

Muestree el script del keepalive para abrir y para cerrar un socket en los puertos especificados del usuario

Contenido

[Introducción](#)

[Antes de comenzar](#)

[Convenciones](#)

[prerrequisitos](#)

[Componentes Utilizados](#)

[Secuencia de comandos de ejemplo](#)

[Información Relacionada](#)

[Introducción](#)

Este documento dirige la implementación de los keepalives en secuencia de comandos. Este script abrirá y cerrará un socket en los puertos especificados del usuario. El cierre será un FIN bastante que un RST. Si uno de los puertos falla, el servicio será declarado abajo. Este método de scripting está el más estrechamente vinculado a las funciones, que está presente en los clientes de marcación manual del Remote Access Server (RAS), los programas para terminal, y las utilidades generales del scripting. Esta característica utiliza el lenguaje de la secuenciación de comandos rico de WebNS.

Complete con un Application Program Interface del socket simple (API) (conecte/desconexión/envían/reciben), un dará al usuario del keepalive en secuencia de comandos 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. Sin las funciones del keepalive en secuencia de comandos, le limitan actualmente al FTP, al HTTP, al ICMP, y al TCP. Con los keepalives en secuencia de comandos, sin embargo, usted puede permanecer encima de los protocolos actuales escribiendo sus propios scripts. Por ejemplo, usted puede desarrollar un script entonado específicamente para conectar con un servidor POP3 sin requerir WebNS construir un tipo de keepalive POP3. Esta característica permite que los clientes creen su propio Keepalives de encargo para adaptarse a sus requisitos específicos. Aunque éste sea un componente del Content Services Switch (CSS), las secuencias de comandos personalizadas no son soportadas por el Centro de Asistencia Técnica de Cisco (TAC de Cisco).

Los keepalives en secuencia de comandos abajo no son soportados oficialmente por TAC, sino se han probado, y están disponibles para el uso en su propia discreción.

[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](#)

La información que contiene este documento se basa en las versiones de software y hardware indicadas a continuación.

- WebNS versión 3.x y posteriores
- Serie CSS 11000

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.

[Secuencia de comandos de ejemplo](#)

El script abajo se puede utilizar para abrir y para cerrar un socket en los puertos especificados del usuario.

```
!--- No echo. !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! !--- Filename:
ap-kal-tcp-ports !--- Parameters: Service Address, TCP Port(s) ! !--- Description: !--- This
script will open and close a socket on the user specified ports. !--- The close will be a FIN
rather than a RST. If one of the ports fails, !--- the service will be declared down ! !---
Failure Upon: !--- 1. Not establishing a connection with the host on one of the specified ports.
! !--- Note: Does not use output. !--- Will handle out of sockets scenario. ! !--- Tested: KGS
12/18/01 ! !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! set OUT-OF-SOCKETS
"785" set NO-CONNECT "774" !--- Make sure the user has a qualified number of arguments. if
${ARGS}[#] "LT" "2" echo "Usage: ap-kal-tcp-ports \'ipAddress tcpPort1 [tcpPort2 tcpPort3...]\'"
exit script 1 endbranch set SERVICE "${ARGS}[1]" !--- echo "SERVICE = ${ARGS}[1]" var-shift ARGS
while ${ARGS}[#] "GT" "0" set TCP-PORT "${ARGS}[1]" var-shift ARGS function SOCKET_CONNECT call
!--- If out of sockets, exit, and look for sockets on the next KAL interval. if RETURN "=="
"${OUT-OF-SOCKETS}" set EXIT_MSG "Exceeded number of available sockets, skipping until next
interval." exit script 0 endbranch !--- Valid connection, look to see if it was good. if RETURN
"==" "${NO-CONNECT}" set EXIT_MSG "Connect: Failed to connect to ${SERVICE}:${TCP-PORT}" exit
script 1 endbranch endbranch no set EXIT_MSG exit script 0 function SOCKET_CONNECT begin set
CONTINUE_ON_ERROR "1" socket connect host ${SERVICE} port ${TCP-PORT} tcp 2000 set SOCKET-STAT
"${STATUS}" set CONTINUE_ON_ERROR "0" socket disconnect ${SOCKET} graceful function
SOCKET_CONNECT return "${SOCKET-STAT}" function SOCKET_CONNECT end
```

[Información Relacionada](#)

- [Soporte de productos de los CSS 11000 Series Content Services Switch](#)
- [Soporte de productos de los CSS 11500 Series Content Services Switch](#)
- [Software de la descarga CSS11000 \(clientes registrados solamente\)](#)
- [Software de la descarga CSS11500 \(clientes registrados solamente\)](#)
- [Soporte Técnico y Documentación - Cisco Systems](#)