Kamailio Workshop

Quick Getting Started

- ClueCon Deconstructed, August 4-7, 2020 —

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

- Originally from Romania, living in Berlin, Germany
- Computer science software engineer - Polytechnics University Bucharest (2001)
- Researcher in RTC at Fraunhofer Fokus Institute, Berlin, Germany (2002-2005)
- Co-founder, main coordinator and lead developer of Kamailio, an open source SIP Server
- Professional consultancy for SIP, VoIP, Kamailio and all RTC at asipto.com
- Involved in open source real time communications since 2002
- Working with open standard protocols, mainly from IETF, GSMA/3GPP/ITU
- C software developer - mainly VoIP server side infrastructure
- Co-organizer of Kamailio World Conference, FOSDEM RTC DevRoom
- Speaking and promoting OSS RTC at world wide events

KAMAILIO

nowadays

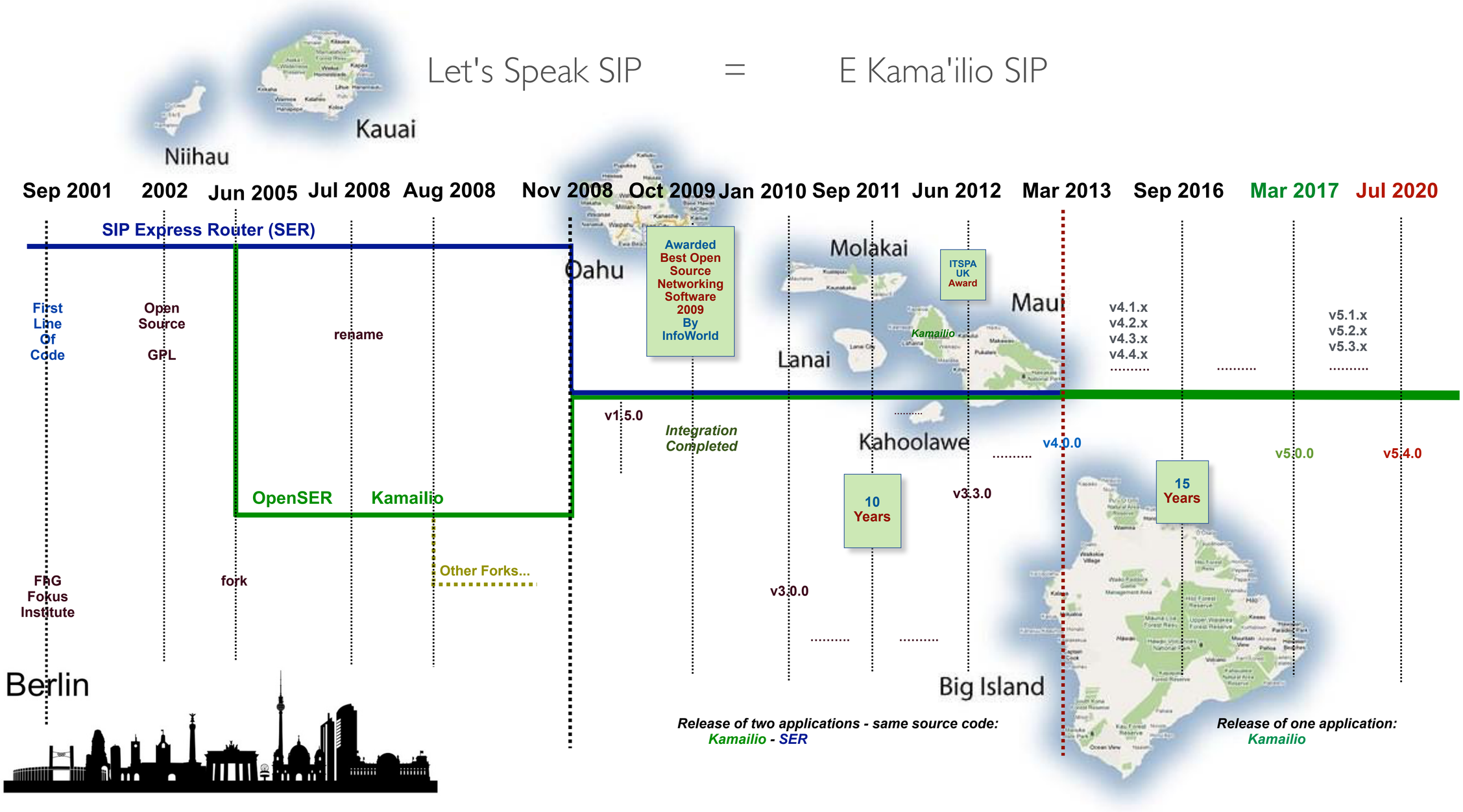
- over 100 yearly active developers - used by large rtc companies (millions of customers, billions of minutes per month) -

KAMAILIO

www.kamailio.org
@kamailio

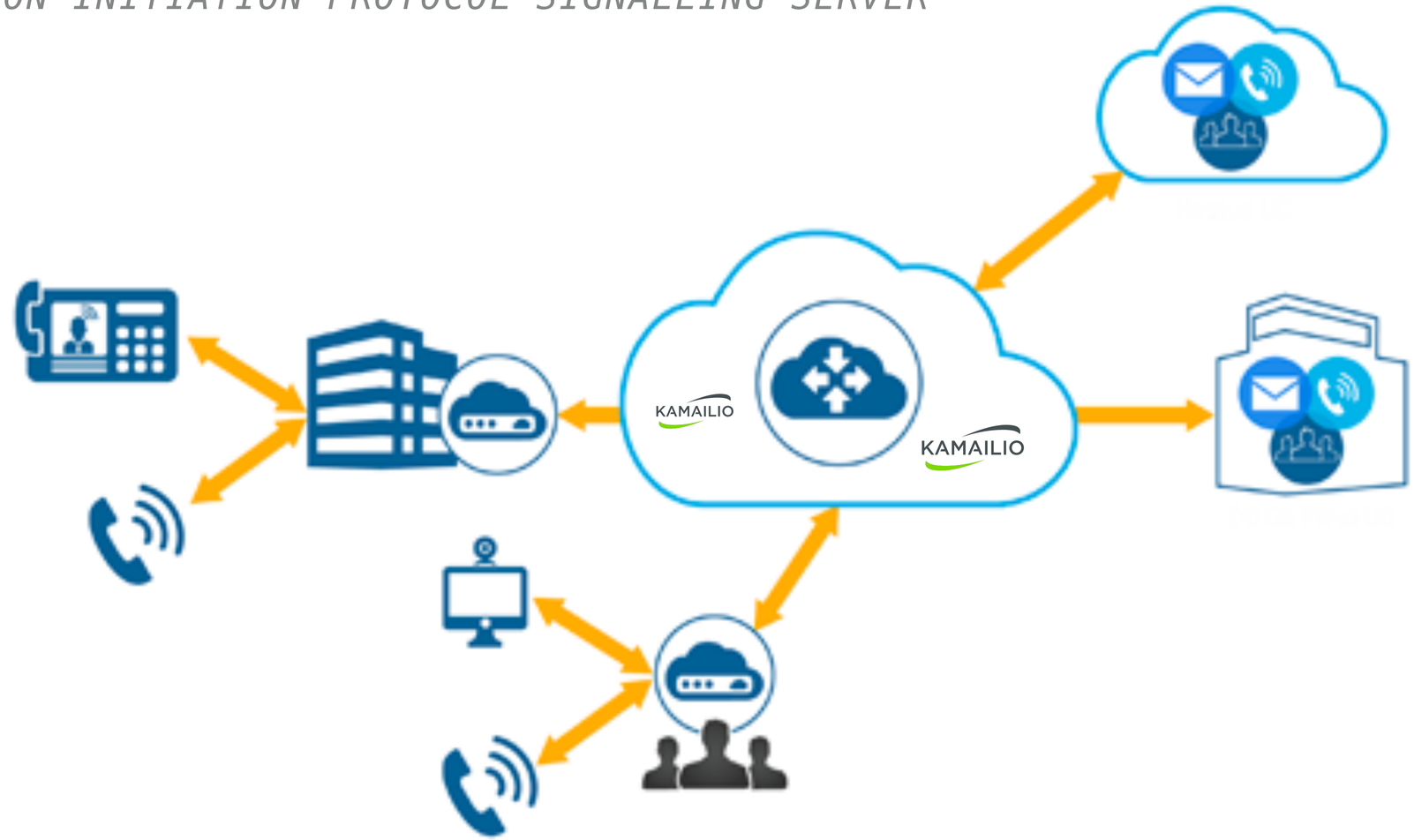
KAMAILIO WORLD
CONFERENCE & EXHIBITION
BERLIN GERMANY APRIL 27-29, 2020

www.kamailioworld.com
@kamailioworld


Let's Speak SIP = E Kama'ilio SIP
Niihau
Kauai
Oahu
Molokai
Lanai
Maui
Kahoolawe
Big Island
Sep 2001
2002
Jun 2005
Jul 2008
Aug 2008
Nov 2008
Oct 2009
Jan 2010
Sep 2011
Jun 2012
Mar 2013
Sep 2016
Mar 2017
Jul 2020
SIP Express Router (SER)
First Line Of Code
Open Source
GPL
rename
fork
OpenSER
Kamailio
Other Forks...
FhG Fokus Institute
Berlin
Awarded Best Open Source Networking Software 2009 By InfoWorld
ITSPA UK Award
Kamailio
v1.5.0
Integration Completed
v3.0.0
10 Years
v3.3.0
v4.0.0
15 Years
v4.1.x
v4.2.x
v4.3.x
v4.4.x
v5.0.0
v5.1.x
v5.2.x
v5.3.x
v5.4.0
Release of two applications - same source code:
Kamailio - SER
Release of one application:
Kamailio

KAMAILIO - CONNECTING HUMANS AND DEVICES USING SIP

SESSION INITIATION PROTOCOL SIGNALLING SERVER

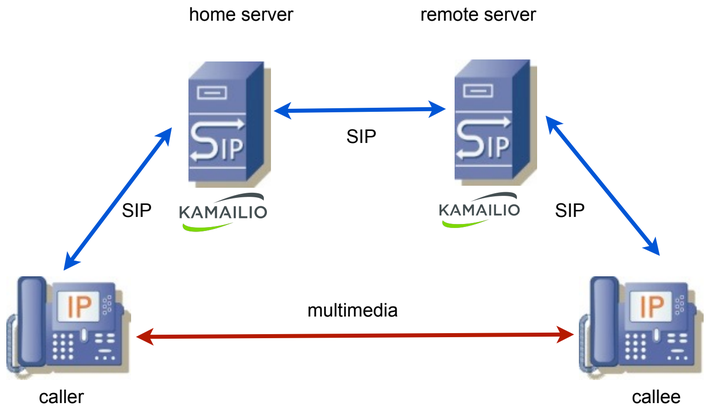


ABOUT KAMAILIO PROJECT

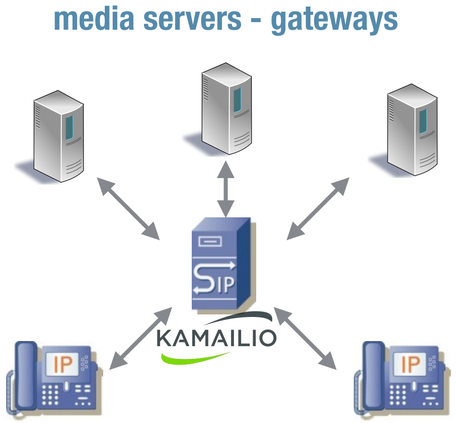
- Open Source SIP (IETF RFC3261) Signaling Server implementation, developed since 2001
  - started at FhG Fokus Institute, Berlin, Germany
- Can be used for VoIP (Voice, Video, VoLTE/IMS, SIP-I/SIP-T), Instant Messaging, Presence, WebRTC, IoT, Diameter, SQL and NoSQL backends
- Designed for modularity, flexibility and scalability
  - thousands of call setups per second, hundred thousands of connected phones per instance
- IPv6/IPv4 - UDP/TCP/TLS/SCTP/WebSocket - asynchronous routing
- Classic SIP - WebRTC gateway using Kamailio + RTPEngine
- Over 200 modules (extensions) - https://www.kamailio.org/docs/modules/stable/
- Community management, with over 50 active developers each year
  - over 300 contributors over the time
- Runs its own conference - Kamailio World
  - Berlin, Germany: https://www.kamailioworld.com
- Used by large telecom, mobile operators and OTT service providers world wide

KAMAILIO

COMMON USE CASES



- authentication, registration and user location
- voice, video, instant messaging and presence
- NAT traversal, RTP relaying, webrtc
- SIP security firewall - DDoS mitigation, anti-fraud
- integration with social networking



- load balancer
- least cost routing
- transport layer gateway (IPv4/IPv6, UDP/TLS)
- topology hiding
- carriers interconnect

AMONG FEATURES

- Implementing IETF RFC3261 and many other RFCs
- SIP proxy, registrar and redirect server
- Support for IPv4 and IPv6 (since 2002)
- Support for UDP, TCP, TLS, SCTP and WebSocket (WebRTC) with asynchronous processing
- User authentication: www-digest with md5 or sha256, client TLS certificate
- ACL based on source IP, group membership, capability string
- Database SQL backends: mysql, postgres, sqlite, oracle, unixodbc
- No-SQL backends: redis, mongodb, cassandra, memcached
- RADIUS and Diameter support
- IMS/VoLTE/5G extensions - I-CSCF, P-CSCF, S-CSCF, online charging, QoS
- postpaid and prepaid accounting
- DNS NAPTR and SRV routing, DNS caching

KAMAILIO

AMONG FEATURES

- ENUM routing
- SIP SIMPLE presence extensions, including embedded XCAP server and client, BLF, SLA
- MSRP relay implementation
- Instant messaging, custom IM conferencing, offline message storing
- Call forwarding, call barring, group calling
- Load balancing
- Least cost routing
- Topology hiding and topology stripping
- Integration with mobile devices for push notifications
- Registration to remote servers
- Number portability
- Generic database (pseudo) clustering
- Integration with statsd and prometeus

KAMAILIO

AMONG FEATURES

- Connector to Kafka
- Active calls tracking, limits for number of active calls
- Traffic shaping with number of requests per second, filtered per IP, user ID, …
- User aliases (multiple identities) and short dialing
- Multi domain support (hosted PBX services)
- Data replication with distributed message queue systems
- rabbitmq, nsq and mqtt connectors
- Internal message queue for processing offloading
- SDP operations (e.g., remove codecs, video media streams)
- API-based routing
- Event API exposure
- STIR/SHAKEN support

KAMAILIO

Resources

- **Website: https://www.kamailio.org**
- Development project: https://github.com/kamailio/kamailio
- Documentation page: https://www.kamailio.org/w/documentation/
- Wiki tutorials: https://www.kamailio.org/wiki/
- Bug tracker: https://github.com/kamailio/kamailio/issues
- Users mailing list: https://lists.kamailio.org/cgi-bin/mailman/listinfo/sr-users
- Realtime channels: IRC: #kamailio (freenode.net) - Matrix: #kamailio (kamailio.dev)
- Business directory: https://www.kamailio.org/w/business-directory/
- Download: https://www.kamailio.org/w/download/ (or OS distro repos)

MAIN COOKBOOKS

- Core CookBook
  - Global parameters
  - Core functions
  - Configuration file elements
  - Routing blocks
  - https://www.kamailio.org/wiki/cookbooks/5.4.x/core
- Variables CookBook
  - List of variables
  - https://www.kamailio.org/wiki/cookbooks/5.4.x/pseudovariables
- Transformations CookBook
  - List of transformations - sort of functions applied to variables
  - https://www.kamailio.org/wiki/cookbooks/5.4.x/transformations

MODULES DOCUMENTATION

https://www.kamailio.org/docs/modules/stable/

KAMAILIO MODULES - V5.4.X (STABLE)

Module	Description	Status
- A - B - C - D - E - F - G - H - I - J - K - L - M - N - O - P - Q - R - S - T - U - V - W - X - Y - Z -		
ACC	Accounting module	released
ACC_DIAMETER	Accounting module for DIAMETER backend	alpha
ACC_JSON	Accounting module generating a JSON document pushed to MQueue	released
ACC_RADIUS	Accounting module for RADIUS backend	released
ALIAS_DB	Database aliases module	released
APP_JAVA	Execute embedded Java applications	released
APP_JSDT	Execute embedded JavaScript scripts	released
APP_LUA	Execute embedded Lua scripts	released
APP_LUA_SR	Export old Lua sr library	released
APP_MONO	Execute embedded managed code (e.g., C#, VisualBasic.NET, Java, Java Script)	released
APP_PERL	Embed execution of Perl functions	released
APP_PYTHON	Execute embedded Python2 scripts	released

KAMAILIO

COMMAND LINE PARAMETERS

```
version: kamailio 5.4.0 (x86_64/darwin) 73c97e
Usage: kamailio [options]
Options:
    -a mode      Auto aliases mode: enable with yes or on,
                  disable with no or off
    --alias=val  Add an alias, the value has to be '[proto:]hostname[:port]'
                  (like for 'alias' global parameter)
    -A define    Add config pre-processor define (e.g., -A WITH_AUTH,
                  -A 'FLT_ACC=1', -A 'DEFVAL="str-val"')
    -b nr        Maximum receive buffer size which will not be exceeded by
                  auto-probing procedure even if  OS allows
    -c           Check configuration file for syntax errors
    -d           Debugging level control (multiple -d to increase the level from 0)
    --debug=val  Debugging level value
    -D           Control how daemonize is done:
                  -D..do not fork (almost) anyway;
                  -DD..do not daemonize creator;
                  -DDD..daemonize (default)
    -e           Log messages printed in terminal colors (requires -E)
    -E           Log to stderr
    -f file      Configuration file (default: /tmp/kamailio-5.4/etc/kamailio/kamailio.cfg)
    -g gid       Change gid (group id)
    -G file      Create a pgid file
    -h           This help message
    --help       Long option for `-h`
    -I           Print more internal compile flags and options
    -K           Turn on "via:" host checking when forwarding replies
    -l address   Listen on the specified address/interface (multiple -l
                  mean listening on more addresses). The address format is
                  [proto:]addr_lst[:port][/advaddr],
                  where proto=udp|tcp|tls|sctp,
                  addr_lst= addr|(addr, addr_lst),
                  addr=host|ip_address|interface_name and
                  advaddr=addr[:port] (advertised address).
                  E.g: -l localhost, -l udp:127.0.0.1:5080, -l eth0:5062,
                  -l udp:127.0.0.1:5080/1.2.3.4:5060,
                  -l "sctp:(eth0)", -l "(eth0, eth1, 127.0.0.1):5065".
                  The default behaviour is to listen on all the interfaces.
```

COMMAND LINE PARAMTERS

```
--loadmodule=name load the module specified by name
--log-engine=log engine name and data
-L path      Modules search path (default: /tmp/kamailio-5.4/lib64/kamailio/modules)
-m nr        Size of shared memory allocated in Megabytes
--modparam=modname:paramname:type:value set the module parameter
              type has to be 's' for string value and 'i' for int value,
              example: --modparam=corex:alias_subdomains:s:kamailio.org
-M nr        Size of private memory allocated, in Megabytes
-n processes Number of child processes to fork per interface
              (default: 8)
-N           Number of tcp child processes (default: equal to `-n')
-O nr        Script optimization level (debugging option)
-P file      Create a pid file
-Q           Number of sctp child processes (default: equal to `-n')
-r           Use dns to check if is necessary to add a "received="
              field to a via
-R           Same as `-r` but use reverse dns;
              (to use both use `-rR`)
--server-id=num set the value for server_id
--subst=exp set a subst preprocessor directive
--substdef=exp set a substdef preprocessor directive
--substdefs=exp set a substdefs preprocessor directive
-S           disable sctp
-t dir       Chroot to "dir"
-T           Disable tcp
-u uid       Change uid (user id)
-v           Version number
--version    Long option for `-v`
-V           Alternative for `-v`
-x name      Specify internal manager for shared memory (shm)
              - can be: fm, qm or tlsf
-X name      Specify internal manager for private memory (pkg)
              - if omitted, the one for shm is used
-Y dir       Runtime dir path
-w dir       Change the working directory to "dir" (default: "/")
-W type      poll method (depending on support in OS, it can be: poll,
              epoll_lt, epoll_et, sigio_rt, select, kqueue, /dev/poll)
```

ARCHITECTURE - SIP TOOLKIT

BUILDING BLOCKS FOR CREATING SIP ROUTING SYSTEMS

- Authentication
- Authorization
- Accounting
- Registration
- Location
- Least cost routing
- Load balancing
- Encryption
- ...

KAMAILIO

AGAIN - BUILDING SIP ROUTING SYSTEMS

VOIP - IP TELEPHONY IS A USE CASE OF SIP

KNOWING SIP IS VERY IMPORTANT!!!

KNOWING SIP IS VERY IMPORTANT!!!

KNOWING SIP IS VERY IMPORTANT!!!

LEARNING THE BASICS OF SCRIPTING LANGUAGES HELPS A LOT AS WELL

SIP: BASIC PROTOCOL ARCHITECTURE

- RFC3261 + hundred extensions

request

Start line	**INVITE sip:user@sipserver.com SIP/2.0**
Message headers	**Via: SIP/2.0/UDP 10.10.10.10:5060;branch=z9hG4bKxy** **From: "Me" <sip:me@sipserver.org>;tag=a012** **To: "User" <sip:user@sipserver.org>** **Call-ID: d@10.10.10.10** **CSeq: 1 INVITE** **Contact: <sip:10.10.10.10:5060>** **User-Agent: SIPTelephone** **Content-Type: application/sdp** **Content-Length: 251**
Message body	**v=0** **o=audio1 0 0 IN IP4 10.10.10.10** **s=session** **c=IN IP4 10.10.10.10** **m=audio 54742 RTP/AVP 4 3** **a=rtpmap:4 G723/8000** **a=rtpmap:3 GSM/8000**

response

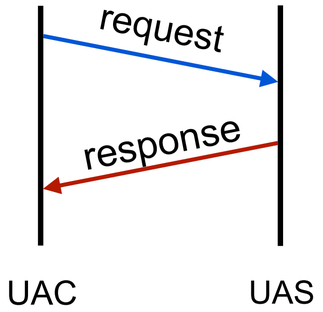
SIP/2.0 200 OK

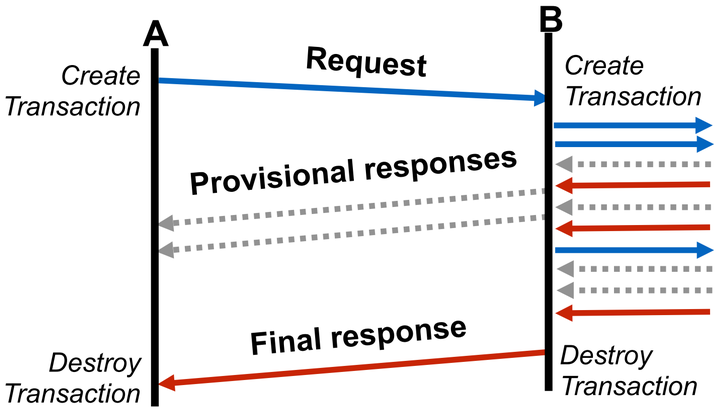
Via: SIP/2.0/UDP 10.10.10.10:5060;branch=z9hG4bKxy **From: "Me" <sip:me@sipserver.org>;tag=a012** **To: "User" <sip:user@sipserver.org>;tag=b034** **Call-ID: d@10.10.10.10** **CSeq: 1 INVITE** **Contact: <sip:10.10.10.20:5060>** **User-Agent: SIPSoftPhone** **Content-Type: application/sdp** **Content-Length: 123**
v=0 **o=audio2 0 0 IN IP4 10.10.10.20** **s=session** **c=IN IP4 10.10.10.20** **m=audio 62043 RTP/AVP 0 4**

response == reply

SIP - THE BASIC KEY CONCEPTS

- SIP request
- SIP reply (response)
  - Provisional: 100 - 199
  - Successful: 200-299
  - Failure: 300-699
- SIP transaction
- SIP dialog
- SIP Request URI
- SIP headers
  - Call-Id, CSeq
  - From, To — and the tag parameter
  - Contact, Via, Record-Route, Route
- Session Description Protocol (SDP)





CONFIGURATION FILE: SCRIPTING SIP ROUTING

- 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
    - Initially designed in 2001-2002, built from scratch
  - Kamailio Embedded Interface (KEMI) languages
    - Introduced in v5.0
    - Reuse existing scripting languages
    - Support for Lua, Python 2/3, JavaScript, Ruby, Squirrel language
    - Allow reloading of scripts without restart
  - Inline execution of scripting languages or REST-API based routing
    - Can be executed inside native scripting language
    - Support for Lua, JavaScript, Python, Perl, .Net (C#, …), Squirrel, Java

```
# global settings
#!define FLT_ACC 1
debug=9
fork=no
listen=192.168.1.34:5060
…
pstn.gw = 1.2.3.4” desc “pstn
gateway ip”
...
```

```
# module settings
mpath=”/usr/local/lib/kamailio/modules/”
loadmodule=”tm.so”
...
modparam("tm", "fr_inv_timer", 30000)
.....
```

```
# routing blocks
request_route {
    xlog(“request received from $si\n”);
    if($si==“10.1.2.10”) {
        route(REDIRECT);
    } else {
        $rd = “10.1.2.5”;
    }
    t_on_reply(“LOGRPL”);
    t_relay();
}
route[REDIRECT] {
    $rd = “10.1.2.3”;
    send_reply(“302”, “Redirected”);
    exit;
}
onreply_route[LOGRPL] {
   xlog(“response received from $si\n”);
}
...
```

KEMI Framework

Empowering Kamailio Since 2016

- introduced in Kamailio v5.0.0
  - use other scripting languages for writing SIP routing logic (route blocks)
  - exports a function one time and becomes available in embedded interpreters
  - ***allows reloading routing script without restart***
  - ***enables the use of an extensive number of extensions available in the scripting languages***
  - ***lua, python2, python3, javascript, ruby, squirrel***

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

Seeing Is Believing


```
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  }
```

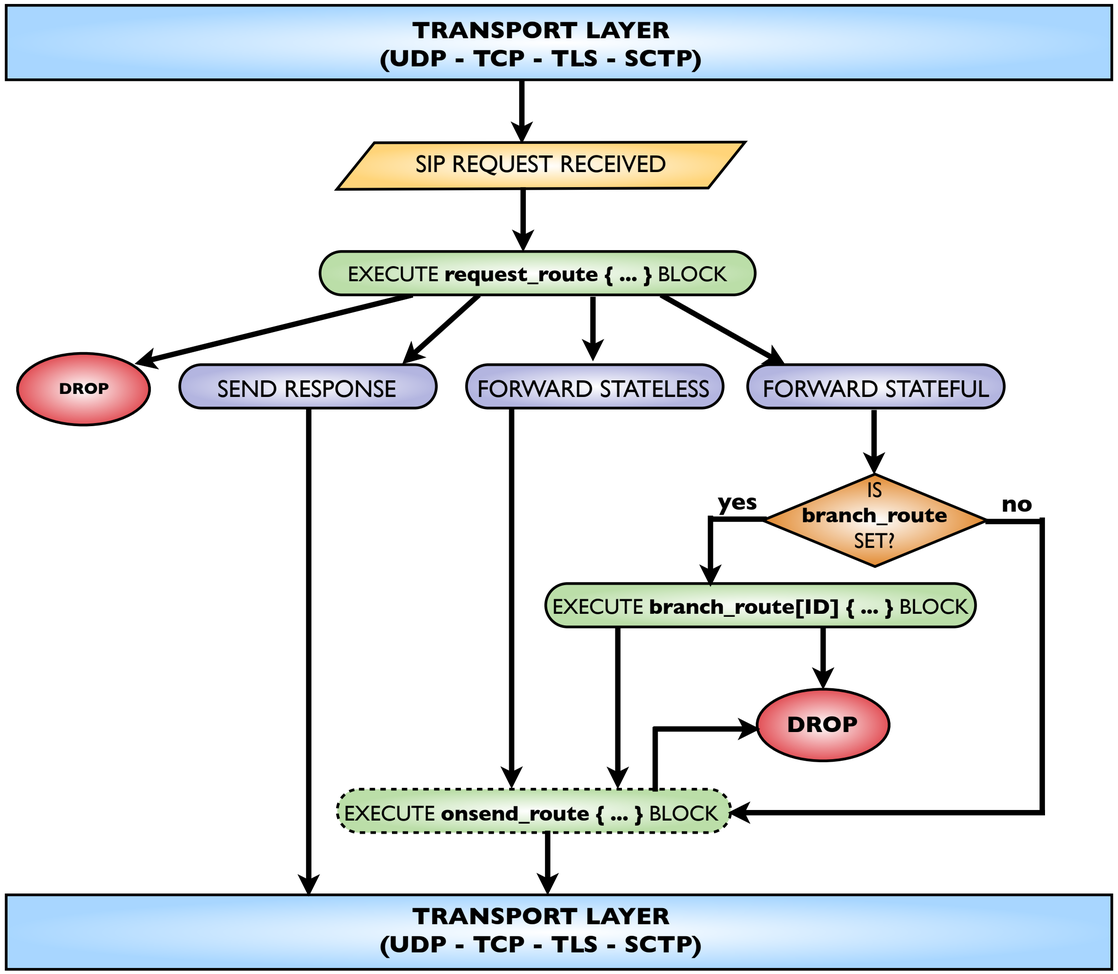
```
324  -- Routing to foreign domains
325  function ksr_route_sipout()
326          if KSR.is_myself(KSR.pv.get("$ru")) then return 1; end
327
328          KSR.hdr.append_hf("P-Hint: outbound\r\n");
329          ksr_route_relay();
330          KSR.x.exit();
331  end
```

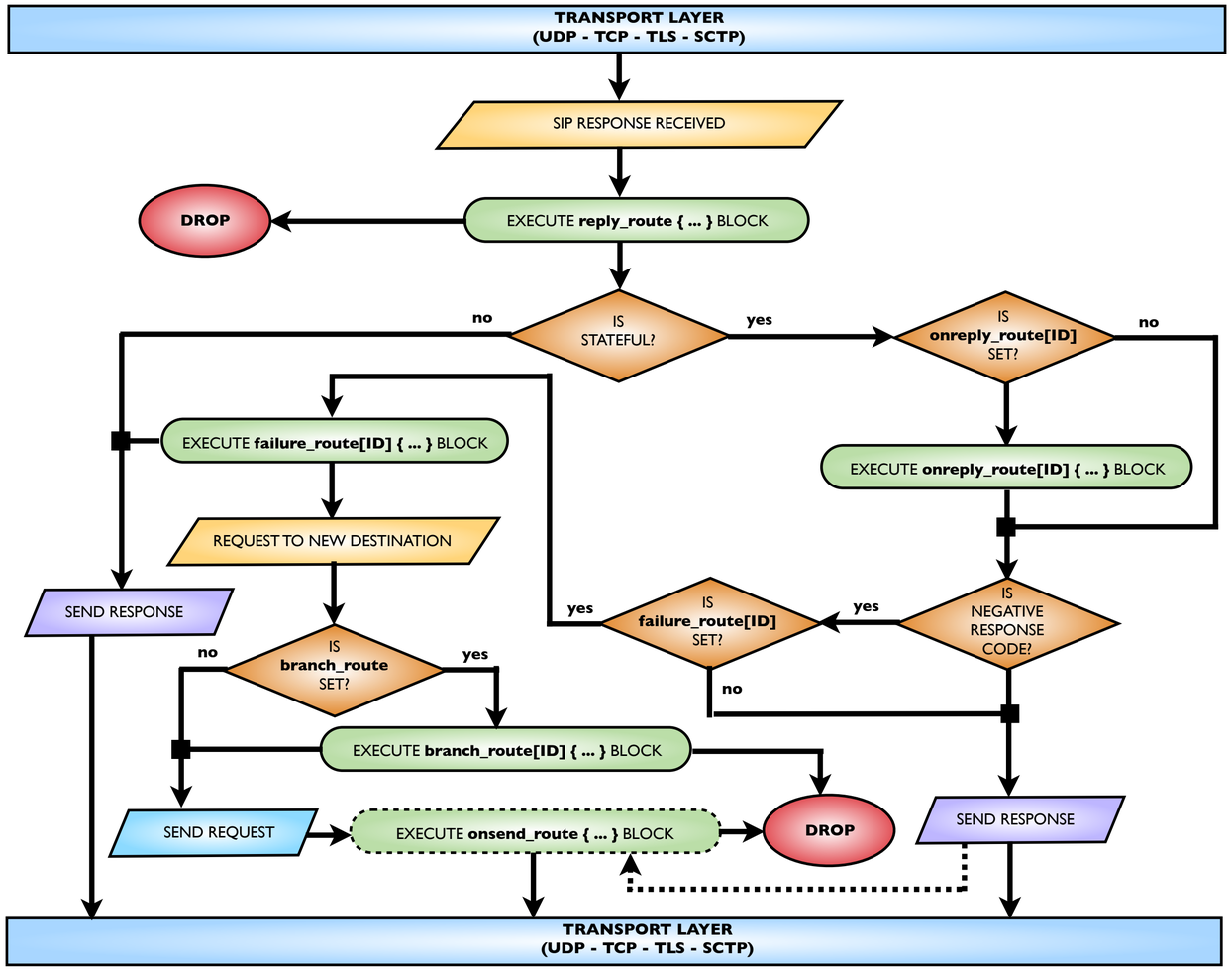
```
332      # Routing to foreign domains
333      def ksr_route_sipout(self, msg):
334          if KSR.is_myself(KSR.pv.get("$ru")) :
335              return 1;
336
337          KSR.hdr.append("P-Hint: outbound\r\n");
338          self.ksr_route_relay(msg);
339          return -255;
```

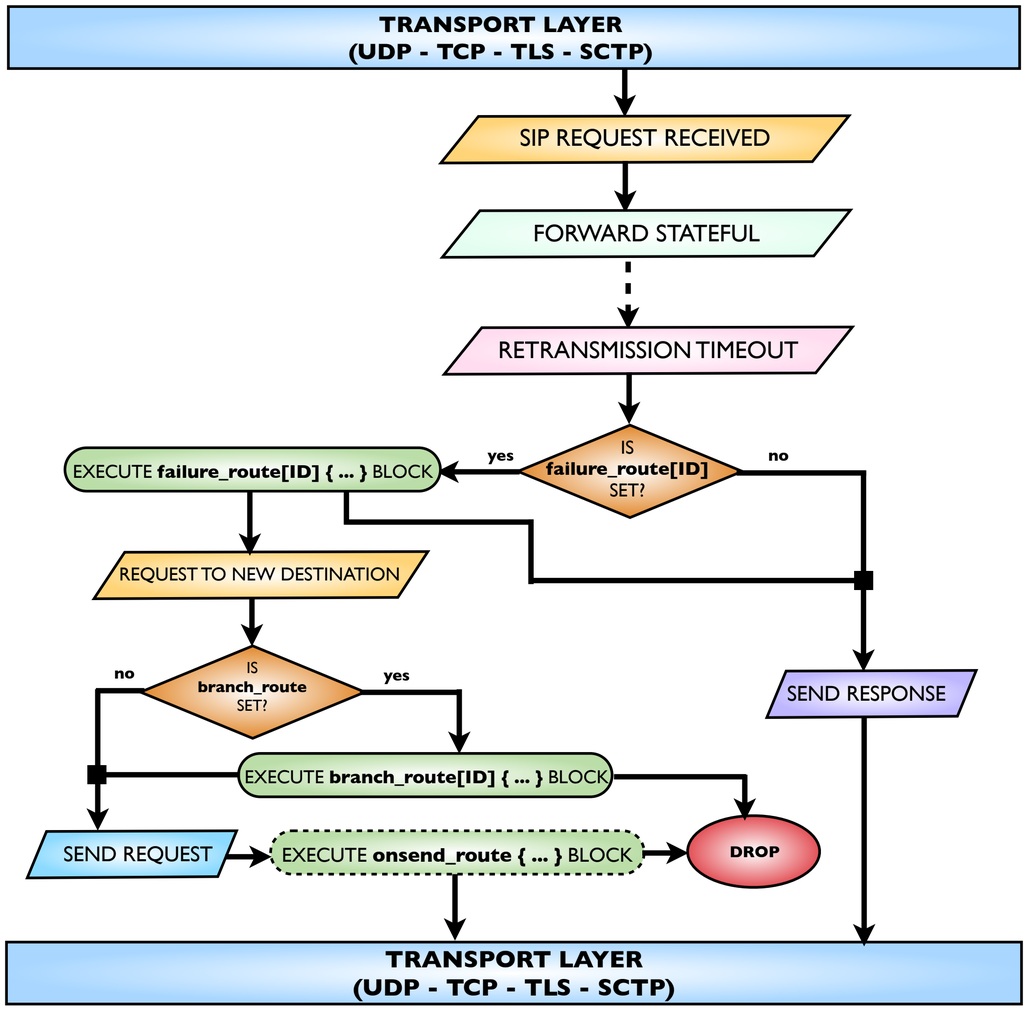
https://github.com/kamailio/kamailio/tree/master/misc/examples/kemi

DEFAULT CONFIGURATION FILE

https://github.com/kamailio/kamailio/blob/5.4/etc/kamailio.cfg

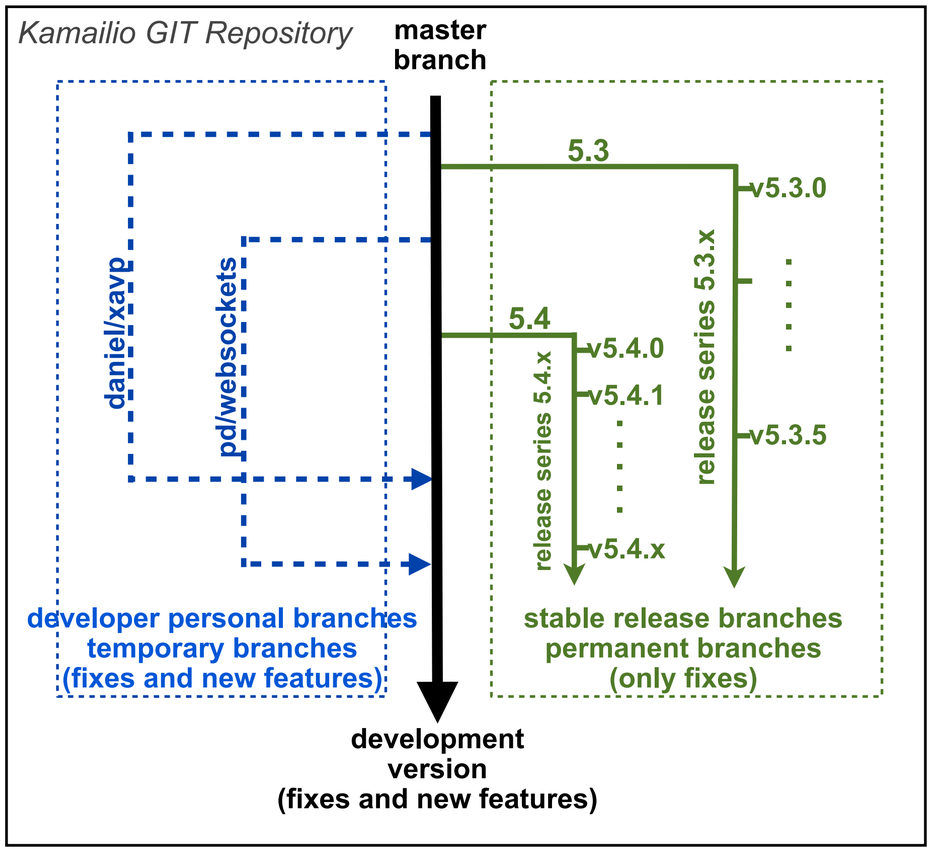

TRANSPORT LAYER
(UDP - TCP - TLS - SCTP)
SIP REQUEST RECEIVED
EXECUTE request_route { ... } BLOCK
DROP
SEND RESPONSE
FORWARD STATELESS
FORWARD STATEFUL
IS
branch_route
SET?
yes
no
EXECUTE branch_route[ID] { ... } BLOCK
DROP
EXECUTE onsend_route { ... } BLOCK
TRANSPORT LAYER
(UDP - TCP - TLS - SCTP)


TRANSPORT LAYER
(UDP - TCP - TLS - SCTP)
SIP RESPONSE RECEIVED
EXECUTE reply_route { ... } BLOCK
DROP
IS STATEFUL?
no
yes
IS onreply_route[ID] SET?
no
EXECUTE onreply_route[ID] { ... } BLOCK
EXECUTE failure_route[ID] { ... } BLOCK
REQUEST TO NEW DESTINATION
SEND RESPONSE
IS NEGATIVE RESPONSE CODE?
yes
IS failure_route[ID] SET?
yes
no
IS branch_route SET?
no
yes
EXECUTE branch_route[ID] { ... } BLOCK
SEND REQUEST
EXECUTE onsend_route { ... } BLOCK
DROP
SEND RESPONSE
TRANSPORT LAYER
(UDP - TCP - TLS - SCTP)


TRANSPORT LAYER
(UDP - TCP - TLS - SCTP)
SIP REQUEST RECEIVED
FORWARD STATEFUL
RETRANSMISSION TIMEOUT
IS
failure_route[ID]
SET?
yes
no
EXECUTE failure_route[ID] { ... } BLOCK
REQUEST TO NEW DESTINATION
IS
branch_route
SET?
no
yes
SEND RESPONSE
EXECUTE branch_route[ID] { ... } BLOCK
SEND REQUEST
EXECUTE onsend_route { ... } BLOCK
DROP
TRANSPORT LAYER
(UDP - TCP - TLS - SCTP)

Kamailio

GIT Repo & Releases


Kamailio GIT Repository
master
branch
daniel/xavp
pd/websockets
5.3
v5.3.0
release series 5.3.x
v5.3.5
5.4
v5.4.0
v5.4.1
release series 5.4.x
v5.4.x
developer personal branches
temporary branches
(fixes and new features)
stable release branches
permanent branches
(only fixes)
development
version
(fixes and new features)

Kamailio

Basic Debugging

LOGGING FUNCTIONS

- **core** - log messages with static strings
  - ***log([level], message);***
    - *level* – integer value, lower value is higher priority
    - *message* – static string message
  - *log(1,"this is one message\n")*
  - *log("this is a second message\n")*
- **xlog** module - log messages with dynamic string
  - log levels and log files
  - ***xlog("level", "message");***
    - *level* – string of level id
    - *message* – dynamic string message
  - ***xdbg("message");***
    - *message* – dynamic string message

xlog("L_ERR", "the SIP method is $rm and RURI is $ru\n")

- core parameters:
  - **debug** - sets the log level threshold for selecting printed messages
  - **log_stderror** - print log messages to stderror or syslog
  - **log_facility** - select the syslog facility where to print messages
  - **log_name** - name of the application to use in syslog messages
  - **log_color** - print using different colors for log levels (for stderror)
  - **log_prefix** - add a prefix to all log messages printed when routing SIP
- log levels
  - L_ALERT (-5) - should be used if the error requires immediate action.
  - L_BUG (-4) - should be used for bug detected cases
  - L_CRIT (-2) - should be used if the error is a critical situation.
  - L_ERR (-1) - should be used to report errors during data processing which do not cause system malfunctioning.
  - L_WARN (0) - should be used to write warning messages.
  - L_NOTICE (1) - should be used to report unusual situations.
  - L_INFO (2) - should be used to write informational messages.
  - L_DBG (3) - should be used to write messages for debugging.

BENCHMARKING

- **core** - latency logs
  - latency_cfg_log - (level) - execution time for request_route and reply_route
  - latency_limit_action - (milli-secs) - limit to print log message when action was slower
  - latency_limit_db - (milli-secs) - limit to print log message when a DB query was slower
  - latency_log - (level) - which level to be used for latency limits logs

- **benchmark** module - how long it takes to execute a part of config script
  - microsecond precision
  - nanosecond precision when compiled with librt
  - metrics of average execution time for a bm-timer

```
...
bm_start_timer("usrloc-lookup");
lookup("location");
bm_log_timer("usrloc-lookup");
...
```

DEBUGGER MODULE

- Step by step execution of configuration file - gdb-like analysis
  - Interactive - commands via RPC interface - not for production systems
  - Run with one SIP worker to be easy to spot the PID
  - Can print values for variables at each execution step
- Config file execution trace
  - Print log messages with executed actions in config
  - Can be enabled for production if needed (still be careful)
- Per module log level
- Print modified SIP message in logs
- Log details about assign acctions
- Print all config variables

```
...
modparam("debugger", "cfgtrace", 1)
...
```

```
...
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=368 a=6 n=route
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=461 a=17 n=if
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=456 a=26 n=mf_process_max
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=466 a=17 n=if
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=461 a=27 n=sanity_check
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=371 a=6 n=route
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=659 a=3 n=return
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=374 a=6 n=route
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=501 a=17 n=if
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=470 a=25 n=has_totag
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=386 a=17 n=if
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=379 a=26 n=is_method
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=386 a=25 n=t_check_trans
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=389 a=6 n=route
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=643 a=3 n=return
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=393 a=26 n=remove_hf
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=398 a=17 n=if
 9(6285) ERROR: *** cfgtrace: c=[/etc/kamailio/debugger.cfg] l=394 a=26 n=is_method
```

CONFIG SYNTAX ERRORS

~# kamailio -c

loading modules under modules

0(52953) : <core> [cfg.y:3411]: yyerror_at(): parse error in config file **/etc/kamailio/kamailio.cfg**, line **65**, column 5: syntax error

0(52953) : <core> [cfg.y:3411]: yyerror_at(): parse error in config file **/etc/kamailio/kamailio.cfg**, line **65**, column 5: '(")' expected (function call)

~# kamailio -c -f /path/to/file.cfg

~# kamailio -c -E -dd

ONLY SYNTAX ERROR - DOESN”T DETECT STARTUP MISCONFIGURATIONS (E.G., WRONG DB PASSWORD)

OTHER MODULES AND TOOLS

- **cfgutils** module
  - show pkg and shm statistics
  - abort()
    - crash on purpose and get a core file
  - conditions in configuration file based on sip traffic or timer routines
- **cfgt** module
  - write a JSON report of config file execution
  - executed actions, variables, …
  - targeting unit testing
- **siptrace** module
  - write sip traffic to database table
- **sipdump** module
  - write sip traffic and metadata to text files with time-based rotation
- **kamctl, kamcli** and **kamcmd**
  - control debug level
  - dump the values for some variables, hash tables, memory structures
  - get internal statistics
    - *kamctl stats*
    - *kamctl rpc pkg.stats*

Kamailio

Related Applications

KAMCTL

~# kamctl help

```
Existing commands:

-- command 'start|stop|restart|trap'

trap ............................... trap with gdb Kamailio processes
restart ............................ restart Kamailio
start .............................. start Kamailio
stop ............................... stop Kamailio

-- command 'acl' - manage access control lists (acl)

acl show [<username>] .............. show user membership
acl grant <username> <group> ....... grant user membership (*)
acl revoke <username> [<group>] .... grant user membership(s) (*)
```

Manage users - add, remove, change password

 # kamctl add alice@wonderland.com mypasswd

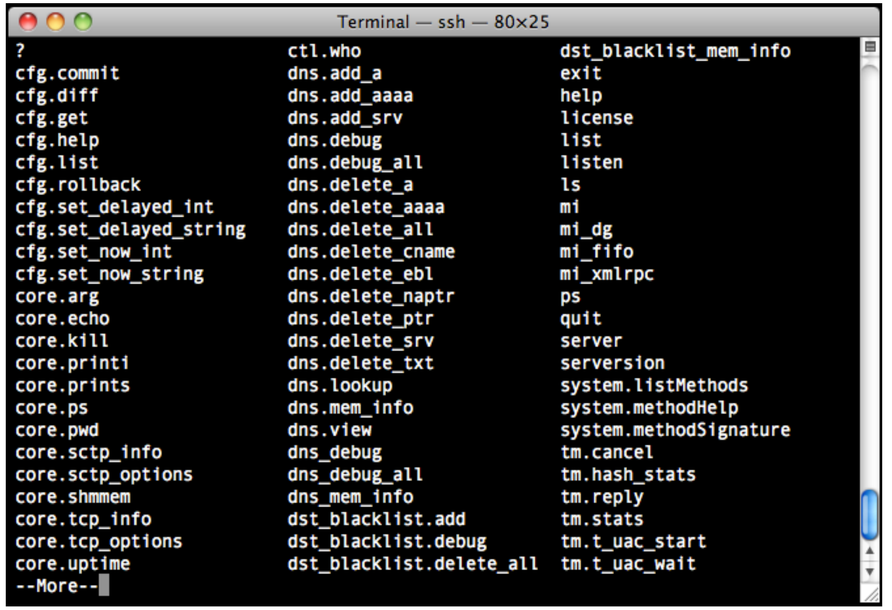
~# kamctl add help

```
-- command 'add|passwd|rm' - manage subscribers

add <username> <password> .......... add a new subscriber (*)
show <username> .................... show subscriber attributes (*)
passwd <username> <passwd> ......... change user's password (*)
rm <username> ...................... delete a user (*)
sets <username> <attr> <val> ....... set string attribute (column value)
setn <username> <attr> <val> ....... set numeric attribute (column value)
```

KAMCMD - KAMCLI

- ***kamcmd*** - part of Kamailio source tree (utils/kamcmd/)
  - a binrpc cli for Kamailio's ctl module written in C
  - previously named sercmd
  - installed along with ctl module
  - auto-completion for rpc command name
    - based on readline library



- ***kamcli*** - rather new Kamailio command line interface tool
  - written in Python
  - extensible via modules
  - easier to handle input values and format output
  - uses RPC (e.g., JSONRPC) control interface
- https://github.com/kamailio/kamcli

```
$ kamcli --help
$ kamcli <command> --help
$ kamcli <command> <subcommand> --help
```

SIREMIS - WEB MANAGEMENT INTERFACE

https://www.siremis.org - https://github.com/asipto/siremis

SIP Tools

Monitor And Analyze SIP Network Traffic
Generate SIP Traffic

SIPSAK

- Generate SIP traffic
- Shoot a SIP OPTIONS request
  - sipsak -s sip:test@sipserver.com
- Simulate various SIP scenario: registration, stress test, call test, message test, …
- Usually available as package in the distro

```
$ ../test/sipsak/sipsak -h
sipsak 0.9.7pre by Nils Ohlmeier
 Copyright (C) 2002-2004 FhG Fokus
 Copyright (C) 2004-2005 Nils Ohlmeier
 report bugs to nils@sipsak.org

 shoot  : sipsak [-f FILE] [-L] -s SIPURI
 trace  : sipsak -T -s SIPURI
 usrloc : sipsak -U [-I|M] [-b NUMBER] [-e NUMBER] [-x NUMBER] [-z NUMBER] -s SIPURI
 usrloc : sipsak -I|M [-b NUMBER] [-e NUMBER] -s SIPURI
 usrloc : sipsak -U [-C SIPURI] [-x NUMBER] -s SIPURI
 message: sipsak -M [-B STRING] [-O STRING] [-c SIPURI] -s SIPURI
 flood  : sipsak -F [-e NUMBER] -s SIPURI
 random : sipsak -R [-t NUMBER] -s SIPURI

 additional parameter in every mode:
   [-a PASSWORD] [-d] [-i] [-H HOSTNAME] [-l PORT] [-m NUMBER] [-n] [-N]
   [-r PORT] [-v] [-V] [-w]
…
```

SIPP

- Generate SIP traffic
- Run as UAC or UAS and generate traffic
  - sipp -sn uac sipserver.com
- Scenarios are defined in xml files and can be a sequence of SIP requests and responses, with or without RTP
- Usually available as package in the distro (can be named sip-scenario)

```
$ sipp -h

Usage:

 sipp remote_host[:remote_port] [options]

Example:

  Run SIPp with embedded server (uas) scenario:
    ./sipp -sn uas
  On the same host, run SIPp with embedded client (uac) scenario:
    ./sipp -sn uac 127.0.0.1

 Available options:


*** Scenario file options:

  -sd             : Dumps a default scenario (embedded in the SIPp executable)
  -sf             : Loads an alternate XML scenario file.  To learn more about XML scenario
…
```

http://sipp.sourceforge.net/

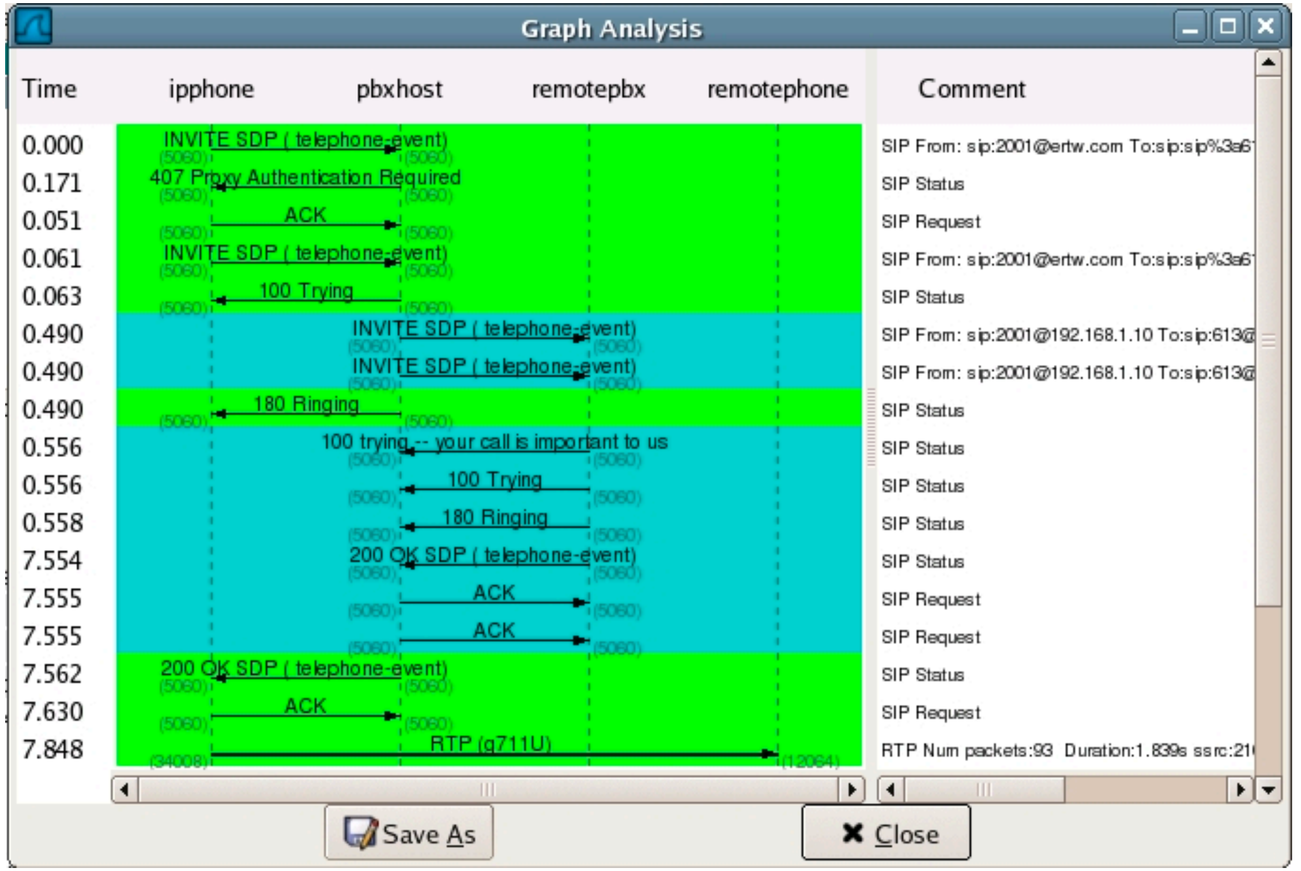
https://github.com/SIPp/sipp

OTHER SIP TRAFFIC GENERATORS

- protoshoot - inside Kamailio source tree: misc/tools/protoshoot
  - Send SIP messages stored in a file via UDP or TCP (replay a SIP message)
- wsctl - send traffic over WebSocket
  - https://github.com/miconda/wsctl
- pjsua - comprehensive command line SIP user agent from PjSIP project
  - https://www.pjsip.org/pjsua.htm
- baresip - command line SIP user agent
  - https://github.com/alfredh/baresip

WIRESHARK

- The most popular network traffic sniffer and analyzer with GUI (Linux, MacOS, Windows)
  - Tshark - command line variant
- http://www.wireshark.org



SNGREP

- Terminal based SIP traffic sniffer with call flow
  - Uses ncurses to draw call flows
- https://github.com/irontec/sngrep

OTHER SIP TRAFFIC MONITORING TOOLS

- ngrep - terminal tool - https://github.com/jpr5/ngrep - http://ngrep.sourceforge.net/usage.html
  - Print text payload of SIP traffic
- sipgrep - terminal tool - https://github.com/sipcapture/sipgrep
- tcpdump - terminal tool - https://www.tcpdump.org/
- sipcapture - web gui - http://sipcapture.org
- sip3 - web gui - https://sip3.io
- flanders - web gui - https://github.com/jonkirkman/flanders

https://kamailio.org/docs/tutorials/5.4.x/kamailio-install-guide-git/

Bonus

Integration With FreeSwitch
For Media Services

PARTICIPATE TO THE DEMO

KAMAILIO SERVER: LAB.KAMAILIO.DEV

TEST SIP ACCOUNTS: 1005, 1006, 1007, ...

ECHO SERVICE VIA FREESWITCH: 9196

JOIN AUDIO CONFERENCE ROUTED TO FREESWITCH - DIAL: 3000

EXTENSIONS ROUTED TO FREESWITCH: 3XXX AND 9XXX

KAMAILIO.CFG CHANGES FOR ROUTING TO FREESWITCH FOR MEDIA SERVICES

FREESWITCH RUNNING ON: 192.168.12.167:5090
- WITH DEFAULT DIALPLAN
- EXTENSIONS 3XXX AND 9XXX ARE ROUTED TO FREESWITCH
- EXAMPLE MEDIA SERVICES:
  - ECHO: 9196 - DELAYED ECHO: 9195
  - AUDIO CONFERENCE: 3000

```
--- etc/kamailio.cfg  2020-08-04 10:23:28.548474917 -0700
+++ etc/kamailio-fs.cfg     2020-08-04 04:56:06.847052158 -0700
@@ -562,6 +562,12 @@
                exit;
        }

+       if($rU =~ "^9[0-9][0-9][0-9]$" || $rU =~ "^3[0-9][0-9][0-9]$") {
+               $ru = "sip:" + $rU + "@192.168.23.167:5090";
+               route(RELAY);
+               exit;
+       }
+
        # dispatch destinations to PSTN
        route(PSTN);
```

KAMAILIO 5.4.0 RUN FROM SOURCE CODE BUILD

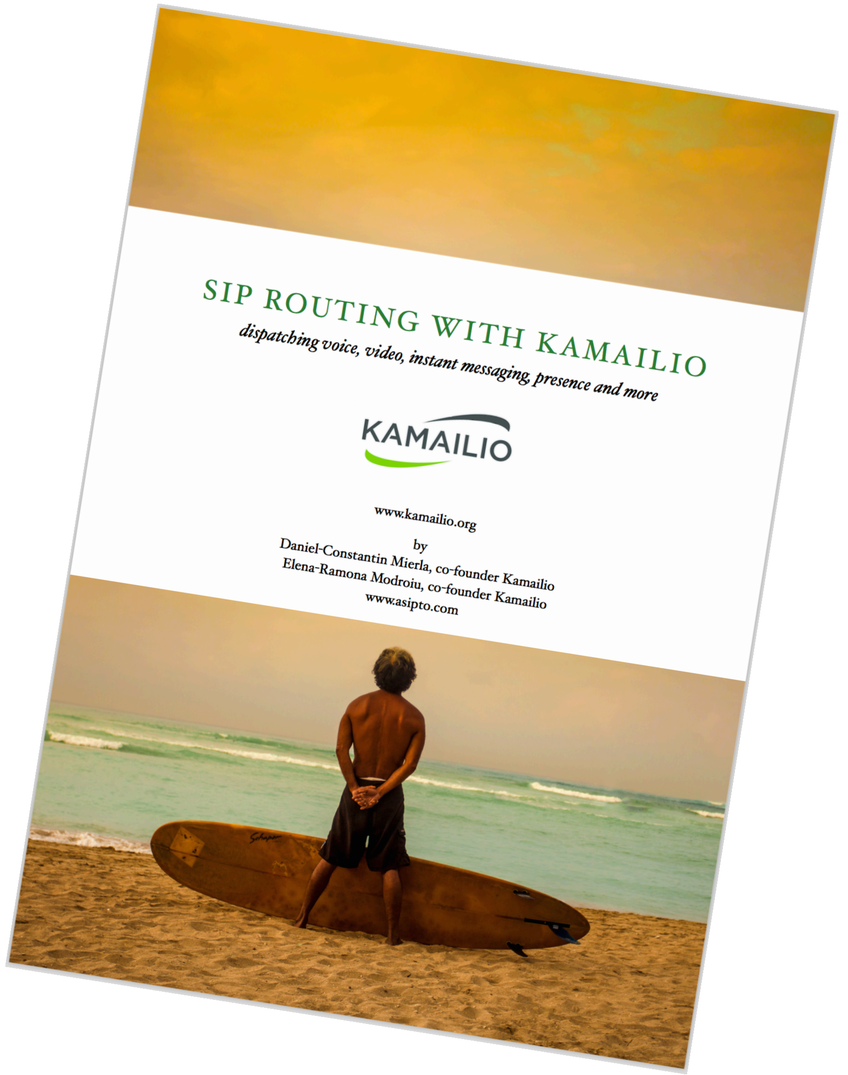
```
cd /usr/local/src/
mkdir kamailio-5.4
cd kamailio-5.4
git clone --depth 1 --no-single-branch https://github.com/kamailio/kamailio kamailio
cd kamailio
git checkout -b 5.4 origin/5.4
make include_modules="db_mysql" cfg
make all
```

START KAMAILIO FROM SOURCE CODE DIRECTORY WITH:

./src/kamailio -w . -f etc/kamailio.cfg -L src/modules/ -A WITH_NAT -A WITH_RTPENGINE -a no --alias=lab.kamailio.dev -l udp:192.168.23.166:5060/76.9.245.166:5060

KAMAILIO SYSTEM: **A CLOUD NATTED VIRTUAL MACHINE**

KAMAILIO LOCAL IP: **192.168.23.166**
KAMAILIO PUBLIC IP: **76.9.245.166**
SIP DOMAIN: **LAB.KAMAILIO.DEV**
RTP NAT TRAVERSAL: **DONE WITH RTPENGINE (HAS TO BE INSTALLED AS WELL)**



https://www.asipto.com/sw/kamailio-admin-book/

Kamailio World Updates
Online Event

September 2-3, 2020

https://www.kamailio.org/w/2020/08/kamailio-world-updates-online-celebration-party-sep-2-3-2020/

THANK YOU!

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

Anniversary Edition 2021
www.kamailioworld.com