Feature Referral in the Session Initiation Protocol (SIP)
draft-audet-sipping-feature-ref-00

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 August 20, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2008).

Abstract

Feature referral allows for an application to make a high level request to a User Agent to perform an action or "feature", and let the User Agent actually execute the feature as it sees fit.
Feature referral uses the SIP REFER method with a Refer-To header field containing a URN.

Table of Contents

1. Terminology .............................................. 3
2. Introduction ............................................ 3
3. Overview .................................................. 5
4. User Agent Behavior ....................................... 6
   4.1. Dialog usage ......................................... 6
   4.2. Addressing the relevant parties ..................... 6
5. Call flows .................................................. 7
   5.1. Answer Call Operation ................................ 8
   5.2. Clear Connection ..................................... 9
   5.3. Deflect Call .......................................... 12
   5.4. Hold Call ............................................. 13
   5.5. Retrieve Call ......................................... 14
   5.6. Single Step Transfer Call Flow Example ............. 15
   5.7. Conference Calls ..................................... 17
   5.8. Separate Calls ....................................... 17
6. Security Considerations .................................... 17
7. IANA Considerations ....................................... 17
8. Acknowledgments .......................................... 17
9. References ................................................ 17
   9.1. Normative References ................................ 17
   9.2. Informational References ............................. 17
Authors' Addresses .......................................... 19
Intellectual Property and Copyright Statements ............. 20
1. Terminology

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

To simplify discussions of the REFER method and its extensions, the three terms below are being used throughout the document:

- REFER-Issuer: the UA issuing the REFER request
- REFER-Recipient: the UA receiving the REFER request
- REFER-Target: the UA designated in the Refer-To Uniform Resource Identifier (URI), which, for this specification, is a Uniform Resource Name (URN)

2. Introduction

Feature referral allows for an application (such as a proxy or a user agent) to make a high level request to a SIP [RFC3261] User Agent (UA) to perform an action or "feature", and let the the User Agent actually execute the feature as it sees fit. Feature referral uses the SIP REFER method [RFC3515] with a Refer-To header field containing a URN [RFC2141].

Feature referral is useful for collections of loosely coupled User Agents which would like to present a coordinated user experience (i.e., when the Application is co-resident in the UA). Among other things, this allows User Agents which handle orthogonal media types but which would like to be present in a single conversation to add and remove each other from the conversation as needed. This is especially appropriate when coordinating conversations among organizers, general purpose computers, and special purpose communications appliances like telephones, Internet televisions, in-room video systems, electronic whiteboards, and gaming devices. For example using feature referral, an Instant Messaging client could initiate a multiplayer gaming session and an audio session to a chat conversation. Likewise a telephone could add an electronic whiteboard session to a voice conversation. Finally, a computer or organizer could cause a nearby phone to dial from numbers or URIs in a document, email, or address book; allow users to answer or deflect incoming calls without removing hands from the computer keyboard; place calls on hold; and join other sessions on the phone or otherwise.

Feature referral is also useful for a wide range of third party applications that need to remotely control or influence a User Agent (for example, in Contact center environment). In pre-SIP
environments, these environments have been using "Computer Telephony Integration": for example, traditional PBXs use CTI protocols such as CSTA [ECMA269] to provide this functionality. CSTA works fine for legacy PBXs with legacy phones but is problematic in a SIP environment. For example, SIP includes totally new capabilities such as presence and instant messaging. SIP also supports multiple users with multiple devices operating at once, and with complex User Interfaces. Furthermore, multiple applications may want to simultaneously wish to interact with the device. Because of the lack of a native mechanism mechanism to achieve such control for SIP, implementors have had to implement such techniques as mapping CSTA’s ASN.1 encoding to XML then encapsulate it into SIP INFO requests in order to tunnel it to a SIP B2BUA [ECMA323], which then maps it to proprietary device control protocols or to SIP with proprietary and incompatible extensions. This document provides a clean and native way to meet the requirements.

CTI fundamentally requires two components:

- Monitoring - to learn the state of the UA
- Control - request the UA to perform certain features

SIP already provides some capabilities for monitoring, including the following:

- Dialog package - call states
- Registration package - phone status
- Conference package - conference status

SIP also provide a method for requesting UAs do perform certain task, i.e., REFER [RFC3515], but today is it limited. Specically:

- REFER does not allow for a UA to request another UA to respond to requests, e.g.,
  * A UA cannot request another UA to answer a call
  * A UA cannot request another UA to reject a call
- REFER does not allow for a UA to request another UA to invoke features, e.g.,
  * REFER does not allow for a UA to request another UA to place a call on hold, or to mute it
  * REFER does not allow for a UA to request another UA to transfer, conference, or park a call

Feature referral is consistent with the SIP call control framework [I-D.ietf-sipping-cc-framework] and is a natural expansion of the Application Interaction Framework [I-D.ietf-sipping-app-interaction-framework] which allows for referral to SIP resources (through the SIP URI scheme) and Web pages
(through the HTTP URI scheme).

3. Overview

A prototypical feature referral flow looks as per section 4.1 of [RFC3515]. The Refer-To URI in the REFER message includes a URN describing the feature. The first part of the URN, i.e., the Namespace Identifier, is intended to be in the formal space and assigned by IANA, as per the procedures of [RFC3406]. An alternative would be to use the service URN space [RFC5031]. Until this is resolved, this document will use the following namespace: "feature". The second part of the URN includes the feature name, and may be followed by a semi-colon and additional feature-specific parameters.

Feature referral are sent to a GRUU when a specific instance of a UA is the desired target. When the feature referral needs to be correlated to a specific dialog, the Target-Dialog header field is used [RFC4538]. Some primitives require a second dialog identifier (such as ConferenceCalls which causes the media from two dialogs to be mixed). The mechanism to convey this second dialog identifier is TBD.

The following is a list of sample features (using the CSTA TR/87 [TR87] minimal profile as a starting point):

- Answer call - urn:feature:AnswerCall
- Clear connection - urn:feature:ClearConnection
- Deflect call - urn:feature:DeflectCall
- Hold call - urn:feature:HoldCall
- Retrieve call - urn:feature:RetrieveCall
- Single step transfer -urn:feature:SingleStepTransfer
- Conference calls - urn:feature:ConferenceCalls
- Separate calls - urn:feature:SeparateCalls

Note that the very important "Make call" CTI primitive does not require a feature referral URN since it is accomplished by sending a normal REFER with a URI identifying the resource (e.g., a sip, sips or tel URI).

Of course, other features could also be added, beyond the realm of traditional telephony, e.g.:

- Add buddy to list - urn:feature:AddBuddy;sip@bob@example.com
- Send vCard - urn:feature:SendVCard
4. User Agent Behavior

4.1. Dialog usage

This document attempts to avoid using multiple dialog usages, for the reasons described in [RFC5057]. Therefore, this document will make use of the GRUU [I-D.ietf-sip-gruu], and the Target-Dialog header field [RFC4538] to associated and existing INVITE usage with a REFER arriving on a new dialog to facilitate authorization of that REFER.

In many use cases of feature referral, receiving notifications about the status of a REFER request are superfluous, as the Refer issuer often maintains a long duration subscription to the dialog package [RFC4235]. Suppression of the REFER notifications is done with the norefersub option-tag, defined in section 7 of [RFC4488]. When the norefersub option tag is present, a REFER request which would have created a new subscription and dialog becomes a standalone transaction instead, eliminating a multiple dialog usage. Each such standalone REFER transaction use a new (unique) Call-ID header field value.

In the most common usage, the controller maintains a long duration subscription to the dialog package, and sends REFER requests in separate dialogs. Each REFER would include the norefersub option-tag in a Supported header field.

In some cases, the controller does not maintain a dialog package subscription for the Refer-Receiver. This might be the case for a "webdialer" or other application which associates with other UAs on an adhoc and intermittent basis. An initial REFER request is sent to start a new dialog, which is followed by notifications for the refer event type (the norefersub option-tag is not used in this case).

4.2. Addressing the relevant parties

REFER requests contain a number of URIs which need to address the appropriate parties. A list of the relevant fields include the Request-URI, To URI, From URI, Contact URI, Refer-To URI, and the Referred-By URI, as well as the Target-Dialog itself. This section attempts to clarify what needs to be placed in each field.

In most cases, feature referral applies to dialogs or sessions on a specific UA, in which case a GRUU [I-D.ietf-sip-gruu] for a single UA (i.e., Contact URI) is used. Contact URIs for a UA can be discovered by subscribing to the registration package [RFC3680] for the relevant AORs.

In the cases where the controller does not care which specific UA it
manipulates, an AOR can be used instead. When an AOR is used, the REFER request can include appropriate caller-preferences to encourage selection of an appropriate Contact. The norefs sub option-tag is not used when the REFER Request-URI is an AOR, as the REFER Request could fork and cause very odd behavior. While, the controller can discourage a proxy from forking remote call control request by using the Request-Disposition: no-fork header field, insuring that no proxy forks requires the use of the callerpref option-tag in a Proxy-Require header field value. Use of Proxy-Require is not normally advised because any proxy in the chain of this request which did not support caller preferences would cause the request to fail.

The To header field in the REFER request normally contains the same URI as in the Request-URI. The From identifies the AOR of the controller. The Refer-To URI is the feature referral URN.

Many uses of feature referral require that the Refer-Receiver take some action in the context of an existing dialog. For example, the controller might want the Refer-Receiver to send terminate an existing dialog. To select the appropriate dialog from which to source the request, the Target-Dialog header specified in [RFC4538] is used.

5. Call flows

This sample provides non-normative sample calls flows for the features listed in Section 3. It is important to understand that the actual "realization" of the feature (i.e., the actual procedures invoked) are the sole responsibility of the Refer-Recipient. This document in no way attempts to standardize those procedures, and the call flow below are merely examples.

In all cases, the "controller" (i.e., the Refer-Issuer) could be Alice’s PC, PDA, or a third party application. The controlled device is Alice’s phone (i.e., the Refer-Recipient). The Refer-Target is obviously the feature referral URN. In all cases, it is assumed that the controller is subscribed to Alice’s Phone’s dialog package.

The call flows in this document use the following conventions. The dialog each message is sent in is shown on the left hand side. Selected Request-URI and header fields are shown. The contents of message bodies are shown for dialog-info+xml, sd, and sipfrag message bodies. For responses, the method is shown in parentheses. For reference, the messages are labeled F1, F2, etc.
5.1. Answer Call Operation

In message 1, Bob makes a call to Alice’s Phone. A notification of "trying" is sent to the controller. Alice’s phone automatically sends a "ringing" to Bob. Another notification of "early" is then sent to the controller. The controller then tells the phone to answer the call. Alice’s phone sends a notification of "confirmed" to the controller.

```
Controller       Alice        Bob
<< Controller subscribed >>
<< to Alice’s dialog events >>
F1 INVITE sip:Alice-AOR
F2 NOTIFY sip:Controller-GRUU
dialog-info+xml: dialog1=trying
F3 200 (NOTIFY)
F4 180 (INVITE)
F5 NOTIFY sip:Controller-GRUU
dialog-info+xml: dialog1=early
F6 200 (NOTIFY)
F7 REFER sip:Alice-GRUU
   To: sip:Alice-GRUU
   Refer-To: urn:feature:AnswerCall
   Target-Dialog: dialog1
F8 202 (REFER)
F9 200 (INVITE)
F10 ACK
F11 NOTIFY sip:Controller-GRUU
dialog-info+xml: dialog1=confirmed
F12 200 (NOTIFY)
```
Answer Call Flow Example

5.2. Clear Connection

Clear Connection is a perfect example of a feature whose treatment (and consequently, the resulting call flow) depends on the situation, for example, the state of the dialog between the remote parties.

Alice’s Phone and Bob are currently in an established dialog. The controller tells Alice’s phone to "clear the connection" with Bob’s phone.

Controller Alice Bob
<< Controller subscribed to >>> <<< Established dialog1 >>>>
<<< Alice’s dialog events >>>>
dialog3 F1 REFER sip:Alice-GRUU
To: sip:Alice-GRUU
Refer-To: urn:feature:ClearConnection
Target-Dialo: dialog1
------------------------------>
dialog3 F2 202 (REFER)
<-----------------------------
dialog1 F3 BYE sip:Bob-GRUU
------------------------------>
dialog1 F4 200 (BYE)
<-----------------------------
dialog2 F5 NOTIFY sip:Controller-GRUU
dialog-info+xml: dialog2=local-bye
<-----------------------------
dialog2 F6 200 (NOTIFY)
------------------------------>

Clear Connection in Established Dialog Call Flow Example

If Alice’s Phone and Bob are in an early dialog with Bob calling Alice, the call flow could be as follows.
Controller        Alice        Bob
<< Controller subscribed to >> << Alice’s dialog events >>>
dialog1             F1 INVITE sip:Alice-AOR
                     (dialog2)
<--------------------------->
dialog2             F2 NOTIFY sip:Controller-GRUU
dialog-info+xml: dialog1=trying
<--------------------------->
dialog2             F3 200 (NOTIFY)
-------------------------->
dialog1             F4 180 (INVITE)
-------------------------->
dialog2             F5 NOTIFY sip:Controller-GRUU
dialog-info+xml: dialog1=early
<--------------------------->
dialog2             F6 200 (NOTIFY)
<--------------------------->
dialog3             F7 REFER sip:Alice-GRUU
To: sip:Alice-GRUU
Refer-To: urn:ietf:feature:ClearConnection
Target-Dialog: dialog1
<--------------------------->
dialog3             F8 202 (REFER) (dialog3)
<--------------------------->
dialog1             F9 480 (INVITE)
<--------------------------->
dialog1             F10 ACK
<--------------------------->
dialog2             F11 NOTIFY (Controller-GRUU)
dialog-info+xml: dialog1=rejected
<--------------------------->
dialog2             F12 200 (NOTIFY)
<--------------------------->

Clear Connection in Early Dialog Call Flow Example

If Alice’s Phone and Bob are in an early dialog with Alice calling
Bob, the call flow could be as follows.
Controller | Alice | Bob
---|---|---
<< Controller subscribed to >> | << Alice’s dialog events >> |

dialog1 | F1 INVITE sip:Bob-AOR | -------------------------->

dialog2 | F2 NOTIFY sip:Controller-GRUU |
| dialog-info+xml: dialog1=trying |
| <-----------------------------| |

dialog2 | F3 200 (NOTIFY) | <-----------------------------| |

dialog1 | F4 180 (INVITE) | <-----------------------------| |

dialog2 | F5 NOTIFY sip:Controller-GRUU |
| dialog-info+xml: dialog1=early |
| <-----------------------------| |

dialog2 | F6 200 (NOTIFY) | <-----------------------------| |

dialog3 | F7 REFER sip:Alice-GRUU |
| To: sip:Alice-GRUU |
| Refer-To: urn:feature:ClearConnection |
| Target-Dialog: dialog1 |
| <-----------------------------| |

dialog3 | F8 202 (REFER) |
| <-----------------------------| |

dialog1 | F9 CANCEL | <-----------------------------| |

dialog1 | F10 200 (CANCEL) | <-----------------------------| |

dialog1 | F11 487 (INVITE) | <-----------------------------| |

dialog1 | F12 ACK | <-----------------------------| |

dialog1 | F13 NOTIFY sip:Controller-GRUU |
| dialog-info+xml: dialog1=rejected |
| <-----------------------------| |

dialog2 | F14 200 (NOTIFY) | <-----------------------------| |

Clear Connection Initiated Call Flow Example
### 5.3. Deflect Call

Bob makes a call to Alice’s Phone. A notification of "trying" is sent to the controller. Alice’s phone automatically sends a "ringing" to Bob. Another notification of "early" is then sent to the controller. The controller tells the phone to deflect the call to Cathy. Alice’s phone sends a notification of "terminated" to the controller. Bob’s will attempt the call to Cathy.

<table>
<thead>
<tr>
<th>Controller</th>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>dialog1</td>
<td>F1</td>
<td>INVITE sip:Alice-AOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;&lt; Controller subscribed to &gt;&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;&lt;&lt; Alice’s dialog events &gt;&gt;&gt;</td>
</tr>
<tr>
<td>dialog2</td>
<td>F2</td>
<td>NOTIFY sip:Controller-GRUU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dialog-info+xml: dialog1=trying</td>
</tr>
<tr>
<td>dialog2</td>
<td>F3</td>
<td>200 (NOTIFY)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--------------------------------</td>
</tr>
<tr>
<td>dialog1</td>
<td>F4</td>
<td>180 (INVITE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--------------------------&gt;</td>
</tr>
<tr>
<td>dialog2</td>
<td>F5</td>
<td>NOTIFY sip:Controller-GRUU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dialog-info+xml: dialog1=early</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--------------------------------</td>
</tr>
<tr>
<td>dialog2</td>
<td>F6</td>
<td>200 (NOTIFY)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--------------------------------</td>
</tr>
<tr>
<td>dialog3</td>
<td>F7</td>
<td>REFER sip:Alice-GRUU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To: sip:Alice-GRUU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer-To: urn:feature:DeflectCall;target=(Cathy-AOR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Target-Dialog: dialog1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--------------------------------</td>
</tr>
<tr>
<td>dialog3</td>
<td>F8</td>
<td>202 (REFER)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--------------------------------</td>
</tr>
<tr>
<td>dialog1</td>
<td>F9</td>
<td>302 (INVITE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contact: sip:Cathy-AOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--------------------------------</td>
</tr>
<tr>
<td>dialog1</td>
<td>F10</td>
<td>ACK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--------------------------------</td>
</tr>
<tr>
<td>dialog2</td>
<td>F11</td>
<td>NOTIFY sip:Controller-GRUU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dialog-info+xml: dialog1=rejected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--------------------------------</td>
</tr>
<tr>
<td>dialog2</td>
<td>F12</td>
<td>200 (NOTIFY)</td>
</tr>
</tbody>
</table>
5.4. Hold Call

The controller tells Alice’s phone to put on hold the already established dialog with Bob. Alice’s phone sends a re-INVITE to Bob’s contact to put the media stream on hold. Note that a call hold is different concept than held media. In fact, a user can be placed on hold, and be provided with music on hold. A held call is a logical state which could be useful for a number of things such as monitoring the amount of time a user stays in a queue.
5.5. Retrieve Call

The controller tells Alice’s phone to retrieve an held call with Bob. Alice’s phone sends a re-INVITE to Bob’s contact to resume the media stream which was already on hold.
5.6. Single Step Transfer Call Flow Example

Alice’s phone and Bob are currently in an established dialog. The controller tells Alice’s phone to transfer the call to Cathy. Alice’s phone sends a REFER to Bob to transfer the call to Cathy. Cathy’s phone rings, is answered. Bob sends a notification to Alice’s phone of completion of REFER (using the implicit subscription). Alice’s phone then terminates the session with Bob and sends a notification of "terminated" to the controller.
dialog3
  F2  202 (REFER)
<-----------------------------

dialog4
  F3  REFER sip:Bob-GRUU
      Refer-To: (Cathy-AOR)
--------------------------->

dialog4
  F4  200 (REFER)
<-----------------------------

dialog4
  F5  NOTIFY sip:Alice-GRUU
      sipfrag: 100
<--------------------------->

dialog4
  F6  200 (NOTIFY)
<-----------------------------

dialog4
  Cathy

dialog5
    F7  INVITE sip:Cathy-AOR
    <-----------------------------

dialog5
    F8  180
    <-----------------------------

dialog5
    F9  200
    <-----------------------------

dialog5
    F10 ACK
    <-----------------------------

dialog4
    F11 NOTIFY sip:Alice-GRUU
        sipfrag: 200
    <-----------------------------

dialog4
    F12 200 (NOTIFY)
    <-----------------------------

dialog1
    F13 BYE
    <-----------------------------

dialog1
    F14 200 (BYE)
    <-----------------------------

dialog2
  F15 NOTIFY sip:Controller-GRUU
      dialog-info+xml: dialog1=terminated
    <-----------------------------

dialog2
  F16 200 (NOTIFY)
    <----------------------------->
5.7. Conference Calls

T.B.D.

5.8. Separate Calls

T.B.D.

6. Security Considerations

The functionality described in this document allows an authorized party to manipulate SIP sessions and dialogs in arbitrary ways. Any user agent that accepts these types of requests needs to be very careful in who it authorizes to send these types of requests. The same security considerations as [RFC3515] apply.

7. IANA Considerations

T.B.D. Need to register urn namespace according to procedures of [RFC3406].

8. Acknowledgments

Many thanks to Sean Olson, Orit Levin, Robert Sparks, Jonathan Rosenberg, and John Elwell.

9. References

9.1. Normative References


9.2. Informational References


Authors' Addresses

Francois Audet
Nortel
4655 Great America Parkway
Santa Clara, CA  95054
US

Phone: +1 408 495 2456
Email: audet@nortel.com

Alan Johnston
Avaya
St. Louis, MO  63124
US

Email: alan@sipstation.com

Rohan Mahy
Plantronics
345 Encinal Street
Santa Cruz, CA
US

Email: rohan@ekabal.com

Cullen Jennings
Cisco Systems
170 West Tasman Drive
Mailstop SJC-21/2
San Jose, CA  95134
US

Phone: +1 408 902-3341
Email: fluffy@cisco.com
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.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).