Modular And Test Driven SIP Routing With Lua

Sebastian Damm

E: damm@sipgate.de
T: @_SebastianDamm

Who we are, what we do

- Düsseldorf based VoIP service provider (since 2004)
- Active in Germany and UK
- Full MVNO in the Telefónica network
- Private and Business customers
- VoIP and Mobile products
- Some 100k active customers
- Almost 100 million minutes each month

Of course, we use Kamailio

Of course, we use Kamailio

All over our network.

Of course, we use Kamailio

All over our network.
But we have a huge problem!

What problem?

at sipgate

- methods of software development have changed:
  - Agile, fast cycles, many releases
  - small modules
  - unit tests, continuous deployment
  - permanent refactoring of functions

What problem?

at sipgate

- methods of software development have changed:
  - Agile, fast cycles, many releases
  - small modules
  - unit tests, continuous deployment
  - permanent refactoring of functions
- telephony side:
  - "organically grown" routing logic (since 2004)
  - 40 developers, but only 4 people can read/write it
  - learnings from software development haven't made it to this part

What problem?

Nobody wants to touch it!

What problem?

1. Readability

What problem?

2. damm@mindel: /etc/kamailio (ssh)

```
else if( uri =~ "^sip:[1-9]{1}[0-9]{6}[pgy][0-9]+@(.*)" ) {
        # Mark Call as LI needed
        setsflag(0);

        route(34);
        route(13);
        route(12);
        route(10);
        route(blockNonVerifiedOutgoing);
        route(checkBlockedIn);
        route(23);
        route(24);
        route(handleNAT);

        route(checkDebugGroupOnFromUser);
        route(checkDebugOnIncomingCalls);

        if (!isflagset(15)) {
                append_hf("X-CONF: transfer-in\r\n");
        }
        route(careAboutXCID);
        $(avp(s:calltype)) = $(avp(s:calltype)) + ',group';

        $ru = $(avp(s:mastersipid-in));
        route(sendToTransferOrForward);
        $rU = $oU;

        xlog("L_NOTICE", "main_route: Some log line here. F=$fu T=$tu R=$ru\n");
}
                                                        1206,0-1      33%
```

sipgate

What problem?

1. Readability

What problem?

2. Config file size

What problem?

What problem?

2. Config file size

What problem?

3. Testability

What problem?

What problem?

It's a mess!

But how do we fix that?

But how do we fix that?

- app_*:
  - app_lua
  - app_python
  - app_jsdt
  - app_java, app_mono, app_perl
  - missing: app_ruby ;)

But how do we fix that?

- app_*:
  - app_lua
  - app_python
  - app_jsdt
  - app_java, app_mono, app_perl
  - missing: app_ruby ;)

- KEMI
  - lua
  - python
  - jsdt
  - sqlang

Why Lua?

1. Speed

Interpreter	Average	Min	Max
Native	302.28	6	3824
Lua	308.32	6	3596
Python	393.71	23	3266

All times in Microseconds.
Source: https://www.kamailio.org/wiki/devel/config-engines#interpreters_performances

Why Lua?

2. Easy to learn

- None of us had written Lua (production) code before.
- If you've written code before, you'll be able to start right away.
- Less fear of changing something within Kamailio.

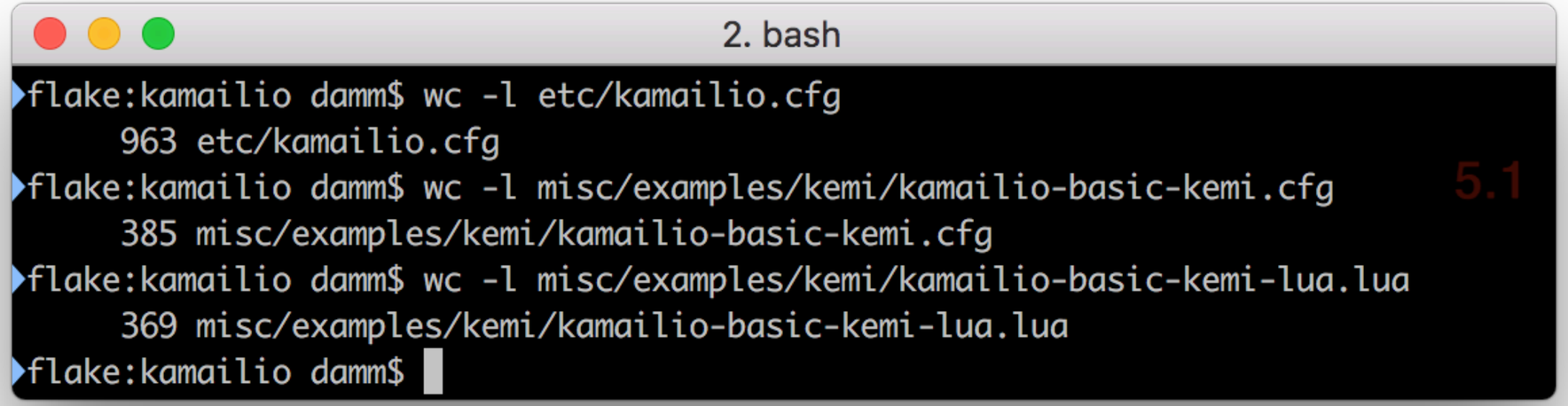
Why Lua?

3. You can work test-driven

- Impossible with native Kamailio language
- Know if you broke something before you deploy it!

Standard way: one big Lua file

Still many lines of code



Standard way: one big Lua file

Still have to know Kamailio syntax all the time

```
function ksr_request_route()

   -- per request initial checks
   ksr_route_reqinit();

   -- NAT detection
   ksr_route_natdetect();

   -- CANCEL processing
   if KSR.pv.get("$rm") == "CANCEL" then
      if KSR.tm.t_check_trans()>0 then
         ksr_route_relay();
      end
      return 1;
   end

   -- handle requests within SIP dialogs
   ksr_route_withindlg();

   -- -- only initial requests (no To tag)

   -- handle retransmissions
   if KSR.tmx.t_precheck_trans()>0 then
      KSR.tm.t_check_trans();
      return 1;
   end
   if KSR.tm.t_check_trans()==0 then return 1 end

   -- authentication
   ksr_route_auth();

   -- remove preloaded route headers
   KSR.hdr.remove("Route");
   if string.find("INVITE|SUBSCRIBE", KSR.pv.get("$rm")) then
      KSR.rr.record_route();
   end

   -- account only INVITEs
   if KSR.pv.get("$rm")=="INVITE" then
      KSR.setflag(FLT_ACC); -- do accounting
   end

   -- dispatch requests to foreign domains
   ksr_route_sipout();

   -- -- requests for my local domains

   -- handle registrations
   ksr_route_registrar();

   if KSR.pv.is_null("$rU") then
      -- request with no Username in RURI
      KSR.sl.sl_send_reply(484,"Address Incomplete");
      return 1;
   end

   -- user location service
   ksr_route_location();

   return 1;
end
```

Standard way: one big Lua file

Still not testable

```
function ksr_request_route()

   -- per request initial checks
   ksr_route_reqinit();

   -- NAT detection
   ksr_route_natdetect();

   -- CANCEL processing
   if KSR.pv.get("$rm") == "CANCEL" then
      if KSR.tm.t_check_trans()>0 then
         ksr_route_relay();
      end
      return 1;
   end

   -- handle requests within SIP dialogs
   ksr_route_withindlg();

   -- -- only initial requests (no To tag)

   -- handle retransmissions
   if KSR.tmx.t_precheck_trans()>0 then
      KSR.tm.t_check_trans();
      return 1;
   end
   if KSR.tm.t_check_trans()==0 then return 1 end

   -- authentication
   ksr_route_auth();

   -- remove preloaded route headers
   KSR.hdr.remove("Route");
   if string.find("INVITE|SUBSCRIBE", KSR.pv.get("$rm")) then
      KSR.rr.record_route();
   end

   -- account only INVITEs
   if KSR.pv.get("$rm")=="INVITE" then
      KSR.setflag(FLT_ACC); -- do accounting
   end

   -- dispatch requests to foreign domains
   ksr_route_sipout();

   -- -- requests for my local domains

   -- handle registrations
   ksr_route_registrar();

   if KSR.pv.is_null("$rU") then
      -- request with no Username in RURI
      KSR.sl.sl_send_reply(484,"Address Incomplete");
      return 1;
   end

   -- user location service
   ksr_route_location();

   return 1;
end
```

How about a library?

- Encapsulate Kamailio functions in easy-to-use native functions

```
if KSR.pv.get("$rm") == "INVITE" then
  [..]
end

if kamailio.is_invite() then
  [..]
end
```

- Re-use often used calls
- Testable
- Fix logic errors before deploying them into production

How about a library?

Reduce the main script to a minimum

```
kamailio = require("kamailio")

function ksr_request_route()
  kamailio.process_request()
end
```

So how do we get there?

- Install dependencies
  - Interpreter: lua, version 5.1 (apt-get install lua-5.1)
  - Testing framework: busted (apt-get install lua-busted)
- Create your library.
- Define your function.
- Write tests.
- Make the function work.

Create your library

- Can be just one file containing all your methods
  `/etc/kamailio/kamailio-functions.lua`
  `/usr/local/share/lua/5.1/kamailio-functions.lua`

Create your library

- Can be just one file containing all your methods
  `/etc/kamailio/kamailio-functions.lua`
  `/usr/local/share/lua/5.1/kamailio-functions.lua`

- Can be a directory containing multiple files

```
/usr/local/share/lua/5.1/kamailio/
├── actions.lua
├── init.lua
├── message.lua
├── message_state.lua
├── nathandling.lua
├── security.lua
└── traffic.lua
```

Create your library

- Can be just one file containing all your methods
  `/etc/kamailio/kamailio-functions.lua`
  `/usr/local/share/lua/5.1/kamailio-functions.lua`

- Can be a directory containing multiple files
  `/usr/local/share/lua/5.1/kamailio/`

```
├── actions.lua
├── init.lua
├── message.lua
├── message_state.lua
├── nathandling.lua
├── security.lua
└── traffic.lua
```

Create your library

- Can be just one file containing all you
  /etc/kamailio/kamailio-functi
  /usr/local/share/lua/5.1/kama

- Can be a directory containing multipl
  /usr/local/share/lua/5.1/kama
  ├── actions.lua
  ├── init.lua
  ├── message.lua
  ├── message_state.lua
  ├── nathandling.lua
  ├── security.lua
  └── traffic.lua

```
damm@stretch:~$ lua
Lua 5.1.5  Copyright (C) 1994-2012 Lua.org, PUC-Rio
> foo = require "foo"
stdin:1: module 'foo' not found:
    no field package.preload['foo']
    no file './foo.lua'
    no file '/usr/local/share/lua/5.1/foo.lua'
    no file '/usr/local/share/lua/5.1/foo/init.lua'
    no file '/usr/local/lib/lua/5.1/foo.lua'
    no file '/usr/local/lib/lua/5.1/foo/init.lua'
    no file '/usr/share/lua/5.1/foo.lua'
    no file '/usr/share/lua/5.1/foo/init.lua'
    no file './foo.so'
    no file '/usr/local/lib/lua/5.1/foo.so'
    no file '/usr/lib/x86_64-linux-gnu/lua/5.1/foo.so'
    no file '/usr/lib/lua/5.1/foo.so'
    no file '/usr/local/lib/lua/5.1/loadall.so'
stack traceback:
    [C]: in function 'require'
    stdin:1: in main chunk
    [C]: ?
>
```

Create your library

```
traffic = require "kamailio.traffic"
message = require "kamailio.message"
security = require "kamailio.security"
rex = require "rex_pcre"

local kamailio = {}

function kamailio:process_request()
  [..]
end

function kamailio:process_reply()
  [..]
end

return kamailio
```

Create your library

```
traffic = require "kamailio.traffic"
message = require "kamailio.message"
security = require "kamailio.security"
rex = require "rex_pcre"

local kamailio = {}

function kamailio:process_request()
  [..]
end

function kamailio:process_reply()
  [..]
end

return kamailio
```

Write your function

kamailio.cfg

```
route["clir"] {
  if ($rU =~ "^\*31[0-9]+$") {
    strip(3);
    setflag(21);
  } else if ($rU =~ "^\*31[*#][0-9]+$") {
    strip(4);
    setflag(21);
  } else if ($rU =~ "^\*31\%23[0-9]+$") {
    strip(6);
    setflag(21);
  }
  if (isflagset(21) && method=="INVITE") {
    xlog("L_NOTICE", "User wants to suppress his number for this call. F=$fU T=$tU
D=$fn\n");
    append_hf("Privacy: id\r\n");
  }
}
```

Write your function

kamailio/headers.lua

```
rex = require "rex_pcre"
message = require "kamailio.message"

local headers = {}

function headers.get_request_user()
  return KSR.pv.get("$rU")
end

function headers.set_request_user(value)
  KSR.pv.sets("$rU", value)
end

function headers.set_privacy_header()
  KSR.hdr.append("Privacy: id")
end

function headers.suppress_cid_if_needed()
  request_user = headers.get_request_user()
  if rex.find(request_user, "^\\*31([#]|%23)?") then
    headers.set_request_user(rex.gsub(request_user, "^\\*31([#]|%23)?", ""))
    if message.is_invite() then headers.set_privacy_header() end
  end
end

return headers
```

Write your function

kamailio/headers.lua

```lua
rex = require "rex_pcre"
message = require "kamailio.message"

local headers = {}

function headers.get_request_user()
  return KSR.pv.get("$rU")
end

function headers.set_request_user(value)
  KSR.pv.sets("$rU", value)
end

function headers.set_privacy_header()
  KSR.hdr.append("Privacy: id")
end

function headers.suppress_cid_if_needed()
  request_user = headers.get_request_user()
  if rex.find(request_user, "^\\*31([#]|%23)?") then
    headers.set_request_user(rex.gsub(request_user, "^\\*31([#]|%23)?", ""))
    if message.is_invite() then headers.set_privacy_header() end
  end
end

return headers
```

Write your function

kamailio/headers.lua

```
rex = require "rex_pcre"
message = require "kamailio.message"

local headers = {}

function headers.get_request_user()
  return KSR.pv.get("$rU")
end

function headers.set_request_user(value)
  KSR.pv.sets("$rU", value)
end

function headers.set_privacy_header()
  KSR.hdr.append("Privacy: id")
end

function headers.suppress_cid_if_needed()
  request_user = headers.get_request_user()
  if rex.find(request_user, "^\\*31([#]|%23)?") then
    headers.set_request_user(rex.gsub(request_user, "^\\*31([#]|%23)?", ""))
    if message.is_invite() then headers.set_privacy_header() end
  end
end

return headers
```

Write your function

kamailio/headers.lua

```
rex = require "rex_pcre"
message = require "kamailio.message"

local headers = {}

function headers.get_request_user()
  return KSR.pv.get("$rU")
end

function headers.set_request_user(value)
  KSR.pv.sets("$rU", value)
end

function headers.set_privacy_header()
  KSR.hdr.append("Privacy: id")
end

function headers.suppress_cid_if_needed()
  request_user = headers.get_request_user()
  if rex.find(request_user, "^\\*31([#]|%23)?") then
    headers.set_request_user(rex.gsub(request_user, "^\\*31([#]|%23)?", ""))
    if message.is_invite() then headers.set_privacy_header() end
  end
end

return headers
```

Test it

spec/headers_spec.lua

```
require 'busted.runner'()

local headers = require "../kamailio/headers"

describe("Check and enable CLIR ->", function()
  it("Caller dials *31021112345", function()
    -- Call the function
    headers.suppress_cid_if_needed()
    -- Now check if everything is as expected
    assert.spy(KSR.pv.sets).was.called_with( "$rU", "021112345" )
    assert.spy(KSR.hdr.append).was.called_with("Privacy: id")
 end)
end)
```

Test it

spec/headers_spec.lua

```lua
require 'busted.runner'()

local headers = require "../kamailio/headers"

describe("Check and enable CLIR ->", function()
  it("Caller dials *31021112345", function()
    -- Call the function
    headers.suppress_cid_if_needed()
    -- Now check if everything is as expected
    assert.spy(KSR.pv.sets).was.called_with( "$rU", "021112345" )
    assert.spy(KSR.hdr.append).was.called_with("Privacy: id")
 end)
end)
```

Test it

spec/headers_spec.lua

```
require 'busted.runner'()

local headers = require "../kamailio/headers"

describe("Check and enable CLIR ->", function()
  it("Caller dials *31021112345", function()
    -- Call the function
    headers.suppress_cid_if_needed()
    -- Now check if everything is as expected
    assert.spy(KSR.pv.sets).was.called_with( "$rU", "021112345" )
    assert.spy(KSR.hdr.append).was.called_with("Privacy: id")
  end)
end)
```

Test it

spec/headers_spec.lua

```lua
require 'busted.runner'()

local headers = require "../kamailio/headers"

describe("Check and enable CLIR ->", function()
  it("Caller dials *31021112345", function()
    -- Call the function
    headers.suppress_cid_if_needed()
    -- Now check if everything is as expected
    assert.spy(KSR.pv.sets).was.
    assert.spy(KSR.hdr.append).w
 end)
end)
```

```
2. damm@stretch: ~/kamailioworld (ssh)
damm@stretch:~/kamailioworld$ busted tests/test_headers.lua
*
0 successes / 0 failures / 1 error / 0 pending : 0.00134 seconds

Error → tests/test_headers.lua @ 22
Check and enable CLIR Caller dials *31021112345
./src////kamailio/headers.lua:7: attempt to index global 'KSR' (a nil value)
damm@stretch:~/kamailioworld$
```

Fix your tests

```
require 'busted.runner'()

local function init_mock(options)
  -- mock global variable 'KSR'
  local ksr_mock = {
    pv = {
      get = function(key)
        if key == "$rU" then
          if options.rU ~= nil then return options.rU else return "01234567" end
        end
      end,
      sets = function(k, v) end,
    },
    hdr = {
      append = function(header) end,
    }
  }
  _G["KSR"] = mock(ksr_mock)

  local message_mock = {
    is_invite = function() return options.is_invite end
  }
  _G["message"] = mock(message_mock)
end

local headers = require "kamailio/headers"
[..]
```

Fix your tests

```
require 'busted.runner'()

local function init mock(options)
  -- mock global variable 'KSR'
  local ksr_mock = {
    pv = {
      get = function(key)
        if key == "$rU" then
          if options.rU ~= nil then return options.rU else return "01234567" end
        end
      end,
      sets = function(k, v) end,
    },
    hdr = {
      append = function(header) end,
    }
  }
  _G["KSR"] = mock(ksr_mock)

  local message_mock = {
    is_invite = function() return options.is_invite end
  }
  _G["message"] = mock(message_mock)
end

local headers = require "kamailio/headers"
[..]
```

Fix your tests

```
require 'busted.runner'()

local function init_mock(options)
  -- mock global variable 'KSR'
  local ksr_mock = {
    pv = {
      get = function(key)
        if key == "$rU" then
          if options.rU ~= nil then return options.rU else return "01234567" end
        end
      end,
      sets = function(k, v) end,
    },
    hdr = {
      append = function(header) end,
    }
  }
  _G["KSR"] = mock(ksr_mock)

  local message_mock = {
    is_invite = function() return options.is_invite end
  }
  _G["message"] = mock(message_mock)
end

local headers = require "kamailio/headers"
[..]
```

Fix your tests

```
require 'busted.runner'()

local function init_mock(options) [..] end

describe("Check and enable CLIR ->", function()
  it("Caller dials *31021112345", function()
    -- Initialize the mock
    init_mock{ rU = "*31021112345", is_invite = true }
    -- Call the function
    headers.suppress_cid_if_needed()
    -- Now check if everything is as expected
    assert.spy(KSR.pv.sets).was.called_with("$rU", "021112345")
    assert.spy(KSR.hdr.append).was.called_with("Privacy: id")
  end)
end)
```

Fix your tests

```
require 'busted.runner'()

local function init_mock(options) [..] end

describe("Check and enable CLIR ->", function()
  it("Caller dials *31021112345", function()
    -- Initialize the mock
    init_mock{ rU = "*31021112345", is_invite = true }
    -- Call the function
    headers.suppress_cid_if_needed()
    -- Now check if everything is as expected
    assert.spy(KSR.pv.sets).was.called_with("$rU", "021112345")
    assert.spy(KSR.hdr.append).was.called_with("Privacy: id")
  end)
end)
```

Fix your tests

```lua
require 'busted.runner'()

local function init_mock(options) [..] end

describe("Check and enable CLIR ->", function()
  it("Caller dials *31021112345", function()
    -- Initialize the mock
    init_mock{ rU = "*31021112345", is_invite = true }
    -- Call the function
    headers.suppress_cid_if_needed()
    -- Now check if everything is as expected
    assert.spy(KSR.pv.sets).was.called_with("$rU", "021112345")
    assert.spy(KSR.hdr.append).w
  end)
end)
```

```
2. damm@stretch: ~/kamailioworld (ssh)
damm@stretch:~/kamailioworld$ busted tests/test_headers.lua
●
1 success / 0 failures / 0 errors / 0 pending : 0.00123 seconds
damm@stretch:~/kamailioworld$
```

Add more tests

```
require 'busted.runner'()

local function init_mock(options) [..] end

describe("Check and enable CLIR ->", function()
  it("Caller dials *31021112345", function() .. end)
  it("Caller dials *31*021112345", function() .. end)
  it("Caller dials *31#021112345", function() .. end)
  it("Caller dials *31%23021112345", function() .. end)
  it("Caller dials +4921112345", function() .. end)
  it("Caller dials *31+4921112345, non-INVITE", function() .. end)
end)
```

Add more tests

```
require 'busted.runner'()

local function init_mock(options) [..] end

describe("Check and enable CLIR ->", function()
  it("Caller dials *31021112345", function() .. end)
  it("Caller dials *31*021112345", function() .. end)
  it("Caller dials *31#021112345", function() .. end)
  it("Caller dials *31%23021112345", function() .. end)
  it("Caller dials +4921112345", function()
    init_mock{ rU = "+4921112345", is_invite = true }
    headers.suppress_cid_if_needed()
    assert.spy(KSR.pv.sets).was.called(0)
    assert.spy(KSR.hdr.append).was.called(0)
  end)
  it("Caller dials *31+4921112345, non-INVITE", function()
    init_mock{ rU = "*31+4921112345" }
    headers.suppress_cid_if_needed()
    assert.spy(KSR.pv.sets).was.called_with("$rU", "+4921112345")
    assert.spy(KSR.hdr.append).was.called(0)
  end)
end)
```

Add more tests

```
require 'busted.runner'()

local function init_mock(options) [..] end

describe("Check and enable CLIR ->", function()
  it("Caller dials *31021112345", function() .. end)
  it("Caller dials *31*021112345", function() .. end)
  it("Caller dials *31#021112345", function() .. end)
  it("Caller dials *31%23021112345", function() .. end)
  it("Caller dials +4921112345", function()
    init_mock{ rU = "+4921112345", is_invite = true }
    headers.suppress_cid_if_needed()
    assert.spy(KSR.pv.sets).was.called(0)
    assert.spy(KSR.hdr.append).was.called(0)
  end)
  it("Caller dials *31+4921112345, non-INVITE", function()
    init_mock{ rU = "*31+4921112345" }
    headers.suppress_cid_if_needed()
    assert.spy(KSR.pv.sets).was.called_with("$rU", "+4921112345")
    assert.spy(KSR.hdr.append).was.called(0)
  end)
end)
```

Add more tests

```
require 'busted.runner'()

local function init_mock(options) [..] end

describe("Check and enable CLIR ->", function()
  it("Caller dials *31021112345", function() .. end)
  it("Caller dials *31*021112345", function() .. end)
  it("Caller dials *31#021112345", function() .. end)
  it("Caller dials *31%23021112345", function() .. end)
  it("Caller dials +4921112345", function()
    init_mock{ rU = "+4921112345", is_invite = true }
    headers.suppress_cid_if_needed()
    assert.spy(KSR.pv.sets).was.called(0)
    assert.spy(KSR.hdr.append).was.called(0)
  end)
  it("Caller dials *31+4921112345, non-INVITE", function()
    init_mock{ rU = "*31+4921112345" }
    headers.suppress_cid_if_needed()
    assert.spy(KSR.pv.sets).was.called_with("$rU", "+4921112345")
    assert.spy(KSR.hdr.append).was.called(0)
  end)
end)
```

Add more tests

```
require 'busted.runner'()

local function init_mock(options) [..] end

describe("Check and enable CLIR ->", function()
  it("Caller dials *31021112345", function() .. end)
  it("Caller dials *31*021112345", function() .. end)
  it("Caller dials *31#021112345", function() .. end)
  it("Caller dials *31%230211123
  it("Caller dials +4921112345",
    init_mock{ rU = "+4921112345
    headers.suppress_cid_if_need
    assert.spy(KSR.pv.sets).was.
    assert.spy(KSR.hdr.append).w
  end)
  it("Caller dials *31+492111234
    init_mock{ rU = "*31+4921112
    headers.suppress_cid_if_need
    assert.spy(KSR.pv.sets).was.
    assert.spy(KSR.hdr.append).w
  end)
end)
```

```
2. damm@stretch: ~/kamailioworld (ssh)
damm@stretch:~/kamailioworld$ busted -l spec/headers_spec.lua
spec/headers_spec.lua:29: Check and enable CLIR -> Caller dials *31021112345
spec/headers_spec.lua:38: Check and enable CLIR -> Caller dials *31*021112345
spec/headers_spec.lua:47: Check and enable CLIR -> Caller dials *31#021112345
spec/headers_spec.lua:56: Check and enable CLIR -> Caller dials *31%23021112345
spec/headers_spec.lua:65: Check and enable CLIR -> Caller dials +4921112345
spec/headers_spec.lua:74: Check and enable CLIR -> Caller dials *31+4921112345, non-INVITE
damm@stretch:~/kamailioworld$ busted spec/headers_spec.lua
•■••••
5 successes / 1 failure / 0 errors / 0 pending : 0.008387 seconds

Failure → spec/headers_spec.lua @ 38
Check and enable CLIR -> Caller dials *31*021112345
spec/headers_spec.lua:44: Function was not called with the arguments
damm@stretch:~/kamailioworld$
```

Fix your code

kamailio/headers.lua

```
rex = require "rex_pcre"
message = require "kamailio.message"

local headers = {}

function headers.get_request_user()
  return KSR.pv.get("$rU")
end

function headers.set_request_user(value)
  KSR.pv.sets("$rU", value)
end

function headers.set_privacy_header()
  KSR.hdr.append("Privacy: id")
end

function headers.suppress_cid_if_needed()
  request_user = headers.get_request_user()
  if rex.find(request_user, "^\\*31([#]|%23)?") then
    headers.set_request_user(rex.gsub(request_user, "^\\*31([#]|%23)?", ""))
    if message.is_invite() then headers.set_privacy_header() end
  end
end

return headers
```

Fix your code

kamailio/headers.lua

```
rex = require "rex_pcre"
message = require "kamailio.message"

local headers = {}

function headers.get_request_user()
  return KSR.pv.get("$rU")
end

function headers.set_request_user(value)
  KSR.pv.sets("$rU", value)
end

function headers.set_privacy_header()
  KSR.hdr.append("Privacy: id")
end

function headers.suppress_cid_if_needed()
  request_user = headers.get_request_user()
  if rex.find(request_user, "^\*31([#]|%23)?") then
    headers.set_request_user(rex.gsub(request_user, "^\*31([#]|%23)?", ""))
    if message.is_invite() then headers.set_privacy_header() end
  end
end

return headers
```

Fix your code

kamailio/headers.lua

```
rex = require "rex_pcre"
message = require "kamailio.message"

local headers = {}

function headers.get_request_user()
  return KSR.pv.get("$rU")
end

function headers.set_request_user(value)
  KSR.pv.sets("$rU", value)
end

function headers.set_privacy_header()
  KSR.hdr.append("Privacy: id")
end

function headers.suppress_cid_if_needed()
  request_user = headers.get_request_user()
  if rex.find(request_user, "^\\*31([*#]|%23)?") then
    headers.set_request_user(rex.gsub(request_user, "^\\*31([*#]|%23)?", ""))
    if message.is_invite() then headers.set_privacy_header() end
  end
end

return headers
```

Fix your code

kamailio/headers.lua

```
rex = require "rex_pcre"
message = require "kamailio.message"

local headers = {}

function headers.get_request_user()
  return KSR.pv.get("$rU")
end

function headers.set_request_user(value)
  KSR.pv.sets("$rU", value)
end

function headers.set_privacy_header()
  KSR.hdr.append("Privacy: id")
end

function headers.suppress_cid_if_needed()
  request_user = headers.get_request_user()
  if rex.find(request_user, "^\*31([*#]|%23)?") then
    headers.set_request_user(rex.gsub(request_user, "^\*31([*#]|%23)?", ""))
    if message.is_invite() then headers.set_privacy_header() end
  end
end

return headers
```

Rerun your tests

```
require 'busted.runner'()

local function init_mock(options) [..] end

describe("Check and enable CLIR ->", function()
  it("Caller dials *31021112345", function() .. end)
  it("Caller dials *31*021112345", function() .. end)
  it("Caller dials *31#021112345", function() .. end)
  it("Caller dials *31%23021112345", function() .. end)
  it("Caller dials +4921112345", function() .. end)
  it("Caller dials *31+4921112345, non-INVITE", function() .. end)
end)
```

Rerun your tests

```lua
require 'busted.runner'()

local function init_mock(options) [..] end

describe("Check and enable CLIR ->", function()
  it("Caller dials *31021112345", function() .. end)
  it("Caller dials *31*021112345", function() .. end)
  it("Caller dials *31#021112345", function() .. end)
  it("Caller dials *31%23021112345", function() .. end)
  it("Caller dials +4921112345", function() .. end)
  it("Caller dials *31+492111234
end)
```

2. damm@stretch: ~/kamailioworld (ssh)

```
damm@stretch:~/kamailioworld$ busted -l spec/headers_spec.lua
spec/headers_spec.lua:29: Check and enable CLIR -> Caller dials *31021112345
spec/headers_spec.lua:38: Check and enable CLIR -> Caller dials *31*021112345
spec/headers_spec.lua:47: Check and enable CLIR -> Caller dials *31#021112345
spec/headers_spec.lua:56: Check and enable CLIR -> Caller dials *31%23021112345
spec/headers_spec.lua:65: Check and enable CLIR -> Caller dials +4921112345
spec/headers_spec.lua:74: Check and enable CLIR -> Caller dials *31+4921112345, non-INVITE
damm@stretch:~/kamailioworld$ busted spec/headers_spec.lua
●●●●●●
6 successes / 0 failures / 0 errors / 0 pending : 0.006695 seconds
damm@stretch:~/kamailioworld$
```

Too dry?

Try it yourself:
https://github.com/sipgate/lua-kamailio

That's it

Questions?

Want to chat? Stop by!