Session Initiation Proposal
Investigation Working Group
Internet-Draft
Expires: January 8, 2005

V. Hilt
Bell Labs/Lucent Technologies
G. Camarillo
Ericsson

July 10, 2004

Evaluating Scenarios for Session-specific Policies
draft-hilt-sipping-policy-scenarios-00

Status of this Memo

This Internet-Draft will expire on January 8, 2005.

Copyright Notice

Abstract

This draft describes detailed call flows for different use cases of

session-specific policies. It compares the two approaches that are currently being discussed for session-specific policies, namely the piggyback model and the separate channel model.

Table of Contents

1. Introduction

The concept of session-specific SIP session policies [3] has been around for some time. However, it has proven that the mechanisms for establishing session-specific policies are non-trivial and most likely require to sacrifice some of the requirements defined in [5].

In this draft, we compare two approaches that have been proposed for session-specific policies: the piggyback model and the separate channel model. We analyze detailed call flows of use cases for both models and discuss advantages and drawbacks of each model.

The main purpose of this draft is to spark the discussion about the two models and to come to a conclusion on which if the models is the most appropriate approach for 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, [1] and indicate requirement levels for compliant implementations.

3. Scenario

All use cases in the subsequent sections are based on the following scenario (see Figure 1). The user agent UA A is registered at proxy P A, which is responsible for domain A. UA B is registered at P B in domain B. Both domains A and B are separate and they are connected through a transit network.

It is assumed that user agent and proxy of each domain have a relationship (e.g. UA A is a customer of provider running domain A). It is also assumed that the entities in different domains do not necessarily have a relationship. This corresponds to a scenario where a customer of one provider is establishing a session with a customer of another provider. As a consequence, entities in one domain can't make any assumptions about the capabilities of entities in the other domain. In particular, it can't be assumed that session policies are

supported in the other domain. Additionally, it is assumed that entities in one domain are not willing to disclose network internals such as session policies to the other domain.

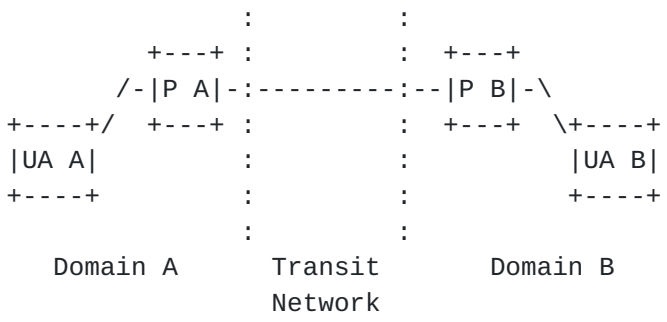


Figure 1

4. Use Cases

4.1 NAT Traversal

In this scenario, each domain is connected to the public Internet through a NAT. UA A and UA B have local, non-routable addresses. The proxies P A and P B implement MIDCOM [6] agents an control an associated NAT that connects their domain to the Internet.

Session policies are needed to accomplish the following tasks for NAT traversal:

- Enable proxies to examine the media addresses and ports in the session description created by its associated UA (can either be an offer or an answer). This information is needed to configure NAT rules for incoming media traffic.
- Enable proxies to modify the media addresses and ports in the session description created by its associated UA (offer or answer). The modification is needed to replace the local addresses with globally routable addresses at which the associated UA is reachable from outside.
- Enable a proxy to examine the media addresses and ports in the session description created by the remote UA (offer or answer). This information is needed to configure NAT rules for outgoing media traffic.

4.1.1 Piggyback Model

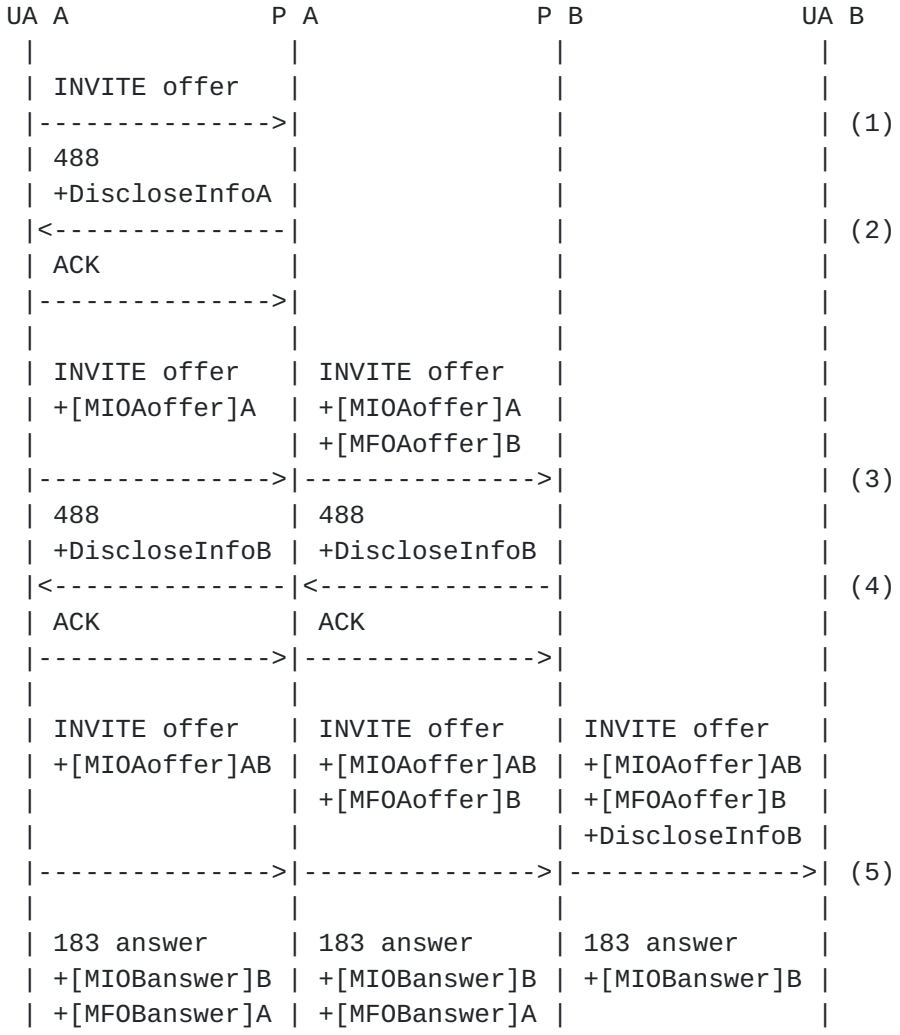
In the piggyback model, session policies are piggybacked on the SIP messages used for the corresponding SDP offer/answer exchange.

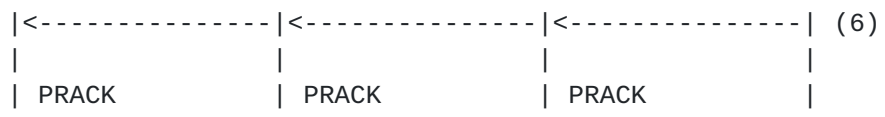
4.1.1.1 Offer in Request - Alternative 1

The call flow in Figure 2 describes the piggyback model for INVITE

requests carrying a session description offer. This alternative is based on encryption to protect MIOs and MFOs from being inspected by unauthorized network entities (e.g. in the transit network). It corresponds to the piggyback model that has been discussed so far (e.g. in [3])

It is important to note that this alternative still requires that the UAs on both sides support session-specific policies, even if policies are only used in one domain. In other words, to enable the use of policies between UA A and P A in domain A, UA B in domain B also needs to support policies, even if policies are not used in this domain. Furthermore, encryption can only protect policies from being inspected in the transit network. Entities in both domains must be able to inspect the policies of the other domain.




|<----------------|<----------------|<----------------| (6)
| | | |
| PRACK | PRACK | PRACK |

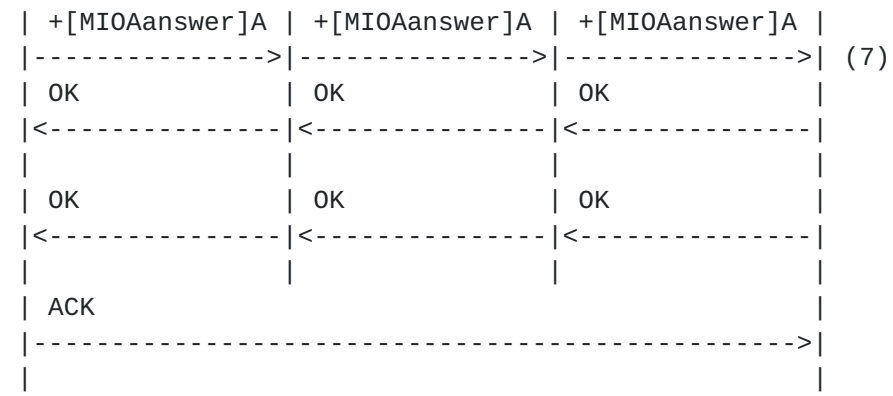


Figure 2

Steps (1) and (2) are needed if P A detects that UA A does not disclose the required aspects of its session description offer in a Media Interface Object A (MIOAoffer). In this case, P A returns a 488 response that requests the disclosure of these aspects. This steps could be avoided, for example, by providing information about what to disclose as part of the device configuration [4].

In step (3) UA A creates Media Interface Object A (MIOAoffer) that discloses the IP addresses and ports it has used in the offer. UA A encrypts MIOAoffer with a key known to P A ([MIOAoffer]A). P A can now perform its MIDCOM functionalities based on the data in MIOAoffer and creates a Media Filter Object for MIOAoffer (MFOAoffer), which contains the external addresses and ports UA B must use to reach UA A. P A encrypts MFOAoffer with a key known to UA B.

In step (4) P B returns a 488 response and asks UA A to disclose the addresses and ports used in the offer. It also asks P A to disclose all policies that affect the addresses and ports in the offer, since these are the addresses and ports that will later be used in the session.

Step (5) is analogous to step (3) except that MIOAoffer and MFOAoffer are now encrypted with a keys known to P B and UA B. Finally, P B asks UA B to disclose the addresses and ports it is going to use in the answer.

In step (6) UA B has accepted the policies contained in MFOAoffer. It

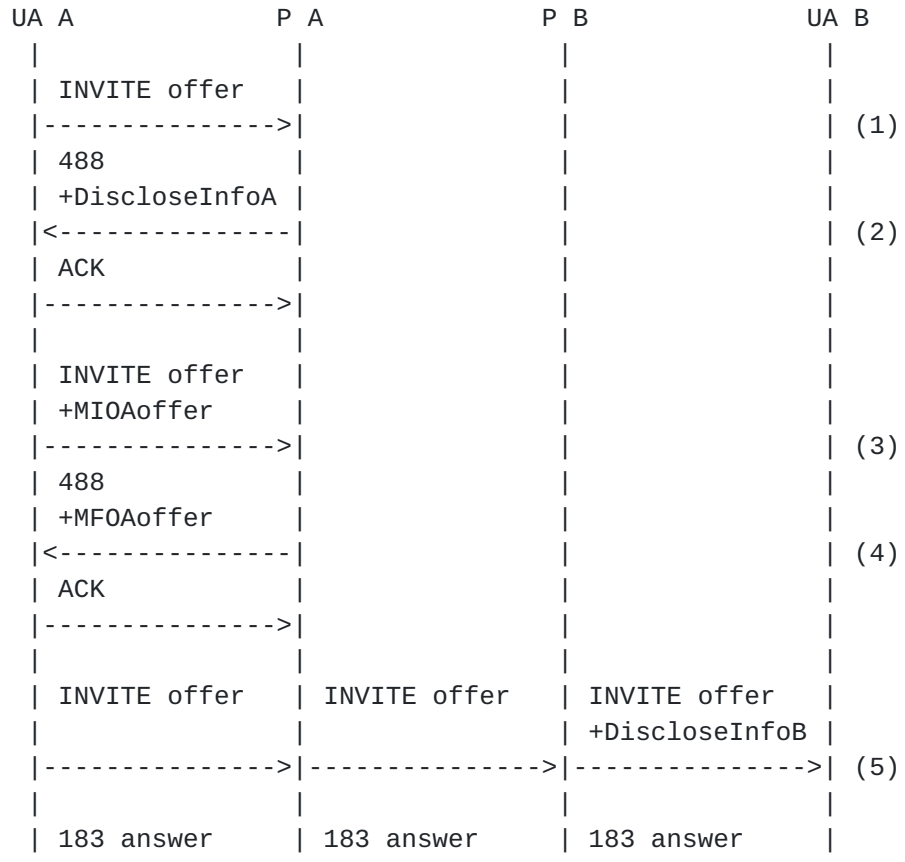
creates a 183 response with its session description answer and a MIOBanswer containing the local IP addresses and ports. UA B encrypts MIOBanswer with a key known to P B. The use of a 183 response instead of a 200 OK later enables UA A to cancel the INVITE transaction if it decides not to accept the requested policies before the INVITE transaction is completed.

P B examines the addresses and ports in MIOBanswer and inserts MFOBanswer containing the external addresses and ports to be used with the session description answer. It encrypts MFOBanswer with a key known to UA A.

In step (7) UA A accepts the policies in MFOBanswer and creates a PRACK. It inserts a MIOAanswer, which contains the addresses and ports it is using to send media to UA B. UA A encrypts MIOAanswer with a key known to P A. Since P A has no policies for the answer, no additional MFOs are needed.

4.1.1.2 Offer in Request - Alternative 2

The call flow in Figure 3 also piggybacks policy information on messages exchanged within a SIP INVITE transaction. In this call flow, these messages are used to exchange policies between UA and proxy. The flow ensures that policy information does not leave the local domain by rejecting messages and removing policy headers.




+MIOBoffer
+MIOBanswer
(6)

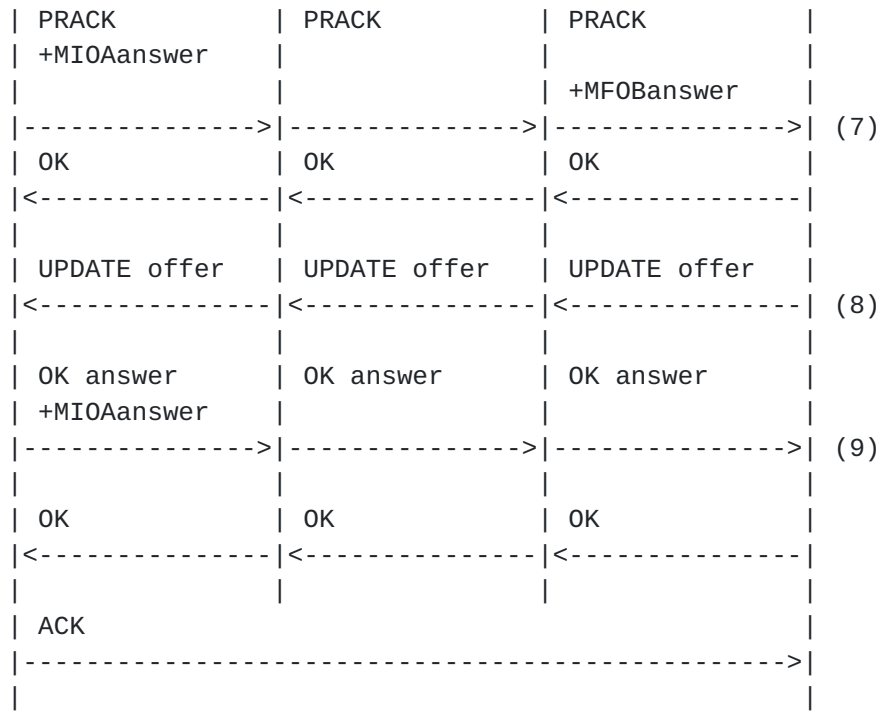


Figure 3

The basic idea of exchanging MIOs and MFOs is the same as in the above flow. Steps (1) - (3) are identical. In step (4) P A returns a MFOAoffer containing the modified addresses and ports for the offer to UA A. UA A can now apply these policies and create a new offer in step (5).

In step (5) P B also requests the disclosure of the addresses used in the offer and answer and receives them from UA B in step (6). Since UA B has not received policies from P B yet, the answer in step (6) is a dummy answer that needs to be updated later.

In step (7) UA A creates a PRACK containing a MIOAanswer which is still based on the dummy answer. P B uses this PRACK message to transmit the addresses and ports it wants UA B to use in its session description to UA B. To make these addresses and ports known to UA A, UA B creates an new offer and sends an UPDATE in step (8) to which UA A responds in step (9). UA A also creates a new MIOAanswer for P A that is now based on the actual session description used in the session.

4.1.1.3 Offer in Response

The piggyback model call flows for INVITEs that carry the session description offer in the response are analogous to the above call

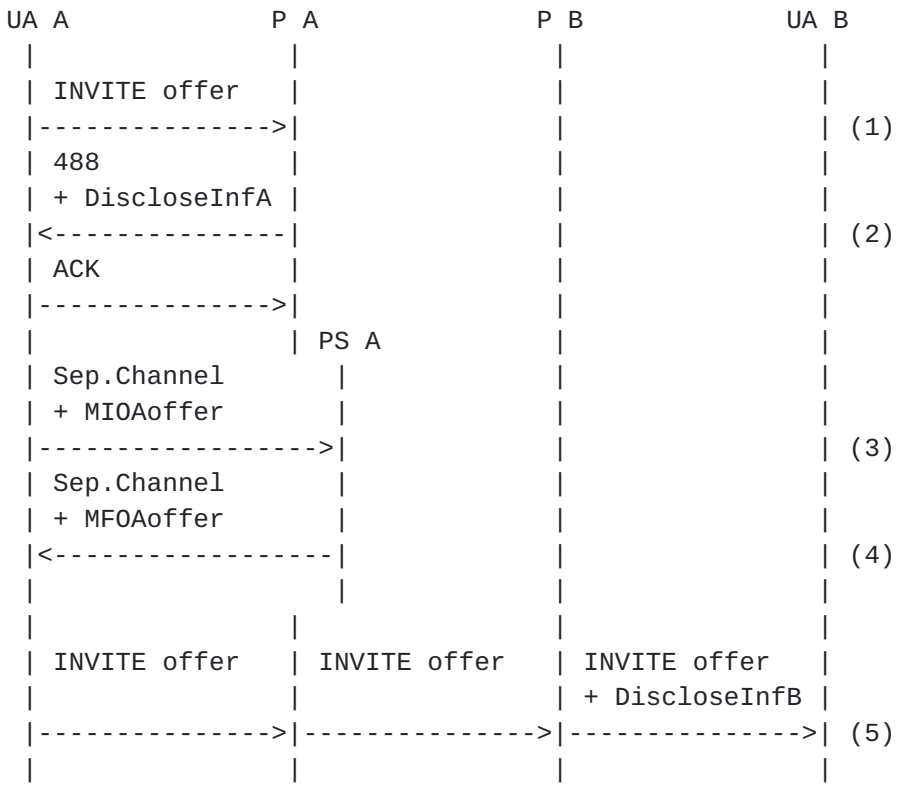
flows. However, these flow are generally more complex that the flows described above for the offer in request scenario.

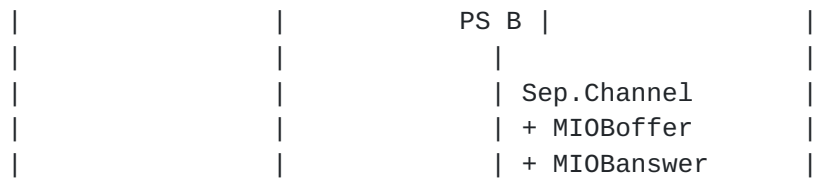
4.1.2 Separate Channel Model

The idea behind the Separate Channel Model is that user agents retrieve session-specific policies through a separate channel before they create the session description offer/answer. The channel can be implemented in different ways, based on SIP or on another protocol. In this document we simply make the assumption that this channel enables a UA to send a MIO to the policy server and to retrieve a MFO as a response.

4.1.2.1 Offer in Request

The call flow in Figure 4 depicts the separate channel model for INVITE requests carrying a session description offer. PS A and PS B are the policy servers in the respective domains. They can be co-located with the proxies P A and P B but do not have to be.




PS B
Sep.Channel
+ MIOBoffer
+ MIOBanswer

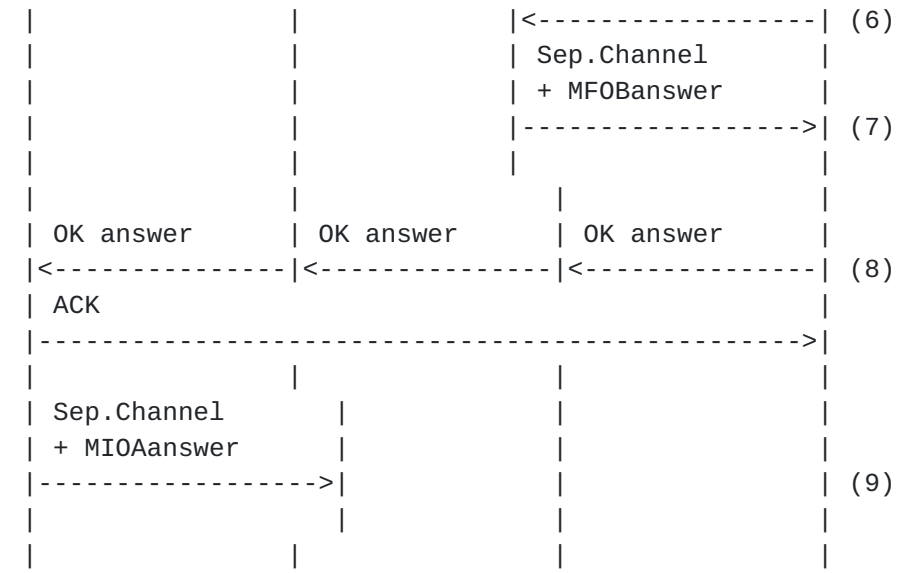


Figure 4

Steps (1) and (2) are needed if P A detects that UA A has not requested policies for the current session before creating the SDP offer. In this case, P A returns a 488 response that contains the address to which UA A should establish a channel to and information about what should be disclosed in an MIO. These steps can be avoided, for example, by providing the information about what to disclosure to where as part of the device configuration.

In step (3) UA A establishes a channel to PS A and submits a MIOAoffer in which it reveals the addresses and ports it is going to use in the offer. PS A uses this information in its function as MIDCOM agent and returns the addresses and ports UA A should include in its offer in an MFOAoffer in step (4).

In step (5) UA A decides to accept the policies in MFOAoffer and creates the offer using the given addresses and ports. P B inserts disclosure information for UA B into this message.

Before creating an answer, UA B retrieves the policies that apply to this session by establishing a channel to its policy server in step (6). It submits the addresses and ports from the offer in MIOBoffer and the addresses and ports it is going to use in its answer in MIOBanswer. PS B returns the addresses and ports to be used in the

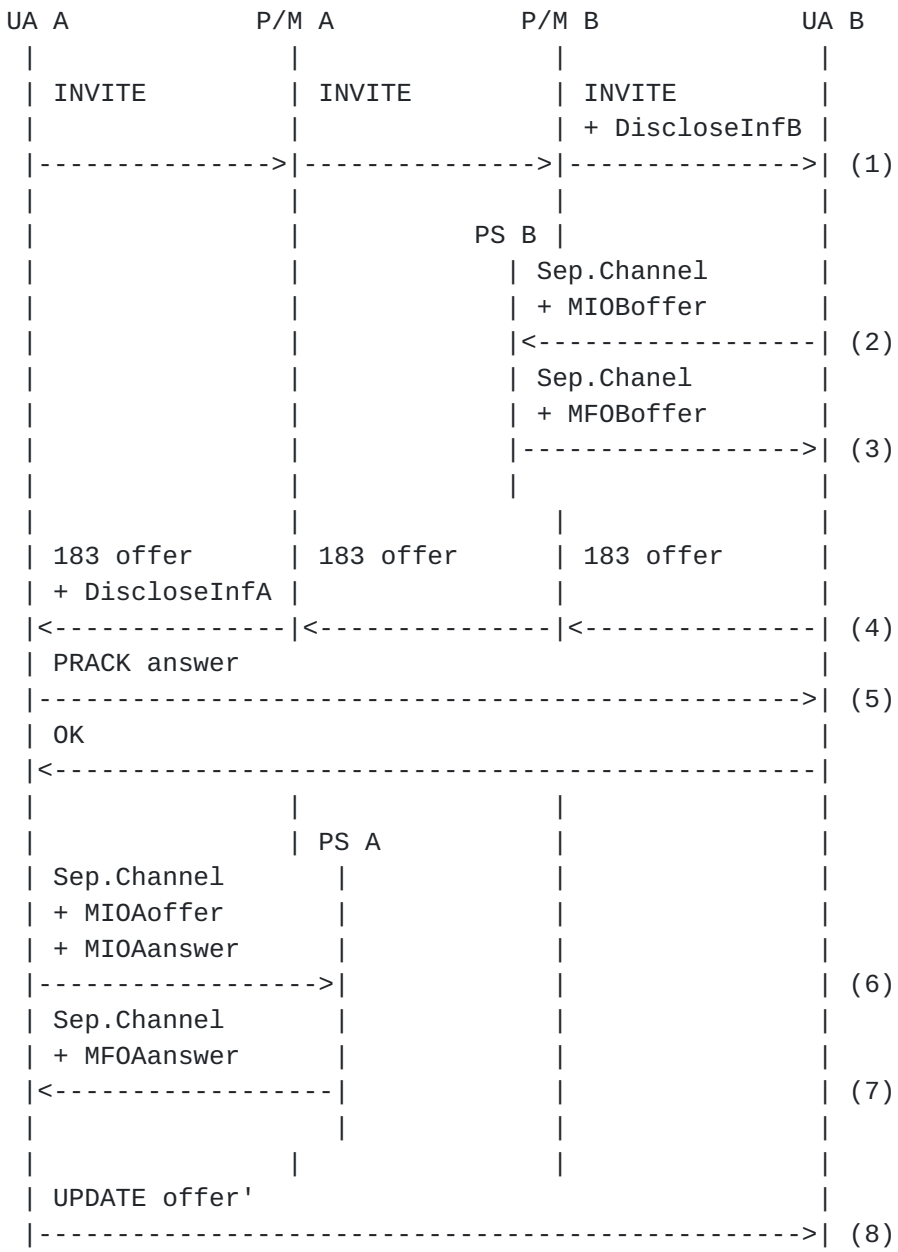
answer in MFOBanswer in step (7). If UA B decides to accept these policies, it creates an answer in step (8). If not, UA B can return a final response rejecting the INVITE.

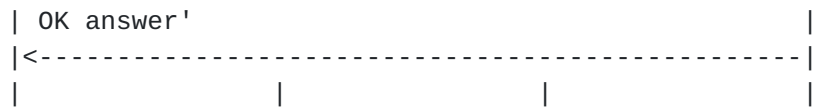
In step (9), UA A submits MIOAanswer to the local policy server

disclosing the addresses and ports received in the answer from UA B.

4.1.2.2 Offer in Response

The call flow for an INVITE carrying the offer in the response is depicted in Figure 5. In contrast to call flow Figure 4, UA A has to wait until it receives an offer from UA B before it can retrieve the policies for the current session.




| OK answer' |
|<----------|
| | | |

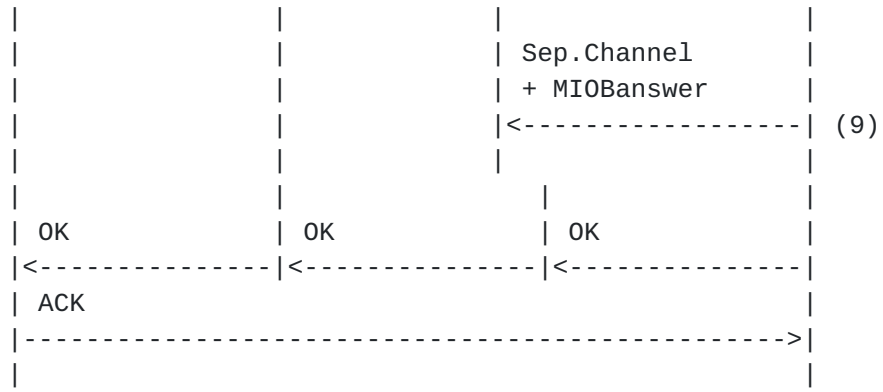


Figure 5

After receiving the 183 response in step (4), UA A must respond immediately with a PRACK to avoid the expiration of timer T1 in UA B and the retransmission of the 183. UA A therefore creates a PRACK with an answer that does not yet consider session-specific policies. It then retrieves the policies for the current session in steps (6) and (7) in which it gets the external addresses and ports from PS A in MFOAanswer. It creates a new offer and sends it to UA B in the UPDATE shown in step (7).

ISSUE: If it can be assumed that UA A and the policy server are located in the same network, there might be enough time for UA A to retrieve policies before generating the PRACK. The sequence of steps would then be (1)-(3),(5)-(6),(4) without a need for the UPDATE in step (7). Is this a reasonable assumption?

4.2 Codec Selection

In this scenario, session-specific policies are used to limit the set of codecs a UA can use. By using session-specific policies, a network provider does not need to reveal the list of allowed codecs to the UA. Instead it can limit the use of certain codecs only if endpoints announce them in an SDP description.

Session policies are needed to accomplish the following tasks for codec selection:

- Enable a proxy to examine the codecs listed in the session description offer (independent of whether the offer was created by the local or the remote UA).
- Enable proxies to remove codecs from the offer (independent of whether the offer was created by the local or the remote UA).

The call flows for both models are analogous to the NAT scenario,

with the difference that the policy servers do not not provide policies for the answer. Instead, they both provide policies for the session description offer. Also, MIOs contain lists of codecs and MFOs identify those codecs that should not be used.

5. Discussion

5.1 Disclosure of Session Descriptions and Policies

In the piggyback model (alternative 1), all MIOs and MFOs travel through the network. End-to-middle and middle-to-end encryption can be used to prevent unauthorized network entities from examining them. However, even with encryption, UAs need to disclose MIOs to all policy-enabled proxies even if they are located in remote networks. Moreover, proxies must disclose their policies to UAs in remote networks and to other proxies that are interested in examining or modifying the same aspect of a session description.

In the piggyback model (alternative 2), the MIOs and MFOs are piggybacked on messaged which are destined at entities outside of the local network. By rejecting messages and removing headers, the proxies keep the MIOs and MFOs within the local network. End-to-middle and middle-to-end encryption can be used to further protect the MIOs and MFOs so that they can't be examined by unauthorized entities even if these packets accidentally leave the local network.

In the separate channel model, UAs exchange MIOs and MFOs on a separate channel directly with the policy server. UAs can therefore disclose different aspects of a session description to each server. Each server can return policies directly to the UA. End-to-end encryption can be used to secure these transmissions. If UA and the policy server are in the same network, the MIOs and MFOs never exit that network.

5.2 UA Support of Policies

In the piggyback model (alternative 1) both UAs need to support policies, even if they are only used in one of the domains.

In the piggyback model (alternative 2) and the separate channel model, it is sufficient if one of the UAs supports policies.

5.3 Re-Use of Document Formats and Mechanisms

The piggyback model (both alternatives) requires that proxy servers insert MFOs into SIP messages. The current standards require the use of headers for this purpose, since a proxy is not allowed to add body

elements to a message. As a consequence, standard document formats that could be used in MIME bodies can't be used for MFOs in the piggyback model. In addition, S/MIME encryption doesn't apply.

In the separate channel model, MIOs and MFOs are exchanged over a separate channel which is potentially able to carry arbitrary documents. This enables the use of existing document formats for MIOs and MFOs and the use of encryption. In particular, the document formats that are defined for session-independent policies [2] can be re-used for session-specific policies. This greatly simplifies UAs which support both types of policies.

5.4 Asynchronous Policies

Some scenarios require that a policy server can update the session policies at any time for ongoing sessions.

In the piggyback model (both alternatives), the exchange of policies is tied to UA initiated offer/answer exchanges of session descriptions (i.e. INVITE, re-INVITE or UPDATE). For this reason, a proxy can't introduce new policies at arbitrary times during a session.

In the separate channel model, the policy server can send updates for the current policy at any time, independent of messages exchanged between the UAs.

5.5 Separation of Tasks

It is generally desirable to develop separate solutions for different tasks. In the piggyback model (both alternatives), the task of exchanging MIOs and MFOs between UA and policy server is coupled to the task of exchanging the offer/answer between UAC and UAS. This increases the complexity of call flows, in particular if the transmission of MIO/MFOs is spread across different SIP transactions, and leads lower re-usability of solutions for each task.

The separate channel model provides a clear separation of tasks.

6 References

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

[2] Hilt, V., Camarillo, G. and J. Rosenberg, "Session-Independent Policies for the Session Initiation Protocol (SIP)", draft-hilt-sipping-session-indep-policy-01 (work in progress), May 2004.

[3] Hilt, V. and J. Rosenberg, "A Framework for Session-Specific Intermediary Session Policies in SIP", draft-hilt-sipping-session-spec-policy-00 (work in progress), September 2003.

[4] Petrie, D., "A Framework for Session Initiation Protocol User Agent Profile Delivery", draft-ietf-sipping-config-framework-03 (work in progress), May 2004.

[5] Rosenberg, J., "Requirements for Session Policy for the Session Initiation Protocol (SIP)", draft-ietf-sipping-session-policy-req-01 (work in progress), February 2004.

[6] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A. and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, August 2002.

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

Appendix A. Acknowledgements

Many thanks to Jonathan Rosenberg and Allison Mankin.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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