

Mobile app messaging with kamailio

Kamailio is amazing when using SIP for voice and video calling and it is certainly very good at providing connectivity.

When doing messaging over SIP, we can get the same level of control over connectivity.

In several projects I have seen messaging being handled by a separate team of engineers and both teams needed to ensure good connectivity therefore solving the problem twice.

In this presentation I will share some recipes you may find useful when using kamailio for messaging.

Presented @ Kamailio World 2024

A little bit about me,

KAMAILIO WORLD CONFERENCE

Julien Chavanton,
20+ years in computers software engineering with a focus on CTI, VoIP and real time communication.
In the past few years, I was also able to mix RTC with eBPF/XDP to write packet manipulation/relay softwares.

I tend to be interested about low level, by that I mean being close to the hardware, protocols, etc.



Testing a voip softphone over starlink
La Vérendrye
natural reserve, Quebec.
Octobre 2023

RAVEN TECHNOLOGIES

Presented @ Kamailio World 2024

RAVEN
TECHNOLOGIES

I apologies in advance, I will be sharing a lot of on screen code in this presentation.

http://www.castmm.com/kw.pdf

I wanted to share some Kamailio recipes, maybe I am inspired by all the time I spent making pies.













mobile app messaging with Kamailio presentation overview

Part 1 - messaging with Kamailio

send, push and store (push server example, registrar module)

register and forward (uac module)

large messages (media file server)

Part 2 - example of using messaging for exchanging information with softphones

message freeswitch packets and jitter buffer reporting to the app

message webrtc getstats from the app

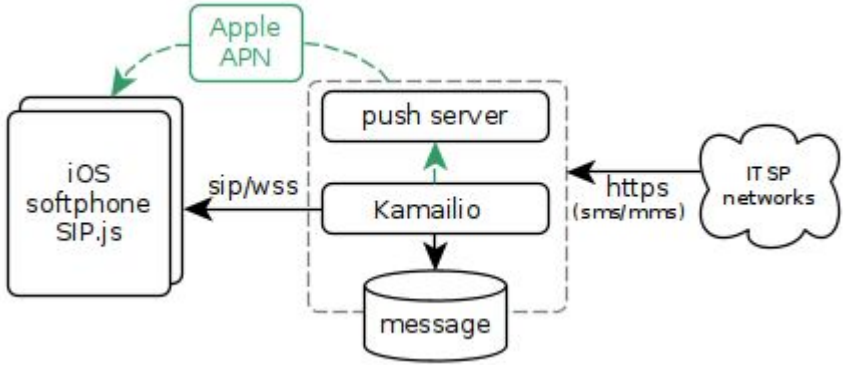
message freeswitch conference status to the app

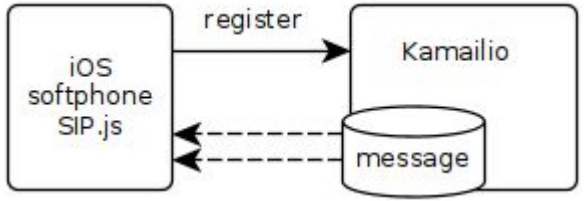
http://www.castmm.com/kw.pdf

message store and forward

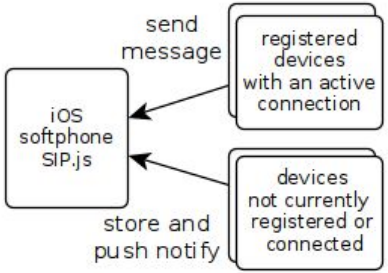
When receiving a message

1. select the devices of the destination user
2. check if we have an active WSS/TCP connections for them
3. try to send the message or send a push notification and store the message





https://github.com/jchavanton/push_server



Using kamailio **registrar** module, select the devices of the destination user, check if we have active WSS/TCP connections for them.

```
if (is_method("REGISTER")) {
    /* Once authenticated, store the deviceid using
the registrar module */
    $avp(device_id) = $hdr(X-DEVICE-TOKEN);
    remove_hf("User-Agent");
    append_hf("User-Agent: $avp(device_id)\r\n");
    msg_apply_changes();
    save("location");
}
```

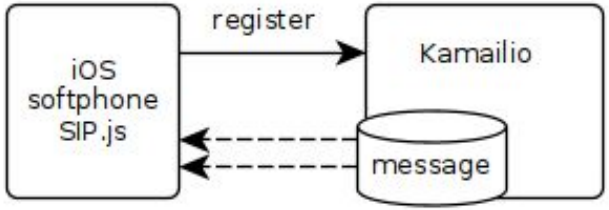
```
route[GET_REGISTRAR_USER_DEVICES] {
   // query the registrar module to retrieve the connections (WSS/TCP)
   $var(i) = 0;
   if (reg_fetch_contacts("location", "$avp(user)", "caller")) {  //
       while($var(i) < $(ulc(caller=>count))) {
           $xavp(devices=>addr)         = $(ulc(caller=>addr)[$var(i)]);
           $xavp(devices[0]=>id)        = $(ulc(caller=>user_agent)[$var(i)]);
           $xavp(devices[0]=>received)  = $(ulc(caller=>received)[$var(i)]);
           $var(conid)                  = $(ulc(caller=>conid)[$var(i)]);
           if ($var(conid) > 0 && !tcp_conid_alive("$var(conid)")) { // verify the connection state, tcpops module
               $xavp(devices[0]=>connected) = 0;  // we need to store and forward when the user registers
               xinfo("[$ci][$xavp(devices=>id)][$avp(user)] tcp connection inactive [$var(conid)] store & push\n");
           } else {
               $xavp(devices[0]=>connected) = 1;  // we can try t_relay() with a failure route
               xinfo("[$ci][$xavp(devices=>id)][$avp(user)] tcp connection active [$var(conid)] connected\n");
           }
           $var(i) = $var(i) + 1;
       }
   }
}
 // also push to unregistered devices
```

KAMAILIO

Message storage

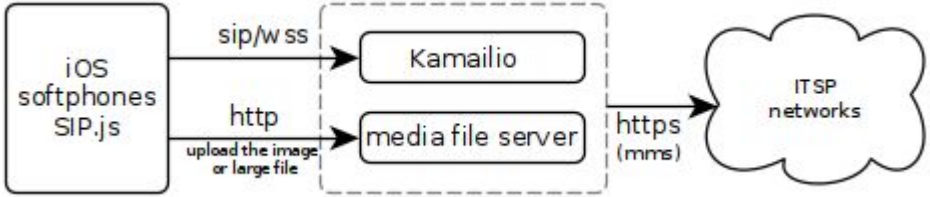
```
1.  modparam("sqlops", "sqlcon", "message=>sqlite:////kamailio_db/message.sqlite" );
2.
    route[MESSAGE_DB_STORE] { // when we fail to relay a message or the device is not connected
3.    $avp(m) = $rb;   // the message is in the SIP request body
4.    $avp(m) = $(avp(m){s.encode.base64});
5.    $var(q) = $_s(INSERT INTO message (user_src, user_dst, device_id, ts, rb)
6.                        VALUES ( '$fU', '$avp(dst_user_id)', '$avp(device_id)', $TV(s), '$avp(m)'));
7.    if (sql_xquery("message", "$var(q)", "rb")) {
8.        xinfo("ok\n");
9.    } else {
10.       xinfo("failure\n");
11.   }
12. }
13.
14. route[MESSAGE_DB_GET] { // when a device is registering
15.   $var(q) = $_s(SELECT id, user_src, user_dst, ts, rb FROM message WHERE
16.    (user_dst like '$fU' or user_src like '$fU') and sending = 0 and device_id like '$avp(device_id)');
17.   if (sql_xquery("message", "$var(q)", "res")) {
18.       while ($xavp(res) != $null) {
19.           $avp(message) = $xavp(res=>rb);
20.           $avp(message) = $(avp(message){s.decode.base64});
21.           $var(q) = $_s(update message set sending = 1 where id = $xavp(res=>id));
22.           sql_xquery("message", "$var(q)", "res");
23.           $xavp(res[0]=>user_dst) = $fU;
24.           route(UAC_SEND_MESSAGE) ;   // forward | send the message, see next slide
25.           pv_unset("$xavp(res)");
26.       }
27.   }
28. }
```

When a device is registering, we can forward all the stored messages with the uac module.




```
1.  loadmodule "uac"
2.
    route[UAC_SEND_MESSAGE] { // $xavp(res) was filled by the route MESSAGE_DB_GET
3.      $uac_req(method) = "MESSAGE";
4.      $uac_req(furi) = $_s(sip:$xavp(res=>user_src)@$shv(hostname));
5.      $uac_req(ruri) = "sip:" + $avp(user_socket);
6.      // if in the context of a received SIP REGISTER
7.      // $su - Source address as URI  Example sip:143.127.32.16:49902;transport=ws
        // $avp(user_socket) = $_s($fU@$(su{s.substr,4,0}));
8.
9.      $uac_req(turi) = $_s(sip:$xavp(res=>user_dst)@$shv(hostname));
10.     $uac_req(body) = $avp(message);
11.     $uac_req(hdrs) = "Content-Type: application/json \r\n";
12.     $uac_req(hdrs) = $uac_req(hdrs) + $_s(X-Received: $xavp(res=>ts)) +"\r\n";
13.     $uac_req(callid) = $_s($(mb{s.md5})_$xavp(res=>id));  // message id is added to the call-id
14.     $uac_req(evroute) = 1;
15.     uac_req_send();
16. }
17.
18. event_route[uac:reply] {
19.     $var(id) = $(uac_req(callid){s.select,1,_});
20.     xinfo("method[$uac_req(method)] code[$uac_req(evcode)] call-id[$uac_req(callid)] id[$var(id)]  \n");
21.     if ($uac_req(method) == 'MESSAGE' && && $uac_req(evcode) == 200) {
22.         route(DELETE_MESSAGE);
23.     }
24. }
```

Large messages and media file server.



In kamailio SIP messages processing is quite optimized and messages are exposed in memory.

Messages are limited in size, even if it is possible to adjust the buffers example *tcp_rd_buf_size*, this is not viable of large files.

This may be a limitation but it is also a blessing since we are forced to protect the signaling channel from large transfer.

In this example I am proposing an approach similar to MMS where we upload the attachment and send a link to it.

A JSON envelope is used, in this case the media section contains a file.

```
MESSAGE sip:20rgkegj@10.125.25.146;transport=udp SIP/2.0
Via: SIP/2.0/WSS e1snsdr6j3d8.invalid;branch=z9hG4bK4784540
To: <sip:bob@kamailio.nowhere:443>
From: <sip:alice@kamailio.nowhere:443>;tag=9r1t254ac6
CSeq: 1 MESSAGE
Call-ID: uhu1p8faoi263jsp3jbq
Content-Type: application/json
Content-Length: 366

{
   "text": "",
   "media": [
     {
       "fileSize": 7814298,
       "fileName": "4608BD07-F484-4FB4-BA89-18448D9BAB88.jpg"
     }
   ],
   "id": "3ffa1aa2-37bf-40ab-98c3-a224f7e24ad6"
 }
```



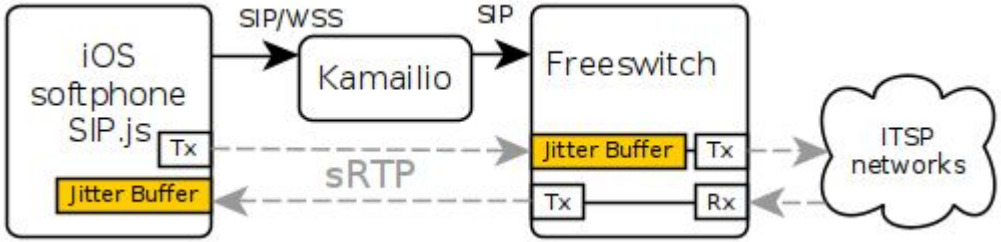
Part 2 - example of using messaging for exchanging information between the backend and the softphones

message freeswitch packets and jitter buffer reporting to the app

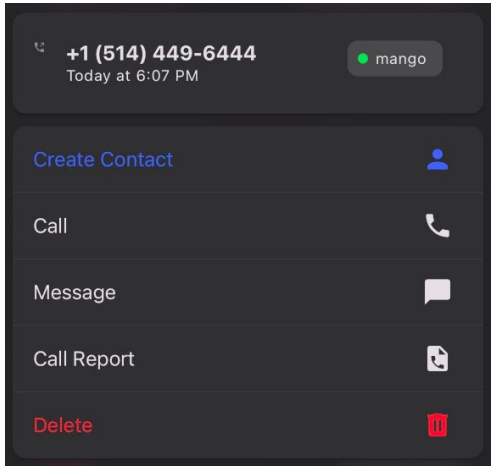
message webrtc getstats from the app

message freeswitch conference status to the app

Was there a one way delay ?
Freeswitch call reports Messages



I do not think we can assume that ITSPs are expecting the kind of crazy jitter we can find on mobile devices roaming around.
For VoIP in and out, when using a JB we may want to measure and report how much time are packet spending in, freeswitch.



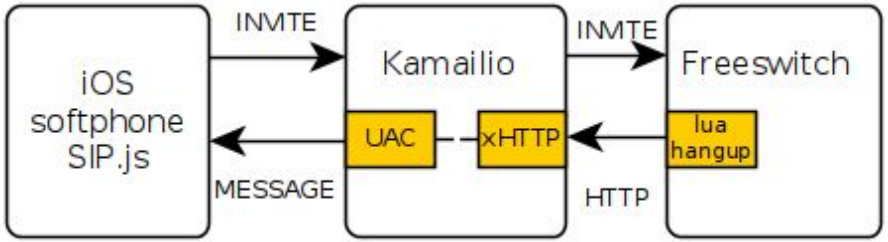

```
"call report": {
    "jb": {
      "size_max_ms": 180,
      "size_est ms": 20,
      "acceleration ms": 580,
      "expand ms": 480,
      "fast acceleration ms": 0,
      "forced_acceleration_ms": 0,
      "jitter max ms": 31,
      "jitter_est_ms": 2
    }
  }
```

```
  "call_report": {
    "rp": {
      "duration": 165,
      "call id in": "99d203f3-7229-123d-c19e-0a9c4706906f",
      "call id out": "mnjbpclkide4uvtnjbhb",
      "report_a": {
        "in": {
          "ssrc": "0xD88ED8F6",
          "remote socket": "142.117.233.136:53741",
          "local socket": "10.125.25.146:16112",
          "codec": "opus",
          "count": 8473,
          "plc": 25
        },
        "out": {
          "ssrc": "0x9A184A89",
          "remote socket": "52.45.164.125:15766",
          "local socket": "10.125.25.146:17490",
          "codec": "PCMU",
          "count": 8473,
          "max_ms": 180,
          "avg_delay_ms": 30.52
        }
      },
    }
}
```

Packet transmission reporting (not merged into freeswitch)
https://github.com/jchavanton/freeswitch/commit/907a95

More elastic jitter buffer in Freeswitch
https://github.com/signalwire/freeswitch/pull/2337

Freeswitch call reports to Kamailio message




```
<!-- freeswitch XML dialplan -->
<action application="set" data="user_socket=${sip_h_X-user_socket}"/>
<action application="set" data="session_in_hangup_hook=true"/>
<action application="set" data="api_hangup_hook=lua /hangup_metrics.lua"/>
```

```
route[INVITE] {
        append_hf( "X-user-socket: $avp(user_socket) \r\n");
        $du = $_s(sip:$shv(freeswitch_ip_port) );
        t_relay();
}

event_route[xhttp:request] {
    if ($hu =~ "^/freeswitch_packet_metrics" ) {
        jansson_get_field( $avp(r), "destination",
"$avp(user_socket)" );
        $avp(message) = $rb;
        route(UAC_SEND_MESSAGE) ; // send the message
        xhttp_reply("200", "OK", "text/plain", "\r\n");
        exit;
    }
    xhttp_reply("404", "Not Found", "", "");
}
```

KAMAILIO

```
-- hangup_metrics.lua
local socket = require("socket")
local https = require("socket.http")
local ip = os.getenv("LOCAL_IPV4")
local report = session:getVariable("packet_stats_report");
local call_id = session:getVariable("sip_call_id");
local user_socket = session:getVariable("user_socket");

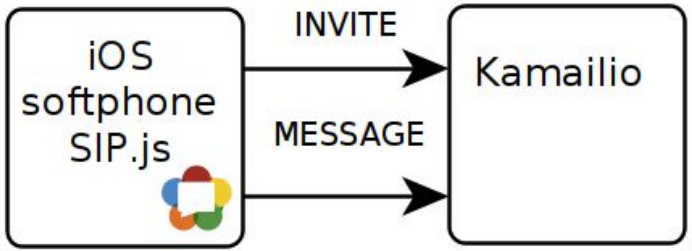
function http_post_duration(v)
   session:consoleLog("info", "[http-kamailio]:"..v.."\n");
   local r, c, h, s = https.request{
       method = 'POST',
       url = "http://"..ip..":80/freeswitch_packet_metrics",
       headers = {
           ["Content-Type"] = "application/json",
           ["Content-Length"] = string.len(v)
       },
       source = ltn12.source.string(v),
       sink = ltn12.sink.table(response_body)
   }
end

local r = '{ "call_id": "'..call_id..'", "user_socket":
"'..user_socket..'", "report": "'..report..'"}'

http_post(report)
```

Lua

SIP.js WebRTC call reports over SIP messages




```
import { UserAgent, Messager } from "sip.js";

async function sendMessage(body) {
  const target = UserAgent.makeURI("sip:stats@kamailio.void");
  const messager = new Messager(userAgent, target, body);
  messager.message()
}

async function jsonReportFromStats(stats) {
  let includedReports = ["inbound-rtp"];
  let reports = [];
  stats.forEach((report) => {
     if (includedReports.includes(report.type))
         reports.push(report);
  });
  statsJson = JSON.stringify(reports);
  return statsJson;
}

const sdh = activeSession.sessionDescriptionHandler;
const pc = sdh.peerConnection;
const stats = await pc.getStats();
const body = await jsonReportFromStats(stats);
sendMessage(body);
```

```
MESSAGE sip:stats@kamailio.void SIP/2.0
Via: SIP/2.0/WSS gfh8ksi2ic1o.invalid;branch=z9hG4bK6482525
To: <sip:stats@kamailio.void>
From: <sip:stats@kamailio.void>;tag=bjeq9qj5rb
CSeq: 1 MESSAGE
Call-ID: rk4e00a9ad76t072aiim
User-Agent: SIP.js/0.21.2
Content-Type: text/plain
Content-Length: 977

[{        "timestamp": 1713158590304,
          "type": "inbound-rtp",
          "ssrc": 3257621132,
          "packetsLost": 0,
          "packetsReceived": 481,
          "audioLevel": 0.00012207403790398877,
          "bytesReceived": 34105,
          "concealedSamples": 23656,
          "fecPacketsDiscarded": 107,
          "fecPacketsReceived": 107,
          "headerBytesReceived": 5772,
          "insertedSamplesForDeceleration": 3006,
          "jitterBufferDelay": 16204.8,
          "removedSamplesForAcceleration": 2558,
          "silentConcealedSamples": 18600,
          "totalAudioEnergy": 0.00939457813620388,
          "totalSamplesDuration": 10.08999999999983,
          "totalSamplesReceived": 484320
```

Was this call a one way audio ?
WebRTC call reports over SIP messages.

```
// stat message received route
route[WEBRTC_STATS] {
   jansson_get_field($rb, "callId", "$avp(call-id)");
   jansson_get_field($rb, "inbound-rtp", "$avp(inbound-rtp)");
   if ($avp(inbound-rtp) != $null) {
       jansson_get_field($avp(inbound-rtp), "totalSamplesDuration", "$avp(inSamplesS)");
       jansson_get_field($avp(inbound-rtp), "totalAudioEnergy", "$avp(inAudioEnergy)");
       lua_run("average_audio_level", "$avp(inAudioEnergy)", "$avp(inSamplesS)");
       $avp(inAvgAudioEnergy) = $avp(lua_ret);
       lua_run("round", "$avp(inSamplesS)", 2);
       $avp(inSamplesDuration) = $avp(lua_ret);
       lua_run("round", "$avp(inAudioEnergy)", 3);
       $avp(inAudioEnergy) = $avp(lua_ret);
   }
   xinfo("playback:[$avp(inAvgAudioEnergy)/1]sampled-duration[$avp(inSamplesS)]\n");
}
```

```
-- utils.lua
function round(v,d)
if (v == nil or d == nil) then
   KSR.pv.sets("$avp(lua_ret)","")
   return
end
f = "%." .. d ..  "f"
KSR.pv.sets("$avp(lua_ret)",string.format(f, tonumber(v)))
end

function average_audio_level(energy,time)
if (energy == nil or time == nil) then
   KSR.pv.sets("$avp(lua_ret)","")
   return
end
e = tonumber(energy)
t = tonumber(time)
if (e == nil or t == nil) then
   KSR.pv.sets("$avp(lua_ret)","")
   return
end
if (e == 0 or t == 0) then
   KSR.pv.sets("$avp(lua_ret)","0")
   return
end
KSR.pv.sets("$avp(lua_ret)", round(math.sqrt(e/t),2))
end
```

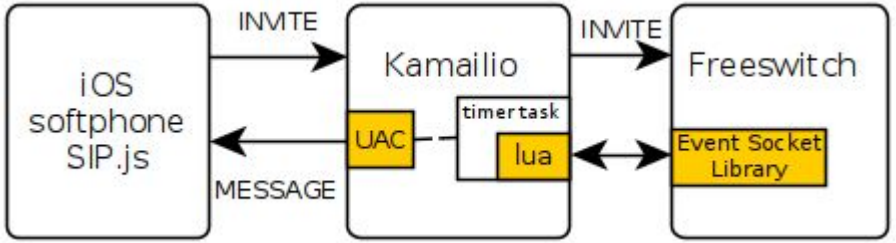
```
playback:[0.05/1]sampled-duration[240.17s]
// not speaking very loudly and/or often
playback:[0.77/1]sampled-duration[240.17s]
// shouting in the microphone almost 100% of the time (very unlikely)
```

Audio Energy :

1.0 - The maximum possible digital level
0.5 - 50%
0 - Silence

https://en.wikipedia.org/wiki/Sound_pressure#Sound_pressure_level

Retrieve conference status from freeswitch




```
modparam("app_lua", "load", "/etc/kamailio/conf_status.lua" )
modparam("rtimer", "timer", "name=trun;interval=5;mode=1;" )
modparam("rtimer", "exec", "timer=trun;route=GET_CONF_STATUS" )
loadmodule "jansson"

// sending one status message to each conference participant
route[GET_CONF_STATUS] {
  lua_run("get_conf_status", "$xavp(rb=>user)");
  if ($avp(conf_status) == $null) return;
  $var(count) = 0; $var(count_members) = 0; $var(members) = 0;

  jansson_array_size("", $avp(conf_status), "$var(size)");
  while ($var(count) < $var(size)) {
    jansson_get("[$var(count)]", $avp(conf_status), "$var(v)");
    jansson_array_size("members", $var(v), "$var(members)");
    while ($var(count_members) < $var(members)) {
      jansson_get("members[$var(count_members)]",$var(v),"$var(member)");
      jansson_get("caller_id_number",$var(member),"$var(userid)");
      route(SEND_CONF_STATUS); // wrapper around UAC_SEND_MESSAGE
      $var(count_members) = $var(count_members) + 1;
    }
    $var(count) = $var(count) + 1;
  }
}
```

KAMAILIO

```
-- /etc/kamailio/conf_status.lua

function get conf status ()
  local uv  = (require "lluv")
  local esl = (require "lluv.esl")
  local cnn = esl.Connection()

  cnn:open(function(self, err)
     if err then
        return
     end
     self:bgapi("conference json_list", function(self, err,
reply)
        KSR.pv.sets("$avp(conf_status)", (err or
reply:getBody()));
        self:close()
     end)
   end)
   uv.run()
end
```

Lua

```
# lua esl installation on Debian bookworm
apt-get install lua5.1 lua-socket luarocks libuv1-dev libexpat1-dev uuid-dev
luarocks install --server=https://luarocks.org/dev lluv-esl
```

Freeswitch conference status message

```
{
   "conf_status" : {
     "conference_name" : "meeting123",
     "member_count" : 1,
     "rate": 8000,
     "conference_uuid" : "b815f40e-ccb4-4e65-b6c8-9611a3d08f67" ,
     "max_bw_in" : 0,
     "force_bw_in" : 0,
     ...
     "running" : true,
     "answered" : true,
     "enforce_min" : true,
     "bridge_to" : false,
     "dynamic" : true,
     "recording" : false,
     "video_bridge" : false,
     "video_floor_only" : false,
     "video_rfc4579" : false,
     "variables" : {},
     "members" : [ { ... } ]  // members
   },
   "to": [
     "bob"
   ]
}
```

```
{     // example of conference member
   "type": "caller",
   "id": 2,
   "flags": {
     "can_hear": true,
     "can_see": true,
     "can_speak": true,
     "hold": false,
     "mute_detect": false,
     "talking": false,
     "has_video": false,
     "video_bridge": false,
     "has_floor": true,
     "is_moderator": false,
     "end_conference": false,
     "mandatory_member_end_conference": false,
     "pass_digits": false
   },
   "uuid": "9db9b1c8-167a-4628-ac72-e57f9c439179",
   "caller_id_name": "bob",
   "caller_id_number": "bob",
   "join_time": 7,
   "last_talking": 5,
   "energy": 100,
   "volume_in": 0,
   "volume_out": 0,
   "output-volume": 0,
   "input-volume": 0
}
```

Entartage (Pieing), home made pies only please !



Q&A

Looking forward future collaboration !

Julien Chavanton
jchavanton@gmail.com



Val-David, Quebec.
April 2023

Presented @ Kamailio World 2024