

# Table of Contents

<b><u>Sample Keepalive Script to Check the Web Page for the Webstring</u></b> .....	1
<u>Document ID: 47383</u> .....	1
<u>Introduction</u> .....	1
<u>Prerequisites</u> .....	1
<u>Requirements</u> .....	1
<u>Components Used</u> .....	1
<u>Sample Script</u> .....	2
<u>Related Information</u> .....	3

# Sample Keepalive Script to Check the Web Page for the Webstring

Document ID: 47383

---

<b>Introduction</b>
<b>Prerequisites</b>
Requirements
Components Used
<b>Sample Script</b>
<b>Related Information</b>

---

## Introduction

This script checks the Web page for the Web string. If the string is missing, mark the service as down. This script is used with any sort of page, especially dynamic ones that are generated via scripts, ColdFusion, and so on. This document also addresses implementation of scripted keepalives. 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.

## Prerequisites

### Requirements

Familiarity with WebNS rich scripting language.

### Components Used

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

- WebNS versions 3.x and higher
- CSS 11x00 Series

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

# Sample Script

The script below can be used to check the Web page for the webstring.

```
!--- No echo.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!--- Filename: ap-kal-httpstring
!--- Parameters: WebsiteIP WebPage WebString [Port]
!--- Requirements: WebNS4.x or higher
!
!--- Uses:
!--- Checks the Web page for the Web string. If the string is missing,
!--- mark the service as down. Used with any sort of page, especially dynamic
!--- ones that are generated via scripts, ColdFusion, and so on.
!
!--- Logic:
!--- The script connects to a Web server on port 80 by default.
!--- It performs a GET on the specified page.
!--- If the Web string is returned, the service stays up.
!--- If anything fails, the service is marked down.
!
!--- Notes:
!--- The Web string is case-sensitive.
!--- Only the first 10Kb of the response is inspected.
!
!
!--- Tested: 04/12/01-KGS
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

no set CONTINUE_ON_ERROR
no set EXIT_MSG

!--- Make sure the user has the proper number of arguments.

if ${ARGS}[#] "LT" "3"
    echo "Usage: ap-kal-httpcheck \'WebsiteIP WebPage WebString [Port]\'"
    exit script 1
endbranch

!--- Set variables corresponding to the args.

set WebSite "${ARGS}[1]"
set WebPage "${ARGS}[2]"
set WebString "${ARGS}[3]"
set WebPort "80"
if ${ARGS}[#] "GT" "3"
    set WebPort "${ARGS}[4]"
endbranch

echo "Requesting ${WebPage} from ${WebSite} on port ${WebPort}."

!--- Connect to the remote server.

set EXIT_MSG "Connect: Failed. Could not connect to ${WebSite} on port ${WebPort}"
set CONTINUE_ON_ERROR "1"
socket connect host ${WebSite} port ${WebPort} tcp
if ${STATUS} "NEQ" "0"
```

```
    exit script 1
endbranch
no set CONTINUE_ON_ERROR

!--- Request the desired Web page.

set EXIT_MSG "Send: Failed. Could not send to ${WebSite}:${WebPort}"
socket send ${SOCKET} "GET ${WebPage} HTTP/1.0\n\nHost: ${WebSite}:${WebPort}\n"

!--- Look for the Web string.

set EXIT_MSG "Waitfor: Failed. Did not find [${WebString}]"
set CONTINUE_ON_ERROR "1"
socket waitfor ${SOCKET} "${WebString}" case-sensitive
if ${STATUS} "NEQ" "0"
    exit script 1
endbranch
no set CONTINUE_ON_ERROR

!--- Disconnect from the server.

no set EXIT_MSG
socket disconnect ${SOCKET} graceful
exit script 0
```

---

## Related Information

- [CSS 11000 Series Content Services Switches Hardware Support](#)
- [CSS 11500 Series Content Services Switches Hardware Support](#)
- [Software Download for CSS11500](#)

---

All contents are Copyright © 1992–2006 Cisco Systems, Inc. All rights reserved. Important Notices and Privacy Statement.

---

Updated: Jan 20, 2006

Document ID: 47383

---