

BLISS	M. Procter	
Internet-Draft	VoIP.co.uk	
Intended status: Informational	August 17, 2008	
Expires: February 18, 2009		

[TOC](#)

Implementing Call Park and Retrieve using the Session Initiation Protocol (SIP)

draft-procter-bliss-call-park-extension-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 February 18, 2009.

Abstract

Call Park and Call Retrieve are useful telephony services that are familiar to many users. Existing implementations using the Session Initiation Protocol (SIP) show that a variety of approaches can be taken, with varying degrees of interoperability. This draft discusses a number of feature variations, and how they may be implemented.

Table of Contents

- [1.](#) Overview
- [2.](#) Parking a call
 - [2.1.](#) Parking a call without an orbit
 - [2.2.](#) Parking a call with a specified orbit
 - [2.3.](#) Parking a call with a Park-Server allocated orbit
 - [2.4.](#) A failed attempt to park a call

- [2.5. Parking a call without a Park Server](#)
- [3. Retrieving a Parked Call](#)
 - [3.1. Retrieving a call from a Park Server](#)
 - [3.2. Retrieving a call from a User Agent](#)
- [4. User Agent Configuration](#)
- [5. Acknowledgements](#)
- [6. Security Considerations](#)
- [7. IANA Considerations](#)
- [8. References](#)
 - [8.1. Normative References](#)
 - [8.2. Informative References](#)
- [§ Author's Address](#)
- [§ Intellectual Property and Copyright Statements](#)

1. Overview

[TOC](#)

Call Park is a feature that enables UAs to make a call inactive but not terminated, in such a way as to allow the call to be resumed by the UA that parked the call, or by a different UA.

This feature is typically used when User A wishes to transfer a call in progress to User B, but doesn't necessarily know how to reach User B's UA directly. In this situation, User A parks the call, and then tells User B where the call is parked. User B may then retrieve the call using a convenient UA.

Other uses include allowing multiple calls to be parked at the same 'location', and forming a queue. In this way, a simple 'ACD' (Automatic Call Distribution) system can be implemented that permits calls to be initially sorted and placed in one of a number of queues, ready to be handled when an appropriate agent becomes available (and retrieves the next call from the queue).

In all cases, the parked call is subsequently identifiable by a short (typically 3 or 4 digit) label known as an 'orbit'. This orbit is often allocated by the user parking the call, but some environments favour allocation of the orbit by a Park Server. Both approaches are described in this document.

Having a Park Server for parked calls is a useful approach, which secures parked calls against User Agent rebooting and other losses of service. However, being able to park and retrieve calls without a Park Server is also a useful model, both in terms of decentralised network design and also for smaller installations that don't necessarily merit a separate Park Server. Therefore, parking with and without a Park Server are discussed in this document.

[TOC](#)

2. Parking a call

A basic call flow for Call Park is given in

[\[I-D.ietf-sipping-service-examples\]](#) (Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, "Session Initiation Protocol Service Examples," July 2008.) (section 2.15), and this forms the basis of the feature. This approach uses the SIP dialog ID between the parked endpoint and the park server itself as the unique parked call identifier. Using the dialog ID has a number of advantages since it is unique and allocated by both the parked user and the Park Server. However, it is also long, which can lead to problems when trying to identify parked calls by verbal or human-written mechanisms.

Traditional PBX users have become accustomed to parking a call against a short number (typically 3 or 4 digits), and then using this identifier to communicate to the retrieving party which call to retrieve. This information may be passed verbally, or by means of small paper notes. Whilst collisions may occur, they are generally avoided satisfactorily by administrative policies.

This draft attempts to reconcile these two models by allowing a short label to be attached to a parked call (the 'orbit'). The retrieving party can then use the same tag to locate the relevant dialog ID in order to retrieve the parked call.

2.1. Parking a call without an orbit

[TOC](#)

Certain environments do not require an 'orbit' to be used, either because calls are parked in a single queue, or the dialog identifiers are readily passed between concerned UAs. In this scenario, the flow described in [\[I-D.ietf-sipping-service-examples\]](#) (Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, "Session Initiation Protocol Service Examples," July 2008.) is followed without deviation.

2.2. Parking a call with a specified orbit

[TOC](#)

The message flow of parking a call in this scenario is identical to that illustrated in [\[I-D.ietf-sipping-service-examples\]](#) (Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, "Session Initiation Protocol Service Examples," July 2008.). The difference that this draft introduces is in the REFER message to the Park Server. The details of the REFER message changes are discussed below.

Alice	Bob	Park Server	Carol
	INVITE F1		
	----->		
	180 Ringing F2		
	<-----		
	200 OK F3		
	<-----		
	ACK F4		
	----->		
	RTP Media		
	<=====		
	Bob Parks Call		
		REFER Refer-To: A F5	
		----->	
		202 F6	
		<-----	
		NOTIFY F7	
		<-----	
		200 F8	
		----->	
	INVITE F9 Replaces: B		
	<-----		
	200 OK F10		
	----->		
	ACK F11		
	<-----		
	(Music-on-Hold or other RTP?)		
	<=====		
	BYE F12		
	----->	NOTIFY F14	
	200 OK F13	<-----	
	<-----	200 OK F15	
		----->	

The URI <sips:park-server@example.com;orbit=1234> is used instead of directing the request to the URI <sips:park@server.example.com>. The addition of the orbit parameter effectively tags the parked call with a short memorable code entered by the user.

F5 REFER Bob -> Park Server

```
REFER sips:park-server@example.com;orbit=1234 SIP/2.0
Via: SIP/2.0/TLS client.biloxi.example.com:5061
    ;branch=z9hG4bKnashds9
Max-Forwards: 70
From: Bob <sips:bob@biloxi.example.com>;tag=02134
To: Park Server <sips:park-server@example.com;orbit=1234>
Call-ID: 4802029847@biloxi.example.com
CSeq: 1 REFER
<allOneLine>
  Refer-To: <sips:alice@client.atlanta.example.com?Replaces=
    12345601%40atlanta.example.com%3Bfrom-tag%3D314159
    %3Bto-tag%3D1234567>
</allOneLine>
  Referred-By: <sips:bob@biloxi.example.com>
  Contact: <sips:bob@client.biloxi.example.com>
  Content-Length: 0
```

2.3. Parking a call with a Park-Server allocated orbit

[TOC](#)

Sometimes an orbit number assignment policy needs to be implemented. This may be to ensure that all orbit numbers are a particular length, or have a form that means that they can be dialled directly (given suitable extensions to an Application Server). It may also be implemented to eliminate the problem of trying to park more than one call on the same orbit.

To enforce a policy, we ensure that the orbit number is not allocated by the UA (entered by the user, or by configuration etc.) but is instead allocated by the Park Server, and relayed to the UA. The approach taken here is analogous to the Conference Factory approach described in [\[RFC4579\] \(Johnston, A. and O. Levin, "Session Initiation Protocol \(SIP\) Call Control - Conferencing for User Agents," August 2006.\)](#). Bob sends a REFER to the preconfigured Park Server URI, but without any 'orbit' parameter added. The Park Server then responds by redirecting Bob to the correct orbit by using a '302 Moved Temporarily' response. The orbit can then be found by inspecting this new target. Note that an intermediate proxy may recurse on this 302 response and Bob may never see the redirect. In this scenario, Bob can still extract the orbit from Contact of 2xx response to the original REFER.

Alice	Bob	Park Server	Carol
	INVITE F1		
	----->		
	180 Ringing F2		
	<-----		
	200 OK F3		
	<-----		
	ACK F4		
	----->		
	RTP Media		
	<=====		
	Bob Parks Call		
		REFER Refer-To: A F5	
		----->	
		302 Orbit allocated F6	
		<-----	
		REFER Refer-To: A F7	
		----->	
		202 Accepted F8	
		<-----	
		NOTIFY	
		<-----	
		200 OK	
		----->	
	INVITE Replaces: Bob		
	<-----		
	200 OK		
	----->		
	ACK		
	<-----		
	(Music-on-Hold or other RTP?)		
	<=====		
	BYE		
	----->		
		NOTIFY	
	200 OK		
	<-----		
		200 OK	
	<-----		
		----->	

F5 REFER Bob -> Park Server

```
REFER sips:park-server@example.com SIP/2.0
Via: SIP/2.0/TLS client.biloxi.example.com:5061
    ;branch=z9hG4bKnashdsB
Max-Forwards: 70
From: Bob <sips:bob@biloxi.example.com>;tag=22134
To: Park Server <sips:park-server@example.com>
Call-ID: 4802029847@biloxi.example.com
CSeq: 1 REFER
<allOneLine>
  Refer-To: <sips:alice@client.atlanta.example.com?Replaces=
    12345601%40atlanta.example.com%3Bfrom-tag%3D314159
    %3Bto-tag%3D1234567>
</allOneLine>
  Referred-By: <sips:bob@biloxi.example.com>
  Contact: <sips:bob@client.biloxi.example.com>
  Content-Length: 0
```

F6 302 Orbit Allocated Park Server -> Bob

```
SIP/2.0 202 Orbit Allocated
Via: SIP/2.0/TLS client.biloxi.example.com:5061
    ;branch=z9hG4bKnashdsB
    ;received=192.0.2.105
From: Bob <sips:bob@biloxi.example.com>;tag=22134
To: Park Server <sips:park-server@example.com>;tag=56324
Call-ID: 4802029848@biloxi.example.com
CSeq: 1 REFER
Contact: <sips:park-server@example.com;orbit=1234>
Content-Length: 0
```

Once Bob's UA learns of the allocated orbit (either by finding it in the 302 response, or by finding it in the Contact of the 202 response to the REFER), it can be passed to the user (Bob) in an appropriate manner.

2.4. A failed attempt to park a call

[TOC](#)

A Park Server may choose to reject a park attempt for many reasons, including prohibiting multiple calls being parked against the same orbit, or prohibiting certain users from parking calls on certain orbits. Whatever the reason, the response sent to Bob will enable Bob to take appropriate action. The following example shows the Park Server rejecting a call due to the orbit already being in use.

Alice	Bob	Park Server	Carol
INVITE F1			
----->			
180 Ringing F2			
<-----			
200 OK F3			
<-----			
ACK F4			
----->			
RTP Media			
<=====			
Bob Parks Call			
	REFER Refer-To: A F5		
	----->		
	486 Busy Here		
	<-----		

When Bob's parking attempt is rejected, Bob may choose to attempt to park the call again, but using a different orbit number. The ability for Bob to recover from failed parking attempts such as this without dropping the call to Alice is an important consequence of Bob sending the REFER to the Park Server, rather than sending the REFER to Alice so that she can park herself.

2.5. Parking a call without a Park Server

[TOC](#)

Sometimes, it is useful to be able to park a call without using a Park Server. The original dialog between Alice and Bob is maintained, even though Bob has notionally parked the call. As a consequence, the only changes that occur may be within Bob's UA, and will not necessarily involve any SIP signalling.

3. Retrieving a Parked Call

[TOC](#)

In order to retrieve a parked call, Carol needs to obtain the dialog identifiers for the dialog between Alice and wherever Alice is parked. The dialog identifiers can be obtained by issuing a SUBSCRIBE for the dialog event package [\[RFC4235\] \(Rosenberg, J., Schulzrinne, H., and R. Mahy, "An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol \(SIP\)," November 2005.\)](#). The resulting NOTIFY will contain details of all pertinent calls, including the dialog identifiers. Carol may (if presented with multiple dialogs) choose

which call to retrieve. Many implementations choose the first dialog listed, although some use the <duration> element to identify which call has been parked for the longest time. Obtaining the dialog information in this way follows the flow described in

[\[I-D.ietf-sipping-service-examples\]](#) (Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, "Session Initiation Protocol Service Examples," July 2008.).

3.1. Retrieving a call from a Park Server

[TOC](#)

By subscribing to the dialog event package [\[RFC4235\]](#) (Rosenberg, J., Schulzrinne, H., and R. Mahy, "An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP)," November 2005.) at the same URI used for parking the call, i.e. <sips:park-server@example.com;orbit=1234>, all the information that is required for the call to be retrieved by C is delivered in the corresponding NOTIFY.

Similarly, if the call was parked in an environment that does not require 'orbit' parameters, subscribing to the URI used for parking the call, i.e. <sips:park-server@example.com>, will still result in the necessary information being provided for the call to be retrieved.

SUBSCRIBE sips:park-server@example.com;orbit=1234 SIP/2.0
Via: SIP/2.0/TLS chicago.example.com:5061;branch=z9hG4bK92bz
Max-Forwards: 70
From: Carol <sips:carol@chicago.example.com>;tag=8672349
To: <sips:park-server@example.com;orbit=1234>
Call-ID: xt4653gs2ham@chicago.example.com
CSeq: 1 SUBSCRIBE
Contact: <sips:carol@client.chicago.example.com>
Event: dialog
Subscription-State: active;expires=0
Accept: application/dialog-info+xml
Content-Length: 0

F2 200 OK Park Server -> Carol

SIP/2.0 200 OK

Via: SIP/2.0/TLS chicago.example.com:5061;branch=z9hG4bK92bz
;received=192.0.2.114

Max-Forwards: 70

From: Carol <sips:carol@chicago.example.com>;tag=8672349

To: <sips:park-server@example.com;orbit=1234>;tag=1234567

Call-ID: xt4653gs2ham@chicago.example.com

CSeq: 1 SUBSCRIBE

Content-Length: 0

F3 NOTIFY Park Server -> Carol

NOTIFY sips:carol@client.chicago.example.com SIP/2.0

Via: SIP/2.0/TLS chicago.example.com:5061;branch=z9hG4bK93ca

Max-Forwards: 70

To: Carol <sips:carol@chicago.example.com>;tag=8672349

From: <sips:park-server@example.com;orbit=1234>;tag=1234567

Call-ID: xt4653gs2ham@chicago.example.com

CSeq: 2 NOTIFY

Contact: <sips:park-server@example.com;orbit=1234>

Event: dialog

Subscription-State: terminated

Content-Type: application/dialog-info+xml

Content-Length: ...

<?xml version="1.0"?>

<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"

version="0" state="full"

entity="sips:park@park.server.example.com;orbit=1234">

<dialog id="94992014524" call-id="12345600@atlanta.example.com"

local-tag="3145678" remote-tag="1234567" direction="recipient"

remote-uri="alice@atlanta.example.com"

remote-target="alice@client.atlanta.example.com">

<state>confirmed</state>

</dialog>

</dialog-info>

F4 200 OK Carol -> Park Server

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS chicago.example.com:5061;branch=z9hG4bK93ca
To: Carol <sips:carol@chicago.example.com>;tag=8672349
From: <sips:park@server.example.com;orbit=1234>;tag=1234567
Call-ID: xt4653gs2ham@chicago.example.com
CSeq: 2 NOTIFY
Contact: <sips:carol@client.chicago.example.com>
Content-Length: 0
```

The remainder of the frames are the same as the corresponding frames from [\[I-D.ietf-sipping-service-examples\] \(Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, "Session Initiation Protocol Service Examples," July 2008.\)](#), since the required dialog ID has been obtained through the SUBSCRIBE / NOTIFY cycle from the Park Server.

3.2. Retrieving a call from a User Agent

[TOC](#)

Retrieving a parked call from a User Agent is very similar to retrieving a call from a Park Server. The main difference is identifying which URI should be used for the initial subscription, in order to find the dialog identifiers for the parked call. The URI most likely to be known for a particular User Agent is the AoR. But this may correspond to multiple UAs. Therefore, if Carol subscribes to the AoR, she should be prepared for multiple NOTIFY responses, should her SUBSCRIBE fork.

It is tempting to state that Carol should subscribe to the GRUU of the UA with the parked call. This will work well if the GRUU is known in advance (possibly by Carol having a long-lived subscription to Bob's UA). If the GRUU is not known in advance, then it may prove too onerous to expect it to be typed in on Carol's UA, just to retrieve a parked call.

Open issue: Is subscribing to the AoR the best we can offer?

Using the 'orbit' parameter in conjunction with parking calls on User Agents can lead to difficulties in ensuring that the 'orbit' parameter is delivered to the User Agent. Current thinking favours an approach based upon [\[RFC4244\] \(Barnes, M., "An Extension to the Session Initiation Protocol \(SIP\) for Request History Information," November 2005.\)](#).

Open issue: Once this has moved on a little, we can ensure it meets our needs.

If the 'orbit' parameter is not used, then Carol will eventually find out about all the dialogs currently in progress at Bob's UA.

Distinguishing the parked call from other dialogs may prove challenging

(assuming that all calls marked as held is not acceptable). There may be an opportunity to reuse some BLISS-MLA work here, which permits dialogs to be marked as 'exclusive', which indicates that other UAs should not attempt to pick them up.

Open issue: How should a parked call be identified when parked on a UA with other (potentially non-parked) dialogs in progress? Does this problem only exist when an orbit is not used?

4. User Agent Configuration

[TOC](#)

For Bob and Carol to be able to park and retrieve calls using a Park Server, both need to be configured with the URI of the Park Server. In addition, Bob and Carol should be configured to understand whether or not an orbit will be required for park and retrieve. Finally, Bob also needs to be configured to determine whether Bob should provide the orbit or whether the orbit will be allocated by the Park Server.

5. Acknowledgements

[TOC](#)

The following individuals were part of the Call Park Design Team, and have helped to shape this document:

Francois Audet, Jason Fischl, Derek Macdonald, Shida Schubert, Sanjay Sinha, Dale Worley and Theo Zourzouvillys.

6. Security Considerations

[TOC](#)

None.

7. IANA Considerations

[TOC](#)

Open issue: presumably need to define the new uri-parameter 'orbit'.

8. References

[TOC](#)

8.1. Normative References

[TOC](#)

[I-D.ietf-sipping-service-examples]	Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, " Session Initiation Protocol Service Examples ," draft-ietf-sipping-service-examples-15 (work in progress), July 2008 (TXT).
[RFC4235]	Rosenberg, J., Schulzrinne, H., and R. Mahy, " An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP) ," RFC 4235, November 2005 (TXT).

8.2. Informative References

[TOC](#)

[RFC4579]	Johnston, A. and O. Levin, " Session Initiation Protocol (SIP) Call Control - Conferencing for User Agents ," BCP 119, RFC 4579, August 2006 (TXT).
[RFC4244]	Barnes, M., " An Extension to the Session Initiation Protocol (SIP) for Request History Information ," RFC 4244, November 2005 (TXT).

Author's Address

[TOC](#)

	Michael Procter
	VoIP.co.uk
	Commerce House
	Telford Road
	Bicester, Oxfordshire OX26 4LD
	UK
Email:	michael@voip.co.uk
URI:	http://voip.co.uk

Full Copyright Statement

[TOC](#)

Copyright © The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE

INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.