Implementing Call Park and Retrieve using the Session Initiation Protocol (SIP)
draft-procter-bliss-call-park-extension-03

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 April 24, 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 using existing techniques. An additional URI parameter is also described, which enables further common use-cases to be implemented.
Table of Contents

1. Overview ......................................................... 3
2. Parking a call ..................................................... 3
   2.1. Parking a call without an orbit .......................... 5
   2.2. Parking a call with an orbit specified by the UA ....... 5
   2.3. Parking a call with an orbit specified by the Park Server ........................................... 6
   2.4. A failed attempt to park a call ......................... 9
3. Retrieving a Parked Call ........................................ 9
4. User Agent Considerations ....................................... 12
5. Park Server Considerations ..................................... 13
6. Acknowledgements ................................................ 13
7. Security Considerations ......................................... 13
8. IANA Considerations .............................................. 14
9. References ....................................................... 14
   9.1. Normative References ..................................... 14
   9.2. Informative References ................................... 14
Author’s Address ..................................................... 14
Intellectual Property and Copyright Statements .................. 16
1. Overview

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 environments 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

A basic call flow for Call Park is given in [I-D.ietf-sipping-service-examples] (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 [I-D.ietf-sipping-service-examples] (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.

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

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

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] (section 2.15) is followed without deviation.

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

The message flow of parking a call in this scenario is identical to that illustrated in [I-D.ietf-sipping-service-examples] (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

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]. 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
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Call</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;============&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob Parks Call</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>REFER Refer-To: A F5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-------------&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>302 Orbit allocated F6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REFER Refer-To: A F7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-------------&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>202 Accepted F8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOTIFY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200 OK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;------------</td>
<td></td>
</tr>
</tbody>
</table>

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%3D134159%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

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

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]. 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] (section 2.15).

By subscribing to the dialog event package [RFC4235] 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.

---

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
<th>Park Server</th>
<th>Carol</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBSCRIBE F1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200 OK F2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOTIFY F3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200 OK F4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;---------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INVITE Replaces: Park Server F5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;-------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>200 F6</td>
<td></td>
</tr>
<tr>
<td>&lt;-------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACK F7</td>
<td></td>
</tr>
<tr>
<td>&lt;-------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTP Media</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;=============================</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BYE F8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;-------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>200 OK F9</td>
<td></td>
</tr>
<tr>
<td>&lt;-------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 [I-D.ietf-sipping-service-examples], since the required dialog ID has been obtained through the SUBSCRIBE / NOTIFY cycle from the Park Server.

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. **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

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. Acknowledgements

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.

7. Security Considerations

None.
8. IANA Considerations

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

According to [RFC3969], 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] ('netann') defines values that are included in the current registry, and it is most definately 'Informational'.

9. References

9.1. Normative References

[I-D.ietf-sipping-service-examples]


9.2. Informative References


Author’s Address

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

Copyright (C) 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.