Kamailio: Tips, Tricks & Notes from the Field

Evariste Systems

Alex Balashov – Evariste Systems LLC
Atlanta, GA, USA

http://www.evaristesys.com/

Evariste Systems?

- Software vendor:
  - CSRP - Class 4 routing (LCR), rating and accounting product for delivering SIP trunking to retail and wholesale customers.
- Kamailio consultancy
  - Professional services related to CSRP.
  - Strong focus on PSTN interconnection in North America.
  - Custom Kamailio projects.
  - Miscellaneous VoIP engineering and consultancy.

Tips, tricks and notes from the field!

- Started working with OpenSER in late 2006.
- Life since then is Kamailio/OpenSER all day, every day.
- Some things I've learned.
  - A lot of it is buried in Kamailio documentation, but it doesn't mean one paid attention to it.
  - What is self-evident to one person is not to another.
    - The hope is that something here was non-obvious to you.

#1: usrloc db_mode when sharing DB

If you have two active proxies storing registration bindings and sharing the same database (using database backing for the usrloc module), use **db_mode 3**.

Alternately, if you have two proxies using Different database servers with master-master replication running between them, also use **db_mode 3**.

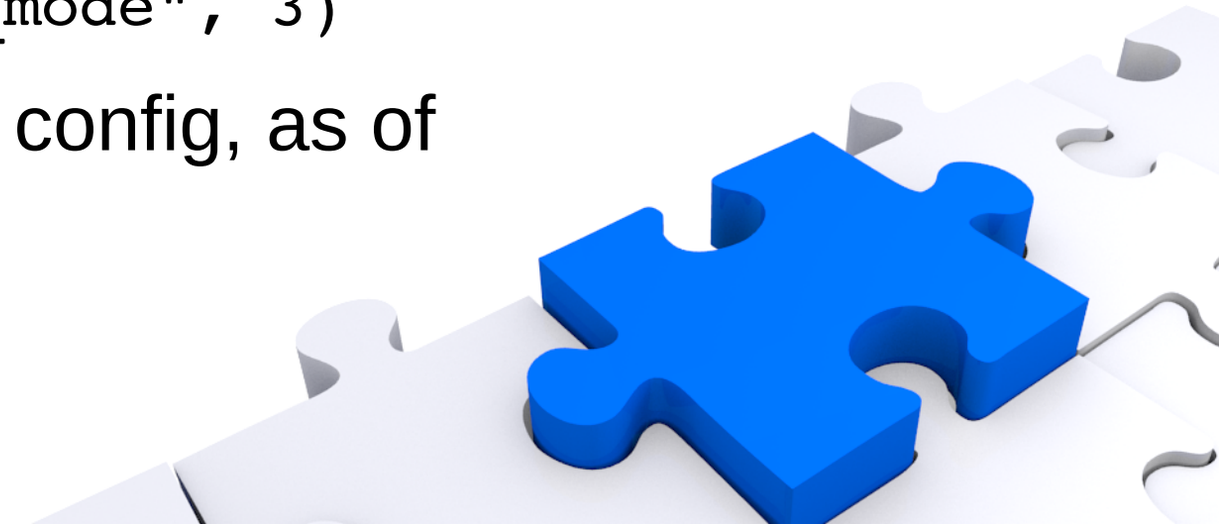
The other modes cause the respective proxies to kick each other's contacts out of the *location* table.

#2: TM 'failure_reply_mode' parameter

- In master:*7f0cdd2* (3.0.2 release), the *failure_reply_mode* modparam was added to control retention of branch data in serial forking.
- If you want behaviour compatible with Kamailio <= 1.5.x (all previous branches discarded), set:

```
modparam("tm", "failure_reply_mode", 3)
```

- This is set by default in the stock config, as of master:*cba4663.*



#3: Arrays are a thing!

- AVPs can be made into arrays.

```
$(avp(x)[0]) = 'a';
$(avp(x)[1]) = 'b';
$(avp(x)[2]) = 'c';
```

- Because they are implemented as lists internally.
- Not just AVPs – some other types of pseudovariables as well:
  - $hdr(...) - headers
  - But not user variables - $var(...).

#4: (X)AVP arrays are LIFO stacks

However, (X)AVPs are LIFO (Last In First Out) stacks:

```
$(avp(x)[0]) = 'a';
$(avp(x)[1]) = 'b';
$(avp(x)[2]) = 'c';

$var(i) = 0;

while(is_avp_set("$(avp(x)[$var(i)])")) {
   xlog("L_INFO", "Array value [$var(i)]: $(avp(x)[$var(i)])\n");
   $var(i) = $var(i) + 1;
}
```

… prints …

```
INFO: Array value [0]: c
INFO: Array value [1]: b
INFO: Array value [2]: a
```

#4: AVPs are LIFO stacks (cont'd)

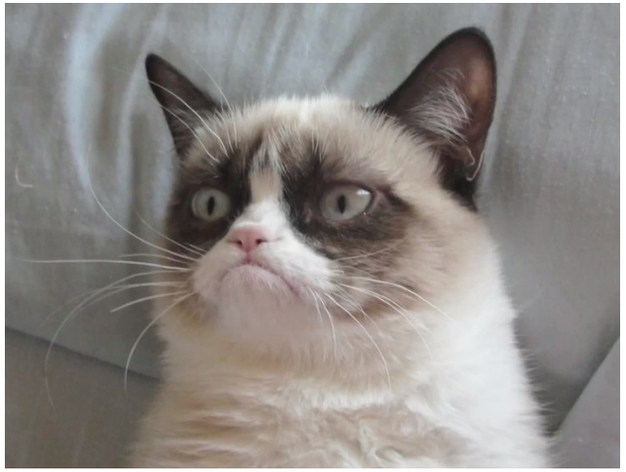
Therefore, this is the right way to iterate over an array in FIFO order:

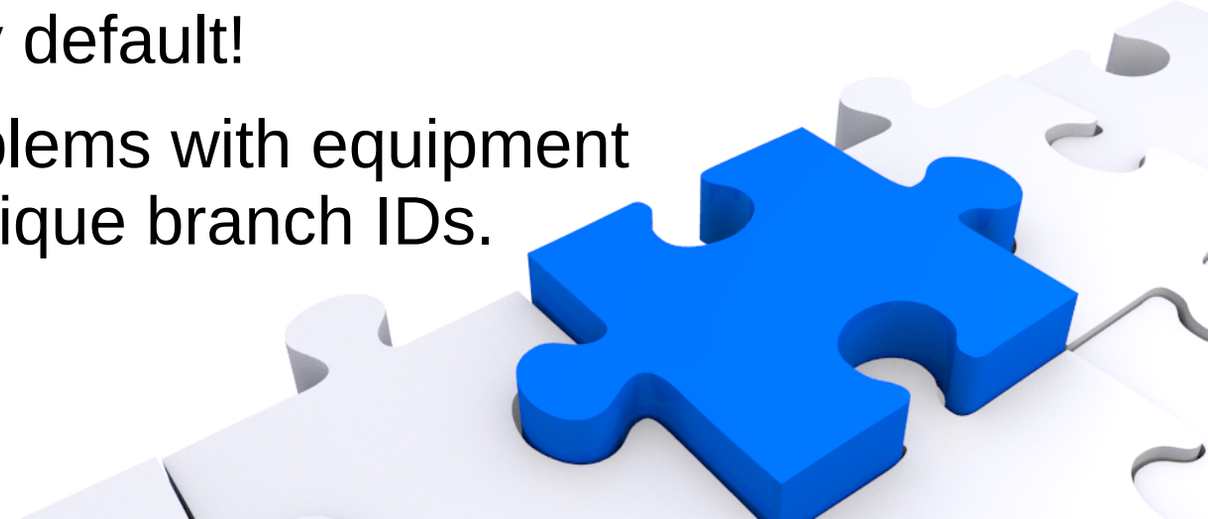
```
$var(i) = 2;

while(is_avp_set("$(avp(x)[$var(i)])")) {
    xlog("L_INFO", "Array value [$var(i)]: $(avp(x)[$var(i)])\n");
    $var(i) = $var(i) - 1;
}
```

#5: syn_branch

- Prior to 4.0.x, the *syn_branch* core parameter existed.
  - Caused the same, simple **branch** value to be reused on stateless forwarding, and also to stateful forwarding of end-to-end ACKs for 2xx requests.
  - It was on (set to 1) by default!
  - It caused a lot of problems with equipment that didn't like non-unique branch IDs.





#5: syn_branch (cont'd)

- In 4.1.x, Henning Westerholt removed it! (master:*6287cae*)
  - From commit: „the performance reason that motivated this functionality are today not valid anymore...“
- Vielen Dank, Henning!

#6: local request event_route (TM)

- Endogenously-generated requests
  - Such as synthetic BYEs generated by the dialog module on dialog timeout
- … don't come through normal request routes.
- So, if you need to intercept these for accounting:

```
event_route[tm:local-request] {
    xlog("L_INFO", "[R-TM-LOCAL-REQUEST:$ci] Local request $rm to $ru\n");
}
```

#7: Non-local bind for failover

- Don't harness Kamailio inside OCF resource agent scripts for Heartbeat.
  - Too complicated.
- Enable non-local bind kernel option:
  - **`echo 1 > /proc/sys/net/ipv4/ip_nonlocal_bind`**
- Kamailio can then bind to the shared IP interface, even if it is not homed on the secondary host.
- Will start receiving traffic when the interface swings over.

#8: Don't use old core functions

- Don't use `rewritehostport()`, `force_send_socket()` & friends.
- They don't accept pseudovariable arguments.
- Modify the request URI instead:
  - `$ru`
- Or the destination set, if you need to:
  - `$du`
- For send socket: `$fs`

#9: Calling request_routes from failure_routes

- You can do it.
- You just need to be careful about not doing things from your `request_route` that you can't do in a `failure_route` context.
- Particularly:
  - Be sure you send all your replies statefully.
  - Or using `send_reply()`, which automatically sends replies statelessly if there is no transaction, or statefully if there is.

#10: append_branch()

- You don't really need it anymore in serial forking cases (circa master:*22f1b99*)
  - In normal failover routing cases.
- Just modify RURI, `t_relay()`, and a new branch will automatically be created.
- Still needed for parallel forking.

#11: $dlg_var(...)

- Store dialog-persistent variables there:
  - `$dlg_var(key) = 'val';`
- If you're tracking all dialogs anyway, very convenient.
- No need to deal with clunky Record-Route parameter manipulation anymore.
- **Caveat**: Only available when processing sequential requests (after `loose_route()`).

#12: Accessing replies from failure_route

- `failure_routes` correspond to a branch failure, not a negative reply *per se*.
- But pseudovariable attributes of the winning reply (from the same transaction) can be accessed:
  - `$T_rpl(...)` container holds these.
  - e.g. `$T_rpl($rs)` = reply code of winning reply

#13: Evaluating PVs in string assignment context

- Substitution of PVs in string literals will be done in function arguments (ones which accept Pvs):
  - `xlog("L_INFO", "Value is: $var(x)\n");`
- But not in bare string literals on assignment!
  - `$var(x) = "From URI: $fu";`
  - This will be evaluated literally.

#13: Evaluating PVs in string assignment context (cont'd)

- So, concatenation is required:
  - `$ru = “sip“: $tU + “@“ + $var(domain);`
- New pseudovariable container for this can help:
  - `$var(x) = $_s(sip:$tU@$var(domain));`

#15: Base64 transformations

- `{s.encode.base64}` / `{s.decode.base64}`
  - Contributed by Sipwise (master:*12f441f*), by Richard Fuchs.
- Good for avoiding delimiter problems in concatenated storage of SIP data.
- Thank you, Sipwise!

#14: Asynchronous side tasks

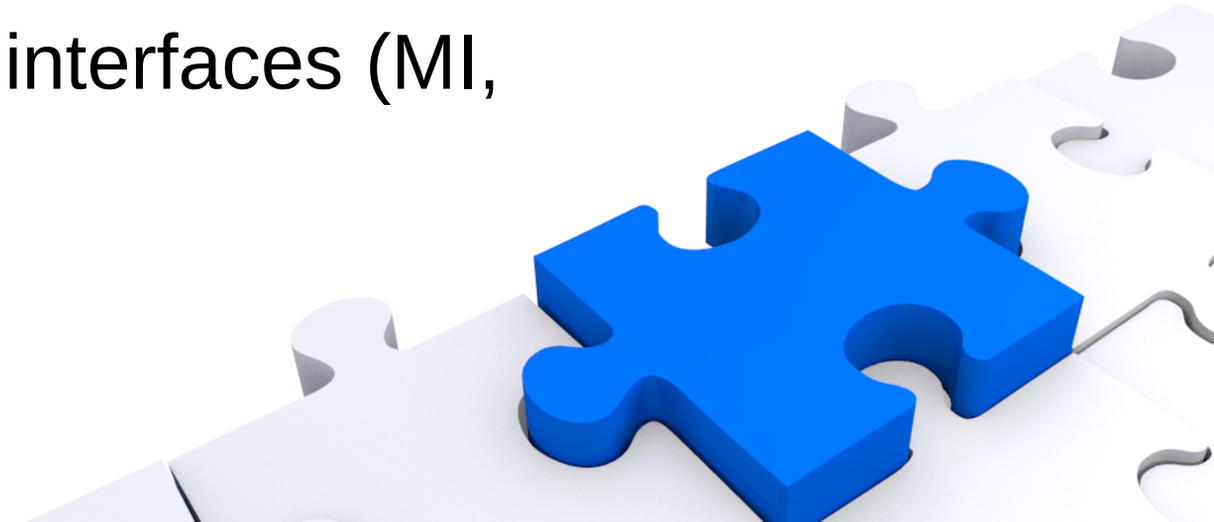
- Have I/O-bound activity on which message/call processing does not depend?
- Put it on an `mqueue`.
- Consume it with `rtimer`.
  - The `rtimer` module has a separate thread pool.

#15: uac_replace_from()/to()

- You can't call `uac_replace_from()/to()`.
  - Not if you want to take advantage of magical concealment of From/To manipulations from the caller.
- Just manipulate a single PV with your desired caller ID/caller display name values.
  - Call `uac_replace_from()` at the end of your call processing.

#16: Export your own statistics

- Use **statistics** module.
- Define your own modparams of statistics for it:
  - `modparam("statistics", "variable", "total_calls")`
- Update them from route script:
  - `update_stat("total_calls", "+1");`
- Easy to access via management interfaces (MI, BINRPC).



That's all!