

Sample Keepalive Script to Open and Close a Socket on the User Specified Ports

Document ID: 24106

Introduction
Before You Begin
Conventions
Prerequisites
Components Used
Sample Script
Related Information

Introduction

This document addresses implementation of scripted keepalives. 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. This method of scripting is most closely related to functionality, which is present in Remote Access Server (RAS) dialup clients, terminal programs, and general scripting utilities. This feature utilizes WebNS's rich scripting language.

Complete with a simple socket Application Program Interface (API) (connect/disconnect/send/receive), a scripted keepalive will give the user the ability to tailor their own protocol, or write their own sequence of steps to provide a reliable ALIVE or DOWN state of a service. Without the scripted keepalive functionality, you are currently limited to FTP, HTTP, ICMP, and TCP. With scripted keepalives, however, you can remain on top of the current protocols by writing your own scripts. For example, you can develop a script specifically toned to connect to a POP3 server without requiring WebNS to build a keepalive type POP3. This feature allows customers to create their own custom keepalives to suit their specific requirements. Although this is a component of the Content Services Switch (CSS), custom scripts are not supported by the Cisco Technical Assistance Center (Cisco TAC).

The scripted keepalives below are not officially supported by TAC, but have been tested, and are available for use at your own discretion.

Before You Begin

Conventions

For more information on document conventions, see the Cisco Technical Tips Conventions.

Prerequisites

There are no specific prerequisites for this document.

Components Used

The information in this document is based on the software and hardware versions below.

- WebNS versions 3.x and higher
- CSS 11000 Series

The information presented in this document was created from devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If you are working in a live network, ensure that you understand the potential impact of any command before using it.

Sample Script

The script below can be used to open and close a socket on the user specified ports.

```
!--- 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
```

Related Information

- [CSS 11000 Series Content Services Switches Product Support](#)
 - [CSS 11500 Series Content Services Switches Product Support](#)
 - [Download CSS 11000 Software](#)
 - [Download CSS 11500 Software](#)
 - [Technical Support – Cisco Systems](#)
-

All contents are Copyright © 2006–2007 Cisco Systems, Inc. All rights reserved. Important Notices and Privacy Statement.

Updated: Jan 31, 2006

Document ID: 24106
