debugコマンドに関する重要な情報を理解する

内容

はじめに

前提条件

- 要件
- 使用するコンポーネント
- 表記法

背景説明

警告

デバッグを行う前に

- デバッグ出力の取得
  - コンソール ポート
  - Aux ポート
  - VTY ポート
  - 内部バッファへのメッセージのロギング
  - メッセージの UNIX Syslog サーバへのロギング
- その他のデバッグ前作業
- デバッグを中止するには

debug ip packet コマンドの使用

警告

条件付きで起動されるデバッグ

関連情報

はじめに

このドキュメントでは、Cisco IOS®プラットフォームで使用可能な debug ip packet コマンドを含む debug コマンドの使用に関する一般的なガイドラインについて説明します。

前提条件

要件

次の項目に関する知識があることが推奨されます。

- コンソール ポート、aux ポートおよび vty ポートを使用したルータへの接続
- 一般的な Cisco IOS 設定の問題

- Cisco IOS のデバッグ出力の解釈

使用するコンポーネント

このドキュメントの内容は、特定のソフトウェアやハードウェアのバージョンに限定されるものではありません。

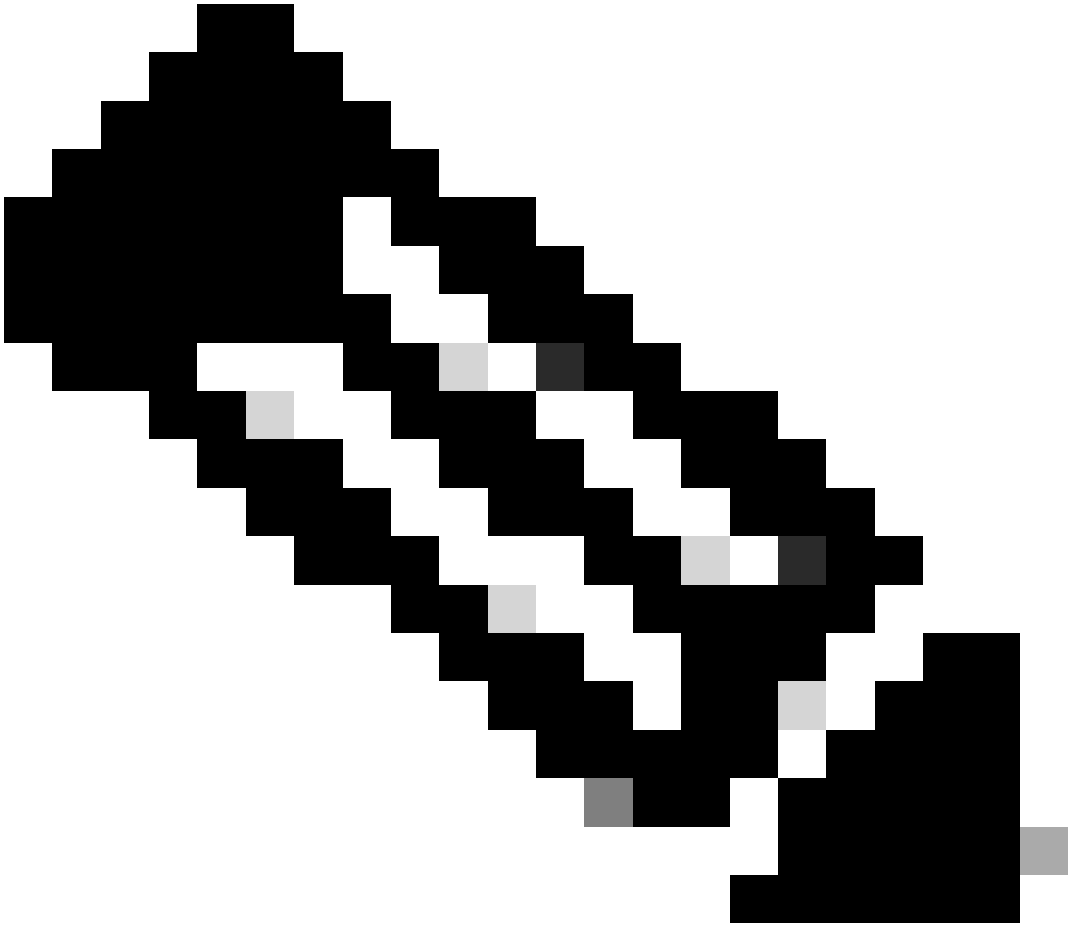
このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、クリアな（デフォルト）設定で作業を開始しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認してください。

表記法

表記法の詳細については、『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。

背景説明

ここでは、Cisco IOSプラットフォームで利用可能なデバッグを使用するための一般的なガイドラインと、コマンドや条件付きデバッグを正しく使用す **debug ip packet** る例について説明します。



注：このドキュメントでは、特定のdebugコマンドの使用方法と出力の解釈方法については説明していません。特定の debug コマンドの詳細については、該当する『Cisco Debug Command Reference』マニュアルを参照してください。

特権EXECコマンドの **debug** の出力には、一般的なプロトコルステータスやネットワークアクティビティに関連したさまざまなインターネットワーキングイベントについての診断情報が記載されています。

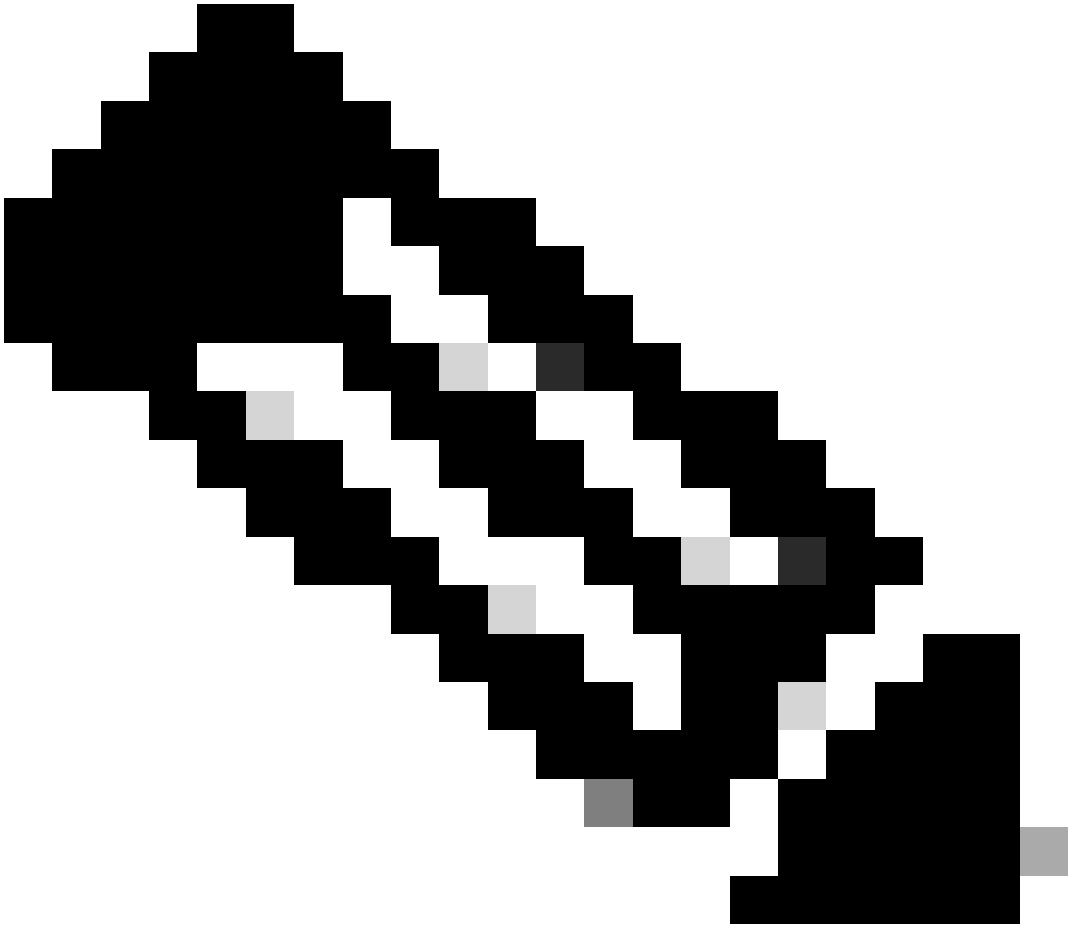
警告

debug コマンドは注意して使用してください。一般に、これらのコマンドは、特定の障害をトラブルシューティングする場合に限り、必ずルータの技術サポート担当者の指示に従って使用することをお勧めします。

インターネットワークに高い負荷が生じているときにデバッグを有効にすると、ルータの動作が中断する場合があります。そのため、ロギングが有効な場合、コンソール ポートがログ メッセージで過負荷になるとアクセス サーバは断続的にフリーズする可能性があります。

コマ **debug** ンドを開始する前に、このコマンドが生成できる出力とそれにかかる時間について必ず考慮してください。たとえば、ルータがBasic Rate Interface（BRI；基本速度インターフェイス）を1つ装備している場合、 **debug isdn q931** によってシステムに悪影響が及ぶことはおそらくありません。しかし、同じデバッグをフルE1設定のAS5800で実行すると、おそらく大量の入力が生成され、ハングして応答しなくなります。

デバッグを行う前に、コマンドを使用してCPUの負荷を調べ **show processes cpu** ます。デバッグを開始する前に、CPU を十分に使用できることを確認します。高い CPU 負荷に対処する方法の詳細については、『Cisco ルータの CPU 使用率が高い場合のトラブルシューティング』を参照してください。たとえば、Cisco 7200ルータにブリッジングを実行するATMインターフェイスを使用している場合、設定されたサブインターフェイスの量によっては、ルータの再起動でCPUを大量に使用する可能性があります。これは、各 Virtual Circuit（VC; 仮想回線）に Bridge Protocol Data Unit（BPDU; ブリッジ プロトコル データ ユニット）パケットを生成する必要があるためです。このような重要な時間にデバッグを開始すると、CPU使用率が大幅に上昇し、ハングやネットワーク接続の喪失を引き起こす可能性があります。



注：デバッグの実行中、特にデバッグの負荷が高いときには、ルータプロンプトは通常表示されません。しかし、ほとんどの場合、no debug all または undebug all コマンドを使用してデバッグを中止できます。デバッグの安全な使用方法の詳細については、「デバッグ出力の取得」セクションを参照してください。

デバッグを行う前に

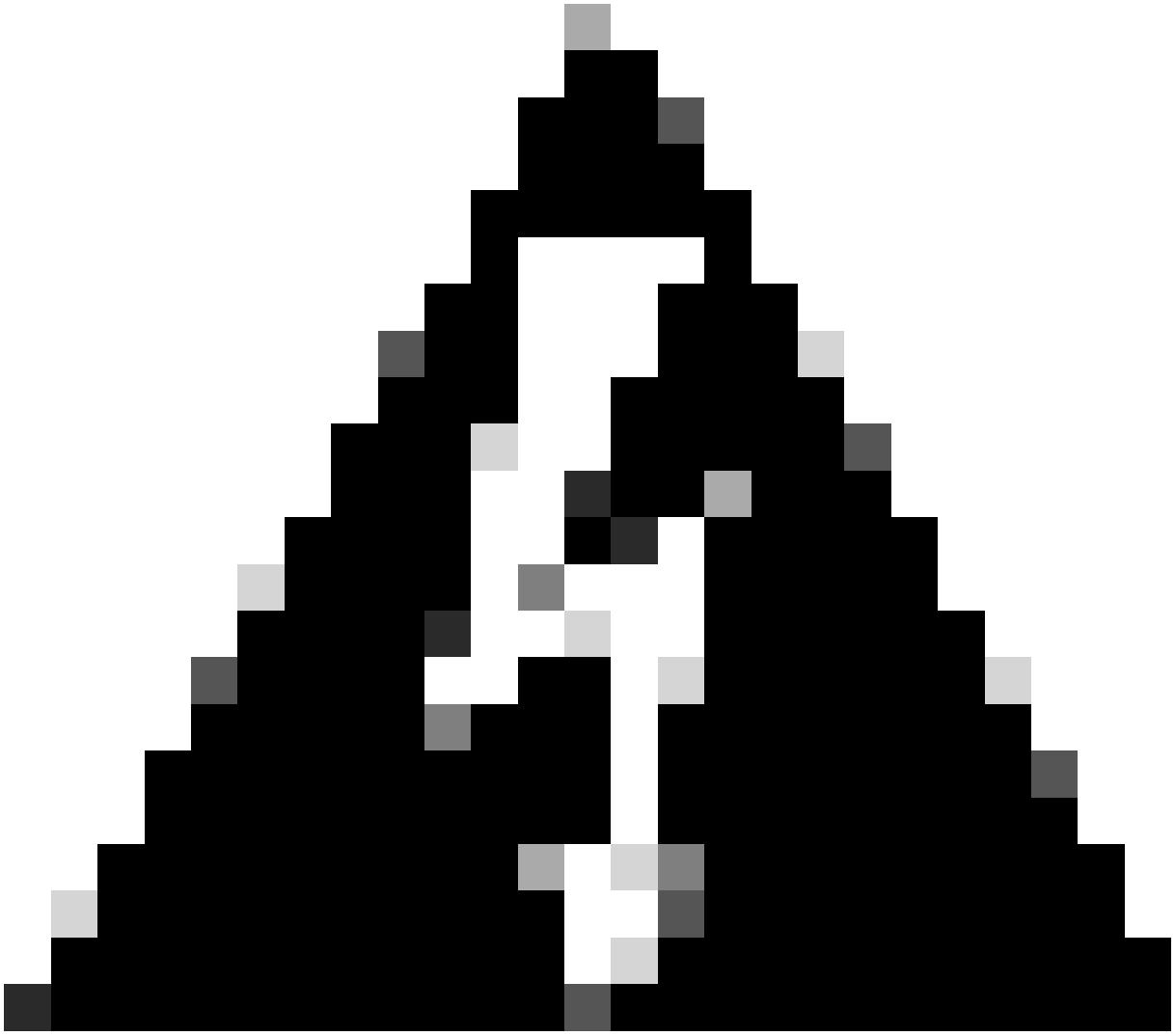
上記の内容に加えて、デバッグがプラットフォームの安定性に与える影響についても、必ず把握してください。また、ルータのどのインターフェイスに接続する必要があるかを考慮する必要があります。このセクションでは、いくつかのガイドラインを示します。

デバッグ出力の取得

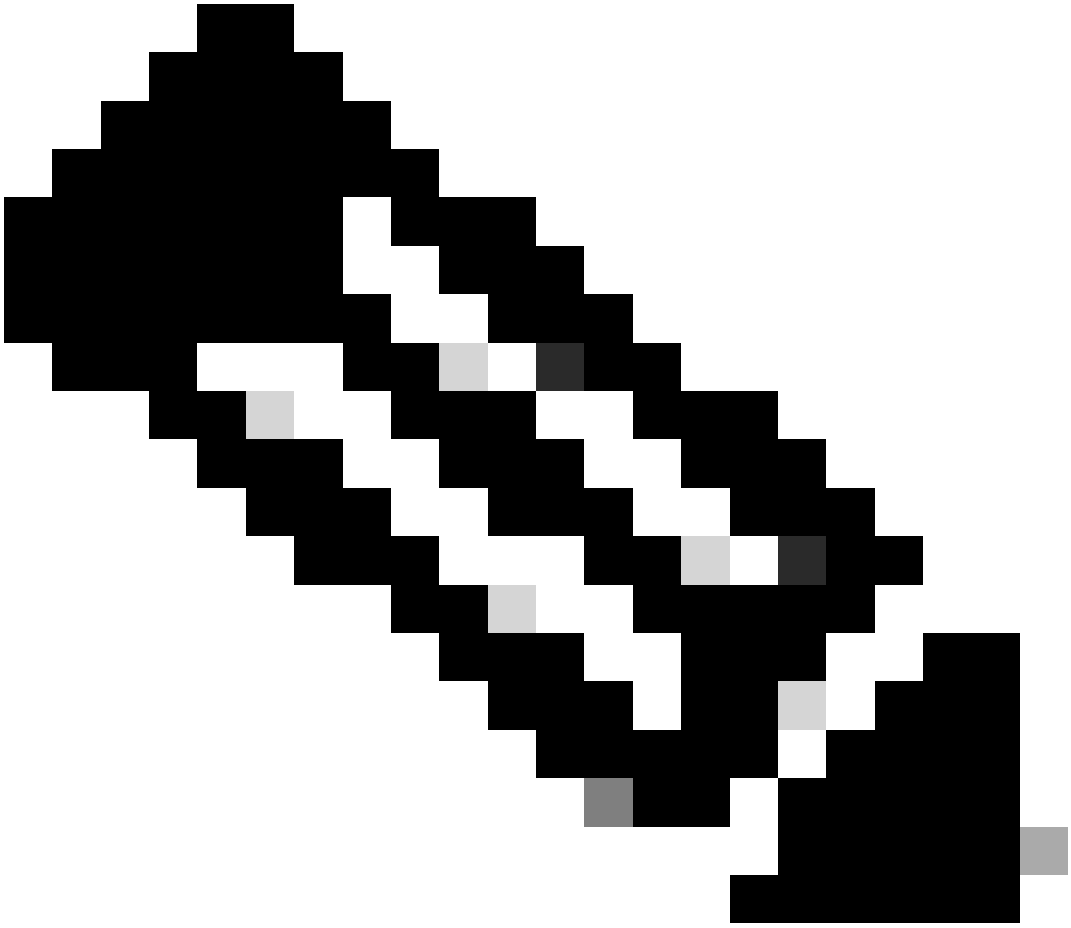
ルータはデバッグ出力を、コンソール ポート、aux ポート、および vty ポートなどさまざまなインターフェイスに表示できます。ルータは、内部バッファへのメッセージを外部 UNIX syslog サーバへログすることもできます。各方式の手順と注意事項については、次で説明します。

コンソール ポート

コンソールから通常の設定で接続しているときは、特別な作業を行う必要はありません。デバッグ出力は自動的に表示される必要があります。ただし、必要に応じ **logging console level** て設定されていること、およびコマンドでロギングが無効になっていないことを確認してくだ **no logging console** さい。



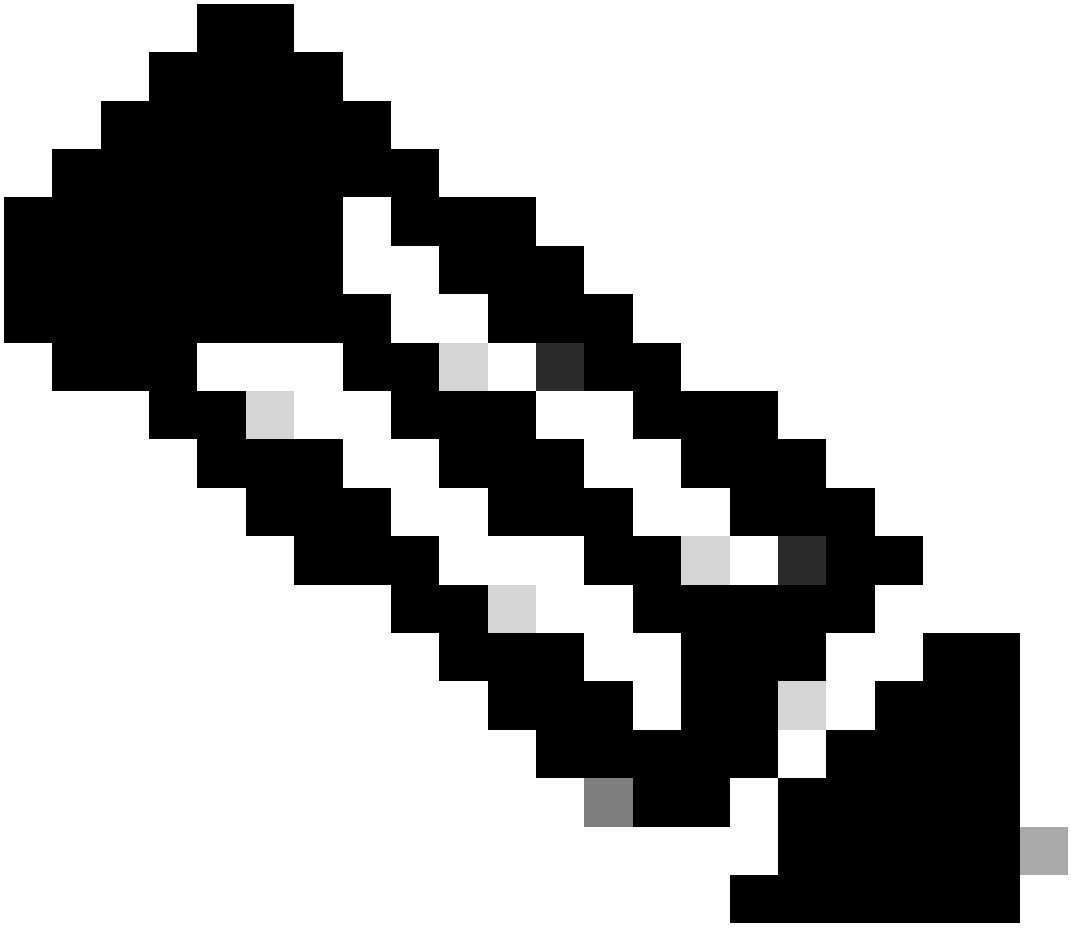
警告：ルータのコンソールポートに対する過剰なデバッグが原因でハングする場合があります。これは、ルータが自動的にコンソール出力を他のルータ機能より優先するためです。そのため、ルータがコンソールポートへの大きなデバッグ出力を処理している場合、ハングする可能性があります。そのためデバッグ出力が多過ぎる場合は、vty（telnet）ポートまたはログ バッファを使用してデバッグを行います。詳細については、次に説明します。



注：コンソールポートのロギングは、デフォルトで有効になっています。そのため、ユーザが実際には他のポートや方法（Aux、vty、バッファなど）を使用して出力結果を取得する場合でも、コンソール ポートは常にデバッグ出力を処理しています。そのため、シスコでは、通常の稼働状況では常に no logging console コマンドを有効にしておいて、デバッグ出力の取得には別の方法を使用することをお勧めします。コンソールを使用する必要がある状況では、一時的に logging console をオンに戻します。

Aux ポート

補助ポートを経由して接続している場合は、コマンドを入力 **terminal monitor** します。また、コマンドがルータ **no logging on** で有効になっていないことも確認します。



注:AUXポートを使用してルータを監視する場合、ルータがリブートしても、AUXポートにはブートシーケンスの出力が表示されないことに注意してください。ブート シーケンスを表示するには、コンソール ポートに接続します。

VTY ポート

補助ポートまたはTelnetを介して接続している場合は、コマンドを入力 **terminal monitor** します。コマンドが使用され **no logging on** ていないことも確認します。

内部バッファへのメッセージのロギング

デフォルトのロギング デバイスはコンソールです。他のデバイスを指定しない限り、すべてのメッセージはコンソールに表示さ

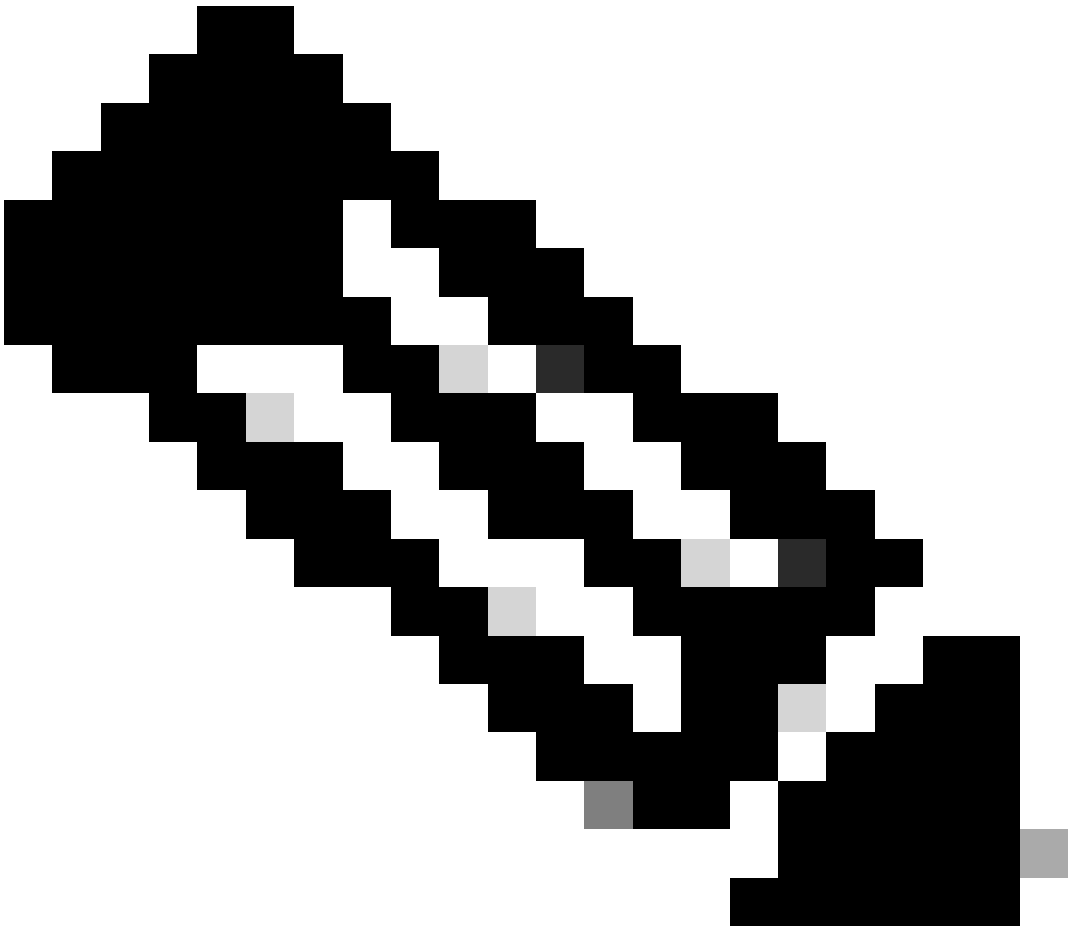
れます。

メッセージを内部バッファにログするには、 **logging buffered** 外部コンフィギュレーションコマンドを使用します。このコマンドの全構文は次のとおりです。

<#root>

logging buffered no logging buffered

logging buffered このコマンドは、ログメッセージをコンソールに書き込むのではなく、内部バッファにコピーします。バッファは実際には循環しており、新しいメッセージによって古いメッセージが上書きされます。バッファ内にログされているメッセージを表示するには、特権EXECコマンド **show logging** （.binファイル）を使用します。最初に表示されるメッセージは、バッファ内で最も古いメッセージです。バッファのサイズを指定できるだけでなく、ログされるメッセージの重大度も指定できます。



注：バッファサイズを入力する前に、ボックスで十分なメモリが使用可能であることを確認してください。使用可能なメモリ量を確認するには、Cisco IOSの show proc mem コマンドを使用します。

no logging buffered このコマンドは、バッファの使用を取り消し、メッセージをコンソール（デフォルト）に書き込みます。

メッセージの UNIX Syslog サーバへのロギング

メッセージを syslog サーバ ホストにログするには、logging ルータ設定コマンドを使用します。このコマンドの全構文は次のとおりです。

<#root>

```
logging <ip-address> no logging <ip-address>
```

loggingこのコマンドは、ロギングメッセージを受信するsyslogサーバホストを識別します。引数 <ip-address> は、ホストの IP アドレスです。このコマンドを何度も発行すると、ロギング メッセージを受信する syslog サーバのリストが作成されます。

no logging コマンドは、指定されたアドレスのsyslogサーバをsyslogのリストから削除します。

その他のデバッグ前作業

- 使用するターミナル エミュレータ ソフトウェア（HyperTerminal など）をセットアップして、デバッグ出力をファイルに取り込みます。たとえば、HyperTerminalで、をクリックし**Transfer**、をクリックして **Capture Text** 、適切なオプションを選択します。詳細は、『ハイパーターミナルからのテキスト出力のキャプチャ』を参照してください。その他のターミナル エミュレーション ソフトウェアについては、付属のマニュアルを参照してください。

- 次のコマンドを使用して、ミリ秒（ミリ秒）のタイムスタンプ **service timestamps** を有効にします。

<#root>

```
router(config)#

service timestamps debug datetime msec

router(config)#

service timestamps log datetime msec
```

これらのコマンドはタイム スタンプを MMM DD HH:MM:SS の形式でデバッグに追加し、日付と時間をシステム クロックに従って表示します。システム クロックをまだ設定していない場合は、日付と時刻の前にアスタリスク (*) が表示されて日付と時刻が正確でないことが示されます。

一般にはミリ秒のタイムスタンプを設定することをお勧めします。これによりデバッグ出力を見るときにより明確になります。タイムスタンプをミリ秒に設定すると、相互に関連しているさまざまなデバッグ イベントのタイミングをより明確に知ることができます。ただし、コンソールポートから大量のメッセージが出力される場合は、イベントの実際のタイミングと関連付けられないことに注意してください。たとえば、200個のVCがあるボックスでallを有 **debug x25** 効にし、出力がバッファに口グされる(**no logging console logging buffered** andcommandsを使用)場合、 (バッファ内の) debug出力に表示されるタイムスタンプは、パケットがインターフェイスを通過する正確な時間にはなりません。そのため、ミリ秒のタイムスタンプはパフォーマンス問題を検査するために使用するのではなく、イベントがいつ発生したかに関する関連情報を取得するために使用します。

デバッグを中止するには

デバッグを停止するには、theorcommandsを使用**no debug allundebug all**します。デバッグがオフになっていることを**show debug**、コマンドを使用して確認します。

no logging console terminal no monitor コマンドで停止されるのは、それぞれコンソールへの出力とAuxまたはvtyへの出力だけであることに注意してください。デバッグは停止されないため、ルータのリソースは消費されています。

debug ip packet コマンドの使用

debug ip packet このコマンドは、ルータによってファーストスイッチングされないパケットに関する情報を生成します。しかし、すべてのパケットの出力を作成するため、出力は大規模になりルータがハングする可能性があります。このため、このセクションで説明する最も厳密なコントロールの下でのみ使 **debug ip packet** 用してください。

debug ip packet の出力を制限する最もよい方法は、デバッグにリンクされたアクセスリストを作成することです。アクセスリストの基準に一致するパケットだけが対象になります **debug ip packet** 。このアクセス リストをインターフェイスに適用する必要は

ありません。デバッグ操作に適用します。

を使用する前に **debugging ip packet** 、ルータがデフォルトでファーストスイッチング、またはCEFスイッチング（そのように設定されている場合）を実行していることに注意してください。これは、これらの技法が設定されていると、パケットはプロセッサに送られず、そのためデバッグには何も表示されないことを示します。これが機能するには、 **no ip route-cache** （ユニキャストパケットの場合）または **no ip mroute-cache** （マルチキャストパケットの場合）を使用して、ルータでのファーストスイッチングを無効にする必要があります。これは、トラフィックが流れるべきインターフェイスに適用する必要があります。コマンドを使用してこれを確認 **show ip route** します。

警告

- ファースト スイッチングを無効にしているルータが大量のパケットを処理すると、CPU の使用率が瞬間的に上昇し、ハングしたりピアへの接続が不通になったりする場合があります。

- Multi Protocol Label Switching（MPLS）を実行しているルータのファースト スイッチングを無効にしないでください。MPLS は CEF と併用されます。したがって、インターフェイス上のファースト スイッチングを無効にすると重大な影響を与える場合があります。

次のシナリオ例を参照してください。

ルータ 122 には、次のアクセス リストが設定されています。

<#root>

access-list 105 permit icmp host 10.10.10.2 host 10.1.1.1 access-list 105 permit icmp host 10.1.1.1 host

このアクセスリストでは、ホストrouter_121（IPアドレス10.10.10.2）からホストrouter_123（IPアドレス10.1.1.1）へのインターネット制御メッセージプロトコル(ICMP)パケットと、その逆方向が許可されています。いずれかの方向のパケットを許可することが

重要です。許可されていないと、ルータは戻りのICMPパケットをドロップする可能性があります。

router_122 の 1 つのインターフェイスだけでファースト スイッチングを解除します。つまり、パケットを代行受信するCisco IOSの観点から見ると、そのインターフェイスを宛先とするパケットのデバッグ情報だけが表示されます。デバッグの観点から見ると、そのようなパケットは「d=」付きで表示されます。他のインターフェイスではファーストスイッチングをオフにしていないため、戻りのパケットは対象になりません **debug ip packet** 。次の出力に、ファースト スイッチングを無効にする方法を示します。

<#root>

router_122(config)#

interface virtual-template 1

router_122(config-if)#

no ip route-cache

router_122(config-if)#

end

ここで、先に定義したアクセスリスト(access-list 105)を使用してアク **debug ip packet** ティブ化する必要があります。

<#root>

router_122#

debug ip packet

detail 105 IP packet debugging is on (detailed) for access list 105 router_122# 00:10:01: IP: s=10.1.1

次に、他のインターフェイス(router_122)のファーストスイッチングを削除します。つまり、これら2つのインターフェイスを通過するすべてのパケットは、パケット交換されます(これは、 **debug ip packet**

<#root>

router_122(config)#

interface serial 3/0

router_122(config-if)#

no ip route-cache

router_122(config-if)#

end

router_122# 00:11:57: IP:

s=10.10.10.2

(Virtual-Access1),

d=10.1.1.1

(Serial3/0), g=172.16.1.6, len 100, forward 00:11:57:

ICMP type=8

, code=0 *! -- ICMP packet (echo) from 10.10.10.2 to 10.1.1.1* 00:11:57: IP:

s=10.1.1.1

(Serial3/0),

d=10.10.10.2

(Virtual-Access1), g=10.10.10.2, len 100, forward 00:11:57:

ICMP type=0

, code=0 *! -- ICMP return packet (echo-reply) from 10.1.1.1 to 10.10.10.2* 00:11:57: IP: s=10.10.10.2

debug ip packet の出力には、アクセス リストの条件に一致しないパケットは表示されないことに注意してください。この手順のその他の情報については、『ping および traceroute コマンドについて』を参照してください。

アクセス リストの作成方法についての詳細は、『標準 IP アクセス リスト ロギング』を参照してください。

条件付きで起動されるデバッグ

デバッグの条件付き起動機能を有効にすると、ルータはルータを指定したインターフェイスで出入りするパケットのデバッグ メッセージを生成します。このルータは別のインターフェイスに出入りするパケットのデバッグ出力を生成しません。

条件付きデバッグの簡単な実装を確認します。次のシナリオを考えてみます。次に示すルータ(trabol)には、HDLCカプセル化を実行している2つのインターフェイス (シリアル0とシリアル3) があります。

debug serial interface コマンドnormalcommandを使用すると、HDLCキープアライブがすべてのインターフェイスで受信されることを確認できます。キープアライブを両方のインターフェイスで確認できます。

<#root>

traxbol#

debug serial interface

Serial network interface debugging is on traxbol# *Mar 8 09:42:34.851:

Serial0: HDLC

myseq 28, mineseen 28*, yourseen 41, line up *! -- HDLC keeplaive on interface Serial 0* *Mar 8 09:42:

Serial3: HDLC

myseq 26, mineseen 26*, yourseen 27, line up *! -- HDLC keeplaive on interface Serial 3* *Mar 8 09:42:

インターフェイス シリアル 3 の条件付きデバッグを有効にします。インターフェイス シリアル 3 のデバッグ情報だけが表示されるようにします。コマ **debug interface <interface_type interface_number** >ンドを使用します。

<#root>

traxbol#

debug interface serial 3

Condition 1 set

コ **show debug condition** マンドを使用して、条件付きデバッグが有効なことを確認します。インターフェイス シリアル 3 の条件が有効なことに注意してください。

<#root>

traxbol#

show debug condition

Condition 1: interface Se3 (1 flags triggered) Flags: Se3 traxbol#

今度はインターフェイス シリアル 3 のデバッグだけが表示されていることに注意してください。

<#root>

*Mar 8 09:43:04.855:

Serial3: HDLC

myseq 29, mineseen 29*, yourseen 30, line up *Mar 8 09:43:14.855:

Serial3: HDLC

myseq 30, mineseen 30*, yourseen 31, line up

条件付 **undebug interface <interface_type interface_number**> きデバッグを解除するには、このコマンドを使用します。条件付きの起動を解除する前に、 (undebug all などを使用して) デバッグをオフにしておくことをお勧めします。これは、条件が解除されたときデバッグ出力が集中するのを回避するためです。

<#root>

traxbol#

undebug interface serial 3

This condition is the last interface condition set. Removing all conditions can cause a flood of debug

y

```
 Condition 1 has been removed traxbol
```

今度は、シリアル 0 とシリアル 3 の両インターフェイスのデバッグが表示されていることに注意してください。

<#root>

*Mar 8 09:43:34.927:

Serial3: HDLC

```
 myseq 32, mineseen 32*, yourseen 33, line up *Mar 8 09:43:44.923:
```

Serial0: HDLC

```
 myseq 35, mineseen 35*, yourseen 48, line up
```

警告：一部のデバッグ操作は、もともと条件付きです。1 つの例としては ATM デバッグがあります。ATMデバッグでは、デバッグを有効にするインターフェイスを明示的に指定する必要があります。すべてのatmインターフェイスでデバッグを有効にして条件を指定する必要はありません。

次のセクションでは、ATM パケット デバッグを 1 つのサブインターフェイスに制限する正しい方法を説明します。

<#root>

arielle-nrp2#

debug atm packet interface atm 0/0/0.1

!--- Note that you explicitly specify the sub-interface to be used for debugging ATM packets debuggi

Displaying packets on interface ATM0/0/0.1 only

arielle-nrp2# *Dec 21 10:16:51.891: ATM0/0/0.1(O): VCD:0x1 VPI:0x1 VCI:0x21 DM:0x100 SAP:AAAA CTL:03 0

(適用された条件で) すべてのインターフェイスを有 **atm debugging** 効にしようとすると、ルータに大量のATMサブインターフェイスがある場合はハングする可能性があります。ここで示されているのは ATM デバッグの誤った方法の一例です。

この場合には条件が適用されていることがわかりますが、効果がないこともわかります。別のインターフェイスからのパケットも確認できます。この実験シナリオでは、インターフェイスは 2 つだけでトラフィックもほとんどありません。インターフェイスの数が多いと、すべてのインターフェイスのデバッグ出力は極端に高くなり、ルータが停止する原因になります。

<#root>

arielle-nrp2#

show debugging condition

Condition 1: interface AT0/0/0.1

(1 flags triggered) Flags: AT0/0/0.1 *! -- A condition for a specific interface.* arielle-nrp2#

debug atm packet

ATM packets debugging is on Displaying all ATM packets arielle-nrp2# *Dec 21 10:22:06.727:

ATM0/0/0.2

(O): *! -- You see debugs from interface ATM0/0/0/.2, even though the condition ! -- specified ONLY AT0*

ATM0/0/0.1

(O): *!--- You also see debugs for interface ATM0/0/0.1 as you wanted.* VCD:0x1 VPI:0x1 VCI:0x21 DM:0x1

関連情報

- ダイヤルおよびアクセスに関するサポート ページ
- シスコのテクニカルサポートとダウンロード

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。