WebNS 4.0 スクリプト キープアライブの設定例

目次

概要

このドキュメントでは、スクリプト キープアライブの初期実装について説明します。 このスクリプティング方法は、Reliability, Availability, and Serviceability（RAS）のダイヤルアップ クライアント、ターミナル プログラム、および一般的なスクリプト ユーティリティに存在する機能と密接に関連しています。 この機能は、CSS の豊富なスクリプト言語を使用します。

シンプルなソケット API（接続/切断/送信/受信）でこの機能を設定すると、ユーザは自身のプロトコルをカスタマイズしたり、独自のステップ シーケンスを記述してサービスの信頼できる ALIVE または DOWN 状態を提供したりできます。 現在は、FTP、HTTP、ICMP、および TCP に限定されています。 この新機能を使用すると、独自のスクリプトを記述することで現在のプロトコルに加えて優れた機能を実現できます。 たとえば、ユーザは、Cisco を必要とせずに POP3 サーバに接続するように調整されたスクリプトを開発することで、自分のニーズに合うようにキープアライブ型 POP3 を構築できます。 この機能により、顧客は特定の要件に合わせて独自のカスタム キープアライブを作成できます。

はじめに

表記法

ドキュメント表記の詳細は、『シスコ テクニカル ティップスの表記法』を参照してください。

前提条件

このドキュメントに関する固有の要件はありません。

使用するコンポーネント

このドキュメントは、WebNS のソフトウェアを搭載した CSS11000/CSS 11500 または CSS11800 に適用されます。

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されたものです。 このドキュメントで使用するすべてのデバイスは、クリアな（デフォルト）設定で作業を開始しています。 対象のネットワークが実稼働中である場合には、どのような作業についても、その潜在的な影響について確実に理解しておく必要があります。

背景理論

メモ帳などのテキスト エディタを使用してコマンドライン インターフェイス（CLI）スクリプトをオフラインで開発したら、そのスクリプトを CSS の /script ディレクトリにアップロードし、サービスに対してスクリプトのキープアライブ オプションを設定できます。 システムにスクリプトが存在しない場合でも、スクリプト キープアライブを作成できます。 システムにスクリプトがない状態でスクリプト キープアライブを実行する場合、DOWN 状態がサービスに常に残ります。 これにより管理者は、すべてのスクリプトの記述を完成させる（またはアップロードする）前に、コンフィギュレーションを記述して実装できます。

スクリプト キープアライブに見えるようにするには、このスクリプトが /<現在実行中のバージョン>/script/ ディレクトリに存在する必要があります。 スクリプト キープアライブを設定する際、パス名は受け付けられません。スクリプト名だけが受け付けられます。 スクリプトがシステムのどこか他の場所に存在する場合、スクリプト キープアライブはそのスクリプトが存在しないものとみなします。 つまり、システムでソフトウェアをアップグレードすると、古いスクリプトは、古いバージョンのスクリプト ディレクトリに残存します。 古いディレクトリからスクリプトをコピーする方法については、このドキュメントの「スクリプト ファイルのコピー」の項を参照してください。

引用符で囲まれた引数として 128 文字を渡すことができます。 引数あたり標準 7 文字を想定すると、1 つのスクリプトに約 18 個の引数を指定できます。 コマンドラインは、90 文字のみ受け付けます。

多くのスクリプトが、接続、リクエストの送信、特定のタイプの応答の待機など複数ステップのプロセスを含んでいるので、スクリプト キープアライブは標準のキープアライブよりも低い頻度で設定することを推奨します。 このような理由から、キープアライブが完全に終了する時間を確保するために、頻度を 10 秒以上に設定することを推奨します。 そうしないと、状態の遷移がより頻繁に発生する可能性があります。

スクリプトは、ゼロまたは非ゼロのステータス コードを返す可能性があります。 ゼロ以外が返された場合、CSS はサービス状態を DOWN のフラグに設定します。 ゼロが返された場合、CSS はサービス状態を ALIVE のフラグに設定します。 次のスクリプトを例として参照してください。

```
!--- Connect to the remote host. socket connect host einstein port 25 tcp !--- Validate that you did connect. if $SOCKET "==" "-1" exit script 1 endbranch
```

${SOCKET} 変数が -1 に設定されている場合、CSS は常に DOWN の状態を返します。 スクリプトのロジックをチェックし、必ず正しい値が返されるようにすることが非常に重要です。

設定

この項では、このドキュメントで説明する機能の設定に必要な情報を提供します。

大量のサービスがキープアライブ スクリプトを使用する必要がある場合は、比較的少ないグローバル キープアライブのサブセットを使用して仕事を処理することを推奨します。

スクリプト キープアライブを設定するには、他のすべてのタイプのキープアライブと同じガイドラインに従います。 次に、コマンド構文を示します。

```
CS100(config-service[serv1]# keepalive type script ap-kal-smtp "einstein"
```

または

```
CS100(config-service[serv1])# keepalive type script ap-kal-pop3 "einstein vxworks mipspci"
```

最初の行は、現在のサービス キープアライブのタイプを **script** に設定し、スクリプト名が **ap-kal-smtp**、引数が **einstein** になるように指定します。 2 行目は、最初のスクリプトと似ていますが、次の 3 つの引数を渡す **ap-kal-pop3** を示します。 **einstein**、**vxworks**、および **mipspci**。 また、**Tab**（または**?**）キーを押して、/<現在実行中のバージョン>/script ディレクトリで利用できるすべてのスクリプトの全体リストを表示することもできます。 ここでは、**Tab** または **?** キーを押した際に利用可能なキープアライブ スクリプトを明確に確認できるようなスクリプトの命名規則を採用してください。 スクリプトには、**ap-kal-type** の命名規則が使用されます。 たとえば、SMTP スクリプトには、**ap-kal-smtp** という名前を付けます。 オプションで、後日アップロードするつもりの、このリストにはないスクリプトを入力することができます。 スクリプトは **archive**、**clear**、**および copy** コマンドで操作でき、スイッチ上の /script ディレクトリに対してアップロードまたはダウンロードできます。

スクリプト名には最大 32 文字の長さを使用できます。 引数は引用符で囲んだ文字列として渡す必要があり、その長さは、最大 128 文字まで可能です。 スクリプト キープアライブを無効にするには、次のコマンドを発行し、サービスを再設定します。

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

サービス内のスクリプト キープアライブの表示

サービスにスクリプト キープアライブが設定されている場合、スクリプト名は、**show service** コマンドの出力結果で確認できます。 スクリプトは、キープアライブ タイプの下に表示され、引数があればその行の直下の Script Arguments: フィールドの URL のみが置換されます。 スクリプトの引数が存在しない場合、このフィールドは表示されません。 次の例は、**vxworks** および **mipspci** という引数が指定された **ap-kal-pop3** という名前のスクリプトを持つサービス **cs100** です。

この機能により、**ap-kal-pop3** のようなスクリプトは、POP3 サーバのホスト名、ユーザ名、およびパスワードに一致する 3 つのパラメータを取得できます。 単に POP サーバにログインするだけで、このサーバが ALIVE 状態にあることをスクリプトが判定できます。

show running-config コマンドの出力結果を次に示します。

上記のコマンド出力から、本サービスにどのようなスクリプト（および引数）が設定されているかが正確に判断できます。 引数が存在しない場合、スクリプト名の隣には、引用符で囲まれたテキストが存在しません。

スクリプト ロギングを使用する場合の設定は次のとおりです。

この例は、**testlog.log** にスクリプト キープアライブのログ出力を記録しているサービスを示しています。

ソケット プリミティブ

構造化されたプロトコルの構築を支援するために、スクリプト キープアライブで使用できるいくつかの **socket** コマンドがあります。 ASCII または 16 進数の送信および受信機能にはソケット プリミティブが可能です。 オプションの RAW キーワードが指定されている各コマンドは、標準の ASCII のデータを 16 進数変換に変更します。 たとえば、ASCII で **abcd** は、0x61 0x62 0x63 0x64 を意味する **61626364** で表されます。

詳細については、次の **socket** コマンドのリストを参照してください。

- **socket connect host hostname port port tcp | udp [integer] [session]**：このコマンドは、TCP または UDP 接続のいずれかを実行します。 TCP 接続を行うと、特定の IP/ポートとのハンドシェイク（SYN-SYN ACK...）が実行されます。 UDP 接続を行うと、単にホスト/ポートが予約されるだけです。 ソケット値は、スクリプト内の ${SOCKET} 変数で受け取られます。**注:** スイッチ上のすべてのスクリプト全体で一度に開くことができるソケットは 32 個のみです。次のリストは、各パラメータについての情報を提供します。**host**：リモート システムのホスト名または IP アドレスを指すキーワード。**port**： 接続をネゴシエートするポートを指すキーワード。**tcp**：TCP を使用する接続。**udp**：UDP を使用する接続。**integer**：ネットワーク確立のためのタイムアウト値（秒単位）。 接続が正常に行われている前に制限時間に達すると、試行が失敗します。 UDP はコネクションレス型なので、これは TCP 接続にのみ適用されます。 デフォルト値は 5 秒のタイムアウトです。**session** ：セッションが終了するまでソケットを開いたままにするように指定するキーワード。 つまり、セッションでソケットを開き、それ自身でソケットを閉じないスクリプトは、ユーザがログアウトするまで開いたままになります。
- **socket send socket# string [raw | base64]**：このコマンドは、以前に接続された TCP 接続経由でデータを書き込みます。次のリストは、各パラメータについての情報を提供します。**socket#** ：ソケット ファイル記述子（整数形式）。 この記述子は接続から返されます。**string** ：最大 128 文字の長さの、引用符で囲まれたテキスト。**raw** ：指定すると、文字列値が単純な文字列としてではなく実際の 16 進数のバイトとして転送されます。 たとえば、**0D0A** は ‘0’ ‘D’ ‘0’ ‘A' としてではなく、**0x0D 0x0A**（またはキャリッジ リターン、ライン フィード）として送信されます。**base64**：これは、文字列を接続経由で送信する前に base64 エンコードします。 パスワードで保護された Web サイトに接続する際に、HTTP 基本認証に役立ちます。
- **socket receive socket# [integer] [raw]**：このコマンドは、リモート ホストから入ってくるデータを最大 10K まで内部バッファにため込みます。 このコマンドは、その後、新しいデータがこの内部バッファに入らないようにバッファをロックします。 引き続き、inspect を使用し、この内部 10K バッファに存在するすべてのデータを標準出力にダンプできます。 **注:** 新しいデータが入ってくる前に、10K 内部バッファ内に存在する以前のすべてのデータがフラッシュされます。次のリストは、各パラメータについての情報を提供します。**socket#** ：ソケット ファイル記述子（整数形式）。 この記述子は接続から返されます。**integer**：内部 10K バッファをロックし、ユーザに戻る前に待機する時間（ミリ秒）を表す整数値。 整数タイムアウトが指定されていない場合、receive はユーザに戻る前に 100 ミリ秒待機します。**raw** ：指定すると、文字列値が単純な文字列としてではなく実際の 16 進数のバイトとして転送されます。 たとえば、**0D0A** は ‘0’ ‘D’ ‘0’ ‘A' としてではなく、**0x0D 0x0A**（またはキャリ

ッジ リターン、ライン フィード）として送信されます。

- **socket waitfor socket# string [integer] [case-sensitive] [offset] [raw]**：このコマンドは、指定された文字列の引数を見つけるとすぐに戻る以外は、socket receive に似ています。 指定した文字列が見つかると、ゼロの ${STATUS} を返します。 それ以外の場合は 1 を返します。 受け取ったデータは、**socket inspect** コマンドを発行して表示できます。次のリストは、各パラメータについての情報を提供します。**socket#** ：ソケット ファイル記述子（整数形式）。 この記述子は接続から返されます。**string**：ゼロの ${STATUS} 結果（見つかったことを意味する）を返す必要のある特定の文字列。 文字列が見つかった場合、すぐに戻り、整数のタイムアウトの期間待機しません。**integer**：内部 10K バッファを口ックし、ユーザに戻る前に待機する時間（ミリ秒）を表す整数値。 整数タイムアウトが指定されていない場合、receive はユーザに戻る前に 100 ミリ秒待機します。**case-sensitive** ：指定すると、文字列比較で大文字と小文字が区別されます。 たとえば、**User:** は **user:** とは同じではありません。**offset**：受信したデータの何バイトまで文字列を検索するか指定します。 たとえば、**a0** を検索する際に 10 のオフセットを指定すると、受信したデータのうち 10 バイトまで **a0** を検索します。**raw** ：指定すると、文字列値が単純な文字列としてではなく実際の 16 進数のバイトとして転送されます。 たとえば、**0D0A** は ‘0’ ‘D’ ‘0’ ‘A' としてではなく、**0x0D 0x0A**（またはキャリッジ リターン、ライン フィード）として送信されます。
- **socket inspect socket# [pretty][raw]**：このコマンドは、ソケット データ バッファ（内部）から実際のデータを検査します。 データが見つかった場合、そのデータが標準出力に表示されます。 表示される文字が表示不可能な場合は、読みやすくするためにそれらの文字はドット（.）で表されます。次のリストは、各パラメータについての情報を提供します。**socket#** ：ソケット ファイル記述子（整数形式）。 この記述子は接続から返されます。**raw** ：指定すると、文字列値が単純な文字列としてではなく実際の 16 進数のバイトとして表示されます。 標準出力に **ABCD** と表示する代わりに、**41424344**（1 バイトの 16 進数で同様に表したもの）と表示します。**pretty** ：出力のプリティプリント。 各行にはデータの各バイトに相当する ASCII または 16 進数の両方が含まれます。 各行には 16 バイトが表示されます。 たとえば、**0x41 0x42 0x43 0x44 0x10 0x05ABCD** などです。
- **socket disconnect socket# [graceful]**：このコマンドは、リモート ホストへの接続を閉じます。 これは、データ送信が終わったことを認識するように、リモート ホストに RST を送信することによって行われます。次のリストは、各パラメータについての情報を提供します。**socket#** ：ソケット ファイル記述子（整数形式）。 この記述子は接続から返されます。**graceful**：リモート ホストに RST ではなく FIN を送信して接続を終了する正常（グレースフル）な切断。

ソケット管理

スイッチ上では常に、存在できるオープン（使用中）ソケットには 32 個の制限があります。 ユーザが socket connect を使用した後、そのソケットのファイル記述子（${SOCKET} 内に保存）に対して socket disconnect で終了しない場合、そのソケットをパラメータに指定して socket disconnect を呼び出さない限り、ソケットが開いたままになります。 スクリプトの中で開かれたソケットは、（セッション引数が socket connect に渡されない限り）スクリプトが終了すると閉じます。 セッションが終了すると、セッション内で開かれたソケットが閉じます。

ソケットが開いたままの場合は、通常、ユーザが適切に閉じずに接続した場合です。 そのようなソケットは、閉じられるか、スクリプト（またはターミナル接続）が閉じられるまで開いたままで使用されます。 **show sockets** コマンドは、開いている、または閉じているソケットをユーザが認識できるように、現在使用されているすべてのソケット ファイル記述子を一覧表示するために実装されています。

注: リモート ホストでソケットがタイムアウトした場合、またはソケットがリモート ホストによって閉じられた場合、ソケットの優れたアーキテクチャによってクリーンアップされ、使用済みソケットのリストから（**show sockets** コマンドを発行して検索され）除外されます。 これは、リモート ホストによって閉じられたソケットに対してユーザが別の転送を試行すると発生します（それ以外の場合、ユーザが必要とするまでアイドル状態で待機します）。

show sockets コマンドの出力結果を次に示します。

画面には、ソケット ID（ファイル記述子）、接続されたホスト/ポートのペア、ユーザ、および記述子が開いている期間を示すタイマーが一覧表示されます。 User フィールドは、**show lines** コマンドで CLI に出力される行 ID を表します。

socket receive を使用したデータの取得では、一度に 10K 分のデータをバッファできます。 ユーザが再度 socket receive を実行するまでこのバッファは変更されず、実行されると、その時点でバッファがクリーンアップされ、ネットワークから送信されてくる他のデータで再度満たされます。（ソケット コネクトから作成された）各ソケット記述子には、独自の 10K バッファがあります。

スクリプト ファイルのコピー

新しいバージョンのコードにアップグレードする場合、以前のバージョンの script ディレクトリで変更したすべてのスクリプト ファイルを、新しいバージョンにコピーする必要があります。

スイッチをアップグレードする前に、次の手順を実行します。

1. CSS スイッチに FTP 接続します。 管理ポートまたは VLAN 回路アドレスを使用してください。
2. script ディレクトリに移動します。
3. 編集したすべてのスクリプトをローカル マシンにダウンロードします。
4. スイッチをアップグレードします。
5. スイッチの新しい script ディレクトリにローカル マシンからスクリプトをアップロードします。

スクリプトのサンプル

いくつかのデフォルトを導入するために次のスクリプトが用意されています。 これらのサンプルには、DNS、Echo、NetBios、Finger、Time、HttpList、PingList、HttpAuth、Imap4、CookieSet、POP3、HttpTag、MailHost、および SMTP 用に記述されたスクリプトが含まれます。

show script ? コマンドを発行し、 スクリプトを表示します。 次にコマンドの出力例を示します。

```
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
```

確認

現在、この設定に使用できる確認手順はありません。

トラブルシューティング

次のガイドラインに従って、設定のトラブルシューティングを行います。

- **script play** コマンドを発行してコマンドラインからスクリプトを実行します。 スーパーユーザとしてログインしているときにこのコマンドを発行し、エラーなく終了することを確認してください。 エラーが発生する場合は、エラーの原因となった行が出力されるはずです。
- スクリプトを実行した際に実際に返ってくる結果と、スクリプトを実行した結果として予想される内容を比較観察するために、CSS と Web サーバ間でスニファ トレースを取得します。
- 5.x バージョンのコードでは、**use-output** コマンドが追加されました。 このコマンドは、**grep** コマンドを使用するすべてのスクリプトで使用する必要があります。 たとえば、**keepalive type script ap-kal-dns の引数には use-output** を指定します。

関連情報

- Web Network Services のソフトウェア サポート ページ
- CSS 11000 シリーズ コンテント サービス スイッチのハードウェア サポート (英語)
- CSS 11500
- Cisco WebNS CSS11500 ソフトウェアのダウンロード ページ
- Cisco WebNS CSS11000 ソフトウェアのダウンロード ページ
- テクニカルサポート - Cisco Systems