Debug 명령에 대한 중요 정보 이해

목차

소개

이 문서에서는 Cisco IOS® 플랫폼에서 사용 가능한 debug 명령을 비롯한 debug ip packet 명령 사용에 대한 일반적인 지침을 설명합니다.

사전 요구 사항

요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- 콘솔, aux 및 vty 포트를 사용하여 라우터에 연결
- 일반적인 Cisco IOS 컨피그레이션 문제

- Cisco IOS 디버그 출력 해석

사용되는 구성 요소

이 문서는 특정 소프트웨어 및 하드웨어 버전으로 한정되지 않습니다.

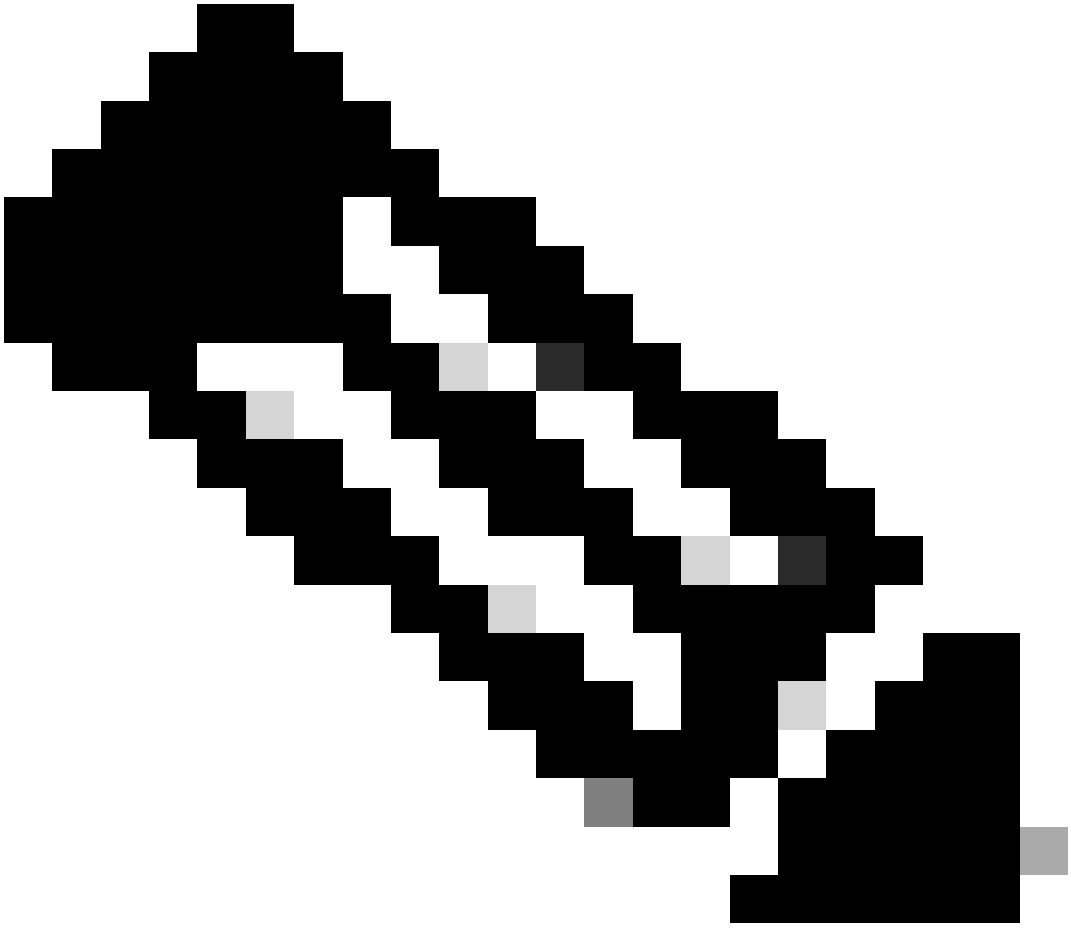
이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

표기 규칙

문서 규칙에 대한 자세한 내용은 Cisco 기술 팁 표기 규칙을 참고하십시오.

배경 정보

이 페이지에서는 Cisco IOS 플랫폼에서 사용할 수 있는 디버깅 사용에 대한 몇 가지 일반적인 지침과 명령 및 조건부 디버깅을 올바르게 사용하는 예 **debug ip packet** 를 제공합니다.



참고: 이 문서에서는 특정 debug 명령 및 출력을 사용하고 해석하는 방법에 대해 설명하지 않습니다. 특정 debug 명령에 대한 자세한 내용은 해당 Cisco Debug 명령 참조 설명서를 참조하십시오.

특권 EXEC **debug** 명령의 출력은 프로토콜 상태 및 네트워크 활동과 관련된 다양한 인터네트워킹 이벤트를 포함하는 진단 정보를 제공합니다.

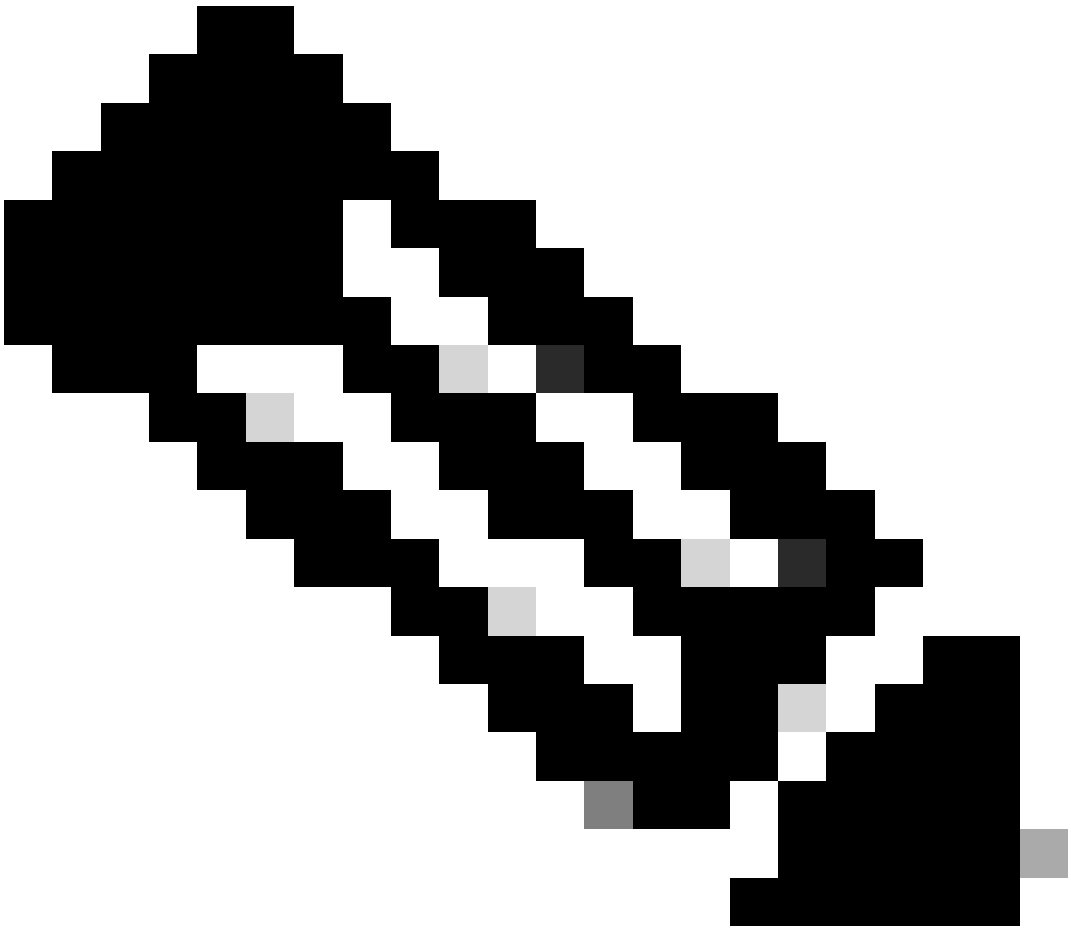
경고

debug 주의해서 명령을 사용합니다. 일반적으로 이러한 명령은 특정 문제를 트러블슈팅하는 경우 라우터 기술 지원 담당자의 지시 하에서만 사용하는 것이 좋습니다.

디버깅을 활성화하면 내부 네트워크에서 높은 로드 조건이 발생할 때 라우터의 작업이 중단될 수 있습니다. 따라서 로깅이 활성화된 경우 콘솔 포트에 로그 메시지가 오버로드되는 즉시 액세스 서버가 간헐적으로 중지될 수 있습니다.

명령을 시작하기 **debug** 전에 이 명령이 생성할 수 있는 출력과 이 명령이 걸리는 시간을 항상 고려하십시오. 예를 들어, 하나의 기본 속도 인터페이스(BRI)가 있는 라우터가 있는 경우, **debug isdn q931** 아마도 시스템에 해가 되지 않을 것입니다. 그러나 전체 E1 컨피그레이션을 사용하는 AS5800에서 동일한 디버그를 수행하면 너무 많은 입력이 생성되어 응답하지 않고 멈출 수 있습니다.

디버깅하기 전에 이 명령을 사용하여 CPU 로드를 **show processes cpu** 확인합니다. 디버그를 시작하기 전에 사용 가능한 충분한 CPU가 있는지 확인합니다. CPU 로드가 많은 경우 이를 처리하는 방법에 대한 자세한 내용은 Cisco 라우터에서 CPU 사용률이 높은 문제 해결을 참조하십시오. 예를 들어, ATM 인터페이스가 브리징을 수행하는 Cisco 7200 라우터가 있는 경우 구성된 하위 인터페이스의 양에 따라 라우터를 다시 시작하면 많은 CPU를 사용할 수 있습니다. 여기서 이유는 각 가상 회로(VC)마다 BPDU(Bridge Protocol Data Unit) 패킷을 생성해야 하기 때문이다. 이러한 중요한 시간 동안 디버그를 시작하면 CPU 사용률이 크게 상승하여 중단 또는 네트워크 연결 손실이 발생할 수 있습니다.



참고: 디버그가 실행 중일 때는 특히 디버그가 많은 경우 일반적으로 라우터 프롬프트가 표시되지 않습니다. 그러나 대부

분의 경우 디버그를 중지하기 위해 no debug all 또는 undebug all 명령을 사용할 수 있습니다. 디버그를 안전하게 사용하는 방법에 대한 자세한 내용은 디버그 출력 가져오기 섹션을 참조하십시오.

디버깅 전

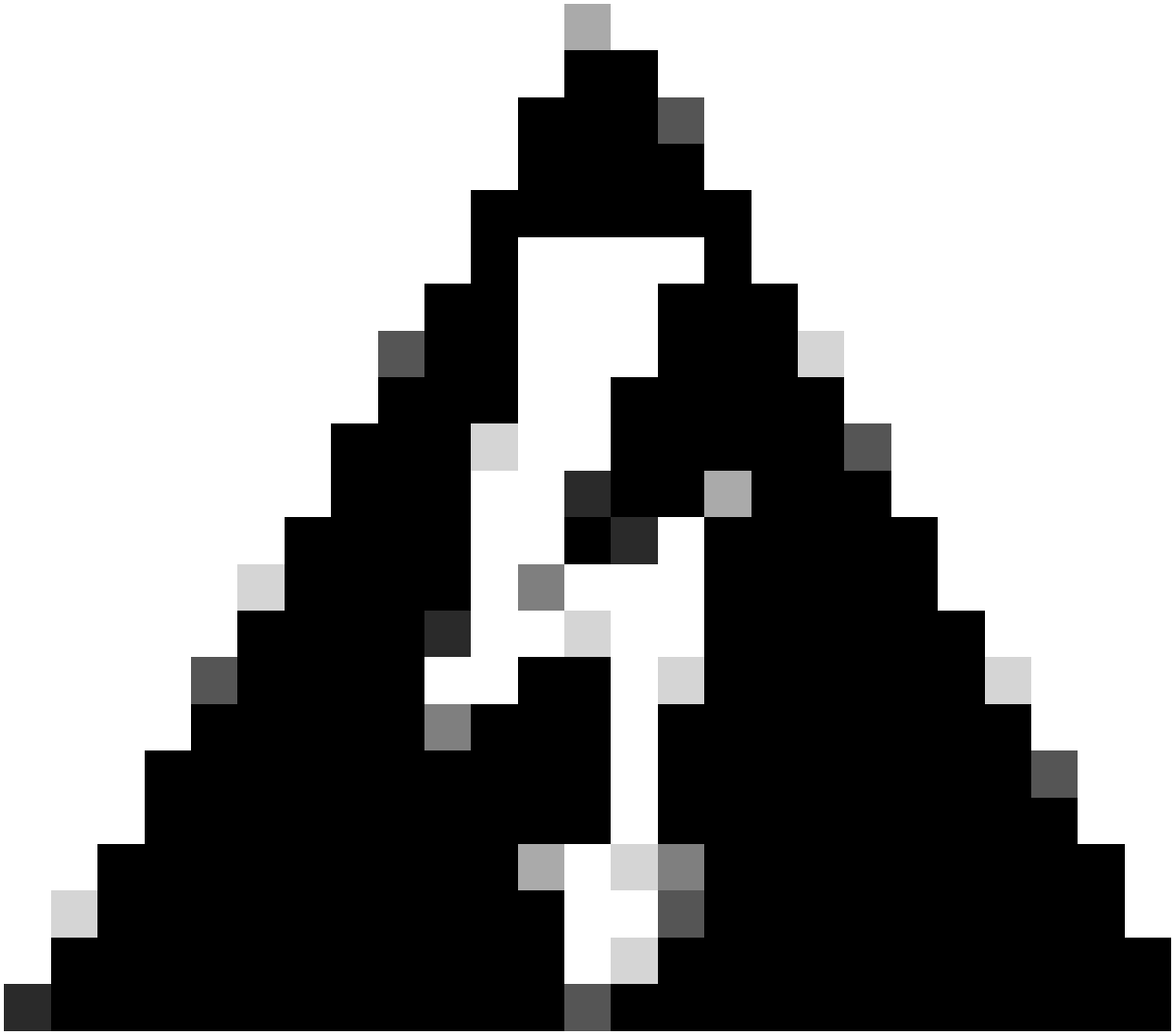
위에서 언급한 점 외에, 디버깅이 플랫폼의 안정성에 미치는 영향을 이해해야 합니다. 라우터에서 연결해야 하는 인터페이스도 고려해야 합니다. 이 섹션에는 몇 가지 지침이 있습니다.

디버그 출력 가져오기

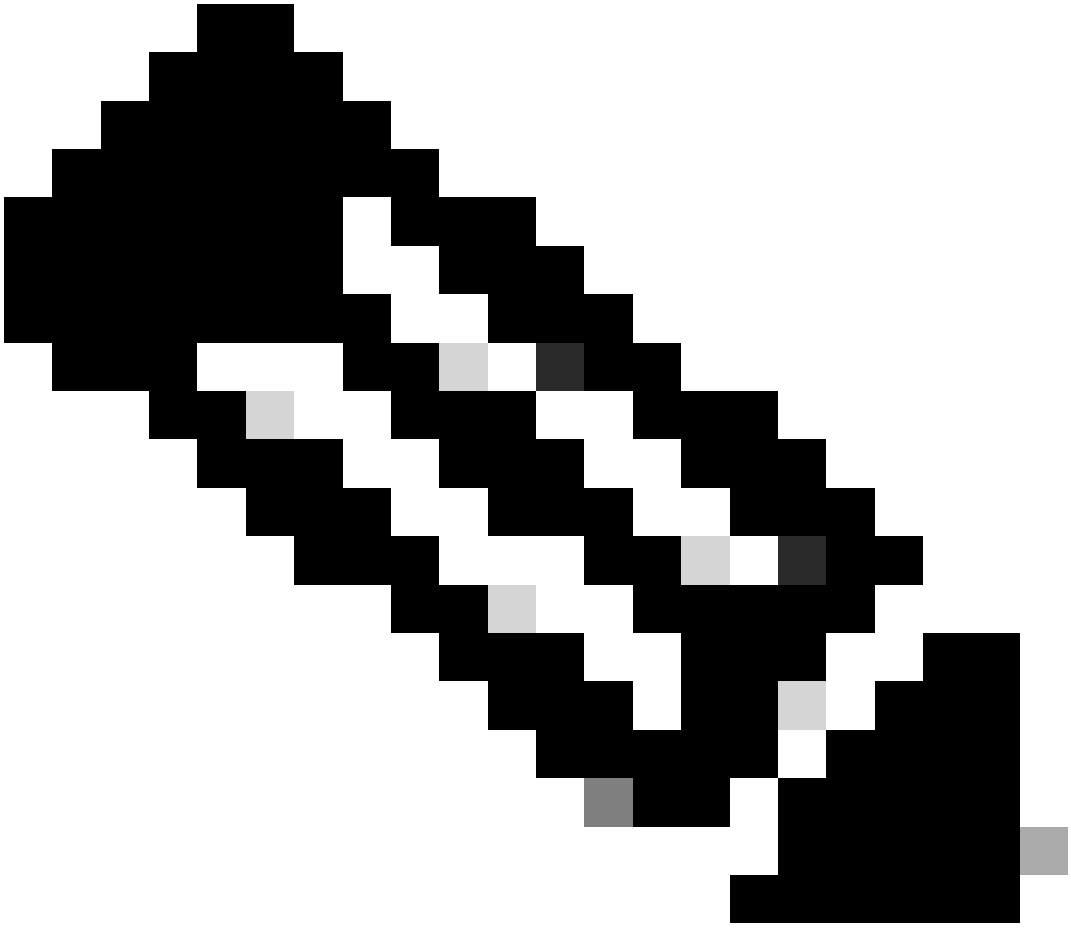
라우터는 콘솔, aux 및 vty 포트를 비롯한 다양한 인터페이스에 디버그 출력을 표시할 수 있습니다. 라우터는 내부 버퍼에 외부 unix syslog 서버에 메시지를 로깅할 수도 있습니다. 각 방법에 대한 지침 및 주의 사항은 다음에 설명되어 있습니다.

콘솔 포트

콘솔에 연결된 경우 일반 컨피그레이션에서는 추가 작업을 수행할 필요가 없습니다. 디버그 출력이 자동으로 표시되어야 합니다. 그러나 이 **logging console level** 가 원하는 대로 설정되었는지, 그리고 로깅이 이 명령으로 비활성화되지 않았는지 **no logging console** 확인합니다.



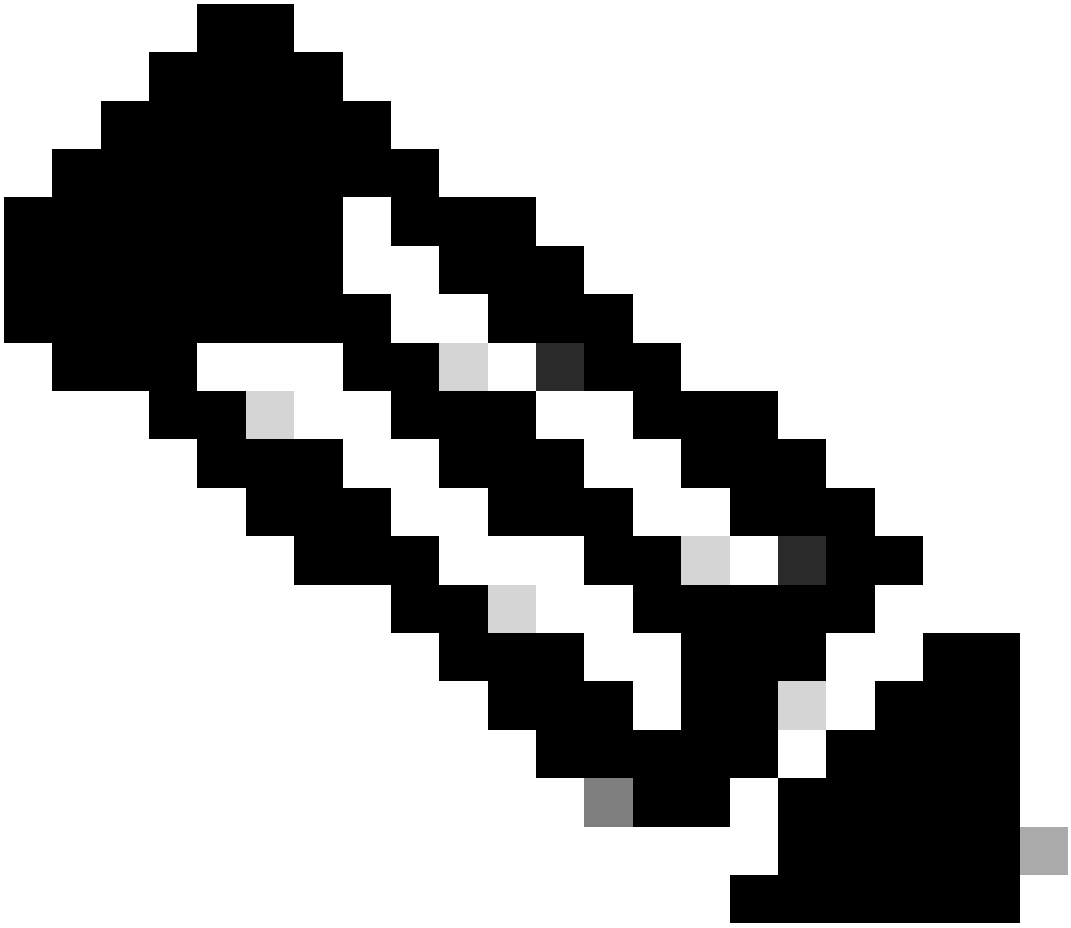
경고: 라우터의 콘솔 포트에 대한 과도한 디버깅이 중단될 수 있습니다. 이는 라우터가 다른 라우터 기능보다 콘솔 출력의 우선 순위를 자동으로 지정하기 때문입니다. 따라서 라우터가 콘솔 포트에 대한 대규모 디버그 출력을 처리하는 경우 중단될 수 있습니다. 따라서 디버그 출력이 과도한 경우 vty(telnet) 포트 또는 로그 버퍼를 사용하여 디버그를 가져옵니다. 자세한 내용은 다음에 제공됩니다.



참고: 기본적으로 로깅은 콘솔 포트에서 활성화되어 있습니다. 따라서 콘솔 포트는 출력을 캡처하기 위해 실제로 다른 포트나 메서드(예: Aux, vty 또는 버퍼)를 사용하는 경우에도 항상 디버그 출력을 처리합니다. 따라서 Cisco에서는 정상 작동 조건에서 no logging console 명령을 항상 활성화하고 다른 방법을 사용하여 디버그를 캡처할 것을 권장합니다. 콘솔을 사용해야 하는 경우 일시적으로 로깅 콘솔을 다시 켭니다.

Aux 포트

보조 포트를 통해 연결된 경우 명령을 **terminal monitor** 입력합니다. 또한 라우터 **no logging on** 에서 명령이 활성화되지 않았는지 확인합니다.



참고: Aux 포트를 사용하여 라우터를 모니터링하는 경우 라우터가 재부팅될 때 Aux 포트는 부팅 시퀀스 출력을 표시하지 않습니다. 부팅 시퀀스를 보려면 콘솔 포트에 연결합니다.

VTY 포트

보조 포트 또는 텔넷을 통해 연결된 경우 명령을 **terminal monitor** 입력합니다. 또한 명령 **no logging on** 이 사용되지 않았는지 확인합니다.

내부 버퍼에 메시지 로깅

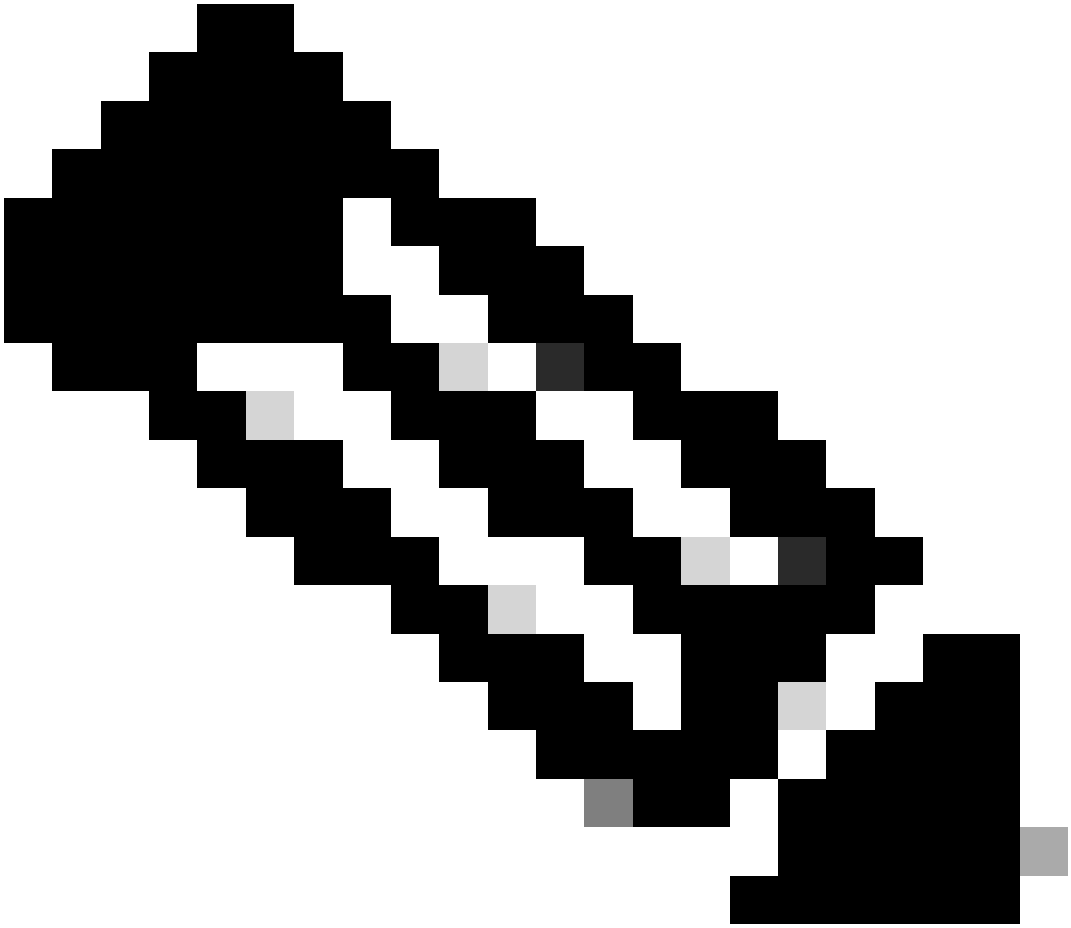
기본 로깅 장치는 콘솔입니다. 달리 지정되지 않는 한 모든 메시지가 콘솔에 표시됩니다.

내부 버퍼에 메시지를 로깅하려면 outer **logging buffered** 컨피그레이션 명령을 사용합니다. 다음은 이 명령의 전체 구문입니다.

<#root>

logging buffered no logging buffered

logging buffered 이 명령은 로그 메시지를 콘솔에 쓰지 않고 내부 버퍼에 복사합니다. 버퍼는 원래 순환형이므로 새 메시지가 이전 메시지를 덮어씁니다. 버퍼에 로깅된 메시지를 표시하려면 특권 EXEC 명령을 사용합니다 **show logging** . 표시되는 첫 번째 메시지는 버퍼에서 가장 오래된 메시지입니다. 로깅할 메시지의 심각도 수준과 버퍼의 크기를 지정할 수 있습니다.



참고: 버퍼 크기를 입력하기 전에 상자에 충분한 메모리가 있는지 확인하십시오. 사용 가능한 메모리를 show proc mem 보려면 Cisco IOS 명령을 사용합니다.

---

no logging buffered 이 명령은 버퍼 사용을 취소하고 콘솔(기본값)에 메시지를 씁니다.

UNIX Syslog 서버에 메시지 로깅

syslog 서버 호스트에 메시지를 로깅하려면 logging router configuration 명령을 사용합니다. 이 명령의 전체 구문은 다음과 같습니다.

<#root>

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

logging이 명령은 로깅 메시지를 수신할 syslog 서버 호스트를 식별합니다. **< ip-address**> 인수는 호스트의 IP 주소입니다. 이 명령을 두 번 이상 실행하면 로깅 메시지를 수신하는 syslog 서버 목록을 작성합니다.

no logging 이 명령은 지정된 주소의 syslog 서버를 syslog 목록에서 삭제합니다.

기타 사전 디버그 작업

- 디버그 출력을 파일에 캡처할 수 있도록 터미널 에뮬레이터 소프트웨어(예: HyperTerminal)를 설정합니다. 예를 들어 하이퍼터미널에서 을 클릭한**Transfer** 다음 을 클릭하고 **Capture Text** 적절한 옵션을 선택합니다. 자세한 내용은 하이퍼터미널에서 텍스트 출력 캡처를 참조하십시오. 다른 터미널 에뮬레이터 소프트웨어는 소프트웨어 설명서를 참조하십시오.

- 다음 명령을 사용하여 밀리초(msec) 타임스탬프 **service timestamps** 를 활성화합니다.

<#root>

```
router(config)#
```

```
service timestamps debug datetime msec
```

```
router(config)#
```

```
service timestamps log datetime msec
```

이러한 명령은 MMM DD HH:MM:SS 형식의 디버깅에 타임스탬프를 추가하여 시스템 시계에 따라 날짜와 시간을 나타냅니다. 시스템 시계가 설정되지 않은 경우 날짜 및 시간 앞에 별표(*)가 표시되어 날짜와 시간이 정확하지 않을 수 있음을 나타냅니다.

디버그 출력을 볼 때 밀리초 타임스탬프를 구성하는 것이 좋습니다. 이 타임스탬프는 높은 수준의 명확성을 제공합니다. 밀리초의 타임스탬프는 서로 관련된 다양한 디버그 이벤트의 타이밍을 더 잘 나타냅니다. 그러나 콘솔 포트에서 많은 메시지를 출력하면 이벤트의 실제 타이밍과 상관관계를 분석할 수 없습니다. 예를 들어, VC가 200개인 상자에서 **debug x25** 모두 활성화하고 출력이 버퍼에 로깅된 경우(**no logging console logging buffered** theandcommands 사용), 디버그 출력에 표시된 타임스탬프(버퍼 내)는 패킷이 인터페이스를 통과하는 정확한 시간이 될 수 없습니다. 따라서 msec 타임스탬프를 사용하여 성능 문제를 증명하지 말고 이벤트가 발생하는 시기에 대한 상대 정보를 얻으십시오.

디버깅을 중지하려면

디버그를 중지하려면 theorcommands를 **no debug allundebug all**사용합니다. 명령을 사용하여 디버그가 꺼져 있는지 확인합니다**show debug**.

commandsandonly를 **no logging console terminal no monitor** 사용하면 콘솔, Aux 또는 vty에 각각 출력이 출력되지 않습니다. 디버깅을 중지하지 않으므로 라우터 리소스를 모두 사용합니다.

debug ip packet 명령 사용

debug ip packet 이 명령은 라우터가 빠르게 전환하지 않는 패킷에 대한 정보를 생성합니다. 그러나 모든 패킷에 대해 출력을 생성하므로 출력이 광범위하여 라우터가 정지할 수 있습니다. 따라서 이 절에서 **debug ip packet** 설명한 대로 가장 엄격한 제어 하에 사용해야 합니다.

출력을 제한하는 가장 좋은 방법 **debug ip packet** 은 디버깅에 연결된 액세스 목록을 만드는 것입니다. 액세스 목록 기준과 일치하는 패킷에만 적용 가능합니다 **debug ip packet** . 이 액세스 목록은 인터페이스에 적용할 필요가 없으며 디버그 작업에 적용됩니다.

사용하기 **debugging ip packet** 전에 라우터가 기본적으로 고속 스위칭을 수행 중이거나, 구성된 경우 CEF 스위칭을 수행 중일 수 있습니다. 즉, 이러한 기술이 적용되면 패킷이 프로세서에 제공되지 않으므로 디버깅에 아무 것도 표시되지 않습니다. 이 기능이 작동하려면 (유니캐스트 패킷의 경우) 또는 **no ip route-cache** (멀티캐스트 패킷의 **no ip mroute-cache** 경우)를 사용하여 라우터에서 고속 스위칭을 비활성화해야 합니다. 이는 트래픽이 흐르도록 해야 하는 인터페이스에 적용되어야 합니다. 명령을 사용하여 이 **show ip route** 를 확인합니다.

경고

- 많은 패킷을 처리하는 라우터에서 빠른 스위칭을 비활성화하면 CPU 사용률이 급증하여 상자가 중단되거나 피어에 대한 연결이 끊어질 수 있습니다.

- MPLS(Multi Protocol Label Switching)를 실행하는 라우터에서 고속 스위칭을 비활성화하지 마십시오. MPLS는 CEF와 함께 사용됩니다. 따라서 인터페이스에서 고속 스위칭을 비활성화하면 치명적인 영향을 미칠 수 있습니다.

이 샘플 시나리오를 고려해 보십시오.

router_122에 구성된 액세스 목록은 다음과 같습니다.

<#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

이 액세스 목록은 호스트 router_121(IP 주소 10.10.10.2)에서 호스트 router_123(IP 주소 10.1.1.1)으로 그리고 다른 방향으로 ICMP(Internet Control Message Protocol) 패킷을 허용합니다. 패킷을 어느 방향으로든 허용해야 합니다. 그렇지 않으면 라우터가 반환 ICMP 패킷을 삭제할 수 있습니다.

router_122에서 하나의 인터페이스에서만 고속 스위칭을 제거합니다. 즉, 패킷을 가로채는 Cisco IOS의 관점에서 볼 때 해당 인터페이스로 향하는 패킷의 디버그만 볼 수 있습니다. 디버그에서 이러한 패킷은 "d="로 표시됩니다. 다른 인터페이스에서 빠른 스위칭을 아직 해제하지 않았으므로 반환 패킷이 영향을 받지 않습니다 **debug ip packet** . 이 출력은 빠른 스위칭을 비활성화하는 방법을 보여줍니다.

<#root>

router_122(config)#

interface virtual-template 1

router_122(config-if)#

no ip route-cache

router_122(config-if)#

end

이제 이전에 **debug ip packet** 정의된 access-list(access-list 105)로 활성화해야 합니다.

<#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

이제 다른 인터페이스(router_122)에서 고속 스위칭을 제거합니다. 즉, 이 두 인터페이스의 모든 패킷이 이제 패킷 스위칭됩니다(의 요구 사항). **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
```

디버그 ip 패킷 출력에는 access-list 기준과 일치하지 않는 패킷이 표시되지 않습니다. 이 절차에 대한 자세한 내용은 Ping 및 Traceroute 명령 이해를 참조하십시오.

액세스 목록을 작성하는 방법에 대한 자세한 내용은 표준 IP 액세스 목록 로깅을 참조하십시오.

조건부로 트리거된 디버그

조건부 트리거 디버깅 기능이 활성화된 경우 라우터는 지정된 인터페이스에서 라우터를 시작하거나 종료하는 패킷에 대한 디버깅 메시지를 생성합니다. 라우터는 다른 인터페이스를 통해 시작하거나 종료하는 패킷에 대한 디버깅 출력을 생성하지 않습니다.

조건부 디버그의 간단한 구현을 살펴보십시오. 이 시나리오를 고려해 보십시오. 다음(trabol)에 표시된 라우터에는 HDLC 캡슐화를 실행하는 두 개의 인터페이스(serial 0 및 serial 3)가 있습니다.

모든 인터페이스에서 **debug serial interface** 수신된 HDLC 킵얼라이브를 관찰하려면 normalcommand를 사용할 수 있습니다. 두 인터페이스에서 킵얼라이브를 관찰할 수 있습니다.

<#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:
```

인터페이스 일련 번호 3에 대해 조건부 디버그를 활성화합니다. 즉, 인터페이스 serial 3에 대한 디버깅만 표시됩니다. 명령 **debug interface <interface_type interface_number >**을 사용합니다.

<#root>

traxbol#

debug interface serial 3

```
 Condition 1 set
```

조건부 디버그 **show debug condition** 가 활성 상태인지 확인하려면 명령을 사용합니다. 인터페이스 일련 번호 3의 조건은 활성 상태입니다.

<#root>

traxbol#

show debug condition

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

이제 인터페이스 serial 3에 대한 디버깅만 표시됩니다

<#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

조건부 디버그 **undebug interface <interface_type interface_number>** 를 제거하려면 이 명령을 사용합니다. 조건부 트리거를 제거하기 전에 디버그를 끄는 것이 좋습니다(예: 모두 디버그 해제 사용). 이는 조건이 제거될 때 디버그 출력이 폭주하는 것을 방지하기 위한 것이다.

<#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

이제 인터페이스 serial 0과 serial 3 모두에 대한 디버그가 표시되는지 확인할 수 있습니다.

<#root>

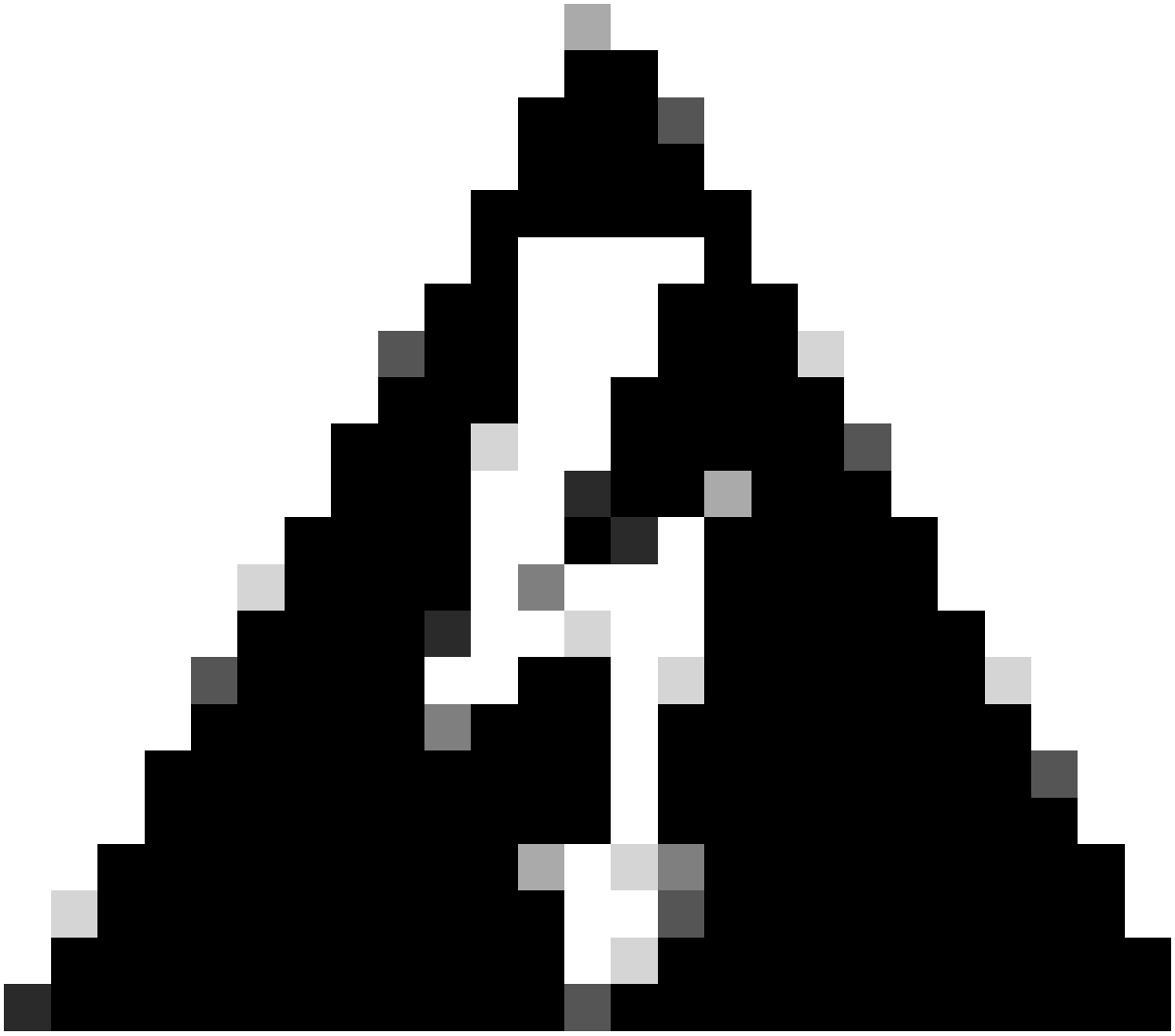
*Mar 8 09:43:34.927:

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



경고: 일부 디버깅 작업은 자체적으로 수행됩니다. 예를 들면 atm 디버깅이 있습니다. ATM 디버깅을 사용하는 경우 모든 ATM 인터페이스에서 디버그를 활성화하고 조건을 지정하기보다는 디버그를 활성화해야 할 인터페이스를 명시적으로 지정해야 합니다.

이 섹션에서는 ATM 패킷 디버깅을 하나의 하위 인터페이스로 제한하는 올바른 방법을 보여줍니다.

<#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 O

조건이 적용된 모든 **atm debugging** 인터페이스를 활성화하려고 하면 라우터에 ATM 하위 인터페이스 수가 많을 경우 라우터가 정지될 수 있습니다. atm 디버깅에 대한 잘못된 메서드의 예가 표시됩니다.

이 경우 조건이 적용되지만 효과가 없음을 알 수 있습니다. 다른 인터페이스의 패킷은 계속 볼 수 있습니다. 이 Lab 시나리오에서는 인터페이스가 두 개뿐이고 트래픽이 매우 적습니다. 인터페이스 수가 많을 경우 모든 인터페이스의 디버그 출력이 매우 높으므로 라우터가 정지될 수 있습니다.

<#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

관련 정보

- **전화 걸기 및 액세스 기술 지원**
- **Cisco 기술 지원 및 다운로드**

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.