Configuración básica de OpenSER

A continuación se muestra un ejemplo comentado de configuración básica de OpenSER, el que se pueden realizar llamadas de dispositivo a dispositivo, estando la configuración almacenada en una Basde de Datos de MySQL.

Fichero: **openser.cfg**

```
debug=3
fork=no
log_stderror=yes

listen=udp:10.68.42.134
port=5060
children=4
dns=no
rev_dns=no

mpath="/lib/openser/modules/"

loadmodule "mysql.so"
loadmodule "sl.so"
loadmodule "tm.so"
loadmodule "rr.so"
loadmodule "maxfwd.so"
loadmodule "usrloc.so"
loadmodule "registrar.so"
loadmodule "mi_fifo.so"
loadmodule "textops.so"
loadmodule "xlog.so"
loadmodule "auth.so"
loadmodule "auth_db.so"
loadmodule "uri.so"
loadmodule "uri_db.so"
loadmodule "domain.so"

modparam("mi_fifo", "fifo_name", "/tmp/openser_fifo")
modparam("auth_db|uri_db|usrloc", "db_url", "mysql://openser:openserrw@localhost/openser")
modparam("auth_db", "calculate_ha1", no)
modparam("auth_db", "password_column", "ha1")
modparam("auth_db", "password_column_2", "ha1b")

modparam("usrloc", "db_mode", 2)
modparam("rr", "enable_full_lr", 1)

## Tiempo para la llamada
modparam("tm", "fr_inv_timer", 45)

modparam("domain", "db_url", "mysql://openser:openserrw@localhost/openser")
modparam("domain", "db_mode", 1)   ## Habilitamos la cache se la tabla domain
```

```
# ------------------------------------------------------------------
# Main routing logic
# ------------------------------------------------------------------

route{

      # ------------------------------------------------------------------
      # LOG section
      # ------------------------------------------------------------------
      ## LOG del mensaje recibido

      xlog("L_INFO","\n\n$Cbg[  $mi --  $rm  --  $ua ($si:$sp)  --  FROM: $fu  --  TO: $tU  --
]$Cxx\n");

      # ------------------------------------------------------------------
      # Sanity Check Section
      # ------------------------------------------------------------------
      ## Comprobamos que el mensaje no sea demasiado largo ni halla superado max_forwards


      if(!mf_process_maxfwd_header("10"))
      {
            sl_send_reply("483","Too Many Hops");
            xlog("L_ERROR","\n\n$CwrToo Many Hops$Cxx\n");
            exit;
      };
      if(msg:len > max_len) {
            sl_send_reply("513","Message Overflow");
            xlog("L_ERROR","\n\n$CwrMessage Overflow$Cxx\n");
            exit;
      }


      # ------------------------------------------------------------------
      # Record Route Section
      # ------------------------------------------------------------------
      ## Nos ponemos en medio, poniendo una cabecera Record-Route, para que los
      ##mensajes pasen por nosotros.
      ## Pero si es REGISTER no se debe hacer.

      if (method!="REGISTER") {
            record_route();
      };


      # ------------------------------------------------------------------
      # Loose Route Section
      # ------------------------------------------------------------------
      ## Los mensajes pertenecientes a un mismo dialogo deben tomar el camino
      ##indicado por Record-Route

      if (loose_route()) {
            xlog("L_INFO","\n\n *** Estamos en loose_route() ***\n\n");
            route(1);
            exit;
      };


      # ------------------------------------------------------------------
      # Call Type Processing Section
      # ------------------------------------------------------------------
      ## Mensajes con destino distinto de nuestro servidor, hacemos relay, pero .
      ## Ruta de salientes

      if (!is_uri_host_local()) {
            if (is_from_local()) {
```

```
                route(4);
        }
        else {
                sl_send_reply("403", "Forbidden");
        };
        exit;
    }

    ## Mensajes con destino nuestro servidor (por ejemplo llamadas a
    ##usuarios registrados aquí
    ## Tenemos que tratar explicitamente el ACK
    if (method=="ACK") {
        xlog("L_INFO","$Cbx--- Procesando un ACK *not within a dialog*$Cxx\n");
        route(1);
        exit;
    }

    ## CANCEL messages can be safely processed with a simple call to t_relay()
    ##because SER will automatically match the CANCEL message to the original
    ##INVITE message (stateful).
    ## Para los CANCEL, con la ruta por defecto vale.
    else if (method=="CANCEL") {
        route(1);
        exit;
    }

    ## Los REGISTER los tratamos a parte (en la ruta 2)
    else if (method=="REGISTER") {
        route(2);
        exit;
    }

    ## Los INVITEs a la ruta 3 (autenticacion,...)
    else if (method=="INVITE") {
        route(3);
        exit;
    }

    ## Resto de casos (OPTIONS, REFER, BYE...)
    else {
        # Puede que venga a nosotros pero tengamos definido un alias a
        ##fuera. lookup("aliases") nos da la nueva URI que puede sea !=myself.
        lookup("aliases");
        if (!is_uri_host_local()) {
            xlog("L_INFO","$CrxNot my URI after the alias lookup$Cxx\n");
            ## A las salientes
            route(4);
            exit;
        };

        ## Miramos si existe el destino en nuestra tabla "location".
        if (!lookup("location")) {
            xlog("L_INFO","$Crx404 User Not Found$Cxx\n");
            sl_send_reply("404", "Not Found");
            exit;
        };

        ## Si hemos llegado hasta aquí enrutamos el mensaje al destino por la
        ##ruta por defecto.
        route(1);
        exit;
     };
}
```

```
# ----------------------------------------------------------------
# Default message handler
# ----------------------------------------------------------------

route[1]
{
      ## Indicamos que las respuestas que se originen aqui vayan a la ruta
      ##onreply_route[1], así sabemos lo que pasa
      t_on_reply("1");

      if(!t_relay()) {
            sl_reply_error();
      };
      xlog("L_INFO","$CbxMessage is relayed; now exiting$Cxx\n");
      exit;
}

# ----------------------------------------------------------------
# REGISTER message handler
# ----------------------------------------------------------------

route[2]
{
      sl_send_reply("100", "Trying");
      if (!www_authorize("","subscriber")) {
            xlog("L_INFO","$CbxSe necesita autenticacion para el REGISTER$Cxx\n");
            www_challenge("","0");
            exit;
      }
      else if (!check_to()) {
            ## Validate the supplied To: header against the previously
            ##validated digest credentials. If they do not match then we must
            ##reject the REGISTER.
            xlog("L_INFO","$Crx*** check_to() = NO!! ***$Cxx\n");
            sl_send_reply("401", "Unauthorized");
            exit;
      };
      xlog("L_INFO","$Cbx*** REGISTER correcto ***$Cxx\n");

      ## Eliminamos las cabeceras relativas a la autenticacion, porque ya no son
      ##necesarias y no vamos a ir mandandolas por ahi...
      consume_credentials();

      ## Informo si es un UNREGISTER (RFC3261 -- 10.2.2)
      if ($hdr(contact)=~";expires=0") || ($hdr(expires)=="0") {
            xlog("L_INFO","$Cbx*** UNREGISTER ***$Cxx\n");
      }

      ## Guardamos la localización en la tabla "location".
      if (!save("location")) {
            sl_reply_error();
      };
}
```

```
# ------------------------------------------------------------------
# INVITE Message Handler
# ------------------------------------------------------------------

route[3] {

      ## Es necesario autenticarse para poder llamar
      if (!proxy_authorize("","subscriber")) {
            xlog("L_INFO","$CbxSe necesita autenticacion para el INVITE$Cxx\n");
            proxy_challenge("","0");
            exit;
      }

      ## Tienen que coincidir el nombre de usuario con el de la cabecera FROM
      else if (!check_from()) {
            xlog("L_INFO","$Crx*** check_from() = NO!! ***$Cxx\n");
            sl_send_reply("403", "Use From=ID");
            exit;
      };
      xlog("L_INFO","$Cbx*** INVITE correcto ***$Cxx\n");
      consume_credentials();

      # Puede que venga a nosotros pero tengamos definido un alias a fuera.
      #lookup("aliases") nos da la nueva URI que puede sea !=myself.
      lookup("aliases");
      if (!is_uri_host_local()) {
            ## A las salientes
            route(4);
            exit;
      };

      if (!lookup("location")) {
            xlog("L_INFO","$Crx404 User Not Found$Cxx\n");
            sl_send_reply("404", "User Not Found");
            exit;
      };

      ## El usuario se ha autenticado y a quien llama existe en "location" así que lo rutamos.
      route(1);

}

# ------------------------------------------------------------------
# Outgoing
# ------------------------------------------------------------------

route[4]
{
      xlog("L_INFO","$Cbx*** Llamada saliente ***$Cxx\n");
      route(1);
      exit;
}

# ------------------------------------------------------------------
# onreply_route[1] -- Para ver las respuestas a los INVITE.
# ------------------------------------------------------------------
## Si un usuario hace un INVITE las respuestas del llamado (Trying, Ringing, OK...) pasan por
aquí.

onreply_route[1]
{
      xlog("L_INFO","\n\n$Cbc[Respuesta][  $rs ($rr) desde $si:$sp -- Peticion: ($rm)  ]
$Cxx\n");
}
```

Fichero: **openserctlrc**

```
## your SIP domain
SIP_DOMAIN=tu_dominio_o_ip

## database type: MYSQL or PGSQL, by defaulte none is loaded
DBENGINE=MYSQL

## database host
DBHOST=localhost

## database name
DBNAME=openser

## database read/write user
DBRWUSER=openser

## database read only user
DBROUSER=openserro

## password for database read only user
DBROPW=openserro

## database super user
DBROOTUSER="root"

## type of aliases used: DB - database aliases; UL - usrloc aliases
## - default: none
ALIASES_TYPE="DB"

## control engine: FIFO or UNIXSOCK
## - default FIFO
CTLENGINE="FIFO"

## path to FIFO file
#OSER_FIFO="/tmp/openser_fifo"

## check ACL names; default on (1); off (0)
# VERIFY_ACL=1

## ACL names - if VERIFY_ACL is set, only the ACL names from below list
## are accepted
# ACL_GROUPS="local ld int voicemail free-pstn"

## presence of serweb tables - default "no"
# HAS_SERWEB="yes"

## verbose - debug purposes - default '0'
# VERBOSE=1

## do (1) or don't (0) store plaintext passwords
## in the subscriber table - default '1'
STORE_PLAINTEXT_PW=0
```

El ejemplo mostrado es funcional, para utilizarlo basta con tener OpenSER instalado con soporte MySQL y seguir los siguientes pasos:

1. Instalar OpenSER con soporte MySQL: http://www.saghul.net/blog/2007/08/13/howto-compilar-openser-con-soporte-para-mysql/

2. Crear las bases de datos, ejecutando:

   #openser_mysql.sh

3. Crear los ficheros de ejemplo arriba expuestos en /etc/openser

4. Añadir el dominio local (o la IP del servidor) a la tabla 'domain'.

5. Agregar cuentas de usuario:

   #openserctl add 200 1234 saghul@gmail.com

   (Nos añade el usuario 200 con la contraseña 1234)

6. Probar!

NOTA: Para entender bien el ejemplo conviene estudiar el funcionamiento de OpenSER, para ello, estos recursos pueden resultar de interés:
http://www.saghul.net/blog/2007/08/14/recursos-para-aprender-sip-y-openser/