Ejemplo de configuración de los keepalives en secuencia de comandos de WebNS 4.0

Contenido

Introducción

Este documento describe la instrumentación inicial de los keepalives en secuencia de comandos. Este método de scripting está el más estrechamente vinculado a las funciones que está presente en los clientes de marcación manual de la confiabilidad, de la Disponibilidad, y de la utilidad (RAS), los programas para terminal, y las utilidades generales del scripting. Esta característica utiliza el lenguaje de la secuenciación de comandos rico CSS.

Configurando esto con un dará al usuario del socket simple API (conecte/desconexión/envían/reciben) la capacidad de adaptar su propio protocolo, o escribió su propia la secuencia de para proporcionar un estado ALIVE o DOWN confiable de algún servicio de los pasos. Le limitan actualmente al FTP, al HTTP, al ICMP, y al TCP. Con esta nueva función, usted puede permanecer encima de los protocolos actuales escribiendo sus propios scripts. Por ejemplo, un usuario puede desarrollar un script entonado específicamente para conectar con un servidor POP3 sin requerir Cisco construir un tipo de keepalive POP3 para adaptarse a sus necesidades. Esta característica permitirá que los clientes creen su propio Keepalives de encargo para adaptarse a sus requisitos específicos.

Antes de comenzar

Convenciones

Para obtener más información sobre las convenciones del documento, consulte Convenciones de Consejos Técnicos de Cisco.

prerrequisitos

No hay requisitos previos específicos para este documento.

Componentes Utilizados

Este documento se aplica a CSS 11000/CSS 11500 o al CSS11800 con el software de WebNS.

La información que se presenta en este documento se originó a partir de dispositivos dentro de un ambiente de laboratorio específico. Todos los dispositivos que se utilizan en este documento se pusieron en funcionamiento con una configuración verificada (predeterminada). Si la red está funcionando, asegúrese de haber comprendido el impacto que puede tener un comando antes de ejecutarlo.

Teoría Precedente

Una vez que usted ha desarrollado un script del comando line interface(cli) off-liné usando un editor de textos tal como libreta, usted puede cargar que script al directorio de /script del CSS y configura la opción de keepalive del script en un servicio. A le se permite crear un keepalive del script sin tener un script presente en el sistema. En el caso de un keepalive del script que se ejecuta sin un script en el sistema, seguirá habiendo un `estado inactivo` constante en el servicio. Esto permite que un administrador escriba una configuración y implemente la configuración antes de poner (o de cargarlos) todos los scripts en escrito.

Un script debe residir en el/<current que ejecuta el directorio version>/script/para que el keepalive del script encuentre el script. Los nombres del trayecto no se validan, solamente los nombres de secuencia de comandos al configurar el keepalive del script. Si el script está presente a otra parte en el sistema, el keepalive del script asumirá que no existe. Este los medios, sin embargo, cuando el software se actualiza en el sistema, los viejos scripts residen en el directorio del script de la versión antigua. Para copiar los scripts del viejo directorio, vea la sección Copia de archivos de secuencias de comandos de este documento.

Usted puede pasar los caracteres 128 en un argumento citado. Si se asume que un estándar de siete caracteres por argumento, usted puede conseguir cerca de 18 argumentos en un script. La línea de comando validará solamente 90 caracteres sin embargo.

Se recomienda que el Keepalives del script esté configurado con una frecuencia más baja que un keepalive estándar porque muchos scripts tendrán un proceso de varias fases tal como conexión, enviando una petición, y esperar un tipo específico de respuesta. Debido a esto, es a tener recomendado frecuencia de diez los segundos o más alto de modo que el keepalive tenga tiempo para acabar totalmente. Si no, las transiciones de estado pueden ocurrir más a menudo.

Un script puede devolver un código de estado de cero o de no-cero. En una vuelta de no-cero, el CSS señalará al estado de servicio por medio de una bandera como `ABAJO`. En una vuelta de cero, el CSS señalará al estado de servicio por medio de una bandera como `VIVO`. Refiera al script siguiente por un ejemplo:

```
!--- Connect to the remote host. socket connect host einstein port 25 tcp !--- Validate that you
```

did connect. `if $SOCKET "==" "-1" exit script 1 endbranch`
El CSS volverá siempre un estado de `ABAJO` si la variable `$ {SOCKET}` se fija a la negativa una. Es muy importante marcar la lógica de sus scripts para asegurarse de que el valor correcto consigue vuelto.

Configurar

En esta sección encontrará la información para configurar las funciones descritas en este documento.

Para un gran número de servicios que requieran el uso del Keepalives del script, se recomienda altamente que utilizan un subconjunto de keepalives globales más pequeño para manejar el trabajo para él.

Para configurar un keepalive del script, siga las mismas guías de consulta que para el resto de los tipos de keepalive. La sintaxis de los comandos se muestra abajo.

```
CS100(config-service[serv1)# keepalive type script ap-kal-smtp "einstein"
```
o
```
CS100(config-service[serv1])# keepalive type script ap-kal-pop3 "einstein vxworks mipspci"
```
La primera línea configurará el keepalive del servicio actual para estar de **script del** tipo y para tener el nombre de secuencia de comandos ap-kal-**S TP** con el argumento einstein. La segunda línea demostraciones **ap-kal-pop3**, que es similar al primer script, pero pasará tres argumentos: **einstein**, **vxworks**, y **mipspci**. Es también posible golpear la clave de la **lengueta** (o**?**) para ver que una lista completa de toda scripts disponible en el/<current que ejecuta el directorio version>/script. **¿**Esto le anima a adoptar a una convención para nombres del script de modo que cuando usted golpea la **lengueta** o**?** , usted puede ver claramente los scripts del keepalive disponibles para utilizar. Utilizan a una convención para nombres del ap-kal-**tipo** para los scripts. Por ejemplo, un script S TP sería nombrado ap-kal-**S TP**. Opcionalmente, usted puede teclear adentro uno no encontrado en esa lista, asumiendo que usted desea cargarla más adelante. Los scripts se pueden manipular a través de los **comandos archive, clear, y copy**, y pueden ser carga/descarga del directorio de /script en el Switch.

El nombre de secuencia de comandos puede ser hasta 32 caracteres de largo. Los argumentos se deben pasar en una cadena apostrofada, y pueden estar hasta los caracteres 128 de largo. Para inhabilitar el keepalive del script, usted configuraría de nuevo el servicio publicando el siguiente comando:

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

Ver un keepalive del script en un servicio

Cuando un keepalive del script se configura bajo servicio, el nombre de secuencia de comandos se puede considerar en la salida del **comando show service**. El script aparece conforme al tipo de keepalive, y cualquier argumento potencial se puede encontrar directamente bajo esa línea en los `argumentos del guión:` campo. Si no hay argumentos del guión, ese campo no se visualiza. Un ejemplo del servicio **cs100** con un script llamó **ap-kal-pop3** con los **vxworks de los** argumentos y el **mipspci** se muestra abajo.

Con este poder, un script como **ap-kal-pop3** podría tomar tres parámetros, que harían juego el

nombre de host, el nombre de usuario, y la contraseña para un servidor POP3. Simplemente la apertura de sesión al servidor POP es bastante para que el script determine que este servidor está en el estado `VIVO`.

La salida del **comando show running-config** se muestra abajo.

La salida de comando antedicha muestra exactamente qué script (y los argumentos) se han configurado en este servicio. Si no hay argumentos presentes, el texto citado después del nombre de secuencia de comandos no estará presente.

Al trabajar con la función de registro de la secuencia de comandos, la configuración es como sigue:

Este ejemplo muestra el servicio que registra su keepalive del script hecho salir a **testlog.log**.

Socketes primitivos

Hay algunos **comandos socket** que pueden ser utilizados en un keepalive del script para ayudar a construir un protocolo estructurado. Los socketes primitivos permitirán el ASCII o el hexadecimal envían y reciben las funciones. Cada comando que tiene una palabra clave opcional de RAW cambiará los datos de un ASCII estándar a una conversión hexadecimal. Por ejemplo, el **abcd** en el ASCII sería representado por **61626364**, que denota 0x61 0x62 0x63 0x64.

Refiera a la lista siguiente de **comando socket** para más información:

- **el socket conecta el** *puerto* **tcp del puerto del** *nombre de host del* **host | UDP [integer] [session]**- este comando realiza un TCP o la conexión UDP. La ejecución de una conexión TCP implica un apretón de manos (el SYN-SYNACK…) a un IP/port específico. La ejecución de una conexión UDP es simplemente una reserva del /port del host. El valor del socket se recibe en una variable `$ {SOCKET}` en el script.**Nota:** Solamente 32 socketes se pueden abrir a cualquier momento a través de todos los scripts en el Switch.La lista abajo proporciona la información sobre cada parámetro.palabra clave del **Host-a** que se debe seguir por el nombre de host o la dirección IP del sistema remoto.**puerto** - palabra clave que se debe seguir por el puerto en con el cual negociar una conexión.**tcp** - una conexión usando el TCP.**UDP** - una conexión usando el UDP.**número entero** - un valor de agotamiento del tiempo para el establecimiento de la red en los segundos. Si expira el límite de tiempo antes de que la conexión se haya hecho con éxito, la tentativa falla. Esto se aplica solamente a una conexión TCP, pues el UDP está sin conexión. El valor predeterminado es un descanso de cinco segundos.**sesión** - una palabra clave que dice el socket seguir siendo abierto hasta que se acabe la sesión. Eso significa que ninguna scripts que los socketes abiertos en la sesión y no los cierran en sus los propio seguirá habiendo abierta hasta que el usuario termine la sesión.
- **el socket envía la cadena del socket- [sin procesar | base64]**- este comando escribe los datos a través de una conexión TCP previamente conectada.La lista abajo proporciona la información sobre cada parámetro.**socket-** - la descripción del archivo del socket (forma del número entero). Este descriptor se vuelve de conecta.**cadena** - cadena de texto citada hasta los caracteres 128 de largo.**sin procesar** - si está especificado, hace los valores de la cadena ser transferido como bytes hexadecimales reales bastante que una cadena simple. Por ejemplo, **0D0A** no se envía como “0" “D” “0" “A,” sino bastante como **0x0D 0x0A** (o retorno de carro, Line Feedavance de línea).**base64** - esto base64 codificará la cadena antes de enviarla a través de la conexión. Útil para la autenticación básica HTTP cuando la conexión con una

contraseña protegió el sitio web.

- **el socket recibe el socket- [integer] [raw]**- este comando llena el búfer interno 10K de los datos que vienen adentro del host remoto. Este comando entonces bloquea el buffer de modo que no se ponga ningunos nuevos datos en este búfer interno. Usted puede proceder al uso examina para vaciar todos los datos que residen en este buffer interno 10K a la salida estándar. **Nota:** Todos los datos anteriores en el búfer interno 10K se enjuagan antes de que se introduzcan los nuevos datos.La lista abajo proporciona la información sobre cada parámetro.**socket-** - la descripción del archivo del socket (forma del número entero). Este descriptor se vuelve de conecta.**número entero** - un valor del número entero que representa el número de milisegundos para esperar antes de bloquear el buffer interno 10K y de volver al usuario. Si no se especifica ninguna hora del número entero hacia fuera, reciba esperará 100ms antes de volver al usuario.**sin procesar** - si está especificado, hace los valores de la cadena ser transferido como bytes hexadecimales reales bastante que una cadena simple. Por ejemplo, **0D0A** no se envía como “0" “D” “0" “A,” sino bastante como **0x0D 0x0A** (o retorno de carro, Line Feedavance de línea).
- **cadena del socket- del socket waitfor [integer] [case-sensitive] [offset] [raw]**- este comando es similar al socket recibe, salvo que vuelve inmediatamente sobre encontrar el argumento de la cadena especificada. Una vez que se encuentra la cadena especificada, volverá `$ {ESTATUS}` de cero. Si no, vuelve 1. Los datos extraídos pueden ser vistos más a fondo publicando el **comando socket inspect**.La lista abajo proporciona la información sobre cada parámetro.**socket-** - la descripción del archivo del socket (forma del número entero). Este descriptor se vuelve de conecta.**cadena** - la cadena específica que se debe encontrar para dar lugar a un `$ {ESTATUS}` de cero, que indica encontrado. Si se encuentra la cadena, vuelve inmediatamente y no espera el entero longitud de tiempo agotado de espera del entero.**número entero** - un valor del número entero que representa el número de milisegundos para esperar antes de bloquear el buffer interno 10K y de volver al usuario. Si no se especifica ninguna hora del número entero hacia fuera, reciba esperará 100ms antes de volver al usuario.**caso sensible** - si está especificado, indica que la comparación de cadenas debe ser con diferenciación entre mayúsculas y minúsculas. Por ejemplo, **usuario:** no sea equivalente al **usuario:**.**desplazamiento** - cuántos bytes en los datos recibidos la cadena debe ser encontrada. Por ejemplo, si usted busca para el **a0** y da un desplazamiento de diez, usted buscará el **a0** diez bytes en los datos recibidos.**sin procesar** - si está especificado, hace los valores de la cadena ser transferido como bytes hexadecimales reales bastante que una cadena simple. Por ejemplo, **0D0A** no se envía como “0" “D” “0" “A,” sino bastante como **0x0D 0x0A** (o retorno de carro, Line Feedavance de línea).
- **el socket examina el socket- [pretty] [raw]**- este comando examina el búfer de datos del socket (interno) para los datos reales. Si se encuentran los datos, esos datos se visualizan a la salida estándar. Si los caracteres visualizados son NON-imprimibles, serán representados por un punto (.) para la legibilidad.La lista abajo proporciona la información sobre cada parámetro.**socket-** - la descripción del archivo del socket (forma del número entero). Este descriptor se vuelve de conecta.**sin procesar** - si está especificado, hace los valores de la cadena ser visualizado como bytes hexadecimales reales bastante entonces una cadena simple. Bastante entonces imprimiendo el **ABCD al** estándar, imprimiría **41424344** (1 byte equivalente hexadecimal).impresión **guapita de cara de la** salida. Cada línea contendrá el ASCII o el equivalente hexadecimal para cada byte de dato. Habrá 16 bytes impresos en cada línea. Por ejemplo, **0x41 0x42 0x43 0x44 0x10 0x05 ABCD**.
- **[graceful] del socket- de la desconexión del socket**- este comando cierra la conexión al host remoto. Esto es hecha enviando el RST al host remoto de modo que sepa que le hacen que

envía los datos.La lista abajo proporciona la información sobre cada parámetro.**socket-** - la descripción del archivo del socket (forma del número entero). Este descriptor se vuelve de conecta.**agraciado** - desconexión agraciada que envía un FIN bastante que un RST al host remoto para cerrar la conexión.

Administración de socket

Hay una limitación de 32 socketes (funcionando) abiertos en el Switch a cualquier momento. Si un usuario hace un socket conecta y no termina con una desconexión del socket para la descripción del archivo de ese socket (guardada en `$ {SOCKET}`), el socket sigue siendo abierto hasta que una desconexión del socket se haya llamado con ese socket como el parámetro. Los socketes abiertos dentro de los scripts son cerrados cuando el script termina (a menos que el argumento de la sesión se pasa en el socket conecta). Los socketes abiertos dentro de las sesiones son cerrados cuando la sesión termina.

Si un socket sigue siendo abierto, esto es generalmente un caso donde el usuario ha hecho una conexión sin correctamente el cierre. Que el socket seguirá abierto y utilizado hasta que él es cerrado o hasta el script (o el conexión de la terminal) se ha cerrado. Han implementado al **comando show sockets** de enumerar todas las descripciones del archivo usadas del socket que son actualmente funcionando de modo que el usuario conozca cuál está abierto y cuál es cerrado.

Nota: Si un host remoto mide el tiempo hacia fuera de un socket, o el socket es cerrado por un host remoto, la arquitectura de socket es bastante elegante limpiarlo, y le toma la lista de los de socketes usados (encontrados publicando el **comando show sockets**). Esto ocurrirá solamente después de que un usuario intente hacer otro transfiera sobre un socket que ha sido cerrado por el host remoto (si no sienta la marcha lenta que aguarda las necesidades de usuarios).

La salida del **comando show sockets** se muestra abajo.

La pantalla enumera el ID del socket (descripción del archivo), los pares del /port del host conectado, el usuario, y a un temporizador de cuánto tiempo el descriptor ha estado abierto. El campo del `usuario` representaría la línea identificador como se ve en el **comando show lines** hecho salir en el CLI.

Extrayendo los datos usando el socket reciba puede mitigar 10K de información al mismo tiempo. Seguirá habiendo este buffer sin cambiar hasta que el usuario haga otro socket reciba, se limpia y se rellena el momento en el cual el buffer con más datos que vienen apagado el alambre. Cada descriptor de socket (creado del socket conecte) tendrá su propio buffer 10K.

Archivos de secuencia de comandos de copiado

Al actualizar a una nueva versión del código, todos los archivos de secuencia de comandos que usted ha modificado en el directorio del script de la versión anterior necesitarán ser copiados a la nueva versión.

Siga los siguientes pasos antes de actualizar su Switch:

1. FTP al switch CSS. Utilice el puerto de administración o a la dirección de circuito del VLA N.
2. CD al directorio del script.
3. Descargue todos los scripts que usted ha editado a su máquina local.
4. Actualice su Switch.

5. Cargue los scripts de su máquina local al nuevo directorio del script del Switch.

Muestras del script

Los scripts siguientes están disponibles para proporcionarle algunas implementaciones predeterminadas. Estas muestras contienen los scripts escritos por el DNS, la generación de eco, el NetBios, el finger, el tiempo, HttpList, PingList, HttpAuth, Imap4, CookieSet, el POP3, HttpTag, MailHost, y el S TP.

¿Publique el **script de la demostración?** ordene para ver los scripts. A continuación, se incluye un resultado de ejemplo del comando

```
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
```

Verificación

Actualmente, no hay un procedimiento de verificación disponible para esta configuración.

Troubleshooting

Utilice las guías de consulta siguientes para resolver problemas su configuración:

- Ejecute el script de la línea de comando publicando el **comando script play**. Publique este comando cuando está abierto una sesión como un superusuario para aseegurarse la completa sin el error. Si hace el error hacia fuera, la línea que causó el error debe ser publicada.
- Tome una traza de sniffer entre el CSS y el servidor Web para observar qué se vuelve realmente cuando el script se ejecuta contra lo que está esperando el script ver.
- En las versiones del código 5.x, agregaron al **comando use-output**. Este comando se debe utilizar con todos los scripts que utilicen el **comando grep**. Por ejemplo, uso-**salida de los** *argumentos del* **Ap-kal-dns del script del tipo de keepalive**.

Información Relacionada

- Software support del Web Network Services
- Soporte del hardware de los CSS 11000 Series Content Services Switch
- Soporte del hardware de los CSS 11500 Series Content Services Switch
- Descargas del software de Cisco WebNS CSS11500
- Descargas del software de Cisco WebNS CSS11000
- Soporte Técnico - Cisco Systems