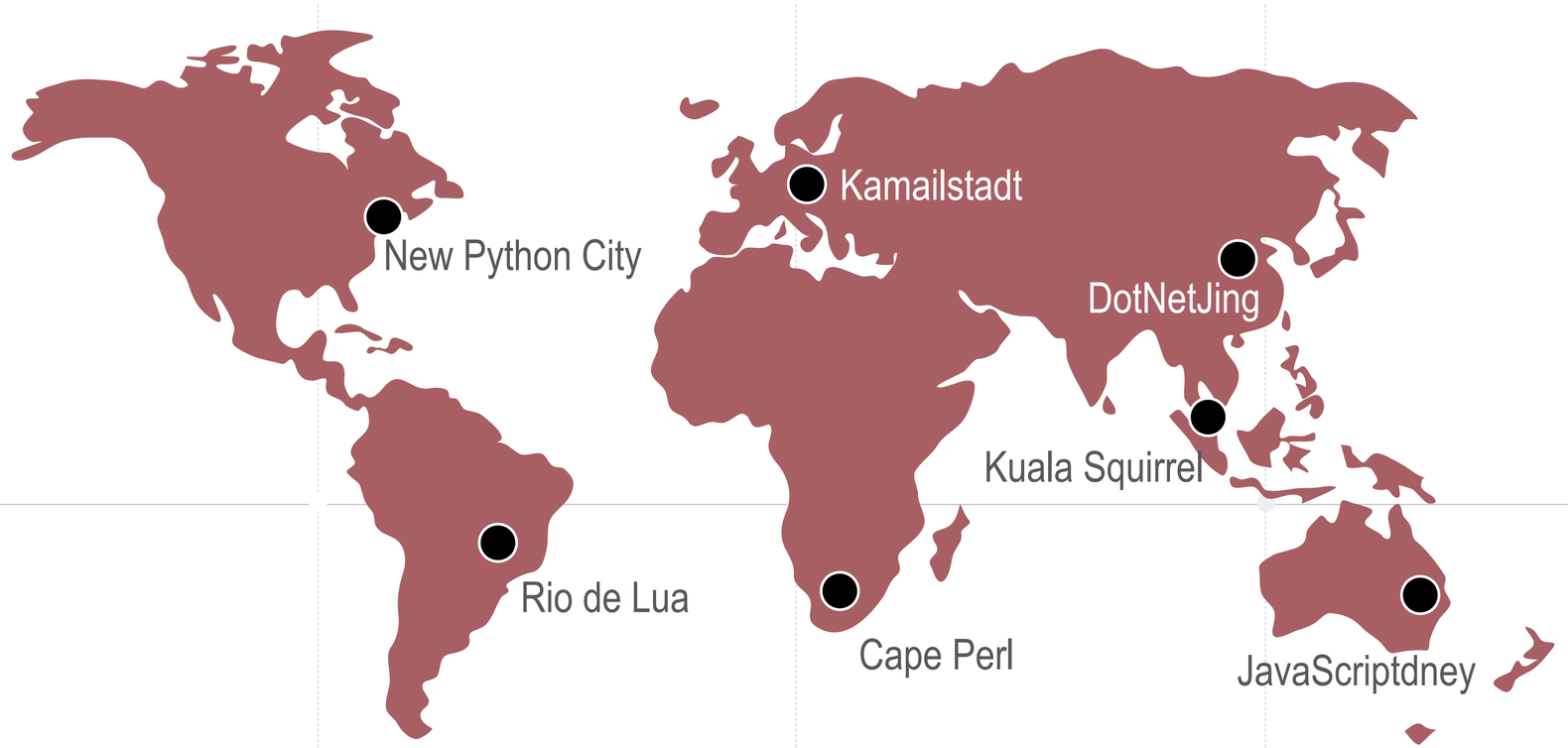


The World Of KEMI Scripting

Migrating SIP Routing Logic To KEMI Scripting

KAMAILIO WORLD
CONFERENCE & EXHIBITION
BERLIN GERMANY MAY 14-16, 2018

Daniel-Constantin Mierla
Co-Founder Kamailio Project
@miconda
asipto.com

evosip
the world outloud

netaxis
solutions

2600Hz

sipgate

sip:wise

digium

SIMWOOD
AUTHENTIC WHOLESALE TELECOMMUNICATIONS

KAMAILIO WORLD
CONFERENCE & EXHIBITION
BERLIN GERMANY MAY 14-16, 2018

EVARISTE
SYSTEMS

pascom®

ngvoice

LOD

KAZOOcon

vuc

cnd core network dynamics

Fraunhofer
FOKUS

asipto

Fraunhofer
Forum Berlin

Kamailio Configuration File

- Two main roles
  - Kamailio application initialization
    - Done once at startup (passive scope)
    - Global parameters, loading modules and modules'parameters
    - Many values can be changed at runtime via RPC (no restart)
  - Rules for handling SIP traffic
    - Done during runtime to decide the routing of SIP messages
    - No reload without restart for native kamailio.cfg scripting language
    - KEMI routing scripts can be reloaded without restart (v5.0+)
- Scripting languages
  - Native scripting language
    - built from scratch, routing blocks with set of actions
  - Kamailio Embedded Interface (KEMI) languages
    - reuse existing (popular) scripting languages
    - replace the routing blocks from native scripting language
    - allow reloading of scripts without restart and more features
  - Inline execution of scripting languages
    - can be executed inside native scripting language
    - support for Lua, JavaScript, Python, Perl, .Net (C#, …), Squirrel, Java


```
298  # Routing to foreign domains
299  route[SIPOUT] {
300          if (uri==myself) return;
301
302          append_hf("P-hint: outbound\r\n");
303          route(RELAY);
304          exit;
305  }
```

Scripting Based SIP Routing

Native (Custom) Routing Language Since 2001

Kamailio Embedded Interpreter Interface - K E M I

- introduced in Kamailio v5.0.0
  - split parts: **passive** (global and module params) and **active** (routing blocks)
  - live reload of active part
- working for next scripting languages (embedded interpreters)
  - Lua
  - JavaScript
  - Python - Python3
  - Squirrel
- not yet updated for next existing embedded interpreters
  - can be used with inline execution
  - Perl
  - Mono (.NET, C#, …)
  - Java (?)
- *native language still available*
  - *the old kamailio.cfg scripting language is still there, maintained and developed*

```
298  # Routing to foreign domains
299  route[SIPOUT] {
300          if (uri==myself) return;
301
302          append_hf("P-hint: outbound\r\n");
303          route(RELAY);
304          exit;
305  }
```

Scripting SIP Routing In Lua

- app_lua
  - existing since 2010 offering inline execution of Lua scripts
  - https://www.kamailio.org/docs/modules/stable/modules/app_lua.html
- highlights
  - very small interpreter, fast
  - decent number of extensions (native lua libraries)
  - popular in gaming space, also in RTC (Asterisk, Freeswitch)

https://github.com/kamailio/kamailio/blob/master/misc/examples/kemi/kamailio-basic-kemi-lua.lua

```
324  -- Routing to foreign domains
325  function ksr_route_sipout()
326          if KSR.is_myself(KSR.pv.get("$ru")) then return 1; end
327
328          KSR.hdr.append_hf("P-Hint: outbound\r\n");
329          ksr_route_relay();
330          KSR.x.exit();
331  end
```

Scripting SIP Routing In JavaScript

- introduced in Kamailio v5.1
  - not in stable version
  - https://www.kamailio.org/docs/modules/devel/modules/app_jsdt.html
- highlights
  - duktape.org engine - small, easy to embed by importing source code files
  - very popular scripting language
  - no external dependencies
  - reload of routing script
  - not many extensions

https://github.com/kamailio/kamailio/blob/master/misc/examples/kemi/kamailio-basic-kemi-jsdt.js

```
331  // Routing to foreign domains
332  function ksr_route_sipout()
333  {
334          if (KSR.is_myself(KSR.pv.get("$ru"))) { return; }
335
336          KSR.hdr.append_hf("P-Hint: outbound\r\n");
337          ksr_route_relay();
338          KSR.x.exit();
339  }
```

Scripting SIP Routing In Python

- app_python - app_python3
  - introduced in 2010 for inline execution of Python scripts
  - https://www.kamailio.org/docs/modules/stable/modules/app_python.html
  - https://www.kamailio.org/docs/modules/devel/modules/app_python3.html
- highlights
  - popular scripting language with extensive number of extensions
  - object oriented, perceived as slower than other scripting languages
  - reloading of the routing script not implemented yet

https://github.com/kamailio/kamailio/blob/master/misc/examples/kemi/kamailio-basic-kemi-python.py

```
332     # Routing to foreign domains
333     def ksr_route_sipout(self, msg):
334         if KSR.is_myself(KSR.pv.get("$ru")) :
335             return 1;
336
337         KSR.hdr.append("P-Hint: outbound\r\n");
338         self.ksr_route_relay(msg);
339         return -255;
```

Scripting SIP Routing In Squirrel

- app_sqlang
  - introduced in Kamailio 5.1
  - https://www.kamailio.org/docs/modules/devel/modules/app_sqlang.html
- highlights
  - http://www.squirrel-lang.org
  - very small interpreter
  - embedded by importing its source code
  - no external dependency
  - reload of scripting script
  - pre-compilation options

Squirrel
the programming language

https://github.com/kamailio/kamailio/blob/master/misc/examples/kemi/kamailio-basic-kemi-sqlang.sq

```
332  // Routing to foreign domains
333  function ksr_route_sipout()
334  {
335          if (KSR.is_myself(KSR.pv.get("$ru"))) { return; }
336
337          KSR.hdr.append_hf("P-Hint: outbound\r\n");
338          ksr_route_relay();
339          KSR.x.exit();
340  }
```

KEMI Insights

how functions are exported

the relation between C function and KEMI export.

- estimated 1024 max KSR functions
- auto generated 1024 Lua exports functions
- index KSR functions in one-to-one relation

Exported To KEMI (Lua)	Internal Kamailio Structure
sr_kemi_lua_exec_func_0	sr_kemi_lua_export_t(t_relay)
…	…
sr_kemi_lua_exec_func_100	sr_kemi_lua_export_t(sl_send_reply)
…	…
null	null
…	…

```
static int sr_kemi_lua_exec_func_0(lua_State *L)
{
	return sr_kemi_lua_exec_func(L, 0);
}

...

static int sr_kemi_lua_exec_func_1023(lua_State *L)
{
	return sr_kemi_lua_exec_func(L, 1023);
}

...
typedef struct sr_kemi_lua_export {
	lua_CFunction pfunc;
	sr_kemi_t *ket;
} sr_kemi_lua_export_t;

static sr_kemi_lua_export_t _sr_kemi_lua_export_list[] = {
	{ sr_kemi_lua_exec_func_0, NULL},
...
	{ sr_kemi_lua_exec_func_1023, NULL},
	{NULL, NULL}
};
```

KEMI Insights

how functions are exported

the relation between C function and KEMI export.

```
sr_kemi_t *sr_kemi_lua_export_get(int idx)
{
	if(idx<0 || idx>=SR_KEMI_LUA_EXPORT_SIZE)
		return NULL;
	return _sr_kemi_lua_export_list[idx].ket;
}

...

int sr_kemi_lua_exec_func(lua_State* L, int eidx)
{
	sr_kemi_t *ket;

	ket = sr_kemi_lua_export_get(eidx);
	return sr_kemi_lua_exec_func_ex(L, ket, 0);
}
```

- *Performances*
- testing system
  - VirtualBox with 2GB or RAM and 2 processors, having Linux Mint as guest OS. The host was a MAC OS X running on a Mid 2012 model of Macbook Pro
- results - micro-seconds (1 / 1 000 000 of a second)
  - INTERPRETER - AVERAGE - MIN - MAX
  - NATIVE - 302.275 - 6 - 3824
  - LUA - 308.32 - 6 - 3596

KEMI Insights

how functions are exported

the relation between C function and KEMI export.

```
static int ki_set_source_address(sip_msg_t *msg, str *saddr)
{
	sr_phostp_t rp;
	union sockaddr_union faddr;
	char cproto;
	int ret;

	if(msg==NULL || saddr==NULL || saddr->len<=0) {
		LM_ERR("bad parameters\n");
		return -1;
	}

	if(parse_protohostport(saddr, &rp)<0) {
		LM_ERR("failed to parse the address [%.*s]\n", saddr->len, saddr->s);
		return -1;
	}

	cproto = (char)rp.proto;
	ret = sip_hostport2su(&faddr, &rp.host, (unsigned short)rp.port, &cproto);
	if(ret!=0) {
		LM_ERR("failed to resolve address [%.*s]\n", saddr->len, saddr->s);
		return -1;
	}

	msg->rcv.src_su=faddr;
	su2ip_addr(&msg->rcv.src_ip, &faddr);
	msg->rcv.src_port=rp.port;

	return 1;
}
```

```
static int w_set_source_address(sip_msg_t *msg, char *paddr, char *p2)
{
	str saddr;
	if (fixup_get_svalue(msg, (gparam_t*)paddr, &saddr) != 0 || saddr.len<=0) {
		LM_ERR("cannot get source address value\n");
		return -1;
	}
	return ki_set_source_address(msg, &saddr);
}
```

Exported to native configuration interpreter
- evaluate parameter with pseudo-variables
- execute the function exported to KEMI

- set_source_address(“udp:$var(ip):5080”)

Exported to KEMI

- KSR.corex.set_source_address(“udp:127.0.0.1:5080”)
- KSR.corex.set_source_address(“udp:” .. KSR.pv.get(“$var(ip)”) .. “:5080”)

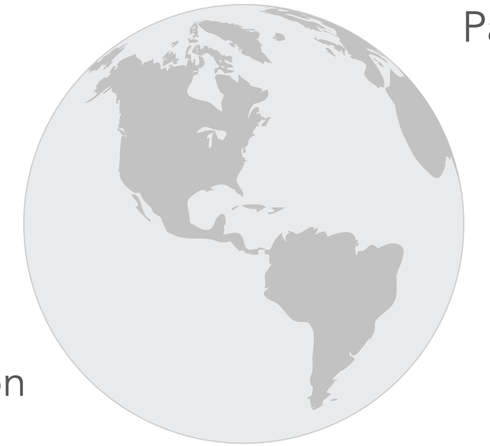
Native vs. KEMI

the major characteristics

Routing callbacks

Function names

Parameters to functions



Return code evaluation

Stop script execution

http://kamailio.org/docs/tutorials/devel/kamailio-kemi-framework/

Native vs. KEMI

routing callbacks

Native Scripting Language	KEMI Scripting Language
request_route { ... }	ksr_request_route()
reply_route { ... }	ksr_reply_route()
onsend_route { ... }	ksr_onsend_route{}
route[NAME] { ... }	any function name
event_route[NAME] { ... }	name specified by event_callback parameter for modules
branch_route[NAME] { ... }	any function name and engaged via KSR.tm.t_on_branch("F")
onreply_route[NAME] { ... }	any function name and engaged via KSR.tm.t_on_reply("F")
failure_route[NAME] { ... }	any function name and engaged via KSR.tm.t_on_failure("F")
event_route[tm:branch-failure:NAME] { ... }	any function name and engaged via KSR.tm.t_on_branch_failure("F")

Native vs. KEMI

function names

- **Native scripting language**
- ***C style naming***
  - Just function name
  - *my_function(params)*
    - t_relay()
  - Most of the modules use a common prefix
    - tm: t_relay(), t_reply(), t_replicate
  - Functions exported by core via script interpreter
    - Can have special prototype
  - Most of the functions exported by modules

- **KEMI scripting language**
- ***Object-oriented style naming***
  - The module becomes an object
  - *KSR.module.function(params)*
    - KSR.tm.t_relay()
  - KSR - static object (module) apart of Python
  - Most of function names were preserved, even if they have a duplicate id in the prefix
    - KSR.acc.acc_request(…)
  - Some functions exported by the core, majority by modules
  - Functions exported by core: KSR.function(…)
  - Special KSR submodules: KSR.hdr, KSR.pv and KSR.x

Native vs. KEMI

parameters to functions

- **Native scripting language**

- ***Tokens, int and string parameters***
  - Tokens only for few core functions
    - are aliases to specific values - uri:host
  - Integer parameters to module functions provided as string
    - ds_select_dst(“1”, “4”)
  - String parameters to module functions can have pseudo-variables evaluated at runtime
    - xlog(“this is the r-uri: $ru\n”);
  - Some functions support expressions
    - Better avoid them if not sure
  - Up to 6 parameters
  - Variadic number of parameter for same function
    - int ds_select_dst(set, alg [, limit])

- **KEMI scripting language**

- ***Int and string parameters***
  - Format defined by the scripting language
  - Integer parameters provided as numbers
    - KSR.dispatcher.ds_select_dst(1, 4)
  - String parameters must be given as concatenation expression with pseudo-variables
    - KSR.log(“this is the r-uri: “ .. KSR.pv.get(“$ru”) .. “\n”)
  - Expressions in parameters are supported by the scripting languages
  - Up to 6 parameters
  - Fixed number of parameters
    - int KSR.dispatcher.ds_select_dst(int set, int alg)
    - int KSR.dispatcher.ds_select_dst_limit(int set, int alg, int limit)

Native vs. KEMI

return code evaluation

Native scripting language
return value is an integer
return code evaluation for functions and routing blocks:

- if the return code is < 0, then it is evaluated to false in logical expression.
- if the return code is > 0, then it is evaluated to true in logical expression
- if the return code is = 0, then the interpreter stops the execution of the config file script

```
# handle retransmissions
if (!is_method("ACK")) {
        if(t_precheck_trans()) {
                t_check_trans();
                exit;
        }
        t_check_trans();
}
```

```
-- handle retransmissions
if not KSR.is_ACK() then
        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
end
```

KEMI scripting language
return value can be bool, integer or string (special cases)

- Return of a string is done by the function getting pseudo-variable value
- Most of the functions return integer
- Some (mostly from core) functions return bool
- *Important:* very few functions return 0 to request stop of script execution - KSR.tm.t_check_trans(), KSR.tm.t_newtran(), KSR.websocket.handle_handshake()

Native vs. KEMI

- stop script execution - exit - drop -

Native scripting language

exit()

drop()

return(0)

```
# handle retransmissions
if (!is_method("ACK")) {
        if(t_precheck_trans()) {
                t_check_trans();
                exit;
        }
        t_check_trans();
}
```

```
-- handle retransmissions
if not KSR.is_ACK() then
        if KSR.tmx.t_precheck_trans()>0 then
                KSR.tm.t_check_trans();
                return 1;
        end
        if KSR.tm.t_check_trans()==0 then
                KSR.x.exit();
        end
end
```

KEMI scripting language

very careful: standard exit from scripting language can stop kamailio

- Python 2/3: safe to use their exit() or os.exit()
- Python 2/3: cfg-script drop is: KSR.set_drop() + os.exit()
- Other KEMI scripting languages (Lua, JavaScript, Squirrel):
  - KSR.x.exit()
  - KSR.x.drop()
- Do not forget to evaluate Kamailio-exported functions for return of 0 value

Using KEMI SIP Routing Scripts

Remarks - Tips

- **Benefits**
  - Extensive scripting languages
    - More statements, expressions, functions
  - Less maintenance efforts
    - Developed and tested elsewhere
  - Better documentation and learning resources
    - tutorials, examples, … (for the language itself)
  - Reload support
    - No need to restart Kamailio anymore
    - Takes effect with the next script executing in each Kamailio process
  - Comparable performances with native scripting language
    - Static exports for Lua, JavaScript, Squirrel
    - Python creates a dynamic object (slightly slower)

- **Drawbacks**
  - Working with pseudo-variables has to be done via functions
    - KSR.pv.get(“$rU”)
    - KSR.pv.sets(“$rU”, “test”)
    - KSR.pv.seti(“$var(x)”, 10)
    - KSR.pv_is_null(“$rU”)
    - tip: store value on local variable (e.g., srcip = KSR.pv.get(“$si”))
  - Function static parameter pre-compilation
    - Not needed in most of the cases, because parameters with variables are evaluated each time at runtime
    - Here is mainly about replace/substitute with regexp
    - subst(“/alice/bob/g”) vs. subst(“/alice/$var(user)/g”)
    - Performance impact is not noticeable anyhow
    - tip: search and many other operations can be done with functions from scripting language itsef
  - Not all modules exported to KEMI
    - Main missing group now is the IMS modules
    - tip: time for pull requests

devel examples

work with the code - play with scripts

evosip
the world outloud

netaxis
solutions

2600Hz

sipgate

sip:wise

digium

SIMWOOD
AUTHENTIC WHOLESALE TELECOMMUNICATIONS

KAMAILIO WORLD
CONFERENCE & EXHIBITION
BERLIN GERMANY MAY 14-16, 2018

EVARISTE
SYSTEMS

pascom®

ngvoice

LOD

KAZOOcon

vuc

cnd core network dynamics

Fraunhofer
FOKUS

asipto

Fraunhofer
Forum Berlin