OpenH323 Gatekeeper - The GNU Gatekeeper

Este manual é mantido por: *Leandro Caetano Gonçalves Lustosa* <leandro@nce.ufrj.br> v2.0.6, 24 Setembro 2003

Este é o Manual do Usuário para quem deseja compilar, instalar, configurar e monitorar o OpenH323 Gatekeeper - The GNU Gatekeeper.

Conteúdo

1 Introdução

1.1 Sobre

OpenH323 Gatekeeper - The GNU Gatekeeper <http://www.gnugk.org/> é um projeto de código-fonte aberto que implementa um gatekeeper H.323. Um gatekeeper provê serviços de controle para terminais H.323. Ele representa uma parte importante de muitas instalações de telefonia internet que são baseadas no padrão H.323.

De acordo com a Recomendação H.323, um gatekeeper deverá prover os seguintes serviços:

- Tradução de Endereço
- Controle de Admissão
- Controle de Banda Passante
- Gerenciamento da Zona Administrativa
- Sinalização de Controle das Chamadas
- Autorização de Chamadas
- Gerenciamento da Banda Passante
- Gerenciamento das Chamadas

The GNU Gatekeeper implementa muitas destas funções através do uso da biblioteca *OpenH323* <http://sourceforge.net/projects/openh323> que contém o stack completo do protocolo.

A Recomendação H.323 é um padrão internacional publicado pela *ITU* <http://www.itu.int/>. Sendo um padrão de comunicação de áudio, vídeo e dados através da Internet. Veja também o site de Paul Jones' *Padrão H.323* <http://www.packetizer.com/voip/h323/>.

Para uma descrição detalhada sobre a funcionalidade do gatekeeper, verifique este site *aqui* <http://www.iec.org/online/tutorials/h323/topic06.html>.

1.2 Direitos Autorais

1.3 Nome

O nome formal deste projeto é *OpenH323 Gatekeeper - The GNU Gatekeeper*, e o apelido é *OpenH323GK* ou *GnuGk*. Por favor não confunda este projeto de gatekeeper com outros.

Existem diversos projetos de gatekeeper de código-fonte aberto baseados na biblioteca OpenH323.

- *OpenGatekeeper* `<http://opengatekeeper.sourceforge.net/>` - desenvolvido pela Egoboo

  Um gatekeeper cheio de funcionalidade e disponível gratuitamente através de licença MPL. O projeto está inativo já faz algum tempo.

- *OpenGK* `<http://sourceforge.net/projects/openh323>` - desenvolvido pela Equivalence

  Somente suporta funcionalidades bem primárias.

- OpenH323 Gatekeeper - este gatekeeper aqui.

Ter diferentes gatekeepers com nomes muitos similares realmente confunde a maioria dos usuários. Como nosso "OpenH323 Gatekeeper"foi o primeiro a aparecer em cena, não é nossa culpa que os outros tenham escolhido nomes similares. Mas para fazer a distinção um pouco mais clara sem confundir as pessoas ainda mais, nós decidimos dar ao projeto um subtítulo "OpenH323 Gatekeeper - The GNU Gatekeeper"e começamos a usar o nome `gnugk` como o nome do executável.

1.4 Características

A versão 2.0.6 é uma release de conserto de bugs, e possui os seguintes melhoramentos:

FileAcct - Módulo de log de CDR em formato texto

Opção de linha de comando (-u) para alterar o proprietário do processo gatekeeper.

Aperfeiçoamento de filas virtuais (virtual queues).

Suporte completo a serviço pré-pago - tanto para terminais registrados (ARQ), quanto para terminais não registrados (Q.931 Setup). Agora a limitação de duração de chamada é completamente suportada pela autenticação via RADIUS.

A versão 2.0.5 é uma release de conserto de bugs, e adiciona os seguintes melhoramentos:

A primeira implementação de transferência de chamadas.

Autenticação via RADIUS H.235 (username/password) e alias.

Framework modular para contabilidade.

Módulo RADIUS para contabilidade.

Canal de sinalização autenticação/autorização (Q.931/H.225.0 Setup). Limite de duração de ligações (serviço pré-pago) pode ser controlado utilizando-se este esquema. Módulo RADIUS Q.931 de autenticação/autorização é suportado.

A versão 2.0.3 é uma release de conserto de bugs, e acrescenta apenas um pequeno melhoramente:

- Redireciona um Setup da chamada para o terminal especificado diretamente ao receber a mensagem Q.931 Facility com o campo reason configurado em **callForwarded**.
- Permite manualmente especificar terminais NATed.
- Adicionado um formulário simples para distribuir chamadas inbound. As chamadas para uma VirtualQueue podem ser encaminhadas para agentes por meio de uma aplicação de distribuição externa.

As novas funcionalidades adicionadas a versão 2.0.2 são:

- Adicionada a tecnologia Citron de NAT que permite transparentemente a penetração em NAT boxes. Suporte a múltiplos terminais e chamadas concorrentemente.
- O gatekeeper pode ficar situado atrás de uma NAT box e registrar terminais que estejam usando IPs públicos.
- Nova estrutura extendida fd_set, que permite ao gatekeeper suportar milhares de chamadas concorrentes no modo GK routed.
- Suporte a QoS ao adicionar o flag TOS para os pacotes RTP/RTCP.
- Autenticação na porta de status por usuário e senha.

E é claro, a maioria das funções da versão 2.0 também estão incluídas:

- A tabela de registro e a tabela de chamadas foram reprojetadas de maneira a ficarem thread-safe, e muito mais eficientes. Suportando dezenas de milhares de registros e milhares de chamadas concorrentes.
- Uma nova arquitetura GK-routed que suporta o encaminhamento de H.225.0/Q.931 e H.245 sem a criação de threads adicionais. Assim sendo, o limite no número de threads não restringirá o número de chamadas concorrentes.
- Suporte a função proxy H.323 ao encaminhar todos os canais lógicos, incluindo os fluxos RTP/RTCP dos canais de mídia e os canais de dados T.120. Os canais lógicos abertos pelo tunelamente H.245 e o procedimento fast-connect também são suportados. No modo proxy, não existe tráfego direto entre as entidades chamador e o chamado. Desta forma, isto se torna útil se você tiver algum terminal usando endereço IP privado atrás de um NAT box querendo se comunicar com outros terminais usando endereços IP públicos fora do NAT box.
- Suporte a clusters de gatekeepers pela troca de mensagens LRQ/LCF/LRJ (função de vizinhança). Se o destino de um LRQ recebido for desconhecido, o GnuGk pode redireciona-lo para o próximo hop. Assim sendo, o GnuGk pode trabalhar como um directory gatekeeper.
- Suporte a vários métodos de autenticação para as requisições RAS, incluindo senha H.235 (MD5 & SHA-1 e CAT), casamento de padrão por IP ou por prefixo. Bancos de dados MySQL e LDAP também são suportados como backend para a autenticação.
- Suporte a gatekeepers alternativos para redundância e balanceamento de carga. Se o GnuGk estiver sobrecarregado, os terminais podem ser redirecionados para outros gatekeepers.
- Pode trabalhar como um endpoint (gateway ou terminal) ao registrar=se com um gatekeeper pai. Com este recurso, criar hierarquias de gatekeepers é fácil.
- Monitora e controla o GnuGk via porta de status TCP, incluindo registros e estatisticas da chamada.
- Imprime um CDR(registro detalhado da chamada) na porta de status para sistemas de contabilização de backend. O CDR contém identificador da chamada, os endereços IPs do chamador e chamado, o inicio e o fim da ligação e a duração da mesma.
- Muitas configurações podem ser refeitas em tempo de execução, sem precisar desligar o gatekeeper. O GnuGk faz a re-leitura das configurações ao receber o comando '`reload`' via porta de status, ou ao receber um sinal `HUP` (plataforma Unix).

1.5 Download

A versão mais recente/estável e a versão de desenvolvimento estão disponíveis na *página de download* <http://www.gnugk.org/h323download.html>.

A última versão do código fonte está disponível no CVS em *Sourceforge* <http://sourceforge.net/cvs/?group_id=4797> (*Web-GUI* <http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/openh323gk/>). Cuidado - esta versão de desenvolvimento é a versão que esta sendo modificada.

Você também pode fazer o download dos executáveis a partir de *Sourceforge* <http://sourceforge.net/project/showfiles.php?group_id=4797>. Somente algumas versões estão disponíveis como executáveis.

1.6 Lista de Discussão

Existem duas listas de discussão para o projeto, uma para os desenvolvedores e uma para os usuários.

As questões genéricas do usuários deveriam ser enviadas para *lista de discussão dos usuários* <mailto:Openh323gk-users@sourceforge.net>. Voce pode encontrar a lista dos emails arquivados desta lista *aqui* <http://sourceforge.net/mailarchive/forum.php?forum_id=8549>. Para entrar nesta lista de discussão, clique *aqui* <http://lists.sourceforge.net/lists/listinfo/openh323gk-users>.

Para relatar problemas ou submeter bugs/patches, envie emails para a *lista de discussão dos desenvolvedores* <mailto:Openh323gk-developer@sourceforge.net>. Os emails arquivados desta lista estão *aqui* <http://sourceforge.net/mailarchive/forum.php?forum_id=3079>. Por favor envie as questões de nível usuário para a lista de discussão de usuários enquanto que mantenha esta lista para o desenvolvimento somente! Se você quiser contribuir com o projeto, por favor *participe da lista de discussão dos desenvolvedores* <http://lists.sourceforge.net/lists/listinfo/openh323gk-developer>.

Note: Por favor, não envie suas questões de maneira privada diretamente para um desenvolvedor em particular. Nós normalmente estamos ocupados. Nós **não** queremos ser seus consultores particulares, ao menos que você esteja disposto a pagar. Envie seus problemas para a lista de discussão pública apropriada desta forma todo mundo poderá ajudá-lo.

Por favor também não envie problemas específicos do GnuGk na lista de discussão do OpenH323, ou vice-versa. Ou você poderá ser definitivamente ignorado. Eles são projetos diferentes, embora fortemente relacionados. Os desenvolvedores também são diferentes, embora eles possam trabalhar juntos de alguma maneira.

Antes de envia um email, tenha certeza de que voce leu os documentos relacionados com o assunto cuidadosamente. Descreva seus problemas claramente e precisamente. Nos mostre as mensagens de erro ou logs se existir algum.

1.7 Contribuidores do projeto

O coordenador do projeto atualmente é *Jan Willamowius* <http://www.willamowius.de/> <*jan@willamowius.de*>

As principais funções e recursos da versão 2.0 foram desenvolvidas por Chih-Wei Huang <*cwhuang@linux.org.tw*> e *Citron Network Inc.* <http://www.citron.com.tw/>, incluindo as tabelas de registro e chamadas thread-safe, nova arquitetura de GK-routed, proxy H.323, autenticação H.235 e backend MySQL.

Uma equipe da *mediaWays* <http://www.mediaways.net/> está trabalhando em um subsistema de banco de dados LDAP, reescrevendo mecanismos de encaminhamento mais avançados.

A versão inicial do gatekeeper foi desenvolvida por Xiang Ping Chen, Joe Metzger e Rajat Todi.

2 Compilando e Instalando

2.1 Compilando o Gatekeeper

Para gerar uma release do gatekeeper você precisará de pelo menos a PWLib 1.2 e OpenH323 1.8 ou maior (as versões recomendadas são PWLib 1.4.11 ou mais recente e Openh323 1.11.7 ou mais recente). A versão de desenvolvimento do gatekeeper normalmente requer a versão mais recente disponível do OpenH323. Estas bibliotecas estão disponíveis em *Página de Download do OpenH323* <http://sourceforge.net/projects/openh323>. Verifique as instruções sobre *como compilar o código OpenH323* <http://www.voxgratia.org/docs/faq.html>.

Ordem de bibliotecas para compilar:

1. PWLib (versões release + debug)
2. OpenH323
3. OpenH323 aplicativo de teste (não é totalmente necessário, é apenas para ter certeza que está tudo funcionando bem)
4. E finalmente o Gatekeeper

No Unix digite o `make debug` ou `make opt` no diretório do gatekeeper para compilar as versões de debug ou release, respectivamente. Use o comando `make both` para compilar ambas as versões. Note que é necessário o uso do GCC 2.95.2 ou superior. Versões mais antigas podem não funcionar. Uma boa prática é executar um `make debugdepend` ou `make optdepend` no diretório do gatekeer antes de iniciar a compilação atual (make debug ou make opt) - estes comandos compilam a lista de dependências apropriadas. Então posteriormente, quando você atualizar seus fontes via CVS, todos os arquivos afetados serão recompilados. Caso contrário, você pode gerar um Gatekeeper parcialmete compilado com os headers antigos e parcialmente compilado com os headers atualizados - o que é uma coisa muito ruim.

No Windows apenas abra o projeto disponibilizado (`gk.dsw`) e compile no ambiente Microsoft Visual C++ 6.0 ou 7.0 (Visual C++ 5.0 é muito antigo).

Desde a versão 2.0, o Gatekeeper suporta os bancos de dados de backend MySQL e LDAP. Se você não quiser o suporte MySQL, você deve configurar a variável de ambiente NO_MYSQL antes de compilar:

```
$ NO_MYSQL=1 make both
```

Para desabilitar o suporta ao LDAP:

```
$ NO_LDAP=1 make both
```

Ou para desabilitar ambos use

```
$ NO_MYSQL=1 NO_LDAP=1 make both
```

Desde a versão 2.0.1 o Gatekeeper implementou uma estrutura fd_set extendida que permite ao Gatekeeper suportar milhares de chamadas concorrentes no modo Gk-routed. Para habilitar este recurso, exporte a variável de ambiente LARGE_FDSET para o número máximo de descritores de arquivos. Por exemplo,

```
$ LARGE_FDSET=16384 make opt
```

Desde a versão 2.0.4 o Gatekeeper implementa o protocolo cliente do Radius que permite o registro/admissão, autenticação e autorização em servidores Radius. Esta característica é habilitada por padrão. Para desabilitar a compilação destes módulos radius, configurar a variável de ambiente NO_RADIUS antes de compilar:

```
$ NO_RADIUS=1 make both
```

Desde a versão 2.05 o Gatekeeper é capaz de fazer contabilização. Atualmente, somente os módulos de contabilização RADIUS e arquivo texto estão disponíveis. A contabilização ainda é uma característica experiemental (apesar de funcionar corretamente), e dessa forma, não é compilada por padrão. Para habilitar a contabilização, configurar a variável de ambiente HAS_ACCT antes de compilar:

```
$ HAS_ACCT=1 make both
```

2.2 Instalando o Gatekeeper

Não existe nenhum procedimento especial para a instalação. Apenas copie o executável no diretório que você quiser e crie um arquivo de configuração para ele. Existem diversos exemplos de configuração no subdiretório `etc/` na árvores dos arquivos fonte. Veja a seção 4.2 (Arquivo de Configuração) para explicações detalhadas.

Por exemplo, na plataforma Linux x86, o executável gerado `gnugk` está localizado no subdiretório `obj_linux_x86_r/`. Você deve copiar ele para `/usr/sbin/`, criar um arquivo de configuração em `/etc/gnugk.ini` e iniciar o processo usando o comando

```
$ /usr/sbin/gnugk -c /etc/gnugk.ini -o /var/log/gnugk.log -ttt
```

Veja a seção 4.1 (Opções da Linha de Comando) para maiores detalhes.

2.3 Binários Pré-Compilados

Se você não quiser compilar o gatekeeper a partir do código fonte, existem diversos "pacotes"pré-compilados disponíveis em *SourceForge* `<http://sourceforge.net/project/showfiles.php?group_id=4797>`. Nem todas as versões estarão disponíveis como binários - verifique quais estão disponíveis.

Pacotes Red Hat (.rpm)

Download os RPMs e entre com os seguintes comandos como `root`, substitua com o nome do arquivo que você fez download.

```
$ rpm -Uvh gnugk-x.x.x.rpm
```

Pacotes Debian (.deb)

Se você estiver usando uma versão 'estável' do Debian, você pode instalar o gatekeeper pelo uso do seguinte comando como `root`:

```
$ apt-get install openh323gk
```

3 Iniciando o uso (Tutorial)

3.1 Um primeiro experimento

Para ver se todos os componentes estão funcionando, use 2 estações de trabalho Linux, ambas conectadas a uma LAN. É preciso ter instalado previamente pelo menos a versão 1.1 do OpenH323 e OhPhone. Na primeira máquina execute o gatekeeper e ohphone (em diferentes consoles):

```
jan@machine1 > gnugk -ttt
```

Agora que o gatekeeper está executando no modo direto. A opção "-ttt"diz ao gatekeeper que ele deve gerar um certa quantidade de logs na saída de console (você pode direcionar a saída para um arquivo com a opção "-o logfile").

```
jan@machine1 > ohphone -l -a -u jan
```

Esta configuração (-l) faz com que o OhPhone se prepare para receber chamadas e aceite automaticamente estas (-a). O próximo parâmetro especifica que o usuário será registrado como jan com o gatekeeper, este último sendo detectado automaticamente. (Se a auto detecção falhar por alguma razão use o parâmetro "-g 1.2.3.4"para especificar o endereço IP do gatekeeper desejado.)

Na segunda máquina execute o ohphone:

```
peter@machine2 > ohphone -u peter jan
```

Esta segunda instância do OhPhone se registra através da auto-detecção como sendo o usuário peter e tenta realizar uma ligação para o usuário jan. O gatekeeper irá resolver a partir do apelido do usuário, o seu endereço IP com o qual o usuário jan esteja registrado (machine1 neste caso) e desta forma o OhPhone irá realizar uma chamada para a outra instância do OhPhone da machine1.

A primeira instância do OhPhone irá aceitar automaticamente esta chamada e então Peter e Jan poderão conversar.

3.2 Usando a interface de Status para "espiar"o gatekeeper

Agora, para verificarmos que as mensagens foram manipuladas pelo gatekeeper. Na console da machine1 usaremos o telnet para conectar com o gatekeeper:

```
jan@machine1 > telnet machine1 7000
```

Muito provavelmente nós veremos a mensagem "Access forbidden!"(Acesso Proibido), porque não é permitido a qualquer um "espiar"o gatekeeper.

Assim sendo, precisamos criar um arquivo básico chamado gatekeeper.ini e colocar ele no mesmo diretório de onde iniciamos o gatekeeper. O arquivo gatekeeper.ini vai conter somente 4 linhas:

```
[Gatekeeper::Main]
Fourtytwo=42
[GkStatus::Auth]
rule=allow
```

Pare o gatekeeper com Ctrl-C e reinicie. Quando nós tentarmos usar o telnet novamente, nós então nos conectaremos com o gatekeeper. Repetindo o primeiro experimento onde Peter chama Jan e vê quais mensagens são manipuladas pelo gatekeeper no modo direto. Existe um conjunto de comandos que pode ser colocado na sessão telnet de status: Digite "help"para visualizar estes comandos. Para finalizar a sessão telnet com o gatekeeper digite "quit"e pressione Enter.

3.3 Iniciando o gatekeeper em modo GK-routed

Iniciar o gatekeeper no modo GK-routed significa que o gatekeeper usará o "modo de sinalização roteada pelo gatekeeper"para todas as chamadas. Neste modo o gatekeeper todas as mensagens passam necessariamente pelo gatekeeper o que dá muito mais controle sobre as chamadas.

```
jan@machine1 > gnugk -r
```

Agora que nós estamos executando o gatekeeper no modo GK-routed. Faça Telnet na porta de status e realize uma chamada para ver que mensagens são manipuladas pelo gatekeeper.

Observe que todos os pacotes de mídia (audio e video) ainda são enviados diretamente entre os endpoints (ou seja, as 2 instancias do ohphone).

Como a sinalização roteada pelo gatekeeper é muito mais complicada é provável usar este modo para traçar bugs, e manter o registro de todas as mensagens. ;-)

3.4 Um PBX virtual: Desconectado chamadas

Até agora o gatekeeper agiu como um mecanismo de resolução de nomes para endereços IP. Esta é uma funcionalidade importante, mas não é muito empolgante.

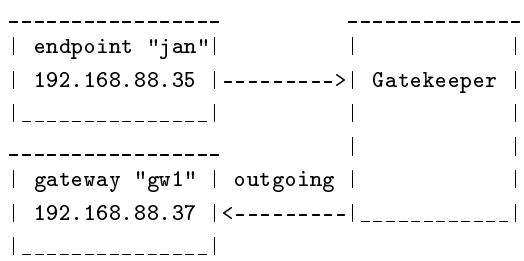
Como o gatekeeper tem uma série de controles sobre as chamadas, ele terminar elas por exemplo. Assim, quando nós estivermos conectados na porta de status, nós poderemos gerar a lista de todas as chamadas ativas com o comando "PrintCurrentCalls". Para finalizar uma chamada, nós podemos digitar "Disconnectip 1.2.3.4"para desconectar um dos terminais H.323.

Alguém poderia escrever um script simple que se conectaria a porta de status e verificaria todas as chamadas ativas e terminaria aquelas que excedem-se 5 minutos, desta forma nenhum usuário iria abusar dos recursos.

Este mecanismo também poderia ser usado para transferir chamadas entre usuários ou encaminhar chamadas. (Mas isso ainda não está implementado no gatekeeper.)

3.5 Roteando chamadas através de um gateway para alcançar usuários externos

Sem usar um gateway você pode chamar somente outras pessoas com outro telefone IP através da Internet. Para alcançar pessoas com telefones normais você deve usar um gateway.



O gatekeeper deve conhecer quais prefixos serão suportados pelo gateway e quais números podem ser alcançados diretamente. Usando a seção [RasSrv::GWPrefixes] do arquivo de configuração é explicitado quais prefixos de números que poderão ser roteados pelo gateway.

```
[RasSrv::GWPrefixes]
gw1=0
```

Esta entrada do exemplo diz que o gatekeeper irá encaminhar todas as chamadas com número E.164 numbers iniciando com 0 serão encaminhadas para o gateway registrado com o apelido (alias) H.323 "gw1". Se não houver nenhum gateway com este alias a chamada irá falhar. (Observe que você deve usar o apelido para identificar o gateway - você não pode simplesmente configurar no gatekeeper o endereço IP do gateway.)

3.6 Reescrevendo números E.164

Quando usamos um gateway frequentemente temos que usar um plano de numeração interna e rescrever esta numeração discada antes de ser enviada para dentro da rede telefonica. Portanto, nós podemos fazer uso da seção 4.2.6 (RasSrv::RewriteE164) para configurar esta funcionalidade.

Exemplo: Nós queremos chamar o número 12345 com o nosso telefone IP e isto será interpretado automaticamente como uma ligação para o número 08765 que está atrás do gateway "gw1".

```
[RasSrv::GWPrefixes]
gw1=0

[RasSrv::RewriteE164]
12345=08765
```

4 Usando o Gatekeeper (Referencia)

O comportamento do gatekeeper é completamente determinado pelas opções da linha de comando e pelo arquivo de configuração. Algumas opções da linha de comandos podem sobrepor configurações feitas no arquivo .ini. Por exemplo, a opção -l sobrepõe a configuração TimeToLive presente no arquivo de configuração.

4.1 Opções da Linha de Comando

Quase todas as opções tem um formato curto e um longo, ex., -c é o mesmo que -config.

4.1.1 Básico

`-h -help`

Mostra todas as opções disponíveis e sai do programa.

`-c -config filename`

Especifica qual arquivo de configurações será usado.

`-s -section section`

Especifica qual seção principal usar do arquivo de configuração. O default é [Gatekeeper::Main].

-i -interface IP

Especifica a interface (IP) que o gatekeeper vai ficar ouvindo. Você pode deixar opção de lado para que o gatekeeper tente determinar automaticamente qual IP ele ficará ouvindo, ao menos que você realmente deseje que o gatekeeper faça "bind"a partir de um IP especifico.

-l -timetolive n

Especifica o tempo de vida em segundos do registro do terminal. Ele sobrepõe a configuração TimeToLive no arquivo de configuração. Veja 4.2.1 (ttl) para maiores detalhes.

-b -bandwidth n

Especifica o total de banda passante disponível para o gatekeeper. Sem a especificação desta opção, o gerenciamento de banda é desabilitado por default.

-pid filename

Especifica o arquivo pid, somente válido para versão Unix.

Especifica o total de banda passante disponível para o gatekeeper. Sem a especificação desta opção, o gerenciamento de banda é desabilitado por default.

-user name

Executa o processo gatekeeper como este usuário, somente válido para versão Unix.

4.1.2 Modos de Sinalização Gatekeeper

As opções nesta subseção sobrepõem as configurações na 4.2.2 (seção [RoutedMode]) do arquivo de configuração.

-d -direct

Use esta opção para configurar o modo de sinalização direta de chamadas.

-r -routed

Use esta opção para configurar o modo de sinalização roteada pelo gatekeeper.

-rr -h245routed

Usar o modo de sinalização roteada pelo gatekeeper e encaminhe também o canal de controle H.245.

4.1.3 Informações de Depuração

-o -output filename

Escrever em um arquivo de log especificado todo o trace do programa

-t -trace

Configurar o número de "verbose"(profundidade da depuração) para o trace. Quanto mais -t você adicionar, mais verbose (informações de debug) será exibido. Por exemplo, usando -ttttt você está configurando o nível de trace para 5.

4.2 Arquivo de Configuração

O arquivo de configuração é um arquivo de texto normal. O formato básico é:

```
[Section String]
Key Name=Value String
```

Comentários são criados com o caracter hash (#) ou um ponto e virgula (;) no começo de uma linha.

O arquivo complete.ini contém todas as seções disponíveis do GnuGk. Para muitos casos não faz sentido usar todas estas seções de uma só vez. Este arquivo é só uma referencia de uma coleção de exemplos para diversas configurações.

O arquivo de configuração pode sofrer alterações em tempo de execução. Uma vez modificado o arquivo de configuração, você deve digitar o comando reload via porta de status, ou enviar um sinal HUP para o processo gatekeeper no Unix. Por exemplo,

```
kill -HUP ‘cat /var/run/gnugk.pid‘
```

Note É especificado em algumas seções da versão 2.0 do GnuGk que [RasSrv::*], deve ser usado, enquanto que [RasSvr::*] pode ser usado por outras. Esta inconsistencia pode confundir os usuários. Na versão 2.0.1 todas as seções são corrigidas para [RasSrv::*]. Assim se você atualizar uma versão 2.0 para uma versão mais nova, não esqueça de trocar os nomes das seções, ou o GnuGk se recusará a iniciar.

4.2.1 Seção [Gatekeeper::Main]

- Fourtytwo=42
  Default: N/A Esta configuração é usada para testar a presença do arquivo de configuração. Se ela não for encontrada, uma mensagem de warning será mostrada. Logo, verifique sempre se esta configuração esta presente em todos os arquivos de configuração.
- Name=OpenH323GK
  Default: OpenH323GK Identificador do Gatekeeper. O Gatekeeper somente responderá os GRQs que forem direcionados para este ID e o usará em todas das mensagens com os terminais.
- Home=192.168.1.1
  Default: 0.0.0.0 O gatekeeper irá receber requisicões a partir deste endereço IP. Por default, o gatekeeper "ouve"todas as interfaces de seu host. Você pode deixar esta opção sem configuração, ao menos que se queira configurar o gatekeeper para realizar o bind de um endereço IP específico.
- NetworkInterfaces=192.168.1.1/24,10.0.0.1/0
  Default: N/A Especifica as interfaces de rede do gatekeeper. Por default o gatekeeper irá detectar as interfaces do seu host automaticamente. Existem duas situações onde você pode querer esta opção. Uma é se a deteção automática falhar, e a outra é se o estiver atrás de um NAT box e quer permitir que terminais com endereços IP públicos se registrem neste gatekeeper. Neste caso você deve configurar a opção somente se o gatekeeper estiver sendo executado sob a NAT box.

- EndpointIDSuffix=_gk1
  Default: _endp O gatekeeper irá associar um identificador único a cada terminal registrado. Esta opção pode ser usada para especificar um sufixo a ser anexado como identificador do terminal. Isto é útil somente quando usamos mais de um gatekeeper.

- TimeToLive=300
  Default: -1 Um registro de terminal com um gatekeeper deveria ter um ciclo de vida limitado. O gatekeeper especifica a duração de registro de um terminal ao incluir o campo timeToLive na mensagem RCF. Após o tempo especificado, o registro terá expirado. O terminal deve periodicamente enviar mensagem RRQ tendo o bit keepAlive ativado antes que o tempo expire. Tal mensagem deve incluir uma quantidade mínima de informações conforme descrito em H.225.0. Isto é chamado de lightweight RRQ.

  Esta configuração especifica o tempo de vida do temporizador em segundos até que o registro expire. Observe que o terminal pode requisitar um tempo de timeToLive menor em sua mensagem de RRQ para o gatekeeper. Para evitar uma sobrecarga de mensagens RRQ, o gatekeeper automaticamente ajusta este timer para 60 seconds se o valor recebido for inferior a este!

  Após o tempo de permanência do registro, o gatekeeper irá enviar subsequentemente duas mensagens IRQ para pesquisar se o terminal ainda está vivo. Se o terminal responder com um IRR, o registro será estendido. De outro modo, se não receber esta confirmação, o gatekeeper enviará um URQ com a razão ttlExpired para o terminal. O terminal então deverá re-registrar com o gatekeeper usando uma mensagem RRQ completa.

  Para desabilitar esta característica, configure em -1.

- TotalBandwidth=100000
  Default: -1 A Banda Passante Total disponível para os endpoints.

- RedirectGK=Endpoints $>$ 100 $||$ Calls $>$ 50
  Default: N/A Esta opção permite a você redirecionar os terminais para gatekeepers alternativos quando o gatekeeper estiver sobrecarregado. Por exemplo, com a configuração descrita acima, o gatekeeper irá rejeitar um pedido de RRQ se os terminais registrados excederem 100, ou rejeitará um ARQ se as chamadas concorrentes excederem 50.

  Além do mais, você pode explicitamente redirecionar todos os terminais pelo uso da opção temporary ou permanent. O gatekeeper irá retornar uma mensagem de rejeição RAS com uma lista de gatekeepers alternativos definidos em AlternateGKs. Observe que um redirecionamento permanent fará com que os terminais não se registrem mais neste gatekeeper. Atenção, observe que esta funcionalidade somente tem efeito em terminais compatíveis com a versão 4 do H.323.

- AlternateGKs=1.2.3.4:1719:false:120:OpenH323GK
  Default: N/A É permitida a existência de outro gatekeeper para prover redundância. Isto é implementado de uma maneira active-active. Atualmente, você poderá entrar em uma situação (válida!) onde alguns terminais estejam registrados com o primeiro e outros estejam registrados com o segundo gatekeeper. Você poderia inclusive ser capaz de usar os dois gatekeepers de maneira round-robin para dividir a carga (embora, isto não tenha sido testado). Se você notar uma mensagem, "primary GK"isto referencia o gatekeeper que você está configurando e "alternate GK"significa o outro. O primary GK inclui um campo em RCF que estabelece qual é o IP alternativo e o identificador de gatekeeper a ser usado. Mas, por outro lado o GK alternativo

precisa conhecer cada registro feito com o GK primário ou senão ele poderá rejeitar ligações. Além do mais, nosso gatekeeper pode encaminhar cada RRQ recebido para o endereço IP alternativo.

A opção de configuração AlternateGKs especifica o campo contido no RCF do GK primário. O primeiro e segundo campos desta string define para onde (IP, port) encaminhar. O terceiro elemento diz aos terminais se eles precisam se registrar no GK alternativo antes de realizar ligações. Normalmente ele não vão precisar pois replicamos os RRQs, de modo que eles já estão registrados também com o alternate GK. O quarto campo especifica a prioridade deste GK. Quanto menor o valor melhor, normalmente o GK primário é considerado com prioridade 1. O último campo especifica o identificador deste gatekeeper alternativo.

- SendTo=1.2.3.4:1719
  Default: N/A Embora esta informação esteja contida no AlternateGKs, você pode ainda assim especificar para qual endereço redirecionar os RRQs. Esto pode ser diferente do AlternateGK, portanto ela é usada como uma opção separada (pense em ambientes residenciais com muitas máquinas).

- SkipForwards=1.2.3.4:5.6.7.8
  Default: N/A Para evitar o encaminhamento circular, você não deveria redirecionar RRQs que você recebeste de outros GK (esta regra é verdadeira para tanto, GK primário e alternativo). Dois mecanismos são usados para identificar se uma requisição está sendo encaminhada. O primeiro mecanismo verifica um flag no RRQ. Como poucos endpoints implementam isto, nós precisamos de um segundo mecanismo, mais confiável. Portanto, especifique os IPs dos outros gatekeepers nesta lista.

- StatusPort=7000
  Default: 7000 Porta de Status para monitorar o gatekeeper. Veja 5 (esta seção) com os detalhes.

Muitos usuários quase nunca precisarão trocar quaiquer dos próximos valores. Eles são usados principalmente para testes ou aplicações mais sofisticadas.

- UseBroadcastListener=0
  Default: 1 Define se o canal de broadcast para RAS estará funcionando. Isto requer que seja feito o *bind* de todas as interfaces daquela máquina, assim sendo se você quiser criar multiplas instâncias de gatekeeper em uma mesma máquina você deverá desligar esta opção.

- UnicastRasPort=1719
  Default: 1719 O identificador TSAP do canal RAS unicast.

- MulticastPort=1718
  Default: 1718 O identificador TSAP do canal RAS multicast.

- MulticastGroup=224.0.1.41
  Default: 224.0.1.41 O grupo multicast do canal RAS.

- EndpointSignalPort=1720
  Default: 1720 Porta default do canal de sinalização de chamadas dos endpoints.

- ListenQueueLength=1024
  Default: 1024 Tamanho da fila de requisições de conexão TCP.

- SignalReadTimeout=1000
  Default: 1000 Tempo em milisegundos para a leitura expirar dos canais de sinalização (Q931).
- StatusReadTimeout=3000
  Default: 3000 Tempo em milisegundos para a leitura expirar do canal de status.
- StatusWriteTimeout=5000
  Default: 5000 Tempo em milisegundos para a escrita expirar no canal status

4.2.2 Seção [RoutedMode]

As mensagens de sinalização podem ser redirecionadas de duas maneiras. O primero método é o Direct Endpoint Call Signalling, neste caso as mensagens de sinalização de estabelecimento de chamada serão trocadas diretamente pelos endpoints. O segundo método é o de GK-Routed. Neste método, as mensagens de sinalização de estabelecimento de chamada serão encaminhadas através do gatekeeper para os endpoints. A escolha de qual metodo será usado deve feito pelo administrador do gatekeeper.

Quando a sinalização Gatekeeper Routed for usada, o administrador também poderá escolher se deseja encaminhar as mensagens de controle H.245 e os canais lógicos.

Caso I.

O gatekeeper não realiza encaminhamento. Os canais de controle H.245 e os canais lógicos são estabelecidos diretamente entre os endpoints.

Caso II.

O canal de controle H.245 é redirecionado entre endpoints através do gatekeeper, enquanto os canais lógicos são estabelecidos diretamente entre os endpoints.

Caso III.

O gatekeeper encaminha o canal de controle H.245, assim como todos os canais lógicos, incluindo RTP/RTCP para áudio e vídeo, e o canal T.120 para dados. Neste caso, nenhum tráfego será passado diretamente entre os endpoints. Isto é normalmente conhecido como H.323 Proxy, que na verdade poderia ser especificado como um gateway H.323-H.323.

Esta seção define as opções do modo GK-routed (caso I & II). O recurso proxy está definido na 4.2.3 (próxima seção). Todas as configurações desta seção serão afetadas pela reinicio do gatekeeper.

- GKRouted=1
  Default: 0 Habilitar ou não o modo GK-routed.
- H245Routed=1
  Default: 0 Habilitar ou não o encaminhamento do canal de controle H.245 pelo GK. Isso só funciona se GKRouted=1.
- CallSignalPort=0
  Default: 1721 A porta de sinalização do gatekeeper. A porta padrão é 1721. Nós não usamos a bem conhecida porta 1720 de modo que é possível rodar um terminal H.323 na mesma máquina do gatekeeper. Você pode configurar com a opção 0 para deixar o gatekeeper escolher arbitrariamente uma porta.

- CallSignalHandlerNumber=2
  Default: 1 O número de manipuladores de sinalização de chamada. Você poderá aumentar este numero em um gatekeeper com alta carga. O número somente pode ser incrementado em tempo de execução. Se você tiver uma máquina multiprocessada, é possível configurar este número com o número de CPUs.

- AcceptNeighborsCalls=1
  Default: 1 Com este recurso habilitado, a thread de sinalização de chamada aceitará chamadas sem um CallRec pré-existente na CallTable, dado que o endpoint correspondente ao destinationAddress na mensagem Setup possa ser encontrado na RegistrationTable, e a parte chamadora esteja na lista de vizinhos ou seja de um GK pai. O gatekeeper também irá colocar seu próprio endereço de sinalização na LCF correspondente a uma LRQ. Isto significa que a sinalização da chamada será roteada para GK2 em chamadas GK-GK. Como resultado disso, os CDRs no GK2 poderão ser gerados corretamente mostrando o tempo de conexão, ao invés de "sem conexão".

- AcceptUnregisteredCalls=1
  Default: 0 Com este recurso habilitado, o gatekeeper aceitará chamadas de quaisquer endpoints não registrados. Entretanto, isso aumenta os riscos na segurança. Cuidado no uso deste recurso.

- RemoveH245AddressOnTunneling=1
  Default: 0 Alguns terminais enviam o h245Address na UUIE do Q.931 mesmo que o h245Tunneling esteja configurado com TRUE. Isto poderá causar problemas de interoperabilidade. Se a opção for TRUE, o gatekeeper irá remover o h245Address quando a flag h245Tunneling estiver TRUE. Isto força a parte remoto a permanecer no modo de tunelamento.

- RemoveCallOnDRQ=0
  Default: 1 Com esta opção desabilitada, o gatekeeper não desconectará uma chamada se ela receber um DRQ. Isto evita problemas potenciais de "race conditions"quando o DRQ sobrepõe um Release Complete. Isto só tem sentido no modo GK-routed visto que no modo direto, o único mecanismo de sinalização de fim de chamada é pelo uso de DRQ.

- DropCallsByReleaseComplete=1
  Default: 0 De acordo com a Recomendação H.323, o gatekeeper deveria derrubar uma chamada pelo envio do comando RAS DisengageRequest para os endpoints. Entretanto, alguns endpoints ruins podem ignorar este comando. Quando esta opção estiver em on, o gatekeeper enviará o Q.931 Release Complete ao invés do RAS DRQ para ambos os endpoints para forçar eles a derrubar a chamada.

- SendReleaseCompleteOnDRQ=1
  Default: 0 Ao desconectar, o endpoint envia tanto Release Complete da especificação H.225/Q.931 e DRQ da especificação RAS. Pode acontecer do DRQ ser processado primeiro, fazendo com que o gatekeeper feche o canal de sinalização de chamada, desta forma prevenindo o Release Complete de ser redirecionado para o outro endpoint. Pois o gatekeeper fecha o canal TCP com o destino, alguns endpoints (ex. Cisco CallManager) não derrubam a chamada mesmo que o canal de sinalização tenha sido fechado. Isto resulta em telefones que ficam tocando se o chamador colocar no gancho antes do chamado atender. Configurando este parâmetro com 1 faz com que o gatekeeper sempre envie Release Complete para ambos os endpoints antes de fechar o canal quando ele receber DRQ de uma das partes envolvidas.

- SupportNATedEndpoints=1
  Default: 0 Permitir ou não que um endpoints atrás de uma NAT box se registre com o gatekeeper. Se sim, o gatekeeper irá traduzir o endereço IP especificado nos canais Q.931 e H.245 para o endereço do NAT box.

  Desde a versão 2.0.2, o GnuGk suporta chamadas que saiam do ambiente NAT (de um endpoint que esteja atrás do NAT para a rede IP pública) diretamente sem quaisquer modificações necessárias dos endpoints ou da NAT box. Para tanto, é preciso apenas registrar o endpoint com o GnuGk e realizar a ligação a qualquer momento.

- ScreenDisplayIE=MyID
  Default: N/A Modificar o campo DisplayIE do Q.931 (que mostra o usuário chamador) para o valor especificado.

- ScreenCallingPartyNumberIE=0965123456
  Default: N/A Modificar o campo CallingPartyNumberIE do Q.931 para o valor especificado.

- ForwardOnFacility=1
  Default: 1 Se verdadeiro, ao receber a Q.931 Facility com reason configurada para callForwarded, o gatekeeper irá redirecionar o Setup da chamada diretamente para o endpoint encaminhado, ao invés de passar a mensagem de volta para o chamador. Se tiver endpoints defeituosos que não saibam manipular a Q.931 Facility com reason callForwarded, habilite esta opção.

- ShowForwarderNumber=0
  Default: 0 Reescrever ou não o endereço da parte chamada pelo número redirecionado. Isto é normalmente usado para fins de bilhetagem. Somente válido se ForwardOnFacility=1.

- Q931PortRange=20000-20999
  Default: 0 (random) Especificar uma faixa de portas TCP dos canais de sinalização Q.931. Observe que esta faixa pode limitar o número de chamadas concorrentes.

- ConnectTimeout=60000
  Default: 180000 Tempo limite de espera em milisegundos antes de se remover chamadas não completadas da tabela de chamadas. Este é um tempo de segurança para não permitir que chamadas fiquem penduradas na tabela de chamadas indefinidamente. Note que tornar este valor muito curto pode resultar em chamadas sendo canceladas antes de serem atendidas.

4.2.3 Seção [Proxy]

A seção define os recursos de H.323 proxy. Isto indica que o gatekeeper irá redirecionar todo o tráfego entre os endpoints chamador e chamado, de modo que não exista nenhum tráfego direto entre os dois endpoints. Assim sendo isso é muito útil para ser usado em um ambiente com IP privado atrás de um NAT box e alguns endpoints usando endereços IP públicos fora do NAT.

O gatekeeper pode fazer o proxy dos canais lógicos de RTP/RTCP (audio e video) e T.120 (data). Os canais lógicos abertos com os procedimentos de fast-connect ou tunelamento H.245 também são suportados.

Observe que para o proxy funcionar, o gatekeeper deve ter conexão direta com ambas as redes do chamador e chamado.

- Enable=1
  Default: 0 Habilitar ou não a função proxy. Você terá que habilitar o modo Gatekeeper routed primeiro (veja a 4.2.2 (seção anterior)). Você não precisa necessariamente especificar o roteamento do H.245. Ele será automaticamente usado caso seja necessário.

- InternalNetwork=10.0.1.0/24
  Default: N/A Definir as redes atrás do proxy. Múltiplas redes internas são permitidas. O proxy roteia os canais somente se a comunicação entre endpoints for entre uma rede interna e uma externa. Se você não especificar esta, todas as chamadas serão roteadas via proxy.

  Format:

  InternalNetwork=network address/netmask[,network address/netmask,...]

  A máscara pode ser expressa em notação decimal com ponto ou notação CIDR (tamanho do prefixo), como mostrado no exemplo.

  Exemplo:

  InternalNetwork=10.0.0.0/255.0.0.0,192.168.0.0/24

- T120PortRange=40000-40999
  Default: 0 (variável) Especificar a faixa de portas TCP para os canais de dados T.120. Observe que esta faixa pode limitar o número de chamadas concorrentes.

- RTPPortRange=50000-59999
  Default: 10000-59999 Especificar a faixa de portas UDP para os canais RTP/RTCP. Observe que esta faixa pode limitar o número de chamadas concorrentes.

- ProxyForNAT=1
  Default: 1 Se verdadeiro, o gatekeeper irá realizar o proxy das chamadas nas quais os endpoints participantes estejam atrás de um NAT box. Isto garante que o fluxo RTP/RTCP possa penetrar na NAT box sem modifica-la. Entretanto, o endpoint atrás do NAT box pode usar a mesma porta para enviar e receber fluxo RTP/RTCP. Se você não tiver um endpoint que satisfaça esta pré-condição, é melhor desabilitar esta função e deixar o NAT box redirecionar o fluxo RTP/RTCP para você.

- ProxyForSameNAT=0
  Default: 0 Fazer o proxy ou não de chamadas entre endpoints que estejam no mesmo lado do NAT box. Em geral, você não precisa habilitar esta funcionalidade, pois normalmente os endpoints do mesmo NAT box podem ser comunicar mutuamente.

4.2.4 Seção [GkStatus::Auth]

Define um conjunto de regras que descrevem quem poderá conectar na porta de status.

- rule=allow
  Default: forbid Possíveis valores são

  - forbid - desabilita quaisquer conexões.
  - allow - permite quaisquer conexões
  - explicit - lê um pârametro ip=value onde ip é o endereço do cliente de gerencia/monitoramento, value é 1,0 ou allow,forbid ou yes,no. Se ip não estiver listado no parâmetro default será usado.

- regex - o IP do cliente deve casar com o padrão ditado pela expressão regular.

  Exemplo:

  Para permitir clientes de gerencia da faixa 195.71.129.0/24 e 195.71.131.0/24:

  ```
  regex=^195\.71\.(129|131)\.[0-9]+$
  ```

- password - o usuário deve colocar um username e password apropriado para logar. O formato de username/password é o mesmo da seção 4.2.16 ([Password]).

Além disso, estas regras podem ser combinadas por "|"ou "&". Por exemplo,

- rule=explicit | regex
  O endereço IP do cliente deve casar explicitamente explicit ou com regra regex.
- rule=regex & password
  O endereço IP do cliente deve casar com a regra regex, e o usuário tem que logar provendo o username e password.

- default=allow
  Default: forbid Somente usando quando rule=explicit.
- Shutdown=forbid
  Default: allow Permitir ou não o shutdown do gatekeeper via porta de status.

4.2.5 Seção [RasSrv::GWPrefixes]

Esta seção lista quais números E.164 são roteados para um gateway específico.

Formato:

```
gw-alias=prefix[,prefix,...]
```

Repare que você tem que especificar o alias (apelido) do gateway. Se um gateway tiver se registrado com aquele alias, todos os números começando com os prefixos expressos são redirecionados para este gateway.

Exemplo:

```
test-gw=02,03
```

4.2.6 Seção [RasSrv::RewriteE164]

Esta seção define as regras para a reescrita dos dialedDigits (número E.164).

Formato:

```
[!]original-prefix=target-prefix[,target-prefix,...]
```

Se o número começar com original-prefix, ele é reescrito para target-prefix. Multiplos destinos são possíveis. Se o flag '!' preceder o original-prefix, o sentido é invertido.

Exemplo:

```
08=18888
```

Se você discar 08345718, isso será reescrito para 18888345718.

Opcional:

- Fastmatch=08
  Default: N/A Somente reescreve o dialDigits que comece com o prefixo especificado.

4.2.7 Seção [RasSrv::PermanentEndpoints]

Nesta seção você pode configurar endpoints que não tenham suporte a RAS ou não querem ter seus registros expirados. Estes registros estarão sempre armazenados na tabela de registro do gatekeeper. Entretanto, você ainda pode desregistrar este endpoint via porta de status.

Formato:

```
IP[:port]=alias[,alias,...;prefix,prefix,...]
```

Exemplo:

Para gateway,

```
10.0.1.5=Citron;009,008
```

Para terminal,

```
10.0.1.10:1720=700
```

4.2.8 Seção [RasSrv::Neighbors]

Se o destino de um ARQ for desconhecido, o gatekeeper envia LRQs para seus vizinhos para perguntar se eles tem o destino procurado como endpoint. Um vizinho é selecionado se o seu prefixo casar com o destino sendo procurado ou se tiver o prefixo "*". Normalmente somente um prefixo é suportado.

Usualmente, o gatekeeper apenas responde os LRQs enviados por vizinhos que estejam na sua lista de vizinhos definida nesta seção. Se você especificar um prefixo vazio, nenhum LRQ será enviado para aquele vizinho, mas o gatekeeper aceitará LRQs deste.

O campo password é usado para autenticar os LRQs daquele vizinho. Veja seção 4.2.15 ([Gatekeeper::Auth]) para detalhes.

Formato:

```
GKID=ip[:port;prefix;password;dynamic]
```

Exemplo:

```
GK1=192.168.0.5;*
GK2=10.0.1.1:1719;035;gk2
GK3=gk.citron.com.tw;;gk3;1
```

4.2.9 Seção [RasSrv::LRQFeatures]

Define algumas características de LRQ e LCF.

- NeighborTimeout=1
  Default: 2 Valores para expirar a espera de respostas dos vizinhos. Se nenhum resposta de todos os vizinhos for recebida até o timeout, o gatekeeper irá responder ARJ para o endpoint que está esperando ARQ.

- ForwardHopCount=2
  Default: N/A Se o gatekeeper recebe um LRQ cujo destino seja desconhecido, ele pode redirecionar esta mensagem para os seus vizinhos. Quando o gatekeeper recebe o LRQ e decide que aquela mensagem pode ser redirecionada para outro gatekeeper, em primeiro lugar ele decrementa o valor do campo hopCount do LRQ. De modo que se hopCount chegar a 0, o gatekeeper não irá redirecionar a mensagem. Esta opção define o número de gatekeepers pelos quais um LRQ pode propagar. Observar que isso somente afeta o emissor do LRQ, não o re-encaminhador.

- AlwaysForwardLRQ=1
  Default: 0 Força o gatekeeper a direcionar LRQ mesmo que não haja hopCount no LRQ. Cuidado com os loops de LRQ, evite, você deve usar esta opção com muito cuidado.

- AcceptForwardedLRQ=1
  Default: 1 Aceita ou não um pacote LRQ que foi redirecionado dos vizinhos.

- IncludeDestinationInfoInLCF=0
  Default: 1 O gatekeeper responde com LCFs contendo os campos destinationInfo e destinationType, os aliases e o tipo de terminais (PC, GW) do endpoint destino. O gatekeeper vizinho pode então armazenar esta informação para suprimir LRQ mais tarde. Entretanto, alguns fabricantes de gatekeeper não usam corretamente esta característica, o que pode resultar em problemas de interoperabilidade. Somente desligue esta opção se você encontrar problemas na comunicação com um gatekeeper de outros fabricante.

- CiscoGKCompatible=1
  Default: 0 Inclui um parâmetro NonStandardParameter nos LRQs para ficar compatível com os gatekeepers Cisco.

4.2.10 Seção [RasSrv::RRQFeatures]

- AcceptEndpointIdentifier=1
  Default: 1 Aceitar ou não o endpointIdentifier especificado em um RRQ completo.

- AcceptGatewayPrefixes=1
  Default: 1 Um gateway pode registrar seus prefixos com o gatekeeper usando o campo supportedPrefixes no campo terminalType do RRQ. Esta opção define se será aceito ou não o prefixo especificado de um gateway.

- OverwriteEPOnSameAddress=1
  Default: 0 Em algumas redes um endereço IP de endpoint pode mudar inesperadamente. Isto pode acontecer quando um endpoint estiver usando uma conexão PPP (ex. modem ou ADSL). Esta opção define como manipular uma requisição de registro (RRQ) de um endereço IP que nã corresponde ao que estava armazenado. A ação default é rejeitar a requisição. Com esta opção habilitada a requisição conflitante irá causar um unregister request (URQ) para ser enviado para o endereço IP existente e a entrada será removida permitindo ao endpoint se registrar com um novo endereço

4.2.11 Seção [RasSrv::ARQFeatures]

- ArjReasonRouteCallToSCN=0
  Default: 1 Se verdadeiro, o gatekeeper irá rejeitar ligações de um gateway para ele mesmo pela reason routeCallToSCN.

- ArjReasonRouteCallToGatekeeper=1
  Default: 1 Se verdadeiro, o gatekeeper rejeita um ARQ respondido sem um registro pre-existente CallRec na CallTable pela reason routeCallToGatekeeper no modo Gk-routed. O endpoint deverá liberar a chamada imediatamente e re-enviar o Setup da chamada para o gatekepeer.

- CallUnregisteredEndpoints=0
  Default: 1 Com esta opção configurada, o gatekeeper irá aceitar um ARQ de um endpoint registrado com destCallSignalAddress, não importando se o endereço pertença a um endpoint registrado ou não. Isto significa que você pode explicitar especificamente o IP do endpoint (registrado ou não) que você deseja chamar.

- RemoveTrailingChar=#
  Default: N/A Especificar o caracter de trailer que será removida da destinationInfo. Por exemplo, se seu endpoint incorretamente contiver o caracter de terminação como '#' no destinationInfo, você poderá remove-lo através desta opção.

4.2.12 Seção [CallTable]

- GenerateNBCDR=0
  Default: 1 Gerar CDRs (Call Detail Record) das chamadas que vierem das zonas vizinhas. O IP e o endpoint ID da parte chamadora é exibido vazia. Isto é normalmente usado para fins de debug.

- GenerateUCCDR=0
  Default: 0 Gerar CDRs para chamadas tidas como não-conectadas. Isto normalmente é usado para fins de pesquisa. Observe que uma chamada é considerada não-conectada somente se o gatekeeper estiver roteando a sinalização e a mensagem Connect do Q.931 não foi recebida pelo gatekeeper. No modo de sinalização direta, uma chamada é sempre como conectada.

- DefaultCallDurationLimit=3600
  Default: 0 Tempo máximo em segundos para a duração das ligações. Configure como 0 para desabilitar este recurso.

  Limite máximo padrão de duração de chamadas (segundos). Configure como 0 para desabilitar este recurso e não limitar o tempo de duração das chamdas.

4.2.13 Seção [Endpoint]

O gatekeeper pode trabalhar como um endpoint podendo se registrar com outro gatekeeper. Com este recurso, você pode criar facilmente hierarquias de gatekeepers. A seção define as características de endpoint de um gatekeeper.

- Gatekeeper=10.0.1.1
  Default: no Define um gatekeeper pai para o endpoint (gatekeeper) de modo que ele se registre com este. E não tente se registrar consigo mesmo, ou o programa ficará confuso. Para desabilitar este recurso, configure o campo como no.

- Type=Gateway
  Default: Gateway Define o tipo de terminal para este endpoint. Os valores válidos são Gateway ou Terminal.

- H323ID=CitronProxy
  Default: <Name> Especifica os aliases (apelidos) H.323 deste endpoint. Múltiplos aliases podem ser separados por vírgulas.

- E164=18888600000,18888700000
  Default: N/A Especifica os aliase E.164 (dialedDigits) deste endpoint. Múltiplos aliases podem ser separados por vírgulas.

- Password=123456
  Default: N/A Especifica um senha a ser enviada para o gatekeeper pai. Todas as requisições RAS irão conter a senha no campo cryptoTokens (MD5 & HMAC-SHA1-96) e no campo tokens (CAT). Para enviar requisições RAS sem o campos cryptoTokens e tolkens, configure configure a senha como vazio.

  Além disso, a senha também é usada nos LRQs a serem enviados entre gatekeepers vizinhos.

- Prefix=188886,188887
  Default: N/A Registra os prefixos especificados com o gatekeeper pai. Somente tem efeito quando o Type especificado for Gateway.

- TimeToLive=900
  Default: N/A Sugere um valor em segundos para o tempo-de-vida do registro. Note que o verdadeiro valor deste temporizador do tempo-de-vida é atribuido pelo gatekeeper pai quando responder com RCF uma requisição RRQ.

- RRQRetryInterval=10
  Default: 10 Define um interface de retransmissão em segundos para os RRQs caso nenhuma resposta seja recebida pelo gatekeeper pai.

- ARQTimeout=2
  Default: 2 Definir o valor do temporizador em segundo para os ARQs.

- UnregisterOnReload=1
  Default: 0 Definir se o gatekeeper filho deverão desregistrar ou re-registrar com seu pai toda vez que receberem o comando Reload.

- NATRetryInterval=60
  Default: 60 Intervalo de retransmissão em segundos para o socket NAT. Deixe em branco, caso você não entenda este parâmetro.

- NATKeepaliveInterval=86400
  Default: 86400 Intervalo de keepalive em segundos do socket NAT. Deixe em branco, caso você não entenda este parâmetro.

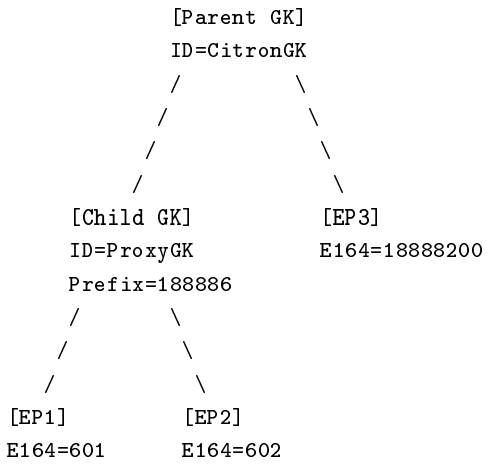
4.2.14 Seção [Endpoint::RewriteE164]

Uma vez que você tiver especificado prefixos para o endpoint gatekeeper, o gatekeeper pai irá redirecionar chamadas cujo dialedDigits comecem com aqueles prefixos. O gatekeeper filho pode reescrever o destino de acordo a regras especificadas nesta seção. Ao contrário, quando um endpoint interno chamar um endpoint registrado no gatekeeper pai, a fonte será reescrita reversamente.

Formato:

```
external prefix=internal prefix
```

Por exemplo, se você tiver a seguinte configuração,



Com esta regra:

```
188886=6
```

Quando o EP1 chamar o EP3 através de 18888200, a parte do número chamadora CallingPartyNumber do Setup Q.931 será reescrito para 18888601. Assim sendo, EP3 poderá alcançar os EP1 e EP2 chamando 18888601 e 18888602, respectivamente. Portanto, um endpoint registro no GK filho com prefixo '6' irá parece como se fosse um endpoint com prefixo '188886', para os endpoints registrados no gatekeeper pai.

Esta seção não está relacionada com a seção 4.2.6 (RasSrv::RewriteE164), embora esta última tenha seu efeito aplicado em primeiro lugar.

4.2.15 Seção [Gatekeeper::Auth]

Esta seção define os mecanismos de autenticação com o gatekeeper.

Sintaxe:

```
authrule=actions

<authrule> := SimplePasswordAuth | AliasAuth | PrefixAuth | RadAuth | RadAliasAuth | ...
<actions>  := <control>[;<ras>|<q931>,<ras>|<q931>,...]
<control>  := optional | required | sufficient
<ras>      := GRQ | RRQ | URQ | ARQ | BRQ | DRQ | LRQ | IRQ
<q931>     := Setup
```

A aplicação de um regra resulta em um dos três possíveis resultados: ok, fail, next.

- ok - A requisição é autenticada pelo módulo.
- fail - A autenticação falha e poderá ser rejeitada.
- next - A regrga não pode determinar o tratamento para a requisição.

Existem também três maneiras de controlar uma regra:

- optional - Se esta regra não puder tratar a requisição, esta deve passar para a próxima regra.
- required - As requisições devem ser autenticadas por este módulos, ou então serão rejeitadas. A requisição autenticada então poderia passar para a próxima regra.
- sufficient - Se a for autenticada, ela será aceita, ou poderia ser rejeitada. Assim, esta regra determina o destino da requisição. Nenhuma regra deve ser colocada após um regra sufficient, pois ela não terá nenhum efeito.

Módulos suportados atualmente:

- SimplePasswordAuth/MySQLPasswordAuth/LDAPPasswordAuth/ExternalPasswordAuth Estes módulos checam os campos tokens ou cryptoTokens da mensagem RAS. Os tokens deveriam conter pelo menos o generalID e a senha. No que diz respeito aos cryptoTokens, tanto tokens cryptoEPPwdHash criptografados por MD5 simples quanto os tokens nestedcryptoToken criptogradados pelo HMAC-SHA1-96 (libssl deve estar instalada!) são suportados atualmente. Para tokens tokens criptografados por CAT (Cisco Accesss Token) e em texto puro usuário/senha são puportados atualmente. O ID e senha são lidos da seção 4.2.16 ([Password]), ou de um banco de dados MySQL, ou LDAP ou um programa externo, logo eles devem ser configurados com as seções SimplePasswordAuth, MySQLPasswordAuth, LDAPPasswordAuth e ExternalPasswordAuth, respectivamente. O suporte a outros banco de dados de backend são facilmente adicionados.
- NeighborPasswordAuth Este módulo é usado para autenticar os LRQs vindos dos vizinhos definidos na seção 4.2.8 ([RasSrv::Neighbors]).
- AliasAuth/MySQLAliasAuth/LDAPAliasAuth Este módulo pode ser usado somente para autenticar RegistrationRequest (RRQ). O IP de um endpoint com certo alias poderia casar com um padrão especificado. Para o AliasAuth o padrão é definido na seção 4.2.19 ([RasSrv::RRQAuth]). Para o MySQLAliasAuth, o padrão é obtido do banco de dados MySQL, definido na seção 4.2.20 ([MySQLAliasAuth]). Para o LDAPAliasAuth o alias (default: atributo mail) e o IP (default: atributo voIPIpAddress) devem ser encontrados em uma entrada LDAP.
- PrefixAuth Originalmente conhecido como GkAuthorize. O IP ou aliases de uma requisição para certo prefixo devem casar com um determinado padrão. Veja a seção 4.2.21 ([PrefixAuth]) para maiores detalhes. Atualmente o módulo pode autorizar somente AdmissionRequest (ARQ) e LocationRequest (LRQ).
- RadAuth Provê autenticação baseada em username/password do esquema de segurança H.235. Autentica RRQs e/ou ARQs através de servidores RADIUS remotos. Ele passa aos servidores RADIUS os usernames e passwords extraidos dos tokens CAT (Cisco Access Tokens) transportados dentro de pacotes RRQ e ARQ. Caso seus endpoints não suportem CATs CATs ou você não precisa de um esquema de autenticação baseado em usernames/password associados individualmente - este módulo não funcionará com você (mas você pode checar o módulo RadAliasAuth). Veja a seção 4.2.22 ([RadAuth]) para maiores detalhes.
- RadAliasAuth Provê autenticação baseado nos aliases dos endpoints e/ou endereços IP de sinalização com servidores RADIUS remotos. Ele não precisa de qualquer tokens H.235 dentro das mensagem RAS, desta forma ele pode ser usado por uma variedade maior de sistema comparado ao RadAuth. As mensagens RRQ, ARQ e Q.931 Setup podem ser autenticadads usando este módulo. Veja a seção 4.2.23 ([RadAliasAuth]) para maiores detalhes.

Você pode configurar uma regra para checar somente alguma mensagem RAS em particular. O exemplo seguinte configura o SimplePasswordAuth como uma regra opcional para checar RRQ e ARQ. Se um RRQ não tiver checado (ele não contiver os campos tokens ou cryptoTokens), ele será checado pelo AliasAuth. O default é aceitar todas as requisições.

Example 1:

```
SimplePasswordAuth=optional;RRQ,ARQ
AliasAuth=sufficient;RRQ
default=allow
```

Example 2:

```
RadAliasAuth=required;Setup
default=allow
```

4.2.16 Seção [Password]

Esta seção define os pares userid e password usados pelo módulo SimplePasswordAuth. Use o comando Use 'make addpasswd' para gerar o utilitário de geração de senhas addpasswd.

Como usar o addpasswd:

```
addpasswd config userid password
```

Opções:

- KeyFilled=123
  Default: 0 Valor padrão de inicialização da chave criptográfica.
- CheckID=1
  Default: 0 Checa se os aliases casam com os ID no tokens.
- PasswordTimeout=120
  Default: -1 O módulo SimplePasswordAuth e todos os seus descendentes podem fazer um cache das senhas autenticadas. Este parâmetro define o valor do tempo de vida do cache em segundos. 0 significa que o cache não é usado para armazenar senhas, enquanto que um valor negativo significa que o cache nunca expira.

4.2.17 Seção [MySQLAuth]

Define o banco de dados MySQL, tabelas e campos a serem usados para extrair o userid e password.

- Host=localhost
  Default: localhost Nome do host ou IP do servidor MySQL.
- Database=billing
  Default: billing O banco de dados a ser conectado.
- User=cwhuang
- Password=123456
  O nome do usuário e senha para conectar ao banco de dados.

- Table=customer
  A tabela no banco de dados a ser usada para pesquisas/atualizações.
- IDField=IPN
  O nome do campo do identificador do usuário (user id).
- PasswordField=Password
  O nome do campo de senha.
- ExtraCriterion=Kind > 0
  Default: N/A Especificar critérios extra.

O comando SQL pode ser escrito da seguinte forma:

```
SELECT $PasswordField FROM $Table WHERE $IDField = %id [AND $ExtraCriterion]
```

4.2.18 Seção [ExternalPasswordAuth]

Especificar um programa externo que recupere a senha. O programa pode aceitar o ID da entrada padrão e imprimir a senha na saída padrão.

- PasswordProgram=/usr/local/bin/getpasswd
  Default: N/A Para executar programas externos.

4.2.19 Seção [RasSrv::RRQAuth]

Especifica a ação a ser tomada na recepção do RRQ (confirmar ou negar) pelo módulo AliasAuth. O primeiro alias (normalmente o H323ID) do endpoint a ser registrado será verificado nesta seção. Se um parâmetro for encontrado o valor será aplicado como uma regra. Uma regra consite de condições separadas por "&". Um registro será aceito quando todas as condições se aplicarem.

Sintaxe:

```
<authrules> :=  empty  |  <authrule> "&" <authrules>

  <authrule>  := <authtype> ":" <authparams>
  <authtype>  := "sigaddr" | "sigip"
  <autparams> := [!&]*
```

A notação e significado de <authparams> depende do <authtype>:

- sigaddr - expressão regular estendida que deve casar com a representação "PrintOn(ostream)" do endereço de sinalização da requisição. Exemplo:

  ```
  sigaddr:.*ipAddress .* ip = .* c0 a8 e2 a5 .*port = 1720.*
  ```

- sigip - forma especializada do 'sigaddr'. Escreva o endereço IP de sinalização usando (o normalmente usado) formato de notação decimail: "byteA.byteB.byteC.byteD:port". Exemplo:

  ```
  sigip:192.168.242.165:1720
  ```

- allow - sempre aceita o alias.
- deny - sempre rejeita o alias.

4.2.20 Seção [MySQLAliasAuth]

Define o banco de dados MySQL, tabelas e campos a serem usados para extrair um padrão para um alias.

- Host=localhost
  Default: localhost Nome do host ou IP do servidor MySQL.
- Database=billing
  Default: billing O banco de dados a ser conectado.
- User=cwhuang
- Password=123456
  O nome do usuário e senha para conectar ao banco de dados.
- Table=customer
  A tabela no banco de dados a ser usada para pesquisas/atualizações.
- IDField=IPN
  O nome do campo do identificador do usuário (user id).
- IPField=Address
  O nome do campo com a especificação do endereço IP.
- ExtraCriterion=Kind > 0
  Default: N/A Especificar critérios extra.

O comando SQL pode ser escrito da seguinte forma:

```
SELECT $IPField FROM $Table WHERE $IDField = %alias [AND $ExtraCriterion]
```

4.2.21 Seção [PrefixAuth]

Esta seção define as regras de autenticação do módulo PrefixAuth. Atualmente, somente os ARQs e LRQs podem ser autorizados por este módulo.

Primeiramente, o prefixo mais especifico é selecionado de acordo com o campo destinationInfo da requisição recebida. Desta forma a requisição é aceita ou rejeitada de acordo com regras casadas com mascaras de sub-rede específicas. Se nenhum prefixo casar no padrão, e a opção default estiver especificada, logo a requisição será aceita ou rejeita de acordo com esta última. Entretanto, ela pode ser rejeitada ou ser passada para o próximo módulo de autenticação de acordo com o requisito do módulo.

Formato:

```
prefix=authrule[|authrule|...]
```

Sintaxe:

```
<authrule> :=  <result> <authrule>

  <result>    := deny | allow
  <authrule>  := [!]ipv4:<iprule> | [!]alias:<aliasrule>
```

Onde o <iprule> pode ser especificado em notação decimal ou notação CIDR, <aliasrule> é expresso com expressões regulares. Se a flag '!' preceder a regra, o sentido será invertido.

Exemplo:

```
555=deny ipv4:10.0.0.0/27|allow ipv4:0/0
5555=allow ipv4:192.168.1.1|deny ipv4:192.168.1.0/255.255.255.0
86=deny !ipv4:172.16.0.0/24
09=deny alias:^188884.*
ALL=allow ipv4:ALL
```

Nesta configuração, todos os endpoints excepto os da rede 10.0.0.0/27 tem permissão para fazer chamadas para o prefixo 555 (menos o 5555). Os endpoints da rede 192.168.1.0/24 não tem permissão de chamar os prefixos 5555, exceto o endereço IP 192.168.1.1. Os endpoints que não não sejam da rede 172.16.0.0/24 são proibidos de realizar chamadas para o prefixo 86. Os endpoints que tiverem seus alias começando com 188884 não tem permissão de chamar o prefixo 09. Todas as outras situações são permitidas.

4.2.22 Section [RadAuth]

Esta seção define as configurações que habilitam a autenticação RADIUS baseado nos CATs (Cisco Access Tokens) H.235 presentes em requisições RAS RRQ e ARQ. Este esquema de autenticação é útil somente para terminais registrados no gatekeeper.

- Servers=SERVER1[:AUTH_PORT];SERVER2[:AUTH_PORT];...
  Default: N/A A lista de servidores RADIUS a serem usados para a autenticação das requisições RAS. Esta lista pode conter um número arbitrário de servidores. A ordem dos servidores é importante, pois eles serão pesquisados pelo módulo RADIUS na ordem especificada. Se nenhuma informação sobre porta for colocada, o valor DefaultAuthPort será usado. Se estes atributos não forem configurados, o arquivo services será checado para verificar a porta do serviço 'radius'. Os servidores podem estar no formato de endereços IP no nomes DNS.

  Exemplo:

  ```
  Servers=192.168.3.1:1645;192.168.3.2:1812;radius.mycompany.com
  ```

- LocalInterface=IP_OR_FQDN
  Default: N/A Interface de rede local que o cliente RADIUS estará usando para se comunicar com os servidores RADIUS. Este parâmetro pode ser útil em máquinas NAT para restringir o número de interfaces de rede usadas para comunicação com RADIUS. Por default este valor é vazio e permite que as requisições RADIUS sejam enviadas por qualquer (melhor possível) interface de rede. Se você não estiver bem certo do que está fazendo, é melhor deixar esta opção sem configuração.

- RadiusPortRange=10000-11000
  Default: N/A Por default (se esta opção não estiver configurada) o cliente RADIUS alocará portas dinamicamente conforme especificado pelo sistema operacional. Se você quiser restringir o cliente RADIUS para usar uma faixa particular de portas - configure este parâmetro.

- DefaultAuthPort=PORT_NO
  Default: Uma entrada de serviço 'radius' no arquivo services ou o valor 1812 A porta padrão a ser usada pelas requisições de autenticação RADIUS (Access-Request packets), se não forem sobrepostas pelo atributo Servers.

- SharedSecret=SECRET
  Default: N/A (empty string) Segredo usado para autenticar este GNU GK (cliente NAS) com o servidor RADIUS. Ela deve ser uma senha criptograficamente forte.

- RequestTimeout=TIMEOUT_MS
  Default: 2000 (miliseconds) Timeout (em milisegundos) para o servidor RADIUS responder uma requisição enviada pelo GNU GK. Se nenhuma resposta for recebida dentro deste período de tempo, o próximo servidor RADIUS será utilizado.

- IdCacheTimeout=TIMEOUT_MS
  Default: 9000 (miliseconds) Timeout (em milisegundos) do identificador de 8-bits da requisição RADIUS ser único. Se toda a faixa de identificadores de 8-bit acabar dentro deste período, um novo socket cliente (socket UDP) será alocado pelo módulo RADIUS. Vamos tomar como exemplo: temos aproximadamente 60 RRQs/sec - após ca. 4 segundos os identificadores de 8-bit acabam - um próximo socket é alocado - após os próximos 4 segundos a segunda faixa de identificadores de 8-bit também acaba - um terceiro socket é alocado - após o 9º grupo de identificadores tiver sido alocado, o pool 1 estarão disponíveis novamente - ... . Em geral, as mensagens too long timeout - too much resources consumed, too short timeout - são os servidores RADIUS podem ter pacotes como duplicados e desta forma descartaram.

- SocketDeleteTimeout=TIMEOUT_MS
  Default: 60000 (miliseconds) - 60 s Timeout para os sockets RADIUS serem fechados. Ele é usado em conjunto com IdCacheTimeout - sockets adicionais criados devido a períodos de alta carga para servir requisições são fechados após períodos sem utilização.

- RequestRetransmissions=NUMBER
  Default: 2 Quantas vezes uma simples requisição será transmitido para cada servidor RADIUS configurado (se nenhuma resposta for recebida). 1 significa nenhuma retransmissão, 2 - retransmissão simple, ... O método de retransmissão exato é definido pelo atributo RoundRobinServers.

- RoundRobinServers=BOOLEAN
  Default: 1 O método de retransmissão das requisições RADIUS.

  Se tiver configurado como 1, as requisições RADIUS serão retransmitidas do seguinte forma (até que a resposta seja recebida):

  ```
  Servidor #1 Tentativa #1, Servidor #2 Tentativa #1, ..., Servidor #N Tentativa #1
  ...
  Servidor #1 Tentativa #MaxRetrans, ..., Servidor #1 Tentativa #MaxRetrans
  ```

  Se for configurado com 0, a seguinte sequencia é preservada:

  ```
  Servidor #1 Tentativa #1, ..., Servidor #1 Tentativa #MaxRetrans
  ...
  Servidor #N Tentativa #1, ..., Servidor #N Tentativa #MaxRetrans
  ```

- AppendCiscoAttributes=BOOLEAN
  Default: 0 Se configurado, os atributos RADIUS Especificos da Cisco serão incluidos nas requisições RADIUS (h323-conf-id,h323-call-origin,h323-call-type).

- IncludeTerminalAliases=BOOLEAN
  Default: 1 Se configurado, o atributo Cisco VSA 'h323-ivr-out' é enviado com a lista de aliases registrados pelo endpoint (RRQ.m_terminalAlias). Este atributo é fornecido para prover um controle refinado sobre a lista de aliases dos quais permiti-se que o endpoint se registre. O formato deste atributo é:

```
        Cisco-AV-Pair = "h323-ivr-out=terminal-alias:" alias [,alias] [;]
Example:
        Cisco-AV-Pair = "h323-ivr-out=terminal-alias:helpdesk,support,77771;"
```

  IncludeEndpointIP=BOOLEAN
  Default: 1

  Se configurado, o endereço IP será passado como atributo RADIUS Framed-IP-Address dentro dos Access-Requests. Este pode ser usado para autenticação adicional (do username/password). Para o RRQs, este atributo é igual aos campos RRQ.m_radAddress[0] ou RRQ.m_callSignalAddress[0]. Para os ARQs, este atributo é igual ao campo ARQ.m_srcCallSignalAddress ou ao endereço de sinalização de chamada armazenado pela tabela de registro do gatekeeper.

4.2.23 Seção [RadAliasAuth]

Esta seção define as configurações que habilitam a autenticação RADIUS baseado nos alias ou endereços IP presentes em requisições RAS RRQ , RAS ARQ e Q.931 Setup. Este esquema de autenticação é util tanto para terminais registrados no gatekeeper (ARQ,RRQ) quanto para chamadas originadas de terminais não registrados (Setup).

- Servers=SERVER1[:AUTH_PORT];SERVER2[:AUTH_PORT];...
  Default: N/A A lista de servidores RADIUS a serem usados para a autenticação das requisições RAS. Esta lista pode conter um número arbitrário de servidores. A ordem dos servidores é importante, pois eles serão pesquisados pelo módulo RADIUS na ordem especificada. Se nenhuma informação sobre porta for colocada, o valor DefaultAuthPort será usado. Se estes atributos não forem configurados, o arquivo services será checado para verificar a porta do serviço 'radius'. Os servidores podem estar no formato de endereços IP no nomes DNS.

  Example:

```
Servers=192.168.3.1:1645;192.168.3.2:1812;radius.mycompany.com
```

- LocalInterface=IP_OR_FQDN
  Default: N/A Interface de rede local que o cliente RADIUS estará usando para se comunicar com os servidores RADIUS. Este parâmetro pode ser útil em máquinas NAT para restringir o número de interfaces de rede usadas para comunicação com RADIUS. Por default este valor é vazio e permite que as requisições RADIUS sejam enviadas por qualquer (melhor possível) interface de rede. Se você não estiver bem certo do que está fazendo, é melhor deixar esta opção sem configuração.

- RadiusPortRange=10000-11000
  Default: N/A Por default (se esta opção não estiver configurada) o cliente RADIUS alocará portas dinamicamente conforme especificado pelo sistema operacional. Se você quiser restringir o cliente RADIUS para usar uma faixa particular de portas - configure este parâmetro.

- DefaultAuthPort=PORT_NO
  Default: Uma entrada de serviço 'radius' no arquivo services ou o valor 1812 A porta padrão a ser usada pelas requisições de autenticação RADIUS (Access-Request packets), se não forem sobrepostas pelo atributo Servers.

- SharedSecret=SECRET
  Default: N/A (empty string) Segredo usado para autenticar este GNU GK (cliente NAS) com o servidor RADIUS. Deve ser uma senha criptograficamente forte.

- RequestTimeout=TIMEOUT_MS
  Default: 2000 (miliseconds) Timeout (em milisegundos) para o servidor RADIUS responder uma requisição enviada pelo GNU GK. Se nenhuma resposta for recebida dentro deste período de tempo, o próximo servidor RADIUS será utilizado.

- IdCacheTimeout=TIMEOUT_MS
  Default: 9000 (miliseconds) Timeout (em milisegundos) para o identificador de 8-bits da requisição RADIUS ser único. Se toda a faixa de identificadores de 8-bit acabar dentro deste período, um novo socket cliente (socket UDP) será alocado pelo módulo RADIUS. Vamos tomar como exemplo: temos aproximadamente 60 RRQs/sec - após ca. 4 segundos os identificadores de 8-bit acabam - um próximo socket é alocado - após os próximos 4 segundos a segunda faixa de identificadores de 8-bit também acaba - um terceiro socket é alocado - após o 9º grupo de identificadores tiver sido alocado, o pool 1 estarão disponíveis novamente - ... . Em geral, as mensagens too long timeout - too much resources consumed, too short timeout - são os servidores RADIUS podem ter pacotes como duplicados e desta forma descartaram.

- SocketDeleteTimeout=TIMEOUT_MS
  Default: 60000 (miliseconds) - 60 s Timeout para os sockets RADIUS serem fechados. Ele é usado em conjunto com IdCacheTimeout - sockets adicionais criados devido a períodos de alta carga para servir requisições são fechados após períodos sem utilização.

- RequestRetransmissions=NUMBER
  Default: 2 Quantas vezes uma simples requisição será transmitido para cada servidor RADIUS configurado (se nenhuma resposta for recebida). 1 significa nenhuma retransmissão, 2 - retransmissão simple, ... O método de retransmissão exato é definido pelo atributo RoundRobinServers.

- RoundRobinServers=BOOLEAN
  Default: 1 O método de retransmissão das requisições RADIUS.

  Se tiver configurado como 1, as requisições RADIUS serão retransmitidas do seguinte forma (até que a resposta seja recebida):

  ```
  Servidor #1 Tentativa #1, Servidor #2 Tentativa #1, ..., Servidor #N Tentativa #1
  ...
  Servidor #1 Tentativa #MaxRetrans, ..., Servidor #1 Tentativa #MaxRetrans
  ```

  Se for configurado com 0, a seguinte sequencia é preservada:

  ```
  Servidor #1 Tentativa #1, ..., Servidor #1 Tentativa #MaxRetrans
  ...
  Servidor #N Tentativa #1, ..., Servidor #N Tentativa #MaxRetrans
  ```

- AppendCiscoAttributes=BOOLEAN
  Default: 1 Se configurado, os atributos RADIUS Especificos da Cisco serão incluidos nas requisições RADIUS (h323-conf-id,h323-call-origin,h323-call-type).

- IncludeTerminalAliases=BOOLEAN
  Default: 1 Se configurado, o atributo Cisco VSA 'h323-ivr-out'

  é enviado com a lista de aliases registrados pelo endpoint (RRQ.m_terminalAlias). Este atributo é fornecido para prover um controle refinado sobre a lista de aliases dos quais permiti-se que o endpoint se registre. O formato deste atributo é:

```
        Cisco-AV-Pair = "h323-ivr-out=terminal-alias:" alias [,alias] [;]
Example:
        Cisco-AV-Pair = "h323-ivr-out=terminal-alias:helpdesk,support,77771;"
```

- IncludeEndpointIP=BOOLEAN
  Default: 1 Se configurado, o endereço IP será passado como atributo RADIUS Framed-IP-Address dentro dos Access-Requests. Este pode ser usado para autenticação adicional (do username/password). Para o RRQs, este atributo é igual aos campos RRQ.m_radAddress[0] ou RRQ.m_callSignalAddress[0]. Para os ARQs, este atributo é igual ao campo ARQ.m_srcCallSignalAddress ou ao endereço de sinalização de chamada armazenado pela tabela de registro do gatekeeper.

- FixedUsername
  Default: N/A Se este parâmetro estiver configurado, ele sobrepõe um valor do atributo User-Name do RADIUS para a requisição. Isto significa que cada Access-Request será autenticado pelo usuário especificado em FixedUsername.

- FixedPassword
  Default: N/A Se não estiver configurado, o User-Password é copiado como User-Name. Por exemplo, se o User-Name é 'john' então o User-Password também será configurado como 'john'. Configurando este parametro sobrepõe este comportamento e o atributo User-Password será sempre configurado com o valor de FixedPassword.

  Exemplo 1:

```
(Nem FixedUsername muito menos FixedPassword configurados)
```

  Todos os endpoints serão autenticados usando seus alias como o username e a senha. Isto significa que, por exemplo o endpoint 'EP1' será autenticado pelo username 'EP1 e a senha 'EP1'.

  Exemplo 2:

```
(O FixedUsername não configurado)
FixedPassword=ppp
```

  Todos os endpoints serão autenticados usando seus alias e a senha 'ppp'.

  Exemplo 3:

```
FixedUsername=ppp
FixedPassword=ppp
```

  Todos os endpoints serão autenticados usando o username 'ppp' e a senha 'ppp'.

4.2.24 Seção [GkLDAP::LDAPAttributeNames]

Esta seção define quais nomes de atributos LDAP usar.

- H323ID
  O alias do endpoint H.323. Precisam ser únicos dentro de uma árvore LDAP (isto é porque nós usamos por default o endereço de email).

- TelephonNo
  O alias E.164 do endpoint.
- voIPIpAddress
  O endereço IP comparado deve ser comparado quando usando o LDAPAliasAuth Por hora, somente um único valor é permitido aqui.
- H235PassWord
  A senha em formato texto a ser comparada quando usarmos o H.235 (LDAPPasswordAuth in Gatekeeper::Auth). Por hora, somente um único valor é permitido aqui.

4.2.25 Seção [GkLDAP::Settings]

Esta seção define o servidor LDAP e o parâmetros padrão do cliente LDAP a serem usados.

- ServerName
  Default: ldap O nome DNS do servidor LDAP.
- ServerPort
  Default: 389 A porta TCP do servidor LDAP (normalmente 389).
- SearchBaseDN
  Default: o=University of Michigan, c=US Ponto de entrada na estrutura de árvore do servidor LDAP. As pesquisas são realizadas somente abaixo deste nó raiz.
- BindUserDN
  Default: cn=Babs Jensen,o=University of Michigan, c=US Este nome distinguível é usado pelo gatekeeper para conectar ao servidor LDAP. Deixe ele vazio se você quiser fazer o acesso ao servidor LDAP como anônimo.
- BindUserPW
  Default: ReallySecretPassword Se voce especificar o BindUserDN, então aqui estará a senha correspondendo ao usuário que foi conectado aqui.
- sizelimit
  Default: 0 Número máximo de resultados que o servidor pode retornar em uma uma única pesquisa. O gatekeeper espera que cada LDAP possa conter uma ou zero resultados, de modo que este parâmetro é pouco usado.
- timelimit
  Default: 0 Número máximo de segundos que uma pesquisa pode esperar antes de ser considerada como "falha".

4.2.26 Seção [Gatekeeper::Acct]

A seção define uma lista de módulos de contabilização de chamadas. Estes módulos fazem o log de dados CDR (Call Detail Record) para um meio específico de armazenamento. A configuração é muito similar a uma de autenticação de gatekeeper (veja).

Syntax:

```
acctmod=actions

<acctmod> := FileAcct | RadAcct | ...
```

```
<actions> := <control>[;<event>,<event>,...]
<control> := optional | required | sufficient
<event>   := start | stop | update | on | off
```

Esta lista de eventos diz ao gatekeeper quais eventos devem gerar logs CDR para um dado módulo (caso o evento seja suportado pelo módulo):

- start - uma chamada foi iniciada (o que não quer dizer que tenha sido completada - a mensagem Setup foi recebida ou um ACF foi enviado; no caso de modo direto de roteamento),
- update - uma chamada está ativa e uma atualizaçao de período é executada; para refletir a nova duração da chamada,
- stop - uma chamada foi terminada (removida da tabela de chamadas do GK),
- on - o gatekeeper foi iniciado,
- off - o gatekeeper foi parado (terminado).

A geração de um log CDR por um módulo pode resultar em um dos três códigos: ok, fail ou next.

- ok - o log CDR foi registrado com sucesso por este módulo.
- fail - o módulo falhou em registrar o log CDR (se evento=start ou update, a chamada será rejeitada ou terminada).
- next - o log CDR não pôde ser registrado, porém a rejeição/término da chamada não deverá ocorrer.

O processamento dos módulos de contabilização e seus resultados finais podem ser controlados por flags de controle:

- optional - se o módulo não pôde registrar o log CDR, passe para o próximo módulo,
- required - se o módulo não pôde registrar o log CDR, processe os módulos restantes, mas o resultado final será fail,
- sufficient - se o módulo registrou o log CDR com sucesso, pare de processar e retorne o resultado de sucesso; caso contrário, passe para o próximo módulo.

Módulos de contabilização suportados atualmente:

- FileAcct Log CDR em arquivo texto. Registra linhas de status como linhas CDR em um arquivo texto especificado. Este módulo suporta somente o evento de contabilização stop. As opções de configuração são lidas da seção FileAcct.
- RadAcct Este módulo executa contabilização via RADIUS. Suporta todos os tipos de eventos (/bf/start, stop, update, on, off/), entretanto, o update ainda não está completamente implementado. Veja a seção RadAcct para detalhes de configuração.

Um exemplo de configuração de contabilização poderia ser assim (sempre registra CDRs em arquivo texto, tenta enviar dados de contabilização para um servidor RADIUS):

Exemplo:

```
RadAcct=optional;start,stop
FileAcct=required
```

4.2.27 Seção [FileAcct]

Este módulo de contabilização escreve linhas de status compatíveis com linhas CDR em um arquivo texto especificado.

- DetailFile=FULL_PATH_AND_FILENAME
  Default: none Especifica um caminho completo para o arquivo de texto CDR. Caso o arquivo exista e Rotate seja 0, os CDRs serão apendados, caso contrário, o log é rotacionado e os CDRs são armazenados a um novo arquivo criado

- Rotate=0 or 1
  Default: 0 Determina se o arquivo texto CDR deve ser rotacionado a cada restart ou reload do gatekeeper. Se for 0, então os CDRs serão apendados ao arquivo existente. Se for 1, então o arquivo existente é renomeado para CURRENT_FILENAME.YYYYMMDD-HHMMSS, onde YYYYMMDD-HHMMSS é substituido pela data corrente, e os CDRs são escritos em um arquivo vazio.

  Examplo:

```
[FileAcct]
DetailFile=/var/log/gk/cdr.log
Rotate=0
```

4.2.28 Seção [RadAcct]

Este módulo de contabilização envia dados de contabilização para um servidor RADIUS. Sua configuração é quase a mesma utilizada para autenticação via RADIUS.

- Servers=SERVER1[:ACCT_PORT];SERVER2[:ACCT_PORT];...
  Default: N/A Servidores RADIUS aos quais enviar os dados de contabilização. Se nenhuma informação a respeito da porta é dada, o número de porta DefaultAcctPort é utilizado. Se estes atributos não forem configurados, então o arquivo services do SO será consultado a respeito da porta do serviço 'radaccr'. Servidores podem ser enderoços IP ou nomes DNS.

- LocalInterface=IP_OR_FQDN
  Default: N/A Interface de rede local que o cliente RADIUS deve utilizar para se comunicar com os servidores RADIUS.

- RadiusPortRange=10000-11000
  Default: N/A Por default (se esta opção não estiver configurada) o cliente RADIUS alocará portas dinamicamente conforme especificado pelo sistema operacional. Se você quiser restringir ao cliente RADIUS o uso de uma faixa particular de portas - configure este parâmetro.

- DefaultAcctPort=PORT_NO
  Default: An entry for 'radius' service in services file or 1813 Porta padrão a ser usada pelas requisições de contabilização RADIUS, se não sobrepostas pelo atributo Servers.

- SharedSecret=SECRET
  Default: N/A (empty string) Segredo usado para autenticar este GNU GK (cliente NAS) com o servidor RADIUS. Deve ser uma senha criptograficamente forte.

- RequestTimeout=TIMEOUT_MS
  Default: 2000 (miliseconds) Timeout (em milisegundos) para o servidor RADIUS responder uma requisição enviada pelo GNU GK. Se nenhuma resposta for recebida dentro deste período de tempo, o próximo servidor RADIUS será utilizado.

- IdCacheTimeout=TIMEOUT_MS
  Default: 9000 (miliseconds) Timeout (em milisegundos) para o identificador de 8-bits da requisição RADIUS ser único.

- SocketDeleteTimeout=TIMEOUT_MS
  Default: 60000 (miliseconds) - 60 s Timeout para os sockets RADIUS serem fechados.

- RequestRetransmissions=NUMBER
  Default: 2 How many times a single RADIUS request is transmissed to every configured RADIUS server (if no response is received).

  Quantas vezes uma simples requisição será transmitido para cada servidor RADIUS configurado (se nenhuma resposta for recebida).

- RoundRobinServers=BOOLEAN
  Default: 1 O método de retransmissão das requisições RADIUS.

- AppendCiscoAttributes=BOOLEAN
  Default: 0 Se configurado, os atributos RADIUS Especificos da Cisco serão incluidos nas requisições RADIUS (h323-conf-id,h323-call-origin,h323-call-type).

- IncludeEndpointIP=BOOLEAN
  Default: 1 Se configurado, o endereço IP será passado como atributo RADIUS Framed-IP-Address dentro dos Access-Requests.

4.2.29 Seção [NATedEndpoints]

O gatekeeper pode automaticamente detectar se um endpoint está atrás de NAT. Entretanto, se a detecção falhar, você pode especificar esta manualmente com esta seção.

Formato:

```
alias=true,yes,1,...
```

Exemplo:

Especificar um endpoint com alias 601 estando atrás do NAT.

```
601=true
```

4.2.30 Seção [CTI::Agents]

Esta seção permite configurações para filas virtuais que permitem a distribuição da chamada para uma aplicação externa. Uma fila virtual pode ser um alias H.323 que pode ser chamado como um endpoint.

Na chegada de um ARQ em uma fila virtual, o gatekeeper sinaliza um RouteRequest na porta de status e espera que uma aplicação externa responda com tanto uma rejeição (então o ARQ será rejeitado) ou com RouteToAlias que fará com que o ARQ seja reescrito de modo que a chamada seja roteada para o alias (ex. agente de call center) especificado pela aplicação externa.

Se nenhuma resposta for recebida após um período de timeout, a chamada será finalizada.

Você pode ter múltiplas filas virtuais. Apenas é preciso separa os nomes com virgulas (sem espaços!), ex. VirtualQueue=hotline,sales.

Verifique a seção de monitoramento para mais detalhes destas mensagens e respostas.

- VirtualQueue
  Default: none Isto define os aliases H.323 para as filas virtuais.
- CTI_Timeout
  Default: 10 Timeout em segundos para aplicações externas responderem o RouteRequest. Se nenhuma resposta for recebida durante este tempo um ARJ será enviado para o chamador.

5 Monitorando o Gatekeeper (Referencia)

5.1 Interface Status

A Interface Status é a interface externa para monitoramento e controle do gatekeeper. O gatekeeper irá enviar mensagens a respeito sobre as chamadas de todos os clientes conectados e irá receber comandos via esta interface.

A interface é uma porta TCP única (default: 7000), você pode conectar com o telnet ou com outro cliente. Um exemplo de um cliente diferente é o Java GUI, aka GkGUI.

5.1.1 Areas de Aplicação

O que você pode fazer com os poderes da Interface de Status é com você. Mas aqui estão algumas idéias possíveis:

- Monitoramento de Chamadas
- Monitoramento dos endpoints registrados
- Graphical User Interface

  Veja GkGUI.
- Aplicações de Contabilização de Chamada

  Analise as mensagens CDR e as redirecione para uma aplicação de billing.
- Interface com extensões externas

  Se não quiser publicar o código fonte deste recursos adicionais, apenas publique o núcleo de funcionalidades e a interface que é preciso com a interface de status e mantenha privado esta parte externa do código.

5.1.2 Exemplos

Suponha que você esteja interessado nos CDRs (registros detalhados das chamadas) e você quer processar eles em intervalos regulares.

Aqui temos um script Perl (gnugk_cdr.pl) que inicia o gatekeeper e replica um cliente simplificado para a Interface de Status e escreve todos os CDRs em um arquivo de logs.

```
#!/usr/bin/perl
# programa de exemplo para demonstrar como escrever CDRs para um arquivo de log
use strict;
use IO::Socket;
use IO::Handle;

my $logfile = "/home/jan/cdr.log";
my $gk_host = "localhost";
my $gk_port = 7000;
my $gk_pid;

if ($gk_pid = fork()) {
        # processo pai irá ouvir o status da gatekeeper
        sleep(1);       # aguarda o gk inicializar
        my $sock = IO::Socket::INET->new(PeerAddr => $gk_host, PeerPort => $gk_port, Proto => 'tcp')
        if (!defined $sock) {
                die "Can't connect to gatekeeper at $gk_host:$gk_port";
        }
        $SIG{HUP} = sub { kill 1, $gk_pid; };   # pass HUP to gatekeeper
        $SIG{INT} = sub { close (CDRFILE); kill 2, $gk_pid; };  # close file when terminated

        open (CDRFILE, ">>$logfile");
        CDRFILE->autoflush(1);  # don't buffer output
        while (!$sock->eof()) {
                my $msg = $sock->getline();
                $msg = (split(/;/, $msg))[0];   # remove junk at end of line
                my $msgtype = (split(/\|/, $msg))[0];
                if ($msgtype eq "CDR") {
                        print CDRFILE "$msg\n";
                }
        }
        close (CDRFILE);
} else {
        # processo filho inicia o gatekeeper
        exec("gnugk");
}
```

5.1.3 Interface Gráfica (GUI) do Gatekeeper

Existem diversos frontend gráficos (GUI) para o gatekeeper.

- Java GUIDesenvolvido por Jan Willamowius. Você pode monitorar os registros e chamadas que passam pelo gatekeeper. Um clique-direito em um botão apresenta um menu popup para aquele endpoint.

  Esta GUI funciona com Java 1.0 ambiente implementado em muitos browsers web. Por razões de segurança o GUI deve ser executado como uma aplicação standalone ou executado por um servidor web no mesmo número IP que o gatekeeper (você não poderá executar o applet a partir do arquivo local).

  O programa está disponível em *H323GUI* <http://www.gnugk.org/h323gui.html>¡C

- GkGUIUm programa standalone em Java desenvolvido por *Citron Network Inc.* <http://www.citron.com.tw/> Ele requer Java 1.4. E os seus recursos incluem:
  - Monitorar múltiplos gatekeepers simultaneamente.
  - Dois modos de visalização: Lista de Botões e Árvore.
  - Registros das Chamadas (CDR) e estatisticas.
  - Log da Interface de Status do GK.
  - Diferentes cores para diferentes tipos de endpoints.
  - Modificar a configuração do gatekeeper.
  - Forçar o desregistro dos endpoints.
  - Armazenar e imprimir os logs do status e os CDR.

  O GkGUI está sob a licença GNU General Public License, disponível em <http://www.gnugk.org/h323develop.html#java>¡C

5.2 Comandos (Referencia)

O comando help ou h irá mostrar uma lista de todos os comandos disponíveis.

- Reload Recarrega a configuração.
- Version, v Mostra a versão e a informação sobre o S.O. do gatekeeper.
- Statistics, s Mostra informações estatísticas do gatekeeper.

  Exemplo:

```
Statistics
-- Endpoint Statistics --
Total Endpoints: 21  Terminals: 17  Gateways: 4  NATed: 2
Cached Endpoints: 1  Terminals: 1  Gateways: 0
-- Call Statistics --
Current Calls: 1 Active: 1 From Neighbor: 0 From Parent: 0
Total Calls: 1539  Successful: 1076  From Neighbor: 60  From Parent: 5
Startup: Fri, 21 Jun 2002 10:50:22 +0800  Running: 11 days 04:22:59
;
```

- PrintAllRegistrations, r, ? Mostra todos os endpoints registrados.

  Formato:

```
AllRegistrations
RCF|IP:Port|Aliases|Terminal_Type|EndpointID
...
Number of Endpoints: n
;
```

  Example:

```
AllRegistrations
RCF|10.1.1.10:1720|800:dialedDigits=Wei:h323_ID|terminal|1289_endp
RCF|10.0.1.43:1720|613:dialedDigits=Jacky Tsai:h323_ID|terminal|1328_endp
RCF|10.0.1.55:1720|705:dialedDigits=Sherry Liu:h323_ID|terminal|1333_endp
Number of Endpoints: 3
;
```

- PrintAllRegistrationsVerbose, rv, ?? Mostra detalhes de todos os endpoints registrados.

  Formato:

  ```
  AllRegistrations
  RCF|IP:Port|Aliases|Terminal_Type|EndpointID
  Registration_Time C(Active_Call/Connected_Call/Total_Call) <r>
  [Prefixes: ##] (gateway only)
  ...
  Number of Endpoints: n
  ;
  ```

  Exemplo:

  ```
  AllRegistrations
  RCF|10.0.1.8:1720|Accel-GW2:h323_ID|gateway|1322_endp
  Wed, 26 Jun 2002 16:40:03 +0800 C(1/5/33) <1>
  Prefixes: 09,002
  RCF|10.1.1.10:1720|800:dialedDigits=Wei:h323_ID|terminal|1289_endp
  Wed, 26 Jun 2002 16:40:55 +0800 C(0/32/39) <1>
  RCF|10.0.1.66:1720|716:dialedDigits=Vicky:h323_ID|terminal|1425_endp
  Wed, 26 Jun 2002 16:40:58 +0800 C(1/47/53) <1>
  Number of Endpoints: 2
  ;
  ```

- PrintCurrentCalls, c, ! Mostra todas as chamadas correntes.

  Formato:

  ```
  CurrentCalls
  Call No. # | CallID | Call_Duration | Left_Time
  Dialed_Number
  ACF|Caller_IP:Port|Caller_EPID|CRV
  ACF|Callee_IP:Port|Callee_EPID|CRV
  ...
  Number of Calls: Current_Call Active: Active_Call From Neighbor: Call_From_Neighbor \
  From Parent: Call_From_Parent
  ;
  ```

  Exemplo:

  ```
  CurrentCalls
  Call No. 29 | CallID bd c6 17 ff aa ea 18 10 85 95 44 45 53 54 77 77 | 109 | 491
  Dial 0953378875:dialedDigits
  ACF|10.0.1.49:1720|4048_CGK1|25263
  ACF|10.1.1.1:1720|4037_CGK1|25263
  Call No. 30 | CallID 70 0e dd c0 9a cf 11 5e 00 01 00 05 5d f9 28 4d | 37 | 563
  Dial 0938736860:dialedDigits
  ACF|10.0.1.48:1032|4041_CGK1|11896
  ACF|10.1.1.1:1720|4037_CGK1|11896
  Number of Calls: 2 Active: 2 From Neighbor: 0 From Parent: 0
  ;
  ```

- PrintCurrentCallsVerbose, cv, !! Mostra detalhes de todas as chamadas correntes.

  Formato:

  ```
  CurrentCalls
  Call No. # | CallID | Call_Duration | Left_Time
  Dialed_Number
  ```

```
ACF|Caller_IP:Port|Caller_EPID|CRV
ACF|Callee_IP:Port|Callee_EPID|CRV
# Caller_Aliases|Callee_Aliases|Bandwidth|Connected_Time <r>
...
Number of Calls: Current_Call Active: Active_Call From NB: Call_From_Neighbor
;
```

Exemplo:

```
CurrentCalls
Call No. 48 | CallID 7d 5a f1 0a ad ea 18 10 89 16 00 50 fc 3f 0c f5 | 30 | 570
Dial 0225067272:dialedDigits
ACF|10.0.1.200:1720|1448_endp|19618
ACF|10.0.1.7:1720|1325_endp|19618
# Sherry:h323_ID|Accel-GW1:h323_ID|200000|Wed, 26 Jun 2002 17:29:55 +0800 <2>
Number of Calls: 1 Active: 1 From NB: 0
;
```

- Find, f Encontrar um endpoint registro a partir de um alias ou prefixo.

  Formato:

  ```
  Find Alias
  RCF|IP:Port|Aliases|Terminal_Type|EndpointID
  ;
  ```

  Exemplo:

  ```
  f 800
  RCF|10.1.1.10:1720|800:dialedDigits=Wei:h323_ID|terminal|1289_endp
  ;
  f 801
  SoftPBX: alias 801 not found!
  ```

- FindVerbose, fv Encontra detalhes de um endpoint registrado com um alias ou um prefixo.

  Formato:

  ```
  FindVerbose Alias
  RCF|IP:Port|Aliases|Terminal_Type|EndpointID
  Registration_Time C(Active_Call/Connected_Call/Total_Call) <r>
  [Prefixes: ##] (gateway only)
  ;
  ```

  Exemplo:

  ```
  fv 02
  RCF|10.0.1.100:1720|TFN:h323_ID|gateway|4037_CGK1
  Wed, 26 Jun 2002 17:47:29 +0800 C(0/84/120) <1>
  Prefixes: 02,09
  ;
  ```

- UnregisterIP Força o desregistro de um endpoint pelo endereço IP e porta de sinalização.

  Formato:

  ```
  UnregisterIP IP[:Port]
  ```

  Example:

```
UnregisterIP 10.0.1.31:1720
URQ|10.0.1.31:1032|1326_endp|maintenance;
SoftPBX: Endpoint 10.0.1.31:1720 unregistered!
```

- UnregisterAlias Força o desregistro de um endpoint por um ou mais dos seus aliases.

  Formato:

  ```
  UnregisterAlias Alias
  ```

  Exemplo:

  ```
  UnregisterAlias 601
  URQ|10.0.1.31:1032|1326_endp|maintenance;
  SoftPBX: Endpoint 601 unregistered!
  ```

- UnregisterAllEndpoints Força o desregistro de todos os endpoints registrados.

  Formato:

  Example:

  ```
  UnregisterAllEndpoints
  URQ|10.0.1.7:1024|1325_endp|maintenance;
  URQ|10.0.1.8:1024|1322_endp|maintenance;
  URQ|10.0.1.32:1032|1324_endp|maintenance;
  URQ|10.0.1.36:1032|1323_endp|maintenance;
  URQ|10.0.1.42:1032|1318_endp|maintenance;
  Done
  ;
  ```

- DisconnectCall Desconecta uma chamada dado um certo número.

  Formato:

  ```
  DisconnectCall Number
  ```

  Exemplo:

  ```
  DisconnectCall 1533
  ```

- DisconnectIP Desconecta todas as chamada de um endpoint por IP e porta de sinalização.

  Formato:

  ```
  DisconnectIP IP[:Port]
  ```

  Exemplo:

  ```
  DisconnectIP 10.0.1.31:1720
  ```

- DisconnectAlias Desconecta todas as chamada de um endpoint por um ou mais dos seus aliases.

  Formato:

  ```
  DisconnectAlias Alias
  ```

  Exemplo:

  ```
  DisconnectAlias 601
  ```

- ClearCalls Desconecta todas as chamadas do gatekeeper.

- GK Mostra informações sobre o gatekeeper pai.
- Debug Somente usado para propósitos de debug. Opções:
  - trc [+|-|n] Mostra/Modifica o nível de debug.
  - cfg SEC PAR Lê ou imprime um parâmetro de configuração em uma seção.
  - set SEC PAR VAL Escreve um valor de um parâmetro de configuração em uma seção.
  - remove SEC PAR Remove o valor de um parâmetro de configuração de uma seção.
  - remove SEC Remove uma seção.
  - printrm VERBOSE Imprime todos os registros removidos de endpoints.

  Exemplo:

  ```
  debug trc 3
  debug set RoutedMode H245Routed 1
  ```

- Who Mostra todas as pessoas na porta de status.
- RouteReject Termina esta chamada que está em uma fila virtual.

  Formato:

  ```
  RouteReject CallingEndpointID CallRef
  ```

  Exemplo:

  ```
  RouteReject endp_4711 1234
  ```

- RouteToAlias, rta Roteia esta chamada em uma fila virtual para o alias especificado.

  Formato:

  ```
  RouteToAlias Alias CallingEndpointID CallRef
  ```

  Exemplo:

  ```
  RouteToAlias Suzi endp_4711 1234
  ```

- Exit, q Desconecta da porta de status.

5.3 Mensagens (Referencia)

Esta seção descreve as saídas das mensagens da interface de status.

- GCF|IP|Aliases|Endpoint_Type; O gatekeeper recebe um GatekeeperRequest (GRQ) e responde com um GatekeeperConfirm (GCF).
- GRJ|IP|Aliases|Endpoint_Type|RejectReason; O gatekeeper recebe um GatekeeperRequest (GRQ) e responde com um GatekeeperReject (GRJ).
- RCF|IP:Port|Aliases|Endpoint_Type|EndpointID; O gatekeeper recebe um RegistrationRequest (RRQ) e responde com um RegistrationConfirm (RCF).
- RRJ|IP|Aliases|Endpoint_Type|RejectReason; O gatekeeper recebe um RegistrationRequest (RRQ) e responde com um RegistrationReject (RRJ).
- ACF|Caller_IP:Port|Caller_EndpointID|CRV|DestinationInfo|SrcInfo|IsAnswered; O gatekeeper recebe um AdmissionRequest (ARQ) e responde com um AdmissionConfirm (ACF).

- ARJ|Caller_IP:Port|DestinationInfo|SrcInfo|IsAnswered|RejectReason; O gatekeeper recebe um AdmissionRequest (ARQ) e responde com um AdmissionReject (ARJ).
- DCF|IP|EndpointID|CRV|DisengageReason; O gatekeeper recebe um DisengageRequest (DRQ) e responde com um DisengageConfirm (DCF).
- DRJ|IP|EndpointID|CRV|RejectReason; O gatekeeper recebe um DisengageRequest (DRQ) e responde com um DisengageReject (DRJ).
- LCF|IP|EndpointID|DestinationInfo|SrcInfo; O gatekeeper recebe um LocationRequest (LRQ) e responde com um LocationConfirm (LCF).
- LRJ|IP|DestinationInfo|SrcInfo|RejectReason; O gatekeeper recebe um LocationRequest (LRQ) e responde com um LocationReject (LRJ).
- BCF|IP|EndpointID|Bandwidth; O gatekeeper recebe um BandwidthRequest (BRQ) e responde com um BandwidthConfirm (BCF).
- BRJ|IP|EndpointID|Bandwidth|RejectReason; O gatekeeper recebe um BandwidthRequest (BRQ) e responde com um BandwidthReject (BRJ).
- UCF|IP|EndpointID; O gatekeeper recebe um UnregistrationRequest (URQ) e responde com um UnregistrationConfirm (UCF).
- URJ|IP|EndpointID|RejectReason; O gatekeeper recebe um UnregistrationRequest (URQ) e responde com um UnregistrationReject (URJ).
- IRQ|IP:Port|EndpointID; O gatekeeper envia um InfoRequest (IRQ) para o endpoint para verificar se ele ainda está ativo. O endpoint deverá responder com um InfoRequestResponse (IRR) imediatamente.
- URQ|IP:Port|EndpointID|Reason; O gatekeeper envia um UnregistrationRequest (URQ) para um endpoint para cancelar o seu registro. O endpoint deverá responder com um UnregistrationConfirm (UCF).
- CDR|CallNo|CallId|Duration|Starttime|Endtime|CallerIP|CallerEndId| \ CalledIP|CalledEndId|DestinationInfo|SrcInfo|GatekeeperID; Após a ligação ter sido desconectada, o registro dos detalhes da chamada é mostrado (em uma linha).
- RouteRequest|CallerIP:Port|CallerEndpointId|CallRef|VirtualQueue|CallerAlias; Requer que uma aplicação externa roteie uma chamada vindo de uma fila virtual.