1 Creación do dialplan

O *dialplan* ou plan de marcado constitúe unha das partes máis importantes de Asterisk. Realmente é nel onde se amosa toda a potencia, versatilidade e flexibilidade que pode chegar a ter unha centralita telefónica software.

O *dialplan* é o centro de Asterisk. Cada díxito que se marca nun terminal recorrerá o *dialplan*, buscando "qué facer". O *dialplan* poderíase asimilar a unha táboa de enrutado, de forma que o usuario marca unha secuencia de díxitos ou caracteres e nela se atopan as accións a realizar para a secuencia de marcado.

1.1 Sumario

1.2 Esquema de traballo

O *dialplan* de Asterisk atópase no arquivo extensions.conf e está composto por contextos, extensións e prioridades.

As **extensións** son números (ou caracteres alfanuméricos) onde o usuario pode chamar. Estas extensións, conteñen accións asociadas, que son executadas de xeito secuencial, por orden de **prioridade**.

Os **contextos** son agrupacións lóxicas de extensións, e se empregan para dividir o *dialplan* en diversos entes lóxicos. Esta división é necesaria para poder dispor dun plan sostible, escalable e con posibilidades de ofrecer diversos entornos de marcado aillados.

A sintaxis para as extensións dun contexto é a seguinte:

```
exten => numero de extensión, prioridade, aplicación(argumentos)
```

Ó definir un usuario no *sip.conf* ou *iax.conf*, asociámoslle un contexto, polo que ese usuario só poderá marcar as extensións incluídas no seu contexto.

```
[proba]
exten => 1234,1,NoOp(Isto é unha proba)
exten => 1234,n,NoOp(Isto é outra proba)
```

No exemplo de arriba amósase un sencillo contexto cunha extensión e duas prioridades. Sempre é necesaria a prioridade 1, pero para as seguintes prioridades pódese usar a "n", que fará que se incremente automáticamente nunha unidade, evitando así ter que face-lo manualmente.

Ó marcar unha extensión vanse executando as aplicacións correspondentes de xeito secuencial, de acordo á súa prioridade:

```
exten => 1234,1,Aplicación 1
exten => 1234,2,Aplicación 2
```

O uso de prioridades numéricas pode dificultar a modificación do *dialplan* nun futuro, xa que si se desexa insertar unha acción no medio, sería necesario modificar a prioridade de todas as seguintes.

Para evitar isto podemos empregar a prioridade "n", que simplemente indica "a seguinte prioridade".

```
exten => 1234,1,Aplicación 1
exten => 1234,n,Aplicación 2
...
exten => 1234,n,Aplicación 9
```

Como pode observarse, deste xeito é moito máis sinxelo insertar novas aplicacións en medio, sin necesidade de preocuparnos de cambiar os números de prioridade.

En lugar de ter que expresar todas as combinacións de extensións posibles, podemos empregar patróns e variables, ó obxecto de dispor dun *dialplan* máis claro e sostible.

Os patróns comezan cun guión baixo "_" e empregan o seguintes caracteres especiais:

```
X: Calqueira díxito do 0 ó 9
Z: Calqueira díxito do 1 ó 9
N: Calqueira díxito do 2 ó 9
[]: Calqueira díxito que se atope entre os corchetes, por exemplo [123] implica o 1,o 2 e o 3.

. (punto): Calqueira cousa, por exemplo _9.
Implica calqueira número que comece por 9, sin ter en conta o 9 en si mesmo.

!: Caracter de desambigüación.
Indica que o procesamento ten que ser detido tan pronto como se atope un patrón axeitado.
```

Empregando os patróns é posible simplificar o *dialplan* e na práctica empréganse patróns como os seguintes:

_[789]XXXXXXXX: Números fixos nacionais (que comecen por 7,8 ou 9 e teñen nove díxitos)

_6XXXXXXXX: Números de móbil (comezan por 6 e teñen nove díxitos)

1.3 Aplicacións e funcións

Ó marcar unha extensión execútase a aplicación asociada á prioridade correspondente. A diferencia das aplicacións, as funcións soamente poden ser empregadas dentro de ditas aplicacións. Vexamos un exemplo:

```
exten => 2000,1,Dial(SIP/carlos)
```

Ó marcar a extensión 2000 executarase a aplicación *Dial*, á que lle indicamos que chame a Carlos, que é un usuario SIP.

Primeiro indicamos a tecnoloxía (SIP/) e despois o nome do usuario que definimos no arquivo correspondente, neste caso *sip.conf*.

As funcións, en cambio execútanse dentro das aplicacións, por exemplo:

```
exten => 1234,1,NoOp(Estas chamándome dende o : ${CALLERID(num)})
```

Neste caso execútase a aplicación *Noop*, que únicamente imprime a mensaxe que se lle pasa como argumento por pantalla, e no seu interior execútase a función *CALLERID* con argumento *num*, que devolve o número chamante.

Listaxe 1: Exemplo de dialplan básico

```
[a-extensions]
exten => 2000,1,Dial(SIP/carlos,60,Tt)
exten => 2001,1,Dial(SIP/mendez,60,Tt)

[a-servizos]
exten => 1234,1,Playback(tt-monkeys)
exten => 1235,1,Dial(IAX2/guest@pbx.digium.com/s@default)

[dende-usuarios]
include => a-extensions
include => a-servizos
```

Neste exemplo definimos tres contextos: o contexto *a-extensions* inclúe as extensións 2000 e 2001, que fan uso da aplicación *Dial* para chamar a carlos e a mendez respectivamente.

O contexto *a-servizos* contén dous servizos: 1234, que reproduce unha locución de exemplo (o sonido duns monos) e 1235, que fai unha chamada a Digium a través do IAX2.

O contexto *dende-usuarios* inclúe os outros dous contextos, e é o que asignaremos ás extensións no sip.conf, de xeito que sexan capaces de marcar todas as extensións incluídas por este contexto.

Podería facerse todo nun só contexto, pero iso derivaría nun *dialplan* pouco sostible.

1.4 Macros

En moitas ocasións atoparémonos con fragmentos do '**dialplan'** que son idénticos, e nos que soamente cambia un pequeno detalle.

Seguindo a filosofía do *dialplan* da **Listaxe 2**, teríamos que copiar o mesmo para a extensión 2001,2002,2003, modificando en cada fragmento o protocolo a usar (SIP ou IAX), nome do usuario e o número de extensión. Isto, ademais de ser bastante traballoso, complicaría o día de mañá calquera modificación.

Listaxe 2: Fragmento dun dialplan

```
[a-extensions]
exten => 2000,1,Dial(SIP/mendez,20)
exten => 2000,n,Goto(2000-${DIALSTATUS},1)

; si está ocupado, reprodúcese a mensaxe correspondente
exten => 2000-BUSY,1,VoiceMail(2000,b)
exten => 2000-BUSY,n,Hangup

; si non contesta, reprodúcese a mensaxe de "no dispoñible'
exten => 2000-NOANSWER,1,VoiceMail(2000,u)
exten => 2000-NOANSWER,n,Hangup

; en calqueira outro caso, facemos o mesmo que si non contesta.
exten => 2000-.,1,Goto(2000-NOANSWER)
```

Para solucionar este problema, poderemos definir **macros**, que son o equivalentes a "funcións" ou "subrutinas" das linguaxes de programación. Do mesmo xeito que as funcións, unha macro recibirá uns parámetros que poderemos utilizar dentro da súa definición.

As macros defínense no ficheiro *extensions.conf*, como un contexto máis, tendo en conta unha serie de condicións:

1. O nome do "contexto" empezará por "macro-" para identificar que se trata dunha macro, e non dun contexto normal.
2. As macros sempre empezarán pola extensión "s". Ademais, non se permite un "goto" dende outra parte do *dialplan* a un punto diferente da macro que non sexa ó seu inicio.
3. Dentro da definición da macro, os parámetros recibidos son accesibles mediante ${ARG1}, ${ARG2}... E a extensión dende a que se chamou á macro, podémola obter en ${MACRO_EXTEN}.

Para executar unha macro, no *dialplan* temos dispoñible a aplicación "Macro":

```
exten => 1234,1,Macro(nome_macro, parametro1, parametro2, parametro3, ...)
```

No exemplo que vemos na **Listaxe 3** usamos unha macro para o código que tiñamos, saltando á caixa de correos de voz se está ocupado. A esta macro chamarémola "extension-VM".

Listaxe 3: Extension VM

```
[macro-extensionVM]
;esta macro recibirá dous parámetros
;o primero indica o dispositivo ó que hai que chamar
;o segundo é o número do buzón de voz ó que desviar a chamada
exten => s,1,NoOp(chamada para ${ARG1})
exten => s,n,Dial(${ARG1},20)
exten => s,n,Goto(s-${DIALSTATUS},1)

; si está ocupado, reprodúcese a mensaxe correspondente
exten => s-BUSY,1,VoiceMail(${ARG2},b)
exten => s-BUSY,n,Hangup

; si non contesta, reprodúcese a mensaxe de "non dispoñible"
exten => s-NOANSWER,1,VoiceMail(${ARG2},u)
exten => s-NOANSWER,n,Hangup

; en calqueira outro caso, facemos o mesmo que si non contesta.
exten => s-.,1,Goto(s-NOANSWER,1)
```

E adaptamos o contexto *[a-extensions]* para que use esta macro para chamar ós usuarios:

```
[a-extensions]
exten => 2000,1,Macro(extensionVM,SIP/mendez,2000)
exten => 2001,1,Macro(extensionVM,SIP/carlos,2001)
exten => 2002,1,Macro(extensionVM,SIP/laura,2002)
exten => 2003,1,Macro(extensionVM,IAX2/juan,2003)
exten => 2004,1,Macro(extensionVM,IAX2/maria,2004)
```

Isto facilita moitísimo o mantemento do *dialplan*, e faino ao mesmo tempo máis lexible.

1.5 Gardar información na base de datos ASTDB

Asterisk mantén determinada información sobre o seu estado actual nunha base de datos coñecida como **AstDB**. A información que se almacena nesta base de datos inclúe os datos de rexistro dos usuarios, estado das colas, etc. Isto permite que o sistema poida recuperar esta información en caso de ter que reiniciarse.

A información na *AstDB* organízase en familias e identifícanse mediante unha chave que será única dentro da familia. Para cada familia e chave pódese almacenar un valor.

Dende o *CLI* dispomos de distintos comandos cos que podemos acceder á información da *AstDB*, podendo tanto ler como escribir nela:

database show <familia> <chave>: Amosa o contido da base de datos, podendo filtrar por familia e chave. Vemos un exemplo na *Listaxe 4*.

database del <familia> <chave>: Permite borrar un valor da base de datos.

database deltree <familia> <árbore-de-chaves>: . Borra toda unha árbore de claves da base de datos. Por exemplo, pode borrar toda a árbore "Registry" da familia "IAX".

database put <familia> <chave> <valor>: . Permite introducir información na base de datos (ver *Listaxe 5*).

Listaxe 4: Contido da base de datos AstDB

```
PBX*CLI> database show
/IAX/Registry/caracas  : 192.168.15.209:4569:60
/IAX/Registry/juan    : 192.168.15.2:4570:60
/SIP/Registry/carlos  : 192.168.15.2:5060:3600:carlos:sip:carlos@192.168.15.2

PBX*CLI> database show iax
/IAX/Registry/caracas        : 192.158.15.209:4569:60
/IAX/Registry/juan             : 192.168.15.2:4570:60

PBX*CLI> database show iax registry/juan
/IAX/Registry/juan       : 192.168.15.2:4570:60

PBX*CLI> database showkey iax/registry/juan
/IAX/Registry/juan       : 192.168.15.2:4570:60
```

Listaxe 5: Introducindo información

```
PBX*CLI> database put familia chave valor
Updated database successfully

PBX*CLI> database show
/IAX/Registry/caracas        : 192.168.15.209:4569:60
/IAX/Registry/caracas        : 192.168.15.2:4570:60
/IAX/Registry/caracas        : 192.168.15.2:5060:3600:carlos:sip:carlos@192.168.15.2
/familia/chave               : valor
```

Ler a información que Asterisk garda na base de datos pode resultar interesante para determinadas aplicacións, pero o verdadeiramente interesante desta base de datos é que nós podemos tamén gardar e ler información propia, para utiliza-la dende o *dialplan*. Para isto dispomos dunha serie de funcións:

Gardar un valor na base de datos:

```
exten => s,1,Set( DB(familia/chave) = valor )
```

Ler un valor da base de datos:

```
exten => s,1,Set( var = ${DB(familia/chave)})
```

Comprobar se existe un valor para unha familia e chave:

```
exten => s,1,GotoIf ($[${DB_EXISTS(familia/chave)} = 0]?s,chamar)
```

No caso de existir un valor para a familia/chave, *DB_EXISTS* garda este valor na variable *${DB_RESULT}*.

Podemos probar a chamar ao número *78 e despois executar database *show* no *CLI*. Veremos que se gardou un rexistro na base de datos, da forma: *DND/num_extension*.

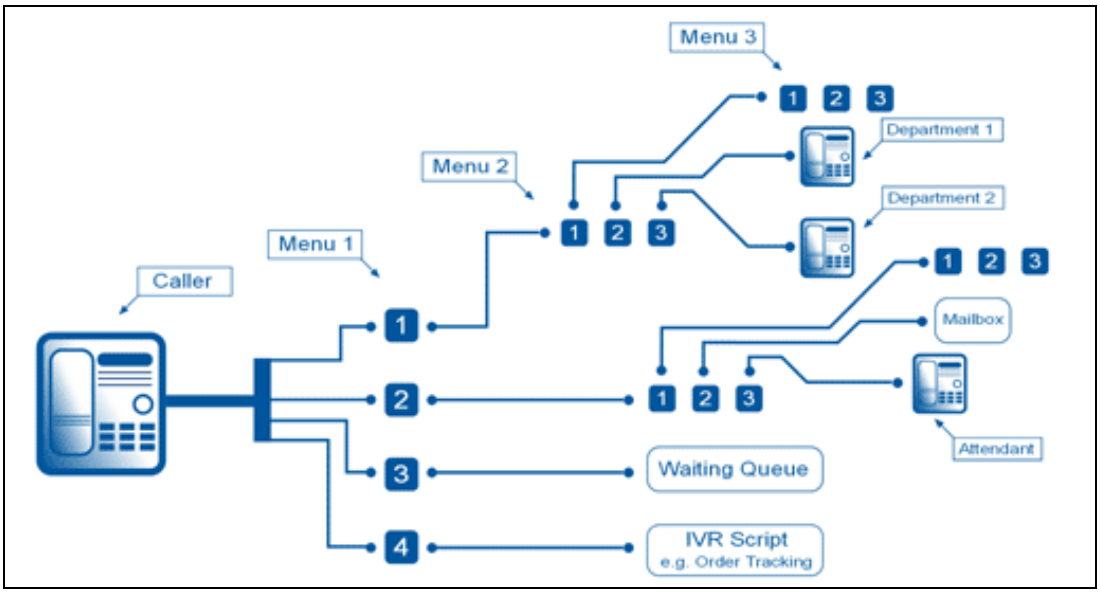
Agora actualizaremos a nosa macro ex-tensionVM para que teña en conta se o usuario ten activado o "non-molestar", como podemos ver na *Listaxe 6*.

Listaxe 6: Actualizando extension VM

```
[macro-extensionVM]
; esta macro recibirá dous parámetros
; o primeiro indica o dispositivo ó que hai que chamar
; o segundo é o número do buzón de voz ó que desviar a chamada
exten => s,1,NoOp(chamada para ${ARG1})

; si NON ten activado o no-molestar, saltamos a "chamar"
exten => s,n,Gotolf($[${DB_EXISTS(DND/${MACRO_EXTEN})} = 0]?s,chamar)
exten => s,n(DND),Playback(vm-extension)
exten => s,n,Playback(vm-isunavail)
exten => s,n,Hangup
exten => s,n(chamar),Dial(${ARG1},20)
exten => s,n,Goto(s-${DIALSTATUS},1)

; si está ocupado, reprodúcese a mensaxe correspondente
exten => s-BUSY,1,VoiceMail(${ARG2},b)
exten => s-BUSY,n, Hangup

; si non contesta, reprodúcese a mensaxe de "non dispoñible"
exten => s-NOANSWER,1,VoiceMail(${ARG2},u)
exten => s-NOANSWER,n,Hangup
exten => s-NOMOLESTAR,1,Playback(vm-extension)
exten => s-NOMOLESTAR,n,Playback(vm-isunavail)
exten => s-NOMOLESTAR,n,Hangup

; en calqueira outro caso, facemos o mesmo que si non contesta.
exten => s-.,1,Goto(s-NOANSWER,1)
```

1.6 Menú de voz interactivo ou IVR

Un IVR é un menú onde o usuario pode interactuar mediante pulsaciones DTMF (tonos multifrecuencia) co sistema telefónico. As empresas adoitan usar estes menús, reproducindo unha mensaxe de benvida cando reciben unha chamada, e ofrecéndonos despois as típicas opcións de "pulse 1 para falar co departamento de ventas; 2 para falar con contabilidade...".



Ésta sería a forma máis sinxela de IVR coñecido tamén como **Operadora Automática**.

Asterisk ofrécenos todas as ferramentas necesarias para crear dende o *IVR* máis simple ata os sistemas máis complexos.

O primeiro que necesitaremos para crear un IVR son os sons que se van a reproducir. Estes deben atoparse no directorio de sonidos do Asterisk, normalmente */var/lib/asterisk/sounds/*, e ter un formato recoñecido por éste.

Xunto con Asterisk se distribúe un xogo de sons estándar en inglés. Existen xogos de voces de gran calidade en castelán, pero normalmente necesitaremos algúns sons personalizados ("grazas por chamar ó Instituto San Clemente", "pulse 1 para falar co departamento de informática...", etc.).

Aínda que é posible usar a aplicación Record de Asterisk para gravar as mensaxes , para obter a mellor calidade aconséllase gravalos cunha aplicación especializada (Audacity, por exemplo). Neste caso debemos gravar os arquivos a 8KHz e 16 bits.

Para gravar mensaxes dende o *dialplan* emprégase a aplicación *Record*, coa sintaxe:

```
Record(nome.formato|silencio|maxDuracion|opcións)
```

Si dentro do nome do arquivo engadimos os caracteres %d, éstes serán sustituídos por un número secuencial, para evitar que sobreescribamos unha gravación existente. Neste caso, a variable *${RECORDED_FILE}* conterá o nome final do arquivo. A aplicación Record gravará a mensaxe ata que o usuario pulse a tecla #, ou ben ata que chegue á duración máxima indicada como parámetro ou se detecten tantos segundos de silencio como se indicaron.

```
[servizos]
exten => *77,1,Record(sonido-%d.alaw)
exten => *77,n,Playback(beep)
exten => *77,n,Playback(${RECORDED_FILE})
exten => *77,n,Hangup
```

Para programar os nosos IVR disporemos de algunhas aplicacións interesantes para o *dialplan*:

1. **Playback(sonido)**: Reproduce un sonido
2. **WaitExten(tempo)**: Espera a que o usuario teclee unha opción (ou un número de extensión).
3. **Background(sonido)**: Reproduce un son, pero o usuario pode interrumpir a reproducción, tecleando un número de opción. Sería equivalente a empregar as aplicacións Playback y WaitExten de xeito simultáneo.
4. **GotoIfTime(hora|dias_semana|dias_mes|ano?si_certo:si_falso)**: Realiza un salto a outro punto do *dialplan* dependendo da data e a hora. Resulta moi útil para actuar de xeito diferente si estamos en horario de oficina ou non.

Para que non entre nun bucle infinito dispomos de dous tipos de retardo: tempo inter-dixito e tempo de resposta:

```
Set(TIMEOUT(digit)=3)
Set(TIMEOUT(response)=9)
```

Unha vez que o usuario comeza a teclear unha opción, o tempo máximo permitido entre díxitos será *TIMEOUT(digit)*. Si pasa este tempo dende o último díxito tecleado, o sistema considera que o usuario rematou de teclear o número da opción e saltará a esa extensión (ou opción). Si non existe, saltará á extensión "i".

Por outra parte, si o usuario non comeza a teclear unha opción no tempo indicado por *TIMEOUT(response)*, o *IVR* saltará á extensión "t". (Ver Listaxe 7).

Listaxe 7: Programando o IVR

```
; IVR ó que chegarán as chamadas entrantes
; A mensaxe de Benvida dice "Grazas por chamar ó IES San Clemente
; Para falar có departamento de Informática pulse 1,
; para falar có departamento de Idiomas pulse 2.
; Se coñece a extensión da persoa
;  coa que desexa falar, márquea agora

[entrada]
exten => s,1,Set(TIMEOUT(digit)=3)
exten => s,n,Set(TIMEOUT(response)=9)        '

; Comprobamos si estamos no horario do instituto
exten => s,n,GotoIfTime(09:00-22:30|mon-fri|*|*?dentro_horario)
exten => s,n,Playback(estamos-pechados-deixe-mensaxe)
exten => s,n,VoiceMail(999,s)
exten => s,n,Hangup
exten => s,n(dentro_horario),Background(benvida)
exten => s,n,WaitExten(10)
exten => s,n, Goto (operadora)

; Si elixe a opción 1, mandamos ó usuario á cola de Informatica
exten => 1,1,Queue(informatica,t,60)
exten => 1,n, Hangup

; Si elixe a opción 2, mandamos ó usuario á cola de Idiomas
exten => 2,1,Queue(idiomas,t,60)
exten => 2,n,Hangup

; Si elixe unha opción incorrecta, volvemos a darlle as instruccions
exten => i,1,Goto(s,1)

; Si non selecciona ningunha opción, o mandamos á cola de operadora
exten => t,1,Queue(operadora)

; Ó incluir o contexto a-extensions, permitimos que o usuario
; teclee un número de extensión directamente
include => a-extensions
```

1.7 Sala de Conferencias

As salas de conferencia permiten que varios usuarios manteñan unha conversación entre eles, como si estiveran reunidos nunha mesma sala.

As posibilidades que ofrece Asterisk para a creación de salas de conferencias son enormes, podendo crear salas nas que unha persoa pode falar mentras todos os demais escoitan, ou ben que todos poidan intervir ó mesmo tempo.

Tamén existe a posibilidade de enmudecer ou expulsar a un usuario dunha sala durante a conferencia, bloquear para que non poidan entrar máis usuarios, gravar a conferencia, etc.

Podemos definir unha serie de salas de xeito estático, no arquivo *meetme.conf*. Este arquivo contén unha sección *[general]* cun único parámetro *audiobuffers* para indicar a cantidade de buffers que empregará a aplicación para amortiguar o jitter (retardo).

Despois da sección *[general]* hay unha sección *[rooms]* coa definición das distintas salas de conferencias. Ver na **listaxe 8** un exemplo do arquivo *meetme.conf*.

Listaxe 8: Arquivo meetme.conf

```
[general]
audiobuffers=5
```

```
[rooms]
; Sala de conferencias nº 100, sin PIN
conf => 100

; Sala de conferencias nº 101 con PIN 1234
conf => 101,1234

; Sala de conferencias nº 102, con PIN 1234 e PIN de adminstrador 5544
conf => 102,1234,5544
```

Aínda tamén é posible crear salas dinámicamente, por parte dos mesmos usuarios.

A aplicación que empregaremos no *dialplan* para acceder ás salas de conferencias é *Meetme*:

```
exten => xxxx,n,Meetme(numero_de_sala[|opcions[|PIN]]])
```

As opcións que podemos especificar son as seguintes: **a.** Entrar en modo Administrador. O administrador pode bloquear a conferencia para que non entren máis participantes. (ver opción "s")

A. Entrar en modo "marcado". Cando este usuario salga da conferencia, ésta finaliza.

b. Executa o script AGI indicado en {MEETME_AGI_BACKGROUND}. Por defecto: conf-background.agi só funciona se todas as canles da conferencia son ZAP.

c. Reprodúcese un aviso indicando cantos usuarios hai na conferencia.

d. Crea a conferencia dinámicamente.

D. Crea a conferencia dinámicamente, e asígnalle un PIN.

Ampliemos o noso contexto *[servizos]*, cunha extensión para crear e unirse a unha conferencia:

```
exten => 800,1,Meetme(,DMIsr)
exten => 800,n,hangup
```

Cando un usuario chame á extensión 800, o sistema pediralle que introduza un número de conferencia, terminando con #. O usuario pode asignar o número que desexe (con calquera cantidade de dígitos).

Seguidamente terá que asignar un PIN a esta conferencia, e dicir o seu propio nome. Desta forma créase unha nova conferencia dinámicamente, co número asignado polo usuario (parámetro D).

O usuario que creou a conferencia quedarase conectado, escoitando unha música en espera ata que se conecte un segundo usuario (parámetro M).

Para que os demais usuarios se poidan unir á conferencia soamente teñen que marcar a extensión 800, teclear o número de conferencia que creou o primeiro usuario, e o mesmo PIN.

Cada usuario que se une á conferencia deberá dicir o seu nome. Asterisk usará este nome para reproducir un aviso aos demais usuarios, indicando o nome de quen se une á conferencia, e de quen sae da mesma parámetro I).

Durante a conferencia, cada usuario poderá marcar a tecla * para enmudecer a súa terminal ou adaptar os niveis de volume do audio que envía/recibe (parámetro s).

E por último, a conferencia gravarase nun arquivo co nome *meetme-conf-rec-<nº conferencia>-<ID unico>.wav* no directorio */var/lib/asterisk/sounds/* (parámetro r).

Tamén existe unha aplicación para administrar salas de conferencias desde o mesmo *dialplan*. Esta aplicación é *MeetmeAdmin* e a súa sintaxis é:

```
exten => xxx.n,MeetmeAdmin (nº_conferencia|comando[|usuario])
```

Esta aplicación, xunto coa opción "X" de *Meetme*, que permite que o usuario poida saír da conferencia tecleando un número de extensión dun só dígito, seranos moi útil para dar ao administrador da conferencia o poder de expulsar ou silenciar a aqueles usuarios molestos na conferencia.