A Framework for Session-Specific Session Policies in the Session Initiation Protocol (SIP)
draft-hilt-sipping-session-spec-policy-01

Status of this Memo

This document is an Internet-Draft and is subject to all provisions of section 3 of RFC 3667. 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 become aware will be disclosed, in accordance with RFC 3668.

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, 2005.

Copyright Notice

Copyright (C) The Internet Society (2004).

Abstract

This specification defines a framework for session-specific policies in Session Initiation Protocol (SIP) sessions. It enables intermediaries to request the use of policies in a SIP session and
defines the transport mechanisms needed for creating and delivering policies on a per-session basis.

Table of Contents

1. Introduction ................................................. 3
2. Terminology ................................................. 4
3. Architecture ................................................. 4
4. Overview of Operation ....................................... 6
   4.1 Offer in Request ........................................ 6
   4.2 Offer in Response ........................................ 7
5. Distributing Policy Server URIs ............................ 9
   5.1 UAC Behavior ............................................ 9
   5.2 UAS Behavior ............................................ 11
   5.3 Proxy Behavior .......................................... 12
   5.4 Header Definition and Syntax .......................... 12
6. Policy Channel ................................................ 14
7. Updating Policies ............................................. 14
8. Security Considerations ..................................... 14
9. IANA Considerations ......................................... 14
   Authors’ Addresses ......................................... 15
10. References .................................................... 14
A. Acknowledgements ........................................... 16
   Intellectual Property and Copyright Statements ........ 17
1. Introduction

Some domains have policies in place, which impact the sessions established using the Session Initiation Protocol (SIP). These policies are often needed to support the network infrastructure or the execution of services. For example, wireless networks usually have limited resources for media traffic. During periods of high activity, a wireless network provider may want to restrict codec usage on the network to lower rate codecs.

In another example, a SIP user agent is using a network which connects to the public Internet through a firewall at the network edge. The provider would like to tell the user agent to direct the media streams to the appropriate open ip/ports on that firewall. Knowing this policy enables these user agents to setup sessions with user agents on the public Internet.

In a third example, a domain A has a limited bandwidth pipe to another domain B. The pipe is realized through two routers that are managed. Domain A would like to direct each call to one of the routers based on their capacity. With session policies, the domain can inform the user agent about the route with the most capacity available.

Domains sometimes enforce policies they have in place. For example, a domain might have a configuration in which all packets containing a certain audio codec are dropped. Unfortunately, enforcement mechanisms usually do not inform the user about the policies they are enforcing and silently keep the user from doing anything against them. This may lead to the malfunctioning of devices that is incomprehensible to the user. With session policies, the user knows about the restricted codecs and can use a different codec or simply connect to a domain with less stringent policies. Session policies provide an important combination of consent coupled with enforcement. That is, the user becomes aware of the policy and needs to act on it, but the provider still retains the right to enforce the policy.

This specification defines a framework for delivering session-specific policies to user agents. Session-specific policies are conveyed to user agents on a per session basis. They are created with knowledge about the details of a session that is being established and apply only to this session. This framework defines mechanisms to announce the presence of session-specific policy servers to user agents. It defines a protocol for user agents to submit session details to a policy server in a secure way and to deliver policies for this session in response.

Session policies that are independent of a certain session and
generally apply to the sessions of a user agent can be delivered using the framework for session-independent policies [4]. The current situation in imposing session policies and the drawbacks of these practices as well as the requirements for a session policy solution are discussed in [5].

Note: The difference between session-independent and session-specific policies needs to be discussed here.

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. Architecture

![Figure 1](attachment:image.png)

This framework enables a network domain to set up policies for specific SIP sessions. The following entities are involved in setting up session-specific policies (see Figure 1): a user agent, a proxy, a policy server and possibly a router acting as policy enforcement point. The proxy’s role is to convey the URI of the policy server in its domain to user agents. The proxy does not provide the actual policies. It merely ensures that the user agents know where to retrieve policies from.
A policy server is a separate logical entity that may be physically
co-located with the proxy. Each domain has one policy server. Its task is to deliver session policies to user agents. The policy server and the user agent communicate through an end-to-end channel.

A policy-enabled user agent receives the URI of a policy server from the proxy. The user agent contacts this policy server and retrieves session policies. The user agent may provide information about the current session to the policy server and may receive session policies in response. It is also possible for the policy server to update the policies asynchronously during the course of a session and to receive a notification when the session is terminated.

A network may have routers in place, which enforce session policies acting as policy enforcement points. This specification assumes that enforcement points, if they are present, have knowledge about the current policies and their enforcement. The goal of this framework is to support enforcement points by informing user agents about the current policies helping them to avoid policy violations.

Note: The reasoning for the separate channel model needs to be discussed here.

The communication between a user agent and a policy server to set up session-specific policies involves two main operations:

1. The UA disclose information about the current session and the offer/answer exchange to a policy server.
2. The policy server sends instructions to the UA.

Some types of policies do not involve sending instructions, but only information disclosure. Still, a general session-specific policy mechanism needs to support both operations.

The same way, some policy servers only need to inspect the offer, but not the answer. Nevertheless, a general mechanism needs to consider policy servers which need to inspect both.

OPEN ISSUE: Is the inspection of the answer needed or can it be avoided (on a case-by-case basis or even in general) for simplification purposes.

Finally, some policy servers need to asynchronously update session policies or need to receive a notification when a session is terminated. For example, a policy server managing firewalls may want to free resources once a session is terminated. A general mechanism needs to enable these interactions between policy server and user agent.
4. Overview of Operation

The section provides example call flows to illustrate the establishment of session policies. It does not contain a normative protocol definition.

In this scenario, there are two domains (domain A and B), which both have session-specific policies for the UAs in their domain. Both do not provide session policies to the UAs outside of their domain. The two domains have a proxy (P A and P B) and a policy server (PS A and PS B). The two policy servers have policies for the session description offer and answer.

4.1 Offer in Request

The first call flow depicts an INVITE transaction with the offer in the request. It is assumed that UAC does not have previous knowledge about the policy server in its domain.

UA A creates an INVITE that is routed to proxy P A. P A knows that policies apply to this session. It rejects the INVITE and delivers the URI of PS A to UA A. UA A contacts PS A, discloses information about the session description offer to PS A and receives policies for the offer. P A does not reject the second INVITE since the UAC has now included a token, which it has received from PS A along with the policies, in the Policy-Id header. P A removes the Policy-Id header when forwarding the INVITE. P B provides the URI of PS B to UA B in the Policy-Contact header. UA B contacts PS B, discloses information about the offer and the answer it is about to return to the UA A. It receives policies for the offer and answer. It applies these policies and returns the answer in the 200 OK. Finally, UA A contacts PS A with the answer and retrieves policies from PS A.
4.2 Offer in Response

This call flow depicts an INVITE transaction with the offer in the response.

The steps in this call flow are similar to the steps in the previous flow. An important difference is that UA A now contacts PS A after receiving an offer in the 200 OK and before returning the answer in the ACK. This approach is discussed in more detail in the sections below.
UA A    P A    P B    UA B

INVITE offer
------------->
488
+ Policy-Contact
<-------------
ACK
------------->
PS A
PolicyChannel
-------------->
PolicyChannel
<-------------

INVITE
+ Policy-Id
------------->
INVITE
+ Policy-Contact
<-------------
PS B
PolicyChannel
+ InfoOffer
<-------------
PolicyChannel
+ PolicyOffer
------------->

OK offer
<-------------
OK offer
<-------------
OK offer
<-------------
PolicyChannel
+ InfoOffer
+ InfoAnswer
------------->
PolicyChannel
+ PolicyOffer
+ PolicyAnswer
<-------------
ACK answer
------------->
5. Distributing Policy Server URIs

The first step in setting up policies for a session is to convey the URIs of policy servers to the UAs, which may have policies for the current session. The policy server URIs need to be conveyed to the UAs in the initial INVITE that sets up the session as well as in re-INVITEs and UPDATEs during the session.

5.1 UAC Behavior

When a UA compliant to this specification generates an INVITE or UPDATE request, it MUST include the Supported header field with the option tag "policy" in the request.

A UA that supports session-specific policies SHOULD cache the URI of the policy server in the local domain. It receives this URI from the local proxy after sending an INVITE or UPDATE request. The UAC SHOULD use the cached policy server URI to retrieve session policies before sending subsequent INVITE or UPDATE requests. Contacting the local policy server before sending a request avoids the retransmission of the policy server URI for every new request.

The UA SHOULD NOT cache policy server URIs that are outside of the local domain since these servers may not be relevant for new sessions a UA sets up. A new session might traverse different networks and have a different destination. Some policies require that the request is routed in part to a proxy before policies can be provided by a policy server. A proxy marks the URI of such policy servers as non-cacheable when providing them to the UA. The UA SHOULD NOT cache a non-cacheable policy server URI and SHOULD remove the current URI from the cache when it receives one. In addition, a UA SHOULD NOT cache policy server URIs it receives when acting as UAS.

If the UAC generates a re-INVITE or UPDATE request within an existing session, it SHOULD contact all policy server URIs it has contacted when generating the previous INVITE or UPDATE request in this session.
before sending the new request.

The UAC receives a token from each policy server it has successfully contacted when generating a INVITE or UPDATE request. The UAC must include this token into the request. This enables a proxy to determine whether the UAC has already contacted the local policy server for this request or whether the policy server URI still needs to be conveyed to the UAC. The UAC MUST include all tokens it has received when generating a request into the Policy-Id header field. If it has not received a token, for example, because the policy server was unreachable, the UAC SHOULD use the policy server URI as the token. If no policy server has been contacted for a request, the UAC SHOULD NOT include a Policy-Id field. The syntax of the Policy-Id header is defined in Section 5.4.

The creation of policy server tokens is out of scope for this specification. The token needs to enable a proxy to verify that the policy server in its domain is known to the UAC and has successfully been contacted.

The UAC may receive a 488 response with a Policy-Contact header field. The Policy-Contact header contains a policy server URI. The UAC SHOULD use this URI to retrieve session policies for the current request. It should re-try the request after applying the received policies. The UAC MUST add the token it receives from this policy server to the Policy-Id header field. The UAC should cache the policy server URI if the policy server is in the same domain as the UAC and the Policy-Contact header does not contain the header field parameter "non-cacheable". The syntax of the Policy-Contact header is defined in Section 5.4.

If the UAC has received multiple 488 responses for the same request, it MUST use the policy server URIs in the order in which the 488 responses were received. The UAC MUST contact the first policy server and apply the received policies before contacting the next server. When it discloses session information to a policy server, it MUST use the most recent session information that considers the policies received previously. This ensures that policy servers are contacted in the order in which the request travels through the proxies. The order in which policy servers are contacted and session policies are applied to a session is significant for the final result of the policy process. For example, a policy for NAT traversal requires that the policy server can examine the IP address on which the media stream will be sent/received in the previous domain.

If required by the policy, the UAC should contact the policy servers again after receiving a 2xx response. In this case, the UAC MUST contact the same policy servers it has contacted for the successful
If the 2xx to an INVITE request contains a session description answer (i.e. the UAC did include an offer in the request), the UAC MUST send the ACK before it contacts the policy servers. If however the 2xx to an INVITE request contains an offer and the UAC must respond with an answer, the UAC MUST first contact the policy servers before sending the ACK. A policy server must respond immediately to requests on the policy channel (see Section 6). The UAC uses these policies to formulate an answer and ships this answer in the ACK. This is analogous to call flow I defined in [7].

OPEN ISSUE: Retrieving policies from the policy server before sending the ACK may delay the transmission of the ACK and trigger retransmissions of the 2xx by the UAS. An alternative approach is to generate a preliminary answer without contacting a policy server and return this answer in the ACK. The UAC then contacts the policy servers after the ACK has been sent. If the retrieved policies change the preliminary answer, the UAC must generate a new UPDATE request which contains an offer that considers these policies. This offer effectively replaces the preliminary answer sent in the ACK.

The advantage of contacting the policy server after sending the ACK is that it avoids 2xx retransmissions caused by policy servers not responding in time. A disadvantage is that the preliminary answer might trigger the use of policy enforcement mechanisms since it may not be compliant with policies.

The advantage of contacting the policy server before sending the ACK is simplicity. There is no overhead for updating the session description answer in policed sessions. A drawback is that retransmissions of 2xx may occur in cases where the UAC needs to contact a number of policy servers or policy servers respond poorly. However, such configurations are problematic in general and should be avoided. If they occur, the number of 2xx retransmissions and the overhead associated with them is however usually very small.

A policy may require the UAC to contact the policy server when a session is terminated. In this case, the UAC MUST contact the same policy server URIs it has used for the last successful request within this session, when the session is terminated.

5.2 UAS Behavior

If an incoming INVITE or UPDATE request includes a Policy-Contact header containing a list of policy server URIs, the UAS SHOULD use
these URIs to retrieve session policies. The UAS MUST use the policy server URIs in the order they were contained in the Policy-Contact header, starting with the topmost value.

If the UAS receives an ACK with a session description answer, it may need to contact the policy servers again. In this case, it MUST contact the same policy servers it has contacted after receiving the corresponding INVITE request.

A policy may require the UAS to contact the policy server when a session is terminated. In this case, the UAS MUST contact the same policy server URIs it has used for the last successful request within this session, when the session is terminated.

5.3 Proxy Behavior

A proxy may provide the contact URI of the local policy server to the UAC or the UAS when processing a INVITE or UPDATE request.

If a INVITE or UPDATE request contains a Supported header field with the option tag "policy", the proxy MAY reject the request with a 488 response to provide the policy server URI to the UAC. If the request contains a Policy-Id header field, the proxy MUST determine if it contains a token identifying the local policy server. If such a token is not present or if the Policy-Id header field is not present, the proxy MAY reject the request with a 488. The proxy MUST include a Policy-Contact header in the 488 with the URI of the local policy server as the value. Some policies require that the request is routed in part to a proxy before policies can be provided to the UAC. If this is the case, the proxy SHOULD add the header field parameter "non-cacheable" to the Policy-Contact header. This will prevent the UAC from contacting the policy server before the request is sent.

A token from the local policy server in the Policy-Id header indicates that the UAC has already contacted the policy server for this request. In this case, the proxy SHOULD remove the token from the Policy-Id header before forwarding the request.

The proxy MAY insert a Policy-Contact header field into a INVITE or UPDATE request in order to provide the policy server URI to the UAS. If the request already contains a Policy-Contact header field, the proxy MUST insert the URI ahead of all existing values at the beginning of the list. A proxy MUST NOT change the order of existing Policy-Contact header values.

5.4 Header Definition and Syntax
The Policy-Id header field is inserted into a INVITE or UPDATE
request by the UAC. It identifies all policy servers the UAC has contacted for the request. A Policy-Id header value can either be a token that identifies a policy server to the proxy in the same domain or the URI of a policy server.

The syntax of the Policy-Id header field is:

\[
\text{Policy-Id} = "\text{Policy-Id}\" \text{ HCOLON pol-id-params} \\
\text{pol-id-params} = \text{token / absoluteURI}
\]

The Policy-Contact header field can be inserted into INVITE and UPDATE requests by a proxy. It contains an ordered list of policy server URIs that need to be contacted by the UAS. The UAS starts to process the header field at the topmost value of this list. New header field values are inserted at the top. The Policy-Contact header field effectively forms a stack. The "non-cacheable" header field parameter MUST NOT be inserted into a request.

The Policy-Contact header field can also be inserted into a 488 response to an INVITE or UPDATE request by a proxy. It contains a policy server URI that needs to be contacted by the UAC. A proxy MAY add the "non-cacheable" header field parameter to indicate that the UAC should not cache the policy server URI.

The syntax of the Policy-Contact header field is:

\[
\text{Policy-Contact} = "\text{Policy-Contact}\" \text{ HCOLON policyURI} \\
\text{policyURI} = \text{absoluteURI [ SEMI "non-cacheable"]}
\]

The BNF for absoluteURI is defined in [8].

Table 1 is an extension of Tables 2 and 3 in [8]. The column 'UPD' is for the UPDATE method [6].

<table>
<thead>
<tr>
<th>Header field</th>
<th>where</th>
<th>proxy</th>
<th>ACK</th>
<th>BYE</th>
<th>CAN</th>
<th>INV</th>
<th>OPT</th>
<th>REG</th>
<th>UPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy-Id</td>
<td>R</td>
<td>rd</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>o</td>
<td>-</td>
<td>-</td>
<td>o</td>
</tr>
<tr>
<td>Policy-Contact</td>
<td>R</td>
<td>a</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>o</td>
<td>-</td>
<td>-</td>
<td>o</td>
</tr>
<tr>
<td>Policy-Contact</td>
<td>488</td>
<td>a</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>o</td>
<td>-</td>
<td>-</td>
<td>o</td>
</tr>
</tbody>
</table>

Table 1: Policy-Id and Policy-Contact Header Fields
6. Policy Channel

A UA and policy server compliant with this specification MUST use the mechanism defined in this section for the policy channel.

[TBD.]

OPEN ISSUE: The policy channel mechanism needs to be defined. One option is the use of a SUBSCRIBE/NOTIFY based mechanism. It would allow policy servers to deliver the initial policies and asynchronously update those policies if needed. This would however require the use of SDP announcements as filter in the SUBSCRIBE request, which is a debatable approach.

7. Updating Policies

A UA may receive policy updates through a policy channel. The UA SHOULD apply these policies to the current session. It MUST generate a re-INVITE or UPDATE request if the updated policies modify aspects of the session that need to be communicated to the peer UA.

8. Security Considerations

In particular authentication and authorization are critical issues that need to be addressed here.

[TBD.]

9. IANA Considerations

[TBD.]

10 References


Authors’ Addresses

Volker Hilt
Bell Labs/Lucent Technologies
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

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, NJ 07054
USA

EMail: jdrosen@dynamicsoft.com
Appendix A. Acknowledgements

Many thanks to Allison Mankin for the discussions and the suggestions for this draft.
Intellectual Property Statement

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.

Disclaimer of Validity

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 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.

Copyright Statement

Copyright (C) The Internet Society (2004). 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.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.