Ramais SIP=sip.conf

Ramais IAX=iax.conf

Plano de Discagem=extensions.conf

Filas de Atendimento=queues.conf

Agentes de Atendimento=agents.conf

Salas de Conferência=meetme.conf

SIP

|  |
| --- |
| **[general]**  **context=interno**  **bindport=5060**  **bindaddr=0.0.0.0**  **srvlookup=yes**  **disallow=all**  **allow=alaw**  **allow=gsm**  **[sip](!)**  **context=interno**  **type=friend**  **host= dynamic**  **nat=yes**  **[2000](sip)**  **callerid = “2000″**  **username=2000**  **secret=1234**  **[2001](sip)**  **callerid = “2001″**  **username=2001**  **secret=1234** |

Agora vejamos o que significa cada item, mas antes vejamos a sintaxe.

A sintaxe usada no asterisk é bem simples :

[seção]

Opção = valor

Opção = valor

E assim sucessivamente.

Agora sim, vejamos quais opções foram habilitadas :

A seção **general**, são opções que se aplicam a todas as outras seções do arquivo, ou seja, todas as seções herdarão essas opções.

**bindport :**Especifica a porta de comunicação do Asterisk para aceitar conexões. A porta TCP e UDP 5060 é padrão para SIP. Já para RTP é utilizada de 10000 e 20000.

**bindaddr**Especifica o endereço IP que escutará e aguardará conexões. Quando usado 0.0.0.0 , escutará em qualquer interface configurada.

**disallow**Desabilita codecs específicos, caso utilizado all, desabilita todos codecs.

**allow**Especifica quais codecs serão habilitados , pode usar várias linhas, ou separa-los por vírgula, em nosso caso habilitamos o alaw que é o G711a, usado pelas operadoras no padrão, e o já conhecido GSM, muito utilizado nas operadores de telefonia móvel.

**language**Define o idioma para os arquivos de som utilizados pelo Asterisk.

Na próxima seção, conhecida como seção de canais ou ainda **peer,**faremos a configuração dos ramais.

Num primeiro momento usei um modelo, que servirá para todos os ramais que criar, será como um padrão seguido pelo ramais que criaremos posteriormente.

Neste caso o valor **sip,**seguido do ponto de exclamação. Essa pontuação que informa para o asterisk que isso é um modelo de uso, na realidade chamado de template. Vamos ver o que significa todos os dados.:

**[sip](!):**Uma nova seção chamada sip, que será um template.

**context**Um contexto , usado para definir regras de discagem, isso será abordado posteriormente.

**type**Define se o canal pode ligar, receber ligação ao ambos, sendo user para fazer chamada, peer para receber e friend para ambos.

**host**Define o endereço ip para o canal, no caso dynamic, aceitará um ip dinâmico.

**secret**Senha para autenticar um canal no servidor.

**nat**Se o cliente está em uma rede interna e sofrerá Nat.

**[2000](sip)    Neste item estamos criando o ramal 2000, usando sip como base.**

**callerid     Define o destino de chamada, pode se usar nome como número.**

**secret         Senha para autenticação no Servidor**

|  |
| --- |
| **general]**  **bindport = 5060**  **bindaddr = 0.0.0.0**  **srvlookup = yes**  **disallow = all**  **allow = alaw**  **allow = gsm**  **language = pt\_BR**    **[template](!)**  **type = friend**  **context = ramais**  **host = dynamic**    **[2001](template)**  **secret = 1234**  **mailbox = 2001**    **[2002](template)**  **secret = 1234**  **mailbox = 2002** |

Neste arquivo podemos ver que não há nada de diferente há não ser a entrada mailbox. Este opção nós iremos usar mais para frente quando configurarmos a caixa postal do usuário para receber mensagens de voz. Neste contexto ele simplesmente consulta a caixa de mensagens em questão e traz essa informação na tela de exibição do Telefone IP ou software.

IAX

|  |
| --- |
| **[general]**  **bindport = 4569**  **bindaddr = 0.0.0.0**  **disallow = all**  **allow = alaw**  **allow = gsm**  **language = pt\_BR**    **[template2](!)**  **type = friend**  **context = ramais**  **host = dynamic**    **[2003](template2)**  **secret = 1234**  **mailbox = 2003**  **requirecalltoken = no**    **[2004](template2)**  **secret = 1234**  **mailbox = 2004**  **requirecalltoken = no** |

A configuração do arquivo iax é muito semelhante ao sip, temos bindport (porta UDP) bindaddr (endereço), os protocolos em disallow e allow, a língua (language). Usamos o modelo template2, para aplicar nos ramais 2003 e 2004, senhas padrão 1234, host dinâmico, ou seja, o ramal não está configurado para um ip especifico, o único item novo é o requirecalltoken, que vai permitir o registro do ramal sem a necessidade de validar a chamada por token.

Os ramais já estão prontos, agora devemos carregá-lo no Asterisk, executando os comandos abaixo:

|  |
| --- |
| #asterisk -r  \*CLI> sip reload  \*CLI> iax2 reload  \*CLI> sip show peers    Name/username Host Dyn Nat ACL Port Status  2001/2001 192.168.0.104 D N 35458 Unmonitored  2002 (Unspecified) D N 5060 Unmonitored  2 sip peers [Monitored: 0 online, 0 offline Unmonitored: 2 online, 0 offline]    \*CLI> iax2 show peers    Name/Username Host Mask Port Status  2004 (Unspecified) (D) 255.255.255.255 0 Unmonitored  2003 192.168.0.102 (D) 255.255.255.255 4569 Unmonitored  2 iax2 peers [0 online, 0 offline, 2 unmonitored] |

Extensions

O Plano de discagem envolve o contexto que nada mais é que um nome dado à que contexto será aplicado determinados ramais, as extensões, que são entradas ( seqüência de caracteres ) recebidas pelo Asterisk, prioridade, ou seja a ordem que serão processadas as regras e por ultimo a aplicação que será executada. O Asterisk possui muitas aplicações, mas de 100 com certeza.

No caso vimos duas , Dial e HangUP.

A sintaxe das extensões é:

**exten => EXTENSÃO,PRIORIDADE,APLICAÇÃO**

A aplicação Dial conecta dois canais de comunicação o valor após a aplicação, no caso SIP/2000 , informa qual tecnologia que está sendo usada e qual a identificação. Sua sintaxe completa é :

**Dial(tecnologia/Identificação[,timeout][,opções]**

Em outra ocasião abordaremos todas as opções, basta saber que o timeout é o tempo que o Asterisk vai aguardar atender a chamada para ser completada e opções temos várias como desligar chamada após alguns segundos, habilitar transferência , habilitar gravação, etc.

Já o HangUP encerra a chamada , fechando todos os canais.

Com isso sim temos já nossos ramais prontos para testar.

Vamos acessar o prompt de comando do asterisk , conforme abaixo e executar os comandos :

**#asterisk –r**

**CLI> sip reload**

**CLI> sip show peers**

Name/username Host Dyn Nat ACL Port Status

2001 (Unspecified) D 0 UNKNOWN

2000 (Unspecified) D 0 UNKNOWN

2 sip peers [Monitored: 0 online, 2 offline Unmonitored: 0 online, 0 offline]

**CLI> dialplan reload**

Dialplan reloaded.

**CLI> dialplan show**

[ Context 'interno' created by 'pbx\_config' ]

’2000′ => 1. Dial(SIP/2000) [pbx\_config]

2. HangUP() [pbx\_config]

’2001′ => 1. Dial(SIP/2001) [pbx\_config]

2. HangUP() [pbx\_config]

O comando **asterisk –r,**acessa o console do asterisk, **sip reload e dialplan reload,**recarregam tanto o arquivo sip.conf como extesions.conf para dentro do asterisk, e então **sip show e dialplan show,**vão mostrar as configurações atuais.

Após isso basta usar um cliente qualquer para testar, no Windows é possível usar o Zoiper em <http://www.zoiper.com/softphone/classic/>, tendo ele versão para Windows como para Linux.

Espero que aproveitem, e até o próximo tutorial Asterisk.

Agora devemos configurar o plano de discagem, o que ira permitir os ramais se comunicarem. Iremos usar o arquivo extensions.conf, e novamente iremos começar a partir de um arquivo sem conteúdo .

|  |
| --- |
| **# >/etc/asterisk/extensions.conf**  **#vim /etc/asterisk/extensions.conf**      **[ramais]**  **;SIP**  **exten =>2001,1,Dial(SIP/2001)**  **exten =>2002,1,Dial(SIP/2002)**    **;IAX**  **exten =>2003,1,Dial(IAX2/2003)**  **exten =>2004,1,Dial(IAX2/2004)** |

Como nos itens anteriores deveremos recarregar estas configurações, vejamos como:

|  |
| --- |
| #asterisk -r  \*CLI> dialplan reload  \*CLI> dialplan show ramais    [ Context 'ramais' created by 'pbx\_config' ]  ’2001′ => 1. Dial(SIP/2001) [pbx\_config]  ’2002′ => 1. Dial(SIP/2002) [pbx\_config]  ’2003′ => 1. Dial(IAX2/2003) [pbx\_config]  ’2004′ => 1. Dial(IAX2/2004) [pbx\_config]    -= 4 extensions (4 priorities) in 1 context. =- |

Depois desse processo, basta testar, lembrando que você deverá ter um software compatível com IAX2, tal como Zoiper, [www.zoiper.com](http://www.zoiper.com/). Agora vamos começar a ver as novas configurações, o assunto deste Post, a URA.

Configurações da URA

Deveremos editar o arquivo extensions.conf para podermos configurar nossa URA. O processo neste exemplo vai consistir em gravar um mensagem através do Telefone IP ou SoftFone, e depois usar-la para resgatar. Depois de gravarmos nossas mensagens iremos configurar a URA propriamente dito.

Algumas novidades serão utilizadas nessa configuração, em relação à criação de extensões e aplicações que deveremos usar.

Algumas aplicações importantes para o uso da URA , lembrando que o Asterisk possui mais de 170 aplicações. Mas antes de falarmos das aplicações, vejamos como são chamadas.

Normalmente temos um contexto (contexto têm entradas do tipo “[]“, por exemplo [ramais]), dentro do contexto teremos as entradas com extensão, prioridade e aplicação:

**exten =>**extensão, prioridade, aplicação()

A extensão pode ser uma seqüência de caracteres, não podendo ser superior a 79 caracteres. Dentro de cada contexto, nós definimos uma ou mais extensão. Uma extensão é uma instrução que o Asterisk irá seguir, acionada por uma chamada de entrada ou por dígitos discados num canal. As extensões especificam o que acontece às chamadas enquanto elas seguem seu caminho pelo dialplan. Apesar de as instruções poderem ser usadas para especificar ramais de telefones no sentido tradicional, elas podem ser utilizadas para muito mais no Asterisk.

Existem algumas extensões especiais, muito utilizadas como: s (start), i (extensão inválida), t (quando o tempo para a escolha de uma opção termina), fax (desvia automaticamente a extensão fax)

Na prioridade a extensão pode ter vários passos, chamados prioridades. Cada prioridade é numerada seqüencialmente, começando por 1. Cada prioridade executa uma aplicação específica. Como exemplo, a seguinte extensão pode responder a uma chamada (na prioridade 1) e então desligar (na prioridade 2):

exten => 2001,1,Answer()

exten => 2001,2,HangUP()

E aplicação , como nome diz, será o que o Asterisk, irá executar, cada aplicação executa uma ação especifica. Abaixo veremos as aplicações necessárias para nossa URA.

**Answer ():**Está aplicação atende o canal se estiver sendo solicitado, normalmente antes de executar outra aplicação, ele deve ser atendida, e o responsável é essa aplicação:

Ex: exten => s,1,Answer()

Essa extensão inicia a aplicação Answer com prioridade 1, o valor s é de start, 1 é a prioridade e Answer a aplicação propriamente dita. Depois disse ele segue o que o arquivo de extensão solicita, através do Dialplan.

**Wait() :**Aguarda por um número de segundos

Ex: exten = s,2,Wait(1)

Essa extensão aguarda por 1 segundo antes de prosseguir.

**PlayBack()**: Toca um arquivo previamente gravado para um canal. O PlayBack não permite que o usuário interaja com o sistema, esses sons devem ficar em /var/lib/asterisk/sounds

Ex: exten = s,3,PlayBack(goodbye)

Essa extensão ira tocar o arquivo goodbye no canal especificado.

**Background() :** Similar ao PlayBack, mas este permite que o usuário interaja para que possa ser digitado um valor na Ura. Após essa interação, é bem provável que use o Goto, para encaminhar para extensão solicitada.

Ex: s,2,BackGround(saudação)

Irá tocar o arquivo saudação no canal especificado, aguardando uma interação do usuário para encaminhar para aplicação solicitada.

**Record()**: Usado para fazer gravações, as gravações são salvas em /var/lib/asterisk/sounds. Posteriormente podem-se usar tais gravações em nossa usa. Com certeza iremos brincar com isso.

Ex: 2005,3,Record(menu.alaw)

Iŕa gravar o arquivo menu.alaw na contexto 2005 ( Normalmente o próprio ramal atendido)

**GoTo() :**Pula para um contexto, extensão e prioridade especifica. Mas pode desviar a aplicação para uma extensão, prioridade dentro do mesmo contexto, ou até uma prioridade especifica dentro do mesmo contexto e extensão.

Visto as aplicações mais importantes, vamos gradativamente implantar nossa URA, para começarmos, iremos fazer um exemplo bem simples, escolheremos um ramal, onde ele ira atender com a aplicação Answer, Tocar algum arquivo ( os arquivos instalados do pacotes asterisk-core-sounds-en-gsm, ficam em /usr/share/asterisk/sounds ),e após ouvirmos o arquivo, o ramal será desligado via hangup.

Para isso devemos editar o arquivo extensions.conf novamente e adicionar as linhas abaixo, o restante que já havia sido configurado não precisa ser alterado.

**exten => 2000,1,Answer()**

**exten => 2000,2,PlayBack(demo-enterkeywords)**

**exten => 2000,3,hangup()**

Não se esqueça de recarregar o DialPlan. Depois de feito esse processo basta discar para o ramal 2000 que será ouvido uma mensagem. Na primeira linha temos o primeiro passo, quando o asterisk detectar o valor 2000, ira executar o primeiro comando, pois a prioridade 1 , como se fosse uma ordem, e após isso o 2 e o 3 dentro da mesma extensão. Na primeira linha o Asterisk atende a chamada, passa para a segunda linha e Toca o conteúdo do arquivo demo-enterkeyword.gsm, que encontra-se no diretório informado anteriormente, e por fim desliga com a aplicação hangup.

Já podemos ter idéia de como funciona uma Ura, ao invés de dar um hangup, poderia facilmente mandar para um novo contexto, que analisasse dígitos tais como 1 para departamento 1, dois para departamento 2, e assim sucessivamente, conforme a necessidade.

Alias porque não fazê-lo.

Podemos então melhor nosso script, mas nesse caso, vamos fazer uma gravação primeiro, dos departamentos para então criarmos a ura.

Iremos alterar as configurações do ramal 2000, agora para gravar uma mensagem, após isso tocá-la para ouvirmos e então desligar. Devemos alterar o código anterior do ramal 2000 para o seguinte:

**exten=>2000,1,Answer()**

**exten => 2000,2,Wait(1)**

**exten=>2000,3,Record(menu.gsm)**

**exten=>2000,4,Wait(1)**

**exten=>2000,5,Playback(menu)**

**exten=>2000,6,Wait(1)**

**exten=>2000,7,Hangup**

Novamente não podemos esquecer-nos de dar um reload no DialPlan. Nestas linhas temos as seguintes informações, linha 1 , que atende a ligação, na linha 2, 4 e 6 o Wait, é usado para aguardar 1 segundo antes de passar para a próxima aplicação, o Record na linha 3 irá gravar a mensagem com o nome de arquivo menu.gsm, e temos a opção PlayBack para poderemos escutarmos a gravação, e por fim na ultima linha é desligado.

**Atenção:**Cuidado com a pasta /usr/share/asterisk, caso não pertença ao usuário e grupo asterisk, a permissão de gravação será negada. E agora uma dica, em vez de numerar as linhas com as prioridades 1, 2, 3, etc, basta numerar a primeira com o número 1 e as seguintes com a letra n, que era seguirá o fluxo linha a linha. E por ultimo cuidado com rede wireless, pois tive muitos problemas usando os codecs padrões.

Ao final da gravação deve-se digitar a tecla sustenido (#), para então ouvirmos a gravação. Grave algo do tipo dois ramais que serão redirecionados como, por exemplo, suporte e vendas. Grave algo do tipo: Digite 1 para suporte, 2 para vendas e 3 para RH.

Enfim depois de todo esse processo podemos então configurar a URA. Muita calma está quase acabando. Eu acho rsrs!! O próximo passo será agora a criação da URA,logo abaixo teremos o arquivo completo do extensions.conf:

|  |
| --- |
| **[ramais]**  **;SIP**  **exten =>2001,1,Dial(SIP/2001)**  **exten =>2002,1,Dial(SIP/2002)**    **;IAX**  **exten =>2003,1,Dial(IAX2/2003)**  **exten =>2004,1,Dial(IAX2/2004)**      **exten => 2000,1,Goto(ura,s,1)**    **[ura]**  **exten => s,1,Answer()**  **exten => s,2,Wait()**  **exten => s,3,Background(menu)**    **exten => 1,1,PlayBack(tt-monkeys)**  **exten => 1,n,HangUp()**    **exten => 2,1,Dial(SIP/2001)**  **exten => 2,n,HangUP()**    **exten => 3,1,Dial(AIX2,2003)**  **exten => 3,n,HangUp()** |

Neste arquivo podemos ver novos elementos inseridos, e como havia já comentando, acho que não é novidade, temos o Goto e o Background, como novidades nesse momento. Os primeiros itens relacionados aos ramais SIP e AIX já foram comentados e não tem nenhuma novidade.

Agora alinha referente ao ramal 2000 traz a aplicação Goto, que ira desviar o fluxo para o contexto **ura,**extensão **x**e prioridade 1, que está logo abaixo.

Na extensão ura, o primeiro item executado é atender a chamada, aguardar por tempo indeterminado, já que não foi informado o tempo, e em Background narrará o menu gravado anteriormente. Quando o usuário escolher a opção entre 1,2 e 3, será desviado para tais extensões. Temos a extensão 1, tocando um som chamado tt-monkeys, na extensão 2 um desvio para o ramal 2001 e por fim extensão 3 para o ramal 2003.

A URA praticamente está pronta.

Recarregue o Dialplan novamente e teste, discando para o ramal 2000, e posteriormente para onde quer ser redirecionado.

Ufffa!!! Cabou ???? Naum!!!!

Temos um grande problema aqui: E se o usuário digitar uma extensão que não existe? E se pior, o cara teve uma dor de barriga, foi ao banheiro e deixou o telefone fora do gancho sem digitar nada?

O asterisk não tem nenhuma ação configurada, com isso teremos um sério problema. Por isso devemos tratar ramais inválidos, e tempo de aguardo. Lembram da extensão especial “i” e “t”. E justamente ela que nos tirará deste problema.

Devemos para finalizar finalmente, adicionar essas entradas na URA. Agora enfim o arquivo final:

|  |
| --- |
| **[general]**  **autofallthrough = no**      **[ramais]**  **;SIP**  **exten =>2001,1,Dial(SIP/2001)**  **exten =>2002,1,Dial(SIP/2002)**    **;IAX**  **exten =>2003,1,Dial(IAX2/2003)**  **exten =>2004,1,Dial(IAX2/2004)**      **exten => 2000,1,Goto(ura,s,1)**    **[ura]**  **exten => s,1,Set(TIMEOUT(response)=10)**  **exten => s,n,Set(TIMEOUT(digit)=2)**  **exten => s,n,Answer()**  **exten => s,n(repete),Background(menu)**    **exten => 1,1,PlayBack(tt-monkeys)**  **exten => 1,n,HangUp()**    **exten => 2,1,Dial(SIP/2001)**  **exten => 2,n,HangUP()**    **exten => 3,1,Dial(AIX2,2003)**  **exten => 3,n,HangUp()**    **exten => i,1,PlayBack(invalid)**  **exten => i,n,Goto(s,repete)**    **exten => t,1,PlayBack(vm-goodbye)**  **exten => t,n,HangUp()** |

Bom, agora temos uma URA realmente. Vamos entender o arquivo:

* O parâmetro **autofallthrough**permite o uso de tempo, com TIMEOUT , para digitação dos dígitos na URA.
* Quando ligamos para os ramais 2001 até 2004, é feito uma chamada normalmente.
* Ao ligarmos para o ramal 2000, caímos na URA, onde é feito o desvio para o contexto **ura,**extensão **s**e prioridade 1.
* Já na ura, o primeiro item é o TIMEOUT, como o nome diz , tem relação ao tempo. Nesse primeiro caso, ela limita o tempo que o usuário tem para digitar a extensão, se em 10 segundo ele não digitar, automaticamente vai ser encaminhado para extensão t, onde será tocado o arquivo **vm-goodbye.**
* No segundo item TIMEOUT(digit) , especificamos o tempo máximo entre um dígito e outro. Obviamente em nosso caso não há necessidade, pois as extensões têm apenas um digito, mas caso houvesse mais, seria o tempo padrão de 2 segundos.
* A ura ira atender em **Answer,**e em seguida iniciará tocar o que está especificado em Background, que é justamente o menu gravado anteriormente. O item repete nada mais que um nome que demos, ou label, que será usado com o Goto em algum momento.
* Extensões 1, 2,3 encaminham a ligação para seus respectivos ramais.
* A extensão “i”, é utilizado quando é digitado um ramal inexistente, neste caso será utilizada o arquivo invalid para tocar, para informar ao usuário que o ramal é invalido, em seguida é novamente utilizado o Goto para desviar a ligação para o começo da ura, no label repete.
* E por fim a extensão “t”, utilizado com os primeiros itens de TIMEOUT, desliga a chamada.

A única desvantagem é justamente a língua, para esse problema temos o pacote de sons em português, que pode ser baixado diretamente de<http://sourceforge.net/projects/disc-os/files/Disc-OS%20Sounds/1.0-RELEASE/Disc-OS-Sounds-1.0-pt_BR.tar.gz/download>.

O uso de variáveis, macros, template, podem nos auxiliar a diminuir o trabalho que seria necessário para configurações de nossos ramais, e planos de discagem (Dial Plan).

Imagine a situação que deveremos criar 100 ramais, e desses 100 ramais teremos no mínimo 100 entradas para os Planos de Discagem. Fazer isso manualmente vai gerar gasto de tempo, e muito trabalho, sem contar que o arquivo de configuração ficará gigante. Através das máscaras que são valores coringas, variáveis poderão trazer valores, por exemplo, do número discado, da extensão e prioridade, entre outras informações, e também podemos observar nos outros Post o uso de template que pode minimizar a quantidade de comandos digitados.

Por fim, ainda veremos nesse Post, algumas aplicações para tomada de decisão, por exemplo, imagine que o usuário digitou 021, uma ligação iniciando para o Rio de Janeiro, eu posso desviar essa ligação, e mandar sair por um canal especifico, usando uma aplicação como **Gotoif (Vá para, se).**Citarei algumas aplicações que poderemos usar em Posts futuros, tais como Correio de Voz, Fila ou Mensagens com Webmail para voice.

Então vamos ao Post.

**Máscara de Discagem**

A máscara de discagem são como metacaracteres, ou caracteres coringas, que substituirão valores nas extensões do Dial Plan, evitando uma grande quantidade de digitação.

Vamos imaginar criar um Dial Plan para 300 ramais, seria necessário criar uma regra no plano de descaem para cada um desses ramais, sendo que será necessária mais de linha para as regras. Com o uso de máscara de discagem, poderemos substituir um ou uma faixa de numeração.

Só devemos prestar atenção em um item:**“Todas as vezes que fizermos uso de máscara, deveremos preceder a expressão com o \_ (underline).**

Veja abaixo a tabela de Caracteres para uso com máscara de discagem:

|  |  |
| --- | --- |
| **Máscara** | **Descrição** |
| X | Substitui qualquer dígito da faixa 0 a 9. |
| Z | Substitui qualquer dígito da faixa 1 a 9. |
| N | Substitui qualquer dígito da faixa 2 a 9. |
| [145-9] | Substitui quaisquer caracteres, ou intervalo, no exemplo os valores substituídos serão 1,4 e de 5 a 9. |
| . | Corresponde a qualquer um ou mais caracteres (inclusivo) |
| ! | Corresponde a qualquer caractere exclusivo. |

Vejamos alguns exemplos:

* \_2XXX                  Qualquer Ramal começando com 2.
* \_NXXXXXXX      Qualquer número de telefone começando de 2 a 9.
* \_9090.                  Qualquer número iniciando com 9090 e possua ao menos mais um dígito.

Agora que já vimos os valores das máscaras, podemos aplicar em nosso contexto. Mas é interessante antes vermos algumas variáveis. Para que possamos fazer um uso correto das funções.

**Variáveis**

O conceito de variável aqui é comum a qualquer linguagem de programação.

Podemos criar nossas próprias variáveis ou fazer uso das variáveis já existentes, ou criadas em tempo de execução do próprio Asterisk.

Vamos imaginar criar um Dial Plan, para 200 ramais, ramais simples. Todos SIP ou AIX. Normalmente estaríamos colocando uma entrada para cada, como visto no Post anterior: **[Asterisk Parte 2 – Criando uma URA](http://www.stato.blog.br/wordpress/?p=583)**

|  |
| --- |
| **[ramais];SIP**  **exten =>2001,1,Dial(SIP/2001)**  **exten =>2002,1,Dial(SIP/2002)**  **exten =>2003,1,Dial(SIP/2003)** |

Vimos que a primeira parte, quanto aos números de Ramais podemos usar uma máscara que resolverá de forma bem simples o valores de entradas. O grande problema é para onde discar, por exemplo, (SIP/2001), onde não podemos colocar o valor de X como Máscara para ramal SIP. Neste caso se faz necessário o uso de uma variável do Asterisk, que traga justamente o número discado, que nessa caso é a variável **EXTEN.**Está variável traz a extensão que esta sendo processada no momento.

Precisamos primeiro, antes de falar propriamente das variáveis do Asterisk, visualizar como a usamos.

Existem dois tipos de variáveis: Globais e Canal.

* **Globais:**Criadas na seção Global, dentro do arquivo /etc/Asterisk/extensions.conf.
* **Canais:** Criadas no momento da criação do Canal, que ficará associada a está canal somente.
* **Ambiente:**Variáveis do ambiente Linux, pode ser utilizado usando o comando ENV, como $ {ENV (Variável Linux)}

Para criar uma variável, devemos dar um nome e simplesmente dar-lhe um conteúdo, veja abaixo:

|  |
| --- |
| **OPERADORA=041SAIDA1=E1**  **OPERADORACEL=021** |

Neste exemplo acima, foram criada 3 variáveis com seus respectivos conteúdos. Já para usarmos a variável devemos utilizar o símbolo de $, como no Shell Script, mas também devemos colocar o nome da variável dentro de chaves, veja abaixo como podemos recuperar o conteúdo das variáveis criadas anteriormente.

|  |
| --- |
| **$ {OPERADORA}$ {SAIDA1}**  **$ {OPERADORACEL}** |

Podemos ainda criar uma variável em tempo de execução, por exemplo em um Plano de Discagem, usamos o comando **SET**para isso, veja abaixo:

**exten =>             2001,1,SET(RAMAL=2001)**

**exten =>             2001,2,DIAL(SIP/${RAMAL})**

**exten =>             2001,3,Hangup()**

Neste exemplo acima, criamos a variável “RAMAL” em tempo de execução, ou seja, na criação do Canal, ou da chamada.

As variáveis de canais são as mais utilizadas normalmente, para trazer informação dos canais normalmente, como nome do canal , nome do contexto, extensão, etc. Vejamos abaixo algumas das variáveis mais usadas:

${CALLERID}                                      Identificador da chamada (nome ou número)

${CALLERIDNAME}                         O nome do id. da chamada

${CALLERIDNUM}                           O número do id. da chamada.

${CHANNEL}                                     Nome do Canal atual ou corrente.

${CONTEXT}                                      Nome do Contexto atual ou corrente.

${DATATIME}                                   Data e hora no formato o YYYY-MM-DD\_HH:MM:SS.

${DIALPEERNUMBER}                   Número de que foi chamada.

${DIALSTATUS}                                Status da Chamada.

${EXTEN}                                            Extensão da chamada atual ou corrente.

${HANGUPCAUSE}                         Causa do término da chamada

${LANGUAGE}                                 A Linguagem Atual.

${PRIORITY}                                      Prioridade Atual

Existem muitas outras variáveis de canais que podem ser consultadas em  <https://wiki.asterisk.org/wiki/display/AST/Asterisk+Standard+Channel+Variables>.

Vejamos alguns exemplos:

exten => \_20XX,1,Dial(SIP/${EXTEN})

exten => \_0ZZZZNXXXXXXX.,1, Dial(${SAIDAE1/g1/0${OPERADORA}${EXTEN:4},35,t)

Neste exemplo podemos ver o uso de máscara e da variável do Asterisk EXTEN. Esse é justamente o exemplo que havia falado acima, para evitar o  a criação de uma infinidades de canais. Quando um ramal começando com 20 e mais dois números for solicitado, irá chamar a aplicação DIAL para discar para o ramal ( SIP/${EXTEN}), onde esta variável trará o próprio numero discado. Assim precisamos apenas de uma entrada para todos os ramais do tipo 20XX.

Já no segundo exemplo estamos filtrando ligações que comecem com 0 seguida de 4números que vão de 1 a 9, que significa código de operadora seguida do código da área , em seguida qualquer numero de 2 a 9 (N)  , em seguida qualquer número 7 vezes, o que totaliza, 8 números, ou o número do telefone. Este número usou como máscara inicial o valor **“N”,** para que sejam discado valores iniciais entre 2 e 9, que inclui qualquer tipo de telefone, convencional e celulares. Mas depois disso fazemos uso de duas variáveis, que poderiam ser pré-estabelecidas anteriormente, a OPERADORA, que será o operadora usa para DDD, e a extensão com uma alteração, usaremos somente os números digitados a partir do 4 º digito, descartando os três primeiros dígitos. Ou seja, a pessoa digitou 0215129298989, será removido o 021, e pelo nosso Plano de Discagem, será inserido um zero, seguido da operadora e por fim a extensão. Supondo que a operadora seja 41, o resultado será 0415129298989, mudando assim o valor da operadora, o valor 35 é referente, ao tempo máximo para chamada, no caso 35 segundos, e o t habilita a transferência da chamada para o lado destino da chamada.

Mas notou que a variável EXTEN usou um valor 3 logo depois de declarar a variável, e uso de “:” (dois pontos). Este é outro aspecto de variáveis, é possível manipular, atribuir, fazer expressões lógicas para futuros desvio  e tomadas de decisão, etc.. Para isso veremos como manipular variáveis no subtópico abaixo: Manipulando variáveis.

**Manipulando Variáveis**

Agora que sabemos como criar devemos saber como manipular tais variáveis, em termos de strings, operações, etc.

É bem similar ao Linux, no sentindo de remover parte da String, vejamos a sintaxe abaixo:

**${variável: posição inicial: comprimento}**

Sendo que o valor da variável é obrigatório, e o restante, tanto a posição inicial que inicia em 0, quanto o comprimento que você deseja capturar da variável são optativos. Digamos que temos seguinte variável:

**VAR=123456**

Veja abaixo vários exemplo , de como resgatar diversos valores:

|  |  |
| --- | --- |
| **Variável** | **Retorno** |
| ${VAR:1} | 23456 |
| ${VAR:0:1} | 1 |
| ${VAR:-2} | 56 |
| ${VAR:-4:2} | 34 |

Como podemos ver acima, podemos ainda utilizar valores negativos que indicarão que a posição inicial deve ser contada da direita para esquerda, o contrário do padrão. Um exemplo bem simples disso é no próprio Plano de Discagem, quando usamos o valor 0 para discar de uma ramal para um número de telefone padrão:

**exten => \_0NXXXXXXX,1,Dial(DGV/g1/${EXTEN:1},10,t)**

Neste exemplo estamos removendo o valor 0 (zero) e discando na saída tronco o número do telefone .

Outro exemplo que pode ser citado é o visto anterior , no exemplo “exten => \_0ZZZZNXXXXXXX.,1, Dial(${SAIDAE1/g1/0${OPERADORA}${EXTEN:4},35,t)” . Neste exemplo removemos os o valor 0ZZ (Código operadora), e adicionamos antes da variável ${EXTEN} o valor 0 seguido da operadora, e o restante do número que DDD + o telefone convencional.

Ainda é possível executar expressões lógicas e matemáticas, abaixo os operadores e descrição:

* Operadores Lógicos:
  + &                                           E
  + |                                            OU
  + =                                            Igual
  + >                                            Maior que
  + <                                            Menor que
  + >=                                         Maior ou igual que
  + <=                                         Menor ou igual que
  + !=                                          Diferente
  + Operadores Matemáticos:
    - +                                            Soma
    - -                                             Subtração
    - \*                                            Multiplicação
    - /                                            Divisão
    - %                                           Módulo

Vejamos mais alguns exemplos:

|  |  |
| --- | --- |
| **Expressão** | **Descrição** |
| $[${VALOR} + 5 | Irá retornar o VALOR + 5 |
| $[$OPERADORA} = 021] | Retornará 0 se verdadeiro e se falso retornará 1 |
| $[${OPER1} = 021 | ${OPER1} = 041 ] | Retornará 0 se o valor de OPER1 for 021 ou 041, retornará falso somente se ambos forem diferentes. |
| $[${OPER1} = 041 & $[EXTEN:4:2} = 11 | Retornará verdadeiro se o valor de OPER1 é igual 041 e os dois caracteres , a partir da 4º caractere de EXTEN for 11 retornará verdadeiro, caso um deles não seja o esperado, retornará falso |

Bom, já deu para ter ideia né...?

No primeiro exemplo um expressão simples que soma o conteúdo da variável VALOR com 5, O segundo exemplo irá comparar se a variável OPERADORA e igual a 21, no terceiro e quarto fazemos usos das operações Lógicas and e or. O terceiro exemplo verifica se o valor de OPER1 é igual a 021 ou 041, se um deles retornar como verdadeiro a sentença é considerado verdadeira, para ser falso, ambos têm que retornar falso. Já o ultimo compara se o Valor de OPER1 é igual a 041 e se o DDD é 11, ambos devem ser verdadeiros para a sentença ser verdadeira.

Para finalizar essa parte de aproveitamento de código, devemos ver agora macros.

**Usando Macros**

Macros são rotinas, ou melhor, sub-rotinas que são usadas mais de uma vez dentro do plano de discagem, evitando à reedificação de dados.  Facilitando operações,  organização, onde os códigos podem ser reutilizados.

Similar uma função, passamos o nome da rotina, e quantos argumentos como variáveis serão usadas, na hora de chamar basta fazer referencia a macros usando a quantidade de argumentos que foram especificados anteriormente, na criação da macro.

Vejamos a sintaxe da macro:

**Macro(nome da macro, arg1,arg2)**

Onde temos :

* Macro                                  A chamada para criação da Macro
* nome da macro               O nome da macro que está sendo criada, e será referenciado posteriormente.
* arg(n)                                  Argumentos que serão passados para macro

A macro deverá ser criado antes do uso nos contextos, por exemplo:

|  |
| --- |
| **[macro-exemplo]exten => s,1,Dial(${ARG1}/${ARG2},${ARG3})**    **[interno]**  **exten => \_4XXX,1,Macro(exemplo,SIP,${EXTEN},30)** |

Nesta macro acima estamos usando a aplicação Dial. Lembram-se ? Não ? Ok , vejamos a sintaxe da aplicação Dial:

**Dial(tecnologia/identificação[,timeout][,opções])**

**Para ver mais vejam mais em :**[**Introdução ao Asterisk – Parte 1**](http://www.stato.blog.br/wordpress/?p=522)**.**

Desta forma, temos uma macro chamada exemplo, que chama a aplicação Dial, com os argumentos padrão, Tecnologia ( SIP,AIX,DGV, etc), identificação da extensão , timeout ( opcional no Dial). Podemos facilmente ter digitado :

**Dia(SIP/${EXTEN},30)**

Não existe uma diferença muito grande, em digitar daquela forma e essa, pois usamos uma linha somente, agora imagine, várias linhas em vez de uma.

Vejamos outro exemplo:

|  |
| --- |
| **[macro-hello]**  **exten => s,1,Answer**  **exten => s,n,Playback(${ARG1})**    **[ramais]**  **exten => 1234,1,Macro(ola)** |

Desta forma fica um pouco mais claro o uso. Criamos à macro hello, que atenderá a chamada através da aplicação Answer, e depois tocará uma música que será passada como argumento na chamada da macro, através da aplicação Playback.

Um uso interessante poderia ser no uso de Status da discagem, para Ocupada, Não Responde etc. Veja abaixo os retornos que pode ser retornados pela variável DIALSTATUS:

* CHANUNAVAIL                               Canal indisponível. Em SIP, peer não registrado.
* ANSWER                             A chamada é atendida.
* NOANSWER                      Nenhuma resposta. . O número tocou por muito tempo, então a ligação expirou.
* BUSY                                    Sinal Ocupado.
* CANCEL                               Chamada for cancelada. O comando de discagem alcançou o seu número, mas o receptor desligou antes de atender.
* CONGESTION                   Congestionamento.  Este Status é geralmente é um sinal de que o número não é reconhecido.
* DONTCALL                         Modo de privacidade. Chamadas são rejeitadas.
* TORTURE                            Modo de privacidade. O receptor escolheu enviar a chamada para o menu.
* INVALIDARGS                  Erro de análise de argumentos na discagem.

Vejamos um exemplo abaixo:

|  |
| --- |
| **[macro-discar]exten =>s, 1, Dial(${ARG1}/$ARG2},${ARG3})**  **exten => s,n,Goto(call-${DIALSTATUS},1)**    **exten => call-NOANSWER,1,Playback(naoatende)**  **exten => call-NOANSWER,n,HangUP**    **exten => call-BUSY,1,Playback(ocupado)**  **exten => call-BUSY,n,HangUP** |

Vimos acima o uso de uma macro, usando a aplicação Dial, e seus argumentos. A novidade ai é aplicação **Goto.**Essa aplicação desvia o fluxo para outra extensão e prioridade. No caso estamos desviado para o valor call + o retorno da variável status, que poderá ser uma das vistas anteriores. Neste exemplo user somente NOANSWER e BUSY, mas pode ser implantada para todas.

Enfim, a aplicação desviará o fluxo no caso do NOANSWER, para call + NOANSWER, que será o contexto call-NOANSWER,  que rodará a aplicação PlayBack tocando o som naoatende, e por fim desligará a chamada.

Para fechar esse Post, veremos um exemplo para criação do plano de discagem, seguindo os obrigações abaixo:

1-      Criar as variáveis OPERADORA, SAIDAE1 e GRUPO, para os valores 041,DGV e g1.

2-      Criar os ramais SIP 2001,2003,2004,2005,2006,2007,2008,2101,2102,2103,2104, 2105, 2106, 2107.

3-      Criar os ramais AIX 3001,3002,3003,3004,3005,3008,3009

4-      Crie uma macro para verificar o status quando NOANWSER,BUSY e CHANUNAVAIL.

|  |
| --- |
| **[globais]**  **OPERADORA=041**  **SAIDAE1=DGV**  **GRUPO=g1**      **[macro-Discar]**  **exten => s, 1, Dial(${ARG1}/$ARG2},${ARG3})**  **exten => s,n,Goto(call-${DIALSTATUS})**  **exten => s,n,(call-NOASNWER),PlayBack(naoatende)**  **exten => s,n,HangUp()**  **exten => s,n,(call-BUSY),PlayBack(ocupado)**  **exten => s,n,HangUp()**  **exten => s,n,(call-CHANUNAVAIL),PlayBack(indisponivel)**  **exten => s,n,HangUp()**        **[interno]**      **exten  =>  \_200[1345678],1,Macro(Discar,SIP,${EXTEN},30,t)**  **exten =>  \_200[1345678],n,HangUp()**    **exten  => \_210[1-567],1,Macro(Discar,SIP,${EXTEN,},30,t)**  **exten =>  \_210[1-567],n,HangUp()**    **exten  => \_300[1-589],1,Macro(Discar,AIX2,${EXTEN},30 ,t)**  **exten  => \_300[1-589],n,HangUp()** |

Vamos começar comentando o contexto de interno, nele temos a criação dos ramais conforme solicitado, mas também temo o uso de expressões como [0-9] (de 0 a 9). Como uso de expressão regular, podemos colocar dentro do colchete vários caracteres , que neste caso são números, ou intervalo usando o hífen.

Já no uso do macro, temos na primeira extensão, uma discagem normal, caso não atendida seguirá o fluxo, sendo desviada pelo Goto para o contexto apropriado entre os status usados, que são os 3 acimas.

Estão,  depois de tudo isso, vimos que podemos facilitar muito o nosso Plano de Discagem , através de variáveis, mascaras e macros. O que ainda não vimos foram outras aplicações como Gotoif, Gosubif,GotoifTime, e mais umas trocentas….rsrs