Network Working Group
Request for Comments: 4028
Category: Standards Track

S. Donovan
J. Rosenberg
Cisco Systems

April 2005

Session Timers in the Session Initiation Protocol (SIP)

Status of This Memo

Copyright Notice

Abstract

This document defines an extension to the Session Initiation Protocol (SIP). This extension allows for a periodic refresh of SIP sessions through a re-INVITE or UPDATE request. The refresh allows both user agents and proxies to determine whether the SIP session is still active. The extension defines two new header fields: Session-Expires, which conveys the lifetime of the session, and Min-SE, which conveys the minimum allowed value for the session timer.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [2] does not define a keepalive mechanism for the sessions it establishes. Although the user agents may be able to determine whether the session has timed out by using session specific mechanisms, proxies will not be able to do so. The result is that call stateful proxies will not always be able to determine whether a session is still active. For instance, when a user agent fails to send a BYE message at the end of a session, or when 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 method to determine 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-INVITE or UPDATE [3] requests (referred to as session refresh requests) to keep the session alive. The interval for the session refresh requests is determined through a negotiation mechanism defined here. If a session refresh request 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.

This refresh mechanism has additional applications. A user agent would like to determine whether the session is still active for the same reasons a call stateful proxy server would. 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 [5]. However, it is desirable to separate indications of SIP session liveness from the details of the particular session.

Another application of the session timer is in the construction of a SIP Network Address Translator (NAT) Application Level Gateway (ALG) [6]. 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 provides an extension to SIP that defines a session expiration mechanism. Periodic refreshes, through re-INVITEs or UPDATEs, are used to keep the session active. The extension is sufficiently backward compatible with SIP that it works as long as either one of the two participants in a dialog understands the extension. Two new header fields (Session-Expires and Min-SE) and a new response code (422) are defined. Session-Expires conveys the duration of the session, and Min-SE conveys the minimum allowed value for the session expiration. The 422 response code indicates that the session timer duration was too small.

2. Terminology

In this document, the key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' are to be interpreted as described in RFC 2119 [1] and indicate requirement levels for compliant SIP implementations.

Additionally, we define the following terms:

Session Interval: The maximum amount of time that can occur between session refresh requests in a dialog before the session will be considered timed out. The session interval is conveyed in the Session-Expires header field, which is defined here. The UAS obtains this value from the Session-Expires header field in a 2xx response to a session refresh request that it sends. Proxies and UACs determine this value from the Session-Expires header field in a 2xx response to a session refresh request that they receive.

Minimum Timer: Because of the processing load of mid-dialog requests, all elements (proxy, UAC, UAS) can have a configured minimum value for the session interval that they are willing to accept. This value is called the minimum timer.

Session Expiration: The time at which an element will consider the session timed out, if no successful session refresh transaction occurs beforehand.

Session Refresh Request: An INVITE or UPDATE request processed according to the rules of this specification. If the request generates a 2xx response, the session expiration is increased to the current time plus the session interval obtained from the response. A session refresh request is not to be confused with a target refresh request, defined in Section 6 of [2], which is a request that can update the remote target of a dialog.

Initial Session Refresh Request: The first session refresh request sent with a particular Call-ID value.

Subsequent Session Refresh Request: Any session refresh request sent with a particular Call-ID after the initial session refresh request.

Refresh: Same as a session refresh request.

3. Overview of Operation

This section provides a brief overview of the operation of the extension. It is tutorial in nature and should not be considered normative.

This extension has the property that it works even when only one UA in a dialog supports it. The processing steps differ for handling each of the four cases (the UAC does or doesn’t support it, and the UAS does or doesn’t support it). For simplicity’s sake, this section

will describe basic operation in the case where both sides support the extension.

A UAC starts by sending an INVITE. This includes a Supported header field with the option tag 'timer', indicating support for this extension.

This request passes through proxies, any one of which may have an interest in establishing a session timer. Each proxy can insert a Session-Expires header field and a Min-SE header field into the request (if none is already there) or alter the value of existing Session-Expires and Min-SE header fields as described below.

The Min-SE header field establishes the lower bound for the session refresh interval; i.e., the fastest rate any proxy servicing this request will be allowed to require. The purpose of this header field is to prevent hostile proxies from setting arbitrarily short refresh intervals so that their neighbors are overloaded. Each proxy processing the request can raise this lower bound (increase the period between refreshes) but is not allowed to lower it.

The Session-Expires header field establishes the upper bound for the session refresh interval; i.e., the time period after processing a request for which any session-stateful proxy must retain its state for this session. Any proxy servicing this request can lower this value, but it is not allowed to decrease it below the value specified in the Min-SE header field.

If the Session-Expires interval is too low for a proxy (i.e., lower than the value of Min-SE that the proxy would wish to assert), the proxy rejects the request with a 422 response. That response contains a Min-SE header field identifying the minimum session interval it is willing to support. The UAC will try again, this time including the Min-SE header field in the request. The header field contains the largest Min-SE header field it observed in all 422 responses previously received. This way, the minimum timer meets the constraints of all proxies along the path.

After several INVITE/422 iterations, the request eventually arrives at the UAS. The UAS can adjust the value of the session interval as if it were a proxy; when done, it places the final session interval into the Session-Expires header field in a 2xx response. The Session-Expires header field also contains a 'refresher' parameter, which indicates who is doing the refreshing -- the UA that is currently the UAC, or the UA that is currently the UAS. As the 2xx response travels back through the proxy chain, each proxy can observe the final session interval but can't change it.

From the Session-Expires header field in the response, both UAs know that a session timer is active, when it will expire, and who is refreshing. At some point before the expiration, the currently active refresher generates a session refresh request, which is a re-INVITE or UPDATE [3] request. If the refresher never gets a response to that session refresh request, it sends a BYE to terminate the session. Similarly, if the other side never gets the session refresh request before the session expires, it sends a BYE.

The refresh requests sent once the session is established are processed identically to the initial requests, as described above. This means that a successful session refresh request will extend the session, as desired.

The extension introduces additional complications beyond this basic flow to support cases where only one of the UAs supports it. One such complication is that a proxy may need to insert the Session-Expires header field into the response, in the event that the UAS doesn't support the extension. The negotiation of the role of refresher is also affected by this capability; it takes into consideration which participants support the extension.

Note that the session timer refreshes the session, not the dialog used to establish the session. Of course, the two are related. If the session expires, a BYE is sent, which terminates the session and, generally, the dialog.

4. Session-Expires Header Field Definition

The Session-Expires header field conveys the session interval for a SIP session. It is placed only in INVITE or UPDATE requests, as well as in any 2xx response to an INVITE or UPDATE. Like the SIP Expires header field, it contains a delta-time.

The absolute minimum for the Session-Expires header field is 90 seconds. This value represents a bit more than twice the duration that a SIP transaction can take in the event of a timeout. This allows sufficient time for a UA to attempt a refresh at the halfpoint of the session interval, and for that transaction to complete normally before the session expires. However, 1800 seconds (30 minutes) is RECOMMENDED as the value for the Session-Expires header field. In other words, SIP entities MUST be prepared to handle Session-Expires header field values of any duration greater than 90 seconds, but entities that insert the Session-Expires header field SHOULD NOT choose values of less than 30 minutes.

Small session intervals can be destructive to the network. They cause excessive messaging traffic that affects both user agents and proxy servers. They increase the possibility of 'glare' that can occur when both user agents send a re-INVITE or UPDATE at the same time. Since the primary purpose of the 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 because 95% of phone calls are shorter than this duration. However, the 30 minute minimum is listed as a SHOULD, and not as 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, not 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 default value of the Session-Expires header field is undefined. This means that the absence of the Session-Expires header field implies no expiration of the session, using the mechanism defined in this specification. Note that other mechanisms not defined in this specification, such as locally configured timers, may apply.

The syntax of the Session-Expires header field is as follows:

```
Session-Expires  = ("Session-Expires" / "x") HCOLON delta-seconds
                     *(SEMI se-params)
se-params        = refresher-param / generic-param
refresher-param  = "refresher" EQUAL  ("uas" / "uac")
```

Note that a compact form, the letter x, has been reserved for Session-Expires. The BNF for delta-seconds and generic-param is defined in Section 25 of RFC 3261 [2].

Table 1 is an extension of Tables 2 and 3 in [2] for the Session-Expires and Min-SE header fields. The column 'PRA' is for the PRACK method [7], 'UPD' is for the UPDATE method [3], 'SUB' is for the SUBSCRIBE method [8], and 'NOT' is for the NOTIFY method [8].

Header	where	proxy	ACK	BYE	CAN	INV	OPT	REG	PRA	UPD	SUB	NOT
Session-Expires	R	amr	-	-	-	o	-	-	-	o	-	-
Session-Expires	2xx	ar	-	-	-	o	-	-	-	o	-	-
Min-SE	R	amr	-	-	-	o	-	-	-	o	-	-
Min-SE	422		-	-	-	m	-	-	-	m	-	-

Table 1: Session-Expires and Min-SE Header Fields

5. Min-SE Header Field Definition

The Min-SE header field indicates the minimum value for the session interval, in units of delta-seconds. When used in an INVITE or UPDATE request, it indicates the smallest value of the session interval that can be used for that session. When present in a request or response, its value MUST NOT be less than 90 seconds.

When the header field is not present, its default value for is 90 seconds.

The Min-SE header field MUST NOT be used in responses except for those with a 422 response code. It indicates the minimum value of the session interval that the server is willing to accept.

The syntax of the Min-SE header field is as follows:

Min-SE = "Min-SE" HCOLON delta-seconds *(SEMI generic-param)

6. 422 Response Code Definition

This extension introduces the 422 (Session Interval Too Small) response code. It is generated by a UAS or proxy when a request contains a Session-Expires header field with a duration below the minimum timer for the server. The 422 response MUST contain a Min-SE header field with the minimum timer for that server.

7. UAC Behavior

7.1. Generating an Initial Session Refresh Request

A UAC that supports the session timer extension defined here MUST include a Supported header field in each request (except ACK), listing the option tag 'timer' [2]. It MUST do so even if the UAC is not requesting usage of the session timer for this session.

The UAC MAY include a Require header field in the request with the value 'timer' to indicate that the UAS must support the session timer to participate in the session. This does not mean that the UAC is requiring the UAS to perform the refreshes, only that it is requiring the UAS to support the extension. In addition, the UAC MAY include a Proxy-Require header field in the request with the value 'timer' to indicate that proxies must support the 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 field containing 'timer' MUST still be included, even if the Require or Proxy-Require header fields are present containing 'timer'.

A UAC MAY include the Min-SE header field in the initial INVITE request.

A UAC MAY include a Session-Expires header field in an initial session refresh request if it wants a session timer applied to the session. The value of this header field indicates the session interval desired by the UAC. If a Min-SE header is included in the initial session refresh request, the value of the Session-Expires MUST be greater than or equal to the value in Min-SE.

The UAC MAY include the refresher parameter with value 'uac' if it wants to perform the refreshes. However, it is RECOMMENDED that the parameter be omitted so that it can be selected by the negotiation mechanisms described below.

7.2. Processing a 2xx Response

The session timer requires a UA to create and maintain state. This state includes the session interval, the session expiration, and the identity of the refresher. This state is associated with the dialog on which the session has been negotiated.

When a 2xx response to a session refresh request arrives, it may or may not contain a Require header field with the value 'timer'. If it does, the UAC MUST look for the Session-Expires header field to process the response.

If there was a Require header field in the response with the value 'timer', the Session-Expires header field will always be present. UACs MUST be prepared to receive a Session-Expires header field in a response, even if none were present in the request. The 'refresher' parameter will be present in the Session-Expires header field, indicating who will perform the refreshes. The UAC MUST set the identity of the refresher to the value of this parameter. If the parameter contains the value 'uac', the UAC will perform them. It is possible that the UAC requested the session timer (and thus included a Session-Expires header field in the request) and that there was no Require or Session-Expires header field in the 2xx response. This will happen when the UAS doesn't support the session timer extension and only the UAC has asked for a session timer (no proxies have requested it). In this case, if the UAC still wishes to use the session timer (which is purely for its benefit alone), it has to perform them. To do this, the UAC follows the procedures defined in this specification as if the Session-Expires header field were in the 2xx response, and its value was the same as that in the request, but with a refresher parameter of 'uac'.

If the 2xx response did not contain a Session-Expires header field, there is no session expiration. In this case, no refreshes need to be sent. A 2xx without a Session-Expires can come for both initial and subsequent session refresh requests. This means that the session timer can be 'turned-off' in mid dialog by receiving a response without a Session-Expires header field.

The UAC remembers the session interval for a session as the value of the delta-time from the Session-Expires header field in the most recent 2xx response to a session refresh request on a dialog. It is explicitly allowed for there to be differing session intervals (or none at all) on differing dialogs established as a result of a single INVITE. The UAC also remembers whether it or its peer is the refresher on for the session.

If the UAC must perform the refreshes, it computes the session expiration for that session. The session expiration is the time of reception of the last 2xx response to a session refresh request on that dialog plus the session interval for that session. If the UA seeks to continue with the session beyond the session expiration, it MUST generate a refresh before the session expiration. It is

RECOMMENDED that this refresh be sent once half the session interval has elapsed. Additional procedures for this refresh are described in Section 10.

Similarly, a re-INVITE or UPDATE request sent within a dialog for purposes other than session refreshes will also have the effect of refreshing the session, and its processing will follow the procedures defined in this specification.

7.3. Processing a 422 Response

If the response to a session refresh request is a 422 (Session Interval Too Small) response message, then the UAC MAY retry the request. The procedures for retrying are described in Section 7.4. This new request constitutes a new transaction and SHOULD have the same value as the Call-ID, To, and From of the previous request, but the CSeq should contain a new sequence number that is one higher than the previous.

7.4. Generating Subsequent Session Refresh Requests

The values of Supported, Require, and Proxy-Require used in the initial Session refresh request MUST be used.

The UAC MUST insert the Min-SE header field into a session refresh request for a particular dialog if it has ever received a 422 response to a previous session refresh request on the same dialog, or if it has received a session refresh request on that dialog that contained a Min-SE header field. Similarly, if no dialog has been established yet, a UAC MUST insert the Min-SE header field into an INVITE request if it has ever received a 422 response to a previous INVITE request with the same Call-ID.

The value of the Min-SE header field present in a session refresh request MUST be the largest value among all Min-SE header field values returned in all 422 responses or received in session refresh requests, on the same dialog, if a dialog has been established. If no dialog has been established, the Min-SE header field value is set to the largest value among all Min-SE header field values returned in all 422 responses for an INVITE request with the same Call-ID. A result of this rule is that the maximum value of the Min-SE is effectively 'cleared' once the dialog is established, and from that point on, only the values from proxies known to be on the proxy path will end up being used.

The UAC may have its own opinions about the minimum session interval. In that case, if the value above is too small, the UAC MAY increase it.

In a session refresh request sent within a dialog with an active session timer, the Session-Expires header field SHOULD be present. When present, it SHOULD be equal to the maximum of the Min-SE header field (recall that its default value when not present is 90 seconds) and the current session interval. Inclusion of the Session-Expires header field with this value avoids certain denial-of-service attacks, as documented in Section 11. As such, a UA should only ignore the SHOULD in unusual and singular cases where it is desirable to change the session interval mid-dialog.

If the session refresh request is not the initial one, it is RECOMMENDED that the refresher parameter be set to 'uac' if the element sending the request is currently performing refreshes, and to 'uas' if its peer is performing the refreshes. This way, the role of refresher does not change on each refresh. However, if it wishes to explicitly change the roles, it MAY use a value of 'uas' if it knows that the other side supports the session timer. It could know this by having received a request from its peer with a Supported header field containing the value 'timer'. If it seeks to reselect the roles, it MAY omit the parameter.

A re-INVITE generated to refresh the session is a normal re-INVITE, and an UPDATE generated to refresh a session is a normal UPDATE. If a UAC knows that its peer supports the UPDATE method, it is RECOMMENDED that UPDATE be used instead of a re-INVITE. A UA can make this determination if it has seen an Allow header field from its peer with the value 'UPDATE', or through a mid-dialog OPTIONS request. It is RECOMMENDED that the UPDATE request not contain an offer [4], but a re-INVITE SHOULD contain one, even if the details of the session have not changed. In that case, the offer MUST indicate that it has not changed. In the case of SDP, this is accomplished by including the same value for the origin field as did previous SDP messages to its peer. The same is true for an answer exchanged as a result of a session refresh request; if it has not changed, that MUST be indicated.

8. Proxy Behavior

Session timers are mostly of interest to call stateful proxy servers (that is, to servers that maintain the state of calls and dialogs established through them). However, a stateful proxy server (that is, a server which is aware of transaction state but does not retain call or dialog state) MAY also follow the rules described here. Stateless proxies MUST NOT attempt to request session timers. Proxies that ask for session timers SHOULD record-route, as 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.

8.1. Processing of Requests

Processing of requests is identical for all session refresh requests.

To request a session timer for a session, a proxy makes sure that a Session-Expires header field is present in a session refresh request for that session. A proxy MAY insert a Session-Expires header field in the request before forwarding it if none was present in the request. This Session-Expires header field may contain any desired expiration time the proxy would like, but not with a duration lower than the value in the Min-SE header field in the request, if it is present. The proxy MUST NOT include a refresher parameter in the header field value.

If the request already had a Session-Expires header field, the proxy MAY reduce its value but MUST NOT set it to a duration lower than the value in the Min-SE header field in the request, if it is present. If the value of the Session-Expires header field is greater than or equal to the value in the Min-SE header field (recall that the default is 90 seconds when the Min-SE header field is not present), the proxy MUST NOT increase the value of the Session-Expires header field. If the value of the Session-Expires header field is lower than the value of the Min-SE header field (possibly because the proxy increased the value of the Min-SE header field, as described below), the proxy MUST increase the value of the Session-Expires header field to make it equal to Min-SE header field value. The proxy MUST NOT insert or modify the value of the 'refresher' parameter in the Session-Expires header field.

If the request contains a Supported header field with a value 'timer', the proxy MAY reject the INVITE request with a 422 (Session Interval Too Small) response if the session interval in the Session-Expires header field is smaller than the minimum interval defined by the proxy's local policy. When sending the 422 response, the proxy MUST include a Min-SE header field with the value of its minimum interval. That minimum MUST NOT be lower than 90 seconds.

If the request doesn't indicate support for the session timer but contains a session interval that is too small, the proxy cannot usefully reject the request, as this would result in a call failure. Rather, the proxy SHOULD insert a Min-SE header field containing its minimum interval. If a Min-SE header field is already present, the proxy SHOULD increase (but MUST NOT decrease) the value to its minimum interval. The proxy MUST then increase the Session-Expires

header field value to be equal to the value in the Min-SE header field, as described above. A proxy MUST NOT insert a Min-SE header field or modify the value of an existing header field in a proxied request if that request contains a Supported header field with the value 'timer'. This is needed to protect against certain denial of service attacks, described in Section 11.

Assuming that the proxy has requested a session timer (and thus has possibly inserted the Session-Expires header field or reduced it), the proxy MUST remember that it is using a session timer, and also remember the value of the Session-Expires header field from 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 field with the value 'timer'. If the request did not contain a Supported header field with the value 'timer', the proxy MAY insert a Require header field with the value 'timer' into the request. However, this is NOT RECOMMENDED. This allows the proxy to insist on a session timer for the session. This header field is not needed if a Supported header field was in the request; in this case, the proxy would already be sure the session timer can be used for the session.

8.2. Processing of Responses

When the final response to the request arrives, it is examined by the proxy.

If the response does not contain a Session-Expires header field but the proxy remembers that it requested a session timer in the request (by inserting, modifying, or examining and accepting the Session-Expires header field in the proxied request), this means that the UAS did not support the session timer. If the proxy remembers that the UAC did not support the session timer either, the proxy forwards the response upstream normally. There is no session expiration for this session. If, however, the proxy remembers that the UAC did support the session timer, additional processing is needed.

Because there is no Session-Expires or Require header field in the response, the proxy knows that it is the first session-timer-aware proxy to receive the response. This proxy MUST insert a Session-Expires header field 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 add the 'timer'

option tag to any Require header field in the response, and if none was present, add the Require header field with that value before forwarding it upstream.

If the received response contains a Session-Expires header field, no modification of the response is needed.

In all cases, if the 2xx response forwarded upstream by the proxy contains a Session-Expires header field, its value represents the session interval for the session associated with that response. The proxy computes the session expiration as the time when the 2xx response is forwarded upstream, plus the session interval. This session expiration MUST update any existing session expiration for the session. The refresher parameter in the Session-Expires header field in the 2xx response forwarded upstream will be present, and it indicates which UA is performing the refreshes. There can be multiple 2xx responses to a single INVITE, each representing a different dialog, resulting in multiple session expirations, one for each session associated with each dialog.

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

8.3. Session Expiration

When the current time equals or passes the session expiration for a session, the proxy MAY remove associated call state, and MAY free any resources associated with the call. Unlike the UA, it MUST NOT send a BYE.

9. UAS Behavior

The UAS must respond to a request for a session timer by the UAC or a proxy in the path of the request, or it may request that a session timer be used itself.

If an incoming request contains a Supported header field with a value 'timer' and a Session Expires header field, the UAS MAY reject the INVITE request with a 422 (Session Interval Too Small) response if the session interval in the Session-Expires header field is smaller than the minimum interval defined by the UAS' local policy. When sending the 422 response, the UAS MUST include a Min-SE header field with the value of its minimum interval. This minimum interval MUST NOT be lower than 90 seconds.

If the UAS wishes to accept the request, it copies the value of the Session-Expires header field from the request into the 2xx response.

The UAS response MAY reduce its value but MUST NOT set it to a duration lower than the value in the Min-SE header field in the request, if it is present; otherwise the UAS MAY reduce its value but MUST NOT set it to a duration lower than 90 seconds. The UAS MUST NOT increase the value of the Session-Expires header field.

If the incoming request contains a Supported header field with a value 'timer' but does not contain a Session-Expires header, it means that the UAS is indicating support for timers but is not requesting one. The UAS may request a session timer in the 2XX response by including a Session-Expires header field. The value MUST NOT be set to a duration lower than the value in the Min-SE header field in the request, if it is present.

The UAS MUST set the value of the refresher parameter in the Session-Expires header field in the 2xx response. This value specifies who will perform refreshes for the dialog. The value is based on the value of this parameter in the request, and on whether the UAC supports the session timer extension. The UAC supports the extension if the 'timer' option tag was present in a Supported header field in the request. Table 2 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 it is disallowed by the protocol.

UAC supports?	refresher parameter in request	refresher parameter in response
N	none	uas
N	uac	NA
N	uas	NA
Y	none	uas or uac
Y	uac	uac
Y	uas	uas

Table 2: UAS Behavior

The fourth row of Table 2 describes a case where both the UAC and UAS support the session timer extension, and where the UAC did not select who will perform refreshes. This allows the UAS to decide whether it or the UAC will perform the refreshes. However, as the table indicates, the UAS cannot override the UAC's choice of refresher, if it made one.

If the refresher parameter in the Session-Expires header field in the 2xx response has a value of 'uac', the UAS MUST place a Require header field into the response with the value 'timer'. This is

because the uac is performing refreshes and the response has to be processed for the UAC to know this. If the refresher parameter in the 2xx response has a value of 'uas' and the Supported header field in the request contained the value 'timer', the UAS SHOULD place a Require header field into the response with the value 'timer'. In this case, the UAC is not refreshing, but it is supposed to send a BYE if it never receives a refresh. Since the call will still succeed without the UAC sending a BYE, insertion of the Require is a SHOULD here, and not a MUST.

Just like the UAC, the UAS stores state for the session timer. This state includes the session interval, the session expiration, and the identity of the refresher. This state is bound to the dialog used to set up the session. The session interval is set to the value of the delta-time from the Session-Expires header field in the most recent 2xx response to a session refresh request on that dialog. It also remembers whether it or its peer is the refresher on the dialog, based on the value of the refresher parameter from the most recent 2xx response to a session refresh request on that dialog. If the most recent 2xx response had no Session-Expires header field, there is no session expiration, and no refreshes have to be performed.

If the UAS must refresh the session, it computes the session expiration. The session expiration is the time of transmission of the last 2xx response to a session refresh request on that dialog plus the session interval. If UA wishes to continue with the session beyond the session expiration, it MUST generate a refresh before the session expiration. It is RECOMMENDED that this refresh be sent once half the session interval has elapsed. Additional procedures for this refresh are described in Section 10.

10. Performing Refreshes

The side generating a refresh does so according to the UAC procedures defined in Section 7. Note that only a 2xx response to a session refresh request extends the session expiration. This means that a UA could attempt a refresh and receive a 422 response with a Min-SE header field that contains a value much larger than the current session interval. The UA will still have to send a session refresh request before the session expiration (which has not changed), even though this request will contain a value of the Session-Expires that is much larger than the current session interval.

If the session refresh request transaction times out or generates a 408 or 481 response, then the UAC sends a BYE request as per Section 12.2.1.2 of RFC 3261 [2]. If the session refresh request does not generate a 2xx response (and, as a result, the session is not refreshed), and a response other than 408 or 481 is received, the UAC

SHOULD follow the rules specific to that response code and retry if possible. For example, if the response is a 401, the UAC would retry the request with new credentials. However, the UAC SHOULD NOT continuously retry the request if the server indicates the same error response.

Similarly, if the side not performing refreshes does not receive a session refresh request before the session expiration, it SHOULD send a BYE to terminate the session, slightly before the session expiration. The minimum of 32 seconds and one third of the session interval is RECOMMENDED.

> Firewalls and NAT ALGs may be very unforgiving about allowing SIP traffic to pass after the expiration time of the session. This is why the BYE should be sent before the expiration.

11. Security Considerations

The session timer introduces the capability of a proxy or UA element to force compliant UAs to send refreshes at a rate of the element's choosing. This introduces the possibility of denial-of-service attacks with significant amplification properties. These attacks can be launched from 'outsiders' (elements that attempt to modify messages in transit) or by 'insiders' (elements that are legitimately in the request path but are intent on doing harm). Fortunately, both cases are adequately handled by this specification.

11.1. Inside Attacks

This introduces the possibility of rogue proxies or UAs introducing denial-of-service attacks. However, the mechanisms in this specification prevent that from happening.

First, consider the case of a rogue UAC that wishes to force a UAS to generate refreshes at a rapid rate. To do so, it inserts a Session-Expires header field into an INVITE with a low duration and a refresher parameter equal to uas. Assume it places a Supported header field into the request. The UAS or any proxy that objects to this low timer will reject the request with a 422, thereby preventing the attack. If no Supported header field was present, the proxies will insert a Min-SE header field into the request before forwarding it. As a result, the UAS will not choose a session timer lower than the minimum allowed by all elements on the path. This too prevents the attack.

Next, consider the case of a rogue UAS that wishes to force a UAC to generate refreshes at a rapid rate. In that case, the UAC has to support session timer. The initial INVITE arrives at the rogue UAS,

which returns a 2xx with a very small session interval. The UAC uses this timer and quickly sends a refresh. Section 7.4 requires that the UAC copy the current session interval into the Session-Expires header field in the request. This enables the proxies to see the current value. The proxies will reject this request and provide a Min-SE with a higher minimum, which the UAC will then use. Note, that if the proxies did not reject the request, but rather proxied the request with a Min-SE header field, an attack would still be possible. The UAS could discard this header field in a 2xx response and force the UAC to continue to generate rapid requests.

In a similar fashion, a rogue proxy cannot force either the UAC or UAS to generate refreshes unless the proxy remains on the signaling path and sees every request and response.

11.2. Outside Attacks

An element that can observe and modify a request or response in transit can force rapid session refreshes. To prevent this, requests and responses have to be protected by message integrity. Since the session timer header fields are not end-to-end and are manipulated by proxies, the SIP S/MIME capabilities are not suitable for this task. Rather, integrity has to be protected by using hop-by-hop mechanisms. As a result, it is RECOMMENDED that an element send a request with a Session-Expires header field or a Supported header field with the value 'timer' by using TLS. As adequate protection is obtained only if security is applied on each hop, it is RECOMMENDED that the SIPS URI scheme be used in conjunction with this extension. This means that proxies that record-route and request session timer SHOULD record-route with a SIPS URI. A UA that inserts a Session-Expires header into a request or response SHOULD include a Contact URI that is a SIPS URI.

12. IANA Considerations

This extension defines two new header fields, a new response code, and a new option tag. SIP [2] defines IANA procedures for registering these.

12.1. IANA Registration of Min-SE and Session-Expires Header Fields

The following is the registration for the Min-SE header field:

RFC Number: RFC 4028
Header Name: Min-SE
Compact Form: none

The following is the registration for the Session-Expires header field:

RFC Number: RFC 4028
Header Name: Session-Expires
Compact Form: x

12.2. IANA Registration of the 422 (Session Interval Too Small) Response Code

The following is the registration for the 422 (Session Interval Too Small) response code:

Response Code: 422
Default Reason Phrase: Session Interval Too Small
RFC Number: RFC 4028

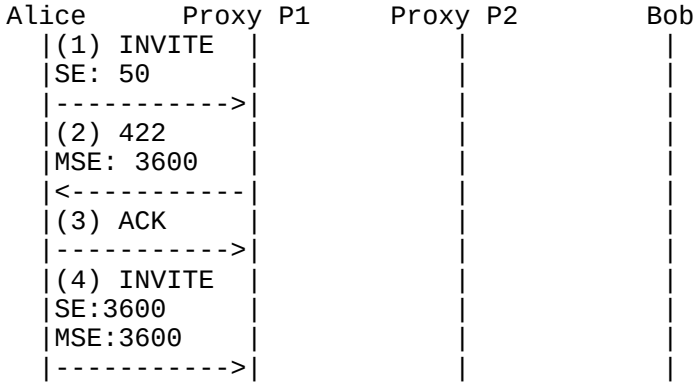
12.3. IANA Registration of the 'timer' Option Tag

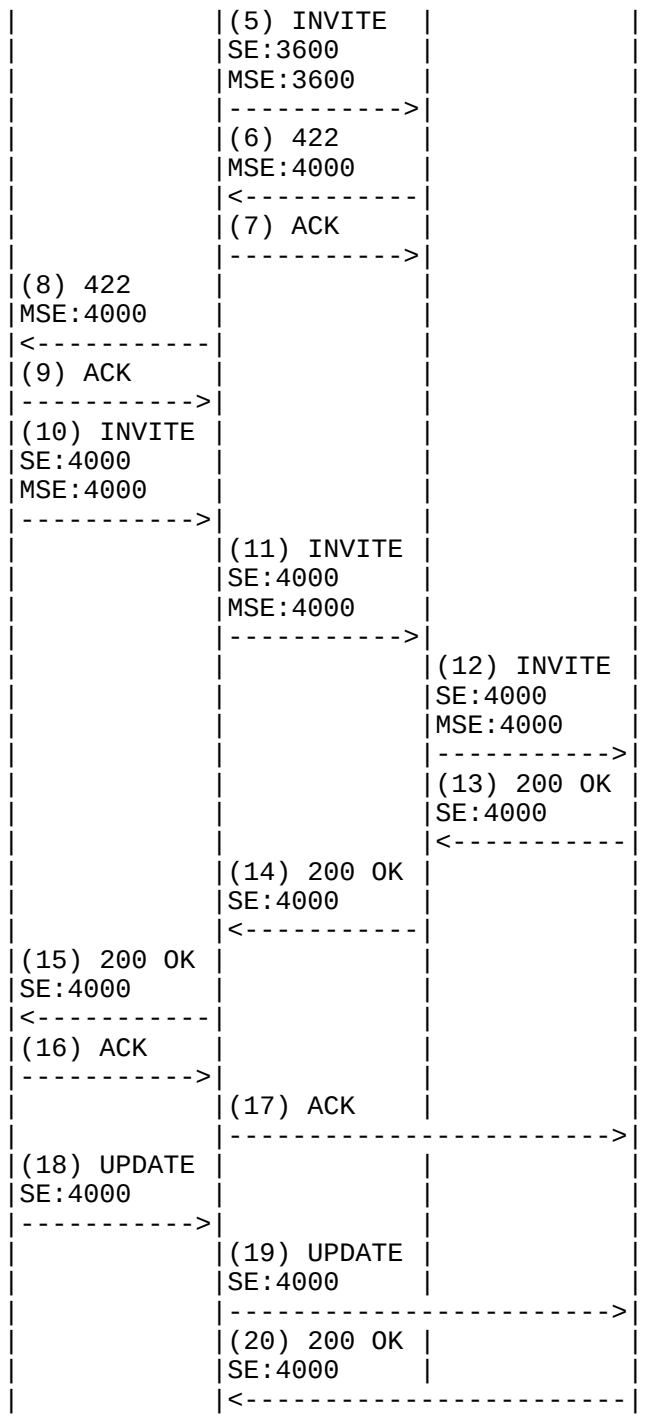
The following is the registration for the 'timer' option tag:

Name: timer
Description: This option tag is for support of the session timer extension. Inclusion in a Supported header field in a request or response indicates that the UA can perform refreshes according to that specification. Inclusion in a Require header in a request means that the UAS must understand the session timer extension to process the request. Inclusion in a Require header field in a response indicates that the UAC must look for the Session-Expires header field in the response and process it accordingly.

13. Example Call Flow

Example Session Timer Flow




(5) INVITE
SE:3600
MSE:3600
(6) 422
MSE:4000
(7) ACK
(8) 422
MSE:4000
(9) ACK
(10) INVITE
SE:4000
MSE:4000
(11) INVITE
SE:4000
MSE:4000
(12) INVITE
SE:4000
MSE:4000
(13) 200 OK
SE:4000
(14) 200 OK
SE:4000
(15) 200 OK
SE:4000
(16) ACK
(17) ACK
(18) UPDATE
SE:4000
(19) UPDATE
SE:4000
(20) 200 OK
SE:4000

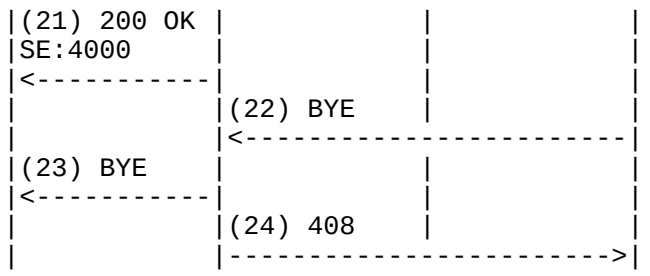


Figure 1: Example Session Timer Flow

Figure 1 gives an example of a call flow that makes use of the session timer. In this example, both the UAC and UAS support the session timer extension. The initial INVITE request generated by the UAC, Alice (message 1), might look like this:

```
INVITE sips:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com;branch=z9hG4bKnashds8
Supported: timer
Session-Expires: 50
Max-Forwards: 70
To: Bob <sips:bob@biloxi.example.com>
From: Alice <sips:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sips:alice@pc33.atlanta.example.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)
```

This request indicates that Alice supports the session timer, and is requesting session refreshes every 50 seconds. This arrives at the first proxy, P1. This session interval is below the minimum allowed value of 3600. So P1 rejects the request with a 422 (message 2):

```
SIP/2.0 422 Session Interval Too Small
Via: SIP/2.0/TLS pc33.atlanta.example.com;branch=z9hG4bKnashds8
  ;received=192.0.2.1
Min-SE: 3600
To: Bob <sips:bob@biloxi.example.com>;tag=9a8kz
From: Alice <sips:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
```

This response contains a Min-SE header field with the value 3600. Alice then retries the request. This time, the request contains a Min-SE header, as Alice has received a 422 for other INVITE requests with the same Call-ID. The new request (message 4) might look like this:

```
INVITE sips:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com;branch=z9hG4bKnashds9
Supported: timer
Session-Expires: 3600
Min-SE: 3600
Max-Forwards: 70
To: Bob <sips:bob@biloxi.example.com>
From: Alice <sips:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314160 INVITE
Contact: <sips:alice@pc33.atlanta.example.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)
```

Proxy P1 record-routes. Since the session interval is now acceptable to it, it forwards the request to P2 (message 5). However, the session interval is below its minimum configured amount of 4000. So it rejects the request with a 422 response code (message 6) and includes a Min-SE header field with the value of 4000. Once more, Alice retries the INVITE. This time, the Min-SE header field in her INVITE is the maximum of all Min-SE she has received (3600 and 4000). Message 10 might look like this:

```
INVITE sips:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com;branch=z9hG4bKnashds10
Supported: timer
Session-Expires: 4000
Min-SE: 4000
Max-Forwards: 70
To: Bob <sips:bob@biloxi.example.com>
From: Alice <sips:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314161 INVITE
Contact: <sips:alice@pc33.atlanta.example.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)
```

P1 record-routes once again, but P2 does not (this wouldn't normally happen; presumably, if it asked for session timer, it would record-route the subsequent request). The UAS receives the request. It copies the Session-Expires header from the request to the response and adds a refresher parameter with value 'uac'. This 200 OK is forwarded back to Alice. The response she receives (message 15) might look like this:

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS pc33.atlanta.example.com;branch=z9hG4bKnashds10
 ;received=192.0.2.1
Require: timer
Supported: timer
Record-Route: sips:p1.atlanta.example.com;lr
Session-Expires: 4000;refresher=uac
To: Bob <sips:bob@biloxi.example.com>;tag=9as888nd
From: Alice <sips:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314161 INVITE
Contact: <sips:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 142

(Bob's SDP not shown)
```

Alice generates an ACK (message 16), which is routed through P1 and then to Bob. Since Alice is the refresher, around 2000 seconds later Alice sends an UPDATE request to refresh the session. Because this request is part of an established dialog and Alice has not received any 422 responses or requests on that dialog, there is no Min-SE header field in her request (message 18):

```
UPDATE sips:bob@192.0.2.4 SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com;branch=z9hG4bKnashds12
Route: sips:p1.atlanta.example.com;lr
Supported: timer
Session-Expires: 4000;refresher=uac
Max-Forwards: 70
To: Bob <sips:bob@biloxi.example.com>;tag=9as888nd
From: Alice <sips:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314162 UPDATE
Contact: <sips:alice@pc33.atlanta.example.com>
```

This is forwarded through P1 to Bob. Bob generates a 200 OK, copying the Session-Expires header field into the response. This is forwarded through P1 and arrives at Alice. The response she receives (message 21) might look like this:

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS pc33.atlanta.example.com;branch=z9hG4bKnashds12
  ;received=192.0.2.1
Require: timer
Session-Expires: 4000;refresher=uac
To: Bob <sips:bob@biloxi.example.com>;tag=9as888nd
From: Alice <sips:alice@atlanta.example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314162 UPDATE
Contact: <sips:bob@192.0.2.4>
```

Shortly afterward, Alice's UA crashes. As a result, she never sends a session refresh request. 3968 seconds later, Bob times out and sends a BYE request (message 22). This is sent to P1. P1 attempts to deliver it but fails (because Alice's UA has crashed). P1 then returns a 408 (Request Timeout) to Bob.

14. Acknowledgements

The authors wish to thank Brett Tate for his contributions to this work. Brian Rosen completed the editing of the document.

15. References

15.1. Normative References

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

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

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

[4] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

15.2. Informative References

[5] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[6] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[7] Rosenberg, J. and H. Schulzrinne, "Reliability of Provisional Responses in Session Initiation Protocol (SIP)", RFC 3262, June 2002.

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

Authors' Addresses

Steve Donovan
Cisco Systems, Inc.
2200 E. President George Bush Turnpike
Richardson, Texas 75082
US

EMail: srd@cisco.com

Jonathan Rosenberg
Cisco Systems, Inc.
600 Lanidex Plaza
Parsippany, NJ 07054
US

EMail: jdrosen@cisco.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.