Internet Engineering Task Force
Internet Draft
draft-ietf-sip-session-timer-05.txt
July 20, 2001
Expires: February 2002

SIP WG
S.Donovan,J.Rosenberg
dynamicsoft

The SIP Session Timer

STATUS OF THIS MEMO

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 two new general headers, Session-Expires, which conveys the lifetime of the session, and Min-SE, which conveys the minimum allowed value for the session timer.

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 application of the session timer is in the construction of a SIP NAT ALG. The ALG embedded in a NAT will need to maintain state for the duration of a call. This state must eventually be 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. Two new general headers, Session-Expires and Min-SE, are defined. Session-Expires conveys the duration of the session, and Min-SE conveys the minimum allowed value for the session expiration.

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" [3]. 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. The UAC MAY include the "refresher" parameter in the Session-Expires header with the value "uac". This indicates that the caller will perform refreshes for the session.

Proxies on the signaling path may have their own requirements for the refresh interval of the session. If no Session-Expires was present, the proxy MAY insert one if it so desires. If one was present, the proxy can lower its value, but not to a value lower than the value in the Min-SE header in the request, if present. The proxy MUST NOT increase the value of the Session-Expires header. The proxy remembers the value of Session-Expires that was finally used in the request. The proxy also notes whether the UAC supports session timer, based on whether the Supported header was present with the tag "timer". In all cases, the proxy MUST NOT insert or modify the value of the "refresher" parameter in the Session-Expires header.

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, but not lower than the value in the Min-SE header, if present. The UAS also sets the value of the refresher parameter, based on rules in Section 7, which determines who will be generating the re-INVITEs for the refreshing. If the UAC is doing refreshes, the UAS additionally inserts a Require header into the 200 class response, with the tag "timer". This informs the

UAC that special processing is needed for this response.

If the UAS does not support session timer, it behaves as a normal UAS. Because of this, any 200 class response it generates will not contain the Session-Expires or Require header with the tag "timer".

The 200 class 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 (meaning the UAS does not support session timer), additional proxy processing is needed. If the proxy remembers that the UAC supported session timer, the proxy must return the value

of the session timer to the UAC. So, it inserts the Session-Expires header into the response (using the value it remembered inserting), sets the "refresher" parameter to "uac", and adds a Require header with a value of "timer". If 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.

If the response received by the proxy does contain the Session-Expires header, the "refresher" parameter will indicate which side is performing refreshes.

The response then arrives at the UAC. If it contains a Require header with the value "timer", the UAC knows that special processing of the response is needed. The response will also contain a Session-Expires header, and the value of that header is used as the interval for refreshes. The refresher parameter indicates who will be performing the refreshes. If it contains the value "uac", the UAC knows it is performing them. If it contains the value "uas", it knows that the UAS is performing them.

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

Whichever side is supposed to perform the refreshes MUST send a re-INVITE before the expiration time. This re-INVITE SHOULD contain a Session-Expires header. The processing of this re-INVITE by proxies and UAS is identical to that of the initial INVITE.

If the side not performing refreshes 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 200 class 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 2xx 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 minutes is RECOMMENDED. In other words, SIP entites 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 minutes.

Small session timer values can be destructive to the network. They cause excessive messaging traffic that affects both user agents and proxy servers. They increase the possibility of re-INVITE collisions

(when both parties re-INVITE each other at the same time). Since the primary purpose of session timer is to provide a means to time out state in SIP elements, very small values won't generally be needed. 30 minutes was chosen since 95% of phone calls are less than this duration. However, the 30 minute minimum is listed as a SHOULD, and not a MUST, since the exact value for this number is dependent on many network factors, including network bandwidths and latencies, computing power, memory availability, network topology, and of course, the application scenario. After all, SIP can set up any kind of session, nut just a phone call. At the time of publication of this document, 30 minutes seems appropriate. Advances in technologies may result in the number being excessively large five years in the future.

The syntax of the Session-Expires header is:

```
Session-Expires  =  ("Session-Expires" | "x") ":" ( SIP-date |
                    delta-seconds ) [refresher]
refresher        =  ";" "refresher" "=" "uas"|"uac"
```

OPEN ISSUE: Is there really any reason for absolute times? It just complicates things, since all operations really need to be done using the delta-seconds. Plus, it will require a Date header, so that a proxy will have to subtract the two to arrive at a delta-time anyway. RECOMMENDATION: Remove absolute time.

The optional refresher parameter indicates who will be doing the refreshing. It is RECOMMENDED that this parameter not be present in an initial INVITE, so that the negotiation mechanisms can successfully determine who will perform them. A response with the Session-Expires header MUST contain this parameter, indicating who will perform the refreshes. If the UAC wishes to insist on performing the refreshes, it MAY insert the parameter with a value of "uac" in the initial INVITE. It MUST NOT use a value of "uas" in the initial INVITE, since it doesn't know whether the UAS is capable of refreshes. However, in a re-INVITE, it MAY use a value of "uas" if it

knows that the other side supports session timer.

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:

	where	enc	e-e	ACK	BYE	CAN	INV	OPT	REG
Session-Expires	R	n	h	-	-	-	o	-	-
Session-Expires	2xx	n	h	-	-	-	o	-	-

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 Min-SE Header Field Definition

The Min-SE request header indicates the minimum value for the duration of the session timer. It is in units of delta-seconds. An INVITE for a particular call leg which has generated a 422 response MUST contain the Min-SE header, and the value MUST be the largest value amongst all Session-Expires values returned in 422 responses on that leg. A proxy or UAS MUST NOT reduce the value of the session timer below the value in this header, when present.

The syntax of the Min-SE header is:


```
Min-SE  =  "Min-SE" ":" delta-seconds
```

A UAC MAY include the Min-SE in an INVITE request, even if it never received a 422 previously.

Table 2 is an extension of tables 4 and 5 in [1] for the Session-Expires header:

	where	enc	e-e	ACK	BYE	CAN	INV	OPT	REG
Min-SE	R	n	h	-	-	-	o	-	-

Table 2: 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.

5 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" [3]. 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 the refresher parameter with value "uac" if it wishes to perform the refreshes.

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. The Supported header containing "timer" MUST still be included even if the Require or Proxy-Require headers are present containing "timer".

When a 2xx response to the initial INVITE request arrives, it may or

may not contain a Require header with the value "timer". If it does, the UAC MUST look for the Session-Expires header to process the response.

If there was a Require header in the response with the value "timer", the Session-Expires header will always be present. UACs MUST be prepared to receive a Session-Expires header in a response even if none were present in the request. The "refresher" parameter will be present, indicating who will be performing the refreshes. If the parameter contains the value "uac", the UAC will perform them. It is possible that the UAC requested session timer (and thus included a Session-Expires in the request, but there was no Require or Session-Expires in the 200 class response. This will happen when only the UAC supports session timer. In this case, the UAC needs to perform keepalives. To do this, the UAC follows the procedures defined in this specification as if the Session-Expires header were in the 200 class response, and its value was the same as the one in the request (but with a refresher parameter of "uac").

If the UAC must refresh, it computes the expiration time of the session. This is based on the value of the Session-Expires in the response. 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. The procedures for performing this refresh are described in Section 8.

If the response to the initial INVITE request is a 422 Session Timer Too Small response message then the UAC MAY retry the INVITE with a longer session timer value in the Session-Expires header. The UAC MUST use a value that is bigger than any value returned in a Session-Expires header in any 422 response within the call leg. This requires the UA to store the largest session timer duration returned in any 422 for the duration of the call leg. This behavior is needed to prevent certain denial of service attacks, described in Section 9.

6 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. Proxies which ask for session timers SHOULD 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.

6.1 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, but not with a duration lower than the value in the Min-SE header in the request, if present.

If the request already had a Session-Expires header, the proxy MAY reduce the value in the Session-Expires header, but MUST NOT set it to a duration lower than the value in the Min-SE header in the request, if present. The proxy MUST NOT increase the value of the duration.

The proxy MAY reject the INVITE request if the requested session timer value in the Session-Expires header is smaller than the minimum value defined in the proxies local policy. The proxy does so by sending a 422 Session Timer Too Small response message. When sending the 422 response message, the proxy MUST include a Session-Expires header with a value for the session timer that is acceptable to the proxy.

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 of the Session-Expires header it placed in the proxied request. This MUST be remembered for the duration of the transaction. The proxy MUST remember, for the duration of the transaction, whether the request contained the Supported header with the value "timer".

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.

6.2 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, amongst all proxies which requested session timer.

When the UAS sends the response, it will not contain a Session-Expires or Require header. This response is forwarded upstream, and it will eventually arrive at a proxy which was interested in session timers. Because there is no Session-Expires or Require in the response, the proxy knows it is the first session-timer-aware proxy to receive the response. This proxy MUST insert a Session-Expires header into the response with the value it remembered from the forwarded request. It MUST set the value of the "refresher" parameter to "uac". 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 with the refresher parameter set to "uas". The proxy forwards the

response upstream.

CASE IV: UAC supports, UAS supports. In this case, the final response contains a Session-Expires header. The refresher parameter indicates who is performing the refreshes. 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. 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.

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.

7 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 class response (assuming the UAS accepts the call). The UAS MAY reduce the expiration time when it places this header into the response, but MUST NOT set it to a duration lower than the value in the Min-SE header in the request, if present. The UAS MUST NOT increase the value of the duration.

The UAS MAY reject the INVITE request if the requested session timer value in the Session-Expires header is smaller than the minimum value defined in UAS local policy. The UAS does so by sending a 422 Session Timer Too Small response message. When sending the 422 response message, the UAS MUST include a Session-Expires header with a value that is acceptable to it.

If the inital INVITE did not contain a Session-Expires header, and the UAS wishes to use refreshes for the session, it MUST insert a Session-Expires header into the response. This header MAY have any desired expiration time greater than the value in the Min-SE header from the request, if present.

The UAS MUST set the value of the refresher parameter in the Session-Expires header present in a 200 class response. This value specifies who will perform refreshes for the call leg. The value is set based on the value of this parameter in the request and on whether the UAC supports session timer. Table 3 defines how the value in the response is set. A value of 'none' in the 2nd column means that there was no refresher parameter in the request. A value of 'NA' in the third column means that this particular combination shouldn't happen, as its disallowed by the protocol.

UAC supports?	refresher parameter	refresher parameter

	in request	in response
N	none	uas
N	uac	NA
N	uas	NA
Y	none	uas or uac
Y	uac	uac
Y	uas	uas

Table 3: UAS Behavior

In the fourth row of Table 3, since the UAC supports the session timer, and so does the UAS, the UAS can elect for itself or the UAC to perform the refreshes.

If the refresher parameter in the Session-Expires header in the 200 class response has a value of "uac", the UAS MUST place a Require header into the response with the value "timer".

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. 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 transmission of the response plus that delta time. Let the difference in time between the transmission 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.

If the 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. The procedures for performing this refresh are described in Section 8.

8 Performing Refreshes

The 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 in

Sections 5 and 7. However, as discussed in Section 4, if the UAC received any 422 responses for this call leg, the re-INVITE MUST contain a Min-SE header with the largest of all durations returned in all 422 responses on that call leg. The re-INVITE SHOULD contain a Session-Expires header, with an duration equal to the last duration. For example, if the initial INVITE sequence results in a session timer of 200 seconds, the re-INVITE SHOULD contain a session timer with a value of 200 seconds. The re-INVITE MAY contain a different expiration, but this should only be done if the UA wishes to change the session expiration.

If the Session-Timer is placed in the re-INVITE, the refresher parameter SHOULD be present, and SHOULD identify the element currently responsible for refreshes. This means that a re-INVITE may have a refresher parameter with value "uas", if the element not performing refreshes sends a re-INVITE for some other, non-session-timer reason. The refresher parameter MAY be set to identify the element not performing refreshes at this time. This will result in a change of roles.

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 even though they did so the first time.

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 [4], this is accomplished by using a different value for the origin field.

If the refreshing re-INVITE is used solely for refreshing, it SHOULD 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. The same is true for the 200 class response to a re-INVITE used solely for refreshing. The response MUST contain a session description with

an indication that it has not changed. This is accomplished in the same way as for the request.

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.

It is possible that the calling UA is generating refreshes, and then it receives a re-INVITE. After following the rules for UAS described above, 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 may 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.

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 (generally, it will reject the call unless it explicitly is capable of recovering lost calls). If From tags are used, 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. Therefore, a UAC MUST insert a From tag in an initial INVITE if it supports session timer. A UAS that wishes to reject a re-INVITE for a call that it believes is already terminated SHOULD respond with a 481. A UAC receiving a 481 to a session timer refresh MUST generate a BYE to terminate that call leg.

> Without From tags, A could INVITE B without a From tag. B inserts a tag in the 200 OK. Now, B sends a re-INVITE to A. Meantime, A has crashed and rebooted. This re-INVITE has a From tag, but no To tag. It therefore cannot be distinguished for a new INVITE in which the UAC inserts a From tag. This ambiguity is resolved by mandating use of From tag with session timer.

> The requirement for From tags and responding with a 481 to stale re-INVITEs has been added to the updated version of RFC2543. However, to eliminate a dependency between this spec and the new version of SIP, these two features are specified here as well.

9 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 or UASes introducing denial-of-service attacks. Use of short refresh intervals allows the proxies or UAS to create network load. This is preventable using the 422 response and Min-SE header mechanism. Any proxy or the UAS can reject an INVITE with a low session timer, and the UAC will then include this minimum in the Min-SE header in subsequent requests. No compliant proxy will then use a session timer with a lower value. Rogue proxies could strip the Min-SE, but in this case, UA configuration should ideall prevent the attacks by refusing to use a session timer lower than a specific value (30 minutes is recommended here).

It is also 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.

10 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, Min-SE and Require are intentionally left out of the SIP messages.

10.1 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;refresher=uac   Called UA starts timer on send
                                            Calling UA starts timer on recei
Calling UA -> Called UA
       ACK

60 seconds later:

Calling UA -> Called UA
       INVITE
       Supported: timer
       Session-Expires: 120;refresher=uac

Calling UA <- Called UA
       200 OK
       Require: timer
       Session-Expires: 120;refresher=uac   Called UA starts timer on send
                                            Calling UA starts timer on recei
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
```

10.2 Basic negotiation of Session Time

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

```
   Calling UA -> proxy
           INVITE
           Supported: timer
           Session-Expires: 240

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

   proxy <- Called UA
           200 OK                                Called UA wants a shorter timer
           Session-Expires: 120;refresher=uac    Called UA starts timer
           Require: timer

   Calling UA <- proxy
           200 OK
           Session-Expires: 120;refresher=uac    Proxy starts timer on send
           Require: timer                        Calling UA starts timer on recei

   Calling UA -> proxy
          ACK

   proxy -> Called UA
          ACK
```

For whatever reason, the calling UA decides not to refresh. So, after

110 seconds, it sends a BYE.

```
Calling UA -> proxy
        BYE

proxy -> Called UA
       BYE
```



```
proxy <- Called UA
       200 OK

Calling UA <- proxy
       200 OK
```

10.3 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 -> proxy
       INVITE                  No Session-Expires
       Supported: timer

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

proxy <- Called UA
       200 OK
       Session-Expires: 120;refresher=uac   Called UA starts timer on send
       Require: timer

Calling UA <- proxy
       200 OK
```

```
        Session-Expires: 120;refresher=uac   Proxy starts timer on send
        Require: timer                       Calling UA starts timer on recei

   Calling UA -> proxy
          ACK

   proxy -> Called UA
          ACK
```

10.4 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 -> proxy
           INVITE

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

   proxy <- Called UA
           200 OK                               Called UA wants a shorter timer
           Session-Expires: 120;refresher=uas   Called UA starts timer on send

   Calling UA <- proxy
           200 OK
           Session-Expires: 120;refresher=uas   Proxy starts timer on send
                                                Called UA starts timer on receip

   Calling UA -> proxy
          ACK

   proxy -> 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 proxy adds the Session-Expires and Require headers into the response from the calling UA.

```
proxy <- Called UA
       INVITE
       Supported: timer
       Session-Expires: 120;refresher=uac

Calling UA <- proxy
       INVITE
       Supported: timer
       Session-Expires:120;refresher=uac

Calling UA -> proxy
       200 OK

proxy -> Called UA
       200 OK
```



```
       Session-Expires: 120;refresher=uac  Proxy updates timer on send
       Require: timer                      Called UA updates timer on receip


proxy <- Called UA
       ACK

Calling UA <- proxy
       ACK
```

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

```
Calling UA -> proxy
       BYE

proxy -> Called UA
       BYE
```

```
proxy <- Called UA
       200 OK

Calling UA <- proxy
       200 OK
```

10.5 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 -> proxy
       INVITE
       k: timer

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



```
proxy <- Called UA
       200 OK                 Called UA doesn't understand session timer

Calling UA <- proxy
       200 OK
       Session-Expires: 180;refresher=uac  Proxy adds S-E and Require
       Require: timer                      Proxy starts timer on send
                                           Calling UA starts timer on receip

Calling UA -> proxy
       ACK
```

```
proxy -> 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.

```
Calling UA -> proxy
       INVITE
       k: timer
       Session-Expires: 180;refresher=uac   UA asks for timer this time
                                            This is not mandatory

proxy -> Called UA
       INVITE                    proxy does not reduce Session-Expires heade
       k: timer
       Session-Expires: 180;refresher=uac

proxy <- Called UA
       400 Rejected Media        Called UA doesn't understand session timer

Calling UA <- proxy
       400 Rejected Media

Calling UA -> proxy
       ACK

proxy -> Called UA
       ACK
```


10.6 Proxy insists on session timer

In this scenario, the calling UA does not support the session timer, but the proxy 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 -> proxy
        INVITE

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

proxy <- Called UA
        420 Bad Extension
        Unsupported: timer

Calling UA <- proxy
        420 Bad Extension
        Unsupported: timer

Calling UA -> proxy
       ACK

proxy -> Called UA
       ACK
```

10.7 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 -> proxy
       INVITE

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

proxy <- Called UA
       200 OK                  Called UA doesn't understand session timer

Calling UA <- proxy         proxy doesn't add S-E since it knows Calling UA
       200 OK               doesn't support it

Calling UA -> proxy
       ACK

proxy -> Called UA
       ACK
```

10.8 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 -> proxy
        INVITE
        Supported: timer
        Session-Expires: 240

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

proxy <- Called UA
        200 OK                              Called UA supports timer
```

```
          Session-Expires: 240;refresher=uac  Inserts Require, Session-Expires
          Require: timer                      Called UA starts timer on send

   Calling UA <- proxy                        Calling UA sees refresher=uac
          200 OK                              It is refreshing
          Session-Expires: 240;refresher=uac  Proxy starts timer on send
          Require: timer                      Calling UA starts timer on recei

   Calling UA -> proxy
          ACK

   proxy -> Called UA
          ACK
```

The called UA (which is a UAC for this transaction) now sends a re-INVITE. For whatever reason, it decides to switch roles by explicitly designating the itself as the refresher.

```
   proxy <- Called UA
          INVITE
          Supported: timer
          Session-Expires: 240;refresher=uac

   Calling UA <- proxy
          INVITE                proxy wants timer, no change in value
          Supported: timer
          Session-Expires: 240;refresher=uac

   Calling UA -> proxy
          200 OK                              Calling UA supports timer
          Session-Expires: 240;refresher=uac  Inserts Session-Expires
          Require: timer                      Calling UA updates timer on send

   proxy -> Called UA                         Called UA sees 200 w/refresher=u
          200 OK                              It is refreshing
          Session-Expires: 240;refresher=uac  Proxy updates timer on send
          Require: timer                      Called UA updates timer on recei
```

```
proxy <- Called UA
       ACK

Calling UA <- proxy
       ACK
```


10.9 Proxy Rejects Timer

In this call flow, the calling UA sends an INVITE with a Session-Expires header that is low. This arrives at a proxy, which rejects it with a 422 because it is too low. The proxy offers a larger minimum timer. The calling UA retries with a new Session-Expires and with a Min-SE header.

```
Calling UA -> Proxy
      INVITE
      Supported: timer
      Session-Expires: 10              UAC uses low timer

Calling UA <- Proxy
      422 Timer Too Low
      Session-Expires: 200             Proxy says minimum is 200s

Calling UA -> Proxy
      ACK

Calling UA -> Proxy
      INVITE
      Supported: timer
      Session-Expires: 300             UAC chooses larger SE
      Min-SE: 200                      Minimum is 200

Proxy -> Called UA
      INVITE
      Supported: timer
      Session-Expires: 250             Proxy lowers to 250
      Min-SE: 200

Proxy <- Called UA
```

```
          200
          Require: timer
          Session-Expires: 250;refresher=uac   UAS lowers to 200

   Calling UA <- Proxy
          200
          Require: timer
          Session-Expires: 250;refresher=uac

   Calling UA -> Proxy
          ACK

   Proxy -> Called UA
```



```
          ACK
```

11 Acknowledgements

The authors wish to thank Brett Tate for his contributions to this work.

12 Changes since -04

- Added requirement for From tags with session timer, to handle this crash and reboot case. Discussed when a UA would want to recover calls this way.
- Removed text about inserting Session-Expires:0 when you want to indicate that the call is down. Rather, send a 481.
- Made handling of a 481 a MUST for UAC.
- Added clarification to call flows on when timer is started and updated.
- Added wording indicating that it is bad to do usage billing using SIP.
- Added wording indicating that the UAS should not change the

SDP for a re-INVITE that is used solely for refreshing the session timer.

- Added text about using 422 Session Timer Too Small message to reject an INVITE with a session expiration value that is smaller that policy at a proxy or UAC.

- Changed SPS to proxy in call flows

- Mentioned that low session timer values can lead to re-INVITE glare.

- Added discussion of why minimum of 30 minutes is a SHOULD and not a MUST.

- Added Min-SE header and related processing.

- Added refresher parameter to Session-Expires, which has simplified processing.

13 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

14 Bibliography

[1] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," Request for Comments 2543, Internet Engineering Task Force, Mar. 1999.

[2] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," Request for Comments 1889, Internet Engineering Task Force, Jan. 1996.

[3] J. Rosenberg and H. Schulzrinne, "The SIP supported header," Internet Draft, Internet Engineering Task Force, Feb. 2001. Work in progress.

[4] M. Handley and V. Jacobson, "SDP: session description protocol," Request for Comments 2327, Internet Engineering Task Force, Apr. 1998.

Table of Contents