SIPCORE Working Group
Internet-Draft
Intended status: Informational
Expires: November 25, 2010

C. Holmberg
Ericsson

May 24, 2010

Indication of support for keep-alive
draft-ietf-sipcore-keep-03.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 are not implicitly negotiated as part of the SIP Outbound negotiation.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Section 3.5 of SIP Outbound [RFC5626] defines two keep-alive mechanisms. Eventhough the keep-alive mechanisms are separated from the rest of the SIP Outbound mechanism, it is currently not possible to explicitly negotiate usage of the keep-alive mechanisms, since usage of keep-alives in most cases are implicitly negotiated as part of the SIP Outbound negotiation.

However, there are SIP Outbound use-cases where the usage of keep-alives are not implicitly negotiated as part of the SIP Outbound negotiation. In addition, there are cases where SIP Outbound is not supported, where it cannot be applied, but where there is still a need to be able to negotiate usage of keep-alives. For those cases, a mechanism to explicitly negotiate the usage of keep-alives is needed.

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

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

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

In some cases a User Agent Client (UAC) does not register itself before it establishes a session, but where it still needs to be able to establish a session and send keep-alives in order to maintain NAT bindings open during the duration of the call. A typical example is an emergency calls, where a registration is not always required.

1.2. Use-case: SIP Outbound not supported

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

1.3. Use-case: SIP dialog initiated Outbound flows

SIP Outbound allows the establishment of flows using initial SIP dialog requests. As specified in [RFC5626], the usage keep-alives are not implicitly negotiated for such flows. Therefor there is a

need to be able to explicitly negotiate the usage of the keep-alives.

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 [RFC5626], 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 entity as the Authoritative Proxy.

Keep-alives: Refers to keep-alive messages as defined in SIP Outbound [RFC5626].

"keep" parameter: A SIP Via header field parameter that a SIP entity can insert in its Via header field of a request to explicitly indicate willingness to send keep-alives. A SIP entity can add a "yes" parameter value to a "keep" parameter in the top-most Via header field of a recieved SIP request, to indicate willingness to receive keep-alives from the adjacent downstream SIP entity (associated with the top-most Via header field of the received request) from which it received the request.

SIP entity: SIP User Agent (UA), or proxy, as defined in [RFC3261].

4. User Agent and Proxy behavior

4.1. General

This section describes of SIP UAs and proxies negotiate the sending or receiving of keep-alives within a registration and within a dialog. It also describes which types of SIP requests and responses can be used in order to negotiate the sending and receiving of keep-alives, and the lifetime of the negotiated keep-alives.

SIP requests are used by SIP entities to indicate willingness to send keep-alives towards the adjacent upstream SIP entity. The associated responses are used by SIP entities to indicate willingness to receive

keep-alives. The procedures to indicate willingness to send and received keep-alives are identical for UAs and proxies.

NOTE: Since there are SIP entities that already use CRLF keep-alives, and SIP entities are expected to be able to receive those, this specification does not forbid the sending of CRLF keep-alives towards an SIP entity even if it has not indicated willingess to receive keep-alives using the "keep" parameter. However, the "keep" parameter is still important in order for a SIP entity to indicate that it supports CRLF keep-alives, so that the adjacent SIP entity does not use other mechanisms (e.g. short registration refresh intervals) in order to make sure the NAT bindings are kept open.

4.2. Scope and duration of keep-alives

4.2.1. General

The sending and receving of keep-alives can be negotiated within a registration, or within a dialog. The scope of the negotiated keep-alives depends on what SIP request methods are used for the keep-alive negotiation.

The sending and receiving of keep-alives can be negotiated when a registration or dialog is initiated, or later during the registration or dialog. However, once a SIP entity has negotiated the sending of keep-alives within a registration or dialog, it can not re-negotiate the sending of keep-alives within the same registration or dialog. Likewise, once a SIP entity has indicated willingness to receive keep-alives within a registration or dialog, it MUST NOT indicate willingness to receive keep-alives in a response to a subsequent request within that registration or dialog.

A SIP entity that has indicated willingess to receive keep-alives within a dialog can still, in a subsequent request within the dialog, indicate willingness to send keep-alives within the same dialog. Likewise, a SIP entity that has negotiated the sending of keep-alives within a dialog can in a response to a subsequent request indicate willingness to receive keep-alives within the same dialog.

4.2.2. Keep-alives within registration

SIP entities use the REGISTER method in order to negotiate the sending and reciving of keep-alives within a registration. The keep-alives can be negotiated when the registration is established, or later within the registration. Once negotiated, the keep-alives are sent during the lifetime of the registration, until it is terminated.

In case a SIP entity establishes multiple registration flows

[RFC5626], the sending and receiving of keep-alives is done separately for each individual registration flow. The SIP entity MUST NOT send keep-alives on registration flows where it has not received an indicator that the adjacent upstream SIP entity is willing to receive keep-alives withing that registration flow.

4.2.3. Keep-alives within dialog

SIP entities use a initial request for a dialog, or a mid-dialog target refresh request [RFC3261] in order to negotiate the sending and reciving of keep-alives within a dialog. The keep-alives can be negotiated when the dialog is established, or later within the dialog. Once negotiated, the keep-alives are sent during the lifetime of the dialog, until it is terminated.

Since an ACK request does not have an associated response, it can not be used to negotiate the sending and reciving of keep-alives. Therefor a SIP entity MUST NOT insert a "keep" parameter in its Via header field of an ACK request. If a SIP entity receives a "keep" parameter in an ACK request, it MUST ignore the parameter.

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

As defined in [RFC5626], a SIP entity that supports the sending of keep-alives must act as a Session Traversal Utilities for NAT (STUN) client [RFC5389]. The SIP entity must support the amount of STUN which is required to apply the STUN keep-alive mechanism defined in [RFC5626], and it must support the CRLF keep-alive mechanism defined in [RFC5626].

When a SIP entity sends or forwards a request, if it wants to negotiate the sending of keep-alives within the registration (in case of a REGISTER request) or dialog (in case of an initial request for a dialog, or a mid-dialog target refresh request), and if it has not previously negotiated the sending of keep-alives within the same registration or dialog, it MUST insert a "keep" parameter in its Via header field of the request.

When the SIP entity receives the associated response, if the "keep" parameter in its Via header field in the response contains a "yes" parameter value, it MUST start to send 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 reuqests (if the keep-alive negotiation is for a dialog). Subsequent mid-dialog requests are addressed based on the dialog route set.

If the response contains a Flow-Timer header field, the SIP entity MUST remove the header field before it forwards the response towards another SIP entity.

When a SIP entity is about to send a keep-alive, if the SIP entity at the same time is also about to send or forward a SIP request within the same registration or dialog, for which the keep-alive is to be sent, the SIP entity MAY choose not to send the keep-alive, as the SIP request will perform the same keep-alive action.

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

NOTE: If a SIP entity inserts a "keep" parameter in its Via header field of an INVITE request, and it receives multiple responses (provisional or final) associated with the request, as long as at least one of the responses, for a specific dialog, contains a "keep" parameter with a "yes" value it is seen as an indication that the adjacent upstream SIP entity is willing to receive keep-alives within the dialog.

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

As defined in [RFC5626], a SIP entity that supports receiving of keep-alives must act as a STUN server [RFC5389]. The SIP entity must support the amount of STUN which is required to apply the STUN keep-alive mechanism defined in [RFC5626], and it must support the CRLF keep-alive mechanism defined in [RFC5626].

When a SIP entity receives request that can be used in order to negotiate the sending and receiveing of keep-alives, the top-most Via header field of the request contains a "keep" parameter, and the SIP entity has not previously indicated willingess to receive keep-alives from the adjacent downstream SIP entity within the registration (in case of a REGISTER request) or dialog (in case of a initial request for a dialog, or a mid-dialog target refresh request), if it is willing to receive keep-alives from the adjacent downstream SIP entity it MUST add a "yes" parameter value to the "keep" parameter of

the top-most Via header field of the request, before forwarding the request or creating a response. In addition, the SIP entity MAY insert a Flow-Timer header field [RFC5626] in the associated response, which indicates the recommended keep-alive frequency for the registration or dialog.

5. Keep-alive frequency

If a SIP entity receives a SIP response, where its Via header field contains a "keep" parameter with a "yes" value, also contains a Flow-Timer header field [RFC5626], according to [RFC5626] the SIP entity MUST send keep-alives at least as often as this number of seconds, and if the SIP entity uses the server-recommended keep-alive frequency it should send its keep-alives so that the interval between each keep-alive israndomly distributed between 80% and 100% of the server-provided time.

If the SIP entity does not receive a Flow-Timer header field from the edge proxy, it can send keep-alives at its discretion. [RFC5626] provides additional guidance on selecting the keep-alive frequency in case a Flow-Timer header field is not received.

OPEN ISSUE: It has been suggested that, instead of using the Flow-Timer header field in order to provide the recommented keep-alive frequency value, the value would be added as a parameter to the "keep" parameter, instead of the "yes" value.

6. Overlap with connection reuse

The connect-reuse specification [I-D.ietf-sip-connect-reuse] specifies how to use connection-oriented transports to send requests in the reverse direction. SIP entity A opens a connection to entity B in order to send a request. Under certain conditions entity B can reuse that connection for sending requests in the backwards direction to A as well. However, the connect-reuse specification does not define a keep-alive mechanism for this connection.

The mechanism specified in this draft is thus orthogonal to the purpose of connection reuse. An entity that wants to use connection-reuse as well as indicate keep-alive mechanism on that connection will insert both the "alias" parameter defined in [connect-reuse] as well as the "keep" parameter defined in this memo. Inserting only one of these parameters is not a substitute for the other. Thus, while the presence of a "keep" parameter will indicate that the enity supports keep-alives in order to keep the connection open, no inference can be drawn on whether that connection can be used for

requests in the backwards direction.

7. Examples

7.1. General

This section shows example flows where the usage of keep-alives is negotiated between different SIP entities, within a registration or within a dialog.

7.2. Keep-alive negotiation: UA-proxy within registration

The figure 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) supports the keep-alive mechanism, and is willing to receive keep-alives from Alice during the registration, so it adds a "yes" value to the "keep" parameter in the Via header field of the UAC, before it forwards the request towards the registrar.

When P1 receives the associated response, it inserts a Flow-Timer header field, with a recommended keep-alive frequency interval of 30 seconds, in the response, 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 within the registration. For the duration of the registration, the UAC then sends periodic keep-alives (in this example using the STUN keep-alive technique) towards P1, using the recommended keep-alive frequency indicated in the Flow-Timer header field of the response.

```
     Alice                        P1                      REGISTRAR
       |                          |                           |
       |--- REGISTER------------->|                           |
       |    Via: UAC;keep         |                           |
       |                          |--- REGISTER-------------->|
       |                          |    Via: P1                |
       |                          |    Via: UAC;keep=yes      |
       |                          |                           |
       |                          |<-- 200 OK ----------------|
       |                          |    Via: P1                |
       |                          |    Via: UAC;keep=yes      |
       |<-- 200 OK ---------------|                           |
       |    Via: UAC;keep=yes     |                           |
       |    Flow-Timer: 30        |                           |
       |                          |                           |
       |                          |                           |
       |                   *** Timeout ***                    |
       |                          |                           |
       |=== STUN request ========>|                           |
       |<== STUN response ========|                           |
       |                          |                           |
       |                   *** Timeout ***                    |
       |                          |                           |
       |=== STUN request ========>|                           |
       |<== STUN response ========|                           |
       |                          |                           |
```

Figure 1: Example call flow

7.3. Keep-alive negotiation: UA-proxy within dialog

The figure 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. P1 also supports the keep-alive mechanism, and is willing to receive keep-alives from Alice during the dialog, so it adds a "yes" value to the "keep" parameter in the Via header field of Alice, before it forwards the request towards Bob.

When P1 receives the associated response, it inserts a Flow-Timer header field, with a recommended keep-alive frequency interval of 30 seconds, in the response, before it forwards the response towards Alice.

When Alice receives the response, she determines from its Via header field that P1 is willing to receive keep-alives within the dialog. For the duration of the dialog, she then sends periodic keep-alives (in this example using the STUN keep-alive technique) towards P1, using the recommended keep-alive frequency indicated in the Flow-Timer header field of the response.

```
     Alice                        P1                         Bob
       |                          |                           |
       |--- INVITE -------------->|                           |
       |    Via: UAC;keep         |                           |
       |                          |--- INVITE --------------->|
       |                          |    Via: P1                |
       |                          |    Via: UAC;keep=yes      |
       |                          |    Record-Route: P1       |
       |                          |                           |
       |                          |<-- 200 OK ----------------|
       |                          |    Via: P1                |
       |                          |    Via: UAC;keep=yes      |
       |                          |    Record-Route: P1       |
       |<-- 200 OK ---------------|                           |
       |    Via: UAC;keep=yes     |                           |
       |    Flow-Timer: 30        |                           |
       |    Record-Route: P1      |                           |
       |                          |                           |
       |--- ACK ----------------->|                           |
       |                          |                           |
       |                          |--- ACK ------------------>|
       |                          |                           |
       |                   *** Timeout ***                    |
       |                          |                           |
       |=== STUN request ========>|                           |
       |<== STUN response ========|                           |
       |                          |                           |
       |                   *** Timeout ***                    |
       |                          |                           |
       |=== STUN request ========>|                           |
       |<== STUN response ========|                           |
       |                          |                           |
       |                          |                           |
       |--- BYE ----------------->|                           |
       |                          |                           |
       |                          |--- BYE ------------------>|
       |                          |                           |
       |                          |<-- 200 OK ----------------|
       |                          |                           |
```

Figure 2: Example call flow

7.4. Keep-alive negotiation: UA-UA within dialog

The figure 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, and it does not indicate willingness to receive keep-alives from Alice.

When Alice receives the response, she determines from her Via header field that P1 is not willing to receive keep-alives 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 from Alice during the dialog, so he adds a "yes" value to the "keep" parameter in the Via header field of Alice, before he creates and sends a response towards her. Bob also inserts a Flow-Timer header field in the response, with a recommended keep-alive frequency interval of 30 seconds.

When Alice receives the response, she determines from her Via header field that Bob is willing to receive keep-alives from her within the dialog. For the duration 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 in the Flow-Timer header field of the response.

```
     Alice                        P1                         Bob
       |                          |                           |
       |--- INVITE -------------->|                           |
       |    Via: UAC;keep         |                           |
       |                          |--- INVITE --------------->|
       |                          |    Via: P1                |
       |                          |    Via: UAC;keep          |
       |                          |                           |
       |                          |<-- 200 OK ----------------|
       |                          |    Via: P1                |
       |                          |    Via: UAC;keep          |
       |<-- 200 OK ---------------|                           |
       |    Via: UAC;keep         |                           |
       |                          |                           |
       |                                                      |
       |--- ACK --------------------------------------------->|
       |                                                      |
       |--- UPDATE ------------------------------------------>|
       |    Via: UAC;keep                                     |
       |                                                      |
       |<-- 200 OK ------------------------------------------>|
       |    Via: UAC;keep=yes                                 |
       |    Flow-Timer: 30                                    |
       |                                                      |
       |                                                      |
       |                   *** Timeout ***                    |
       |                                                      |
       |=== STUN request ====================================>|
       |<== STUN response ====================================|
       |                                                      |
       |                   *** Timeout ***                    |
       |                                                      |
       |=== STUN request ====================================>|
       |<== STUN response ====================================|
       |                                                      |
       |                                                      |
       |--- BYE --------------------------------------------->|
       |                                                      |
       |<-- 200 OK -------------------------------------------|
       |                                                      |
```

Figure 3: Example call flow

8. Grammar

This specification defines a new Via header field parameter, "keep". The grammar includes the definitions from [RFC5626].

The ABNF [RFC5234] is:

```
via-params =/ keep

keep       = "keep" [ EQUAL "yes" ]
```

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 [RFC5626]. 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

This specification does not introduce security consideritions in additions to those specified 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 and John Elwell for their comments on the list. Thanks to Vijay Gurbani for providing text about the relationship with the connect-reuse specification.

12. References

12.1. 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.

12.2. Informative References

[I-D.ietf-sip-connect-reuse] Gurbani, V., Mahy, R., and B. Tate, "Connection Reuse in the Session Initiation Protocol (SIP)", draft-ietf-sip-connect-reuse-14 (work in progress), August 2009.

Author's Address

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: christer.holmberg@ericsson.com