Call Pickup Examples in SIP
draft-worley-sipping-pickup-02

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

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

This Internet-Draft will expire on December 23, 2006.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This document walks through various examples and call flows for implementing "call pickup" operations in SIP telephony. It focuses on distributing as much processing as possible in the user agents in accordance with the philosophy of end-point controlled call-control (EPCC).

Various difficulties in implementing call pickup are discussed, and further suggestions and discussion are solicited.
Table of Contents

1. Introduction ............................................. 3
2. Related work ............................................ 4
3. Basic call flow ......................................... 5
4. Detailed call flow ....................................... 7
5. Call pickup with proxy .................................. 10
6. Call pickup with proxy and state agent ................... 13
7. Call pickup with a pickup agent ......................... 15
8. With improved pickup agent .............................. 17
9. With proxy and improved pickup agent ................... 19
10. Group call pickup ....................................... 22
    10.1. Picking up any call to any member of a group of
         individuals ........................................... 22
    10.2. Picking up a call routed to several individuals ..... 22
    10.3. Picking up a call that is not routed to a specific
         user agent ............................................ 22
11. Call park ................................................. 23
12. Alternative retrieval mechanism for call park ............ 24
13. Security considerations .................................. 27
14. Revision history ......................................... 28
    14.1. Changes from draft-worley-sipping-pickup-01 to
         draft-worley-siping-pickup-02 ...................... 28
15. Informative References .................................. 28
Author’s Address ............................................. 29
Intellectual Property and Copyright Statements ............... 30
1. Introduction

There are several different schemes for implementing call pickup. The basic method is the one specified in the Sylantro "SIP-B" specification, which despite its proprietary air, uses standard SIP features in an end-point call control (EPCC) style [1]. All other methods are variations on the same theme, usually by using an agent process (in a proxy or communications server) to provide a feature that the user agents are lacking. Almost all the software development effort to support call pickup is in implementing these agent processes, so we can easily see the trade-off between development effort and which features must be provided by the user agents.

Like call transfer, effecting call pickup requires some support from the caller’s end. These caller-end features will, therefore, soon come to be considered necessary for any "quality" SIP implementation.

This document shows message flows for executing call pickup using various configurations of SIP agents, including a proxy that dispatches requests for AORs to their registered UAs, "call state agents" that generate dialog events on behalf of UAs that cannot generate them, and "pickup agents" that execute the call pickup sequence on behalf of UAs that cannot do so themselves. It also discusses various complications that can arise.

This document is not intended to advance toward standardization, but rather to serve as examples for implementors, and to start discussion of problems in SIP that may arise when implementing call pickup.
2. Related work

Several other documents have explored SIP call pickup:

1. draft-ietf-sipping-service-examples [2] discusses a large number of EPCC implementations of call-control features, including implementations of call pickup operations.

2. draft-procter-sipping-call-park-extension [3] discusses an alternative mechanism to park calls with designated, short parking orbit numbers, and how to retrieve such calls.
3. Basic call flow

The SIP-B pickup mechanism is implemented in the user agents. Thus, it requires a more sophisticated user agent to execute the pickup. It also depends on the callee phone producing dialog events, or there being a state agent doing so on its behalf.

The basic SIP-B pickup sequences is as follows. (Only principal messages are shown.) Suppose the incoming call is to the callee phone, extension 123, and the phone executing the pickup is extension 456:

```
1 INVITE
----------------------> 2 Call Pickup on 123
                           
3 SUBSCRIBE
<------------------------

4 NOTIFY
----------->

5 INVITE/Replaces
<------------------------

6 CANCEL
----------->

7 200 OK
----------->
```

1. The caller (AOR Caller@example.com) sends an INVITE with URI 123@example.com to phone 123.

2. The user of phone 456 activates the call pickup feature for extension 123.
3. Phone 456 sends a SUBSCRIBE with URI 123@example.com to phone 123, requesting "Event: dialog" and "Expires: 0".

4. Phone 123 sends one NOTIFY to phone 456 giving the status of its current dialogs for AOR 123@example.com, which includes the early dialog of INVITE 1 from Caller, and gives the "remote identity" and "remote target" of the dialog (which are the From: and Contact: of INVITE 1), one of which is "Caller@example.com".

5. Phone 456 sends INVITE 5 to Caller@example.com. It has a Replaces: header specifying the dialog parameters sent in the NOTIFY. The Replaces: header contains the "early-only" option, so that the pickup operation fails if extension 123 answers the call.

6. As a consequence of executing the INVITE/Replaces, the caller sends a CANCEL of its INVITE 1 to phone 123.

7. The caller sends a 200 response to the INVITE 5 from phone 456.
   At this point, Caller is talking to phone 456.
4. Detailed call flow

A more complete packet diagram of a SIP-B pickup sequence is as follows. (100 responses are present only if the transport is UDP, and are omitted on this diagram.) Messages in the same transaction are labeled with the same number but successive letters, e.g., 7a, 7b, 7c.
<table>
<thead>
<tr>
<th>Caller</th>
<th>Target 123</th>
<th>Executing 456</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a INVITE</td>
<td>---------------------</td>
<td></td>
</tr>
<tr>
<td>1b 180 Ringing</td>
<td>&lt;----------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 Call Pickup on 123</td>
</tr>
<tr>
<td>3a SUBSCRIBE</td>
<td>&lt;----------------------</td>
<td></td>
</tr>
<tr>
<td>3b 200 OK</td>
<td>----------------------&gt;</td>
<td></td>
</tr>
<tr>
<td>4a NOTIFY</td>
<td>----------------------&gt;</td>
<td></td>
</tr>
<tr>
<td>4b 200 OK</td>
<td>&lt;----------------------</td>
<td></td>
</tr>
<tr>
<td>5a INVITE/Replaces</td>
<td>&lt;----------------------</td>
<td></td>
</tr>
<tr>
<td>5b 200 OK</td>
<td>----------------------&gt;</td>
<td></td>
</tr>
<tr>
<td>5c ACK</td>
<td>&lt;----------------------</td>
<td></td>
</tr>
<tr>
<td>1c CANCEL</td>
<td>----------------------&gt;</td>
<td></td>
</tr>
<tr>
<td>1d 200 OK</td>
<td>&lt;----------------------</td>
<td></td>
</tr>
<tr>
<td>1d 487 Request Term.</td>
<td>&lt;----------------------</td>
<td></td>
</tr>
<tr>
<td>1e ACK</td>
<td>----------------------&gt;</td>
<td></td>
</tr>
</tbody>
</table>
1. The caller (AOR Caller@example.com) sends an INVITE 1a with URI 123@example.com to phone 123.

2. The user of phone 456 activates the call pickup feature for extension 123. This may be activated by a special button on the phone, or through a dialing prefix which is intercepted by the phone.

3. Phone 456 sends a SUBSCRIBE 3a with URI 123@example.com to extension 123, requesting "Event: dialog" and "Expires: 0". This SUBSCRIBE requests one NOTIFY containing the status of all dialogs involving AOR 123@example.com.

   This requires that the other phones in the installation can produce dialog event notifications. We discuss below using a state agent to support phones that cannot produce dialog event notifications.

   Since dialog event notifications can contain sensitive information, the ability to SUBSCRIBE to them should be controlled by authentication. This will be discussed below.

4. Phone 123 sends one NOTIFY 4a to phone 456 giving the status of its current dialogs for AOR 123@example.com, which includes the early dialog of INVITE 1a from Caller, and gives the "remote identity" and/or "remote target" of the dialog (which are the From: and Contact: of INVITE 1a), which is "Caller@example.com".

5. Phone 456 sends to Caller@example.com an INVITE 5a. It has a Replaces: header specifying the dialog parameters sent in NOTIFY 4a. Its URI is the remote target address, and its has a Replaces: header specifying the dialog parameters sent in the NOTIFY.

   If the NOTIFY 4a does not include a "remote target" URI (which should always identify the caller UA), the executing phone will have to fall back to the "remote identity" URI (in hopes that it routes to the caller UA). The "remote target" URI should be the Contact URI from INVITE 1a. In order for INVITE 5a to succeed, that Contact needs to be globally routable.

1c. The caller sends a CANCEL 1c of its INVITE 1a to phone 123.

1d. The caller sends a 200 response to the INVITE 5a from phone 456.

   At this point, Caller is talking to phone 456.
5. Call pickup with proxy

Now let us consider call pickup when the organization has a proxy for routing calls.

<table>
<thead>
<tr>
<th>Caller</th>
<th>Proxy</th>
<th>Target</th>
<th>Executing</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORWARDED</td>
<td></td>
<td>123</td>
<td>456</td>
</tr>
</tbody>
</table>

1a INVITE

1b INVITE

1c 180 Ring.

1d 180 Ringing

2 Call Pickup on 123

3a SUBSCRIBE

3b SUBSCRIBE

3c OK

3d 200 OK

4a NOTIFY

4b 200 OK

5a INVITE/Replaces

5b 200 OK

5c ACK

1e CANCEL

1f CANCEL
The major differences from the previous example are:

1. INVITE 1a and later messages in its transaction go through the proxy, thus allowing the proxy to convert 123@example.com to a series of contact points.

3. SUBSCRIBE 3a also goes through the proxy, so it is also sent to a series of contact points. The proxy should fork the SUBSCRIBE in the same way as it would fork an INVITE, except that it should fork all the contact points in parallel, even if they would fork serially for INVITEs, so phone 456 can find its target call quickly.

As SUBSCRIBE 3a is forked, its request URI is changed from the AOR 123@example.com to various contact URIs, usually the same user name with host parts that are the IP addresses of the phone user agents. This will match the request URIs of the INVITEs 1b, so information about those dialogs will be reported in NOTIFYs.

4. Because SUBSCRIBE 3a can be forked, phone 456 should be prepared to receive multiple NOTIFYs from various phones for AOR 123, and be able to select the correct dialog if it discovers more than one.

Phone 456 should beware that it may see multiple instances of a single forked INVITE (which have the same Call-Id), and not present them to the user as alternatives.

Phone 456 should beware that one AOR might be forwarded to a URI with a different user name. E.g., "123@example.com" might fork first to an user agent on 123’s desk ("123@10.1.200.12") and then to a back-up responder ("789@example.com"). However, a UA that receives SUBSCRIBE 3a will see the same request URI as it saw when
it received INVITE 1a, and thus will include any dialog it creates in its subsequent NOTIFY 4a. But it may include dialogs that were originally sent to other AORs as well. Unfortunately, there is no clean way to distinguish the dialogs that were originally for 123, as the best approximation to that information is the "local identity" (i.e., "To:" URI), but that may have originally been an address that forwards to 123. In the end, we should avoid forwarding two AORs to the same request-URI, so as to avoid this ambiguity.

5. The INVITE/Replaces 5a may be forwarded through the proxy as well. Doing so makes no essential difference to the operation.
6. Call pickup with proxy and state agent

This case is when the phones themselves cannot generate dialog event notices, so that work is delegated to a state agent which receives information from the site’s proxy.

<table>
<thead>
<tr>
<th>Caller</th>
<th>Proxy</th>
<th>State Agent</th>
<th>Target</th>
<th>Executing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>123</td>
<td>456</td>
<td></td>
</tr>
</tbody>
</table>

```
1a INVITE
------------->
1b INVITE
------------->
1c 180 Ring.
<-----------
1d 180 Ringing
-------------

State Info.
++++++++++++++>

2 Call Pickup on 123

3a SUBSCRIBE
<------------------
3b SUBSCRIBE
------------->
3c OK
<----------
3d 200 OK

4a NOTIFY
----------->
4b 200 OK
<---------

[Remainder of operation is as before.]
```

The major differences in processing are:
The proxy monitors the progress of dialogs and reports state information via a possibly non-SIP channel to the state agent. Since the state agent needs to report only early dialog information, one possible channel is to do a non-blocking fork to the state agent of every call to 123. (This trick was invented by Scott Lawrence.) However, if the state agent will be used to support applications that require knowledge of committed dialogs (e.g., attendant consoles), this technique alone will not be sufficient.

3. The proxy routes SUBSCRIBE 3a not to phone 123 but rather to the state agent. The state agent uses the request URI or the From: URI to determine the user on whose behalf it should generate events.

Since a SIP call can be maintained indefinitely without any signaling, it is possible for the state agent to miss the termination of a call and believe that the call is continuing forever. There needs to be a way to clear such state. Early dialog state can be cleared due to the global time limit on INVITE transactions, but committed dialogs have no intrinsic time limit. One way to reset the state agent would be to define a dialing prefix, and any call to an extension prepended with the dialing prefix would be routed to the state agent, which would interpret the INVITE as a request to reset the state of the specified extension.
### 7. Call pickup with a pickup agent

A further case allows calls to be picked up by phones that do not implement SIP-B pickup. Essentially, the phone makes a call that is routed to a "pickup agent" which executes the SIP-B pickup process on the phone’s behalf.

<table>
<thead>
<tr>
<th>Caller</th>
<th>Target</th>
<th>pickup</th>
<th>Executing</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td></td>
<td></td>
<td>456</td>
</tr>
</tbody>
</table>

```
|  | 1a INVITE |        |          |
|  | --------------|--------|-----------|
| 1b 180 Ringing |        |          |
|  | --------------|--------|-----------|
|  | 2 Dials *78123 |          |          |
|  | 3a INVITE |        |          |
|  | --------------|--------|-----------|
|  | 4a SUBSCRIBE |        |          |
|  | 4b 200 OK |        |          |
|  | --------------|--------|-----------|
|  | 5a NOTIFY |        |          |
|  | 5b OK |        |          |
|  | --------------|--------|-----------|
|  | 3b INVITE/Replaces |        |          |
|  | 3c 200 OK |        |          |
|  | --------------|--------|-----------|
|  | 3d 200 OK |        |          |
|  | 3e ACK |        |          |
|  | --------------|--------|-----------|
```

[Remainder of operation is as before.]

The major differences in processing are:
2. The executing user dials a special prefix and then the extension from which the call is to be picked up.

3. Because the dialed number starts with the pickup prefix "*78", the call is routed (either by a phone dialplan or an outgoing call proxy) to the pickup agent.

4 and 5. The pickup agent executes the same sequence of actions as a SIP-B phone would.

3b. After determining the Caller’s URI, the pickup agent appends a Replaces: header to the INVITE and forwards it as if it was proxying the call.

If the target phone is represented by a state agent, then the pickup agent and the state agent could communicate by non-SIP means rather than using SUBSCRIBE and NOTIFY.
8. With improved pickup agent

The previous packet flow is suboptimal in that the pickup agent acts as a non-standard proxy, rewriting an INVITE into an INVITE with Replaces: header. We can adopt a technique from Dan Petrie to be strictly conformant: The pickup agent returns a 302 (Moved Temporarily) response to the executing phone, bearing a Contact: header with a URI <caller-URI?Replaces=dialog-parameters>. 
The major differences in processing are:

3. Instead of forwarding the INVITE to the Caller, the pickup agent returns a 302 response, whose Contact is a URI for the Caller with a Replaces header parameter specifying dialog 1a. Phone 456 then sends INVITE/Replaces 4a to Caller.
9. With proxy and improved pickup agent

Let’s combine some of the above flows to show a realistic case, where all signaling goes through the proxy, and we use the "improved" pickup agent to support a phone that does not do call pickup by itself.

<table>
<thead>
<tr>
<th>Caller</th>
<th>Proxy</th>
<th>Target</th>
<th>Pickup</th>
<th>Executing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a INVITE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1b INVITE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1c 180 Rin.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1d 180 Ringing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 Dials *78123</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3a INVITE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3b INVITE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4a SUBSCRIBE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4b SUBSC.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4c 200 OK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4d 200 OK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5a NOTIFY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5b NOTIFY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The overall call flow is:

1. Caller sends invite to the target (123).

2. Executing phone (456) dials *78123.

3. Due to the dialing prefix, the INVITE is routed to the pickup agent.
4. The pickup agent recognizes the dialing prefix *78 and puts the transaction aside for a moment. It initiates a SUBSCRIBE with "Expires:0" (one-time) to 123.

5. The target sends a NOTIFY back to the pickup agent.

3d. The pickup agent selects dialog 1 to be picked up, and returns a 302 for INVITE 3b to the proxy. The 302 contains "Contact: <caller-URI?Replaces=dialog-1-parameters>".

3e. The proxy starts another fork of INVITE 3a, sending an INVITE 3e with Replaces header to the Caller. INVITE 3e is accepted, replacing the dialog initiated by INVITE 1a. Caller then sends CANCEL 1e for INVITE 1a, leading to a 200 OK response for the CANCEL and a 487 response for INVITE 1a, as well as an ACK for the 487 response.
10. Group call pickup

There are three kinds of group call pickup:

10.1. Picking up any call to any member of a group of individuals

This is a pickup directed to "any call to any of the people in my group", that is, being able to pick up a call without having to first determine the specific extension to which it was directed.

If the proxy is configured to provide a URI that forwards to all members of the group, a SUBSCRIBE to that URI will be forwarded to all of the relevant user agents. However, no incoming call contains the group URI as part of its dialog identification, so it becomes questionable how the executing phone determines that it can pickup the call.

The common convention is that a pickup that has several possible targets should pick up the call that has been waiting longest. The dialog event package contains a "duration" parameter.

10.2. Picking up a call routed to several individuals

In terms of the old key phone systems, this situation is like an extension number that rings on several people’s multi-line phones.

This can be handled without additional work, because the proxy will already be routing this URI to all of the individuals’ user agents, and so it will route a SUBSCRIBE to that URI to all of the user agents as well, ensuring that the executing phone will see information about the incoming call.

10.3. Picking up a call that is not routed to a specific user agent

This is the situation of a call when the caller dials a number that is not immediately routed to any particular phone, but rather "rings in limbo" until someone picks it up. In SIP, this service is implemented via an intangible user agent for the AOR that is the endpoint of an early dialog, but never answers it.

Call pickup from this sort of "calling group" is very similar to call pickup from a phone or individual line, as long as the intangible UA implements the needed SIP features to participate in a call pickup.
11. Call park

Retrieving a call from call parking is very much like call pickup of a ringing call.

To park a call, the phone transfers it to an extension that is maintained by a "park server", a multi-line software UA that accepts and maintains dialogs to a set of extensions that are defined as "parking orbits".

To retrieve a parked call, a phone executes a call pickup operation, except that the parameters of the Replaces header do not include the "early-only" operation, so that the confirmed dialog with the park server can be retrieved.

When call pickup and call retrieve are implemented by a pickup agent, the pickup agent is the only component that needs to distinguish the parking orbits and generate the Replaces header specification in its 302 response differently.
12. Alternative retrieval mechanism for call park

Section 11 envisions that calls are retrieved from parking much like ringing calls are picked up, which requires that the park server can respond to SUBSCRIBE requests for the dialog event package. To avoid this requirement, call retrieval can be implemented by sending the INVITE for call retrieval directly to the park server, which will itself determine the call to be retrieved. The park server can put the Executor in contact with the Caller by accepting the INVITE and then executing a "consultative transfer" operation, sending a 302 response, or other techniques.

This technique can reduce the number of SIP features that the park server needs to support, at the expense of implementing more application-level processing.
This example shows the park server returning a 302 response.

<table>
<thead>
<tr>
<th>Caller</th>
<th>Park Server 123</th>
<th>Executing 456</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Existing dialog</td>
<td>&lt;===============</td>
<td></td>
</tr>
<tr>
<td>2 Call Pickup on 123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3a INVITE *78123</td>
<td>&lt;=--------------</td>
<td></td>
</tr>
<tr>
<td>3b 302 Caller</td>
<td>&lt;=--------------</td>
<td></td>
</tr>
<tr>
<td>4a INVITE/Replaces</td>
<td>&lt;=--------------</td>
<td></td>
</tr>
<tr>
<td>4b 200 OK</td>
<td>&lt;=--------------</td>
<td></td>
</tr>
<tr>
<td>4c ACK</td>
<td>&lt;=--------------</td>
<td></td>
</tr>
<tr>
<td>5a BYE</td>
<td>&lt;=--------------</td>
<td></td>
</tr>
<tr>
<td>5b 200 OK</td>
<td>&lt;=--------------</td>
<td></td>
</tr>
</tbody>
</table>

1. We start at the point where Caller has a dialog established with the park server on parking orbit 123.

2. The user of phone 456 activates the call pickup feature for extension 123. This may be activated by a special button on the phone, or through a dialing prefix which is intercepted by the phone. In this example, the user dials "*78123".

3. The INVITE to *78123 is routed to the park server.
The park server determines that it has a dialog established on parking orbit 123, and extracts the dialog’s identifiers. The park server sends a 302 (Moved Temporarily) response, with a Contact header that specifies the Caller’s contact URI, and provides a header parameter for a Replaces header specifying the dialog identifiers for the call on orbit 123.

4. Due to the 302 response, UA 456 sends an INVITE with a Replaces header to Caller. Caller accepts the INVITE and replaces dialog 1 with it.

5. Caller terminates the dialog with the park server.
13. Security considerations

All of these mechanisms (with the exception of using an integrated state agent and pickup agent) depend on SUBSCRIBE/NOTIFY for the dialog event package. But SUBSCRIBE reveals information which could be sensitive -- we have just shown that anyone who can successfully SUBSCRIBE to an extension can take calls from that extension!

Thus, we need to ensure that the acceptance of SUBSCRIBEes is controlled by suitable security mechanisms.
14. Revision history

14.1. Changes from draft-worley-sipping-pickup-01 to draft-worley-siping-pickup-02

Added Alternative retrieval mechanism, Revision history, and Related work sections.

Updated contact information.

15. Informative References


Author’s Address

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

Phone: +1 781 647 9199
Email: worley@ariadne.com