Asterisk

Configuración Avanzada

Discado automático

- Archivos .call
  - Se utilizan para iniciar llamadas desde una aplicación externa.
  - Son archivos de texto que al copiarse en el directorio /var/spool/asterisk/outgoing, Asterisk notará su presencia e inmediatamente activará la extensión en la prioridad especificada en el archivo .call.
  - Generalmente, se combinan con el programador de tareas de Linux: el cron.
  - Algunos ejemplos de uso son: soluciones de callback, despertador telefónico, anuncios automáticos.

Discado automático: ejemplo

- En el siguiente ejemplo: se llama al 43424444 en el canal Zap/1Dahdi/1 y envía la llamada al contexto [mensajes-salida], extensión 84, Prioridad 1:
  - Archivo mensaje.call

```
Channel: Zap/1Dahdi/1/43424444
MaxRetries: 2
RetryTime: 60
WaitTime: 30
Context: mensajes-salida
Extension: 84
Priority: 1
```

  - extensions.conf

```
[mensajes-salida]
exten => 84,1,Playback(anuncio)
exten => 84,2,Playback(vm-goodbye)
exten => 84,3,Hangup
```

Manejo de colas

- El sistema de colas en Asterisk se compone de:
  - Llamadas entrantes que son ubicadas en una cola.
  - Miembros que contestan las llamadas en la cola (extensiones o bien usuarios que se loguean como agentes).
  - Una estrategia sobre cómo manejar la cola y repartir las llamadas entre los miembros
  - Música que se reproduce durante la espera en la cola.
  - Anuncios para miembros y emisores de llamadas.

Manejo de colas: Config

- La configuración de las colas se define:
  - Estáticamente: en el archivo queues.conf
  - Dinámicamente: la configuración se almacena en una BD, “disponibilizando” los cambios sin necesidad de realizar un reload.
- La configuración de los agentes se define en el archivo agents.conf.

Manejo de colas: agentes

- Los agentes atienden las llamadas de una cola específica.
- Un agente debe realizar un **login** (llamando a una extensión especial que contiene la aplicación **AgentLogin**) indicando que está listo para tomar llamadas.
- Los miembros son aquellos canales disponibles que están activamente atendiendo la cola. Pueden ser tanto agentes como también canales regulares (sip/juan).

Manejo de colas: estrategias

- Las llamadas son distribuidas entre los miembros de una cola siguiendo alguna de las siguientes estrategias:
  - **ringall:** hace sonar todos los canales disponibles hasta que alguno responda (configuración por defecto).
  - **roundrobin:** hace sonar cada interfaz disponible por turnos.
  - **leastrecent:** hace sonar la interfaz que fue menos recientemente llamada por esta cola.
  - **fewestcalls:** hace sonar la interfaz con la menor cantidad de llamadas completas.
  - **random:** hace sonar una interfaz al azar.
  - **rrmemory:** igual que el round robin pero recuerda cual fue el último teléfono que atendió una llamada y continúa con el siguiente.

Manejo de colas: penalties y priorities

- **Penalty:** se le asigna una penalidad a cada agente, de manera tal que primero se derivan las llamadas (vía la estrategia definida) a los agentes con el menor valor de penalidad. En el caso de estar todos ocupados, se continúa con la siguiente penalty y así sucesivamente.
  - Ejemplo: sólo si la telefonista está ocupada, se deriva el llamado a la oficina.
- **Priority**: se le asigna una prioridad a cada llamada entrante, permitiendo situarla en un lugar más adelante de la cola (no siempre al final).
  - Ejemplo: las llamadas al nro. 110 tienen más prioridad que las del 111.

Manejo de colas: aplicaciones

- Aplicaciones principales, utilizadas en extensions.conf:
  - **Queue:** aplicación utilizada para encolar una llamada (toma como parámetro las colas definidas en queue.conf).
  - **AddQueueMember:** agrega dinámicamente un miembro a la cola.
  - **RemoveQueueMember:** remueve dinámicamente un miembro de la cola.
  - **AgentLogin:** login de un agente a una cola.

Manejo de colas: CLI

- Comandos relacionados de la CLI:
  - **show agents**: muestra los agentes .
  - **show queues**: lista todas las colas.
  - **show queue**: muestra datos de una cola en particular.
  - **queue add member**: agrega un miembro a la cola.
  - **queue remove member**: elimina un miembro de la cola.

Manejo de colas: ejemplo

- queue.conf

```
[MyQueue] ; nombre de la cola
music=default ; musica que coloca antes que lo atiendan
strategy=ringall ; estrategia de ring
timeout=15 ; tiempo maximo de ring a un miembro
retry=5 ; tiempo de espera hasta reintentar con otro miembro
wrapuptime=0 ; tiempo de espera luego de intentar con todos los miembros
maxlen = 0 ;
announce-frequency = 0 ; cantidad de veces que sale el anuncio
announce-holdtime = no ;  si dice el tpo de espera aproximado
member => Agent/1001,1
member => Agent/1002,1
member => Agent/1003,1
```

Manejo de colas: ejemplo

- agents.conf

```
[agents]
agent => 1001,1111,Juan
agent => 1002,2222,Pedro
agent => 1003,3333,Pablo
```

- extensions.conf

```
exten => 2020,1,Answer
exten => 2020,3,Wait(2)
exten => 2020,4,SetMusicOnHold(default)
exten => 2020,5,Queue(MyQueue|r)
exten => 2020,6,Hangup
```

ENUM

- ENUM proviene de tElephone Number Mapping.
- Se asocian los números telefónicos convencionales (E.164) a nombre DNS en *.e164.arpa*.
  - +34 944012345 → 5.4.3.2.1.0.4.4.9.4.3. e164.arpa
- El servidor DNS que alberga la entrada, tiene registrados los servicios (sip, mail, http) publicados para dicho número.
  - Ejemplo de entrada en Bind:

```
$ORIGIN 5.4.3.2.1.0.4.4.9.4.3.e164.arpa.
NAPTR 10 100 "u" "E2U+sip" "!^.*$!sip:fulano@foo.com!" .
NAPTR 10 101 "u" "E2U+msg" "!^.*$!mailto:fulano@foo.com!" .
```

- Prioridad 1: contactar vía SIP con fulano@foo.com
- Prioridad 2: contactar por correo con fulano@foo.com

ENUM

- Ejemplo:
  - Llamada desde un teléfono IP al +34944991234
    1. Consulta del teléfono al servidor DNS sobre la dirección 4.3.2.1.9.9.4.4.9.4.3.e164.arpa
    2. El DNS responde:
       sip:fulano@foo.com
       mailto:fulano@gmail.com
    3. El teléfono llama a sip:fulano@foo.com

ENUM

- Estado del arte:
  - ENUM soportado por algunos proxies, Asterisk y algunos teléfonos SIP.
  - El DNS .e164.arpa no tiene aún las zonas subdelegadas en muchos países.
- Alternativa: **e164.org**
  - Servidor DNS privado independiente. Permite al usuario asociar su número de teléfono convencional a una dirección VoIP, correo, web, etc.

Registro de llamadas

- Asterisk permite llevar un control exhaustivo de todas las llamadas que se han realizado o recibido.
- Es interesante para control propio de facturación, independiente del proveedor (si no se es uno de ellos).
- Permite realizar estadísticas.
- Este control se denomina CDR: Call Detail Record.

Registro de llamadas

- El registro del CDR se escribe por defecto en el archivo /var/log/asterisk/cdr-csv/Master.csv
- Existen extensiones al cdr: cdr_mysql por ejemplo, que permiten almacenar los registros en una base de datos.
- cdr_mysql está disponible en asterisk-addons
- El CDR se configura en el archivo cdr.conf, para el módulo de MySQL, se utiliza cdr_mysql.conf
- Para confirmar el estado del CDR desde el CLI, se puede ejecutar:

  CLI> cdr status

Registro de llamadas

- Asterisk genera un CDR (registro) para cada llamada.
- Algunos de los campos más importantes son:
  - **accountcode:** código de la cuenta a utilizar.
  - **src**: número del caller ID.
  - **dst:** extensión destino.
  - **dcontext:** contexto destino.
  - **start:** comienzo de la llamada (fecha/hora).
  - **answer:** respuesta de la llamada (fecha/hora).
  - **end:** fin de la llamada (fecha/hora).
  - **duration**: duración de la llamada en segundos, desde que fue discada hasta el corte.
  - **billsec:** duración de la llamada en segundos, desde que fue atendida hasta el corte.
  - **disposition:** estado de la llamada (atendida, no atentendida, ocupado, fallida).

Registro de llamadas

- Existen muchas aplicaciones que permite gestionar el CDR. Desarrollar una propia no es realmente muy complejo.
- Algunas aplicaciones open source son:
  - **Astbill:** es una de las mejores aplicaciones opensource para tarificación, control de cuentas y llamadas.
  - **Areski Stat v2:** se trata de una aplicación para listar y realizar estadísticas de las llamadas realizas o enviadas.
  - **A2Billing**

Sistema de logs

- En el archivo /etc/asterisk/logger.conf se encuentra la configuración del sistema de logging de Asterisk.
- Los distintos niveles de información a capturar en los logs son:
  - **Verbose:** mensajes generales sobre lo que está ocurriendo en el sistema (por ej, si el valor de verbosity es mayor a 3, muestra las instrucciones del plan de marcación).
  - **Debug:** mensajes con información extendida, en general utilizados por programadores.
  - **Notice:** notificaciones no críticas.
  - **Warning:** mensajes de alerta posiblemente críticos.
  - **Error:** mensajes indicando que ocurrió algo grave.

Sistema de logs

- En el contexto [logfiles] del archivo logger.conf se indican los archivos y mensajes a loguear en c/u, la sintaxis es: *archivo => nivel1,...,niveln*
- Los archivos de log se crean por defecto en /var/log/asterisk/ (esto se puede cambiar /etc/asterisk/asterisk.conf).
- Ejemplos:

  debug => debug

  full => notice,warning,error,debug,verbose

Sistema de logs

- Para enviar a la consola, hay que definir el archivo especial console:

  console => notice,warning,error,debug

- También se pueden enviar al syslog:

  syslog.local0 => debug, warning, error, notice, verbose

  Configurando además en /etc/syslog.conf:

  local0.* @ip_servidor

Sistema de logs: CLI

- Los comandos relacionados con el manejo del log del CLI son:
  - **logger reload**: reabre los archivos de log del Asterisk y recarga la configuración del logger .
  - **logger rotate**: rota los archivos de log y luego hace un logger reload.
  - **core set verbose:** cambia el nivel de información a mostrar en la consola. Por ej:

    set verbose 999.

AGI

- La AGI (o Asterisk Gateway Interface) provee una interfaz estándar para que programas externos puedan controlar el plan de marcación.
- Generalmente, los scripts AGI se utilizan para realizar lógica avanzada, comunicarse con base de datos relacionales, etc.
- Los lenguajes más comunes de programación de scripts AGI son: PHP, Python y Perl, aunque se puede utilizar cualquier otro lenguaje.

AGI

- El intercambio de información del script con Asterisk se realiza vía los canales de comunicación: STDIN, STDOUT y STDERR.
  - Lee desde STDIN para obtener información.
  - Escribe en STDOUT para enviar información.
  - Escribe en STDERR para enviar información de debugging.
- El script AGI envía comandos a Asterisk escribiendo en el STDOUT. Seguidamente Asterisk envía una respuesta por cada uno de ellos que es leída por el script.

AGI

- Algunos ejemplos de comandos son:
  - **ANSWER:** atiende.
  - **HANGUP:** cuelga.
  - **SAY [NUMBER | DIGITS | ALPHA | PHONETICS]:** dice un número, dígito, caracter o una cadena fonéticamente.
  - **SET [CONTEXT | EXTENSION | PRIORITY]:** establece un nuevo contexto, extensión o prioridad luego de finalizada la ejecución de script.
  - **VERBOSE:** imprime un mensaje en el log.
  - **WAIT FOR DIGIT:** espera que se presione un dígito.
  - **[SET | GET] VARIABLE:** asigna u obtiene el valor de una variable del plan de marcación.

AGI

- El programa debe:
  - Tener derechos de ejecución y presentar un intérprete válido
    - Ej yum –y install php; chmod 755 mi_cript.php
  - Ser localizado por defecto en /var/lib/asterisk/agi-bin
- Cómo llamar al script desde el dialplan:

  exten => 123,1,Answer()

  exten => 123,2,AGI(mi_script.php|argumentos)

AGI: ejemplo en PHP

- El siguiente script está escrito en PHP y dicta los números que se encuentran en el archivo que se le pasa como parámetro:

```
#!/usr/bin/php -q
<?php
// Esta línea es para que que haga no mantenga en un buffer el output
ob_implicit_flush(true);
set_time_limit(6);
error_reporting(0);

// Se abren los diferentes archivos (STDIN, STDOUT y un archivo de log del AGI)
$in = fopen("php://stdin","r");
$out = fopen("php://stdout","w");
$stdlog = fopen("/var/log/asterisk/my_agi.log", "w");

// Si debug es true, escribe en el archivo de log definido anteriormente
$debug = true;
```

AGI: ejemplo en PHP

```
// Toma el nombre del archivo con los números a dictar del primer parámetro
$archivo = $argv[1];

// Define la funcion read, que lee el input del STDIN
function read() {
  global $in, $debug, $stdlog;
  $input = str_replace("\n", "", fgets($in, 4096));
  if ($debug) fputs($stdlog, "read: $input\n");
  return $input;
}

// Define la funcion write, que escribe el output en el STDOUT
function write($line) {
  global $debug, $stdlog, $out;
  if ($debug) fputs($stdlog, "write: $line\n");
  fputs($out,$line."\n");
  fflush($out);
}
```

AGI: ejemplo en PHP

```
// Lee el archivo que se paso como parámetro
$lines = file($archivo);

// Reproduce los dígitos contenidos en cada línea del mismo
// informando en el log del Asterisk la accion realizada
foreach ($lines as $line) {
    $line=trim($line);
    for ($i=0;$i<strlen($line);$i++) {
     write("VERBOSE \"REPRODUCIENDO DIGITO $line[$i]\"");
     read();
     write("SAY DIGITS $line[$i] \"\"");
     read();
        sleep(1);
    }
}
// Se cierran todos los handlers de archivos
fclose($in);
fclose($out);
fclose($stdlog);
exit; ?>
```

AGI: ejemplo en PHP

- Por ejemplo, si se quiere asociar el script a la extensión 200, se debe agregar al dialplan:

```
exten => 200,1,Answer();
exten => 200,2,AGI(dicta.php|/tmp/numeros.txt)
exten => 200,3,Hangup()
```

CLI

- Más allá de los comandos que se vieron a través del curso, se revisarán algunos otros adicionales:
- Canal de consola (console channel)
  - **console dial**: permite hacer un llamada desde la consola.
  - **console answer**: permite contestar una llamada desde la consola.
  - **console hangup**: cuelga la llamada en curso en la consola.

CLI

- Administración del servidor:
  - **stop/restart gracefully**: parar/recomenzar Asterisk cuando no haya llamadas en curso y sin aceptar nuevas llamadas.
  - **stop/restart now**: parar/recomenzar inmediatamente.
  - **stop/restart when convenient**: parar/recomenzar Asterisk cuando no haya llamadas en curso.
  - **reload**: recarga toda la configuración .
  - **module load/unload**: cargar/descargar un módulo específico.
  - **module show**: mostrar todos los módulos levantados.

CLI

- Generales:
  - **show applications:** mostrar aplicaciones registradas en Asterisk.
  - **show channels:** listar los canales definidos.
  - **show codecs:** mostrar información sobre los codecs instalados.
  - **show translation:** mostrar un cuadro de doble entrada con los tiempos de conversión entre formatos de codecs.
  - **soft hangup:** colgar una llamada en un canal.
  - **debug channel:** realizar un debug de un canal.

CLI

Plan de marcación:

- **dialplan show**: mostrar el plan de marcación actual.
- **dialplan save**: guardar los cambios realizados.
- **dialplan add/remove extension**: agregar / eliminar una extensión en un contexto dado al plan de marcación.
- **dialplan add/remove include**: incorporar / eliminar un include en un contexto dado en el plan de marcación.

ASTERISK MANAGER API

- Permite a una aplicación cliente conectarse a una instancia de Asterisk vía TCP/IP y ejecutar comandos o leer eventos.
- Generalmente, se utiliza el puerto 5038.
- Utiliza un protocolo en modo texto que consiste en líneas de tipo "clave: valor".
- Conjunto de líneas: paquete.

MANAGER: PROTOCOLO

1) Se debe establecer una sesión con el manager antes de ejecutar comandos.
2) Los paquetes pueden ser transmitidos en ambas direcciones.
3) El orden de las líneas dentro del paquete es indistinto.
4) Las líneas se delimitan con CRLF y una línea en blanco (2 CRLF consecutivos) indica el final del paquete.

MANAGER: TIPOS DE PAQUETE

El tipo de paquete está dado por las siguientes claves:

- **Action:** paquete originado en el cliente requiriendo llevar a cabo una acción particular. Contiene el nombre de la acción y los parámetros de la misma.
- **Response:** la respuesta del Asterisk a la Acción requerida por el cliente.
- **Event:** datos correspondientes a un evento generado dentro del núcleo de Asterisk o módulo.

MANAGER: AUTENTICACIÓN

- Las cuentas de usuario se configuran en /etc/asterisk/manager.conf. Por ej:

```
[general]
enabled=yes

[admin]
secret = claveadmin
deny=0.0.0.0/0.0.0.0
permit=127.0.0.1/255.255.255.0
read = system,call,log,verbose,command,agent,user
write = system,call,log,verbose,command,agent,user
```

- En este caso, "admin" es el nombre de usuario, la clave es "claveadmin" y sólo se permiten conexiones para este usuario vía localhost. El resto de las líneas establecen permisos (r,w,r/w) para cada clase (system, call, etc.).

MANAGER: CONEXIÓN

- Enviar un mensaje con acción "Login", junto con el usuario y la clave como parámetros. Por ej:

```
Action: login
Username: admin
Secret: adminclave
Events: off
```

(La última línea indica que la conexión no recibirá eventos por parte del Asterisk.)

MANAGER: CONEXIÓN

- Se recibirá por parte del servidor

```
Asterisk Call Manager/1.0
Response: Success
Message: Authentication accepted
```

- O, en caso de error:

```
Asterisk Call Manager/1.0
Response: Error
Message: Authentication failed
```

MANAGER: PAQUETES ACTION

- Se pueden proveer parámetros adicionales (por ej, un número a llamar o canal a desconectar).
- En el caso que la acción determine la ejecución de una entrada del plan de marcación, también se pueden proveer variables.
- Formato:

```
Action: <action type><CRLF>
<Key 1>: <Value 1><CRLF>
<Key 2>: <Value 2><CRLF>
...
Variable: <Variable 1>=<Value 1><CRLF>
Variable: <Variable 2>=<Value 2><CRLF>
...
<CRLF>
```

MANAGER: PAQUETES ACTION

- Algunos ejemplos de acciones

- **Command:** ejecuta un comando (por ej, reload) (privilege: command,all)
- **Events:** controla el flujo de los eventos
- **Hangup:** colgar canal (privilege: call,all)
- **IAXpeers:** lista los peers IAX (privilege: system,all)
- **ListCommands:** lista los comando disponibles del manager
- **Logoff:** logoff del manager
- **MailboxCount:** verifica la cantidad de mensajes en el mailbox (privilege: call,all)
- **MailboxStatus:** verifica el status del mailbox (privilege: call,all)
- **Originate**: origina llamada (privilege: call,all)
- **ParkedCalls:** lista las parked calls

MANAGER: PAQUETES ACTION

- Algunos ejemplos de acciones (cont.)

- **QueueAdd:** agrega un miembro a la cola (privilege: agent,all)
- **QueueRemove:** remueve un miembro de la cola (privilege: agent,all)
- **SIPpeers:** lista los peers SIP (privilege: system,all)
- **Status:** Status (privilege: call,all)
- **ZapHangup:** cuelga un canal Zap
- **ZapTransfer:** transfiere un canal Zap
- **ZapShowChannels:** muestras los canales Zap

- (será recomendable meternos con esto?)

MANAGER: EJEMPLO LLAMADA

- El cliente envía:

  ACTION: Originate
  Channel: SIP/12345
  Exten: 1234
  Priority: 1
  Context: default

- El cliente recibe, en caso de éxito:

  Event: Newchannel
  Channel: SIP/12345-ed8f
  State: Down
  CallerID:
  Uniqueid: 1124982019.19157

MANAGER: EJEMPLO LLAMADA

- El cliente recibe, en caso de éxito (cont):

  Event: Newchannel
  Channel: SIP/12345-ed8f
  State: Ringing
  CallerID:
  Uniqueid: 1124982019.19157

  Event: Newstate
  Channel: SIP/12345-ed8f
  State: Up
  CallerID:
  Uniqueid: 1124982019.19157

MANAGER: EJEMPLO LLAMADA

- El cliente recibe, en caso de éxito (cont):

  Event: Newexten
  Channel: SIP/12345-ed8f
  Context: default
  Extension: 1234
  Priority: 1
  Application: SetVar
  AppData: extension=1234
  Uniqueid: 1124982019.19157

  Response: Success
  Message: Originate successfully queued

MANAGER: EJEMPLO LLAMADA

- El cliente recibe, en caso de error:

  Event: Newexten
  Channel: OutgoingSpoolFailed
  Context: default
  Extension: failed
  Priority: 1
  Application: SetVar
  AppData: extension=failed
  Uniqueid: 1124981514.58775

MANAGER: EJEMPLO LLAMADA

- El cliente recibe, en caso de error (cont):

```
Event: Hangup
Channel: OutgoingSpoolFailed
Uniqueid: 1124981514.58775
Cause: 0

Response: Error
Message: Originate failed
```

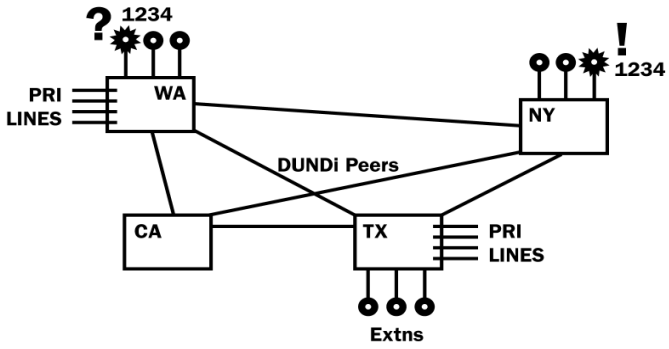
MANAGER: PROBLEMAS

- La documentación sobre el protocolo y la funcionalidad del manager está incompleta.
- Tiene problemas con el manejo de varias conexiones a la vez (> 5).
- Es recomendada la utilización de un Proxy (por, ej ProxyAstMan), para sistemas que hagan un uso intensivo del manager, comp pueden ser sistemas de monitoreo y campaña telefónica.

DUNDI

- Sistema peer-to-peer que permite localizar gateways para servicios telefónicos.
- No hay autoridad central que se encargue de la administración (como ser, el ENUM).
- Es completamente distribuido.
- Se puede considerar como un directorio telefónico.

Diagrama DUNDI



dundi.conf

[general]
; puerto mediante el cual se comunica el protocolo dundi
port=4520
; identificador de mi Asterisk en la nube dundi, se recomienda sea la MAC
; de la tarjeta de red eth0
entityid=00:0C:29:0C:AB:C2
; Tiempo que permanece en cache la ruta para ir a las extensiones aprendidas
; mediante dundi
cachetime=5
; Máximo número de saltos que se harán buscando el destino dentro de la red dundi
ttl=12
; si al preguntar a un par, el ACK tarda más de 2 segundos, se cancela la búsqueda
; a través de ese peer; es útil cuando no está activo el peer o cuando hay una conexión
; mala hacia el peer
autokill=yes

Publicación de mis números en la red dundi (contextos dundi)

- Los contextos en esta sección se enlazan con los contextos de extensions.conf
- **El contexto referenciado de extensions.conf es donde se controla qué numeros se publican en la red dundi.**
- **Al configurar un peer con el que nos vamos a enlazar, se puede determinar a cuales de los mapas puede tener acceso el peer.**

Contextos dundi

nombre_contexto =>
contexto_extensions,peso,proto,destino[,opciones]]

nombre_contexto. Nombre del contexto al que se hace referencia en una petición dundi.
contexto_extensions. Nombre del contexto en el extensions.conf, donde se buscarán los números que son solicitados en la nube dundi.
peso. Deberá ser 0 en caso de que nuestro conmutador publique directamente los números, en caso contrario, deberá tener el valor del número de saltos que necesita para llevar al destino.
proto. Cualquiera de los protocolos IP (sip, iax,323) con el cual el peer remoto se debe comunicar con nosotros.
destino. Es la información que se entrega al peer para que pueda llegar al número que está marcando.

.....Contextos dundi

opciones. Pueden ser varios argumentos los cuales indican el comportamiento que se tendrá cuando no tengamos nosotros el número que se está solicitando.
- **nounsolicited. No se permiten llamadas que no sean solicitadas.**
- **nocomunsolicited. No se permiten llamadas comerciales que no sean solicitadas.**
- **nopartial. No se permiten búsquedas para patrones parciales.**
- **residential, comercial,mobile. Le indica a los pares qué tipo de números son los que se están publicando a través de este contexto.**

[mappings]
priv => dundi-priv-local,0,IAX2,priv:${SECRET}@**132.248.175.91**/${NUMBER},nounsolicited,nocomunsolicit,nopartial

.....Contextos dundi

Plantilla. Permite especificar una respuesta genérica.

${SECRET}. Es reemplazado por el password almacenado en la base de datos.

${NUMBER}. Es el número solicitado.

${IPADDR}. Es la IP de nuestro Asterisk. No se recomienda su uso.

Generación de llaves

La autenticación en una red dundi normalmente se hace mediante llave pública/privada

cd /var/lib/asterisk/keys
astgenkey -n **dundi_ext_44xx**
yum -y install php
cp dundi_ext_44xx.pub /var/www/html
service httpd start
wget -c http://132.248.175.90/dundi_principal.pub

CLI>reload res_crypto.so
CLI>reload pbx_dundi.so
CLI>dundi lookup 4400@priv

Dundi: configuración de pares

;Par principal, el principal publica las demás extensiones
[00:0c:29:d8:86:c0] ; MAC address del par principal
model = symmetric
host = 132.248.175.90
inkey = dundi_principal ; nombre de la llave pública del par
outkey = **dundi_ext_44xx** ; nombre de mi llave privada
include = priv
permit = priv
qualify = yes
dynamic=yes

extensions.conf

; tratamos de ir en el contexto local al número de extensión,
; después vamos a buscar a otro contexto.
; dundi-priv-lookup apunta a la directiva switch la cual permite
; buscar las extensiones en otros conmutadores. Esto posibilita la
; convergencia entre múltiples servidores Asterisk con diferentes
; números de bloques de extensiones

[macro-dundi-lookup]
exten => s,1,Goto(${ARG1},1)
include => dundi-priv-local
include => dundi-priv-lookup

Definición de contextos

; Son las extensiones locales, por ejemplo: **44XX**
[dundi-priv-local]
exten => _**44XX**,1,Macro(Dial,${EXTEN})

; Buscamos con nuestros pares
[dundi-priv-lookup]
switch => DUNDi/priv

; Cuando se trata de una llamada de un par privado, llega aquí.
[dundi-priv-incoming]
include => dundi-priv-local

Búsqueda en pares

;--
; Contexto para llamadas salientes, aquí nos comunicamos
; con la red dundi, pero agregamos 4 dígitos
;--
[pares-dundi]
exten => _**XXXX**,1,Macro(dundi-lookup,${EXTEN})

[internos]

include => pares-dundi

iax.conf

[priv]
type=user
dbsecret=dundi/secret
context=internos
disallow=all
allow=ulaw
allow=alaw
allow=gsm

Curso elaborado por
Júlian Dunayevich, Lázaro Baca, Andrés Brassara y Santiago Alberch
julian@dunayevich.com
lazaro.baca@gmail.com
abrassara@gmail.com
salberch@gmail.com

Basándose en:
Irontec: **contacto@irontec.com** (CC)
Asterisk, The Future of Telephony, Jim Meggelen, Jared Smith, and Leif Madsen, O´REILLY, 2005