```
#!KAMAILIO
#
# Kamailio (OpenSER) SIP Server v4.4 - default configuration script
#     - web: http://www.kamailio.org
#     - git: http://sip-router.org
#
# Direct your questions about this file to: <sr-users@lists.sip-router.org>
#
# Refer to the Core CookBook at http://www.kamailio.org/wiki/
# for an explanation of possible statements, functions and parameters.
#
# Several features can be enabled using '#!define WITH_FEATURE' directives:
#
# *** To run in debug mode:
#     - define WITH_DEBUG
#
# *** To enable mysql:
#     - define WITH_MYSQL
#
# *** To enable authentication execute:
#     - enable mysql
#     - define WITH_AUTH
#     - add users using 'kamctl'
#
# *** To enable IP authentication execute:
#     - enable mysql
#     - enable authentication
#     - define WITH_IPAUTH
#     - add IP addresses with group id '1' to 'address' table
#
# *** To enable persistent user location execute:
#     - enable mysql
#     - define WITH_USRLOCDB
#
# *** To enable presence server execute:
#     - enable mysql
#     - define WITH_PRESENCE
#
# *** To enable nat traversal execute:
#     - define WITH_NAT
#     - install RTPProxy: http://www.rtpproxy.org
#     - start RTPProxy:
#        rtpproxy -l _your_public_ip_ -s udp:localhost:7722
#     - option for NAT SIP OPTIONS keepalives: WITH_NATSIPPING
#
# *** To enable PSTN gateway routing execute:
#     - define WITH_PSTN
#     - set the value of pstn.gw_ip
#     - check route[PSTN] for regexp routing condition
#
# *** To enable database aliases lookup execute:
#     - enable mysql
#     - define WITH_ALIASDB
#
# *** To enable speed dial lookup execute:
#     - enable mysql
#     - define WITH_SPEEDDIAL
#
# *** To enable multi-domain support execute:
#     - enable mysql
#     - define WITH_MULTIDOMAIN
```

```
#
# *** To enable TLS support execute:
#     - adjust CFGDIR/tls.cfg as needed
#     - define WITH_TLS
#
# *** To enable XMLRPC support execute:
#     - define WITH_XMLRPC
#     - adjust route[XMLRPC] for access policy
#
# *** To enable anti-flood detection execute:
#     - adjust pike and htable=>ipban settings as needed (default is
#       block if more than 16 requests in 2 seconds and ban for 300 seconds)
#     - define WITH_ANTIFLOOD
#
# *** To block 3XX redirect replies execute:
#     - define WITH_BLOCK3XX
#
# *** To enable VoiceMail routing execute:
#     - define WITH_VOICEMAIL
#     - set the value of voicemail.srv_ip
#     - adjust the value of voicemail.srv_port
#
# *** To enhance accounting execute:
#     - enable mysql
#     - define WITH_ACCDB
#     - add following columns to database
#!ifdef ACCDB_COMMENT
  ALTER TABLE acc ADD COLUMN src_user VARCHAR(64) NOT NULL DEFAULT '';
  ALTER TABLE acc ADD COLUMN src_domain VARCHAR(128) NOT NULL DEFAULT '';
  ALTER TABLE acc ADD COLUMN src_ip varchar(64) NOT NULL default '';
  ALTER TABLE acc ADD COLUMN dst_ouser VARCHAR(64) NOT NULL DEFAULT '';
  ALTER TABLE acc ADD COLUMN dst_user VARCHAR(64) NOT NULL DEFAULT '';
  ALTER TABLE acc ADD COLUMN dst_domain VARCHAR(128) NOT NULL DEFAULT '';
  ALTER TABLE missed_calls ADD COLUMN src_user VARCHAR(64) NOT NULL DEFAULT '';
  ALTER TABLE missed_calls ADD COLUMN src_domain VARCHAR(128) NOT NULL DEFAULT '';
  ALTER TABLE missed_calls ADD COLUMN src_ip varchar(64) NOT NULL default '';
  ALTER TABLE missed_calls ADD COLUMN dst_ouser VARCHAR(64) NOT NULL DEFAULT '';
  ALTER TABLE missed_calls ADD COLUMN dst_user VARCHAR(64) NOT NULL DEFAULT '';
  ALTER TABLE missed_calls ADD COLUMN dst_domain VARCHAR(128) NOT NULL DEFAULT '';
#!endif

####### Include Local Config If Exists #########
import_file "kamailio-local.cfg"

####### Defined Values #########

#!define WITH_MYSQL
#!define WITH_AUTH
#!define WITH_NAT

# *** Value defines - IDs used later in config
#!ifdef WITH_MYSQL
# - database URL - used to connect to database server by modules such
#       as: auth_db, acc, usrloc, a.s.o.
#!ifndef DBURL
#!define DBURL "mysql://kamailio:kamailiorw@localhost/kamailio"
#!endif
#!endif
#!ifdef WITH_MULTIDOMAIN
# - the value for 'use_domain' parameters
#!define MULTIDOMAIN 1
```

```
#!else
#!define MULTIDOMAIN 0
#!endif

# - flags
#   FLT_ - per transaction (message) flags
#   FLB_ - per branch flags
#!define FLT_ACC 1
#!define FLT_ACCMISSED 2
#!define FLT_ACCFAILED 3
#!define FLT_NATS 5

#!define FLB_NATB 6
#!define FLB_NATSIPPING 7

####### Global Parameters #########

#!define WITH_DEBUG

### LOG Levels: 3=DBG, 2=INFO, 1=NOTICE, 0=WARN, -1=ERR
#!ifdef WITH_DEBUG
debug=4
log_stderror=yes
#!else
debug=2
log_stderror=no
#!endif

memdbg=5
memlog=5

log_facility=LOG_LOCAL0

# number of SIP routing processes
children=8

/* uncomment the next line to disable TCP (default on) */
#disable_tcp=yes

/* uncomment the next line to disable the auto discovery of local aliases
   based on reverse DNS on IPs (default on) */
#auto_aliases=no

/* add local domain aliases */
#alias="sip.mydomain.com"

/* uncomment and configure the following line if you want Kamailio to
   bind on a specific interface/port/proto (default bind on all available) */
#listen=udp:10.0.0.10:5060

/* port to listen to */
#port=5060

#!ifdef WITH_TLS
enable_tls=yes
#!endif

# life time of TCP connection when there is no traffic
# - a bit higher than registration expires to cope with UA behind NAT
tcp_connection_lifetime=3605
```

```
####### Custom Parameters #########

# These parameters can be modified runtime via RPC interface
# - see the documentation of 'cfg_rpc' module.
#
# Format: group.id = value 'desc' description
# Access: $sel(cfg_get.group.id) or @cfg_get.group.id
#

#!ifdef WITH_PSTN
# PSTN GW Routing
#
# - pstn.gw_ip: valid IP or hostname as string value, example:
# pstn.gw_ip = "10.0.0.101" desc "My PSTN GW Address"
#
# - by default is empty to avoid misrouting
pstn.gw_ip = "" desc "PSTN GW Address"
pstn.gw_port = "" desc "PSTN GW Port"
#!endif

#!ifdef WITH_VOICEMAIL
# VoiceMail Routing on offline, busy or no answer
#
# - by default Voicemail server IP is empty to avoid misrouting
voicemail.srv_ip = "" desc "VoiceMail IP Address"
voicemail.srv_port = "5060" desc "VoiceMail Port"
#!endif

####### Modules Section ########

# set paths to location of modules (to sources or installation folders)
#!ifdef WITH_SRCPATH
mpath="modules/"
#!else
mpath="/home/CloudShare/lib/kamailio/modules/"
#!endif

#!ifdef WITH_MYSQL
loadmodule "db_mysql.so"
#!endif

loadmodule "mi_fifo.so"
loadmodule "kex.so"
loadmodule "corex.so"
loadmodule "tm.so"
loadmodule "tmx.so"
loadmodule "sl.so"
loadmodule "rr.so"
loadmodule "pv.so"
loadmodule "maxfwd.so"
loadmodule "usrloc.so"
loadmodule "registrar.so"
loadmodule "textops.so"
loadmodule "siputils.so"
loadmodule "xlog.so"
loadmodule "sanity.so"
loadmodule "ctl.so"
loadmodule "cfg_rpc.so"
loadmodule "mi_rpc.so"
loadmodule "acc.so"
```

```
#!ifdef WITH_AUTH
loadmodule "auth.so"
loadmodule "auth_db.so"
#!ifdef WITH_IPAUTH
loadmodule "permissions.so"
#!endif
#!endif

#!ifdef WITH_ALIASDB
loadmodule "alias_db.so"
#!endif

#!ifdef WITH_SPEEDDIAL
loadmodule "speeddial.so"
#!endif

#!ifdef WITH_MULTIDOMAIN
loadmodule "domain.so"
#!endif

#!ifdef WITH_PRESENCE
loadmodule "presence.so"
loadmodule "presence_xml.so"
#!endif

#!ifdef WITH_NAT
loadmodule "nathelper.so"
loadmodule "rtpproxy.so"
#!endif

#!ifdef WITH_TLS
loadmodule "tls.so"
#!endif

#!ifdef WITH_ANTIFLOOD
loadmodule "htable.so"
loadmodule "pike.so"
#!endif

#!ifdef WITH_XMLRPC
loadmodule "xmlrpc.so"
#!endif

#!ifdef WITH_DEBUG
loadmodule "debugger.so"
#!endif

# ----------------- setting module-specific parameters ---------------


# ----- mi_fifo params -----
#modparam("mi_fifo", "fifo_name", "/var/run/kamailio/kamailio_fifo")

# ----- ctl params -----
#modparam("ctl", "binrpc", "unix:/var/run/kamailio/kamailio_ctl")

# ----- tm params -----
# auto-discard branches from previous serial forking leg
modparam("tm", "failure_reply_mode", 3)
# default retransmission timeout: 30sec
modparam("tm", "fr_timer", 30000)
```

```
# default invite retransmission timeout after 1xx: 120sec
modparam("tm", "fr_inv_timer", 120000)


# ----- rr params -----
# set next param to 1 to add value to ;lr param (helps with some UAs)
modparam("rr", "enable_full_lr", 0)
# do not append from tag to the RR (no need for this script)
modparam("rr", "append_fromtag", 0)


# ----- registrar params -----
modparam("registrar", "method_filtering", 1)
/* uncomment the next line to disable parallel forking via location */
# modparam("registrar", "append_branches", 0)
/* uncomment the next line not to allow more than 10 contacts per AOR */
#modparam("registrar", "max_contacts", 10)
# max value for expires of registrations
modparam("registrar", "max_expires", 3600)
# set it to 1 to enable GRUU
modparam("registrar", "gruu_enabled", 0)


# ----- acc params -----
/* what special events should be accounted ? */
modparam("acc", "early_media", 0)
modparam("acc", "report_ack", 0)
modparam("acc", "report_cancels", 0)
/* by default ww do not adjust the direct of the sequential requests.
   if you enable this parameter, be sure the enable "append_fromtag"
   in "rr" module */
modparam("acc", "detect_direction", 0)
/* account triggers (flags) */
modparam("acc", "log_flag", FLT_ACC)
modparam("acc", "log_missed_flag", FLT_ACCMISSED)
modparam("acc", "log_extra",
    "src_user=$fU;src_domain=$fd;src_ip=$si;"
    "dst_ouser=$tU;dst_user=$rU;dst_domain=$rd")
modparam("acc", "failed_transaction_flag", FLT_ACCFAILED)
/* enhanced DB accounting */
#!ifdef WITH_ACCDB
modparam("acc", "db_flag", FLT_ACC)
modparam("acc", "db_missed_flag", FLT_ACCMISSED)
modparam("acc", "db_url", DBURL)
modparam("acc", "db_extra",
    "src_user=$fU;src_domain=$fd;src_ip=$si;"
    "dst_ouser=$tU;dst_user=$rU;dst_domain=$rd")
#!endif


# ----- usrloc params -----
/* enable DB persistency for location entries */
#!ifdef WITH_USRLOCDB
modparam("usrloc", "db_url", DBURL)
modparam("usrloc", "db_mode", 2)
modparam("usrloc", "use_domain", MULTIDOMAIN)
#!endif


# ----- auth_db params -----
#!ifdef WITH_AUTH
```

```
modparam("auth_db", "db_url", DBURL)
modparam("auth_db", "calculate_ha1", yes)
modparam("auth_db", "password_column", "password")
modparam("auth_db", "load_credentials", "")
modparam("auth_db", "use_domain", MULTIDOMAIN)

# ----- permissions params -----
#!ifdef WITH_IPAUTH
modparam("permissions", "db_url", DBURL)
modparam("permissions", "db_mode", 1)
#!endif

#!endif


# ----- alias_db params -----
#!ifdef WITH_ALIASDB
modparam("alias_db", "db_url", DBURL)
modparam("alias_db", "use_domain", MULTIDOMAIN)
#!endif


# ----- speeddial params -----
#!ifdef WITH_SPEEDDIAL
modparam("speeddial", "db_url", DBURL)
modparam("speeddial", "use_domain", MULTIDOMAIN)
#!endif


# ----- domain params -----
#!ifdef WITH_MULTIDOMAIN
modparam("domain", "db_url", DBURL)
# register callback to match myself condition with domains list
modparam("domain", "register_myself", 1)
#!endif


#!ifdef WITH_PRESENCE
# ----- presence params -----
modparam("presence", "db_url", DBURL)

# ----- presence_xml params -----
modparam("presence_xml", "db_url", DBURL)
modparam("presence_xml", "force_active", 1)
#!endif


#!ifdef WITH_NAT
# ----- rtpproxy params -----
modparam("rtpproxy", "rtpproxy_sock", "udp:127.0.0.1:7722")

# ----- nathelper params -----
modparam("nathelper", "natping_interval", 30)
modparam("nathelper", "ping_nated_only", 1)
modparam("nathelper", "sipping_bflag", FLB_NATSIPPING)
modparam("nathelper", "sipping_from", "sip:pinger@kamailio.org")

# params needed for NAT traversal in other modules
modparam("nathelper|registrar", "received_avp", "$avp(RECEIVED)")
modparam("usrloc", "nat_bflag", FLB_NATB)
#!endif
```

```
#!ifdef WITH_TLS
# ----- tls params -----
modparam("tls", "config", "/home/CloudShare/etc/kamailio/tls.cfg")
#!endif

#!ifdef WITH_ANTIFLOOD
# ----- pike params -----
modparam("pike", "sampling_time_unit", 2)
modparam("pike", "reqs_density_per_unit", 16)
modparam("pike", "remove_latency", 4)

# ----- htable params -----
# ip ban htable with autoexpire after 5 minutes
modparam("htable", "htable", "ipban=>size=8;autoexpire=300;")
#!endif

#!ifdef WITH_XMLRPC
# ----- xmlrpc params -----
modparam("xmlrpc", "route", "XMLRPC");
modparam("xmlrpc", "url_match", "^/RPC")
#!endif

#!ifdef WITH_DEBUG
# ----- debugger params -----
modparam("debugger", "cfgtrace", 1)
modparam("debugger", "log_level_name", "exec")
#!endif

####### Routing Logic ########


# Main SIP request routing logic
# - processing of any incoming SIP request starts with this route
# - note: this is the same as route { ... }
request_route {

    # per request initial checks
    route(REQINIT);

    # NAT detection
    route(NATDETECT);

    # CANCEL processing
    if (is_method("CANCEL")) {
        if (t_check_trans()) {
            route(RELAY);
        }
        exit;
    }

    # handle requests within SIP dialogs
    route(WITHINDLG);

    ### only initial requests (no To tag)

    # handle retransmissions
    if(t_precheck_trans()) {
        t_check_trans();
        exit;
```

```
    }
    t_check_trans();

    # authentication
    route(AUTH);

    # record routing for dialog forming requests (in case they are routed)
    # - remove preloaded route headers
    remove_hf("Route");
    if (is_method("INVITE|SUBSCRIBE")) {
        record_route();
    }

    # account only INVITEs
    if (is_method("INVITE")) {
        setflag(FLT_ACC); # do accounting
    }

    # dispatch requests to foreign domains
    route(SIPOUT);

    ### requests for my local domains

    # handle presence related requests
    route(PRESENCE);

    # handle registrations
    route(REGISTRAR);

    if ($rU==$null) {
        # request with no Username in RURI
        sl_send_reply("484","Address Incomplete");
        exit;
    }

    # dispatch destinations to PSTN
    route(PSTN);

    # user location service
    route(LOCATION);
}

# Wrapper for relaying requests
route[RELAY] {

    # enable additional event routes for forwarded requests
    # - serial forking, RTP relaying handling, a.s.o.
    if (is_method("INVITE|BYE|SUBSCRIBE|UPDATE")) {
        if(!t_is_set("branch_route")) t_on_branch("MANAGE_BRANCH");
    }
    if (is_method("INVITE|SUBSCRIBE|UPDATE")) {
        if(!t_is_set("onreply_route")) t_on_reply("MANAGE_REPLY");
    }
    if (is_method("INVITE")) {
        if(!t_is_set("failure_route")) t_on_failure("MANAGE_FAILURE");
    }

    if (!t_relay()) {
        sl_reply_error();
    }
    exit;
```

```
}

# Per SIP request initial checks
route[REQINIT] {
#!ifdef WITH_ANTIFLOOD
    # flood detection from same IP and traffic ban for a while
    # be sure you exclude checking trusted peers, such as pstn gateways
    # - local host excluded (e.g., loop to self)
    if(src_ip!=myself) {
        if($sht(ipban=>$si)!=$null) {
            # ip is already blocked
            xdbg("request from blocked IP - $rm from $fu (IP:$si:$sp)\n");
            exit;
        }
        if (!pike_check_req()) {
            xlog("L_ALERT","ALERT: pike blocking $rm from $fu (IP:$si:$sp)\n");
            $sht(ipban=>$si) = 1;
            exit;
        }
    }
    if($ua =~ "friendly-scanner|sipcli") {
        # silent drop for scanners - uncomment next line if want to reply
        # sl_send_reply("200", "OK");
        exit;
    }
#!endif

    if (!mf_process_maxfwd_header("10")) {
        sl_send_reply("483","Too Many Hops");
        exit;
    }

    if(is_method("OPTIONS") && uri==myself && $rU==$null) {
        sl_send_reply("200","Keepalive");
        exit;
    }

    if(!sanity_check("1511", "7")) {
        xlog("Malformed SIP message from $si:$sp\n");
        exit;
    }
}

# Handle requests within SIP dialogs
route[WITHINDLG] {
    if (!has_totag()) return;

    # sequential request withing a dialog should
    # take the path determined by record-routing
    if (loose_route()) {
        route(DLGURI);
        if (is_method("BYE")) {
            setflag(FLT_ACC); # do accounting ...
            setflag(FLT_ACCFAILED); # ... even if the transaction fails
        } else if ( is_method("ACK") ) {
            # ACK is forwarded statelessy
            route(NATMANAGE);
        } else if ( is_method("NOTIFY") ) {
            # Add Record-Route for in-dialog NOTIFY as per RFC 6665.
            record_route();
        }
```

```
        route(RELAY);
        exit;
    }

    if (is_method("SUBSCRIBE") && uri == myself) {
        # in-dialog subscribe requests
        route(PRESENCE);
        exit;
    }
    if ( is_method("ACK") ) {
        if ( t_check_trans() ) {
            # no loose-route, but stateful ACK;
            # must be an ACK after a 487
            # or e.g. 404 from upstream server
            route(RELAY);
            exit;
        } else {
            # ACK without matching transaction ... ignore and discard
            exit;
        }
    }
    sl_send_reply("404","Not here");
    exit;
}

# Handle SIP registrations
route[REGISTRAR] {
    if (!is_method("REGISTER")) return;

    if(isflagset(FLT_NATS)) {
        setbflag(FLB_NATB);
#!ifdef WITH_NATSIPPING
        # do SIP NAT pinging
        setbflag(FLB_NATSIPPING);
#!endif
    }
    if (!save("location")) {
        sl_reply_error();
    }
    exit;
}

# User location service
route[LOCATION] {

#!ifdef WITH_SPEEDDIAL
    # search for short dialing - 2-digit extension
    if($rU=~"^[0-9][0-9]$") {
        if(sd_lookup("speed_dial")) {
            route(SIPOUT);
        }
    }
#!endif

#!ifdef WITH_ALIASDB
    # search in DB-based aliases
    if(alias_db_lookup("dbaliases")) {
        route(SIPOUT);
    }
#!endif
```

```
    $avp(oexten) = $rU;
    if (!lookup("location")) {
        $var(rc) = $rc;
        route(TOVOICEMAIL);
        t_newtran();
        switch ($var(rc)) {
            case -1:
            case -3:
                send_reply("404", "Not Found");
                exit;
            case -2:
                send_reply("405", "Method Not Allowed");
                exit;
        }
    }

    # when routing via usrloc, log the missed calls also
    if (is_method("INVITE")) {
        setflag(FLT_ACCMISSED);
    }

    route(RELAY);
    exit;
}

# Presence server processing
route[PRESENCE] {
    if(!is_method("PUBLISH|SUBSCRIBE")) return;

    if(is_method("SUBSCRIBE") && $hdr(Event)=="message-summary") {
        route(TOVOICEMAIL);
        # returns here if no voicemail server is configured
        sl_send_reply("404", "No voicemail service");
        exit;
    }

#!ifdef WITH_PRESENCE
    if (!t_newtran()) {
        sl_reply_error();
        exit;
    }

    if(is_method("PUBLISH")) {
        handle_publish();
        t_release();
    } else if(is_method("SUBSCRIBE")) {
        handle_subscribe();
        t_release();
    }
    exit;
#!endif

    # if presence enabled, this part will not be executed
    if (is_method("PUBLISH") || $rU==$null) {
        sl_send_reply("404", "Not here");
        exit;
    }
    return;
}

# IP authorization and user authentication
```

```
route[AUTH] {
#!ifdef WITH_AUTH

#!ifdef WITH_IPAUTH
    if((!is_method("REGISTER")) && allow_source_address()) {
        # source IP allowed
        return;
    }
#!endif

    if (is_method("REGISTER") || from_uri==myself) {
        # authenticate requests
        if (!auth_check("$fd", "subscriber", "1")) {
            auth_challenge("$fd", "0");
            exit;
        }
        # user authenticated - remove auth header
        if(!is_method("REGISTER|PUBLISH"))
            consume_credentials();
    }
    # if caller is not local subscriber, then check if it calls
    # a local destination, otherwise deny, not an open relay here
    if (from_uri!=myself && uri!=myself) {
        sl_send_reply("403","Not relaying");
        exit;
    }

#!endif
    return;
}

# Caller NAT detection
route[NATDETECT] {
#!ifdef WITH_NAT
    force_rport();
    if (nat_uac_test("19")) {
        if (is_method("REGISTER")) {
            fix_nated_register();
        } else {
            if(is_first_hop()) {
                set_contact_alias();
            }
        }
        setflag(FLT_NATS);
    }
#!endif
    return;
}

# RTPProxy control and signaling updates for NAT traversal
route[NATMANAGE] {
#!ifdef WITH_NAT
    if (is_request()) {
        if(has_totag()) {
            if(check_route_param("nat=yes")) {
                setbflag(FLB_NATB);
            }
        }
    }
    if (!(isflagset(FLT_NATS) || isbflagset(FLB_NATB))) return;
```

```
    if(nat_uac_test("8")) {
        rtpproxy_manage("co");
    } else {
        rtpproxy_manage("cor");
    }

    if (is_request()) {
        if (!has_totag()) {
            if(t_is_branch_route()) {
                add_rr_param(";nat=yes");
            }
        }
    }
    if (is_reply()) {
        if(isbflagset(FLB_NATB)) {
            if(is_first_hop())
                set_contact_alias();
        }
    }
#!endif
    return;
}

# URI update for dialog requests
route[DLGURI] {
#!ifdef WITH_NAT
    if(!isdsturiset()) {
        handle_ruri_alias();
    }
#!endif
    return;
}

# Routing to foreign domains
route[SIPOUT] {
    if (uri==myself) return;

    append_hf("P-hint: outbound\r\n");
    route(RELAY);
    exit;
}

# PSTN GW routing
route[PSTN] {
#!ifdef WITH_PSTN
    # check if PSTN GW IP is defined
    if (strempty($sel(cfg_get.pstn.gw_ip))) {
        xlog("SCRIPT: PSTN routing enabled but pstn.gw_ip not defined\n");
        return;
    }

    # route to PSTN dialed numbers starting with '+' or '00'
    #     (international format)
    # - update the condition to match your dialing rules for PSTN routing
    if(!($rU=~"^(\+|00)[1-9][0-9]{3,20}$")) return;

    # only local users allowed to call
    if(from_uri!=myself) {
        sl_send_reply("403", "Not Allowed");
        exit;
    }
```

```
    if (strempty($sel(cfg_get.pstn.gw_port))) {
        $ru = "sip:" + $rU + "@" + $sel(cfg_get.pstn.gw_ip);
    } else {
        $ru = "sip:" + $rU + "@" + $sel(cfg_get.pstn.gw_ip) + ":"
                    + $sel(cfg_get.pstn.gw_port);
    }

    route(RELAY);
    exit;
#!endif

    return;
}

# XMLRPC routing
#!ifdef WITH_XMLRPC
route[XMLRPC] {
    # allow XMLRPC from localhost
    if ((method=="POST" || method=="GET")
            && (src_ip==127.0.0.1)) {
        # close connection only for xmlrpclib user agents (there is a bug in
        # xmlrpclib: it waits for EOF before interpreting the response).
        if ($hdr(User-Agent) =~ "xmlrpclib")
            set_reply_close();
        set_reply_no_connect();
        dispatch_rpc();
        exit;
    }
    send_reply("403", "Forbidden");
    exit;
}
#!endif

# Routing to voicemail server
route[TOVOICEMAIL] {
#!ifdef WITH_VOICEMAIL
    if(!is_method("INVITE|SUBSCRIBE")) return;

    # check if VoiceMail server IP is defined
    if (strempty($sel(cfg_get.voicemail.srv_ip))) {
        xlog("SCRIPT: VoiceMail routing enabled but IP not defined\n");
        return;
    }
    if(is_method("INVITE")) {
        if($avp(oexten)==$null) return;

        $ru = "sip:" + $avp(oexten) + "@" + $sel(cfg_get.voicemail.srv_ip)
                + ":" + $sel(cfg_get.voicemail.srv_port);
    } else {
        if($rU==$null) return;

        $ru = "sip:" + $rU + "@" + $sel(cfg_get.voicemail.srv_ip)
                + ":" + $sel(cfg_get.voicemail.srv_port);
    }
    route(RELAY);
    exit;
#!endif

    return;
}
```

```
# Manage outgoing branches
branch_route[MANAGE_BRANCH] {
    xdbg("new branch [$T_branch_idx] to $ru\n");
    route(NATMANAGE);
}

# Manage incoming replies
onreply_route[MANAGE_REPLY] {
    xdbg("incoming reply\n");
    if(status=~"[12][0-9][0-9]") {
        route(NATMANAGE);
    }
}

# Manage failure routing cases
failure_route[MANAGE_FAILURE] {
    route(NATMANAGE);

    if (t_is_canceled()) exit;

#!ifdef WITH_BLOCK3XX
    # block call redirect based on 3xx replies.
    if (t_check_status("3[0-9][0-9]")) {
        t_reply("404","Not found");
        exit;
    }
#!endif

#!ifdef WITH_VOICEMAIL
    # serial forking
    # - route to voicemail on busy or no answer (timeout)
    if (t_check_status("486|408")) {
        $du = $null;
        route(TOVOICEMAIL);
        exit;
    }
#!endif
}
```