Indication of support for keep-alive
draft-ietf-sipcore-keep-08.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-alarms 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 April 22, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ............................................. 3
   1.1. Use-case: Dialog from non-registered UAs .............. 3
   1.2. Use-case: SIP Outbound not supported .................. 3
   1.3. Use-case: SIP dialog initiated Outbound flows ........ 3
2. Conventions .............................................. 4
3. Definitions .............................................. 4
4. User Agent and Proxy behavior ............................. 4
   4.1. General .............................................. 4
   4.2. Lifetime of keep-ales ................................. 5
      4.2.1. General ........................................... 5
      4.2.2. Keep-alives associated with registration .......... 5
      4.2.3. Keep-alives associated with dialog ............... 6
   4.3. Behavior of a SIP entity willing to send keep-alives ... 6
   4.4. Behavior of a SIP entity willing to receive keep-alives . 7
5. Keep-alive frequency ...................................... 8
6. Connection reuse ......................................... 9
7. Examples .................................................. 9
   7.1. General .............................................. 9
   7.2. Keep-alive negotiation associated with registration:
      UA-proxy .............................................. 9
   7.3. Keep-alive negotiation associated with dialog: UA-proxy 10
   7.4. Keep-alive negotiation associated with dialog: UA-UA ... 12
8. Grammar ................................................... 14
9. IANA Considerations ....................................... 15
   9.1. keep ................................................ 15
10. Security Considerations .................................. 15
11. Acknowledgements ....................................... 16
12. Change Log ............................................. 16
13. References ............................................. 16
   13.1. Normative References ................................. 16
   13.2. Informative References ............................... 17
Author’s Address ........................................... 17
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-ales can be implicitly negotiated as part of the SIP Outbound negotiation.

However, there are SIP Outbound use-cases where usage of keep-ales 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, [RFC5626] 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 all SIP entities that need to be able to negotiate the usage 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 [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 [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: The keep-alive messages 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 towards its adjacent downstream SIP entity. A SIP entity can also insert the header field 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 [RFC3261].

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 the same 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.

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 \[RFC5626\], 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
deefined in \[RFC5626\], and it must support the CRLF keep-alive
mechanism defined in \[RFC5626\]. \[RFC5626\] 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 its Via header
field of the request to indicate willingness to send keep-alives.

When the SIP entity receives the associated response, if the "keep"
parameter in its Via header field of the response contains a "keep"
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 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 its 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 its 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 its Via header field of a subsequent SIP request associated with the dialog, in case the new adjacent SIP downstream 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 [RFC5626], 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 [RFC5626], and it must support the CRLF keep-alive mechanism defined in [RFC5626].

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 can contain a recommended keep-alive frequency, given in seconds, or a zero value.
When a SIP entity indicates willingness to receive keep-alives in a response to an INVITE request, it MUST insert a "keep" parameter in at least one reliable response to the request. The SIP entity MAY insert an identical "keep" parameter value in other responses to the same request. The SIP entity MUST NOT insert "keep" parameters with differing values in responses to a single INVITE 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 its 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. [RFC5626] 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 [RFC5626], the receiving SIP entity may consider a connection to be dead in such situations.

If a SIP entity that uses the "keep" parameter 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, without forcing entities to re-write the value of the Flow-Timer header field.

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 MUST 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 [RFC5923] 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.

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

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) 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.

<table>
<thead>
<tr>
<th>Alice</th>
<th>P1</th>
<th>REGISTRAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>--- REGISTER------------&gt;</td>
<td>--- REGISTER--------------------------&gt;</td>
<td></td>
</tr>
<tr>
<td>Via: Alice;keep</td>
<td>Via: P1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Via: Alice;keep</td>
<td></td>
</tr>
<tr>
<td>--- 200 OK --------------</td>
<td>--- 200 OK --------------------------</td>
<td></td>
</tr>
<tr>
<td>Via: Alice;keep=30</td>
<td>Via: P1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Via: Alice;keep</td>
<td></td>
</tr>
<tr>
<td>*** Timeout ***</td>
<td>*** Timeout ***</td>
<td></td>
</tr>
<tr>
<td>=== STUN request ======&gt;</td>
<td>&lt;= STUN response =========</td>
<td></td>
</tr>
<tr>
<td>*** Timeout ***</td>
<td>*** Timeout ***</td>
<td></td>
</tr>
<tr>
<td>=== STUN request ======&gt;</td>
<td>&lt;= STUN response =========</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Example call flow

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

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, 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

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, 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 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 Bob, using the recommended keep-alive frequency indicated by the "keep" parameter value.
Figure 3: Example call flow

8. Grammar

This specification defines a new Via header field parameter, "keep".

The ABNF [RFC5234] is:
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:

<table>
<thead>
<tr>
<th>Header Field</th>
<th>Parameter Name</th>
<th>Predefined Values</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Via</td>
<td>keep</td>
<td>No</td>
<td>[RFCXXXX]</td>
</tr>
</tbody>
</table>

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 are required to use a dedicated connection reuse mechanism, in conjunction with the keep-alive negotiation mechanism.

Unless SIP messages are integrity protected, 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.

Downstream SIP entities can modify Via header fields identifying other SIP entities, and cause keep-alives to be sent (at high rates) to entities that do not support the keep-alive mechanism. SIP entities can prevent this, when a SIP response is received, by examining their own Via header field to determine that downstream entities have not added a "keep" parameter or set an existing "keep" parameter to a value not supported by the implementation.
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 and Adam Roach 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-07
  o Last paragraph of section 4.2.2 removed
  o Reference correction

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

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

13. References

13.1. Normative References


13.2. Informative References


Author’s Address

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: christer.holmberg@ericsson.com