Exemplo de configuração dos keepalive de script de WebNS 4.0

Índice

Introdução

Este documento descreve a implementação inicial dos keepalive de script. Este método do script é o mais estreitamente relacionado à funcionalidade que esta presente em clientes dialup da confiança, da Disponibilidade, e da utilidade (RAS), em programas terminal, e em utilidades gerais do script. Esta característica utiliza o linguagem de script rico do CSS.

Configurar isto com um soquete simples API (conecte/disconexão/enviam/recebem) dará ao usuário a capacidade para costurar seu próprio protocolo, ou escreve sua própria sequência das etapas para fornecer um `VIVO` ou um `estado inativo` seguro de um serviço. Você é limitado atualmente ao FTP, ao HTTP, ao ICMP, e ao TCP. Com estes novos recursos, você pode permanecer sobre os protocolos atual escrevendo seus próprios scripts. Por exemplo, um usuário pode desenvolver um script tonificado especificamente para conectar a um server POP3 sem exigir Cisco construir um tipo keepalive POP3 para serir suas necessidades. Esta característica permitirá que os clientes criem seu próprio Keepalives feito sob encomenda para serir suas exigências específicas.

Antes de Começar

Convenções

Para obter mais informações sobre convenções de documento, consulte as Convenções de dicas técnicas Cisco.

Pré-requisitos

Não existem requisitos específicos para este documento.

Componentes Utilizados

Este documento aplica-se a CSS 11000/CSS 11500 ou a CSS11800 com software webns.

As informações neste documento foram criadas a partir de dispositivos em um ambiente de laboratório específico. Todos os dispositivos utilizados neste documento foram iniciados com uma configuração (padrão) inicial. Se você estiver trabalhando em uma rede ativa, certifique-se de que entende o impacto potencial de qualquer comando antes de utilizá-lo.

Material de Suporte

Uma vez que você desenvolveu um script do comando line interface(cli) que usa off line um editor de texto tal como o bloco de notas, você pode transferir arquivos pela rede esse script ao diretório de /script do CSS e configurar a opção de keepalive do script em um serviço. É permitido você criar um keepalive do script sem ter um script atual no sistema. No caso de um keepalive do script que é executado sem um script no sistema, um `estado inativo` constante permanecerá no serviço. Isto permite que um administrador escreva uma configuração e execute a configuração antes de escrever para fora (ou dos transferir arquivos pela rede) todos os scripts.

Um script deve residir no/<current que executa o diretório version>/script/para que o keepalive do script encontre o script. Os nomes de caminho não são aceitados, simplesmente nomes de script ao configurar o keepalive do script. Se o script esta presente em outra parte no sistema, o keepalive do script suporá que não existe. Este os meios, contudo, quando o software é promovido no sistema, os scripts velhos residem no diretório do script da versão velha. Para copiar os scripts do diretório velho, veja a seção Copiando Arquivos de Script deste documento.

Você pode passar os caráteres 128 em um argumento citado. Supondo um padrão de sete caracteres por argumento, você pode obter aproximadamente 18 argumentos em um script. A linha de comando aceitará somente 90 caráteres contudo.

Recomenda-se que o Keepalives do script esteja configurado com uma frequência mais baixa do que um keepalive padrão porque muitos scripts terão um processo da multi-etapa tal como a conexão, enviando um pedido, e a espera de um tipo específico de resposta. Devido a isto, recomenda-se ter segundos de uma frequência de dez ou mais alto de modo que o keepalive tenha o tempo para terminar completamente. Se não, as transições de estado podem ocorrer mais frequentemente.

Um script pode retornar um código de status de zero ou de diferente de zero. Em um retorno de diferente de zero, o CSS embandeirará o estado do serviço como `PARA BAIXO`. Em um retorno de zero, o CSS embandeirará o estado do serviço como `VIVO`. Refira o seguinte script para um exemplo:

```
!--- Connect to the remote host. socket connect host einstein port 25 tcp !--- Validate that you did connect. if $SOCKET "==" "-1" exit script 1 endbranch
```

O CSS retornará sempre um estado de `PARA BAIXO` se a variável `$ {SOQUETE}` é ajustada ao negativo um. É muito importante verificar a lógica de seus scripts para assegurar-se de que o valor correto obtenha retornado.

Configurar

Nesta seção, você encontrará informações para configurar os recursos descritos neste documento.

Para um grande número serviços que exigem o uso do Keepalives do script, é altamente recomendado que usam um subconjunto de keepalives globais menor para segurar o trabalho para ele.

Para configurar um keepalive do script, siga as mesmas diretrizes que para todos tipos keepalive restantes. A sintaxe de comando é mostrada abaixo.

```
CS100(config-service[serv1)# keepalive type script ap-kal-smtp "einstein"
```

ou

```
CS100(config-service[serv1])# keepalive type script ap-kal-pop3 "einstein vxworks mipspci"
```

A primeira linha configurará o keepalive atual do serviço para ser do tipo **script** e para ter o nome de script ap-kal-**S TP** com argumento einstein. A segunda linha mostras **ap-kal-pop3**, que é similar ao primeiro script, mas passará três argumentos: **einstein**, **vxworks**, e **mipspci**. É igualmente possível bater a chave da **aba** (ou**?**) para ver que de toda uma lista completa passa pelo processo de script disponível no/<current que executa o diretório version>/script. Isto incentiva-o adotar uma convenção de nomeação do script de modo que quando você bate a **aba** ou**?** , você pode claramente ver os scripts do keepalive disponíveis para usar-se. Uma convenção de nomeação do ap-kal-**tipo** é usada para scripts. Por exemplo, um script S TP seria nomeado ap-kal-**S TP**. Opcionalmente, você pode datilografar dentro um não encontrado nessa lista, supondo o desejo para transferi-lo arquivos pela rede em um outro dia. Os scripts podem ser manipulados através dos **comandos archive, clear, e copy**, e podem ser transferência de arquivo pela rede/transferência do diretório de /script no interruptor.

O nome de script pode ser até 32 caráteres por muito tempo. Os argumentos devem ser passados em um string entre aspas, e podem ser até os caráteres 128 por muito tempo. Para desabilitar o keepalive do script, você reconfiguraria o serviço emitindo o comando seguinte:

```
CS100(config-service[cs100])# keepalive type none
```

Vendo um Keepalive do script em um serviço

Quando um keepalive do script é configurado sob um serviço, o nome de script pode ser considerado na saída do **comando show service**. O script aparece sob o tipo keepalive, e todos os argumentos potenciais podem ser encontrados diretamente sob essa linha nos `argumentos de script:` campo. Se não há nenhum argumento de script, esse campo não está indicado. Um exemplo do serviço **cs100** com um script chamou **ap-kal-pop3** com os **vxworks dos** argumentos e o **mipspci** é mostrado abaixo.

Com esta potência, um script como **ap-kal-pop3** poderia tomar três parâmetros, que combinariam o hostname, o username, e a senha para um server POP3. Simplesmente entrar ao server POP é bastante para que o script determine que este server está no estado `VIVO`.

A saída do **comando show running-config** é mostrada abaixo.

A saída acima do comando mostra exatamente que script (e argumentos) foram configurados neste serviço. Se nenhum argumento esta presente, o texto citado depois do nome de script não esta presente.

Ao trabalhar com logging de script, a configuração é como segue:

Este exemplo mostra o serviço que registra seu keepalive do script output a **testlog.log**.

Primitivos de soquete

Há alguns **comandos socket** que podem ser usados em um keepalive do script para ajudar em construir um protocolo estruturado. Os primitivos de soquete permitirão o ASCII ou o hexadecimal enviam e recebem a funcionalidade. Cada comando que tem um palavra-chave RAW opcional mudará os dados de um padrão ASCII a uma conversão hexadecimal. Por exemplo, o **abcd no** ASCII seria representado por **61626364**, que denota 0x61 0x62 0x63 0x64.

Refira a seguinte lista de **comando socket** para mais informação:

- **o soquete conecta a** *porta* **tcp da porta do** *hostname do* **host | [session] do [integer] UDP**- este comando executa um TCP ou a conexão de UDP. Executar uma conexão de TCP envolve um aperto de mão (o SYN-SYNACK…) a um IP/port específico. Executar uma conexão de UDP é meramente uma reserva da /porta do host. O valor do soquete é recebido em uma variável `$ {SOQUETE} no` script.**Nota:** Somente 32 soquetes podem ser abertos a qualquer altura através de todos os scripts no interruptor.A lista abaixo fornece a informação em cada parâmetro.palavra-chave do **Host-a** que deve ser seguida pelo hostname ou pelo endereço IP de Um ou Mais Servidores Cisco ICM NT do sistema remoto.**porta** - palavra-chave que deve ser seguida pela porta em que para negociar uma conexão com.**tcp** - uma conexão usando o TCP.**UDP** - uma conexão usando o UDP.**inteiro** - um valor de timeout para o estabelecimento de rede nos segundos. Se o limite de tempo expira antes que a conexão esteja feita com sucesso, a tentativa falha. Isto aplica-se somente a uma conexão de TCP, porque o UDP é sem conexão. O valor padrão é um intervalo de cinco segundos.**sessão** - uma palavra-chave que diga o soquete para permanecer aberto até que a sessão estiver terminada. Isso significa que nenhuns scripts que os soquetes abertos na sessão e não os fecham no seus próprios permanecerão abertos até os log de usuário para fora.
- **o soquete envia a corda do socket- [crua | base64]**- este comando redige dados através de uma conexão de TCP previamente conectada.A lista abaixo fornece a informação em cada parâmetro.**socket-** - o descritor de arquivo do soquete (formulário do inteiro). Este descritor é retornado de conecta.**corda** - sequência de caracteres de texto citada até os caráteres 128 de comprimento.**cru** - se especificado, faz com que os valores de série sejam transferidos como bytes hexadecimais reais um pouco do que uma corda simples. Por exemplo, **0D0A** não é enviado como “0" “D” “0" “A,” mas um pouco como **0x0D 0x0A** (ou tecla semelhante a tecla ENTER, alimentação de linha).**base64** - isto base64 codificará a corda antes de enviá-la através da conexão. Útil para a autenticação básica HTTP quando conectar a uma senha protegeu o site.
- **o soquete recebe o [raw] do [integer] do socket-**- este comando enche acima o buffer interno 10K com os dados que vêm dentro do host remoto. Este comando trava então o buffer de modo que nenhum dados novo seja posto neste buffer interno. Você pode continuar ao uso

inspeciona para despejar todos os dados que residem neste buffer 10K interno à saída padrão. **Nota:** Todos os dados precedentes no buffer interno 10K são lavados antes que os dados novos estejam postos dentro.A lista abaixo fornece a informação em cada parâmetro.**socket-** - o descritor de arquivo do soquete (formulário do inteiro). Este descritor é retornado de conecta.**inteiro** - um valor de número inteiro que representa o número de milissegundos para esperar antes de travar o buffer 10K interno e de retornar ao usuário. Se nenhuma hora do inteiro para fora não é especificada, receba esperará 100ms antes de retornar ao usuário.**cru** - se especificado, faz com que os valores de série sejam transferidos como bytes hexadecimais reais um pouco do que uma corda simples. Por exemplo, **0D0A** não é enviado como “0" “D” “0" “A,” mas um pouco como **0x0D 0x0A** (ou tecla semelhante a tecla ENTER, alimentação de linha).

- **[raw] do [offset] do [case-sensitive] do [integer] da corda do socket- do waitfor de soquete**- este comando é similar ao soquete recebe, salvo que retorna imediatamente em cima de encontrar o argumento da série especificada. Uma vez que a série especificada é encontrada, retornará `$ {ESTADO} de` zero. Se não, retorna 1. Os dados recuperados podem mais ser vistos emitindo o **comando socket inspect**.A lista abaixo fornece a informação em cada parâmetro.**socket-** - o descritor de arquivo do soquete (formulário do inteiro). Este descritor é retornado de conecta.**corda** - a corda específica que deve ser encontrada para conduzir a `$ {ESTADO} de` zero, que indica encontrado. Se a corda é encontrada, retorna imediatamente e não espera o tamanho do integer timeout inteiro.**inteiro** - um valor de número inteiro que representa o número de milissegundos para esperar antes de travar o buffer 10K interno e de retornar ao usuário. Se nenhuma hora do inteiro para fora não é especificada, receba esperará 100ms antes de retornar ao usuário.**diferencia maiúscula e minúscula** - se especificado, indica que a comparação de série deve ser diferenciando maiúsculas e minúsculas. Por exemplo, **usuário:** não seja equivalente ao **usuário:**.**offset** - quantos bytes nos dados recebidos a corda deve ser encontrada. Por exemplo, se você procura por **a0** e dá um offset de dez, você procurará os dez bytes **a0 nos** dados recebidos.**cru** - se especificado, faz com que os valores de série sejam transferidos como bytes hexadecimais reais um pouco do que uma corda simples. Por exemplo, **0D0A** não é enviado como “0" “D” “0" “A,” mas um pouco como **0x0D 0x0A** (ou tecla semelhante a tecla ENTER, alimentação de linha).
- **o soquete inspeciona o [raw] do [pretty] do socket-**- este comando inspeciona o buffer de dados do soquete (interno) para dados reais. Se os dados são encontrados, esses dados estão indicados à saída padrão. Se os caráteres indicados são NON-imprimíveis, estarão representados por um ponto (.) para a legibilidade.A lista abaixo fornece a informação em cada parâmetro.**socket-** - o descritor de arquivo do soquete (formulário do inteiro). Este descritor é retornado de conecta.**cru** - se especificado, faz com que os valores de série sejam indicados como bytes hexadecimais reais um pouco então uma corda simples. Um pouco então imprimindo o **ABCD ao** padrão para fora, imprimiria **41424344** (1 equivalente hexadecimal de byte).**consideravelmente** - bela impressão da saída. Cada linha conterá o ASCII ou o equivalente hexadecimal para cada byte de dados. Haverá 16 bytes impressos em cada linha. Por exemplo, **0x41 0x42 0x43 0x44 0x10 0x05 ABCD**.
- **[graceful] do socket- da disconexão do soquete**- este comando fecha a conexão ao host remoto. Isto é feito enviando o RST ao host remoto de modo que saiba que você está feito que envia dados.A lista abaixo fornece a informação em cada parâmetro.**socket-** - o descritor de arquivo do soquete (formulário do inteiro). Este descritor é retornado de conecta.**gracioso** - disconexão graciosa que envia um FIN um pouco do que um RST ao host remoto para fechar a conexão.

A administração de soquete

Há uma limitação de 32 (no uso) soquetes abertos no interruptor a qualquer altura. Se um usuário faz um soquete conecta e não termina com uma disconexão do soquete para o descritor de arquivo desse soquete (salvar em `$ {SOQUETE}`), o soquete permanece aberto até que uma disconexão do soquete esteja chamada com esse soquete como o parâmetro. Os soquetes abertos dentro dos scripts estão fechados quando o script termina (a menos que o argumento de sessão está passado no soquete conecta). Os soquetes abertos dentro das sessões são fechados quando a sessão termina.

Se um soquete permanece aberto, este é geralmente um caso onde o usuário faça uma conexão sem corretamente se fechar. Que o soquete permanecerá aberto e usado até que ele é fechado ou até o script (ou a conexão terminal) foi fechado. **O comando show sockets** foi executado alistar todos os descritores de arquivo usados do soquete que são atualmentes em uso de modo que o usuário conheça o que estão aberto e o que é fechado.

Nota: Se um host remoto cronometra para fora um soquete, ou o soquete está fechado por um host remoto, a arquitetura de soquete é esperta bastante limpá-lo, e toma-o fora da lista de soquetes usados (encontrados emitindo o **comando show sockets**). Isto ocorrerá somente depois que um usuário tenta fazer uma outra transferência em cima de um soquete que esteja fechado pelo host remoto (se não senta a quietude que espera as necessidades de usuários).

A saída do **comando show sockets** é mostrada abaixo.

A tela alista o ID de soquete (descritor de arquivo), pares da /porta do host conectado, usuário, e um temporizador de quanto tempo o descritor esteve aberto. O campo do `usuário` representaria a linha identificador como visto no **comando show lines** output no CLI.

Recuperando os dados que usam o soquete receba pode proteger ao mesmo tempo o valor 10K dos dados. Este buffer permanecerá inalterado até que o usuário faça um outro soquete receber, que no ponto o buffer está limpado para fora e reenchido com mais dados que vêm fora o fio. Cada descritor de soquete (criado do soquete conecte) terá seu próprio buffer 10K.

Arquivos de script de copi

Ao promover a uma nova versão do código, todos os arquivos de script que você alterou no diretório do script da versão anterior deverão ser copiados à nova versão.

Siga estas etapas antes de promover seu interruptor:

1. FTP ao switch CSS. Use a porta de gerenciamento ou o endereço do circuito VLAN.
2. CD ao diretório do script.
3. Transfira todos os scripts que você editou a sua máquina local.
4. Promova seu interruptor.
5. Transfira arquivos pela rede os scripts de sua máquina local ao diretório novo do script do interruptor.

Passar pelo processo de script amostras

Os seguintes scripts estão disponíveis a fim fornecer-lhe algumas aplicações do padrão. Estas amostras contêm os scripts escritos para o DNS, o eco, o NetBios, o finger, o tempo, o HttpList, o

PingList, o HttpAuth, o Imap4, o CookieSet, o POP3, o HttpTag, o MailHost, e o S TP.

Emita o **script da mostra?** comande para ver os scripts. A saída de exemplo de comando está mostrada abaixo.

```
CS150# show script
  ap-kal-dns          NOV 17 09:58:36          1555
  ap-kal-echo         NOV 17 09:58:36          1920
  ap-kal-finger       NOV 17 09:58:36          1172
  ap-kal-httpauth     NOV 17 09:58:36          1927
  ap-kal-httplist     NOV 17 09:58:36          1674
  ap-kal-httptag      NOV 17 09:58:36          1180
  ap-kal-imap4        NOV 17 09:58:36          1556
  ap-kal-ldap         NOV 17 09:58:36          1640
  ap-kal-mailhost     NOV 17 09:58:36          2437
  ap-kal-netbios      NOV 17 09:58:36          1632
  ap-kal-pinglist     NOV 17 09:58:36           739
  ap-kal-pop3         NOV 17 09:58:36          1568
  ap-kal-setcookie    NOV 17 09:58:38          1436
  ap-kal-smtp         NOV 17 09:58:38          1310
  ap-kal-ssl          NOV 17 09:58:38          2053
  ap-kal-time         NOV 17 09:58:38          1064
  cache.map           NOV 17 09:58:38          1615
  commit_redundancy   NOV 17 09:58:38        109224
  commit_vip_redund.. NOV 17 09:58:38        132147
  default-profile     NOV 17 09:58:38          1240
  dnslookup           NOV 17 09:58:40          8009
  eql-cacheable       NOV 17 09:58:40          1186
  eql-graphics        NOV 17 09:58:40           234
  eql-multimedia      NOV 17 09:58:40           279
  flowinfo            NOV 17 09:58:40          5665
  monitor             NOV 17 09:58:40          3734
  pcm-collect-cfgs    NOV 17 09:58:40          2373
  pcm-repeat-cmd      NOV 17 09:58:40          4995
  service-load        NOV 17 09:58:40           920
  setup               NOV 17 09:58:40         24328
  showtech            NOV 17 09:58:40          2528
  testpeering         NOV 17 09:58:40         34142
  upgrade             NOV 17 09:58:40         17117
  ap-kal-ssl.txt      NOV 24 09:18:00          2053
```

Verificar

No momento, não há procedimento de verificação disponível para esta configuração.

Troubleshooting

Use as seguintes diretrizes para pesquisar defeitos sua configuração:

- Execute o script da linha de comando emitindo o **comando script play**. Emita este comando quando entrado como um superuser para se certificar d termina sem erro. Se faz o erro para fora, a linha que causou o erro deve ser emitida.
- Tome um farejador de rastreamento entre o CSS e o servidor de Web para observar o que é retornado realmente quando o script é executado contra o que o script está esperando ver.
- Nas versões de código 5.x, o **comando use-output** foi adicionado. Este comando deve ser usado com todos os scripts que usam o **comando grep**. Por exemplo, uso-**saída dos**

argumentos do Ap-kal-dns do script do tipo keepalive.

Informações Relacionadas

- Suporte de software do Web Network Services
- Suporte a hardware dos CSS 11000 Series Content Services Switch
- Suporte a hardware dos CSS 11500 Series Content Services Switch
- Downloads do software de Cisco WebNS CSS11500
- Downloads do software de Cisco WebNS CSS11000
- Suporte Técnico - Cisco Systems