Exemple de configuration de Keepalives à base de script WebNS 4.0

Contenu

Introduction

Ce document décrit l'implémentation initiale des keepalives à base de script. Cette méthode de script le plus étroitement est liée à la fonctionnalité qui est présente dans des clients distants du Reliability, Availability, and Serviceability (RAS), des programmes de terminal, et des utilitaires généraux de script. Cette caractéristique utilise le langage de script riche du CSS.

Configurer ceci avec un socket simple API (connectez/débranchement/send/receive) donnera à l'utilisateur la capacité de travailler leur propre protocole, ou écrit leur propre ordre des étapes pour fournir un `ACTIF` ou un état d'indisponibilité fiable d'un service. Vous êtes actuellement limité au FTP, au HTTP, à l'ICMP, et au TCP. Avec cette nouvelle configuration, vous pouvez rester sur les protocoles en cours à côté d'écrire vos propres scripts. Par exemple, un utilisateur peut développer un script spécifiquement modifié la tonalité pour se connecter à un serveur POP3 sans exiger de Cisco d'établir un type de keepalive POP3 pour adapter à leurs besoins. Cette caractéristique permettra à des clients pour créer leur propre Keepalives fait sur commande pour satisfaire à leurs exigences spécifiques.

Avant de commencer

Conventions

Pour plus d'informations sur les conventions des documents, référez-vous aux Conventions utilisées pour les conseils techniques de Cisco.

Conditions préalables

Aucune condition préalable spécifique n'est requise pour ce document.

Composants utilisés

Ce document s'applique à CSS 11000/CSS 11500 ou à CSS 11800 avec le logiciel de WebNS.

Les informations présentées dans ce document ont été créées à partir de périphériques dans un environnement de laboratoire spécifique. Tous les périphériques utilisés dans ce document ont démarré avec une configuration effacée (par défaut). Si vous travaillez dans un réseau opérationnel, assurez-vous de bien comprendre l'impact potentiel de toute commande avant de l'utiliser.

Théorie générale

Une fois que vous avez développé un script de l'interface de ligne de commande (CLI) off-line utilisant un éditeur de texte tel que Notepad, vous pouvez télécharger que script au répertoire de /script du CSS et configurez l'option de keepalive de script à un service. On te permet pour créer une keepalive de script sans avoir un script actuel sur le système. Dans le cas d'une exécution de keepalive de script sans script sur le système, un état d'indisponibilité de constante demeurera au service. Ceci permet à un administrateur pour écrire une configuration et pour implémenter la configuration avant d'écrire (ou les télécharger) tous les scripts.

Un script doit résider dans le/<current exécutant le répertoire version>/script/pour que la keepalive de script trouve le script. Des noms de chemin ne sont pas reçus, seulement des noms de script en configurant la keepalive de script. Si le script est présent ailleurs sur le système, la keepalive de script supposera qu'elle n'existe pas. Ce les moyens, cependant, quand le logiciel est mis à jour sur le système, les vieux scripts résident dans de la vieille le répertoire du script version. Pour copier les scripts à partir du vieux répertoire, voyez la section de fichiers script copiante de ce document.

Vous pouvez passer 128 caractères dans un argument cité. Assumant un niveau de sept caractères par argument, vous pouvez obtenir environ 18 arguments en un script. La ligne de commande recevra seulement 90 caractères cependant.

Le Keepalives de script d'il est recommandé que soit configuré avec une fréquence inférieure qu'une keepalive standard parce que beaucoup de scripts auront un processus multipas tel que se connecter, envoyant une demande, et attendre un type spécifique de réponse. En raison de ceci, il est recommandé pour avoir une fréquence de dix secondes ou plus élevé de sorte que la keepalive ait le temps pour terminer complètement. Autrement, les transitions d'état peuvent se produire plus souvent.

Un script peut renvoyer code d'état de zéro ou de différent de zéro. Sur un retour de différent de zéro, le CSS signalera l'état de service en tant que VERS LE BAS. Sur un retour de zéro, le CSS signalera l'état de service comme ACTIF. Référez-vous au script suivant pour un exemple :

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

```
did connect. if $SOCKET "==" "-1" exit script 1 endbranch
```
Le CSS renverra toujours un état de `VERS LE BAS` si la variable `$ {SOCKET}` est placée au négatif un. Il est très important de vérifier la logique de vos scripts pour s'assurer que la valeur correcte obtient retourné.

Configurez

Cette section vous fournit des informations pour configurer les fonctionnalités décrites dans ce document.

Pour un grand nombre de services qui exigent l'utilisation du Keepalives de script, on le recommande fortement qu'ils emploient un plus petit sous-ensemble de Keepalives global pour manipuler le travail pour eux.

Pour configurer une keepalive de script, suivez les mêmes instructions que pour tous autres types de keepalive. La syntaxe de commande est affichée ci-dessous.

```
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"
```
La première ligne configurera la keepalive en cours de service pour être de **script de** type et pour avoir l'AP-kal-**SMTP de** nom de script avec l'argument **einstein**. La deuxième ligne les expositions **ap-kal-pop3**, qui est semblable au premier script, mais passera trois arguments : **einstein**, **vxworks**, et **mipspci**. Il est également possible d'enfoncer la touche d'**onglet** (ou **?**) pour voir une liste complète de tous les scripts disponibles dans le/<current exécutant le répertoire version>/script. Ceci vous encourage à adopter un script nommant la convention de sorte que quand vous frappez l'**onglet** ou **?** , vous pouvez clairement voir les scripts de keepalive disponibles pour utiliser. Une convention nommante d'AP-kal-**type** est utilisée pour des scripts. Par exemple, un script de SMTP serait nommé AP-kal-**SMTP**. Sur option, vous pouvez saisir un non trouvé dans cette liste, supposant que vous souhaitez la télécharger à une date ultérieure. Des scripts peuvent être manipulés par les **archives**, l'**espace libre**, et les Commandes **COPY**, et peuvent être téléchargement/téléchargement à partir du répertoire de /script sur le commutateur.

Le nom de script peut être jusqu'à 32 caractères longs. Les arguments doivent être passés dans une chaîne protégée, et peuvent être jusqu'à 128 caractères longs. Pour désactiver la keepalive de script, vous modifieriez le service en émettant la commande suivante :

```
CS100(config-service[cs100])# keepalive type none
```
Visionnement d'une keepalive de script dans un service

Quand une keepalive de script est configurée sous un service, le nom de script peut être vu dans la sortie de commande de **show service**. Le script apparaît sous le type de keepalive, et tous les arguments potentiels peuvent être trouvés directement sous cette ligne dans les `arguments de script :` champ. S'il n'y a aucun argument de script, ce champ n'est pas affiché. Un exemple du service **cs100** avec un script a appelé **ap-kal-pop3** avec les **vxworks d'**arguments et le **mipspci** est affiché ci-dessous.

```
Name: cs100                 Index: 1
 Type: Local                State: Suspended
 Rule ( 13.1.1.2  TCP  23 )
 Redirect Domain:
 Keepalive: (SCRIPT ap-kal-pop3  255   3   5 )
 Script Arguments: "einstein vxworks mipspci"
 Mtu:                1500         State Transitions:  0
 Connections:        0            Max Connections:    0
 Total Connections: 0             Total Reused Conns: 0
 Weight:             1            Load:               255
```

Avec cette alimentation, un script comme **ap-kal-pop3** pourrait prendre trois paramètres, qui apparieraient l'adresse Internet, le nom d'utilisateur, et le mot de passe pour un serveur POP3. Ouvrir une session simplement au serveur POP est assez pour que le script détermine que ce serveur est dans l'état ACTIF.

La sortie de **commande show running-config** est affichée ci-dessous.

```
service cs100
  protocol tcp
  port 23
  ip address 13.1.1.2
  keepalive frequency 255
  keepalive type script ap-kal-pop3 "einstein vxworks mipspci"
```

La sortie de commande ci-dessus affiche exactement quel script (et arguments) ont été configuré à ce service. Si aucun argument n'est présent, le texte cité après le nom de script ne sera pas présent.

En fonctionnant avec le script se connectant, la configuration est comme suit :

```
service S1
  ip address 192.168.2.3
  keepalive type script ap-kal-pop3 "192.168.2.3 lab labtest1"
  keepalive frequency 10
  keepalive logging file testlog.log
  active
```

Cet exemple affiche le service se connectant sa keepalive de script sortie à **testlog.log**.

Primitifs de socket

Il y a quelques commandes de **douille** qui peuvent être utilisées dans une keepalive de script pour aider à établir un protocole structuré. Les primitifs de socket tiendront compte de l'ASCII ou de l'hexadécimal envoient et reçoivent la fonctionnalité. Chaque commande qui a un mot clé CRU facultatif changera les données d'une ASCII standard à une conversion hexadécimale. Par exemple, l'**abcd** dans l'ASCII serait représenté par **61626364**, qui dénote 0x61 0x62 0x63 0x64.

Référez-vous à la liste suivante de pour en savoir plus de commande de **socket** :

- **le socket connectent le TCP de** *port de* **port d'***adresse Internet d'***hôte | UDP [entier] [session]**- cette commande exécute une connexion de TCP ou d'UDP. Exécuter une connexion TCP implique une prise de contact (SYN-SYNACK…) à une particularité IP/port. Exécuter une connexion d'UDP est simplement une réservation de l'hôte/du port. La valeur de socket est reçue dans une variable `$ {SOCKET}` dans le script.**Remarque:** Seulement 32 sockets peuvent être ouverts en même temps à travers tous les scripts sur le commutateur.La liste ci-dessous fournit des informations sur chaque paramètre.**hôte** - un mot clé qui devrait être suivi par l'adresse Internet ou l'adresse IP du système distant.**port** - mot clé qui devrait être suivi par le port sur avec lequel pour négocier une connexion.**TCP** - une connexion utilisant le TCP.**UDP** - une connexion utilisant l'UDP.**entier** - une valeur du dépassement de durée pour l'établissement de réseau en quelques secondes. Si le délai expire avant que le rapport ait été avec succès établi, la tentative échoue. Ceci s'applique seulement à une connexion TCP, car l'UDP est sans connexion. La valeur par défaut est un délai d'attente de cinq secondes.**session** - un mot clé qui indique le socket demeurer ouvert jusqu'à ce que la session soit de finition. Cela signifie qu'aucun script qui les sockets ouverts en session et ne les ferment pas tout seul restera ouvert jusqu'aux journaux de l'utilisateur.
- **le socket envoient la chaîne de socket# [crue | base64]**- cette commande écrit des données par une connexion TCP précédemment connectée.La liste ci-dessous fournit des informations sur chaque paramètre.**socket#** - le descripteur de fichier de socket (forme d'entier). Ce descripteur est retourné de se connectent.**chaîne** - chaîne de texte jusqu'à 128 caractères de longueur.**cru** - si spécifié, cause les valeurs de chaîne d'être transféré en tant qu'octets hexadécimaux réels plutôt qu'une chaîne simple. Par exemple, **0D0A** n'est pas envoyé en tant que « 0" « D » « 0" « A, » mais plutôt comme **0x0D 0x0A** (ou retour chariot, retour à la ligne).**base64** - ceci base64 encodera la chaîne avant de l'envoyer par la connexion. Utile pour l'authentification de base de HTTP en se connectant à un site Web protégé par mot de passe.
- **le socket reçoivent le socket# [l'entier] [cru]**- cette commande remplit la mémoire tampon 10K interne de données étant livré dedans à partir du serveur distant. Cette commande verrouille alors la mémoire tampon de sorte qu'aucune nouvelle donnée ne soit mise dans cette mémoire tampon interne. Vous pouvez poursuivre à l'utilisation examinez pour vider toutes les données résidant dans cette mémoire tampon 10K interne à la sortie standard. **Remarque:** Toutes les données précédentes dans la mémoire tampon 10K interne sont rincées avant que de nouvelles données soient mises dedans.La liste ci-dessous fournit des informations sur chaque paramètre.**socket#** - le descripteur de fichier de socket (forme d'entier). Ce descripteur est retourné de se connectent.**entier** - une valeur entière représentant le nombre de millisecondes pour attendre avant de verrouiller la mémoire tampon 10K interne et le renvoi à l'utilisateur. Si aucune heure d'entier n'est spécifiée, recevez attendra 100ms avant le renvoi à l'utilisateur.**cru** - si spécifié, cause les valeurs de chaîne d'être transféré en tant qu'octets hexadécimaux réels plutôt qu'une chaîne simple. Par exemple, **0D0A** n'est pas envoyé en tant que « 0" « D » « 0" « A, » mais plutôt comme **0x0D 0x0A** (ou retour chariot, retour à la ligne).
- **décalage] de chaîne de socket# de waitfor de socket [[entier] [case-sensitive] [cru]**- cette commande est semblable au socket reçoivent, sauf qu'elle retourne immédiatement en trouvant l'argument spécifié de chaîne. Une fois la chaîne spécifiée est trouvée, il renverra `$ {ÉTAT} de` zéro. Autrement, il renvoie 1. Les données récupérées peuvent plus plus loin être visualisées en émettant le **socket examinent la** commande.La liste ci-dessous fournit des informations sur chaque paramètre.**socket#** - le descripteur de fichier de socket (forme d'entier). Ce descripteur est retourné de se connectent.**chaîne** - la chaîne spécifique qui doit s'avérer pour avoir comme conséquence un `$ {ÉTAT} de` zéro, qui indique trouvé. Si la chaîne

est trouvée, elle retourne immédiatement et n'attend pas la longueur entière de délai d'attente d'entier.**entier** - une valeur entière représentant le nombre de millisecondes pour attendre avant de verrouiller la mémoire tampon 10K interne et le renvoi à l'utilisateur. Si aucune heure d'entier n'est spécifiée, recevez attendra 100ms avant le renvoi à l'utilisateur.**case-sensitive** - si spécifié, indique que la comparaison de chaîne devrait distinguer les majuscules et minuscules. Par exemple, **utilisateur :** ne soyez pas équivalent à l'**utilisateur :**.**décalage** - combien d'octets dans les données reçues la chaîne devrait être trouvé. Par exemple, si vous recherchez **a0** et donnez un décalage de dix, vous rechercherez **a0** dix octets dans les données reçues.**cru** - si spécifié, cause les valeurs de chaîne d'être transféré en tant qu'octets hexadécimaux réels plutôt qu'une chaîne simple. Par exemple, **0D0A** n'est pas envoyé en tant que « 0" « D » « 0" « A, » mais plutôt comme **0x0D 0x0A** (ou retour chariot, retour à la ligne).

- **le socket examinent le socket# [joli] [cru]**- cette commande examine le tampon de données du socket (interne) pour assurer les données réelles. Si des données sont trouvées, ces données sont affichées à la sortie standard. Si les caractères affichés sont non-imprimables, ils seront représentés par un point (.) pour la lisibilité.La liste ci-dessous fournit des informations sur chaque paramètre.**socket#** - le descripteur de fichier de socket (forme d'entier). Ce descripteur est retourné de se connectent.**cru** - si spécifié, cause les valeurs de chaîne d'être affiché en tant qu'octets hexadécimaux réels plutôt puis une chaîne simple. Plutôt alors imprimant **ABCD à la** norme, il imprimerait **41424344** (1 équivalent d'hexadécimal d'octet).impression **gentillette de** sortie. Chaque ligne contiendra l'ASCII ou l'hexadécimal équivalent pour chaque octet de données. Il y aura 16 octets imprimés sur chaque ligne. Par exemple, **0x41 0x42 0x43 0x44 0x10 0x05 ABCD**.
- **socket# de débranchement de socket [gracieux]**- cette commande ferme la connexion au serveur distant. Ceci est fait en envoyant RST au serveur distant de sorte qu'il sache que vous êtes fait envoyant des données.La liste ci-dessous fournit des informations sur chaque paramètre.**socket#** - le descripteur de fichier de socket (forme d'entier). Ce descripteur est retourné de se connectent.**gracieux** - débranchement gracieux qui envoie une FIN plutôt qu'un RST au serveur distant pour clôturer le connecter.

Gestion de socket

Il y a une limite de 32 sockets (en service) ouverts sur le commutateur en même temps. Si un utilisateur fait un socket se connectent et ne finit pas avec un débranchement de socket pour le descripteur de fichier de ce socket (enregistré dans `$ {SOCKET}`), le socket reste ouvert jusqu'à ce qu'un débranchement de socket se soit appelé avec ce socket comme paramètre. Les sockets ouverts dans des scripts sont fermés quand le script finit (à moins que l'argument de session est passé dans le socket se connectent). Les sockets ouverts dans des sessions sont fermés quand la session finit.

Si un socket demeure ouvert, c'est habituellement un cas où l'utilisateur a établi un rapport sans se fermer correctement. Que le socket demeurera ouvert et utilisé jusqu'à ce qu'il est fermé ou jusqu'au script (ou à la connexion terminale) a été fermé. La commande de **show sockets** a été mise en application de répertorier tous les descripteurs de fichier utilisés de socket qui sont actuellement en service de sorte que l'utilisateur sache ce qui est ouvert et ce qui est fermé.

Remarque: Si un serveur distant chronomètre un socket, ou le socket est fermé par un serveur distant, l'architecture de socket est assez intelligente pour le nettoyer, et le prend hors de la liste de sockets utilisés (trouvés en émettant la commande de **show sockets**). Ceci se produira seulement après que des tentatives d'un utilisateur de faire un autre transfert sur un socket qui a

été fermé par le serveur distant (autrement il repose l'inactif attendant les besoins de l'utilisateur).

La sortie de commande de **show sockets** est affichée ci-dessous.

```
Socket ID      Remote Host:Port      User         Time
---------    --------------------    ----      ------------
   38           192.168.2.3:80       console    0 days 00:00:23
   37           192.168.2.4:80       console    0 days 00:00:27
   36           192.168.2.4:80       console    0 days 00:29:23
   40           192.168.2.3:07       console    0 days 00:00:19
   45           192.168.2.4:80       vty1    0 days 00:00:02
   44           192.168.2.4:80       vty1    0 days 00:00:03
```

L'écran répertorie l'ID de socket (descripteur de fichier), l'hôte connecté/paires de port, l'utilisateur, et un temporisateur de combien de temps le descripteur a été ouvert. Le champ d'`utilisateur` représenterait la ligne identifiant comme vu dans la sortie de commande de **shows line** sur le CLI.

En récupérant des données utilisant le socket recevez peut mettre en mémoire tampon la valeur 10K des données en même temps. Cette mémoire tampon demeurera sans changement jusqu'à ce que l'utilisateur fasse une autre douille reçoivent, laquelle au point la mémoire tampon est nettoyée et remplie avec plus de données étant livré hors fonction le fil. Chaque descripteur de douille (créé du socket connectez) aura sa propre mémoire tampon 10K.

Copier des fichiers script

Quand améliorant à une nouvelle version de code, tous les fichiers script que vous avez modifiés dans le répertoire du script de la version préalable devront être copiés sur la nouvelle version.

Suivez ces étapes avant d'améliorer votre commutateur :

1. FTP au commutateur CSS. Utilisez l'adresse de port de gestion ou de circuit VLAN.
2. CD au répertoire de script.
3. Téléchargez tous les scripts que vous avez édités à votre ordinateur local.
4. Améliorez votre commutateur.
5. Téléchargez les scripts de votre ordinateur local au nouveau répertoire du script du commutateur.

Échantillons de script

Les scripts suivants sont disponibles afin de te fournir quelques réalisations par défaut. Ces échantillons contiennent des scripts écrits pour des DN, l'écho, le Netbios, le Finger, le temps, le HttpList, le PingList, le HttpAuth, IMAP4, le CookieSet, le POP3, le HttpTag, le MailHost, et le SMTP.

Émettez le **script d'exposition ?** commande de voir les scripts. L'exemple de sortie de commande est affiché ci-dessous.

```
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
```

Vérifiez

Aucune procédure de vérification n'est disponible pour cette configuration.

Dépannez

Employez les instructions suivantes pour dépanner votre configuration :

- Exécutez le script de la ligne de commande en émettant la commande de **jeu de script**. Émettez cette commande une fois ouvert une session comme un super utilisateur pour s'assurer le se termine sans erreur. S'il fait l'erreur, la ligne qui a entraîné l'erreur devrait être émise.
- Prenez un tracé de renifleur entre le CSS et le web server pour observer ce qui est retourné réellement quand le script est exécuté contre ce que le script compte voir.
- Dans les versions 5.x du code, la commande d'utilisation-**sortie** a été ajoutée. Cette commande doit être utilisée avec tous les scripts qui utilisent la commande de **grep**. Par exemple, utilisation-**sortie d'***arguments d'AP-kal*-**dn de script de type de keepalive**.

Informations connexes

- Support logiciel de services réseau de Web
- Support matériel de Commutateurs de services satisfaits de gamme 11000 CSS
- Support matériel pour les commutateurs de services de contenu de la gamme CSS 11500

- Téléchargements logiciels de Cisco WebNS CSS11500
- Téléchargements logiciels de Cisco WebNS CSS11000
- Support technique - Cisco Systems