ASR 9000에서 XR Embedded Packet Tracer 구성

목차

[소개](#)

[사전 요구 사항](#)

- [요구 사항](#)
- [사용된 구성 요소](#)

[배경 정보.](#)

- [XR 임베디드 패킷 추적기 제한 사항 및 제한 사항](#)
- [패킷 추적기 워크플로](#)

[구성](#)

- [패킷 추적기 카운터 및 조건 지우기](#)
- [패킷 추적 시작/중지](#)
- [패킷 추적기 조건](#)
- [패킷 추적기 조건 - 인터페이스](#)
- [패킷 추적기 조건 - 오프셋/값/마스크](#)

[컨피그레이션 예시:](#)

[관련 정보](#)

소개

이 문서에서는 XR Embedded Packet Tracer에 대해 설명합니다. 서비스 검증 및 문제 해결을 위해 맞춤형 패킷 흐름을 추적하는 데 도움이 됩니다.

사전 요구 사항

요구 사항

XR Embedded Packet Tracer는 Cisco IOS® XR 버전 7.1.2부터 제공되며 ASR 9000 Series에서 지원됩니다. 추가 XR 제품군은 향후 업데이트에 대한 지원을 받을 계획입니다.

사용되는 구성 요소

XR Embedded Packet Tracer는 특정 프로토콜에 독립적이며 모든 유형의 유니캐스트 및 멀티캐스트 패킷과 호환됩니다.

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.

배경 정보.

XR Embedded Packet Tracer 프레임워크는 서비스 흐름 검증 및 패킷 전달 문제의 트러블슈팅을 크게 간소화했습니다.

인터페이스에서 패킷 추적이 활성화되면 NP(Network Processor)는 수신 패킷을 평가하여 정의된 기준을 충족하는지 확인합니다. 패킷이 지정된 조건을 충족하면 내부 헤더에 식별자가 추가됩니다. 이 식별자는 라우터 내의 데이터 경로 및 punt-path와 관련된 모든 구성 요소 전반에서 패킷을 쉽게 추적할 수 있습니다.

조건은 어떤 패킷이 라우터를 통과할 때 추적될 수 있는지를 정의하는 기준 또는 규칙 집합을 가리킵니다. 이러한 조건은 시스템이 문제 해결 또는 서비스 검증을 위해 특정 패킷 흐름을 식별하고 모니터링하는 데 도움이 됩니다.

조건은 다음 구성 요소로 구성됩니다.

1. 물리적 인터페이스:

    - 패킷이 도착해야 하는 네트워크 인터페이스를 지정합니다.
    - 예: `packet-trace condition interface Gi0/0/0/1`

2. 오프셋/값/마스크 삼중:

    - 패킷 헤더(또는 페이로드)의 특정 부분을 일치시키기 위한 기준을 정의합니다.
    - 이 세 가지 요소를 통해 프로토콜 헤더의 모든 부분을 나타낼 수 있으므로 프레임워크는 프로토콜에 구애받지 않습니다.

      예:

        - Offset: 패킷 내의 위치(바이트 오프셋)입니다.
        - 가치: 해당 위치의 예상 값입니다.
        - Mask: 정밀도를 위해 일치시킬 비트.

XR 임베디드 패킷 추적기 제한 사항 및 제한 사항

XR 릴리스 7.1.2:

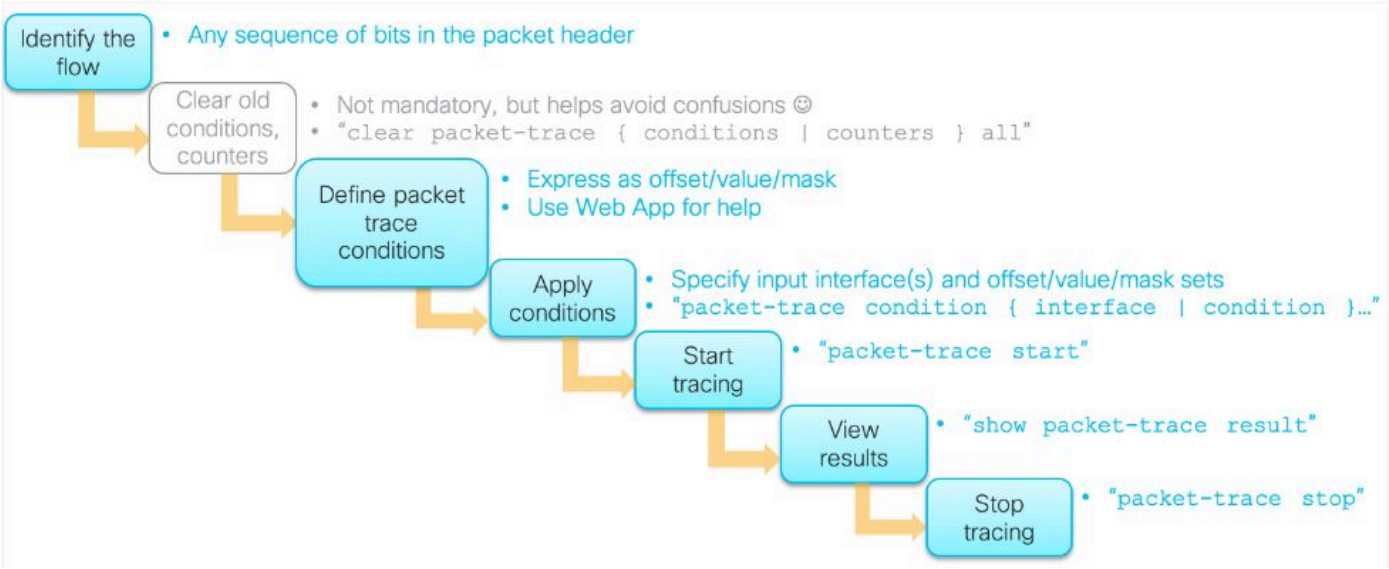
패킷 마킹은 Lightspeed Plus, Lightspeed 및 Tomahawk 라인 카드에서 지원됩니다.
패킷 추적은 앞에서 언급한 유형의 라인 카드에서 지원됩니다.
최대 3개의 4-옥텟 오프셋/값/마스크 세트를 지정할 수 있습니다.

XR 릴리스 7.5.2:

패킷 추적기는 조건이 설정된 시점에 번들 멤버를 자동으로 확인합니다
이제 SPP, NetIO, UDP, TCP의 punt 경로에서 패킷을 추적할 수 있습니다

패킷 추적기 워크플로

이 다이어그램은 패킷 추적기 워크플로의 작동 방식을 보여줍니다.



구성

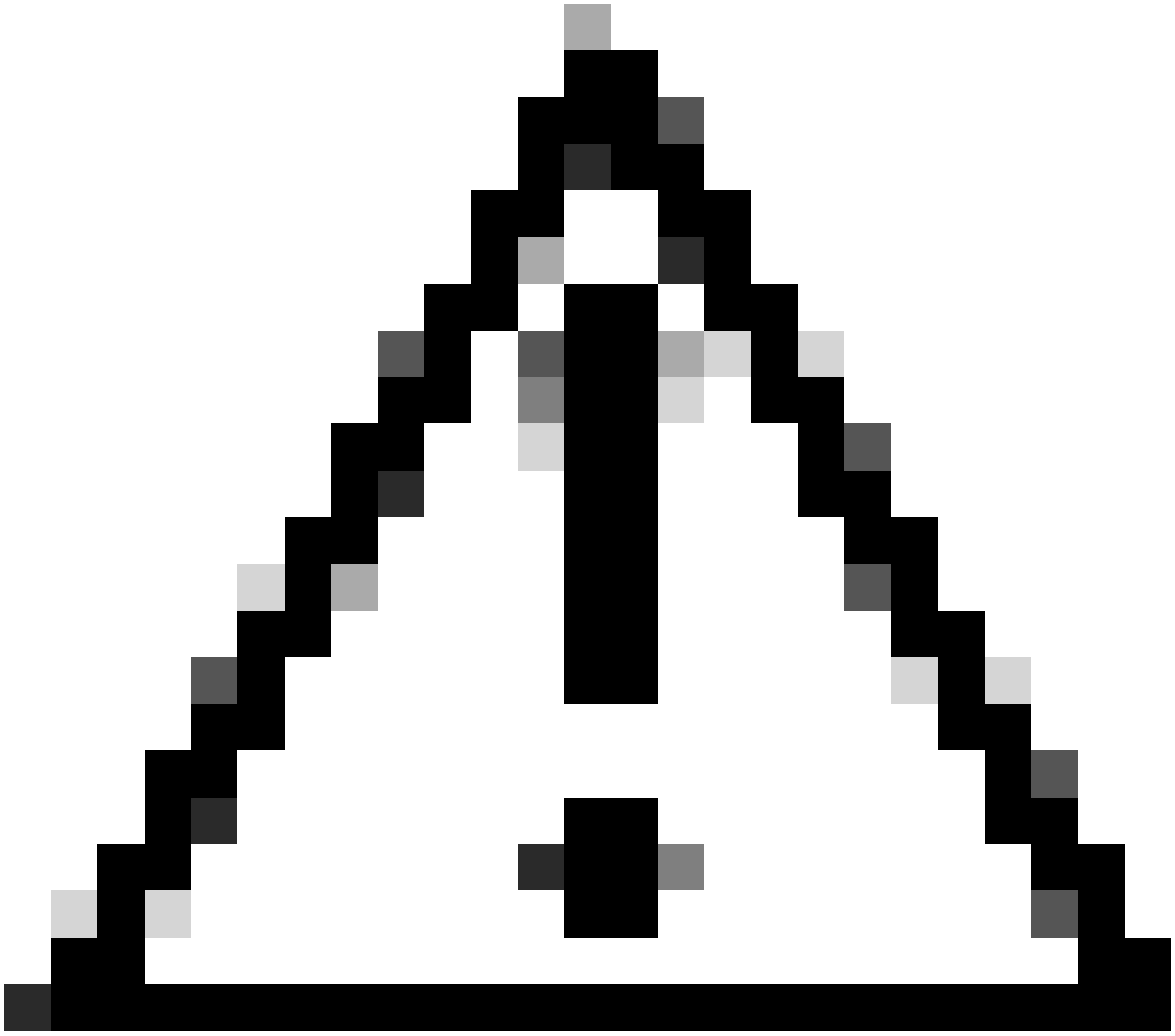
패킷 추적기 카운터 및 조건 지우기

패킷 추적기 카운터를 재설정하는 명령입니다. 패킷 추적기 카운터는 필요할 때마다 재설정될 수 있습니다.

```
clear packet-trace counters all
```

모든 패킷 추적기 조건을 제거하려면 다음 명령을 사용합니다.

```
clear packet-trace conditions all
```



주의: 패킷 추적이 비활성 상태일 때만 패킷 추적기 조건을 지울 수 있습니다.

패킷 추적 시작/중지

패킷 추적의 시작과 끝을 수동으로 지정해야 합니다.

```
RP/0/RP0/CPU0:Device#  packet-trace start
RP/0/RP0/CPU0:Device#  packet-trace stop
```

패킷 추적기 조건

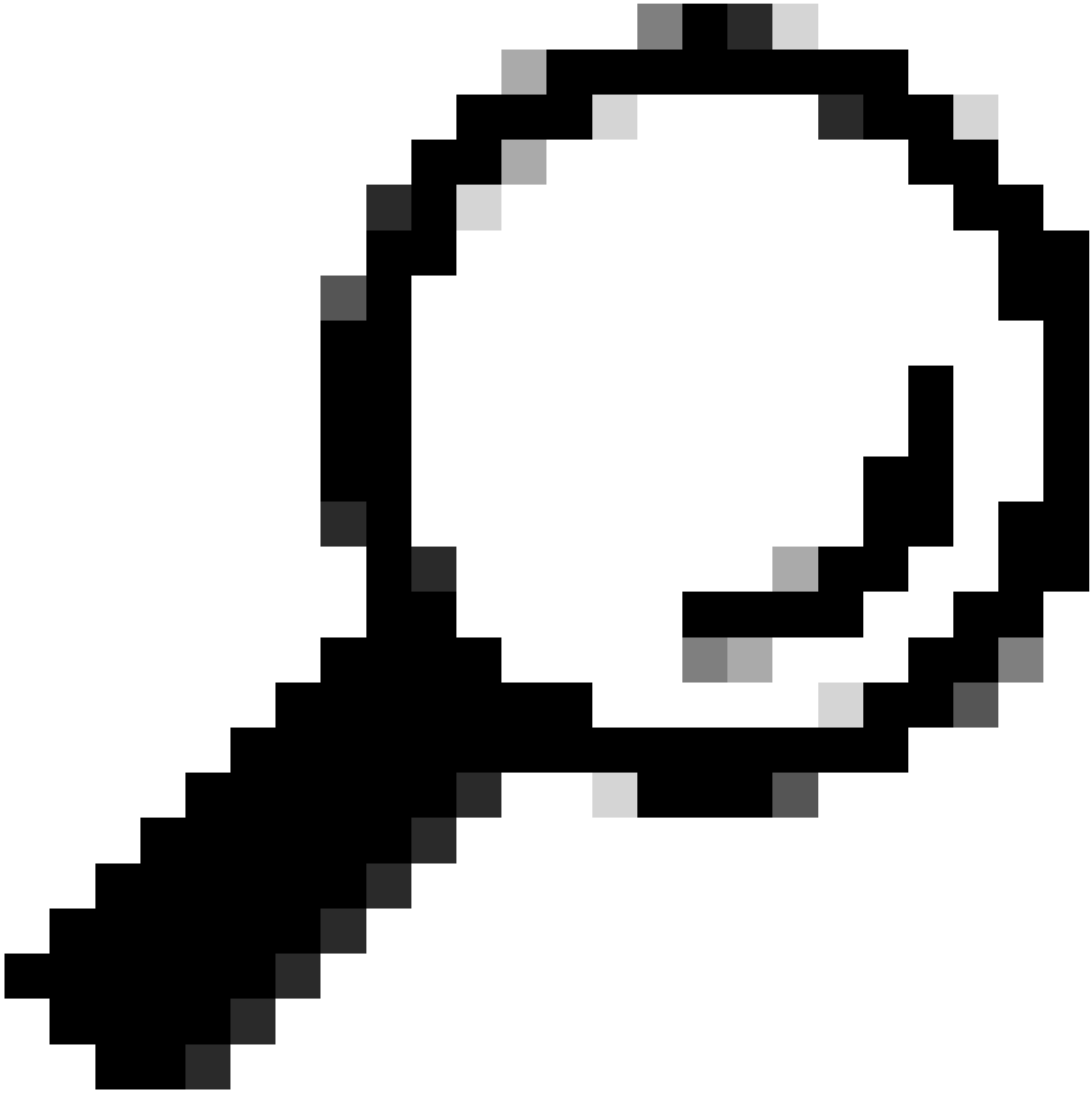
조건은 다음과 같습니다.

1. Physical Interface(s): 패킷이 수신되어야 하는 물리적 인터페이스를 나타냅니다.

2. 오프셋/값/마스크 트리플 관심 흐름을 정의하는 데 도움이 됩니다.

패킷 추적기 조건 - 인터페이스

```
RP/0/RP0/CPU0:Device#packet-trace condition interface GigE0/0/0/0
RP/0/RP0/CPU0:Device#packet-trace condition interface GigE0/0/0/1
```



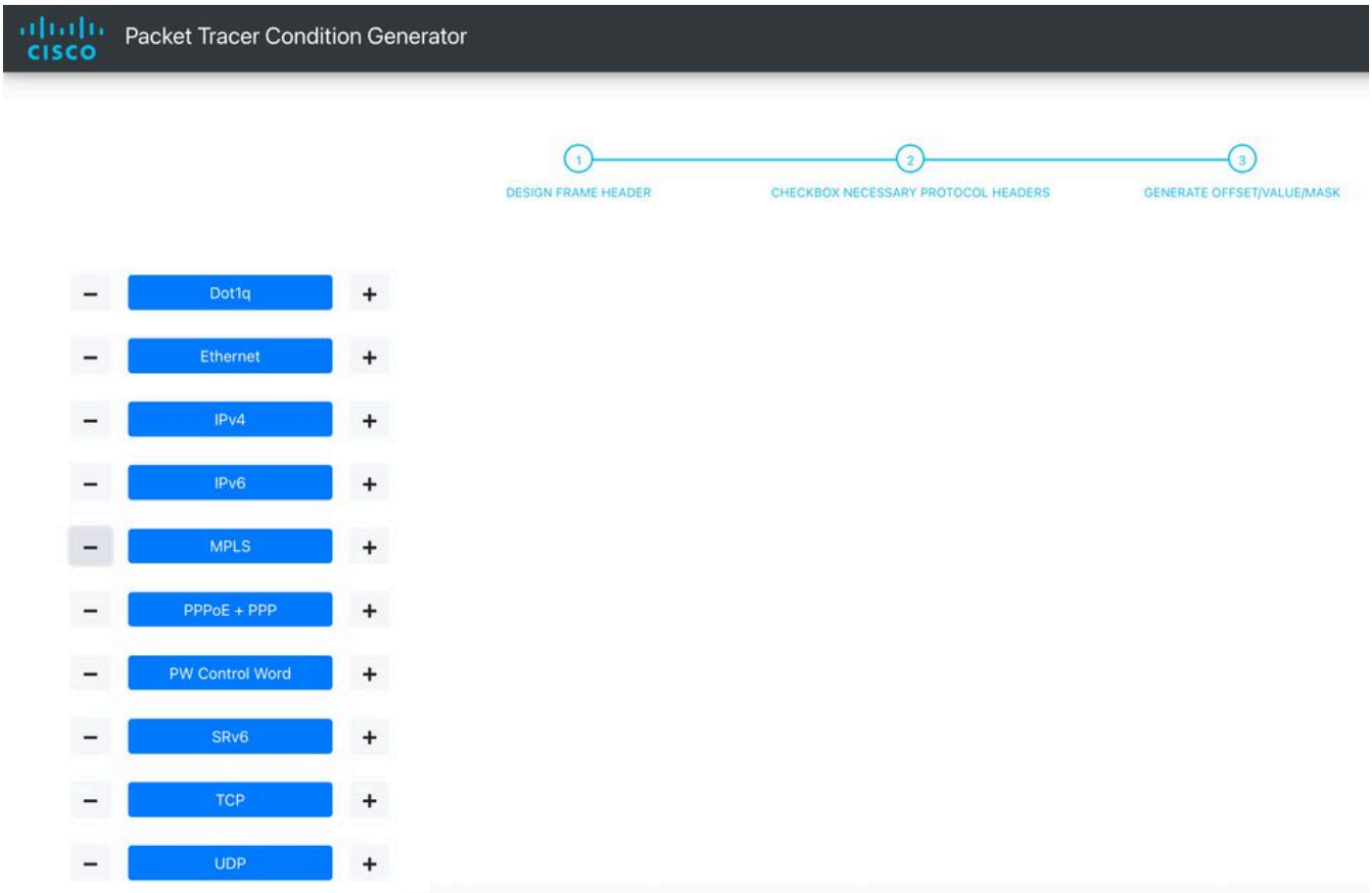
팁: 하위 인터페이스에서 추적할 때 Offset/Value/Mask 사양에서는 dot1q 또는 QinQ 캡슐화를 고려해야 합니다.

패킷 추적기 조건 - 오프셋/값/마스크

"XR Packet Tracer Condition Generator Web App"은 패킷 추적기 조건을 생성하기 위한 도구를 제공합니다.

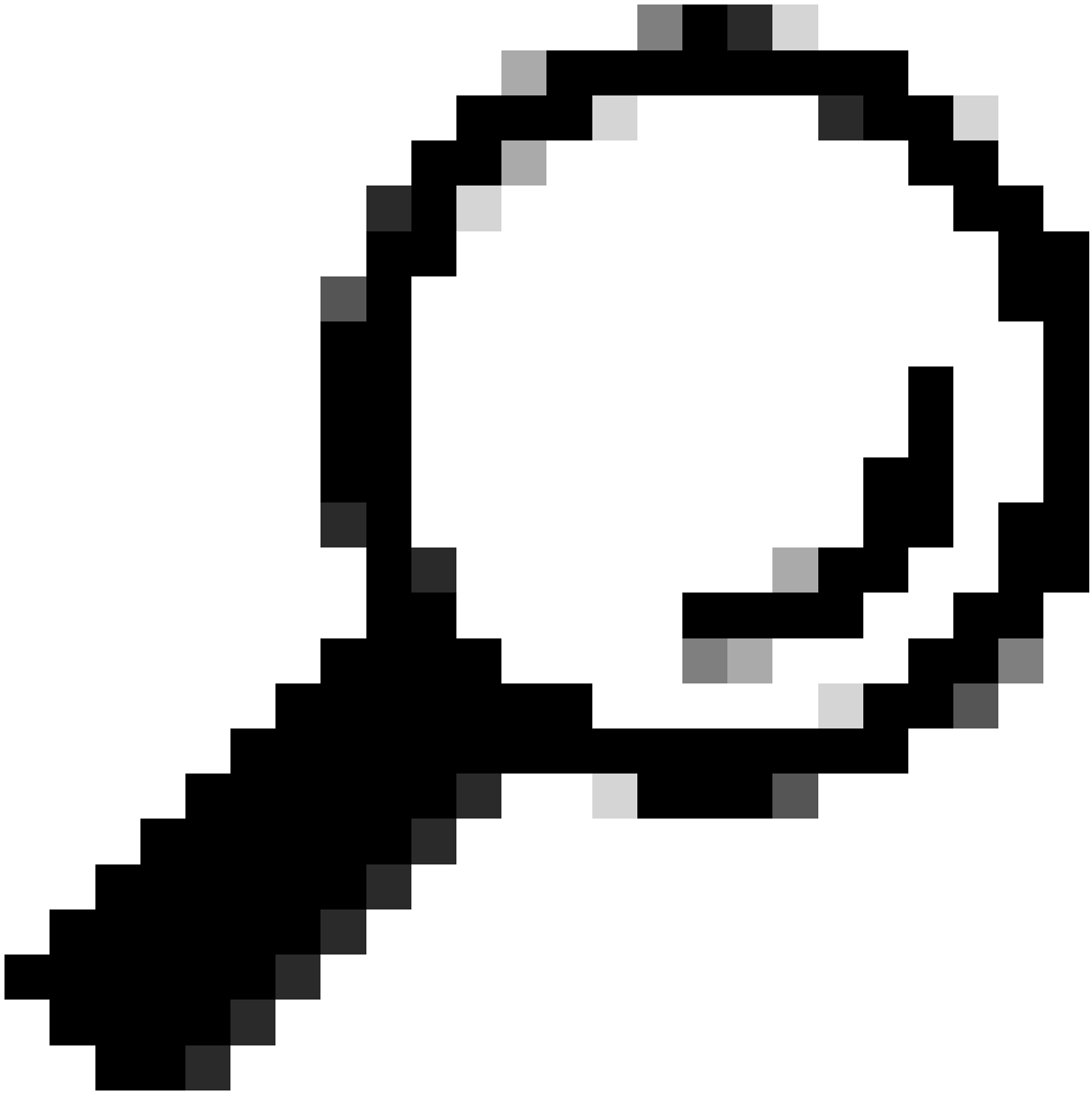
소스 코드 및 설치 지침은 GitHub에서 XR Embedded Packet Tracer - Condition Generator라는 이름으로 액세스할 수 있습니다.

이 애플리케이션을 사용하면 원하는 패킷 플로우에 대한 프로토콜 스택을 시각적으로 구성하고, 조건을 정의하기 위한 관련 레이어를 선택하고, 추적하려는 특정 플로우를 설명하는 값(옵션 마스크 사용)을 입력할 수 있습니다.



Web App의 랜딩 페이지에는 컨피그레이션에 대해 지원되는 프로토콜 헤더 목록이 표시됩니다.

오프셋 계산은 헤더의 순서에 따라 결정되므로 트래픽에 일치시키려는 헤더 앞에 필요한 모든 헤더를 포함해야 합니다.



팁: 사용 중인 PW 제어 단어 헤더를 포함해야 합니다.

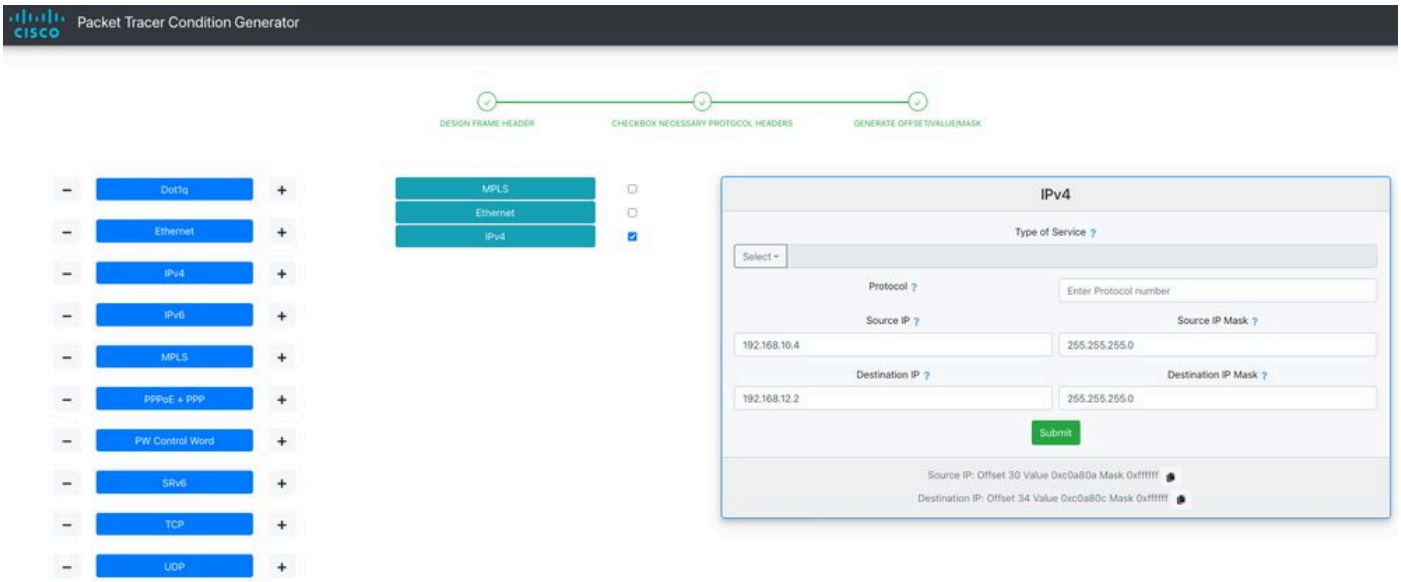
컨피그레이션 예시:

다음은 토폴로지의 예입니다. Cisco의 목적은 패킷이 XRV1 디바이스를 통해 제대로 수신 및 전송되는지 확인하는 것입니다.

1. - 모니터링할 특정 인터페이스에 대한 패킷 추적 조건을 설정합니다.

```
RP/0/RP0/CPU0:xrv-1#packet-trace condition interface Bundle-Ether1
RP/0/RP0/CPU0:xrv-1#packet-trace condition interface Bundle-Ether2
```

2.- 오프셋/값/마스크를 생성하고, 일치시킬 헤더 옆의 확인란을 선택합니다. 필요한 경우 여러 헤더를 선택할 수 있습니다. 선택된 각 헤더에 대해 해당 프레임이 오른쪽에 표시됩니다. 프레임에 원하는 값과 마스크를 입력한 다음 Submit(제출) 버튼을 클릭하여 컨피그레이션을 완료합니다.



3.- 오프셋/값/마스크가 클립보드에 복사되면 이를 사용하여 조건을 정의합니다.

```
RP/0/RP0/CPU0:xrv-1#packet-trace condition 1 Offset 30 Value 0xc0a80a Mask 0xffffff
RP/0/RP0/CPU0:xrv-1#packet-trace condition 5 Offset 34 Value 0xc0a80c Mask 0xffffff
```

4.- 패킷 추적 상태를 확인합니다.

```
RP/0/RP0/CPU0:xrv-1#show packet-trace status
---------------------------------------------------------------
Packet Trace Master Process:
  Buffered Conditions:
    Interface Bundle-Ether1
      Member GigE0/0/0/0
    Interface Bundle-Ether2
      Member GigE0/0/0/1
    1 Offset 30 Value 0xc0a80a Mask 0xffffff
    5 Offset 34 Value 0xc0a80c Mask 0xffffff
  Status: Inactive
RP/0/RP0/CPU0:xrv-1#
RP/0/RP0/CPU0:xrv-1#show packet-trace status detail
---------------------------------------------------------------
Location: 0/0/CPU0
Available Counting Modules: 4

  #1 SPP
    Last errors:
  #2 npu_server_lsp
   Last errors:
  #3 NETIO
    Last errors:
  #4 UDP
    Last errors:

Available Marking Modules: 1

  #1 npu_server_lsp
    Interfaces: 0
    Conditions: 0
    Last errors:
---------------------------------------------------------------
Packet Trace Master Process:

  Buffered Conditions:
    Interface Bundle-Ether1
      Member GigE0/0/0/0
    Interface Bundle-Ether2
      Member GigE0/0/0/1
    1 Offset 30 Value 0xc0a80a Mask 0xffffff
    5 Offset 34 Value 0xc0a80c Mask 0xffffff
  Status: Inactive
---------------------------------------------------------------
Location: 0/RP0/CPU0
Available Counting Modules: 3

  #1 SPP
    Last errors:
  #2 NETIO
    Last errors:
  #3 UDP
    Last errors:

Available Marking Modules: 0
RP/0/RP0/CPU0:xrv-1#
```

5.- 패킷 추적기를 시작합니다.

```
RP/0/RP0/CPU0:xrv-1# packet-trace start
RP/0/RP0/CPU0:xrv-1#
RP/0/RP0/CPU0:xrv-1# show packet-trace status
-------------------------------------------------------------
Packet Trace Master Process:
  Buffered Conditions:
    Interface Bundle-Ether1
      Member GigE0/0/0/0
    Interface Bundle-Ether2
      Member GigE0/0/0/1
    1 Offset 30 Value 0xc0a80a Mask 0xffffff
    5 Offset 34 Value 0xc0a80c Mask 0xffffff
  Status: Active
RP/0/RP0/CPU0:xrv-1#
```

6. - 트래픽을 캡처하기 위해 몇 분 정도 기다리겠습니다.

7.- 결과 확인:

```
RP/0/RP0/CPU0:xrv-1#show packet-trace result
T: D - Drop counter; P - Pass counter
Location     | Source       | Counter                    | T | Last-Attribute
-------------   ------------   -------------------------   -   ----------------------------------------
0/0/CPU0       NP0            PACKET_MARKED                P      GigE0_0_0_0
0/0/CPU0       NP0            PACKET_TO_FABRIC             P
0/0/CPU0       NP0            PACKET_TO_PUNT               P
0/0/CPU0       NP0            PACKET_FROM_FABRIC           P
0/0/CPU0       NP0            PACKET_TO_INTERFACE          P      GigE0_0_0_1
RP/0/RP0/CPU0:xrv-1#
```

8.- show packet-trace description 명령을 사용하여 패킷 추적기 프레임워크에 등록된 모든 카운터를 해당 설명과 함께 확인할 수 있습니다.

```
RP/0/RP0/CPU0:xrv-1#show packet-trace descriptions
NP0                PACKET_MARKED                M   Marked from ingress interface
NP0                PACKET_FROM_INJECT           P   Injected from linecard CPU
NP0                PACKET_FROM_FAB_INJECT       P   Injected from fabric
NP0                PACKET_ING_DROP              D   Dropped on ingress
NP0                PACKET_TO_FABRIC             P   Sent to router fabric
NP0                PACKET_TO_PUNT               P   Punted to linecard for CPU handling
NP0                PACKET_FROM_FABRIC           P   From router fabric
NP0                PACKET_EGR_DROP              D   Dropped on egress
NP0                PACKET_TO_INTERFACE          P   Packet sent to network interface
```

```
RP/0/RP0/CPU0:xrv-1#
```

9.- 패킷 추적을 중지합니다.

```
RP/0/RP0/CPU0:xrv-1#packet-trace stop
```

관련 정보

XR 임베디드 패킷 추적기

Cisco 기술 지원 및 다운로드

ASR 9000 Series 라인 카드 유형 이해

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.