Internet Engineering Task Force                                   SIP WG
Internet Draft                                                    S.Donovan,J.Rosenberg
draft-ietf-sip-session-timer-04.txt                               dynamicsoft
November 22, 2000                                                Expires: May 2001

The SIP Session Timer

STATUS OF THIS MEMO

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

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.

Abstract

This document proposes an extension to the Session Initiation Protocol (SIP). This extension allows for a periodic refresh of SIP sessions through a re-INVITE. The refresh allows both user agents and proxies to determine if the SIP session is still active. The extension defines a new general header, Session-Expires, which conveys the lifetime of the session.

1 Introduction

The Session Initiation Protocol (SIP) [1], does not currently define a keepalive mechanism. The result is that call stateful proxies will not always be able to determine whether a call is still active or not. For instance, when a user agent fails to send a BYE message at the end of a session, or the BYE message gets lost due to network
problems, a call stateful proxy will not know when the session has ended. In this situation, the call stateful proxy will retain state for the call and has no deterministic method of determining when the call state information no longer applies.

To resolve this problem, this extension defines a keepalive mechanism for SIP sessions. UAs send periodic re-INVITEs to keep the session alive. The interval for the re-INVITEs is determined through a negotiation mechanism defined here. If a re-INVITE is not received before the interval passes, the session is considered terminated. Both UAs are supposed to send a BYE, and call stateful proxies can remove any state for the call.

INVITE is used as a refresh, as opposed to another method, to allow sessions to be recovered after a crash and restart of one of the UAs. It makes SIP sessions soft state.

This refresh mechanism has additional applications. For the same reasons a call stateful proxy server would like to determine whether the session is still active, a user agent would like to make this determination. This determination can be made at a user agent without the use of SIP level mechanisms; for audio sessions, periodic RTCP packets serve as an indication of liveness [2]. However, it is desirable to separate SIP call liveness from the details of the particular session.

Another important application of the session timer is in NAT and firewall control [3]. In order for SIP to flow through a NAT or firewall, holes and/or address bindings must be dynamically created to allow the media for the session to flow. These holes and/or address bindings represent state which must be eventually removed. Relying on a BYE to trigger the removal of state, besides being unreliable, introduces a potential denial of service attack.

This document proposes an extension to SIP that defines a session expiration mechanism. Periodic refreshes, through re-INVITEs, are used to keep the session active. The extension is sufficiently backwards compatible with SIP that it works so long as either one of the two participants in a call leg understand the extension. A new general header, the Session-Expires header, is defined. It conveys the expiration time of the session.

2 Protocol Overview

UACs which support the session timer extension defined here MUST include a Supported header in all requests, excepting ACK, containing the option tag "timer" [4]. When a UAC makes a call, it MAY include a
Session-Expires header in the INVITE request. The presence of the Session-Expires header indicates that the UAC wishes to use the session timer for this call. Its value indicates the desired expiration time of the session.

Proxies on the signaling path may have their own requirements for the refresh interval of the session. If the Supported header in the request lists the option tag "timer", a proxy can be certain the UAC understands the session timer. In this case, if no Session-Expires was present, the proxy can insert one if it so desires. If one was present, the proxy can lower, but not increase, the expiration time of the session. The proxy remembers the value of Session-Expires it placed into the request, and also remembers that the UAC supported session timer. The UAC will ultimately be responsible for sending the refreshes for this call leg.

If the Supported header was absent from the request, or was present but didn't include the tag "timer", the proxy knows the UAC cannot generate refreshes, but the called party may be able to. If no Session-Expires was present, the proxy can insert one if it so desires. If one was present, the proxy can lower, but not increase, the expiration time of the session. The proxy MUST remember the value of Session-Expires it placed into the request, and also MUST remember that the UAC did not support the session timer. The UAS may be responsible for sending the refreshes for this call leg. If the proxy wishes to insist that the call is only established if the UAS supports session timer, it MAY insert a Require header into the request, with the value "timer", although this is NOT RECOMMENDED, since it eliminates the possibility of call establishment if the UAS does not support session timers.

Eventually, the initial INVITE reaches the UAS. There are two cases - the UAS supports session timer, or it does not. If it does, and the response is otherwise acceptable to the UAS, the UAS MUST copy the Session-Expires into the 200 class response, or MAY add one into the response if none was present in the request. They UAS MAY reduce the session timer in the 200 class response. If the request also contained the Supported header with the value "timer", the UAS knows that the UAC can do refreshes. To make sure the UAC is aware it must actually do them, the UAS MUST add a Require header to the 200 class response, with the tag "timer", when the request contains a Require header with tag "timer". The UAS does not perform the refreshes in this case. If the request did not contain the Supported header with the value "timer", the UAS knows the UAC cannot perform refreshes. So, it assumes the responsibility. In this case, it MUST add the value of the Session-Expires to the response, but it MUST NOT add a Require header with value "timer". This is because the UAC does not support the extension; the UAS cannot insist on its usage at the UAC,
which is what a Require header in the response would accomplish.

If the UAS does not support session timer, it behaves as a normal UAS. It will, in this case, neither insert Session-Expires in the response nor a Require header with value "timer".

This response travels backwards through the proxies. When it reaches a proxy which remembers that it asked for session timer, the proxy examines the response. If the response did not contain a Session-Expires header, but the proxy remembers that the UAC supported session timer, the proxy knows that the UAC supports session timer, but the UAS does not. So, it inserts the Session-Expires header into the response and also adds a Require header with a value of "timer". If the response the proxy received did have a Session-Expires header, but no Require header with value "timer", the proxy knows that the UAS understands session timer, but not the UAC. It simply forwards this request upstream. If the proxy gets a response without Session-Expires, and the proxy remembers that the UAC did not support session timer, the proxy knows that session timer cannot be used, since neither UAS nor UAC support it. Finally, if the response contains the Session-Expires and Require header with the value "timer", the proxy knows that both UAC and UAS support session timer, and that the UAC will be performing refreshes.

The response then arrives at the UAC. If it contains a Require header with the value "timer", the UAC knows it is responsible for refreshes. The response will also contain a Session-Expires header, and the value of that header is used as the interval for refreshes. If the response contains no Session-Expires header or Require header, but the UAC had inserted a Session-Expires header into the request (since it wanted a session timer), the UAC will also be responsible for refreshes.

The UAC then ACKs the INVITE. The Session-Expires MUST NOT be included in the ACK.

If the calling UA is supposed to perform refreshes and wishes to keep the session alive, it MUST send a re-INVITE before the expiration time. This re-INVITE MAY contain a Session-Expires header. The processing of this re-INVITE by proxies and UAS is identical to that of the initial INVITE, excepting that processing may also take place for non-200 final responses.

If the called UA is supposed to perform refreshes and wishes to keep the session alive, it MUST send a re-INVITE before the expiration time. This re-INVITE MAY contain a Session-Expires header. The processing of this re-INVITE by proxies and UAS is identical to that of the initial INVITE, excepting that processing may also take place
Session Timer

November 22, 2000

for non-200 final responses.

If the calling UA or the called UA is not performing refreshes, and
does not receive a re-INVITE refreshing the session before the
session expires, they SHOULD send a BYE to terminate the call. If a
refreshing UA does not receive a response to the re-INVITE used to
refresh the session, it SHOULD send a BYE to terminate the call.
Similarly, if a proxy doesn’t receive a re-INVITE before expiration
of the session, it MAY remove associated call state, and MAY free any
resources associated with the call. Unlike the UA, it MUST NOT send a
BYE.

3 Session-Expires Header Field Definition

The Session-Expires header conveys the expiration time for a SIP
session described in the body of the message. It is placed only in
INVITE requests, and is allowed in any final response to an INVITE.
Like the SIP Expires header, it can contain either an absolute time
or a delta-time. If it contains an absolute time, this time indicates
the time at which a proxy or UA may safely destroy any state
associated with the call. If it contains a delta time, the expiration
time of the session is defined as that delta plus the time at which
the header is observed in a final response. For example, if a UAS
generates a 200 OK response to a re-INVITE that contained a Session-
Expires header with a value of 3600, the UAS computes the expiration
time of the session as one hour after the time when the 200 OK was
sent. For each proxy, the expiration time is one hour after the time
when the 200 OK was received or sent (assuming these two are
sufficiently close together). For the UAC, the expiration time is one
hour after the receipt of the final response. Any entity that inserts
a Session-Expires header with an absolute time MUST also insert a
Date header into the message. This ensures proper operation for
devices that do not have access to absolute time.

There is no absolute minimum session refresh interval. However, 30
seconds is RECOMMENDED. In other words, SIP entities MUST be prepared
to handle session refresh intervals of any duration, but entities
that insert the Session-Expires header SHOULD NOT choose times that
result in intervals less than 30 seconds.

The syntax of the Session-Expires header is:

Session-Expires = ("Session-Expires" | "x") ":" (SIP-date |
delta-seconds)
Note that a compact form, the letter ‘x’, has been reserved for Session-Expires. Both SIP-Date and delta-seconds are defined in Section 6.20 of RFC 2543 [1].

Table 1 is an extension of tables 4 and 5 in [1] for the Session-Expires header:

<table>
<thead>
<tr>
<th>Header</th>
<th>R</th>
<th>n</th>
<th>h</th>
<th>-</th>
<th>-</th>
<th>o</th>
<th>-</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session-Expires</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Session-Expires</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where enc e-e ACK BYE CAN INV OPT REG

Table 1: Summary of header fields. "o": optional "-": not applicable, "R": request header, "r": response header, "g": general header, "*": needed if message body is not empty. A numeric value in the "type" column indicates the status code the header field is used with.

4 UAC Behavior

A UAC which supports the session timer extension defined here MUST include a Supported header in each request (excepting ACK), listing the option tag "timer" [4]. It MUST do so even if the UAC is not requesting keepalives for the call.

If the UAC wishes to request keepalives for this call, it MUST include a Session-Expires in the INVITE request used to initiate the call. The value of this header indicates the time when the UAC will consider the call expired if no refresh is sent. If the request is being authenticated, the Session-Expires header MUST appear before the Authorization or Proxy-Authorization headers.

The UAC MAY include a Require in the request with the value "timer" to indicate that the UAS must support the session timer to participate in the session. In addition, the UAC MAY include a Proxy-Require header in the request with the value "timer" to indicate that proxies must support session timer in order to correctly process the request. However, usage of either Require or Proxy-Require by the UAC is NOT RECOMMENDED. They are not needed, since the extension works even when only the UAC supports the extension.

When the response to the initial INVITE request arrives, it may or may not contain a Session-Expires header, and may or may not contain a Require header with the value "timer". UACs MUST be prepared to receive a Session-Expires header in a response even if none were present in the request.
Table 2 describes the behavior of the UAC (in terms of handling refreshes) after receiving a 200 class response to an initial INVITE, or any final response to a re-INVITE, based on the headers present in that response.

<table>
<thead>
<tr>
<th>Require</th>
<th>Session-Expires</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
<td>If request contained Session-Expires, UAC is performing refreshes. Otherwise, do nothing.</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>UAS is performing refreshes. Do nothing.</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>Should never happen. Do nothing.</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>UAC MUST perform refreshes.</td>
</tr>
</tbody>
</table>

Table 2: UAC Behavior

If the UAC must refresh, it follows the following procedure. The UAC computes the expiration time of the session. This is based on the value of the Session-Expires in the response. If there is no Session-Expires in the response, but the UAC is refreshing, the value from the request is used (this only happens when the UAC wants session timers, but none of the proxies, nor the UAS, want or support it). If the Session-Expires contains an absolute time, that is the time of expiration. If it contains a delta-time, the expiration time is the time of reception of the response plus that delta time. Let the difference in time between the reception of the response and the session expiration time be called the refresh interval. Note that this expiration applies only to the call leg associated with the response. It is explicitly allowed for there to be differing session timers (or none at all) on differing call legs, in the case where there are multiple 2xx OK responses to an initial INVITE with different tags in the To field.

If UA wishes to continue with the session beyond the expiration, it MUST generate a refresh before the expiration time. It is RECOMMENDED that this refresh be sent once half the refresh interval has elapsed. This refresh is accomplished by sending a re-INVITE request on the given call leg. Sending of the refresh (in terms of this extension), and processing the response are exactly identical to the rules above, excepting that the processing applies to non-200 final responses, as well as 200 class final responses.

A UA MAY use the refreshing re-INVITE as a normal SIP re-INVITE; that is, this re-INVITE MAY contain an updated session description. In the case where the re-INVITE contains an updated session description, the session description MUST somehow indicate that it has changed. In the case of SDP [5], this is accomplished by using a different value for...
If the refreshing re-INVITE is used solely for refreshing, it MUST still contain a session description, unchanged from the previous INVITE. The session description MUST somehow indicate that it has not changed. In the case of SDP, this is accomplished by including the same value for the origin field as previous messages to its peer.

If no response to a refreshing re-INVITE is received before the expiration of the session, the UA SHOULD send a BYE request to terminate the call. It SHOULD send this BYE slightly before expiration of the session. The minimum of ten seconds and one third the session interval is RECOMMENDED.

For example, if the session interval is 120 seconds, one third of this is 40 seconds. Since the minimum of 10 seconds and 40 seconds is 10 seconds, the BYE would be sent 10 seconds before the session expires.

Firewalls and NATs may be very unforgiving about allowing SIP traffic to pass after the expiration time of the session. It is for this reason that the BYE should be sent before the expiration.

Note that it is possible that the calling UA is generating refreshes, and then it receives a re-INVITE. After following the rules for UAS described below, the calling UA now determines it is not supposed to generate refreshes. The UA SHOULD cease generating refreshes in this case, and let the other UA perform them. This also implies that the responsibility for generating refreshes may change during a call. This happens commonly in one specific case - both caller and callee support session timer. The caller will be doing re-INVITEs initially. If the callee re-INVITEs, it becomes the UAC for this transaction, and the rules defined in this specification will result in a change in refresh responsibility to the called party. In general, when both parties support session timer, refreshes become the responsibility of the party which performed the last INVITE transaction.

Note that it is possible for a UAS to believe that an INVITE is an initial INVITE, and for the UAC to believe it is a re-INVITE. This happens when a UA crashes and reboots between refreshes. When the refresh arrives at the rebooted UA, it decides to reject the call. The UAS can detect that the re-INVITE is for an existing call by the existence of the tag in the To field of the re-INVITE. In order to alert the UAC that it believes the call is now down (the UAC believes this request to be a re-INVITE, and so a non-200 OK final response
will not cause it to destroy the call), the UAS SHOULD include a Session-Expires and Require into the non-200 response (assuming session timer is supported by the UAC), with an immediate expiration time.

5 Proxy Behavior

Session expirations are mostly of interest to call stateful proxy servers. However, a stateful proxy server MAY also follow the rules described here. Stateless proxies MUST NOT attempt to request session timers. In general, proxies which ask for session timers will record-route, since they won’t receive refreshes if they don’t.

The proxy processing rules require the proxy to remember information between the request and response, ruling out stateless proxies.

5.1 Processing of requests

Due to local policy, a proxy may have guidelines about the desired maximum lifetime for a call initiated through it. When an initial INVITE is received to establish a call, a proxy MAY insert a Session-Expires header in the request before forwarding it, if none was present in the request. This Session-Expires header may contain any desired expiration time the proxy would like. If the request already had a Session-Expires header, the proxy MAY reduce the value in the Session-Expires header before forwarding the request, but MUST NOT increase it.

Assuming the proxy wishes to use session timer (and thus has possibly inserted the Session-Expires header or reduced it), the proxy MUST remember that it is using session timer, and also remember the value in the request it forwarded, until the final response to the request arrives, or the transaction times out. If the request also contained the Supported header with the value "timer", the proxy MUST remember this as well.

If the request did not contain a Supported header with the value "timer", the proxy MAY insert a Require header into the request, with the value "timer". However, this is NOT RECOMMENDED. This allows the proxy to insist on session timer for the session. This header is not needed if a Supported header was in the request; in this case, the proxy can already be sure that the session timer can be used for the session.

5.2 Processing of Responses
When the final response to the request arrives, it is examined by the proxy. There are four cases.

CASE I: UAC supports, UAS doesn’t. In this case, the request forwarded by the proxy contained a Session-Expires header (possibly inserted by the proxy). Recall that all proxies interested in session timer MUST remember the value of the timer in the forwarded request, in addition to whether the UAC supports it. Handling of the response for this scenario depends on whether the session timer aware proxy is the one closest to the UAS. For the proxy closest to the UAS, the final response to the INVITE does not contain a Session-Expires header, nor does it contain a Require header with the option tag "timer". The proxy MUST insert a Session-Expires header into the response with the value it remembered from the forwarded request. The proxy MUST also insert the Require header into the response, with the value "timer", before forwarding it upstream. The value of the Session-Expires in the forwarded response represents the expiration time of the session. For other proxies, the response will already contain the Session-Expires and Require header, so that this case is indistinguishable from CASE IV.

CASE II: UAC doesn’t support, UAS doesn’t support. In this case, the final response has no Session-Expires header, and the proxy remembers that the UAC did not support the session timer. The proxy forwards the response upstream normally. There are no session timers for this call leg.

CASE III: UAS supports, UAC doesn’t. In this case, the final response contains a Session-Expires header, but no Require header with the tag "timer". The proxy remembers that the UAC did not support session timers. The UAS will perform refreshes in this case. The proxy forwards the response upstream.

CASE IV: UAC supports, UAS supports. In this case, the final response contains a Session-Expires header and a Require header with the tag "timer". The UAC performs refreshes in this case. This case will also occur when the UAC supports session timer, and the UAS doesn’t, but a downstream proxy had recognized this as CASE I and inserted the Session-Expires and Require headers into the response. The proxy
forwards the response upstream.

In all cases, if the final response forwarded upstream by the proxy contains a Session-Expires header, its value represents the expiration time of the session for the call leg associated with that response. Note that there can be multiple 200 class responses to a single INVITE, each representing a different call leg, resulting in multiple session timers, one for each call leg.

In all cases, the proxy MUST NOT modify the value of the Session-Expires header received in the response (assuming one was present) before forwarding it upstream.

The expiration of the call leg will occur at the time indicated in the Session-Expires header. If the Session-Expires header contains a delta-time, the expiration time is the time of receipt of the final response, plus that delta time.

Re-INVITE requests may arrive from either UA, refreshing the session and extending the expiration time. Processing of these re-INVITEs by a proxy is identical to the procedure for processing the initial INVITE, except for the fact that processing includes any final response, not just 200 class.

If the session expires without having seen a response to a re-INVITE, the proxy MAY consider the call leg terminated. This means it MAY flush any state associated with that call leg.

Note that a proxy MUST NOT send a BYE request once the session expires.

6 UAS Behavior

When a UAS receives an INVITE for a new call, and that INVITE contains a Session-Expires header, the UAS MUST place a Session-Expires header in a 200 OK response (assuming the UAS accepts the call). The UAS MAY reduce the expiration time when it places this header into the response, but it MUST NOT increase it. If the initial INVITE did not contain a Session-Expires header, but it did contain a Supported header containing the option tag "timer", the UAS MAY insert a Session-Expires header into the response. This header MAY have any desired expiration time.

If the UAS places a Session-Expires header into the response, and the request contained a Supported header with the value "timer", the UAS MUST place a Require header into the response with the value "timer". In this case, the UAC will generate the refreshes.
If the UAS places a Session-Expires header into the response, and the request did not contain a Supported header with the value "timer", the UAS MUST NOT place a Require header into the response with the value "timer". In this case, the UAS will generate the refreshes.

These cases are summarized in Table 3. The table indicates the behavior of the UAS, assuming it supports session timers, as a function of the presence of the headers in the received request.

<table>
<thead>
<tr>
<th>Supported</th>
<th>Session-Expires</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
<td>If response contains Session-Expires, UAS refreshing. Otherwise, do nothing.</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>Copy Session-Expires to response. May reduce. UAS refreshing.</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>MAY add Session-Expires to response. If added, MUST add Require to response. UAC refreshing.</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>Copy Session-Expires to response. May reduce. Add Require to response. UAC refreshing.</td>
</tr>
</tbody>
</table>

Table 3: UAS Behavior

If the UAS is generating refreshes, it computes the expiration time of the session based on the value of the Session-Expires header in the response it sent. This expiration applies only to the call leg associated with that final response. The processing from this point is as described in section 4 once the UAC determined it was performing refreshes.

As described in Section 4, the refreshing UA will send periodic re-INVITEs to refresh the session. A UAS MUST be prepared to receive and process these re-INVITEs. Processing of the re-INVITE, as far as the session timer is concerned, is identical to the rules for the initial INVITE, described above, except that processing applies to any final response, not just 200 OK. Note that if the 200 OK to the re-INVITE has no Session-Expires, no expiration time exists for the session. This can happen if the proxies and/or UAS change their mind about session timers, and decide they no longer wish to use them. Since each INVITE is treated independently, the proxies or UAS do not need to continue requesting session timer event though they did so the first time.

Security Considerations
The session timer introduces the capability of a proxy to effectively force clients to send refreshes at a rate of the proxies choosing. It can also force the clients to send a BYE by setting the expirations to times that are too short. This introduces the possibility of rogue proxies introducing denial-of-service attacks. Use of short refresh intervals allows the proxies to create network load. The session timer mechanism allows the proxy to be able to terminate established calls - a capability a normal proxy doesn't have in [1].

As a result of these potential attacks, it is RECOMMENDED that IP or transport level security is used when communicating between proxies, and that requests with Session-Expires headers only be accepted over these secure transports.

8 Examples

The following examples are meant to illustrate the functionality associated with the session timer. In the interest of brevity, all headers excepting Supported, Session-Expires and Require are intentionally left out of the SIP messages.

8.1 Basic session timer

In this case, two UAs communicate directly, with no proxies. Both support the session timer. The call is setup with a two minute session expiration. One minute later, the UAC refreshes the session.

Calling UA -> Called UA
   INVITE
   Supported: timer
   Session-Expires: 120

Calling UA <- Called UA
   200 OK
   Require: timer
   Session-Expires: 120
   Called UA starts session timer on send
   Calling UA starts session timer on receipt

Calling UA -> Called UA
   ACK

60 seconds later:

Calling UA -> Called UA
   INVITE
   Supported: timer
   Session-Expires: 120

S.Donovan,J.Rosenberg
Calling UA <- Called UA
200 OK
Require: timer
Session-Expires: 120
Called UA starts session timer on send
Calling UA starts session timer on receipt
Calling UA -> Called UA
ACK

110 seconds later the called UA did not receive a re-INVITE. It therefore considers the call terminated and sends a BYE:

Calling UA <- Called UA
BYE
Calling UA -> Called UA
200 OK

8.2 Basic negotiation of Session Time

In this configuration, two UAs talk through a single stateful proxy server (SPS). Both the SPS and the UAS reduce the session timer.

Calling UA -> SPS
INVITE
Supported: timer
Session-Expires: 240

SPS -> Called UA
INVITE
Supported: timer
Session-Expires: 180
SPS wants a shorter timer

SPS <- Called UA
200 OK
Session-Expires: 120
Called UA wants a shorter timer
Require: timer

Calling UA <- SPS
200 OK
Session-Expires: 120
Proxy starts timer on send
Require: timer
Calling UA starts timer on receipt

Calling UA -> SPS
ACK
SPS -> Called UA
  ACK

For whatever reason, the calling UA decides not to refresh. So, after 110 seconds, it sends a BYE.

Calling UA -> SPS
  BYE

SPS -> Called UA
  BYE

SPS <- Called UA
  200 OK

Calling UA <- SPS
  200 OK

8.3 No Session-Expires Header in INVITE

In this scenario, the UA sends an INVITE without a Session-Expires header and with a Supported header containing the option tag "timer". Since the proxy wants session timer for the call, it adds the Session-Expires header.

Calling UA -> SPS
  INVITE
  No Session-Expires
  Supported: timer

SPS -> Called UA
  INVITE
  Supported: timer
  Session-Expires: 120
  Proxy added Session-Expires

SPS <- Called UA
  200 OK
  Session-Expires: 120
  Require: timer

Calling UA <- SPS
  200 OK
  Session-Expires: 120
Require: timer

Calling UA -> SPS
  ACK

SPS -> Called UA
  ACK

8.4 Session timer without Calling UA Support

In this scenario, the calling UA sends and INVITE without a Session-Expires header and without a Supported header containing the option tag "timer". Since the proxy wants session timer for the call it adds Session-Expires header before proxying the INVITE to the called UA.

Calling UA -> SPS
  INVITE

SPS -> Called UA
  INVITE  SPS adds session expires
  Session-Expires: 180

SPS <- Called UA
  200 OK  Called UA wants a shorter timer
  Session-Expires: 120  Called UA starts timer

Calling UA <- SPS
  200 OK  Proxy starts timer on send
  Session-Expires: 120

Calling UA -> SPS
  ACK

SPS -> Called UA
  ACK

Sixty seconds later, the called UA sends a re-INVITE. Note that the called UA does support session timer, so it includes a header(Supported) header in the request. The SPS adds the Session-Expires and Require headers into the response from the calling UA.

SPS <- Called UA
  INVITE
Supported: timer  
Session-Expires: 120

Calling UA <- SPS
INVITE
Supported: timer  
Session-Expires:120

Calling UA -> SPS
200 OK

SPS -> Called UA
Session-Expires: 120
Require: timer  
200 OK

SPS <- Called UA
ACK

Calling UA <- SPS
ACK

The Calling UA terminates the session for non timer related reasons:

Calling UA -> SPS
BYE

SPS -> Called UA
BYE

SPS <- Called UA
200 OK

Calling UA <- SPS
200 OK

8.5 Session Timer without Called UA Support

In this scenario, the calling UA indicates that it supports the session timer, but does not add the Session-Expires header into the INVITE. The proxy adds it, but session timer is not supported by the UAS. The call is still set up with a session timer, as all that is required is for one of the user agents involved in the call leg to understand the "timer" feature.
Calling UA -> SPS
INVITE
k: timer

SPS -> Called UA
INVITE              SPS adds S-E header
k: timer
Session-Expires: 180

SPS <- Called UA
200 OK              Called UA doesn’t understand session timer

Calling UA <- SPS
200 OK
Session-Expires: 180 SPS adds Session-Expires and Require
Require: timer

Calling UA -> SPS
ACK

SPS -> Called UA
ACK

The UAC then re-invites, which is responded to with a 400 because the new media streams were rejected. Since this is a re-INVITE, the session is still active, and thus the processing of the response to re-INVITE still occurs in the proxy.

Calling UA -> SPS
INVITE              UA asks for specific timer this time
k: timer
Session-Expires: 180 This is not mandatory

SPS -> Called UA
INVITE              SPS does not reduce Session-Expires header
k: timer
Session-Expires: 180

SPS <- Called UA
400 Rejected Media Called UA doesn’t understand session timer

Calling UA <- SPS
400 Rejected Media
Session-Expires: 180
SPS adds Session-Expires and Require
Require: timer

Calling UA -> SPS
ACK

SPS -> Called UA
ACK

8.6 Proxy insists on session timer

In this scenario, the calling UA does not support the session timer, but the SPS on the setup path insists on it by inserting a Require header. The UAS does not support the session timer either, so it rejects the request with a 420 as per standard RFC2543 extension handling. This response is forwarded upstream towards the UAC. The UAC treats it as a normal 400 class response, and then ACKs it. In this case, the call cannot be established.

It is slightly odd that a UAC will send a request without a Require header, and yet get a 420 response back with an Unsupported header. Normal UAs that don’t support session timer should handle this case correctly, treating it just as a normal 400 class response.

Note that proxy insertion of Require is NOT RECOMMENDED.

Calling UA -> SPS
INVITE

SPS -> Called UA
INVITE
SPS adds session expires
Require: timer
SPS adds Require
Session-Expires: 180

SPS <- Called UA
420 Bad Extension
Unsupported: timer

Calling UA <- SPS
420 Bad Extension
Unsupported: timer

Calling UA -> SPS
ACK

SPS -> Called UA
ACK

8.7 Neither UA Supports Session Timer

In this case, neither UA supports the session timer. However, one of
the proxies on the call setup path requests (but does not require)
it. The call completes without session timers.

Calling UA -> SPS
INVITE

SPS -> Called UA
INVITE
S-P adds S-E header compact form
x: 180

SPS <- Called UA
200 OK
Called UA doesn’t understand session timer

Calling UA <- SPS
200 OK
SPS doesn’t add S-E since it knows Calling UA
doesn’t support it

Calling UA -> SPS
ACK

SPS -> Called UA
ACK

8.8 Both UAs Support, Change in Roles

In this case, both user agents support session timer. The initial
INVITE from caller to callee results in refreshes being generated by
the caller. A re-INVITE sent from the callee changes that role so
that the callee refreshes.
Calling UA -> SPS
   INVITE
   Supported: timer
   Session-Expires: 240

SPS -> Called UA
   INVITE
   SPS wants timer, no change in value
   Supported: timer
   Session-Expires: 240

SPS <- Called UA
   200 OK
   Called UA supports timer
   Session-Expires: 240
   Inserts Require, Session-Expires
   Require: timer

Calling UA <- SPS
   200 OK
   Calling UA gets response with Require:timer
   Session-Expires: 240
   It is refreshing
   Require: timer

Calling UA -> SPS
   ACK

SPS -> Called UA
   ACK

The called UA (which is a UAC for this transaction) now sends a re-
INVITE:

SPS <- Called UA
   INVITE
   Supported: timer
   Session-Expires: 240

Calling UA <- SPS
   INVITE
   SPS wants timer, no change in value
   Supported: timer
   Session-Expires: 240

Calling UA -> SPS
   200 OK
   Calling UA supports timer
   Session-Expires: 240
   Inserts Require, Session-Expires
   Require: timer

S.Donovan,J.Rosenberg
SPS -> Called UA
200 OK
Session-Expires: 240
Require: timer

Called UA gets response with Require:timer
It is refreshing

SPS <- Called UA
ACK

Calling UA <- SPS
ACK

9 Changes since -03

o Now handle the case where the UAC wants refreshes, but none of the proxies, nor the UAS, support it. Same in the reverse case where UAS wants it, but the UAC nor any of the proxies want it.

o Added note about proxy insertion of Require resulting in a 420 even though UAC didn’t ask for any extensions.

o Added compact form

o Specified conditions under which refresh responsibilities change. Also added an example showing this case.

10 Author’s Addresses

Steven R. Donovan
dynamicsoft
5100 Tennyson Parkway, Suite 1200
Plano, Texas 75024
email: sdonovan@dynamicsoft.com

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
email: jdrosen@dynamicsoft.com
11 Bibliography


Table of Contents

1 Introduction ........................................ 1
2 Protocol Overview .................................. 2
3 Session-Expires Header Field Definition .......... 5
4 UAC Behavior ...................................... 6
5 Proxy Behavior .................................... 9
5.1 Processing of requests ............................ 9
5.2 Processing of Responses ......................... 9
6 UAS Behavior ...................................... 11
7 Security Considerations ............................ 12
8 Examples .......................................... 13
8.1 Basic session timer ................................ 13
8.2 Basic negotiation of Session Time ............... 14
8.3 No Session-Expires Header in INVITE .......... 15
8.4 Session timer without Calling UA Support .... 16
8.5 Session Timer without Called UA Support ...... 17
8.6 Proxy insists on session timer ................... 19
8.7 Neither UA Supports Session Timer ............. 20
8.8 Both UAs Support, Change in Roles ............ 20
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Changes since -03</td>
<td>22</td>
</tr>
<tr>
<td>10</td>
<td>Author’s Addresses</td>
<td>22</td>
</tr>
<tr>
<td>11</td>
<td>Bibliography</td>
<td>23</td>
</tr>
</tbody>
</table>