BLISS	M. Procter
Internet-Draft	VoIP.co.uk
Intended status: Informational	April 23, 2009
Expires: October 25, 2009	

TOC

Implementing Call Park and Retrieve using the Session Initiation Protocol (SIP)
draft-procter-bliss-call-park-extension-04

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79. 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.
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 October 25, 2009.

Copyright Notice

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 using existing techniques. An additional URI parameter is also described, which enables further common use-cases to be implemented.

Table of Contents

---

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.

Multiple Park Servers can be beneficial in some enviroments for a variety of reasons including load-sharing and administrative policies. This document shows how support for multiple servers can easily be achieved whilst still permitting a single 'well-known' Park Server URI to be advertised for configuration.

---

2. Parking a call

TOC

A basic call flow for Call Park is given in [RFC5359] (Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, “Session Initiation Protocol Service Examples,” October 2008.) (section 2.15), and this forms the basis of the feature. The flow shows Alice and Bob in a call, when Bob decides to park the call by sending a REFER to the Park Server.

It is worth noting that whilst the flow is conceptually similar to an Unattended Transfer [RFC5359] (Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, “Session Initiation Protocol Service Examples,” October 2008.) (section 2.4), the REFER is sent to different endpoints in the two cases. For Unattended Transfer, the Transferor sends the REFER to the Transferee, instructing him to call the Transfer Target. For Call Park, the Transferor (Bob) sends the REFER to the Transfer Target (Park Server), instructing it to call the Transferee (Alice).

By following the Call Park model, we ensure that Bob has visibility over the success or failure of the park attempt. We also ensure that Bob does not rely on Alice to correctly pass the orbit parameter back from the Park Server for the centrally-allocated orbit number situation. Finally, because Bob sends the REFER to the Park Server, we give the Park Server the opportunity to challenge Bob and ensure that appropriate authorisation exists for the feature.

```
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 basic call flow described above 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 calls being parked 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 label to locate the relevant dialog ID in order to retrieve the parked call. Note that the orbit may be allocated by the User Agent parking the call or centrally by the Park Server.

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 [RFC5359] (Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, “Session Initiation Protocol Service Examples,” October 2008.) (section 2.15) is followed without deviation.

2.2. Parking a call with an orbit specified by the UA

TOC

The message flow of parking a call in this scenario is identical to that illustrated in [RFC5359] (Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, “Session Initiation Protocol Service Examples,” October 2008.) (section 2.15). The difference that this document introduces is in the REFER message to the Park Server.
In this scenario, it is assumed that Bob has entered a parking orbit in some manner appropriate to his UA. Once this is done, the REFER is sent to the URI <sips:park@server.example.com;orbit=1234> instead of simply directing the request to the URI <sips:park@server.example.com>. The addition of the orbit parameter to the URI effectively labels 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 an orbit specified by the Park Server

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.

```
           Alice           Bob        Park Server       Carol
             |              |              |              |
             | Active Call  |              |              |
             |<============>|              |              |
             |      Bob Parks Call         |              |
             |              |   REFER Refer-To: A F5      |
             |              |-------------->|              |
             |              |302 Orbit allocated F6       |
             |              |<--------------|              |
             |              |   REFER Refer-To: A F7      |
             |              |-------------->|              |
             |              |202 Accepted F8              |
             |              |<--------------|              |
             |              |   NOTIFY     |              |
             |              |<--------------|              |
             |              |    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
```

This is also the means by which multiple Park Servers can be deployed. A REFER to <sips:park@server.example.com> might result in a 302 response, nominating <sips:park@server-1.example.com;orbit=1234> as the desired target.

Different network architectures may result in different behaviours as seen by Bob. In particular, whether Bob sees the 302 response will depend on whether or not an intermediate proxy recurses on it. Therefore, Bob's UA must be prepared to extract the orbit parameter from either the 302 response (if one is seen) or the Contact header of the 2xx response to his REFER.

Since this technique may also be used to resolve the problem of parking multiple calls on the same orbit, Bob's UA must be prepared to extract the orbit even if it provided one in the initial request. If the orbit differs to the one requested, the extracted orbit should be rendered to Bob in an appropriate manner.

```
F8 202 Accepted Park Server -> Bob

SIP/2.0 202 Accepted
Via: SIP/2.0/TLS client.biloxi.example.com:5061
 ;branch=z9hG4bKnashds9
 ;received=192.0.2.105
From: Bob <sips:bob@biloxi.example.com>;tag=02134
To: Park Server <sips:park@server.example.com>;tag=56323
Call-ID: 4802029847@biloxi.example.com
Contact: <sips:park@server.example.com;orbit=1234>
CSeq: 1 REFER
Content-Length: 0
```

This variation is only possible on Park Servers capable of generating Contact URIs of the correct form, i.e. with an 'orbit' URI parameter, in either a 302 response, or in a 2xx response to the REFER. Park Servers unable to generate URIs of this form are therefore confined to environments that don't require centrally allocated parking orbits.

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.

---

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 [RFC5359] (Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, “Session Initiation Protocol Service Examples,” October 2008.) (section 2.15). 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.

```
                Alice           Bob        Park Server       Carol
                  |              |              |              |
                  |              |              | SUBSCRIBE F1 |
                  |              |              |<-------------|
                  |              |              |  200 OK F2   |
                  |              |              |------------->|
                  |              |              |  NOTIFY F3   |
                  |              |              |------------->|
                  |              |              |  200 OK F4   |
                  |              |              |<-------------|
                  |              |              |              |
                  |              |              |              |
                  |           INVITE Replaces: Park Server F5  |
                  |<-------------------------------------------|
                  |              |              |   200 F6     |
                  |------------------------------------------->|
                  |              |              |    ACK F7    |
                  |<-------------------------------------------|
                  |                  RTP Media                 |
                  |<===========================================>|
                  |           BYE F8            |              |
                  |---------------------------->|              |
                  |          200 OK F9          |              |
                  |<----------------------------|              |
```

F1 SUBSCRIBE Carol -> Park Server

```
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@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 [RFC5359] (Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, “Session Initiation Protocol Service Examples,” October 2008.), since the required dialog ID has been obtained through the SUBSCRIBE / NOTIFY cycle from the Park Server.

TOC

4. User Agent Considerations

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.
Any orbit received from the Park Server, either in the Contact URI of a 302 or 2xx response to REFER, should be rendered to the user in an appropriate manner, even if an orbit was provided in the initial REFER. *** FIXME UNLESS it is the same as requested? This is to allow Park Servers to implement various policy decisions and allocate orbits as required. Failure to do this may lead to a call being parked on a different orbit to the expected one, and hence being effectively lost. If the UA provided an orbit in the REFER request, and no orbit is received from the Park Server, then the UA may assume that the call has been parked against the requested orbit correctly.

---

5. Park Server Considerations

TOC

It is expected that Park Servers will not necessarily support all the feature variations described in this document, at least not simultaneously. Therefore Park Servers should offer the set that is most appropriate for the target environment. For example, some Park Servers may offer centrally allocated orbits, some may not, and some may be configurable. Other policy-related decisions include how to handle more than one call being parked on a particular orbit. *** FIXME more!

---

6. Other Implementations and Interoperability

TOC

Several vendors already implement Call Park/Retrieve in different ways. Many of these approaches use non-standard/proprietary extensions to achieve some of the goals that this document addresses.
Interoperability is rarely a driving concern with proprietary extensions, since a non-interoperable implementation is simply regarded as not fully implemented.
This section describes some of the approaches that have been implemented. It is not intended to be an exhaustive review of all actual or possible implementations, nor is it intended to give sufficient detail for new implementations. Instead, it aims to give an overview sufficient to let the reader compare the different tradeoffs that have been seen in the field.

6.1. Parking by blind transfer

TOC

Some implementations support call park by performing a Blind Transfer to a URI of the form 'sip:xxxx@example.com', where 'xxxx' represents the park number, or orbit. To retrieve a parked call, users generally dial something of the form 'sip:*4xxxx@example.com'. The park number and the retrieval code both exist in the local dialling domain, and tend to be easy for UAs to dial.

This approach is generally supported by most UAs, since performing a blind transfer is a commonly implemented feature. Dialling the retrieval code is also commonly supported as it resembles a 'star code', or 'Vertical Service Code'.

This approach doesn't permit park numbers to be centrally allocated, as the user is required to select one when parking. The park numbers must also be chosen so as to not clash with local extension numbers.

6.2. Parking by blind transfer with central allocation

TOC

At least one implementation supports parking a call by performing a Blind Transfer to a preconfigured URI. The allocated parking orbit is returned to the UA in the 202 response to the REFER, in a custom header. The UA performing the park must understand this header, and render the contents to the user. To retrieve the call, the park number is simply dialled as though it were a local extension.

This approach requires explicit support in the UA used to park calls, as the park number needs to be extracted from the response to the REFER, and displayed to the user. Retrieving a call can be performed by most UAs.

6.3. Parking with orbit parameters

TOC

Some implementations choose an approach that is similar to the one described in this document. A Blind Transfer to a URI of the form 'sip:callpark@example.com;orbit=7001' is performed. Rather than retrieving the call as described above, a call is made to the corresponding URI 'sip:pickup@example.com;orbit=7001'.

This approach requires support in the UA used to park calls, since the target of the Blind Transfer requires a uri parameter to be added. Support is also required in the UA used to retrieve calls, since again the URI in question requires a parameter.

No central allocation of park numbers is supported in this approach. However, the parked numbers may overlap with the local extension plan if desired, since calls are not directly placed to the park numbers.

6.4. Parking on the User Agent

TOC

Some implementations require the User Agents to perform all park-related functionality. To do this, each UA that is expected to park calls is configured with a range of park numbers. Should a UA wish to park a call, it keeps the call on hold locally, and registers itself against one of its park numbers. Any other UA can then retrieve the call by dialling the park number. The call arrives at the parking UA, which can connect it to the held call using 3rd party call control. This approach requires explicit support in the UA used to park calls. Retrieving a call can be performed by most UAs.
Whilst attractive from the perspective of not needing any centralised server support, this approach suffers from the requirement to preconfigure all the UAs with park numbers, in case they need to park a call. Failure to provision enough numbers will prevent calls from being parked on a particular phone. Conversely, provisioning too many numbers will rapidly deplete the local extension numbering space.

---

7. 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.

---

8. Security Considerations

TOC

None.

---

9. IANA Considerations

TOC

Open issue: According to [RFC3969] (Camarillo, G., “The Internet Assigned Number Authority (IANA) Uniform Resource Identifier (URI) Parameter Registry for the Session Initiation Protocol (SIP),” December 2004.), defining a URI parameter can only be done in a

standards-track RFC. That doesn't sound like the sort of thing this document will do, nor the sort of thing BLISS will do either. However, [RFC4240] (Burger, E., Van Dyke, J., and A. Spitzer, “Basic Network Media Services with SIP,” December 2005.) ('netann') defines values that are included in the current registry, and it is most definately 'Informational'.
This specification adds a new value to the IANA "SIP/SIPS URI Parameters" registry as defined in [RFC3969] (Camarillo, G., “The Internet Assigned Number Authority (IANA) Uniform Resource Identifier (URI) Parameter Registry for the Session Initiation Protocol (SIP),” December 2004.).

Parameter Name	Predefined Values	Reference
orbit	No	RFCxxxx

The ABNF [RFC5234] (Crocker, D. and P. Overell, “Augmented BNF for Syntax Specifications: ABNF,” January 2008.) grammar for this parameter is shown below.

```
orbit-param     = "orbit" EQUAL orbit-value
orbit-value     = 1*DIGIT
```

10. References

TOC

10.1. Normative References

TOC

[RFC5359]	Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, “Session Initiation Protocol Service Examples,” BCP 144, RFC 5359, October 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).
[RFC5234]	Crocker, D. and P. Overell, “Augmented BNF for Syntax Specifications: ABNF,” STD 68, RFC 5234, January 2008 (TXT).
[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 (TXT).

10.2. 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).
[RFC4240]	Burger, E., Van Dyke, J., and A. Spitzer, “Basic Network Media Services with SIP,” RFC 4240, December 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