SIPPING Working Group
Internet-Draft
Expires: August 1, 2007

V. Hilt
Bell Labs/Alcatel-Lucent
G. Camarillo
Ericsson
January 28, 2007

A Session Initiation Protocol (SIP) Event Package for Session-Specific Session Policies. draft-ietf-sipping-policy-package-03

Status of this Memo

Copyright Notice

Abstract

This specification defines a Session Initiation Protocol (SIP) event package for session-specific policies. This event package enables user agents to subscribe to session policies for a SIP session and to receive notifications if these policies change.

Table of Contents

1. Introduction

The Framework for Session Initiation Protocol (SIP) [5] Session Policies [7] defines a protocol framework that enables a proxy to define and impact policies on sessions such as the codecs or media types to be used. It identifies two types of session policies: session-specific and session-independent policies. Session-specific policies are policies that are created for one particular session, based on the session description of this session. They enable a network intermediary to inspect the session description a UA is proposing and to return a policy specifically generated for this session description. For example, an intermediary could open pinholes in a firewall/NAT for each media stream in a session and return a policy that replaces the internal IP addresses and ports with external ones. Since session-specific policies are tailored to a session, they only apply to the session they are created for. A user agent requests session-specific policies on a session-by-session basis at the time a session is created and the session description is known. Session-independent policies on the other hand are policies that are created independent of a session and generally apply to all the SIP sessions set up by a user agent (see [7]).

The Framework for SIP Session Policies [7] defines a mechanism that enables UAs to discover the URIs of session-specific policy servers. This specification defines a SIP event package [4] that enables UAs to subscribe to session-specific policies on a policy server. Subscribing to session-specific policies involves the following steps (see [7]):

1. A user agent submits the details of the session it is trying to establish to the policy server and asks whether a session using these parameters is permissible. For example, a user agent might propose a session that contains the media types audio and video.
2. The policy server generates a policy decision for this session and returns the decision to the user agent. Possible policy decisions are (1) to deny the session, (2) to propose changes to the session parameters with which the session would be acceptable, or (3) to accept the session as it was proposed. An example for a policy decision is to disallow the use of video but agree to all other aspects of the proposed session.
3. The policy server can update the policy decision at a later time. A policy decision update can, for example, require additional changes to the session (e.g. because the available bandwidth has changed) or deny a previously accepted session (i.e. disallow the continuation of a session).

The event package for session-specific policies enables a user agent to subscribe to the policies for a SIP session following the above

model. The subscriber initiates a subscription by submitting the details of the session it is trying to establish to the notifier (i.e. the policy server) in the body of a SUBSCRIBE request. The notifier uses this information to determine the policy decision for this session. It conveys the initial policy decision to the subscriber in a NOTIFY and all changes to this decision in subsequent NOTIFYs.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [1] and indicate requirement levels for compliant implementations.

3. Event Package Formal Definition

This document provides the details for defining a SIP event package as required by RFC 3265 [4].

3.1. Event Package Name

The name of the event package defined in this specification is "session-spec-policy".

3.2. Event Package Parameters

This package defines the optional event package parameter "local-only". This parameter is only defined for NOTIFY requests and MUST be ignored if received in a SUBSCRIBE request. The usage of the "local-only" parameter is described in Section 3.3, Section 3.8 and Section 3.9.

3.3. SUBSCRIBE Bodies

A SUBSCRIBE for this event package MUST contain a body that describes a SIP session. The purpose of this body is to enable the notifier to generate the policies the subscriber is interested in. In this event package, the Request-URI, the event package name and event parameters are not sufficient to determine the resource a subscription is for. However, with the session description in the SUBSCRIBE body, the notifier can generate the requested policy decision and create policy events for this resource.

All subscribers and notifiers MUST support the MIME type

"application/session-policy+xml" as defined in the User Agent Profile Data Set for Media Policy [3]. The "application/session-policy+xml" format is the default format for SUBSCRIBE bodies in this event package. Subscribers and notifiers MAY negotiate the use of other formats capable of representing a session.

> Note: encoding the local and remote session description in the Media Policy [3] format has a number of advantages compared to the use of the SDP [9] format in SUBSCRIBE bodies: i) the Media Policy format is more flexible and allows the inclusion of information that can't be expressed in SDP (e.g. the target URI), ii) the Media Policy format enables the encoding of local and remote session descriptions in a single document (not requiring the use of MIME multipart and new content disposition types), and iii) it aligns the formats used for session-specific and session-independent policies. A drawback is that it requires the UA to encode SDP and session information in Media Policy documents.

3.4. Subscription Duration

A subscription to the session-specific policy package is usually established at the beginning of a session and terminated when the corresponding session ends. A typical duration of a phone call is a few minutes.

Since the duration of a subscription to the session-specific policy package is related to the lifetime of the corresponding session, the value for the duration of a subscription is largely irrelevant. However, it SHOULD be longer than the typical duration of a session. The default subscription duration for this event package is set to two hours.

A subscription MAY be terminated before a session ends by the notifier. For example, a notifier may terminate the subscription after the initial policy notification has been sent to the subscriber if it knows that these policies will not change during the session. A subscriber MUST NOT terminate a subscription unless it is terminating the session this subscription is for.

3.5. NOTIFY Bodies

In this event package, the body of a notification contains the session policy requested by the subscriber. All subscribers and notifiers MUST support the format "application/session-policy+xml" [3] as a format for NOTIFY bodies.

The SUBSCRIBE request MAY contain an Accept header field. If no such header field is present, it has a default value of "application/

session-policy+xml". If the header field is present, it MUST include "application/session-policy+xml", and MAY include any other MIME type capable of representing session-specific policies. As defined RFC 3265 [4], the body of notifications MUST be in one of the formats defined in the Accept header of the SUBSCRIBE request or in the default format.

If the notifier uses the same format in NOTIFY bodies that was used by the subscriber in the SUBSCRIBE body (e.g. "application/ session-policy+xml"), the notifier can expect that the subscriber supports all format extensions that were used in the SUBSCRIBE body. The notifier cannot assume that the subscriber supports other extensions beyond that and SHOULD NOT use such extensions.

If the SUBSCRIBE request contained a representation of the local session description and the subscription was accepted, then the NOTIFY body MUST contain a policy for the local session description. If the SUBSCRIBE request of an accepted subscription contained the local and the remote session description, then the NOTIFY body MUST contain two policies, one for the local and one for the remote session description.

3.6. Subscriber generation of SUBSCRIBE requests

The subscriber follows the general rules for generating SUBSCRIBE requests defined in [4]. The subscriber MUST provide sufficient information in the SUBSCRIBE body to fully describe the session for which it seeks to receive session-specific policies. It MUST use the most recent session description as a basis for this information.

If the "application/session-policy+xml" format is used in SUBSCRIBE bodies, the subscriber MUST provide a value for each field that is defined for session information documents [3] and for which the subscriber has information available. In other words, the subscriber MUST fill in the elements of a session information document as complete as possible. If the subscriber supports extensions of the "application/session-policy+xml" format, it MUST also provide a value for each field defined by this extension for session information documents, if possible. Providing as much information as possible avoids that a session is rejected due to a lack of session information and the negotiation of the information to be disclosed between notifier and subscriber.

Subscriptions to this event package are typically created in conjunction with an SDP offer/answer exchange [8] during the establishment of a session (see [7]). If used with an offer/answer exchange, the subscriber MUST insert the representation of the local session description in the SUBSCRIBE body. The local session

description is the one that was created by the subscriber (e.g. the offer if the subscriber has initiated the offer/answer exchange). Under certain circumstances, a UA may not have a session description when subscribing to session-specific policies, for example, when it is composing an empty INVITE request (i.e. an INVITE request that does not contain an offer). In these cases, a UA SHOULD establish a subscription without including a representation of the local session description. The UA MUST send another SUBSCRIBE request that contains this session description as soon as the session description becomes available, for example, because the UA has received a 200 OK to an empty INVITE request. A policy server can choose to admit a session only after the UA has disclosed the session descriptions.

The subscriber SHOULD also include a representation of the remote session description in the SUBSCRIBE body. The remote session description is the one the subscriber has received (i.e. the answer if the subscriber has initiated the offer/answer exchange). In some scenarios, the remote session description is not available to the subscriber at the time the subscription to session-specific policies is established. In this case, the initial SUBSCRIBE message SHOULD only contain a representation of the local session description. When the remote description becomes available, the subscriber SHOULD refresh the subscription by sending another SUBSCRIBE request, which then contains the local and the remote session description, unless the subscriber has received a NOTIFY with the "local-only" parameter. This parameter indicates that the notifier does not need to see the remote session description.

A user agent can change the session description of an ongoing session. A change in the session description will typically affect the policy decisions for this session. A subscriber MUST refresh the subscription to session-specific policies every time the session description of a session changes. It does this by sending a SUBSCRIBE request, which contains the details of the updated session descriptions.

A subscriber may receive a error that indicates a server failure in response to a SUBSCRIBE request. In this case, the subscriber SHOULD try to locate an alternative server, for example, using the procedures described in [6]. If no alternative server can be located, the subscriber MAY continue with the session for which it wanted to receive session-specific policies without subscribing to session-specific policies. This is to avoid that a failed policy server prevents a UA from setting up or continuing with a session. Since the sessions created by the UA may not be policy compliant without this subscription, they may be blocked by policy enforcement mechanisms if they are in place.

Session policies can contain sensitive information. Moreover, policy decisions can significantly impact the behavior of a user agent. A user agent should therefore verify the identity of a policy server and make sure that policies have not been altered in transit. All implementations of this package MUST support TLS [2] and the SIPS URI scheme. A subscriber SHOULD use SIPS URIs when subscribing to session-specific policies so that policies are transmitted over TLS. See Section 4.

3.7. Notifier processing of SUBSCRIBE requests

All subscriptions to session-specific policies SHOULD be authenticated and authorized before approval. However, a policy server may frequently encounter UAs it cannot authenticate. In these cases, the policy server MAY provide a generic policy that does not reveal sensitive information to these UAs. For details see Section 4.

The authorization policy is at the discretion of the administrator. In general, all users SHOULD be allowed to subscribe to the session-specific policies of their sessions. A subscription to this event package will typically be established by a device that needs to know about the policies for its sessions. However, subscriptions may also be established by applications (e.g. a conference server). In those cases, an authorization policy will typically be provided for these applications.

Responding in a timely manner to a SUBSCRIBE request is crucial for this event package. A notifier must minimize the time needed for processing SUBSCRIBE requests and generating the initial NOTIFY. This includes minimizing the time needed to generate an initial policy decision. A short response time is in particular important for this event package since it minimizes the delay for fetching policies during an INVITE transaction and therefore reduces call setup time. In addition, subscriptions to session-specific policies can be established while the subscriber is in an INVITE transaction at a point where it has received the 200 OK but before sending the ACK. Delaying the creation of the initial NOTIFY would delay the transmission of the ACK. A more detailed discussion of this scenario can be found in [7].

A subscriber may not have disclosed enough information in the SUBSCRIBE request to enable the notifier to generate a policy decision. For example, a UA may have subscribed to session-specific policies without including the representation of a session description. The policy server SHOULD accept such a subscription. However, it SHOULD generate a NOTIFY request that indicates that a policy decision could not be made due to insufficient information.

This can be expressed by either generating a NOTIFY request with an empty body or by inserting a corresponding policy decision document into the NOTIFY body.

3.8. Notifier generation of NOTIFY requests

A notifier sends a notification in response to SUBSCRIBE requests as defined in RFC 3265 [4]. In addition, a notifier MAY send a notification at any time during the subscription. Typically, it will send one every time the policy decision this subscription is for has changed. When and why a policy decision changes is entirely at the discretion of the administrator. A policy decision can change for many reasons. For example, a network may become congested due to an increase in traffic and reduces the bandwidth available to an individual user. Another example is a session that has been started during "business hours" and continues into "evening hours" where more bandwidth or video sessions are available to the user according to the service level agreement.

Policy decisions are expressed in the format negotiated for the NOTIFY body (e.g. "application/session-policy+xml"). The policy document in a NOTIFY body MUST represent a complete policy decision. Notifications that contain the deltas to previous policy decisions or partial policy decisions are not supported in this event package.

The notifier SHOULD terminate the subscription if the policy decision is to reject a session and if it can be expected that this decision will not change in the foreseeable future. The notifier SHOULD keep the subscription alive, if it rejects a session but expects that the session can be admitted at a later point in time. For example, if the session was rejected due to a temporary shortage of resources and the notifier expects that these resources will become available again shortly it should keep the subscription alive. A session is admitted by returning a policy decision document that requires some or no changes to the session.

If the notifier has not received enough information to make a policy decision from the subscriber (e.g. because it did not receive a session description), it SHOULD NOT terminate the subscription since it can be expected that the UA refreshes the subscription with a SUBSCRIBE request that contains more information. The notifier SHOULD generate a NOTIFY request with an empty body or with a body that contains a policy decision document indicating that the decision could not be made.

Some session-specific policies do not require the disclosure of the remote session description to the notifier. If a notifier determines that this is the case after receiving a SUBSCRIBE request, it SHOULD

include the "local-only" event parameter in NOTIFY requests.

3.9. Subscriber processing of NOTIFY requests

A subscriber MUST apply the policy decision received in a NOTIFY to the session associated with this subscription. If the UA decides not to apply the received policy decision, it MUST NOT set up the session and MUST terminate the session if it is already in progress. If the UA has a pending INVITE transaction for this session, it MUST cancel or reject the INVITE request.

If the subscriber receives a NOTIFY indicating that the session has been rejected, it MUST NOT attempt to establish this session. The notifier may still keep up the subscription after rejecting a session and may send an updated policy decision to the subscriber at a later time. This is useful, for example, if the session was rejected due to a temporary shortage of resources and the notifier expects that this problem to be resolved shortly. In this case, the subscriber SHOULD not terminate the subscription to session-specific policies. If the notifier has terminated the subscription after rejecting the session, the subscriber SHOULD NOT try to re-subscribe for the same session.

The subscriber may receive a NOTIFY which indicates that the SUBSCRIBER request did not contain enough information. The subscriber SHOULD re-subscribe with more complete information as soon as the missing information (e.g. the session description) is available.

A subscriber may receive an update to a policy decision for a session that is already established. The subscriber MUST apply the new policy decision to this session. If a UA decides that it does not want to apply the new policy decision, it MUST terminate the session. An updated policy decision may require the UA to generate a re-INVITE or UPDATE request in this session if the session description has changed or it may need to terminate this session. A policy update that requires a UA to terminate a session can, for example, be triggered by the users account running out of credit or the detection of an emergency that requires the termination of non-emergency calls.

If the subscriber receives a NOTIFY that contains the "local-only" event parameter, it SHOULD NOT include the remote session description in subsequent SUBSCRIBE requests within this subscription.

3.10. Handling of forked requests

This event package allows the creation of only one dialog as a result of an initial SUBSCRIBE request. The techniques to achieve this

behavior are described in [4].

3.11. Rate of notifications

It is anticipated that the rate of policy changes will be very low. In any case, notifications SHOULD NOT be generated at a rate of more than once every five seconds.

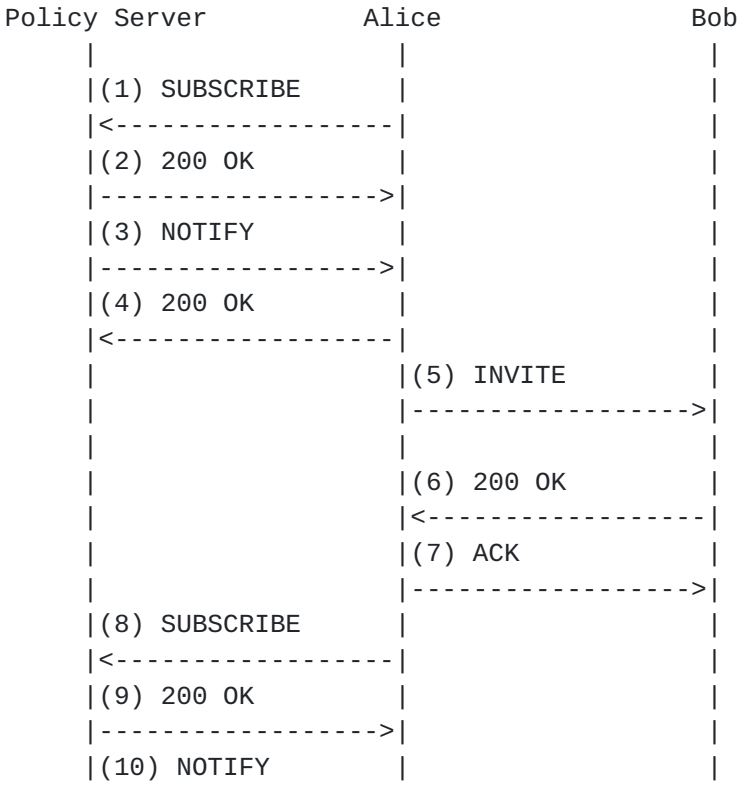
3.12. State Agents

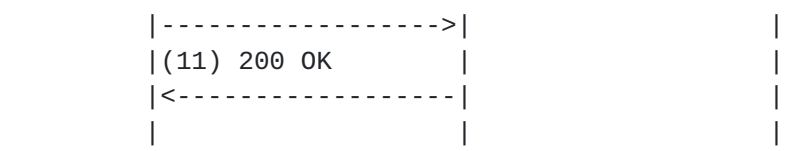
State agents play no role in this package.

3.13. Examples

The following message flow illustrates how a user agent (Alice's phone) can subscribe to session-specific policies when establishing a call (here to Bob's phone). The flow assumes that the user agent has already received the policy server URI (e.g. through configuration or as described in [7]) and it does not show messages for authentication on a transport or SIP level.

These call flow examples are informative and not normative. Implementers should consult the main text of this document for exact protocol details.





Message Details

(1) SUBSCRIBE Alice -> Policy Server

```
SUBSCRIBE sips:policy@biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS pc.biloxi.example.com:5061
 ;branch=z9hG4bK74bf
Max-Forwards: 70
From: Alice <sips:alice@biloxi.example.com>;tag=8675309
To: PS <sips:policy@biloxi.example.com>
Call-ID: rt4353gs2egg@pc.biloxi.example.com
CSeq: 1 SUBSCRIBE
Contact: <sips:alice@pc.biloxi.example.com>
Expires: 7200
Event: session-spec-policy
Accept: application/session-policy+xml
Content-Type: application/session-policy+xml
Content-Length: ...

[Local session description (offer)]
```

(2) 200 OK Policy Server -> Alice

(3) NOTIFY Policy Server -> Alice

```
NOTIFY sips:alice@pc.biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS srvr.biloxi.example.com:5061
 ;branch=z9hG4bK74br
Max-Forwards: 70
From: PS <sips:policy@biloxi.example.com>;tag=31451098
To: Alice <sips:alice@biloxi.example.com>;tag=8675309
Call-ID: rt4353gs2egg@pc.biloxi.example.com
CSeq: 1 NOTIFY
Event: session-spec-policy
Subscription-State: active;expires=7200
Content-Type: application/session-policy+xml
Content-Length: ...

[Policy for local session description (offer)]
```

(4) 200 OK Alice -> Policy Server

(5) INVITE Alice -> Bob

(6) 200 OK Bob -> Alice

(7) ACK Alice -> Bob

(8) SUBSCRIBE Alice -> Policy Server

```
SUBSCRIBE sips:policy@biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS pc.biloxi.example.com:5061
 ;branch=z9hG4bKna998sl
Max-Forwards: 70
From: Alice <sips:alice@biloxi.example.com>;tag=8675309
To: PS <sips:policy@biloxi.example.com>;tag=31451098
Call-ID: rt4353gs2egg@pc.biloxi.example.com
CSeq: 2 SUBSCRIBE
Expires: 7200
Event: session-spec-policy
Accept: application/session-policy+xml
Content-Type: application/session-policy+xml
Content-Length: ...

[Local session description (offer)]
[Remote session description (answer)]
```

(9) 200 OK Policy Server -> Alice

(10) NOTIFY Policy Server -> Alice

```
NOTIFY sips:alice@pc.biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS srvr.biloxi.example.com:5061
 ;branch=z9hG4bKna998sk
Max-Forwards: 70
From: PS <sips:policy@biloxi.example.com>;tag=31451098
To: Alice <sips:alice@biloxi.example.com>;tag=8675309
Call-ID: rt4353gs2egg@pc.biloxi.example.com
CSeq: 2 NOTIFY
Event: session-spec-policy
Subscription-State: active;expires=7200
Content-Type: application/session-policy+xml
Content-Length: ...

[Policy for local session description (offer)]
[Policy for remote session description (answer)]
```

F6 200 OK Alice -> Policy Server

4. Security Considerations

Session policies can significantly change the behavior of a user agent and can therefore be used by an attacker to compromise a user agent. For example, session policies can be used to prevent a user agent from successfully establishing a session (e.g. by setting the available bandwidth to zero). Such a policy can be submitted to the user agent during a session, which may cause the UA to terminate the session.

A user agent transmits session information to a policy server. This information may contain sensitive data the user may not want an eavesdropper or an unauthorized policy server to see. For example, the session information may contain the encryption keys for media streams. Vice versa, session policies may also contain sensitive information about the network or service level agreements the service provider may not want to disclose to an eavesdropper or an unauthorized user agent.

It is therefore important to secure the communication between the user agent and the policy server. The following three discrete attributes need to be protected:

1. authentication of the policy server and, if needed, the user agent,
2. confidentiality of the messages exchanged between the user agent and the policy server and
3. ensuring that private information is not exchanged between the two parties, even over an confidentiality-assured and authenticated session.

The confidentiality of the messages exchanged between the two parties can be protected by encrypting the data stream through a TLS session using the cipher suites specified in [5].

Accordingly, policy servers SHOULD be addressable only through a SIPS URI and it MUST support TLS. In order to send a subscription to the policy server, the user agent MUST support TLS, although it does not need to possess a certificate. In such a case, the policy server SHOULD authenticate the UA using HTTP Digest. The confidentiality of the communication between the policy server and the user agent will be assured as long as the policy server supports TLS and is reached through a SIPS URI.

Authenticating the two parties can performed using X.509 certificates exchanged through TLS and other techniques such as HTTP Digest. When the user agent establishes a TLS session with the policy server, the policy server will present it a X.509 certificate. The user agent SHOULD ensure that the identity of the policy server encoded in the certificate matches the URI that the user received when it used the Framework for SIP Session Policies [7] to retrieve a URI of a session-specific policy server.

When a policy server receives a new subscription (as opposed to a refresh subscription), it SHOULD try to authenticate the user agent using any means at its disposal. If the user agent has a TLS certificate, the identity of the user agent SHOULD be contained in the certificate, or if the user agent does not possess a certificate, the policy server SHOULD challenge the user agent using HTTP Digest. A policy server may frequently encounter UAs it cannot authenticate. In these cases, the policy server MAY provide a generic policy that does not reveal sensitive information to these UAs.

And finally, the fact that the user agent and the policy server have successfully authenticated each other and have established a secure TLS session does not absolve either one from ensuring that they do not communicate sensitive information. For example, a session description may contain sensitive information -- session keys, for example -- that the user agent may not want to share with the policy server; and indeed, the policy server does not need such information to effectively formulate a policy. Thus, the user agent should not insert such sensitive information in a session information document that it sends to the policy server. Likewise, the policy server may have information that is sensitive and of no use to the user agent -- network service level agreements, or network statistics, for example. Thus, the policy server should refrain from transmitting such information to the user agent.

5. IANA Considerations

5.1. Event Package Name

This specification registers an event package, based on the registration procedures defined in RFC 3265 [2]. The following is the information required for such a registration:

Package Name: session-spec-policy

Package or Template-Package: This is a package.

Published Document: RFC XXXX (Note to RFC Editor: Please fill in XXXX

with the RFC number of this specification).

Person to Contact: Volker Hilt, volkerh@bell-labs.com.

Appendix A. Acknowledgements

Many thanks to Jonathan Rosenberg for the many discussions and suggestions for this draft, to Roni Even, Bob Penfield, Mary Barnes and Shida Schubert for reviewing the draft and providing feedback and to Vijay Gurbani for the comments and input to the security considerations section.

6. References

6.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[3] Hilt, V., Camarillo, G., and J. Rosenberg, "A User Agent Profile Data Set for Media Policy", draft-ietf-sipping-media-policy-dataset-02 (work in progress), October 2006.

[4] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[5] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[6] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

[7] Hilt, V., Camarillo, G., and J. Rosenberg, "A Framework for Session Initiation Protocol (SIP) Session Policies", draft-ietf-sip-session-policy-framework-00 (work in progress), October 2006.

6.2. Informative References

[8] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[9] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

Authors' Addresses

Volker Hilt
Bell Labs/Alcatel-Lucent
101 Crawfords Corner Rd
Holmdel, NJ 07733
USA

Email: volkerh@bell-labs.com

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Gonzalo.Camarillo@ericsson.com

Full Copyright Statement

Intellectual Property

Acknowledgment

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