SIP Session-Timers Support in Asterisk

Requirements and Design Document

Created By: Raj Jain {rj2807@gmail.com}

1. Introduction

The Asterisk PBX currently does not have a way to reclaim SIP sessions that do not terminate through normal signaling procedures due to network problems or when the other end-point or an intermediary record-routing proxy dies. Currently, there is no way to detect and therefore reclaim such “leaked” SIP sessions in Asterisk.

> Note: Asterisk is currently capable of detecting RTP inactivity and can teardown SIP sessions as a result of that, but that does not help when the RTP is not flowing through the Asterisk. In addition, it does not help clear state in the intermediary proxies and the remote UA (for instance, when the BYE request does not make it to all the hops including the far-end).

The SIP Session-Timers is an extension of the SIP protocol that allows end-points and proxies to refresh a session periodically. The sessions are kept alive by sending a RE-INVITE or UPDATE request at a negotiated interval. If a session refresh fails then all the entities that support Session-Timers clear their internal session state. In addition, UAs generate a BYE request in order to clear the state in the proxies and the remote UA (this is done for the benefit of SIP entities in the path that do not support Session-Timers).

This document describes the requirements and design for Session-Timers support in the Asterisk SIP stack (a.k.a. SIP channel driver or chan_sip.c). The implementation of Session-Timers feature in Asterisk will be compliant to RFC 4028. There will be certain policies implemented in Asterisk where RFC 4028 leaves the discretion to the implementer. These policies are outlined in this document as a part of requirements.

2. Requirements

This section describes the requirements for SIP Session-Timers implementation in the Asterisk. These requirements also describe the policies implemented in the Asterisk with respect to what Asterisk implementation will choose to do when RFC 4028 leaves it at the discretion of the local policy implemented in the SIP UA (these are typically the behaviors described with MAY or SHOULD keywords in the RFC text).

2.1. Configuration

The Session-Timers can be configured on a system-wide, per-user, or per-peer basis. The per-user/per-peer settings override the global settings. The following new parameters have been added to the sip.conf file.

```
session-timers=["accept", "originate", "refuse"]
session-expires=[integer]
session-minse=[integer]
session-refresher=["uas", "uac"]
```

session-timers:

The session-timers parameter defines the mode of operation of SIP session-timers feature in Asterisk. The Asterisk can be configured in one of the following three modes:

1. **`accept`**: In the "accept" mode, the Asterisk honors session-timers request made by remote end-points. A remote end-point can request Asterisk to engage session-timers by either sending it an INVITE request with Supported: timer header in it or by responding to Asterisk's INVITE with a 200 OK that contains Session-Expires: header in it. In this mode, the Asterisk does not request session-timers from remote end-points. This is the default mode in Asterisk.

2. **`originate`**: In the "originate" mode, the Asterisk requests the remote end-points to engage session-timers in addition to honoring such requests made by the remote end-points. In order to get as much protection as possible against SIP channel leaks due to network or end-point failures, in this mode the Asterisk runs session-timers and sends periodic RE-INVITEs even if a remote end-point does not support the session-timers feature.

3. **`refuse`**: In the "refuse" mode, the Asterisk acts as if it does not support session-timers for inbound or outbound requests. If a remote end-point requests Asterisk to engage session-timers in a dialog (by sending Supported: timer header) then Asterisk simply ignores that request. If a remote end-point requires Asterisk to engage session-timers (by sending Require: timer header in an INVITE) then Asterisk rejects that request with a 420 Bad Extension response.

`session-expires:`

The `session-expires` parameter contains the session refresh interval in seconds chosen by Asterisk. When acting as a UAC, `session-expires` is the value the Asterisk publishes in the Session-Expires: header in an outbound INVITE. When acting as a UAS, if the remote end-point suggests a session refresh interval that is larger than Asterisk's configured `session-expires`, then Asterisk will negotiate the refresh interval down to its configured `session-expires`. Asterisk does this be including a Session-Expires: with the appropriate value in the 200 OK response. The default value for this parameter is 1800 seconds.

This behavior is described with a MAY keyword in RFC 4028 (paragraph from section 9 of RFC 4028 quoted below). In order to get utmost protection with session-timers, the Asterisk implementation chooses to refresh at a lower interval if configured so.

> 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.

`session-minse:`

The `session-minse` parameter contains the minimum session refresh interval in seconds acceptable to Asterisk. This parameter basically serves to throttle back end-points that want to refresh the session too frequently, which can be detrimental to network bandwidth and performance of Asterisk. When acting as a UAC, `session-minse` is the value the Asterisk publishes in the Min-SE: header in an outbound INVITE. If Asterisk receives a 422 response meaning that its `session-minse` is not acceptable to the UAS, then Asterisk adjusts its Min-SE: header value on the fly to what is recommended by the UAS in the 422 responses for that

dialog. When acting as a UAS, if the remote end-point suggests a Min-SE: interval that is lower than Asterisk's configured `session-minse`, then Asterisk will reject that request with a 422 response. The 422 response generated by Asterisk will contain Min-SE: `session-minse` header in it.

The default and the minimum value for this parameter is 90 seconds. According to RFC 4028, this parameter can not be configured to a value lower than 90 seconds in sip.conf. If someone configures this to be less than 90 seconds in sip.conf then Asterisk will ignore that setting and will default to 90 seconds.

`session-refresher:`

The `session-refresher` parameter defines whether the UAC or UAS will be the one sending session refresh requests in a dialog. The possible values for this parameter are "`uac`" or "`uas`". This parameter is meaningful for inbound calls only where the Asterisk acts as a UAS. If the Asterisk receives an INVITE where the refresher= parameter is absent in Session-Expires: header (which means that the UAC is letting the UAS decide the refresher), then Asterisk selects its configured `session-refresher` as the party that will generate session refresh RE-INVITEs. The Asterisk lets the UAC know of this decision by sending refresher= parameter in the 200 response.

When acting as a UAC, the Asterisk leaves the session refresher selection to the UAS. That is, Asterisk does not present refresher= parameter in the Session-Expires: header in an outbound INVITE. This behavior is recommended by RFC 4028.

The default value for this parameter is "`uas`".

2.2. Retry on receiving 422

The Asterisk will retry an outbound call if it gets a 422 (Session Interval Too Small) response. The Asterisk will adjust its Min-SE for the session to the Min-SE that it received in the 422 response.

There may be several retires of a particular request if the intermediary proxies between the UAC and the UAS also participate in Session-Timer. The users should tune their `session-expires` setting in such a way that it meets the Min-SE condition for all the proxies in the path and UAS such that the 422 retry roundtrips are saved.

The Asterisk as a UAS will generate a 422 response if it receives a request with Session-Expires: interval lower than the Asterisk's `session-minse` setting. Asterisk will include a Min-SE: header in the 422 response with the value of its `session-minse`.

2.3. Remote UAS doesn't support Session-Timer

When the Asterisk is configured to "`originate`" Session-Timers but the UAS does not support the feature, then Asterisk will still run the Session-Timers. This works on the principle that every UA must honor a re-INVITE. In this case, the Asterisk UAC will assume the role of the refresher and will periodically send session refresh requests. Since there was no negotiation of the session refresh interval, the Asterisk will pick the refresh interval of its own choice (the value of `session-expires` parameter in sip.conf).

If the Asterisk does not receive a response for a session refresh re-INVITE then it will assume that something has gone wrong and it will tear down the session internally and will issue a BYE requesttowards the UAS.

2.4. Remote UAC doesn't support Session-Timer

When Asterisk receives a SIP INVITE without a Supported: or Require: header field indicating 'timer' option tag then that means the UAC does not support Session-Timer. The onus is now on Asterisk to decide whether it wants to run Session-Timer for the session or not. If the Asterisk is configured in the "`originate`" mode then Asterisk will run the Session-Timer even if the UAC does not support it. The Asterisk will act as a refresher and will refresh the session at the interval indicated in `session-expires` parameter in sip.conf.

2.5. Turning-off Session-Timer Mid Session

The SIP protocol allows Session-Timers to be dynamically turned-off in the middle of a session. This happens when the UAS responds to a mid-dialog session refresh request with a 200 response without a Session-Expires: header field. When the Asterisk is acting as the UAS in a session then it will never turn-off the Session-Timer in the middle of a session (even if sip.conf is updated accordingly and dynamically uploaded).

When the Asterisk is acting as a UAC and if the UAS turns-off the Session-Timer in the middle of a session then the Asterisk will assume the behavior dictated by `session-timers` parameter in sip.conf. If the session-expire was set to "`originate`" then Asterisk will assume the refresher role (if it was not already the one) for the remainder of the session.

2.6. Adjusting Session-Timer Mid Session

The SIP protocol allows Session-Timers to be dynamically adjusted in the middle of a session. This happens when the UAS responds to a mid-dialog session refresh request with a 200 response with a different Session-Expires: header field value than the one in the 200 response for the previous refresh request. When the Asterisk is acting as the UAS in a session then it will never change Session-Timers interval in the middle of a session (even if sip.conf is updated accordingly and dynamically uploaded).

When the Asterisk is acting as a UAC and if the UAS adjusts the Session-Timer in the middle of a session then Asterisk will honor that and react accordingly.

2.7. Session Refresh Methods

According to RFC 4028, a session can be refreshed either using the UPDATE method or the re-INVITE method (when a distinction is not necessary, these methods are generically referred to as session refresh requests in this document) . UPDATE is usually a superior technique for modifying or refreshing a session as compared to a RE-INVITE. UPDATE offers at least the following advantages over RE-INVITE:

1. UPDATE is lighter on the end-points and the network because it involves a request and a response (two SIP messages), whereas, a RE-INVITE involves two SIP requests and a response (three SIP messages).

2. UPDATE (being a separate transaction) can be sent during the initial INVITE transaction. This was one of the main reasons the UPDATE method was invented because it allowed modifications of SDP during early-media. From a Session-Timer perspective, UPDATE can be used to refresh sessions that are in an early-dialog state (re-INVITEs cannot be used until a confirmed dialog is established)

3. UPDATE can be completed without SDP offer/answer exchange, whereas, re-INVITEs must include SDP offer/answer exchange even when the intent is session refresh. When a re-INVITE is used to refresh a session, the UAC must use the same SDP version number in its offer as the previous request and the UAS must figure out that the SDP version did not change from the previous offer. This places additional burden on both the UAC and UAS.

4. UPDATE is more atomic and less prone to race conditions as compared to re-INVITE. A re-INVITE request can have one or more provisional responses which opens up a wider window for re-INVITE glares and other race conditions.

The issue, however, is that the Asterisk SIP stack currently does not handle inbound UPDATE requests. Handling inbound UPDATE requests is a bigger topic than building the Session-Timers feature. SIP does not permit allowing inbound UPDATE just for session-timers without opening up the gates for all other possible usages of UPDATE (when a UA indicates UPDATE in its Allow: header then that means it supports the UPDATE method in general; in other words there is no way of indicating to the other end-point that Asterisk supports UPDATE for session-timers only).

Whenever the inbound UPDATE support is added to Asterisk it will have to be added for all the usages of UPDATE such as session modification, session timers (mid-dialog), session timers (early dialog) and session modification during early-media. Adding support for session modification and session timers usages of UPDATE is simple (we can do the same processing that we do for re-INVITEs today); the complexity arises in session modification during early-media scenarios and also when session refreshes have to happen before a confirmed dialog is established. Handling of these scenarios will require an inherently RFC 3261 compliant SIP stack because these scenarios require running multiple transactions (UPDATE and INVITE) simultaneously. An RFC 3261 compliant SIP stack is somewhat of a pre-requisite if we want to properly support the UPDATE method.

2.8. Requiring Session-Timers

In SIP an end-point can require that the other end-point must support Session-Timers for a session or else the session be rejected. This is done by including the ‘timer’ option tag in Require: or Proxy-Require: instead of the Supported: header field.

In the interest of interoperability with maximum number of end-point implementations, Asterisk will never require the Session-Timers support from the other end-point or the proxies. If the other end-point requires Asterisk to support Session-Timer and if the Asterisk `session-timers` flag is configured to “`refuse`” then asterisk will reject the request with a 420 (Bad Extension) response. Note that the “`refuse`” has a much milder behavior (ignore) when the other end-point sends Supported: timer.

2.9. Session-Refresher

Asterisk will support a configurable setting for refresher= param in Session-Expires header. When Asterisk is acting as the UAC (that is, it generates an outgoing call), then Asterisk will not add refresher= parameter in the outbound INVITE. The reasoning behind this is the following: RFC 4028 does not allow the UAC to send refresher=uas, therefore, if sip.conf dictates that the `session-refresher=uas`, then Asterisk can not reflect the same in refresher= parameter. Secondly, RFC 4028 recommends that the refresher=uac should not be sent in the interest of leaving it to negotiation.

2.10. Interop Testing

This feature will be tested for interoperability with a publicly available SIP UA (such as X-Lite).

2.11. Performance Testing

Some performance testing will be done using SIPp. Linux 'top' command will be used to compare performance with the Session-Timer feature enabled or disabled.

3. Future Enhancements

1. Session-Timer control from dial-plan.
2. Session-Timer parameters read/write through AMI
3. Support of UPDATE method for session refreshes.

4. Implementation Details

Session-Timers Negotiation:

The Session-Timers parameters (active/disabled, session refresh interval, minimum refresh interval, and the refresher) are negotiated on a per-session basis between the UAC and the UAS. This negotiation happens in INVITE/2XX response. The Session-Timers negotiation logic has been added to the handle_request_invite() function to support cases where the Asterisk receives an INVITE and therefore acts as the UAS.

Generating timer events:

The Asterisk scheduler is utilized for generating the session timers stimuli. The ast_sched_add() and ast_sched_del() functions have been utilized for turning session timers on and off.

Interaction with RTP inactivity timeouts:

The SIP Session-Timers have no interaction with RTP inactivity timers. In a system where the RTP is traversing through the Asterisk, it is possible that both these mechanisms may be simultaneously running.

5. Existing Bug Fixes

The following three existing bugs in Asterisk SIP channel were encountered in the development of Session-Timers. These bugs have been resolved and can be checked-in separately if needed:

Asterisk ignores option tags in Require: header: The Asterisk checks for the Require: header in an inbound INVITE and if it sees one or more options tags from the standard sets of options that it does not support then it rejects the INVITE with 420 response. However, if a Require: option is supported such as "replaces" and "timer" then Asterisk was not enabling those options for further processing in the dialog.

Asterisk does not reject Require: unknown options tags: The Asterisk checks for the Require: header in an inbound INVITE and if it sees one or more options tags from the standard sets of options that it does not support then it rejects the INVITE with 420 response. However, if the Require: header contains one or more option tags that are not from standard set of option tags currently coded in Asterisk than Asterisk ignores those options and fails to reject the request with a 420 response. New options tags are frequently being added to the SIP protocol suite let alone proprietary option tags. This issue has now been resolved.

Asterisk processes old SDP as new SDP: The Asterisk was treating an old, unmodified SDP in a RE-INVITE as a new SDP and was incorrectly incrementing its SDP version number in response. The dialog context (sip_pvt) in Asterisl chan_sip.c now keeps track of previous SDP version of the remote end-point and if it does not change in a new RE-INVITE than Asterisk does not alter its own SDP version number either. This issue has now been resolved.

> Note: There is a related issue where Asterisk increments SDP version number from what it sent in183's SDP to what it sends in 200's SDP. That issue has not been resolved because it does not directly effect Session-Timers, but with the "old SDP" hooks now in place, the issue could be resolved somewhat easily.

6. Diagnostics and Debugging

The output of CLI "sip show settings" command has been enhanced to print session-timers related configuration flags. Below is a sample output (note the new output lines in red text):

```
*CLI> sip show settings


Global Settings:
----------------
  SIP Port:               5060
  Bindaddress:            0.0.0.0
```

```
  Videosupport:           No
  Textsupport:            No
  AutoCreatePeer:         No
  MatchAuthUsername:      No
  Allow unknown access:   Yes
  Allow subscriptions:    Yes
  Allow overlap dialing:  No
  Promsic. redir:         No
  SIP domain support:     No
  Call to non-local dom.: Yes
  URI user is phone no:   No
  Our auth realm          asterisk
  Realm. auth:            No
  Always auth rejects:    No
  Call limit peers only:  No
  Direct RTP setup:       No
  User Agent:             Raj's Asterisk
  Reg. context:           (not set)
  Regexten on Qualify:    No
  Caller ID:              asterisk
  From: Domain:
  Record SIP history:     Off
  Call Events:            Off
  IP ToS SIP:             CS0
  IP ToS RTP audio:       CS0
  IP ToS RTP video:       CS0
  IP ToS RTP text:        CS0
  802.1p CoS SIP:         4
  802.1p CoS RTP audio:   5
  802.1p CoS RTP video:   6
  802.1p CoS RTP text:    0
  T38 fax pt UDPTL:       No
  RFC2833 Compensation:   No
  Jitterbuffer enabled:   No
  Jitterbuffer forced:    No
  Jitterbuffer max size:  -1
  Jitterbuffer resync:    -1
  Jitterbuffer impl:
  Jitterbuffer log:       No
  SIP realtime:           Disabled

Network Settings:
---------------------------
  SIP address remapping:  Disabled, no localnet list
  Externhost:             <none>
  Externip:               0.0.0.0:0
  Externrefresh:          10
  Internal IP:            127.0.0.1:5060
  STUN server:            0.0.0.0:0

Global Signalling Settings:
---------------------------
  Codecs:                 0x8000e (gsm|ulaw|alaw|h263)
  Codec Order:            none
  T1 minimum:             100
  Relax DTMF:             No
  Compact SIP headers:    No
```

```
  RTP Keepalive:          0 (Disabled)
  RTP Timeout:            0 (Disabled)
  RTP Hold Timeout:       0 (Disabled)
  MWI NOTIFY mime type:   application/simple-message-summary
  DNS SRV lookup:         Yes
  Pedantic SIP support:   No
  Reg. min duration       60 secs
  Reg. max duration:      3600 secs
  Reg. default duration:  120 secs
  Outbound reg. timeout:  20 secs
  Outbound reg. attempts: 0
  Notify ringing state:   Yes
  Notify hold state:      No
  SIP Transfer mode:      open
  Max Call Bitrate:       384 kbps
  Auto-Framing:           No
  Outb. proxy:            <not set>
  Session Timer Mode:     Originate
  Session Refresher:      uac
  Session-Expires:        1800 secs
  Min-SE:                 90 secs

Default Settings:
-----------------
  Context:                default
  Nat:                    RFC3581
  DTMF:                   rfc2833
  Qualify:                0
  Use ClientCode:         No
  Progress inband:        Never
  Language:
  MOH Interpret:          default
  MOH Suggest:
  Voice Mail Extension:   voicemail

----
*CLI>
```

The output of ‘sip show user’ and ‘sip show peer’ commands have been enhanced to print session-timers related flags.

```
*CLI> sip show user 1000


  * Name       : 1000
  Secret       : <Not set>
  MD5Secret    : <Not set>
  Context      : default
  Language     :
  AMA flags    : Unknown
  Transfer mode: open
  MaxCallBR    : 384 kbps
  CallingPres  : Presentation Allowed, Not Screened
  Call limit   : 0
  Callgroup    :
  Pickupgroup  :
  Callerid     : "" <>
```

```
  ACL          : No
  Codec Order  : (gsm:20,ulaw:20,alaw:20)
  Auto-Framing:  No
  Session-Timers   : Originate
  Session-Refresher: uas
  Session-Expires  : 120 secs
  Min-SE           : 90 secs


*CLI> sip show peer 2000


  * Name       : 2000
  Secret       : <Set>
  MD5Secret    : <Not set>
  Context      : default
  Subscr.Cont. : <Not set>
  Language     :
  AMA flags    : Unknown
  Transfer mode: open
  CallingPres  : Presentation Allowed, Not Screened
  Callgroup    :
  Pickupgroup  :
  Mailbox      :
  VM Extension : asterisk
  LastMsgsSent : 32767/65535
  Call limit   : 0
  Dynamic      : Yes
  Callerid     : "" <>
  MaxCallBR    : 384 kbps
  Expire       : 127
  Insecure     : no
  Nat          : Always
  ACL          : No
  T38 pt UDPTL : No
  CanReinvite  : No
  PromiscRedir : No
  User=Phone   : No
  Video Support: No
  Text Support : No
  Trust RPID   : No
  Send RPID    : No
  Subscriptions: Yes
  Overlap dial : No
  DTMFmode     : rfc2833
  ToHost       :
  Addr->IP     : 192.168.15.100 Port 2915
  Defaddr->IP  : 0.0.0.0 Port 5060
  Def. Username:
  SIP Options  : (none)
  Codecs       : 0xe (gsm|ulaw|alaw)
  Codec Order  : (gsm:20,ulaw:20,alaw:20)
  Auto-Framing:  No
  Status       : Unmonitored
  Useragent    : RTC/1.3.5470 (Messenger 5.1.0701)
  Reg. Contact : sip:192.168.15.100:10201
```

```
  Session-Timers    : Originate
  Session-Refresher: uac
  Session-Expires   : 100 secs
  Min-SE            : 92 secs
```

The ‘sip show channel <call-id>’ CLI command has been enhanced to output session-timers related flags that apply to that particular session. Below is a sample output (note the new output lines in red text):

```
*CLI> sip show channel NjE4MmZkMTl

  * SIP Call
  Curr. trans. direction:  Incoming
  Call-ID:                NjE4MmZkMTlkMDU3NDZjMWUyYzBhOTFjMGZlOGVlMDI.
  Owner channel ID:       SIP/1000-08a626d0
  Our Codec Capability:   4
  Non-Codec Capability (DTMF):   1
  Their Codec Capability:   1038
  Joint Codec Capability:   4
  Format:                 0x4 (ulaw)
  T.38 support            No
  Video support           No
  MaxCallBR:              384 kbps
  Theoretical Address:    159.63.73.115:5061
  Received Address:       159.63.73.115:5061
  SIP Transfer mode:      open
  NAT Support:            RFC3581
  Audio IP:               159.63.73.11 (local)
  Our Tag:                as5a01a88b
  Their Tag:              75710f57
  SIP User agent:         X-Lite release 1006e stamp 34025
  Username:               1000
  Peername:               1000
  Original uri:           sip:1000@159.63.73.115:5061
  Caller-ID:              1000
  Need Destroy:           No
  Last Message:           Rx: ACK
  Promiscuous Redir:      No
  Route:                  sip:1000@159.63.73.115:5061
  DTMF Mode:              rfc2833
  SIP Options:            timer
  Session-Timer:          Active
  Session-Timer Interval:  150
  Session-Timer Refresher: uac
  Session-Timer Expirys:   0
  Session-Timer Sched Id:  9
```

7. Sample sip.conf

```
;--------------------------- SIP Session-Timers -----------------------
; SIP Session-Timers provide a keep-alive mechanism for active SIP
; sessions. They can detect SIP channels that do not terminate through
; normal signaling procedures and reclaim such channels. Session-Timers
; can be configured globally or at user or peer level.
;

session-timers=accept
session-expires=3600
session-minse=90
session-refresher=uac


[2000]
type=peer
regexten=2000
secret=2000
host=dynamic
nat=yes
canreinvite=no
disallow=all
allow=gsm
allow=ulaw
allow=alaw
session-timers=originate
session-expires=1800
session-minse=90
session-refresher=uas
```

8. Test Cases

#	Test Description	Expected Result	Pass/ Fail
	Asterisk receives an incoming call. session-timers = originate UAC does not support Session-Timers.	Asterisk periodically refreshes session periodically sending RE-INVITE based on session-expires setting.	**Pass**
	Asterisk receives an incoming call. session-timers = accept UAC does not support Session-Timers.	Asterisk does not periodically refresh the session.	**Pass**
	Asterisk receives an incoming call. session-timer = refuse UAC does not support Session-Timers.	Asterisk does not periodically refresh the session.	**Pass**
	Asterisk receives an incoming call. session-timer = accept UAC supports Session-Timers. UAC sends Session-Expires lower that Asterisk's Min-SE	Asterisk returns a 422 response. UAC reverts back with updated Session-Expires. Call completes as normal and session-timers are run.	**Pass**
	Asterisk receives an incoming call. session-timer = accept session-refresher = uas UAC supports Session-Timers. UAC sends Session-Expires without the refresher parameter.	Asterisk places refresher=uas in 200 response. Asterisk periodically sends RE-INVITEs.	**Pass**
	Asterisk receives an incoming call. session-timer = accept session-refresher = uac UAC supports Session-Timers. UAC sends Session-Expires without the refresher parameter.	Asterisk places refresher=uac in 200 response. Asterisk waits for RE-INVITEs and resets timer on receiving one.	**Pass**
	Asterisk receives an incoming call. session-timer = accept session-refresher = uas UAC supports Session-Timers. UAC sends Session-Expires without the refresher parameter. Failure condition: UAC becomes unreachable mid-session. Asterisk sends RE-INVITE which times out.	Asterisk tears down the channel.	**Pass**
	Asterisk receives an incoming call. session-timers = refuse UAC supports Session-Timers.	Asterisk does not run session-timers	**Pass**
	Asterisk receives an incoming call. session-timers = refuse	Asterisk rejects the request with a 420 response	**Pass**

	UAC requires Session-Timers.		
	Asterisk makes an outbound call. session-timer = originate Asterisk sends an INVITE which gets rejected by a 422 response.	Asterisk reattempts the INVITE while using the Min-SE in 422 as the Session-Expires in the new INVITE. Note: This scenario was tested using SIPp (X-Lite does not respond with 422).	**Pass**
	Asterisk makes an outbound call. session-timer = originate UAS does not support session-timer	Asterisk periodically refreshes session	**Pass**
	Asterisk makes an outbound call. session-timer = originate UAS supports session-timer	Asterisk runs session-timers based on the refresher- selected by the UAS and the Session-Expires: selected by the UAS	**Pass**
	Asterisk makes an outbound call. session-timer = originate UAS supports session-timer. Asterisk's Session-Expires: is lower than UAS' local Min-SE	Asterisk receives a 422 response. Asterisk sends a new INVITE this time with Session-Expires equal to 422's Min-SE value. Call goes through.	**Pass**
	Asterisk makes an outbound call. session-timers = accept UAS supports session-timer	Asterisk runs session-timers based on the refresher- selected by the UAS and the Session-Expires: selected by the UAS	**Pass**
	Asterisk makes an outbound call. session-timers = refuse UAS supports session-timers	Asterisk does not run session-timers	**Pass**