OpenCNAM Kamailio Integration Guide

A. Introduction

OpenCNAM provides several data channels through which customers can query its Caller ID Name (CNAM) lookup products. One of these is the SIP interface, which uses SIP redirect messages to convey caller identity.

The purpose of this guide is to provide users of the **Kamailio** SIP proxy/ server with specific instructions on how to consume the OpenCNAM SIP interface using its programmatic configuration script.

This document presumes prior familiarity with the OpenCNAM SIP interface. If you are not familiar with how the OpenCNAM SIP interface works or with SIP redirect messages, consult the "OpenCNAM Integration with SIP Interface" guide for the appropriate background information. In the interest of brevity, this document will concern itself with topics specific to Kamailio implementation only.

This document also presumes familiarity with the essentials of Kamailio's configuration scripting language, alsoknown as route script. Kamailio route script is a complex topic and is outside the scope of this document. If you are looking to learn about Kamailio more generally, the **Kamailio project documentation** site is a useful starting point. You may also consider joining the **Kamailio mailing lists** to receive realtime help from its opensource community. Finally, the assistance of professional Kamailio consultants is available on a commercial basis; you can inquire with OpenCNAM Support for recommendations of particular companies.

B. Prerequisites

1. This document presumes that you have a working Kamailio installation and a clear understanding of how SIP `INVITE` requests are routed in your Kamailio configuration. You will need to modify this sequence in order to make use of the OpenCNAM SIP interface.

2. This document presumes that you have followed the "Prerequisites" steps in the "OpenCNAM Integration with SIP Interface" guide in provisioning the IP address(es) of your Kamailio server in the "white list" under the Advanced Authentication Options section of the **customer dashboard**. If you have not done this, please stop here and perform this step before continuing.

C. Implementation

In most effective uses of Kamailio, transactional relay of `INVITE` requests is used via its `tm` module. The stock configuration that ships with Kamailio encapsulates inbound call routing to VoIP endpoints from the PSTN, which is the most common presumption of the OpenCNAM usecase, in the `route[LOCATION]` request route. The actual location of appropriate `INVITE` handling may vary, and you will need to locate it.

You will need to modify this sequence to effect the following result:

1. Relay incoming `INVITE` to OpenCNAM gateway (`sip.opencnam.com`).

2. Receive `300 Multiple Choices` response with `P-Asserted-Identity` header containing CNAM response, as described more fully in the "OpenCNAM Integration with SIP Interface" guide.

3. Extract the CNAM portion of the `P-Asserted-Identity` header (the "display name").

4. Convey the CNAM value to the call recipient in one of the following ways:
   a. Build new `P-Asserted-Identity` header;
   b. Inject the value into the display name portion of the `From` header.

To begin, you must associate this `failure_route` with the current `INVITE` transaction. In the parlance of Kamailio documentation, this is referred to as "arming" the `failure_route`.

```
t_on_failure("CNAM_CATCH");
```

Important: The `t_on_failure()` call must **precede** the invocation of `t_relay()`.

Then, modify your Kamailio route script to relay the incoming `INVITE` request to OpenCNAM by setting the relay destination ("destination URI") of the request to the OpenCNAM server. This must also be done prior to `t_relay()`:

```
$du = 'sip:sip.opencnam.com:5060';
```

Finally, you must create the actual `failure_route`:

```
failure_route[CNAM_CATCH] {
       if(t_is_cancelled() || t_branch_timeout())
           exit;

       # Conceal unexpected reply failures with an opaque 500 Internal
       Server
       # Error response. This course of action should be tailored to
       business
       # requirements.

       if($T_rpl($rs) != 300) {
           xlog("L_INFO", "Unexpected reply $T_tpl($rs) $T_rpl($rr)
           received\n");
           send_reply("500", "Internal Server Error");
           exit;
       }
       # Access PAI header of winning reply.
       $avp(cnam) = $(T_rpl($hdr(P-Asserted-Identity){nameaddr.name}
       {s.replace,",});
       # Re-route call to final destination with implicit append_branch(),
       e.g.
       route(LOCATION); # Only if you did not modify route[LOCATION] to
       route to         # OpenCNAM!
       t_relay();
}
```

The CNAM value should now be stored in `$avp(cnam)` , which is a transaction-persistent variable (AVP, or attributevalue pair) that is accessible in all classes of routes for as long as the current `INVITE` transaction remains active.

The choice of what to do with this value will depend on how you wish to convey it to the recipient, as described in the next two subsections.

Presentation option A Append `P-Asserted-Identity` header

One option, if the receiving SIP endpoint supports it, is to simply append one’s own `P-Asserted-Identity` header:

```
append_hf(“P-Asserted-Identity: \”$avp(cnam)\” <sip:$rU@$Ri:$Rp>\r\n”);
```

Commercial SIP equipment will generally prefer `P-Asserted-Identity` over the `From` display name field for purposes of presentation.

Presentation option B Inject value into `From` display name

Another, more complex option involves injecting the CNAM value into the `From` header. This is more challenging because SIP proxies are prohibited by the standards from modifying the `From` header value in any way in flight.

However, Kamailio has a module called `uac` which offers a rather novel solution to this problem. Simply put, it involves modifying the `From` header statefully in a way that is invisible to the calling party because the From header will be rewritten to the original value on all messages going back to that calling party.

Important: This approach requires that you make use of `Record-Route` (i.e. via `record_route()`) in order to keep the Kamailio proxy in the path of all subsequent requests inside the dialog. Otherwise, the modified `From` header will be leaked through in indialog messages such as reinvites, BYE, etc.

To use the `uac` approach, first load and configure the `uac` module at the top of your configuration:

```
loadmodule "uac"

modparam("uac", "rr_from_store_param", "fromcor")
modparam("uac", "restore_mode", "auto")
modparam("uac", "restore_passwd", "z1pmFqRjTPSA") # Set your own encryption key!
```

Next, you may use the `uac_replace_from()` function exported by this module to replace the `From` display name in the following manner:

```
uac_replace_from("\"$avp(cnam)\"", "$fu");
```

Technical note: This will cause the original `From` value to be encrypted and stored in a `Record-Route` parameter as a "rider". The parameter will be called `fromcor` as per the above example, but this parameter can have an arbitrary name.

D. Success

If the strategies described have worked, you should see correct CNAM values presented on the call recipient's endpoint.

F. Support / Assistance

We are happy to assist with your integration. Our team can be reached in any of the following ways:

Phone: +1-888-315-8356 (TELO) or +1-678-631-8356 (TELO)
Email: support@opencnam.com