SIPPING Working Group
Internet-Draft
Expires: December 25, 2006

V. Hilt
Bell Labs/Lucent Technologies
G. Camarillo
Ericsson
J. Rosenberg
Cisco Systems

June 23, 2006

A Framework for Session Initiation Protocol (SIP) Session Policies
draft-ietf-sipping-session-policy-framework-01

Abstract

Proxy servers play a central role as an intermediary in the Session Initiation Protocol (SIP) as they define and impact policies on call routing, rendezvous, and other call features. However, there is currently no standard mechanism by which a proxy can define or influence policies on sessions such as the codecs or media types to

be used. This document specifies a framework for SIP session policies that provides this capability to proxies. It defines a model, an overall architecture and the protocol components for session policies.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [6] is a signaling protocol for creating, modifying and terminating multimedia sessions. A central element in SIP is the proxy server. Proxy servers are intermediaries that are responsible for request routing, rendezvous, authentication and authorization, mobility, and other signaling services. However, proxies are divorced from the actual sessions - audio, video, and messaging - that SIP establishes. Details of the sessions are carried in the payload of SIP messages, and are usually described with the Session Description Protocol (SDP) [7]. Indeed, SIP provides end-to-end encryption features using S/MIME, so that all information about the sessions can be hidden from eavesdroppers and proxies alike.

However, experience has shown that there is a need for SIP intermediaries to impact aspects of a session. For example, SIP may be used in a wireless network, which has limited resources for media traffic. During periods of high activity, the wireless network provider wants to restrict the amount of bandwidth available to each individual user. With session policies, an intermediary in the wireless network can inform the user agent about the bandwidth it can currently count on. This information enables the user agent to make an informed decision about the number of streams, the media types, and the codecs it can successfully use in a session. Similarly, a network provider may have a service level agreement with a user that defines the set of media types a user can use. With session policies, the network can convey the current set of policies to user agents, enabling them to set up sessions without inadvertently violating any of the network policies.

In another example, a SIP user agent is using a network which is connected to the public Internet through a firewall or a network border device. The network provider would like to tell the user agent that it needs to send its media streams to a specific IP address and port on the firewall or border device to reach the public Internet. Knowing this policy enables the user agent to set up sessions across the firewall or the network border. In contrast to other methods for inserting a media intermediary, the use of session policies does not require the inspection or modification of SIP message bodies.

Domains often enforce the session policies they have in place. For example, a domain might have a policy that disallows the use of video and may enforce this policy by dropping all packets that contain a video encoding. Unfortunately, enforcement mechanisms usually do not inform the user about the policies they are enforcing. Instead, they silently keep the user from doing anything against them. This may

lead to a malfunctioning of devices that is incomprehensible to the user. With session policies, the user knows about the current network policies and can set up policy-compliant sessions or simply connect to a domain with less stringent policies. Thus, 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.

Two types of session policies exist: session-specific policies 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 examine the session description a UA is proposing and to return a policy specifically 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. Session-specific policies are created on a session-by-session basis at the time the session is established.

Session-independent policies on the other hand are policies that are created independent of a session and generally apply to the SIP sessions set up by a user agent. A session-independent policy can, for example, be used to inform user agents about an existing bandwidth limit or media type restrictions. Since these policies are not based on a specific session description, they can be created independent of an attempt to set up a session and only need to be conveyed to the user agent once (e.g. at the time the device is powered on).

This specification defines a framework for SIP session policies. It specifies a model, the overall architecture, and the protocol components that are needed for session-independent and session-specific policies.

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. Session-Independent Policies

Session-independent policies are policies that are created independent of a session and generally apply to the sessions a user agent is setting up. They typically remain stable for a longer period of time and apply to the sessions set up while they are valid. However, session-independent policies may also change over time. For example, a policy that defines a bandwidth limit for a user may change during the day, defining a lower limit during peak hours and allow more bandwidth off-peak.

3.1. Architecture and Overview

```
                    +-------------+
             /------|   policy    |
  +----+    /       |  server 1   |
  |    |---/        +-------------+
  | UA |                 ...
  |    |---\        +-------------+
  +----+    \       |   policy    |
             \------|  server n   |
                    +-------------+
```

Figure 1

A SIP UA may receive session-independent policies from one or more policy servers. In a typical configuration, a UA receives session-independent policies from a policy server in the access or local network domain (i.e. the domain from which the UA receives IP service) and possibly the home network domain (i.e. the domain the UA registers at). The local network may, for example, have policies that support the access network infrastructure (e.g. in a wireless network where bandwidth is scarce, a provider may restrict the bandwidth available to an individual user). The home network may have policies that are needed to support services or policies that reflect the service level agreement with the user. Thus, in most cases, a UA will receive session-independent policies from one or at most two policy servers.

Setting up session-independent policies involves the following steps:

1. A user agent requests session-independent policies from the policy servers in the local network and home domain. A user agent typically requests these policies when it starts up or connects to a new network domain.

2. The policy server selects the policies that apply to this user agent. The policy server may have general policies that apply to all users or maintain separate policies for each individual user. The selected policies are returned to the user agent.
3. The policy server may update the policies, for example, when network conditions change.

3.2. Policy Subscription

A UA requests session-independent policies by subscribing to the policy server in a domain. Subscriptions to session-independent policies are established using the "ua-profile" event package defined in the Framework for SIP User Agent Profile Delivery [4].

The "ua-profile" event package [4] provides a mechanism to discover policy servers in the local network and the home domain. The "local-network" profile-type enables a UA to discover a policy server in the local domain; the "user" profile type in the home domain. A UA compliant to this specification SHOULD attempt to discover and subscribe to the policy servers in these two domains.

A UA SHOULD (re-)subscribe to session-independent policies when the following events occur:

- The UA registers a new AoR or removes a AoR from the set of AoRs it has registered. In these cases, the UA SHOULD establish subscriptions for each new AoR using the "user" and the "local-network" profile-types. The UA SHOULD terminate all subscriptions for AoRs it has removed.
- The UA changes the domain it is connected to. The UA SHOULD terminate all existing subscriptions for the "local-network" profile-type. It SHOULD then create a new subscription for each AoR using the "local-network" profile-type. This way, the UA stops receiving policies from the previous local domain and starts to receive the policies of the new local domain. The UA does not need to change the subscriptions for "user" profiles.

If a subscriber is unable to establish a subscription, it SHOULD NOT attempt to re-try this subscription, unless one of the above events occurs again. This is to limit the number of SUBSCRIBE requests sent within domains that do not support session-independent policies.

A UA compliant to this specification MUST support the User Agent Profile Data Set for Media Policy [3] and the Schema for SIP User Agent Profile Data Sets [8]. To indicate that the UA wants to receive session-independent policies, it includes the MIME type "application/session-policy+xml" in addition to the MIME type of the Schema for SIP User Agent Profile Data Sets in the Accept header of a

SUBSCRIBE request.

A policy server MAY send a notification to the subscriber every time the session-independent policies covered by the subscription changes. The definition of what causes a policy to change is at the discretion of the administrator. A change in the policy may be triggered, for example, by a change in the network status, by the change in the time of day or by an update of the service level agreement with the customer. The session-independent policies contained in a notification MUST represent a complete session-independent policy. Deltas to previous policies or partial policies are not supported.

4. Session-Specific Policies

Session-specific policies are policies that are created specifically for one particular session of a UA. Thus, session-specific policies will typically be different for different sessions. The session-specific policies for a session may change during the course of the session. For example, a user may run out of credit during a session, which will cause the network to disallow the transmission all media streams from this point on.

4.1. Architecture

```
                          domain 1
                       +-----------+
                /------|   proxy   |----...
   +----+    /         +-----------+
   |    |---/          +-----------+
   |    |              |  policy   |
   | UA |============|  server   |
   |    |              +-----------+
   |    |****          +-----------+
   +----+    *         |  policy   |
                *******|enforcement|****...
                       +-----------+

   --- SIP Signaling
   === Policy Channel
   *** Media
```

Figure 2

The following entities are needed for session-specific policies (see Figure 2): a user agent (UA), a proxy, a policy server and possibly a policy enforcement entity.

The role of the proxy is to provide a rendezvous mechanism for UAs and policy servers. It conveys the URI of the policy server in its domain to UAs and ensures that each UA knows where to retrieve policies from. It does not deliver the actual policies to UAs.

The policy server is a separate logical entity that may be physically co-located with the proxy. The role of the policy server is to deliver session policies to UAs. The policy server receives session information, uses this information to determine the policies that apply to the session and returns these policies to the UA. The mechanism for generating policies (i.e. making policy decisions) is outside the scope of this specification. A policy server may, for example, query an external entity to get the policies that apply to a session or it may directly incorporate a policy decision point and generate policies locally.

A UA receives the URI of a policy server from a proxy. It uses this URI to connect to the policy server. It provides information about the current session to the policy server and receives session policies in response. The UA may also receive policy updates from the policy server during the course of a session.

A network may have a policy enforcement infrastructure in place. However, this specification does not make any assumptions about the enforcement of session policies and the mechanisms defined here are orthogonal a policy enforcement infrastructure. Their goal is to provide a mechanism to convey session information to a policy server and to return the policies that apply to a session to the UA.

In principle, each domain that is traversed by SIP signaling messages can define session-specific policies for a session. Each of these domains needs to run a policy server and a proxy that is able to rendezvous a UA with the policy server (as shown in Figure 2). However, it is expected that session-specific policies will often only be provided by the local domain of the user agent.

4.2. Overview

The protocol defined in this specification clearly separates SIP signaling and the exchange of policies. SIP signaling is only used to rendezvous the UA with the policy server. From this point on, UA and policy server communicate directly with each other over a separate policy channel. This is opposed to a piggyback model, where the exchange of policy information between endpoint and a policy server in the network is piggybacked onto the SIP signaling messages that are exchanged between endpoints.

The main advantage of using a separate policy channel is that it

decouples the exchange of signaling messages between endpoints from the exchange of policies between endpoint and policy server. This decoupling provides a number of desirable properties. It enables the use of separate encryption mechanisms on the signaling path (to secure the communication between endpoints) and on the policy channel (to secure the communication between endpoint and policy server). Policies can be submitted directly from the policy server to the endpoint and never travel along the signaling path, possibly crossing many domains. Endpoints set up a separate policy channel to each policy server and can specifically decide which information they want to disclose to which policy server. Finally, policy servers do not need to rely on a SIP signaling message flowing by to send policies or policy updates to an endpoint. A policy server can use the policy channel at any time to update session policies as needed. A disadvantage of the separate channel model is that it requires additional messages for the exchange of policy information.

Following this model, signaling for session-specific policies involves the following two fundamental tasks:

1. UA/policy server rendezvous: a UA setting up a session needs to be able to discover the policy servers that are relevant to this session.
2. Policy channel: once the UA has discovered the relevant policy servers for a session, it needs to connect to these servers, disclose session information and retrieve the policies that apply to this session.

The setting up session-specific policies over the policy channel involves the following steps:

1. A user agent submits information about the session it is trying to establish to the policy server and asks whether a session using these parameters is permissible.
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.
3. The policy server can update the policy decision at a later time. A policy decision update can, for example, propose additional changes to the session (e.g. change the available bandwidth) or deny a previously accepted session (i.e. disallow the continuation of a session).

In many cases, the mechanism for session-specific policies will be used to disclose session information and return session policies. However, some scenarios may only involve the disclosure of session

information to a network intermediary. If an intermediary does not intend to return a policy, it can simply accept the session as it was proposed. Similarly, some session-specific policies only apply to the offer (and therefore only require the disclosure of the offer) whereas others apply to offer and answer. Both types of policies are supported by session-specific policy mechanism.

4.3. Examples

This section provides two examples to illustrate the overall operation of session-specific policies. The call flows depict the rendezvous mechanism between UA and policy server and indicate the points at which the UA exchanges policy information with the policy server.

The example is based on the following scenario: there are two domains (domain A and domain B), which both have session-specific policies for the UAs in their domain. Both domains do not provide 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 policies in both domains involve the session description offer and answer.

4.3.1. Offer in Request

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

(1) UA A sends an INVITE to proxy P A. P A knows that policies apply to this session and (2) returns a 488 to UA A. P A includes the URI of PS A in the 488 response. This step is needed since the UAC has no prior knowledge about the URI of PS A. (3) UA A uses the URI to contact PS A, discloses the session description offer to PS A and (4) receives policies for the offer. (5) UA A reformulates the INVITE request under consideration of the received policies and includes a Policy-Id header to indicate that it has already contacted PS A. P A does not reject the INVITE this time and removes the Policy-Id header when forwarding the INVITE. P B adds a Policy-Contact header containing the URI of PS B. (6) UA B uses this URI to contact PS B and discloses the offer and the answer it is about to send. (7) UA B receives policies from PS B and applies them to the offer and answer respectively. (8) UA B returns the updated answer in the 200 OK. (9) UA A contacts PS A with the answer and (10) retrieves answer policies from PS A.

```
    UA A              P A              P B             UA B
     |                 |                |                 |
     | INVITE offer    |                |                 |
     |---------------->|                |                 | (1)
     | 488             |                |                 |
     | + Policy-Contact|                |                 |
     |<----------------|                |                 | (2)
     | ACK             |                |                 |
     |---------------->|                |                 |
     |                 | PS A           |                 |
     |                    |             |                 |
     | PolicyChannel      |             |                 |
     | + InfoOffer        |             |                 |
     |------------------->|             |                 | (3)
     | PolicyChannel      |             |                 |
     | + PolicyOffer      |             |                 |
     |<-------------------|             |                 | (4)
     |                    |             |                 |
     |                 |                |                 |
     | INVITE offer'   | INVITE offer'  | INVITE offer    |
     | + Policy-Id     |                | + Policy-Contact|
     |---------------->|--------------->|---------------->| (5)
     |                 |                |                 |
     |                 |           PS B |                 |
     |                 |             |                    |
     |                 |             | PolicyChannel      |
     |                 |             | + InfoOffer        |
     |                 |             | + InfoAnswer       |
     |                 |             |<-------------------| (6)
     |                 |             | PolicyChannel      |
     |                 |             | + PolicyOffer      |
     |                 |             | + PolicyAnswer     |
     |                 |             |------------------->| (7)
     |                 |             |                    |
     |                 |                |                 |
     | OK answer       | OK answer      | OK answer       |
     |<----------------|<---------------|<----------------| (8)
     | ACK                                                |
     |--------------------------------------------------->|
     |                 |                |                 |
     |                    |             |                 |
     | PolicyChannel      |             |                 |
     | + InfoAnswer       |             |                 |
     |------------------->|             |                 | (9)
     | PolicyChannel      |             |                 |
     | + PolicyAnswer     |             |                 |
     |<-------------------|             |                 | (10)
     |                    |             |                 |
```

4.3.2. Offer in Response

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

Steps (1) - (8) are analogous to steps (1) - (8) in the previous flow. An important difference is that in steps (9) and (10) UA A contacts PS A after receiving the offer in the 200 OK but before returning the answer in step (11). This enables UA A to return the final answer, which includes all applicable policies, in the ACK. However, it requires that PS A immediately returns a policy to avoid a delay in the transmission of the ACK. This is similar to Flow I in [9].

```
    UA A              P A              P B             UA B
     |                 |                |                 |
     | INVITE          |                |                 |
     |---------------->|                |                 | (1)
     | 488             |                |                 |
     | + Policy-Contact|                |                 |
     |<----------------|                |                 | (2)
     | ACK             |                |                 |
     |---------------->|                |                 |
     |                 | PS A           |                 |
     |                    |             |                 |
     | PolicyChannel      |             |                 |
     |------------------->|             |                 | (3)
     | PolicyChannel      |             |                 |
     |<-------------------|             |                 | (4)
     |                    |             |                 |
     |                 |                |                 |
     | INVITE          | INVITE         | INVITE          |
     | + Policy-Id     |                | + Policy-Contact|
     |---------------->|--------------->|---------------->| (5)
     |                 |                |                 |
     |                 |           PS B |                 |
     |                 |             |                    |
     |                 |             | PolicyChannel      |
     |                 |             | + InfoOffer        |
     |                 |             |<-------------------| (6)
     |                 |             | PolicyChannel      |
     |                 |             | + PolicyOffer      |
     |                 |             |------------------->| (7)
     |                 |             |                    |
     |                 |                |                 |
     | OK offer        | OK offer       | OK offer        |
     |<----------------|<---------------|<----------------| (8)
     |                 |                |                 |
```

```
     |                     |              |                  |
     | PolicyChannel       |              |                  |
     | + InfoOffer         |              |                  |
     | + InfoAnswer        |              |                  |
     |-------------------->|              |                  | (9)
     | PolicyChannel       |              |                  |
     | + PolicyOffer       |              |                  |
     | + PolicyAnswer      |              |                  |
     |<--------------------|              |                  | (10)
     |                     |              |                  |
     | ACK answer                                            |
     |------------------------------------------------------>| (11)
     |                  |                 |                  |
     |                  |              |                     |
     |                  |              | PolicyChannel       |
     |                  |              | + InfoAnswer        |
     |                  |              |<--------------------| (12)
     |                  |              | PolicyChannel       |
     |                  |              | + PolicyAnswer      |
     |                  |              |-------------------->| (13)
     |                  |              |                     |
```

4.4. UA/Policy Server Rendezvous

The first step in setting up session-specific policies is to rendezvous the UAs with the relevant policy servers. This is achieved by providing the URIs of all policy servers relevant for a session to the UAs.

4.4.1. UAC Behavior

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

The UAC may receive a 488 in response to an INVITE or UPDATE request, which contains a Policy-Contact header field. This is a new header defined in this specification that contains the URI of a policy server. A 488 response with this header is generated by a proxy to convey the URI of the local policy server to the UAC. The UAC SHOULD use this URI to contact the policy server using mechanism defined in Section 4.5. The UAC SHOULD apply the policies received and resend the updated request.

The UAC MUST insert a Policy-Id header into a request if it has contacted a policy server for this request. The Policy-Id header MUST include the URIs of all policy servers the UAC has contacted for the request. The Policy-Id header enables a proxy to determine

whether the URI of its policy server is already known to the UAC (and thus the request can be passed through) or whether the URI still needs to be conveyed to the UAC in a 488 response.

In some cases, a request may traverse multiple domains with session-policies in place. Each of these domains may return a 488 response containing a policy server URI. Since the UAC contacts a policy server after receiving a 488 response from a domain and before re-sending the request, session policies are always applied to a request in the order in which the request traverses through the domains. The UAC SHOULD NOT change this implicit order among policy servers.

Some types of session policies only apply to the offer whereas other policies apply to the offer as well as the answer. A UA SHOULD generally disclose the offer and the answer to the policy server. However, the policy server may indicate on the policy channel (after receiving the offer) that the disclosure of the answer is not needed for this session. In this case, the UAC MAY skip the disclosure of the answer for this particular session.

Depending on whether or not the UAC has included an offer in the INVITE request it has sent to the UAS, it will receive an answer or an offer in the response from the UAS. If the response contains an answer (i.e. the request contained an offer), it MUST send the ACK before contacting the policy server with the answer. The UAC MUST contact the same policy servers it has contacted for the offer. If the response contains an offer (i.e. the INVITE request was empty), the UAC MUST first contact the policy server to retrieve session policies and apply these policies before sending the answer in the ACK. The answer in the ACK will therefore already consider the relevant policies.

> This approach assumes that the policy server immediately responds to a policy request and does not require manual intervention to create a policy. A delay in the response from the policy server would delay the transmission of the ACK and could trigger retransmissions of the INVITE response (also see the recommendations for Flow I in [9]).

4.4.2. Caching of Policy Server URIs

A UAC may frequently need to contact the policy server in the local domain. To avoid the retransmission of the local policy server URI for each INVITE or UPDATE request, the UAC SHOULD cache the URI of the local policy server. The UAC may receive this URI in a 488 from the local proxy after sending an INVITE or UPDATE message. Alternatively, the UA may also have received the local policy server URI through configuration or other means. If the UAC has received a

local policy server URI through configuration and receives another one in a 488 response, it SHOULD overwrite the configured URI with the one received in the 488 response. The UAC SHOULD contact the cached local policy server URI when creating a new INVITE or UPDATE request, before they are sent.

Domains can prevent the UAC from caching the local policy server URI. This is useful, for example, if the policy server does not need to be involved in all sessions or the policy server URI changes from session to session. A proxy can mark the URI of such a local policy server as "non-cacheable". The UA SHOULD NOT cache a non-cacheable policy server URI. It SHOULD remove the current URI from the cache when receiving a "non-cacheable" URI.

The UAC SHOULD NOT cache policy server URIs it has received from proxies outside of the local domain. These policy servers may not be relevant for subsequent sessions, which may go to a different destination, traversing different domains.

The UAC SHOULD store the list of policy server URIs is has contacted for a session as part of the session state. The UAC should keep this list until the session is terminated. The UAC SHOULD contact the policy server URIs in this list for mid-dialog INVITE or UPDATE request. Caching these URIs avoids the retransmission of policy server URIs for each mid-dialog requests.

4.4.3. UAS Behavior

An incoming INVITE or UPDATE request may contain a Policy-Contact header with a list of policy server URIs. The UAS SHOULD contact all policy server URIs in a Policy-Contact header. The UAS MUST contact the policy server URIs in the order in which they were contained in the Policy-Contact header, starting with the topmost value.

If the UAS receives an ACK containing an answer, it SHOULD contact the policy servers again with the answer. In this case, it MUST contact the same policy servers it has contacted for the offer. However, the policy server may have indicated in response to the offer that the disclosure of the answer is not needed for this session. In this case, the UAS MAY skip the disclosure of the answer for this particular session.

4.4.4. Proxy Behavior

A proxy provides rendezvous functionality for UAs and the local policy server. This is achieved by conveying the URI of the local policy server to the UAC or the UAS (or both) when processing an INVITE or UPDATE request.

If an 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 local policy server URI to the UAC. Before rejecting a request, the proxy MUST verify that the request does not have a Policy-Id header field, which already contains the local policy server URI. If the request does not have such a header or the local policy server URI is not present in this header, then the proxy MAY reject the request with a 488. The proxy MUST insert a Policy-Contact header in the 488 response that contains the URI of the local policy server. The proxy MAY add the header field parameter "non-cacheable" to prevent the UAC from caching this policy server URI.

If the local policy server URI is already present in the Policy-Id header of an INVITE or UPDATE request, the proxy MUST NOT reject the request as described above. The proxy SHOULD remove this policy server URI from the Policy-Id header field before forwarding the request.

The proxy MAY insert a Policy-Contact header field into an INVITE or UPDATE request in order to convey 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.

4.4.5. Header Definition and Syntax

The Policy-Id header field is inserted into an INVITE or UPDATE request by the UAC. It identifies all policy servers the UAC has contacted for this request. A Policy-Id header value is the URI of a policy server.

The syntax of the Policy-Id header field is:

```
Policy-Id        = "Policy-Id" HCOLON absoluteURI
                   *(COMMA absoluteURI)
```

The Policy-Contact header field can 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 Policy-Contact header field can also 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 used in a request.

The syntax of the Policy-Contact header field is:

```
Policy-Contact   = "Policy-Contact" HCOLON policyURI
                   *(COMMA policyURI)
policyURI        = absoluteURI [ SEMI "non-cacheable" ]
```

The BNF for absoluteURI is defined in [6].

Table 1 is an extension of Tables 2 and 3 in [6]. The column 'UPD' is for the UPDATE method [5].

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG	UPD
Policy-Id	R	rd	-	-	-	o	-	-	o
Policy-Contact	R	a	-	-	-	o	-	-	o
Policy-Contact	488	a	-	-	-	o	-	-	o

Table 1: Policy-Id and Policy-Contact Header Fields

4.5. Policy Subscription

The rendezvous mechanism described in the previous section enables proxies to deliver the URIs of policy servers to the UAC and UAS. This section describes the mechanism for the policy channel, i.e. the protocol UAs use to contact the policy servers. The main task of the policy channel is to enable a UA to submit information about the session it is trying to establish (i.e. the offer and the answer) to a policy server and to receive the resulting session-specific policies and possible updates to these policies in response.

A UA compliant to this specification MUST implement the Event Package for Session-Specific Session Policies [2]. It contacts a policy server by subscribing to this event package.

When subscribing to session-specific policies, the UA discloses information about the session it is trying to establish to the policy server as described in [2]. This information is used by the policy server to determine the session-specific policy for this session. The policy server returns the policies that apply to this session in NOTIFY messages. It returns an initial set of policies when the subscription is established and may notify the UA when there are updates to these policies. Complete call flow examples for session-specific policies that include policy channel messages can be found in Appendix B.

A UA SHOULD use the policies it has received from the policy server

in the current session (i.e. the session the subscription is for).

When a UA receives a policy update, it SHOULD apply the update to the current session. If this update causes a change in the session description of a session, the UA may need to generate a re-INVITE or UPDATE message to re-negotiate the modified session description with its peer UA. For example, if a policy update disallows the use of video and video is part of the current session description, then the UA will need to create an new session description offer without video. After receiving this offer, the peer UA knows that video can't be used any more and responds with the corresponding answer.

5. 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 set up a user agent so that it is unable to successfully establish 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 will cause the UA to terminate the session.

A user agent transmits session information to a policy server for session-specific policies. This session information may contain sensitive data the user may not want an eavesdropper or an unauthorized policy server to see. In particular, 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.

User agents should therefore authenticate a policy server before accepting a session policy. Policy servers should authenticate user agents before sending a session policy. This document does not define the protocols between user agents and policy servers and merely refers to other specifications. The security considerations of these specifications apply and provide the mechanisms needed to secure these protocols.

Administrators should use SIPS URIs as policy server URIs, if possible, so that subscriptions to session policies are transmitted over TLS.

This document defines a new mechanism that enables proxies to rendezvous UAs and policy servers. An attacker can use this mechanism to refer a UA to compromised policy server. The UA can prevent such an attack from being effective by authenticating policy

servers.

An attacker could intercept SIP messages between the UA and proxy and remove the policy headers needed for session-specific policies. This would impede the rendezvous between UA and policy server and, since the UA would not contact the policy server, may prevent a UA from setting up a session. This attack can be prevented by using a secured transport protocol such as TLS between proxies and UA.

6. IANA Considerations

6.1. Registration of the "Policy-Id" Header

Name of Header: Policy-Id

Short form: none

Normative description: Section 4.4.5 of this document

6.2. Registration of the "Policy-Contact" Header

Name of Header: Policy-Contact

Short form: none

Normative description: Section 4.4.5 of this document

6.3. Registration of the "policy" SIP Option-Tag

Name of option: policy

Description: Support for the Policy-Contact and Policy-Id headers.

SIP headers defined: Policy-Contact, Policy-Id

Normative description: This document

Appendix A. Acknowledgements

Many thanks to Allison Mankin for the discussions and the suggestions for this draft. Many thanks to everyone who contributed by providing feedback on the mailing list and in IETF meetings.

Appendix B. Session-Specific Policies - Call Flows

The following call flows illustrate the overall operation of session-specific policies. The call flows contain all messages needed for UA/policy server rendezvous and the policy subscription.

The following abbreviations are used:

o: offer
o': offer modified by a policy
po: offer policy
a: answer
a': answer modified by a policy
pa: answer policy
ps uri: policy server URI (in Policy-Contact header)
ps id: policy server id (in Policy-Id header)

B.1. Offer in Invite

```
   UA A       P A      PS A      PS B       P B      UA B
     |         |         |         |         |         |
     |(1) INV <o>        |         |         |         |
     |-------->|         |         |         |         |
     |(2) 488 <ps uri>   |         |         |         |
     |<--------|         |         |         |         |
     |(3) ACK  |         |         |         |         |
     |-------->|         |         |         |         |
     |(4) SUBSCRIBE <o>  |         |         |         |
     |------------------>|         |         |         |
     |(5) 200 OK         |         |         |         |
     |<------------------|         |         |         |
     |(6) NOTIFY <po>    |         |         |         |
     |<------------------|         |         |         |
     |(7) 200 OK         |         |         |         |
     |------------------>|         |         |         |
     |(8) INV <ps id, o'>|         |         |         |
     |-------->|         |         |         |         |
     |         |(9) INV <o'>       |         |         |
     |         |---------------------------->|         |
     |         |         |         |         |(10) INV <o', ps uri>
     |         |         |         |         |-------->|
     |         |         |         |(11) SUBSCRIBE <o', a>
     |         |         |         |<------------------|
     |         |         |         |(12) 200 OK        |
     |         |         |         |------------------>|
     |         |         |         |(13) NOTIFY <po, pa>
     |         |         |         |------------------>|
     |         |         |         |(14) 200 OK        |
     |         |         |         |<------------------|
     |         |         |         |         |(15) 200 OK <a'>
     |         |         |         |         |<--------|
     |         |(16) 200 OK <a'>   |         |         |
     |         |<----------------------------|         |
     |(17) 200 OK <a'>   |         |         |         |
     |<--------|         |         |         |         |
     |(18) ACK |         |         |         |         |
     |------------------------------------------------>|
     |(19) SUBSCRIBE <o', a'>      |         |         |
     |------------------>|         |         |         |
     |(20) 200 OK        |         |         |         |
     |<------------------|         |         |         |
     |(21) NOTIFY <pa>   |         |         |         |
     |<------------------|         |         |         |
     |(22) 200 OK        |         |         |         |
     |------------------>|         |         |         |
     |         |         |         |         |         |
     |         |         |         |         |         |
```

B.2. Offer in Response

```
   UA A       P A      PS A      PS B       P B      UA B
     |         |         |         |         |         |
     |(1) INV  |         |         |         |         |
     |-------->|         |         |         |         |
     |(2) 488 <ps uri>   |         |         |         |
     |<--------|         |         |         |         |
     |(3) ACK  |         |         |         |         |
     |-------->|         |         |         |         |
     |(4) SUBSCRIBE      |         |         |         |
     |------------------>|         |         |         |
     |(5) 200 OK         |         |         |         |
     |<------------------|         |         |         |
     |(6) NOTIFY         |         |         |         |
     |<------------------|         |         |         |
     |(7) 200 OK         |         |         |         |
     |------------------>|         |         |         |
     |(8) INV <ps id>    |         |         |         |
     |-------->|         |         |         |         |
     |         |(9) INV  |         |         |         |
     |         |---------------------------->|         |
     |         |         |         |         |(10) INV <ps uri>
     |         |         |         |         |-------->|
     |         |         |         |(11) SUBSCRIBE <o> |
     |         |         |         |<------------------|
     |         |         |         |(12) 200 OK        |
     |         |         |         |------------------>|
     |         |         |         |(13) NOTIFY <po>   |
     |         |         |         |------------------>|
     |         |         |         |(14) 200 OK        |
     |         |         |         |<------------------|
     |         |         |         |         |(15) 200 OK <o'>
     |         |         |         |         |<--------|
     |         |(16) 200 OK <o'>   |         |         |
     |         |<----------------------------|         |
     |(17) 200 OK <o'>   |         |         |         |
     |<--------|         |         |         |         |
     |(18) SUBSCRIBE <o', a>       |         |         |
     |------------------>|         |         |         |
     |(19) 200 OK        |         |         |         |
     |<------------------|         |         |         |
     |(20) NOTIFY <po, pa>         |         |         |
     |<------------------|         |         |         |
     |(21) 200 OK        |         |         |         |
     |------------------>|         |         |         |
     |(22) ACK <a'>      |         |         |         |
     |------------------------------------------------>|
```

```
     |         |         |         |(23) SUBSCRIBE <o', a'>
     |         |         |         |<------------------|
     |         |         |         |(24) 200 OK        |
     |         |         |         |------------------>|
     |         |         |         |(25) NOTIFY <po, pa>
     |         |         |         |------------------>|
     |         |         |         |(26) 200 OK        |
     |         |         |         |<------------------|
     |         |         |         |         |         |
     |         |         |         |         |         |
```

7. References

7.1. Normative References

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

[2] Hilt, V. and G. Camarillo, "A Session Initiation Protocol (SIP) Event Package for Session-Specific Session Policies.", draft-ietf-sipping-policy-package-01 (work in progress), April 2006.

[3] Hilt, V., Camarillo, G., and J. Rosenberg, "A User Agent Profile Data Set for Media Policy", draft-ietf-sipping-media-policy-dataset-01 (work in progress), March 2006.

[4] Petrie, D., "A Framework for Session Initiation Protocol User Agent Profile Delivery", draft-ietf-sipping-config-framework-08 (work in progress), March 2006.

[5] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.

[6] 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.

7.2. Informative References

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

[8] Petrie, D., Lawrence, S., Dolly, M., and V. Hilt, "A Schema and Guidelines for Defining Session Initiation Protocol User Agent Profile Data Sets", draft-petrie-sipping-profile-datasets-03

(work in progress), October 2005.

[9] Rosenberg, J., Peterson, J., Schulzrinne, H., and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", BCP 85, RFC 3725, April 2004.

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@cisco.com

Acknowledgment

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