SIPPING Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 23, 2011

V. Hilt
Bell Labs/Alcatel-Lucent
G. Camarillo
Ericsson
J. Rosenberg
jdrosen.net
February 19, 2011

A Framework for Session Initiation Protocol (SIP) Session Policies
draft-ietf-sip-session-policy-framework-10

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. This document specifies a framework for SIP session policies that provides a standard mechanism by which a proxy can define or influence policies on sessions, such as the codecs or media types to be used. It defines a model, an overall architecture and new protocol mechanisms for session policies.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on August 23, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [RFC3261] 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 session-mode 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) [RFC4566].

Experience has shown that there is a need for SIP intermediaries to impact aspects of a session. For example, SIP can be used in a wireless network, which has limited resources for media traffic. During periods of high activity, the wireless network provider could want to restrict the amount of bandwidth available to each user. With session policies, an intermediary in the wireless network can inform the user agent about the bandwidth it has available. 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 can have a service level agreement with a user that defines the set of media types the 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 have the need to enforce the session policies they have in place. For example, a domain might have a policy that disallows the use of video and can have an enforcement mechanism that drops all packets containing a video encoding. Unfortunately, these 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 can 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 the proposed session description. It can then return a policy for the session description that replaces the IP addresses and ports of the UA with the ones opened in the firewall/NAT that are reachable from external. Session-specific policies provide information about a specific session to a domain, which can be used to implement policies for opening pinholes on a firewall/NAT. 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 all 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 when it initializes (e.g., at the time the device is powered on) and when policies are changed.

This specification defines a framework for SIP session policies. It specifies a model, the overall architecture and new protocol mechanisms that are needed for session-independent and session-specific policies. Since session-specific and session-independent policies have different requirements, this specification defines two different mechanisms to deliver them to user agents. These mechanisms are independent of each other and, depending on whether one or both types of session policies are needed, it is possible to use the session-specific or the session-independent mechanism or both to deliver policies to user agents.

It is RECOMMENDED that UAs and intermediaries use the mechanisms defined in this specification for signaling session policies to

endpoints. To ensure backwards compatibility with UAs that do not support this specification, intermediaries may choose to resort to existing mechanisms such as rejecting sessions that are not policy compliant with a 488 response as a fallback solution if a UA does not indicate support for session policies. UAs that do not support session policies will receive the same user experience as they would today. As these techniques are known to have many drawbacks it is RECOMMENDED that UAs and intermediaries use explicit signaling of policies using the mechanisms defined in this specification.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Session-Independent Policies

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

3.1. Architecture and Overview

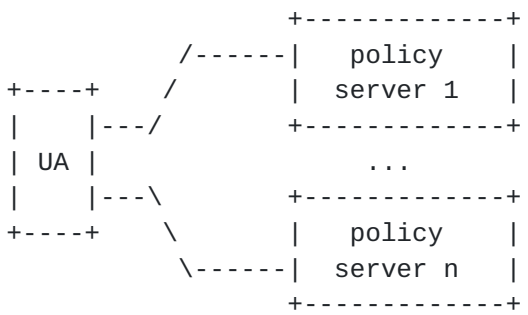


Figure 1

A SIP UA can 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 local network domain

(i.e., the domain from which the UA receives IP service) and possibly the SIP service provider domain (i.e., the domain the UA registers at). The local network can have policies that support the access network infrastructure. For example, in a wireless network where bandwidth is scarce, a provider can restrict the bandwidth available to an individual user. The SIP service provider can 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 two policy servers.

Setting up session-independent policies involves the following steps:

1. A user agent discovers session-independent policy servers in the local network and SIP service provider domain
2. A user agent requests session-independent policies from the discovered policy servers. A user agent typically requests these policies when it starts up or connects to a new network domain.
3. The policy server selects the policies that apply to this user agent. The policy server can 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.
4. The policy server can update the policies, for example, when network conditions change.

3.2. Policy Subscription

3.2.1. UAC Behavior

A UA that supports session-independent policies compliant to this specification MUST attempt to retrieve session-independent policies from the local network and the SIP service provider domain, unless the UA knows (e.g., through configuration) that a domain does not provide session-independent policies (in which case the UA SHOULD NOT retrieve session-independent policies from this specific domain).

A UA that supports session-independent policies compliant to this specification MUST support the retrieval of session-independent policies from the local network and the SIP service provider domain using the "ua-profile" event package defined in the Framework for SIP User Agent Profile Delivery [I-D.ietf-sipping-config-framework]. The UA MAY support other methods of retrieving session-independent policies from local network and SIP service provider domain.

The "ua-profile" event package [I-D.ietf-sipping-config-framework] provides a mechanism to subscribe to session-independent policies. A UA subscribes to the policy server in the local network domain using the procedures defined for the "local-network" profile-type. The UA

uses the procedures defined for the "user" profile type to subscribe to the policy server in the SIP service provider domain.

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

- The UA registers a new address-of-record (AoR) or removes a AoR from the set of AoRs it has registered. In these cases, the UA MUST establish subscriptions for each new AoR using the "user" and the "local-network" profile-types. The UA MUST terminate all subscriptions for AoRs it has removed.
- The UA changes the domain it is connected to. The UA MUST terminate all existing subscriptions for the "local-network" profile-type. The UA MUST then create a new subscription for each AoR it maintains 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 UA is unable to establish a subscription, the UA 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. However, a UA SHOULD retry the subscription with a longer time interval (e.g., once every 24 hours). This enables UAs to detect new policies that are deployed in a network that previously did not have policies.

A UA that supports session-independent policies compliant to this specification MUST support the User Agent Profile Data Set for Media Policy [I-D.ietf-sipping-media-policy-dataset]. To indicate that the UA wants to receive session-independent policies, the UA includes the MIME type "application/media-policy-dataset+xml" in the Accept header field of a SUBSCRIBE request.

A UA MUST apply the session-independent policies it has received and use these policies in the session descriptions it creates. If the UA decides not to use the received policies, then the UA MUST NOT set up a session unless it changes the domain that provided these policies. A UA MAY try to connect to another local network and/or SIP service provider domain with a different set of policies.

If a UA receives both session-independent and session-specific policies, the UA MUST apply the session-independent policies to the session description before the session description is sent to the session-specific policy server (see Section 4). Thus, session-independent policies are always applied before session-specific

policies are retrieved.

3.2.2. UAS Behavior

A policy server MAY send a notification to the UA every time the session-independent policies covered by the subscription change. The definition of what causes a policy to change is at the discretion of the administrator. A change in the policy can 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.

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 can change during the course of the session. For example, a user can 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

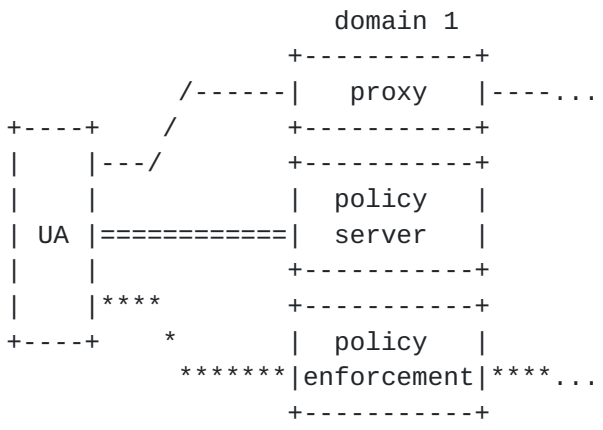


Figure 2

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 ensures that each UA has the URI [RFC3986] of the policy server in its domain and knows where to retrieve policies from. The proxy conveys the policy server URI to UAs in case they have not yet received it (e.g., in a previous call or through configuration). The proxy does not deliver the actual policies to UAs.

The policy server is a separate logical entity that can 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 from the UA, 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 of the scope of this specification. A policy server can, for example, query an external entity to get policies or it can 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 contact the policy server. It provides information about the current session to the policy server and receives session policies in response. The UA can also receive policy updates from the policy server during the course of a session.

A network can 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 to a policy enforcement infrastructure.

In principle, each domain that is traversed by SIP signaling messages can define session-specific policies for a session. Each domain 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 signaling between endpoints from the policy exchange between an endpoint and a policy server. This decoupling has 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 do not travel along the signaling path, possibly crossing many domains. Endpoints set up a separate policy channel to each policy server and can disclose the information requested by the specific policy server (e.g., offer or offer/answer). 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 communication between UA and policy server on 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 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 shown in Figure 3 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 request to proxy P A. P A knows that policies apply to this session and (2) returns a 488 (Not Acceptable Here) response to UA A. P A includes the URI of PS A in the 488 (Not Acceptable Here) 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 field to indicate that it has already contacted PS A. P A does not reject the INVITE request this time and removes the Policy-Id header field when forwarding the INVITE request. P B adds a Policy-Contact header field containing the URI of PS B. (6) UA B uses this URI to contact PS B and disclose 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) response. (9) UA A contacts PS A again with the current offer and answer and (10) retrieves the policies for

both from PS A.

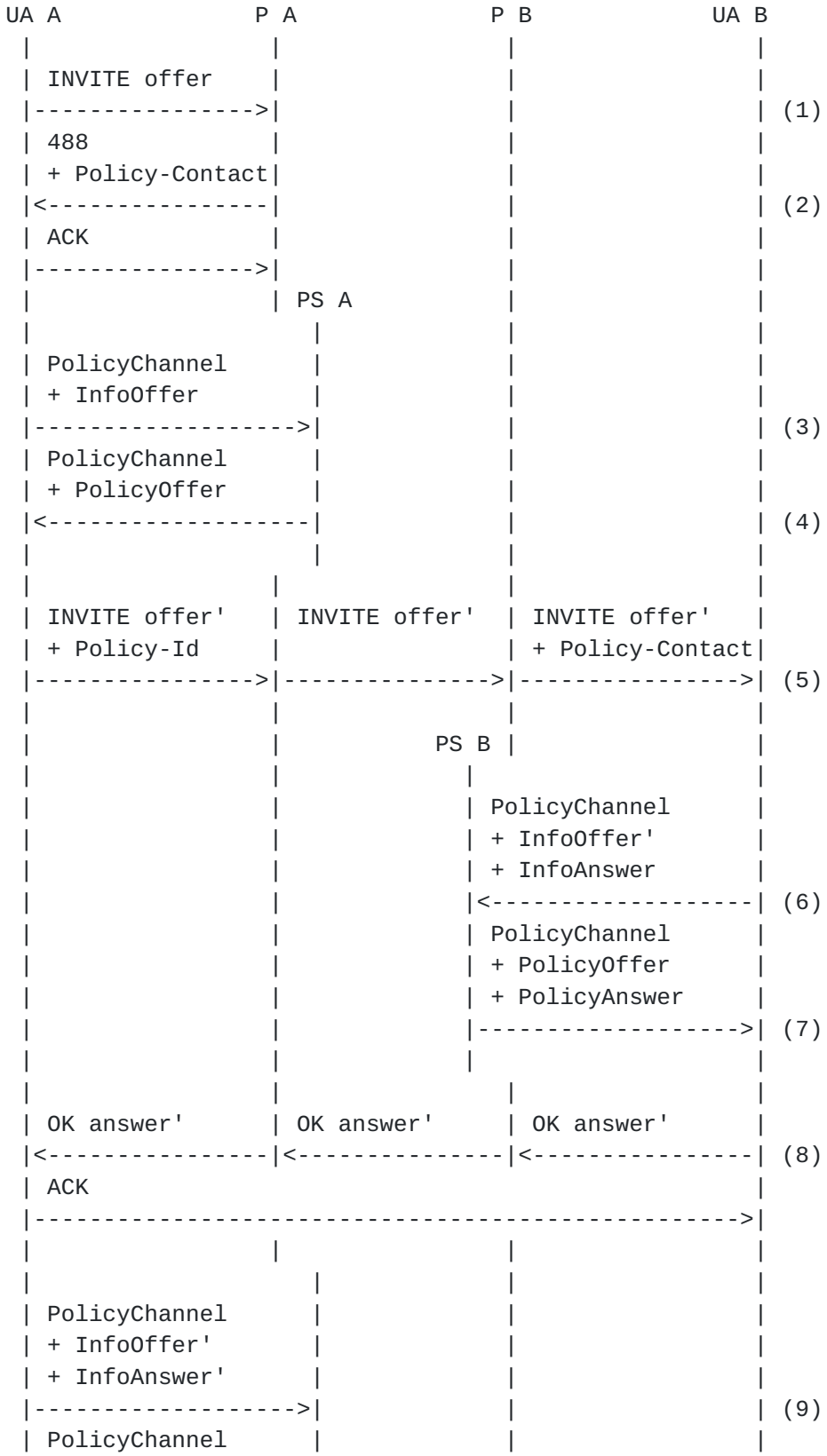
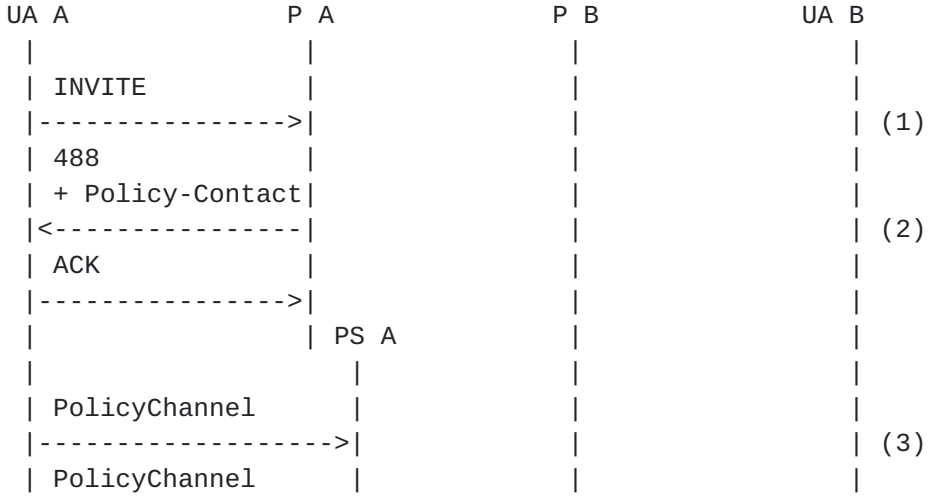


Figure 3

4.3.2. Offer in Response

The call flow shown in Figure 4 depicts an INVITE transaction with the offer in the response.

(1) UA A sends an INVITE request without an offer to proxy P A and (2) P A returns a 488 (Not Acceptable Here) response containing the URI of PS A. (3),(4) UA A uses this policy server URI to set up the policy channel. At this time, UA A does not disclose a session description since it does not have the offer yet. (5) UA A re-sends the INVITE request and includes a Policy-Id header field to indicate that it has contacted PS A. P A does not reject the INVITE request this time and removes the Policy-Id header field when forwarding the INVITE request. P B adds a Policy-Contact header field containing the URI of PS B. (6) UA B uses this URI to discloses the offer to PS B. (7) UA B receives policies from PS B and applies them to the offer. (8) UA B returns the updated offer the 200 (OK) response. (9),(10) UA A contacts PS and discloses the offer and the answer it is about to send. An important difference to the flow in the previous example is that UA A performs steps (9) and (10) before returning the answer in step (11). This enables UA A to return the final answer in the ACK request, which includes all applicable policies. However, it requires that PS A immediately returns a policy to avoid a delay in the transmission of the ACK request. (12),(13) UA B again sends the current offer and answer to PS B and applies the policies it receives to both before using them.



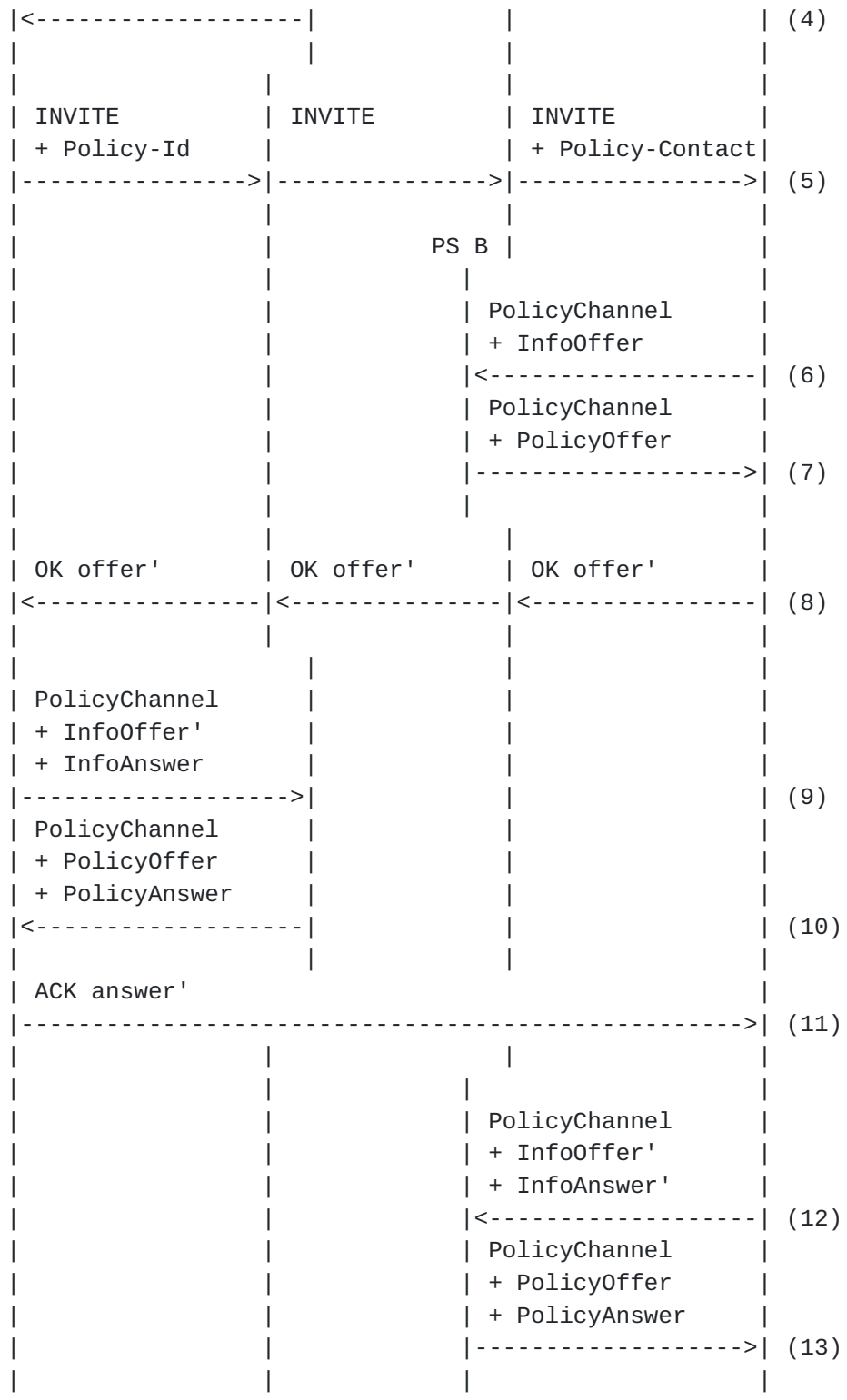

INVITE
+ Policy-Id
INVITE
INVITE
+ Policy-Contact
PS B
PolicyChannel
+ InfoOffer
PolicyChannel
+ PolicyOffer
OK offer'
OK offer'
OK offer'
PolicyChannel
+ InfoOffer'
+ InfoAnswer
PolicyChannel
+ PolicyOffer
+ PolicyAnswer
ACK answer'
PolicyChannel
+ InfoOffer'
+ InfoAnswer'
PolicyChannel
+ PolicyOffer
+ PolicyAnswer
(4)
(5)
(6)
(7)
(8)
(9)
(10)
(11)
(12)
(13)

Figure 4

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

A UAC compliant to this specification MUST include a Supported header field with the option tag "policy" into all requests that can initiate an offer/answer exchange [RFC3264] (e.g., INVITE, UPDATE [RFC3311] and PRACK [RFC3262] requests). The UAC MUST include the "policy" option tag into these requests even if the particular request does not contain an offer or answer (e.g., an INVITE request without an offer). A UAC MAY include the "policy" option tag into all requests.

A UAC can receive a 488 (Not Acceptable Here) response that contains a Policy-Contact header field. The Policy-Contact header field is a new header field defined in this specification. It contains one (or multiple alternative) URIs for a policy server. A 488 (Not Acceptable Here) response with this header field is generated by a proxy to convey an URI of the local policy server to the UAC. After receiving a 488 (Not Acceptable Here) response with a Policy-Contact header field, a UAC compliant to this specification needs to decide if it wants to continue with the session now knowing that there is a policy server. If the UAC decides to continue, the UAC MUST use one of the policy server URIs to contact the policy server using the mechanism defined in Section 4.5.

The Policy-Contact header can contain multiple URIs each with a different URI scheme and containing an "alt-uri" parameter with identical values. These URIs represent alternative policy channel mechanisms for obtaining the same policy. The UAC chooses one of the alternative URIs to use to obtain the policy. The UAC MAY take as a hint the order of the alternative URIs as indicating a preference as to which URI to use. The topmost URI in the list might be more preferred by the domain of the proxy for use to obtain the policy.

After receiving policies from the policy server, the UAC decides if it wants to accept these policies or not. If the UAC accepts these policies, the UAC MUST apply them to the current request and resend the updated request. If no changes are required by policies or no policies have been received, the request can be resent without any policy-induced changes. If the UAC decides that the list of policy servers or the received session policies are unacceptable, then the UAC MUST NOT resend the request.

To protect the integrity of the policy server URI in a Policy-Contact header field, the UAC SHOULD use a secured transport protocol such as Transport Layer Security (TLS) [RFC5246] between UAC and proxy.

The UAC MUST insert a Policy-Id header field into requests for which it has contacted a policy server and accepted the policies received. The Policy-Id header field is a new header field that is defined in this specification. The UA MUST create a Policy-Id header field value for each policy server it has contacted during the preparation of the request. A Policy-Id header field value contains two pieces of information: the policy server URI and an optional token. The policy server URI is the URI the UA has used to contact the policy server. The token is an opaque string the UAC can receive from the policy server. A token can, for example, be contained in the policy document [I-D.ietf-sipping-media-policy-dataset]. If the UAC has received a token from the policy server the UAC MUST include the token in the Policy-Id header field. The format of the Policy-Id header field is defined in Section 4.4.5.

The main purpose of the Policy-Id header field is to enable a proxy to determine if the UAC already knows an URI of the local policy server. If the policy server URI is not yet known to the UAC, the proxy can convey this URI to the UAC by rejecting the request with a 488 (Not Acceptable Here) response.

In some cases, a request can traverse multiple domains with a session-policy server. Each of these domains can return a 488 (Not Acceptable Here) response containing a policy server URI. A UAC contacts a policy server after receiving a 488 (Not Acceptable Here) response from a domain and before re-sending the request. This creates an implicit order between the policy servers in multiple domains. I.e., a UAC contacts the first policy server, re-sends the modified request, contacts the second policy server, re-sends the modified request and so on. This way, session policies are always applied to a request in the order in which the request traverses through the domains. The UAC MUST NOT change this implicit order among policy servers.

A UAC frequently needs to contact the policy server in the local domain before setting up a session. To avoid the retransmission of the local policy server URI in a 488 (Not Acceptable Here) response for each new request, a UA SHOULD maintain a cache that contains the URI of the policy server in the local domain (see Section 4.4.4). The UAC SHOULD use the cached policy server URI to contact the local policy server before sending a request that initiates the offer/answer exchange for a new session (e.g., an INVITE request). The UAC SHOULD NOT cache a policy server URI that is in a different domain than the UAC even if it is the first policy server URI returned. The

first policy server URI returned can be from another domain if the local domain does not have a policy server. Note that UACs perform exact domain comparisons. That is, foo.example.com and example.com are not considered equivalent.

UAs can re-negotiate the session description during a session by initiating a subsequent offer/answer exchange, e.g., in an INVITE, UPDATE or PRACK request. When creating such a mid-dialog request, a UA SHOULD contact all policy servers to which it has established a policy channel during the initial offer/answer exchange (see Section 4.5) before sending the request. This avoids the retransmission of all policy server URIs in 488 (Not Acceptable Here) responses for mid-dialog requests.

4.4.2. Proxy Behavior

A proxy provides rendezvous functionalities for UAs and policy server. This is achieved by conveying the URI of a policy server to the UAC or the UAS (or both) when processing INVITE, UPDATE or PRACK requests (or any other request that can initiate an offer/answer exchange).

If an offer/answer exchange initiating request contains a Supported header field with the option tag "policy", the proxy MAY reject the request with a 488 (Not Acceptable Here) response to provide the local policy server URI to the UAC. Before rejecting a request, the proxy MUST verify that the request does not contain a Policy-Id header field with the local policy server URI as a value. If the request does not contain such a header field or a local policy server URI is not present in this header field, then the proxy MAY reject the request with a 488 (Not Acceptable Here) response. The proxy MUST insert a Policy-Contact header field in the 488 (Not Acceptable Here) response that contains one (or multiple) URIs of its associated policy server. The proxy MAY add the header field parameter "non-cacheable" to prevent the UAC from caching this policy server URI (see Section 4.4.4).

More than one URI for the policy server using different URI schemes MAY be provided by the proxy as alternative URIs to contact the policy. If a proxy includes multiple URIs for the same policy, the proxy MUST include an "alt-uri" parameter for all policy server URIs that are alternatives for obtaining the same policy. The "alt-uri" parameter MUST contain either the domain name of the domain for which all the alternative policy server URIs relate to or a FQDN (e.g., the hostname of a policy server). All URIs that are alternatives for the same policy MUST have the same value for the "alt-uri" parameter. The value used for the "alt-uri" parameter MUST be such that the same value will not be included with other policy server URIs that a UA

needs to contact by any other proxy within the same domain or another domain. A method to create a new unique "alt-uri" parameter value is to examine the value of existing "alt-uri" parameters and to make sure that the new value differs. A proxy MAY hint to a UA at a preference as to which URI to use by including the more preferred URI higher in the list than the other alternative URIs. URIs with the same "alt-uri" parameter MUST use different URI schemes. A SIP or SIPS URI MUST be included even if other URI schemes are defined and used in the future.

If a local policy server URI is present in a Policy-Id header field value of a request, then the proxy MUST NOT reject the request as described above (it can still reject the request for other reasons). The proxy SHOULD remove the Policy-Id header field value of its associated policy server from the Policy-Id header field before forwarding the request. Not removing the Policy-Id header field value will not cause harm, however, the value is not relevant to any other proxy on the path and only increases message size. It also would disclose the policy server URI to subsequent proxies.

The Policy-Id header field serves two main purposes: first and most importantly, it enables the proxy to determine if a UAC already knows the URI of the local policy server. The second purpose of the Policy-Id header field is to enable a domain to route all requests that belong to the same session (i.e., the initial request and requests a UA retransmits after contacting the policy server) to the same proxy and policy server. This is important if a domain has multiple proxy/policy server combinations (e.g., in a proxy/policy server farm that receives requests through a load balancer), which create per-session state in the network. An example for such a scenario is a policy server that is associated with a session border device. The policy server configures the session border device after receiving a session description from the UAC via the policy channel. Retransmitted requests for such a session need to be routed to the same proxy/policy server as the initial request since this proxy/ policy server combination has configured the associated border device for the session.

Routing all requests that belong to the same session to the same proxy can be achieved by using the Policy-Id header field token. It requires that the policy server returns a token to the UAC that uniquely identifies the specific proxy/policy server combination. The UAC includes this token in the Policy-Id header field and it can be used (together with the policy server URI) by the proxies in this domain to route the request along the desired path. The format of this token does not require standardization. The only requirement is that the token provides sufficient information for proxies to route the message inside a domain to the desired proxy/policy server. The

token can, for example, be a numeric identifier or an IP address.

> Note: it has been proposed to use the Policy-Id header field to provide a hint for a proxy that the UAC has actually contacted the policy server. This usage also requires the policy server to return a token to the UA. In addition, the policy server needs to share valid tokens with the proxy. After receiving a request with a Policy-Id header field, the proxy can determine if the token in the Policy-Id header field is valid. If it is valid, the proxy knows that the UA has contacted the policy server for this session. However, this token does not provide any proof that the UA has actually used the policies it has received from the policy server. A malicious UA can simply contact the policy server, discard all policies it receives but still use the token in the Policy-Id header field.

The proxy MAY insert a Policy-Contact header field into INVITE, UPDATE or PRACK requests (or any other request that can initiate an offer/answer exchange) 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 after all existing values at the end of the list. A proxy MUST NOT change the order of existing Policy-Contact header field values.

A proxy MUST use the Record-Route mechanism [RFC3261] if its associated policy server has session policies that apply to mid-dialog requests. The Record-Route header field enables a proxy to stay in the signaling path and re-submit the policy server URIs to UAs during mid-dialog requests that initiate an offer/answer exchange. Re-submitting the policy server URI to UAs ensures that UAs keep contacting the policy server for mid-dialog requests.

A proxy can find out if the UAS supports this extension by examining the Supported header field of responses. The proxy knows that the UAS supports this extension if the Supported header field of a response contains the option tag "policy". A proxy can use this information to determine if the UAS has understood the Policy-Contact header field it has inserted into the request.

To protect the integrity of the policy server URI in a Policy-Contact header field, the proxy SHOULD use a secured transport protocol such as TLS [RFC5246] between proxy and UAs.

4.4.3. UAS Behavior

A UAS can receive an INVITE, UPDATE or PRACK request (or another request that can initiate offer/answer exchanges) that contains a Policy-Contact header field with a list of policy server URIs. A UAS

that receives such a request needs to decide if it wants to accept the session knowing that there are policy servers involved. If the Policy-Contact header contains multiple URIs each with a different URI scheme and containing an "alt-uri" parameter with identical values these URI schemes represent alternative policy channel mechanisms for obtaining the same policy. If the UAS accepts the session, the UAS MUST contact one URI out of each group of URIs with identical "alt-uri" parameter values to obtain the policy. The UAS MAY take as a hint the order of the alternative URIs as indicating a preference as to which URI to use. The topmost URI in the list might be more preferred by the domain of the proxy for use to obtain the policy. The UAS MUST contact all policy server URIs in a Policy-Contact header field that are not part of a group of alternative URIs and MUST contact one URI in each group of alternative URIs. The UAS MUST contact these policy server URIs in the order in which they were contained in the Policy-Contact header field, starting with the topmost value (i.e., the value that was inserted first).

If a UAS decides that it does not want to accept a session because there are policy servers involved or because one of the session policies received from a policy server is not acceptable, the UAS MUST reject the request with a 488 (Not Acceptable Here) response.

The UAS MAY accept a request and continue with setting up a session if it cannot setup a policy channel to the policy server, for example, because the policy server is unreachable or returns an error condition that cannot be resolved by the UAS (i.e., error conditions other than, for example, a 401 (Unauthorized) response). This is to avoid that the failure of a policy server prevents a UA from communicating. Since this session might not be policy compliant without the policy subscription, it can be blocked by policy enforcement mechanisms if they are in place.

A UAS can receive a token from a policy server via the policy channel. Since the UAS does not create a Policy-ID header field, it can simply ignore this token.

A UAS compliant to this specification MUST include a Supported header field with the option tag "policy" into responses to requests that can initiate an offer/answer exchange. The UAS MAY include this option tag in all responses. This way, a proxy that has inserted the Policy-Contact header field can know that the header field was understood by the UAS.

4.4.4. Caching the Local Policy Server URI

A UAC frequently needs to contact the policy server in the local domain before setting up a session. To avoid the retransmission of

the local policy server URI for each session, a UA SHOULD maintain a cache that contains the URI of the local policy server.

A UA can receive this URI in a Policy-Contact header field of a request or a 488 (Not Acceptable Here) response. The UA can also receive the local policy server URI through configuration, for example, via the configuration framework [I-D.ietf-sipping-config-framework]. If a UA has received a local policy server URI through configuration and receives another local policy server URI in a Policy-Contact header field, the UA SHOULD overwrite the configured URI with the most recent one received in a Policy-Contact header field. A policy server URI received in a Policy-Contact header field expires if it has not been refreshed before it reaches the maximum cached URI validity. The default maximum cached URI validity is 24 hours.

Domains can prevent a UA 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 policy server as "non-cacheable". A UA MUST NOT cache a non-cacheable policy server URI. The UA SHOULD remove the current URI from the cache when receiving a local policy server URI that is marked as "non-cacheable". This is to avoid the use of policy server URIs that are outdated.

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

The UA MUST NOT cache tokens it has received from a policy server. A token is only valid for one request.

4.4.5. Header Field Definition and Syntax

4.4.5.1. Policy-Id Header Field

The Policy-Id header field is inserted by the UAC into INVITE, UPDATE or PRACK requests (or any other request that can be used to initiate an offer/answer exchange). The Policy-Id header field identifies all policy servers the UAC has contacted for this request.

The value of a Policy-Id header field consists of a policy server URI and an optional token parameter. The token parameter contains a token the UA might have received from the policy server.

The syntax of the Policy-Id header field is described below in ABNF,

according to RFC5234 [RFC5234], as an extension to the ABNF for SIP in RFC3261 [RFC3261]:

```
  Policy-Id        = "Policy-Id" HCOLON policyURI
                     *(COMMA  policyURI)
  policyURI        = ( SIP-URI / SIPS-URI / absoluteURI )
                     [ SEMI token-param ] *( SEMI generic-param )
  token-param      = "token=" token
```

4.4.5.2. Policy-Contact Header Field

The Policy-Contact header field can be inserted by a proxy into a 488 (Not Acceptable Here) response to INVITE, UPDATE or PRACK requests (or other requests that initiate an offer/answer exchange). The value of a Policy-Contact header field consists of a policy server URI and an optional "non-cacheable" header field parameter. The policy server URI identifies the policy server that needs to be contacted by a UAC. The "non-cacheable" header field parameter indicates that the policy server URI is not intended to be cached by the UAC.

The Policy-Contact header field can also be inserted by a proxy into INVITE, UPDATE and PRACK requests (or other requests that can be used to initiate an offer/answer exchange). It contains an ordered list of policy server URIs that need to be contacted by the UAS. The topmost value of this list identifies the policy server that is contacted first. New header field values are inserted at the end. With this, the Policy-Contact header field effectively forms a fist-in-first-out queue.

The syntax of the Policy-Contact header field is described below in ABNF, according to RFC5234 [RFC5234], as an extension to the ABNF for SIP in RFC3261 [RFC3261]:

```
 Policy-Contact         = "Policy-Contact" HCOLON
                          policyContact-info *(COMMA policyContact-info)

 policyContact-info     = LAQUOT policyContact-uri RAQUOT
                          *( SEMI policyContact-param )

 policyContact-uri      = ( SIP-URI / SIPS-URI / absoluteURI )

 policyContact-param    = ( "non-cacheable" / policyContact-alt-uri
                          / generic-param )

 policyContact-alt-uri  = "alt-uri" EQUAL hostname
```

Tables 1 and 2 are extensions of Tables 2 and 3 in RFC 3261

[RFC3261]. The column "INF" is for the INFO method [RFC2976], "PRA" is for the PRACK method [RFC3262], "UPD" is for the UPDATE method [RFC3311], "SUB" is for the SUBSCRIBE method [RFC3265], "NOT" is for the NOTIFY method [RFC3265], "MSG" is for the MESSAGE method [RFC3428], "REF" is for the REFER method [RFC3515], and "PUB" is for the PUBLISH method [RFC3903].

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

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

Header field	where	proxy	PRA	PUB	SUB	NOT	INF	MSG	REF
Policy-Id	R	rd	c	-	-	-	-	-	-
Policy-Contact	R	a	c	-	-	-	-	-	-
Policy-Contact	488	a	c	-	-	-	-	-	-

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

4.5. Policy Channel

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.

The Event Package for Session-Specific Session Policies [I-D.ietf-sipping-policy-package] defines a SUBSCRIBE/NOTIFY-based [RFC3265] policy channel mechanism. A UA compliant to this specification MUST support the Event Package for Session-Specific Session Policies [I-D.ietf-sipping-policy-package]. The UA MUST use this event package to contact a policy server if the policy server URI is a SIP-URI or SIPS-URI. A UA MAY support other policy channel mechanisms.

4.5.1. Creation and Management

A UA discovers the list of policy servers relevant for a session during the initial offer/answer exchange (see Section 4.4). A UA compliant to this specification MUST set up a policy channel to each of the discovered policy server. If the UA does not want to set up a policy channel to one of the policy servers provided, the UA MUST cancel or reject a pending INVITE transaction for the session or terminate the session if it is already in progress.

A UA MUST maintain the policy channel to each discovered policy server during the lifetime of a session, unless the policy channel is closed by the policy server or the UA discovers that the policy server is no longer relevant for the session as described below.

A UAC can receive a 488 (Not Acceptable Here) response with a Policy-Contact header field containing a new policy server URI in response to a mid-dialog request. This indicates that the set of policy servers relevant for the current session has changed. If this occurs, the UAC MUST retry sending the request as if it was the first request in a dialog (i.e., without applying any policies except the policies from the local policy server). This way, the UAC will re-discover the list of policy servers for the current session. This is necessary since the UAC has no other way of knowing when to contact the newly discovered policy server relative to the existing policy servers and if any of the existing policy servers does not need to be contacted any more. The UAC MUST set up a policy channel to each new policy server. The UAC SHOULD close policy channels to policy server that are not listed any more. If the policy channel to these servers is not closed, the UAC can receive policies that do not apply to the session any more. The UAC MUST contact policy servers in the order in which they were discovered in the most recent request.

If a UAS receives a mid-dialog request with a Policy-Contact header field containing a list of policy server URIs that is different from the list of policy servers to which the UAS has currently established a policy channel, then the UAS MUST set up a policy channel to all new policy servers and contact them. The UAS SHOULD close policy channels to servers that are not listed any more. If the policy channel to these servers is not closed, the UAS can receive policies that do not apply to the session any more. The UAS MUST use policy servers in the order in which they were contained in the most recent Policy-Contact header field.

A UA MUST inform the policy server when a session is terminated (e.g., when the UA has either sent or received a BYE) via the policy channel, unless a policy server indicates via the policy channel that it does not need to be contacted at the end of the session. This enables a policy server to free all resources it has allocated for this session.

4.5.2. Contacting the Policy Server

A UA MUST contact all policy servers to which it has established a policy channel before sending or after receiving a mid-dialog request. The UA MUST contact the policy servers in the order in which they were discovered most recently.

A UA that receives a SIP message containing an offer or answer SHOULD completely process the message (e.g., according to RFC3261 [RFC3261]) before contacting the policy server. The SIP processing of the message includes, for example, updating dialog state and timers as well as creating ACK or PRACK requests as necessary. This ensures that contacting a policy server does not interfere with SIP message processing and timing (e.g., by inadvertently causing timers to expire). This implies, for example, that a UAC which has received a response to an INVITE request would normally finish the processing of the response including transmitting the ACK request before it contacts the policy server. An important exception to this rule is discussed in the next paragraph.

In some cases, a UA needs to use the offer/answer it has received in a SIP message to create an ACK or PRACK request for this message, i.e., it needs to use the offer/answer before finishing the SIP machinery for this message. For example, a UAC that has received an offer in the response to an INVITE request needs to apply policies to the offer and the answer before it can send the answer in an ACK request. In these cases, a UA SHOULD contact the policy server even if this is during the processing of a SIP message. This implies that a UA, which has received an offer in the response of an INVITE request, would normally contact the policy server and apply session policies before sending the answer in the ACK request.

> Note: this assumes that the policy server can always respond immediately to a policy request and does not require manual intervention to create a policy. This will be the case for most policy servers. If, however, a policy server cannot respond with a policy right away, it can return a policy that temporarily denies the session and update this policy as the actual policy decision becomes available. A delay in the response from the policy server to the UA would delay the transmission of the ACK request and could trigger retransmissions of the INVITE response (also see the recommendations for Flow I in RFC3725 [RFC3725]).

The case of multiple policy servers providing policies to the same UA requires additional considerations. A policy returned by one policy server can contain information that needs to be shared with the other policy servers. For example, two policy servers can have the policy to insert a media intermediary by modifying the IP addresses and ports of media streams. In order for media streams to pass through both intermediaries, each intermediary needs to know the IP address and port on which the other media intermediary is expecting the stream to arrive. If media streams are flowing in both directions, this means that each intermediary needs to know IP addresses and ports of the other intermediary.

UACs usually contact a policy server twice during an offer/answer exchange (unless a policy server indicates that it only needs to be contacted once). Therefore the case of multiple policy servers providing policies to a single UAC does not require additional steps in most cases. However, a UAS usually contacts each policy server only once (see Figure 4). If a session policy returned by one of the policy servers requires that information is shared between multiple servers and the UAS receives policies from more than one policy server, then the UAS MUST contact all policy servers a second time after contacting all servers the first time. Whether or not a second round is required is determined by the type of information returned by the policy server. A data format for session policies (e.g., [I-D.ietf-sipping-media-policy-dataset]) needs to explicitly state if a second round is needed for a particular data element. If a UA receives such an element, it knows that is expected to contact policy servers a second time. If such a data element is modified during a mid-call offer/answer exchange and multiple policy servers are providing policies to a UA then all UAs MUST contact policy servers in a first and second round. An example call flow is shown in Appendix B.3.

A UA that supports session-specific policies compliant to this specification MUST support the User Agent Profile Data Set for Media Policy [I-D.ietf-sipping-media-policy-dataset] as data format for session policies.

4.5.3. Using Session Policies

A UA MUST disclose the session description(s) for the current session to policy servers through the policy channel. The UA MUST apply session policies it receives to the offer and, if one is received, to the answer before using the offer/answer. If these policies are unacceptable, the UA MUST NOT continue with the session. This means that, the UA MUST cancel or reject a pending INVITE transaction for the session or terminate the session if it is already in progress. If the UA receives an unacceptable policy in an INVITE response, the UA MUST complete the INVITE transaction and then terminate the session.

When a UA receives a notification about a change in the current policies, the UA MUST apply the updated policies to the current session or the UA MUST terminate the session. If the policy update causes a change in the session description of a session, then the UA needs to re-negotiate the modified session description with its peer UA, for example, using a re-INVITE or UPDATE request. 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

Policy enforcement mechanisms can prevent a UA from communicating with another UA if the UAs are not aware of the policies that are enforced. Policy enforcement mechanisms without policy signaling can therefore create a denial of service condition for UAs. This specification provides a mechanism for intermediaries to signal the policies that are enforced to UAs. It enables UAs to establish sessions that are conform and pass through policy enforcement.

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

A UA transmits session information to a policy server for session-specific policies. This session information can contain sensitive data the user does not want an eavesdropper or an unauthorized policy server to see. Vice versa, session policies can contain sensitive information about the network or service level agreements the service provider does not want to disclose to an eavesdropper or an unauthorized UA.

It is important to secure the communication between the proxy and the UA (for session-specific policies) as well as the UA and the policy server. The following four discrete attributes need to be protected:

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

To protect the integrity of the policy server URI, a UA SHOULD use a secured transport protocol such as TLS [RFC5246] between proxies and the UA. Protecting the integrity of the policy server URI is important since an attacker could intercept SIP messages between the UA and the proxy and remove the policy header fields 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, can prevent a UA from setting up a session.

Instead of removing a policy server URI, an attacker can also modify the policy server URI and point the UA to a compromised policy server. It is RECOMMENDED that a UA authenticates policy servers to prevent such an attack from being effective.

It is RECOMMENDED that the UA only accepts session-independent policies from trustworthy policy servers as these policies affect all sessions of a UA. A list of trustworthy session-independent policy servers can be provided to the UA through configuration. As SIP messages can be affected by any proxy on a path and session-specific policies only apply to a single session, a UA MAY choose to accept session-specific policies from other policy servers as well.

Policy servers SHOULD authenticate UAs to protect the information that is contained in a session policy. However, a policy server can also 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.

It is RECOMMENDED that administrators use SIPS URIs as policy server URIs so that subscriptions to session policies are transmitted over TLS.

The above security attributes are important to protect the communication between the UA and policy server. This document does not define the protocol used for the communication between UA and policy server and merely refers to other specifications for this purpose. The security considerations of these specifications need to address the above security aspects.

6. IANA Considerations

6.1. Registration of the "Policy-Id" Header Field

Name of Header Field: Policy-Id

Short form: none

Normative description: Section 4.4.5 of this document

6.2. Registration of the "Policy-Contact" Header Field

Name of Header Field: Policy-Contact

Short form: none

Normative description: Section 4.4.5 of this document

6.3. Registration of the "non-cacheable" Policy-Contact Header Field Parameter

Registry Name: Header Field Parameters and Parameter Values
Reference: RFC3968 [RFC3968]

Registry:

Header Field	Parameter Name	Predefined Values	Reference
Policy-Contact	non-cacheable	Yes	this document

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

This specification registers a new SIP option tag, as per the guidelines in Section 27.1 of RFC3261 [RFC3261].

This document defines the SIP option tag "policy".

The following row has been added to the "Option Tags" section of the SIP Parameter Registry:

Name	Description	Reference
policy	This option tag is used to indicate that a UA can process policy server URIs for and subscribe to session-specific policies.	this document

Name of option: policy

Description: Support for the Policy-Contact and Policy-Id header fields.

SIP header fields defined: Policy-Contact, Policy-Id

Normative description: This document

7. References

7.1. Normative References

[I-D.ietf-sipping-config-framework]
Channabasappa, S., "A Framework for Session Initiation Protocol User Agent Profile Delivery", draft-ietf-sipping-config-framework-18 (work in progress), October 2010.

[I-D.ietf-sipping-media-policy-dataset]
Hilt, V., Worley, D., Camarillo, G., and J. Rosenberg, "A User Agent Profile Data Set for Media Policy", draft-ietf-sipping-media-policy-dataset-11 (work in progress), February 2011.

[I-D.ietf-sipping-policy-package]
Hilt, V. and G. Camarillo, "A Session Initiation Protocol (SIP) Event Package for Session-Specific Session Policies.", draft-ietf-sipping-policy-package-08 (work in progress), March 2010.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3261] 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.

[RFC3262] Rosenberg, J. and H. Schulzrinne, "Reliability of Provisional Responses in Session Initiation Protocol (SIP)", RFC 3262, June 2002.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3265] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[RFC3311] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.

[RFC3968] Camarillo, G., "The Internet Assigned Number Authority

(IANA) Header Field Parameter Registry for the Session Initiation Protocol (SIP)", BCP 98, RFC 3968, December 2004.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

7.2. Informative References

[RFC2976] Donovan, S., "The SIP INFO Method", RFC 2976, October 2000.

[RFC3428] Campbell, B., Rosenberg, J., Schulzrinne, H., Huitema, C., and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002.

[RFC3515] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

[RFC3725] 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.

[RFC3903] Niemi, A., "Session Initiation Protocol (SIP) Extension for Event State Publication", RFC 3903, October 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

Appendix A. Acknowledgements

Many thanks to Allison Mankin, Andrew Allen, Cullen Jennings and Vijay Gurbani for their contributions to this draft. Many thanks to Roni Even, Bob Penfield, Mary Barnes, Shida Schubert, Keith Drage, Lisa Dusseault, Tim Polk and Pasi Eronen for their reviews and suggestions.

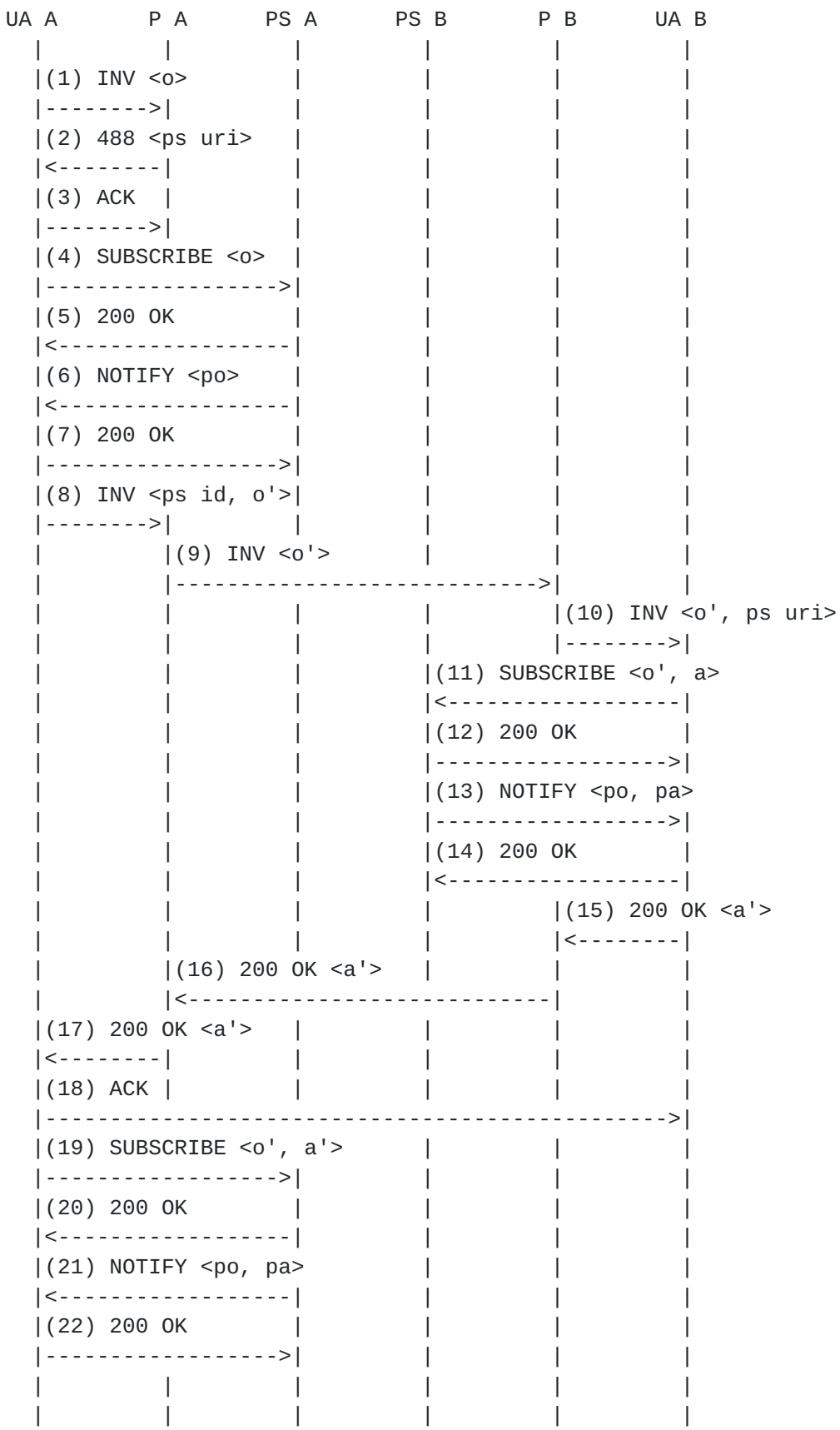
Appendix B. Session-Specific Policies - Call Flows

The following call flows illustrate the overall operation of session-specific policies including the policy channel protocol as defined in the SIP Event Package for Session-Specific Session Policies [I-D.ietf-sipping-policy-package].

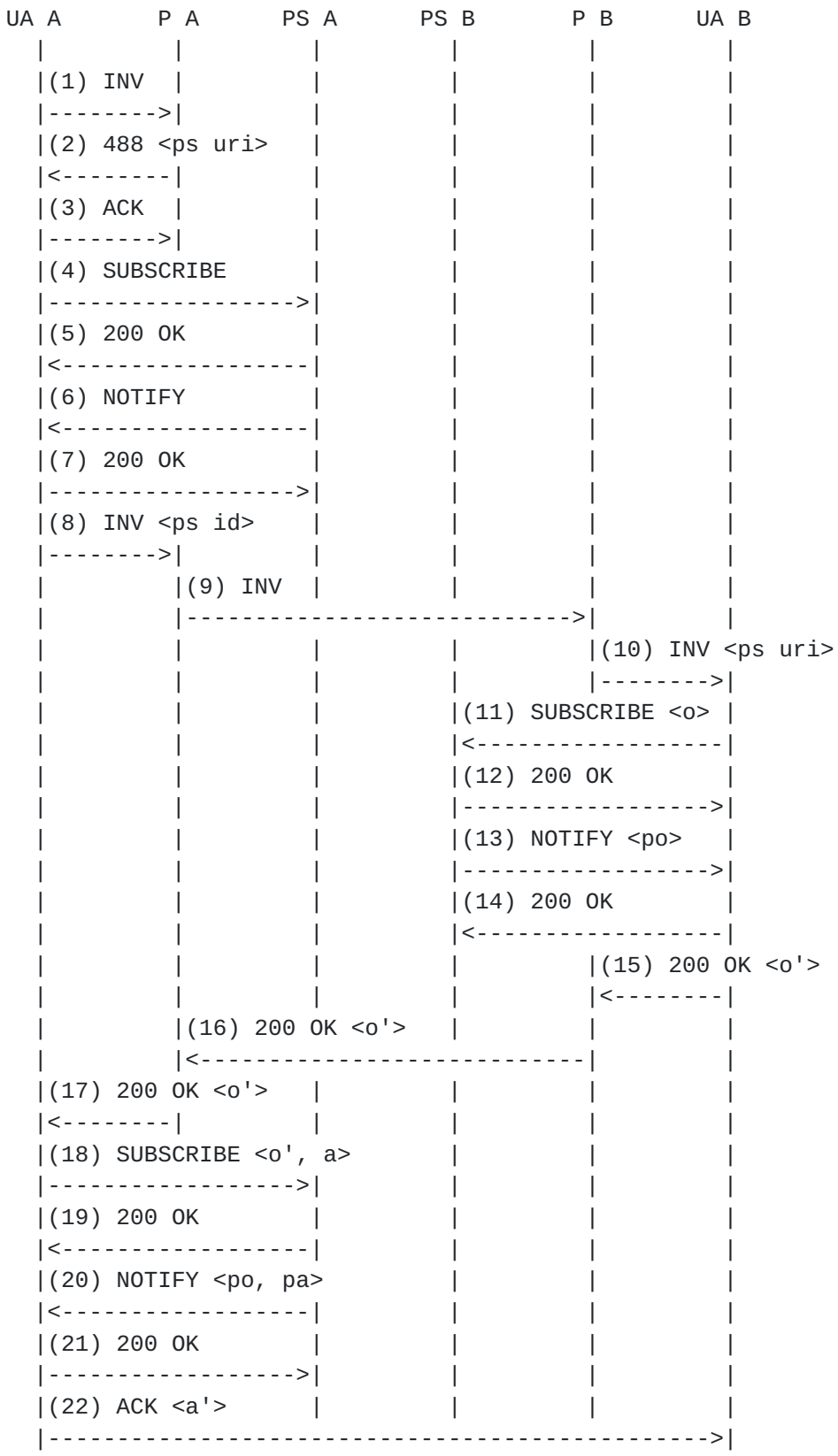
The following abbreviations are used:

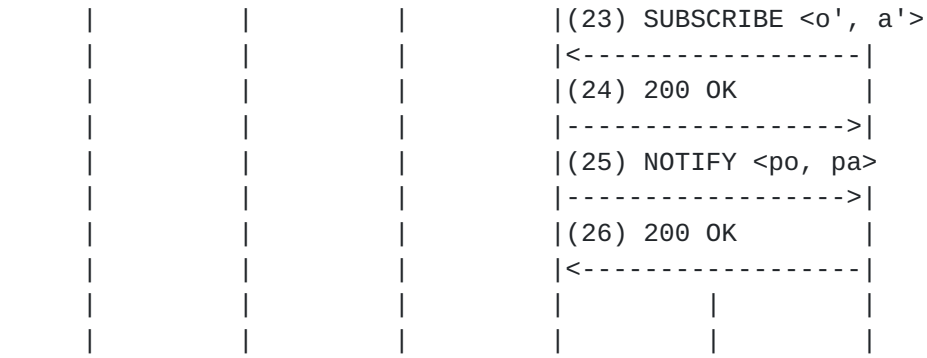
o: offer
o': offer modified by a policy
po: offer policy
a: answer
a': answer modified by a policy
pa: answer policy
ps uri: policy server URI (in Policy-Contact header field)
ps id: policy server id (in Policy-Id header field)

B.1. Offer in Invite

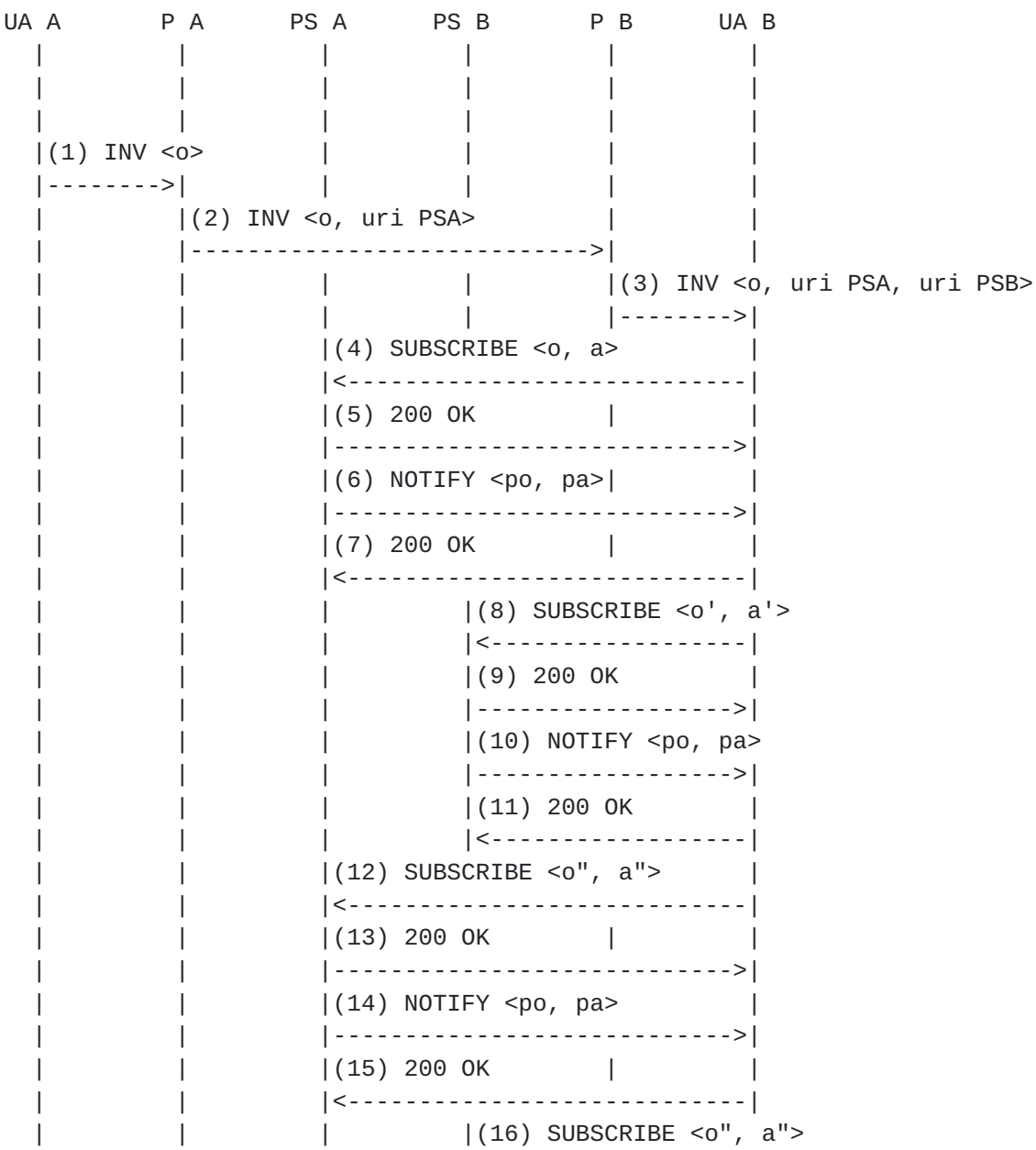

UA A
P A
PS A
PS B
P B
UA B
(1) INV <o>
(2) 488 <ps uri>
(3) ACK
(4) SUBSCRIBE <o>
(5) 200 OK
(6) NOTIFY <po>
(7) 200 OK
(8) INV <ps id, o'>
(9) INV <o'>
(10) INV <o', ps uri>
(11) SUBSCRIBE <o', a>
(12) 200 OK
(13) NOTIFY <po, pa>
(14) 200 OK
(15) 200 OK <a'>
(16) 200 OK <a'>
(17) 200 OK <a'>
(18) ACK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

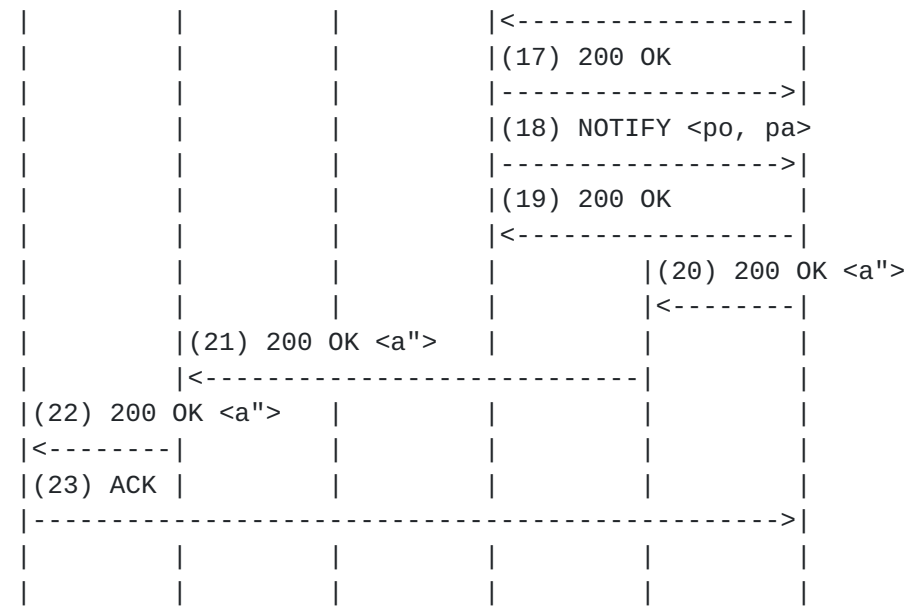
B.2. Offer in Response





B.3. Multiple Policy Servers for UAS





Authors' Addresses

Volker Hilt
Bell Labs/Alcatel-Lucent
791 Holmdel-Keyport 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
jdrosen.net
Monmouth, NJ
USA

Email: jdrosen@jdrosen.net