mistake), the value is rewritten adding the new appearance number. There MUST NOT be more than one appearance parameter in an Alert-Info header field.

If no special ringtone is desired, a normal ringtone SHOULD be indicated using the urn:alert:service:normal in the Alert-Info, as per [I-D.ietf-salud-alert-info-urns]. The appearance number present in an Alert-Info header field SHOULD be rendered by the UA to the user, following the guidelines in Section 5.3. If the INVITE is forwarded to another AOR, the appearance parameter in the Alert-Info SHOULD be removed before forwarding outside the group.

The determination as to what value to use in the appearance parameter can be done at the proxy that forks the incoming request to all the registered UAs.

There are a variety of ways the proxy can use to determine what value it should use to populate this parameter. For example, the proxy could fetch this information by initiating a SUBSCRIBE request with Expires: 0 to the Appearance Agent for the AOR to fetch the list of lines that are in use. Alternatively, it could act like a UA that is a part of the appearance group and SUBSCRIBE to the State-Agent like any other UA. This would ensure that the active dialog information is available without having to poll on a need basis. It could keep track of the list of active calls for the appearance AOR based on how many unique INVITE requests it has forked to or received from the appearance AOR. Another approach would be for the Proxy to first send the incoming INVITE to the Appearance Agent which would redirect to the appearance group URI and escape the proper Alert-Info header field for the Proxy to

recurse and distribute to the other UAs in the group. The Appearance Agent needs to know about all incoming requests to the AOR in order to seize the appearance number. One way in which this could be done is for the Appearance Agent to register against the AOR with a higher q value. This will result in the INVITE being sent to the Appearance Agent first, then being offered to the UAs in the group.

8. User Interface Considerations

The "appearance number" allocated to a call is an important concept that enables calls to be handled by multiple devices with heterogeneous user interfaces in a manner that still allows users to see a consistent model. Careful treatment of the appearance number is essential to meet the expectations of the users. Also, rendering the correct call/appearance state to users is also important.

8.1. Appearance Number Rendering

Since different UAs have different user interface capabilities, it is usual to find that some UAs have restrictions that others do not. Perfect interoperability across all UAs is clearly not possible, but by careful design, interoperability up to the limits of each UA can be achieved.

The following guidelines suggest how the appearance number should be handled in three typical user interface implementations.

8.1.1. Single Appearance UAs

These devices are constrained by only having the capability of displaying status indications for a single appearance. The UA SHOULD still send messages annotated with appearance number "1". Any call indications for appearances other than for number "1" SHOULD be rejected with a 486 or 480 response. Note that this means that a single appearance UA cannot answer its own call to the shared AOR, since this call would use a second appearance number.

8.1.2. Dual Appearance UAs

These devices are essentially single appearance phones that implement call waiting. They have a very simple user interface that allows them to switch between two appearances (toggle or flash hook) and perhaps audible tones to indicate the status of the other appearance. Only appearance numbers "1" and "2" will be used by these UAs.

8.1.3. Shared Appearance UAs with Fixed Appearance Number

This UA is the typical 'business-class' hard-phone. A number of appearances are typically configured statically and labeled on buttons, and calls may be managed using these configured appearances. Any calls outside this range should be rejected, and not mapped to a free button. Users of these devices often seize specific appearance numbers for outgoing calls, and the UA will need to seize the appearance number and wait for confirmation from the Appearance Agent before proceeding with calls.

8.1.4. Shared Appearance UAs with Variable Appearance Number

This UA is typically a soft-phone or graphically rich user interface hard-phone. In these cases, even the idea of an appearance index may seem unnecessary. However, for these phones to be able to interwork successfully with other phone types, it is important that they still use the appearance index to govern the order of appearance of calls in progress. No specific guidance on presentation is given except that the order should be consistent. These devices can typically make calls without waiting for confirmation from the Appearance Agent on the appearance number.

8.1.5. Example User Interface Issues

The problems faced by each style of user interface are readily seen in this example:

1. A call arrives at the shared appearance group, and is assigned an appearance number of "1". All UAs should be able to render to the user the arrival of this call.
2. Another call arrives at the shared appearance group, and is assigned an appearance number of "2". The single appearance UA should not present this call to the user. Other user agents should have no problems presenting this call distinctly from the first call.
3. The first call clears, releasing appearance number "1". The single appearance UA should now be indicating no calls since it is unable to manage calls other than on the first appearance. Both shared appearance UAs should clearly show that appearance number "1" is now free, but that there is still a call on appearance number "2".
4. A third call arrives, and is assigned the appearance number of "1". All UAs should be able to render the arrival of this new call to the user. Multiple appearance UAs should continue to indicate the presence of the second call, and should also ensure that the presentation order is related to the appearance number and not the order of call arrival.

8.2. Call State Rendering

UAs that implement the shared appearance feature typically have a user interface that provides the state of other appearances in the group. As dialog state NOTIFYs from the Appearance Agent are processed, this information can be rendered. Even the simplest user interface typically has three states: idle, active, and hold. The idle state, usually indicated by lamp off, is indicated for an appearance when the appearance number is not associated with any dialogs, as reported by the Appearance Agent. The active state, usually indicated by a lamp on, is indicated by an appearance number being associated with at least one dialog, as reported by the Appearance Agent. The hold state, often indicated by a blinking lamp, means the call state from the perspective of the UA in the shared appearance group is hold. This can be determined by the presence of the "+sip.rendering=no" feature tag [RFC3840] with the local target URI. Note that the hold state of the remote target URI is not relevant to this display. For joined dialogs, the state is rendered as hold only if all local target URIs are indicated with the "+sip.rendering=no" feature tag.

9. Interoperability with non-Shared Appearance UAs

It is desirable to allow a basic UA that does not directly support shared appearance to be part of a shared appearance group. To support this the Proxy must collaborate with the Appearance Agent. This is not required in the basic shared appearance architecture, consequently shared appearance interoperability with non-shared appearance UAs will not be available in all shared appearance deployments.

First, a UA which does not support dialog events or the shared appearance feature will be discussed. Then, a UA which does support dialog events but not the shared appearance feature will be discussed.

9.1. Appearance Assignment

A UA that has no knowledge of appearances must will only have appearance numbers for outgoing calls if assigned by the Appearance Agent. If the non-shared appearance UA does not support Join or Replaces, all dialogs SHOULD be marked "exclusive" to indicate that these options are not available. Marking these dialogs "exclusive" provides a better user experience and avoids extra SIP messaging failures.

9.2. Appearance Release

In all cases the Appearance Agent must be aware of dialog lifetime to release appearances back into the group.

It is also desirable that any dialog state changes (such as hold, etc) be made available to other UAs in the group through the Dialog Event Package. If the Appearance Agent includes a proxy which Record-Routes for dialogs from the non-shared appearance aware UA, the Appearance Agent will know about the state of dialogs including hold, etc. This information could be determined from inspection of non-end-to-end-encrypted INVITE and re-INVITE messages and added to the dialog information conveyed to other UAs.

9.3. UAs Supporting Dialog Events but Not Shared Appearance

Interoperability with UAs which support dialog events but not the shared appearance feature is more straightforward. As before, all appearance number assignment must be done by the Appearance Agent. The Appearance Agent SHOULD still include appearance information in NOTIFYs - this UA will simply ignore this extra information. This type of UA will also ignore appearance number limitations and may attempt to Join or Replace dialogs marked exclusive. As a result, the Proxy or UAs need to reject such requests or the dialogs will get joined or taken.

10. Provisioning Considerations

UAs can automatically discover if this feature is active for an AOR by looking for the 'shared' Event header field parameter in a response to a dialog package SUBSCRIBE to the AOR, so no provisioning for this is needed.

The registrar will need to be provisioned to accept either first or third party registrations for the shared AOR. First party registration means the To and From URIs in the REGISTER request are the shared AOR URI. Third party registration means the To URI is the shared AOR URI and the From URI is a different AOR, perhaps that of the individual user. Either the credentials of the shared AOR or the user MUST be accepted by the registrar and the Appearance Agent, depending on the authorization policy in place for the domain.

If the Appearance Agent needs to subscribe to the dialog state of the UAs, then the Appearance Agent and the UAs need to be provisioned with credentials so the UAs can authenticate the Appearance Agent.

In some cases, UAs in the shared appearance group might have a UI

limitation on the number of appearances that can be rendered. Typically this will be hard phones with buttons/lamps instead of more flexible UIs. In this case, it can be useful for the Appearance Agent to know this maximum number. This can allow the Appearance Agent to apply policy when this limit is reached, e.g. deny a call. However, this mechanism does not provide any way to discover this by protocol means.

11. Example Message Flows

The next section shows call flow and message examples. The flows and descriptions are non-normative. Note that in these examples, all INVITES sent by a UA in the group will be From the shared AOR (sip:HelpDesk@example.com in this case), and all INVITES sent to the group will have a Request-URI of the shared AOR. Any other requests would not apply to this feature and would be handled using normal SIP mechanisms.

Note that the first twelve examples assume the Appearance Agent is aware of dialog state events. The example in Section 11.13 shows the case where this is not the case and as a result the Appearance Agent initiates a subscription to users of the shared AOR. Any of the other call flow examples could have shown this mode of operation as it is equally valid.

11.1. Registration and Subscription

Bob and Alice are in an appearance group identified by the shared appearance AOR sip:HelpDesk@example.com. Bob REGISTERS using contact sip:bob@ua2.example.com. Alice REGISTERS with contact sip:alice@ua1.example.com.

User Agents for Alice and Bob subscribe to the dialog package for the appearance AOR and publish dialog state to the Appearance Agent. Message exchanges between the Registrar, Appearance Agent, Alice, and Bob are shown below. The call flow examples below do not show the authentication of subscriptions, publications, and notifications. It should be noted that for security purposes, all publications and subscriptions must be authorized before they are accepted.

Also note that registrations and subscriptions must all be refreshed by Alice at intervals determined by the expiration intervals returned by the Registrar or Appearance Agent.

Registrar	Appearance Agent	Alice	Bob

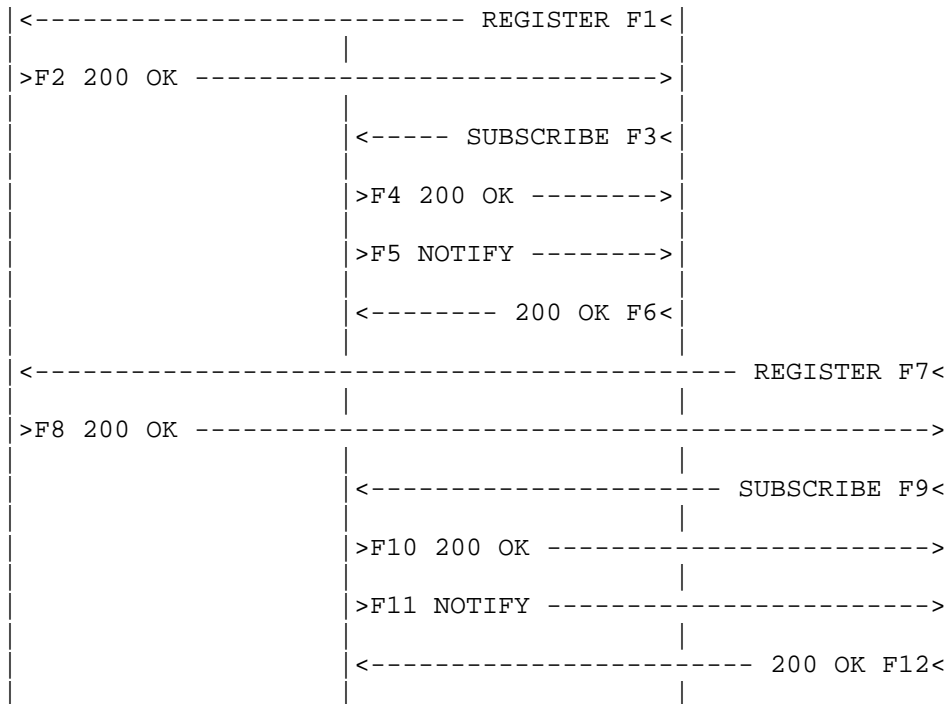


Figure 1.

F1-F2: Alice registers AOR with
contact: <sip:alice@ual.example.com>

F1 Alice ----> Registrar

```

REGISTER sip:registrar.example.com SIP/2.0
Via: SIP/2.0/UDP ual.example.com;branch=z9hG4bK527b54da8ACC7B09
From: <sip:alice@example.com>;tag=CDF9A668-909E2BDD
To: <sip:HelpDesk@example.com>
CSeq: 2 REGISTER
Call-ID: d3281184-518783de-cc23d6bb
Contact: <sip:alice@ual.example.com>
Max-Forwards: 70
Expires: 3600
Content-Length: 0
  
```

F2 Registrar ----> Alice

```

SIP/2.0 200 OK
Via: SIP/2.0/UDP ual.example.com;branch=z9hG4bK527b54da8ACC7B09
  
```

CSeq: 2 REGISTER
Call-ID: d3281184-518783de-cc23d6bb
From: <sip:alice@example.com>;tag=CDF9A668-909E2BDD
To: <sip:HelpDesk@example.com>;tag=1664573879820199
Contact: <sip:alice@ual.example.com>;expires=3600
Content-Length: 0

F3 to F6: Alice also subscribes to the events associated with the Appearance AOR. Appearance Agent notifies Alice of the status.

F3 Alice ----> Appearance Agent

SUBSCRIBE sip:HelpDesk@example.com SIP/2.0
Via: SIP/2.0/UDP ual.example.com;branch=z9hG4bKf10fac97E7A76D6A
From: <sip:alice@example.com>;tag=925A3CAD-CEBB276E
To: <sip:HelpDesk@example.com>
CSeq: 91 SUBSCRIBE
Call-ID: ef4704d9-bb68aa0b-474c9d94
Contact: <sip:alice@ual.example.com>
Event: dialog;shared
Accept: application/dialog-info+xml
Max-Forwards: 70
Expires: 3700
Content-Length: 0

F4 Appearance Agent ----> Alice

SIP/2.0 200 OK
Via: SIP/2.0/UDP ual.example.com;branch=z9hG4bKf10fac97E7A76D6A
CSeq: 91 SUBSCRIBE
Call-ID: ef4704d9-bb68aa0b-474c9d94
From: <sip:alice@example.com>;tag=925A3CAD-CEBB276E
To: <sip:HelpDesk@example.com>;tag=1636248422222257
Allow-Events: dialog
Expires: 3700
Contact: <sip:appearanceagent.example.com>
Content-Length: 0

F5 Appearance Agent ----> Alice

NOTIFY sip:alice@ual.example.com SIP/2.0
From: <sip:HelpDesk@example.com>;tag=1636248422222257
To: <sip:alice@example.com>;tag=925A3CAD-CEBB276E
Call-ID: ef4704d9-bb68aa0b-474c9d94
CSeq: 232 NOTIFY

Via: SIP/2.0/UDP appearanceagent.example.com;branch=z9hG4bK1846
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Event: dialog;shared
Subscription-State: active;expires=3000
Contact: <appearanceagent.example.com>
Content-Length: ...

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
              version="40"
              state="full"
              entity="sip:HelpDesk@example.com">
</dialog-info>
```

F6 Alice ----> Appearance Agent

SIP/2.0 200 OK
Via: SIP/2.0/UDP appearanceagent.example.com;branch=z9hG4bK1846
From: <sip:HelpDesk@example.com>;tag=163624842222257
To: <sip:alice@example.com>;tag=925A3CAD-CEBB276E
CSeq: 232 NOTIFY
Call-ID: ef4704d9-bb68aa0b-474c9d94
Contact: <sip:alice@ua1.example.com>
Content-Length: 0

F7 Bob ----> Registrar

REGISTER sip:registrar.example.com SIP/2.0
Via: SIP/2.0/UDP ua2.example.com;branch=z9hG4b53b54d87B
From: <sip:bob@example.com>;tag=34831131
To: <sip:HelpDesk@example.com>
CSeq: 72 REGISTER
Call-ID: 139490230230249348
Contact: <sip:bob@ua2.example.com>
Max-Forwards: 70
Expires: 3600
Content-Length: 0

F8 Registrar ----> Bob

SIP/2.0 200 OK
Via: SIP/2.0/UDP ua2.example.com;branch=z9hG4b53b54d87B
From: <sip:bob@example.com>;tag=34831131
To: <sip:HelpDesk@example.com>;tag=fkwlwqil

```

CSeq: 72 REGISTER
Call-ID: 139490230230249348
Contact: <sip:alice@ua1.example.com>;expires=3200
Contact: <sip:bob@ua2.example.com>;expires=3600
Content-Length: 0

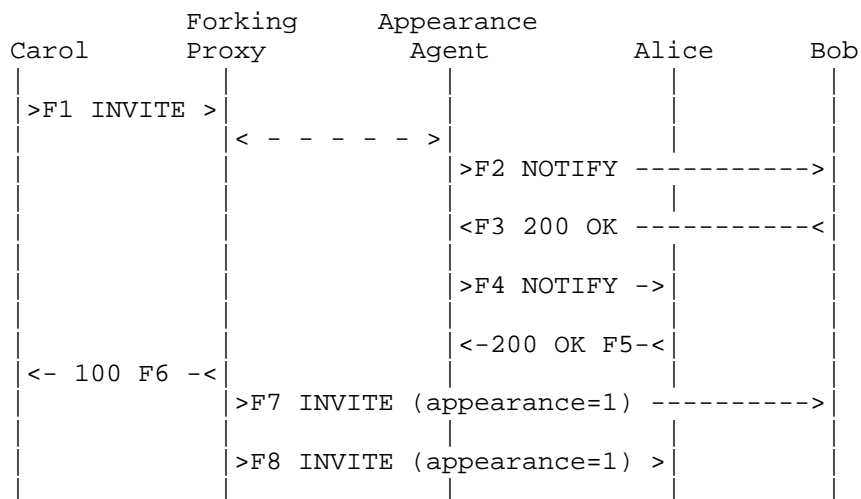
```

11.2. Appearance Selection for Incoming Call

In the call flow below Bob and Alice are in an appearance group. Carol places a call to the appearance group AOR. The Appearance Agent sends NOTIFYs to Alice and Bob telling them what appearance the call is using. Both Alice and Bob's devices are alerted of the incoming call. Bob answers the call.

Note that it is possible that both Alice and Bob answer the call and send 200 OK responses to Carol. It is up to Carol to resolve this situation. Typically, Carol will send ACKs to both 200 OKs but send a BYE to terminate one of the dialogs. As a result, either Alice or Bob will receive the BYE and publish that their dialog is over. However, if Carol answers both Alice and Bob and keeps both dialogs active, then the Appearance Agent will need to resolve the situation by moving either Alice or Bob's dialog to a different appearance.

All NOTIFY messages in the call flow below carry dialog events and only dialog states are mentioned for simplicity. For brevity, the details of some messages are not shown below. Note that the order of F2 - F5 and F7 - F8 could be reversed.



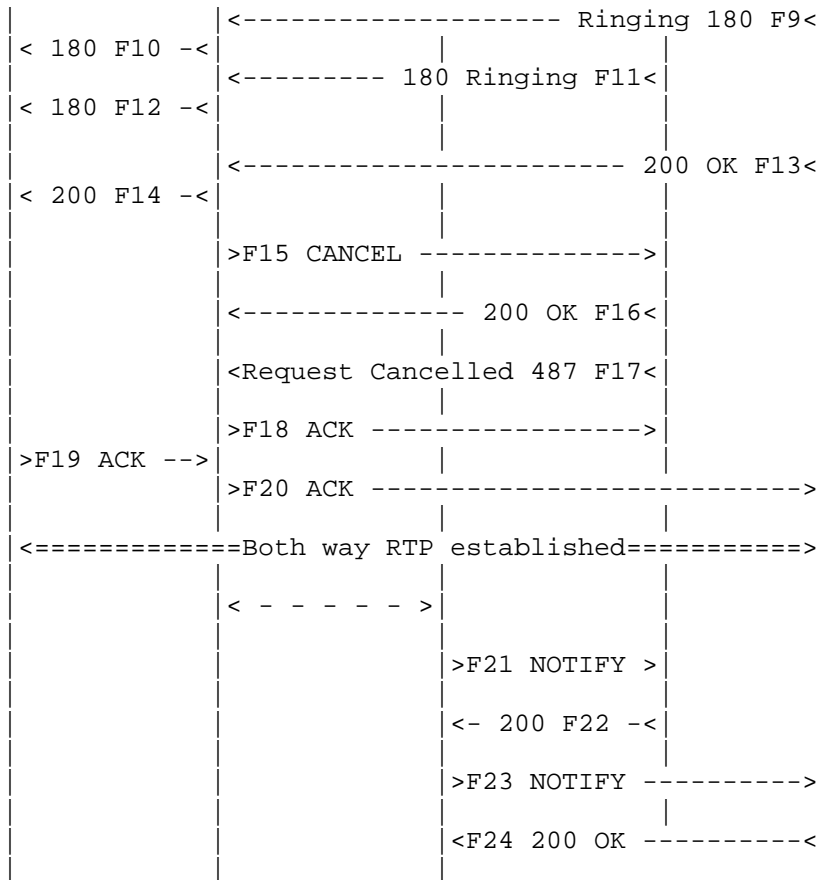


Figure 2.

F4 Appearance Agent ----> Alice

```

NOTIFY sip:alice@ual.example.com SIP/2.0
From: <sip:HelpDesk@example.com>;tag=151702541050937
To: <sip:alice@example.com>;tag=18433323-C3D237CE
Call-ID: 1e361d2f-a9f51109-bafe31d4
CSeq: 12 NOTIFY
Via: SIP/2.0/UDP appearanceagent.example.com;branch=z9hG4bK1403
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Event: dialog;shared
Subscription-State: active;expires=2800
Contact: <appearanceagent.example.com>
Content-Length: ...
  
```

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
  version="13"
  state="partial"
  entity="sip:HelpDesk@example.com">
  <dialog id="2a7294823093f5274e3fd2ec54a2d76c"
    call-id="14-1541707345"
    remote-tag="44BAD75D-E3128D42"
    direction="recipient">
    <sa:appearance>1</sa:appearance>
    <state>trying</state>
    <remote>
      <identity>sip:carol@ua.example.com</identity>
    </remote>
  </dialog>
</dialog-info>
```

F7 Proxy ----> Bob

```
INVITE sip:bob@ua2.example.com SIP/2.0
Via: SIP/2.0/UDP ua3.example.com;branch=z9hG4bK4324ea
Via: SIP/2.0/UDP proxy.example.com;branch=z9hG4bK38432ji
From: <sip:carol@example.com>;tag=44BAD75D-E3128D42
To: <sip:HelpDesk@example.com>
CSeq: 106 INVITE
Call-ID: 14-1541707345
Contact: <sip:carol@ua3.example.com>
Max-Forwards: 69
Alert-Info: <urn:alert:service:normal>;appearance=1
Content-Type: application/sdp
Content-Length: ...
```

```
v=0
o=- 1102980499 1102980499 IN IP4 ua3.example.com
s=
c=IN IP4 ua3.example.com
t=0 0
m=audio 2238 RTP/AVP 0 8 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
```

F21 Appearance Agent ----> Alice

```
NOTIFY sip:alice@ua1.example.com SIP/2.0
```

```

From: <sip:HelpDesk@example.com>;tag=151702541050937
To: <sip:alice@example.com>;tag=18433323-C3D237CE
Call-ID: 1e361d2f-a9f51109-bafe31d4
CSeq: 13 NOTIFY
Via: SIP/2.0/UDP appearanceagent.example.com;branch=z9hG4bK4164F03j
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Event: dialog;shared
Subscription-State: active;expires=2500
Contact: <appearanceagent.example.com>
Content-Length: ...

```

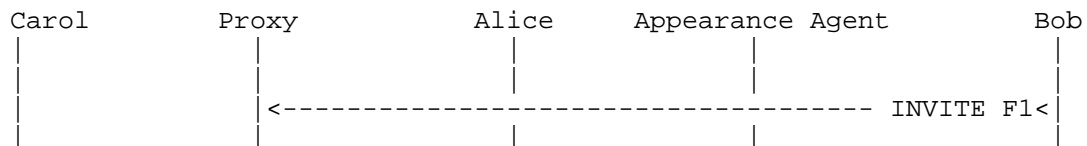
```

<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
  version="17"
  state="partial"
  entity="sip:HelpDesk@example.com">
  <dialog id="2a7294823093f5274e3fd2ec54a2d76c"
    call-id="14-1541707345"
    remote-tag="44BAD75D-E3128D42"
    local-tag="7349dsfjkFD03s"
    direction="recipient">
    <sa:appearance>l</sa:appearance>
    <state>confirmed</state>
    <local>
      <target>sip:bob@ua2.example.com</target>
    </local>
    <remote>
      <identity>sip:carol@ua.example.com</identity>
    </remote>
  </dialog>
</dialog-info>

```

11.3. Outgoing Call without Appearance Seizure

In this scenario, Bob's UA places a call without first selecting/seizing an appearance number. After Bob sends the INVITE, the appearance assigns an appearance number for it and notifies both Alice and Bob.



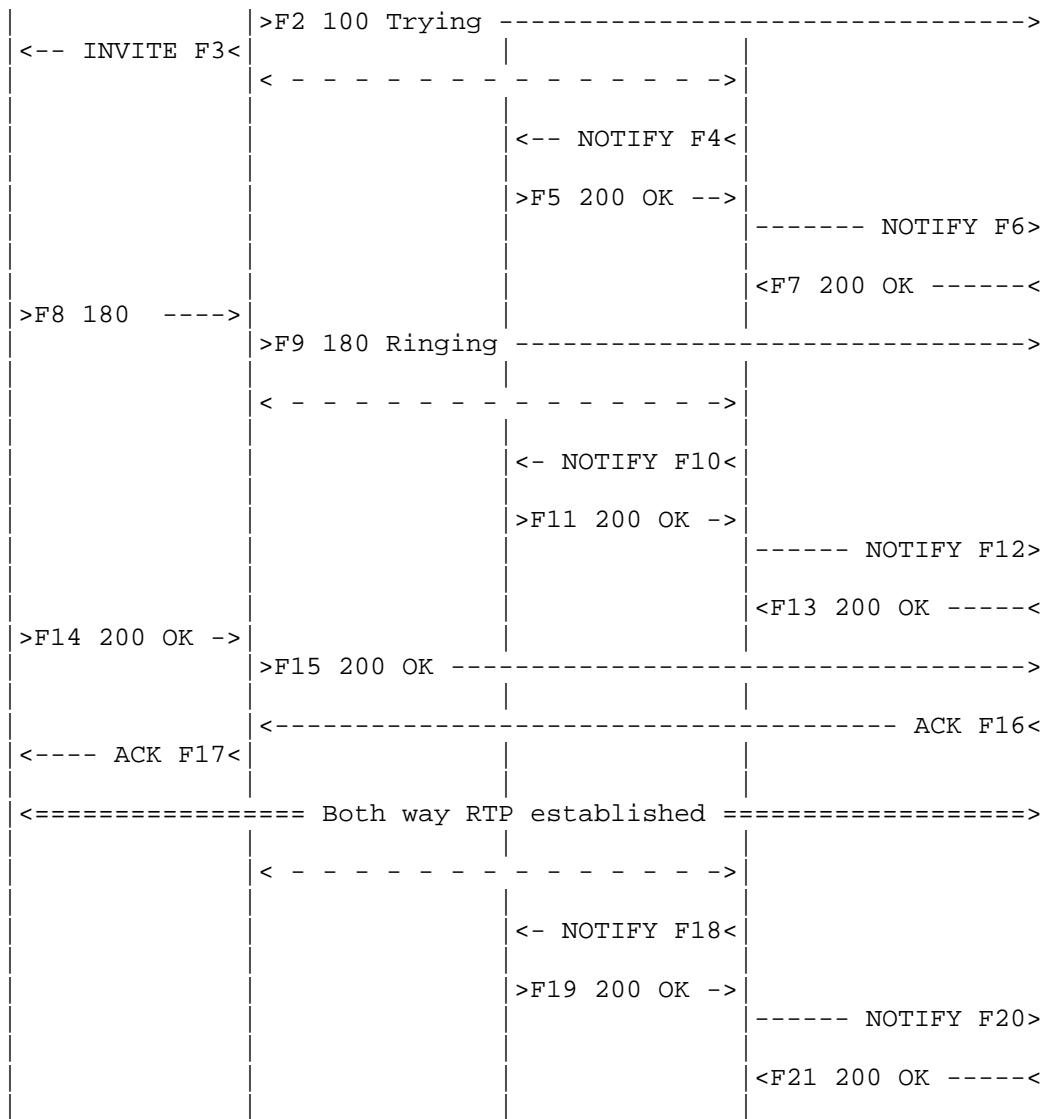


Figure 3.

F1 Bob -----> Proxy

```

INVITE sip:carol@example.com SIP/2.0
Via: SIP/2.0/UDP ua2.example.com;branch=z9hG4bK98c87c52123A08BF
From: <sip:HelpDesk@example.com>;tag=15A3DE7C-9283203B
To: <sip:carol@example.com>
  
```

```
CSeq: 1 INVITE
Call-ID: f3b3cbd0-a2c5775e-5df9f8d5
Contact: <sip:bob@ua2.example.com>
Max-Forwards: 70
Content-Type: application/sdp
Content-Length: 223

v=0
o=- 1102980499 1102980499 IN IP4 ua2.example.com
s=IP SIP UA
c=IN IP4 ua2.example.com
t=0 0
a=sendrecv
m=audio 2236 RTP/AVP 0 8 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
```

F4 Appearance Agent ----> Alice

```
NOTIFY sip:alice@ua1.example.com SIP/2.0
Via: SIP/2.0/UDP appearanceagent.example.com;branch=z9hG4bK8ld84f62
From: <sip:HelpDesk@example.com>;tag=1636248422222257
To: <sip:alice@example.com>;tag=925A3CAD-CEBB276E
Call-ID: ef4704d9-bb68aa0b-474c9d94
CSeq: 233 NOTIFY
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Event: dialog;shared
Subscription-State: active;expires=2200
Contact: <appearanceagent.example.com>
Content-Length: ...
```

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
  version="27"
  state="partial"
  entity="sip:HelpDesk@example.com">
  <dialog id="fa02538339df3ce597f9e3e3699e28fc"
    call-id="f3b3cbd0-a2c5775e-5df9f8d5"
    local-tag="15A3DE7C-9283203B" direction="initiator">
    <sa:appearance>1</sa:appearance>
    <sa:exclusive>false</sa:exclusive>
    <state>trying</state>
    <local>
      <target uri="sip:bob@ua2.example.com">
```

```

        </target>
      </local>
    </dialog>
  </dialog-info>

```

F6 Appearance Agent ----> Bob

```

NOTIFY sip:bob@ua1.example.com SIP/2.0
From: <sip:HelpDesk@example.com>;tag=497585728578386
To: <sip:bob@example.com>;tag=633618CF-B9C2EDA4
Call-ID: a7d559db-d6d7dcad-311c9e3a
CSeq: 7 NOTIFY
Via: SIP/2.0/UDP appearanceagent.example.com
    ;branch=z9hG4bK1711759878512309
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Event: dialog;shared
Subscription-State: active;expires=2000
Contact: <sip:appearanceagent.example.com>
Content-Length: ...

```

```

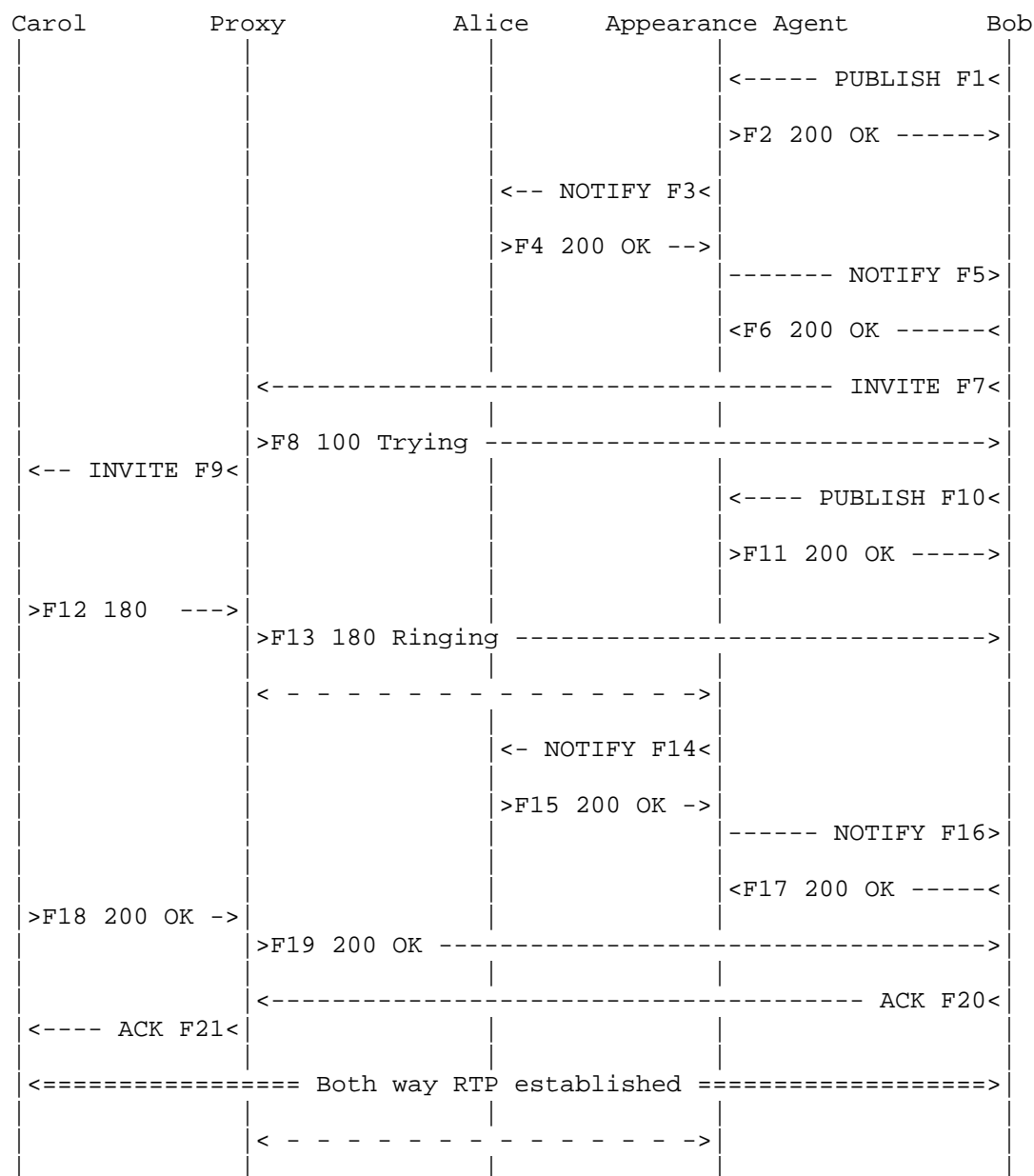
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
  version="78"
  state="partial"
  entity="sip:HelpDesk@example.com">
  <dialog id="02538339hfgdf3ce597f9e3egkl3699e28fc"
    call-id="f3b3cbd0-a2c5775e-5df9f8d5"
    local-tag="15A3DE7C-9283203B"    direction="initiator">
    <sa:appearance>1</sa:appearance>
    <sa:exclusive>false</sa:exclusive>
    <state>trying</state>
    <local>
      <target uri="sip:bob@ua2.example.com">
      </target>
    </local>
  </dialog>
</dialog-info>

```

11.4. Outgoing Call with Appearance Seizure

In this scenario, Bob's UA sends out a dialog event PUBLISH with state (trying) selecting/seizing an appearance number before sending the INVITE. After receiving the 200 OK from the Appearance Agent confirming the appearance number, Bob's UA sends the INVITE to Carol and establishes a session. For brevity, details of some of the

messages are not included in the message flows. Bob's UA puts as much of the dialog information from F7 as can be determined in advance. In this case, the minimum of the Contact URI is included which allows the Appearance Agent to correlate the INVITE with the PUBLISH.



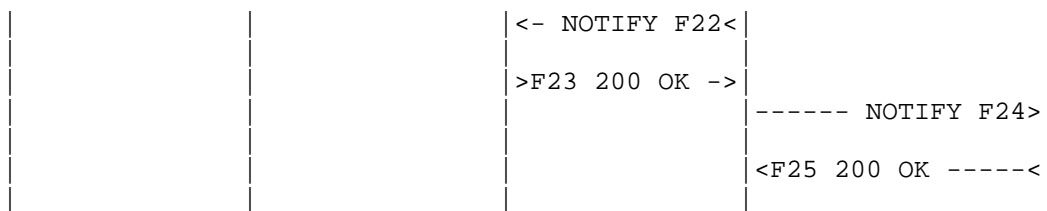


Figure 4.

F1 to F4: Bob uses the shared appearance of the Help Desk on his UA to place an outgoing call (e.g., he goes off-hook). Before sending the outgoing INVITE request, Bob publishes to the Appearance Agent reserving appearance number 1. The Appearance Agent notifies Alice (and all other UAs, including Bob) of the event by sending NOTIFYs.

F1 Bob ----> Appearance Agent

```

PUBLISH sip:HelpDesk@example.com SIP/2.0
Via: SIP/2.0/UDP ua2.example.com;branch=z9hG4bK61314d6446383E79
From: <sip:bob@example.com>;tag=44150CC6-A7B7919D
To: <sip:HelpDesk@example.com>
CSeq: 7 PUBLISH
Call-ID: 44fwF144-F12893K38424
Contact: <sip:bob@ua2.example.com>
Event: dialog;shared
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Content-Length: ...
  
```

```

<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
  version="6"
  state="full"
  entity="sip:HelpDesk@example.com">
  <dialog id="id3d4f9c83" direction="initiator">
    <sa:appearance>1</sa:appearance>
    <sa:exclusive>false</sa:exclusive>
    <state>trying</state>
    <local>
      <target uri="sip:bob@ua2.example.com">
        </target>
      </local>
    </dialog>
  </dialog-info>
  
```


F2 Appearance Agent ----> Bob

SIP/2.0 200 OK
Via: SIP/2.0/UDP ua2.example.com;branch=z9hG4bK61314d6446383E79
From: <sip:bob@example.com>;tag=44150CC6-A7B7919D
To: <sip:HelpDesk@example.com>
CSeq: 7 PUBLISH
Call-ID: 44fwF144-F12893K38424
Contact: <sip:bob@ua2.example.com>
Event: dialog;shared
SIP-Etag: 482943245
Allow-Events: dialog
Expires: 60
Content-Length: 0

F7 Bob ---> Proxy

INVITE sip:carol@example.com SIP/2.0
Via: SIP/2.0/UDP ua2.example.com;branch=z9hG4bK342122
Max-Forwards: 70
From: <sip:HelpDesk@example.com>;tag=15A3DE7C-9283203B
To: <sip:carol@example.com>
Call-ID: f3b3cbd0-a2c5775e-5df9f8d5
CSeq: 31 INVITE
Contact: <sip:bob@ua2.example.com>
Content-Type: application/sdp
Content-Length: ...

(SDP Not Shown)

F10 Bob ----> Appearance Agent

PUBLISH sip:HelpDesk@example.com SIP/2.0
Via: SIP/2.0/UDP ua2.example.com;branch=z9hG4bK6d644638E7
From: <sip:bob@example.com>;tag=0CCf6-A7FdsB79D
To: <sip:HelpDesk@example.com>
CSeq: 437 PUBLISH
Call-ID: fwF14d4-F1FFF2F2893K38424
Contact: <sip:bob@ua2.example.com>
Event: dialog;shared
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Content-Length: ...

<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"

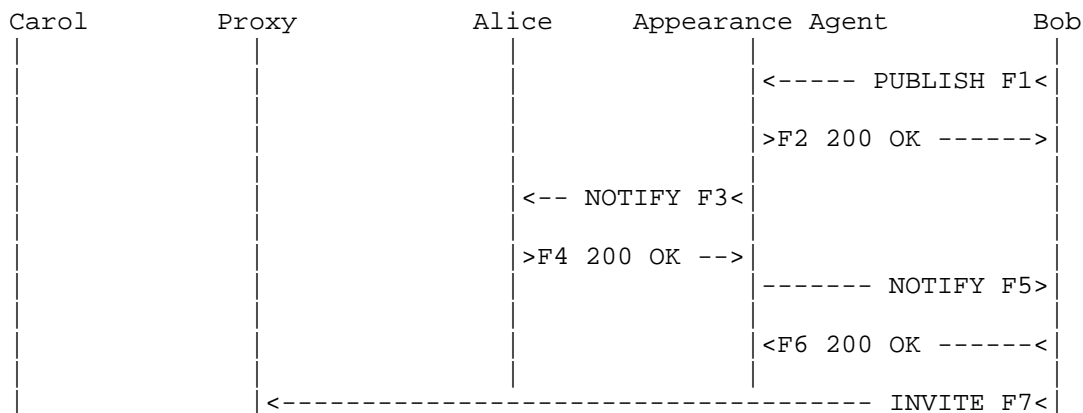
```

        xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
        version="6"
        state="full"
        entity="sip:HelpDesk@example.com">
<dialog id="id3d4f9c83"
    call-id="f3b3cbd0-a2c5775e-5df9f8d5"
    local-tag="15A3DE7C-9283203B"
                                direction="initiator">
    <sa:appearance>1</sa:appearance>
    <sa:exclusive>false</sa:exclusive>
    <state>trying</state>
    <local>
        <target uri="sip:bob@ua2.example.com">
            </target>
    </local>
    <remote>
        <identity uri="sip:carol@example.com">
            </identity>
    </remote>
</dialog>
</dialog-info>

```

11.5. Outgoing Call without using an Appearance Number

In this scenario, Bob's UA sends out a dialog event PUBLISH with state (trying) indicating that he does not want to utilize an appearance number for this dialog. The PUBLISH does not have an appearance element but does have the 'shared' Event header field parameter. As a result, the Appearance Agent knows the UA does not wish to use an appearance number for this call. If the Appearance Agent does not wish to allow this, it would reject the PUBLISH with a 400 (Bad Request) response and the UA would know to re-PUBLISH selecting/seizing an appearance number.



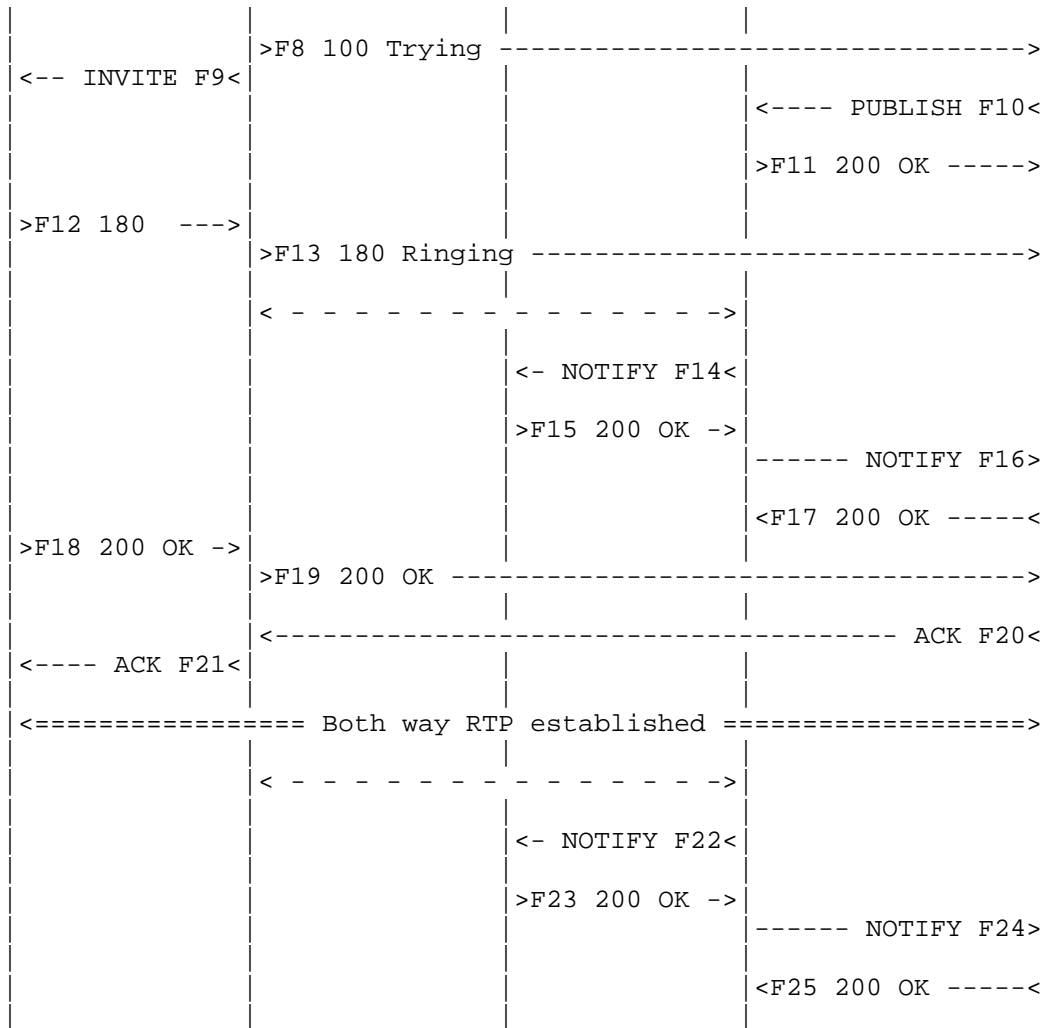


Figure 5.

F1 Bob ----> Appearance Agent

```

PUBLISH sip:appearanceagent.example.com SIP/2.0
Via: SIP/2.0/UDP ua2.example.com;branch=z9hG4bK61314d6446383E79
From: <sip:bob@example.com>;tag=4415df82k39sf
To: <sip:HelpDesk@example.com>
CSeq: 7 PUBLISH
Call-ID: 44fwF144-F12893K38424
Contact: <sip:bob@ua2.example.com>
  
```

```

Event: dialog;shared
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Content-Length: ...

<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
             version="6"
             state="full"
             entity="sip:HelpDesk@example.com">
  <dialog id="id3d4f9c83" direction="initiator">
    <sa:exclusive>false</sa:exclusive>
    <state>trying</state>
    <local>
      <target uri="sip:bob@ua2.example.com">
        </target>
      </local>
    </dialog>
  </dialog-info>

```

Note that F7 would be the same as the previous example.

11.6. Appearance Release

Bob and Carol are in a dialog, created, for example as in Section 11.3. Carol sends a BYE to Bob to terminate the dialog and the Appearance Agent de-allocates the appearance number used, sending notifications out to the UAs in the shared group.

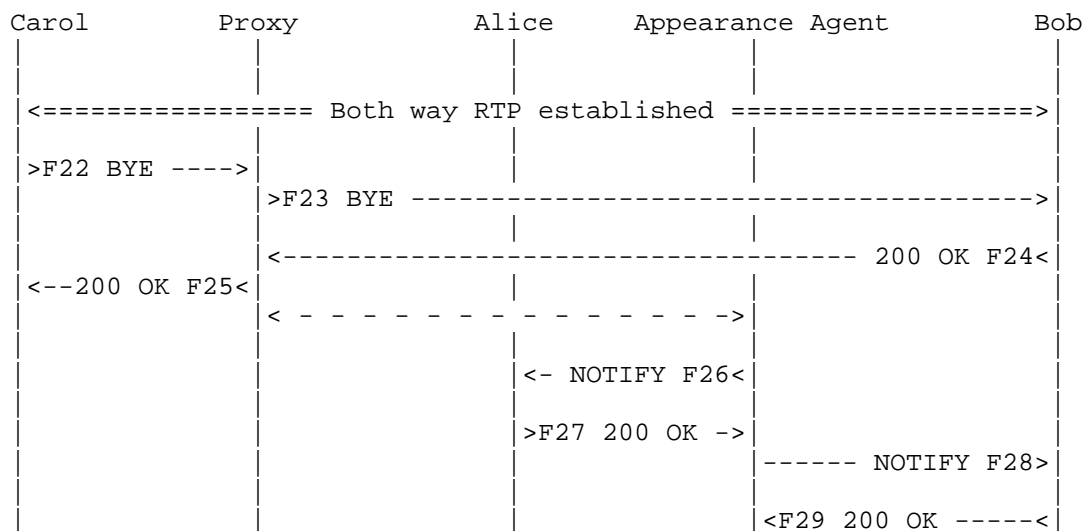


Figure 6.

F28 Appearance Agent ----> Bob

```

NOTIFY sip:bob@ua1.example.com SIP/2.0
From: <sip:HelpDesk@example.com>;tag=497585728578386
To: <sip:bob@example.com>
Call-ID: a7d559db-d6d7dcad-311c9e3a
CSeq: 7 NOTIFY
Via: SIP/2.0/UDP appearanceagent.example.com
    ;branch=z9hG4bK759878512309
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Event: dialog;shared
Subscription-State: active;expires=1800
Contact: <sip:appearanceagent.example.com>
Content-Length: ...

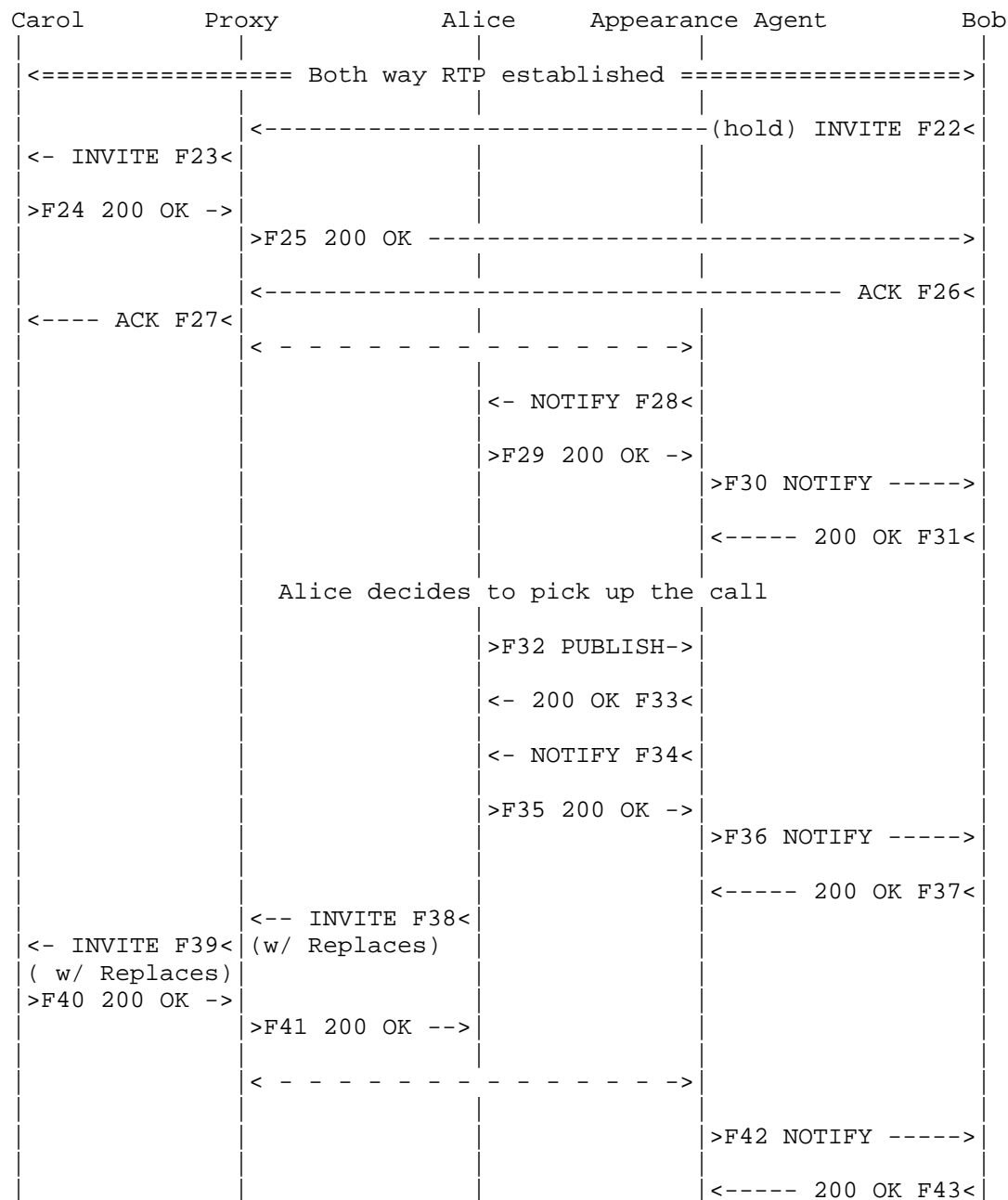
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
             xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
             version="27"
             state="partial"
             entity="sip:HelpDesk@example.com">
  <dialog id="fa02538339df3ce597f9e3e3699e28fc"
        call-id="f3b3cbd0-a2c5775e-5df9f8d5"
        local-tag="15A3DE7C-9283203B"
        remote-tag="65a98f7c-1dd2-11b2-88c6-b0316298f7c"
              direction="initiator">
    <sa:appearance>1</sa:appearance>
    <sa:exclusive>false</sa:exclusive>
    <state>terminated</state>
    <local>
      <target uri="sip:bob@ua2.example.com">
    </target>
    </local>
  </dialog>
</dialog-info>

```

11.7. Appearance Pickup

In this scenario, Bob has an established dialog with Carol created using the call flows of Figure 1 or Figure 2. Bob then places Carol on hold. Alice receives a notification of this and renders this on Alice's UI. Alice subsequently picks up the held call and has a established session with Carol. Finally, Carol hangs up. Alice must PUBLISH F32 to indicate that the INVITE F38 will be an attempt to

pickup the dialog between Carol and Bob, and hence may use the same appearance number. This example also shows Secure SIP (sips) being used.



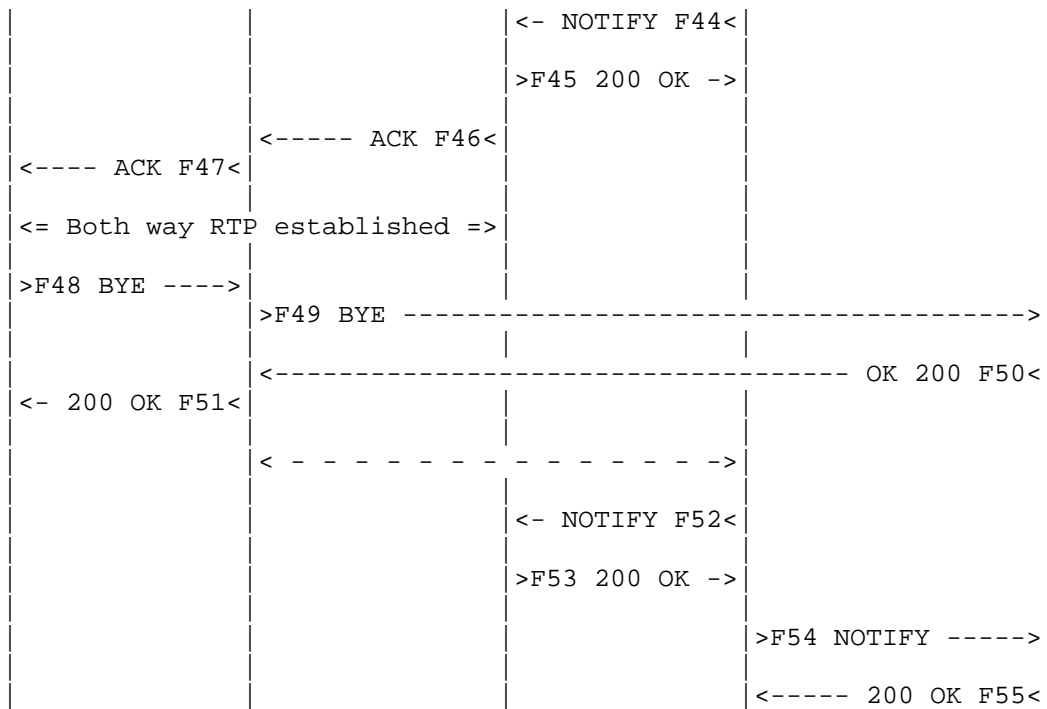


Figure 7.

F28 Appearance ----> Alice

```

NOTIFY sips:alice@ual.example.com SIP/2.0
From: <sips:HelpDesk@example.com>;tag=151702541050937
To: <sips:alice@example.com>;tag=18433323-C3D237CE
Call-ID: 1e361d2f-a9f51109-bafe31d4
CSeq: 12 NOTIFY
Via: SIP/2.0/TLS appearanceagent.example.com
    ;branch=z9hG4bK1403
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Event: dialog;shared
Subscription-State: active;expires=1800
Contact: <sips:appearanceagent.example.com>
Content-Length: ...

<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
    xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
    version="10"
  
```

```

        state="partial"
        entity="sips:HelpDesk@example.com">
<dialog id="id3d4f9c83"
  call-id="f3b3cbd0-a2c5775e-5df9f8d5"
  local-tag="15A3DE7C-9283203B"
  remote-tag="65a98f7c-1dd2-11b2-88c6-b0316298f7c"
  direction="initiator">
  <sa:appearance>1</sa:appearance>
  <sa:exclusive>false</sa:exclusive>
  <state>active</state>
  <local>
    <target uri="sips:bob@ua2.example.com">
      <param pname="+sip.rendering" pval="no" />
    </target>
  </local>
  <remote>
    <identity>sips:carol@example.com</identity>
    <target uri="sips:carol@ua3.example.com" />
  </remote>
</dialog>
</dialog-info>

```

F32 Alice ----> Appearance Agent

```

PUBLISH sips:HelpDesk@example.com SIP/2.0
Via: SIP/2.0/TLS ua2.example.com;branch=z9hG4bKa5d6cf61F5FBC05A
From: <sips:HelpDesk@example.com>;tag=44150CC6-A7B7919D
To: <sips:alice@example.com>;tag=428765950880801
CSeq: 11 PUBLISH
Call-ID: 87837Fkw87asfds
Contact: <sips:alice@ua2.example.com>
Event: dialog;shared
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Content-Length: ...

```

```

<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
  version="10"
  state="full"
  entity="sips:HelpDesk@example.com">
  <dialog id="id3d4f9c83"
    call-id="3d57cd17-47deb849-dca8b6c6"
    local-tag="8C4183CB-BCEAB710" >
    <sa:appearance>1</sa:appearance>
    <sa:exclusive>false</sa:exclusive>

```



```
<sa:replaced-dialog
  call-id="f3b3cbd0-a2c5775e-5df9f8d5"
  from-tag="15A3DE7C-9283203B"
  to-tag="65a98f7c-1dd2-11b2-88c6-b03162323164+65a98f7c" />
<state>trying</state>
<local>
  <target uri="sips:alice@ua1.example.com">
    <param pname="+sip.rendering" pval="yes"/>
  </target>
</local>
<remote>
  <target uri="sips:carol@ua3.example.com" />
</remote>
</dialog>
</dialog-info>
```

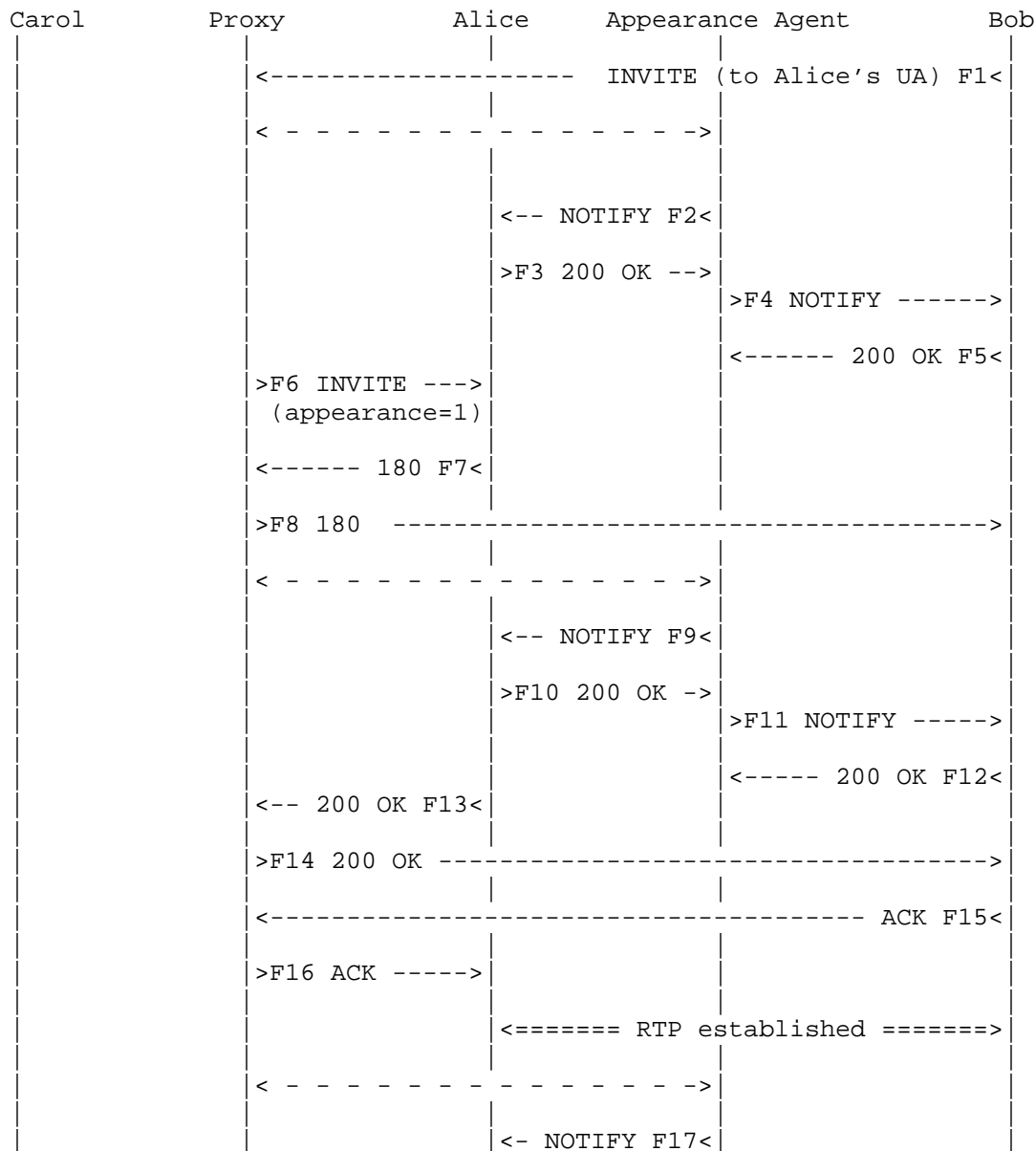
F38 Alice ----> Proxy

```
INVITE sips:carol@example.com SIP/2.0
Via: SIP/2.0/TLS ua1.example.com;branch=z9hG4bK4ea695b5B376A60C
From: <sips:HelpDesk@example.com>;tag=8C4183CB-BCEAB710
To: <sips:carol@example.com:5075>
CSeq: 1 INVITE
Call-ID: 3d57cd17-47deb849-dca8b6c6
Contact: <sips:alice@ua1.example.com>
<all-one-line>
Replaces: f3b3cbd0-a2c5775e-5df9f8d5;to-tag=65a98f7c
-1dd2-11b2-88c6-b03162323164+65a98f7c;from-tag=15A3DE7C-9283203B
</all-one-line>
Max-Forwards: 70
Content-Type: application/sdp
Content-Length: 223
```

```
v=0
o=- 1102980497 1102980497 IN IP4 ua1.example.com
s=IP SIP UA
c=IN IP4 ua1.example.com
t=0 0
a=sendrecv
m=audio 2238 RTP/AVP 0 8 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
```

11.8. Calls between UAs within the Group

In this scenario, Bob calls Alice who is also in the Appearance group. Only one appearance number is used for this dialog. This example also shows the use of the 'exclusive' tag to indicate that other UAs in the group can not join or take this dialog.



		>F18 200 OK ->	
			>F19 NOTIFY ----->
			<----- 200 OK F24<

Figure 8.

F19 Appearance Agent -----> Bob

```
NOTIFY sip:bob@ua1.example.com SIP/2.0
From: <sip:HelpDesk@example.com>;tag=497585728578386
To: <sip:bob@example.com>;tag=633618CF-B9C2EDA4
Call-ID: a7d559db-d6d7dcad-311c9e3a
CSeq: 7 NOTIFY
Via: SIP/2.0/UDP appearanceagent.example.com
    ;branch=z9hG4bK1711759878512309
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Event: dialog;shared
Subscription-State: active;expires=1500
Contact: <sip:appearanceagent.example.com>
Content-Length: ...
```

```
<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
    xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
    version="10"
    state="partial"
    entity="sip:HelpDesk@example.com">
  <dialog id="3xdsd4f9c83"
    call-id="b3cbd0-ad2c5775e-5df9f8d5"
    local-tag="34322kdf234f"
    remote-tag="3153DE7C-928203B"
    direction="initiator">
    <sa:exclusive>true</sa:exclusive>
    <sa:appearance>1</sa:appearance>
    <state>confirmed</state>
    <local>
      <target uri="sip:bob@ua2.example.com">
      </target>
    </local>
    <remote>
      <identity>sip:HelpDesk@example.com</identity>
      <target uri="sip:alice@ua1.example.com" />
    </remote>
```

```

</dialog>

<dialog id="4839589"
  call-id="b3cbd0-ad2c5775e-5df9f8d5"
  local-tag="3153DE7C-928203B"
  remote-tag="34322kdfr234f"
  direction="responder">
  <sa:exclusive>true</sa:exclusive>
  <sa:appearance>1</sa:appearance>
  <state>confirmed</state>
  <local>
    <target uri="sip:alice@ua1.example.com" />
  </local>
  <remote>
    <identity>sip:HelpDesk@example.com</identity>
    <target uri="sip:bob@ua2.example.com" />
  </remote>
</dialog>

</dialog-info>

```

11.9. Consultation Hold with Appearances

In this scenario, Bob has a call with Carol. Bob makes a consultation call to Alice by putting Carol on hold and calling Alice. Bob's UA chooses not to have an appearance number for the call to Alice since it is treating it as part of the call to Carol. He indicates this in the PUBLISH F32 which contains the 'shared' Event header field parameter but no <appearance> element. The PUBLISH is sent before the INVITE to Alice to ensure no appearance number is assigned by the Appearance Agent. Finally, Bob hangs up with Alice and resumes the call with Carol. Dialog notifications of the consultation call are not shown, as they are not used.

Note that if Carol hangs up while Bob is consulting with Alice, Bob can decide if he wants to reuse the appearance number used with Carol for the call with Alice. If not, Bob publishes the termination of the dialog with Carol and the Appearance Agent will re-allocate the appearance. If he wants to keep the appearance, Bob will publish the termination of the dialog with Carol and also publish the appearance with the dialog with Alice. This will result in Bob keeping the appearance number until he reports the dialog with Alice terminated.

Note that the call flow would be similar if Bob called a music on hold server instead of Alice to implement a music on hold service as described in [I-D.worley-service-example].

Carol	Proxy	Alice	Appearance Agent	Bob
-------	-------	-------	------------------	-----

```

<===== Both way RTP established =====>
<- INVITE F23<
>F24 200 OK ->
<----- ACK F27<
<- INVITE F22<
>F25 200 OK ----->
<----- ACK F26<
<- NOTIFY F28<
>F29 200 OK ->
>F30 NOTIFY ----->
<----- 200 OK F31<
Bob makes a consultation call to Alice
<----- PUBLISH F32<
>F33 200 OK ----->
<- INVITE F34<
>F35 INVITE -->
<-- 200 OK F36<
>F37 200 OK ----->
<----- ACK F38<
>F39 ACK ----->
<===== RTP established =====>
Bob hangs up with Alice
<----- BYE F40<
>F41 BYE ----->
<-- 200 OK F42<

```

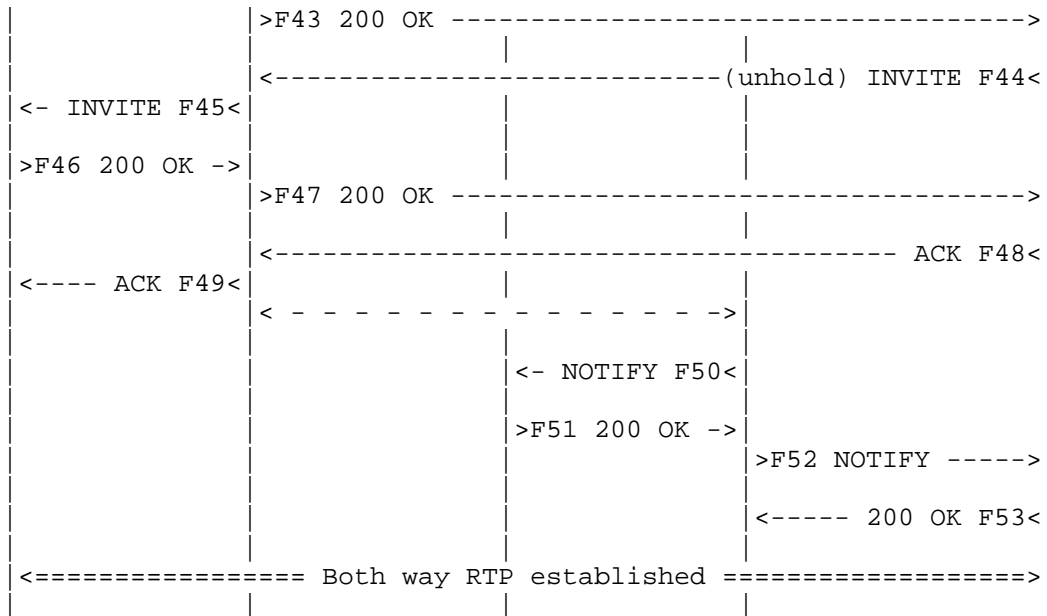


Figure 9.

F32 Bob -----> Appearance Agent

```

PUBLISH sip:HelpDesk@example.com SIP/2.0
Via: SIP/2.0/UDP ua2.example.com;branch=z9hG4bKa5d6cf61F5FBC05A
From: <sip:bob@example.com>;tag=44150CC6-A7B7919D
To: <sip:HelpDesk@example.com>;tag=428765950880801
CSeq: 11 PUBLISH
Call-ID: 44fwF144-F12893K38424
Contact: <sip:bob@ua2.example.com>
Event: dialog;shared
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Content-Length: ...
  
```

```

<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
  version="10"
  state="full"
  entity="sip:HelpDesk@example.com">
  <dialog id="id3d4f9c83"
    call-id="b3cbd0-ad2c5775e-5df9f8d5"
    local-tag="3153DE7C-928203B"
    direction="initiator">
  
```

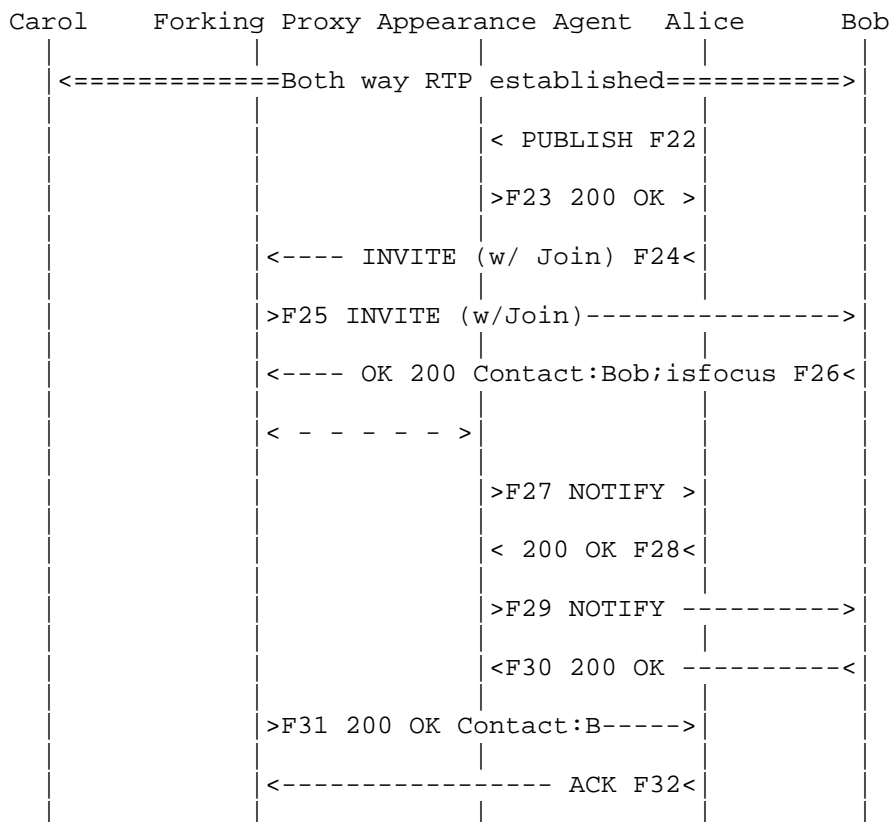
```

    <sa:exclusive>true</sa:exclusive>
    <state>trying</state>
    <local>
      <target uri="sip:bob@ua2.example.com">
      </target>
    </local>
    <remote>
      <identity>sip:HelpDesk@example.com</identity>
      <target uri="sip:alice@ua1.example.com" />
    </remote>
  </dialog>
</dialog-info>

```

11.10. Joining or Bridging an Appearance

In this call flow, a call answered by Bob is joined by Alice or "bridged". The Join header field is used by Alice to request this bridging. If Bob did not support media mixing, Bob could obtain conferencing resources as described in [RFC4579].



	>ACK F33----->	
	<-----INVITE Contact:Bob;isfocus F34<	
<-INVITE F35		
>F26 200 -->		
	>F37 200 OK ----->	
	<-----ACK F38<	
<--- ACK F39		
		<==RTP==>
<=====	Both way RTP established=====	
	< - - - - - >	
		>F40 NOTIFY >
		< 200 OK F41<
		>F42 NOTIFY ----->
		<F43 200 OK -----<

Figure 10.

F22 Alice ----> Appearance Agent

```

PUBLISH sip:HelpDesk@example.com SIP/2.0
Via: SIP/2.0/UDP ua2.example.com;branch=z9hG4bKa5d6cf61F5FBC05A
From: <sip:alice@example.com>;tag=44150CC6-A7B7919D
To: <sip:HelpDesk@example.com>;tag=428765950880801
CSeq: 11 PUBLISH
Call-ID: 87837Fkw87asfds
Contact: <sip:alice@ua2.example.com>
Event: dialog;shared
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Content-Length: ...

```

```

<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
  version="10"
  state="full"
  entity="sip:HelpDesk@example.com:5060">
  <dialog id="id3d4f9c83"
    call-id="dc95da63-60dblabd-d5a74b48"

```



```

    local-tag="605AD957-1F6305C2" >
      <sa:appearance>1</sa:appearance>
      <sa:exclusive>false</sa:exclusive>
      <sa:joined-dialog
        call-id="14-1541707345"
        from-tag="44BAD75D-E3128D42"
        to-tag="d3b06488-1dd1-11b2-88c5-b03162323164+d3e48f4c" />
      <state>trying</state>
      <local>
        <target uri="sip:alice@ual.example.com">
          </target>
        </local>
      <remote>
        <target uri="sip:bob@example.com" />
      </remote>
    </dialog>
  </dialog-info>

```

F24 Alice ----> Proxy

```

INVITE sip:bob@ua.example.com SIP/2.0
Via: SIP/2.0/UDP ual.example.com;branch=z9hG4bKcc9d727c2C29BE31
From: <sip:HelpDesk@example.com>;tag=605AD957-1F6305C2
To: <sip:bob@ua.example.com>
CSeq: 2 INVITE
Call-ID: dc95da63-60db1abd-d5a74b48
Contact: <sip:alice@ual.example.com>
<all-one-line>
Join: 14-1541707345;to-tag=d3b06488-1dd1-11b2-88c5
-b03162323164+d3e48f4c;from-tag=44BAD75D-E3128D42
</all-one-line>
Max-Forwards: 70
Content-Type: application/sdp
Content-Length: 223

```

```

v=0
o=- 1103061265 1103061265 IN IP4 ual.example.com
s=IP SIP UA
c=IN IP4 ual.example.com
t=0 0
a=sendrecv
m=audio 2236 RTP/AVP 0 8 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000

```

11.11. Appearance Allocation - Loss of Appearance

Bob reserves an appearance with a PUBLISH, sends an INVITE to Carol, then becomes unreachable. When he fails to refresh his publication to the appearance agent, the Appearance Agent declares the dialog terminated and frees up the appearance using NOTIFYs F14 and F16. After retransmitting the NOTIFY to Bob (in not shown messages F17, F18, etc.), the subscription is terminated.

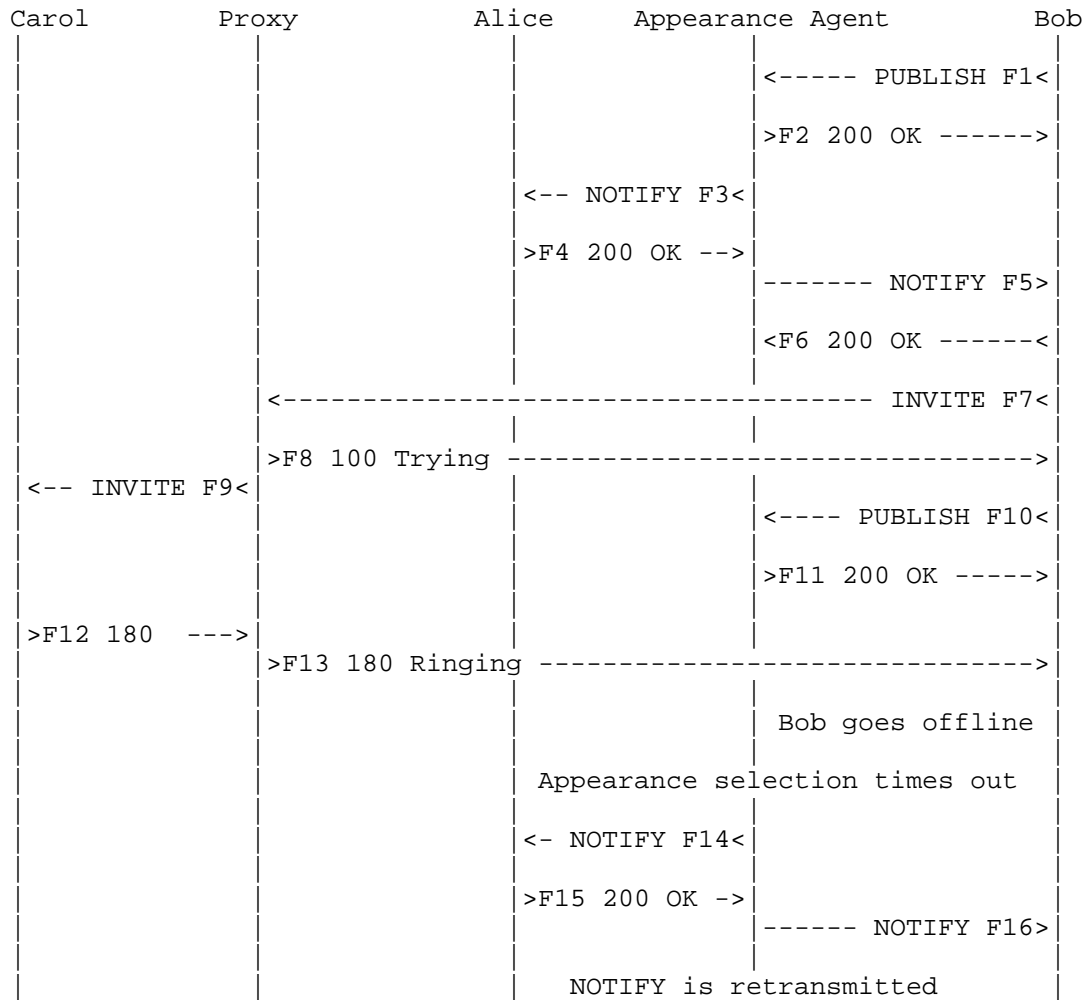


Figure 11.

11.12. Appearance Seizure Contention Race Condition

Bob and Alice both try to reserve appearance 2 by publishing at the same time. The Appearance Agent allocates the appearance to Bob by sending a 200 OK and denies it to Alice by sending a 400 (Bad Request) response. After the NOTIFY F5, Alice learns that Bob is using appearance 2. Alice then attempts to reserve appearance 3 by publishing, which is then accepted.

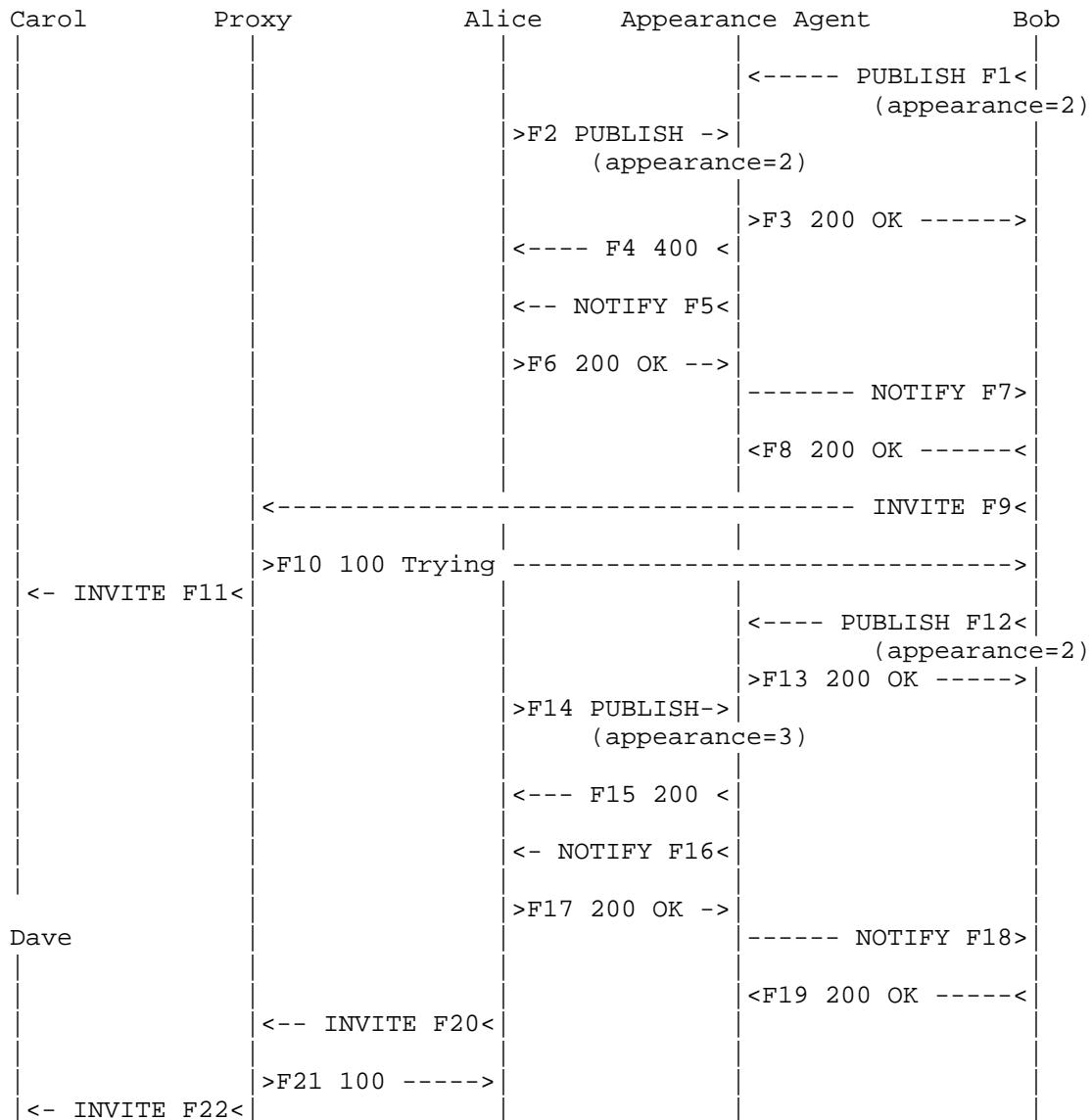
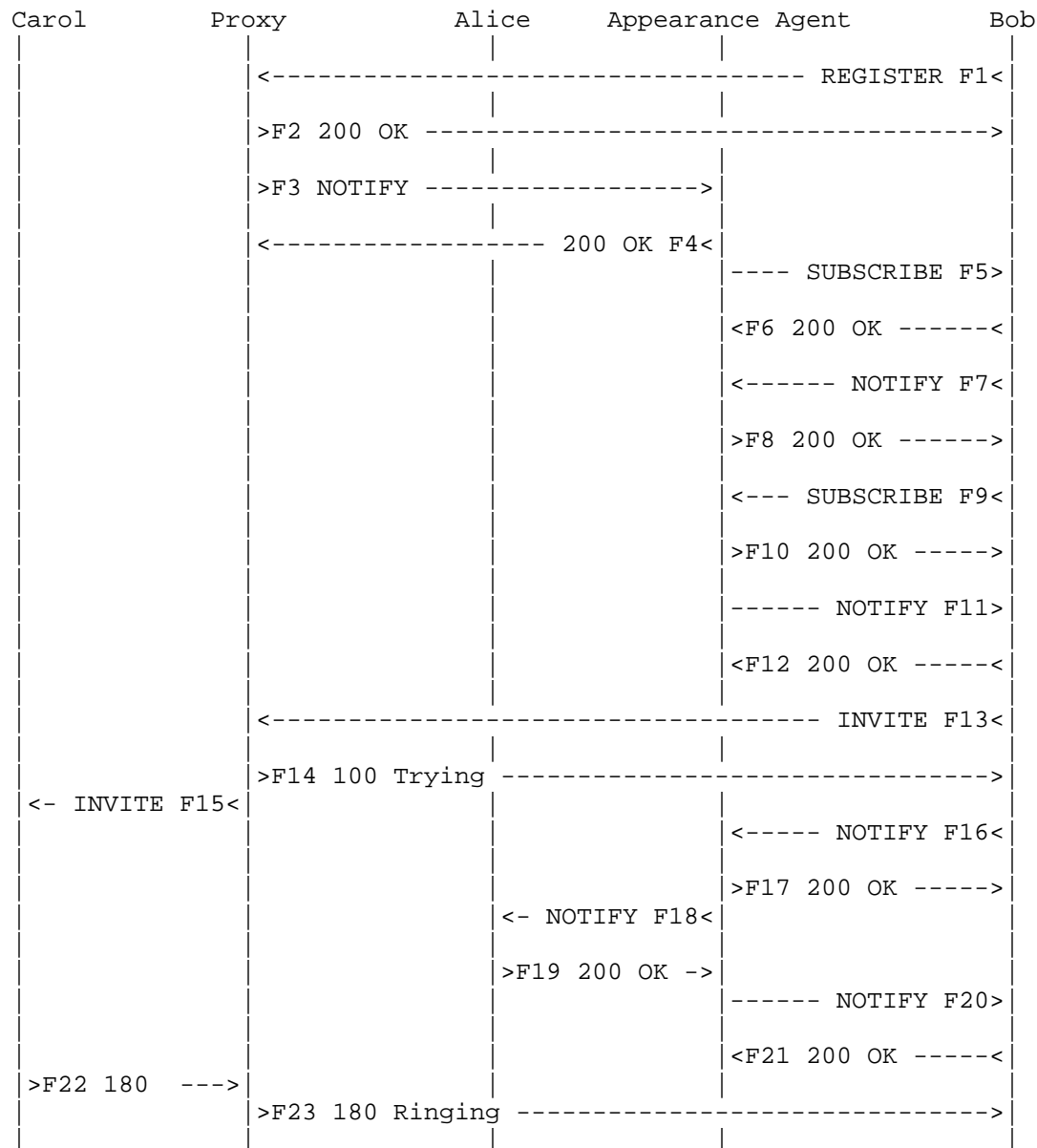


Figure 12.

11.13. Appearance Agent Subscription to UAs

In this scenario, the Appearance Agent does not have any way of knowing Bob's dialog state information, except through Bob. This could be because the Appearance Agent is not part of a B2BUA, or

perhaps Bob is remotely registering. When Bob registers, the Appearance Agent receives a registration event package notification from the registrar. The Appearance Agent then SUBSCRIBES to Bob's dialog event state using Event:dialog in the SUBSCRIBE. Whenever Bob's dialog state changes, Bob's UA sends a NOTIFY to the Appearance Agent which then notifies the other other UAs in the group.



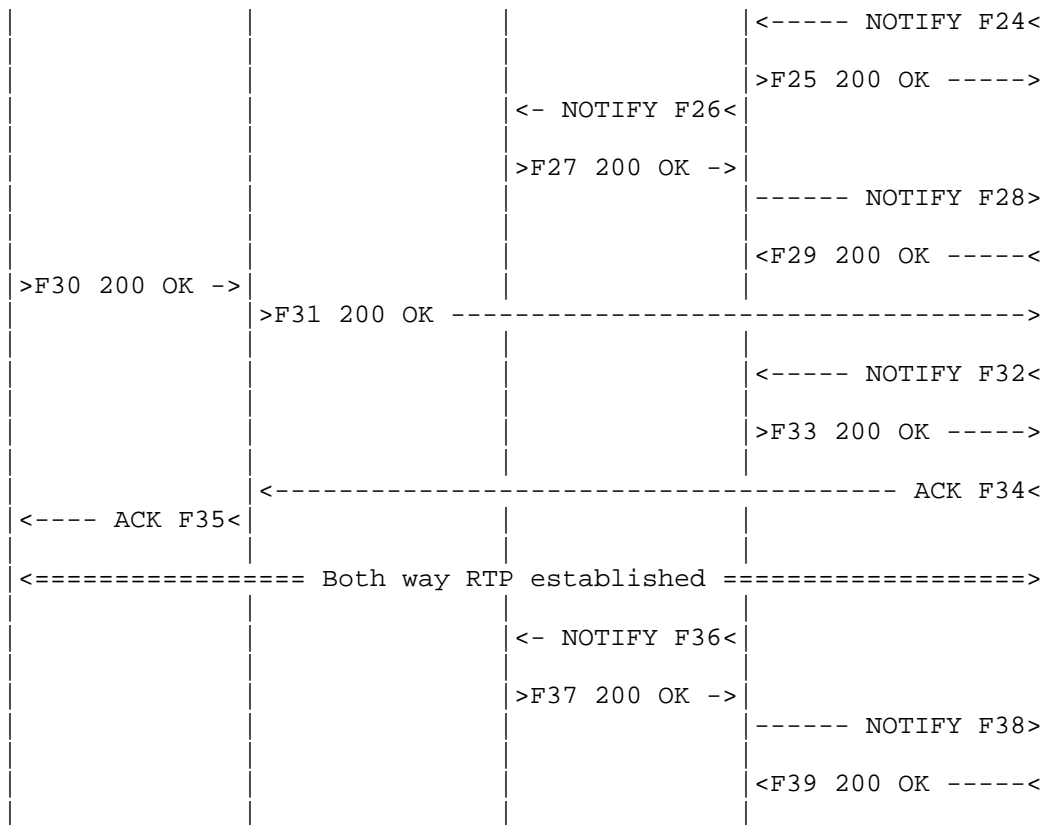
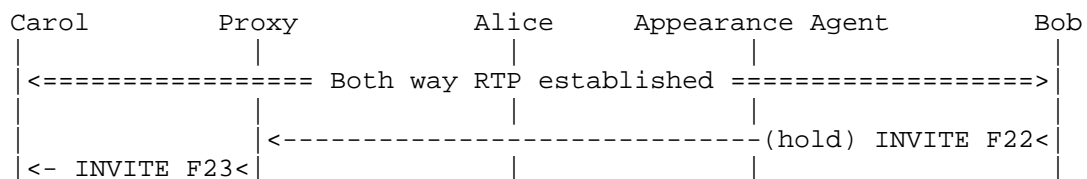


Figure 13.

11.14. Appearance Pickup Race Condition Failure

In this scenario, Bob has an established dialog with Carol created using the call flows of Figure 1 or Figure 2. Bob then places Carol on hold. Alice receives a notification of this and renders this on Alice's UI. Alice attempts to pick up the call but Carol hangs up before the pickup can complete. Alice cancels the pickup attempt with the PUBLISH F48. Note that the call flow for a failed Join would be almost identical.



>F24 200 OK ->	>F25 200 OK ----->		
<----- ACK F27<	<----- ACK F26<		
	< - - - - ->		
	<- NOTIFY F28<		
	>F29 200 OK ->	>F30 NOTIFY ----->	
		<----- 200 OK F31<	
	Alice decides to pick up the call		
	>F32 PUBLISH->		
	<- 200 OK F33<		
	<- NOTIFY F34<		
	>F35 200 OK ->	>F36 NOTIFY ----->	
		<----- 200 OK F37<	
>F38 BYE ----->	>F39 BYE ----->		
	<----- OK 200 F40<		
<- 200 OK F41<	<-- INVITE F42<		
<- INVITE F43<	(w/ Replaces)		
(w/ Replaces)			
>F44 481 ----->	>F45 481 ----->		
<----- ACK F46<	<----- ACK F47<		
	>F48 PUBLISH->		
	<- 200 OK F49<		
	<- NOTIFY F50<		
	>F51 200 OK ->		

```

|                                     |>F52 NOTIFY ----->|
|                                     |<----- 200 OK F53<|

```

Figure 14.

F48 Alice -----> Appearance Agent

```

PUBLISH sip:HelpDesk@example.com SIP/2.0
Via: SIP/2.0/UDP ua2.example.com;branch=z9hG4bKa5d6cf61F5FBC05A
From: <sip:alice@example.com>;tag=44150CC6-A7B7919D
To: <sip:HelpDesk@example.com>;tag=428765950880801
CSeq: 11 PUBLISH
Call-ID: 87837Fkw87asfds
Contact: <sip:alice@ua2.example.com>
Event: dialog;shared
Max-Forwards: 70
Content-Type: application/dialog-info+xml
Content-Length: ...

<?xml version="1.0"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
              xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
              version="10"
              state="full"
              entity="sip:HelpDesk@example.com">
  <dialog id="id3d4f9c83"
    call-id="dc95da63-60db1abd-d5a74b48"
    local-tag="605AD957-1F6305C2" >
    <sa:appearance>1</sa:appearance>
    <sa:exclusive>false</sa:exclusive>
    <sa:replaced-dialog
      call-id="14-1541707345"
      from-tag="44BAD75D-E3128D42"
      to-tag="d3b06488-1dd1-11b2-88c5-b03162323164+d3e48f4c" />
    <state>terminated</state>
    <local>
      <target uri="sip:alice@ua1.example.com">
        </target>
      </local>
    <remote>
      <target uri="sip:carol@ua3.example.com" />
    </remote>
  </dialog>
</dialog-info>

```


11.15. Appearance Seizure Incoming/Outgoing Contention Race Condition

Alice tries to seize appearance 2 at the same time appearance 2 is allocated to an incoming call. The Appearance Agent resolves the conflict by sending a 400 (Bad Request) to Alice. After the NOTIFY F6, Alice learns that the incoming call is using appearance 2. Alice republishes for appearance 3, which is accepted. Note that this example shows the INVITE being received before the NOTIFY from the Appearance Agent.

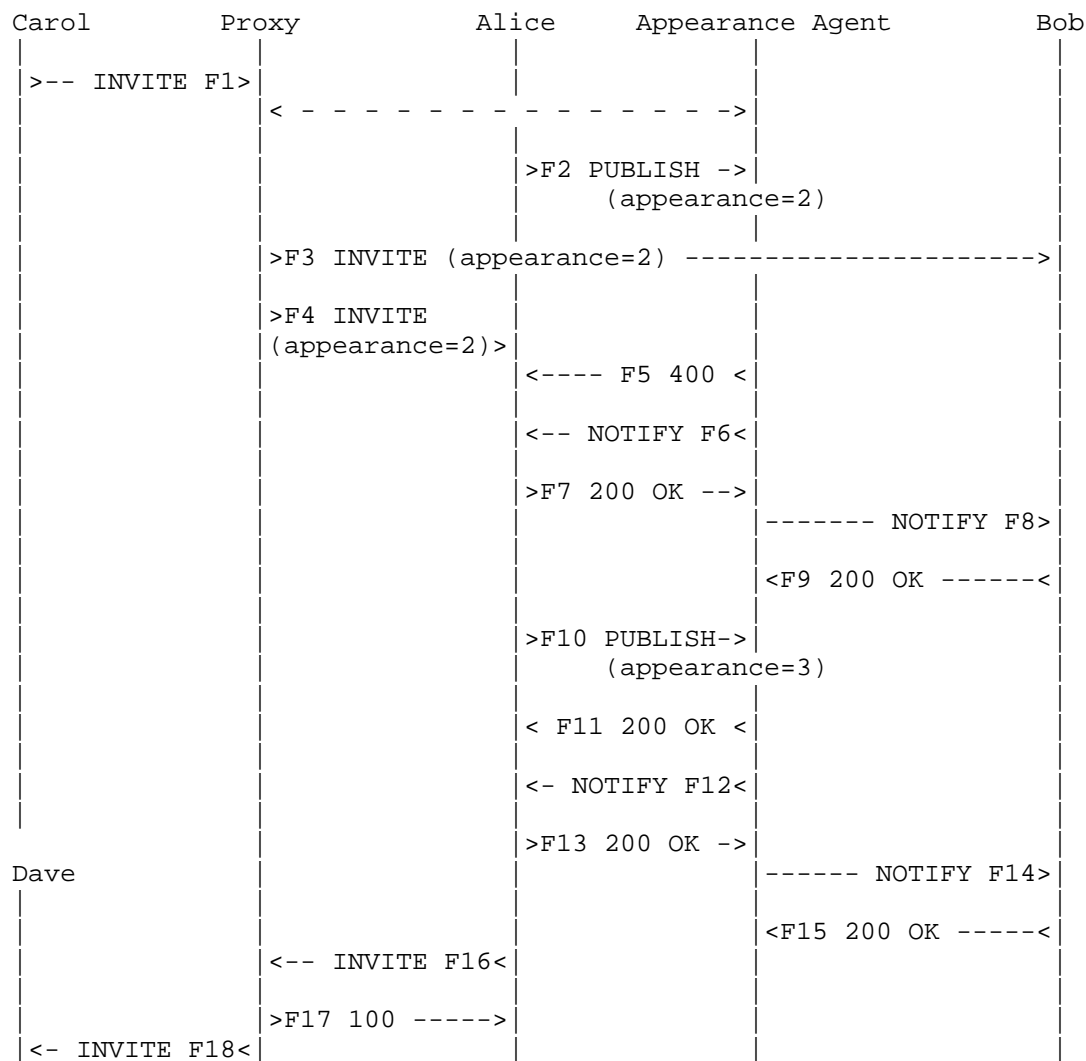


Figure 15.

12. Security Considerations

Since multiple line appearance features are implemented using semantics provided by SIP [RFC3261], the SIP Event Package for Dialog State [RFC4235], and the SIP Event Framework [I-D.ietf-sipcore-rfc3265bis] and [RFC3903], security considerations in these documents apply to this document as well.

To provide confidentiality, NOTIFY or PUBLISH message bodies that provide the dialog state information and the dialog identifiers MAY be encrypted end-to-end using the standard mechanisms such as S/MIME described in [RFC3261]. Alternatively, sending the NOTIFY and PUBLISH requests over TLS also provides confidentiality, although on a hop-by-hop basis. All SUBSCRIBES and PUBLISHES between the UAs and the Appearance Agent MUST be authenticated. Without proper authentication and confidentiality, a third party could learn information about dialogs associated with a AOR and could try to use this information to hijack or manipulate those dialogs using SIP call control primitives.

This feature relies on standard SIP call control primitives such as Replaces and Join. Proper access controls on their use MUST be used so that only members of the appearance group can use these mechanisms. All INVITES with Replaces or Join header fields MUST only be accepted if the peer requesting dialog replacement or joining has been properly authenticated using a standard SIP mechanism (such as Digest or S/MIME), and authorized to request a replacement. Otherwise, a third party could disrupt or hijack existing dialogs in the appearance group.

For an emergency call, a UA MUST NOT wait for a confirmed seizure of an appearance before sending an INVITE. Waiting for confirmation could inadvertently delay or block the emergency call, which by its nature needs to be placed as expeditiously as possible. Instead, a emergency call MUST proceed regardless of the status of the PUBLISH transaction.

13. IANA Considerations

This section registers the SIP Event header field parameter 'shared', the SIP Alert-Info header field parameter 'appearance' and the XML namespace extensions to the SIP Dialog Package.

13.1. SIP Event Header Field Parameter: shared

This document defines the 'shared' header field parameter to the Event header field in the "SIP Header Field Parameters and Parameter

Values" registry defined by [RFC3968].

Header Field	Parameter Name	Predefined Values	Reference
-----	-----	-----	-----
Event	shared	No	[RFC-to-be]

13.2. SIP Alert-Info Header Field Parameter: appearance

This document defines the 'appearance' parameter to the Alert-Info header in the "SIP Header Field Parameters and Parameter Values" registry defined by [RFC3968].

Header Field	Parameter Name	Predefined Values	Reference
-----	-----	-----	-----
Alert-Info	appearance	No	[RFC-to-be]

13.3. URN Sub-Namespace Registration: sa-dialog-info

This section registers a new XML namespace per the procedures in [RFC3688].

URI: urn:ietf:params:xml:ns:sa-dialog-info.

Registrant Contact: IETF BLISS working group, <bliss@ietf.org>, Alan Johnston <alan.b.johnston@gmail.com>

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
    "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1"/>
  <title>Shared Appearance Dialog Information Namespace</title>
</head>
<body>
  <h1>Namespace for Shared Appearance Dialog Information</h1>
  <h2>urn:ietf:params:xml:ns:sa-dialog-info</h2>
  <p>See <a href="ftp://ftp.rfc-editor.org/in-notes/rfcXXXX.txt">
    RFCXXXX</a>.</p>
</body>
</html>
END
```

13.4. XML Schema Registration

This section registers an XML schema per the procedures in [RFC3688].

URI: urn:ietf:params:xml:schema:sa-dialog-info.

Registrant Contact: IETF BLISS working group, <bliss@ietf.org>, Alan Johnston <alan.b.johnston@gmail.com>

The XML for this schema can be found in Section 6.

14. Acknowledgements

The following individuals were part of the shared appearance Design team and have provided input and text to the document (in alphabetical order):

Martin Dolly, Andrew Hutton, Raj Jain, Fernando Lombardo, Derek MacDonald, Bill Mitchell, Michael Procter, Theo Zourzouvillys.

Thanks to Chris Boulton for helping with the XML schema.

Much of the material has been drawn from previous work by Mohsen Soroushnejad, Venkatesh Venkataramanan, Paul Pepper and Anil Kumar, who in turn received assistance from:

Kent Fritz, John Weald, and Sunil Veluvali of Sylanro Systems, Steve Towlson, and Michael Procter of Citel Technologies, Rob Harder and Hong Chen of Polycom Inc, John Elwell, J D Smith of Siemens Communications, Dale R. Worley of Pingtel, Graeme Dollar of Yahoo Inc.

Also thanks to Geoff Devine, Paul Kyzivat, Jerry Yin, John Elwell, Dan York, Spenser Dawkins, Martin Dolly, and Brett Tate for their comments.

Thanks to Carolyn Beeton, Francois Audet, Andy Hutton, Tim Ross, Raji Chinnappa, and Harsh Mendiratta for their detailed review of the document.

15. References

15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC3515] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

- [I-D.ietf-sipcore-rfc3265bis]
Roach, A., "SIP-Specific Event Notification",
draft-ietf-sipcore-rfc3265bis-09 (work in progress),
April 2012.
- [RFC3903] Niemi, A., "Session Initiation Protocol (SIP) Extension
for Event State Publication", RFC 3903, October 2004.
- [RFC3891] Mahy, R., Biggs, B., and R. Dean, "The Session Initiation
Protocol (SIP) "Replaces" Header", RFC 3891,
September 2004.
- [RFC4235] Rosenberg, J., Schulzrinne, H., and R. Mahy, "An INVITE-
Initiated Dialog Event Package for the Session Initiation
Protocol (SIP)", RFC 4235, November 2005.
- [RFC3911] Mahy, R. and D. Petrie, "The Session Initiation Protocol
(SIP) "Join" Header", RFC 3911, October 2004.
- [RFC3840] Rosenberg, J., Schulzrinne, H., and P. Kyzivat,
"Indicating User Agent Capabilities in the Session
Initiation Protocol (SIP)", RFC 3840, August 2004.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
January 2004.
- [I-D.ietf-salud-alert-info-urns]
Liess, L., Jesske, R., Johnston, A., Worley, D., and P.
Kyzivat, "Alert-Info URNs for the Session Initiation
Protocol (SIP)", draft-ietf-salud-alert-info-urns-07 (work
in progress), October 2012.

15.2. Informative References

- [RFC5359] Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and
K. Summers, "Session Initiation Protocol Service
Examples", BCP 144, RFC 5359, October 2008.
- [RFC4579] Johnston, A. and O. Levin, "Session Initiation Protocol
(SIP) Call Control - Conferencing for User Agents",
BCP 119, RFC 4579, August 2006.
- [RFC3680] Rosenberg, J., "A Session Initiation Protocol (SIP) Event
Package for Registrations", RFC 3680, March 2004.
- [I-D.worley-service-example]
Worley, D., "Session Initiation Protocol Service Example
-- Music on Hold", draft-worley-service-example-10 (work

in progress), August 2012.

Authors' Addresses

Alan Johnston (editor)
Avaya
St. Louis, MO 63124

Email: alan.b.johnston@gmail.com

Mohsen Soroushnejad
Sylantro Systems Corp

Email: msoroush@gmail.com

Venkatesh Venkataramanan
Sylantro Systems Corp

Email: vvenkatar@gmail.com

