SIPCORE Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 24, 2011

C. Holmberg
Ericsson

January 20, 2011

Indication of support for keep-alive
draft-ietf-sipcore-keep-12.txt

Abstract

This specification defines a new Session Initiation Protocol (SIP) Via header field parameter, "keep", which allows adjacent SIP entities to explicitly negotiate usage of the Network Address Translation (NAT) keep-alive mechanisms defined in SIP Outbound, in cases where SIP Outbound is not supported, cannot be applied, or where usage of keep-alives is not implicitly negotiated as part of the SIP Outbound negotiation.

Status of this Memo

This Internet-Draft is submitted to IETF 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 July 24, 2011.

Copyright Notice

Table of Contents

1. Introduction

Section 3.5 of SIP Outbound [RFC5626] defines two keep-alive mechanisms. Even though the keep-alive mechanisms are separated from the rest of the SIP Outbound mechanism, SIP Outbound does not define a mechanism to explicitly negotiate usage of the keep-alive mechanisms. In some cases usage of keep-alives can be implicitly negotiated as part of the SIP Outbound negotiation.

However, there are SIP Outbound use-cases where usage of keep-alives is not implicitly negotiated as part of the SIP Outbound negotiation. In addition, there are cases where SIP Outbound is not supported, or where it cannot be applied, but where there is still a need to be able to negotiate usage of keep-alives. Last, SIP Outbound only allows keep-alives to be negotiated between a UA and an edge proxy, and not between other SIP entities.

This specification defines a new Session Initiation Protocol (SIP) [RFC3261] Via header field parameter, "keep", which allows adjacent SIP entities to explicitly negotiate usage of the NAT keep-alive mechanisms defined in SIP Outbound. The "keep" parameter allows SIP entities to indicate willingness to send keep-alives, to indicate willingness to receive keep-alives, and for SIP entities willing to receive keep-alives to provide a recommended keep-alive frequency.

The following sections describe use-cases where a mechanism to explicitly negotiate usage of keep-alives is needed.

1.1. Use-case: Dialog from non-registered UAs

In some cases a User Agent Client (UAC) does not register itself before it establishes a dialog, but in order to maintain NAT bindings open during the lifetime of the dialog it still needs to be able to negotiate sending of keep-alives towards its adjacent downstream SIP entity. A typical example is an emergency call, where a registration is not always required in order to make the call.

1.2. Use-case: SIP Outbound not supported

In some cases some SIP entities that need to be able to negotiate the use of keep-alives might not support SIP Outbound. However, they might still support the keep-alive mechanisms defined in SIP Outbound, and need to be able to negotiate usage of them.

1.3. Use-case: SIP dialog initiated Outbound flows

SIP Outbound allows the establishment of flows using the initial request for a dialog. As specified in RFC 5626 [RFC5626], usage of

keep-alives is not implicitly negotiated for such flows.

2. Conventions

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 BCP 14, RFC 2119 [RFC2119].

3. Definitions

Edge proxy: As defined in RFC 5626, a SIP proxy that is located topologically between the registering User Agent (UA) and the Authoritative Proxy.

NOTE: In some deployments the edge proxy might physically be located in the same SIP entity as the Authoritative Proxy.

Keep-alives: The keep-alive messages defined in RFC 5626.

"keep" parameter: A SIP Via header field parameter that a SIP entity can insert in the topmost Via header field that it adds to the request, to explicitly indicate willingness to send keep-alives towards its adjacent downstream SIP entity. A SIP entity can add a parameter value to the "keep" parameter in a response to explicitly indicate willingness to receive keep-alives from its adjacent upstream SIP entity.

SIP entity: SIP User Agent (UA), or proxy, as defined in RFC 3261.

Adjacent downstream SIP entity: The adjacent SIP entity in the direction towards which a SIP request is sent.

Adjacent upstream SIP entity: The adjacent SIP entity in the direction from which a SIP request is received.

4. User Agent and Proxy behavior

4.1. General

This section describes how SIP UAs and proxies negotiate usage of keep-alives associated with a registration, or a dialog, which types of SIP requests can be used in order to negotiate the usage, and the lifetime of the negotiated keep-alives.

SIP entities indicate willingness to send keep-alives towards the adjacent downstream SIP entity using SIP requests. The associated responses are used by SIP entities to indicate willingness to receive keep-alives. SIP entities that indicate willingness to receive keep-alives can provide a recommended keep-alive frequency.

The procedures to negotiate usage of keep-alives are identical for SIP UAs and proxies.

In general, it can be useful for SIP entities to indicate willingness to send keep-alives, even if they are not aware of any necessity for them to send keep-alives, since the adjacent downstream SIP entity might have knowledge about the necessity. Similarly, if the adjacent upstream SIP entity has indicated willingness to send keep-alives, it can be useful for SIP entities to indicate willingness to receive keep-alives, even if they are not aware of any necessity for the adjacent upstream SIP entity to send them.

NOTE: Usage of keep-alives is negotiated per direction. If a SIP entity has indicated willingness to receive keep-alives from an adjacent SIP entity, sending of keep-alives towards that adjacent SIP entity needs to be separately negotiated.

NOTE: Since there are SIP entities that already use a combination of Carriage Return and Line Feed (CRLF) as keep-alive messages, and SIP entities are expected to be able to receive those, this specification does not forbid the sending of double-CRLF keep-alive messages towards an adjacent SIP entity even if usage of keep-alives with that SIP entity has not been negotiated. However, the "keep" parameter is still important in order for a SIP entity to indicate that it supports sending of double-CRLF keep-alive messages, so that the adjacent downstream SIP entity does not use other mechanisms (e.g. short registration refresh intervals) in order to keep NAT bindings open.

4.2. Lifetime of keep-alives

4.2.1. General

The lifetime of negotiated keep-alives depends on whether the keep-alives are associated with a registration or a dialog. This section describes the lifetime of negotiated keep-alives.

4.2.2. Keep-alives associated with registration

SIP entities use a registration request in order to negotiate usage of keep-alives associated with a registration. Usage of keep-alives can be negotiated when the registration is established, or later

during the registration. Once negotiated, keep-alives are sent until the registration is terminated, or until a subsequent registration refresh request is sent or forwarded. When a subsequent registration refresh request is sent or forwarded, if a SIP entity is willing to continue sending keep-alives associated with the registration, usage of keep-alives MUST be re-negotiated. If usage is not successfully re-negotiated, the SIP entity MUST cease sending of keep-alives associated with the registration.

NOTE: Sending of keep-alives associated with a registration can only be negotiated in the direction from the registering SIP entity towards the registrar.

4.2.3. Keep-alives associated with dialog

SIP entities use an initial request for a dialog, or a mid-dialog target refresh request [RFC3261], in order to negotiate sending and receiving of keep-alives associated with a dialog. Usage of keep-alives can be negotiated when the dialog is established, or later during the lifetime of the dialog. Once negotiated, keep-alives MUST be sent for the lifetime of the dialog, until the dialog is terminated. Once usage of keep-alives associated with a dialog has been negotiated, it is not possible to re-negotiate the usage associated with the dialog.

4.3. Behavior of a SIP entity willing to send keep-alives

As defined in RFC 5626, a SIP entity that supports sending of keep-alives must act as a Session Traversal Utilities for NAT (STUN) client [RFC5389]. The SIP entity must support those aspects of STUN that are required in order to apply the STUN keep-alive mechanism defined in RFC 5626, and it must support the CRLF keep-alive mechanism defined in RFC 5626. RFC 5626 defines when to use STUN, respectively double-CRLF, for keep-alives.

When a SIP entity sends or forwards a request, if it wants to negotiate the sending of keep-alives associated with a registration, or a dialog, it MUST insert a "keep" parameter in the topmost Via header field that it adds to the request, to indicate willingness to send keep-alives.

When the SIP entity receives the associated response, if the "keep" parameter in the topmost Via header field of the response contains a "keep" parameter value, it MUST start sending keep-alives towards the same destination where it would send a subsequent request (e.g. REGISTER requests and initial requests for dialog) associated with the registration (if the keep-alive negotiation is for a registration), or where it would send subsequent mid-dialog requests

(if the keep-alive negotiation is for a dialog). Subsequent mid-dialog requests are addressed based on the dialog route set.

Once a SIP entity has negotiated sending of keep-alives associated with a dialog towards an adjacent SIP entity, it MUST NOT insert a "keep" parameter in any subsequent SIP requests, associated with the dialog, towards that adjacent SIP entity. Such "keep" parameter MUST be ignored, if received.

Since an ACK request does not have an associated response, it can not be used to negotiate usage of keep-alives. Therefore, a SIP entity MUST NOT insert a "keep" parameter in the topmost Via header field of an ACK request. Such "keep" parameter MUST be ignored, if received.

A SIP entity MUST NOT indicates willingness to send keep-alives associated with a dialog, unless it has also inserted itself in the dialog route set [RFC3261].

NOTE: When a SIP entity sends an initial request for a dialog, if the adjacent downstream SIP entity does not insert itself in the dialog route set using a Record-Route header field [RFC3261], the adjacent downstream SIP entity will change once the dialog route set has been established. If a SIP entity inserts a "keep" parameter in the topmost Via header field of an initial request for a dialog, and the "keep" parameter in the associated response does not contain a parameter value, the SIP entity might choose to insert a "keep" parameter in the topmost Via header field of a subsequent SIP request associated with the dialog, in case the new adjacent downstream SIP entity (based on the dialog route set) is willing to receive keep-alives (in which case it will add a parameter value to the "keep" parameter).

If an INVITE request is used to indicate willingness to send keep-alives, as long as at least one response (provisional or final) to the INVITE request contains a "keep" parameter with a parameter value, it is seen as an indication that the adjacent downstream SIP entity is willing to receive keep-alives associated with the dialog on which the response is received.

4.4. Behavior of a SIP entity willing to receive keep-alives

As defined in RFC 5626, a SIP entity that supports receiving of keep-alives must act as a STUN server [RFC5389]. The SIP entity must support those aspects of STUN that are required in order to apply the STUN keep-alive mechanism defined in RFC 5626, and it must support the CRLF keep-alive mechanism defined in RFC 5626.

When a SIP entity sends or forwards a response, and the adjacent

upstream SIP entity indicated willingness to send keep-alives, if the SIP entity is willing to receive keep-alives associated with the registration, or the dialog, from the adjacent upstream SIP entity it MUST add a parameter value to the "keep" parameter, before sending or forwarding the response. The parameter value, if present and with a value other than zero, represents a recommended keep-alive frequency, given in seconds.

There might be multiple responses to an INVITE request. When a SIP entity indicates willingness to receive keep-alives in a response to an INVITE request, it MUST add a parameter value to the "keep" parameter in at least one reliable response to the request. The SIP entity MAY add identical parameter values to the "keep" parameters in other responses to the same request. The SIP entity MUST NOT add different parameter value to the "keep" parameters in responses to the same request. The SIP entity SHOULD indicate the willingness to receive keep-alives as soon as possible.

A SIP entity MUST NOT indicates willingness to receive keep-alives associated with a dialog, unless it has also inserted itself in the dialog route set [RFC3261].

5. Keep-alive frequency

If a SIP entity receives a SIP response, where the topmost Via header field contains a "keep" parameter with a non-zero value that indicates a recommended keep-alive frequency, given in seconds, it MUST use the procedures defined for the Flow-Timer header field [RFC5626]. According to the procedures, the SIP entity must send keep-alives at least as often as the indicated recommended keep-alive frequency, and if the SIP entity uses the recommended keep-alive frequency then it should send its keep-alives so that the interval between each keep-alive is randomly distributed between 80% and 100% of the recommended keep-alive frequency.

If the received "keep" parameter value is zero, the SIP entity can send keep-alives at its discretion. RFC 5626 provides additional guidance on selecting the keep-alive frequency in case a recommended keep-alive frequency is not provided.

This specification does not specify actions to take if negotiated keep-alives are not received. As defined in RFC 5626, the receiving SIP entity may consider a connection to be dead in such situations.

If a SIP entity that adds a parameter value to the "keep" parameter, in order to indicate willingness to receive keep-alives, also inserts a Flow-Timer header field (that can happen if the SIP entity is using

both the Outbound mechanism and the keep-alive mechanism) in the same SIP message, the header field value and the "keep" parameter value MUST be identical.

SIP Outbound uses the Flow-Timer header field to indicate the server-recommended keep-alive frequency. However, it will only be sent between a UA and an edge proxy. Using the "keep" parameter, however, the sending and receiving of keep-alives might be negotiated between multiple entities on the signalling path. In addition, since the server-recommended keep-alive frequency might vary between different SIP entities, a single Flow-Timer header field can not be used to indicate all the different frequency values.

6. Connection reuse

Keep-alives are often sent in order to keep NAT bindings open, so that the NAT may be passed by SIP requests sent in the reverse direction, reusing the same connection, or for non-connection-oriented transport protocols, reusing the same path. This specification does not define such connection reuse mechanism. The keep-alive mechanism defined in this specification is only used to negotiate the sending and receiving of keep-alives. Entities that want to reuse connections need to use another mechanism to ensure that security aspects associated with connection reuse are taken into consideration.

RFC 5923 [RFC5923] specifies a mechanism for using connection-oriented transports to send requests in the reverse direction, and an entity that wants to use connection-reuse as well as indicate support of keep-alives on that connection will insert both the "alias" parameter defined in RFC 5923 as well as the "keep" parameter defined in this specification.

SIP Outbound specifies how registration flows are used to send requests in the reverse direction.

7. Examples

7.1. General

This section shows example flows where usage of keep-alives, associated with a registration and a dialog, is negotiated between different SIP entities.

NOTE: The examples do not show the actual syntactical encoding of the request lines, response lines and the Via header fields, but rather a

pseudo code in order to identity the message type and to which SIP entity a Via header field is associated.

7.2. Keep-alive negotiation associated with registration: UA-proxy

Figure 1 shows an example where Alice sends an REGISTER request. She indicates willingness of sending keep-alive by inserting a "keep" parameter in her Via header field of the request. The edge proxy (P1) forwards the request towards the registrar.

P1 is willing to receive keep-alives from Alice for the duration of the registration, so when P1 receives the associated response it adds a "keep" parameter value, which indicates a recommended keep-alive frequency of 30 seconds, to Alice’s Via header field, before it forwards the response towards Alice.

When Alice receives the response, she determines from her Via header field that P1 is willing to receive keep-alives associated with the registration. Until the registration expires, or Alice sends a registration refresh request, Alice then sends periodic keep-alives (in this example using the STUN keep-alive technique) towards P1, using the recommended keep-alive frequency indicated by the "keep" parameter value.

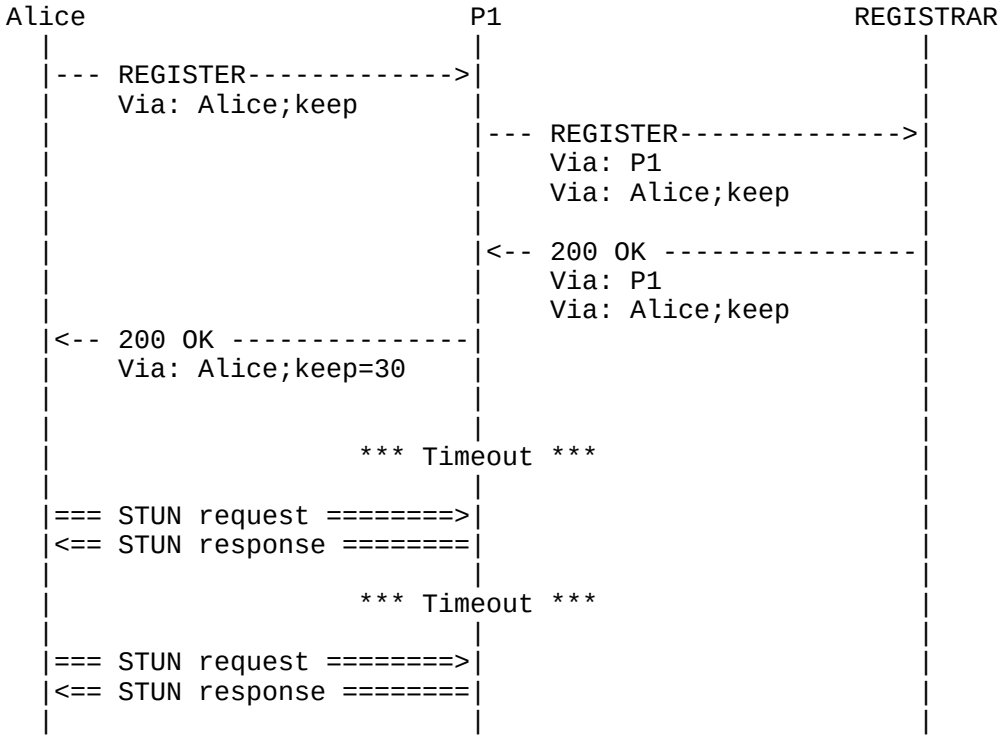


Figure 1: Example call flow

7.3. Keep-alive negotiation associated with dialog: UA-proxy

Figure 2 shows an example where Alice sends an initial INVITE request for a dialog. She indicates willingness to send keep-alive by inserting a "keep" parameter in her Via header field of the request. The edge proxy (P1) adds itself to the dialog route set by adding itself to a Record-Route header field, before it forwards the request towards Bob.

P1 is willing to receive keep-alives from Alice for the duration of the dialog, so When P1 receives the associated response it adds a "keep" parameter value, which indicates a recommended keep-alive frequency of 30 seconds, to Alice’s Via header field, before it forwards the response towards Alice.

When Alice receives the response, she determines from her Via header field that P1 is willing to receive keep-alives associated with the dialog. For the lifetime of the dialog, Alice then sends periodic

keep-alives (in this example using the STUN keep-alive technique) towards P1, using the recommended keep-alive frequency indicated by the "keep" parameter value.

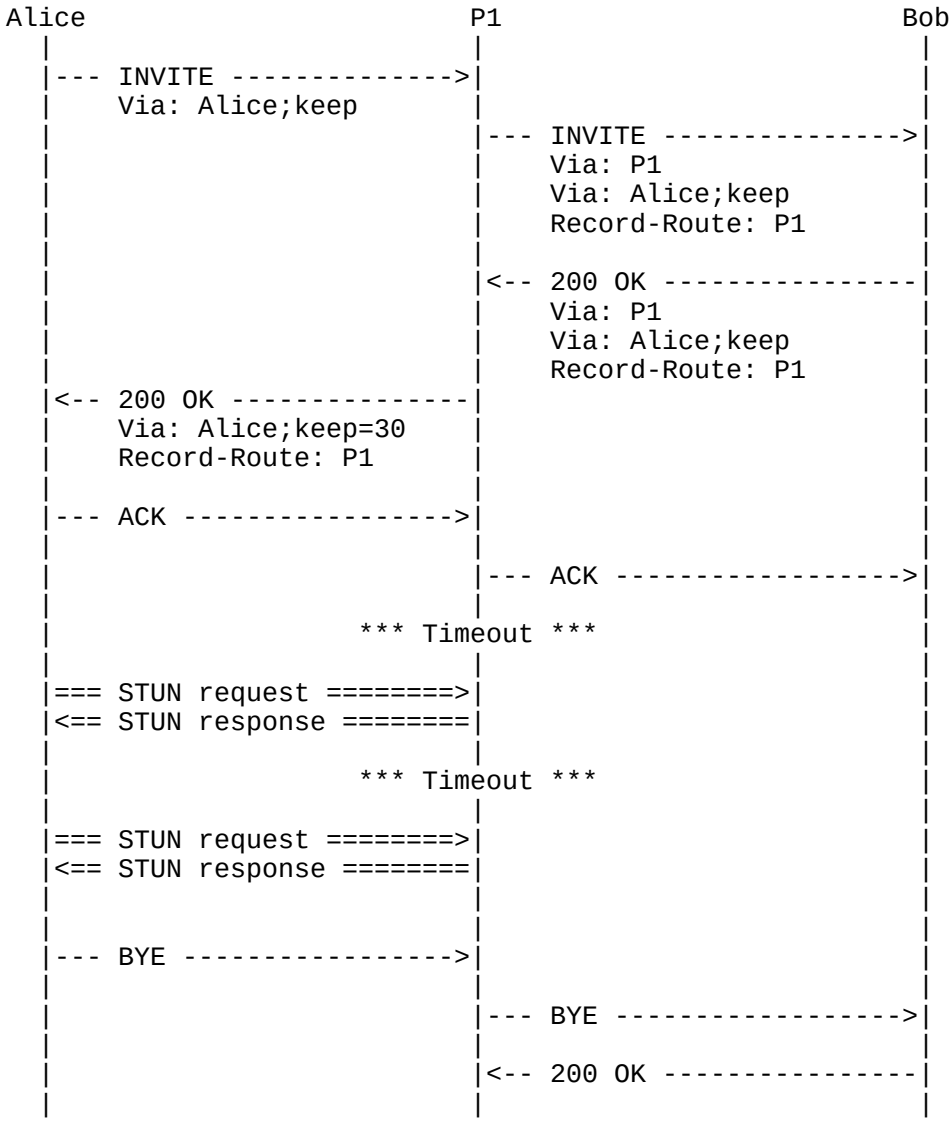


Figure 2: Example call flow

7.4. Keep-alive negotiation associated with dialog: UA-UA

Figure 3 shows an example where Alice sends an initial INVITE request for a dialog. She indicates willingness to send keep-alive by inserting a "keep" parameter in her Via header field of the request. The edge proxy (P1) does not add itself to the dialog route set, by adding itself to a Record-Route header field, before it forwards the request towards Bob.

When Alice receives the response, she determines from her Via header field that P1 is not willing to receive keep-alives associated with the dialog from her. When the dialog route set has been established, Alice sends a mid-dialog UPDATE request towards Bob (since P1 did not insert itself in the dialog route set), and she once again indicates willingness to send keep-alives by inserting a "keep" parameter in her Via header field of the request. Bob supports the keep-alive mechanism, and is willing to receive keep-alives associated with the dialog from Alice, so he creates a response and adds a "keep" parameter value, which indicates a recommended keep-alive frequency of 30 seconds, to Alice’s Via header field, before he forwards the response towards Alice.

When Alice receives the response, she determines from her Via header field that Bob is willing to receive keep-alives associated with the dialog. For the lifetime of the dialog, Alice then sends periodic keep-alives (in this example using the STUN keep-alive technique) towards Bob, using the recommended keep-alive frequency indicated by the "keep" parameter value.

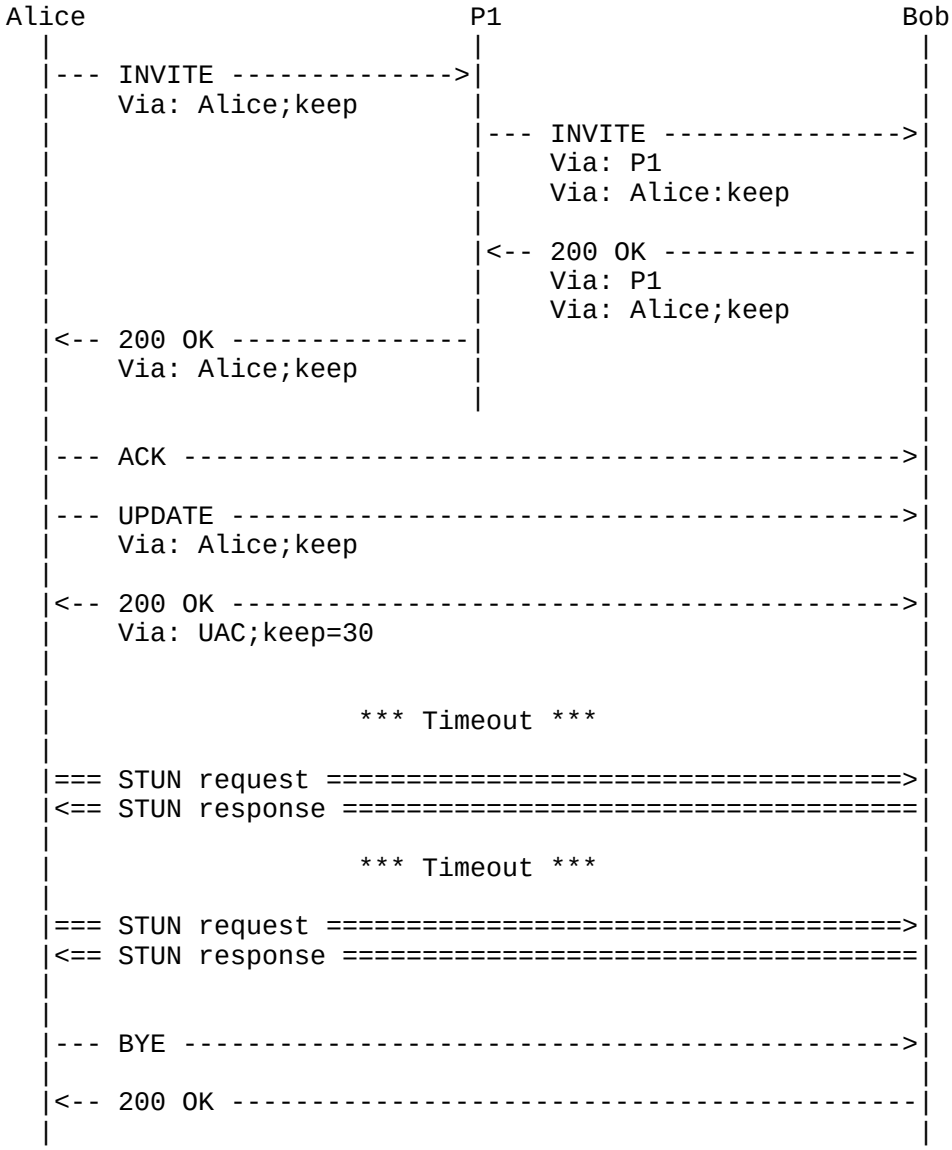


Figure 3: Example call flow

8. Grammar

8.1. General

This section describes the syntax extensions to the ABNF syntax defined in RFC 3261, by defining a new Via header field parameter, "keep". The ABNF defined in this specification is conformant to RFC 5234 [RFC5234].

8.2. ABNF

```
via-params =/ keep

keep       = "keep" [ EQUAL 1*(DIGIT) ]
```

9. IANA Considerations

9.1. keep

This specification defines a new Via header field parameter called keep in the "Header Field Parameters and Parameter Values" sub-registry as per the registry created by [RFC3968]. The syntax is defined in Section 8. The required information is:

Header Field	Parameter Name	Predefined Values	Reference
Via	keep	No	[RFCXXXX]

10. Security Considerations

SIP entities that send or receive keep-alives are often required to use a connection reuse mechanism, in order to ensure that requests sent in the reverse direction, towards the sender of the keep-alives, traverse NATs etc. This specification does not specify a connection reuse mechanism, and it does not address security issues related to connection reuse. SIP entities that wish to reuse connections need to use a dedicated connection reuse mechanism, in conjunction with the keep-alive negotiation mechanism.

Unless SIP messages are integrity protected hop-by-hop, e.g. using Transport Layer Security (TLS) [RFC5246] or Datagram Transport Layer Security (DTLS) [RFC4347], a man-in-the-middle can modify Via header fields used by two entities to negotiate sending of keep-alives, e.g.

by removing the indications used to indicate willingness to send and receive keep-alives, or by decreasing the timer value to a very low value, which might trigger additional resource consumption due to the frequently sent keep-alives.

The behavior defined in Sections 4.3 and 4.4 require a SIP entity using the mechanism defined in this specification to place a value in the "keep" parameter in the topmost Via header field value of a response the SIP entity sends. They do not instruct the entity to place a value in a "keep" parameter of any request it forwards. In particular, SIP proxies MUST NOT place a value into the keep parameter of the topmost Via header field value of a request it receives before forwarding it. A SIP proxy implementing this specification SHOULD remove any keep parameter values in any Via header field values below the topmost one in responses it receives before forwarding them.

When requests are forwarded across multiple hops, it is possible for a malicious downstream SIP entity to tamper with the accrued values in the Via header field. The malicious SIP entity could place a value, or change an existing value in a "keep" parameter in any of the Via header field values, not just the topmost value. A proxy implementation that simply forwards responses by stripping the topmost Via header field value and not inspecting the resulting new topmost Via header field value risks being adversely affected by such a malicious downstream SIP entity. In particular, such a proxy may start receiving STUN requests if it blindly forwards a response with a keep parameter with a value it did not create in the topmost Via header field.

To lower the chances of the malicious SIP entity’s actions having adverse affects on such proxies, when a SIP entity sends STUN keep-alives to an adjacent downstream SIP entity and does not receive a response to those STUN messages, it MUST, based on the procedure in section 4.4.2 of RFC 5626, after 7 retransmissions, or when an error response is received for the STUN request, stop sending keep-alives for the remaining duration of the dialog (if the sending of keep-alives were negotiated for a dialog) or until the sending of keep-alives is re-negotiated for the registration (if the sending keep-alives were negotiated for a registration).

Apart from the issues described above, this specification does not introduce security considerations in addition to those specified for keep-alives in [RFC5626].

11. Acknowledgements

Thanks to Staffan Blau, Francois Audet, Hadriel Kaplan, Sean Schneyer and Milo Orsic for their comments on the initial draft. Thanks to Juha Heinaenen, Jiri Kuthan, Dean Willis, John Elwell, Paul Kyzivat, Peter Musgrave, Dale Worley, Adam Roach and Robert Sparks for their comments on the list. Thanks to Vijay Gurbani for providing text about the relationship with the connect reuse specification.

12. Change Log

[RFC EDITOR NOTE: Please remove this section when publishing]

Changes from draft-ietf-sipcore-keep-11
- Editorial fixes based on last call comments by Peter Saint-Andre (Jan 11th)
- - TLS and DTLS references added
- - Clarification that the sending of keep-alives stops after 7 retranmissions
- Editorial fixes based on last call comments by Alexey Melnikov (Jan 12th)
- - Additional text added to Grammar section
- Editorial fixes based on last call comments by Adrian Farrel (Jan 16th)
- Editorial fixes based on last call comments by Sean Turner (Jan 20th)
- Reference clean-ups

Changes from draft-ietf-sipcore-keep-10
- Editorial fixes based on last call comments by Juergen Schoenwaelder (Dec 21st)
- Editorial fixes based on last call comments by Roni Even (Dec 28th)

Changes from draft-ietf-sipcore-keep-09
- Changes based on AD review comments by Robert Sparks
- Redundant paragraph removed from security considerations

Changes from draft-ietf-sipcore-keep-08
- Changes based on AD review comments by Robert Sparks
- Additional security considerations text provided by Robert Sparks
- http://www.ietf.org/mail-archive/web/sipcore/current/msg03779.html (Nov 23rd)
- http://www.ietf.org/mail-archive/web/sipcore/current/msg03780.html (Nov 23rd)

Changes from draft-ietf-sipcore-keep-07

- Last paragraph of section 4.2.2 removed
- Reference correction

Changes from draft-ietf-sipcore-keep-06
- New text added to the security considerations

Changes from draft-ietf-sipcore-keep-05
- New section about connection reuse added
- Clarify that the specification does not define a mechanism for connection reuse
- New text added to the security considerations
- CRLF changed to double-CRLF in some places

13. References

13.1. Normative References

[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.

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

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5626] Jennings, C., Mahy, R., and F. Audet, "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, October 2009.

13.2. Informative References

[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.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security

(TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5923] Gurbani, V., Mahy, R., and B. Tate, "Connection Reuse in the Session Initiation Protocol (SIP)", RFC 5923, June 2010.

Author's Address

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: christer.holmberg@ericsson.com

Network Working Group
Internet Draft
Expires: Mar 4, 2012
Intended Status: Standards Track (PS)

James Polk
Cisco Systems
Brian Rosen
Jon Peterson
NeuStar

Sept 4, 2011

Location Conveyance for the Session Initiation Protocol
draft-ietf-sipcore-location-conveyance-09.txt

Abstract

This document defines an extension to the Session Initiation Protocol (SIP) to convey geographic location information from one SIP entity to another SIP entity. The SIP extension covers end-to-end conveyance as well as location-based routing, where SIP intermediaries make routing decisions based upon the location of the Location Target.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on Mar 4, 2012.

Copyright Notice

Table of Contents

1. Conventions and Terminology used in this document

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 [RFC2119]. This document furthermore uses numerous terms defined in RFC 3693 [RFC3693], including Location Object, Location Recipient, Location Server, Target, Rulemaker and Using Protocol.

2. Introduction

Session Initiation Protocol (SIP) [RFC3261] creates, modifies and terminates multimedia sessions. SIP carries certain information related to a session while establishing or maintaining calls. This document defines how SIP conveys geographic location information of a Target to a Location Recipient (LR). SIP acts as a Using Protocol of location information, as defined in RFC 3693.

In order to convey location information, this document specifies three new SIP header fields, Geolocation, Geolocation-Routing and Geolocation-Error, which carry a reference to a Location Object (LO), grant permission to route a SIP request based on the location-value and provide error notifications specific to location errors respectively. The Location Object (LO) may appear in a MIME body attached to the SIP request, or it may be a remote resource in the network.

A Target is an entity whose location is being conveyed, per RFC 3693. Thus, a Target could be a SIP user agent (UA), some other IP device (a router or a PC) that does not have a SIP stack, a non-IP device (a person or a black phone) or even a non-communications device (a building or store front). In no way does this document assume that the SIP user agent client which sends a request containing a location object is necessarily the Target. The location of a Target conveyed within SIP typically corresponds to that of a device controlled by the Target, for example, a mobile phone, but such devices can be separated from their owners, and moreover, in some cases the user agent may not know its own location.

In the SIP context, a location recipient will most likely be a SIP UA, but due to the mediated nature of SIP architectures, location information conveyed by a single SIP request may have multiple recipients, as any SIP proxy server in the signaling path that inspects the location of the Target must also be considered a Location Recipient. In presence-like architectures, an intermediary

that receives publications of location information and distributes them to watchers acts as a Location Server per RFC 3693. This location conveyance mechanism can also be used to deliver URIs pointing to such Location Servers where prospective Location Recipients can request Location Objects.

3. Overview of SIP Location Conveyance

An operational overview of SIP location conveyance can be shown in 4 basic diagrams, with most applications falling under one of the following basic use cases. Each is separated into its own subsection here in section 3.

Each diagram has Alice and Bob as UAs. Alice is the Target, and Bob is an LR. A SIP intermediary appears in some of the diagrams. Any SIP entity that receives and inspects location information is an LR, therefore in any of the diagrams the SIP intermediary that receives a SIP request is potentially an LR - though that does not mean such an intermediary necessarily has to route the SIP request based on the location information. In some use cases, location information passes through the LS on the right of each diagram.

3.1 Location Conveyed by Value

We start with the simplest diagram of Location Conveyance, Alice to Bob, where no other layer 7 entities are involved.

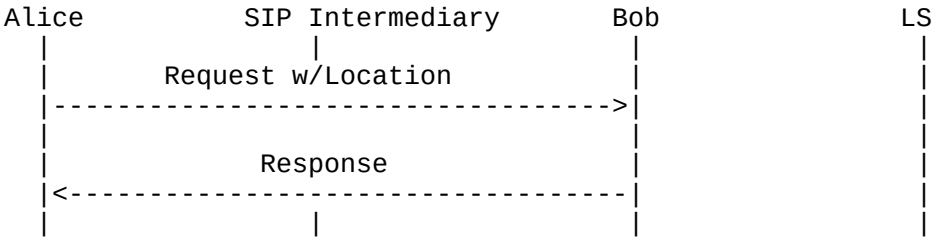


Figure 1. Location Conveyed by Value

In Figure 1, Alice is both the Target and the LS that is conveying her location directly to Bob, who acts as an LR. This conveyance is point-to-point - it does not pass through any SIP-layer intermediary. A Location Object appears by-value in the initial SIP request as a MIME body, and Bob responds to that SIP request as appropriate. There is a 'Bad Location Information' response code introduced within this document to specifically inform Alice if she conveys bad location to Bob (e.g., Bob "cannot parse the location provided", or "there is not enough location information to determine where Alice is").

3.2 Location Conveyed as a Location URI

Here we make Figure 1 a little more complicated by showing a diagram of indirect Location Conveyance from Alice to Bob, where Bob's entity has to retrieve the location object from a 3rd party server.

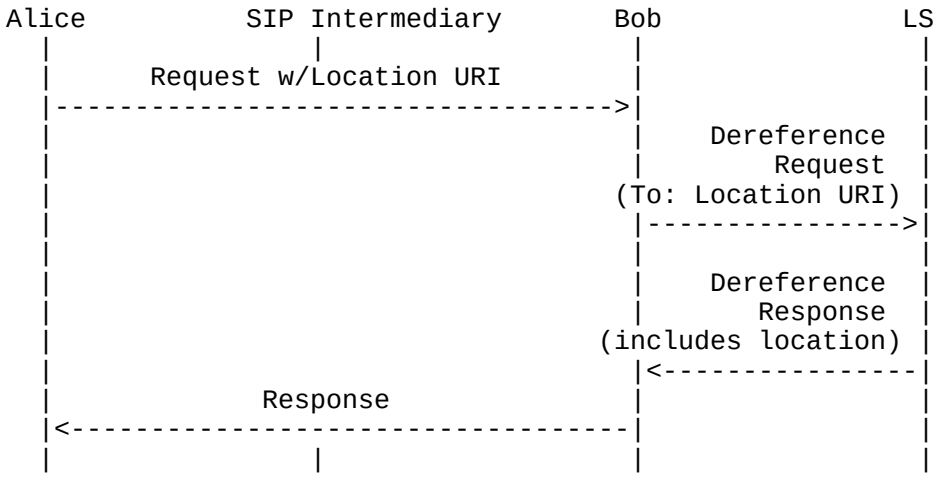


Figure 2. Location Conveyed as a Location URI

In Figure 2, location is conveyed indirectly, via a Location URI carried in the SIP request (more of those details later). If Alice sends Bob this Location URI, Bob will need to dereference the URI - analogous to Content Indirection [RFC4483] - in order to request the location information. In general, the LS provides the location value to Bob instead of Alice directly for conveyance to Bob. From a user interface perspective, Bob the user won't know that this information was gathered from an LS indirectly rather than culled from the SIP request, and practically this does not impact the operation of location-based applications.

The example given in this section is only illustrative, not normative. In particular, applications can choose to dereference a location URI at any time, possibly several times, or potentially not at all. Applications receiving a Location URI in a SIP transaction need to be mindful of timers used by different transactions. In particular, if the means of dereferencing the Location URI might take longer than the SIP transaction timeout (Timer C for INVITE transactions, Timer F for non-INVITE transactions), then it needs to rely on mechanisms other than the transaction's response code to convey location errors, if returning such errors are necessary.

3.3 Location Conveyed though a SIP Intermediary

In Figure 3, we introduce the idea of a SIP intermediary into the example to illustrate the role of proxying in the location architecture. This intermediary can be a SIP proxy or it can be a back-to-back-user-agent (B2BUA). In this message flow, the SIP intermediary could act as a LR, in addition to Bob. The primary use case for intermediaries consuming location information is location-based routing. In this case, the intermediary chooses a next hop for the SIP request by consulting a specialized location service which selects forwarding destinations based on geographical location.

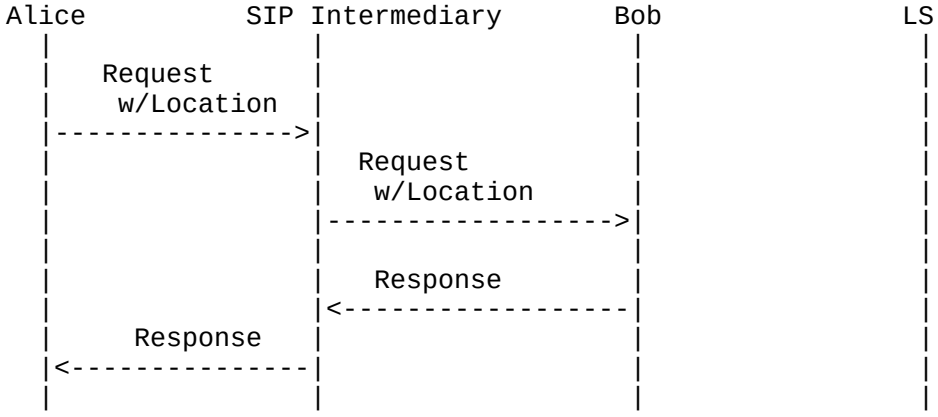


Figure 3. Location Conveyed though a SIP Intermediary

However, the most common case will be one in which the SIP intermediary receives a request with location information (conveyed either by-value or by-reference) and does not know or care about Alice's location, or support this extension, and merely passes it on to Bob. In this case, the intermediary does not act as a Location Recipient. When the intermediary is not an LR, this use case is the same as the one described in Section 3.1.

Note that an intermediary does not have to perform location-based routing in order to be a Location Recipient. It could be the case that a SIP intermediary which does not perform location-based routing does care when Alice includes her location; for example, it could care that the location information is complete or that it correctly identifies where Alice is. The best example of this is intermediaries that verify location information for emergency calling, but it could also be for any location based routing - e.g., contacting your favorite local pizza delivery service, making sure that organization has Alice's proper location in the initial SIP request.

There is another scenario in which the SIP intermediary cares about location and is not an LR, one in which the intermediary inserts another location of the Target, Alice in this case, into the request, and forwards it. This secondary insertion is generally not

advisable because downstream SIP entities will not be given any guidance about which location to believe is better, more reliable, less prone to error, more granular, worse than the other location or just plain wrong.

This document takes a "you break it, you bought it" approach to dealing with second locations placed into a SIP request by an intermediary entity. That entity becomes completely responsible for all location within that SIP request (more on this in Section 4).

3.4 SIP Intermediary Replacing Bad Location

If the SIP intermediary rejects the message due to unsuitable location information, the SIP response will indicate there was 'Bad Location Information' in the SIP request, and provide a location specific error code indicating what Alice needs to do to send an acceptable request (see Figure 4 for this scenario).

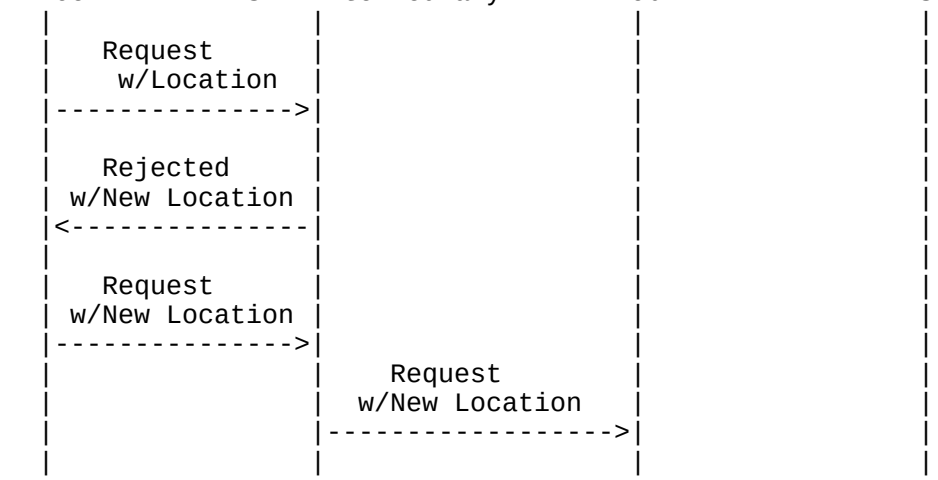


Figure 4. SIP Intermediary Replacing Bad Location

In this last use case, the SIP intermediary wishes to include a Location Object indicating where it understands Alice to be. Thus, it needs to inform her user agent what location it will include in any subsequent SIP request that contains her location. In this case, the intermediary can reject Alice's request and, through the SIP response, convey to her the best way to repair the request in order for the intermediary to accept it.

Overriding location information provided by the user requires a deployment where an intermediary necessarily knows better than an end user - after all, it could be that Alice has an on-board GPS, and the SIP intermediary only knows her nearest cell tower. Which is more accurate location information? Currently, there is no way to

tell which entity is more accurate, or which is wrong - for that matter. This document will not specify how to indicate which location is more accurate than another.

As an aside, it is not envisioned that any SIP-based emergency services request (i.e., IP-911, or 112 type of call attempt) will receive a corrective 'Bad Location Information' response from an intermediary. Most likely, the SIP intermediary would in that scenario act as a B2BUA and insert into the request by-value any appropriate location information for the benefit of Public Safety Answering Point (PSAP) call centers to expedite call reception by the emergency services personnel; thereby, minimizing any delay in call establishment time. The implementation of these specialized deployments is, however, outside the scope of this document.

4. SIP Extensions for Geolocation Conveyance

The following sections detail the extensions to SIP for location conveyance.

4.1 The Geolocation Header Field

This document defines "Geolocation" as a new SIP header field registered by IANA, with the following ABNF [RFC5234]:

```
message-header    /= Geolocation-header ; (message-header from 3261)
Geolocation-header = "Geolocation" HCOLON locationValue
                     *( COMMA locationValue )
locationValue      =  LAQUOT locationURI RAQUOT
                       *(SEMI geoloc-param)
locationURI        =  sip-URI / sips-URI / pres-URI
                       / http-URI / https-URI
                       / cid-url ; (from RFC 2392)
                       / absoluteURI ; (from RFC 3261)
geoloc-param       =  generic-param;  (from RFC 3261)
```

HCOLON, COMMA, LAQUOT, RAQUOT, and SEMI are defined in RFC3261 [RFC3261].

sip-URI, sips-URI and absoluteURI are defined according to [RFC3261].

The pres-URI is defined in [RFC3859].

http-URI and https-URI are defined according to [RFC2616] and [RFC2818], respectively.

The cid-url is defined in [RFC2392] to locate message body parts. This URI type is present in a SIP request when location is conveyed as a MIME body in the SIP message.

GEO-URIs [RFC5870] are not appropriate for usage in the SIP

Geolocation header, because it does not include retention and re-transmission flags as part of the location information. Other URI schemes used in the location URI MUST be reviewed against the RFC 3693 [RFC3693] criteria for a Using Protocol. Section 4.6 discusses how URI schemes are communicated using this SIP extension, and what to do if a URI scheme is received that cannot be supported.

The generic-param in the definition of locationValue is included as a mechanism for future extensions that might require parameters. This document defines no parameters for use with locationValue. If a Geolocation header field is received that contains generic-params, each parameter SHOULD be ignored, and SHOULD NOT be removed when forwarding the locationValue. If a need arises to define parameters for use with locationValue, a revision/extension to this document is required.

The Geolocation header field MUST have at least one locationValue. A SIP intermediary SHOULD NOT add location to a SIP request that already contains location. This will quite often lead to confusion within LRs. However, if a SIP intermediary adds location, even if location was not previously present in a SIP request, that SIP intermediary is fully responsible for addressing the concerns of any 424 (Bad Location Information) SIP response it receives about this location addition, and MUST NOT pass on (upstream) the 424 response. A SIP intermediary that adds a locationValue MUST position the new locationValue as the last locationValue within the Geolocation header field of the SIP request.

This document defines the Geolocation header field as valid in the following SIP requests:

INVITE [RFC3261],	REGISTER [RFC3261],
OPTIONS [RFC3261],	BYE [RFC3261],
UPDATE [RFC3311],	INFO [RFC6086],
MESSAGE [RFC3428],	REFER [RFC3515],
SUBSCRIBE [RFC3265],	NOTIFY [RFC3265],
PUBLISH [RFC3903]	

The Geolocation header field MAY be included in any one of the above listed requests by a UA, and a 424 response to any one of the requests sent above. Fully appreciating the caveats/warnings mentioned above, a SIP intermediary MAY add the Geolocation header field.

A SIP intermediary MAY add a Geolocation header field if one is not present - for example, when a user agent does not support the Geolocation mechanism but their outbound proxy does and knows the Target’s location, or any of a number of other use cases (see Section 3).

The Geolocation header field MAY be present in a SIP request or response without the presence of a Geolocation-Routing header

(defined in Section 4.2). As stated in Section 4.2, the default value of Geolocation-Routing header-value is "no", meaning SIP intermediaries MUST NOT view (i.e., process, inspect or actively dereference) any direct or indirect location within this SIP message. This is for at least two fundamental reasons,

1) to make the possibility of retention of the Target's location moot (because it was not viewed in the first place); and

2) to prevent a different treatment of this SIP request based on the contents of the Location Information in the SIP request.

Any locationValue MUST be related to the original Target. This is equally true for the location information in a SIP response, i.e., from a SIP intermediary back to the Target as explained in Section 3.4. SIP intermediaries SHOULD NOT modify or delete any existing locationValue(s). A use-case in which this would not apply would be where the SIP intermediary is an anonymizer. The problem with this scenario is that the geolocation included by the Target then becomes useless for the purpose or service they wanted to use (include) it for. For example, 911/emergency calling or finding the nearest (towing company/pizza delivery/dry cleaning) service(s) will not yield intended results if the Location Information were to be modified or deleted from the SIP request.

4.2 The Geolocation-Routing Header Field

This document defines "Geolocation-Routing" as a new SIP header field registered by IANA, with the following ABNF [RFC5234]:

```
message-header    /= Georouting-header ; (message-header from 3261)
Georouting-header  = "Geolocation-Routing" HCOLON
                     ( "yes" / "no" / generic-value )
generic-value      =  generic-param;  (from RFC 3261)
```

HCOLON is defined in RFC3261 [RFC3261].

The only defined values for the Geolocation-Routing header field are "yes" or "no". When the value is "yes", the locationValue can be used for routing decisions along the downstream signaling path by intermediaries. Values other than "yes" or "no" are permitted for future extensions. Implementations not aware of an extension MUST treat any other received value the same as "no".

If no Geolocation-Routing header field is present in a SIP request, a SIP intermediary MAY insert this header. Without knowledge from a Rulemaker, the SIP intermediary inserting this header-value SHOULD NOT set the value to "yes", as this may be more permissive than the originating party intends. An easy way around this is to have the Target always insert this header-value as "no".

When this Geolocation-Routing header-value is set to "no", this means no locationValue (inserted by the originating UAC or any intermediary along the signaling path) can be used by any SIP intermediary to make routing decisions. Intermediaries that attempt to use the location information for routing purposes in spite of this counter indication could end up routing the request improperly as a result. Section 4.4 describes the details on what a routing intermediary does if it determines it needs to use the location in the SIP request in order to process the message further. The practical implication is that when the Geolocation-Routing header-value is set to "no", if a cid:url is present in the SIP request, intermediaries MUST NOT view the location (because it is not for intermediaries to consider when processing the request), and if a location URI is present, intermediaries MUST NOT dereference it. UAs are allowed to view location in the SIP request even when the Geolocation-Routing header-value is set to "no". An LR MUST by default consider the Geolocation-Routing header-value as set to "no", with no exceptions, unless the header field value is set to "yes".

A Geolocation-Routing header-value that is set to "no" has no special security properties. It is at most a request for behavior within SIP intermediaries. That said, if the Geolocation-Routing header-value is set to "no", SIP intermediaries are still to process the SIP request and send it further downstream within the signaling path if there are no errors present in this SIP request.

The Geolocation-Routing header field satisfies the recommendations made in section 3.5 of RFC 5606 [RFC5606] regarding indication of permission to use location-based routing in SIP.

SIP implementations are advised to pay special attention to the policy elements for location retransmission and retention described in RFC 4119.

The Geolocation-Routing header field cannot appear without a header-value in a SIP request or response (i.e., a null value is not allowed). The absence of a Geolocation-Routing header-value in a SIP request is always the same as the following header field:

```
Geolocation-Routing: no
```

The Geolocation-Routing header field MAY be present without a Geolocation header field in the same SIP request. This concept is further explored in Section 4.2.1.

4.2.1 Explaining Geolocation-Routing header-value States

The Geolocation header field contains a Target’s location, and MUST NOT be present if there is no location information in this SIP request. The location information is contained in one or more

locationValues. These locationValues MAY be contained in a single Geolocation header field, or distributed among multiple Geolocation header fields. (See section 7.3.1 of RFC3261.)

The Geolocation-Routing header field indicates whether or not SIP intermediaries can view and then route this SIP request based on the included (directly or indirectly) location information. The Geolocation-Routing header field MUST NOT appear more than once in any SIP request, and MUST NOT lack a header-value. The default or implied policy of a SIP request that does not have a Geolocation-Routing header field is the same as if one were present and the header-value were set to "no".

There are only 3 possible states regarding the Geolocation-Routing header field

- "no"
- "yes"
- no header-field present in this SIP request

The expected results in each state are:

If the Geolocation-Routing	Only possible interpretations:
"no"	SIP intermediaries MUST NOT process included geolocation information within this SIP request. SIP intermediaries inserting a locationValue into a Geolocation header field (whether adding to an existing header-value or inserting the Geolocation header field for the first time) MUST NOT modify or delete the received "no" header-value.
"yes"	SIP intermediaries can process included geolocation information within this SIP request, and can change the policy to "no" for intermediaries further downstream.
Geolocation-Routing absent	If a Geolocation header field exists (meaning a locationValue is already present), a SIP intermediary MUST interpret the lack of a Geolocation-Routing header field as if there were one present and the header-value is set to "no".

If there is no Geolocation header field in this SIP request, the default Geolocation-Routing is open and can be set by a SIP intermediary or not at all.

4.3 424 (Bad Location Information) Response Code

This SIP extension creates a new location-specific response code, defined as follows,

424 (Bad Location Information)

The 424 (Bad Location Information) response code is a rejection of the request due to its location contents, indicating location information that was malformed or not satisfactory for the recipient's purpose, or could not be dereferenced.

A SIP intermediary can also reject a location it receives from a Target when it understands the Target to be in a different location. The proper handling of this scenario, described in Section 3.4, is for the SIP intermediary to include the proper location in the 424 Response. This SHOULD be included in the response as a MIME message body (i.e., a location value), rather than as a URI; however, in cases where the intermediary is willing to share location with recipients but not with a user agent, a reference might be necessary.

As mentioned in Section 3.4, it might be the case that the intermediary does not want to chance providing less accurate location information than the user agent; thus it will compose its understanding of where the user agent is in a separate <geopriv> element of the same PIDF-LO [RFC4119] message body in the SIP response (which also contains the Target's version of where it is). Therefore, both locations are included - each with different <method> elements. The proper reaction of the user agent is to generate a new SIP request that includes this composed location object, and send it towards the original LR. SIP intermediaries can verify that subsequent requests properly insert the suggested location information before forwarding said requests.

SIP intermediaries that are forwarding (as opposed to generating) a 424 response MUST NOT add, modify, or delete any location appearing in that response. This specifically applies to intermediaries that are between the 424 response generator and the original UAC. Geolocation and Geolocation-Error header fields and PIDF-LO body parts MUST remain unchanged, never added to or deleted.

Section 4.4 describes a Geolocation-Error header field to provide more detail about what was wrong with the location information in the request. This header field MUST be included in the 424 response.

It is only appropriate to generate a 424 response when the responding entity needs a locationValue and there are no values in the request that are usable by the responder, or when the responder has additional location information to provide. The latter case is shown in Figure 4 of section 3.4. There, a SIP intermediary is informing the upstream UA which location to include in the next SIP request.

A 424 MUST NOT be sent in response to a request that lacks a Geolocation header entirely, as the user agent in that case may not support this extension at all. If a SIP intermediary inserted a locationValue into a SIP request where one was not previously present, it MUST take any and all responsibility for the corrective action if it receives a 424 to a SIP request it sent.

A 424 (Bad Location Information) response is a final response within a transaction, and MUST NOT terminate an existing dialog.

4.4 The Geolocation-Error Header Field

As discussed in Section 4.3, more granular error notifications specific to location errors within a received request are required if the location inserting entity is to know what was wrong within the original request. The Geolocation-Error header field is used for this purpose.

The Geolocation-Error header field is used to convey location-specific errors within a response. The Geolocation-Error header field has the following ABNF [RFC5234]:

```
message-header          /= Geolocation-Error
                           ; (message-header from 3261)
Geolocation-Error        = "Geolocation-Error" HCOLON
                             locationErrorValue
locationErrorValue       = location-error-code
                            *(SEMI location-error-params)
location-error-code      = 1*3DIGIT
location-error-params    = location-error-code-text
                           / generic-param ; from RFC3261
location-error-code-text = "code" EQUAL quoted-string ; from RFC3261
```

HCOLON, SEMI, and EQUAL are defined in RFC3261 [RFC3261]. DIGIT is defined in RFC5234 [RFC5234].

The Geolocation-Error header field MUST contain only one locationErrorValue to indicate what was wrong with the locationValue the Location Recipient determined was bad. The locationErrorValue contains a 3-digit error code indicating what was wrong with the

location in the request. This error code has a corresponding quoted error text string that is human understandable. The text string is OPTIONAL, but RECOMMENDED for human readability, similar to the string phrase used for SIP response codes. That said, the strings are complete enough for rendering to the user, if so desired. The strings in this document are recommendations, and are not standardized - meaning an operator can change the strings - but MUST NOT change the meaning of the error code. Similar to how RFC 3261 specifies, there MUST NOT be more than one string per error code.

The Geolocation-Error header field MAY be included in any response to one of the SIP Methods mentioned in Section 4.1, so long as a locationValue was in the request part of the same transaction. For example, Alice includes her location in an INVITE to Bob. Bob can accept this INVITE, thus creating a dialog, even though his UA determined the location contained in the INVITE was bad. Bob merely includes a Geolocation-Error header value in the 200 OK to the INVITE informing Alice the INVITE was accepted but the location provided was bad.

If, on the other hand, Bob cannot accept Alice’s INVITE without a suitable location, a 424 (Bad Location Information) is sent. This message flow is shown in Figures 1, 2 or 3 in Sections 3.1, 3.2 and 3.3 respectively.

If Alice is deliberately leaving location information out of the LO because she does not want Bob to have this additional information, implementations should be aware that Bob could error repeatedly in order to receive more location information about Alice in a subsequent SIP request. Implementations MUST be on guard for this, by not allowing continually more information to be revealed unless it is clear that any LR is permitted by Alice to know all that Alice knows about her location. A limit on the number of such rejections to learn more location information SHOULD be configurable, with a RECOMMENDED maximum of 3 times for each related transaction.

A SIP intermediary that requires Alice’s location in order to properly process Alice’s INVITE also sends a 424 with a Geolocation-Error code. This message flow is shown in Figure 4 of Section 3.4.

If more than one locationValue is present in a SIP request and at least one locationValue is determined to be valid by the LR, the location in that SIP request MUST be considered good as far as location is concerned, and no Geolocation-Error is to be sent.

Here is an initial list of location based error code ranges for any SIP response, including provisional responses (other than 100 Trying) and the new 424 (Bad Location Information) response. These error codes are divided into 3 categories, based on how the response receiver should react to these errors. There MUST be no more than one Geolocation-Error code in a SIP response, regardless of how many

locationValues there are in the correlating SIP request. There is no guidance given in this document as to which locationValue, when more than one was present in the SIP request, is related to the Geolocation-Error code; meaning that, somehow not defined here, the LR just picks one to error.

- o 1XX errors mean the LR cannot process the location within the request

  A non-exclusive list of reasons for returning a 1XX is

  - the location was not present or could not be found,
  - there was not enough location information to determine where the Target was,
  - the location information was corrupted or known to be inaccurate,

- o 2XX errors mean some specific permission is necessary to process the included location information.

- o 3XX errors mean there was trouble dereferencing the Location URI sent.

Dereference attempts to the same request SHOULD be limited to 10 attempts within a few minutes. This number SHOULD be configurable, but result in a Geolocation-Error: 300 error once reached.

It should be noted that for non-INVITE transactions, the SIP response will likely be sent before the dereference response has been received. This document does not alter that SIP protocol reality. This means the receiver of any non-INVITE response to a request containing location SHOULD NOT consider a 200 OK to mean the act of dereferencing has concluded and the dereferencer (i.e., the LR) has successfully received and parsed the PIDF-LO for errors and found none. The end of section 3.2 discusses how transaction timing considerations lead to this requirement.

Additionally, if an LR cannot or chooses not to process location from a SIP request, a 500 (Server Internal Error) SHOULD be used with or without a configurable Retry-After header field. There is no special location error code for what already exists within SIP today.

Within each of these ranges, there is a top level error as follows:

```
Geolocation-Error: 100 ; code="Cannot Process Location"

Geolocation-Error: 200 ; code="Permission To Use Location
                               Information"

Geolocation-Error: 300 ; code="Dereference Failure"
```

If an error recipient cannot process a specific error code (such as the 201 or 202 below), perhaps because it does not understand that specific error code, the error recipient SHOULD process the error code as if it originally were a top level error code where the X in X00 matches the specific error code. If the error recipient cannot process a non-100 error code, for whatever reason, then the error code 100 MUST be processed.

There are two specific Geolocation-Error codes necessary to include in this document, both have to do with permissions necessary to process the SIP request; they are

Geolocation-Error: 201 ; code="Permission To Retransmit Location Information to a Third Party"

This location error is specific to having the Presence Information Data Format (PIDF-LO) [RFC4119] <retransmission-allowed> element set to "no". This location error is stating it requires permission (i.e., PIDF-LO <retransmission-allowed> element set to "yes") to process this SIP request further. If the LS sending the location information does not want to give this permission, it will not change this permission in a new request. If the LS wants this message processed with the <retransmission-allowed> element set to "yes" it MUST choose another logical path (if one exists) for this SIP request.

Geolocation-Error: 202 ; code="Permission to Route based on Location Information"

This location error is specific to having the Geolocation-Routing header value set to "no". This location error is stating it requires permission (i.e., the Geolocation-Routing header value set to "yes") to process this SIP request further. If the LS sending the location information does not want to give this permission, it will not change this permission in a new request. If the LS wants this message processed with the <retransmission-allowed> element set to "yes" it MUST choose another logical path (if one exists) for this SIP request.

4.5 Location URIs in Message Bodies

In the case where an LR sends a 424 response and wishes to communicate suitable location by reference rather than by value, the 424 MUST include a content-indirection body per RFC 4483.

4.6 Location Profile Negotiation

The following is part of the discussion started in Section 3, Figure 2, which introduced the concept of sending location indirectly.

If a location URI is included in a SIP request, the sending user agent MUST also include a Supported header field indicating which location profiles it supports. Two option tags for location profiles are defined by this document: "geolocation-sip" and "geolocation-http". Future specifications MAY define further location profiles per the IANA policy described in Section 8.3.

The "geolocation-sip" option tag signals support for acquiring location information via the presence event package of SIP ([RFC3856]). A location recipient who supports this option can send a SUBSCRIBE request and parse a resulting NOTIFY containing a PIDF-LO object. The URI schemes supported by this option include "sip", "sips" and "pres".

The "geolocation-http" option tag signals support for acquiring location information via an HTTP ([RFC2616]). A location recipient who supports this option can request location with an HTTP GET and parse a resulting 200 response containing a PIDF-LO object. The URI schemes supported by this option include "http" and "https". A failure to parse the 200 response, for whatever reason, will return a "Dereference Failure" indication to the original location sending user agent to inform it that location was not delivered as intended.

If the location URI receiver does not understand the URI scheme sent to it, it will return an Unsupported header value of the option-tag from the SIP request, and include the option-tag of the preferred URI scheme in the response's Supported header field.

See [ID-GEO-FILTERS] or [ID-HELD-DEREF] for more details on dereferencing location information.

5. Geolocation Examples

5.1 Location-by-value (in Coordinate Format)

This example shows an INVITE message with a coordinate location. In this example, the SIP request uses a sips-URI [RFC3261], meaning this message is protected using TLS on a hop-by-hop basis.

```
INVITE sips:bob@biloxi.example.com SIP/2.0
Via: SIPS/2.0/TLS pc33.atlanta.example.com;branch=z9hG4bK74bf9
Max-Forwards: 70
To: Bob <sips:bob@biloxi.example.com>
From: Alice <sips:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
Geolocation: <cid:target123@atlanta.example.com>
Geolocation-Routing: no
Accept: application/sdp, application/pidf+xml
CSeq: 31862 INVITE
Contact: <sips:alice@atlanta.example.com>
Content-Type: multipart/mixed; boundary=boundary1
```

```
   Content-Length: ...

   --boundary1

   Content-Type: application/sdp

   ...SDP goes here

   --boundary1

   Content-Type: application/pidf+xml
   Content-ID: <target123@atlanta.example.com>
   <?xml version="1.0" encoding="UTF-8"?>
       <presence
          xmlns="urn:ietf:params:xml:ns:pidf"
          xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
          xmlns:gbp="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
          xmlns:cl="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"
          xmlns:gml="http://www.opengis.net/gml"
          xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
          entity="pres:alice@atlanta.example.com">
        <dm:device id="target123-1">
          <gp:geopriv>
            <gp:location-info>
              <gml:location>
                <gml:Point srsName="urn:ogc:def:crs:EPSG::4326">
                  <gml:pos>32.86726 -97.16054</gml:pos>
                </gml:Point>
             </gml:location>
            </gp:location-info>
            <gp:usage-rules>
              <gbp:retransmission-allowed>false
              </gbp:retransmission-allowed>
              <gbp:retention-expiry>2010-11-14T20:00:00Z
              </gbp:retention-expiry>
            </gp:usage-rules>
            <gp:method>802.11</gp:method>
          </gp:geopriv>
          <dm:deviceID>mac:1234567890ab</dm:deviceID>
          <dm:timestamp>2010-11-04T20:57:29Z</dm:timestamp>
        </dm:device>
      </presence>
   --boundary1--
```

The Geolocation header field from the above INVITE:

```
   Geolocation: <cid:target123@atlanta.example.com>
```

... indicates the content-ID location [RFC2392] within the multipart message body of where location information is. The other message body part is SDP. The "cid:" eases message body parsing and disambiguates multiple parts of the same type.

If the Geolocation header field did not contain a "cid:" scheme, for example, it could look like this location URI:

Geolocation: <sips:target123@server5.atlanta.example.com>

... the existence of a non-"cid:" scheme indicates this is a location URI, to be dereferenced to learn the Target's location. Any node wanting to know where the target is located would subscribe to the SIP presence event package [RFC3856] at

sips:target123@server5.atlanta.example.com

(see Figure 2 in Section 3.2 for this message flow).

5.2 Two Locations Composed in Same Location Object Example

This example shows the INVITE message after a SIP intermediary rejected the original INVITE (say, the one in section 5.1). This INVITE contains the composed LO sent by the SIP intermediary which includes where the intermediary understands Alice to be. The rules of RFC 5491 [RFC5491] are followed in this construction.

This example is here, but ought not be taken as occurring very often. In fact, this example is believed to be a corner case of location conveyance applicability.

```
INVITE sips:bob@biloxi.example.com SIP/2.0
Via: SIPS/2.0/TLS pc33.atlanta.example.com;branch=z9hG4bK74bf0
Max-Forwards: 70
To: Bob <sips:bob@biloxi.example.com>
From: Alice <sips:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188512@atlanta.example.com
Geolocation: <cid:target123@atlanta.example.com>
Geolocation-Routing: no
Accept: application/sdp, application/pidf+xml
CSeq: 31863 INVITE
Contact: <sips:alice@atlanta.example.com>
Content-Type: multipart/mixed; boundary=boundary1
Content-Length: ...

--boundary1

Content-Type: application/sdp

...SDP goes here

--boundary1

Content-Type: application/pidf+xml
Content-ID: <target123@atlanta.example.com>
```

```
<?xml version="1.0" encoding="UTF-8"?>
    <presence
       xmlns="urn:ietf:params:xml:ns:pidf"
       xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
       xmlns:gbp="urn:ietf:params:xml:ns:pidf:geopriv10:basicPolicy"
       xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
       xmlns:cl="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"
       xmlns:gml="http://www.opengis.net/gml"
       entity="pres:alice@atlanta.example.com">
     <dm:device id="target123-1">
       <gp:geopriv>
         <gp:location-info>
           <gml:location>
             <gml:Point srsName="urn:ogc:def:crs:EPSG::4326">
               <gml:pos>32.86726 -97.16054</gml:pos>
             </gml:Point>
           </gml:location>
         </gp:location-info>
         <gp:usage-rules>
           <gbp:retransmission-allowed>false
           </gbp:retransmission-allowed>
          <gbp:retention-expiry>2010-11-14T20:00:00Z
           </gbp:retention-expiry>
         </gp:usage-rules>
         <gp:method>802.11</gp:method>
       </gp:geopriv>
       <dm:deviceID>mac:1234567890ab</dm:deviceID>
       <dm:timestamp>2010-11-04T20:57:29Z</dm:timestamp>
     </dm:device>
     <dm:person id="target123">
       <gp:geopriv>
         <gp:location-info>
           <cl:civicAddress>
             <cl:country>US</cl:country>
             <cl:A1>Texas</cl:A1>
             <cl:A3>Colleyville</cl:A3>
             <cl:RD>Treemont</cl:RD>
             <cl:STS>Circle</cl:STS>
             <cl:HNO>3913</cl:HNO>
             <cl:FLR>1</cl:FLR>
             <cl:NAM>Haley's Place</cl:NAM>
             <cl:PC>76034</cl:PC>
           </cl:civicAddress>
         </gp:location-info>
         <gp:usage-rules>
           <gbp:retransmission-allowed>false
           </gbp:retransmission-allowed>
           <gbp:retention-expiry>2010-11-14T20:00:00Z
           </gbp:retention-expiry>
         </gp:usage-rules>
         <gp:method>triangulation</gp:method>
       </gp:geopriv>
```

```
          <dm:timestamp>2010-11-04T12:28:04Z</dm:timestamp>
        </dm:person>
      </presence>
   --boundary1--
```

6. Geopriv Privacy Considerations

Location information is considered by most to be highly sensitive information, requiring protection from eavesdropping and altering in transit. [RFC3693] originally articulated rules to be followed by any protocol wishing to be considered a "Using Protocol", specifying how a transport protocol meets those rules. [RFC6280] updates the guidance in RFC3693 to include subsequently introduced entities and concepts in the geolocation architecture.

RFC5606 explores the difficulties inherent in mapping the GEOPRIV architecture onto SIP elements. In particular, the difficulties of defining and identifying recipients of location information are given in that document, along with guidance in Section 3.3.2 on the use of location by-reference mechanisms to preserve confidentiality of location information from unauthorized recipients.

In a SIP deployment, location information may be added by any of several elements, including the originating user agent or a proxy server. In all cases, the Rule Maker associated with that location information decides which entity adds location information and what access control rules apply. For example, a SIP user agent that does not support the Geolocation header may rely on a proxy server under the direction of the Rule Maker adding a Geolocation header with a reference to location information. The manner in which the Rule Maker operates on these devices is outside the scope of this document.

The manner in which SIP implementations honor the Rule Maker’s stipulations for access control rules (including retention and retransmission) is application-specific and not within the scope of SIP protocol operations. Entities in SIP networks that fulfill the architectural roles of the Location Server or Location Recipient treat the privacy rules associated with location information per the guidance in [RFC6280] section 4.2.1. In particular, RFC4119 (especially 2.2.2) gives guidance for handling access control rules; SIP implementations should furthermore consult the emendations in RFC5606.

7. Security Considerations

Conveyance of physical location of a UA raises privacy concerns, and depending on use, there probably will be authentication and integrity concerns. This document calls for conveyance to be accomplished through secure mechanisms, like S/MIME encrypting

message bodies (although this is not widely deployed), TLS protecting the overall signaling or conveyance location by-reference and requiring all entities that dereference location to authenticate themselves. In location-based routing cases, encrypting the location payload with an end-to-end mechanism such as S/MIME is problematic, because one or more proxies on the path need the ability to read the location information to retarget the message to the appropriate new destination UAS. Data can only be encrypted to a particular, anticipated target, and thus if multiple recipients need to inspect a piece of data, and those recipients cannot be predicted by the sender of data, encryption is not a very feasible choice. Securing the location hop-by-hop, using TLS, protects the message from eavesdropping and modification in transit, but exposes the information to all proxies on the path as well as the endpoint. In most cases, the UA has no trust relationship with the proxy or proxies providing location-based routing services, so such end-to-middle solutions might not be appropriate either.

When location information is conveyed by reference, however, one can properly authenticate and authorize each entity that wishes to inspect location information. This does not require that the sender of data anticipate who will receive data, and it does permit multiple entities to receive it securely, but it does not however obviate the need for pre-association between the sender of data and any prospective recipients. Obviously, in some contexts this pre-association cannot be presumed; when it is not, effectively unauthenticated access to location information must be permitted. In this case, choosing pseudo-random URIs for location by-reference, coupled with path encryption like SIPS, can help to ensure that only entities on the SIP signaling path learn the URI, and thus restores rough parity with sending location by-value.

Location information is especially sensitive when the identity of its Target is obvious. Note that there is the ability, according to [RFC3693] to have an anonymous identity for the Target’s location. This is accomplished by use of an unlinkable pseudonym in the "entity=" attribute of the <presence> element [RFC4479]. Though, this can be problematic for routing messages based on location (covered in the document above). Moreover, anyone fishing for information would correlate the identity at the SIP layer with that of the location information referenced by SIP signaling.

When a UA inserts location, the UA sets the policy on whether to reveal its location along the signaling path - as discussed in Section 4, as well as flags in the PIDF-LO [RFC4119]. UAC implementations MUST make such capabilities conditional on explicit user permission, and MUST alert the user that location is being conveyed.

This SIP extension offers the default ability to require permission to process location while the SIP request is in transit. The default for this is set to "no". There is an error explicitly

describing how an intermediary asks for permission to view the Target's location, plus a rule stating the user has to be made aware of this permission request.

There is no end-to-end integrity on any locationValue or locationErrorValue header field parameter (or middle-to-end if the value was inserted by a intermediary), so recipients of either header field need to implicitly trust the header field contents, and take whatever precautions each entity deems appropriate given this situation.

8. IANA Considerations

The following are the IANA considerations made by this SIP extension. Modifications and additions to all these registrations require a standards track RFC (Standards Action).

[Editor's Note: RFC-Editor - within the IANA section, please replace "this doc" with the assigned RFC number, if this document reaches publication.]

8.1 IANA Registration for the SIP Geolocation Header Field

The SIP Geolocation Header Field is created by this document, with its definition and rules in Section 4.1 of this document, and should be added to the IANA sip-parameters registry with the following actions

1. Update the Header Fields registry with

Registry:

Header Name	compact	Reference
Geolocation		[this doc]

8.2 IANA Registration for the SIP Geolocation-Routing Header Field

The SIP Geolocation-Routing Header Field is created by this document, with its definition and rules in Section 4.2 of this document, and should be added to the IANA sip-parameters registry with the following action

1. Update the Header Fields registry with

Registry:

Header Name	compact	Reference
Geolocation-Routing		[this doc]

8.3 IANA Registration for Location Profiles

This document defines two new SIP option tags: "geolocation-sip" and "geolocation-http" to be added to the IANA sip-parameters Options Tags registry.

Name	Description	Reference
geolocation-sip	The "geolocation-sip" option tag signals support for acquiring location information via the presence event package of SIP (RFC 3856). A location recipient who supports this option can send a SUBSCRIBE request and parse a resulting NOTIFY containing a PIDF-LO object. The URI schemes supported by this option include "sip", "sips" and "pres".	[this doc]
geolocation-http	The "geolocation-http" option tag signals support for acquiring location information via an HTTP ([RFC2616]). A location recipient who supports this option can request location with an HTTP GET and parse a resulting 200 response containing a PIDF-LO object. The URI schemes supported by this option include "http" and "https".	[this doc]

The names of profiles are SIP option-tags, and the guidance in this document does not supersede the option-tag assignment guidance in [RFC3261] (which requires a Standards Action for the assignment of a new option tag). This document does however stipulate that option-tags included to convey the name of a location profile per this definition MUST begin with the string "geolocation" followed by a dash. All such option tags should describe protocols used to acquire location by reference: these tags have no relevance to location carried in SIP requests by value, which use standard MIME typing and negotiation.

8.4 IANA Registration for 424 Response Code

In the SIP Response Codes registry, the following is added

Reference: RFC-XXXX (i.e., this document)
Response code: 424 (recommended number to assign)
Default reason phrase: Bad Location Information

Registry:

Response Code	Reference
Request Failure 4xx	

424 Bad Location Information [this doc]

This SIP Response code is defined in section 4.3 of this document.

8.5 IANA Registration of New Geolocation-Error Header Field

The SIP Geolocation-error header field is created by this document, with its definition and rules in Section 4.4 of this document, to be added to the IANA sip-parameters registry with two actions

1. Update the Header Fields registry with

Registry:

Header Name	compact	Reference
Geolocation-Error		[this doc]

2. In the portion titled "Header Field Parameters and Parameter Values", add

Header Field	Parameter Name	Predefined Values	Reference
Geolocation-Error	code	yes	[this doc]

8.6 IANA Registration for the SIP Geolocation-Error Codes

This document creates a new registry for SIP, called "Geolocation-Error Codes." Geolocation-Error codes provide reason for the error discovered by Location Recipients, categorized by action to be taken by error recipient. The initial values for this registry are shown below.

Registry Name: Geolocation-Error Codes
Reference: [this doc]
Registration Procedures: Specification Required

Code	Default Reason Phrase	Reference
100	"Cannot Process Location"	[this doc]
200	"Permission To Use Location Information"	[this doc]
201	"Permission To Retransmit Location Information to a Third Party"	[this doc]
202	"Permission to Route based on Location Information"	[this doc]
300	"Dereference Failure"	[this doc]

Details of these error codes are in Section 4.4 of this document.

9. Acknowledgements

To Dave Oran for helping to shape this idea.

To Dean Willis for guidance of the effort.

To Allison Mankin, Dick Knight, Hannes Tschofenig, Henning Schulzrinne, James Winterbottom, Jeroen van Bemmel, Jean-Francois Mule, Jonathan Rosenberg, Keith Drage, Marc Linsner, Martin Thomson, Mike Hammer, Ted Hardie, Shida Shubert, Umesh Sharma, Richard Barnes, Dan Wing, Matt Lepinski, John Elwell, Thomas Stach, Jacqueline Lee and Adam Roach for constructive feedback and nits checking.

Special thanks to Paul Kyzivat for his help with the ABNF in this document and to Robert Sparks for many helpful comments and the proper construction of the Geolocation-Error header field.

And finally, to Spencer Dawkins for giving this doc a good scrubbing to make it more readable.

10. References

10.1 Normative References

[RFC3261] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, May 2002.

[RFC4119] J. Peterson, "A Presence-based GEOPRIV Location Object Format", RFC 4119, December 2005

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

[RFC2392] E. Levinson, "Content-ID and Message-ID Uniform Resource Locators", RFC 2392, August 1998

[RFC3856] J. Rosenberg, "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, August 2004

[RFC3859] J. Peterson, "Common Profile for Presence (CPP)", RFC 3859, August 2004

[RFC3428] B. Campbell, Ed., J. Rosenberg, H. Schulzrinne, C. Huitema, D. Gurle, "Session Initiation Protocol (SIP) Extension for

Instant Messaging" , RFC 3428, December 2002

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

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

[RFC6086] C. Holmberg, E. Burger, H. Kaplan, "Session Initiation Protocol (SIP) INFO Method and Package Framework", RFC 6086, January 2011

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

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

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

[RFC4479] J. Rosenberg, "A Data Model for Presence", RFC 4479, July 2006

[RFC4483] E. Berger, "A Mechanism for Content Indirection in SIP", RFC 4483, May 2006

[RFC5491] J. Winterbottom, M. Thomson, H. Tschofenig, "GEOPRIV PIDF-LO Usage Clarification, Considerations, and Recommendations ", RFC 5491, March 2009

[RFC5870] A. Mayrhofer, C. Spanring, "A Uniform Resource Identifier for Geographic Locations (’geo’ URI)", RFC 5870, June 2010

[RFC2616] R. Fielding, J. Gettys, J., Mogul, H. Frystyk, L., Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1", RFC 2616, June 1999

10.2 Informative References

[RFC3693] J. Cuellar, J. Morris, D. Mulligan, J. Peterson. J. Polk, "Geopriv Requirements", RFC 3693, February 2004

[RFC2818] E. Rescorla, "HTTP Over TLS", RFC 2818, May 2000

[RFC5606] J. Peterson, T. Hardie, J. Morris, "Implications of ’retransmission-allowed’ for SIP Location Conveyance", RFC5606, Oct 2008

[ID-GEO-FILTERS] R. Mahy, B. Rosen, H. Tschofenig, "Filtering Location Notifications in SIP", draft-ietf-geopriv-loc-filters, "work

in progress", March 2010

[ID-HELD-DEREF] J. Winterbottom, H. Tschofenig, H. Schulzrinne, M. Thomson, M. Dawson, "A Location Dereferencing Protocol Using HELD", "work in progress", June 2011

[RFC6280] R. Barnes, M. Lepinski, A. Cooper, J, Morris, H. Tschofenig, H. Schulzrinne, "An Architecture for Location and Location Privacy in Internet Applications", draft-ietf-geopriv-arch, "work in progress", October 2010

Authors' Addresses

James Polk
Cisco Systems
3913 Treemont Circle
Colleyville, Texas 76034

33.00111N
96.68142W

Phone: +1-817-271-3552

Email: jmpolk@cisco.com

Brian Rosen
NeuStar, Inc.
470 Conrad Dr.
Mars, PA 16046

40.70497N
80.01252W

Phone: +1 724 382 1051
Email: br@brianrosen.net

Jon Peterson
NeuStar, Inc.

Email: jon.peterson@neustar.biz

Appendix A. Requirements for SIP Location Conveyance

The following subsections address the requirements placed on the UAC, the UAS, as well as SIP proxies when conveying location. This is from the original requirements draft that has since evolved into the solution document (that is above). This has been kept for historical reasons.

If a requirement is not obvious in intent, a motivational statement is included below it.

A.1 Requirements for a UAC Conveying Location

UAC-1 The SIP INVITE Method [RFC3261] must support location conveyance.

UAC-2 The SIP MESSAGE method [RFC3428] must support location conveyance.

UAC-3 SIP Requests within a dialog should support location conveyance.

UAC-4 Other SIP Requests may support location conveyance.

UAC-5 There must be one, mandatory to implement means of transmitting location confidentially.

Motivation: to guarantee interoperability.

UAC-6 It must be possible for a UAC to update location conveyed at any time in a dialog, including during dialog establishment.

Motivation: if a UAC has moved prior to the establishment of a dialog between UAs, the UAC must be able to send location information. If location has been conveyed, and the UA moves, the UAC must be able to update the location previously conveyed to other parties.

UAC-7 The privacy and security rules established within [RFC3693] that would categorize SIP as a 'Using Protocol' MUST be met.

UAC-8 The PIDF-LO [RFC4119] is a mandatory to implement format for location conveyance within SIP.

Motivation: interoperability with other IETF location protocols and Mechanisms.

UAC-9 There must be a mechanism for the UAC to request the UAS send its location.

UAC-9 has been DEPRECATED by the SIP WG, due to the many problems this requirement would have caused if implemented. The solution is for the above UAS to send a new request to the original UAC with the UAS's location.

UAC-10 There must be a mechanism to differentiate the ability of the UAC to convey location from the UACs lack of knowledge of its location

Motivation: Failure to receive location when it is expected can happen because the UAC does not implement this extension, or because the UAC implements the extension, but does not know where the Target is. This may be, for example, due to the failure of the access network to provide a location acquisition mechanism the UAC supports. These cases must be differentiated.

UAC-11 It must be possible to convey location to proxy servers along the path.

Motivation: Location-based routing.

A.2 Requirements for a UAS Receiving Location

The following are the requirements for location conveyance by a UAS:

UAS-1 SIP Responses must support location conveyance.

The SIPCORE WG reached consensus that this be allowed, but not to communicate the UAS's location; rather for a SIP intermediary to inform the UAC which location to include in its next SIP request (as a matter of correcting what was originally sent by the UAC).

UAS-2 There must be a unique 4XX response informing the UAC it did not provide applicable location information.

In addition, requirements UAC-5, 6, 7 and 8 also apply to the UAS.

A.3 Requirements for SIP Proxies and Intermediaries

The following are the requirements for location conveyance by a SIP proxies and intermediaries:

Proxy-1 Proxy servers must be capable of adding a Location header field during processing of SIP requests.

Motivation: Provide network assertion of location when UACs are unable to do so, or when network assertion is more reliable than UAC assertion of location

Note: Because UACs connected to SIP signaling networks can have widely varying access network arrangements, including VPN tunnels and roaming mechanisms, it can be difficult for a network to reliably know the location of the endpoint. Proxies SHOULD NOT assert location of an endpoint unless the SIP signaling network has reliable knowledge of the actual location of the Targets.

Proxy-2 There must be a unique 4XX response informing the UAC it did not provide applicable location information.

SIPCORE
Internet-Draft
Updates: 3261 (if approved)
Intended status: Standards Track
Expires: June 22, 2011

G. Camarillo, Ed.
C. Holmberg
Ericsson
Y. Gao
ZTE

December 19, 2010

Re-INVITE and Target-refresh Request Handling in the Session Initiation Protocol (SIP)
draft-ietf-sipcore-reinvite-08

Abstract

The procedures for handling SIP re-INVITEs are described in RFC 3261. Implementation and deployment experience has uncovered a number of issues with the original documentation, and this document provides additional procedures that update the original specification to address those issues. In particular, this document defines in which situations a UAS (User Agent Server) should generate a success response and in which situations a UAS should generate an error response to a re-INVITE. Additionally, this document defines further details of procedures related to target-refresh requests.

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 June 22, 2011.

Table of Contents

1. Introduction

As discussed in Section 14 of RFC 3261 [RFC3261], an INVITE request sent within an existing dialog is known as a re-INVITE. A re-INVITE is used to modify session parameters, dialog parameters, or both. That is, a single re-INVITE can change both the parameters of its associated session (e.g., changing the IP address where a media stream is received) and the parameters of its associated dialog (e.g., changing the remote target of the dialog). A re-INVITE can change the remote target of a dialog because it is a target refresh request, as defined in Section 6 of RFC 3261 [RFC3261].

A re-INVITE transaction has an offer/answer [RFC3264] exchange associated to it. The UAC (User Agent Client) generating a given re-INVITE can act as the offerer or as the answerer. A UAC willing to act as the offerer includes an offer in the re-INVITE. The UAS (User Agent Server) then provides an answer in a response to the re-INVITE. A UAC willing to act as answerer does not include an offer in the re-INVITE. The UAS then provides an offer in a response to the re-INVITE becoming, thus, the offerer.

Certain transactions within a re-INVITE (e.g., UPDATE [RFC3311] transactions) can also have offer/answer exchanges associated to them. A UA (User Agent) can act as the offerer or the answerer in any of these transactions regardless of whether the UA was the offerer or the answerer in the umbrella re-INVITE transaction.

There has been some confusion among implementors regarding how a UAS should handle re-INVITEs. In particular, implementors requested clarification on which type of response a UAS should generate in different situations. In this document, we clarify these issues.

Additionally, there has also been some confusion among implementors regarding target refresh requests, which include but are not limited to re-INVITEs. In this document, we also clarify the process by which remote targets are refreshed.

> Indented passages such as this one are used in this document to provide additional information and clarifying text. They do not contain normative protocol behavior.

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].

UA: User Agent.

UAC: User Agent Client.

UAS: User Agent Server.

> Note that the terms UAC and UAS are used with respect to an INVITE or re-INVITE transaction and do not necessarily reflect the role of the UA concerned with respect to any other transaction, such as an UPDATE transaction occurring within the INVITE transaction.

3. Changing the Session State During a Re-INVITE

The following sections discuss how to change the state of the session during a re-INVITE transaction.

3.1. Background on Re-INVITE Handling by UASs

A UAS receiving a re-INVITE will need to, eventually, generate a response to it. Some re-INVITEs can be responded to immediately because their handling does not require user interaction (e.g., changing the IP address where a media stream is received). The handling of other re-INVITEs requires user interaction (e.g., adding a video stream to an audio-only session). Therefore, these re-INVITEs cannot be responded to immediately.

An error response to a re-INVITE has the following semantics. As specified in Section 12.2.2 of RFC 3261 [RFC3261], if a re-INVITE is rejected, no state changes are performed. These state changes include state changes associated to the re-INVITE transaction and all other transactions within the re-INVITE (this section deals with changes to the session state; target refreshes are discussed in Section 4.2). That is, the session state is the same as before the re-INVITE was received. The example in Figure 1 illustrates this point.

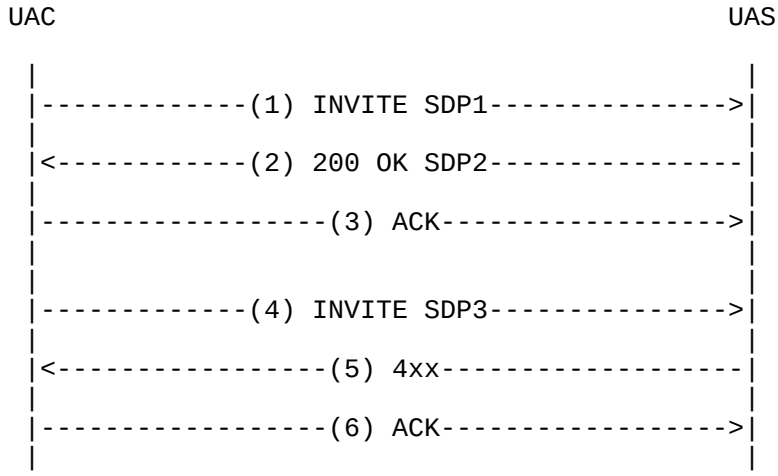


Figure 1: Rejection of a re-INVITE

The UAs perform an offer/answer exchange to establish an audio-only session:

```
SDP1:
   m=audio 30000 RTP/AVP 0

SDP2:
   m=audio 31000 RTP/AVP 0
```

At a later point, the UAC sends a re-INVITE (4) in order to add a video stream to the session.

```
SDP3:
   m=audio 30000 RTP/AVP 0
   m=video 30002 RTP/AVP 31
```

The UAS is configured to automatically reject video streams. Consequently, the UAS returns an error response (5). At that point, the session parameters in use are still those resulting from the initial offer/answer exchange, which are described by SDP1 and SDP2. That is, the session state is the same as before the re-INVITE was received.

In the previous example, the UAS rejected all the changes requested in the re-INVITE by returning an error response. However, there are

situations where a UAS wants to accept some but not all the changes requested in a re-INVITE. In these cases, the UAS generates a 200 (OK) response with an SDP indicating which changes were accepted and which were not. The example in Figure 2 illustrates this point.

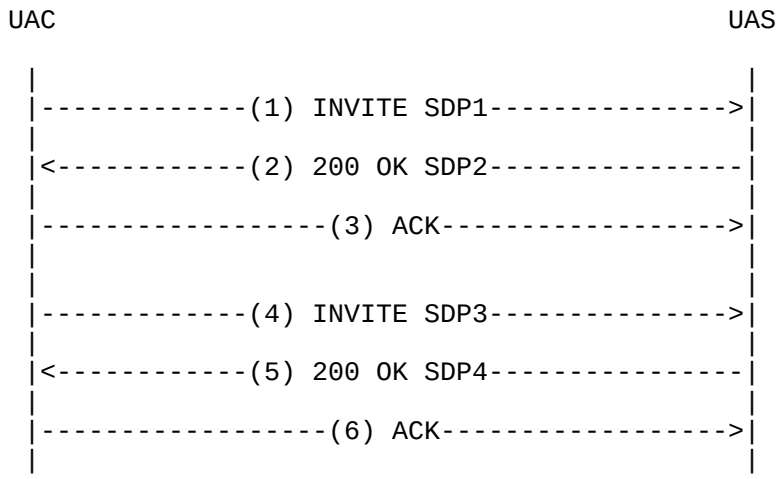


Figure 2: Automatic rejection of a video stream

The UAs perform an offer/answer exchange to establish an audio only session:

```
SDP1:
   m=audio 30000 RTP/AVP 0
   c=IN IP4 192.0.2.1

SDP2:
   m=audio 31000 RTP/AVP 0
   c=IN IP4 192.0.2.5
```

At a later point, the UAC moves to an access that provides a higher-bandwidth. Therefore, the UAC sends a re-INVITE (4) in order to change the IP address where it receives the audio stream to its new IP address and add a video stream to the session.

```
SDP3:
   m=audio 30000 RTP/AVP 0
   c=IN IP4 192.0.2.2
   m=video 30002 RTP/AVP 31
   c=IN IP4 192.0.2.2
```

The UAS is automatically configured to reject video streams. However, the UAS needs to accept the change of the audio stream's remote IP address. Consequently, the UAS returns a 200 (OK) response and sets the port of the video stream to zero in its SDP.

```
SDP4:
   m=audio 31000 RTP/AVP 0
   c=IN IP4 192.0.2.5
   m=video 0 RTP/AVP 31
```

In the previous example, the UAS was configured to automatically reject the addition of video streams. The example in Figure 3 assumes that the UAS requires its user's input in order to accept or reject the addition of a video stream and uses reliable provisional responses [RFC3262] (PRACK transactions are not shown for clarity).

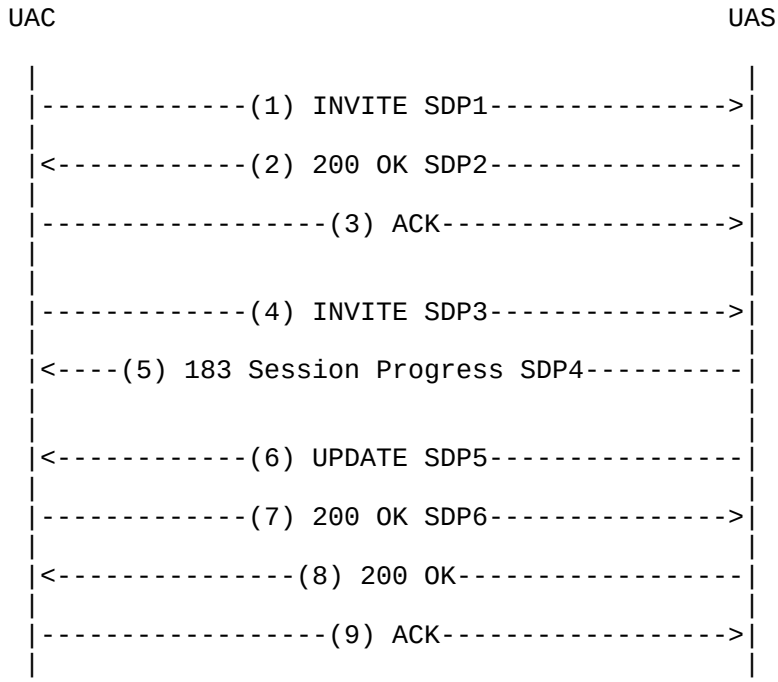


Figure 3: Manual rejection of a video stream by the user

Everything up to (4) is identical to the previous example. In (5), the UAS accepts the change of the audio stream's remote IP address but does not accept the video stream yet (it provides a null IP address instead of setting the stream to 'inactive' because inactive streams still need to exchange RTCP traffic).

```
SDP4:
   m=audio 31000 RTP/AVP 0
   c=IN IP4 192.0.2.5
   m=video 31002 RTP/AVP 31
   c=IN IP4 0.0.0.0
```

At a later point, the UAS's user rejects the addition of the video stream. Consequently, the UAS sends an UPDATE request (6) setting the port of the video stream to zero in its offer.

```
SDP5:
   m=audio 31000 RTP/AVP 0
   c=IN IP4 192.0.2.5
   m=video 0 RTP/AVP 31
   c=IN IP4 0.0.0.0
```

The UAC returns a 200 (OK) response (7) to the UPDATE with the following answer:

```
SDP6:
   m=audio 30000 RTP/AVP 0
   c=IN IP4 192.0.2.2
   m=video 0 RTP/AVP 31
```

The UAS now returns a 200 (OK) response (8) to the re-INVITE.

In all the previous examples, the UAC of the re-INVITE transaction was the offerer. Examples with UACs acting as the answerers would be similar.

3.2. Problems with Error Responses and Already-executed Changes

Section 3.1 contains examples on how a UAS rejects all the changes requested in a re-INVITE without executing any of them by returning an error response (Figure 1), and how a UAS executes some of the changes requested in a re-INVITE and rejects some of them by returning a 2xx response (Figure 2 and Figure 3). A UAS can accept

and reject different sets of changes simultaneously (Figure 2) or at different times (Figure 3).

The scenario that created confusion among implementors consists of a UAS that receives a re-INVITE, executes some of the changes requested in it, and then wants to reject all those already-executed changes and revert to the pre-re-INVITE state. Such a UAS may consider returning an error response to the re-INVITE (the message flow would be similar to the one in Figure 1), or using an UPDATE request to revert to the pre-re-INVITE state and then returning a 2xx response to the re-INVITE (the message flow would be similar to the one in Figure 3). This section explains the problems associated with returning an error response in these circumstances. In order to avoid these problems, the UAS should use the latter option (UPDATE request plus a 2xx response). Section 3.3 and Section 3.4 contain the normative statements needed to avoid these problems.

The reason for not using an error response to undo already executed changes is that an error response to a re-INVITE for which changes have already been executed (e.g., as a result of UPDATE transactions or reliable provisional responses) is effectively requesting a change in the session state. However, the UAC has no means to reject that change if it is unable to execute them. That is, if the UAC is unable to revert to the pre-re-INVITE state, it will not be able to communicate this fact to the UAS.

3.3. UAS Behavior

UASs should only return an error response to a re-INVITE if no changes to the session state have been executed since the re-INVITE was received. Such an error response indicates that no changes have been executed as a result of the re-INVITE or any other transaction within it.

If any of the changes requested in a re-INVITE or in any transaction within it have already been executed, the UAS SHOULD return a 2xx response.

A change to the session state is considered to have been executed if an offer/answer without preconditions [RFC4032] for the stream has completed successfully or the UA has sent or received media using the new parameters. Connection establishment messages (e.g., TCP SYN), connectivity checks (e.g., when using ICE [RFC5245]), and any other messages used in the process of meeting the preconditions for a stream are not considered media.

Normally, a UA receiving media can easily detect when the new parameters for the media stream are used (e.g., media is received on a new port). However, in some scenarios the UA will have to process incoming media packets in order to detect whether they use the old or the new parameters.

The successful completion of an offer/answer exchange without preconditions indicates that the new parameters for the media stream are already considered to be in use. The successful completion of an offer/answer exchange with preconditions means something different. The fact that all mandatory preconditions for the stream are met indicates that the new parameters for the media stream are ready to be used. However, they will not actually be used until the UAS decides to use them. During a session establishment, the UAS can wait before using the media parameters until the callee starts being alerted or until the callee accepts the session. During a session modification, the UAS can wait until its user accepts the changes to the session. When dealing with streams where the UAS sends media more or less continuously, the UAC notices that the new parameters are in use because the UAC receives media that uses the new parameters. However, this mechanism does not work with other types of streams. Therefore, it is RECOMMENDED that when a UAS decides to start using the new parameters for a stream for which all mandatory preconditions have been met, the UAS either sends media using the new parameters or sends a new offer where the precondition-related attributes for the stream have been removed. As indicated above, the successful completion of an offer/answer exchange without preconditions indicates that the new parameters for the media stream are already considered to be in use.

3.4. UAC Behavior

A UAC that receives an error response to a re-INVITE that undoes already-executed changes within the re-INVITE may be facing a legacy UAS that does not support this specification (i.e., a UAS that does not follow the guidelines in Section 3.3). There are also certain race condition situations that get both user agents out of synchronization. In order to cope with these race condition situations, a UAC that receives an error response to a re-INVITE for which changes have been already executed SHOULD generate a new re-INVITE or UPDATE request in order to make sure that both UAs have a common view of the state of the session (the UAC uses the criteria in Section 3.3 in order to decide whether or not changes have been executed for a particular stream). The purpose of this new offer/answer exchange is to synchronize both UAs, not to request changes that the UAS may choose to reject. Therefore, session parameters in the offer/answer exchange SHOULD be as close to those in the pre-re-INVITE state as possible.

3.5. Glare Situations

Section 4 of RFC 3264 [RFC3264] defines glare conditions as a user agent receiving an offer after having sent one but before having received an answer to it. That section specifies rules to avoid glare situations in most cases. When despite following those rules a glare conditions occurs (as a result of a race condition), it is handled as specified in Sections 14.1 and 14.2 of RFC 3261 [RFC3261]. The UAS returns a 491 (Request Pending) response and the UAC retries the offer after a randomly-selected time, which depends on which user agent is the owner of the Call-ID of the dialog. The rules in RFC 3261 [RFC3261] not only cover collisions between re-INVITEs that contain offers; they cover collisions between two re-INVITEs in general, even if they do not contain offers. Sections 5.2 and 5.3 of RFC 3311 [RFC3311] extend those rules to also cover collisions between an UPDATE request carrying an offer and another message (UPDATE, PRACK or INVITE) also carrying an offer.

The rules in RFC 3261 [RFC3261] do not cover collisions between an UPDATE request and a non-2xx final response to a re-INVITE. Since both the UPDATE request and the reliable response could be requesting changes to the session state, it would not be clear which changes would need to be executed first. However, the procedures discussed in Section 3.4 already cover this type of situation. Therefore, there is no need to specify further rules here.

3.6. Example of UAS Behavior

This section contains an example of a UAS that implements this specification using an UPDATE request and a 2xx response to a re-INVITE in order to revert to the pre-re-INVITE state. The example, which is shown in Figure 4, assumes that the UAS requires its user’s input in order to accept or reject the addition of a video stream and uses reliable provisional responses [RFC3262] (PRACK transactions are not shown for clarity).

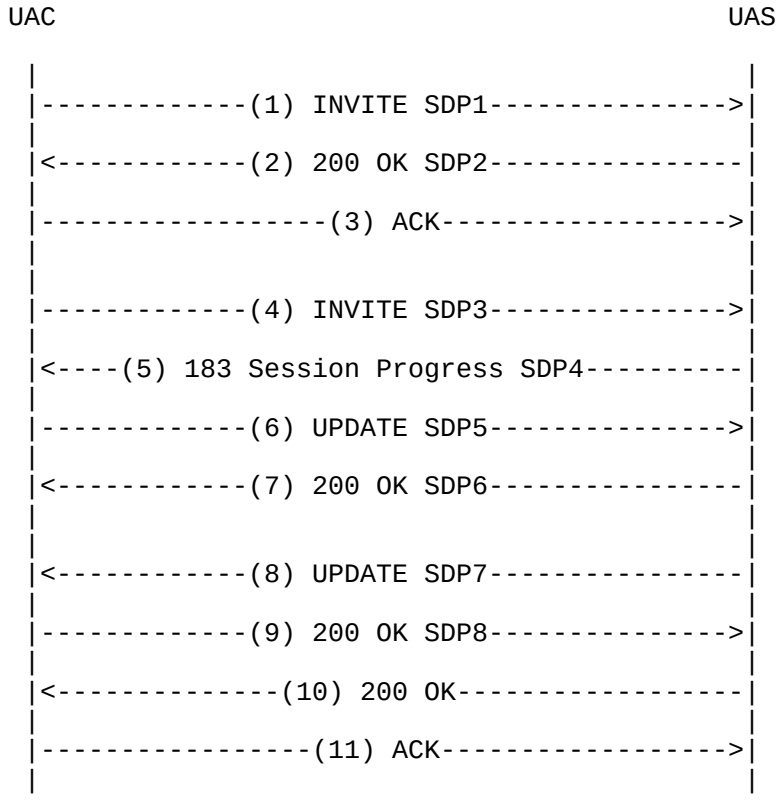


Figure 4: Rejection of a video stream by the user

The UAs perform an offer/answer exchange to establish an audio only session:

```
SDP1:
   m=audio 30000 RTP/AVP 0
   c=IN IP4 192.0.2.1

SDP2:
   m=audio 31000 RTP/AVP 0
   c=IN IP4 192.0.2.5
```

At a later point, the UAC sends a re-INVITE (4) in order to add a new codec to the audio stream and to add a video stream to the session.

```
SDP3:
   m=audio 30000 RTP/AVP 0 3
   c=IN IP4 192.0.2.1
   m=video 30002 RTP/AVP 31
   c=IN IP4 192.0.2.1
```

In (5), the UAS accepts the addition of the audio codec but does not accept the video stream yet (it provides a null IP address instead of setting the stream to 'inactive' because inactive streams still need to exchange RTCP traffic).

```
SDP4:
   m=audio 31000 RTP/AVP 0 3
   c=IN IP4 192.0.2.5
   m=video 31002 RTP/AVP 31
   c=IN IP4 0.0.0.0
```

At a later point, the UAC sends an UPDATE request (6) to remove the original audio codec from the audio stream (the UAC could have also used the PRACK to (5) to request this change).

```
SDP5:
   m=audio 30000 RTP/AVP 3
   c=IN IP4 192.0.2.1
   m=video 30002 RTP/AVP 31
   c=IN IP4 192.0.2.1

SDP6:
   m=audio 31000 RTP/AVP 3
   c=IN IP4 192.0.2.5
   m=video 31002 RTP/AVP 31
   c=IN IP4 0.0.0.0
```

Yet at a later point, the UAS's user rejects the addition of the video stream. Additionally, the UAS decides to revert to the original audio codec. Consequently, the UAS sends an UPDATE request (8) setting the port of the video stream to zero and offering the original audio codec in its SDP.

```
SDP7:
   m=audio 31000 RTP/AVP 0
   c=IN IP4 192.0.2.5
   m=video 0 RTP/AVP 31
   c=IN IP4 0.0.0.0
```

The UAC accepts the change in the audio codec in its 200 (OK) response (9) to the UPDATE request.

```
SDP8:
   m=audio 30000 RTP/AVP 0
   c=IN IP4 192.0.2.1
   m=video 0 RTP/AVP 31
   c=IN IP4 192.0.2.1
```

The UAS now returns a 200 (OK) response (10) to the re-INVITE. Note that the media state after this 200 (OK) response is the same as the pre-re-INVITE media state.

3.7. Example of UAC Behavior

Figure 5 shows an example of a race condition situation in which the UAs end up with different views of the state of the session.

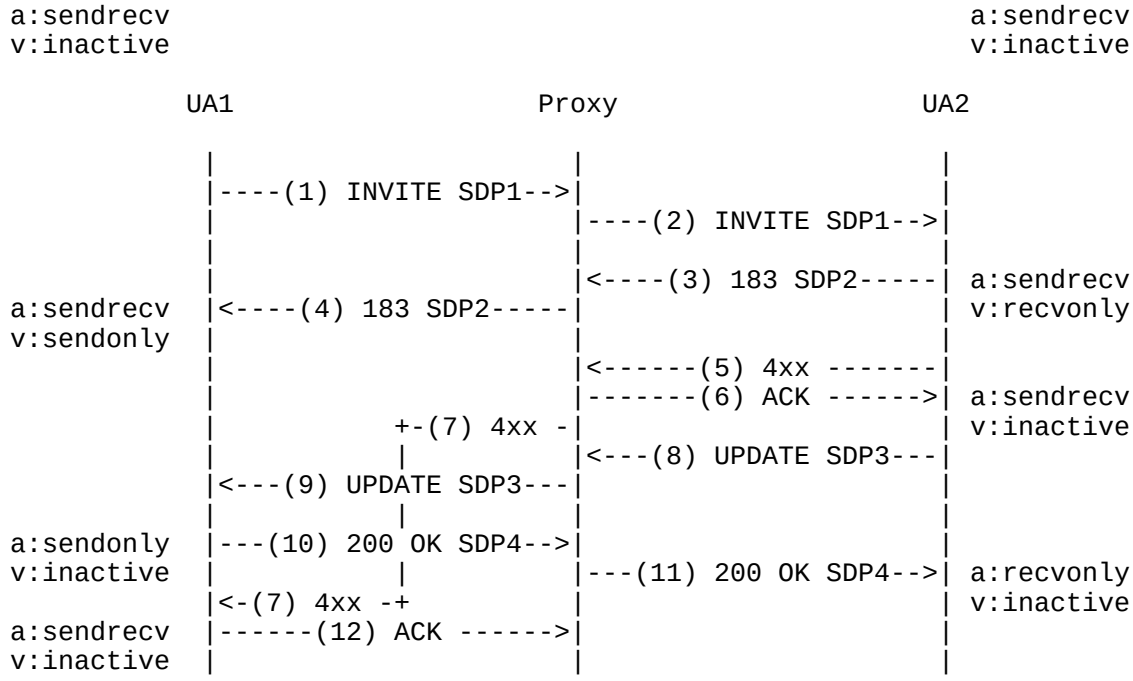


Figure 5: Message flow with race condition

The UAs in Figure 5 are involved in a session that, just before the message flows in the figures starts, includes a sendrecv audio stream and an inactive video stream. UA1 sends a re-INVITE (1) requesting to make the video stream sendrecv.

```
SDP1:
   m=audio 20000 RTP/AVP 0
   a=sendrecv
   m=video 20002 RTP/AVP 31
   a=sendrecv
```

UA2 is configured to automatically accept incoming video streams but to ask for user input before generating an outgoing video stream. Therefore, UAS2 makes the video stream recvonly by returning a 183 (Session Progress) response (2).

```
SDP2:
   m=audio 30000 RTP/AVP 0
   a=sendrecv
   m=video 30002 RTP/AVP 31
   a=recvonly
```

When asked for input, UA2’s user chooses not to have either incoming or outgoing video. In order to make the video stream inactive, UA2 returns a 4xx error response (5) to the re-INVITE. The ACK request (6) for this error response is generated by the proxy between both user agents. Note that this error response undoes already-executed changes. So, UA2 is a legacy UA that does not support this specification.

The proxy relays the 4xx response (7) towards UA1. However, the 4xx response (7) takes time to arrive to UA1 (e.g., the response may have been sent over UDP and the first few retransmissions were lost). In the meantime, UA2’s user decides to put the audio stream on hold. UA2 sends an UPDATE request (8) making the audio stream recvonly. The video stream, which is inactive, is not modified and, thus, continues being inactive.

```
SDP3:
   m=audio 30000 RTP/AVP 0
   a=recvonly
   m=video 30002 RTP/AVP 31
   a=inactive
```

The proxy relays the UPDATE request (9) to UA1. The UPDATE request (9) arrives at UA1 before the 4xx response (7) that had been previously sent. UA1 accepts the changes in the UPDATE request and returns a 200 (OK) response (10) to it.

```
SDP4:
   m=audio 20000 RTP/AVP 0
   a=sendonly
   m=video 30002 RTP/AVP 31
   a=inactive
```

At a later point, the 4xx response (7) finally arrives at UA1. This response makes the session return to its pre-re-INVITE state. Therefore, for UA1, the audio stream is sendrecv and the video stream is inactive. However, for UA2, the audio stream is recvonly (the video stream is also inactive).

After the message flow in Figure 5, following the recommendations in this section, when UA1 received an error response (7) that undid already-executed changes, UA1 would generate an UPDATE request with an SDP reflecting the pre-re-INVITE state (i.e., sendrecv audio and inactive video). UA2 could then return a 200 (OK) response to the UPDATE request making the audio stream recvonly, which is the state UA2's user had requested. Such an UPDATE transaction would get the UAs back into synchronization.

3.8. Clarifications on Cancelling Re-INVITEs

Section 9.2 of RFC 3261 [RFC3261] specifies the behavior of a UAS responding to a CANCEL request. Such a UAS responds to the INVITE request with a 487 (Request Terminated) at the 'should' level. Per the rules specified in Section 3.3, if the INVITE request was a re-INVITE and some of its requested changes had already been executed, the UAS would return a 2xx response instead.

4. Refreshing a Dialog's Targets

The following sections discuss how to refresh the targets of a dialog.

4.1. Background and Terminology on a Dialog's Targets

As described in Section 12 of RFC 3261 [RFC3261], a UA involved in a dialog keeps a record of the SIP or SIPS URI at which it can communicate with a specific instance of its peer (this is called the "dialog's remote target URI" and is equal to the URI contained in the

Contact header of requests and responses it receives from the peer). This document introduces the complementary concept of the "dialog's local target URI", defined as a UA's record of the SIP or SIPS URI at which the peer can communicate with it (equal to the URI contained in the Contact header of requests and responses it sends to the peer). These terms are complementary because the "dialog's remote target URI" according to one UA is the "dialog's local target URI" according to the other UA, and vice-versa.

4.2. Background on Target-refresh Requests

A target-refresh request is defined as follows in Section 6 of RFC 3261 [RFC3261]:

> "A target-refresh request sent within a dialog is defined as a request that can modify the remote target of the dialog."

Additionally, 2xx responses to target-refresh requests can also update the remote target of the dialog. As discussed in Section 12.2 of RFC 3261 [RFC3261], re-INVITEs are target-refresh requests.

RFC 3261 [RFC3261] specifies the behavior of UASs receiving target-refresh requests and of UACs receiving a 2xx response for a target-refresh request.

Section 12.2.2 of RFC 3261 [RFC3261] says:

> "When a UAS receives a target-refresh request, it MUST replace the dialog's remote target URI with the URI from the Contact header field in that request, if present."

Section 12.2.1.2 of RFC 3261 [RFC3261] says:

> "When a UAC receives a 2xx response to a target-refresh request, it MUST replace the dialog's remote target URI with the URI from the Contact header field in that response, if present."

The fact that re-INVITEs can be long-lived transactions and can have other transactions within them makes it necessary to revise these rules. Section 4.3 specifies new rules for the handling of target-refresh requests. Note that the new rules apply to any target-refresh request, not only to re-INVITEs.

4.3. Clarification on the Atomicity of Target-Refresh Requests

The local and remote targets of a dialog are special types of state information because of their essential role in the exchange of SIP messages between UAs in a dialog. A UA involved in a dialog receives

the remote target of the dialog from the remote UA. The UA uses the received remote target to send SIP requests to the remote UA.

The dialog's local target is a piece of state information that is not meant to be negotiated. When a UA changes its local target (i.e., the UA changes its IP address), the UA simply communicates its new local target to the remote UA (e.g., the UA communicates its new IP address to the remote UA in order to remain reachable by the remote UA). UAs need to follow the behavior specified in Section 4.4, Section 4.5, Section 4.6, and Section 4.7 of this specification instead of that specified in RFC 3261 [RFC3261], which was discussed in Section 4.2. The new behavior regarding target-refresh requests implies that a target-refresh request can, in some cases, update the remote target even if the request is responded with a final error response. This means that target-refresh requests are not atomic.

4.4. UA Updating the Dialog's Local Target in a Request

In order to update its local target, a UA can send a target-refresh request. If the UA receives an error response to the target-refresh request, the remote UA has not updated its remote target.

> This allows UASs to authenticate target-refresh requests (see Section 26.2 of RFC 3261 [RFC3261]).

If the UA receives a reliable provisional response or a 2xx response to the target-refresh request, or the UA receives an in-dialog request on the new local target, the remote UA has updated its remote target. The UA can consider the target refresh operation completed.

> Even if the target request was a re-INVITE and the final response to the re-INVITE was an error response, the UAS would not revert to the pre-re-INVITE remote target.

A UA SHOULD NOT use the same target refresh request to refresh the target and to make session changes unless the session changes can be trivially accepted by the remote UA (e.g., an IP address change). Piggybacking a target refresh with more complicated session changes would make it unnecessarily complicated for the remote UA to accept the target refresh while rejecting the session changes. Only in case the target refresh request is a re-INVITE and the UAS supports reliable provisional response or UPDATE requests, the UAC MAY piggyback session changes and a target refresh in the same re-INVITE.

4.5. UA Updating the Dialog's Local Target in a Response

A UA processing an incoming target refresh request can update its local target by returning a reliable provisional response or a 2xx

response to the target-refresh request. The response needs to contain the updated local target URI in its Contact header field. On sending the response, the UA can consider the target refresh operation completed.

4.6. A Request Updating the Dialog's Remote Target

Behavior of a UA after having received a target-refresh request updating the remote target:

If the UA receives a target-refresh request that has been properly authenticated (see Section 26.2 of RFC 3261 [RFC3261]), the UA SHOULD generate a reliable provisional response or a 2xx response to the target-refresh request. If generating such responses is not possible (e.g., the UA does not support reliable provisional responses and needs user input before generating a final response), the UA SHOULD send an in-dialog request to the remote UA using the new remote target (if the UA does not need to send a request for other reasons, the UAS can send an UPDATE request). On sending a reliable provisional response or a 2xx response to the target-refresh request, or a request to the new remote target, the UA MUST replace the dialog's remote target URI with the URI from the Contact header field in the target-refresh request.

> Reliable provisional responses in SIP are specified in RFC 3262 [RFC3262]. In this document, reliable provisional responses are those that use the mechanism defined in RFC 3262 [RFC3262]. Other specifications may define ways to send provisional responses reliably using non-SIP mechanisms (e.g., using media-level messages to acknowledge the reception of the SIP response). For the purposes of this document, provisional responses using those non-SIP mechanisms are considered unreliable responses. Note that non-100 provisional responses are only applicable to INVITE transactions [RFC4320].

If instead of sending a reliable provisional response or a 2xx response to the target-refresh request, or a request to the new target, the UA generates an error response to the target-refresh request, the UA MUST NOT update its dialog's remote target.

4.7. A Response Updating the Dialog's Remote Target

If a UA receives a reliable provisional response or a 2xx response to a target-refresh request, the UA MUST replace the dialog's remote target URI with the URI from the Contact header field in that response, if present.

If a UA receives an unreliable provisional response to a target-

refresh request, the UA MUST NOT refresh the dialog's remote target.

4.8. Race Conditions and Target Refreshes

SIP provides request ordering by using the Cseq header field. That is, a UA that receives two requests at roughly the same time can know which one is newer. However, SIP does not provide ordering between responses and requests. For example, if a UA receives a 200 (OK) response to an UPDATE request and an UPDATE request at roughly the same time, the UA cannot know which one was sent last. Since both messages can refresh the remote target, the UA needs to know which message was sent last in order to know which remote target needs to be used.

This document specifies the following rule to avoid the situation just described. If the protocol allows a UA to use a target-refresh request at the point in time that UA wishes to refresh its local target, the UA MUST use a target-refresh request instead of a response to refresh its local target. This rule implies that a UA only uses a response (i.e., a reliable provisional response or a 2xx response to a target-refresh request) to refresh its local target if the UA is unable to use a target-refresh request at that point in time (e.g., the UAS of an ongoing re-INVITE without support for UPDATE).

4.9. Early Dialogs

The rules given in this section about which messages can refresh the target of a dialog also apply to early dialogs created by an initial INVITE transaction. Additionally, as specified in Section 13.2.2.4 of RFC 3261 [RFC3261], on receiving a 2xx response to the initial INVITE, the UAC recomputes the whole route set of the dialog, which transitions from the "early" state to the "confirmed" state.

Section 12.1 of RFC 3261 allows unreliable provisional responses to create early dialogs. However, per the rules given in this section, unreliable provisional responses cannot refresh the target of a dialog. Therefore, the UAC of an initial INVITE transaction will not perform any target refresh as a result of the reception of an unreliable provisional response with an updated Contact value on an (already-established) early dialog. Note also that a given UAS can establish additional early dialogs, which can have different targets, by returning additional unreliable provisional responses with different To tags.

5. A UA Losing its Contact

The following sections discuss the case where a UA loses its transport address during an ongoing re-INVITE transaction. Such a UA will refresh the dialog's local target so that it reflects its new transport address. Note that target refreshes that do not involve changes in the UA's transport address are outside of the scope of this section. Also, UAs losing their transport address during a non-re-INVITE transaction (e.g., a UA losing its transport address right after having sent an UPDATE request before having received a response to it) are out of scope as well.

The rules given in this section are also applicable to initial INVITE requests that have established early dialogs.

5.1. Background on re-INVITE Transaction Routing

Re-INVITEs are routed using the dialog's route set, which contains all the proxy servers that need to be traversed by requests sent within the dialog. Responses to the re-INVITE are routed using the Via entries in the re-INVITE.

ACK requests for 2xx responses and for non-2xx final responses are generated in different ways. As specified in Sections 14.1 and 13.2.1 of RFC 3261 [RFC3261], ACK requests for 2xx responses are generated by the UAC core and are routed using the dialog's route set. As specified in Section 17.1.1.2 of RFC 3261 [RFC3261], ACK requests for non-2xx final responses are generated by the INVITE client transaction (i.e., they are generated in a hop-by-hop fashion by the proxy servers in the path) and are sent to the same transport address as the re-INVITE.

5.2. Problems with UAs Losing their Contact

Refreshing the dialog's remote target during a re-INVITE transaction (see Section 4.3) presents some issues because of the fact that re-INVITE transactions can be long lived. As described in Section 5.1, the way responses to the re-INVITE and ACKs for non-2xx final responses are routed is fixed once the re-INVITE is sent. The routing of this messages does not depend on the dialog's route set and, thus, target refreshes within an ongoing re-INVITE do not affect their routing. A UA that changes its location (i.e., performs a target refresh) but is still reachable at its old location will be able to receive those messages (which will be sent to the old location). However, a UA that cannot be reachable at its old location any longer will not be able to receive them.

The following sections describe the errors UAs face when they lose

their transport address during a re-INVITE. On detecting some of these errors, UAs following the rules specified in RFC 3261 [RFC3261] will terminate the dialog. When the dialog is terminated, the only option for the UAs is to establish a new dialog. The following sections change the requirements RFC 3261 [RFC3261] places on UAs when certain errors occur so that the UAs can recover from those errors. In short, the UAs generate a new re-INVITE transaction to synchronize both UAs. Note that there are existing UA implementations deployed that already implement this behavior.

5.3. UAS Losing its Contact: UAC Behavior

When a UAS that moves to a new contact and loses its old contact generates a non-2xx final response to the re-INVITE, it will not be able to receive the ACK request. The entity receiving the response and, thus, generating the ACK request will either get a transport error or a timeout error, which, as described in Section 8.1.3.1 of RFC 3261 [RFC3261], will be treated as a 503 (Service Unavailable) response and as a 408 (Request Timeout) response, respectively. If the sender of the ACK request is a proxy server, it will typically ignore this error. If the sender of the ACK request is the UAC, according to Section 12.2.1.2 of RFC 3261 [RFC3261], it is supposed to (at the "should" level) terminate the dialog by sending a BYE request. However, because of the special properties of ACK requests for non-2xx final responses, most existing UACs do not terminate the dialog when ACK request fails, which is fortunate.

A UAC that accepts a target refresh within a re-INVITE MUST ignore transport and timeout errors when generating an ACK request for a non-2xx final response. Additionally, UAC SHOULD generate a new re-INVITE in order to make sure that both UAs have a common view of the state of the session.

> It is possible that the errors ignored by the UAC were not related to the target refresh operation. If that was the case, the second re-INVITE would fail and the UAC would terminate the dialog because, per the rules above, UACs only ignore errors when they accept a target refresh within the re-INVITE.

5.4. UAC Losing its Contact: UAS Behavior

When a UAC moves to a new contact and loses its old contact, it will not be able to receive responses to the re-INVITE. Consequently, it will never generate an ACK request.

As described in Section 16.9 of RFC 3261 [RFC3261], a proxy server that gets an error when forwarding a response does not take any measures. Consequently, proxy servers relaying responses will

effectively ignore the error.

If there are no proxy servers in the dialog's route set, the UAS will get an error when sending a non-2xx final response. The UAS core will be notified of the transaction failure, as described in Section 17.2.1 of RFC 3261 [RFC3261]. Most existing UASs do not terminate the dialog on encountering this failure, which is fortunate.

Regardless of the presence or absence of proxy servers in the dialog's route set, a UAS generating a 2xx response to the re-INVITE will never receive an ACK request for it. According to Section 14.2 of RFC 3261 [RFC3261], such a UAS is supposed to (at the "should" level) terminate the dialog by sending a BYE request.

A UAS that accepts a target refresh within a re-INVITE and never receives an ACK request after having sent a final response to the re-INVITE SHOULD NOT terminate the dialog if the UA has received a new re-INVITE with a higher CSeq sequence number than the original one.

5.5. UAC Losing its Contact: UAC Behavior

When a UAC moves to a new contact and loses its old contact, it will not be able to receive responses to the re-INVITE. Consequently, it will never generate an ACK request.

Such a UAC SHOULD generate a CANCEL request to cancel the re-INVITE and cause the INVITE client transaction corresponding to the re-INVITE to enter the "Terminated" state. The UAC SHOULD also send a new re-INVITE in order to make sure that both UAs have a common view of the state of the session.

> Per Section 14.2 of RFC 3261 [RFC3261], the UAS will accept new incoming re-INVITEs as soon as it has generated a final response to the previous INVITE request, which had a lower CSeq sequence number.

6. Security Considerations

This document does not introduce any new security issue. It just clarifies how certain transactions should be handled in SIP. Security issues related to re-INVITEs and UPDATE requests are discussed in RFC 3261 [RFC3261] and RFC 3311 [RFC3311].

In particular, in order not to reduce the security level for a given session, re-INVITEs and UPDATE requests SHOULD be secured using a mechanism equivalent to or stronger than the initial INVITE request that created the session. For example, if the initial INVITE request

was end-to-end integrity protected or encrypted, subsequent re-INVITEs and UPDATE requests should also be so.

7. IANA Considerations

There are no IANA actions associated with this document.

8. Acknowledgements

Paul Kyzivat provided useful ideas on the topics discussed in this document.

9. References

9.1. Normative References

[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.

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

[RFC4032] Camarillo, G. and P. Kyzivat, "Update to the Session Initiation Protocol (SIP) Preconditions Framework", RFC 4032, March 2005.

9.2. Informative References

[RFC4320] Sparks, R., "Actions Addressing Identified Issues with the Session Initiation Protocol's (SIP) Non-INVITE Transaction", RFC 4320, January 2006.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

Authors' Addresses

Gonzalo Camarillo (editor)
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Gonzalo.Camarillo@ericsson.com

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Christer.Holmberg@ericsson.com

Yang Gao
ZTE
China

Email: gao.yang2@zte.com.cn

Network Working Group
Internet-Draft
Obsoletes: 3265 (if approved)
Updates: 3261, 4660
(if approved)
Intended status: Standards Track
Expires: November 1, 2012

A. B. Roach
Tekelec

April 30, 2012

SIP-Specific Event Notification
draft-ietf-sipcore-rfc3265bis-09

Abstract

This document describes an extension to the Session Initiation Protocol (SIP) defined by RFC 3261. The purpose of this extension is to provide an extensible framework by which SIP nodes can request notification from remote nodes indicating that certain events have occurred.

Note that the event notification mechanisms defined herein are NOT intended to be a general-purpose infrastructure for all classes of event subscription and notification.

This document represents a backwards-compatible improvement on the original mechanism described by RFC 3265, taking into account several years of implementation experience. Accordingly, this document obsoletes RFC 3265. This document also updates RFC 4660 slightly to accommodate some small changes to the mechanism that were discussed in that document.

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 November 1, 2012.

Table of Contents

1. Introduction

The ability to request asynchronous notification of events proves useful in many types of SIP services for which cooperation between end-nodes is required. Examples of such services include automatic callback services (based on terminal state events), buddy lists (based on user presence events), message waiting indications (based on mailbox state change events), and PSTN and Internet Internetworking (PINT) [RFC2848] status (based on call state events).

The methods described in this document provide a framework by which notification of these events can be ordered.

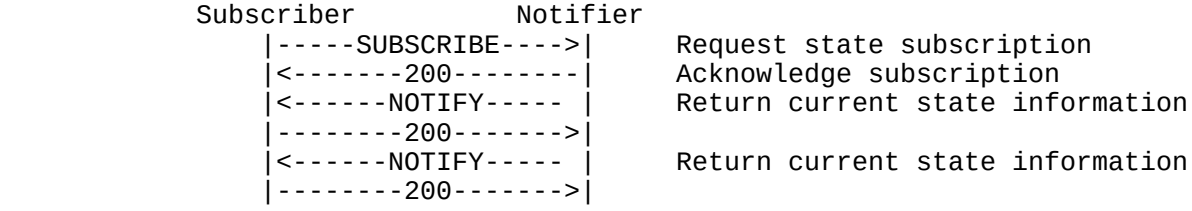
The event notification mechanisms defined herein are NOT intended to be a general-purpose infrastructure for all classes of event subscription and notification. Meeting requirements for the general problem set of subscription and notification is far too complex for a single protocol. Our goal is to provide a SIP-specific framework for event notification which is not so complex as to be unusable for simple features, but which is still flexible enough to provide powerful services. Note, however, that event packages based on this framework may define arbitrarily elaborate rules which govern the subscription and notification for the events or classes of events they describe.

This document does not describe an extension which may be used directly; it must be extended by other documents (herein referred to as "event packages"). In object-oriented design terminology, it may be thought of as an abstract base class which must be derived into an instantiatable class by further extensions. Guidelines for creating these extensions are described in Section 5.

1.1. Overview of Operation

The general concept is that entities in the network can subscribe to resource or call state for various resources or calls in the network, and those entities (or entities acting on their behalf) can send notifications when those states change.

A typical flow of messages would be:



Subscriptions are expired and must be refreshed by subsequent SUBSCRIBE requests.

1.2. Documentation Conventions

There are several paragraphs throughout this document which provide motivational or clarifying text. Such passages are non-normative, and are provided only to assist with reader comprehension. These passages are set off from the remainder of the text by being indented thus:

> This is an example of non-normative explanatory text. It does not form part of the specification, and is used only for clarification.

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 [RFC2119].

In particular, implementors need to take careful note of the meaning of "SHOULD" defined in RFC 2119. To rephrase: violation of SHOULD-strength requirements requires careful analysis and clearly enumerable reasons. It is a protocol violation to fail to comply with "SHOULD"-strength requirements whimsically or for ease of implementation.

The use of quotation marks next to periods and commas follows the convention used by the American Mathematical Society; although contrary to traditional American English convention, this usage lends clarity to certain passages.

2. Definitions

Event Package: An event package is an additional specification which defines a set of state information to be reported by a notifier to a subscriber. Event packages also define further syntax and semantics based on the framework defined by this document required to convey such state information.

Event Template-Package: An event template-package is a special kind of event package which defines a set of states which may be applied to all possible event packages, including itself.

Notification: Notification is the act of a notifier sending a NOTIFY request to a subscriber to inform the subscriber of the state of a resource.

Notifier: A notifier is a user agent which generates NOTIFY requests for the purpose of notifying subscribers of the state of a resource. Notifiers typically also accept SUBSCRIBE requests to create subscriptions.

Subscriber: A subscriber is a user agent which receives NOTIFY requests from notifiers; these NOTIFY requests contain information about the state of a resource in which the subscriber is interested. Subscribers typically also generate SUBSCRIBE requests and send them to notifiers to create subscriptions.

Subscription: A subscription is a set of application state associated with a dialog. This application state includes a pointer to the associated dialog, the event package name, and possibly an identification token. Event packages will define additional subscription state information. By definition, subscriptions exist in both a subscriber and a notifier.

Subscription Migration: Subscription migration is the act of moving a subscription from one notifier to another notifier.

3. SIP Methods for Event Notification

3.1. SUBSCRIBE

The SUBSCRIBE method is used to request current state and state updates from a remote node. SUBSCRIBE requests are target refresh requests, as that term is defined in [RFC3261].

3.1.1. Subscription Duration

SUBSCRIBE requests SHOULD contain an "Expires" header field (defined in [RFC3261]). This expires value indicates the duration of the subscription. In order to keep subscriptions effective beyond the duration communicated in the "Expires" header field, subscribers need to refresh subscriptions on a periodic basis using a new SUBSCRIBE request on the same dialog as defined in [RFC3261].

If no "Expires" header field is present in a SUBSCRIBE request, the implied default MUST be defined by the event package being used.

200-class responses to SUBSCRIBE requests also MUST contain an "Expires" header field. The period of time in the response MAY be

shorter but MUST NOT be longer than specified in the request. The notifier is explicitly allowed to shorten the duration to zero. The period of time in the response is the one which defines the duration of the subscription.

An "expires" parameter on the "Contact" header field has no semantics for the SUBSCRIBE method and is explicitly not equivalent to an "Expires" header field in a SUBSCRIBE request or response.

A natural consequence of this scheme is that a SUBSCRIBE request with an "Expires" of 0 constitutes a request to unsubscribe from the matching subscription.

> In addition to being a request to unsubscribe, a SUBSCRIBE request with "Expires" of 0 also causes a fetch of state; see Section 4.4.3.

Notifiers may also wish to cancel subscriptions to events; this is useful, for example, when the resource to which a subscription refers is no longer available. Further details on this mechanism are discussed in Section 4.2.2.

3.1.2. Identification of Subscribed Events and Event Classes

Identification of events is provided by three pieces of information: Request URI, Event Type, and (optionally) message body.

The Request URI of a SUBSCRIBE request, most importantly, contains enough information to route the request to the appropriate entity per the request routing procedures outlined in [RFC3261]. It also contains enough information to identify the resource for which event notification is desired, but not necessarily enough information to uniquely identify the nature of the event (e.g., "sip:adam@example.com" would be an appropriate URI to subscribe to for my presence state; it would also be an appropriate URI to subscribe to the state of my voice mailbox).

Subscribers MUST include exactly one "Event" header field in SUBSCRIBE requests, indicating to which event or class of events they are subscribing. The "Event" header field will contain a token which indicates the type of state for which a subscription is being requested. This token will be registered with the IANA and will correspond to an event package which further describes the semantics of the event or event class.

If the event package to which the event token corresponds defines behavior associated with the body of its SUBSCRIBE requests, those semantics apply.

Event packages may also define parameters for the Event header field; if they do so, they must define the semantics for such parameters.

3.1.3. Additional SUBSCRIBE Header Field Values

Because SUBSCRIBE requests create a dialog usage as defined in [RFC3261], they MAY contain an "Accept" header field. This header field, if present, indicates the body formats allowed in subsequent NOTIFY requests. Event packages MUST define the behavior for SUBSCRIBE requests without "Accept" header fields; usually, this will connote a single, default body type.

Header values not described in this document are to be interpreted as described in [RFC3261].

3.2. NOTIFY

NOTIFY requests are sent to inform subscribers of changes in state to which the subscriber has a subscription. Subscriptions are created using the SUBSCRIBE method. In legacy implementations, it is possible that other means of subscription creation have been used. However, this specification does not allow the creation of subscriptions except through SUBSCRIBE requests and (for backwards-compatibility) REFER requests [RFC3515].

NOTIFY is a target refresh request, as that term is defined in [RFC3261].

A NOTIFY request does not terminate its corresponding subscription; in other words, a single SUBSCRIBE request may trigger several NOTIFY requests.

3.2.1. Identification of Reported Events, Event Classes, and Current State

Identification of events being reported in a notification is very similar to that described for subscription to events (see Section 3.1.2).

As in SUBSCRIBE requests, NOTIFY request "Event" header fields MUST contain a single event package name for which a notification is being generated. The package name in the "Event" header field MUST match the "Event" header field in the corresponding SUBSCRIBE request.

Event packages may define semantics associated with the body of their NOTIFY requests; if they do so, those semantics apply. NOTIFY request bodies are expected to provide additional details about the nature of the event which has occurred and the resultant resource

state.

When present, the body of the NOTIFY request MUST be formatted into one of the body formats specified in the "Accept" header field of the corresponding SUBSCRIBE request (or the default type according to the event package description, if no Accept header field was specified). This body will contain either the state of the subscribed resource or a pointer to such state in the form of a URI (see Section 5.4.13).

4. Node Behavior

4.1. Subscriber Behavior

4.1.1. Detecting Support for SIP Events

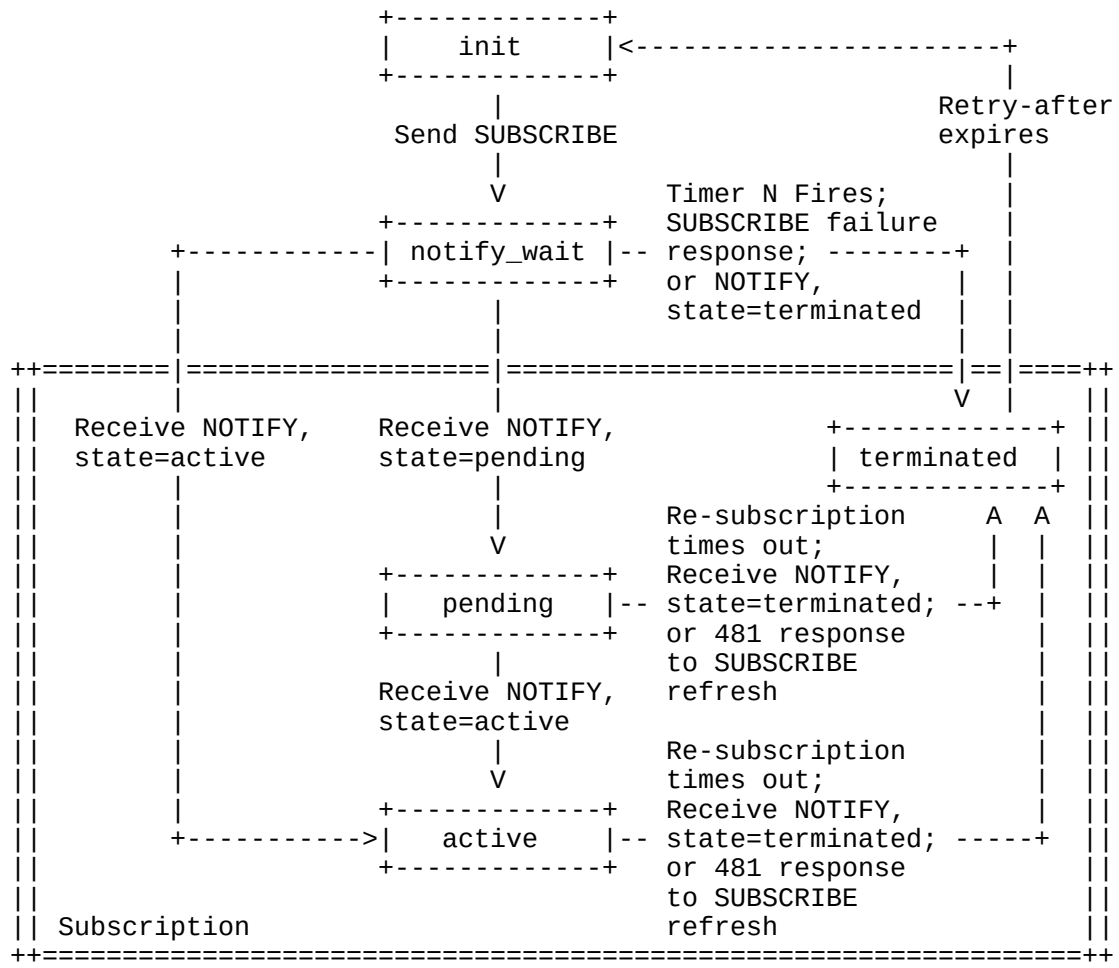
The extension described in this document does not make use of the "Require" or "Proxy-Require" header fields; similarly, there is no token defined for "Supported" header fields. Potential subscribers may probe for the support of SIP Events using the OPTIONS request defined in [RFC3261].

The presence of "SUBSCRIBE" in the "Allow" header field of any request or response indicates support for SIP Events; further, in the absence of an "Allow" header field, the simple presence of an "Allow-Events" header field is sufficient to indicate that the node that sent the message is capable of acting as a notifier (see Section 4.4.4).

> The "methods" parameter for Contact may also be used to specifically announce support for SUBSCRIBE and NOTIFY requests when registering. (See [RFC3840] for details on the "methods" parameter).

4.1.2. Creating and Maintaining Subscriptions

From the subscriber’s perspective, a subscription proceeds according to the following state diagram. Events which result in a transition back to the same state are not represented in this diagram.



In the state diagram, "Re-subscription times out" means that an attempt to refresh or update the subscription using a new SUBSCRIBE request does not result in a NOTIFY request before the corresponding Timer N expires.

Any transition from "notify_wait" into a "pending" or "active" state results in a new subscription. Note that multiple subscriptions can be generated as the result of a single SUBSCRIBE request (see Section 4.4.1). Each of these new subscriptions exists in its own independent state machine, and runs its own set of timers.

4.1.2.1. Requesting a Subscription

SUBSCRIBE is a dialog-creating method, as described in [RFC3261].

When a subscriber wishes to subscribe to a particular state for a resource, it forms a SUBSCRIBE request. If the initial SUBSCRIBE request represents a request outside of a dialog (as it typically will), its construction follows the procedures outlined in [RFC3261] for UAC request generation outside of a dialog.

This SUBSCRIBE request will be confirmed with a final response. 200-class responses indicate that the subscription has been accepted, and that a NOTIFY request will be sent immediately.

The "Expires" header field in a 200-class response to SUBSCRIBE request indicates the actual duration for which the subscription will remain active (unless refreshed). The received value might be smaller than the value indicated in the SUBSCRIBE request, but cannot be larger; see Section 4.2.1 for details.

Non-200 class final responses indicate that no subscription or new dialog usage has been created, and no subsequent NOTIFY request will be sent. All non-200 class responses (with the exception of "489", described herein) have the same meanings and handling as described in [RFC3261]. For the sake of clarity: if a SUBSCRIBE request contains an "Accept" header field, but that field does not indicate a media type that the notifier is capable of generating in its NOTIFY requests, then the proper error response is 406 (Not Acceptable).

4.1.2.2. Refreshing of Subscriptions

At any time before a subscription expires, the subscriber may refresh the timer on such a subscription by sending another SUBSCRIBE request on the same dialog as the existing subscription. The handling for such a request is the same as for the initial creation of a subscription except as described below.

If a SUBSCRIBE request to refresh a subscription receives a 404, 405, 410, 416, 480-485, 489, 501, or 604 response, the subscriber MUST consider the subscription terminated. (See [RFC5057] for further details and notes about the effect of error codes on dialogs and usages within dialog, such as subscriptions). If the subscriber wishes to re-subscribe to the state, he does so by composing an unrelated initial SUBSCRIBE request with a freshly-generated Call-ID and a new, unique "From" tag (see Section 4.1.2.1.)

If a SUBSCRIBE request to refresh a subscription fails with any error code other than those listed above, the original subscription is

still considered valid for the duration of the most recently known "Expires" value as negotiated by the most recent successful SUBSCRIBE transaction, or as communicated by a NOTIFY request in its "Subscription-State" header field "expires" parameter.

> Note that many such errors indicate that there may be a problem with the network or the notifier such that no further NOTIFY requests will be received.

When refreshing a subscription, a subscriber starts Timer N, set to 64*T1, when it sends the SUBSCRIBE request. If this Timer N expires prior to the receipt of a NOTIFY request, the subscriber considers the subscription terminated. If the subscriber receives a success response to the SUBSCRIBE request that indicates that no NOTIFY request will be generated -- such as the 204 response defined for use with the optional extension described in [RFC5839] -- then it MUST cancel Timer N.

4.1.2.3. Unsubscribing

Unsubscribing is handled in the same way as refreshing of a subscription, with the "Expires" header field set to "0". Note that a successful unsubscription will also trigger a final NOTIFY request.

The final NOTIFY request may or may not contain information about the state of the resource; subscribers need to be prepared to receive final NOTIFY requests both with and without state.

4.1.2.4. Confirmation of Subscription Creation

The subscriber can expect to receive a NOTIFY request from each node which has processed a successful subscription or subscription refresh. To ensure that subscribers do not wait indefinitely for a subscription to be established, a subscriber starts a Timer N, set to 64*T1, when it sends a SUBSCRIBE request. If this Timer N expires prior to the receipt of a NOTIFY request, the subscriber considers the subscription failed, and cleans up any state associated with the subscription attempt.

Until Timer N expires, several NOTIFY requests may arrive from different destinations (see Section 4.4.1). Each of these requests establish a new dialog usage and a new subscription. After the expiration of Timer N, the subscriber SHOULD reject any such NOTIFY requests that would otherwise establish a new dialog usage with a "481" response code.

Until the first NOTIFY request arrives, the subscriber should consider the state of the subscribed resource to be in a neutral

state. Event package specifications MUST define this "neutral state" in such a way that makes sense for their application (see Section 5.4.7).

Due to the potential for out-of-order messages, packet loss, and forking, the subscriber MUST be prepared to receive NOTIFY requests before the SUBSCRIBE transaction has completed.

Except as noted above, processing of this NOTIFY request is the same as in Section 4.1.3.

4.1.3. Receiving and Processing State Information

Subscribers receive information about the state of a resource to which they have subscribed in the form of NOTIFY requests.

Upon receiving a NOTIFY request, the subscriber should check that it matches at least one of its outstanding subscriptions; if not, it MUST return a "481 Subscription does not exist" response unless another 400- or 500-class response is more appropriate. The rules for matching NOTIFY requests with subscriptions that create a new dialog usage are described in Section 4.4.1. Notifications for subscriptions which were created inside an existing dialog match if they are in the same dialog and the "Event" header fields match (as described in Section 8.2.1).

If, for some reason, the event package designated in the "Event" header field of the NOTIFY request is not supported, the subscriber will respond with a "489 Bad Event" response.

To prevent spoofing of events, NOTIFY requests SHOULD be authenticated, using any defined SIP authentication mechanism, such as those described in sections 22.2 and 23 of [RFC3261].

NOTIFY requests MUST contain "Subscription-State" header fields which indicate the status of the subscription.

If the "Subscription-State" header field value is "active", it means that the subscription has been accepted and (in general) has been authorized. If the header field also contains an "expires" parameter, the subscriber SHOULD take it as the authoritative subscription duration and adjust accordingly. The "retry-after" and "reason" parameters have no semantics for "active".

If the "Subscription-State" value is "pending", the subscription has been received by the notifier, but there is insufficient policy information to grant or deny the subscription yet. If the header field also contains an "expires" parameter, the subscriber SHOULD

take it as the authoritative subscription duration and adjust accordingly. No further action is necessary on the part of the subscriber. The "retry-after" and "reason" parameters have no semantics for "pending".

If the "Subscription-State" value is "terminated", the subscriber MUST consider the subscription terminated. The "expires" parameter has no semantics for "terminated" -- notifiers SHOULD NOT include an "expires" parameter on a "Subscription-State" header field with a value of "terminated," and subscribers MUST ignore any such parameter, if present. If a reason code is present, the client should behave as described below. If no reason code or an unknown reason code is present, the client MAY attempt to re-subscribe at any time (unless a "retry-after" parameter is present, in which case the client SHOULD NOT attempt re-subscription until after the number of seconds specified by the "retry-after" parameter). The reason codes defined by this document are:

deactivated: The subscription has been terminated, but the subscriber SHOULD retry immediately with a new subscription. One primary use of such a status code is to allow migration of subscriptions between nodes. The "retry-after" parameter has no semantics for "deactivated".

probation: The subscription has been terminated, but the client SHOULD retry at some later time (as long as the resource's state is still relevant to the client at that time). If a "retry-after" parameter is also present, the client SHOULD wait at least the number of seconds specified by that parameter before attempting to re-subscribe.

rejected: The subscription has been terminated due to change in authorization policy. Clients SHOULD NOT attempt to re-subscribe. The "retry-after" parameter has no semantics for "rejected".

timeout: The subscription has been terminated because it was not refreshed before it expired. Clients MAY re-subscribe immediately. The "retry-after" parameter has no semantics for "timeout". This reason code is also associated with polling of resource state, as detailed in Section 4.4.3

giveup: The subscription has been terminated because the notifier could not obtain authorization in a timely fashion. If a "retry-after" parameter is also present, the client SHOULD wait at least the number of seconds specified by that parameter before attempting to re-subscribe; otherwise, the client MAY retry immediately, but will likely get put back into pending state.

noresource: The subscription has been terminated because the resource state which was being monitored no longer exists. Clients SHOULD NOT attempt to re-subscribe. The "retry-after" parameter has no semantics for "noresource".

invariant: The subscription has been terminated because the resource state is guaranteed not to change for the foreseeable future. This may be the case, for example, when subscribing to the location information of a fixed-location land-line telephone. When using this reason code, notifiers are advised to include a "retry-after" parameter with a large value (for example, 31536000 -- or one year) to prevent older, RFC 3265-compliant clients from periodically resubscribing. Clients SHOULD NOT attempt to resubscribe after receiving a reason code of "invariant," regardless of the presence of or value of a "retry-after" parameter.

Other specifications may define new reason codes for use with the "Subscription-State" header field.

Once the notification is deemed acceptable to the subscriber, the subscriber SHOULD return a 200 response. In general, it is not expected that NOTIFY responses will contain bodies; however, they MAY, if the NOTIFY request contained an "Accept" header field.

Other responses defined in [RFC3261] may also be returned, as appropriate. In no case should a NOTIFY transaction extend for any longer than the time necessary for automated processing. In particular, subscribers MUST NOT wait for a user response before returning a final response to a NOTIFY request.

4.1.4. Forking of SUBSCRIBE Requests

In accordance with the rules for proxying non-INVITE requests as defined in [RFC3261], successful SUBSCRIBE requests will receive only one 200-class response; however, due to forking, the subscription may have been accepted by multiple nodes. The subscriber MUST therefore be prepared to receive NOTIFY requests with "From:" tags which differ from the "To:" tag received in the SUBSCRIBE 200-class response.

If multiple NOTIFY requests are received in different dialogs in response to a single SUBSCRIBE request, each dialog represents a different destination to which the SUBSCRIBE request was forked. Subscriber handling in such situations varies by event package; see Section 5.4.9 for details.

4.2. Notifier Behavior

4.2.1. Subscription Establishment and Maintenance

Notifiers learn about subscription requests by receiving SUBSCRIBE requests from interested parties. Notifiers MUST NOT create subscriptions except upon receipt of a SUBSCRIBE request. However, for historical reasons, the implicit creation of subscriptions as defined in [RFC3515] is still permitted.

> [RFC3265] allowed the creation of subscriptions using means other than the SUBSCRIBE method. The only standardized use of this mechanism is the REFER method [RFC3515]. Implementation experience with REFER has shown that the implicit creation of a subscription has a number of undesirable effects, such as the inability to signal the success of a REFER request while signaling a problem with the subscription; and difficulty performing one action without the other. Additionally, the proper exchange of dialog identifiers is difficult without dialog re-use (which has its own set of problems; see Section 4.5).

4.2.1.1. Initial SUBSCRIBE Transaction Processing

In no case should a SUBSCRIBE transaction extend for any longer than the time necessary for automated processing. In particular, notifiers MUST NOT wait for a user response before returning a final response to a SUBSCRIBE request.

> This requirement is imposed primarily to prevent the non-INVITE transaction timeout timer F (see [RFC3261]) from firing during the SUBSCRIBE transaction, since interaction with a user would often exceed 64*T1 seconds.

The notifier SHOULD check that the event package specified in the "Event" header field is understood. If not, the notifier SHOULD return a "489 Bad Event" response to indicate that the specified event/event class is not understood.

The notifier SHOULD also perform any necessary authentication and authorization per its local policy. See Section 4.2.1.3.

The notifier MAY also check that the duration in the "Expires" header field is not too small. If and only if the expiration interval is greater than zero AND smaller than one hour AND less than a notifier-configured minimum, the notifier MAY return a "423 Interval Too Brief" error which contains a "Min-Expires" header field field. The "Min-Expires" header field is described in [RFC3261].

Once the notifier determines that it has enough information to create the subscription (i.e., it understands the event package, the subscription pertains to a known resource, and there are no other barriers to creating the subscription), it creates the subscription and a dialog usage, and returns a 200 (OK) response.

When a subscription is created in the notifier, it stores the event package name as part of the subscription information.

The "Expires" values present in SUBSCRIBE 200-class responses behave in the same way as they do in REGISTER responses: the server MAY shorten the interval, but MUST NOT lengthen it.

> If the duration specified in a SUBSCRIBE request is unacceptably short, the notifier may be able to send a 423 response, as described earlier in this section.

200-class responses to SUBSCRIBE requests will not generally contain any useful information beyond subscription duration; their primary purpose is to serve as a reliability mechanism. State information will be communicated via a subsequent NOTIFY request from the notifier.

The other response codes defined in [RFC3261] may be used in response to SUBSCRIBE requests, as appropriate.

4.2.1.2. Confirmation of Subscription Creation/Refreshing

Upon successfully accepting or refreshing a subscription, notifiers MUST send a NOTIFY request immediately to communicate the current resource state to the subscriber. This NOTIFY request is sent on the same dialog as created by the SUBSCRIBE response. If the resource has no meaningful state at the time that the SUBSCRIBE request is processed, this NOTIFY request MAY contain an empty or neutral body. See Section 4.2.2 for further details on NOTIFY request generation.

Note that a NOTIFY request is always sent immediately after any 200-class response to a SUBSCRIBE request, regardless of whether the subscription has already been authorized.

4.2.1.3. Authentication/Authorization of SUBSCRIBE Requests

Privacy concerns may require that notifiers apply policy to determine whether a particular subscriber is authorized to subscribe to a certain set of events. Such policy may be defined by mechanisms such as access control lists or real-time interaction with a user. In general, authorization of subscribers prior to authentication is not particularly useful.

SIP authentication mechanisms are discussed in [RFC3261]. Note that, even if the notifier node typically acts as a proxy, authentication for SUBSCRIBE requests will always be performed via a "401" response, not a "407". Notifiers always act as a user agents when accepting subscriptions and sending notifications.

> Of course, when acting as a proxy, a node will perform normal proxy authentication (using 407). The foregoing explanation is a reminder that notifiers are always UAs, and as such perform UA authentication.

If authorization fails based on an access list or some other automated mechanism (i.e., it can be automatically authoritatively determined that the subscriber is not authorized to subscribe), the notifier SHOULD reply to the request with a "403 Forbidden" or "603 Decline" response, unless doing so might reveal information that should stay private; see Section 6.2.

If the notifier owner is interactively queried to determine whether a subscription is allowed, a 200 (OK) response is returned immediately. Note that a NOTIFY request is still formed and sent under these circumstances, as described in the previous section.

If subscription authorization was delayed and the notifier wishes to convey that such authorization has been declined, it may do so by sending a NOTIFY request containing a "Subscription-State" header field with a value of "terminated" and a reason parameter of "rejected".

4.2.1.4. Refreshing of Subscriptions

When a notifier receives a subscription refresh, assuming that the subscriber is still authorized, the notifier updates the expiration time for subscription. As with the initial subscription, the server MAY shorten the amount of time until expiration, but MUST NOT increase it. The final expiration time is placed in the "Expires" header field in the response. If the duration specified in a SUBSCRIBE request is unacceptably short, the notifier SHOULD respond with a "423 Interval Too Brief" response.

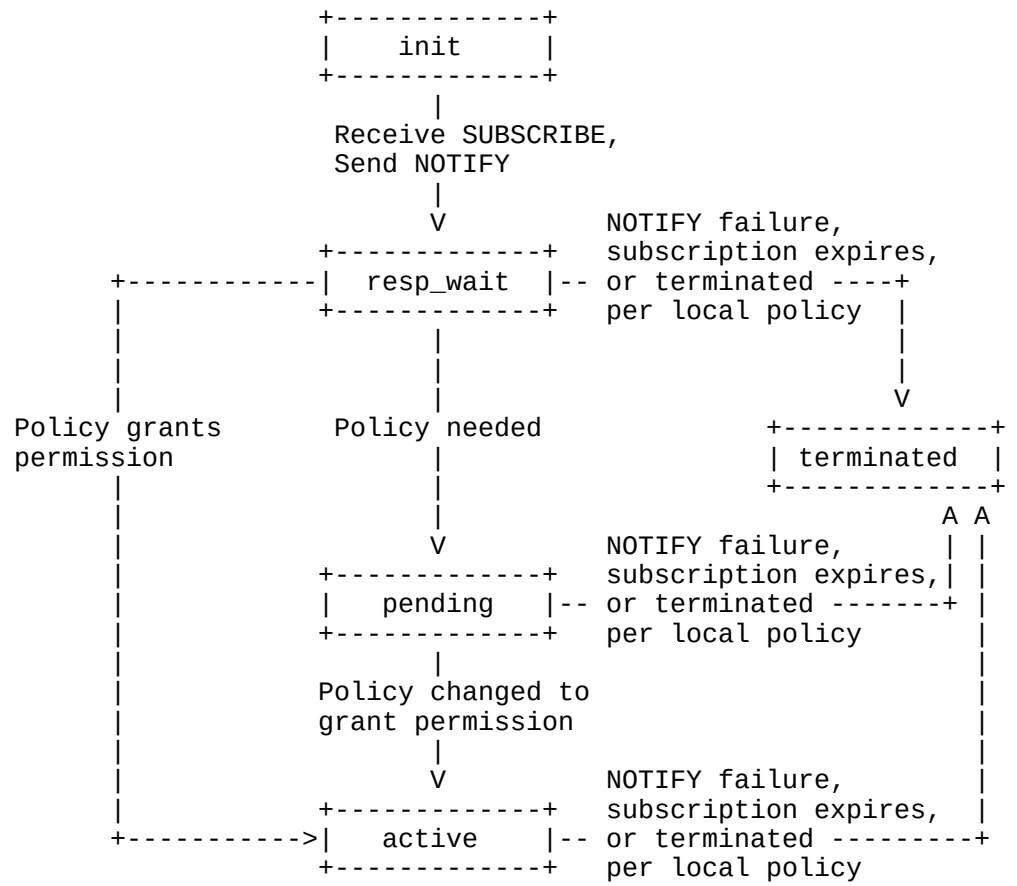
If no refresh for a notification address is received before its expiration time, the subscription is removed. When removing a subscription, the notifier SHOULD send a NOTIFY request with a "Subscription-State" value of "terminated" to inform it that the subscription is being removed. If such a request is sent, the "Subscription-State" header field SHOULD contain a "reason=timeout" parameter.

Clients can cause a subscription to be terminated immediately by sending a SUBSCRIBE request with an "Expires" header field set to '0'. Notifiers largely treat this the same way as any other subscription expiration: they send a NOTIFY request containing a "Subscription-State" of "terminated", with a reason code of "timeout." For consistency with state polling (see Section 4.4.3) and subscription refreshes, the notifier may choose to include resource state in this final NOTIFY request. However, in some cases, including such state makes no sense. Under such circumstances, the notifier may choose to omit state information from the terminal NOTIFY request.

The sending of a NOTIFY request when a subscription expires allows the corresponding dialog usage to be terminated, if appropriate.

4.2.2. Sending State Information to Subscribers

Notifiers use the NOTIFY method to send information about the state of a resource to subscribers. The notifier's view of a subscription is shown in the following state diagram. Events which result in a transition back to the same state are not represented in this diagram.



When a SUBSCRIBE request is answered with a 200-class response, the notifier MUST immediately construct and send a NOTIFY request to the subscriber. When a change in the subscribed state occurs, the notifier SHOULD immediately construct and send a NOTIFY request, unless the state transition is caused by a NOTIFY transaction failure. The sending of this NOTIFY message is also subject to authorization, local policy, and throttling considerations.

If the NOTIFY request fails due to expiration of SIP Timer F (transaction timeout), the notifier SHOULD remove the subscription.

This behavior prevents unnecessary transmission of state information for subscribers who have crashed or disappeared from the network. Because such transmissions will be sent multiple times, per the retransmission algorithm defined in [RFC3261] (instead of the typical single transmission for functioning clients), continuing to service them when no client is available

to acknowledge them could place undue strain on a network. Upon client restart or reestablishment of a network connection, it is expected that clients will send SUBSCRIBE requests to refresh potentially stale state information; such requests will re-install subscriptions in all relevant nodes.

If the NOTIFY transaction fails due to the receipt of a 404, 405, 410, 416, 480-485, 489, 501, or 604 response to the NOTIFY request, the notifier MUST remove the corresponding subscription. See [RFC5057] for further details and notes about the effect of error codes on dialogs and usages within dialog (such as subscriptions).

A notify error response would generally indicate that something has gone wrong with the subscriber or with some proxy on the way to the subscriber. If the subscriber is in error, it makes the most sense to allow the subscriber to rectify the situation (by re-subscribing) once the error condition has been handled. If a proxy is in error, the periodic sending of SUBSCRIBE requests to refresh the expiration timer will re-install subscription state once the network problem has been resolved.

NOTIFY requests MUST contain a "Subscription-State" header field with a value of "active", "pending", or "terminated". The "active" value indicates that the subscription has been accepted and has been authorized (in most cases; see Section 6.2). The "pending" value indicates that the subscription has been received, but that policy information is insufficient to accept or deny the subscription at this time. The "terminated" value indicates that the subscription is not active.

If the value of the "Subscription-State" header field is "active" or "pending", the notifier MUST also include in the "Subscription-State" header field an "expires" parameter which indicates the time remaining on the subscription. The notifier MAY use this mechanism to shorten a subscription; however, this mechanism MUST NOT be used to lengthen a subscription.

Including expiration information for active and pending subscriptions is necessary in case the SUBSCRIBE request forks, since the response to a forked SUBSCRIBE request may not be received by the subscriber. [RFC3265] allowed the notifier some discretion in the inclusion of this parameter, so subscriber implementations are warned to handle the lack of an "expires" parameter gracefully. Note well that this "expires" value is a parameter on the "Subscription-State" header field, NOT an "Expires" header field.

> The period of time for a subscription can be shortened to zero by the notifier. In other words, it is perfectly valid for a SUBSCRIBE request with a non-zero expires to be answered with a NOTIFY request that contains "Subscription-Status: terminated;reason=expired". This merely means that the notifier has shortened the subscription timeout to zero, and the subscription has expired instantaneously. The body may contain valid state, or it may contain a neutral state (see Section 5.4.7).

If the value of the "Subscription-State" header field is "terminated", the notifier SHOULD also include a "reason" parameter. The notifier MAY also include a "retry-after" parameter, where appropriate. For details on the value and semantics of the "reason" and "retry-after" parameters, see Section 4.1.3.

4.2.3. PINT Compatibility

The "Event" header field is considered mandatory for the purposes of this document. However, to maintain compatibility with PINT (see [RFC2848]), notifiers MAY interpret a SUBSCRIBE request with no "Event" header field as requesting a subscription to PINT events. If a notifier does not support PINT, it SHOULD return "489 Bad Event" to any SUBSCRIBE requests without an "Event" header field.

4.3. Proxy Behavior

Proxies need no additional behavior beyond that described in [RFC3261] to support SUBSCRIBE and NOTIFY transactions. If a proxy wishes to see all of the SUBSCRIBE and NOTIFY requests for a given dialog, it MUST add a Record-Route header field to the initial SUBSCRIBE request and all NOTIFY requests. It MAY choose to include Record-Route in subsequent SUBSCRIBE requests; however, these requests cannot cause the dialog’s route set to be modified.

Proxies that did not add a Record-Route header field to the initial SUBSCRIBE request MUST NOT add a Record-Route header field to any of the associated NOTIFY requests.

> Note that subscribers and notifiers may elect to use S/MIME encryption of SUBSCRIBE and NOTIFY requests; consequently, proxies cannot rely on being able to access any information that is not explicitly required to be proxy-readable by [RFC3261].

4.4. Common Behavior

4.4.1. Dialog Creation and Termination

Dialogs usages are created upon completion of a NOTIFY transaction for a new subscription, unless the NOTIFY request contains a "Subscription-State" of "terminated."

Because the dialog usage is established by the NOTIFY request, the route set at the subscriber is taken from the NOTIFY request itself, as opposed to the route set present in the 200-class response to the SUBSCRIBE request.

NOTIFY requests are matched to such SUBSCRIBE requests if they contain the same "Call-ID", a "To" header field "tag" parameter which matches the "From" header field "tag" parameter of the SUBSCRIBE request, and the same "Event" header field. Rules for comparisons of the "Event" header fields are described in Section 8.2.1.

A subscription is destroyed after a notifier sends a NOTIFY request with a "Subscription-State" of "terminated," or in certain error situations described elsewhere in this document. The subscriber will generally answer such final requests with a "200 OK" response (unless a condition warranting an alternate response has arisen). Except when the mechanism described in Section 4.5.2 is used, the destruction of a subscription results in the termination of its associated dialog.

> A subscriber may send a SUBSCRIBE request with an "Expires" header field of 0 in order to trigger the sending of such a NOTIFY request; however, for the purposes of subscription and dialog lifetime, the subscription is not considered terminated until the NOTIFY transaction with a "Subscription-State" of "terminated" completes.

4.4.2. Notifier Migration

It is often useful to allow migration of subscriptions between notifiers. Such migration may be effected by sending a NOTIFY request with a "Subscription-State" header field of "terminated", and a reason parameter of "deactivated". This NOTIFY request is otherwise normal, and is formed as described in Section 4.2.2.

Upon receipt of this NOTIFY request, the subscriber SHOULD attempt to re-subscribe (as described in the preceding sections). Note that this subscription is established on a new dialog, and does not re-use the route set from the previous subscription dialog.

The actual migration is effected by making a change to the policy (such as routing decisions) of one or more servers to which the

SUBSCRIBE request will be sent in such a way that a different node ends up responding to the SUBSCRIBE request. This may be as simple as a change in the local policy in the notifier from which the subscription is migrating so that it serves as a proxy or redirect server instead of a notifier.

Whether, when, and why to perform notifier migrations may be described in individual event packages; otherwise, such decisions are a matter of local notifier policy, and are left up to individual implementations.

4.4.3. Polling Resource State

A natural consequence of the behavior described in the preceding sections is that an immediate fetch without a persistent subscription may be effected by sending a SUBSCRIBE with an "Expires" of 0.

Of course, an immediate fetch while a subscription is active may be effected by sending a SUBSCRIBE request with an "Expires" equal to the number of seconds remaining in the subscription.

Upon receipt of this SUBSCRIBE request, the notifier (or notifiers, if the SUBSCRIBE request was forked) will send a NOTIFY request containing resource state in the same dialog.

Note that the NOTIFY requests triggered by SUBSCRIBE requests with "Expires" header fields of 0 will contain a "Subscription-State" value of "terminated", and a "reason" parameter of "timeout".

Polling of event state can cause significant increases in load on the network and notifiers; as such, it should be used only sparingly. In particular, polling SHOULD NOT be used in circumstances in which it will typically result in more network messages than long-running subscriptions.

When polling is used, subscribers SHOULD attempt to cache authentication credentials between polls so as to reduce the number of messages sent.

> Due to the requirement on notifiers to send a NOTIFY request immediately upon receipt of a SUBSCRIBE request, the state provided by polling is limited to the information that the notifier has immediate local access to when it receives the SUBSCRIBE request. If, for example, the notifier generally needs to retrieve state from another network server, then that state will be absent from the NOTIFY request that results from polling.

4.4.4. Allow-Events header field usage

The "Allow-Events" header field, if present, MUST include a comprehensive and inclusive list of tokens which indicates the event packages for which the User Agent can act as a notifier. In other words, a user agent sending an "Allow-Events" header field is advertising that it can process SUBSCRIBE requests and generate NOTIFY requests for all of the event packages listed in that header field.

Any user agent that can act as a notifier for one or more event packages SHOULD include an appropriate "Allow-Events" header field indicating all supported events in all methods which initiate dialogs and their responses (such as INVITE) and OPTIONS responses.

> This information is very useful, for example, in allowing user agents to render particular interface elements appropriately according to whether the events required to implement the features they represent are supported by the appropriate nodes.
> On the other hand, it doesn’t necessarily make much sense to indicate supported events inside a dialog established by a NOTIFY request if the only event package supported is the one associated with that subscription.

Note that "Allow-Events" header fields MUST NOT be inserted by proxies.

The "Allow-Events" header field does not include a list of the event template packages supported by an implementation. If a subscriber wishes to determine which event template packages are supported by a notifier, it can probe for such support by attempting to subscribe to the event template packages it wishes to use.

> For example: to check for support for the templatized package "presence.winfo", a client may attempt to subscribe to that event package for a known resource, using an "Expires" header value of 0. if the response is a 489 error code, then the client can deduce that "presence.winfo" is unsupported.

4.5. Targeting Subscriptions at Devices

[RFC3265] defined a mechanism by which subscriptions could share dialogs with invite usages and with other subscriptions. The purpose of this behavior was to allow subscribers to ensure that a subscription arrived at the same device as an established dialog. Unfortunately, the re-use of dialogs has proven to be exceedingly confusing. [RFC5057] attempted to clarify proper behavior in a variety of circumstances; however, the ensuing rules remain confusing

and prone to implementation error. At the same time, the mechanism described in [RFC5627] now provides a far more elegant and unambiguous means to achieve the same goal.

Consequently, the dialog re-use technique described in RFC 3265 is now deprecated.

This dialog-sharing technique has also historically been used as a means for targeting an event package at a dialog. This usage can be seen, for example, in certain applications of the REFER method [RFC3515]. With the removal of dialog re-use, an alternate (and more explicit) means of targeting dialogs needs to be used for this type of correlation. The appropriate means of such targeting is left up to the actual event packages. Candidates include the "Target-Dialog" header field [RFC4538], the "Join" header field [RFC3911], and the "Replaces" header field [RFC3891], depending on the semantics desired. Alternately, if the semantics of those header fields do not match the event package's purpose for correlation, event packages can devise their own means of identifying dialogs. For an example of this approach, see the Dialog Event Package [RFC4235].

4.5.1. Using GRUUs to Route to Devices

Notifiers MUST implement the Globally Routable User-Agent URI (GRUU) extension defined in [RFC5627], and MUST use a GRUU as their local target. This allows subscribers to explicitly target desired devices.

If a subscriber wishes to subscribe to a resource on the same device as an established dialog, it should check whether the remote contact in that dialog is a GRUU (i.e., whether it contains a "gr" URI parameter). If so, the subscriber creates a new dialog, using the GRUU as the request URI for the new SUBSCRIBE request.

> Because GRUUs are guaranteed to route to a a specific device, this ensures that the subscription will be routed to the same place as the established dialog.

4.5.2. Sharing Dialogs

For compatibility with older clients, subscriber and notifier implementations may choose to allow dialog sharing. The behavior of multiple usages within a dialog are described in [RFC5057].

Subscribers MUST NOT attempt to re-use dialogs whose remote target is a GRUU.

Note that the techniques described in this section are included for backwards compatibility purposes only. Because subscribers cannot re-use dialogs with a GRUU for their remote target, and because notifiers must use GRUUs as their local target, any two implementations that conform to this specification will automatically use the mechanism described in Section 4.5.1.

Further note that the prohibition on re-using dialogs does not exempt implicit subscriptions created by the REFER method. This means that implementations complying with this specification are required to use the "Target-Dialog" mechanism described in [RFC4538] when the remote target is a GRUU.

If a subscriber wishes to subscribe to a resource on the same device as an established dialog and the remote contact is not a GRUU, it MAY revert to dialog sharing behavior. Alternately, it MAY choose to treat the remote party as incapable of servicing the subscription (i.e., the same way it would behave if the remote party did not support SIP events at all).

If a notifier receives a SUBSCRIBE request for a new subscription on an existing dialog, it MAY choose to implement dialog sharing behavior. Alternately, it may choose to fail the SUBSCRIBE request with a 403 response. The error text of such 403 responses SHOULD indicate that dialog sharing is not supported.

To implement dialog sharing, subscribers and notifiers perform the following additional processing:

- When subscriptions exist in dialogs associated with INVITE-created application state and/or other subscriptions, these sets of application state do not interact beyond the behavior described for a dialog (e.g., route set handling). In particular, multiple subscriptions within a dialog are expire independently, and require independent subscription refreshes.

- If a subscription’s destruction leaves no other application state associated with the dialog, the dialog terminates. The destruction of other application state (such as that created by an INVITE) will not terminate the dialog if a subscription is still associated with that dialog. This means that, when dialog are re-used, then a dialog created with an INVITE does not necessarily terminate upon receipt of a BYE. Similarly, in the case that several subscriptions are associated with a single dialog, the dialog does not terminate until all the subscriptions in it are destroyed.

- Subscribers MAY include an "id" parameter in SUBSCRIBE request "Event" header field to allow differentiation between multiple subscriptions in the same dialog. This "id" parameter, if present, contains an opaque token which identifies the specific subscription within a dialog. An "id" parameter is only valid within the scope of a single dialog.

- If an "id" parameter is present in the SUBSCRIBE request used to establish a subscription, that "id" parameter MUST also be present in all corresponding NOTIFY requests.

- When a subscriber refreshes a the subscription timer, the SUBSCRIBE request MUST contain the same "Event" header field "id" parameter as was present in the SUBSCRIBE request that created the subscription. (Otherwise, the notifier will interpret the SUBSCRIBE request as a request for a new subscription in the same dialog).

- When a subscription is created in the notifier, it stores any "Event" header field "id" parameter as part of the subscription information (along with the event package name).

- If an initial SUBSCRIBE request is sent on a pre-existing dialog, a matching NOTIFY request merely creates a new subscription associated with that dialog.

4.6. CANCEL Requests for SUBSCRIBE and NOTIFY Transactions

Neither SUBSCRIBE nor NOTIFY requests can be canceled. If a UAS receives a CANCEL request that matches a known SUBSCRIBE or NOTIFY transaction, it MUST respond to the CANCEL request, but otherwise ignore it. In particular, the CANCEL request MUST NOT affect processing of the SUBSCRIBE or NOTIFY request in any way.

UACs SHOULD NOT send CANCEL requests for SUBSCRIBE or NOTIFY transactions.

5. Event Packages

This section covers several issues which should be taken into consideration when event packages based on the SUBSCRIBE and NOTIFY methods are proposed.

5.1. Appropriateness of Usage

When designing an event package using the methods described in this document for event notification, it is important to consider: is SIP an appropriate mechanism for the problem set? Is SIP being selected because of some unique feature provided by the protocol (e.g., user mobility), or merely because "it can be done?" If you find yourself defining event packages for notifications related to, for example, network management or the temperature inside your car's engine, you may want to reconsider your selection of protocols.

> Those interested in extending the mechanism defined in this document are urged to follow the development of "Guidelines for Authors of SIP Extensions" [RFC4485] for further guidance regarding appropriate uses of SIP.

Further, it is expected that this mechanism is not to be used in applications where the frequency of reportable events is excessively rapid (e.g., more than about once per second). A SIP network is generally going to be provisioned for a reasonable signaling volume; sending a notification every time a user's GPS position changes by one hundredth of a second could easily overload such a network.

5.2. Event Template-packages

Normal event packages define a set of state applied to a specific type of resource, such as user presence, call state, and messaging mailbox state.

Event template-packages are a special type of package which define a set of state applied to other packages, such as statistics, access policy, and subscriber lists. Event template-packages may even be applied to other event template-packages.

To extend the object-oriented analogy made earlier, event template-packages can be thought of as templatized C++ packages which must be applied to other packages to be useful.

The name of an event template-package as applied to a package is formed by appending a period followed by the event template-package name to the end of the package. For example, if a template-package called "winfo" were being applied to a package called "presence", the event token used in the "Event" header field would be "presence.winfo".

This scheme may be arbitrarily extended. For example, application of the "winfo" package to the the "presence.winfo" state of a resource would be represented by the name "presence.winfo.winfo". It naturally follows from this syntax that the order in which templates are specified is significant.

For example: consider a theoretical event template-package called "list". The event "presence.winfo.list" would be the application of the "list" template to "presence.winfo", which would presumably be a list of winfo state associated with presence. On the other hand, the event "presence.list.winfo" would represent the application of winfo to "presence.list", which would be represent the winfo state of a list of presence information.

Event template-packages must be defined so that they can be applied to any arbitrary package. In other words, event template-packages cannot be specifically tied to one or a few "parent" packages in such a way that they will not work with other packages.

5.3. Amount of State to be Conveyed

When designing event packages, it is important to consider the type of information which will be conveyed during a notification.

A natural temptation is to convey merely the event (e.g., "a new voice message just arrived") without accompanying state (e.g., "7 total voice messages"). This complicates implementation of subscribing entities (since they have to maintain complete state for the entity to which they have subscribed), and also is particularly susceptible to synchronization problems.

There are two possible solutions to this problem that event packages may choose to implement.

5.3.1. Complete State Information

In general, event packages need to be able to convey a well-defined and complete state, rather than just a stream of events. If it is not possible to describe complete system state for transmission in NOTIFY requests, then the problem set is not a good candidate for an event package.

For packages which typically convey state information that is reasonably small (on the order of 1 KB or so), it is suggested that event packages are designed so as to send complete state information whenever an event occurs.

In some circumstances, conveying the current state alone may be

insufficient for a particular class of events. In these cases, the event packages should include complete state information along with the event that occurred. For example, conveying "no customer service representatives available" may not be as useful as conveying "no customer service representatives available; representative sip:46@cs.xyz.int just logged off".

5.3.2. State Deltas

In the case that the state information to be conveyed is large, the event package may choose to detail a scheme by which NOTIFY requests contain state deltas instead of complete state.

Such a scheme would work as follows: any NOTIFY request sent in immediate response to a SUBSCRIBE request contains full state information. NOTIFY requests sent because of a state change will contain only the state information that has changed; the subscriber will then merge this information into its current knowledge about the state of the resource.

Any event package that supports delta changes to states MUST include a version number that increases by exactly one for each NOTIFY transaction in a subscription. Note that the state version number appears in the body of the message, not in a SIP header field.

If a NOTIFY request arrives that has a version number that is incremented by more than one, the subscriber knows that a state delta has been missed; it ignores the NOTIFY request containing the state delta (except for the version number, which it retains to detect message loss), and re-sends a SUBSCRIBE request to force a NOTIFY request containing a complete state snapshot.

5.4. Event Package Responsibilities

Event packages are not required to reiterate any of the behavior described in this document, although they may choose to do so for clarity or emphasis. In general, though, such packages are expected to describe only the behavior that extends or modifies the behavior described in this document.

Note that any behavior designated with "SHOULD" or "MUST" in this document is not allowed to be weakened by extension documents; however, such documents may elect to strengthen "SHOULD" requirements to "MUST" strength if required by their application.

In addition to the normal sections expected in standards-track RFCs and SIP extension documents, authors of event packages need to address each of the issues detailed in the following subsections.

For clarity: well-formed event package definitions contain sections addressing each of these issues, ideally in the same order and with the same titles as these subsections.

5.4.1. Event Package Name

This section, which MUST be present, defines the token name to be used to designate the event package. It MUST include the information which appears in the IANA registration of the token. For information on registering such types, see Section 7.

5.4.2. Event Package Parameters

If parameters are to be used on the "Event" header field to modify the behavior of the event package, the syntax and semantics of such header fields MUST be clearly defined.

Any "Event" header field parameters defined by an event package MUST be registered in the "Header Field Parameters and Parameter Values" registry defined by [RFC3968]. An "Event" header field parameter, once registered in conjunction with an event package, MUST NOT be re-used with any other event package. Non-event-package specifications MAY define "Event" header field parameters that apply across all event packages (with emphasis on "all", as opposed to "several"), such as the "id" parameter defined in this document. The restriction of a parameter to use with a single event package only applies to parameters that are defined in conjunction with an event package.

5.4.3. SUBSCRIBE Request Bodies

It is expected that most, but not all, event packages will define syntax and semantics for SUBSCRIBE request bodies; these bodies will typically modify, expand, filter, throttle, and/or set thresholds for the class of events being requested. Designers of event packages are strongly encouraged to re-use existing media types for message bodies where practical. See [RFC4288] for information on media type specification and registration.

This mandatory section of an event package defines what type or types of event bodies are expected in SUBSCRIBE requests (or specify that no event bodies are expected). It should point to detailed definitions of syntax and semantics for all referenced body types.

5.4.4. Subscription Duration

It is RECOMMENDED that event packages give a suggested range of times considered reasonable for the duration of a subscription. Such packages MUST also define a default "Expires" value to be used if

none is specified.

5.4.5. NOTIFY Request Bodies

The NOTIFY request body is used to report state on the resource being monitored. Each package MUST define what type or types of event bodies are expected in NOTIFY requests. Such packages MUST specify or cite detailed specifications for the syntax and semantics associated with such event body.

Event packages also MUST define which media type is to be assumed if none are specified in the "Accept" header field of the SUBSCRIBE request.

5.4.6. Notifier processing of SUBSCRIBE requests

This section describes the processing to be performed by the notifier upon receipt of a SUBSCRIBE request. Such a section is required.

Information in this section includes details of how to authenticate subscribers and authorization issues for the package.

5.4.7. Notifier generation of NOTIFY requests

This section of an event package describes the process by which the notifier generates and sends a NOTIFY request. This includes detailed information about what events cause a NOTIFY request to be sent, how to compute the state information in the NOTIFY, how to generate neutral or fake state information to hide authorization delays and decisions from users, and whether state information is complete or deltas for notifications; see Section 5.3. Such a section is required.

This section may optionally describe the behavior used to process the subsequent response.

5.4.8. Subscriber processing of NOTIFY requests

This section of an event package describes the process followed by the subscriber upon receipt of a NOTIFY request, including any logic required to form a coherent resource state (if applicable).

5.4.9. Handling of forked requests

Each event package MUST specify whether forked SUBSCRIBE requests are allowed to install multiple subscriptions.

If such behavior is not allowed, the first potential dialog-

establishing message will create a dialog. All subsequent NOTIFY requests which correspond to the SUBSCRIBE request (i.e., match "To", "From", "From" header field "tag" parameter, "Call-ID", "Event", and "Event" header field "id" parameter) but which do not match the dialog would be rejected with a 481 response. Note that the 200-class response to the SUBSCRIBE request can arrive after a matching NOTIFY request has been received; such responses might not correlate to the same dialog established by the NOTIFY request. Except as required to complete the SUBSCRIBE transaction, such non-matching 200-class responses are ignored.

If installing of multiple subscriptions by way of a single forked SUBSCRIBE request is allowed, the subscriber establishes a new dialog towards each notifier by returning a 200-class response to each NOTIFY request. Each dialog is then handled as its own entity, and is refreshed independent of the other dialogs.

In the case that multiple subscriptions are allowed, the event package MUST specify whether merging of the notifications to form a single state is required, and how such merging is to be performed. Note that it is possible that some event packages may be defined in such a way that each dialog is tied to a mutually exclusive state which is unaffected by the other dialogs; this MUST be clearly stated if it is the case.

5.4.10. Rate of notifications

Each event package is expected to define a requirement (SHOULD or MUST strength) which defines an absolute maximum on the rate at which notifications are allowed to be generated by a single notifier.

Each package MAY further define a throttle mechanism which allows subscribers to further limit the rate of notification.

5.4.11. State Aggregation

Many event packages inherently work by collecting information about a resource from a number of other sources -- either through the use of PUBLISH [RFC3903], by subscribing to state information, or through other state gathering mechanisms.

Event packages that involve retrieval of state information for a single resource from more than one source need to consider how notifiers aggregate information into a single, coherent state. Such packages MUST specify how notifiers aggregate information and how they provide authentication and authorization.

5.4.12. Examples

Event packages SHOULD include several demonstrative message flow diagrams paired with several typical, syntactically correct, and complete messages.

It is RECOMMENDED that documents describing event packages clearly indicate that such examples are informative and not normative, with instructions that implementors refer to the main text of the document for exact protocol details.

5.4.13. Use of URIs to Retrieve State

Some types of event packages may define state information which is potentially too large to reasonably send in a SIP message. To alleviate this problem, event packages may include the ability to convey a URI instead of state information; this URI will then be used to retrieve the actual state information.

[RFC4483] defines a mechanism that can be used by event packages to convey information in such a fashion.

6. Security Considerations

6.1. Access Control

The ability to accept subscriptions should be under the direct control of the notifier's user, since many types of events may be considered sensitive for the purposes of privacy. Similarly, the notifier should have the ability to selectively reject subscriptions based on the subscriber identity (based on access control lists), using standard SIP authentication mechanisms. The methods for creation and distribution of such access control lists is outside the scope of this document.

6.2. Notifier Privacy Mechanism

The mere act of returning certain 4xx and 6xx responses to SUBSCRIBE requests may, under certain circumstances, create privacy concerns by revealing sensitive policy information. In these cases, the notifier SHOULD always return a 200 (OK) response. While the subsequent NOTIFY request may not convey true state, it MUST appear to contain a potentially correct piece of data from the point of view of the subscriber, indistinguishable from a valid response. Information about whether a user is authorized to subscribe to the requested state is never conveyed back to the original user under these circumstances.

Individual packages and their related documents for which such a mode of operation makes sense can further describe how and why to generate such potentially correct data. For example, such a mode of operation is mandated by [RFC2779] for user presence information.

6.3. Denial-of-Service attacks

The current model (one SUBSCRIBE request triggers a SUBSCRIBE response and one or more NOTIFY requests) is a classic setup for an amplifier node to be used in a smurf attack.

Also, the creation of state upon receipt of a SUBSCRIBE request can be used by attackers to consume resources on a victim's machine, rendering it unusable.

To reduce the chances of such an attack, implementations of notifiers SHOULD require authentication. Authentication issues are discussed in [RFC3261].

6.4. Replay Attacks

Replaying of either SUBSCRIBE or NOTIFY requests can have detrimental effects.

In the case of SUBSCRIBE requests, attackers may be able to install any arbitrary subscription which it witnessed being installed at some point in the past. Replaying of NOTIFY requests may be used to spoof old state information (although a good versioning mechanism in the body of the NOTIFY requests may help mitigate such an attack). Note that the prohibition on sending NOTIFY requests to nodes which have not subscribed to an event also aids in mitigating the effects of such an attack.

To prevent such attacks, implementations SHOULD require authentication with anti-replay protection. Authentication issues are discussed in [RFC3261].

6.5. Man-in-the middle attacks

Even with authentication, man-in-the-middle attacks using SUBSCRIBE requests may be used to install arbitrary subscriptions, hijack existing subscriptions, terminate outstanding subscriptions, or modify the resource to which a subscription is being made. To prevent such attacks, implementations SHOULD provide integrity protection across "Contact", "Route", "Expires", "Event", and "To" header fields of SUBSCRIBE requests, at a minimum. If SUBSCRIBE request bodies are used to define further information about the state of the call, they SHOULD be included in the integrity protection

scheme.

Man-in-the-middle attacks may also attempt to use NOTIFY requests to spoof arbitrary state information and/or terminate outstanding subscriptions. To prevent such attacks, implementations SHOULD provide integrity protection across the "Call-ID", "CSeq", and "Subscription-State" header fields and the bodies of NOTIFY requests.

Integrity protection of message header fields and bodies is discussed in [RFC3261].

6.6. Confidentiality

The state information contained in a NOTIFY request has the potential to contain sensitive information. Implementations MAY encrypt such information to ensure confidentiality.

While less likely, it is also possible that the information contained in a SUBSCRIBE request contains information that users might not want to have revealed. Implementations MAY encrypt such information to ensure confidentiality.

To allow the remote party to hide information it considers sensitive, all implementations SHOULD be able to handle encrypted SUBSCRIBE and NOTIFY requests.

The mechanisms for providing confidentiality are detailed in [RFC3261].

7. IANA Considerations

(This section is not applicable until this document is published as an RFC.)

With the exception of Section 7.2, the subsections here are for current reference, carried over from the original specification. The only IANA actions requested here are updating all registry references that point to RFC 3265 to instead indicate this document, and creating the new "reason code" registry described in Section 7.2.

7.1. Event Packages

This document defines an event-type namespace which requires a central coordinating body. The body chosen for this coordination is the Internet Assigned Numbers Authority (IANA).

There are two different types of event-types: normal event packages,

and event template-packages; see Section 5.2. To avoid confusion, template-package names and package names share the same namespace; in other words, an event template-package are forbidden from sharing a name with a package.

Policies for registration of SIP event packages and SIP event package templates are defined in section 4.1 of [RFC5727].

Registrations with the IANA are required to include the token being registered and whether the token is a package or a template-package. Further, packages must include contact information for the party responsible for the registration and/or a published document which describes the event package. Event template-package token registrations are also required to include a pointer to the published RFC which defines the event template-package.

Registered tokens to designate packages and template-packages are disallowed from containing the character ".", which is used to separate template-packages from packages.

7.1.1. Registration Information

As this document specifies no package or template-package names, the initial IANA registry for event types will be empty. The remainder of the text in this section gives an example of the type of information to be maintained by the IANA; it also demonstrates all five possible permutations of package type, contact, and reference.

The table below lists the event packages and template-packages defined in "SIP-Specific Event Notification" [RFC xxxx]. Each name is designated as a package or a template-package under "Type".

Package Name	Type	Contact	Reference
example1	package	[Roach]	
example2	package	[Roach]	[RFC xxxx]
example3	package		[RFC xxxx]
example4	template	[Roach]	[RFC xxxx]
example5	template		[RFC xxxx]

PEOPLE

[Roach] Adam Roach <adam.roach@tekelec.com>

REFERENCES

[RFC xxxx] A.B. Roach, "SIP-Specific Event Notification", RFC XXXX,

7.1.2. Registration Template

To: ietf-sip-events@iana.org
Subject: Registration of new SIP event package

Package Name:

(Package names must conform to the syntax described in Section 8.2.1.)

Is this registration for a Template Package:

(indicate yes or no)

Published Specification(s):

(Template packages require a published RFC. Other packages may reference a specification when appropriate).

Person & email address to contact for further information:

7.2. Reason Codes

This document further defines "reason" codes for use in the "Subscription-State" header field (see Section 4.1.3).

Following the policies outlined in "Guidelines for Writing an IANA Considerations Section in RFCs" [RFC5226], new reason codes require a Standards Action.

Registrations with the IANA include the reason code being registered and a reference to a published document which describes the event package. Insertion of such values takes place as part of the RFC publication process or as the result of inter-SDO liaison activity, the result of which will be publication of an associated RFC. New reason codes must conform to the syntax of the ABNF "token" element defined in [RFC3261].

[RFC4660] defined a new reason code prior to the establishment of an IANA registry. We include its reason code ("badfilter") in the initial list of reason codes to ensure a complete registry.

The IANA registry for reason code will be initialized with the following values:

Reason Code	Reference
deactivated	[RFC xxxx]
probation	[RFC xxxx]
rejected	[RFC xxxx]
timeout	[RFC xxxx]
giveup	[RFC xxxx]
noresource	[RFC xxxx]
invariant	[RFC xxxx]
badfilter	[RFC 4660]

REFERENCES

[RFC xxxx] A.B. Roach, "SIP-Specific Event Notification", RFC XXXX, Monthname 20XX

[RFC 4660] Khartabil, H., Leppanen, E., Lonnfors, M., and J. Costa-Requena, "Functional Description of Event Notification Filtering", September 2006.

7.3. Header Field Names

This document registers three new header field names, described elsewhere in this document. These header fields are defined by the following information, which is to be added to the header field sub-registry under http://www.iana.org/assignments/sip-parameters.

Header Name: Allow-Events
Compact Form: u

Header Name: Subscription-State
Compact Form: (none)

Header Name: Event
Compact Form: o

7.4. Response Codes

This document registers two new response codes. These response codes are defined by the following information, which is to be added to the method and response-code sub-registry under http://www.iana.org/assignments/sip-parameters.

Response Code Number: 202
Default Reason Phrase: Accepted

Response Code Number: 489
Default Reason Phrase: Bad Event

8. Syntax

This section describes the syntax extensions required for event notification in SIP. Semantics are described in Section 4. Note that the formal syntax definitions described in this document are expressed in the ABNF format used in [RFC3261], and contain references to elements defined therein.

8.1. New Methods

This document describes two new SIP methods: SUBSCRIBE and NOTIFY.

8.1.1. SUBSCRIBE method

"SUBSCRIBE" is added to the definition of the element "Method" in the SIP message grammar.

Like all SIP method names, the SUBSCRIBE method name is case sensitive. The SUBSCRIBE method is used to request asynchronous notification of an event or set of events at a later time.

8.1.2. NOTIFY method

"NOTIFY" is added to the definition of the element "Method" in the SIP message grammar.

The NOTIFY method is used to notify a SIP node that an event which has been requested by an earlier SUBSCRIBE method has occurred. It may also provide further details about the event.

8.2. New Header Fields

8.2.1. "Event" Header Field

Event is added to the definition of the element "message-header field" in the SIP message grammar.

For the purposes of matching NOTIFY requests with SUBSCRIBE requests, the event-type portion of the "Event" header field is compared byte-by-byte, and the "id" parameter token (if present) is compared byte-by-byte. An "Event" header field containing an "id" parameter never matches an "Event" header field without an "id" parameter. No other parameters are considered when performing a comparison. SUBSCRIBE responses are matched per the transaction handling rules in [RFC3261].

> Note that the forgoing text means that "Event: foo; id=1234" would match "Event: foo; param=abcd; id=1234", but not "Event: foo" (id does not match) or "Event: Foo; id=1234" (event portion does not match).

This document does not define values for event-types. These values will be defined by individual event packages, and MUST be registered with the IANA.

There MUST be exactly one event type listed per event header field. Multiple events per message are disallowed.

The "Event" header field is defined only for use in SUBSCRIBE and NOTIFY requests, and other requests whose definition explicitly calls for its use. It MUST NOT appear in any other SIP requests, and MUST NOT appear in responses.

8.2.2. "Allow-Events" Header Field

Allow-Events is added to the definition of the element "general-header field" in the SIP message grammar. Its usage is described in Section 4.4.4.

User Agents MAY include the "Allow-Events" header field in any request or response, as long as its contents comply with the behavior described in Section 4.4.4.

8.2.3. "Subscription-State" Header Field

Subscription-State is added to the definition of the element "request-header field" in the SIP message grammar. Its usage is described in Section 4.1.3. "Subscription-State" header fields are defined for use in NOTIFY requests only. They MUST NOT appear in other SIP requests or responses.

8.3. New Response Codes

8.3.1. "202 Accepted" Response Code

For historical purposes, the 202 (Accepted) response code is added to the "Success" header field definition.

This document does not specify the use of the 202 response code in conjunction with the SUBSCRIBE or NOTIFY methods. Previous versions of the SIP Events Framework assigned specific meaning to the 202 response code.

Due to response handling in forking cases, any 202 response to a

SUBSCRIBE request may be absorbed by a proxy, and thus it can never be guaranteed to be received by the UAC. Furthermore, there is no actual processing difference for a 202 as compared to a 200; a NOTIFY request is sent after the subscription is processed, and it conveys the correct state. SIP interoperability tests found that implementations were handling 202 differently from 200, leading to incompatibilities. Therefore, the 202 response is being deprecated to make it clear there is no such difference and 202 should not be handled differently than 200.

Implementations conformant with the current specification MUST treat an incoming 202 response as identical to a 200 response, and MUST NOT generate 202 response codes to SUBSCRIBE or NOTIFY requests.

This document also updates [RFC4660], which reiterates the 202-based behavior in several places. Implementations compliant with the present document MUST NOT send a 202 response to a SUBSCRIBE request, and will send an alternate success response (such as 200) in its stead.

8.3.2. "489 Bad Event" Response Code

The 489 event response is added to the "Client-Error" header field field definition. "489 Bad Event" is used to indicate that the server did not understand the event package specified in a "Event" header field.

8.4. Augmented BNF Definitions

The Augmented BNF definitions for the various new and modified syntax elements follows. The notation is as used in [RFC3261], and any elements not defined in this section are as defined in SIP and the documents to which it refers.

```
   SUBSCRIBEm        = %x53.55.42.53.43.52.49.42.45 ; SUBSCRIBE in caps
   NOTIFYm           = %x4E.4F.54.49.46.59 ; NOTIFY in caps
   extension-method = SUBSCRIBEm / NOTIFYm / token

   Event             =  ( "Event" / "o" ) HCOLON event-type
                        *( SEMI event-param )
   event-type        =  event-package *( "." event-template )
   event-package     =  token-nodot
   event-template    =  token-nodot
   token-nodot       =  1*( alphanum / "-"  / "!" / "%" / "*"
                          / "_" / "+" / "`" / "'" / "~" )

   ; The use of the "id" parameter is deprecated; it is included
   ; for backwards compatibility purposes only.
   event-param      =  generic-param / ( "id" EQUAL token )

   Allow-Events =  ( "Allow-Events" / "u" ) HCOLON event-type
                  *(COMMA event-type)

   Subscription-State   = "Subscription-State" HCOLON substate-value
                          *( SEMI subexp-params )
   substate-value       = "active" / "pending" / "terminated"
                          / extension-substate
   extension-substate   = token
   subexp-params        =   ("reason" EQUAL event-reason-value)
                          / ("expires" EQUAL delta-seconds)
                          / ("retry-after" EQUAL delta-seconds)
                          / generic-param
   event-reason-value   =   "deactivated"
                          / "probation"
                          / "rejected"
                          / "timeout"
                          / "giveup"
                          / "noresource"
                          / "invariant"
                          / event-reason-extension
   event-reason-extension = token
```

9. References

9.1. Normative References

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

[RFC2848] Petrack, S. and L. Conroy, "The PINT Service Protocol: Extensions to SIP and SDP for IP Access to Telephone Call

Services", RFC 2848, June 2000.

[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.

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

[RFC4483] Burger, E., "A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages", RFC 4483, May 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5627] Rosenberg, J., "Obtaining and Using Globally Routable User Agent URIs (GRUUs) in the Session Initiation Protocol (SIP)", RFC 5627, October 2009.

[RFC5727] Peterson, J., Jennings, C., and R. Sparks, "Change Process for the Session Initiation Protocol (SIP) and the Real-time Applications and Infrastructure Area", BCP 67, RFC 5727, March 2010.

9.2. Informative References

[RFC2779] Day, M., Aggarwal, S., Mohr, G., and J. Vincent, "Instant Messaging / Presence Protocol Requirements", RFC 2779, February 2000.

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

[RFC3840] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.

[RFC3891] Mahy, R., Biggs, B., and R. Dean, "The Session Initiation Protocol (SIP) "Replaces" Header", RFC 3891, September 2004.

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

[RFC3911] Mahy, R. and D. Petrie, "The Session Initiation Protocol (SIP) "Join" Header", RFC 3911, October 2004.

[RFC4235] Rosenberg, J., Schulzrinne, H., and R. Mahy, "An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP)", RFC 4235, November 2005.

[RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

[RFC4485] Rosenberg, J. and H. Schulzrinne, "Guidelines for Authors of Extensions to the Session Initiation Protocol (SIP)", RFC 4485, May 2006.

[RFC4538] Rosenberg, J., "Request Authorization through Dialog Identification in the Session Initiation Protocol (SIP)", RFC 4538, June 2006.

[RFC4660] Khartabil, H., Leppanen, E., Lonnfors, M., and J. Costa-Requena, "Functional Description of Event Notification Filtering", RFC 4660, September 2006.

[RFC5057] Sparks, R., "Multiple Dialog Usages in the Session Initiation Protocol", RFC 5057, November 2007.

[RFC5839] Niemi, A. and D. Willis, "An Extension to Session Initiation Protocol (SIP) Events for Conditional Event Notification", RFC 5839, May 2010.

Appendix A. Acknowledgements

Thanks to the participants in the Events BOF at the 48th IETF meeting in Pittsburgh, as well as those who gave ideas and suggestions on the SIP Events mailing list. In particular, I wish to thank Henning Schulzrinne of Columbia University for coming up with the final three-tiered event identification scheme, Sean Olson for miscellaneous guidance, Jonathan Rosenberg for a thorough scrubbing of the -00 draft, and the authors of the "SIP Extensions for Presence" document for their input to SUBSCRIBE and NOTIFY request semantics.

I also owe a debt of gratitude to all the implementors who have provided feedback on areas of confusion or difficulty in the original specification. In particular, Robert Sparks' Herculean efforts

organizing, running, and collecting data from the SIPit events have proven invaluable in shaking out specification bugs. Robert Sparks is also responsible for untangling the dialog usage mess, in the form of RFC 5057 [RFC5057].

Appendix B. Changes from RFC 3265

This document represents several changes from the mechanism originally described in RFC 3265. This section summarizes those changes. Bug numbers refer to the identifiers for the bug reports kept on file at http://bugs.sipit.net/.

B.1. Bug 666: Clarify use of expires=xxx with terminated

Strengthened language in Section 4.1.3 to clarify that expires should not be sent with terminated, and must be ignored if received.

B.2. Bug 667: Reason code for unsub/poll not clearly spelled out

Clarified description of "timeout" in Section 4.1.3. (n.b., the text in Section 4.4.3 is actually pretty clear about this).

B.3. Bug 669: Clarify: SUBSCRIBE for a duration might be answered with a NOTIFY/expires=0

Added clarifying text to Section 4.2.2 explaining that shortening a subscription to zero seconds is valid. Also added sentence to Section 3.1.1 explicitly allowing shortening to zero.

B.4. Bug 670: Dialog State Machine needs clarification

The issues associated with the bug deal exclusively with the handling of multiple usages with a dialog. This behavior has been deprecated and moved to Section 4.5.2. This section, in turn, cites [RFC5057], which addresses all of the issues in Bug 670.

B.5. Bug 671: Clarify timeout-based removal of subscriptions

Changed Section 4.2.2 to specifically cite Timer F (so as to avoid ambiguity between transaction timeouts and retransmission timeouts).

B.6. Bug 672: Mandate expires= in NOTIFY

Changed strength of including of "expires" in a NOTIFY from SHOULD to MUST in Section 4.2.2.

B.7. Bug 673: INVITE 481 response effect clarification

This bug was addressed in [RFC5057].

B.8. Bug 677: SUBSCRIBE response matching text in error

Fixed Section 8.2.1 to remove incorrect "...responses and..." -- explicitly pointed to SIP for transaction response handling.

B.9. Bug 695: Document is not explicit about response to NOTIFY at subscription termination

Added text to Section 4.4.1 indicating that the typical response to a terminal NOTIFY is a "200 OK".

B.10. Bug 696: Subscription state machine needs clarification

Added state machine diagram to Section 4.1.2 with explicit handling of what to do when a SUBSCRIBE never shows up. Added definition of and handling for new Timer N to Section 4.1.2.4. Added state machine to Section 4.2.2 to reinforce text.

B.11. Bug 697: Unsubscription behavior could be clarified

Added text to Section 4.2.1.4 encouraging (but not requiring) full state in final NOTIFY request. Also added text to Section 4.1.2.3 warning subscribers that full state may or may not be present in the final NOTIFY.

B.12. Bug 699: NOTIFY and SUBSCRIBE are target refresh requests

Added text to both Section 3.1 and Section 3.2 explicitly indicating that SUBSCRIBE and NOTIFY are target refresh methods.

B.13. Bug 722: Inconsistent 423 reason phrase text

Changed reason code to "Interval Too Brief" in Section 4.2.1.1 and Section 4.2.1.4, to match 423 reason code in SIP [RFC3261].

B.14. Bug 741: guidance needed on when to not include Allow-Events

Added non-normative clarification to Section 4.4.4 regarding inclusion of Allow-Events in a NOTIFY for the one-and-only package supported by the notifier.

B.15. Bug 744: 5xx to NOTIFY terminates a subscription, but should not

Issue of subscription (usage) termination versus dialog termination is handled in [RFC5057]. The text in Section 4.2.2 has been updated to summarize the behavior described by 5057, and cites it for additional detail and rationale.

B.16. Bug 752: Detection of forked requests is incorrect

Removed erroneous "CSeq" from list of matching criteria in Section 5.4.9.

B.17. Bug 773: Reason code needs IANA registry

Added Section 7.2 to create and populate IANA registry.

B.18. Bug 774: Need new reason for terminating subscriptions to resources that never change

Added new "invariant" reason code to Section 4.1.3, ABNF syntax.

B.19. Clarify handling of Route/Record-Route in NOTIFY

Changed text in Section 4.3 mandating Record-Route in initial SUBSCRIBE and all NOTIFY requests, and adding "MAY" level statements for subsequent SUBSCRIBE requests.

B.20. Eliminate implicit subscriptions

Added text to Section 4.2.1 explaining some of the problems associated with implicit subscriptions, normative language prohibiting them. Removed language from Section 3.2 describing "non-SUBSCRIBE" mechanisms for creating subscriptions. Simplified language in Section 4.2.2, now that the soft-state/non-soft-state distinction is unnecessary.

B.21. Deprecate dialog re-use

Moved handling of dialog re-use and "id" handling to Section 4.5.2. It is documented only for backwards-compatibility purposes.

B.22. Rationalize dialog creation

Section 4.4.1 has been updated to specify that dialogs should be created when the NOTIFY arrives. Previously, the dialog was established by the SUBSCRIBE 200, or by the NOTIFY transaction. This was unnecessarily complicated; the newer rules are easier to implement (and result in effectively the same behavior on the wire).

B.23. Refactor behavior sections

Reorganized Section 4 to consolidate behavior along role lines (subscriber/notifier/proxy) instead of method lines.

B.24. Clarify sections that need to be present in event packages

Added sentence to Section 5 clarifying that event packages are expected to include explicit sections covering the issues discussed in this section.

B.25. Make CANCEL handling more explicit

Text in Section 4.6 now clearly calls out behavior upon receipt of a CANCEL. We also echo the "...SHOULD NOT send..." requirement from [RFC3261].

B.26. Remove State Agent Terminology

As originally planned, we anticipated a fairly large number of event packages that would move back and forth between end-user devices and servers in the network. In practice, this has ended up not being the case. Certain events, like dialog state, are inherently hosted at end-user devices; others, like presence, are almost always hosted in the network (due to issues like composition, and the ability to deliver information when user devices are offline). Further, the concept of State Agents is the most misunderstood by event package authors. In my expert review of event packages, I have yet to find one that got the concept of State Agents completely correct -- and most of them start out with the concept being 100% backwards from the way RFC 3265 described it.

Rather than remove the ability to perform the actions previously attributed to the widely misunderstood term "State Agent," we have simply eliminated this term. Instead, we talk about the behaviors required to create state agents (state aggregation, subscription notification) without defining a formal term to describe the servers that exhibit these behaviors. In effect, this is an editorial change to make life easier for event package authors; the actual protocol does not change as a result.

The definition of "State Agent" has been removed from Section 2. Section 4.4.2 has been retooled to discuss migration of subscription in general, without calling out the specific example of state agents. Section 5.4.11 has been focused on state aggregation in particular, instead of state aggregation as an aspect of state agents.

B.27. Miscellanous Changes

The following changes are relatively minor revisions to the document that resulted primarily from review of this document in the working group and IESG, rather than implementation reports.

- Clarified scope of Event header field parameters. In RFC3265, the scope is ambiguous, which causes problems with the RFC3968 registry. The new text ensures that Event header field parameters are unique across all event packages.

- Removed obsoleted language around IANA registration policies for event packages. Instead, we now cite RFC5727, which supersedes RFC3265, and is authoritative on event package registration policy.

- Several editorial updates after input from working group, including proper designation of "dialog usage" rather than "dialog" where appropriate.

- Clarified two normative statements about subscription termination by changing from plain English prose to RFC2119 language.

- Removed "Table 2" expansions, per WG consensus on how SIP table 2 is to be handled.

- Removed 202 response code.

- Clarified that "Allow-Events" does not list event template packages.

- Added clarification about proper response when the SUBSCRIBE indicates an unknown media type in its Accept header field.

- Minor clarifications to Route and Record-Route behavior.

- Added non-normative warning about the limitations of state polling.

- Added information about targeting subscriptions at specific dialogs.

- Added RFC 3261 to list of documents updated by this one (rather than the "2543" indicated by RFC3265).

- Clarified text in Section 3.1.1 explaining the meaning of "Expires: 0".

- Changed text in definition of "probation" reason code to indicate that subscribers don't need to re-subscribe if the associated state is no longer of use to them.

- Specified that the termination of a subscription due to a NOTIFY transaction failure does not require sending another NOTIFY message.

- Clarified how order of template application affects the meaning of an Event header field value. (e.g., "foo.bar.baz" is different than "foo.baz.bar").

Author's Address

Adam Roach
Tekelec
17210 Campbell Rd.
Suite 250
Dallas, TX 75252
US

Email: adam@nostrum.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 13, 2011

C. Jennings
Cisco Systems
K. Ono
Columbia University
R. Sparks
B. Hibbard, Ed.
Tekelec

February 9, 2011

Example call flows using Session Initiation Protocol (SIP) security mechanisms
draft-ietf-sipcore-sec-flows-09

Abstract

This document shows example call flows demonstrating the use of Transport Layer Security (TLS), and Secure/Multipurpose Internet Mail Extensions (S/MIME) in Session Initiation Protocol (SIP). It also provides information that helps implementers build interoperable SIP software. To help facilitate interoperability testing, it includes certificates used in the example call flows and processes to create certificates for testing.

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 13, 2011.

Copyright Notice

Table of Contents

1. Introduction

This document is informational and is not normative on any aspect of SIP.

SIP with TLS ([RFC5246]) implementations are becoming very common. Several implementations of the S/MIME ([RFC5751]) portion of SIP ([RFC3261]) are also becoming available. After several interoperability events, it is clear that it is difficult to write these systems without any test vectors or examples of "known good" messages to test against. Furthermore, testing at the events is often hindered due to the lack of a commonly trusted certificate authority to sign the certificates used in the events. This document addresses both of these issues by providing messages that give detailed examples that implementers can use for comparison and that can also be used for testing. In addition, this document provides a common certificate and private key that can be used to set up a mock Certificate Authority (CA) that can be used during the SIP interoperability events. Certificate requests from the users will be signed by the private key of the mock CA. The document also provides some hints and clarifications for implementers.

A simple SIP call flow using SIPS URIs and TLS is shown in Section 3. The certificates for the hosts used are shown in Section 2.2, and the CA certificates used to sign these are shown in Section 2.1.

The text from Section 4.1 through Section 4.3 shows some simple SIP call flows using S/MIME to sign and encrypt the body of the message. The user certificates used in these examples are shown in Section 2.3. These host certificates are signed with the same mock CA private key.

Section 5 presents a partial list of items that implementers should consider in order to implement systems that will interoperate.

Scripts and instructions to make certificates that can be used for interoperability testing are presented in Appendix A, along with methods for converting these to various formats. The certificates used while creating the examples and test messages in this document are made available in Appendix B.

Binary copies of various messages in this document that can be used for testing appear in Appendix C.

2. Certificates

2.1. CA Certificates

The certificate used by the CA to sign the other certificates is shown below. This is a X509v3 certificate. Note that the X.509v3 Basic Constraints in the certificate allows it to be used as a CA, certificate authority. This certificate is not used directly in the TLS call flow; it is used only to verify user and host certificates.

```
Version: 3 (0x2)
Serial Number:
    96:a3:84:17:4e:ef:8a:4c
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit,
        OU=Sipit Test Certificate Authority
Validity
    Not Before: Jan 27 18:36:05 2011 GMT
    Not After : Jan  3 18:36:05 2111 GMT
Subject: C=US, ST=California, L=San Jose, O=sipit,
        OU=Sipit Test Certificate Authority
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (2048 bit)
        Modulus (2048 bit):
            00:ab:1f:91:61:f1:1c:c5:cd:a6:7b:16:9b:b7:14:
            79:e4:30:9e:98:d0:ec:07:b7:bd:77:d7:d1:f5:5b:
            2c:e2:ee:e6:b1:b0:f0:85:fa:a5:bc:cb:cc:cf:69:
            2c:4f:fc:50:ef:9d:31:2b:c0:59:ea:fb:64:6f:1f:
            55:a7:3d:fd:70:d2:56:db:14:99:17:92:70:ac:26:
            f8:34:41:70:d9:c0:03:91:6a:ba:d1:11:8f:ac:12:
            31:de:b9:19:70:8d:5d:a7:7d:8b:19:cc:40:3f:ae:
            ff:de:1f:db:94:b3:46:77:6c:ae:ae:ff:3e:d6:84:
            5b:c2:de:0b:26:65:d0:91:c7:70:4b:c7:0a:4a:bf:
            c7:97:04:dd:ba:58:47:cb:e0:2b:23:76:87:65:c5:
            55:34:10:ab:27:1f:1c:f8:30:3d:b0:9b:ca:a2:81:
            72:4c:bd:60:fe:f7:21:fe:0b:db:0b:db:e9:5b:01:
            36:d4:28:15:6b:79:eb:d0:91:1b:21:59:b8:0e:aa:
            bf:d5:b1:6c:70:37:a3:3f:a5:7d:0e:95:46:f6:f6:
            58:67:83:75:42:37:18:0b:a4:41:39:b2:2f:6c:80:
            2c:78:ec:a5:0f:be:9c:10:f8:c0:0b:0d:73:99:9e:
            0d:d7:97:50:cb:cc:45:34:23:49:41:85:22:24:ad:
            29:c3
        Exponent: 65537 (0x10001)
X509v3 extensions:
    X509v3 Subject Key Identifier:
        95:45:7E:5F:2B:EA:65:98:12:91:04:F3:63:C7:68:9A:58:16:77:27
    X509v3 Authority Key Identifier:
```

```
          95:45:7E:5F:2B:EA:65:98:12:91:04:F3:63:C7:68:9A:58:16:77:27

      X509v3 Basic Constraints:
          CA:TRUE
      Signature Algorithm: sha1WithRSAEncryption
   06:5f:9e:ae:a0:9a:bc:b5:b9:5b:7e:97:33:cc:df:63:98:98:
   94:cb:0d:66:a9:83:e8:aa:58:2a:59:a1:9e:47:31:a6:af:5c:
   3f:a2:25:86:f8:df:05:92:b7:db:69:a1:69:72:87:66:c5:ab:
   35:89:01:37:19:c9:74:eb:09:d1:3f:88:7b:24:13:42:ca:2d:
   fb:45:e6:cc:4b:f8:21:78:f3:f5:97:ec:09:92:24:a2:f0:e6:
   94:8d:97:4a:00:94:00:bd:25:b8:17:2c:52:53:5d:cc:5c:48:
   a4:a1:1d:2d:f6:50:55:13:a4:d3:b2:a2:f4:f1:b9:6d:48:5e:
   5c:f3:de:e0:fc:59:09:a1:d9:14:61:65:bf:d8:3f:b9:ba:2e:
   7c:ed:5c:24:9b:6b:ca:aa:5f:f1:c1:1e:b0:a8:da:82:0f:fb:
   4c:71:3b:4d:7b:38:c8:e3:8a:2a:19:34:44:26:0b:ea:f0:47:
   38:46:28:65:04:e2:01:52:dd:ec:3d:e5:f5:53:74:77:74:75:
   6d:c6:d9:c2:0a:ac:3b:b8:98:5c:55:53:34:74:52:a8:26:b1:
   2f:30:22:d0:8b:b7:f3:a0:dd:68:07:33:d5:ae:b7:81:b2:94:
   58:72:4e:7c:c6:72:2f:bd:6c:69:fb:b5:17:a8:2a:8d:d7:2c:
   91:06:c8:0c
```

The certificate content shown above and throughout this document was rendered by the OpenSSL "x509" tool. These dumps are included only as informative examples. Output may vary among future revisions of the tool. At the time of this document's publication, there were some irregularities in the presentation of Distinguished Names (DN). In particular, note that in the "Issuer" and "Subject" fields, it appears the intent is to present DNs in Lightweight Directory Access Protocol (LDAP) format. If this was intended, the spaces should have been omitted after the delimiting commas, and the elements should have been presented in order of most-specific to least-specific. Please refer to Appendix A of [RFC4514]. Using the "Issuer" DN from above as an example and following guidelines in [RFC4514], it should have instead appeared as:

Issuer: OU=Sipit Test Certificate Authority,O=sipit,L=San Jose, ST=California,C=US

The ASN.1 parse of the CA certificate is shown below.

```
  0:l= 949 cons: SEQUENCE
  4:l= 669 cons:  SEQUENCE
  8:l=   3 cons:   cont [ 0 ]
 10:l=   1 prim:    INTEGER           :02
 13:l=   9 prim:   INTEGER           :96A384174EEF8A4C
 24:l=  13 cons:   SEQUENCE
 26:l=   9 prim:    OBJECT            :sha1WithRSAEncryption
```

```
 37:l=   0 prim:    NULL
 39:l= 112 cons:   SEQUENCE
 41:l=  11 cons:    SET
 43:l=   9 cons:     SEQUENCE
 45:l=   3 prim:      OBJECT            :countryName
 50:l=   2 prim:      PRINTABLESTRING   :US
 54:l=  19 cons:    SET
 56:l=  17 cons:     SEQUENCE
 58:l=   3 prim:      OBJECT            :stateOrProvinceName
 63:l=  10 prim:      UTF8STRING
  43 61 6c 69 66 6f 72 6e-69 61                      California
 75:l=  17 cons:    SET
 77:l=  15 cons:     SEQUENCE
 79:l=   3 prim:      OBJECT            :localityName
 84:l=   8 prim:      UTF8STRING
  53 61 6e 20 4a 6f 73 65-                           San Jose
 94:l=  14 cons:    SET
 96:l=  12 cons:     SEQUENCE
 98:l=   3 prim:      OBJECT            :organizationName
103:l=   5 prim:      UTF8STRING
  73 69 70 69 74                                     sipit
110:l=  41 cons:    SET
112:l=  39 cons:     SEQUENCE
114:l=   3 prim:      OBJECT            :organizationalUnitName
119:l=  32 prim:      UTF8STRING
  53 69 70 69 74 20 54 65-73 74 20 43 65 72 74 69   Sipit Test Certi
  66 69 63 61 74 65 20 41-75 74 68 6f 72 69 74 79   ficate Authority
153:l=  32 cons:   SEQUENCE
155:l=  13 prim:    UTCTIME           :110127183605Z
170:l=  15 prim:    GENERALIZEDTIME   :21110103183605Z
187:l= 112 cons:   SEQUENCE
189:l=  11 cons:    SET
191:l=   9 cons:     SEQUENCE
193:l=   3 prim:      OBJECT            :countryName
198:l=   2 prim:      PRINTABLESTRING   :US
202:l=  19 cons:    SET
204:l=  17 cons:     SEQUENCE
206:l=   3 prim:      OBJECT            :stateOrProvinceName
211:l=  10 prim:      UTF8STRING
  43 61 6c 69 66 6f 72 6e-69 61                      California
223:l=  17 cons:    SET
225:l=  15 cons:     SEQUENCE
227:l=   3 prim:      OBJECT            :localityName
232:l=   8 prim:      UTF8STRING
  53 61 6e 20 4a 6f 73 65-                           San Jose
242:l=  14 cons:    SET
244:l=  12 cons:     SEQUENCE
246:l=   3 prim:      OBJECT            :organizationName
```

```
251:l=   5 prim:      UTF8STRING
  73 69 70 69 74                                     sipit
258:l=  41 cons:    SET
260:l=  39 cons:     SEQUENCE
262:l=   3 prim:      OBJECT            :organizationalUnitName
267:l=  32 prim:      UTF8STRING
  53 69 70 69 74 20 54 65-73 74 20 43 65 72 74 69   Sipit Test Certi
  66 69 63 61 74 65 20 41-75 74 68 6f 72 69 74 79   ficate Authority
301:l= 290 cons:   SEQUENCE
305:l=  13 cons:    SEQUENCE
307:l=   9 prim:     OBJECT            :rsaEncryption
318:l=   0 prim:     NULL
320:l= 271 prim:    BIT STRING
  00 30 82 01 0a 02 82 01-01 00 ab 1f 91 61 f1 1c   .0...........a..
  c5 cd a6 7b 16 9b b7 14-79 e4 30 9e 98 d0 ec 07   ...{....y.0.....
  b7 bd 77 d7 d1 f5 5b 2c-e2 ee e6 b1 b0 f0 85 fa   ..w...[,........
  a5 bc cb cc cf 69 2c 4f-fc 50 ef 9d 31 2b c0 59   .....i,O.P..1+.Y
  ea fb 64 6f 1f 55 a7 3d-fd 70 d2 56 db 14 99 17   ..do.U.=.p.V....
  92 70 ac 26 f8 34 41 70-d9 c0 03 91 6a ba d1 11   .p.&.4Ap....j...
  8f ac 12 31 de b9 19 70-8d 5d a7 7d 8b 19 cc 40   ...1...p.].}...@
  3f ae ff de 1f db 94 b3-46 77 6c ae ae ff 3e d6   ?.......Fwl...>.
  84 5b c2 de 0b 26 65 d0-91 c7 70 4b c7 0a 4a bf   .[...&e...pK..J.
  c7 97 04 dd ba 58 47 cb-e0 2b 23 76 87 65 c5 55   .....XG..+#v.e.U
  34 10 ab 27 1f 1c f8 30-3d b0 9b ca a2 81 72 4c   4..'...0=.....rL
  bd 60 fe f7 21 fe 0b db-0b db e9 5b 01 36 d4 28   .`..!......[.6.(
  15 6b 79 eb d0 91 1b 21-59 b8 0e aa bf d5 b1 6c   .ky....!Y......l
  70 37 a3 3f a5 7d 0e 95-46 f6 f6 58 67 83 75 42   p7.?.}..F..Xg.uB
  37 18 0b a4 41 39 b2 2f-6c 80 2c 78 ec a5 0f be   7...A9./l.,x....
  9c 10 f8 c0 0b 0d 73 99-9e 0d d7 97 50 cb cc 45   ......s.....P..E
  34 23 49 41 85 22 24 ad-29 c3 02 03 01 00 01      4#IA."$.)......
595:l=  80 cons:   cont [ 3 ]
597:l=  78 cons:    SEQUENCE
599:l=  29 cons:     SEQUENCE
601:l=   3 prim:      OBJECT            :X509v3 Subject Key Identifier
606:l=  22 prim:      OCTET STRING
  04 14 95 45 7e 5f 2b ea-65 98 12 91 04 f3 63 c7   ...E~_+.e.....c.
  68 9a 58 16 77 27                                 h.X.w'
630:l=  31 cons:     SEQUENCE
632:l=   3 prim:      OBJECT            :X509v3 Authority Key Identifier
637:l=  24 prim:      OCTET STRING
  30 16 80 14 95 45 7e 5f-2b ea 65 98 12 91 04 f3   0....E~_+.e.....
  63 c7 68 9a 58 16 77 27-                          c.h.X.w'
663:l=  12 cons:     SEQUENCE
665:l=   3 prim:      OBJECT            :X509v3 Basic Constraints
670:l=   5 prim:      OCTET STRING
  30 03 01 01 ff                                    0....
677:l=  13 cons:  SEQUENCE
679:l=   9 prim:   OBJECT            :sha1WithRSAEncryption
```

```
690:l=   0 prim:   NULL
692:l= 257 prim:  BIT STRING
  00 06 5f 9e ae a0 9a bc-b5 b9 5b 7e 97 33 cc df   .._.......[~.3..
  63 98 98 94 cb 0d 66 a9-83 e8 aa 58 2a 59 a1 9e   c.....f....X*Y..
  47 31 a6 af 5c 3f a2 25-86 f8 df 05 92 b7 db 69   G1..\?.%.......i
  a1 69 72 87 66 c5 ab 35-89 01 37 19 c9 74 eb 09   .ir.f..5..7..t..
  d1 3f 88 7b 24 13 42 ca-2d fb 45 e6 cc 4b f8 21   .?.{$.B.-.E..K.!
  78 f3 f5 97 ec 09 92 24-a2 f0 e6 94 8d 97 4a 00   x.......$......J.
  94 00 bd 25 b8 17 2c 52-53 5d cc 5c 48 a4 a1 1d   ...%..,RS].\H...
  2d f6 50 55 13 a4 d3 b2-a2 f4 f1 b9 6d 48 5e 5c   -.PU........mH^\
  f3 de e0 fc 59 09 a1 d9-14 61 65 bf d8 3f b9 ba   ....Y....ae..?..
  2e 7c ed 5c 24 9b 6b ca-aa 5f f1 c1 1e b0 a8 da   .|.\$.k.._......
  82 0f fb 4c 71 3b 4d 7b-38 c8 e3 8a 2a 19 34 44   ...Lq;M{8...*.4D
  26 0b ea f0 47 38 46 28-65 04 e2 01 52 dd ec 3d   &...G8F(e...R..=
  e5 f5 53 74 77 74 75 6d-c6 d9 c2 0a ac 3b b8 98   ..Stwtum.....;..
  5c 55 53 34 74 52 a8 26-b1 2f 30 22 d0 8b b7 f3   \US4tR.&./0"....
  a0 dd 68 07 33 d5 ae b7-81 b2 94 58 72 4e 7c c6   ..h.3......XrN|.
  72 2f bd 6c 69 fb b5 17-a8 2a 8d d7 2c 91 06 c8   r/.li....*..,...
  0c                                                .
```

2.2. Host Certificates

The certificate for the host example.com is shown below. Note that the Subject Alternative Name is set to example.com and is a DNS type. The certificates for the other hosts are shown in Appendix B.

```
Version: 3 (0x2)
Serial Number:
    96:a3:84:17:4e:ef:8a:4f
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit,
        OU=Sipit Test Certificate Authority
Validity
    Not Before: Feb  7 19:32:17 2011 GMT
    Not After : Jan 14 19:32:17 2111 GMT
Subject: C=US, ST=California, L=San Jose, O=sipit, CN=example.com
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (2048 bit)
        Modulus (2048 bit):
            00:dd:74:06:02:10:c2:e7:04:1f:bc:8c:b6:24:e7:
            9b:94:a3:48:37:85:9e:6d:83:12:84:50:1a:8e:48:
            b1:fa:86:8c:a7:80:b9:be:52:ec:a6:ca:63:47:84:
            ad:f6:74:85:82:16:7e:4e:36:40:0a:74:2c:20:a9:
            6a:0e:6a:7f:35:cf:70:71:63:7d:e9:43:67:81:4c:
            ea:b5:1e:b7:4c:a3:35:08:7b:21:0d:2a:73:07:63:
            9d:8d:75:bf:1f:d4:8e:e6:67:60:75:f7:ea:0a:7a:
```

```
                6c:90:af:92:45:e0:62:05:9a:8a:10:98:dc:7c:54:
                8b:e4:61:95:3b:04:fc:10:50:ef:80:45:ba:5e:84:
                97:76:c1:20:25:c1:92:1d:89:0a:f7:55:62:64:fa:
                e8:69:a2:62:4c:67:d3:08:d9:61:b5:3d:16:54:b6:
                b7:44:8d:59:2b:90:d4:e9:fb:c7:7d:87:58:c3:12:
                ac:33:78:00:50:ba:07:05:b3:b9:01:1a:63:55:6c:
                e1:7a:ec:a3:07:ae:3b:02:83:a1:69:e0:c3:dc:2d:
                61:e9:b2:e3:b3:71:c8:a6:cf:da:fb:3e:99:c7:e5:
                71:b9:c9:17:d4:ed:bc:a0:47:54:09:8c:6e:6d:53:
                9a:2c:c9:68:c6:6f:f1:3d:91:1a:24:43:77:7d:91:
                69:4b
            Exponent: 65537 (0x10001)
    X509v3 extensions:
        X509v3 Subject Alternative Name:
            DNS:example.com, URI:sip:example.com
        X509v3 Basic Constraints:
            CA:FALSE
        X509v3 Subject Key Identifier:
            CC:06:59:5B:8B:5E:D6:0D:F2:05:4D:1B:68:54:1E:FC:F9:43:19:17
        X509v3 Authority Key Identifier:
            95:45:7E:5F:2B:EA:65:98:12:91:04:F3:63:C7:68:9A:58:16:77:27

        X509v3 Key Usage:
            Digital Signature, Non Repudiation, Key Encipherment
        X509v3 Extended Key Usage:
            TLS Web Server Authentication, 1.3.6.1.5.5.7.3.20
        Signature Algorithm: sha1WithRSAEncryption
    6a:9a:d1:db:00:4b:90:86:b0:53:ea:6f:30:31:89:1e:9b:09:
    14:bd:6f:b9:02:aa:6f:58:ee:30:03:b8:a1:fd:b3:41:72:ff:
    b3:0d:cb:76:a7:17:c6:57:38:06:13:e5:f3:e4:30:17:4d:f7:
    97:b5:f3:74:e9:81:f8:f4:55:a3:0d:f5:82:38:c3:98:43:52:
    1f:84:cd:1a:b4:a3:45:9f:3d:e2:31:fd:cb:a2:ad:ed:60:7d:
    fa:d2:aa:49:2f:41:a9:80:01:bb:ed:b6:75:c9:97:69:7f:0c:
    91:60:f1:c4:5a:36:e8:5c:ac:e1:a8:e7:9a:55:e5:e0:cd:01:
    f4:de:93:f4:38:6c:c1:71:d2:fd:cd:1b:5d:25:eb:90:7b:31:
    41:e7:37:0e:e5:c0:01:48:91:f7:34:dd:c6:1f:74:e6:34:34:
    e6:cd:93:0f:3f:ce:94:ad:91:d9:e2:72:b1:9f:1d:d3:a5:7d:
    5e:e2:a4:56:c5:b1:71:4d:10:0a:5d:a6:56:e6:57:1f:48:a5:
    5c:75:67:ea:ab:35:3e:f6:b6:fa:c1:f3:8a:c1:80:71:32:18:
    6c:33:b5:fa:16:5a:16:e1:a1:6c:19:67:f5:45:68:64:6f:b2:
    31:dc:e3:5a:1a:b2:d4:87:89:96:fd:87:ba:38:4e:0a:19:07:
    03:4b:9b:b1
```

The example host certificate above, as well as all the others presented in this document, are signed directly by a root CA. These certificate chains have a length equal to two: the root CA and the host certificate. Non-root CAs exist and may also sign certificates.

The certificate chains presented by hosts with certificates signed by non-root CAs will have a length greater than two. For more details on how certificate chains are validated, see Sections 6.1 and 6.2 of [RFC5280].

2.3. User Certificates

User certificates are used by many applications to establish user identity. The user certificate for fluffy@example.com is shown below. Note that the Subject Alternative Name has a list of names with different URL types such as a sip, im, or pres URL. This is necessary for interoperating with a Common Profile for Instant Messaging (CPIM) gateway. In this example, example.com is the domain for fluffy. The message could be coming from any host in *.example.com, and the AOR in the user certificate would still be the same. The others are shown in Appendix B.1. These certificates make use of the Extended Key Usage (EKU) extension discussed in [RFC5924]. Note that the X509v3 Extended Key Usage attribute refers to the SIP OID introduced in [RFC5924], which is 1.3.6.1.5.5.7.3.20

```
Version: 3 (0x2)
Serial Number:
    96:a3:84:17:4e:ef:8a:4d
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit,
        OU=Sipit Test Certificate Authority
Validity
    Not Before: Feb  7 19:32:17 2011 GMT
    Not After : Jan 14 19:32:17 2111 GMT
Subject: C=US, ST=California, L=San Jose, O=sipit,
         CN=fluffy
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (2048 bit)
        Modulus (2048 bit):
            00:a3:2c:59:0c:e9:bc:e4:ec:d3:9e:fb:99:02:ec:
            b1:36:3a:b7:d3:1d:4d:c3:3a:b6:ae:50:bd:5f:55:
            08:77:8c:7e:a4:e9:f0:68:31:28:8f:23:32:56:19:
            c3:22:97:a7:6d:fd:a7:22:2a:01:b5:af:61:bd:5f:
            7e:c1:14:e5:98:29:b4:34:4e:38:8a:26:ee:0d:da:
            db:27:b9:78:d6:ac:ac:04:78:32:98:c2:75:e7:6a:
            b7:2d:b3:3c:e3:eb:97:a5:ef:8b:59:42:50:17:7b:
            fe:a7:81:af:37:a7:e7:e3:1f:b0:8d:d0:72:2f:6c:
            14:42:c6:01:68:e1:8f:fd:56:4d:7d:cf:16:dc:aa:
            05:61:0b:0a:ca:ca:ec:51:ec:53:6e:3d:2b:00:80:
            fe:35:1b:06:0a:61:13:88:0b:44:f3:cc:fd:2b:0e:
            b4:a2:0b:a0:97:84:14:2e:ee:2b:e3:2f:c1:1a:9e:
            86:9a:78:6a:a2:4c:57:93:e7:01:26:d3:56:0d:bd:
```

```
            b0:2f:f8:da:c7:3c:01:dc:cb:2d:31:8c:6c:c6:5c:
            b4:63:e8:b2:a2:40:11:bf:ad:f8:6d:12:01:97:1d:
            47:f8:6a:15:8b:fb:27:96:73:44:46:34:d7:24:1c:
            cf:56:8d:d4:be:d6:94:5b:f0:a6:67:e3:dd:cf:b4:
            f2:d5
        Exponent: 65537 (0x10001)
X509v3 extensions:
    X509v3 Subject Alternative Name:
        URI:sip:fluffy@example.com, URI:im:fluffy@example.com,
           URI:pres:fluffy@example.com
    X509v3 Basic Constraints:
        CA:FALSE
    X509v3 Subject Key Identifier:
        85:97:09:B8:D3:55:37:24:8A:DC:DE:E3:91:72:E4:22:CF:98:87:52
    X509v3 Authority Key Identifier:
        95:45:7E:5F:2B:EA:65:98:12:91:04:F3:63:C7:68:9A:58:16:77:27

    X509v3 Key Usage:
        Digital Signature, Non Repudiation, Key Encipherment
    X509v3 Extended Key Usage:
        E-mail Protection, 1.3.6.1.5.5.7.3.20
    Signature Algorithm: sha1WithRSAEncryption
a8:a9:8f:d8:8a:0b:88:ed:ff:4f:bf:e5:cd:8f:9e:7b:b8:e6:
f2:2c:aa:e3:23:5b:9a:71:5e:fd:20:a3:dd:d9:d3:c1:f2:e8:
f0:be:77:db:33:cc:8a:7b:4f:91:2b:8d:d6:f7:14:c3:8d:e0:
60:d3:34:50:bc:be:67:22:cd:f5:74:7b:f4:9a:68:a2:52:2b:
81:2f:46:d3:09:9f:25:c3:20:e8:10:d5:ef:38:7b:d1:17:d4:
f1:d7:54:67:56:f1:13:cf:2f:fc:8b:83:fc:14:e7:01:82:59:
83:cc:b1:8d:f0:c7:da:4e:b1:dc:cc:54:cf:6c:3b:47:47:59:
87:d9:16:ec:af:af:e1:12:13:23:1e:0a:db:f5:b5:ff:5d:ab:
15:0e:e3:25:91:00:0e:90:db:d8:07:11:90:81:01:3a:48:a8:
aa:9e:b0:62:d3:36:f0:0c:b7:2f:a7:17:92:52:36:29:14:0a:
d6:65:86:67:73:74:6e:aa:3c:ee:47:38:1e:c8:6e:06:81:85:
1c:2e:f0:b6:04:7d:6c:38:db:81:9c:b8:07:e3:07:be:f5:2f:
09:68:63:04:6b:87:0e:36:b9:a1:a3:fb:c8:30:0c:a0:63:8d:
6d:ab:0a:f8:44:b0:78:19:1a:38:7e:fa:6a:a1:d4:4b:4b:75:
75:bf:6f:09
```

Versions of these certificates that do not make use of EKU are also included in Appendix B.2

3. Callflow with Message Over TLS

3.1. TLS with Server Authentication

The flow below shows the edited SSLDump output of the host example.com forming a TLS [RFC5246] connection to example.net. In this example mutual authentication is not used. Note that the client proposed three protocol suites including TLS_RSA_WITH_AES_128_CBC_SHA defined in [RFC5246]. The certificate returned by the server contains a Subject Alternative Name that is set to example.net. A detailed discussion of TLS can be found in SSL and TLS [EKR-TLS]. For more details on the SSLDump tool, see the SSLDump Manual [ssldump-manpage].

This example does not use the Server Extended Hello (see [RFC5246]).

```
New TCP connection #1: example.com(50738) <-> example.net(5061)
1 1  0.0004 (0.0004)  C>SV3.1(101)  Handshake
      ClientHello
        Version 3.1
        random[32]=
          4c 09 5b a7 66 77 eb 43 52 30 dd 98 4d 09 23 d3
          ff 81 74 ab 04 69 bb 79 8c dc 59 cd c2 1f b7 ec
        cipher suites
        TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
        TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
        TLS_DHE_RSA_WITH_AES_256_SHA
        TLS_RSA_WITH_AES_256_CBC_SHA
        TLS_DSS_RSA_WITH_AES_256_SHA
        TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
        TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
        TLS_DHE_RSA_WITH_AES_128_CBC_SHA
        TLS_RSA_WITH_AES_128_CBC_SHA
        TLS_DHE_DSS_WITH_AES_128_CBC_SHA
        TLS_ECDHE_RSA_WITH_DES_192_CBC3_SHA
        TLS_ECDH_RSA_WITH_DES_192_CBC3_SHA
        TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
        TLS_RSA_WITH_3DES_EDE_CBC_SHA
        TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
        TLS_ECDHE_RSA_WITH_RC4_128_SHA
        TLS_ECDH_RSA_WITH_RC4_128_SHA
        TLS_RSA_WITH_RC4_128_SHA
        TLS_RSA_WITH_RC4_128_MD5
        TLS_DHE_RSA_WITH_DES_CBC_SHA
        TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
        TLS_RSA_WITH_DES_CBC_SHA
        TLS_RSA_EXPORT_WITH_DES40_CBC_SHA
        TLS_DHE_DSS_WITH_DES_CBC_SHA
```

```
          TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
          TLS_RSA_EXPORT_WITH_RC4_40_MD5
          compression methods
                    NULL
   1 2  0.0012 (0.0007)  S>CV3.1(48)  Handshake
         ServerHello
          Version 3.1
          random[32]=
            4c 09 5b a7 30 87 74 c7 16 98 24 d5 af 35 17 a7
            ef c3 78 0c 94 d4 94 d2 7b a6 3f 40 04 25 f6 e0
          session_id[0]=

          cipherSuite         TLS_RSA_WITH_AES_256_CBC_SHA
          compressionMethod                   NULL
   1 3  0.0012 (0.0000)  S>CV3.1(1858)  Handshake
         Certificate
   1 4  0.0012 (0.0000)  S>CV3.1(14)  Handshake
         CertificateRequest
          certificate_types                   rsa_sign
          certificate_types                   dss_sign
          certificate_types                unknown value
         ServerHelloDone
   1 5  0.0043 (0.0031)  C>SV3.1(7)  Handshake
         Certificate
   1 6  0.0043 (0.0000)  C>SV3.1(262)  Handshake
         ClientKeyExchange
   1 7  0.0043 (0.0000)  C>SV3.1(1)  ChangeCipherSpec
   1 8  0.0043 (0.0000)  C>SV3.1(48)  Handshake
   1 9  0.0129 (0.0085)  S>CV3.1(170)  Handshake
   1 10 0.0129 (0.0000)  S>CV3.1(1)  ChangeCipherSpec
   1 11 0.0129 (0.0000)  S>CV3.1(48)  Handshake
   1 12 0.0134 (0.0005)  C>SV3.1(32)  application_data
   1 13 0.0134 (0.0000)  C>SV3.1(496)  application_data
   1 14 0.2150 (0.2016)  S>CV3.1(32)  application_data
   1 15 0.2150 (0.0000)  S>CV3.1(336)  application_data
   1 16 12.2304 (12.0154)  S>CV3.1(32)  Alert
   1    12.2310 (0.0005)  S>C  TCP FIN
   1 17 12.2321 (0.0011)  C>SV3.1(32)  Alert
```

3.2. MESSAGE Transaction Over TLS

Once the TLS session is set up, the following MESSAGE request (as defined in [RFC3428] is sent from fluffy@example.com to kumiko@example.net. Note that the URI has a SIPS URL and that the VIA indicates that TLS was used. In order to format this document, the <allOneLine> convention from [RFC4475] is used to break long lines. The actual message does not contain the line breaks contained within those tags.

```
MESSAGE sips:kumiko@example.net:5061 SIP/2.0
<allOneLine>
Via: SIP/2.0/TLS 192.0.2.2:15001;
     branch=z9hG4bK-d8754z-c785a077a9a8451b-1---d8754z-;
     rport=50738
</allOneLine>
Max-Forwards: 70
To: <sips:kumiko@example.net:5061>
From: <sips:fluffy@example.com:15001>;tag=1a93430b
Call-ID: OTZmMDE2OWNlYTVjNDkzYzBhMWRlMDU4NDExZmU4ZTQ.
CSeq: 4308 MESSAGE
<allOneLine>
Accept: multipart/signed, text/plain, application/pkcs7-mime,
        application/sdp, multipart/alternative
</allOneLine>
Content-Type: text/plain
Content-Length: 6

Hello!
```

When a User Agent (UA) goes to send a message to example.com, the UA can see if it already has a TLS connection to example.com and if it does, it may send the message over this connection. A UA should have some scheme for reusing connections as opening a new TLS connection for every message results in awful performance. Implementers are encouraged to read [RFC5923] and [RFC3263].

The response is sent from example.net to example.com over the same TLS connection. It is shown below.

```
SIP/2.0 200 OK
<allOneLine>
Via: SIP/2.0/TLS 192.0.2.2:15001;
     branch=z9hG4bK-d8754z-c785a077a9a8451b-1---d8754z-;
     rport=50738
</allOneLine>
To: <sips:kumiko@example.net:5061>;tag=0d075510
From: <sips:fluffy@example.com:15001>;tag=1a93430b
Call-ID: OTZmMDE2OWNlYTVjNDkzYzBhMWRlMDU4NDExZmU4ZTQ.
CSeq: 4308 MESSAGE
Content-Length: 0
```

4. Callflow with S/MIME-secured Message

4.1. MESSAGE Request with Signed Body

Below is an example of a signed message. The values on the Content-Type line (multipart/signed) and on the Content-Disposition line have been broken across lines to fit on the page, but they are not broken across lines in actual implementations.

```
MESSAGE sip:kumiko@example.net SIP/2.0
<allOneLine>
Via: SIP/2.0/TCP 192.0.2.2:15001;
     branch=z9hG4bK-d8754z-3a922b6dc0f0ff37-1---d8754z-;
     rport=50739
</allOneLine>
Max-Forwards: 70
To: <sip:kumiko@example.net>
From: <sip:fluffy@example.com>;tag=ef6bad5e
Call-ID: N2NiZjI0NjRjNDQ0MTY1NDRjNWNmMGU1MDA2MDRhYmI.
CSeq: 8473 MESSAGE
<allOneLine>
Accept: multipart/signed, text/plain, application/pkcs7-mime,
        application/sdp, multipart/alternative
</allOneLine>
<allOneLine>
Content-Type: multipart/signed;boundary=3b515e121b43a911;
              micalg=sha1;protocol="application/pkcs7-signature"
</allOneLine>
Content-Length: 774

--3b515e121b43a911
Content-Type: text/plain
Content-Transfer-Encoding: binary

Hello!
--3b515e121b43a911
Content-Type: application/pkcs7-signature;name=smime.p7s
<allOneLine>
Content-Disposition: attachment;handling=required;
                     filename=smime.p7s
</allOneLine>
Content-Transfer-Encoding: binary

**********
* BINARY BLOB 1 *
**********
--3b515e121b43a911--
```

It is important to note that the signature ("BINARY BLOB 1") is computed over the MIME headers and body, but excludes the multipart boundary lines. The value on the Message-body line ends with CRLF. The CRLF is included in the boundary and is not part of the signature computation. To be clear, the signature is computed over data starting with the "C" in the "Content-Type" and ending with the "!" in the "Hello!".

```
Content-Type: text/plain
Content-Transfer-Encoding: binary

Hello!
```

Following is the ASN.1 parsing of encrypted contents referred to above as "BINARY BLOB 1". Note that at address 30, the hash for the signature is specified as SHA-1. Also note that the sender's certificate is not attached as it is optional in [RFC5652].

```
 0  472: SEQUENCE {
 4    9:   OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
15  457:   [0] {
19  453:     SEQUENCE {
23    1:       INTEGER 1
26   11:       SET {
28    9:         SEQUENCE {
30    5:           OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
37    0:           NULL
       :           }
       :         }
39   11:       SEQUENCE {
41    9:         OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
       :         }
52  420:       SET {
56  416:         SEQUENCE {
60    1:           INTEGER 1
63  125:           SEQUENCE {
65  112:             SEQUENCE {
67   11:               SET {
69    9:                 SEQUENCE {
71    3:                   OBJECT IDENTIFIER countryName (2 5 4 6)
76    2:                   PrintableString 'US'
       :                   }
       :                 }
80   19:               SET {
82   17:                 SEQUENCE {
84    3:                   OBJECT IDENTIFIER
       :                     stateOrProvinceName (2 5 4 8)
89   10:                   UTF8String 'California'
```

```
      :                   }
      :                 }
  101   17:               SET {
  103   15:                 SEQUENCE {
  105    3:                   OBJECT IDENTIFIER localityName (2 5 4 7)
  110    8:                   UTF8String 'San Jose'
      :                   }
      :                 }
  120   14:               SET {
  122   12:                 SEQUENCE {
  124    3:                   OBJECT IDENTIFIER
      :                     organizationName (2 5 4 10)
  129    5:                   UTF8String 'sipit'
      :                   }
      :                 }
  136   41:               SET {
  138   39:                 SEQUENCE {
  140    3:                   OBJECT IDENTIFIER
      :                     organizationalUnitName (2 5 4 11)
  145   32:                   UTF8String 'Sipit Test Certificate
                                        Authority'
      :                   }
      :                 }
      :               }
  179    9:             INTEGER 00 96 A3 84 17 4E EF 8A 4D
      :             }
  190    9:           SEQUENCE {
  192    5:             OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
  199    0:             NULL
      :             }
  201   13:           SEQUENCE {
  203    9:             OBJECT IDENTIFIER
      :               rsaEncryption (1 2 840 113549 1 1 1)
  214    0:             NULL
      :             }
  216  256:           OCTET STRING
      :             74 4D 21 39 D6 E2 E2 2C 30 5A AA BC 4E 60 8D 69
      :             A7 E5 79 50 1A B1 7D 4A D3 C1 03 9F 19 7D A2 76
      :             97 B3 CE 30 CD 62 4B 96 20 35 DB C1 64 D9 33 92
      :             96 CD 28 03 98 6E 2C 0C F6 8D 93 40 F2 88 DA 29
      :             AD 0B C2 0E F9 D3 6A 95 2C 79 6E C2 3D 62 E6 54
      :             A9 1B AC 66 DB 16 B7 44 6C 03 1B 71 9C EE C9 EC
      :             4D 93 B1 CF F5 17 79 C5 C8 BA 2F A7 6C 4B DC CF
      :             62 A3 F3 1A 1B 24 E4 40 66 3C 4F 87 86 BF 09 6A
      :             7A 43 60 2B FC D8 3D 2B 57 17 CB 81 03 2A 56 69
      :             81 82 FA 78 DE D2 3A 2F FA A3 C5 EA 8B E8 0C 36
      :             1B BC DC FD 1B 8C 2E 0F 01 AF D9 E1 04 0E 4E 50
      :             94 75 7C BD D9 0B DD AA FA 36 E3 EC E4 A5 35 46
```

```
         :             BE A2 97 1D AD BA 44 54 3A ED 94 DA 76 4A 51 BA
         :             A4 7D 7A 62 BF 2A 2F F2 5C 5A FE CA E6 B9 DC 5D
         :             EA 26 F2 35 17 19 20 CE 97 96 4E 72 9C 72 FD 1F
         :             68 C1 6A 5C 86 42 F2 ED F2 70 65 4C C7 44 C5 7C
         :           }
         :         }
         :       }
         :     }
         :   }
```

SHA-1 parameters may be omitted entirely, instead of being set to NULL, as mentioned in [RFC3370]. The above dump of Blob 1 has SHA-1 parameters set to NULL. Below are the same contents signed with the same key, but omitting the NULL according to [RFC3370]. This is the preferred encoding. This is covered in greater detail in Section 5.

```
 0  468: SEQUENCE {
 4    9:   OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
15  453:   [0] {
19  449:     SEQUENCE {
23    1:       INTEGER 1
26    9:       SET {
28    7:         SEQUENCE {
30    5:           OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
       :           }
       :         }
37   11:       SEQUENCE {
39    9:         OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
       :         }
50  418:       SET {
54  414:         SEQUENCE {
58    1:           INTEGER 1
61  125:           SEQUENCE {
63  112:             SEQUENCE {
65   11:               SET {
67    9:                 SEQUENCE {
69    3:                   OBJECT IDENTIFIER countryName (2 5 4 6)
74    2:                   PrintableString 'US'
       :                   }
       :                 }
78   19:               SET {
80   17:                 SEQUENCE {
82    3:                   OBJECT IDENTIFIER
       :                     stateOrProvinceName (2 5 4 8)
87   10:                   UTF8String 'California'
       :                   }
       :                 }
```

```
   99   17:              SET {
    101   15:                SEQUENCE {
    103    3:                  OBJECT IDENTIFIER localityName (2 5 4 7)
    108    8:                  UTF8String 'San Jose'
          :                  }
          :                }
    118   14:              SET {
    120   12:                SEQUENCE {
    122    3:                  OBJECT IDENTIFIER
          :                    organizationName (2 5 4 10)
    127    5:                  UTF8String 'sipit'
          :                  }
          :                }
    134   41:              SET {
    136   39:                SEQUENCE {
    138    3:                  OBJECT IDENTIFIER
          :                    organizationalUnitName (2 5 4 11)
    143   32:                  UTF8String 'Sipit Test Certificate
                                        Authority'
          :                  }
          :                }
          :              }
    177    9:            INTEGER 00 96 A3 84 17 4E EF 8A 4D
          :            }
    188    7:          SEQUENCE {
    190    5:            OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
          :            }
    197   13:          SEQUENCE {
    199    9:            OBJECT IDENTIFIER
          :            rsaEncryption (1 2 840 113549 1 1 1)
    210    0:            NULL
          :            }
    212  256:          OCTET STRING
          :            74 4D 21 39 D6 E2 E2 2C 30 5A AA BC 4E 60 8D 69
          :            A7 E5 79 50 1A B1 7D 4A D3 C1 03 9F 19 7D A2 76
          :            97 B3 CE 30 CD 62 4B 96 20 35 DB C1 64 D9 33 92
          :            96 CD 28 03 98 6E 2C 0C F6 8D 93 40 F2 88 DA 29
          :            AD 0B C2 0E F9 D3 6A 95 2C 79 6E C2 3D 62 E6 54
          :            A9 1B AC 66 DB 16 B7 44 6C 03 1B 71 9C EE C9 EC
          :            4D 93 B1 CF F5 17 79 C5 C8 BA 2F A7 6C 4B DC CF
          :            62 A3 F3 1A 1B 24 E4 40 66 3C 4F 87 86 BF 09 6A
          :            7A 43 60 2B FC D8 3D 2B 57 17 CB 81 03 2A 56 69
          :            81 82 FA 78 DE D2 3A 2F FA A3 C5 EA 8B E8 0C 36
          :            1B BC DC FD 1B 8C 2E 0F 01 AF D9 E1 04 0E 4E 50
          :            94 75 7C BD D9 0B DD AA FA 36 E3 EC E4 A5 35 46
          :            BE A2 97 1D AD BA 44 54 3A ED 94 DA 76 4A 51 BA
          :            A4 7D 7A 62 BF 2A 2F F2 5C 5A FE CA E6 B9 DC 5D
          :            EA 26 F2 35 17 19 20 CE 97 96 4E 72 9C 72 FD 1F
```

```
          :             68 C1 6A 5C 86 42 F2 ED F2 70 65 4C C7 44 C5 7C
          :           }
          :         }
          :       }
          :     }
          :   }
```

4.2. MESSAGE Request with Encrypted Body

Below is an example of an encrypted text/plain message that says "Hello!". The binary encrypted contents have been replaced with the block "BINARY BLOB 2".

```
MESSAGE sip:kumiko@example.net SIP/2.0
<allOneLine>
Via: SIP/2.0/TCP 192.0.2.2:15001;
     branch=z9hG4bK-d8754z-c276232b541dd527-1---d8754z-;
     rport=50741
</allOneLine>
Max-Forwards: 70
To: <sip:kumiko@example.net>
From: <sip:fluffy@example.com>;tag=7a2e3025
Call-ID: MDYyMDhhODA3NWE2ZjEyYzAwOTZlMjExNWI2ZWQwZGM.
CSeq: 3260 MESSAGE
<allOneLine>
Accept: multipart/signed, text/plain, application/pkcs7-mime,
        application/sdp, multipart/alternative
</allOneLine>
<allOneLine>
Content-Disposition: attachment;handling=required;
                     filename=smime.p7
</allOneLine>
Content-Transfer-Encoding: binary
<allOneLine>
Content-Type: application/pkcs7-mime;smime-type=enveloped-data;
              name=smime.p7m
</allOneLine>
Content-Length: 565

*****************
* BINARY BLOB 2 *
*****************
```

Following is the ASN.1 parsing of "BINARY BLOB 2". Note that at address 454, the encryption is set to aes128-CBC.

```
 0  561: SEQUENCE {
```

```
 4    9:   OBJECT IDENTIFIER envelopedData (1 2 840 113549 1 7 3)
15  546:   [0] {
19  542:     SEQUENCE {
23    1:       INTEGER 0
26  409:       SET {
30  405:         SEQUENCE {
34    1:           INTEGER 0
37  125:           SEQUENCE {
39  112:             SEQUENCE {
41   11:               SET {
43    9:                 SEQUENCE {
45    3:                   OBJECT IDENTIFIER countryName (2 5 4 6)
50    2:                   PrintableString 'US'
       :                   }
       :                 }
54   19:               SET {
56   17:                 SEQUENCE {
58    3:                   OBJECT IDENTIFIER
       :                     stateOrProvinceName (2 5 4 8)
63   10:                   UTF8String 'California'
       :                   }
       :                 }
75   17:               SET {
77   15:                 SEQUENCE {
79    3:                   OBJECT IDENTIFIER localityName (2 5 4 7)
84    8:                   UTF8String 'San Jose'
       :                   }
       :                 }
94   14:               SET {
96   12:                 SEQUENCE {
98    3:                   OBJECT IDENTIFIER
       :                     organizationName (2 5 4 10)
 103    5:                   UTF8String 'sipit'
       :                   }
       :                 }
 110   41:               SET {
 112   39:                 SEQUENCE {
 114    3:                   OBJECT IDENTIFIER
       :                     organizationalUnitName (2 5 4 11)
 119   32:                   UTF8String 'Sipit Test Certificate
                                         Authority'
       :                   }
       :                 }
       :               }
 153    9:             INTEGER 00 96 A3 84 17 4E EF 8A 4E
       :             }
 164   13:           SEQUENCE {
 166    9:             OBJECT IDENTIFIER
```

```
      :               rsaEncryption (1 2 840 113549 1 1 1)
  177  0:             NULL
      :             }
  179 256:          OCTET STRING
      :             B9 12 8F 32 AB 4A E2 38 C1 E0 53 69 88 D6 25 E7
      :             40 03 B1 DE 79 21 A3 E8 23 5A 1B CB FB 58 F4 97
      :             48 A7 C8 F0 3D DF 41 A3 5A 90 32 70 82 FA B0 DE
      :             D8 94 7C 6C 2E 01 FE 33 BD 62 CB 07 4F 58 DE 6F
      :             EA 3F EF B4 FB 46 72 58 9A 88 A0 85 BC 23 D7 C8
      :             09 0B 90 8D 4A 5F 3F 96 7C AC D4 E2 19 E8 02 B6
      :             0E F3 0D F2 91 4A 67 A9 EE 51 6A 97 D7 86 6D EC
      :             78 6E C6 E0 83 7C E1 00 1F 5A 40 59 60 0C D7 EB
      :             A3 FB 04 B3 C9 A5 EB 79 ED B3 56 F8 F6 51 B2 5E
      :             58 E2 D8 17 28 33 A6 B8 35 8C 0E 14 7F 90 D0 7B
      :             03 00 6C 3D 81 29 F5 D7 E5 AC 75 5E E0 F0 DD E3
      :             3E B2 06 97 D6 49 A9 CB 38 08 F1 84 05 F5 C0 BC
      :             55 A6 D4 C9 D8 FD A4 AC 40 9F 9D 51 5B F7 3A C3
      :             C3 CD 3A E7 6D 21 05 D0 50 75 4F 14 D8 77 76 C6
      :             13 A6 48 12 7B 25 CC 22 5D 73 BD 40 E4 15 02 A2
      :             39 4A CB D9 55 08 A4 EE 4E 8A 5E BA C4 4A 46 9C
      :           }
      :         }
  439 124:      SEQUENCE {
  441   9:        OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
  452  29:        SEQUENCE {
  454   9:          OBJECT IDENTIFIER
      :             aes128-CBC (2 16 840 1 101 3 4 1 2)
  465  16:          OCTET STRING
      :             CA 35 CA BD 1E 78 83 D9 20 6C 47 B9 9F DC 91 88
      :           }
  483  80:        [0]
      :           1B AE 12 C4 0E 55 96 AB 99 CC 1C 7F B5 98 A4 BF
      :           D2 D8 7F 94 BB B5 38 05 59 F2 38 A1 CD 29 75 17
      :           1D 63 1B 0B B0 2D 88 06 7F 78 80 F3 5A 3E DC 35
      :           BF 22 1E 03 32 59 98 DA FD 81 5F D9 41 63 3A 18
      :           FD B5 84 14 01 46 0B 40 EB 56 29 86 47 8B D1 EE
      :         }
      :       }
      :     }
      :   }
```

4.3. MESSAGE Request with Encrypted and Signed Body

In the example below, some of the header values have been split across multiple lines. Where the lines have been broken, the <allOneLine> convention has been used. This was only done to make it fit in the RFC format. Specifically, the application/pkcs7-mime

Content-Type line is one line with no whitespace between the "mime;" and the "smime-type". The values are split across lines for formatting, but are not split in the real message. The binary encrypted content has been replaced with "BINARY BLOB 3", and the binary signed content has been replaced with "BINARY BLOB 4".

```
MESSAGE sip:kumiko@example.net SIP/2.0
<allOneLine>
Via: SIP/2.0/TCP 192.0.2.2:15001;
     branch=z9hG4bK-d8754z-97a26e59b7262b34-1---d8754z-;
     rport=50742
</allOneLine>
Max-Forwards: 70
To: <sip:kumiko@example.net>
From: <sip:fluffy@example.com>;tag=379f5b27
Call-ID: MjYwMzdjYTY3YWRkYzgzMjU0MGI4Mzc2Njk1YzJlNzE.
CSeq: 5449 MESSAGE
<allOneLine>
Accept: multipart/signed, text/plain, application/pkcs7-mime,
        application/sdp, multipart/alternative
</allOneLine>
<allOneLine>
Content-Type: multipart/signed;boundary=e8df6e1ce5d1e864;
              micalg=sha1;protocol="application/pkcs7-signature"
</allOneLine>
Content-Length: 1455

--e8df6e1ce5d1e864
<allOneLine>
Content-Type: application/pkcs7-mime;smime-type=enveloped-data;
              name=smime.p7m
</allOneLine>
<allOneLine>
Content-Disposition: attachment;handling=required;
                     filename=smime.p7
</allOneLine>
Content-Transfer-Encoding: binary

*****************
* BINARY BLOB 3 *
*****************
--e8df6e1ce5d1e864
Content-Type: application/pkcs7-signature;name=smime.p7s
<allOneLine>
Content-Disposition: attachment;handling=required;
                     filename=smime.p7s
</allOneLine>
Content-Transfer-Encoding: binary

*****************
* BINARY BLOB 4 *
*****************
--e8df6e1ce5d1e864--
```

Below is the ASN.1 parsing of "BINARY BLOB 3".

```
 0  561: SEQUENCE {
 4    9:   OBJECT IDENTIFIER envelopedData (1 2 840 113549 1 7 3)
15  546:   [0] {
19  542:     SEQUENCE {
23    1:       INTEGER 0
26  409:       SET {
30  405:         SEQUENCE {
34    1:           INTEGER 0
37  125:           SEQUENCE {
39  112:             SEQUENCE {
41   11:               SET {
43    9:                 SEQUENCE {
45    3:                   OBJECT IDENTIFIER countryName (2 5 4 6)
50    2:                   PrintableString 'US'
       :                   }
       :                 }
54   19:               SET {
56   17:                 SEQUENCE {
58    3:                   OBJECT IDENTIFIER
       :                     stateOrProvinceName (2 5 4 8)
63   10:                   UTF8String 'California'
       :                   }
       :                 }
75   17:               SET {
77   15:                 SEQUENCE {
79    3:                   OBJECT IDENTIFIER localityName (2 5 4 7)
84    8:                   UTF8String 'San Jose'
       :                   }
       :                 }
94   14:               SET {
96   12:                 SEQUENCE {
98    3:                   OBJECT IDENTIFIER
       :                     organizationName (2 5 4 10)
 103    5:                   UTF8String 'sipit'
       :                   }
       :                 }
 110   41:               SET {
 112   39:                 SEQUENCE {
 114    3:                   OBJECT IDENTIFIER
       :                     organizationalUnitName (2 5 4 11)
 119   32:                   UTF8String 'Sipit Test Certificate
                                         Authority'
       :                   }
       :                 }
       :               }
 153    9:             INTEGER 00 96 A3 84 17 4E EF 8A 4E
```

```
      :             }
  164  13:           SEQUENCE {
  166   9:             OBJECT IDENTIFIER
      :               rsaEncryption (1 2 840 113549 1 1 1)
  177   0:             NULL
      :             }
  179 256:           OCTET STRING
      :             49 11 0B 11 52 A9 9D E3 AA FB 86 CB EB 12 CC 8E
      :             96 9D 85 3E 80 D2 7C C4 9B B7 81 4B B5 FA 13 80
      :             6A 6A B2 34 72 D8 C0 82 60 DA B3 43 F8 51 8C 32
      :             8B DD D0 76 6D 9C 46 73 C1 44 A0 10 FF 16 A4 83
      :             74 85 21 74 7D E0 FD 42 C0 97 00 82 A2 80 81 22
      :             9C A2 82 0A 85 F0 68 EF 9A D7 6D 1D 24 2B A9 5E
      :             B3 9A A0 3E A7 D9 1D 1C D7 42 CB 6F A5 81 66 23
      :             28 00 7C 99 6A B6 03 3F 7E F6 48 EA 91 49 35 F1
      :             FD 40 54 5D AC F7 84 EA 3F 27 43 FD DE E2 10 DD
      :             63 C4 35 4A 13 63 0B 6D 0D 9A D5 AB 72 39 69 8C
      :             65 4C 44 C4 A3 31 60 79 B9 A8 A3 A1 03 FD 41 25
      :             12 E5 F3 F8 47 CE 8C 42 D9 26 77 A5 57 AF 1A 95
      :             BF 05 A5 E9 47 F2 D1 AE DC 13 7E 1B 83 5C 8C C4
      :             1F 31 BC 59 E6 FD 6E 9A B0 91 EC 71 A6 7F 28 3E
      :             23 1B 40 E2 C0 60 CF 5E 5B 86 08 06 82 B4 B7 DB
      :             00 DD AC 3A 39 27 E2 7C 96 AD 8A E9 C3 B8 06 5E
      :           }
      :         }
  439 124:      SEQUENCE {
  441   9:        OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
  452  29:        SEQUENCE {
  454   9:          OBJECT IDENTIFIER
      :           aes128-CBC (2 16 840 1 101 3 4 1 2)
  465  16:          OCTET STRING
      :             88 9B 13 75 A7 66 14 C3 CF CD C6 FF D2 91 5D A0
      :           }
  483  80:        [0]
      :           80 0B A3 B7 57 89 B4 F4 70 AE 1D 14 A9 35 DD F9
      :           1D 66 29 46 52 40 13 E1 3B 4A 23 E5 EC AB F9 35
      :           A6 B6 A4 BE C0 02 31 06 19 C4 39 22 7D 10 4C 0D
      :           F4 96 04 78 11 85 4E 7E E3 C3 BC B2 DF 55 17 79
      :           5F F2 4E E5 25 42 37 45 39 5D F6 DA 57 9A 4E 0B
      :         }
      :       }
      :     }
      :   }
```

Below is the ASN.1 parsing of "BINARY BLOB 4".

```
   0  472: SEQUENCE {
```

```
 4    9:   OBJECT IDENTIFIER signedData (1 2 840 113549 1 7 2)
15  457:   [0] {
19  453:     SEQUENCE {
23    1:       INTEGER 1
26   11:       SET {
28    9:         SEQUENCE {
30    5:           OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
37    0:           NULL
       :           }
       :         }
39   11:       SEQUENCE {
41    9:         OBJECT IDENTIFIER data (1 2 840 113549 1 7 1)
       :         }
52  420:       SET {
56  416:         SEQUENCE {
60    1:           INTEGER 1
63  125:           SEQUENCE {
65  112:             SEQUENCE {
67   11:               SET {
69    9:                 SEQUENCE {
71    3:                   OBJECT IDENTIFIER countryName (2 5 4 6)
76    2:                   PrintableString 'US'
       :                   }
       :                 }
80   19:               SET {
82   17:                 SEQUENCE {
84    3:                   OBJECT IDENTIFIER
       :                     stateOrProvinceName (2 5 4 8)
89   10:                   UTF8String 'California'
       :                   }
       :                 }
 101   17:               SET {
 103   15:                 SEQUENCE {
 105    3:                   OBJECT IDENTIFIER localityName (2 5 4 7)
 110    8:                   UTF8String 'San Jose'
       :                   }
       :                 }
 120   14:               SET {
 122   12:                 SEQUENCE {
 124    3:                   OBJECT IDENTIFIER
       :                     organizationName (2 5 4 10)
 129    5:                   UTF8String 'sipit'
       :                   }
       :                 }
 136   41:               SET {
 138   39:                 SEQUENCE {
 140    3:                   OBJECT IDENTIFIER
       :                     organizationalUnitName (2 5 4 11)
```

```
    145   32:                   UTF8String 'Sipit Test Certificate
                                          Authority'
          :                   }
          :                 }
          :               }
    179    9:             INTEGER 00 96 A3 84 17 4E EF 8A 4D
          :             }
    190    9:           SEQUENCE {
    192    5:             OBJECT IDENTIFIER sha1 (1 3 14 3 2 26)
    199    0:             NULL
          :             }
    201   13:           SEQUENCE {
    203    9:             OBJECT IDENTIFIER
          :               rsaEncryption (1 2 840 113549 1 1 1)
    214    0:             NULL
          :             }
    216  256:           OCTET STRING
          :             6E 51 AC 24 2E BA 7C A1 EE 80 A8 55 BC D4 64 5D
          :             E5 29 09 5F B2 AF AA 6F 91 D2 97 79 32 5B AF CA
          :             FE A1 73 FC E5 57 4E C6 3B 67 35 AA E4 78 1E 59
          :             93 EE 67 63 77 1E 7A 82 BC 1E 26 0F 39 75 0C A6
          :             26 92 01 6A B7 5D F0 C0 2C 51 46 FB A7 36 44 E3
          :             64 C6 11 CB 0B 6B FD F3 6D 7C FD 3E AE 2E 91 BB
          :             78 9E F4 1B A1 20 68 B9 DE D3 E3 0C FC F7 14 9A
          :             2C 64 AB 27 52 BD 52 EC 27 88 14 BD DB C3 54 C7
          :             EA 48 DB 07 E9 9B 2E C8 BE 62 A2 76 83 53 37 E8
          :             02 4B D1 86 E9 DF 2E BD 93 39 EC 2F 01 53 A0 7F
          :             1A B9 A6 31 FC E7 91 1C DB 22 4A 67 83 94 B2 4E
          :             28 A9 CD DE 4A 04 6A E0 86 90 7B 58 5F DB 7A 96
          :             96 A0 25 61 C2 58 A2 28 E5 B3 B2 F1 6D 51 06 9C
          :             78 61 0D D8 3A A7 9F A3 B5 87 0B 80 11 C2 A9 1A
          :             E5 17 1C EB 82 55 AB CD 04 E7 D9 5B 11 E8 B7 47
          :             FE FD CC B7 DB 47 6F 77 85 9E 24 D8 11 E1 E4 7D
          :           }
          :         }
          :       }
          :     }
          :   }
```

5. Observed Interoperability Issues

This section describes some common interoperability problems. These were observed by the authors at SIPit interoperability events. Implementers should be careful to verify that their systems do not introduce these common problems, and, when possible, make their clients forgiving in what they receive. Implementations should take extra care to produce reasonable error messages when interacting with software that has these problems.

Some SIP clients incorrectly only do SSLv3 and do not support TLS. See Section 26.2.1 of [RFC3261].

Many SIP clients were found to accept expired certificates with no warning or error. See Section 4.1.2.5 of [RFC5280].

When used with SIP, TLS and S/MIME provide the identity of the peer that a client is communicating with in the Subject Alternative Name in the certificate. The software checks that this name corresponds to the identity the server is trying to contact. Normative text describing path validation can be found in Section 7 of [RFC5922] and Section 6 of [RFC5280]. If a client is trying to set up a TLS connection to good.example.com and it gets a TLS connection set up with a server that presents a valid certificate but with the name evil.example.com, it will typically generate an error or warning of some type. Similarly with S/MIME, if a user is trying to communicate with sip:fluffy@example.com, one of the items in the Subject Alternate Name set in the certificate will need to match according to the certificate validation rules in Section 23 of [RFC3261] and Section 6 of [RFC5280].

Some implementations used binary MIME encodings while others used base64. It is advisable that implementations send only binary and are prepared to receive either. See Section 3.2 of [RFC5621].

In several places in this document, the messages contain the encoding for the SHA-1 digest algorithm identifier. The preferred form for encoding as set out in Section 2 of [RFC3370] is the form in which the optional AlgorithmIdentifier parameter field is omitted. However, [RFC3370] also says the recipients need to be able to receive the form in which the AlgorithmIdentifier parameter field is present and set to NULL. Examples of the form using NULL can be found in Section 4.2 of [RFC4134]. Receivers really do need to be able to receive the form that includes the NULL because the NULL form, while not preferred, is what was observed as being generated by most implementations. Implementers should also note that if the algorithm is MD5 instead of SHA-1, then the form that omits the AlgorithmIdentifier parameters field is not allowed and the sender

has to use the form where the NULL is included.

The preferred encryption algorithm for S/MIME in SIP is AES as defined in [RFC3853].

Observed S/MIME interoperability has been better when UAs did not attach the senders' certificates. Attaching the certificates significantly increases the size of the messages, which should be considered when sending over UDP. Furthermore, the receiver cannot rely on the sender to always send the certificate, so it does not turn out to be useful in most situations.

Please note that the certificate path validation algorithm described in Section 6 of [RFC5280] is a complex algorithm for which all of the details matter. There are numerous ways in which failing to precisely implement the algorithm as specified in Section 6 of [RFC5280] can create a security flaw, a simple example of which is the failure to check the expiration date that is already mentioned above. It is important for developers to ensure that this validation is performed and that the results are verified by their applications or any libraries that they use.

6. Additional Test Scenarios

This section provides a non-exhaustive list of tests that implementations should perform while developing systems that use S/MIME and TLS for SIP.

Much of the required behavior for inspecting certificates when using S/MIME and TLS with SIP is currently underspecified. The non-normative recommendations in this document capture the current folklore around that required behavior, guided by both related normative works such as [RFC4474] (particularly, Section 13.4 Domain Names and Subordination) and informative works such as [RFC2818] Section 3.1. To summarize, test plans should:

- For S/MIME secured bodies, assure that the peer's URI (address-of-record, as per [RFC3261] Section 23.3) appears in the subjectAltName of the peer's certificate as a uniformResourceIdentifier field.

- For TLS, assure that the peer's hostname appears as described in [RFC5922]. Also:

  * assure an exact match in a dNSName entry in the subjectAltName if there are any dNSNames in the subjectAltName. Wildcard matching is not allowed against these dNSName entries. See Section 7.1 of [RFC5922].

  * assure that the most specific CommonName in the Subject field matches if there are no dNSName entries in the subjectAltName at all (which is not the same as there being no matching dNSName entries). This match can be either exact, or against an entry that uses the wildcard matching character '*'

  The peer's hostname is discovered from the initial DNS query in the server location process [RFC3263].

- IP addresses can appear in subjectAltName ([RFC5280]) of the peer's certificate, e.g. "IP:192.168.0.1". Note that if IP addresses are used in subjectAltName, there are important ramifications regarding the use of Record-Route headers that also need to be considered. See Section 7.5 of [RFC5922]. Use of IP addresses instead of domain names is inadvisable.

For each of these tests, an implementation will proceed past the verification point only if the certificate is "good". S/MIME protected requests presenting bad certificate data will be rejected. S/MIME protected responses presenting bad certificate information will be ignored. TLS connections involving bad certificate data will

not be completed.

1. S/MIME : Good peer certificate

2. S/MIME : Bad peer certificate (peer URI does not appear in subjectAltName)

3. S/MIME : Bad peer certificate (valid authority chain does not end at a trusted CA)

4. S/MIME : Bad peer certificate (incomplete authority chain)

5. S/MIME : Bad peer certificate (the current time does not fall within the period of validity)

6. S/MIME : Bad peer certificate (certificate, or certificate in authority chain, has been revoked)

7. S/MIME : Bad peer certificate ("Digital Signature" is not specified as an X509v3 Key Usage)

8. TLS : Good peer certificate (hostname appears in dNSName in subjectAltName)

9. TLS : Good peer certificate (no dNSNames in subjectAltName, hostname appears in CN of Subject)

10. TLS : Good peer certificate (CN of Subject empty, and subjectAltName extension contains an iPAddress stored in the octet string in network byte order form as specified in RFC 791 [RFC0791])

11. TLS : Bad peer certificate (no match in dNSNames or in the Subject CN)

12. TLS : Bad peer certificate (valid authority chain does not end at a trusted CA)

13. TLS : Bad peer certificate (incomplete authority chain)

14. TLS : Bad peer certificate (the current time does not fall within the period of validity)

15. TLS : Bad peer certificate (certificate, or certificate in authority chain, has been revoked)

16. TLS : Bad peer certificate ("TLS Web Server Authentication" is not specified as an X509v3 Key Usage)

17. TLS : Bad peer certificate (Neither "SIP Domain" nor "Any Extended Key Usage" specified as an X509v3 Extended Key Usage, and X509v3 Extended Key Usage is present)

7. IANA Considerations

No IANA actions are required.

8. Acknowledgments

Many thanks to the developers of all the open source software used to create these call flows. This includes the underlying crypto and TLS software used from openssl.org, the SIP stack from www.resiprocate.org, and the SIMPLE IMPP agent from www.sipimp.org. The TLS flow dumps were done with SSLDump from http://www.rtfm.com/ssldump. The book "SSL and TLS" [EKR-TLS] was a huge help in developing the code for these flows. It's sad there is no second edition.

Thanks to Jim Schaad, Russ Housley, Eric Rescorla, Dan Wing, Tat Chan, and Lyndsay Campbell who all helped find and correct mistakes in this document.

Vijay Gurbani and Alan Jeffrey contributed much of the additional test scenario content.

9. Security Considerations

Implementers must never use any of the certificates provided in this document in anything but a test environment. Installing the CA root certificates used in this document as a trusted root in operational software would completely destroy the security of the system while giving the user the impression that the system was operating securely.

This document recommends some things that implementers might test or verify to improve the security of their implementations. It is impossible to make a comprehensive list of these, and this document only suggests some of the most common mistakes that have been seen at the SIPit interoperability events. Just because an implementation does everything this document recommends does not make it secure.

This document does not show any messages to check certificate revocation status (see Sections 3.3 and 6.3 of [RFC5280]) as that is not part of the SIP call flow. The expectation is that revocation status is checked regularly to protect against the possibility of certificate compromise or repudiation. For more information on how certificate revocation status can be checked, see [RFC2560] (Online Certificate Status Protocol) and [RFC5055] (Server-Based Certificate Validation Protocol).

10. Changelog

(RFC Editor: remove this section)

-02 to -03

* Re-worded "should" and "must" so that the document doesn't sound like it is making normative statements. Actual normative behavior is referred to in the respective RFCs.

* Section 5: re-worded paragraphs 4 and 5 regarding subjectAltName, and added references.

* Section 6: added references, clarified use of IP addresses, and clarified which From/To URI is used for comparison (from section 23.2). Added an EKU test case.

* Section 9: added text about certificate revocation checking.

* Appendix B.3: new section to present certificate chains longer than 2 (non-root CA).

* Made examples consistently use <allOneLine> convention.

* CSeq looks more random.

* Serial numbers in certificates are non-zero.

* All flows re-generated using new certificates. IP addresses conform to RFC 5737.

* Updated references.

-01 to -02

* Draft is now informational, not standards track. Normative-sounding language and references to RFC 2119 removed.

* Add TODO: change "hello" to "Hello!" in example flows for consistency.

* Add TODO: Fix subjectAltName DNS:com to DNS:example.com and DNS:net to DNS:example.net.

* Add TODO: use allOneLine convention from RFC4475.

* Section 3: updated open issue regarding contact headers in MESSAGE.

* Section 3.2: added some text about RFC 3263 and connection reuse and closed open issue.

* Section 5: clarified text about sender attaching certs, closed issue.

* Section 5: clarified text about observed problems, closed issue.

* Section 5: closed issue about clients vs. servers vs. proxies.

* Section 6: updated section text and open issue where IP address is in subjectAltName.

* Section 6: added normative references and closed "folklore" issue.

* Section 6: added cases about cert usage and broken chains, updated OPEN ISSUE: we need a SIP EKU example.

* References: updated references to drafts and re-categorized informative vs. normative.

* Section 9: added some text about revocation status and closed issue.

* Appendix B: open issue: do we need non-root-CA certs and host certs signed by them for help in testing cases in Section 6?

* Miscellaneous minor editorial changes.

-00 to -01

* Addition of OPEN ISSUES.

* Numerous minor edits from mailing list feedback.

to -00

* Changed RFC 3369 references to RFC 3852.

* Changed draft-ietf-sip-identity references to RFC 4474.

* Added an ASN.1 dump of CMS signed content where SHA-1 parameters are omitted instead of being set to ASN.1 NULL.

* Accept headers added to messages.

* User and domain certificates are generated with EKU as specified in Draft SIP EKU.

* Message content that is shown is computed using certificates generated with EKU.

* Message dump archive returned.

* Message archive contains messages formed with and without EKU certificates.

prior to -00

* Incorporated the Test cases from Vijay Gurbani's and Alan Jeffrey's Use of TLS in SIP draft

* Began to capture the folklore around where identities are carried in certificates for use with SIP

* Removed the message dump archive pending verification (will return in -02)

11. References

11.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC2560] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.

[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.

[RFC3263] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

[RFC3370] Housley, R., "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370, August 2002.

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

[RFC3853] Peterson, J., "S/MIME Advanced Encryption Standard (AES) Requirement for the Session Initiation Protocol (SIP)", RFC 3853, July 2004.

[RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, August 2006.

[RFC5055] Freeman, T., Housley, R., Malpani, A., Cooper, D., and W. Polk, "Server-Based Certificate Validation Protocol (SCVP)", RFC 5055, December 2007.

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

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5621] Camarillo, G., "Message Body Handling in the Session

Initiation Protocol (SIP)", RFC 5621, September 2009.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

[RFC5922] Gurbani, V., Lawrence, S., and A. Jeffrey, "Domain Certificates in the Session Initiation Protocol (SIP)", RFC 5922, June 2010.

[RFC5923] Gurbani, V., Mahy, R., and B. Tate, "Connection Reuse in the Session Initiation Protocol (SIP)", RFC 5923, June 2010.

[RFC5924] Lawrence, S. and V. Gurbani, "Extended Key Usage (EKU) for Session Initiation Protocol (SIP) X.509 Certificates", RFC 5924, June 2010.

[X.509] International Telecommunications Union, "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks", ITU-T Recommendation X.509 (2005), ISO/IEC 9594-8:2005.

[X.683] International Telecommunications Union, "Information technology - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications", ITU-T Recommendation X.683 (2002), ISO/IEC 8824-4:2002, 2002.

11.2. Informative References

[EKR-TLS] Rescorla, E., "SSL and TLS - Designing and Building Secure Systems", 2001.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC4134] Hoffman, P., "Examples of S/MIME Messages", RFC 4134, July 2005.

[RFC4475] Sparks, R., Hawrylyshen, A., Johnston, A., Rosenberg, J., and H. Schulzrinne, "Session Initiation Protocol (SIP) Torture Test Messages", RFC 4475, May 2006.

[RFC4514] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names", RFC 4514, June 2006.

[ssldump-manpage]
Rescorla, E., "SSLDump manpage".

Appendix A. Making Test Certificates

These scripts allow you to make certificates for test purposes. The certificates will all share a common CA root so that everyone running these scripts can have interoperable certificates. WARNING - these certificates are totally insecure and are for test purposes only. All the CA created by this script share the same private key to facilitate interoperability testing, but this totally breaks the security since the private key of the CA is well known.

The instructions assume a Unix-like environment with openssl installed, but openssl does work in Windows too. OpenSSL version 0.9.8j was used to generate the certificates used in this document. Make sure you have openssl installed by trying to run "openssl". Run the makeCA script found in Appendix A.1; this creates a subdirectory called demoCA. If the makeCA script cannot find where your openssl is installed you will have to set an environment variable called OPENSSLDIR to whatever directory contains the file openssl.cnf. You can find this with a "locate openssl.cnf". You are now ready to make certificates.

To create certificates for use with TLS, run the makeCert script found in Appendix A.2 with the fully qualified domain name of the proxy you are making the certificate for. For example, "makeCert host.example.net domain eku". This will generate a private key and a certificate. The private key will be left in a file named domain_key_example.net.pem in Privacy Enhanced Mail (PEM) format. The certificate will be in domain_cert_example.net.pem. Some programs expect both the certificate and private key combined together in a Public-key Cryptography Standards (PKCS) #12 format file. This is created by the script and left in a file named example.net.p12. Some programs expect this file to have a .pfx extension instead of .p12 - just rename the file if needed. A file with a certificate signing request, called example.net.csr, is also created and can be used to get the certificate signed by another CA.

A second argument indicating the number of days for which the certificate should be valid can be passed to the makeCert script. It is possible to make an expired certificate using the command "makeCert host.example.net 0".

Anywhere that a password is used to protect a certificate, the password is set to the string "password".

The root certificate for the CA is in the file root_cert_fluffyCA.pem.

For things that need DER format certificates, a certificate can be

converted from PEM to DER with "openssl x509 -in cert.pem -inform PEM -out cert.der -outform DER".

Some programs expect certificates in PKCS #7 format (with a file extension of .p7c). You can convert these from PEM format to PKCS #7 with "openssl crl2pkcs7 -nocrl -certfile cert.pem -certfile demoCA/cacert.pem -outform DER -out cert.p7c"

IE (version 8), Outlook Express (version 6), and Firefox (version 3.5) can import and export .p12 files and .p7c files. You can convert a PKCS #7 certificate to PEM format with "openssl pkcs7 -in cert.p7c -inform DER -outform PEM -out cert.pem".

The private key can be converted to PKCS #8 format with "openssl pkcs8 -in a_key.pem -topk8 -outform DER -out a_key.p8c"

In general, a TLS client will just need the root certificate of the CA. A TLS server will need its private key and its certificate. These could be in two PEM files, a single file with both certificate and private key PEM sections, or a single .p12 file. An S/MIME program will need its private key and certificate, the root certificate of the CA, and the certificate for every other user it communicates with.

A.1. makeCA script

```
#!/bin/sh
set -x

rm -rf demoCA

mkdir demoCA
mkdir demoCA/certs
mkdir demoCA/crl
mkdir demoCA/newcerts
mkdir demoCA/private
# This is done to generate the exact serial number used for the RFC
echo "4902110184015C" > demoCA/serial
touch demoCA/index.txt

# You may need to modify this for where your default file is
# you can find where yours in by typing "openssl ca"
for D in /etc/ssl /usr/local/ssl /sw/etc/ssl /sw/share/ssl; do
  CONF=${OPENSSLDIR:=$D}/openssl.cnf
  [ -f ${CONF} ] && break
done

CONF=${OPENSSLDIR}/openssl.cnf
```

```
   if [ ! -f $CONF  ]; then
       echo "Can not find file $CONF - set your OPENSSLDIR variable"
       exit
   fi

   cp $CONF openssl.cnf

   cat >> openssl.cnf  <<EOF
   [ sipdomain_cert ]
   subjectAltName=\${ENV::ALTNAME}
   basicConstraints=CA:FALSE
   subjectKeyIdentifier=hash
   authorityKeyIdentifier=keyid,issuer
   keyUsage = nonRepudiation,digitalSignature,keyEncipherment
   extendedKeyUsage=serverAuth,1.3.6.1.5.5.7.3.20

   [ sipdomain_req ]
   basicConstraints = CA:FALSE
   subjectAltName=\${ENV::ALTNAME}
   subjectKeyIdentifier=hash

   [ sipuser_cert ]
   subjectAltName=\${ENV::ALTNAME}
   basicConstraints=CA:FALSE
   subjectKeyIdentifier=hash
   authorityKeyIdentifier=keyid,issuer
   keyUsage = nonRepudiation,digitalSignature,keyEncipherment
   extendedKeyUsage=emailProtection,1.3.6.1.5.5.7.3.20

   [ sipuser_req ]
   basicConstraints = CA:FALSE
   subjectAltName=\${ENV::ALTNAME}
   subjectKeyIdentifier=hash

   [ sipdomain_noeku_cert ]
   subjectAltName=\${ENV::ALTNAME}
   basicConstraints=CA:FALSE
   subjectKeyIdentifier=hash
   authorityKeyIdentifier=keyid,issuer
   keyUsage = nonRepudiation,digitalSignature,keyEncipherment

   [ sipdomain_noeku_req ]
   basicConstraints = CA:FALSE
   subjectAltName=\${ENV::ALTNAME}
   subjectKeyIdentifier=hash

   [ sipuser_noeku_cert ]
   subjectAltName=\${ENV::ALTNAME}
```

```
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
keyUsage = nonRepudiation,digitalSignature,keyEncipherment

[ sipuser_noeku_req ]
basicConstraints = CA:FALSE
subjectAltName=\${ENV::ALTNAME}
subjectKeyIdentifier=hash

EOF


cat > demoCA/private/cakey.pem <<EOF
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFDjBABgkqhkiG9w0BBQ0wMzAbBgkqhkiG9w0BBQwwDgQIlwtc771DlNUCAggA
MBQGCCqGSIb3DQMHBAhRD3Z1i2TavwSCBMgXoXo0H/dTplHwnqfW7Uh1dr776z7B
lsNxlenMA6lYmALF/4E1tqOE2/aEbr8W3wTVjNpew9r5TBsbA1I9/FMMe+USc1ra
5pIdDLx7ynzHvxcUWJ1xbWGeLcEmXGOvzkwW/oOg49Yq1ce1GtlLSV2L7Wi93TUQ
Q8i5l0X0xjx7cB7kaHTOTyaN0sxUE3qlQ2sXTbbHWUfIaNpEZUI5ITrDUflfMnxb
RogQGv+5owsM7zwzfyGz3QocM9WaZwKFOEOqBvEfGaaZ9ml+cn1Rz/1Id7tSBlRH
3ucN2mGdEVIUvzSACZ9LPuIO7WBGM56enDRsqZji4WfqDHdXa4gkJKqPEJeBnLVA
jxCmLJSyikM25kHDm8LWuOckO/Rk+7999h13Qv1Ynm7yCincorqdlTrAdmq1Z8Tj
QPgXioTlx6++6yxiDCV7Mwkydox31K9y/Tf2cZ//dWuf/lfMaaq8HfpSNl4RKqsz
ufL41K5sCzPRIugUdooUQSGPC0JgcskPcifT6zvrI62KLPFVrwG5HT9PdevQvC6O
VgglxbEGJ7I4vllzmY62/0LtQKIA6bh8pszvvmHjGo9s+f+p7KJVYygEHNEmRTm+
8M2owk67033sV6IClDOAdRL8siTHmcmM+r1x9VVIppsDrzjqQqYVGYBbjEJW8eQp
t7kAjuN48tDD1mS8E6DstPv/6S0AjzAqCbjkuPJ0WU5fD1cY+iTpo9vcunohcj+i
KVXsM34wOsBpMBjFQ+Aww5bsIkEV1liOYLav1F7/BvP2s0gc3puM5W35y1cbKLu2
ThJV7mIWoV770aQYpJba0UAk9OzBVEvPNahrDI1NucbEkFrhN2pfnOs7k4UvrjiK
uknKrm3gocDOdstyMZX81Beyj06NhpcJH+bOSvROk/d68aAsapy6qS9hLijNNbcd
itQ/fo+1o9MDujT/huj7ZFqdzNM3KA6vxf0kmmVM+GJbYke+cjXk6WB80lF9lYcB
0pWPd+fgwFL252FUoFcjvUWFXkvbR1+IMkv6sNdKcXHHazAE6nl6yPl9bVwCaS1I
WNqEfHntblNZbeW+3qH8ov1ZXVCqEmaHkajSAhFJKXCgpSXaIx2FSntzpVFbRpnw
Yd9eml9xwgE3l9aRuvR6p61fd051LzCh7KjvorV1CemPUT6YRBamFNCBoT7cqjhE
kqMQfowKkMEY0p2dzMnGzsSPKk10nI53RgPyD/8FT5dPuq073SyjxTKhAbvl+kVl
lrfZ6b7P/UKwLBCT3bLG6uU/Es84euWN+U2JXIADPoCcVeWrUqkf4j368c2Z8Zdd
A27X4ZJ+q+YfsFNiOA7vshHi3Am3gBzQhEEGsRdzgkf8qmtlRGhq/823GExoUfu
8SiOOjoU08HGAkTtPWjV5+0C6Q6RW9SmNMwz7msZHoKTQ8kz2LKXUwb6DBwWcw6/
UTUgzVXqhA8HmjsnVe9ftDKL66v9zlp4RVRdDzm4TYUybYh5uigFbjJFLlnJnJho
TcnusHO80Cxgs64khLRzM46Oi+JSEPv7o7zHcfWNOVtNW908EKCubtEDZtnQn9VC
0Sky9R/WzunaLlG3LZ3BRUhWpyyvdNxlNq3ie4tcRMlXIEe14UZN0sPCKZY//NEn
BEc=
-----END ENCRYPTED PRIVATE KEY-----
EOF

cat > demoCA/cacert.pem <<EOF
-----BEGIN CERTIFICATE-----
```

```
   MIIDtTCCAp2gAwIBAgIJAJajhBdO74pMMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
   BAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMREwDwYDVQQHDAhTYW4gSm9zZTEO
   MAwGA1UECgwFc2lwaXQxKTAnBgNVBAsMIFNpcGl0IFRlc3QgQ2VydGlmaWNhdGUg
   QXV0aG9yaXR5MCAXDTExMDEyNzE4MzYwNVoYDzIxMTEwMTAzMTgzNjA1WjBwMQsw
   CQYDVQQGEwJVUzETMBEGA1UECAwKQ2FsaWZvcm5pYTERMA8GA1UEBwwIU2FuIEpv
   c2UxDjAMBgNVBAoMBXNpcGl0MSkwJwYDVQQLDCBTaXBpdCBUZXN0IENlcnRpZmlj
   YXRlIEF1dGhvcml0eTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKsf
   kWHxHMXNpnsWm7cUeeQwnpjQ7Ae3vXfX0fVbLOLu5rGw8IX6pbzLzM9pLE/8UO+d
   MSvAWer7ZG8fVac9/XDSVtsUmReScKwm+DRBcNnAA5FqutERj6wSMd65GXCNXad9
   ixnMQD+u/94f25SzRndsrq7/PtaEW8LeCyZl0JHHcEvHCkq/x5cE3bpYR8vgKyN2
   h2XFVTQQqycfHPgwPbCbyqKBcky9YP73If4L2wvb6VsBNtQoFWt569CRGyFZuA6q
   v9WxbHA3oz+lfQ6VRvb2WGeDdUI3GAukQTmyL2yALHjspQ++nBD4wAsNc5meDdeX
   UMvMRTQjSUGFIiStKcMCAwEAAaNQME4wHQYDVR0OBBYEFJVFfl8r6mWYEpEE82PH
   aJpYFncnMB8GA1UdIwQYMBaAFJVFfl8r6mWYEpEE82PHaJpYFncnMAwGA1UdEwQF
   MAMBAf8wDQYJKoZIhvcNAQEFBQADggEBAAZfnq6gmry1uVt+lzPM32OYmJTLDWap
   g+iqWCpZoZ5HMaavXD+iJYb43wWSt9tpoWlyh2bFqzWJATcZyXTrCdE/iHskE0LK
   LftF5sxL+CF48/WX7AmSJKLw5pSNl0oAlAC9JbgXLFJTXcxcSKShHS32UFUTpNOy
   ovTxuW1IXlzz3uD8WQmh2RRhZb/YP7m6LnztXCSba8qqX/HBHrCo2oIP+0xxO017
   OMjjiioZNEQmC+rwRzhGKGUE4gFS3ew95fVTdHd0dW3G2cIKrDu4mFxVUzR0Uqgm
   sS8wItCLt/Og3WgHM9Wut4GylFhyTnzGci+9bGn7tReoKo3XLJEGyAw=
   -----END CERTIFICATE-----

   EOF


   # uncomment the following lines to generate your own key pair


   # openssl req -newkey rsa:2048 -passin pass:password \
   #     -passout pass:password -set_serial 0x96a384174eef8a4c \
   #     -sha1 -x509 -keyout demoCA/private/cakey.pem \
   #     -out demoCA/cacert.pem -days 36500 -config ${CONF} <<EOF
   # US
   # California
   # San Jose
   # sipit
   # Sipit Test Certificate Authority
   #
   #
   # EOF


   # either randomly generate a serial number, or set it manually
   # hexdump -n 4 -e '4/1 "%04u"' /dev/random > demoCA/serial
   echo 96a384174eef8a4d > demoCA/serial


   openssl crl2pkcs7 -nocrl -certfile demoCA/cacert.pem \
```

```
           -outform DER -out demoCA/cacert.p7c

   cp demoCA/cacert.pem root_cert_fluffyCA.pem
```

A.2. makeCert script

```
  #!/bin/sh
  set -x

  # Make a symbolic link to this file called "makeUserCert"
  # if you wish to use it to make certs for users.

  # ExecName=$(basename $0)
  #
  # if [ ${ExecName} == "makeUserCert" ]; then
  #   ExtPrefix="sipuser"
  # elif [ ${ExecName} == "makeEkuUserCert" ]; then
  #   ExtPrefix="sipuser_eku"
  # elif [ ${ExecName} == "makeEkuCert" ]; then
  #   ExtPrefix="sipdomain_eku"
  # else
  #   ExtPrefix="sipdomain"
  # fi


  if [  $# == 3  ]; then
    DAYS=36500
  elif [ $# == 4 ]; then
    DAYS=$4
  else
    echo "Usage: makeCert test.example.org user|domain eku|noeku [days]"
    echo "       makeCert alice@example.org [days]"
    echo "days is how long the certificate is valid"
    echo "days set to 0 generates an invalid certificate"
    exit 0
  fi

  ExtPrefix="sip"${2}

  if [ $3 == "noeku" ]; then
    ExtPrefix=${ExtPrefix}"_noeku"
  fi



  DOMAIN=‘echo $1 | perl -ne ’{print "$1\n" if (/(\w+\..*)$/)}’   ‘
```

```
  USER=‘echo $1 | perl -ne ’{print "$1\n" if (/(\w+)\@(\w+\..*)$/)}’   ‘
  ADDR=$1
  echo "making cert for $DOMAIN ${ADDR}"

  if [ $2 == "user" ]; then
    CNVALUE=$USER
  else
    CNVALUE=$DOMAIN
  fi

  rm -f ${ADDR}_*.pem
  rm -f ${ADDR}.p12

  case ${ADDR} in
  *:*) ALTNAME="URI:${ADDR}" ;;
  *@*) ALTNAME="URI:sip:${ADDR},URI:im:${ADDR},URI:pres:${ADDR}" ;;
  *)   ALTNAME="DNS:${DOMAIN},URI:sip:${ADDR}" ;;
  esac

  rm -f demoCA/index.txt
  touch demoCA/index.txt
  rm -f demoCA/newcerts/*

  export ALTNAME

  openssl genrsa  -out ${ADDR}_key.pem 2048
  openssl req -new  -config openssl.cnf -reqexts ${ExtPrefix}_req \
          -sha1 -key ${ADDR}_key.pem \
          -out ${ADDR}.csr -days ${DAYS} <<EOF
  US
  California
  San Jose
  sipit

  ${CNVALUE}



  EOF

  if [ $DAYS == 0 ]; then
  openssl ca -extensions ${ExtPrefix}_cert -config openssl.cnf \
      -passin pass:password -policy policy_anything \
      -md sha1 -batch -notext -out ${ADDR}_cert.pem \
      -startdate 990101000000Z \
      -enddate 000101000000Z \
       -infiles ${ADDR}.csr
  else
```

```
  openssl ca -extensions ${ExtPrefix}_cert -config openssl.cnf \
      -passin pass:password -policy policy_anything \
      -md sha1 -days ${DAYS} -batch -notext -out ${ADDR}_cert.pem \
       -infiles ${ADDR}.csr
  fi

  openssl pkcs12 -passin pass:password \
      -passout pass:password -export \
      -out ${ADDR}.p12 -in ${ADDR}_cert.pem \
      -inkey ${ADDR}_key.pem -name ${ADDR} -certfile demoCA/cacert.pem

  openssl x509 -in ${ADDR}_cert.pem -noout -text

  case ${ADDR} in
  *@*) mv ${ADDR}_key.pem user_key_${ADDR}.pem; \
       mv ${ADDR}_cert.pem user_cert_${ADDR}.pem ;;
  *)   mv ${ADDR}_key.pem domain_key_${ADDR}.pem; \
       mv ${ADDR}_cert.pem domain_cert_${ADDR}.pem ;;
  esac
```

Appendix B. Certificates for Testing

This section contains various certificates used for testing in PEM format.

B.1. Certificates Using EKU

These certificates make use of the EKU specification described in [RFC5924].

Fluffy's user certificate for example.com:

```
-----BEGIN CERTIFICATE-----
MIIEGTCCAwGgAwIBAgIJAJajhBdO74pNMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMREwDwYDVQQHDAhTYW4gSm9zZTEO
MAwGA1UECgwFc2lwaXQxKTAnBgNVBAsMIFNpcGl0IFRlc3QgQ2VydGlmaWNhdGUg
QXV0aG9yaXR5MCAXDTExMDIwNzE5MzIxN1oYDzIxMTEwMTE0MTkzMjE3WjBWMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpv
c2UxDjAMBgNVBAoTBXNpcGl0MQ8wDQYDVQQDEwZmbHVmZnkwggEiMA0GCSqGSIb3
DQEBAQUAA4IBDwAwggEKAoIBAQCjLFkM6bzk7NOe+5kC7LE2OrfTHU3DOrauUL1f
VQh3jH6k6fBoMSiPIzJWGcMil6dt/aciKgG1r2G9X37BFOWYKbQ0TjiKJu4N2tsn
uXjWrKwEeDKYwnXnarctszzj65el74tZQlAXe/6nga83p+fjH7CN0HIvbBRCxgFo
4Y/9Vk19zxbcqgVhCwrKyuxR7FNuPSsAgP41GwYKYROIC0TzzP0rDrSiC6CXhBQu
7ivjL8EanoaaeGqiTFeT5wEm01YNvbAv+NrHPAHcyy0xjGzGXLRj6LKiQBG/rfht
EgGXHUf4ahWL+yeWc0RGNNckHM9WjdS+1pRb8KZn493PtPLVAgMBAAGjgc0wgcow
UQYDVR0RBEowSIYWc2lwOmZsdWZmeUBleGFtcGxlLmNvbYYVaW06Zmx1ZmZ5QGV4
YW1wbGUuY29thhdwcmVzOmZsdWZmeUBleGFtcGxlLmNvbTAJBgNVHRMEAjAAMB0G
A1UdDgQWBBSFlwm401U3JIrc3uORcuQiz5iHUjAfBgNVHSMEGDAWgBSVRX5fK+pl
mBKRBPNjx2iaWBZ3JzALBgNVHQ8EBAMCBeAwHQYDVR0lBBYwFAYIKwYBBQUHAwQG
CCsGAQUFBwMUMA0GCSqGSIb3DQEBBQUAA4IBAQCoqY/YiguI7f9Pv+XNj557uOby
LKrjI1uacV79IKPd2dPB8ujwvnfbM8yKe0+RK43W9xTDjeBg0zRQvL5nIs31dHv0
mmiiUiuBL0bTCZ8lwyDoENXvOHvRF9Tx11RnVvETzy/8i4P8FOcBglmDzLGN8Mfa
TrHczFTPbDtHR1mH2Rbsr6/hEhMjHgrb9bX/XasVDuMlkQAOkNvYBxGQgQE6SKiq
nrBi0zbwDLcvpxeSUjYpFArWZYZnc3RuqjzuRzgeyG4GgYUcLvC2BH1sONuBnLgH
4we+9S8JaGMEa4cONrmho/vIMAygY41tqwr4RLB4GRo4fvpqodRLS3V1v28J
-----END CERTIFICATE-----
```

Fluffy's private key for user certificate for example.com:

-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAoyxZDOm85OzTnvuZAuyxNjq30x1Nwzq2rlC9X1UId4x+pOnw
aDEojyMyVhnDIpenbf2nIioBta9hvV9+wRTlmCm0NE44iibuDdrbJ7l41qysBHgy
mMJ152q3LbM84+uXpe+LWUJQF3v+p4GvN6fn4x+wjdByL2wUQsYBaOGP/VZNfc8W
3KoFYQsKysrsUexTbj0rAID+NRsGCmETiAtE88z9Kw60ogugl4QULu4r4y/BGp6G
mnhqokxXk+cBJtNWDb2wL/jaxzwB3MstMYxsxly0Y+iyokARv634bRIBlx1H+GoV
i/snlnNERjTXJBzPVo3UvtaUW/CmZ+Pdz7Ty1QIDAQABAoIBAH+bSvjiQir1WnnW
YM78s4mpWeDr5chrvjmMQsyu/zQe11u4551T9FgcOl1DQGtpFjLaTz5Ug4nGYjVq
3QG6ieL5mkfddDH2R+zl3sWuMmYQG2ZTaZ41VWdo+V/v8Ap+T9YhA2UGiwQSoA/3
R0PLN3lTaws8nE+hwiaGGsweujBvcaIJu4RQrGHRHaeEplU+tfjcHHElfzUAmKyM
cMgF8IpdUcA1pyHe3Pyc0oGnLyEVnv291xGWQfWT7nqf7K0QDLA6+TvbG3fGEYIw
WK4DMraUbZ66Jlnj1XfADoxWOTsygV+KYhZcbwjBWAUSOSduAtfwa6b72OnWd28J
8KYvrXECgYEA1eCJZZSavxhlfxqsWC/WdQ8S3SimI62KSLrN3bI0RO/60KiU2ap3
16ZhNLq8t3DjpkWiZrukixs2odsU7k3z6q+qm++P0TUwL7z3Bri0FimqUeVSYgAf
ZmFgGz7wLAM29zhv0hTZjGrrwMlNSyJ2tjyqpiO1XqkbdBpPBxKPrdcCgYEAw09f
4M2QKQBFzjecPeQpwJqnh8cuoHS+2CNLYGjlmjd/zAUgVF2+WPA1R1DmjAqJ9iwh
15Yx3CbknpKbfhfilmHkcGyA+fjQaisq/NzN3Ya0FP9Waht0FoBsAHt9X5xFwXH6
YBKUrqoPF5DAy427EL1nsIRa+LtoPaTdqpphFzMCgYEAlgSOO0s2FA43uyTpeF3t
rmQpVilaB7KFSaiGGBgUY7p0koF9DwRsVT4l9sd48a7kb09ur2K08sHe2z8BenoB
Oj+HiyNJHHSTXRjNqNBLuTP2fMU+uPDfFX/92n6WFjkXB+d1P8VSJxUkUjCg36/H
1uHMzQZFBKXXVOPTROG3GDcCgYEAoPFmq8QZOIA+BbnzqVi8QzfuN8geFyE9JrSm
55JpKdT0HbZXts3tDjMbZGI5KUuB9nbViGb/PVBbcoSTV6vtD0kpyq7O9a5gaCyc
ZvS5PARFn0vt9NAcsHIxDZC1drU7EjaPQN3u4aPHff7NsK9haGD78gyPPoqIUsvp
0i0XNtsCgYEAxIUikI+5wXIrnC1FUt0gt6+4T0zc7qEO0EpQRtktZ/1saNXEhA6N
EUqWLJMOnClhp72V5IvXsKgjxU8VpgIZeHIIt5jZb8XMmBiSQxiVTf6rp3s8PqlM
EtXfh7TdJzKuRP7d0g2uG4boJMFf590nqNjrxj9VeSxEWUrSK3YG/h8=
-----END RSA PRIVATE KEY-----

Kumiko’s user certificate for example.net:

```
-----BEGIN CERTIFICATE-----
MIIEGTCCAwGgAwIBAgIJAJajhBdO74pOMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMREwDwYDVQQHDAhTYW4gSm9zZTEO
MAwGA1UECgwFc2lwaXQxKTAnBgNVBAsMIFNpcGl0IFRlc3QgQ2VydGlmaWNhdGUg
QXV0aG9yaXR5MCAXDTExMDIwNzE5MzIxN1oYDzIxMTEwMTE0MTkzMjE3WjBWMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpv
c2UxDjAMBgNVBAoTBXNpcGl0MQ8wDQYDVQQDEwZrdW1pa28wggEiMA0GCSqGSIb3
DQEBAQUAA4IBDwAwggEKAoIBAQDL5odVdA3gFf/MuGIqbMY8Kl7g7kUfexWkpXbT
ptxlxf2D8hzUX8/PUn2XXcTbP019DqA+MkMiX4NNGpDZyeoIrcquKUXK7UQlRoKy
Q6Va11DijHTqdPTWFIrRhbRUhPjj0WvG1AFPYRRG/IZfRQcH8Aw1w8XSp614m1mY
9XwL5LuHNimAgjADHMrSk1obmHws0thU9nV0t1UG1SA1lA32JZX81bqKDg3Tq1Ho
fsKU3GwoBZG507lVG5bcV2ByA5HnCFpFeDTDYE23197USLhqRtIqrxxr64SFo9Dn
P0mYH6e3lRveAZhdKIbCHgGaKqIr7+SZDnLdCyKDrFSPC/lbAgMBAAGjgc0wgcow
UQYDVR0RBEowSIYWc2lwOmt1bWlrb0BleGFtcGxlLm5ldIYVaW06a3VtaWtvQGV4
YW1wbGUubmV0hhdwcmVzOmt1bWlrb0BleGFtcGxlLm5ldDAJBgNVHRMEAjAAMB0G
A1UdDgQWBBQ02bNX/rnbbYoEy6wU7oyst63WbDAfBgNVHSMEGDAWgBSVRX5fK+pl
mBKRBPNjx2iaWBZ3JzALBgNVHQ8EBAMCBeAwHQYDVR0lBBYwFAYIKwYBBQUHAwQG
CCsGAQUFBwMUMA0GCSqGSIb3DQEBBQUAA4IBAQCTN2SNTLUcvgtVnBi3RBRtD0+p
aiFPtWQ+YWbyCG/+NetesegCwi7xBOgSK+GxUWpTVuDW5smyTTZyvrMQhpkckcyO
KvuUVz0/yK67oSume1vo75KY8BvgfeZXZG4PjqqelJ3czB0XLfeb6KFmtoiHQ/R7
4i/O9+MhB3Zoeg5bm5f2g9ljYwRbD1Uav/aH9WeGEX992d9XJ/bpGGPrAdgmV3jo
KDFKh8ys1yfmM3xVdU0qPtos2nlzGNaqoceeFZoYaMf8uTzoaan6KZkQDTiMDRpt
YKxyS721re/840FwDvt67w+GIfFf7ISrAlkHwroYt0NMnLv610rka8qnVvaQ
-----END CERTIFICATE-----
```

Kumiko’s private key for user certificate for example.net:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAy+aHVXQN4BX/zLhiKmzGPCpe4O5FH3sVpKV206bcZcX9g/Ic
1F/Pz1J9l13E2z9NfQ6gPjJDIl+DTRqQ2cnqCK3KrilFyu1EJUaCskOlWtdQ4ox0
6nT01hSK0YW0VIT449FrxtQBT2EURvyGX0UHB/AMNcPF0qeteJtZmPV8C+S7hzYp
gIIwAxzK0pNaG5h8LNLYVPZ1dLdVBtUgNZQN9iWV/NW6ig4N06tR6H7ClNxsKAWR
udO5VRuW3FdgcgOR5whaRXg0w2BNt9fe1Ei4akbSKq8ca+uEhaPQ5z9JmB+nt5Ub
3gGYXSiGwh4BmiqiK+/kmQ5y3Qsig6xUjwv5WwIDAQABAoIBAHCXmrGgRS0xWLBW
PLbKm+iLSRsR14+bqwbg663SHTAB1Yzvu+W2Bo2oMnvMJrEe0o407l2J6bJoZZvF
CKmKqrYiKaJkXgrBW/jtZ6xCWGPCNAL1pnX1IWG5tDIgj8SALOO4N7hyR0rrA4Rz
W0vuVQSYFFX4BhvdxZesyRwCqn3x0pPSff95Ad+vuJd5CYuFZCuyGkszQ3fi+Nia
Gqs01EuyolEv72rsw2E5+wtx3qXB8Z4HXr+Yq9NbE8lp2CWd1Uh1qIHl8kwWmnIG
V3oLKiIowV+M6Zx/uzwAMF0Rdn5kET+b5DOlIksUAAa8LZsf95rOvkLgw7aZaj5e
sXhAdGECgYEA8930YqU2+AcEkjC5hygw1M/X5k/IcvZp0a8/in2hJW7iZgGh0AFE
jjxuoIVXbxSf9cZ+M6g76Svww9ecmovLArqbhFaLfbZCsrLeEAhQtGcu3wv7o6px
N0EbbF5FmOK7qaQ1Sgqj0NF5zP2JsrxGNoRmgFFwVdcpP/3Jp/IlZEsCgYEA1guI
/7I8h9og1dmTPzMpvpnANdRF/iuMX9AE4LNRp09Hjx0B7Vuat1ABtx09/ZN1hLhZ
BTZ5R2R2RjbzSHXZ3FdoMgSx9Q3qa+xuPel4RcppHNjdYkPDhPLnOUwQBqFL6kyU
nTEF+k6VIZvNsmGbB6wpHU1cjDAZUx71p6W49TECgYAMHpa7pExUDT076rH9tpCe
sume544lsHtX0WbOAipVCuqzeRdKmBWJIBW7YoUS3yqH82JoPM8lamqfwQJmZ9Yh
/5YlAIwUJk+wQ9VnZJJmNM6OhTDvVFQmE9VCEHlS/Mmox6FiWZ8EjLSJ7HvAZzzy
Dqhtbh6wFW5WYM15zD3xewKBgQCRmIkY/QGFm0+Ih5ZMgB3eI7GGLB1sNe0nY1Ve
Dzv0pc3UQHQGI7CLDuYLy91V9o8St17+V76JXIHDYy97U4bdBau/kkgGm++gd9PJ
U11Xg8aaM73rUJLXhW7ZH68rA16jQnI4tpcNW5S/pr51n0UYI/hXkT7psPIZA08w
OV8lkQKBgQDaGzCYC/6WumGJUerVCzZd/H6+E3ntZmtz273c8+wV89oRtZzUoJY4
bVNrYFs9iKFxLtNGRECEU2VzDXHUAguqe05rbzPudAZ4wSsrNchUyw8LkIXHDckt
pVLs0vhRK2gW/W2I+p2exSPQPt3Uy8tT6IsB9ZbNg/H4D160heHkuQ==
-----END RSA PRIVATE KEY-----
```

Domain certificate for example.com:

```
-----BEGIN CERTIFICATE-----
MIID9DCCAtygAwIBAgIJAJajhBdO74pPMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMREwDwYDVQQHDAhTYW4gSm9zZTEO
MAwGA1UECgwFc2lwaXQxKTAnBgNVBAsMIFNpcGl0IFRlc3QgQ2VydGlmaWNhdGUg
QXV0aG9yaXR5MCAXDTExMDIwNzE5MzIxN1oYDzIxMTEwMTE0MTkzMjE3WjBbMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpv
c2UxDjAMBgNVBAoTBXNpcGl0MRQwEgYDVQQDEwtleGFtcGxlLmNvbTCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAN10BgIQwucEH7yMtiTnm5SjSDeFnm2D
EoRQGo5IsfqGjKeAub5S7KbKY0eErfZ0hYIWfk42QAp0LCCpag5qfzXPcHFjfelD
Z4FM6rUet0yjNQh7IQ0qcwdjnY11vx/UjuZnYHX36gp6bJCvkkXgYgWaihCY3HxU
i+RhlTsE/BBQ74BFul6El3bBICXBkh2JCvdVYmT66GmiYkxn0wjZYbU9FlS2t0SN
WSuQ1On7x32HWMMSrDN4AFC6BwWzuQEaY1Vs4XrsoweuOwKDoWngw9wtYemy47Nx
yKbP2vs+mcflcbnJF9TtvKBHVAmMbm1TmizJaMZv8T2RGiRDd32RaUsCAwEAAaOB
ozCBoDAnBgNVHREEIDAeggtleGFtcGxlLmNvbYYPc2lwOmV4YW1wbGUuY29tMAkG
A1UdEwQCMAAwHQYDVR0OBBYEFMwGWVuLXtYN8gVNG2hUHvz5QxkXMB8GA1UdIwQY
MBaAFJVFfl8r6mWYEpEE82PHaJpYFncnMAsGA1UdDwQEAwIF4DAdBgNVHSUEFjAU
BggrBgEFBQcDAQYIKwYBBQUHAxQwDQYJKoZIhvcNAQEFBQADggEBAGqa0dsAS5CG
sFPqbzAxiR6bCRS9b7kCqm9Y7jADuKH9s0Fy/7MNy3anF8ZXOAYT5fPkMBdN95e1
83Tpgfj0VaMN9YI4w5hDUh+EzRq0o0WfPeIx/cuire1gffrSqkkvQamAAbvttnXJ
l2l/DJFg8cRaNuhcrOGo55pV5eDNAfTek/Q4bMFx0v3NG10l65B7MUHnNw7lwAFI
kfc03cYfdOY0NObNkw8/zpStkdnicrGfHdOlfV7ipFbFsXFNEApdplbmVx9IpVx1
Z+qrNT72tvrB84rBgHEyGGwztfoWWhbhoWwZZ/VFaGRvsjHc41oastSHiZb9h7o4
TgoZBwNLm7E=
-----END CERTIFICATE-----
```

Private key for domain certificate for example.com:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA3XQGAhDC5wQfvIy2JOeblKNIN4WebYMShFAajkix+oaMp4C5
vlLspspjR4St9nSFghZ+TjZACnQsIKlqDmp/Nc9wcWN96UNngUzqtR63TKM1CHsh
DSpzB2OdjXW/H9SO5mdgdffqCnpskK+SReBiBZqKEJjcfFSL5GGVOwT8EFDvgEW6
XoSXdsEgJcGSHYkK91ViZProaaJiTGfTCNlhtT0WVLa3RI1ZK5DU6fvHfYdYwxKs
M3gAULoHBbO5ARpjVWzheuyjB647AoOhaeDD3C1h6bLjs3HIps/a+z6Zx+VxuckX
1O28oEdUCYxubVOaLMloxm/xPZEaJEN3fZFpSwIDAQABAoIBAB9s231ni4Dk4OwM
u7w48acCFLlsSLMZqoMEKwCN6FO4zDTo23LaqaJxje0UMuuKVXfEYWAP6r6RBcIM
yHQLQMoOCdLNX4y+d+2tUJErLq+9aUUu093ebDxcMntkfh6yNyUS/mk/KQMbpFRT
1dn8oWxSJc19I6yxArkB7/9UEcDut6vzdbz+agXpHZH4Tje5OWZQXkHzsYobM8Y8
c2XwudP1zdQtvOrrOeirexxpOQf4CBQnBxoGmbae9Wf27Kw2bBm5+blZFgdqNxoh
6Q3rJ9EDyWkrVMAq9a67a59wST1ymyC0c6FmfToCMGlgoMPHcEdvuNYPWd2322oK
ZdfsawECgYEA+AewMiTdhAE+9TId2qilLQV+y8bdTHQ9rSqW9SF+q5ShOpZa79ER
asuDuqxU+TiewS0ircrkIyzQmCc1fnfBJh5y6GukpUk8HdLLkA29fV3ZJe+Y4ZbL
b4TEy/RxEECQREgtnQiaw08yOlT1dobNwxzVsi3mrhtOpfbPBERZUSsCgYEA5JG2
aGRCkyzASGAnZmqqXCP/pImU+tJb2OCgQ6/3gsxi/l91LwtRhFgx/ptYCgZWlpbz
+mpnDqexKtowldbjorrUADw84zG4u9d+uWOCXEpCVIEu4DZsRURdy3OzpK1vJaUm
NLgBiDj8JkUFrXTi4Rzx1Xysf6ndWAxDPDdI+GECgYEAoyFrYY+dohSvs9UijY4e
FV5n5t8E7iQF7L72SoOdLHy1DjOV2+VF71erbDusJ751q9hj1qp7Iid3ips/M87P
2qJsMTGbOJrST0s1V6mx16LCD5Fmm/jyFIbeaMZ9FpNgT4ipd38RSyPrhTIbv7kp
3Ao7AtXtwtVzBPUvcz8A/8ECgYEAw2ps2F13qdql3nsO1Ho3gqVoaGUUUUlOK2MI
wjYM1/AkZrR4PKthm1PIEpT/tTpsBz2yBBO6XoYya5+10DWz0yoGHNljeR7GgRqh
hqC0EHGQuizkRd9hu+rSgiI+oXmCQF4tBv+Wl7+YnKOAUidP3gTgIZUA6fjxe9io
FzBxG6ECgYEAyAHvSeqqwmdotdpWgR3Fk1CmtH7ZPnF2rsuRBaBoYnWtU619ote+
+Bmd4fBUB9tQOzUC9desRtoK3+wlJKHEPjm/0FxtQQi9ogHEn4e6P9jOwXJNkSsa
GjGUfzQ3Vm2baeNMg7sH8C5mQ9nskDuCzdlVAB2bMp23oPl6cvPIb0E=
-----END RSA PRIVATE KEY-----
```

Domain certificate for example.net:

```
-----BEGIN CERTIFICATE-----
MIID9DCCAtygAwIBAgIJAJajhBdO74pQMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMREwDwYDVQQHDAhTYW4gSm9zZTEO
MAwGA1UECgwFc2lwaXQxKTAnBgNVBAsMIFNpcGl0IFRlc3QgQ2VydGlmaWNhdGUg
QXV0aG9yaXR5MCAXDTExMDIwNzE5MzIxOFoYDzIxMTEwMTE0MTkzMjE4WjBbMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpv
c2UxDjAMBgNVBAoTBXNpcGl0MRQwEgYDVQQDEwtleGFtcGxlLm5ldDCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAOwsdgPVSPMweLWsBDHUSXJS6Vk6pu6K
sVg8IWMf1g0TWTPc5jUAQlWlLNtmN4gcSzq5z1ecvf3rLMomJPZaWbektTTg1KZl
2wQgyP+vx/Hf1BByj3s2DE/KZoLnQjFQawHHMc+kCtSa6dCFTmD9nA5cYDVxNmKG
Kz/+5HYxe6ByI6NZGNlSB8ADPULcFg6UchO06JvrGFt1n9tAtMf5C31+YYGpqXBl
qZOV8Wo0Gp6Vlnd4LrvDZkwjpQ/o7EuFbiK34Gvh3cuh9EkMbk+IPgVv7ohjWPDl
6WygTkE2VXHDhhdN4MXPKyenXX35sB52fNytN+2qM8bo4QPfTZlGrx0CAwEAAaOB
ozCBoDAnBgNVHREEIDAeggtleGFtcGxlLm5ldIYPc2lwOmV4YW1wbGUubmV0MAkG
A1UdEwQCMAAwHQYDVR0OBBYEFNiNYjKOu6f046JHy28GDRVMeR7sMB8GA1UdIwQY
MBaAFJVFfl8r6mWYEpEE82PHaJpYFncnMAsGA1UdDwQEAwIF4DAdBgNVHSUEFjAU
BggrBgEFBQcDAQYIKwYBBQUHAxQwDQYJKoZIhvcNAQEFBQADggEBAHUzR2H2IWrQ
ls3iqNlG7815mOjm9mgQX6WP2ILwBOTOqtPJ9uE2XZU9qw6d9vdcbAgLpp4Em4T7
Whcs0zVTrgKpWjDlho/boRS1gP2Qu9I86zJzf2R3mhTHUsbpxIwMCcHQg/fdIIeP
5Ar8R5DZXx/Q9zdQLE+cjMSjxo7q7uOV8DRkgMpYtp7BURg5ZXhnkAhEHxa3/SbU
YGfy3PzRoAMQmRZieAXArsIxEfkaC4Dtox/D4XLvY7njBFv8H6wqlvQyDsKXWlUH
8dS9i/3wFEpQtymUUeXwk8gzf2ytT6hgrX70s6BLy/IeRU+wLJ3k5YZpopQZjDm1
fNQG/O8TJlQ=
-----END CERTIFICATE-----
```

Private key for domain certificate for example.net:

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEA7Cx2A9VI8zB4tawEMdRJclLpWTqm7oqxWDwhYx/WDRNZM9zm
NQBCVaUs22Y3iBxLOrnPV5y9/essyiYk9lpZt6S1NODUpmXbBCDI/6/H8d/UEHKP
ezYMT8pmgudCMVBrAccxz6QK1Jrp0IVOYP2cDlxgNXE2YoYrP/7kdjF7oHIjo1kY
2VIHwAM9QtwWDpRyE7Tom+sYW3Wf20C0x/kLfX5hgampcGWpk5XxajQanpWWd3gu
u8NmTCOlD+jsS4VuIrfga+Hdy6H0SQxuT4g+BW/uiGNY8OXpbKBOQTZVccOGF03g
xc8rJ6ddffmwHnZ83K037aozxujhA99NmUavHQIDAQABAoIBABfBYR2BlpTfi0S6
yLE6aSjWriILhD76NFxrr/AIg79M8uwEjCNIo2N5+ckXvv4x2l9N0U0+tt2Tii3L
KGyfKecO6isncjxKgn0nzw/o3nO1z97Xpxb9mL9t3GHOYRoUvK6xGpGILo60BlCz
F+8pk0jegc7eVFoUpMULHm/FCmpY30N5cvCHcAE/ncW49bZmH3gQ+cmr5UcKKDUY
baJyLd8Q1f+uSmtrfYZzRT5c+4wmrBUjv3w9poMJuEo4slRaDnyeKJPSNR/6/LJk
tqnqgNif9cj9wqF6hWA23dDmmU/kSRtn1KOz5XmV9Jbo4Fu64Fvn/m/hj5Og4CP9
hZUWIQECgYEA+nV2pzspCfS7jSebVnvjChvqJ0nJAilSqCmrSQIT5PRmO+GQs6UT
PVN4GE0Ms8TTJyvxVkpoagQ36VLw/Wr0jUm+Z+dv1TIlFWTas8RNmdZHMv0LvfEe
Qu2fTI68l2d/L9GBMUCYa/sucX5E9q+3LC+Qo9jw8ehWjQZsWYER4dsCgYEA8WYX
AqDdKjHRqu2h248gZsuogiZq05iuzXhk2VTQoiM92mu8m1Htak+eov3/3wojqxuw
TAQbf/t8EfQ7LIGjaKqAua7mgG/aNB6MGGwdpBAPUZDL+DuKfbDbzTOL/IuaW0Fp
40RC0Up5nTU9wzIKB7a6n5S5R0KXxiGUIphfcGcCgYA6IYdPmziUOfxJ79ZrBUgV
8ZKwWbzQxpyLsVgzEsthSaRs45a9S2QiyLvIECIRm25S2i0ilRSU/rOncPvEJc3q
+SG7Zgkb146p34WvUbGdMhHGcNsH0+3tJM/jagG1tmzbwWmV7+MwtNT7vI3vH6uJ
EuUkUlbiHsXv53zAbWekHwKBgBy5HwfLCEXbA62o9NdhImPY28YQuClRQ4tjReyu
MNz6AIQayahZiTxbGO8f9fAeDrxvYPzKiFMkI1EnlFrpWf48O3DcpMSninklIVpO
kwBQgOIdrods3j+yaZTzCzcTjVxKXkUSfDjW+b2A9kZhj9v3HCGc2qbl/5Utraio
JMMFAoGAHb+k+C4e8WrW+jXbbG/DgAkSokK5vZwZLHeWBig9bEi626xN/oFEQVXp
zqwyNo6zQaofmS6anT6P2M7NClSGJxh27eBTiTLp1NCXlGTWAQEtXmYtvnAZNzXC
5Ur0wvS5bLx0nbhJwN8ZBwzJhYup0kU3pn99GcF+vkj5Eg7Zftg=
-----END RSA PRIVATE KEY-----

B.2. Certificates NOT Using EKU

These certificates do not make use of the EKU specification described in [RFC5924]. Most existing certificates fall in this category.

Fluffy’s user certificate for example.com:

```
-----BEGIN CERTIFICATE-----
MIID+jCCAuKgAwIBAgIJAJajhBdO74pRMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMREwDwYDVQQHDAhTYW4gSm9zZTEO
MAwGA1UECgwFc2lwaXQxKTAnBgNVBAsMIFNpcGl0IFRlc3QgQ2VydGlmaWNhdGUg
QXV0aG9yaXR5MCAXDTExMDIwNzE5MzIxOFoYDzIxMTEwMTE0MTkzMjE4WjBWMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpv
c2UxDjAMBgNVBAoTBXNpcGl0MQ8wDQYDVQQDEwZmbHVmZnkwggEiMA0GCSqGSIb3
DQEBAQUAA4IBDwAwggEKAoIBAQC6VyOIP6UANXy766KHiYDxyOpYEFboLJv6SEtw
UWQoZS3hQurFidOu4gkCspblzaMoty7lnUexbFxUKdbJOWGMcB2hrezJ+6rwJPK/
bF5YDijVtVqMRd5lv/Ni5yzteHfrMszWnz3t+ojgak4XTjBJmP2RO0T67GUpEbFV
sDeYtWi+G1ebDAR6bf6Jdba2K6DnmkxT5Rr6oYJHIApYbubk28asBQN6EGBBgPEO
RReJYrjoJR/rBDDe1bxK+ONdFXPlwjI/TRPMpvUYraWgTjJ18tXISgF1htaa/Y1K
YP79Yun2Nl/3UQcPIc/C6CXBs3yAUK3qQO1G6C5pXH9KMMlNAgMBAAGjga4wgasw
UQYDVR0RBEowSIYWc2lwOmZsdWZmeUBleGFtcGxlLmNvbYYVaW06Zmx1ZmZ5QGV4
YW1wbGUuY29thhdwcmVzOmZsdWZmeUBleGFtcGxlLmNvbTAJBgNVHRMEAjAAMB0G
A1UdDgQWBBT7CTXlQ5GKWvxGZNY24mmmVuEnRDAfBgNVHSMEGDAWgBSVRX5fK+pl
mBKRBPNjx2iaWBZ3JzALBgNVHQ8EBAMCBeAwDQYJKoZIhvcNAQEFBQADggEBAKL9
wUWGRhCQdhjzY4bx0R5Kwz+NHvsb8rjlPqfdcbNujBCw+rD+/uux0G3HwW+Mraj5
U2tUehwz87k6SgdqADzL/CP2mjzCJo5uDhi+tzjeg6ZklTSZYQrL3FSv/AgcUfFI
9HuCGkix/htaoEMy2zNZnZOjdtFME9w7wb3GxxqWTUzl9TToloCXYmLeQo/jwuad
40ybun1P5CWkO5Md2Y5zuNfCsRRz5lLYtAVfANtLBfeFV+S87AwrrdeITT+iyB7H
Jj+t24U4IMC8MttcHBlPPBuRVc2kmhNEQuTzelCsldXgY2+kn8ItnLdv1mvLpXA2
2Y41CPLCSj9AlqqZL9I=
-----END CERTIFICATE-----
```

Fluffy’s private key for user certificate for example.com:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAulcjiD+lADV8u+uih4mA8cjqWBBW6Cyb+khLcFFkKGUt4ULq
xYnTruIJArKW5c2jKLcu5Z1HsWxcVCnWyTlhjHAdoa3syfuq8CTyv2xeWA4o1bVa
jEXeZb/zYucs7Xh36zLM1p897fqI4GpOF04wSZj9kTtE+uxlKRGxVbA3mLVovhtX
mwwEem3+iXW2tiug55pMU+Ua+qGCRyAKWG7m5NvGrAUDehBgQYDxDkUXiWK46CUf
6wQw3tW8SvjjXRVz5cIyP00TzKb1GK2loE4ydfLVyEoBdYbWmv2NSmD+/WLp9jZf
91EHDyHPwuglwbN8gFCt6kDtRuguaVx/SjDJTQIDAQABAoIBABtIBLi+8K5eJlvw
/MOxOwKrMrwf8ElftppnGTxhfjN31MbFIFA5hJd3GnCdqwAMIlYks6YEZ+mu/rmH
wp2FXCXOiFgSebd8tCMilbO27v0fXZUkTxR4aj4lY0HYrLg7yfrSXjER8WQ1KPMK
PVKmLOWpk34+2jOOhqUDpR3xhcJClQ81fC1hKe2JoixNDoPdfM3azTq8QUPLQD2I
mjww1IH1677G5o/6qMloOM0Feqv/3cUWiRmvPv4eyGHdNtuFXKFpB4DQQMQL7TD8
FoOHBymHIOzSSF+gYgBFOb0YNgu2CqZrfED9cf0rRotrbXf6tM+akclxfHhkfKaa
JPZosbUCgYEA4MaetKsa7azhEYMc4TK0xhhV5Hi6lj1xR/6h++uYF0OIOBjM9yU3
5n6vLpyghNbW2bK08OIWPO0F4syvyKYR2elmUDraH29DKAtRLEkU9K82RG4AmXmk
G6ZsWOfx6Jf35OnAKVj/7aN9jc4K1v6EFyQGYEXbp4I0fhFfbJBAe28CgYEA1Dmx
iKJD+jWW9ypHk51YJ3r+a5qPPNVmjGKQQje3Y6+rSlxmW0hMwXoCBOYRwhHBRA//
SxH93PZ8rECjNkhxp6Ao87X2Gcol5U6kH+rwfd/3+SsHqPrugaDIwNlgkcu8VRrP
8uP2CgJoDBi5UY2UR97GVK98x8k2Sf6kDT32mQMCgYB/KH3R8VY7jOiKcqTc1UWl
J1E3/gB4S+wQ8YELth0FVCP0sDsLuZdlItfRw7OfUraa01k/SHeSIfiJdIghN6mz
oDFMQ+7vh47zUWurZPCg95n4nk5ihIkNR1nV9elJTudjLcWS3pFyC2JU3XIObE+n
k66zufFoUuWFSCi2juibqwKBgCT6RHe1JjkDe2FniX8r7D88y/W9wXVtDWgqiE4x
XQ/OfP8A6IjBKTaQ5qcp2zBAXbdZPjc7VEta21A8FvQPXVZCrsAAFXha4413zVsO
WYblLlTI7ZXA2yvU8wW/Gnds00zU1iTRGX6W+sAY0rll/M8k/tOknA5HfeEYsEbq
Y/w3AoGASjoC9Fjy2aBvH8SQaimn/Rx3hOFR4myOGWtHxrXmezoO2YdcMO1d8rlz
A/sQRvVofHRwyoaIkZkALprEGyxEqCdMmEs1h9xYAcxfW23RfqC39DYb9RTrRkwa
ArJmcEdRESOsIYhhXGfE1QMGiwj1UXMWeYcLtqQKWiLLDTYYfQE=
-----END RSA PRIVATE KEY-----
```

Kumiko’s user certificate for example.net:

```
-----BEGIN CERTIFICATE-----
MIID+jCCAuKgAwIBAgIJAJajhBdO74pSMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMREwDwYDVQQHDAhTYW4gSm9zZTEO
MAwGA1UECgwFc2lwaXQxKTAnBgNVBAsMIFNpcGl0IFRlc3QgQ2VydGlmaWNhdGUg
QXV0aG9yaXR5MCAXDTExMDIwNzE5MzIxOFoYDzIxMTEwMTE0MTkzMjE4WjBWMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpv
c2UxDjAMBgNVBAoTBXNpcGl0MQ8wDQYDVQQDEwZrdW1pa28wggEiMA0GCSqGSIb3
DQEBAQUAA4IBDwAwggEKAoIBAQDE/QVN7nxDDu5ov6b0cmHIFH93KhNbTEyCisir
i4OeUBiCv9dgRgPBXffrIIVQdIlCoDeLDusHdsC9EfFWvg+pRlKVEDgwccO0F5AV
bq3MK2Njma5I0lwpIa0RXYQ0K//oX/+jZeakhFty/R9yer0KaXWdLRd6KtncISui
z9rFhlTB9lHg6vNJUN9+Xonbcs7siXbj3qZdhb7oipI4PoQlXVetyu+SzAVe6MsU
5lwLmpQpIzQdSsJyxaAsW+AsyxunhWWiPZ888UM4vXjacZuj8GvJ8w2XjgJilQvV
s8ojWMKnAGLaR7grTBmGQ90e6+cg7hWuoGBlQA0R0h8zWQz5AgMBAAGjga4wgasw
UQYDVR0RBEowSIYWc2lwOmt1bWlrb0BleGFtcGxlLm5ldIYVaW06a3VtaWtvQGV4
YW1wbGUubmV0hhdwcmVzOmt1bWlrb0BleGFtcGxlLm5ldDAJBgNVHRMEAjAAMB0G
A1UdDgQWBBR6WwH61Ul7BIWeiKM35fMAiE9xazAfBgNVHSMEGDAWgBSVRX5fK+pl
mBKRBPNjx2iaWBZ3JzALBgNVHQ8EBAMCBeAwDQYJKoZIhvcNAQEFBQADggEBAKE8
y9YyoZlkFw4WxPalK087sSEveKBfzh4TuYQf5YcSIPw0coZGj/gNxn1juiYhE93G
F+Si/hJM0M6cc7SLB5Spq06Tt3PyPBIOZOWk9koh92kDI3axSr6II9Plsvp+Xsrl
bz5Zy8njy/YZrk/qOaHqQ5J6nPNp5qwF+ns2t+5Zl88Lli5nkBgOXFOuE0RIkcdF
CUFRUj026GxAILR6wUThOzfq55Azwl5Y9Y9QmEjFhkbYLls00HxcJdnt+6Sdm/vN
MeMJZdTzp1x+8pfPhJgHoyz7nkAxhgzC9RT33ra33BNkMQ6esRlQONJ+ZRsRLhHP
O7+kvXvmj9AAsA291wY=
-----END CERTIFICATE-----
```

Kumiko’s private key for user certificate for example.net:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAxP0FTe58Qw7uaL+m9HJhyBR/dyoTW0xMgorIq4uDnlAYgr/X
YEYDwV336yCFUHSJQqA3iw7rB3bAvRHxVr4PqUZSlRA4MHHDtBeQFW6tzCtjY5mu
SNJcKSGtEV2ENCv/6F//o2XmpIRbcv0fcnq9Cml1nS0XeirZ3CEros/axYZUwfZR
4OrzSVDffl6J23LO7Il2496mXYW+6IqSOD6EJV1XrcrvkswFXujLFOZcC5qUKSM0
HUrCcsWgLFvgLMsbp4Vloj2fPPFDOL142nGbo/BryfMNl44CYpUL1bPKI1jCpwBi
2ke4K0wZhkPdHuvnIO4VrqBgZUANEdIfM1kM+QIDAQABAoIBADuLR+kwp3sVrlcX
Z34IfSofmBALNeKpA4+KJ/JCr7xQ9bfACXhecZAnuWLnZ6TUNRFgoKl2DvEookYE
gHD57n36dcf9KR7rpH5xiOoRlJNcoiRfNeFpRNZiCZBwNiAXFLnHGtznVnpwT7xI
axMNqsrU6epi0O/quAPkOu5x6e0+j+j3ZauI4EfD1w2R6moBMUtATauZEEyLuC9A
6bFz2AFDchPVLwSjNMu0tAJc8Fss8xKls9HUXGS22eUfHxWfkCGwChuW60obGmas
E7GS7h4g9QvvQ4hGSVy9/MmQ88GmT0LynOyzFBCpuwjOQTHwsD674ldMSL4kXYVK
jcnTAkkCgYEA4bjN2ILis3uWTjvTNnrmWn1QoZBZDhg1LuNs5o1XtOJ7CdkckUvs
nqqQYOzNk/9N8vUs12ds3csXHypuuGrJwAVf648RSPDUUQ2XOoPSL9NeuZt5V1fT
1VyVWanKCBZ5sztISNVPt7Pu8DtGLHch4S/7M+gEUQB1Ogz7fyJHvFsCgYEA32mE
6lN67aHkqMLa06ZI9JIk/3SsFIPpjwZ4tk+sQCqEzawPvkT7qF2+U8lVt0XXKJZL
aexsopsULCGS86TEAPoYtjjk91p6ZZj8mgRZLU55g+gRdTpAFhXMgIctU7U6cDIw
SPa6UxJp9XCa/Gf6YLfas9VBhc/8OC7I4ygjLDsCgYEAgAG7yuM/CSY3MRrARw8f
f4W9qkIgHtwfnP2gjobtjEk8GXOkvcle4QQ9aJoiY6HPZM8hpO6kUIuSCzyXGcKF
s33Yzc+Or9zTqzuX3blQA4tNFtlS0POf0En28KhXSIrmbXxbG+LMmJNUF6yluSW+
cuQxA1i6ye0Gjes63Phl0i0CgYEAuEcILGQpTGMyAYWgC93n5Vu6ir+IxO89sgyL
ew1irhakLiWTYsTxsyGHwQKb4i0IWOEHWVp7DPDPhcs3tCIezhN8WKm7KtAFj1HO
YZfemsFU99lutPwUKmNWqFlXqOkeR7cOHtDsRWM15Q45uKJnYmmkSptHjYFNsGXe
q4fK40sCgYBoAYtsLfMlqt7s3htx4hZSMFbLP/iMGW2DMMAzDW+Xxsvw86ibrcWY
8c3hbohuJBpyAzba4QoR2G+gtRmodLca+tQFMrObETHFglNCY+WoHRSNRImbCS8w
dsszPgHWf1nrxBLBiDFlHZwSqbZtLyBjPlHJ+fTiPNo6UTx8aDQ4Pw==
-----END RSA PRIVATE KEY-----
```

Domain certificate for example.com:

```
-----BEGIN CERTIFICATE-----
MIID1TCCAr2gAwIBAgIJAJajhBdO74pTMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMREwDwYDVQQHDAhTYW4gSm9zZTEO
MAwGA1UECgwFc2lwaXQxKTAnBgNVBAsMIFNpcGl0IFRlc3QgQ2VydGlmaWNhdGUg
QXV0aG9yaXR5MCAXDTExMDIwNzE5MzIxOVoYDzIxMTEwMTE0MTkzMjE5WjBbMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpv
c2UxDjAMBgNVBAoTBXNpcGl0MRQwEgYDVQQDEwtleGFtcGxlLmNvbTCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAKEVuYyZlaqfqks9u9yWQRp9WfI+VsQg
GpJH3vAfastElCdxlBV7+R2CaQ/GnXDnE0lAC5SiKRcvPHq5OLx1VnDADMWmcXBv
wK5n1zN+7MUCy/MISMr7E2Nd+py8Ft3XhjWDIuUljAh4HDO4fxS/BFy8zozADxvP
OfpE40EABF5aj7e+xjtkErdkMybAcSYyo53IHP3wDPxmMzCsOw/fi8bfy9j1GiUD
uz01F9qT/Opz9K1snxgT1IK6GRlktG4JawSiohW1QbARfj9//hR7ZgeB0gO6LLGX
cGXdl87JdA4ZHMZNinN4Cv8ctZYSQZ3dbt1pRRbGtq7elPskiinDuUkCAwEAAaOB
hDCBgTAnBgNVHREEIDAeggtleGFtcGxlLmNvbYYPc2lwOmV4YW1wbGUuY29tMAkG
A1UdEwQCMAAwHQYDVR0OBBYEFFNu6jHPsItA+vy/Jqv81MW7wLJpMB8GA1UdIwQY
MBaAFJVFfl8r6mWYEpEE82PHaJpYFncnMAsGA1UdDwQEAwIF4DANBgkqhkiG9w0B
AQUFAAOCAQEANH+wX56VJd0vVB9+Mef1xItWrSQUyNYZZCBq+y/5vIoOp6Chaupn
xjTjWf50zg6CK8yKBWq8pGlG45GTUx+uCx+nVIbHpyTT5+YDDUzlIhhAUzIOOB33
Fd/XI/1PK5p5ftuJIYXU0rGuaoH8ud/p2nhIf9mwicUHxViTX3PUwlFC7eMbevBo
8/dMYnHb2i40ug6hsiYggsmQDbhHLVLo/yqkpvgzPLSSlkXS4sv2oIoJ/ISuSjhP
QkQ7mh7h01ct/LOa53qWfbCVogQDhMEqPTVdPm+JzTrMlWeZdrk4KbnXGp64Jtpu
xTVI4GcVAGWUT0cmpspDmHbPOKm5kcltkg==
-----END CERTIFICATE-----
```

Private key for domain certificate for example.com:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAoRW5jJmVqp+qSz273JZBGn1Z8j5WxCAakkfe8B9qy0SUJ3GU
FXv5HYJpD8adcOcTSUALlKIpFy88erk4vHVWcMAMxaZxcG/ArmfXM37sxQLL8whI
yvsTY136nLwW3deGNYMi5SWMCHgcM7h/FL8EXLzOjMAPG885+kTjQQAEXlqPt77G
O2QSt2QzJsBxJjKjncgc/fAM/GYzMKw7D9+Lxt/L2PUaJQO7PTUX2pP86nP0rWyf
GBPUgroZGWS0bglrBKKiFbVBsBF+P3/+FHtmB4HSA7ossZdwZd2Xzsl0Dhkcxk2K
c3gK/xy1lhJBnd1u3WlFFsa2rt6U+ySKKcO5SQIDAQABAoIBABI9gIZAOedZLxJY
Cja/ON4EBbRdhLuumvOnecIc/J3JxTD2Nnt8T0gdJUJpDhjjwZZQzz7kYdzDN4j6
Akeszb30sT2MTFob/WiCT6cAH1VrrKZ3cK6zYY2l7aPj1H8IUaUrlT73UnT/DMp6
gMFbo+XQZ18evFc8zubc+BK7KsN4Nb6/zMhw+PXEiyg2EGDN1Fo4TMhxPD4wBIMU
8oLlE8A6GKimxAk3gMuIiS6Ruau2HpGkjkkHkAx/yzU1s8BCMoLDJjyyH19PRISr
n0VFfe0gM0aZpdZ/94ynFPdMnBXTq8BabT09eiycuLKlL0g/ERmj6jIImGSYRWED
GzlzX0UCgYEA0FDUek2uLhyltXwlzhDTldyuItiYZq/MeXaq2eA96zhJlD6aX+55
PQIxEEfhgTNf4e4cKjXQSD7aixy7jp/kFGowFRlB4pwbLDuhlniYSxa8Kv0OpJM4
DTAGue4QFZId5Z43KH755Ub7tjrCEIdQnij44DA3gPnjqXk973pdyVcCgYEAxfUx
/zMXgTp7HxW+QHZD7xXEs4Fp1xjzL5BaHoJnM7WbmkWvUvcMaEE/i9RqpyGlXRiN
jX6KBZ9UVgh/B0/AcYMa3DImTa0+Uie9kN7jTi5pzvIUAdFh+RyQ4tULWr5cgrzv
PjGG9tXMthuIbILSumVEwvC+P6Ksi1r4xp1ezl8CgYEArF51sk2clqM1qpnzXjMm
IJbdsA+w6ycD9m1uqaGXGo8UswmqCz70KrspheM0gQfVisjPnU2x7lWz1/AKcdVz
kEDdUFf54FxzT4J4Dl3zBg7l3FxQRXVbp+3ZYvfNb0vcWSc1VNjcRg8aMIsmES8m
UfhtFnRPOPWMn6qmyQVjnTkCgYB/3zlinkBKq9ooZEU3Iq4TXL5pLemOloFQcjCk
kJvVnTRcXTM5pngPSEaiLp6OQ3+sOVYG1nyV0SwLPwW/VVb8fDH3lzWC66vcKeuc
Dz5JnFWg5mLiIbzly/wTaochIOJlWWI5jIigHc9Uu0hOv9sbqJrYSea6+Hv4sNUO
h01chQKBgQCKLEH7vWQX8fkw+yKnmvAFoZ5H3IHUQw/WYsoCOVnWoY+vowcuuTTt
cbW1VkrtEjJPuYeEPa5NI2kmsNUZGrKCpx/3uq2JfMVopJzJN9biFM4ulcKqf9ie
hiVIFVVmxq+dVmXBgXCknhYK1Mnt9b3BK6mDqerQjK1TKryqAJ2QpQ==
-----END RSA PRIVATE KEY-----
```

Domain certificate for example.net:

```
-----BEGIN CERTIFICATE-----
MIID1TCCAr2gAwIBAgIJAJajhBdO74pUMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMREwDwYDVQQHDAhTYW4gSm9zZTEO
MAwGA1UECgwFc2lwaXQxKTAnBgNVBAsMIFNpcGl0IFRlc3QgQ2VydGlmaWNhdGUg
QXV0aG9yaXR5MCAXDTExMDIwNzE5MzIxOVoYDzIxMTEwMTE0MTkzMjE5WjBbMQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpv
c2UxDjAMBgNVBAoTBXNpcGl0MRQwEgYDVQQDEwtleGFtcGxlLm5ldDCCASIwDQYJ
KoZIhvcNAQEBBQADggEPADCCAQoCggEBAKoWx8g1KbnGX2YEOXrbod2pbR0fpkYW
V7O/tIWHddl+ACLlqqNPKSmIqwAFbZ2uf7S95OkXhkgRJGw3BugftUJS7zDhqVqi
dgPLMUPrdzpFazeh/AwBjc0wNBz/6tkUXrm7y/FwwzaCoKw+8Qm4Ibn2E3bNqWlm
iyKOXnYt4LGmy6J5e64hfQ3Vqe0ze5cfLKcpBbjF/TF75utbnH25zE0C/o1b+x1f
dwyDjsH0NN+A1ZFrI2NdleVAuH6F2vx4ctwZUzUJXyXezFmw5SRzhtWkb0iHO0ER
Ne7hCHLCv2Z6/GfIuHirCsGtNKSQIC6k74MyD7D75nltnLVgJ7Oxt28CAwEAAaOB
hDCBgTAnBgNVHREEIDAeggtleGFtcGxlLm5ldIYPc2lwOmV4YW1wbGUubmV0MAkG
A1UdEwQCMAAwHQYDVR0OBBYEFC1TKpLjuKa/dPumVbeFXEW4UR6EMB8GA1UdIwQY
MBaAFJVFfl8r6mWYEpEE82PHaJpYFncnMAsGA1UdDwQEAwIF4DANBgkqhkiG9w0B
AQUFAAOCAQEAJry8LukecUv4DUs5u/s6IymyqDLpeNvm94yrIIk/eRW72Jtr9rf5
6zF0Pd/+NzDXRYPe99HQgF3EKYndKIfnRUStJzIqiba2UszypDVRTQ6W9cH9e/1q
FdCjjeoVkRvnGo91S8DkgWM4boNRUgZtYwP+1I8hR+0717tp0f4fKjYX+NxPe30r
WzbLYXFDEiPndEgcxHc84Eeupit7VBQm7jxtF+XbaVGiLPGKCiYqdVS08h2ZakRK
8T3xL8Ecs4/rQn7PNPyEfS52R8hC70r66aAxZqLbKNpth/SZ3/hdeAyJ/NnFMW1J
uq3kB5YAJSwMYAUXaQhB1BvxKzXqstzJHQ==
-----END CERTIFICATE-----
```

Private key for domain certificate for example.net:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAqhbHyDUpucZfZgQ5etuh3altHR+mRhZXs7+0hYd12X4AIuWq
o08pKYirAAVtna5/tL3k6ReGSBEkbDcG6B+1QlLvMOGpWqJ2A8sxQ+t3OkVrN6H8
DAGNzTA0HP/q2RReubvL8XDDNoKgrD7xCbghufYTds2paWaLIo5edi3gsabLonl7
riF9DdWp7TN7lx8spykFuMX9MXvm61ucfbnMTQL+jVv7HV93DIOOwfQ034DVkWsj
Y12V5UC4foXa/Hhy3BlTNQlfJd7MWbDlJHOG1aRvSIc7QRE17uEIcsK/Znr8Z8i4
eKsKwa00pJAgLqTvgzIPsPvmeW2ctWAns7G3bwIDAQABAoIBAHIjpV+B5YVITL59
+UCr4JyKVLGlioQf/CygafjtZTVVa6v/aRn8Rkgb8XyrJ9sXvZVBlTqiUbdM4Z9I
8faVSKLAWsj3thkfSojTMzU77x+IdCG6LxSzekAGqAIJ7sRL+iEzl/Fm1WlgEYhl
GIWILgHH01n3O0eCy72dwmAV+2Hazn8eBggkWxMp0fblRC9pVh0FCo+jy1lHasjL
oOBkH51lbmZ4PUuUY072j2665gPm7i0nr25igef842JkbqAV8rAoNlQ26Y7tYLEw
6QyLv0odeb0rHZ8IEzahWAdmIPGCIUcFM7RmyInOatGA0dVEU3uYnkUQQVOi/JTx
46CCMbECgYEA4c1Dv/IVz9pdW1o/0MaJ94zfeg7Pgn5DRXnNMjCsSxVHSMINwlUl
BcYozs77vWbIuXiXO2xQe9mGA2ss3+vNxB0eu6EBQ/fK16cQQQH52nXdrV1sqnkN
5B5elFKcZKPfNVWrg0BC6csDndTcHp9STIKsxWkesLzC3Vz5UXZMsocCgYEAwNYV
+SsCIQGLT8ZZfKyE2nHqRUFknKc/tWQJop5gnE4ws3Lql3SNyCUQr/sDYelxQDE3
6COm197JcZ7jggDq7grigIxMznRxLMeG7bb7FfwPE/SKV0H5uagEB7ktFl8xIJKt
yOCK1ulillQjToSs4uetHLRXKCDSEpRiSw7wRdkCgYEAkDKBXYa/nykYDUqpDi57
1PbFkDD9G5x+YVPTUoX6wUgpabFjEANHzVQqo0dTRDTrYmY8Tdpx22WiS3SaB7WS
hfcCtVewczM++lDZ9GnKoVQ76IaM6qC72j36sEXBUhPEa072ZK8ZDCx1dsmEeJnN
+MZKhxcGXl9tIehJ31foyukCgYB9AUs1PwAeTVX13OrduyhUQ0xOoNmMA491Euh8
FpciPD2t1mzkyZWvjPeIXPwQWLglmMJZJeNeRPnpQcrR165zqXKzSj/wBePn12BM
cTXLRp6vnPKhJg+wno4eQ5hKzGKYbv1hHs5iCuDx+pD4sWExpmW+Gdn2FXCYwsAF
UCXJ4QKBgAKSrm8Y5xQhd8RAMg9JZLGUpPnmTKNU98f3fUFnX7jZEZETasnn18vd
65x04h58cohJJkNxqeL6k3lc3Mw0pzZrvsIha3ZMEoJPCgwBa8zLzrR13YQin6yf
+bAmfTDmhigpORB36ODY4B1kcwxKzQ0n3XAtlrL7NRV5wHr2ejkY
-----END RSA PRIVATE KEY-----
```

B.3. Certificate Chaining with a Non-Root CA

Following is a certificate for a non-root CA in example.net. The certificate was signed by the root CA shown in Section 2.1. As indicated in Sections 4.2.1.9 and 4.2.1.3 [RFC5280], "cA" is set in Basic Constraints, and "keyCertSign" is set in Key Usage. This identifies the certificate holder as a signing authority.

```
Version: 3 (0x2)
Serial Number:
    96:a3:84:17:4e:ef:8a:52
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit,
         OU=Sipit Test Certificate Authority
Validity
    Not Before: Feb  7 20:21:13 2011 GMT
    Not After : Jan 14 20:21:13 2111 GMT
Subject: C=US, ST=California, L=San Jose, O=sipit,
         OU=Test CA for example.net, CN=example.net
Subject Public Key Info:
```

```
        Public Key Algorithm: rsaEncryption
        RSA Public Key: (2048 bit)
            Modulus (2048 bit):
                00:d4:46:65:51:f8:84:1c:b5:93:47:a5:15:14:06:
                ec:dc:2a:77:93:11:5e:75:14:d2:88:54:bd:16:50:
                dd:41:3f:7e:2a:e4:26:d5:a3:33:b0:5e:37:1d:e5:
                96:37:1c:1c:69:80:a4:ef:fd:22:78:d7:ce:d3:c3:
                de:96:fb:87:30:88:bc:06:14:80:5d:f3:ab:d7:64:
                3e:07:31:dc:97:c5:d6:19:26:bc:7d:0b:f8:de:5e:
                f9:0f:dc:9a:45:0f:28:8d:dd:fa:15:56:d5:35:17:
                28:80:d2:fc:1f:d6:95:95:42:0e:2c:47:38:53:ad:
                fd:0e:24:fd:a3:43:33:83:52:65:54:da:48:d8:dc:
                86:42:d5:26:ac:1d:52:54:08:52:e5:3f:4a:76:95:
                77:8d:c6:f2:33:f0:18:87:c8:fc:5b:54:5d:dd:65:
                f1:5c:f5:c8:f4:36:54:8a:b6:7b:6f:f8:55:f8:d8:
                d8:df:a9:7b:40:45:4c:92:0f:aa:b2:2c:a1:a8:64:
                d5:99:22:1e:28:78:a0:d8:e5:51:64:3f:03:14:a9:
                12:47:61:84:d6:b0:69:1a:6b:a3:6e:d8:ca:ce:43:
                50:ad:57:96:2b:87:15:d9:c2:11:03:b0:82:d4:f0:
                80:bf:dd:44:f4:f6:39:0a:2b:e3:4d:d3:f5:e7:aa:
                34:e5
            Exponent: 65537 (0x10001)
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA:TRUE
        X509v3 Subject Key Identifier:
            72:70:CF:66:1E:23:A5:38:FC:6F:40:8F:86:8A:AF:E0:B9:6F:E9:C3
        X509v3 Authority Key Identifier:
            95:45:7E:5F:2B:EA:65:98:12:91:04:F3:63:C7:68:9A:58:16:77:27

        X509v3 Key Usage:
            Certificate Sign
        Signature Algorithm: sha1WithRSAEncryption
    70:73:c0:65:9c:2f:09:39:39:d6:a4:5b:95:e7:7b:43:34:b5:
    b9:b2:5d:76:eb:ef:87:e0:25:b6:68:ab:ee:f8:f7:85:c4:21:
    47:bb:6c:68:62:ff:f8:84:1e:44:5a:30:4e:ce:97:91:cc:3d:
    43:4a:8b:b7:25:26:08:63:c6:71:4a:c1:94:35:81:66:de:23:
    9d:e3:37:de:31:80:ed:58:b7:07:a7:ea:87:d3:cc:da:1b:62:
    c9:82:c2:17:e6:2d:20:e4:b2:69:14:cb:05:43:34:6f:b5:2c:
    60:d8:44:43:f9:e6:e9:3d:7c:54:a2:b9:d9:1e:7d:67:bb:3f:
    32:31:0d:c1:88:78:a8:67:39:f5:d2:3e:08:f7:38:84:a6:8f:
    c2:3e:00:ce:5f:b4:c8:da:a1:b5:2f:c2:89:60:a4:3a:2b:be:
    98:e0:44:34:af:ec:7f:73:26:f1:94:5b:39:09:b9:9f:93:c2:
    9d:7a:96:2f:82:66:c8:4d:f6:db:87:00:8e:bc:2a:b9:51:73:
    6c:cc:ff:e5:31:25:b1:4a:d0:9a:a9:c3:65:35:21:89:76:3d:
    39:f8:84:42:a6:03:0e:b5:c9:2f:5d:18:bc:9d:b9:82:f6:83:
    dd:2b:29:6c:8d:2c:8c:47:d4:7d:be:de:32:13:85:92:32:bc:
    61:62:6b:e5
```

Robert's certificate was signed by the non-root CA in example.net:

```
Version: 3 (0x2)
Serial Number:
    96:a3:84:17:4e:ef:8a:53
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, ST=California, L=San Jose, O=sipit,
         OU=Test CA for example.net,
         CN=example.net
Validity
    Not Before: Feb  7 20:21:13 2011 GMT
    Not After : Jan 14 20:21:13 2111 GMT
Subject: C=US, ST=California, L=San Jose, O=sipit, CN=robert
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (2048 bit)
        Modulus (2048 bit):
            00:d3:dc:14:69:6b:71:09:2c:0b:0f:9d:95:08:c1:
            64:20:66:ef:9f:9c:30:06:30:39:eb:14:16:da:19:
            cc:41:4d:b1:cf:f8:53:5b:a5:0d:76:ec:97:ba:16:
            10:9f:ed:57:b5:fb:6d:4b:9f:8f:d0:9f:0e:15:a7:
            3e:88:c4:e4:ef:35:d1:63:91:20:68:18:f4:8e:3b:
            b4:0f:03:3e:a0:00:d6:c3:26:e7:57:8e:21:92:a3:
            7a:2d:21:44:48:db:01:b9:54:e8:dc:d6:e3:d1:b3:
            f2:4b:26:0f:3f:d4:99:63:e4:7e:14:0a:b2:73:1c:
            5f:3b:41:36:e9:9a:70:be:f7:4f:08:6b:4a:db:44:
            02:e8:bb:50:66:2c:98:94:45:9e:7e:01:0e:9d:c3:
            a9:03:b7:28:15:28:c3:cd:a2:ad:ab:07:f6:ff:69:
            f4:ec:ba:7f:4b:bd:9b:28:8c:0d:87:e2:66:d1:24:
            34:e5:77:be:89:f1:c9:76:4c:37:34:3a:bc:d9:9c:
            36:f5:28:60:01:29:5c:f4:1e:7a:15:19:34:81:1c:
            cf:1a:06:5c:0f:f9:81:67:dc:50:09:e2:a8:d7:9d:
            9f:35:6e:ff:a6:a8:80:74:6c:f8:a1:0a:f3:bb:2b:
            b6:51:8c:21:bc:06:72:59:d0:95:42:d3:02:2c:ce:
            f9:23
        Exponent: 65537 (0x10001)
X509v3 extensions:
    X509v3 Subject Alternative Name:
        URI:sip:robert@example.net, URI:im:robert@example.net,
           URI:pres:robert@example.net
    X509v3 Basic Constraints:
        CA:FALSE
    X509v3 Subject Key Identifier:
        A6:42:BD:62:0D:6B:BF:EE:67:D4:C7:BC:09:3F:0B:3A:12:AB:19:CE
    X509v3 Authority Key Identifier:
        72:70:CF:66:1E:23:A5:38:FC:6F:40:8F:86:8A:AF:E0:B9:6F:E9:C3

    X509v3 Key Usage:
```

```
            Digital Signature, Non Repudiation, Key Encipherment
        X509v3 Extended Key Usage:
            E-mail Protection, 1.3.6.1.5.5.7.3.20
        Signature Algorithm: sha1WithRSAEncryption
    25:99:ea:1a:1e:96:6d:4e:b1:9c:5a:43:77:ea:3a:a7:a1:b7:
    22:db:b9:d4:9a:1e:17:f7:13:2e:b2:ca:80:dd:c9:a5:db:61:
    41:c6:8b:65:ae:0e:fc:9a:46:77:16:e0:e2:3d:1d:20:3c:e5:
    d5:e0:b8:03:41:4f:e7:69:bf:e0:4c:dd:cc:c4:51:b1:da:2f:
    ad:58:e1:ed:c6:5b:04:ea:1e:af:9a:89:cd:be:60:3c:9a:30:
    51:7f:99:5a:6b:5c:8f:5a:d4:b8:ce:b5:8b:31:74:70:b3:cc:
    5c:04:90:d8:8d:b6:75:55:fb:c1:d8:e8:db:cf:3d:80:e4:8d:
    2f:7e:b9:2b:a2:9e:9f:1e:6f:d0:4e:6e:f7:f0:a6:61:3b:9e:
    9b:4b:78:6b:84:37:ad:93:19:0d:7f:46:5a:18:74:89:8b:a8:
    1a:75:bf:db:df:25:43:4b:57:ab:a1:19:2e:7c:7b:b9:b5:50:
    ef:2c:1f:5c:18:8f:6c:66:83:61:eb:25:a3:21:81:2c:61:3b:
    ee:8c:18:1a:89:9a:29:0d:5c:5b:38:f3:71:3d:61:f0:3f:80:
    33:90:f2:60:53:48:fb:7a:65:c9:5f:1f:a3:e8:75:42:42:f5:
    ad:db:60:29:c6:0f:3c:68:00:7a:2b:38:db:c7:17:b9:4e:d8:
    90:d8:52:bc
```

Certificate for CA for example.net in PEM format:

```
-----BEGIN CERTIFICATE-----
MIIDzzCCAregAwIBAgIJAJajhBdO74pSMA0GCSqGSIb3DQEBBQUAMHAxCzAJBgNV
BAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMREwDwYDVQQHDAhTYW4gSm9zZTEO
MAwGA1UECgwFc2lwaXQxKTAnBgNVBAsMIFNpcGl0IFRlc3QgQ2VydGlmaWNhdGUg
QXV0aG9yaXR5MCAXDTExMDIwNzIwMjExM1oYDzIxMTEwMTE0MjAyMTEzWjB9MQsw
CQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpv
c2UxDjAMBgNVBAoTBXNpcGl0MSAwHgYDVQQLExdUZXN0IENBIGZvciBleGFtcGxl
Lm5ldDEUMBIGA1UEAxMLZXhhbXBsZS5uZXQwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQDURmVR+IQctZNHpRUUBuzcKneTEV51FNKIVL0WUN1BP34q5CbV
ozOwXjcd5ZY3HBxpgKTv/SJ4187Tw96W+4cwiLwGFIBd86vXZD4HMdyXxdYZJrx9
C/jeXvkP3JpFDyiN3foVVtU1FyiA0vwf1pWVQg4sRzhTrf0OJP2jQzODUmVU2kjY
3IZC1SasHVJUCFLlP0p2lXeNxvIz8BiHyPxbVF3dZfFc9cj0NlSKtntv+FX42Njf
qXtARUySD6qyLKGoZNWZIh4oeKDY5VFkPwMUqRJHYYTWsGkaa6Nu2MrOQ1CtV5Yr
hxXZwhEDsILU8IC/3UT09jkKK+NN0/XnqjTlAgMBAAGjXTBbMAwGA1UdEwQFMAMB
Af8wHQYDVR0OBBYEFHJwz2YeI6U4/G9Aj4aKr+C5b+nDMB8GA1UdIwQYMBaAFJVF
fl8r6mWYEpEE82PHaJpYFncnMAsGA1UdDwQEAwICBDANBgkqhkiG9w0BAQUFAAOC
AQEAcHPAZZwvCTk51qRbled7QzS1ubJdduvvh+Altmir7vj3hcQhR7tsaGL/+IQe
RFowTs6Xkcw9Q0qLtyUmCGPGcUrBlDWBZt4jneM33jGA7Vi3B6fqh9PM2htiyYLC
F+YtIOSyaRTLBUM0b7UsYNhEQ/nm6T18VKK52R59Z7s/MjENwYh4qGc59dI+CPc4
hKaPwj4Azl+0yNqhtS/CiWCkOiu+mOBENK/sf3Mm8ZRbOQm5n5PCnXqWL4JmyE32
24cAjrwquVFzbMz/5TElsUrQmqnDZTUhiXY9OfiEQqYDDrXJL10YvJ25gvaD3Ssp
bI0sjEfUfb7eMhOFkjK8YWJr5Q==
-----END CERTIFICATE-----
```

Private key for CA for example.net:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA1EZlUfiEHLWTR6UVFAbs3Cp3kxFedRTSiFS9FlDdQT9+KuQm
1aMzsF43HeWWNxwcaYCk7/0ieNfO08PelvuHMIi8BhSAXfOr12Q+BzHcl8XWGSa8
fQv43l75D9yaRQ8ojd36FVbVNRcogNL8H9aVlUIOLEc4U639DiT9o0Mzg1JlVNpI
2NyGQtUmrB1SVAhS5T9KdpV3jcbyM/AYh8j8W1Rd3WXxXPXI9DZUirZ7b/hV+NjY
36l7QEVMkg+qsiyhqGTVmSIeKHig2OVRZD8DFKkSR2GE1rBpGmujbtjKzkNQrVeW
K4cV2cIRA7CC1PCAv91E9PY5CivjTdP156o05QIDAQABAoIBADp/7/pIH7h9vcn3
z7hGNE50kaGBHuPrSh3yJG4a+O67XbzaRW2I3XzUaiIeHGixoY7duha9Txu4dbJc
f2JijR4uAIs4aSv7NDdW09VNw3o8NkWWLEnV288Eo2Tgqc8wXz/BleL9nCJWcH4Y
Jw1rKKwKmTdQpVBCWcPlI9UzduXQdZfBbrsL6+OZ+F3kbvUwYAVhhUuBS9sf4Xib
5GA2CDLPm433giOS3yr9KigpcLvbhAhMiPTXJ6i65m9xGGCcjhxP/drOH0cNczRD
yW0FCbaNRJUg9kEVu+n3uG1aVfOnU7RqcblFXgO7ea7G+mfp3Cfm744kvFEXz04k
8WLW6gECgYEA9lK9mKhMUeB1+xPJB4Za5QvrFc7nLt8ee7/aTNcyMI0l3uXyPDPj
TNEfgaRobptmwd2HVtXjlQ54fE+pE+qS8dOORh2VFoWi91zI4C8WnM/6j5P+QiXY
tcZDPF22bmsSW7uaQyaOhUfIMhzox1BbUH5q5YrcA5DmmQtaxcIZ+IECgYEA3J07
6DamIgy0eJO2GKHU/Hy8RvQZgauzCtmqmLQrWZeOmx9hORe1a71QU5F6Y3HQRcTD
RDDdJua9Y8BJ0WTkasbRgxjmHQlf4pUdT6ycfWgISbcCNFTosgPH+/OZPEh4DKlO
rbldUzHPuZdo2Q72KtSPMk+ikny2lCZ9cm2mKmUCgYEAsGoX4fJ/HpDMzrKf4qTG
Co8bojXZ+wbPVT/Vf/0LtBwTCG3VrGpZG5YWo4n1RWpFEQmwuW9cnE+N2TJQXLQ+
47Vpiyv6r/OsAM9SCsWOw2ZtBFGw4v0qFR3W37AaTUCgGFTnKbq+jhQX/FQaH02c
6KxxsM5fvqoTjX7FVycp5IECgYA4Tq1WpHQcpq99Qv4sJUnuM4v+dBj6fq9Q6qNf
HEUgNc2BDC5NWx7D4+rXmX7qWMc2t3S7N9mKL0RRbGeq2RxvoFUjJ7y71oOxmiuE
BWNfoqjS37HhV3aY0Nw/EzqeJ0T0vlXFg1Utgb4p+VoaZHYyElSGG8s7pjcXcwd7
qD7L/QKBgQCeDLKx5T1d/EqwW8KNK5qD/5lG/T0zu3MCDlzCjfs2BHMasv5RALd+
unMMANDElPHOFs7fSmCfspN8Y7+W15/k9WugpwQfST2Y8dSRVdPFp1FRt8u25yX2
mdRbU3vJSiAqPEEpKpBolXPxLOeLGvoTHFWSazgmCPIKKxq0wL+0+w==
-----END RSA PRIVATE KEY-----
```

Robert’s certificate:

```
-----BEGIN CERTIFICATE-----
MIIEJjCCAw6gAwIBAgIJAJajhBdO74pTMA0GCSqGSIb3DQEBBQUAMH0xCzAJBgNV
BAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMREwDwYDVQQHEwhTYW4gSm9zZTEO
MAwGA1UEChMFc2lwaXQxIDAeBgNVBAsTF1Rlc3QgQ0EgZm9yIGV4YW1wbGUubmV0
MRQwEgYDVQQDEwtleGFtcGxlLm5ldDAgFw0xMTAyMDcyMDIxMTNaGA8yMTExMDEx
NDIwMjExM1owVjELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAP
BgNVBAcTCFNhbiBKb3NlMQ4wDAYDVQQKEwVzaXBpdDEPMA0GA1UEAxMGcm9iZXJ0
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA09wUaWtxCSwLD52VCMFk
IGbvn5wwBjA56xQW2hnMQU2xz/hTW6UNduyXuhYQn+1XtfttS5+P0J8OFac+iMTk
7zXRY5EgaBj0jju0DwM+oADWwybnV44hkqN6LSFESNsBuVTo3Nbj0bPySyYPP9SZ
Y+R+FAqycxxfO0E26ZpwvvdPCGtK20QC6LtQZiyYlEWefgEOncOpA7coFSjDzaKt
qwf2/2n07Lp/S72bKIwNh+Jm0SQ05Xe+ifHJdkw3NDq82Zw29ShgASlc9B56FRk0
gRzPGgZcD/mBZ9xQCeKo152fNW7/pqiAdGz4oQrzuyu2UYwhvAZyWdCVQtMCLM75
IwIDAQABo4HNMIHKMFEGA1UdEQRKMEiGFnNpcDpyb2JlcnRAZXhhbXBsZS5uZXSG
FWltOnJvYmVydEBleGFtcGxlLm5ldIYXcHJlczpyb2JlcnRAZXhhbXBsZS5uZXQw
CQYDVR0TBAIwADAdBgNVHQ4EFgQUpkK9Yg1rv+5n1Me8CT8LOhKrGc4wHwYDVR0j
BBgwFoAUcnDPZh4jpTj8b0CPhoqv4Llv6cMwCwYDVR0PBAQDAgXgMB0GA1UdJQQW
MBQGCCsGAQUFBwMEBggrBgEFBQcDFDANBgkqhkiG9w0BAQUFAAOCAQEAJZnqGh6W
bU6xnFpDd+o6p6G3Itu51JoeF/cTLrLKgN3JpdthQcaLZa4O/JpGdxbg4j0dIDzl
1eC4A0FP52m/4EzdzMRRsdovrVjh7cZbBOoer5qJzb5gPJowUX+ZWmtcj1rUuM61
izF0cLPMXASQ2I22dVX7wdjo2889gOSNL365K6Kenx5v0E5u9/CmYTuem0t4a4Q3
rZMZDX9GWhh0iYuoGnW/298lQ0tXq6EZLnx7ubVQ7ywfXBiPbGaDYesloyGBLGE7
7owYGomaKQ1cWzjzcT1h8D+AM5DyYFNI+3plyV8fo+h1QkL1rdtgKcYPPGgAeis4
28cXuU7YkNhSvA==
-----END CERTIFICATE-----
```

Robert’s private key:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA09wUaWtxCSwLD52VCMFkIGbvn5wwBjA56xQW2hnMQU2xz/hT
W6UNduyXuhYQn+1XtfttS5+P0J8OFac+iMTk7zXRY5EgaBj0jju0DwM+oADWwybn
V44hkqN6LSFESNsBuVTo3Nbj0bPySyYPP9SZY+R+FAqycxxfO0E26ZpwvvdPCGtK
20QC6LtQZiyYlEWefgEOncOpA7coFSjDzaKtqwf2/2n07Lp/S72bKIwNh+Jm0SQ0
5Xe+ifHJdkw3NDq82Zw29ShgASlc9B56FRk0gRzPGgZcD/mBZ9xQCeKo152fNW7/
pqiAdGz4oQrzuyu2UYwhvAZyWdCVQtMCLM75IwIDAQABAoIBAAv+Q3GMUYPRaHbj
1tH+EKr86MfCUb2n8T9rjbefCj8QJOa/CgkAGPkIf7ZbFWnYR8TXjOJhEAUhW+zB
4PphGwynoUjfqFP8RavfmVvYNS1dnsrBYwtD0oa4lmwDnBf7vec99Ui7KX5vj2HN
r8NPR7et8a00xdFaY9G46WDkC0nkH8AqMMymY/Vu2KpH0f01hTpFLmxS7We+d3Uq
mva15GUc8+EL079uphokchr4E0036Ce4luCnqQfOUAKcXCMYK27lG5uue620IXLE
CqeevZPEn8eqWhSNGl981CF15AEb0tApMcMwrfcbpnQMHQuyQHm2XVewgF0gQGLn
UA0i6NECgYEA9TrFg3Kuw1Vfi+kztX6IMjW07YgN443NtB/9+sXKoc0Iz6LoPbOT
VHSVqHHpjicicBUyUa77Kr61HAv7AV0s2FRHAb3M7wOVYGkT52+12o4FH6EMU42G
ISAcsS4vCfHhYq1T0hC91bIY1XXxuBrpo0yb1RkEaSALHN6arAEgWccCgYEA3Sod
gEcahQEnu5P8UY5j9yFaBRqVxdQKWnO2trkfLkyVgtvn7ES31EGojVHg23nr5IsK
IpwFgBiQvEGUgV3dR0Jc5sZTETOweWBLebC/CtZfnhBcCNx8jwX5m/CtTzMHuxVs
VJ1WpUDn+K7+G8KIK0+Kp5QdOCxXptHRLkGPBcUCgYAVgCulFL8B3VBdQfsIpKlo
TZEpak5dbydj7ZIlFIZpnUJyggP+tOnr87TTafliP0gjr5gT1VWsL8BNTzeYrQSr
iugW3P9EzXmhVFUsa3z0RpNobIRaJwRljx0046m4I37xWeUJe/JI9C59OLQSwjlN
2f+ntWPPm8GdrF6/SfH+LQKBgQCyDaf2kEf/cHCmiXuHxVUhrs4kccTGofE75RDi
hqNdyPZNhfFvu9srnTivnY2j5MJPGsksF+Qtvpk3lqySghkVt43HlT9nB/A5p5bb
/7muZexQ+ua9k5UMKElOjDNbIcBFk/fFH26UWG7pPSkC/FhYVg9Q3uOvR7PBcAYy
cUFN6QKBgBw2k5SDvun41wNV4wxGEli9ia+i4lzg8pwJ1DUxnOcDvlDGzAzCNtW9
wPoR+jvhK6V6X1mI0tqqcYZ07pC3CJBEtAckHj2Ik+ZAEjQMf+eH62Rcv6Sbozq0
5dFCBZwzIe2IQomg3J8+OyILSs/uzFkjGjloJIrP+OtPKSrfR+/Y
-----END RSA PRIVATE KEY-----
```

Appendix C. Message Dumps

This section contains a base64 encoded gzipped, compressed tar file of various Cryptographic Message Syntax (CMS) messages used in this document. Saving the data in a file foo.tgz.b64 then running a command like "openssl base64 -d -in foo.tgz.b64 | tar xfz -" would recover the CMS messages and allow them to be used as test vectors.


```
-- BEGIN MESSAGE ARCHIVE --
H4sIAIpaUE0CA+ybeUATxx7HCSCIHIpoqSIQvFECu5tsDhAEDATQhCsQExTZ
JBtIyGUSIEREREU8i1ZRqVYERVHUCqKiUBWP1vusXCJeeIv3LfpCaRUpSF8f
tJXH/JPdmd3fTjYz8/n+fr8JT6LEKSVCCYqTKCMd+YhKp/0LAABEAgHb8Eki
wp98NhSIQACxIAhDBACGIRDCAiCBQCTqYAGdv6HEKFWIQtsVrkKISD9zXVvt
jd8F++HzCyl0r+BgD5oXVimUO0fHSITRMndUjUjkYtRRiqqwwb4BTpAjYNoj
VIg4/37mxBwTgAUp2iNHyBFyBmEAAF24CkTKi3LVUKJoBO5YHJ9MggkaHAUi
CxASgSvAc3kwgQDgQBzu9zYXhVymULnCAImgfQAdUeO8ZYo4RMFXOmNJ2hqm
zBk7quV+uZn28FbIJL+1C8QxAkH8h3aeTOLmokIiXXkIWSAgEHimPcYgYjHO
l+qMZYui49gsdpw/ky9mM33V2mOAwWTDdCpPQ6eFSugsuppOjYbZIraj9rZg
dLIzlkwg4bG/vSfTHh48HipXOWMlMWKVUI4oVE5KYaQU5TtgVaha5SQXI0Kp
AxaRy8VCHqISyqRO8miekoRrmGOf1iv5cocmZhCxClVItU2xqPbJMqkKlapw
zHg5+sdnuXBlMVI+ooh3JQkAIoULAhRAwKMIINBForUujnRVRiGgi1whU8l4
MrHrwD92p8EQoopRoAM/PmwcKo1URWlffsPbN+2BwzW33rxfH79xkxbtOFAK
UAXOS8qT8YXSSGcsVyjV9rXBpA8qFsvs/ozpz/TYRYpIUNdfFy1HOUn58U6q
UCmXKYUN92gNqFQIL0qirXeJQqR8sbYnrgp0coxQoX1/AqEYbc3KZ78AkIw5
b2A0IsUn5YUpxlA3MxlzSFu1XxeDAY0AQ4NuI830dPsDxh8vwYDJmCztJd9r
LwGmAnLQGDAy0AvRN7DQDQkGLQDzhpPuJr8OUaFAppAKEdAc6NlQa2jSPRiR
Yv1kShQ0A0waqkx7mHTTjnihCrQHhjVUGJtggxvOsUxUqcKOQRUqoaDhtaFY
jxhVlEwhVMXrGumkZ8+0ZDyYS//YQ9MPPcRguunoJ2N0VHQ7yoWrVx0AzsZi
RsQC4fra+ID+26b6nduvt7rf1KzYZQUngRPcselYuGo/vwK/OP3EcL3lUgeT
5wu+dX+cWmm/2bjU7NU50VKHeGmpK/cGM9cqT1D11U6qWM9q8sq6I/fo3247
9cwy/tDPu53Wi8dePMXNftLfavB1d8Eo/9kpe41EmjERI9+Wu45kWR6brjci
VDg9+bX60llnp9fZh+7Mu2VCtCq+WG8137EnZmvFFX0zRsCSmISSCuPqja+J
1+5dXwd7/5i1zHrzbirT+f6Syli/wN1rp2q4e0c4PZ7AeXf0RtHFiXeGPoYt
+2FPLktnKFYq6m2j9osmpHg+vv9Yjo77iXoooaVBicOZ9tDp3EWCKpVIJIpT
iZU4BaqUd4QEaIP/AIEENOM/DAH4Lv7/HeU3pGMhbZ/9xzbn/LjgP8l5HokM
IwCJhFAQMgEGuS1zHk/+SHVlC1h3hgEi2MhugA+QYBgEmoJe2QLpG3vVeA+I
UPAEPMBtwnt/JkdCp3pB/iyGlvehIgY1WsPWeEbRWUFiOjWEwKB6qTmSEAKH
GfiB91oT5I+8bw7HX9HYeZaFJvO/g9R/2/Of9Ef9D+Hhrvn/N+v/Vmdka15A
B6wOrXsBra8X//gS8U+4BC1J79+XKGJTqd3G7y+VodExOF6HRgH+e/8fJgJd
878z+P8CroDHhfFkMiSABDwC2Ir/T+gQ/59PQUEKD0aaTHYG1VviT/OF2CJe
PFvjAdJpgXEMUWA8ncUmMCSh0QzIW0RniiUMmu8X4f8L8ESUC8E8MsCDiBCR
0L7+f3Pr7ej/t2W6y/9vf/8/8HP+v9NPV5eaR6TYVRfWLfhu+1lh9PEH1Yl7
3jz7fs3BmbkOBuG3w2pqtwGq9cbhC6OdH96zemPjG4apsSHkbVgcoUcehTFK
OWy7cOCpAU/puqGxdgMgZPODtBLNnvfvJ/vNw+utn/rowmayYapSsrV8Dykr
oeeNqsQnR8adKxop2bOKs3FLYdEZeeaIqUusqmMYP5nVzdYPybwytua2/eLE
H1KtuBdqepaMG+w9Fn8y8krfg0ZDhjr1PcK2W385634htWhFRL3aEne7xP2b
u4b1ewyC5s1GzZ/Pt/LaHLkhZNaNd2YF9k604RuOKkWaQTtOVP5UOGTKnAvB
MxPUO5e9HvBypdFIe7tcIT/uSkuD8v/A/2/kfyP7USlPES9Xofx2VgBt8R/C
```

k5rxn6it6uJ/J+A/SsaDFDJERPB8PpEIEFvhP9wh/Af5EB8gkD7hv8gXr5X7
AIMZDdI1gQCbFaLVAtEgg+YXxWCFxPmz6ABd5B3F0PA+8B8PEYF/iv//A9L+
FNHagm6DLZfG1UGlvcQVlcaiYpkc5eO0SwTyKYklf1QSMBFupKYu+BGJepnJ
ugO1VTa6GB0tHVdo6bhUe/hP0zGoBSjuo1ZeP9XrMm7+knrDUIfaOajutiR+
1V3a4n2njLBOoePccHmXneaWvBeZ59noD3vlpzFMfBpaqZd229hH1D1sCMOD
o7vxgaEUfRl33svcUzD95IYZc0PDjqzPej56ZblXwcnKhcJdgUOTVdhizi77
bUfNr48KjZ0gsN+jCs1aBizgpe9Q7xy1et+m11+dHXyROEVgrS80Of1457vt
tW/N3Q5gfpyvd9ku0U6j/7Vmh5GqICyaIApP8JwVysod4jd9p/skL/eTD49W
SZ2KU4vU5iWxo75POZTx3bDM5IlOg3fnw7OKlDdWzJb1DU3LNfd5GRYb/dB6
q+y8dkzcwoTffGNyxsvum+OjgIQmUgawNjCKSPHBoHr6GF39XrzBx9SKM6eD
a4oSrqXTD71KCoistLQITlMeYfRj+XKQKK/oVeHiy2nwiITFQZutH/DpQeqc
vbIj9dH1R+Zd35uQs2ZJfI11vQnV+q7sweLwNN7g0irbvoHyN18Pm7tpV/GI
rJudnr7/Lv531A6Atv1/uLn/D+DBLv53Av5rf08eAIJ8PAUPCAQAoRX+4zuE
/xREO6i4hE/9fzaBzvJV+7NC8P5MD5AhiobpLHocR8KOY0C+eAbkpWEwfWF/
Ju+L8P/xRATSyis+Hw8QBGSY3E7+P6HR/29uvR39/7ZM/3v9//JP/f8j2qpD
Df5/g3Rp9K676TSPAKzVXpT5r4gANOljVwygpRhA84HZyWMATfnfMMN+DwK0
pxBog/94EoRv7v8TiV3x/87AfxKfCwoIJBIIk0ABv9X4P7Fj+E+GIJBEIjXh
Px3yBTlMLzydGtWw/w/mMDlCjihKxKD6SdgSLzWHqm1j8sX+tI/7/2ACgfJv
5T+fCBJhlEQgg1wuiof57cJ/kADDjQKgufkOCxh0dJzjC4lBMFqAcvKdjFmr
L0ziPSuQYdfIDmX9vIJ7ro5zN3koOb1nZXDiqzQj2PxgRPfvt3692MPesDw3
H0mRFak32LoZXSW5mZVmmHEWzc6t9900ZeP9gYbHTscbvXB5Yuk6d7DnTupR
zS97Jtkejg3IeTE3/yvh5Ko6cXzQpnFhIJ9SYbN5dIplpR4F7337BfKy5v0I
zDy7YUxd/zmPbLdcnxc0VVBa+1w1Y0BGVC/r8WGZ5CdzcoQFugTSlKP97Yfd
t2TaztDc2oZRG848pK4SbVjCjb1sEjbbgDNGrCPKC/ZZ914Usqo/bXj/+OUX
PHUP6r6ca1RTeHPQnKiHZy3STN8T7+wvs31XNpGZbuJJ+1wIYgGsMP1lVUJ3
sWn+UVD3lmNSwPnbj/Z7mvZ4ek1i49fPd4PGduPzY/cLy0eNLY9VYZLKAiTB
K7aM74m3GMg/XX3D/RnboCgzWqWesPS0xb7C07Dt2bQhY0r5C48vzDPpttsi
gMka8temQZdYbY/tqp8Vq0rvxIKvIg7nF71/PmnMyyzd0mn6eVzNU+dvH2w4
c8XBuLdN0YSMHBfhvHnjYjg78aylKLrrTOlyN1qF+PRFl1SrGZNnmU+Wjk05
G+saWZdicn8BeVBe0g/IrbKKiVnCUFr2IltxryU+mccj+kgCvMfeHSVh95o2
ab7u01UQ5f4O5wr9Q1XXhsfVDPUoM4ms3lTHcGaUbinanG12t/ervoYlR5Kr
h0tLLdPDfYcrZxUnxkwetmuDXt7+3WXblV6S9L2mPfpaTl2+Zxt31HGR5UNe
6rSg8xWj7tNcsne/vbDVlTToHJmT3+v2pl599bIm6Cfu3mzn8F4Ve2XiNp9J
uum46AWJRNo1C3J9SyPzv1arHZv5+bP5H2Lz+A8IduV/OoP+40EkIoSHuDAB
5PNhiNSK/gM7RP+REAjFAxDcVP9R2fFa7RflT/XAM1heEEfk1bAPJM6fyRHT
RV5qBssX4rAC4zg0elf+5/8r/9OS9irq/Q20ye8qef/lYGHqhSE33fW2XYq3
y741iGN17M34p8t81v/80LXGI5uTBsmTX/9wqXxJgtgR8w5fwj1m6D/+kuzO
6Afb33grxmekZs4qHlT2s5Fx2gK/SaPTE/LOX+13S3eH2RPTx4v8InPrAkXL
ylIk99TSw5dnJFzRseW4syNMyu5mv9EvOLLubvz9gtCXzwPzw8dfLbccjs/Z
Bc836zMt7fQUPR2x63T7Z2W1eTHhlx9WX3PLN1h2wTf3GLn7o5ndnu0rDsk5
f6S8fm2e++pVgWEvnA8cOOF8U2LX7XRAjH+f8rjYwxY5Pr2nDDk+cKKyxP16
X90sit+xipDua+sYc8N3H/TzXvk57XUUPlpio55RgRXTilZfXJSaFGC1pfdB
s5D0TSuOD5hWuHzt3rPl05bsKSR3Yz8mrzlhH2NpzbMy/gGXajBNnfSE43YR
3jvQRg9iL6+snz6pwoPn/HV94cw+GG9j97uh9im0eWfqutI/HV8+ZH5wcVGI
ChcpU+Ea17N2VAJt7f8Ggeb5HyIJALr4/3eUdgnod02jL3/+d9i/P/5K/hdP
AqGu+d8J9D8eoUAQl8jnAQJAIMC3ov/xlA7R/6iAyEX4MNo0/wsxhByRL8AQ
BYkY1ECAzmSDDKr2mMWQ0GkhIJ3qAdGpQVFsyZex/xvPhUEYBSGQS9C+ahBs
5/xvM+vtmf9tw3RXSO0fCKl1/Qf8jwOzk+d/Ozbz+2fzv1Dz+B9M6Mr/dgb+

```
   U0gIRERhCpcEESEuvrX9X1CH8B9PoghgLvRJ/lfEjqNr+CI2k41ns4Ki2ZpI
   DV0UAtBpvgS6hgcxRNEgW+MnZmi8voj8L0rmC4goyENhPoiS2+n/Xx/zv83N
   d+V///4YpK+5sXlQ7qprG9+kHLvb+/jC9FWz3JLOJhz8buf0sYWvLZJEonyC
   onxfckRlwZiXgfOhedWnYyUrvZX7qZm93n+1doZqlp1q6uV6z33LdJKzkqYP
   XJmVbNpj1sOoBxllEuvBI3PDCzIy3dZXWA8o8zwmWzddMGi4TsIK0Q690YnP
   fe4s8oUf1bszJ+a9mHln9LAx9Zeu9qrmHYT9LHjGEtOMXzYpKML56DjqwWww
   Ir5oQ/YavXqPIb1rn7yknZzvWTE0bh1ra/+le7utu017fGbLRYtEqxkT5h+0
   BYvZN+qlGT8sujc5Z9pwt0FW7lf3RZwKD0vpbpC8fWeVTnWeM2XY1YT0zXNv
   H9hlEP65IGTqdxYx6wV9Dpw6cfj92UUTM5MCkoyzd7LmbH8q32LdJxeufmUt
   sPcOcre44uI3qPbepldwzo61P+7TDoV+BykDp/YaZ/o0XV9tPouReO1AcX5N
   iGX8pMeM2iGeJC/KxOeVrAyG8V+bBl1itcPFqjQwb7Dj7oQ1dUkbQorP8yfW
   2htNyt+6Ubbo7LJ4KGzr0XdrlG9rWYzDLpHwxutqG/a3dZG8OBtNcrHN0J6U
   GJOcoYsxop0TH+5zCPR+s55IvcY/bH7MOLr+iSSh3m2L46I96u+fWq3BRhVd
   OnfN5O2LPhkO/E3DgkqC7g1L7VNSdYD50x2fKsPb3zn+/CM3K3ZGMOmW7tgz
   KbdrHEu+pdxzwgRnTutflAO+vbloQNVAv8gZS/IZw3NPXPLTF11OSZsyflKV
   Jj09cwhSOj5reG1B/iNJoMFKNWJa7rx+dXbhbOMk89Lc/7RvxzQMAgEARRkw
   wNSEMOLlFDBVBGMnFhJsYAABZ4LuJUwkJZCgoQQDdcB7Gv768/VRHG01vNNt
   emZ7DOdvjHOoX11ffrLl2/wL8wbDIgAAAAAAAAAAAJBchjiJbgB4AAA=
   -- END MESSAGE ARCHIVE --
```

Authors' Addresses

Cullen Jennings
Cisco Systems
170 West Tasman Drive
Mailstop SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 421 9990
Email: fluffy@cisco.com

Kumiko Ono
Columbia University

Email: kumiko@cs.columbia.edu

Robert Sparks
Tekelec
17210 Campbell Road
Suite 250
Dallas, TX 75252
USA

Email: rjsparks@estacado.net

Brian Hibbard (editor)
Tekelec
17210 Campbell Road
Suite 250
Dallas, TX 75252
USA

Email: brian@estacado.net