Red Hat Enterprise Linux 7

네트워킹 가이드

RHEL 7에서 네트워크, 네트워크 인터페이스 및 네트워크 서비스 구성 및 관리

Last Updated: 2024-10-04

Red Hat Enterprise Linux 7 네트워킹 가이드

RHEL 7에서 네트워크, 네트워크 인터페이스 및 네트워크 서비스 구성 및 관리

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

법적 공지

초록

Red Hat Enterprise Linux 7 네트워킹 가이드는 Red Hat Enterprise Linux의 네트워크 인터페이스, 네트워크 및 네트워크 서비스의 구성 및 관리와 관련된 정보를 문서화합니다. Linux 및 네트워킹에 대한 기본적인 지식을 갖춘 시스템 관리자를 대상으로 합니다.

차례

I 부. 시작하기 전

이 설명서 부분에서는 Red Hat Enterprise Linux에서 네트워크 서비스의 기본 개념에 대한 개요를 제공합니다.

1장. 네트워킹 주제 개요

1.1. IP와 IP가 아닌 네트워크 비교

네트워크는 파일, 프린터, 애플리케이션 및 인터넷 연결과 같은 공유 정보와 리소스를 공유할 수 있는 상호 연결된 장치의 시스템입니다. 이러한 각 장치에는 프로토콜이라는 규칙 집합을 사용하여 둘 이상의 장치 간에 메시지를 보내고 받을 고유한 인터넷 프로토콜(IP) 주소가 있습니다.

네트워크 통신 카테고리

IP 네트워크

인터넷 프로토콜 주소를 통해 통신하는 네트워크입니다. IP 네트워크는 인터넷과 대부분의 내부 네트워크에서 구현됩니다. 이더넷, Cable Modems, DSL Modems, 전화 업 모뎀, 무선 네트워크 및 VPN 연결은 일반적인 예입니다.

IP 외 네트워크

전송 계층이 아니라 하위 계층을 통해 통신하는 데 사용되는 네트워크입니다. 이러한 네트워크는 거의 사용되지 않습니다. InfiniBand는 13장. *InfiniBand 및 RDMA 네트워크 구성* 에 설명된 IP가 아닌 네트워크입니다.

1.2. 정적 IP 주소와 동적 IP 주소 비교

고정 IP 주소 지정

장치에 정적 IP 주소가 할당되면 수동으로 변경하지 않는 한 시간이 지남에 따라 주소가 변경되지 않습니다. 필요한 경우 고정 **IP** 주소를 사용하는 것이 좋습니다.

- **DNS** 및 인증 서버 등의 서버에 대한 네트워크 주소 일관성을 보장하려면 다음을 수행합니다.
- 다른 네트워크 인프라와 독립적으로 작동하는 대역 외 관리 장치를 사용합니다.

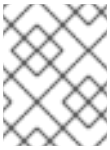
3.1절. "네트워크 구성 방법 선택" 에 나열된 모든 설정 도구를 사용하면 정적 **IP** 주소를 수동으로 할당할 수 있습니다. **nmcli** 도구는 3.3.8절. "nmcli를 사용하여 정적 이더넷 연결 추가 및 구성" 에 설명되어 있습니다.

자동화된 구성 및 관리에 대한 자세한 내용은 *Red Hat Enterprise Linux 7 시스템 관리자 가이드의 OpenLMI 장을 참조하십시오*. *Red Hat Enterprise Linux 7 설치 가이드에서는* 네트워크 설정 할당을 자동화하는 데에도 사용할 수 있는 **Kickstart** 파일 사용에 대해 설명합니다.

동적 IP 주소 지정

장치에 동적 IP 주소가 할당되면 시간이 지남에 따라 주소가 변경됩니다. 따라서 시스템을 재부팅한 후 IP 주소가 변경될 수 있으므로 경우에 따라 네트워크에 연결하는 장치에 사용하는 것이 좋습니다.

동적 IP 주소는 보다 유연하고 설정 및 관리가 용이합니다. DHCP(*동적 호스트 제어 프로토콜*)는 호스트에 네트워크 구성을 동적으로 할당하는 기존의 방법입니다. 자세한 내용은 14.1절. "DHCP를 사용하는 이유는 무엇입니까?"을 참조하십시오. 3.3.7절. "nmcli를 사용하여 동적 이더넷 연결 추가 및 구성" 에 설명된 **nmcli** 도구를 사용할 수도 있습니다.



참고

정적 또는 동적 IP 주소를 사용할 시기를 정의하는 엄격한 규칙은 없습니다. 사용자의 요구 사항, 기본 설정 및 네트워크 환경에 따라 다릅니다.

기본적으로 NetworkManager 는 **DHCP** 클라이언트 **dhclient** 를 호출합니다.

1.3. DHCP 클라이언트 동작 구성

DHCP(Dynamic Host Configuration Protocol) 클라이언트는 클라이언트가 네트워크에 연결할 때마다 DHCP 서버에서 동적 IP 주소와 해당 구성 정보를 요청합니다.

NetworkManager 는 기본적으로 **DHCP** 클라이언트 **dhclient** 를 호출합니다.

IP 주소 요청

DHCP 연결이 시작되면 dhcp 클라이언트에서 **DHCP** 서버의 IP 주소를 요청합니다. dhcp 클라이언트에서 이 요청이 완료될 때까지 기다리는 시간은 기본적으로 60초입니다. **nmcli** 툴 또는 **/etc/sysconfig/network -scripts/ifcfg-*ifname*파일에서 IPV4_DHCP_TIMEOUT 옵션을 사용하여 ipv4.dhcp- timeout** 속성을 구성할 수 있습니다. 예를 들어 **nmcli**: 사용

```
~]# nmcli connection modify enp1s0 ipv4.dhcp-timeout 10
```

이 간격 동안 주소를 가져올 수 없는 경우 IPv4 구성이 실패합니다. 전체 연결도 실패할 수 있으며 **ipv4.may-fail** 속성에 따라 달라집니다.

- **ipv4.may-fail** 가 **yes** (기본값)로 설정된 경우 연결 상태는 IPv6 구성에 따라 다릅니다.

  1. IPv6 구성이 활성화되고 성공하면 연결이 활성화되지만 IPv4 구성은 다시 시도할 수 없습니다.

  2. IPv6 구성을 비활성화하거나 구성하지 않으면 연결에 실패합니다.

- **ipv4.may-fail** 가 **no** 로 설정된 경우 연결이 비활성화됩니다. 이 경우 다음을 수행합니다.

  1. 연결의 자동 연결 속성이 활성화되면 NetworkManager 는 **autoconnect -retries** 속성에 설정된 만큼 연결을 활성화하려고 시도합니다. 기본값은 4입니다.

  2. 연결이 여전히 dhcp 주소를 가져올 수 없는 경우 자동 활성화에 실패합니다.

     5분 후에 자동 연결 프로세스가 다시 시작되고 dhcp 클라이언트에서 dhcp 서버에서 주소를 가져오려고 시도합니다.

갱신 요청

dhcp 주소를 획득하고 IP 주소 리스를 갱신할 수 없는 경우 dhcp 클라이언트는 2분마다 3분마다 다시 시작하여 dhcp 서버에서 리스를 가져오려고 합니다. 매번 리스를 가져오도록 **ipv4.dhcp-timeout** 속성을 초(기본값은 60)로 설정하여 구성됩니다. 시도 중에 응답을 받으면 프로세스가 중지되고 리스를 갱신합니다.

세 번의 시도가 실패한 후 다음을 수행하십시오.

- **ipv4.may-fail** 가 **yes** (기본값)로 설정되어 있고 IPv6가 성공적으로 구성된 경우 연결이 활성화되고 dhcp 클라이언트가 2분마다 다시 시작됩니다.

- **ipv4.may-fail** 가 **no** 로 설정되면 연결이 비활성화됩니다. 이 경우 연결에 **자동 연결 속성이 활성화되어** 있으면 연결이 처음부터 활성화됩니다.

1.3.1. DHCPv4 영구 생성

시작 시 및 리스 갱신 프로세스 중 DHCPv4를 영구적으로 만들려면 **ipv4.dhcp-timeout** 속성을 32비트 정수(MAXINT32)의 최대값으로 설정하려면 **2147483647** 또는 **infinity** 값으로 설정합니다.

```
~]$ nmcli connection modify enps1s0 ipv4.dhcp-timeout infinity
```

결과적으로 **NetworkManager** 는 성공할 때까지 DHCP 서버에서 리스를 가져오거나 갱신하는 것을 중지하지 않습니다.

리스 갱신 프로세스 중에만 DHCP 지속적인 동작을 보장하기 위해 **/etc/sysconfig/network-scripts/ifcfg-*enp1s0*** 구성 파일의 **IPADDR** 속성에 정적 IP를 수동으로 추가하거나 **nmcli**:

```
~]$ nmcli connection modify enp1s0 ipv4.address 192.168.122.88/24
```

IP 주소 리스가 만료되면 고정 IP는 IP 상태를 구성 또는 부분적으로 구성한 대로 보존합니다(IP 주소가 있지만 인터넷에 연결되어 있지 않음) 2분마다 dhcp 클라이언트가 재시작되도록 합니다.

1.4. 무선 규제 도메인 설정

Red Hat Enterprise Linux의 crda 패키지에는 지정된 관할지에 대한 무선 규제 규칙을 커널에 제공하는 중앙 규제 도메인 에이전트가 포함되어 있습니다. 이는 특정 udev 스크립트에 사용되며 udev 스크립트를 디버깅하지 않는 한 수동으로 실행하면 안 됩니다 . 커널은 새 규정 도메인 변경 시 udev 이벤트를 보내 crda 를 실행합니다. 규제 도메인 변경은 Linux 무선 하위 시스템(IEEE-802.11)에 의해 트리거됩니다. 이 하위 시스템은 **regulatory.bin** 파일을 사용하여 규정 데이터베이스 정보를 유지합니다.

setregdomain 유틸리티는 시스템의 규정 도메인을 설정합니다. **Setregdomain** 은 인수를 사용하지 않으며 일반적으로 관리자가 수동으로하지 않고 udev 와 같은 시스템 스크립트를 통해 호출됩니다. 국가 코드 조회에 실패하면 시스템 관리자는 **/etc/sysconfig/regdomain** 파일에 **COUNTRY** 환경 변수를 정의할 수 있습니다.

규제 도메인에 대한 자세한 내용은 다음 도움말 페이지를 참조하십시오.

- **setregdomain(1) 도움말** 페이지 - 국가 코드를 기반으로 규제 도메인을 설정합니다.
- **CRDA(8)** 도움말 페이지 - 지정된 ISO 또는 IEC 3166 alpha2에 대한 무선 규제 도메인을 커널에 전송합니다.
- **regulatory.bin(5) 도움말** 페이지 - Linux 무선 규제 데이터베이스를 보여줍니다.
- **IW(8)** 도움말 페이지 - 무선 장치와 해당 구성을 표시하거나 조작합니다.

1.5. 넷 콘솔 구성

디스크 로깅에 실패하거나 직렬 콘솔을 사용할 수 없는 경우 커널 디버깅을 사용해야 할 수 있습니다. **netconsole** 커널 모듈을 사용하면 네트워크를 통해 커널 메시지를 다른 컴퓨터에 기록할 수 있습니다.

netconsole 을 사용하려면 네트워크에 올바르게 구성된 **rsyslog** 서버가 있어야 합니다.

절차 1.1. netconsole에 대한 rsyslog 서버 구성

1. **/etc/rsyslog.conf 파일의 MODULES 섹션에서 514/udp 포트에서 수신 대기하고 네트워크에서 메시지를 받도록 rsyslogd 데몬을 구성합니다.**

   ```
   $ModLoad imudp
   $UDPServerRun 514
   ```

2. 변경 사항을 적용하려면 **rsyslogd** 서비스를 다시 시작하십시오.

```
]# systemctl restart rsyslog
```

3. **rsyslogd** 가 514/udp 포트에서 수신 대기 중인지 확인합니다.

```
]# netstat -l | grep syslog
udp        0      0 0.0.0.0:syslog          0.0.0.0:*
udp6       0      0 [::]:syslog             [::]:*
```

netstat -l 출력의 **0.0.0.0 :syslog 및 [::]:syslog** 값은 **rsyslogd** 가 **/etc/services** 파일에 정의된 기본 **netconsole** 포트에서 수신 대기함을 의미합니다.

```
]$ cat /etc/services | grep syslog
syslog          514/udp
syslog-conn     601/tcp              # Reliable Syslog Service
syslog-conn     601/udp              # Reliable Syslog Service
syslog-tls      6514/tcp             # Syslog over TLS
syslog-tls      6514/udp             # Syslog over TLS
syslog-tls      6514/dccp            # Syslog over TLS
```

Netconsole 은 initscripts 패키지의 일부인 **/etc/sysconfig/netconsole** 파일을 사용하여 구성됩니다. 이 패키지는 기본적으로 설치되며 **netconsole** 서비스도 제공합니다.

전송 머신을 구성하려면 다음 절차를 따르십시오.

절차 1.2. 전송 머신 구성

1. **syslogd** 서버의 IP 주소와 일치하도록 **/etc/sysconfig/netconsole** 파일의 **SYSLOGADDR** 변수 값을 설정합니다. 예를 들어 다음과 같습니다.

```
SYSLOGADDR=192.168.0.1
```

2. 변경 사항을 적용하려면 **netconsole** 서비스를 다시 시작하십시오.

```
]# systemctl restart netconsole.service
```

3. 시스템을 재부팅한 후 **netconsole.service** 가 실행되도록 활성화합니다.

```
]# systemctl enable netconsole.service
```

4. **/var/log/messages** 파일(기본값) 또는 **rsyslog.conf** 에 지정된 파일에서 클라이언트의 **순 console** 메시지를 확인합니다.

```
]# cat /var/log/messages
```

참고

기본적으로 **rsyslogd** 및 **netconsole.service** 는 포트 514를 사용합니다. 다른 포트를 사용하려면 **/etc/rsyslog.conf** 의 다음 행을 필요한 포트 번호로 변경합니다.

```
$UDPServerRun <PORT>
```

전송 시스템에서 **/etc/sysconfig/netconsole** 파일에서 다음 행의 주석을 제거하고 편집합니다.

```
SYSLOGPORT=514
```

넷콘솔 구성 및 문제 해결에 대한 자세한 내용은 Netconsole 커널 설명서를 참조하십시오.

1.6. SYSCTL로 네트워크 커널 튜닝 가능 항목 사용

sysctl 유틸리티를 통해 특정 커널 튜닝 가능 항목을 사용하여 실행 중인 시스템에서 네트워크 구성을 조정하고 네트워킹 성능에 직접 영향을 미칩니다.

네트워크 설정을 변경하려면 **sysctl** 명령을 사용합니다. 시스템을 다시 시작할 때마다 지속되는 영구 변경 사항은 **/etc/sysctl.conf** 파일에 행을 추가합니다.

사용 가능한 모든 **sysctl** 매개변수 목록을 표시하려면 **root** 로 를 입력합니다.

```
~]# sysctl -a
```

sysctl 을 사용하는 네트워크 커널 튜닝 가능 항목에 대한 자세한 내용은 시스템 관리자 가이드의 여러 인터페이스에서 PTP 사용 섹션을 참조하십시오.

네트워크 커널 튜닝 가능 항목에 대한 자세한 내용은 커널 관리 가이드 의 네트워크 인터페이스 튜닝 가능 항목 을 참조하십시오.

1.7. NCAT 유틸리티를 사용하여 데이터 관리

ncat 네트워킹 유틸리티는 **Red Hat Enterprise Linux 7의 net**cat을 대체합니다. **ncat** 은 다른 애플리케이션 및 사용자에게 네트워크 연결을 제공하는 안정적인 백엔드 도구입니다. 명령줄에서 네트워크를 읽고 쓰고, TCP(Transmission Control Protocol), UDP(User Datagram Protocol), SCTP(Stream Control Transmission Protocol) 또는 통신에 Unix 소켓을 사용합니다. **ncat** 은 **IPv4 및 IPv 6**, 개방형 연결을 처리하고, 패킷을 전송하며, **SSL**, 연결 브로커와 같은 상위 수준의 기능을 지원할 수 있습니다.

동일한 옵션을 사용하여 Thenc **명령을 ncat** 으로 입력할 수도 있습니다. **ncat** 옵션에 대한 자세한 내용은 *마이그레이션 계획 가이드 및 ncat(1) 도움말 페이지의 새로운 네트워킹 유틸리티 (ncat) 섹션* 을 참조하십시오.

ncat 설치

ncat 패키지를 설치하려면 **루트로** 입력하십시오 :

```
~]# yum install ncat
```

ncat 사용 사례의 간략한 선택

예 1.1. 클라이언트와 서버 간 통신 활성화

1. TCP 포트 *8080* 에서 연결을 수신 대기하도록 클라이언트 시스템을 설정합니다:

```
~]$ ncat -l 8080
```

2. 서버 머신에서 클라이언트의 IP 주소를 지정하고 동일한 포트 번호를 사용합니다.

```
~]$ ncat 10.0.11.60 8080
```

연결 양쪽에 메시지를 보내면 로컬 및 원격 시스템에 모두 표시될 수 있습니다.

3. **Ctrl+D** 를 눌러 TCP 연결을 종료합니다.

참고

UDP 포트를 확인하려면 **-u** 옵션과 함께 same **nc** 명령을 사용합니다. 예를 들어 다음과 같습니다.

```
~]$ ncat -u -l 8080
```

예 1.2. 파일 전송

이전 예제에서 언급한 대로 화면에 정보를 출력하는 대신 모든 정보를 파일에 보낼 수 있습니다. 예를 들어 클라이언트에서 서버로 TCP 포트 *8080* 을 통해 파일을 보내려면 다음을 수행합니다.

1. 클라이언트 시스템에서 서버 시스템으로 파일을 전송하는 특정 포트를 수신하려면 다음을 수행합니다.

```
~]$  ncat -l 8080 > outputfile
```

2. 서버 시스템에서 클라이언트의 IP 주소, 포트 및 전송할 파일을 지정합니다.

```
~]$  ncat -l 10.0.11.60 8080 < inputfile
```

파일이 전송되면 연결이 자동으로 닫힙니다.

참고

다른 방향으로도 파일을 전송할 수 있습니다.

```
~]$  ncat -l 8080 < inputfile
```

```
~]$  ncat -l 10.0.11.60 8080 > outputfile
```

예 1.3. HTTP 프록시 서버 생성

localhost 포트 *8080* 에서 HTTP 프록시 서버를 생성하려면 다음을 수행합니다.

```
~]$  ncat -l --proxy-type http localhost 8080
```

예 1.4. 포트 검사

열려 있는 포트를 보려면 **-z** 옵션을 사용하고 검사할 포트 범위를 지정합니다.

```
~]$  ncat -z 10.0.11.60 80-90
   Connection to 192.168.0.1 80 port [tcp/http] succeeded!
```

예 1.5. SSL을 사용하여 보안 클라이언트-서버 통신 설정

서버에서 **SSL** 을 설정합니다.

```
~]$ ncat -e /bin/bash -k -l 8080 --ssl
```

클라이언트 시스템에서 다음을 수행합니다.

```
~]$ ncat --ssl 10.0.11.60 8080
```

참고

SSL 연결의 진정한 기밀성을 보장하려면 서버에 **--ssl-cert** 및 **--ssl-key** 옵션이 필요하며 클라이언트에는 **--ssl-verify** 및 **--ssl-trustfile** 옵션이 필요합니다. **OpenSSL** 에 대한 자세한 내용은 *보안 가이드의 OpenSSL 사용 섹션을 참조하십시오*.

자세한 내용은 *ncat(1)* 도움말 페이지를 참조하십시오.

II 부. IP 네트워킹 관리

이 설명서 부분에서는 Red Hat Enterprise Linux에서 네트워킹을 설정하고 관리하는 방법에 대한 자세한 지침을 제공합니다.

2장. NETWORKMANAGER 시작하기

2.1. NETWORKMANAGER 개요

Red Hat Enterprise Linux 7에서 기본 네트워킹 서비스는 **NetworkManager** 에 의해 제공되며, 이는 네트워크 장치와 연결을 사용할 수 있을 때 활성화되는 동적 네트워크 제어 및 구성 데몬입니다. 기존의 **ifcfg** 유형 구성 파일은 계속 지원됩니다. 자세한 내용은 2.6절. "네트워크 스크립트로 NetworkManager 사용"을 참조하십시오.

2.1.1. NetworkManager 사용의 이점

NetworkManager를 사용할 때의 주요 이점은 다음과 같습니다.

- 네트워크 관리의 용이성: **NetworkManager** 는 네트워크 연결이 작동하는지 확인합니다. 시스템에 네트워크 구성이 없지만 네트워크 장치가 있음을 감지하면 **NetworkManager** 에서 연결을 제공하기 위해 임시 연결을 생성합니다.

- 사용자에게 손쉬운 연결 설정 제공: **NetworkManager** 는 **GUI, nmtui, nmcli** - 등의 다양한 도구를 통해 관리를 제공합니다. 2.5절. "NetworkManager 도구" 의 내용을 참조하십시오.

- 구성 유연성 지원. 예를 들어 **NetworkManager** 가 사용 가능한 와이파이 네트워크를 스캔하고 표시합니다. 인터페이스를 선택할 수 있으며 **NetworkManager** 는 재부팅 프로세스 후 자동으로 연결을 제공하는 필수 자격 증명을 표시합니다. **NetworkManager** 는 네트워크 별칭, IP 주소, 정적 경로, DNS 정보 및 VPN 연결과 많은 연결별 매개 변수를 구성할 수 있습니다. 요구 사항을 반영하도록 구성 옵션을 수정할 수 있습니다.

- 애플리케이션이 네트워크 구성 및 상태를 쿼리하고 제어할 수 있도록 하는 D-Bus를 통해 API 제공. 이러한 방식으로 애플리케이션은 D-BUS를 통해 네트워킹을 확인하거나 구성할 수 있습니다. 예를 들어 **웹 브라우저를 통해 서버를 모니터링하고 구성하는 웹 콘솔** 인터페이스는 **NetworkManager** D-BUS 인터페이스를 사용하여 네트워킹을 구성합니다.

- 재부팅 프로세스 후 장치 상태를 유지하고 다시 시작하는 동안 관리 모드로 설정된 인터페이스를 인수합니다.

- 명시적으로 설정되지 않지만 사용자 또는 다른 네트워크 서비스에 의해 수동으로 제어되는 장치 처리.

2.2. NETWORKMANAGER 설치

NetworkManager 는 기본적으로 Red Hat Enterprise Linux에 설치됩니다. 그렇지 않은 경우 **루트로** 를 입력합니다. :

```
~]# yum install NetworkManager
```

사용자 권한 및 권한 취득에 대한 정보는 *Red Hat Enterprise Linux 시스템 관리자 가이드를 참조하십시오* .

2.3. NETWORKMANAGER 상태 확인

NetworkManager 가 실행 중인지 확인하려면 다음을 수행하십시오.

```
~]$ systemctl status NetworkManager
NetworkManager.service - Network Manager
   Loaded: loaded (/lib/systemd/system/NetworkManager.service; enabled)
```

```
Active: active (running) since Fri, 08 Mar 2013 12:50:04 +0100; 3 days ago
```

NetworkManager 가 실행되지 않는 경우 **systemctl status** 명령은 **Active: inactive(dead)** 를 표시합니다.

2.4. NETWORKMANAGER 시작

NetworkManager 를 시작하려면 다음을 수행합니다.

```
~]# systemctl start NetworkManager
```

부팅 시 NetworkManager 를 자동으로 활성화하려면 다음을 수행합니다.

```
~]# systemctl enable NetworkManager
```

서비스 시작, 중지 및 관리에 대한 자세한 내용은 *Red Hat Enterprise Linux 시스템 관리자 가이드를 참조하십시오*.

2.5. NETWORKMANAGER 도구

표 2.1. NetworkManager 도구 및 애플리케이션 요약

애플리케이션 또는 도구	Description
nmcli	사용자와 스크립트가 NetworkManager 와 상호 작용할 수 있도록 하는 명령줄 도구입니다. NetworkManager 의 모든 측면을 제어하기 위해 서버 등의 GUI가 없는 시스템에서 nmcli 를 사용할 수 있습니다. GUI 도구와 동일한 기능을 합니다.
nmtui	NetworkManager를 위한 간단한 TUI(텍스트 사용자 인터페이스)
nm-connection-editor	본딩 구성 및 티밍 연결 등의 **제어 센터** 유틸리티에서 아직 처리되지 않은 특정 작업에 대한 그래픽 사용자 인터페이스 도구입니다. NetworkManager 에서 저장한 네트워크 연결을 추가, 제거 및 수정할 수 있습니다. 이를 시작하려면 터미널에 nm-connection-editor 를 입력합니다. `~]$ nm-connection-editor`
control-center	데스크탑 사용자가 사용할 수 있는 GNOME 쉘에서 제공하는 그래픽 사용자 인터페이스 도구. 네트워크 설정 도구를 통합합니다. 시작하려면 **Super** 키를 눌러 Activities Overview(활동 개요)를 입력하고 **Network** (네트워크)를 입력한 다음 **Enter** 를 누릅니다. 네트워크 설정 도구가 나타납니다.
네트워크 연결 아이콘	NetworkManager 에서 보고한 네트워크 연결 상태를 나타내는 GNOME Shell 에서 제공하는 그래픽 사용자 인터페이스 도구입니다. 아이콘에는 현재 사용 중인 연결 유형에 대한 시각적 표시로 사용되는 여러 상태가 있습니다.

2.6. 네트워크 스크립트로 NETWORKMANAGER 사용

이 섹션에서는 스크립트를 실행하는 방법과 네트워크 스크립트에서 사용자 지정 명령을 사용하는 방법에 대해 설명합니다.

네트워크 스크립트 라는 용어는 **/etc/init.d/network** 스크립트와 호출되는 기타 설치된 스크립트를 나타냅니다. **NetworkManager** 는 기본 네트워킹 서비스를 제공하지만 스크립트와 **NetworkManager** 는 병렬로 실행되어 함께 작동할 수 있습니다. Red Hat은 먼저 이를 테스트할 것을 권장합니다.

네트워크 스크립트 실행 중

systemctl 명령으로 **만** 네트워크 스크립트를 실행합니다.

```
systemctl start|stop|restart|status network
```

systemctl 유틸리티는 기존 환경 변수를 지우고 올바른 실행을 보장합니다.

Red Hat Enterprise Linux 7에서 **NetworkManager** 가 먼저 시작되고 **/etc/init.d/network** 는 **NetworkManager** 의 연결을 변경하지 않도록 **NetworkManager** 를 확인합니다. **NetworkManager** 는 sysconfig 구성 파일을 사용하여 기본 애플리케이션으로 설정되었으며 **/etc/init.d/network** 는 보조를 사용하도록 설계되었습니다.

/etc/init.d/network 스크립트는 다음을 실행합니다.

1. 수동 - **systemctl** 명령의 **start|stop|restart 네트워크** 사용

   또는

2. **systemctl enable network 명령** 의 결과로 network 서비스가 활성화된 경우 부팅 및 종료.

이 프로세스는 수동 프로세스이며 부팅 후 발생하는 이벤트에 반응하지 않습니다. 사용자는 **ifup 및 if down** 스크립트를 수동으로 호출할 수도 있습니다.

참고

systemctl reload network.service 명령은 initscripts의 기술적 제한으로 인해 작동하지 않습니다. 네트워크 서비스에 대한 새 구성을 적용하려면 **restart** 명령을 사용합니다.

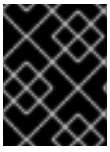
```
~]# systemctl restart network.service
```

이렇게 하면 해제되고 모든 NIC(네트워크 인터페이스 카드)가 작동되어 새 구성을 로드합니다. 자세한 내용은 네트워크 서비스에 대한 Red Hat Knowledgebase 솔루션 다시 로드 및 강제 로드 옵션을 참조하십시오.

네트워크 스크립트에서 사용자 지정 명령 사용

/sbin/ifup-local,ifdown- pre-local 및 **ifdown-local** 스크립트의 사용자 지정 명령은 이러한 장치가 **/etc/init.d/network** 서비스에 의해 제어되는 경우에만 실행됩니다. **ifup-local** 파일은 기본적으로 존재하지 않습니다. 필요한 경우 **/sbin/** 디렉터리에 생성합니다.

ifup-local 스크립트는 **NetworkManager** 가 아니라 initscripts에서만 읽을 수 있습니다. **NetworkManager** 를 사용하여 사용자 지정 스크립트를 실행하려면 **디스패치.d/** 디렉터리에 만듭니다. "Dispatcher 스크립트 실행" 의 내용을 참조하십시오.



중요

initscripts 패키지 또는 관련 rpm에 포함된 파일을 수정하는 것은 권장되지 않습니다. 사용자가 이러한 파일을 수정하는 경우 Red Hat은 지원을 제공하지 않습니다.

사용자 지정 작업은 이전 네트워크 **스크립트와 NetworkManager 를 사용하여 네트워크** 연결이 가동되거나 다운될 때 실행될 수 있습니다. **NetworkManager** 가 활성화되어 있는 경우 **ifup** 및 **ifdown** 스크립트는 **NetworkManager** 에서 해당 인터페이스 관리 여부를 묻습니다. 이 인터페이스는 **NetworkManager** 에서 **ifcfg** 파일의 "DEVICE=" 행에서 찾을 수 있습니다.

NetworkManager 에서 관리하는 장치 :

ifup 호출

ifup 을 호출하고 장치를 **NetworkManager** 에서 **관리하면** 다음 두 가지 옵션이 있습니다.

- 장치가 **아직 연결되지 않은** 경우 **ifup** 에서 **NetworkManager** 에 연결을 시작하도록 요청합니다.
- 장치가 이미 **연결되어** 있으면 할 일이 없습니다.

ifdown 호출

ifdown 을 호출하고 **NetworkManager** 에 의해 장치를 **관리** 할 때 :

- **ifdown** 은 **NetworkManager** 에 연결을 종료하도록 요청합니다.

NetworkManager 에 의해 관리되지 않는 장치 :

ifup 또는 if down 을 호출하면 스크립트는 **NetworkManager** 가 존재하기 전까지 사용한 이전의 비 NetworkManager 메커니즘을 사용하여 연결을 시작합니다.

Dispatcher 스크립트 실행

NetworkManager 는 연결 상태에 따라 추가 사용자 지정 스크립트를 실행하여 서비스를 시작하거나 중지하는 방법을 제공합니다. 기본적으로 **/etc/NetworkManager/dispatcher.d/** 디렉터리가 존재하며 **NetworkManager** 는 알파벳순으로 스크립트를 실행합니다. 각 스크립트는 **root** 가 **소유한** 실행 파일이어야 하며 파일 소유자에 대한 **쓰기 권한만** 있어야 합니다. NetworkManager 디스패처 스크립트 실행에 대한 자세한 내용은 Red Hat Knowledgebase 솔루션에서 NetworkManager 디스패처 스크립트를 작성하여 ethtool 명령을 적용하는 방법을 참조하십시오.

2.7. SYSCONFIG 파일로 NETWORKMANAGER 사용

/etc/sysconfig/ 디렉토리는 구성 파일 및 스크립트의 위치입니다. 대부분의 네트워크 구성 정보는 VPN, 모바일 광대역 및 PPPoE 구성을 제외하고 **/etc/NetworkManager/** 하위 디렉토리에 저장됩니다. 예를 들어 인터페이스 관련 정보는 **/etc/sysconfig/network-scripts/** 디렉터리의 **ifcfg** 파일에 저장됩니다.

전역 설정의 경우 **/etc/sysconfig/network** 파일을 사용하십시오. VPN에 대한 정보, 모바일 광대역 및 PPPoE 연결은 **/etc/NetworkManager/system-connections/** 에 저장됩니다.

Red Hat Enterprise Linux 7에서 **ifcfg** 파일을 편집하는 경우 **NetworkManager** 는 변경 사항을 자동으로 인식하지 못하므로 변경 사항을 알리는 메시지가 표시됩니다. **NetworkManager 프로필 설정을 업데이트하는 툴을 사용하는 경우 NetworkManager** 는 해당 프로필을 사용하여 다시 연결할 때까지 이러한 변경 사항을 구현하지 **않습니다**. 예를 들어 편집기를 사용하여 구성 파일을 변경한 경우 **NetworkManager** 에서 구성 파일을 다시 읽어야 합니다.

이를 확인하려면 **root** 로 를 입력하여 모든 연결 프로필을 다시 로드합니다.

```
~]# nmcli connection reload
```

또는 **변경된 파일만** 다시 로드하려면 **ifcfg-*ifname***:

```
~]# nmcli con load /etc/sysconfig/network-scripts/ifcfg-ifname
```

위의 명령을 사용하여 여러 파일 이름을 지정할 수 있습니다.

nmcli 와 같은 도구를 사용하여 변경한 내용은 다시 로드할 필요가 없지만 연결된 인터페이스를 중단한 다음 다시 실행해야 합니다.

```
~]# nmcli dev disconnect interface-name
```

```
~]# nmcli con up interface-name
```

nmcli 에 대한 자세한 내용은 3.3절. "nmcli로 IP 네트워킹 구성" 를 참조하십시오.

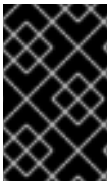
NetworkManager 는 **ifup** 명령이 사용되는 경우 네트워크 스크립트가 실행 중인 경우 **NetworkManager** 를 트리거하려고 시도하지만 네트워크 스크립트를 트리거하지 않습니다. 네트워크 스크립트에 대한 설명은 2.6절. "네트워크 스크립트로 NetworkManager 사용" 을 참조하십시오.

ifup 스크립트는 몇 가지 작업을 수행한 다음 ifup- ***device_name*, ifup- wireless,ifup- ppp** 등과 같은 인터페이스 관련 스크립트를 호출하는 일반 스크립트입니다. 사용자가 **ifup enp1s0을 수동으로 실행하는 경우**:

1. **ifup** 은 **/etc/sysconfig/network-scripts/ifcfg-enp1s0**;이라는 파일을 찾습니다.
2. **ifcfg** 파일이 있는 경우 **ifup** 은 해당 파일에서 **TYPE** 키를 찾아 호출할 유형별 스크립트를 결정합니다.
3. **만약up** 이 **TYPE** 을 기반으로 **ifup-wireless** 또는 **ifup-*device_name*** 을 호출하는 경우
4. 유형별 스크립트는 유형별 설정 수행
5. 유형별 스크립트를 사용하면 일반 함수에서 **DHCP** 또는 정적 설정과 같은 관련 작업을 **IP**를 수행할 수 있습니다.

부팅 시 **/etc/init.d/network** 는 모든 **ifcfg** 파일을 읽고 **ONBOOT=yes** 가 있는 각 파일에 대해 **NetworkManager** 가 이미 **ifcfg** 파일에서 DEVICE를 시작하고 있는지 확인합니다. **NetworkManager** 가 해당 장치를 시작하거나 이미 시작한 경우 해당 파일에 대해 더 이상 수행되지 않으며 다음 **ONBOOT=yes** 파일이 확인됩니다. **NetworkManager** 가 아직 해당 장치를 시작하지 않는 경우 initscripts는 기존 동작을 계속하고 **ifcfg** 파일에 대해 **ifup** 을 호출합니다.

그 결과 **ONBOOT=yes** 가 있는 모든 **ifcfg** 파일은 **NetworkManager** 또는 initscripts에 의해 시스템 부팅 시 시작될 것으로 예상됩니다. 이렇게 하면 **NetworkManager** 가 처리하지 않는 일부 레거시 네트워크 유형(예: ISDN 또는 아날로그 전화 표시 모뎀) 외에도 NetworkManager에서 아직 지원하지 않는 새 애플리케이션은 **NetworkManager** 에서 처리할 **수 없는** 경우에도 initscripts에서 올바르게 시작합니다.



중요

스크립트가 **ifcfg-*** 를 문자 그대로 수행하므로 **/etc** 디렉토리 내의 위치에 백업 파일을 저장하거나 라이브 파일과 동일한 위치에 저장하지 않는 것이 좋습니다. 이러한 확장 기능만 **.old,. orig,.rpmnew,.rpmorig, .rpmsave** 입니다.

sysconfig 파일 사용에 대한 자세한 내용은 3.5절. "ifcfg 파일을 사용하여 IP 네트워킹 구성" 및 *ifcfg(8)* 도움말 페이지를 참조하십시오.

2.8. 추가 리소스

- **man(1) 도움말** 페이지 - 도움말 페이지와 찾기 방법을 설명합니다.
- **NetworkManager(8)** 도움말 페이지 - 네트워크 관리 데몬을 설명합니다.
- **NetworkManager.conf(5) 도움말** 페이지 - **NetworkManager** 구성 파일을 설명합니다.
- **/usr/share/doc/initscripts-*버전*/sysconfig.txt** - 레거시 네트워크 서비스에서 이해한 대로 **ifcfg** 구성 파일과 해당 지시문을 설명합니다.
- **/usr/share/doc/initscripts-*버전*/examples/networking/** - 예제 구성 파일이 포함된 디렉터리입니다.
- **ifcfg(8)** 도움말 페이지 - **ifcfg** 명령을 간단히 설명합니다.

3장. IP 네트워킹 구성

시스템 관리자는 **NetworkManager** 를 사용하여 네트워크 인터페이스를 구성할 수 있습니다.

3.1. 네트워크 구성 방법 선택

- **NetworkManager** 를 사용하여 네트워크 인터페이스를 구성하려면 다음 도구 중 하나를 사용합니다.
  - 텍스트 사용자 인터페이스 도구 **nmtui**. 자세한 내용은 3.2절. "nmtui로 IP 네트워킹 구성" 의 내용을 참조하십시오.
  - 명령줄 도구 **nmcli**. 자세한 내용은 3.3절. "nmcli로 IP 네트워킹 구성" 의 내용을 참조하십시오.
  - 그래픽 사용자 인터페이스 도구, **GNOME GUI**. 자세한 내용은 3.4절. " GNOME GUI를 사용하여 IP 네트워킹 구성 " 의 내용을 참조하십시오.
- **NetworkManager** 를 **사용하지 않고** 네트워크 인터페이스를 구성하려면 다음을 수행합니다.
  - **ifcfg** 파일을 수동으로 편집합니다. 자세한 내용은 3.5절. "ifcfg 파일을 사용하여 IP 네트워킹 구성" 의 내용을 참조하십시오.
  - **ip** 명령을 사용합니다. 이 명령을 사용하여 인터페이스에 IP 주소를 할당할 수 있지만, 재부팅 시 변경 사항이 지속되지 않습니다. 재부팅 시 변경 사항이 손실됩니다. 자세한 내용은 3.6절. "ip 명령을 사용하여 IP 네트워킹 구성" 의 내용을 참조하십시오.
- 루트 파일 시스템이 로컬이 **아닌** 경우 네트워크 설정을 구성하려면 다음을 수행합니다.
  - 커널 명령줄 사용. 자세한 내용은 3.7절. "커널 명령줄에서 IP 네트워킹 구성" 의 내용을 참조하십시오.

3.2. NMTUI로 IP 네트워킹 구성

시스템 관리자는 NetworkManager의 도구 **nmtui** 를 사용하여 네트워크 인터페이스를 구성할 수 있습니다. 2.5절. "NetworkManager 도구" 의 내용을 참조하십시오.

이 절차에서는 텍스트 사용자 인터페이스 도구 **nmtui** 를 사용하여 네트워킹을 구성하는 방법을 설명합니다.

사전 요구 사항

- **nmtui** 툴은 터미널 창에서 사용됩니다. NetworkManager-tui 패키지에 포함되어 있지만 기본적으로 **NetworkManager** 와 함께 설치되지 않습니다. NetworkManager-tui 를 설치하려면 :

```
~]# yum install NetworkManager-tui
```

- **NetworkManager** 가 실행 중인지 확인하려면 2.3절. "NetworkManager 상태 확인" 을 참조하십시오.

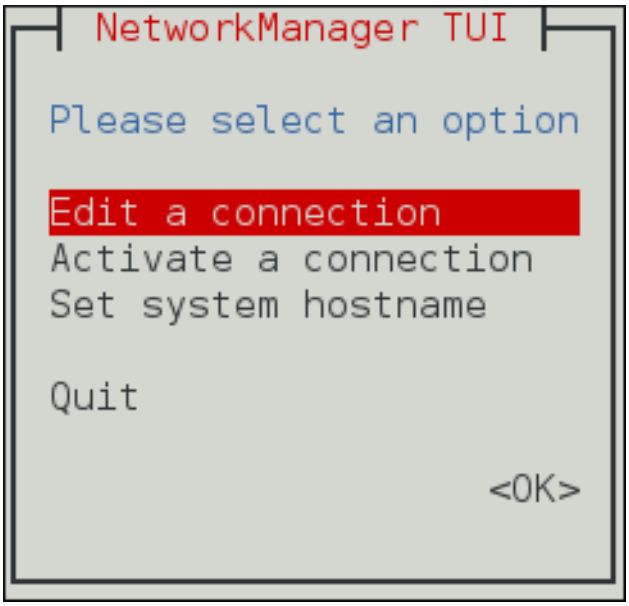
절차

1. **nmtui** 도구를 시작합니다.

```
~]$ nmtui
```

텍스트 사용자 인터페이스가 나타납니다.

그림 3.1. NetworkManager 텍스트 사용자 인터페이스 시작 메뉴



[D]

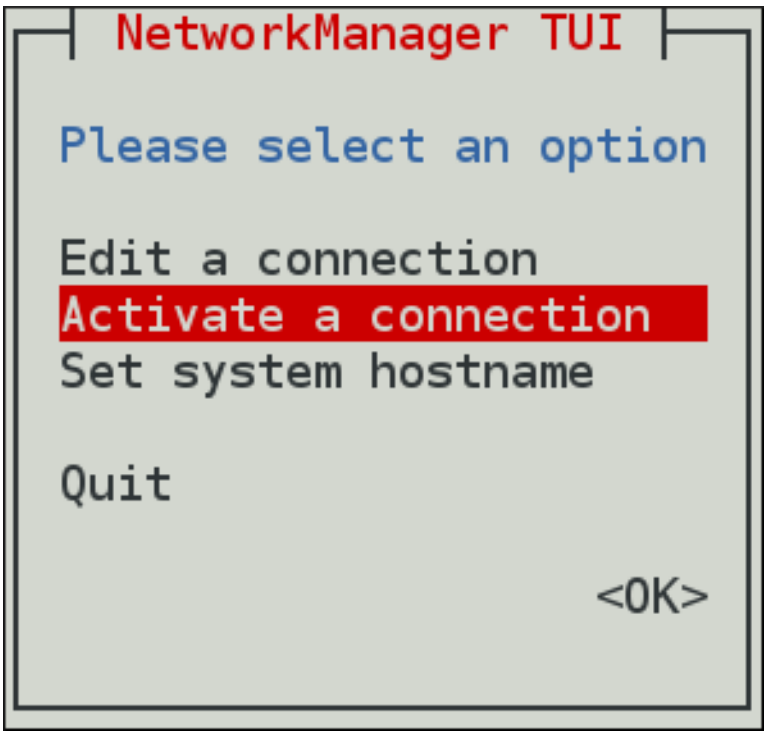
2. 탐색하려면 화살표 키를 사용하거나 **탭을** 눌러 앞으로 이동하고 **Shift+Tab** 을 눌러 옵션을 다시 이동합니다. **Enter** 를 눌러 옵션을 선택합니다. **Space** 표시줄에서 확인란의 상태를 전환합니다.

이미 활성 상태인 수정된 연결 후 변경 사항을 적용하려면 연결을 다시 실행해야 합니다. 이 경우 다음 절차를 따르십시오.

절차

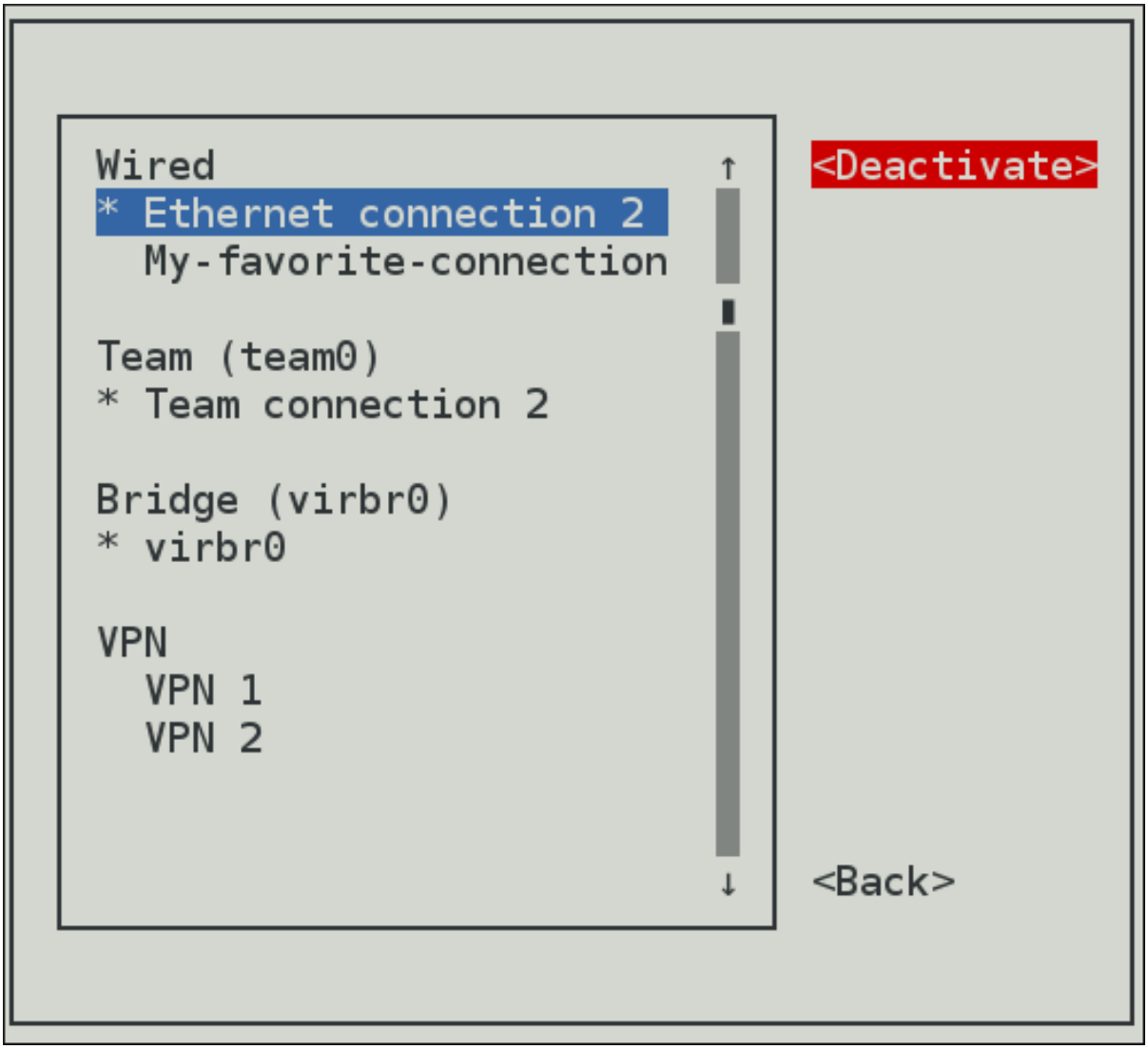
1. **Activate a connection** (연결 메뉴 활성화) 항목을 선택합니다.

그림 3.2. 연결 활성화



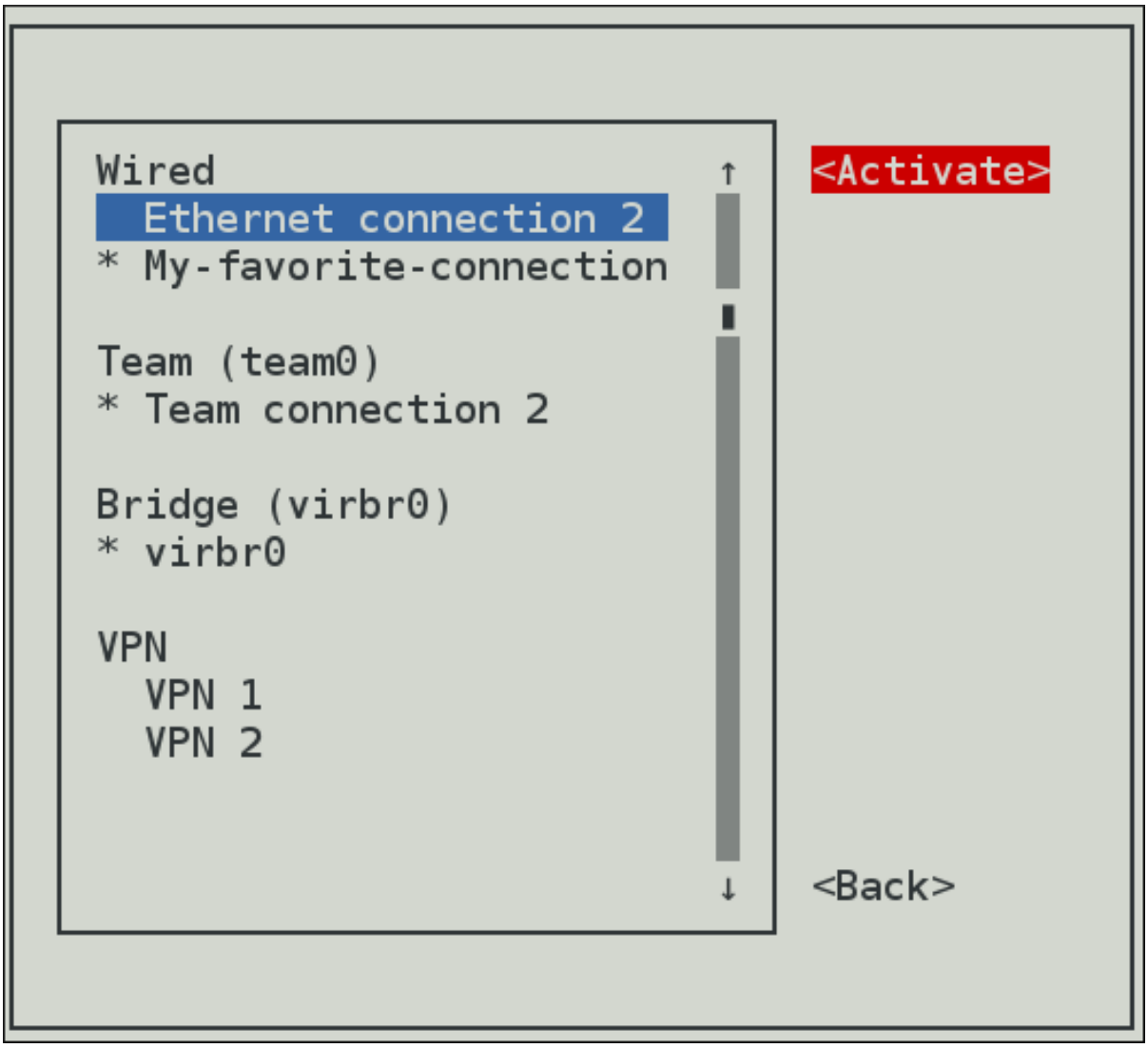
2. 수정된 연결을 선택합니다. 오른쪽에서 **Deactivate** (비활성화) 버튼을 클릭합니다.

그림 3.3. 수정된 연결 비활성화



3. 연결을 다시 선택하고 **Activate** (활성화) 버튼을 클릭합니다.

그림 3.4. 수정된 연결 다시 활성화



다음 명령도 사용할 수 있습니다.

- **nmtui edit *connection-name***

  연결 이름을 지정하지 않으면 선택 메뉴가 표시됩니다. 연결 이름을 제공하고 올바르게 식별하면 관련 **Edit connection(연결 편집)** 화면이 표시됩니다.

- **nmtui connect *connection-name***

  연결 이름을 지정하지 않으면 선택 메뉴가 표시됩니다. 연결 이름을 제공하고 올바르게 식별하면 관련 연결이 활성화됩니다. 잘못된 명령은 사용 메시지를 인쇄합니다.

nmtui 가 모든 유형의 연결을 지원하지는 않습니다. 특히 WPA Enterprise를 사용하여 VPN, 무선 네트워크 연결 또는 **802.1X** 를 사용하여 이더넷 연결을 편집할 수 없습니다.

3.3. NMCLI로 IP 네트워킹 구성

nmcli (NetworkManager 명령줄 인터페이스) 명령줄 유틸리티는 NetworkManager를 제어하고 네트워크 상태를 보고하는 데 사용됩니다. **nm-applet** 또는 기타 그래픽 클라이언트의 대체 방법으로 활용할 수 있습니다. 2.5절. “NetworkManager 도구” **nmcli** 는 네트워크 연결을 생성, 표시, 편집, 삭제, 활성화 및 비활성화하고 네트워크 장치 상태를 제어 및 표시하는 데 사용됩니다.

nmcli 유틸리티는 **NetworkManager** 를 제어하는 데 사용자와 스크립트가 모두 사용할 수 있습니다:

- 서버, 헤드리스 시스템 및 터미널의 경우 **nmcli** 를 사용하여 네트워크 연결 생성, 편집, 시작 및 중지, 네트워크 상태 보기 등 GUI 없이 직접 **NetworkManager** 를 제어할 수 있습니다.
- 스크립트의 경우 **nmcli** 는 스크립트 처리에 더 적합한 terse 출력 형식을 지원합니다. 네트워크 연결을 수동으로 관리하는 대신 네트워크 구성을 통합하는 방법입니다.

nmcli 명령의 기본 형식은 다음과 같습니다.

```
nmcli [OPTIONS] OBJECT { COMMAND | help }
```

. 여기서 OBJECT는 **일반,네트워킹,라디오,연결,장치,에이전트**, **모니터** 중 하나일 수 있습니다. 명령에 이러한 옵션의 접두사를 사용할 수 있습니다. 예를 들어 **nmcli con help,nmcli c help,nmcli connection help 가 동일한 출력을 생성하는 데 도움이 됩니다**.

몇 가지 유용한 선택적 옵션은 다음과 같습니다.

-t, terse

이 모드는 값만 표시하는 terse 출력을 볼 수 있으므로 컴퓨터 스크립트 처리에 사용할 수 있습니다.

예 3.1. 일련의 출력 보기

```
nmcli -t device
ens3:ethernet:connected:Profile 1
lo:loopback:unmanaged:
```

-f, 필드

이 옵션은 출력에 표시할 수 있는 필드를 지정합니다. 예를 들면 NAME,UUID,TYPE,AUTOCONNECT,ACTIVE,DEVICE,STATE입니다. 하나 이상의 필드를 사용할 수 있습니다. 더 많이 사용하려면 쉼표 뒤에 공백을 사용하여 필드를 구분하지 마십시오.

예 3.2. 출력에서 필드 지정

```
~]$ nmcli -f DEVICE,TYPE device
DEVICE  TYPE
ens3    ethernet
lo      loopback
```

또는 스크립팅에 더 나은:

```
~]$ nmcli -t -f DEVICE,TYPE device
ens3:ethernet
lo:loopback
```

-p, 예

이 옵션을 사용하면 **nmcli** 가 사람이 읽을 수 있는 출력을 생성합니다. 예를 들어, 값이 정렬되고 헤더가 인쇄됩니다.

예 3.3. 상당히 모드에서 출력 보기

```
nmcli -p device
==========================
  Status of devices
==========================
DEVICE  TYPE      STATE      CONNECTION
------------------------------------------------------------------
ens3    ethernet  connected  Profile 1
lo      loopback  unmanaged  --
```

-h, help

도움말 정보를 출력합니다.

nmcli 도구에는 몇 가지 컨텍스트에 민감한 도움말이 있습니다.

nmcli 도움말

이 명령은 후속 명령에서 사용할 수 있는 옵션과 개체 이름을 나열합니다.

nmcli *오브젝트* 도움말

이 명령은 지정된 오브젝트와 관련된 사용 가능한 작업 목록을 표시합니다. 예를 들면 다음과 같습니다.

```
nmcli c help
```

3.3.1. nmcli 선택 예

예 3.4. NetworkManager의 전체 상태 확인

```
~]$ nmcli general status
STATE      CONNECTIVITY  WIFI-HW  WIFI     WWAN-HW  WWAN
connected  full          enabled  enabled  enabled  enabled
```

3차 모드에서는 다음을 수행합니다.

```
~]$ nmcli -t -f STATE general
connected
```

예 3.5. NetworkManager 로깅 상태 보기

```
~]$ nmcli general logging
  LEVEL  DOMAINS
  INFO   PLATFORM,RFKILL,ETHER,WIFI,BT,MB,DHCP4,DHCP6,PPP,WIFI_SCAN,IP4,IP6,A
UTOIP4,DNS,VPN,SHARING,SUPPLICANT,AGENTS,SETTINGS,SUSPEND,CORE,DEVICE,OL
PC,
WIMAX,INFINIBAND,FIREWALL,ADSL,BOND,VLAN,BRIDGE,DBUS_PROPS,TEAM,CONCHECK
,DC
B,DISPATCH
```

예 3.6. 모든 연결 보기

```
~]$ nmcli connection show
  NAME       UUID                                  TYPE      DEVICE
Profile 1  db1060e9-c164-476f-b2b5-caec62dc1b05  ethernet  ens3
ens3       aaf6eb56-73e5-4746-9037-eed42caa8a65  ethernet  --
```

예 3.7. 현재 활성화된 연결만 보기

```
~]$ nmcli connection show --active
  NAME       UUID                                  TYPE      DEVICE
Profile 1  db1060e9-c164-476f-b2b5-caec62dc1b05  ethernet   ens3
```

예 3.8. NetworkManager 및 해당 상태에서 인식되는 장치만 보기

```
~]$ nmcli device status
DEVICE  TYPE      STATE      CONNECTION
ens3    ethernet  connected  Profile 1
lo      loopback  unmanaged  --
```

nmcli 명령의 다음 약자를 사용할 수도 있습니다.

표 3.1. 일부 nmcli 명령의 약어

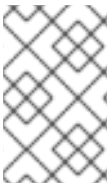
nmcli 명령	약어
nmcli 일반 상태	nmcli g
nmcli 일반 로깅	nmcli g 로그
nmcli 연결 표시	nmcli con show
nmcli 연결 show --active	nmcli con show -a
nmcli 장치 상태	nmcli dev

자세한 내용은 *nmcli-examples(5)* 도움말 페이지를 참조하십시오.

3.3.2. nmcli를 사용하여 네트워크 인터페이스 시작 및 중지

nmcli 툴을 사용하여 컨트롤러를 포함한 네트워크 인터페이스를 시작하고 중지할 수 있습니다. 예를 들어 다음과 같습니다.

```
nmcli con up id bond0
nmcli con up id port0
nmcli dev disconnect bond0
nmcli dev disconnect ens3
```



참고

nmcli connection down 명령은 장치가 추가 자동 활성화되지 않도록 장치의 연결을 비활성화합니다. **nmcli device disconnect** 명령은 장치의 연결을 끊고 장치가 수동 조작 없이 추가 연결을 자동으로 활성화하지 못하도록 합니다.

3.3.3. nmcli 옵션 이해

다음은 몇 가지 중요한 **nmcli** 속성 옵션입니다. *nmcli(1)* 도움말 페이지에 있는 전체 목록을 참조하십시오:

connection.type

연결 유형입니다. 허용되는 값은 adl, bond, bond-slave, bridge-slave, bluetooth, cdma, 이더넷, gsm, infiniband, olpc-mesh, 팀, 팀-슬레이브, vlan, 와이파이, wimax입니다. 각 연결 유형에는 유형별 명령 옵션이 있습니다. **TYPE_SPECIFIC_OPTIONS** 목록은 *nmcli(1)* 도움말 페이지에 있습니다. 예를 들어 다음과 같습니다.

- **gsm** 연결을 사용하려면 **apn** 에 지정된 액세스 지점 이름이 필요합니다.

```
nmcli c add connection.type gsm apn access_point_name
```

- **와이파이** 장치를 사용하려면 **ssid** 에 지정된 서비스 세트 식별자가 필요합니다.

```
nmcli c add connection.type wifi ssid My identifier
```

connection.interface-name

연결과 관련된 장치 이름입니다.

```
nmcli con add connection.interface-name enp1s0 type ethernet
```

connection.id

연결 프로필에 사용되는 이름입니다. 연결 이름을 지정하지 않으면 다음과 같이 생성됩니다.

```
connection.type -connection.interface-name
```

connection.id 는 *연결 프로필* 의 이름이며 장치를 나타내는 인터페이스 이름(**wlp61s0,ens3, em1)과 혼동해서는 안 됩니다**. 그러나 사용자는 인터페이스 다음에 연결의 이름을 지정할 수 있지만 동일한 것은 아닙니다. 장치에 사용할 수 있는 여러 연결 프로필이 있을 수 있습니다. 이는 모바일 장치에 특히 유용하거나 네트워크 케이블을 여러 장치 간에 전환할 때 유용합니다. 구성을 편집하는 대신 필요에 따라 다른 프로필을 만들어 인터페이스에 적용합니다. **id** 옵션은 연결 프로필 이름도 나타냅니다.

show,**up**,**down** 과 같은 **nmcli** 명령에 대한 가장 중요한 옵션은 다음과 같습니다.

id

사용자가 연결 프로필에 할당한 식별 문자열입니다. nmcli connection 명령에서 ID를 사용하여 연결을 식별할 수 있습니다. 명령 출력의 NAME 필드는 항상 연결 ID를 나타냅니다. con-name이 수행하는 것과 동일한 연결 프로필 이름을 나타냅니다.

uuid

연결 프로필에 시스템이 할당한 고유 식별 문자열입니다. **nmcli 연결 명령에서 uuid 를 사용하여 연결을** 식별할 수 있습니다.

3.3.4. nmcli 대화형 연결 편집기 사용

nmcli 도구에는 대화형 연결 편집기가 있습니다. 이를 사용하려면 다음을 수행합니다.

```
~]$ nmcli con edit
```

표시된 목록에서 유효한 연결 유형을 입력하라는 메시지가 표시됩니다. 연결 유형을 입력하면 **nmcli** 프롬프트에 배치됩니다. 연결 유형에 익숙한 경우 유효한 연결 **유형** 옵션을 **nmcli con edit 명령에 추가하고 nmcli 프롬프트로 바로 이동할 수 있습니다. 기존 연결 프로파일을 편집하기 위한 형식은 다음과 같습니다.**

```
nmcli con edit [id | uuid | path] ID
```

새 연결 프로파일을 편집하려면 다음을 수행합니다.

```
nmcli con edit [type new-connection-type] [con-name new-connection-name]
```

nmcli 프롬프트에서 **help** 를 입력하여 유효한 명령 목록을 확인합니다. **describe** 명령을 사용하여 설정 및 해당 속성에 대한 설명을 가져옵니다.

```
describe setting.property
```

예를 들면 다음과 같습니다.

```
nmcli> describe team.config
```

3.3.5. nmcli를 사용하여 연결 프로필 생성 및 수정

연결 프로필에는 데이터 소스에 연결하는 데 필요한 연결 속성 정보가 포함되어 있습니다.

nmcli 를 사용하여 **NetworkManager** 에 대한 새 프로필을 **만들려면** 다음을 수행합니다.

```
nmcli c add {ARGUMENTS}
```

nmcli c add에서는 다음 두 가지 유형의 매개변수를 사용할 수 있습니다.

속성 이름

내부적으로 연결을 설명하는 데 NetworkManager가 사용하는 이름입니다. 가장 중요한 것은 다음과 같습니다.

- connection.type

  ```
  nmcli c add connection.type bond
  ```

- connection.interface-name

  ```
  nmcli c add connection.interface-name enp1s0
  ```

- connection.id

```
nmcli c add connection.id "My Connection"
```

속성 및 설정에 대한 자세한 내용은 **nm-settings(5) 도움말** 페이지를 참조하십시오.

별칭 이름

내부적으로 속성으로 변환되는 사람이 읽을 수 있는 이름입니다. 가장 일반적인 방법은 다음과 같습니다.

- 유형 (연결.type 속성)

```
nmcli c add type bond
```

- ifname (connect.interface-name 속성)

```
nmcli c add ifname enp1s0
```

- con-name (connect.id 속성)

```
nmcli c add con-name "My Connection"
```

이전 버전의 **nmcli** 에서 **별칭** 을 사용하여 필요한 연결을 생성하려면 다음을 수행합니다. 예를 들어, **ifname** enp1s0 및 **con-name** My Connection. 다음 형식의 명령을 사용할 수 있습니다.

```
nmcli c add type ethernet ifname enp1s0 con-name "My Connection"
```

최신 버전에서는 **속성 이름과 별칭** 을 모두 서로 바꿔 사용할 수 있습니다. 다음 예제는 모두 유효하고 동일합니다.

```
nmcli c add type ethernet ifname enp1s0 con-name "My Connection" ethernet.mtu 1600
```

```
nmcli c add connection.type ethernet ifname enp1s0 con-name "My Connection" ethernet.mtu 1600
```

```
nmcli c add connection.type ethernet connection.interface-name enps1s0 connection.id "My Connection" ethernet.mtu 1600
```

인수는 연결 유형에 따라 다릅니다. 모든 연결 유형에 대해 **type** 인수만 필요하며 **ifname** 은 **bond**,**team**,**bridge** 및 **vlan** 을 제외한 모든 유형에 대해 필수입니다.

type *type_name*

연결 유형. 예를 들면 다음과 같습니다.

```
nmcli c add type bond
```

ifname *interface_name*

연결을 바인딩할 인터페이스입니다. 예를 들면 다음과 같습니다.

```
nmcli c add ifname interface_name type ethernet
```

연결 프로필의 속성을 하나 이상 **수정하려면** 다음 명령을 사용합니다.

```
nmcli c modify
```

예를 들어 **연결.id** 를 My Connection에서 My **favorite connection으로 변경하고 connection .interface-name** 을 **enp1s0** 으로 변경하려면 다음과 같이 명령을 실행합니다.

```
nmcli c modify "My Connection" connection.id "My favorite connection" connection.interface-name enp1s0
```

참고

속성 이름을 사용하는 것이 좋습니다. **별칭** 은 호환성을 위해서만 사용됩니다.

또한 이더넷 MTU를 1600으로 설정하려면 다음과 같이 크기를 수정합니다.

```
nmcli c modify "My favorite connection" ethernet.mtu 1600
```

nmcli를 사용하여 수정된 연결 후 변경 사항을 적용하려면 다음 명령을 입력하여 연결을 다시 활성화합니다.

```
nmcli con up con-name
```

예를 들어 다음과 같습니다.

```
nmcli con up My-favorite-connection
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/16)
```

3.3.6. nmcli를 사용하여 네트워크에 연결

현재 사용 가능한 네트워크 연결을 나열하려면 다음을 수행합니다.

```
~]$ nmcli con show
NAME            UUID                                  TYPE           DEVICE
Auto Ethernet   9b7f2511-5432-40ae-b091-af2457dfd988  802-3-ethernet  --
ens3            fb157a65-ad32-47ed-858c-102a48e064a2  802-3-ethernet  ens3
MyWiFi          91451385-4eb8-4080-8b82-720aab8328dd  802-11-wireless wlp61s0
```

출력의 **NAME** 필드는 항상 연결 ID(이름)를 나타냅니다. 이 이름은 인터페이스 이름이 아니며, 인터페이스 이름이 같을 수도 있습니다. 위에 표시된 두 번째 연결에서 NAME **3** 은 인터페이스에 적용된 프로필에 대해 사용자가 제공하는 연결 ID입니다. ens3. 마지막 연결에서 사용자는 인터페이스에 연결 ID **MyWiFi** 를 할당했습니다. wlp61s0.

이더넷 연결을 추가하면 장치에 할당된 구성 프로필을 만듭니다. 새 프로필을 만들기 전에 다음과 같이 사용 가능한 장치를 검토합니다.

```
~]$ nmcli device status
DEVICE  TYPE      STATE         CONNECTION
ens3    ethernet  disconnected  --
ens9    ethernet  disconnected  --
lo      loopback  unmanaged     --
```

3.3.7. nmcli를 사용하여 동적 이더넷 연결 추가 및 구성

동적 이더넷 연결 추가
동적 **IP** 구성을 사용하여 이더넷 구성 프로필을 추가하려면 **DHCP** 에서 네트워크 구성을 할당할 수 있습니다.

```
nmcli connection add type ethernet con-name connection-name ifname interface-name
```

예를 들어 *my-office* 라는 동적 연결 프로필을 생성하려면 다음을 수행합니다.

```
~]$ nmcli con add type ethernet con-name my-office ifname ens3
Connection 'my-office' (fb157a65-ad32-47ed-858c-102a48e064a2) successfully added.
```

이더넷 연결을 열려면 다음을 수행합니다.

```
~]$ nmcli con up my-office
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/5)
```

장치 및 연결의 상태를 검토합니다.

```
~]$ nmcli device status
DEVICE  TYPE      STATE         CONNECTION
ens3    ethernet  connected     my-office
ens9    ethernet  disconnected  --
lo      loopback  unmanaged     --
```

동적 이더넷 연결 구성
호스트에서 **DHCP** 서버로 보낸 호스트 이름을 변경하려면 **dhcp-hostname** 속성을 수정합니다.

```
~]$ nmcli con modify my-office my-office ipv4.dhcp-hostname host-name ipv6.dhcp-hostname host-name
```

호스트에서 **DHCP** 서버로 보낸 **IPv4** 클라이언트 ID를 변경하려면 **dhcp-client-id** 속성을 수정합니다.

```
~]$ nmcli con modify my-office my-office ipv4.dhcp-client-id client-ID-string
```

IPv6 에 대한 **dhcp-client-id** 속성이 없으며 **dhclient** 는 **IPv6** 의 식별자를 만듭니다. 자세한 내용은 **dhclient(8)** 도움말 페이지를 참조하십시오.

DHCP 서버에서 호스트로 전송된 **DNS** 서버를 무시하려면 **ignore-auto-dns** 특성을 수정합니다.

```
~]$ nmcli con modify my-office my-office ipv4.ignore-auto-dns yes ipv6.ignore-auto-dns yes
```

속성 및 설정에 대한 자세한 내용은 **nm-settings(5) 도움말** 페이지를 참조하십시오.

예 3.9. 대화형 편집기를 사용하여 동적 이더넷 연결 구성

대화형 편집기를 사용하여 동적 이더넷 연결을 구성하려면 다음을 수행합니다.

```
~]$ nmcli con edit type ethernet con-name ens3

===| nmcli interactive connection editor |===
```

```
Adding a new '802-3-ethernet' connection

Type 'help' or '?' for available commands.
Type 'describe [<setting>.<prop>]' for detailed property description.

You may edit the following settings: connection, 802-3-ethernet (ethernet), 802-1x, ipv4, ipv6, dcb
nmcli> describe ipv4.method

=== [method] ===
[NM property description]
IPv4 configuration method.  If 'auto' is specified then the appropriate automatic method (DHCP,
PPP, etc) is used for the interface and most other properties can be left unset.  If 'link-local' is
specified, then a link-local address in the 169.254/16 range will be assigned to the interface.  If
'manual' is specified, static IP addressing is used and at least one IP address must be given in the
'addresses' property.  If 'shared' is specified (indicating that this connection will provide network
access to other computers) then the interface is assigned an address in the 10.42.x.1/24 range
and a DHCP and forwarding DNS server are started, and the interface is NAT-ed to the current
default network connection.  'disabled' means IPv4 will not be used on this connection.  This
property must be set.

nmcli> set ipv4.method auto
nmcli> save
Saving the connection with 'autoconnect=yes'. That might result in an immediate activation of the
connection.
Do you still want to save? [yes] yes
Connection 'ens3' (090b61f7-540f-4dd6-bf1f-a905831fc287) successfully saved.
nmcli> quit
~]$
```

기본 조치는 연결 프로필을 영구적으로 저장하는 것입니다. 필요한 경우 **save 임시 명령을 사용하여 다음에 다시 시작할 때까지 메모리에만 프로필을 저장할** 수 있습니다.

3.3.8. nmcli를 사용하여 정적 이더넷 연결 추가 및 구성

정적 이더넷 연결 추가
정적 **IPv4** 구성으로 이더넷 연결을 추가하려면 다음을 수행합니다.

```
nmcli connection add type ethernet con-name connection-name ifname interface-name ip4 address
gw4 address
```

ip 6 및 gw6 옵션을 사용하여 IPv 6 주소 및 게이트웨이 정보를 추가할 수 있습니다.

예를 들어 **IPv4** 주소 및 게이트웨이만 사용하여 정적 이더넷 연결을 생성하려면 다음을 수행합니다.

```
~]$ nmcli con add type ethernet con-name test-lab ifname ens9 ip4 10.10.10.10/24 \
gw4 10.10.10.254
```

선택적으로 장치의 **IPv6** 주소 및 게이트웨이를 동시에 지정합니다.

```
~]$ nmcli con add type ethernet con-name test-lab ifname ens9 ip4 10.10.10.10/24 \
gw4 10.10.10.254 ip6 abbe::cafe gw6 2001:db8::1
Connection 'test-lab' (05abfd5e-324e-4461-844e-8501ba704773) successfully added.
```

두 개의 **IPv4 DNS** 서버 주소를 설정하려면 다음을 수행합니다.

```
~]$ nmcli con mod test-lab ipv4.dns "8.8.8.8 8.8.4.4"
```

이렇게 하면 이전에 설정한 **DNS** 서버가 모두 교체됩니다. **IPv6 DNS** 서버 주소 두 개를 설정하려면 다음을 수행합니다.

```
~]$ nmcli con mod test-lab ipv6.dns "2001:4860:4860::8888 2001:4860:4860::8844"
```

이렇게 하면 이전에 설정한 **DNS** 서버가 모두 교체됩니다. 또는 이전 세트에 다른 **DNS** 서버를 추가하려면 **+** 접두사를 사용합니다.

```
~]$ nmcli con mod test-lab +ipv4.dns "8.8.8.8 8.8.4.4"
```

```
~]$ nmcli con mod test-lab +ipv6.dns "2001:4860:4860::8888 2001:4860:4860::8844"
```

새 이더넷 연결을 열려면 다음을 수행합니다.

```
~]$ nmcli con up test-lab ifname ens9
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/6)
```

장치 및 연결의 상태를 검토합니다.

```
~]$ nmcli device status
DEVICE  TYPE      STATE      CONNECTION
ens3    ethernet  connected  my-office
ens9    ethernet  connected  test-lab
lo      loopback  unmanaged  --
```

새로 구성된 연결에 대한 자세한 정보를 보려면 다음과 같이 명령을 실행합니다.

```
~]$ nmcli -p con show test-lab
===============================================================================

                Connection profile details (test-lab)
===============================================================================

connection.id:                      test-lab
connection.uuid:                    05abfd5e-324e-4461-844e-8501ba704773
connection.interface-name:          ens9
connection.type:                    802-3-ethernet
connection.autoconnect:             yes
connection.timestamp:               1410428968
connection.read-only:               no
connection.permissions:
connection.zone:                    --
connection.master:                  --
connection.slave-type:              --
connection.secondaries:
connection.gateway-ping-timeout:    0
[output truncated]
```

p **, --pretty** 옵션을 사용하면 제목 배너와 섹션이 출력에 추가됩니다.

예 3.10. 대화형 편집기를 사용하여 정적 이더넷 연결 구성

대화형 편집기를 사용하여 정적 이더넷 연결을 구성하려면 다음을 수행합니다.

```
~]$ nmcli con edit type ethernet con-name ens3

===| nmcli interactive connection editor |===

Adding a new '802-3-ethernet' connection

Type 'help' or '?' for available commands.
Type 'describe [>setting<.>prop<]' for detailed property description.

You may edit the following settings: connection, 802-3-ethernet (ethernet), 802-1x, ipv4, ipv6, dcb
nmcli> set ipv4.addresses 192.168.122.88/24
Do you also want to set 'ipv4.method' to 'manual'? [yes]: yes
nmcli>
nmcli> save temporary
Saving the connection with 'autoconnect=yes'. That might result in an immediate activation of the
connection.
Do you still want to save? [yes] no
nmcli> save
Saving the connection with 'autoconnect=yes'. That might result in an immediate activation of the
connection.
Do you still want to save? [yes] yes
Connection 'ens3' (704a5666-8cbd-4d89-b5f9-fa65a3dbc916) successfully saved.
nmcli> quit
~]$
```

기본 조치는 연결 프로필을 영구적으로 저장하는 것입니다. 필요한 경우 **save 임시 명령을 사용하여 다음에 다시 시작할 때까지 메모리에만 프로필을 저장할** 수 있습니다.

NetworkManager 는 내부 매개 변수 **connection.autoconnect** 를 **yes** 로 설정합니다. 또한 **NetworkManager** 는 해당 BOOTPROTO가 **none** 으로 설정되고 ONBOOT가 **yes** 로 설정되는 **/etc/sysconfig/network-scripts/ifcfg-my-office** 에 설정을 작성합니다.

나중에 인터페이스가 표시될 때까지 **NetworkManager** 는 ifcfg 파일에 대한 수동 변경 사항을 인식하지 못합니다. 설정 파일 사용에 대한 자세한 내용은 2.7절. “sysconfig 파일로 NetworkManager 사용”, 3.5절. “ifcfg 파일을 사용하여 IP 네트워킹 구성” 을 참조하십시오.

3.3.9. nmcli를 사용하여 특정 장치로 프로필 잠금

특정 인터페이스 장치에 프로필을 잠그려면 다음을 수행합니다.

```
nmcli connection add type ethernet con-name connection-name ifname interface-name
```

모든 호환 가능 이더넷 인터페이스에 프로필을 사용하려면 다음을 수행합니다.

```
nmcli connection add type ethernet con-name connection-name ifname "*"
```

특정 인터페이스를 설정하지 않으려면 **ifname** 인수를 사용해야 합니다. 와일드카드 문자 ***** 를 사용하여 모든 호환 장치와 함께 프로필을 사용할 수 있도록 지정합니다.

특정 MAC 주소로 프로필을 잠그려면 다음을 수행합니다.

```
nmcli connection add type ethernet con-name "connection-name" ifname "*" mac 00:00:5E:00:53:00
```

3.3.10. nmcli를 사용하여 Wi-Fi 연결 추가

사용 가능한 Wi-Fi 액세스 포인트를 보려면 다음을 수행하십시오.

```
~]$ nmcli dev wifi list
  SSID          MODE  CHAN  RATE     SIGNAL  BARS  SECURITY
  FedoraTest    Infra  11   54 MB/s  98      ▂▄▆█  WPA1
  Red Hat Guest  Infra  6   54 MB/s  97      ▂▄▆█  WPA2
  Red Hat       Infra  6   54 MB/s  77      ▂▄▆_  WPA2 802.1X
* Red Hat       Infra  40  54 MB/s  66      ▂▄▆_  WPA2 802.1X
  VoIP          Infra  1   54 MB/s  32      ▂▄__  WEP
  MyCafe        Infra  11  54 MB/s  39      ▂▄__  WPA2
```

고정 **IP** 구성을 사용하여 Wi-Fi 연결 프로필을 생성하지만 자동 **DNS** 주소 할당을 허용하려면 다음을 수행합니다.

```
~]$ nmcli con add con-name MyCafe ifname wlp61s0 type wifi ssid MyCafe \
ip4 192.168.100.101/24 gw4 192.168.100.1
```

WPA2 암호를 설정하려면 다음을 "실행합니다".

```
~]$ nmcli con modify MyCafe wifi-sec.key-mgmt wpa-psk
~]$ nmcli con modify MyCafe wifi-sec.psk caffeine
```

암호 보안에 대한 정보는 *Red Hat Enterprise Linux 7 보안 가이드* 를 참조하십시오.

Wi-Fi 상태를 변경하려면 다음을 수행합니다.

```
~]$ nmcli radio wifi [on | off ]
```

nmcli를 사용하여 특정 속성 변경

특정 속성을 확인하려면 **mtu**:

```
~]$ nmcli connection show id 'MyCafe' | grep mtu
802-11-wireless.mtu:                    auto
```

설정 속성을 변경하려면 다음을 수행합니다.

```
~]$ nmcli connection modify id 'MyCafe' 802-11-wireless.mtu 1350
```

변경 사항을 확인하려면 다음을 수행합니다.

```
~]$ nmcli connection show id 'MyCafe' | grep mtu
802-11-wireless.mtu:                    1350
```

NetworkManager 는 **802-3-ethernet** 및 **802-11-wireless** 와 같은 매개 변수를 설정으로 참조하며 **mtu** 는 설정 속성으로 참조합니다. 속성 및 설정에 대한 자세한 내용은 **nm-settings(5) 도움말** 페이지를 참조하십시오.

3.3.11. Ignore Certain Devices로 NetworkManager 구성

기본적으로 NetworkManager는 **lo** (loopback) 장치를 제외한 모든 장치를 관리합니다. 그러나 NetworkManager가 이러한 장치를 무시하는 것을 구성하기 위해 **관리되지 않는** 특정 장치를 설정할 수 있습니다. 이 설정을 사용하면 예를 들어 스크립트를 사용하여 이러한 장치를 수동으로 관리할 수 있습니다.

3.3.11.1. 영구적으로 NetworkManager에서 관리되지 않음으로 장치 설정

인터페이스 이름, MAC 주소 또는 장치 유형과 같은 여러 기준에 따라 **관리되지 않는** 장치로 장치를 구성할 수 있습니다. 다음 절차에서는 NetworkManager에서 관리되지 않는 **enp1s0** 인터페이스를 영구적으로 설정하는 방법을 설명합니다.

일시적으로 네트워크 장치를 **관리** 되지 않음으로 구성하려면 3.3.11.2절. "일시적으로 NetworkManager에서 관리되지 않음으로 장치 설정" 을 참조하십시오.

절차

1. 선택 사항: **관리되지 않는** 장치로 설정할 장치 목록을 표시합니다:

```
# nmcli device status
DEVICE  TYPE      STATE         CONNECTION
enp1s0  ethernet  disconnected  --
...
```

2. 다음 콘텐츠를 사용하여 **/etc/NetworkManager/conf.d/99-unmanaged-devices.conf** 파일을 생성합니다.

```
[keyfile]
unmanaged-devices=interface-name:enp1s0
```

여러 장치를 관리되지 않는 것으로 설정하려면 ***unmanaged-devices*** 매개변수의 항목을 세미콜론으로 구분합니다.

```
[keyfile]
unmanaged-devices=interface-name:interface_1;interface-name:interface_2;...
```

3. **NetworkManager** 서비스를 다시 로드합니다.

```
# systemctl reload NetworkManager
```

검증 단계

- 장치 목록을 표시합니다.

```
# nmcli device status
DEVICE  TYPE      STATE      CONNECTION
enp1s0  ethernet  unmanaged  --
...
```

enp1s0 장치 옆의 **관리되지 않는** 상태는 NetworkManager가 이 장치를 관리하지 않음을 나타냅니다.

추가 리소스

장치를 관리되지 않음으로 구성하는 데 사용할 수 있는 기준 목록과 해당 구문은 NetworkManager.conf(5) 도움말 페이지의 *장치 목록 형식* 섹션을 참조하십시오.

3.3.11.2. 일시적으로 NetworkManager에서 관리되지 않음으로 장치 설정

인터페이스 이름, MAC 주소 또는 장치 유형과 같은 여러 기준에 따라 **관리되지 않는** 장치로 장치를 구성할 수 있습니다. 다음 절차에서는 NetworkManager에서 **관리되지 않는 enp1s0** 인터페이스를 일시적으로 설정하는 방법에 대해 설명합니다.

예를 들어 테스트 목적으로 이 방법을 사용합니다. 네트워크 장치를 **관리** 되지 않음으로 영구적으로 구성하려면 3.3.11.1절. "영구적으로 NetworkManager에서 관리되지 않음으로 장치 설정" 을 참조하십시오.

절차

1. 선택 사항: **관리되지 않는** 장치로 설정할 장치 목록을 표시합니다:

```
# nmcli device status
DEVICE  TYPE      STATE         CONNECTION
enp1s0  ethernet  disconnected  --
...
```

2. **enp1s0** 장치를 **Unmanaged** 상태로 설정합니다.

```
# nmcli device set enp1s0 managed no
```

검증 단계

- 장치 목록을 표시합니다.

```
# nmcli device status
DEVICE  TYPE      STATE      CONNECTION
enp1s0  ethernet  unmanaged  --
...
```

enp1s0 장치 옆의 **관리되지 않는** 상태는 NetworkManager가 이 장치를 관리하지 않음을 나타냅니다.

추가 리소스

장치를 관리되지 않음으로 구성하는 데 사용할 수 있는 기준 목록과 해당 구문은 NetworkManager.conf(5) 도움말 페이지의 *장치 목록 형식* 섹션을 참조하십시오.

3.4. GNOME GUI를 사용하여 IP 네트워킹 구성

Red Hat Enterprise Linux 7에서 **NetworkManager** 에는 자체 GUI(그래픽 사용자 인터페이스)가 없습니다. 데스크탑 오른쪽 상단에 있는 네트워크 연결 아이콘은 GNOME Shell의 일부로 제공되며 **네트워크** 설정 구성 도구는 유선, 무선, vpn 연결을 지원하는 새 GNOME **제어 센터** GUI의 일부로 제공됩니다. **nm-connection-editor** 는 GUI 구성을 위한 기본 도구입니다. **제어 센터** 의 기능 외에도 본딩, 팀, 브릿지 연결 구성 **등 GNOME 제어** 센터에서 제공하지 않는 기능도 적용합니다. 이 섹션에서는 다음을 사용하여 네트워크 인터페이스를 구성할 수 있습니다.

- GNOME **control-center** 애플리케이션
- GNOME **nm-connection-editor** 애플리케이션

3.4.1. 제어 센터 GUI를 사용하여 네트워크에 연결

제어 센터 애플리케이션의 **네트워크** 설정 창에 액세스하는 방법은 다음 두 가지가 있습니다.

- **Super** 키를 눌러 Activities Overview(활동 개요)를 입력하고 **Settings** (설정)를 입력한 다음 **Enter** 키를 누릅니다. 그런 다음 왼쪽에서 **Network** (네트워크) 탭을 선택하면 **Network** settings(네트워크 설정) 도구가 표시됩니다. "제어 센터를 사용하여 새 연결 구성" 진행하십시오.
- 화면의 오른쪽 상단에 있는 GNOME Shell 네트워크 연결 아이콘을 클릭하여 메뉴를 엽니다.

그림 3.5. 제어 센터 애플리케이션을 사용한 네트워크 구성

[D]

GNOME 쉘 네트워크 연결 아이콘을 클릭하면 다음이 표시됩니다.

- 현재 연결되어 있는 범주화된 네트워크 목록(예: **Wired** 및 **Wi-Fi**).
- **NetworkManager** 가 감지한 **모든 사용 가능한 네트워크** 목록입니다.
- 구성된 가상 사설 네트워크(VPN)에 연결하기 위한 옵션

  및

- **Network Settings** (네트워크 설정) 메뉴 항목을 선택하기 위한 옵션.

네트워크에 연결되어 있는 경우 연결 이름 **왼쪽에 차단** 기호가 표시됩니다.

Network Settings (네트워크 설정)를 클릭하면 **네트워크** 설정 도구가 나타납니다. "제어 센터를 사용하여 새 연결 구성" 진행하십시오.

3.4.2. GUI를 사용하여 새 연결 및 기존 연결 편집 구성

시스템 관리자는 네트워크 연결을 구성할 수 있습니다. 이를 통해 사용자는 인터페이스의 설정을 적용하거나 변경할 수 있습니다. 이를 위해 다음 두 가지 방법 중 하나를 사용할 수 있습니다.

- GNOME **control-center** 애플리케이션

- GNOME **nm-connection-editor** 애플리케이션

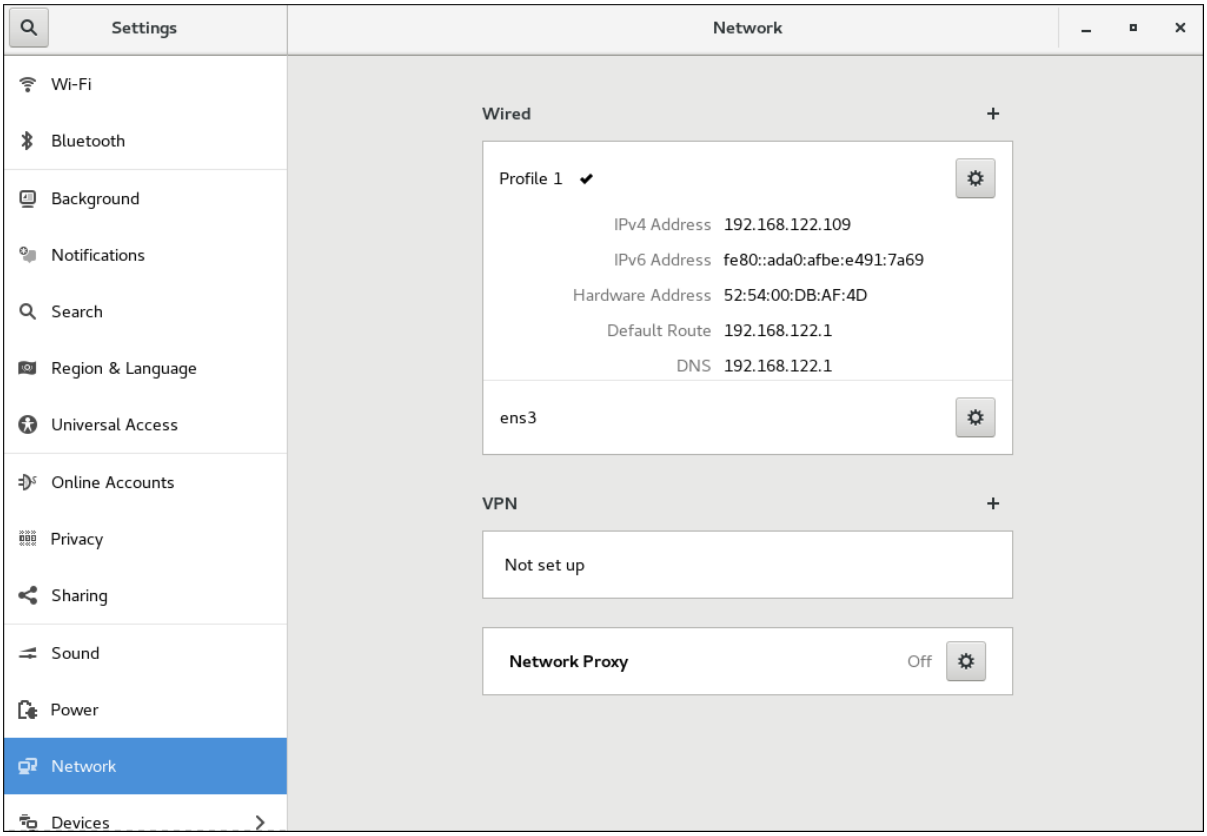
3.4.2.1. 제어 센터를 사용하여 새 연결 및 기존 연결 구성

GNOME **제어 센터** 애플리케이션을 사용하여 네트워크 연결을 생성하고 구성할 수 있습니다.

제어 센터를 사용하여 새 연결 구성
제어 센터 애플리케이션을 사용하여 새 유선, 무선, vpn 연결을 구성하려면 다음과 같이 진행합니다.

1. **Super** 키를 눌러 Activities Overview(활동 개요)를 입력하고 **Settings** (설정)를 입력한 다음 **Enter** 키를 누릅니다. 그런 다음 왼쪽에 있는 **Network** (네트워크) 탭을 선택합니다. **네트워크** 설정 도구는 오른쪽 메뉴에 나타납니다.

   그림 3.6. 네트워크 설정 창 열기



2. 더하기 버튼을 클릭하여 새 연결을 추가합니다.

   구성하려면 다음을 수행합니다.

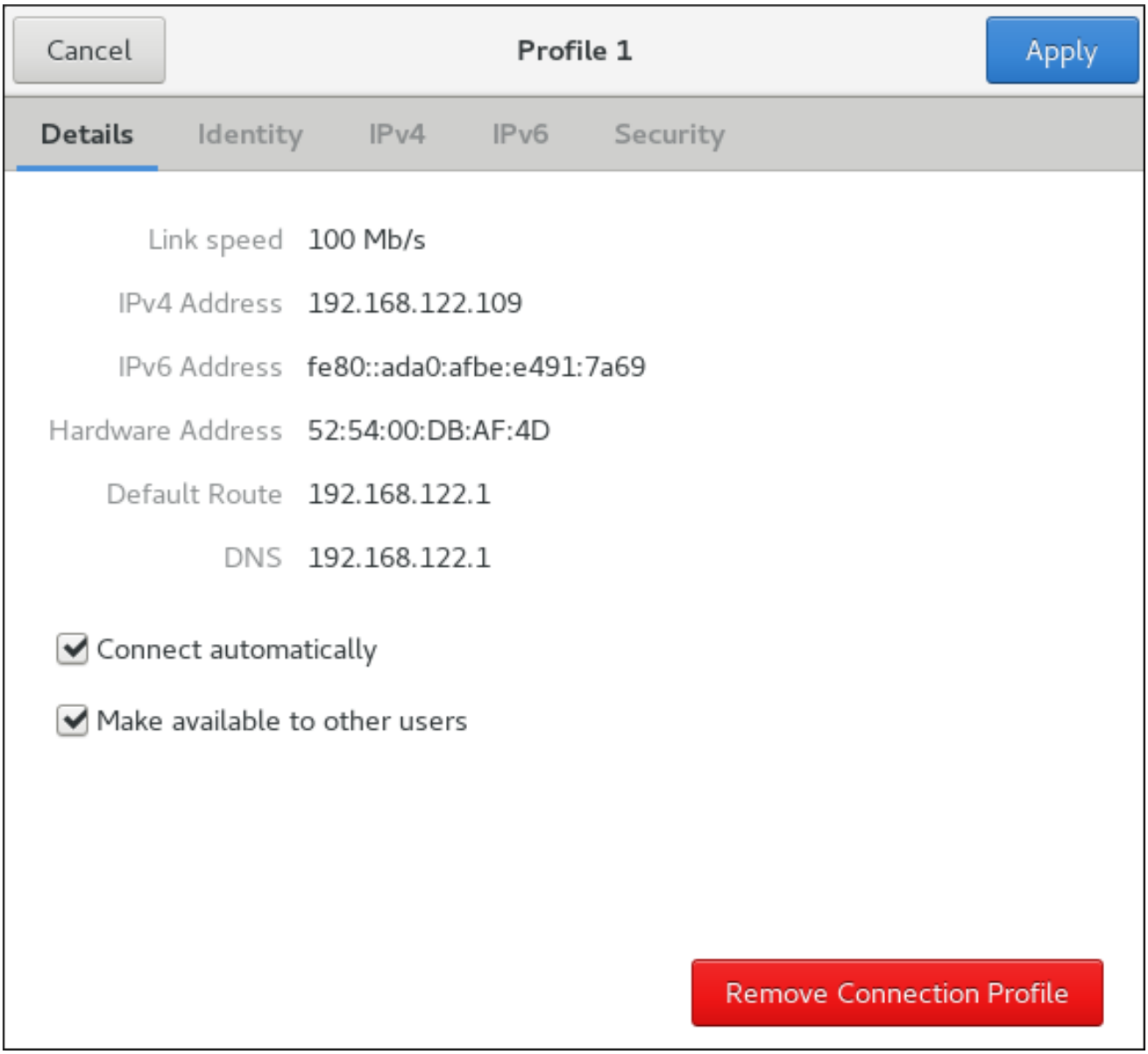
   - **유선 연결, Wired 항목 옆에 있는 더하기 버튼을 클릭하고 3.4.6절. "GUI를 사용하여 Wired (Ethernet) 연결 설정" 진행합니다.**

   - **VPN 연결, VPN** 항목 옆의 더하기 버튼을 클릭하고 계속 진행합니다. 3.4.8.1절. "제어 센터를 사용하여 VPN 연결 설정"

   Wi-Fi 연결 의 경우 **Settings** (설정) 메뉴에서 **Wi-** Fi 항목을 클릭하고 계속 진행합니다. 3.4.7절. "GUI를 사용하여 Wi-Fi 연결 구성"

제어 센터를 사용하여 기존 연결 편집
네트워크 설정 창에서 기존 연결 프로필의 기어 wheel 아이콘을 클릭하면 **IP** 주소 지정, **DNS**, 라우팅 구성과 같은 대부분의 네트워크 구성 작업을 수행할 수 있는 **세부 정보** 창이 열립니다.

그림 3.7. 네트워크 연결 세부 정보 창을 사용하여 네트워크 구성



[D]

추가하거나 구성하는 모든 연결 유형의 경우 **NetworkManager** 를 선택하여 사용 가능한 경우 해당 네트워크에 자동으로 연결할 수 있습니다. 이를 위해 **NetworkManager** 가 사용 가능 여부를 감지할 때마다 **NetworkManager** 가 연결에 **자동으로** 연결되도록 하려면 연결을 선택합니다. **NetworkManager** 가 자동으로 연결하지 않도록 하려면 확인란을 지웁니다. 확인란이 선택 취소되면 네트워크 연결 아이콘의 메뉴에서 해당 연결을 수동으로 선택하여 연결하도록 해야 합니다.

다른 사용자가 연결할 수 있도록 하려면 **Make available to other users** 확인란을 선택합니다.

연결 수정 후에 변경 사항을 적용하려면 연결 창의 오른쪽 상단 모서리에 있는 **Apply** (적용) 버튼을 클릭하면 됩니다.

연결 프로필 제거 빨간색 상자를 클릭하여 연결을 삭제할 수 있습니다.

3.4.2.2. nm-connection-editor를 사용하여 새 연결 및 기존 연결 구성

nm-connection-editor GUI 응용 프로그램을 사용하여 **control-center** 에서 제공하는 것보다 추가 기능으로 원하는 연결을 구성할 수 있습니다. 또한 **nm-connection-editor** 는 본딩, 브리지, VLAN, 팀 연결 구성 **등 GNOME 제어** 센터에서 제공하지 않는 기능을 적용합니다.

nm-connection-editor를 사용하여 새 연결 구성
nm-connection-editor 를 사용하여 새 연결 유형을 추가하려면 다음을 수행합니다.

절차

1. 터미널에서 **nm-connection-editor** 를 입력합니다.

```
~]$ nm-connection-editor
```

Network Connections(네트워크 연결) 창이 표시됩니다.

2. 더하기 버튼을 클릭하여 연결 유형을 선택합니다.

그림 3.8. nm-connection-editor를 사용하여 연결 유형 추가

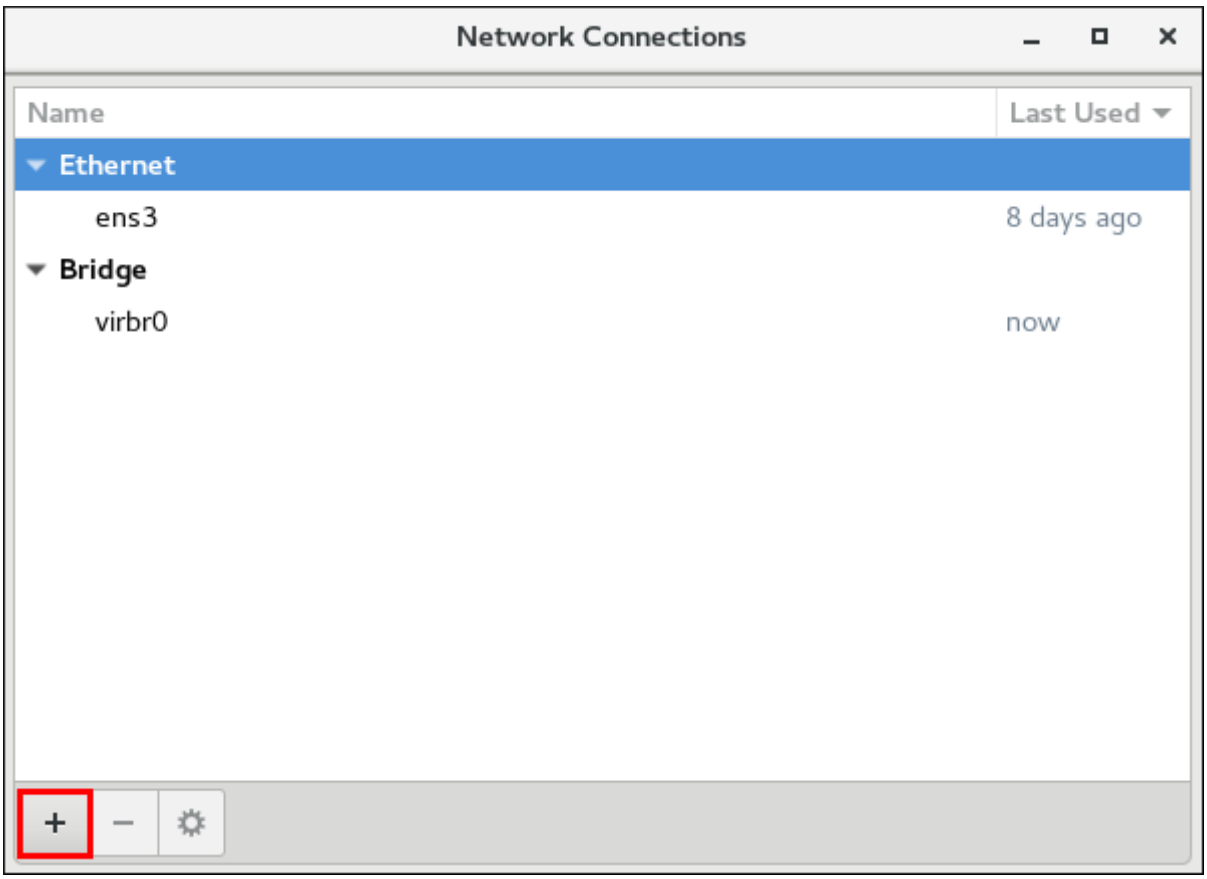


그림 3.9. nm-connection-editor로 연결 유형 선택

생성 및 구성하려면 다음을 수행합니다.

- **본딩 연결**, **Bond** 항목을 클릭하고 7.8.1절. "본딩 연결 설정" 로 진행합니다.
- **브리지 연결을** 클릭하고 **브리지** 항목을 클릭하고 9.4.1절. "GUI를 사용하여 브리지 연결 설정" 로 진행합니다.
- **VLAN 연결**, **VLAN** 항목을 클릭하고 10.5.1절. "VLAN 연결 설정"; 또는,
- **팀 연결을** 클릭하고 **팀** 항목을 클릭하고 8.14절. "GUI를 사용하여 네트워크 팀 만들기" 진행합니다.

nm-connection-editor로 기존 연결 편집
기존 연결 유형의 경우 **Network Connections** (네트워크 연결) 대화 상자에서 기어 wheel 아이콘을 클릭하고 "nm-connection-editor를 사용하여 새 연결 구성" 을 참조하십시오.

3.4.3. nm-connection-editor를 사용하는 일반적인 구성 옵션

nm-connection-editor 유틸리티를 사용하는 경우 다음과 같은 절차에 따라 가장 많은 연결 유형 (ethernet, wifi, 모바일 광대역, DSL)에 대한 일반적인 구성 옵션이 5가지가 있습니다.

절차

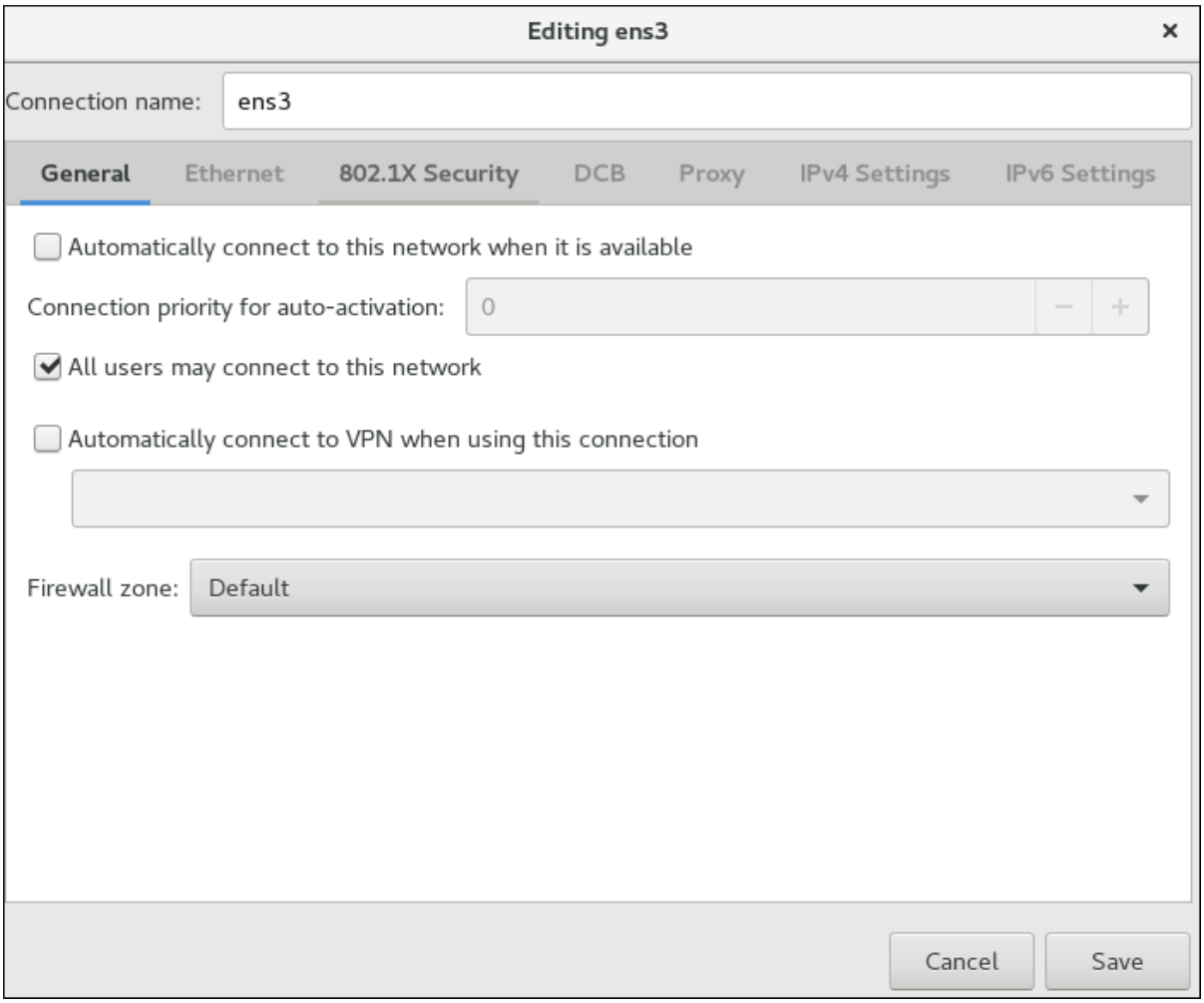
1. 터미널에서 **nm-connection-editor** 를 입력합니다.

```
~]$ nm-connection-editor
```

Network Connections(네트워크 연결) 창이 표시됩니다. 더하기 버튼을 클릭하여 연결 유형 또는 기어 wheel 아이콘을 선택하여 기존 연결을 편집합니다.

2. **편집** 대화 상자에서 **General(일반)** 탭을 선택합니다.

그림 3.10. nm-connection-editor의 설정 옵션



- **연결 이름** - 네트워크 연결에 대한 설명이 포함된 이름을 입력합니다. 이 이름은 **Network** (네트워크) 창의 메뉴에 이 연결을 나열하는 데 사용됩니다.
- **자동 활성화의 연결 우선 순위** - 연결이 자동 연결로 설정되면 번호가 활성화됩니다(**기본적으로 0)**. 숫자가 클수록 우선 순위가 높습니다.
- 사용 가능한 **경우 이 네트워크에 자동으로 연결 - 사용 가능한 경우 NetworkManager 가** 이 연결에 자동으로 연결되도록 하려면 이 상자를 선택합니다. 자세한 내용은 "제어 센터를 사용하여 기존 연결 편집"을 참조하십시오.
- **모든 사용자가 이 네트워크에 연결할 수 있습니다** - 이 상자를 선택하여 시스템의 모든 사용자가 사용할 수 있는 연결을 만듭니다. 이 설정을 변경하려면 root 권한이 필요할 수 있습니다. 자세한 내용은 3.4.5절. "GUI를 사용하여 시스템 전체 및 개인 연결 프로파일 관리" 을 참조하십시오.
- **이 연결을 사용할 때 VPN에 자동으로 연결 - NetworkManager 가 VPN 연결에 자동으로 연결되도록** 하려면 이 상자를 선택합니다. 드롭다운 메뉴에서 VPN을 선택합니다.
- **방화벽 영역** - 드롭다운 메뉴에서 방화벽 영역을 선택합니다. 방화벽 영역에 대한 자세한 내용은 *Red Hat Enterprise Linux 7 보안 가이드* 를 참조하십시오.

참고

VPN 연결 유형의 경우 위의 구성 옵션 중 세 가지만 사용할 수 있습니다. **연결 이름,모든 사용자가 이 네트워크 및 방화벽 영역에 연결할 수 있습니다** .

3.4.4. GUI를 사용하여 자동으로 네트워크에 연결

추가하거나 구성하는 연결 유형의 경우 **NetworkManager** 가 해당 네트워크에 자동으로 연결할지 여부를 선택할 수 있습니다. 다음 방법 중 하나를 사용할 수 있습니다.

- GNOME **control-center** 애플리케이션
- GNOME **nm-connection-editor** 애플리케이션

3.4.4.1. 제어 센터를 사용하여 자동으로 네트워크에 연결

다음과 같이 제어 센터를 사용하여 네트워크에 자동으로 연결할 수 있습니다.

절차

1. **Super** 키를 눌러 Activities Overview(활동 개요)를 입력하고 **Settings** (설정)를 입력한 다음 **Enter** 키를 누릅니다. 그런 다음 왼쪽에 있는 **Network** (네트워크) 탭을 선택합니다. 네트워크 설정 도구가 오른쪽 메뉴에 나타납니다. "제어 센터를 사용하여 새 연결 구성" 을 참조하십시오.
2. 오른쪽 메뉴에서 네트워크 인터페이스를 선택합니다.
3. 오른쪽 메뉴에서 연결 프로필의 기어 wheel 아이콘을 클릭합니다. **Network Details(네트워크** 세부 정보) 창이 표시됩니다.
4. **세부 정보** 메뉴 항목을 선택하고 "제어 센터를 사용하여 기존 연결 편집" 을 참조하십시오.
5. **NetworkManager** 가 사용 가능 여부를 감지할 때마다 **NetworkManager** 가 연결에 **자동으로** 연결되도록 하려면 연결을 자동으로 선택합니다. **NetworkManager** 가 자동으로 연결하지 않도록 하려면 확인란을 지웁니다. 확인란이 선택 취소되면 네트워크 연결 아이콘의 메뉴에서 해당 연결을 수동으로 선택하여 연결하도록 해야 합니다.

3.4.4.2. nm-connection-editor를 사용하여 자동으로 네트워크에 연결

GNOME **nm-connection-editor** 애플리케이션을 사용하여 네트워크에 자동으로 연결할 수도 있습니다. 이를 위해 3.4.3절. "nm-connection-editor를 사용하는 일반적인 구성 옵션" 에 숨겨진 절차에 따라 **General(일반**) 탭에서 **Automatically connect to this network when it is available** (이 네트워크에 자동으로 연결됨) 확인란을 선택합니다.

3.4.5. GUI를 사용하여 시스템 전체 및 개인 연결 프로파일 관리

NetworkManager 는 모든 *연결 프로필을* 저장합니다. 프로필은 인터페이스에 적용할 수 있는 설정의 명명된 컬렉션입니다. **NetworkManager** 는 시스템 전체 사용(시스템 연결)과 모든 *사용자연결프로필을 위해 이러한 연결* 프로필을 저장합니다. 연결 프로파일에 대한 액세스는 **NetworkManager** 에 의해 저장되는 권한에 의해 제어됩니다. **연결** 설정 **권한** 속성에 대한 자세한 내용은 **nm-settings(5) 도움말** 페이지를 참조하십시오. 다음 그래픽 사용자 인터페이스 도구를 사용하여 연결 프로필에 대한 액세스를 제어할 수 있습니다.

- **nm-connection-editor** 애플리케이션
- **Control-center** 애플리케이션

3.4.5.1. nm-connection-editor를 사용하여 연결 프로필의 권한 관리

시스템의 모든 사용자가 사용할 수 있는 연결을 만들려면 3.4.3절. "nm-connection-editor를 사용하는 일반적인 구성 옵션" 에 설명되는 절차를 따르고 **General(일반**) 탭에서 **All users may connect to this network(모든 사용자가 이 네트워크** 연결 가능) 확인란을 선택합니다.

3.4.5.2. 제어 센터를 사용하여 연결 프로필의 권한 관리

다른 사용자가 연결할 수 있도록 하려면 "제어 센터를 사용하여 기존 연결 편집" 에 설명된 절차에 따라 GNOME **제어 센터 네트워크 설정세부 정보 창에서 다른 사용자가 사용할 수 있는 Make (사용 가능)** 확인란을 선택합니다.

반대로 **다른 사용자가 사용할 수 있는 Make** (사용 가능) 확인란을 지워 시스템 전체 대신 사용자별 연결 확인란을 만듭니다.

참고

시스템 정책에 따라 연결의 사용자별 여부 또는 시스템 전체에 대한 변경을 위해 시스템에 대한 루트 권한이 필요할 수 있습니다.

NetworkManager 의 기본 정책은 모든 사용자가 시스템 전체 연결을 만들고 수정할 수 있도록 하는 것입니다. 부팅 시 사용할 수 있는 프로필은 사용자가 로그인할 때까지 표시되지 않으므로 비공개일 수 없습니다. 예를 들어 사용자가 **Connect Automatically(** 자동 연결) 확인란을 사용하여 연결 프로파일 **user-em2** 를 만들지만 **Make available to other users** not selected(선택되지 않은 다른 사용자가 사용 가능하도록 설정) 확인란을 선택한 경우 부팅 시 연결을 사용할 수 없습니다.

연결 및 네트워킹을 제한하기 위해 단독으로 또는 함께 사용할 수 있는 두 가지 옵션이 있습니다.

- 다른 사용자가 사용할 수 있는 **Make available to other users** (다른 사용자가 사용할 수 있도록 설정) 확인란을 선택 취소하면 연결을 수정 가능으로 변경하고 변경 작업을 수행하는 사용자만 사용할 수 있습니다.
- polkit 프레임워크를 사용하여 사용자별로 일반 네트워크 작업의 권한을 제한합니다.

이 두 옵션을 결합하면 세분화된 보안과 네트워킹을 제어할 수 있습니다. **polkit에 대한 자세한 내용은 polkit(8)** 도움말 페이지를 참조하십시오 .

VPN 연결은 Wi-Fi 또는 이더넷 연결보다 개인용으로 간주되기 때문에 **항상** 개인-per-user로 생성됩니다.

3.4.6. GUI를 사용하여 Wired (Ethernet) 연결 설정

다음 두 가지 방법으로 GUI를 사용하여 유선 연결을 구성할 수 있습니다.

- Control-center 애플리케이션
- nm-connection-editor 애플리케이션

3.4.6.1. 제어 센터를 사용하여 유선 연결 구성

절차

1. **Super** 키를 눌러 Activities Overview(활동 개요)를 입력하고 **Settings** (설정)를 입력한 다음 **Enter** 키를 누릅니다. 그런 다음 왼쪽에서 **네트워크** 메뉴 항목을 선택하고 **네트워크** 설정 도구가 나타나면 "제어 센터를 사용하여 새 연결 구성" 을 참조하십시오.

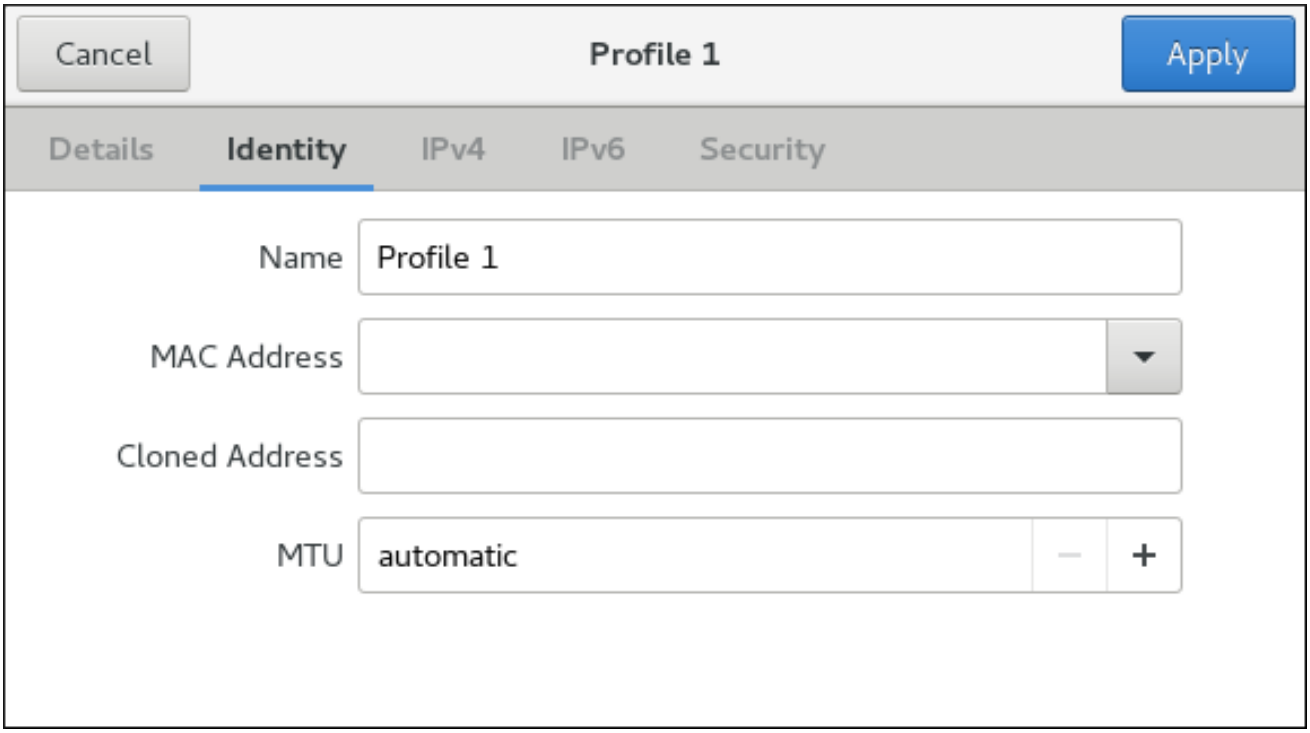
2. 아직 강조 표시되지 않은 경우 **Wired** 네트워크 인터페이스를 선택합니다.

   시스템은 기본적으로 **Wired** 라는 단일 유선 *연결 프로필을* 만들고 구성합니다. 프로필은 인터페이스에 적용할 수 있는 설정의 명명된 컬렉션입니다. 인터페이스에 대해 두 개 이상의 프로필을 생성하고 필요한 대로 적용할 수 있습니다. 기본 프로필은 삭제할 수 없지만 해당 설정은 변경할 수 있습니다.

3. 기어 wheel 아이콘을 클릭하여 기본 **Wired** 프로필을 편집합니다.

기본 설정 옵션
ID 메뉴 항목을 선택하여 **Wired** 대화 상자에서 다음 구성 설정을 확인할 수 있습니다.

그림 3.11. 유선 연결의 기본 설정 옵션



- name - 네트워크 연결을 설명하는 이름을 입력합니다. 이 이름은 **Network** (네트워크) 창의 메뉴에 이 연결을 나열하는 데 사용됩니다.
- **MAC Address** (MAC 주소) - 이 프로필이 적용되어야 하는 인터페이스의 MAC 주소를 선택합니다.
- **복제 주소** - 필요한 경우 사용할 다른 MAC 주소를 입력합니다.
- **MTU** - 필요한 경우 사용할 특정 *최대 전송 단위* (MTU)를 입력합니다. MTU 값은 링크 계층에서 전송할 가장 큰 패킷의 크기(바이트)입니다. 이 기본값은 **1500** 이며 일반적으로 지정하거나 변경할 필요가 없습니다.

추가 Wired 설정 만들기
편집 대화 상자에서 기존 연결을 추가로 구성할 수 있습니다.

구성하려면 다음을 수행합니다.

- 연결의 **IPv4** 설정, **IPv4** 메뉴 항목을 클릭하고 계속 진행합니다. 5.4절. "IPv4 설정 구성"

  또는

- 연결의 **IPv6** 설정, **IPv6** 메뉴 항목을 클릭하고 5.5절. "IPv6 설정 구성" 진행합니다.
- **포트 기반 네트워크 액세스 제어(PNAC)** 는 802.1X **보안** 메뉴 항목을 클릭하고 5.2절. "802.1X 보안 설정" 진행합니다.

새 연결 (또는 수정된 연결) 저장
유선 연결 편집을 마치면 **Apply** (적용) 버튼을 클릭하여 사용자 지정 구성을 저장합니다. 편집하는 동안 프로필이 사용 중인 경우 연결을 다시 시작하여 **NetworkManager** 에서 변경 사항을 적용하도록 합니다.

프로필이 OFF(꺼짐)인 경우 네트워크 연결 아이콘의 메뉴에서 ON(켜짐)으로 설정하거나 선택합니다. 새로운 연결 또는 변경된 연결 사용에 대한 정보는 3.4.1절. "**제어 센터 GUI를 사용하여 네트워크에** 연결" 을 참조하십시오.

새 유선 연결 만들기
새로운 유선 연결 프로파일을 만들려면 더하기 버튼을 클릭하고 "제어 센터를 사용하여 새 연결 구성" 을 참조하십시오.

더하기 버튼을 클릭하여 새 연결을 추가하는 경우 **NetworkManager** 는 해당 연결에 대한 새 구성 파일을 만든 다음 기존 연결을 편집하는 데 사용되는 동일한 대화 상자를 엽니다. "제어 센터를 사용하여 기존 연결 편집" 를 참조하십시오. 이러한 대화 상자의 차이점은 기존 연결 프로필에 **Details(세부 정보)** 메뉴 항목이 있다는 것입니다.

3.4.6.2. nm-connection-editor를 사용하여 Wired 연결 설정

nm-connection-editor GUI 애플리케이션은 **제어 센터** GUI 응용 프로그램보다 더 많은 구성 옵션을 제공합니다. **nm-connection-editor** 를 사용하여 유선 연결을 구성하려면 다음을 수행합니다.

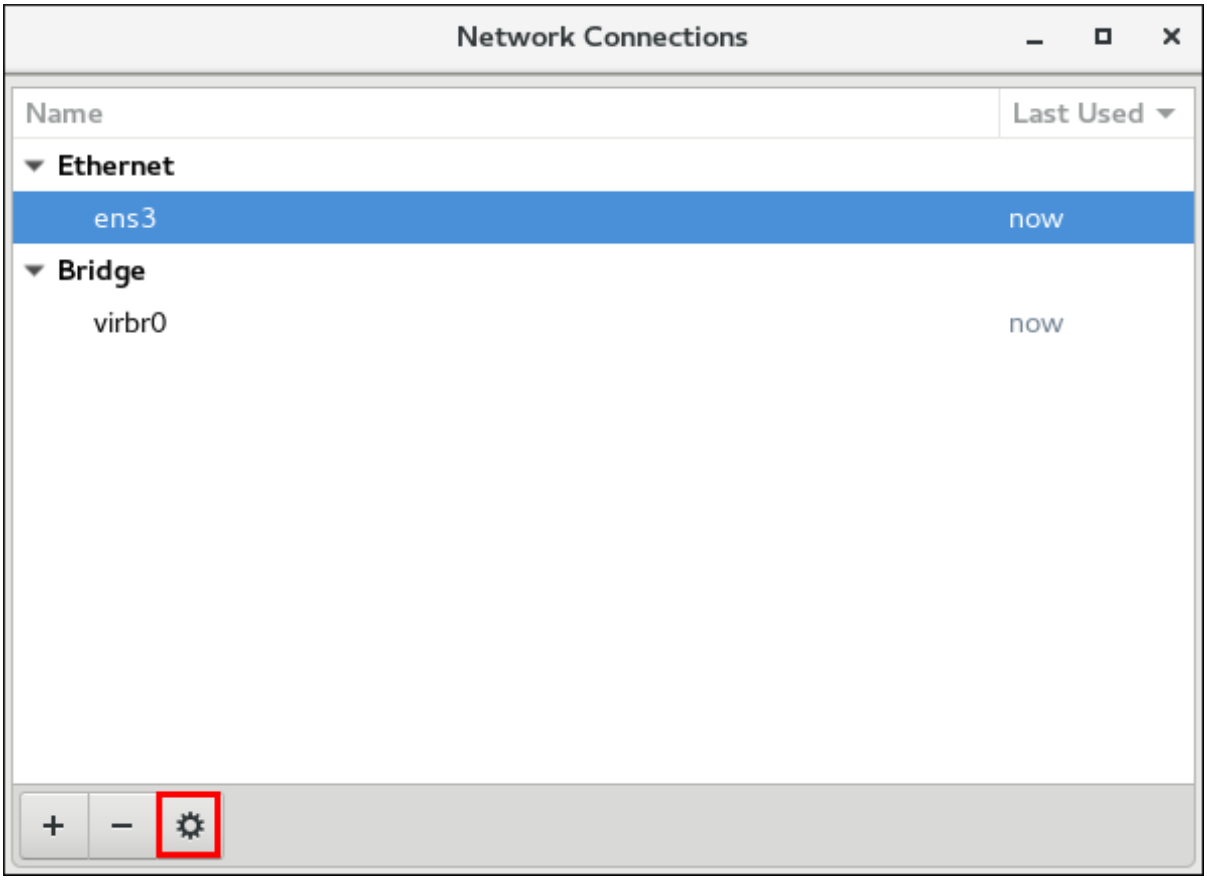
1. 터미널에 **nm-connection-editor** 를 입력합니다.

```
~]$ nm-connection-editor
```

Network Connections(네트워크 연결) 창이 표시됩니다.

2. 편집할 이더넷 연결을 선택하고 톱니바퀴 아이콘을 클릭합니다.

그림 3.12. 유선 연결 편집



Editing(편집) 대화 상자가 표시됩니다.

- 자동으로 네트워크에 연결하고 연결을 제한하려면 **General** (일반) 탭을 클릭하고 3.4.3절. "nm-connection-editor를 사용하는 일반적인 구성 옵션" 을 참조하십시오.
- 네트워킹 설정을 구성하려면 **이더넷** 탭을 클릭하고 "nm-connection-editor를 사용하여 802.3 연결 설정 구성" 를 참조하십시오.
- 유선 연결에 대한 802.1X 보안을 구성하려면 **802.1X 보안** 탭을 클릭하고 5.2.4절. "nm-connection-editor를 사용하여 Wired에 대해 802.1X 보안 설정" 을 참조하십시오.
- IPV4 설정을 구성하려면 IPV4 **Settings(IPV4 설정**) 탭을 클릭하고 "nm-connection-editor를 사용하여 IPV4의 메서드 설정" 를 참조하십시오.
- IPV6 설정을 구성하려면 **IPV6 설정** 탭을 클릭하고 5.5절. "IPv6 설정 구성" 을 참조하십시오.

3.4.7. GUI를 사용하여 Wi-Fi 연결 구성

이 섹션에서는 **NetworkManager** 를 사용하여 **Wi-Fi** (무선 또는 무선*a/b/g/n*) 연결을 액세스 지점에 구성하는 방법을 설명합니다. 액세스 지점은 무선 장치가 네트워크에 연결할 수 있도록 하는 장치입니다.

모바일 광대역(예: 3G) 연결을 구성하려면 3.4.9절. "GUI를 사용하여 모바일 광대역 연결 구성" 을 참조하십시오.

사용 가능한 액세스 지점에 빠르게 연결
절차

1. 네트워크 연결 아이콘을 클릭하여 네트워크 연결 아이콘의 메뉴를 활성화하고 3.4.1절. "**제어 센터 GUI를 사용하여 네트워크에** 연결" 을 참조하십시오.
2. **Wi-Fi** 네트워크 목록에서 액세스 지점의 *Service Set Identifier* (SSID)를 찾습니다.
3. 네트워크의 SSID 를 클릭합니다. 잠김 기호는 액세스 지점에 인증이 필요함을 나타냅니다. 액세스 지점이 안전한 경우 대화 상자에서 인증 키 또는 암호를 묻는 메시지를 표시합니다.

   NetworkManager 는 액세스 지점에서 사용하는 보안 유형을 자동 감지하려고 합니다. 여러 가능성이 있는 경우 **NetworkManager** 는 보안 유형을 추측하고 **Wi-Fi 보안** 드롭다운 메뉴에 표시합니다.

   - 암호가 있는 WPA-PSK 보안(WPA)은 선택하지 않아도 됩니다.
   - WPA Enterprise(802.1X)의 경우 자동 감지할 수 없으므로 보안을 구체적으로 선택해야 합니다.

     확실하지 않은 경우 각 유형에 차례로 연결을 시도합니다.
4. **Password** (암호) 필드에 키 또는 암호를 입력합니다. 40비트 WEP 또는 128비트 WPA 키와 같은 특정 암호 유형은 필수 길이가 아니면 유효하지 않습니다. **연결** 버튼은 선택한 보안 유형에 필요한 길이의 키를 입력할 때까지 비활성 상태로 유지됩니다. 무선 보안에 대한 자세한 내용은 5.2절. "802.1X 보안 설정" 을 참조하십시오.

NetworkManager 가 액세스 지점에 성공적으로 연결하면 네트워크 연결 아이콘이 무선 연결의 신호 강도 그래픽 표시로 변경됩니다.

이러한 자동 생성 액세스 지점 연결 중 하나에 대한 설정을 직접 추가한 것처럼 편집할 수도 있습니다. **네트워크** 창의 **Wi-Fi** 페이지에는 **기록** 버튼이 있습니다. 버튼을 클릭하면 연결하려는 모든 연결 목록이 표시됩니다. 보기 "기존 Wi-Fi 연결 편집"

숨겨진 Wi-Fi 네트워크에 연결

모든 액세스 지점에는 식별할 *Service Set Identifier* (SSID)가 있습니다. 그러나 액세스 지점은 SSID를 브로드캐스트하지 않도록 구성할 수 있으며, 이 경우 *숨겨진* 경우 **NetworkManager** 의 **사용 가능한** 네트워크 목록에 표시되지 않습니다. SSID, 인증 방법 및 비밀을 아는 한 SSID를 숨기고있는 무선 액세스 지점에 계속 연결할 수 있습니다. 숨겨진 무선 네트워크에 연결하려면 다음을 수행합니다.

절차

1. **Super** 키를 눌러 Activities Overview(활동 개요)를 입력하고 **Settings** (설정)를 입력한 다음 **Enter** 키를 누릅니다. 그런 다음 왼쪽에서 **Wi-Fi** 메뉴 항목을 선택합니다.
2. **Connect to Hidden Network** (숨겨진 네트워크에 연결)를 선택합니다. 두 가지 옵션이 있습니다.
   - 이전에 숨겨진 네트워크에 연결한 경우:
     1. **연결** 드롭다운을 사용하여 네트워크를 선택합니다.
     2. **연결**을 클릭합니다.
   - 그렇지 않은 경우 다음과 같이 진행하십시오.
     1. **Connection** (연결) 드롭다운은 **New** (새로 생성)로 그대로 둡니다.
     2. 숨겨진 네트워크의 SSID를 입력합니다.
     3. **Wi-Fi 보안** 방법을 선택합니다.
     4. 올바른 인증 시크릿을 입력합니다.
     5. **연결**을 클릭합니다.

무선 보안 설정에 대한 자세한 내용은 5.2절. “802.1X 보안 설정” 을 참조하십시오.

새 Wi-Fi 연결 구성

절차

1. **Settings** (설정)의 **Wi-Fi** 메뉴 항목을 선택합니다.
2. 연결할 Wi-Fi 연결 이름을 클릭합니다(기본적으로 SSID와 동일합니다).
   - SSID 범위가 아닌 경우 자세한 내용은 “숨겨진 Wi-Fi 네트워크에 연결” 을 참조하십시오.
   - SSID 범위가 범위이면 오른쪽 메뉴에서 **Wi-Fi** 연결 프로필을 클릭합니다. 잠금 기호는 키 또는 암호가 필요함을 나타냅니다. 요청한 경우 인증 세부 정보를 입력합니다.

기존 Wi-Fi 연결 편집

과거 연결에서 시도하거나 성공한 기존 연결을 편집할 수 있습니다.

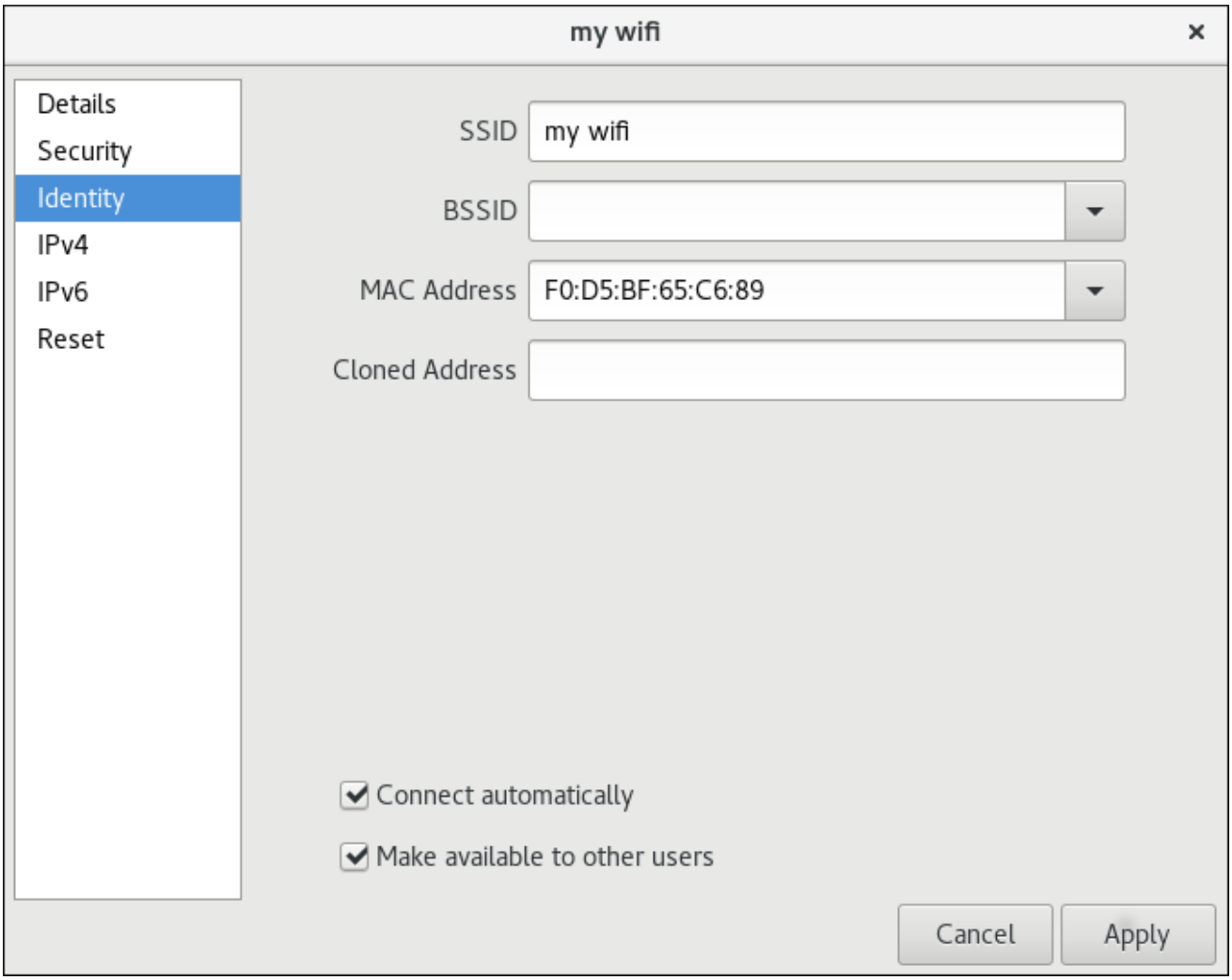
절차

1. **Super** 키를 눌러 Activities Overview(활동 개요)를 입력하고 **Settings** (설정)를 **입력하고 Enter** 키를 누릅니다.
2. 왼쪽 메뉴 항목에서 **Wi-Fi** 를 선택합니다.
3. 편집할 Wi-Fi 연결 이름의 오른쪽에 있는 기어 wheel 아이콘을 선택하면 Edit connection(연결 편집) 대화 상자가 나타납니다. 네트워크가 현재 범위가 아닌 경우 **History** (기록)를 클릭하여 과거 연결을 표시합니다. **Details(세부 정보)** 창에는 연결 세부 정보가 표시됩니다.

Wi-Fi 연결에 대한 기본 설정 옵션

Wi-Fi 연결 설정을 편집하려면 연결 편집 대화 상자에서 **Identity(ID)**를 선택합니다. 다음 설정을 사용할 수 있습니다.

그림 3.13. Wi-Fi 연결에 대한 기본 설정 옵션



SSID

액세스 지점(AP)의SSID(Service *Set Identifier*)입니다.

BSSID

기본 서비스 세트 식별자 (BSSID)는 **인프라** 모드에서 연결할 때 연결하는 특정 무선 액세스 지점의 *하드웨어 주소* 라고도 하는 MAC 주소입니다. 이 필드는 기본적으로 비어 있으며 **BSSID** 를 지정하지 않고도 무선 액세스 지점에 **SSID** 로 연결할 수 있습니다. BSSID를 지정하면 시스템이 특정 액세스 지점에만 연결되도록 강제 적용합니다.

애드혹 네트워크의 경우 애드혹 네트워크가 생성될 때 **mac80211** 하위 시스템에서 **BSSID** 가 무작위로 생성됩니다. **NetworkManager**에 의해 표시되지 않습니다

MAC 주소

사용할 Wi-Fi 인터페이스의 *하드웨어 주소* 라고도 하는 MAC 주소를 선택합니다.

단일 시스템에 하나 이상의 무선 네트워크 어댑터가 연결되어 있을 수 있습니다. 따라서 **MAC 주소** 필드를 사용하면 특정 무선 어댑터를 특정 연결(또는 연결)과 연결할 수 있습니다.

복제된 주소

실제 하드웨어 주소 대신 사용할 복제된 MAC 주소입니다. 필요하지 않은 경우 비워 두십시오.

다음 설정은 대부분의 연결 유형에서 일반적입니다.

- **자동으로 연결** - NetworkManager 가 이 연결을 사용할 수 있을 때 자동으로 연결되도록 하려면 이 상자를 선택합니다. 자세한 내용은 "제어 센터를 사용하여 기존 연결 편집" 을 참조하십시오.
- **다른 사용자가 사용할 수 있도록 설정** - 이 상자를 선택하여 시스템의 모든 사용자가 사용할 수 있는 연결을 만듭니다. 이 설정을 변경하려면 root 권한이 필요할 수 있습니다. 자세한 내용은 3.4.5절. "GUI를 사용하여 시스템 전체 및 개인 연결 프로파일 관리" 을 참조하십시오.

추가 Wi-Fi 구성 만들기
편집 대화 상자에서 기존 연결을 추가로 구성할 수 있습니다.

구성하려면 다음을 수행합니다.

- 무선 연결에 대한 **보안 인증**, 보안을 클릭한 다음 5.2절. "802.1X 보안 설정" **진행합니다**.
- 연결의 **IPv4** 설정, **IPv4** 를 클릭하고 계속 진행 5.4절. "IPv4 설정 구성"

  또는

- 연결의 **IPv6** 설정, **IPv6** 를 클릭하고 5.5절. "IPv6 설정 구성" 진행합니다.

새 연결(또는 수정된 연결 저장)
무선 연결 편집을 마치면 **Apply** (적용) 버튼을 클릭하여 구성을 저장합니다. 올바른 구성이 제공된 경우 네트워크 연결 아이콘의 메뉴에서 이를 선택하여 수정된 연결에 연결할 수 있습니다. 네트워크 선택 및 연결에 대한 자세한 내용은 3.4.1절. "**제어 센터 GUI를 사용하여 네트워크에** 연결" 을 참조하십시오.

3.4.8. GUI를 사용하여 VPN 연결 구성

Libreswan 에서 제공하는 **IPsec** 은 VPN을 생성하는 데 선호되는 방법입니다. **Libreswan** 은 VPN을 위한 오픈 소스 사용자 공간 **IPsec** 구현입니다. 명령줄을 사용하여 **IPsec** VPN 구성은 *Red Hat Enterprise Linux 7 보안 가이드에* 설명되어 있습니다.

3.4.8.1. 제어 센터를 사용하여 VPN 연결 설정

Libreswan 에서 제공하는 **IPsec** 은 Red Hat Enterprise Linux 7에서 VPN을 생성하는 데 선호되는 방법입니다. 자세한 내용은 3.4.8절. "GUI를 사용하여 VPN 연결 구성" 의 내용을 참조하십시오.

아래에 설명된 GNOME 그래픽 사용자 인터페이스 도구에는 NetworkManager-libreswan-gnome 패키지가 필요합니다. 패키지를 설치하려면 **root** 로 다음 명령을 실행하십시오.

```
~]# yum install NetworkManager-libreswan-gnome
```

Red Hat Enterprise Linux에서 새 패키지를 설치하는 방법에 대한 자세한 내용은 Red Hat Enterprise Linux 시스템 관리자 가이드 를 참조하십시오.

VPN(Virtual Private Network)을 설정하면 LAN(Local Area Network)과 다른 원격 LAN 간의 통신이 가능합니다. 이는 인터넷과 같은 중간 네트워크에서 터널을 설정하여 수행됩니다. 설정된 VPN 터널은 일반적으로 인증 및 암호화를 사용합니다. 보안 터널을 사용하여 VPN 연결을 성공적으로 설정한 후 VPN 라우터 또는 게이트웨이는 전송하는 패킷에 대해 다음 작업을 수행합니다.

1. 라우팅 및 인증을 위해 *인증 헤더* 를 추가합니다.

2. 이는 패킷 데이터를 암호화합니다; 및,

3. 암호 해독 및 처리 지침을 구성하는 ESP(Security Payload) 프로토콜에 따라 패킷에 데이터를 포함합니다.

수신 VPN 라우터는 헤더 정보를 제거하고, 데이터를 암호 해독하고, 의도한 대상(네트워크의 워크스테이션 또는 다른 노드)으로 라우팅합니다. 네트워크-네트워크 연결을 사용하여 로컬 네트워크의 수신 노드는 패킷을 이미 암호 해독하고 처리할 준비가 된 패킷을 수신합니다. 따라서 네트워크 간 VPN 연결의 암호화 및 암호 해독 프로세스는 클라이언트에 투명합니다.

여러 인증 및 암호화를 사용하므로 VPN은 통합 인트라넷 역할을 하기 위해 여러 원격 노드를 연결하는 안전하고 효과적인 방법입니다.

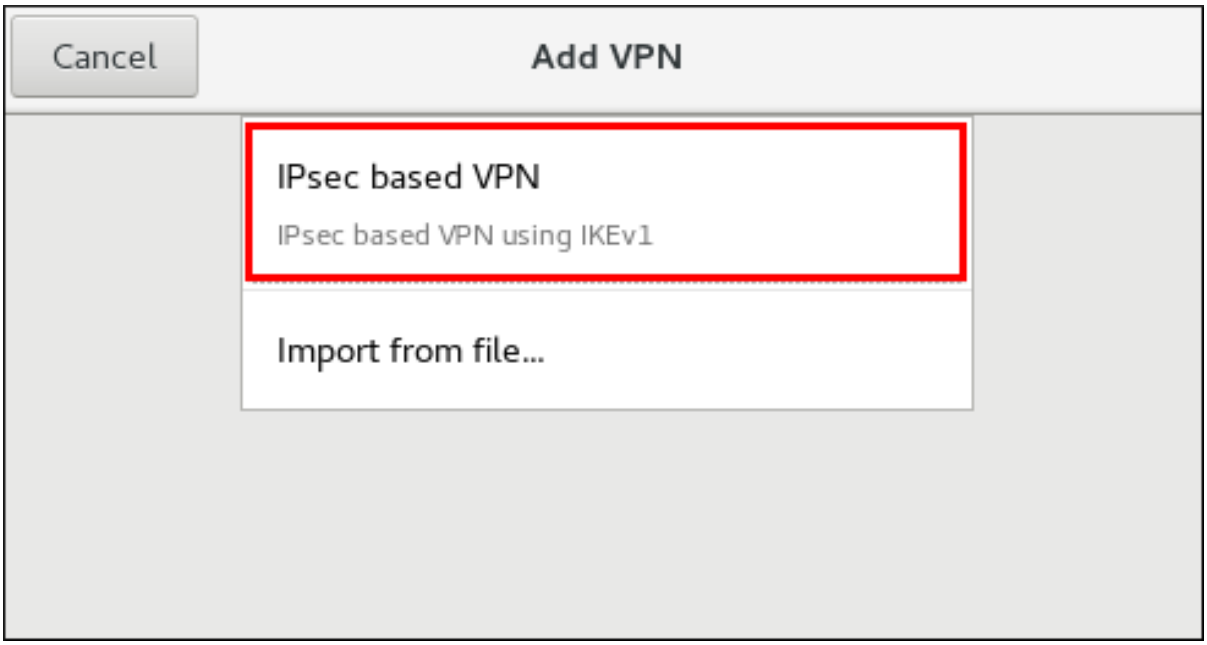
새 IPsec VPN 연결 추가
절차

1. **Super** 키를 눌러 Activities Overview(활동 개요)를 입력하고 **Settings** (설정)를 **입력하고 Enter** 키를 누릅니다. 그런 다음 **Network** 메뉴 항목을 선택하고 **네트워크** 설정 도구가 나타나면 "제어 센터를 사용하여 새 연결 구성" 을 참조하십시오.

2. VPN 항목에서 더하기 버튼을 클릭합니다.

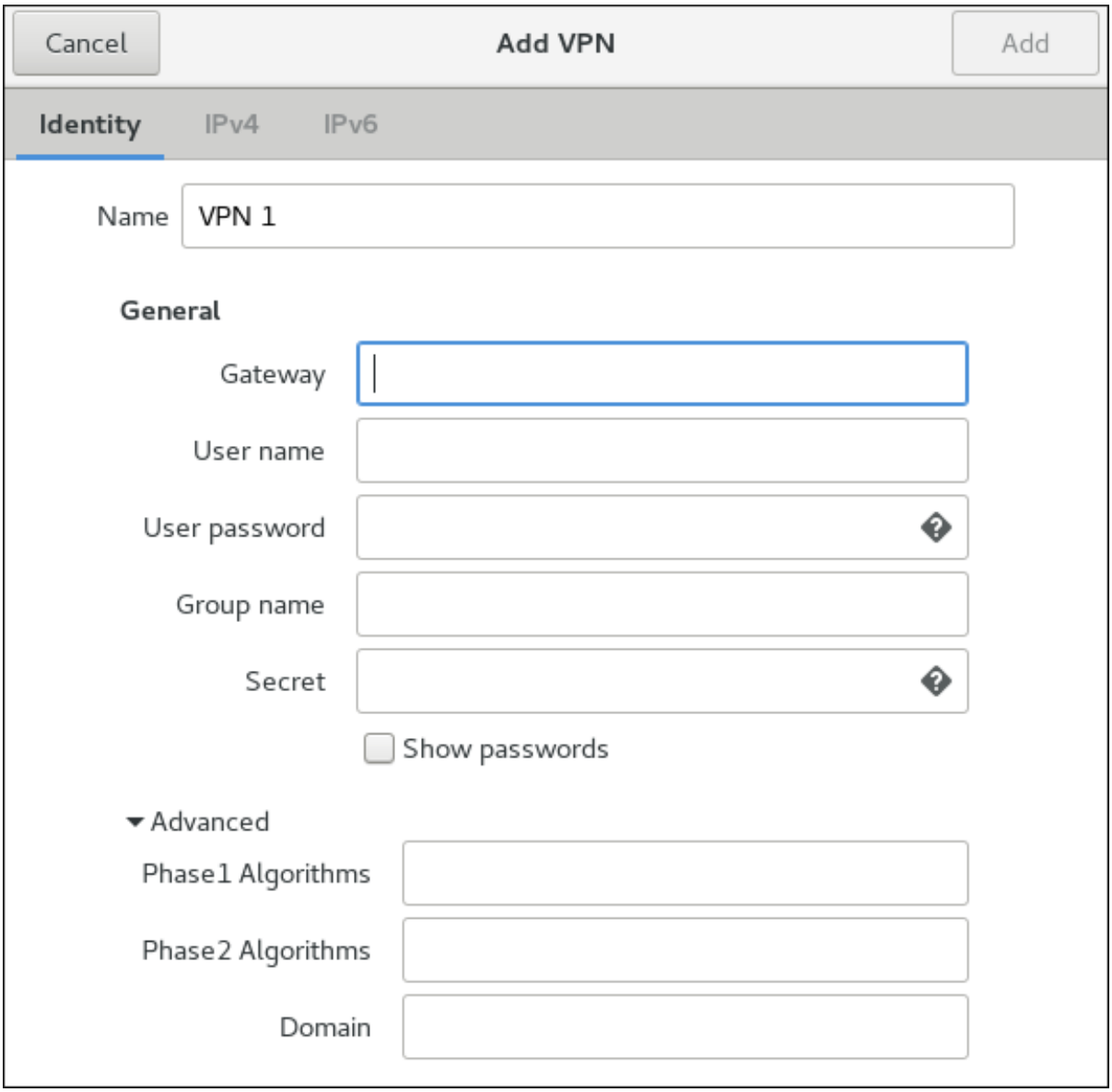
3. **VPN 추가** 창이 나타납니다. 수동으로 구성하려면 **IPsec 기반 VPN** 을 선택합니다.

   그림 3.14. IPsec 모드에서 VPN 구성



4. **ID** 구성 양식에서는 **일반** 및 **고급** 섹션에 필드를 지정할 수 있습니다.

그림 3.15. 일반 및 고급 섹션



- **일반** 섹션에서는 다음을 지정할 수 있습니다.

게이트웨이

원격 VPN 게이트웨이의 이름 또는 **IP** 주소입니다.

사용자 이름

필요한 경우 인증을 위해 VPN 사용자 ID와 연결된 사용자 이름을 입력합니다.

사용자 암호

필요한 경우 인증을 위해 VPN 사용자 ID와 연결된 암호를 입력합니다.

그룹 이름

원격 게이트웨이에 구성된 VPN 그룹의 이름입니다. 비어 있는 경우 기본 Aggressive 모드 대신 IKEv1 주 모드가 사용됩니다.

Secret

사용자 인증 전에 암호화를 초기화하는 데 사용되는 사전 공유 키입니다. 필요한 경우 그룹 이름과 연결된 암호를 입력합니다.

- 다음 구성 설정은 **고급** 섹션에서 사용할 수 있습니다.

1단계 알고리즘

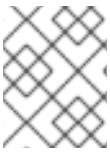
필요한 경우 암호화된 채널을 인증하고 설정하는 데 사용할 알고리즘을 입력합니다.

2단계 알고리즘

필요한 경우 **IPsec** 협상에 사용할 알고리즘을 입력합니다.

도메인

필요한 경우 Domain Name(도메인 이름)을 입력합니다.



참고

NetworkManager 를 사용하지 않고 **IPsec** VPN을 구성하려면 3.4.8절. "GUI를 사용하여 VPN 연결 구성" 을 참조하십시오.

기존 VPN 연결 편집

절차

1. **Super** 키를 눌러 Activities Overview(활동 개요)를 입력하고 **Settings** (설정)를 **입력하고 Enter** 키를 누릅니다. 그런 다음 **Network** 메뉴 항목을 선택하고 **네트워크** 설정 도구가 나타나면 "제어 센터를 사용하여 새 연결 구성" 을 참조하십시오.

2. 편집할 **VPN** 연결을 선택하고 기어 톱니바퀴 아이콘을 클릭하고 **일반** 및 **고급** 섹션을 편집하고 3.4.8.1절. "제어 센터를 사용하여 VPN 연결 설정" 참조하십시오.

새 연결 (또는 수정된) 연결 저장 및 추가 설정 만들기

새 VPN 연결 편집을 마치면 **저장** 버튼을 클릭하여 사용자 지정 구성을 저장합니다. 프로필이 편집하는 동안 사용 중인 경우 **NetworkManager** 에서 변경 사항을 적용하도록 연결의 전원을 켭니다. 프로필이 OFF(꺼짐)인 경우 네트워크 연결 아이콘의 메뉴에서 ON(켜짐)으로 설정하거나 선택합니다. 새로운 연결 또는 변경된 연결 사용에 대한 정보는 3.4.1절. "**제어 센터 GUI를 사용하여 네트워크에** 연결" 을 참조하십시오.

네트워크 창에서 기존 연결을 선택하고 **Configure** (구성)를 클릭하여 **Editing** (편집) 대화 상자로 돌아가 기존 연결을 추가로 구성할 수 있습니다.

그런 다음 다음을 구성하려면 다음을 수행합니다.

- 연결의 **IPv4** 설정을 클릭하고 **IPv4 Settings(IPv4 설정**) 탭을 클릭하고 5.4절. "IPv4 설정 구성" 진행합니다.

3.4.8.2. nm-connection-editor를 사용하여 VPN 연결 구성

nm-connection-editor 를 사용하여 VPN 연결을 추가하고 구성할 수도 있습니다. 이를 위해 다음과 같이 진행하십시오.

절차

1. 터미널에서 **nm-connection-editor** 를 입력합니다. **Network Connections(네트워크 연결**) 창이 표시됩니다. 3.4.3절. "nm-connection-editor를 사용하는 일반적인 구성 옵션" 을 참조하십시오.

2. 더하기 버튼을 클릭합니다. **Choose a Connection Type(연결 유형** 선택) 메뉴가 열립니다.

3. **VPN** 메뉴 항목, **IPsec 기반 VPN 옵션에서** 을 선택합니다.

4. **생성** 을 클릭하여 **편집** 대화 상자를 열고 "새 IPsec VPN 연결 추가" 진행하여 **일반** 및 **고급** 섹션을 편집합니다.

3.4.9. GUI를 사용하여 모바일 광대역 연결 구성

NetworkManager 의 모바일 광대역 연결 기능을 사용하여 다음 *2G 및 3G* 서비스에 연결할 수 있습니다.

- 2G - *GPRS* (*일반 패킷 라디오 서비스*), *EDGE* (GPC *에 대한 향상된 데이터 속도)* 또는 CDMA(코드부 다중 액세스).
- 3G - *UMTS* (*Universal Mobile Telecommunications System*), *HSPA* (*고속 패킷 액세스*) 또는 EVDO(EVolution Data-Only).

연결을 만들려면 컴퓨터에 모바일 광대역 장치(modem)가 있어야 합니다. 이러한 장치는 컴퓨터에 구축될 수도 있고(많은 노트북과 넷북의 경우) 내부 또는 외부 하드웨어로 별도로 제공될 수 있습니다. 예를 들어 PC 카드, USB 모뎀 또는 Dongle, 모바일 또는 태블릿 전화가 모뎀 역할을 할 수 있습니다.

3.4.9.1. nm-connection-editor를 사용하여 모바일 광대역 연결 설정

GNOME **nm-connection-editor** 를 사용하여 모바일 광대역 연결을 구성할 수 있습니다.

새 모바일 광대역 연결 추가
절차

1. 터미널에서 **nm-connection-editor** 를 입력합니다. **Network Connections(네트워크 연결**) 창이 표시됩니다. 3.4.3절. "nm-connection-editor를 사용하는 일반적인 구성 옵션" 을 참조하십시오.

2. 더하기 버튼을 클릭합니다. **Choose a Connection Type(연결 유형** 선택) 메뉴가 열립니다.

3. **Mobile Broadband** 메뉴 항목을 선택합니다.

4. **생성** 을 클릭하여 **Set up a Mobile Broadband Connection** 길잡이를 엽니다.

5. **이 모바일 광대역 장치에 대한 연결 만들기** 에서 연결과 함께 사용할 2G- 또는 3G 가능 장치를 선택합니다. 드롭다운 메뉴가 비활성화된 경우 시스템이 모바일 광대역을 사용할 수 있는 장치를 탐지할 수 없음을 나타냅니다. 이 경우 **Cancel(취소**)을 클릭하고 컴퓨터에서 연결 및 인식할 수 있는 모바일 광대역 장치가 있는지 확인한 다음 이 절차를 다시 시도합니다. **Continue(계속**) 버튼을 클릭합니다.

6. 목록에서 서비스 공급자가 있는 국가를 선택하고 **Continue** (계속) 버튼을 클릭합니다.

7. 목록에서 공급자를 선택하거나 수동으로 입력합니다. **Continue(계속**) 버튼을 클릭합니다.

8. 드롭다운 메뉴에서 결제 계획을 선택하고APN(*액세스 지점 이름*)이 올바른지 확인합니다. **Continue(계속**) 버튼을 클릭합니다.

9. 설정을 검토 및 확인한 다음 **Apply** 단추를 클릭합니다.

10. 다음을 참조하여 모바일 광대역별 설정을 편집합니다. "모바일 광대역 탭 구성"

기존 모바일 광대역 연결 편집
절차

1. 터미널에서 **nm-connection-editor** 를 입력합니다. **Network Connections(네트워크 연결**) 창이 표시됩니다.

2. **모바일 광대역** 탭을 선택합니다.

3. 편집할 연결을 선택하고 톱니바퀴 아이콘을 클릭합니다. 자세한 내용은 3.4.3절. "nm-connection-editor를 사용하는 일반적인 구성 옵션"을 참조하십시오.

4. 다음을 참조하여 모바일 광대역별 설정을 편집합니다. "모바일 광대역 탭 구성"

모바일 광대역 탭 구성
도우미를 사용하여 이미 새로운 모바일 광대역 연결을 추가한 경우(명령 참조) **모바일 광대역** 탭을 편집하여 홈 네트워크를 사용할 수 없는 경우 로밍을 비활성화하거나, 네트워크 ID를 할당하거나, 연결을 사용할 때 **NetworkManager** 에 특정 기술을 선호하도록 지시할 수 있습니다. "새 모바일 광대역 연결 추가"

숫자

GSM 기반 모바일 광대역 네트워크와 PPP 연결을 구축하기 위해 전화하는 번호입니다. 이 필드는 광대역 장치를 초기 설치 중에 자동으로 채울 수 있습니다. 일반적으로 이 필드를 비워 두고 **APN** 을 대신 입력할 수 있습니다.

사용자 이름

네트워크로 인증하는 데 사용되는 사용자 이름을 입력합니다. 일부 공급자는 사용자 이름을 제공하거나 네트워크에 연결할 때 사용자 이름을 허용하지 않습니다.

암호

네트워크 인증에 사용되는 암호를 입력합니다. 일부 공급자는 암호를 제공하지 않거나 암호를 수락합니다.

APN

GSM 기반 *네트워크와의 연결을 설정하는 데 사용되는 액세스 지점 이름* (APN)을 입력합니다. 연결에 올바른 APN을 입력하는 것은 종종 결정되기 때문에 중요합니다.

- 사용자가 네트워크 사용량에 대한 요금을 청구하는 방법
- 사용자가 인터넷, 인트라넷 또는 서브네트워크에 액세스할 수 있는지 여부.

네트워크 ID

네트워크 ID 를 입력하면 **NetworkManager** 는 장치가 특정 네트워크에만 등록되도록 강제 적용합니다. 로밍을 직접 제어할 수 없는 경우 연결이 로밍을 확인하는 데 사용할 수 있습니다.

유형

any - Any 의 기본값은 가장 빠른 네트워크를 선택하도록 모뎀을 남겨 둡니다.

3G(UMTS/HSPA) - 3G 네트워크 기술만 사용하도록 연결을 강제 적용합니다.

2g(GPRS/EDGE) - 연결에서 2G 네트워크 기술만 사용하도록 강제 적용합니다.

3G(UMTS/HSPA) - 먼저 HSPA 또는 UMTS와 같은 3G 기술을 사용하여 연결을 시도하고 실패한 경우에만 GPRS 또는 EDGE로 대체하십시오.

2G (GPRS/EDGE) - 먼저 GPRS 또는 EDGE와 같은 2G 기술을 사용하여 연결을 시도하고 실패하는 경우에만 HSPA 또는 UMTS로 대체하십시오.

홈 네트워크를 사용할 수 없는 경우 로밍 허용

NetworkManager 가 홈 네트워크에서 로밍(roaming)으로 전환하지 않고 연결을 종료하려면 이 상자

를 선택 해제하여 가능한 로밍 요금을 방지합니다. 박스가 선택된 경우 **NetworkManager** 는 홈 네트워크에서 로밍(Raming)으로 전환하여 양호한 연결을 유지하려고 하며 그 반대의 경우도 마찬가지입니다.

PIN

장치의 *SIM* (*구독자 ID 모듈*)이 *PIN* (*개인 식별 번호*)으로 잠겨 있는 경우 **NetworkManager** 가 장치의 잠금을 해제할 수 있도록 PIN을 입력합니다. 장치를 어떠한 용도로든 사용하려면 **NetworkManager** 에서 SIM을 잠금 해제해야 합니다.

CDMA와 EVDO에는 더 적은 수의 옵션이 있습니다. **APN**,**네트워크 ID** 또는 **Type** 옵션이 없습니다.

새 연결 (또는 수정된) 연결 저장 및 추가 설정 만들기
모바일 광대역 연결 편집을 마치면 **Apply** (적용) 버튼을 클릭하여 사용자 지정 구성을 저장합니다. 프로필이 편집하는 동안 사용 중인 경우 **NetworkManager** 에서 변경 사항을 적용하도록 연결의 전원을 켭니다. 프로필이 OFF(꺼짐)인 경우 네트워크 연결 아이콘의 메뉴에서 ON(켜짐)으로 설정하거나 선택합니다. 새로운 연결 또는 변경된 연결 사용에 대한 정보는 3.4.1절. "**제어 센터 GUI를 사용하여 네트워크에** 연결" 을 참조하십시오.

네트워크 연결 창에서 기존 연결을 추가로 선택하고 편집 을 클릭하여 **편집 대화 상자로 돌아갈 수 있습니다.**

그런 다음 다음을 구성하려면 다음을 수행합니다.

- 연결에 대한 **지점 간** 설정, **PPP 설정** 탭을 클릭하고 5.6절. "PPP (Point-to-Point) 설정 구성" 진행합니다.
- 연결의 **IPv4** 설정, **IPv4 설정** 탭을 클릭하고 5.4절. "IPv4 설정 구성"; 또는,
- 연결의 **IPv6** 설정, **IPv6 설정** 탭을 클릭하고 5.5절. "IPv6 설정 구성" 진행합니다.

3.4.10. GUI를 사용하여 DSL 연결 구성

이 섹션은 개인 소비자 또는 SOHO 설치의 일반적인 결합 DSL 모뎀 라우터가 아니라 호스트에 적합한 DSL 카드가 있는 설치를 위한 것입니다.

3.4.10.1. nm-connection-editor를 사용하여 DSL 연결 구성

GNOME **nm-connection-editor** 를 사용하여 DSL 연결을 구성할 수 있습니다.

새 DSL 연결 추가
절차

1. 터미널에서 **nm-connection-editor** 를 입력합니다. **Network Connections(네트워크 연결**) 창이 표시됩니다. 3.4.3절. "nm-connection-editor를 사용하는 일반적인 구성 옵션" 을 참조하십시오.
2. 더하기 버튼을 클릭합니다.
3. **Choose a Connection Type(연결 유형** 선택) 목록이 표시됩니다.
4. **DSL** 을 선택하고 **Create(만들기**) 단추를 누릅니다.
5. **Editing DSL Connection *1*(DSL 연결** 편집 1) 창이 표시됩니다.

기존 DSL 연결 편집

절차

1. 터미널에서 **nm-connection-editor** 를 입력합니다. **Network Connections(네트워크 연결)** 창이 표시됩니다.

2. 편집할 연결을 선택하고 톱니바퀴 아이콘을 클릭합니다. 자세한 내용은 3.4.3절. "nm-connection-editor를 사용하는 일반적인 구성 옵션"을 참조하십시오.

DSL 탭 구성

사용자 이름

서비스 프로바이더로 인증하는 데 사용되는 사용자 이름을 입력합니다.

service

서비스 공급자가 별도로 지시하지 않는 한 비워 두십시오.

암호

서비스 공급자가 제공한 암호를 입력합니다.

새 연결 (또는 수정된) 연결 저장 및 추가 설정 만들기
DSL 연결 편집을 마치면 **Apply** (적용) 버튼을 클릭하여 사용자 지정 구성을 저장합니다. 프로필이 편집하는 동안 사용 중인 경우 **NetworkManager** 에서 변경 사항을 적용하도록 연결의 전원을 켭니다. 프로필이 OFF(꺼짐)인 경우 네트워크 연결 아이콘의 메뉴에서 ON(켜짐)으로 설정하거나 선택합니다. 새로운 연결 또는 변경된 연결 사용에 대한 정보는 3.4.1절. "**제어 센터 GUI를 사용하여 네트워크에 연결**" 을 참조하십시오.

네트워크 연결 창에서 기존 연결을 추가로 선택하고 편집 을 클릭하여 **편집 대화 상자로 돌아갈 수 있습니다.**

구성하려면 다음을 수행합니다.

- **MAC 주소 및 MTU 설정은 Wired** 탭을 클릭하고 "기본 설정 옵션 " 진행합니다.
- 연결에 대한 **지점 간** 설정, **PPP 설정** 탭을 클릭하고 5.6절. "PPP (Point-to-Point) 설정 구성" 진행합니다.
- 연결의 **IPv4** 설정을 클릭하고 **IPv4 Settings(IPv4 설정)** 탭을 클릭하고 5.4절. "IPv4 설정 구성" 진행합니다.

3.5. IFCFG 파일을 사용하여 IP 네트워킹 구성

시스템 관리자는 네트워크 인터페이스를 수동으로 구성하여 **ifcfg** 파일을 편집할 수 있습니다.

인터페이스 구성(ifcfg) 파일은 개별 네트워크 장치에 대한 소프트웨어 인터페이스를 제어합니다. 시스템이 부팅되면 이러한 파일을 사용하여 가져올 인터페이스와 구성 방법을 결정합니다. 이러한 파일의 이름은 일반적으로 **ifcfg-*이름으로*** 지정됩니다. 여기서 접미사 *이름은* 구성 파일이 제어하는 장치의 이름을 나타냅니다. 관례적으로 **ifcfg** 파일의 접미사는 구성 파일 자체의 **DEVICE** 지시문에서 지정한 문자열과 동일합니다.

ifcfg 파일을 사용하여 정적 네트워크 설정을 사용하여 인터페이스 구성

예를 들어 이름이 **enp1s0** 인 인터페이스인 **ifcfg** 파일을 사용하여 정적 네트워크 설정으로 인터페이스를 구성하려면 **/etc/sysconfig/network -scripts/ 디렉터리에 이름이 ifcfg- enp1s0** 인 파일을 만듭니다.

- **IPv4** 구성의 경우

```
DEVICE=enp1s0
BOOTPROTO=none
ONBOOT=yes
PREFIX=24
IPADDR=10.0.1.27
```

- **IPv6** 구성의 경우

```
DEVICE=enp1s0
BOOTPROTO=none
ONBOOT=yes
IPV6INIT=yes
IPV6ADDR=2001:db8::2/48
```

네트워크 또는 브로드캐스트 주소는 **ipcalc** 에 의해 자동으로 계산되므로 지정하지 않아도 됩니다.

추가 **IPv6** ifcfg 구성 옵션은 *nm-settings-ifcfg-rh(5)* 도움말 페이지를 참조하십시오.

중요

Red Hat Enterprise Linux 7에서는 11장. *일관된 네트워크 장치 이름 지정* 에 설명된 대로 네트워크 인터페이스에 대한 명명 규칙이 변경되었습니다. **HWADDR** 지시문을 사용하여 하드웨어 또는 MAC 주소를 지정하면 장치 명명 절차에 영향을 줄 수 있습니다.

ifcfg 파일을 사용하여 동적 네트워크 설정을 사용하여 인터페이스 구성

ifcfg 파일을 사용하여 동적 네트워크 설정으로 *em1* 이라는 인터페이스를 구성하려면 다음을 수행합니다.

1. 이름이 **ifcfg-em1** 인 파일을 **/etc/sysconfig/network-scripts/** 디렉토리에 다음과 같이 만듭니다.

```
DEVICE=em1
BOOTPROTO=dhcp
ONBOOT=yes
```

2. 다른 호스트 이름을 **DHCP** 서버에 보내도록 인터페이스를 구성하려면 **ifcfg** 파일에 다음 행을 추가합니다.

```
DHCP_HOSTNAME=hostname
```

다른 정규화된 도메인 이름(FQDN)을 **DHCP** 서버에 보내도록 인터페이스를 구성하려면 **ifcfg** 파일에 다음 행을 추가합니다.

```
DHCP_FQDN=fully.qualified.domain.name
```

참고

DHCP_HOSTNAME 또는 **DHCP_ FQDN** 중 하나의 지시어만 지정된 **ifcfg** 파일에서 사용해야 합니다. **DHCP_HOSTNAME** 및 **DHCP_FQDN** 이 모두 지정된 경우 후자만 사용됩니다.

3. 특정 **DNS** 서버를 사용하도록 인터페이스를 구성하려면 다음 행을 **ifcfg** 파일에 추가합니다.

```
PEERDNS=no
DNS1=ip-address
DNS2=ip-address
```

여기서 *ip-address* 는 **DNS** 서버의 주소입니다. 그러면 지정된 **DNS** 서버로 네트워크 서비스가 **/etc/resolv.conf** 를 업데이트합니다. 하나의 **DNS** 서버 주소만 필요합니다. 다른 주소는 선택 사항입니다.

4. **ifcfg** 파일에서 정적 경로를 구성하려면 4.5절. "ifcfg 파일에서 정적 경로 구성" 를 참조하십시오.

기본적으로 NetworkManager 는 인터페이스 구성 파일에서 ***BOOTPROTO*** 를 **dhcp** 로 설정하여 자동으로 주소를 가져오도록 프로필이 설정된 경우 **DHCP** 클라이언트 **dhclient** 를 호출합니다. **DHCP** 가 필요한 경우 인터페이스의 모든 인터넷 프로토콜인 **IPv4 및 IPv 6** 에 대해 **dhclient** 인스턴스가 시작됩니다. **NetworkManager** 가 실행 중이 아니거나 인터페이스를 관리하지 않는 경우 레거시 네트워크 서비스는 필요에 따라 **dhclient** 의 인스턴스를 호출합니다. 동적 IP 주소에 대한 자세한 내용은 1.2절. "정적 IP 주소와 동적 IP 주소 비교" 을 참조하십시오.

5. 구성을 적용하려면 다음을 수행합니다.

a. 업데이트된 연결 파일을 다시 로드합니다.

```
# nmcli connection reload
```

b. 연결을 다시 활성화합니다.

```
# nmcli connection up connection_name
```

3.5.1. ifcfg 파일을 사용하여 시스템 전체 및 개인 연결 프로필 관리

권한은 **ifcfg** 파일의 **USERS** 지시문에 해당합니다. **USERS** 지시문이 없으면 모든 사용자가 네트워크 프로필을 사용할 수 있습니다. 예를 들어 **ifcfg** 파일의 다음 명령은 나열된 사용자만 연결을 사용할 수 있도록 합니다.

```
USERS="joe bob alice"
```

또한 **USERCTL** 지시문을 설정하여 장치를 관리할 수 있습니다.

- **yes** 를 설정하면 **root** 가 아닌 사용자가 이 장치를 제어할 수 있습니다.
- **no** 를 설정하면 **root** 가 **아닌** 사용자가 이 장치를 제어할 수 없습니다.

3.6. IP 명령을 사용하여 IP 네트워킹 구성

시스템 관리자는 **ip** 명령을 사용하여 네트워크 인터페이스를 구성할 수 있지만, 재부팅 시 변경 사항은 유지되지 않습니다. 재부팅하면 변경 사항이 손실됩니다.

종종 업스트림 패키지 이름 뒤에 **ip** route2라고 하는 ip 유틸리티의 명령은 **man ip(8)** 페이지에 설명되어 있습니다. Red Hat Enterprise Linux 7의 패키지 이름은 iproute 입니다. 필요한 경우 다음과 같이 해당 버전 번호를 확인하여 **ip** 유틸리티가 설치되었는지 확인할 수 있습니다.

```
~]$ ip -V
ip utility, iproute2-ss130716
```

ip 명령은 **NetworkManager** 와 동시에 인터페이스에 대한 주소와 경로를 추가하고 제거하는 데 사용할 수 있습니다. 이 명령은 해당 명령을 보존하고 **nmcli,nmtui,control-center** 및 D-Bus API에서 인식합니다.

인터페이스를 중단하려면 다음을 수행합니다.

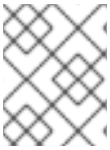
```
ip link set ifname down
```

참고

ip link set *ifname* 명령은 **IFF_UP** 상태에서 네트워크 인터페이스를 설정하고 커널의 범위에서 활성화합니다. 이는 장치의 initscripts 또는 **NetworkManager** 의 활성화 상태에 대한 ***ifup ifname*** 명령과 다릅니다. 실제로 **NetworkManager** 는 현재 연결이 끊어져도 인터페이스를 항상 설정합니다. **nmcli** 도구를 통해 장치 연결을 끊으면 **IFF_UP** 플래그가 제거되지 않습니다. 이러한 방식으로 **NetworkManager** 는 캐리어 상태에 대한 알림을 받습니다.

net-tools 패키지(**ifconfig**)는 InfiniBand 주소를 지원하지 않기 때문에 **ip** 유틸리티는 **ifconfig**유틸리티를 대체합니다.

사용 가능한 OBJECT에 대한 자세한 내용은 **ip help** 명령을 사용하십시오. 예: **ip link help** 및 **ip addr help**.



참고

시스템을 다시 시작한 후에는 명령줄에 지정된 IP **명령이** 유지되지 않습니다. 지속성이 필요한 경우 구성 파일(ifcfg**파일**)을 사용하거나 스크립트에 명령을 추가합니다.

각 작업에 대해 명령줄 및 구성 파일을 사용하는 예는 **nmtui** 및 **nmcli** 예제 뒤에 포함되지만 그래픽 사용자 인터페이스를 **NetworkManager** 에 설명하기 전에 **control-center 및 nm-connection- editor** 에 대한 그래픽 사용자 인터페이스를 설명합니다.

ip 유틸리티를 사용하여 다음 형식을 사용하여 인터페이스에 **IP** 주소를 할당할 수 있습니다.

```
ip addr [ add | del ] address dev ifname
```

ip 명령을 사용하여 정적 주소 할당

IP 주소를 인터페이스에 할당하려면 다음을 수행합니다.

```
~]# ip address add 10.0.0.3/24 dev enp1s0
You can view the address assignment of a specific device:
~]# ip addr show dev enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen
1000
    link/ether f0:de:f1:7b:6e:5f brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.3/24 brd 10.0.0.255 scope global global enp1s0
       valid_lft 58682sec preferred_lft 58682sec
    inet6 fe80::f2de:f1ff:fe7b:6e5f/64 scope link
       valid_lft forever preferred_lft forever
```

추가 예제 및 명령 옵션은 **ip-address(8)** 도움말 페이지에서 찾을 수 있습니다.

ip 명령을 사용하여 여러 주소 구성

ip 유틸리티에서 여러 주소를 동일한 인터페이스에 할당하는 기능을 지원하므로 더 이상 여러 주소를 동일한 인터페이스에 바인딩하는 별칭 인터페이스 방법을 사용할 필요가 없습니다. 주소를 할당하는 **ip** 명령은 여러 주소를 할당하기 위해 여러 번 반복될 수 있습니다. 예를 들어 다음과 같습니다.

```
~]# ip address add 192.168.2.223/24 dev enp1s0
~]# ip address add 192.168.4.223/24 dev enp1s0
~]# ip addr
3: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen
1000
    link/ether 52:54:00:fb:77:9e brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.223/24 scope global enp1s0
    inet 192.168.4.223/24 scope global enp1s0
```

ip 유틸리티 명령에 대한 자세한 내용은 ip(8) 매뉴얼 페이지를 참조하십시오.

참고

시스템을 다시 시작한 후에는 명령줄에 지정된 IP **명령이 유지되지 않습니다**.

3.7. 커널 명령줄에서 IP 네트워킹 구성

인터페이스에서 iSCSI 대상의 루트 파일 시스템에 연결할 때 네트워크 설정은 설치된 시스템에 구성되지 않습니다. 이 문제의 해결 방법:

1. **dracut** 유틸리티를 설치합니다. **dracut** 사용에 대한 자세한 내용은 *Red Hat Enterprise Linux 시스템 관리자 가이드를 참조하십시오*.

2. 커널 명령행에서 **ip** 옵션을 사용하여 설정을 설정합니다.

   ```
   ip<client-IP-number>:[<server-id>]:<gateway-IP-number>:<netmask>:<client-hostname>:
   <interface>:{dhcp|dhcp6|auto6|on|any|none|off}
   ```

   - **DHCP** - DHCP 구성
   - **dhpc6** - DHCP IPv6 구성
   - **auto6** - 자동 IPv6 설정
   - **on,any** protocol - 커널에서 사용 가능한 모든 프로토콜 (기본값)
   - **none,off** - 자동 구성 없음, 정적 네트워크 구성 없음

   예를 들어 다음과 같습니다.

   ```
   ip=192.168.180.120:192.168.180.100:192.168.180.1:255.255.255.0::enp1s0:off
   ```

3. 이름 서버 구성을 설정합니다.

   ```
   nameserver=srv1 [nameserver=srv2 [nameserver=srv3 […]]]
   ```

dracut 유틸리티는 네트워크 연결을 설정하고 **/etc/sysconfig/network-scripts/** 파일에 복사할 수 있는 새 **ifcfg** 파일을 생성합니다.

3.8. IGMP를 사용하여 IP 멀티캐스트 활성화

관리자는 IGMP(Internet Group Management Protocol)를 사용하여 네트워크, 호스트 및 라우터 간 트래픽을 멀티캐스트로 라우팅 및 서브스크립션을 관리할 수 있습니다. Red Hat Enterprise Linux의 커널은 IGMPv3을 지원합니다.

멀티 캐스트 정보를 표시하려면 **ip maddr show** 하위 명령을 사용하십시오. 예를 들면 다음과 같습니다.

```
~]$ ip maddr show dev br0
8: br0
 inet  224.0.0.1
 inet6 ff02::1
 inet6 ff01::1
[output truncated]
```

또는 **ip link show** 명령 출력에서 **MULTICAST** 문자열을 찾습니다. 예를 들면 다음과 같습니다.

```
~]$ ip link show br0
8: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode
DEFAULT qlen 1000
    link/ether 6c:0b:84:67:fe:63 brd ff:ff:ff:ff:ff:ff
```

장치에서 멀티 캐스트를 비활성화하고 *br0* 장치에서 멀티 캐스트가 비활성화되어 있는지 확인하려면 다음을 수행합니다.

```
~]# ip link set multicast off dev br0
~]$ ip link show br0
8: br0: <BROADCAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT qlen
1000
    link/ether 6c:0b:84:67:fe:63 brd ff:ff:ff:ff:ff:ff
```

누락된 **MULTICAST** 문자열은 멀티 캐스트가 비활성화되어 있음을 나타냅니다.

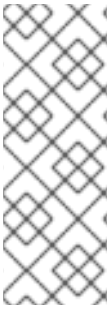
br0 장치에서 멀티 캐스트를 활성화하고 활성화되어 있는지 확인하려면 다음을 수행하십시오.

```
~]# ip link set multicast on dev br0
~]$ ip link show br0
8: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode
DEFAULT qlen 1000
    link/ether 6c:0b:84:67:fe:63 brd ff:ff:ff:ff:ff:ff
```

자세한 내용은 ip Command Cheat Sheet for Red Hat Enterprise Linux 문서 및 **ip(8)** 매뉴얼 페이지를 참조하십시오.

멀티캐스팅에 가입된 IGMP 및 IP 주소의 현재 버전을 확인하려면 **/proc/net/igmp** 파일을 참조하십시오.

```
~]$ cat /proc/net/igmp
```



참고

기본적으로 IGMP는 **firewalld** 에서 활성화되지 않습니다. 영역에 IGMP를 활성화하려면 다음을 수행합니다.

```
~]# firewall-cmd --zone=zone-name --add-protocol=igmp
```

자세한 내용은 Red Hat Enterprise Linux 보안 가이드 의 방화벽 사용 장을 참조하십시오.

3.9. 추가 리소스

설치된 문서

- *ip(8)* 도움말 페이지 - **ip** 유틸리티의 명령 구문을 설명합니다.
- *nmcli(1)* 도움말 페이지 - **NetworkManager** 의 명령줄 도구를 설명합니다.
- *nmcli-examples(5)* 도움말 페이지 - **nmcli** 명령의 예를 표시합니다.
- *nm-settings(5)* 도움말 페이지 - **NetworkManager** 속성 및 해당 설정을 설명합니다.
- *nm-settings-ifcfg-rh(5)* 도움말 페이지 - **ifcfg-rh** 설정 플러그인 설명.

온라인 문서

Red Hat Enterprise Linux 7 보안 가이드

IPsec 기반 VPN 및 해당 구성을 설명합니다. DNSSEC를 사용한 인증된 **DNS** 쿼리 사용에 대해 설명합니다.

RFC 1518 - CIDR(Classless Inter-Domain Routing)

CIDR 주소 할당 및 집계 전략을 설명합니다(변수 서브넷 포함).

RFC 1918 - 사설 인터넷의 주소 할당

는 개인용으로 예약된 **IPv4** 주소 범위를 설명합니다.

RFC 3330 - 특수 사용 IPv4 주소 사용

IANA(Internet Assigned Numbers Authority)에서 할당한 글로벌 및 기타 특수 **IPv4** 주소 블록을 설명합니다.

4장. 정적 경로 및 기본 게이트웨이 구성

이 장에서는 정적 경로 및 기본 게이트웨이의 구성에 대해 설명합니다.

4.1. 라우팅 및 게이트웨이 이해 소개

라우팅은 시스템이 다른 시스템에 대한 네트워크 경로를 찾을 수 있도록 하는 메커니즘입니다. 라우팅은 종종 라우팅 전용 네트워크의 장치에서 처리합니다(단, 라우팅을 수행하도록 모든 장치를 구성할 수 있음). 따라서 Red Hat Enterprise Linux 서버 또는 클라이언트에서 정적 경로를 설정할 필요가 없는 경우가 많습니다. 단, 암호화된 VPN 터널을 통과해야 하는 트래픽 또는 비용이나 보안상의 이유로 특정 경로를 사용해야 하는 트래픽도 예외입니다. 호스트의 라우팅 테이블은 직접 연결된 네트워크에 대한 경로가 자동으로 채워집니다. 경로는 네트워크 인터페이스가 "작동" 될 때 검사됩니다. 원격 네트워크 또는 호스트에 액세스하기 위해 시스템에 트래픽이 전송되어야 하는 게이트웨이의 주소가 지정됩니다.

호스트 인터페이스를 **DHCP** 에서 구성하면 업스트림 네트워크 또는 인터넷으로 이어지는 게이트웨이의 주소가 일반적으로 할당됩니다. 이 게이트웨이는 일반적으로 시스템에 더 나은 경로를 알려져 있지 않고 라우팅 테이블에 있는 경우 사용할 게이트웨이이므로 기본 게이트웨이라고 합니다. 네트워크 관리자는 종종 네트워크의 첫 번째 또는 마지막 호스트 **IP** 주소를 게이트웨이 주소로 사용합니다(예: **192.168.10.1** 또는 **192.168.10.254)**. 네트워크 자체를 나타내는 주소(이 예에서는 **192.168.10.0** 또는 서브넷의 브로드캐스트 주소)와 혼동되는 것이 아닙니다(이 예에서는 **192.168.10.255)**. 기본 게이트웨이는 전통적으로 네트워크 라우터입니다. 기본 게이트웨이는 로컬 네트워크로 향하지 않고 라우팅 테이블에 기본 경로가 지정되지 않은 모든 트래픽에 대한 것입니다.

참고

전문 지식을 확대하려면 Red Hat System Administration I(RH124) 교육 과정에 관심을 가질 수도 있습니다.

4.2. NMCLI를 사용하여 정적 경로 구성

nmcli 도구를 사용하여 정적 경로를 구성하려면 다음 중 하나를 사용합니다.

- nmcli 명령줄
- nmcli 대화형 편집기

예 4.1. nmcli를 사용하여 정적 경로 구성

명령줄을 사용하여 기존 이더넷 연결에 대한 정적 경로를 구성하려면 다음을 수행합니다.

```
~]# nmcli connection modify enp1s0 +ipv4.routes "192.168.122.0/24 10.10.10.1"
```

192.168.122.0/24 서브넷의 트래픽을 **10.10.10.1의 게이트웨이로 전달합니다.**

예 4.2. nmcli Editor를 사용하여 정적 경로 구성

대화형 편집기를 사용하여 이더넷 연결에 대한 정적 경로를 구성하려면 다음을 수행합니다.

```
~]$ nmcli con edit ens3
===| nmcli interactive connection editor |===

Editing existing '802-3-ethernet' connection: 'ens3'
```

```
Type 'help' or '?' for available commands.
Type 'describe [<setting>.<prop>]' for detailed property description.

You may edit the following settings: connection, 802-3-ethernet (ethernet), 802-1x, dcb, ipv4, ipv6,
tc, proxy
nmcli> set ipv4.routes 192.168.122.0/24 10.10.10.1
nmcli> save persistent
Connection 'ens3' (23f8b65a-8f3d-41a0-a525-e3bc93be83b8) successfully updated.
nmcli> quit
```

4.3. GUI를 사용하여 정적 경로 구성

정적 경로를 설정하려면 구성하려는 연결의 **IPv4 또는 IPv6** 설정 창을 엽니다. 이 작업을 수행하는 방법에 대한 지침은 3.4.1절. "**제어 센터 GUI를 사용하여 네트워크에** 연결" 을 참조하십시오.

라우트

address - 원격 네트워크, sub-net 또는 호스트의 **IP** 주소를 입력합니다.

netmask - 위에 입력한 **IP** 주소의 넷마스크 또는 접두사 길이입니다.

게이트웨이 - 원격 네트워크, sub-net 또는 위에 입력한 호스트로 향하는 게이트웨이의 **IP** 주소입니다.

지표 - 네트워크 비용, 이 경로에 제공하는 기본 설정 값입니다. 더 낮은 값이 더 높은 값보다 우선합니다.

자동

Automatic(자동)이 **ON** 이면 **RA** 또는 **DHCP** 의 경로가 사용되지만, 정적 경로를 추가할 수도 있습니다. **OFF** (끄기)인 경우 정의한 정적 경로만 사용됩니다.

이 연결을 네트워크의 리소스에만 사용합니다

연결이 기본 경로가 되지 않도록 하려면 이 확인란을 선택합니다. 일반적인 예는 연결이 VPN 터널 또는 헤드 사무소에 임대 라인인 경우이며 인터넷 바운드 트래픽이 연결을 통과하지 못하도록 하는 경우가 있습니다. 이 옵션을 선택하면 연결을 통해 자동으로 학습되거나 여기에 입력된 경로로 향하는 트래픽만 연결을 통해 라우팅됩니다.

4.4. IP 명령을 사용하여 정적 경로 구성

시스템 관리자는 **ip route** 명령을 사용하여 정적 경로를 구성할 수 있습니다.

IP 라우팅 테이블을 표시하려면 **ip route** 명령을 사용합니다. 예를 들어 다음과 같습니다.

```
~]$ ip route
default via 192.168.122.1 dev ens9  proto static  metric 1024
192.168.122.0/24 dev ens9  proto kernel  scope link  src 192.168.122.107
192.168.122.0/24 dev enp1s0  proto kernel  scope link  src 192.168.122.126
```

ip route 명령은 다음 형식을 취합니다.

```
ip route [ add | del | change | append | replace ] destination-address
```

옵션과 형식에 대한 자세한 내용은 **ip-route(8)** 도움말 페이지를 참조하십시오.

호스트 주소에 정적 경로를 추가하려면 단일 IP 주소에 다음을 실행합니다.

```
~]# ip route add 192.0.2.1 via 10.0.0.1 [dev interface]
```

여기서 *192.0.2.1*은 점으로 인한 호스트의 **IP** 주소이며, *10.0.0.1*은 다음 홉 주소가고 *인터페이스*는 다음 홉으로 이어지는 종료 인터페이스입니다.

정적 경로를 네트워크에 추가하려면 **IP 주소 범위를 나타내는 IP 주소에 다음을 실행합니다.**

```
~]# ip route add 192.0.2.0/24 via 10.0.0.1 [dev interface]
```

여기서 *192.0.2.0* 은 점으로 된 10진수 표기법에서 대상 네트워크의 IP 주소이고 */24* 는 네트워크 접두사입니다. 네트워크 접두사는 서브넷 마스크에서 활성화된 비트 수입니다. 이 형식의 네트워크 주소 슬래시 네트워크 접두사 길이는CIDR(Classless *inter-domain routing*) 표기법이라고 합니다.

할당된 정적 경로를 제거하려면 다음을 수행합니다.

```
~]# ip route del 192.0.2.1
```

ip route 를 사용하여 라우팅 테이블을 변경한 사항은 시스템 재부팅 시 유지되지 않습니다. 정적 경로를 영구적으로 구성하려면 인터페이스의 /etc/sysconfig/network -scripts/ 디렉터리에 route- *interface* 파일을 만들어 해당 경로를 구성할 수 있습니다. 예를 들어 *enp1s0* 인터페이스에 대한 정적 경로는 /etc/sysconfig/network-scripts/route-*enp1s0* 파일에 저장됩니다. 경로*인터페이스 파일에 대한 변경 사항은 네트워크 서비스 또는 인터페이스*를 다시 시작할 때까지 적용되지 않습니다. route-*interface* *파일에*는 다음 두 가지 형식이 있습니다.

- **IP 명령 인수를 참조하십시오. "IP 명령 인수 형식을 사용하는 정적 경로" 의 내용을 참조하십시오.**

  및

- **네트워크/넷마스크 지시문. "네트워크/넷마스크 지시문 형식을 사용하는 정적 경로" 을 참조하십시오.**

ip route 명령에 대한 자세한 내용은 **ip-route(8)** 도움말 페이지를 참조하십시오.

4.5. IFCFG 파일에서 정적 경로 구성

시스템이 종료되거나 다시 시작되면 명령 프롬프트에서 **ip** 명령을 사용하여 설정된 정적 경로가 손실됩니다. 시스템을 다시 시작한 후 정적 경로를 영구적으로 구성하려면 **/etc/sysconfig/network-scripts/** 디렉토리의 인터페이스별 구성 파일에 배치해야 합니다. 파일 이름은 **route-***인터페이스* 형식이어야 합니다. 구성 파일에서 사용할 명령 유형은 다음 두 가지입니다.

IP 명령 인수 형식을 사용하는 정적 경로

인터페이스별 구성 파일에 필요한 경우(예: **/etc/sysconfig/network-scripts/route-enp1s0**) 첫 번째 행의 기본 게이트웨이로 경로를 정의합니다. 이 작업은 게이트웨이가 **DHCP** 를 통해 설정되지 않고 **/etc/sysconfig/network** 파일에 전역적으로 설정되지 않은 경우에만 필요합니다.

```
default via 192.168.1.1 dev interface
```

여기서 *192.168.1.1* 은 기본 게이트웨이의 **IP** 주소입니다. *인터페이스* 는 기본 게이트웨이에 연결되거나 에 연결할 수 있는 인터페이스입니다. **dev** 옵션은 생략할 수 있으며 선택 사항입니다. 이 설정은 **/etc/sysconfig/network** 파일의 설정보다 우선합니다.

원격 네트워크에 대한 경로가 필요한 경우 다음과 같이 정적 경로를 지정할 수 있습니다. 각 행은 개별 경로로 구문 분석됩니다.

```
10.10.10.0/24 via 192.168.1.1 [dev interface]
```

여기서 *10.10.10.0/24* 는 원격 또는 대상 네트워크의 네트워크 주소 및 접두사 길이입니다. 주소 *192.168.1.1* 은 원격 네트워크로 향하는 **IP** 주소입니다. 가급적 *다음 홉 주소* 이지만 종료 인터페이스의 주소가 작동합니다. "다음 홉" 은 링크의 원격 끝(예: 게이트웨이 또는 라우터)을 의미합니다. **dev** 옵션은 종료 인터페이스 *인터페이스*를 지정하는 데 사용할 수 있지만 필수는 아닙니다. 필요한 만큼 정적 경로를 추가합니다.

다음은 **ip** 명령 인수 형식을 사용하는 경로*인터페이스* 파일의 예입니다. 기본 게이트웨이는 **192.168.0.1, interface**입니다. **enp1s0** 리스드 라인 또는 **WAN** 연결은 **192.168.0.10** 에서 사용할 수 있습니다. 두 개의 정적 경로는 **10.10.10.0/24** 네트워크 및 **172.16.1.10/32** 호스트에 도달하기 위한 것입니다.

```
default via 192.168.0.1 dev enp1s0
10.10.10.0/24 via 192.168.0.10 dev enp1s0
172.16.1.10/32 via 192.168.0.10 dev enp1s0
```

위 예제에서 **local 192.168.0.0/24** 네트워크로 이동하는 패킷은 해당 네트워크에 연결된 인터페이스를 전달하게 됩니다. **10.10.10.0/24** 네트워크 및 **172.16.1.10/32** 호스트로 이동하는 패킷은 **192.168.0.10** 으로 이동합니다. 알 수 없는 원격 네트워크에 대한 패킷은 기본 게이트웨이를 사용하므로 기본 경로가 적합하지 않은 경우에만 원격 네트워크 또는 호스트에 대해 정적 경로를 구성해야 합니다. 이 컨텍스트의 원격 연결은 시스템에 직접 연결되지 않은 네트워크 또는 호스트를 의미합니다.

IPv6 구성의 경우 **ip** 경로 형식 의 **route6-*interface*** 파일의 예는 다음과 같습니다.

```
2001:db8:1::/48 via 2001:db8::1 metric 2048
2001:db8:2::/48
```

종료 인터페이스 지정은 선택 사항입니다. 트래픽을 특정 인터페이스에서 강제로 중단하려는 경우 유용할 수 있습니다. 예를 들어 **VPN**의 경우 원격 네트워크로의 트래픽을 강제로 통과하여 전달할 수 있습니다. **tun0** 인터페이스가 대상 네트워크와 다른 서브넷에 있는 경우에도 인터페이스입니다.

ip 경로 형식은 소스 주소를 지정하는 데 사용할 수 있습니다. 예를 들어 다음과 같습니다.

```
10.10.10.0/24 via 192.168.0.10 src 192.168.0.2
```

여러 라우팅 테이블을 지정하는 기존 정책 기반 라우팅 구성을 정의하려면 4.5.1절. “정책 라우팅 이해” 을 참조하십시오.

중요

기본 게이트웨이가 **DHCP** 에서 이미 할당되어 동일한 지표가 있는 동일한 게이트웨이가 구성 파일에 지정된 경우 시작 중에 오류가 발생하거나 인터페이스를 시작할 때 오류가 발생합니다. 다음 오류 메시지가 표시될 수 있습니다. **"RTNET"** 답변: 파일이 존재합니다. 이 오류는 무시될 수 있습니다.

네트워크/넷마스크 지시문 형식을 사용하는 정적 경로

경로*인터페이스* 파일에 네트워크/넷마스크 지시문 형식을 사용할 수도 있습니다. 다음은 네트워크/넷마스크 형식에 대한 템플릿이며 다음에 나오는 지침이 있습니다.

```
ADDRESS0=10.10.10.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.1.1
```

- ADDRESS0=*10.10.10.0* 은 연결할 원격 네트워크 또는 호스트의 네트워크 주소입니다.
- NETMASK0=*255.255.255.0* 은 ADDRESS0=*10.10.10.0* 으로 정의된 네트워크 주소의 넷마스크입니다.
- GATEWAY0=*192.168.1.1* 은 기본 게이트웨이이거나 ADDRESS0=*10.10.10.0*에 *도달하는 데 사용할 수 있는 IP 주소입니다.*

다음은 network/netmask 지시문 형식을 사용하는 경로*인터페이스* 파일의 예입니다. 기본 게이트웨이는 192.168.0.1 이지만 임대된 라인 또는 WAN 연결은 192.168.0.10 에서 사용할 수 있습니다. 두 개의 정적 경로는 10.10.10.0/24 및 172.16.1.0/24 네트워크에 도달하기 위한 것입니다.

```
ADDRESS0=10.10.10.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.0.10
ADDRESS1=172.16.1.10
NETMASK1=255.255.255.0
GATEWAY1=192.168.0.10
```

이후 정적 경로는 순차적으로 번호를 지정해야 하며 어떤 값도 건너뛸 수 없습니다. 예를 들면 ADDRESS0, ADDRESS1,ADDRESS2 등과 같습니다.

기본적으로 한 인터페이스에서 다른 인터페이스로 또는 동일한 인터페이스로 패킷 전달은 보안상의 이유로 비활성화됩니다.

그러면 시스템이 외부 트래픽의 라우터 역할을 합니다. 연결을 공유하거나 VPN 서버를 구성할 때와 같이 외부 트래픽을 라우팅하는 시스템이 필요한 경우 IP 전달을 활성화해야 합니다. 자세한 내용은 *Red Hat Enterprise Linux 7 보안 가이드* 를 참조하십시오.

4.5.1. 정책 라우팅 이해

정책 라우팅 은 보다 유연한 라우팅 구성을 위한 메커니즘입니다. 라우팅 결정은 일반적으로 패키지의 대상 IP 주소를 기반으로 결정됩니다. 정책 라우팅을 사용하면 소스 IP 주소, 소스 포트, 프로토콜 유형과 같은 다른 라우팅 속성에 따라 경로를 유연하게 선택할 수 있습니다. 라우팅 테이블은 네트워크에 대한 경로 정보를 저장합니다. 이 값은 숫자 값 또는 이름으로 식별되며, 이 값은 /etc/iproute2/rt_tables 파일에서 구성할 수 있습니다. 기본 테이블은 254 로 식별됩니다. 정책 라우팅을 사용하면 규칙도 필요합니다. 규칙은 패킷의 특정 속성에 따라 라우팅 테이블을 선택하는 데 사용됩니다.

initscripts의 경우 라우팅 테이블은 테이블 인수를 통해 구성할 수 있는 경로의 속성입니다. **ip** 경로 형식은 여러 라우팅 테이블을 지정하는 기존 정책 기반 라우팅 구성을 정의하는 데 사용할 수 있습니다.

```
10.10.10.0/24 via 192.168.0.10 table 1
10.10.10.0/24 via 192.168.0.10 table 2
```

initscripts에서 라우팅 규칙을 지정하려면 **IPv4**의 **/etc/sysconfig/network-scripts/rule-*enp1s0*** 파일 또는 **IPv 6** 의 경우 **/etc/sysconfig/network-scripts/rule6-*enp1s0*** 파일로 편집합니다.

NetworkManager 는 **policy-routing**을 지원하지만 규칙은 아직 지원되지 않습니다. 규칙은 사용자 지정 스크립트를 실행하는 사용자가 구성해야 합니다. 수동 정적 경로마다 라우팅 테이블을 선택할 수 있습니다.

- **IPv 4**의 **ipv4.route-table**

  및

- **IPv 6**용 **ipv6.route-table.**

경로를 특정 테이블에 설정하면 **DHCP,autoconf6, DHCP 6** 의 모든 경로가 해당 특정 테이블에 배치됩니다. 또한 주소가 이미 구성된 서브넷의 모든 경로는 해당 라우팅 테이블에 배치됩니다. 예를 들어 ***192.168.1.10/24*** 주소를 구성하면 ***192.168.1.0/24*** 서브넷이 **ipv4.route-table**에 포함됩니다.

정책 라우팅 규칙에 대한 자세한 내용은 **ip-rule(8)** 도움말 페이지를 참조하십시오. 라우팅 테이블은 **ip-route(8)** 도움말 페이지를 참조하십시오.

4.6. 기본 게이트웨이 구성

기본 게이트웨이는 먼저 **/etc/sysconfig/network** 파일을 구문 분석한 다음 "작동" 중인 인터페이스에 대한 네트워크 인터페이스 **ifcfg** 파일을 구문 분석하는 네트워크 스크립트에 따라 결정됩니다. **ifcfg** 파일은 숫자로 오름차순으로 구문 분석되며 읽을 마지막 **GATEWAY** 지시문은 라우팅 테이블의 기본 경로를 작성하는 데 사용됩니다.

글로벌 또는 인터페이스별 구성 파일에서 **GATEWAY** 지시문을 사용하여 기본 경로를 지정할 수 있습

니다. 그러나 **Red Hat Enterprise Linux**에서는 글로벌 **/etc/sysconfig/network** 파일을 사용하지 않으며 게이트웨이를 이제 인터페이스별 구성 파일에서만 수행해야 합니다.

NetworkManager 에서 모바일 호스트를 관리하는 동적 네트워크 환경에서는 게이트웨이 정보가 인터페이스별로 지정될 가능성이 높으며 **DHCP** 에서 할당하는 것이 가장 좋습니다. 게이트웨이에 도달하기 위해 **NetworkManager** 의 종료 인터페이스에 영향을 주는 특별한 경우에는 기본 게이트웨이로 이어지지 않는 **ifcfg** 파일에서 **DEFROUTE=no** 명령을 사용하십시오.

5장. 네트워크 연결 설정 구성

이 장에서는 네트워크 연결 설정의 다양한 구성을 설명하고 **NetworkManager** 를 사용하여 구성하는 방법을 보여줍니다.

5.1. 802.3 링크 설정 구성

다음 구성 매개변수를 수정하여 이더넷 연결의 **802.3** 링크 설정을 구성할 수 있습니다.

- **802-3-ethernet.auto-negotiate**
- **802-3-ethernet.speed**
- **802-3-ethernet.duplex**

802.3 링크 설정을 다음 세 가지 기본 모드로 구성할 수 있습니다.

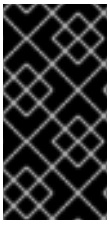
- 링크 협상 무시
- 자동 협상 활성화 시행
- 수동으로 속도 및 이중 링크 설정

링크 협상 무시

이 경우 **NetworkManager** 는 이더넷 연결에 대한 링크 구성을 무시하고 장치에 이미 구성을 유지합니다.

링크 협상을 무시하려면 다음 매개변수를 설정합니다.

```
802-3-ethernet.auto-negotiate = no
802-3-ethernet.speed = 0
802-3-ethernet.duplex = NULL
```



중요

auto-negotiate 매개 변수가 **no** 로 설정되어 있지만 속도 및 이중 값이 설정되지 않은 경우 자동 협상이 비활성화되어 있음을 의미하는 것은 아닙니다.

자동 협상 활성화 적용

이 경우 **NetworkManager** 는 장치에 자동 협상을 적용합니다.

자동 협상 활성화를 적용하려면 다음 옵션을 설정합니다.

```
802-3-ethernet.auto-negotiate = yes
802-3-ethernet.speed = 0
802-3-ethernet.duplex = NULL
```

링크 속도 및 이중화 수동 설정

이 경우 링크의 속도 및 이중 설정을 수동으로 구성할 수 있습니다.

속도 및 이중 링크 설정을 수동으로 설정하려면 다음과 같이 앞서 언급한 매개변수를 설정합니다.

```
802-3-ethernet.auto-negotiate = no
802-3-ethernet.speed = [speed in Mbit/s]
802-3-ethernet.duplex = [half |full]
```

중요

속도 및 이중 값을 모두 설정해야 합니다. 그렇지 않으면 **NetworkManager** 가 링크 구성을 업데이트하지 않습니다.

시스템 관리자는 다음 옵션 중 하나를 사용하여 **802.3** 링크 설정을 구성할 수 있습니다.

- **nmcli** 도구
- **nm-connection-editor** 유틸리티

nmcli 도구를 사용하여 **802.3** 링크 설정 구성

절차

1. ***enp1s0*** 장치에 대한 새 이더넷 연결을 만듭니다.
2. **802.3** 링크 설정을 선택한 구성으로 설정합니다. 자세한 내용은 를 참조하십시오. 5.1절. “802.3 링크 설정 구성”

   예를 들어 속도 옵션을 수동으로 설정하려면 ***100 Mbit/s*** 및 **duplex** 를 ***full*** 로 설정합니다.

```
nmcli connection add con-name MyEthernet type ethernet ifname enp1s0 \
802-3-ethernet.auto-negotiate no \
802-3-ethernet.speed 100 \
802-3-ethernet.duplex full
```

nm-connection-editor를 사용하여 **802.3** 연결 설정 구성

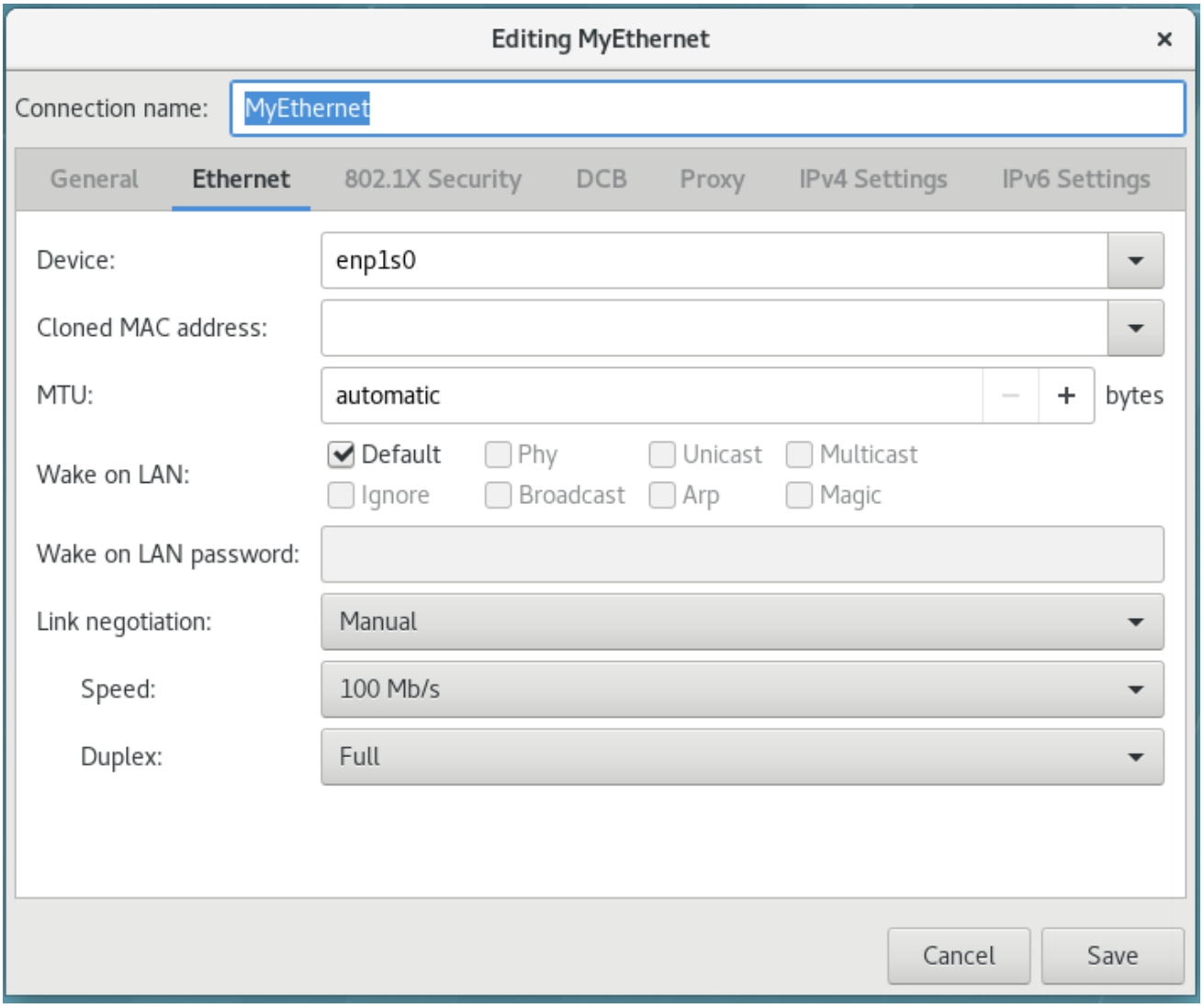
절차

1. 터미널에서 **nm-connection-editor** 를 입력합니다.
2. 편집할 이더넷 연결을 선택하고 기어 **wheel** 아이콘을 클릭하여 편집 대화 상자로 이동합니다. 자세한 내용은 3.4.3절. “nm-connection-editor를 사용하는 일반적인 구성 옵션”을 참조하십시오.

3. 선택한 대화 상자를 선택합니다.

- **ignore:** 링크 구성을 건너뜁니다(기본값).
- **Automatic:** 링크 자동 협상은 장치에 적용됩니다.
- **Manual: Speed** 및 **Duplex** 옵션을 지정하여 링크 협상을 적용할 수 있습니다.

그림 **5.1. nm-connection-editor**를 사용하여 **802.3** 링크 설정을 구성합니다.



5.2. 802.1X 보안 설정

802.1x 보안은 *포트 기반 네트워크 액세스 제어* (PNAC)에 대한 IEEE 표준의 이름입니다. 이를 ***WPA Enterprise*** 라고 합니다. **802.1x** 보안은 물리적 네트워크에서 *논리적 네트워크에* 대한 액세스를 제어하는 방법입니다. 논리적 네트워크에 가입하려는 모든 클라이언트는 올바른 **802.1X** 인증 방법을 사용하여 서버(예: 라우터)로 인증해야 합니다.

802.1x 보안은 무선 네트워크(WLANs) 보안과 가장 자주 관련되지만 LAN(네트워크에 대한 물리적 액세스)을 가진 침입을 방지하는 데에도 사용할 수 있습니다.

과거에는 DHCP 서버가 권한이 없는 사용자에게 IP 주소를 임대하도록 구성되지 않았지만, 이러한 방법이 비실용적이며 안전하지 않은 여러 가지 이유로 더 이상 권장되지 않습니다. 대신 **802.1X** 보안은 포트 기반 인증을 통해 논리적으로 보안된 네트워크를 보장하는 데 사용됩니다.

802.1x는 **vRAM** 및 **LAN** 액세스 제어를 위한 프레임워크를 제공하며, **EAP(Extensible Authentication Protocol)** 유형 중 하나를 전달하기 위한 봉투 역할을 합니다. **EAP** 유형은 네트워크에서 보안을 구현하는 방법을 정의하는 프로토콜입니다.

5.2.1. nmcli로 Wi-Fi에 대한 802.1X 보안 설정

절차

1. 인증된 키-mgmt (키 관리) 프로토콜을 설정합니다. 보안 와이파이 연결에 대한 키 지정 메커니즘을 구성합니다. 속성에 대한 자세한 내용은 ***nm-settings(5)*** 도움말 페이지를 참조하십시오.

2. **802-1x** 인증 설정을 구성합니다. **TLS(Transport Layer Security)** 인증은 "TLS 설정 구성"을 참조하십시오.

표 5.1. 802-1x 인증 설정

802-1x 인증 설정	이름
802-1x.identity	ID
802-1x.ca-cert	CA 인증서

802-1x 인증 설정	이름
802-1x.client-cert	사용자 인증서
802-1x.private-key	개인 키
802-1x.private-key-password	개인 키 암호

예를 들어 **EAP-TLS** 인증 방법을 사용하여 **WPA2 Enterprise**를 구성하려면 다음 설정을 적용합니다.

```
nmcli c add type wifi ifname wlo61s0 con-name 'My Wifi Network' \
    802-11-wireless.ssid 'My Wifi' \
    802-11-wireless-security.key-mgmt wpa-eap \
    802-1x.eap tls \
    802-1x.identity identity@example.com \
    802-1x.ca-cert /etc/pki/my-wifi/ca.crt \
    802-1x.client-cert /etc/pki/my-wifi/client.crt \
    802-1x.private-key /etc/pki/my-wifi/client.key \
    802-1x.private-key-password s3cr3t
```

5.2.2. nmcli로 Wired에 대한 802.1X 보안 설정

nmcli 도구를 사용하여 유선 연결을 구성하려면 **802-11- wireless .ssid** 및 **802-11- wireless-security. key-mgmt** 설정을 제외한 무선 연결과 동일한 절차를 따르십시오.

5.2.3. GUI를 사용하여 Wi-Fi에 대한 802.1X 보안 설정

절차

1. 네트워크 창을 엽니다(**3.4.1**절. “제어 센터 **GUI**를 사용하여 네트워크에 연결”참조).
2. 오른쪽 메뉴에서 무선 네트워크 인터페이스를 선택합니다. 필요한 경우 심볼릭 전원 버튼을 **ON** (켜짐)으로 설정하고 하드웨어 스위치가 켜졌는지 확인합니다.
3. 새 연결의 연결 이름을 선택하거나 **802.1X** 보안을 구성할 기존 연결 프로필의 기어 **wheel** 아이콘을 클릭합니다. 새 연결의 경우 연결을 완료하기 위해 인증 단계를 완료한 다음 기어 **wheel** 아이콘을 클릭합니다.

4.

Security(보안)를 선택합니다.

다음 구성 옵션을 사용할 수 있습니다.

보안

none - Wi-Fi 연결을 암호화하지 마십시오.

WEP 40/128-bit Key - IEEEruntime 표준의 **WEP(Wired Equivalent Privacy)**입니다. 단일 사전 공유 키**(PSK)**를 사용합니다.

WEP 128비트 암호 **-** 암호의 **MD5** 해시를 사용하여 **WEP** 키를 파생합니다.

LEAP - Cisco 시스템의 경량 확장 인증 프로토콜.

동적 **WEP(802.1X) - WEP** 키가 동적으로 변경됩니다. 다음과 함께 사용 **“TLS** 설정 구성”

WPA 및 **WPA2** 개인용 **- IEEE IEEEi** 표준 의 **Wi-Fi** 보호 액세스**(WPA). WEP** 교체**. WPA2(WPA2) Wi-Fi Protected Access II(WPA2).** 개인 모드는 사전 공유 키**(WPA-PSK)**를 사용합니다.

WPA 및 **WPA2 Enterprise - IEEE 802.1X** 네트워크 액세스 제어를 제공하기 위해 **RADIUS** 인증 서버와 함께 사용하기 위한 **WPA.** 다음과 함께 사용 **“TLS** 설정 구성”

암호

인증 프로세스에서 사용할 암호를 입력합니다.

5.

드롭다운 메뉴에서 다음 보안 방법 중 하나를 선택합니다. **LEAP,Dynamic WEP(802.1X)** 또

는 **WPA** 및 **WPA2 Enterprise.**

보안 드롭다운 메뉴에서 선택한 *확장 가능 인증 프로토콜* **(EAP)** 유형에 대한 설명은 **"TLS 설정 구성"** 을 참조하십시오.

5.2.4. nm-connection-editor를 사용하여 Wired에 대해 802.1X 보안 설정

절차

1. 터미널에 **nm-connection-editor** 를 입력합니다.

```
~]$ nm-connection-editor
```

Network Connections(네트워크 연결) 창이 표시됩니다.

2. 편집할 이더넷 연결을 선택하고 기어 **wheel** 아이콘을 클릭하고 **3.4.6.2절. "nm-connection-editor를 사용하여 Wired 연결 설정"** 을 참조하십시오.

3. **Security**(보안)를 선택하고 심볼릭 전원 단추를 **ON** (켜짐)으로 설정하여 설정 구성을 활성화합니다.

4. 다음 인증 방법 중 하나를 선택합니다.

- *전송 계층 보안을 위해* ***TLS*** *를* 선택하고 **"TLS 설정 구성"** 로 진행합니다.
- *보안 튜닝을 통해 유연한 인증을 위해* ***FAST*** *를 선택하고 "터널 TLS 설정 구성" 진행합니다.*
- **Tunneled** ***Transport Layer Security****에 대해* ***Tunneled*** **TLS** 를 선택합니다. 그러지 않으면 **TTLS** 또는 **EAP-TTLS**라고 하며 **"터널 TLS 설정 구성"** 진행합니다.
- 보호된 *확장 가능 인증 프로토콜(PEAP)에 대해 보호된* **EAP(PEAP)** 를 선택하고 **"보호된 EAP(PEAP) 설정 구성"** 진행합니다.

TLS 설정 구성

TLS(Transport Layer Security)에서는 클라이언트와 서버가 TLS 프로토콜을 사용하여 상호 인증됩니다. 서버는 디지털 인증서를 보유하고 있음을 입증하고 클라이언트는 클라이언트 측 인증서를 사용하여 자체 ID를 증명하고 주요 정보를 교환합니다. 인증이 완료되면 TLS 터널이 더 이상 사용되지 않습니다. 대신 클라이언트 및 서버는 교환 키를 사용하여 AES, TKIP 또는 WEP를 사용하여 데이터를 암호화합니다.

인증서를 인증하려는 모든 클라이언트에 배포해야 한다는 팩트는 EAP-TLS 인증 방법이 매우 강력하지만 설정하기에 더 복잡함을 의미합니다. TLS 보안을 사용하려면 PKI(공개 키 인프라)의 오버헤드가 인증서를 관리해야 합니다. TLS 보안을 사용할 때의 이점은 손상된 암호가 (W)LAN에 대한 액세스를 허용하지 않는다는 것입니다. 침입자는 또한 인증 클라이언트의 개인 키에 액세스할 수 있어야 합니다.

NetworkManager 는 지원되는 TLS 버전을 확인하지 않습니다. NetworkManager 는 사용자가 입력한 매개 변수를 수집하여 프로시저를 처리하는 wpa_supplicant 데몬에 전달합니다. 그런 다음 OpenSSL을 사용하여 TLS 터널을 설정합니다. OpenSSL 자체가 SSL/TLS 프로토콜 버전을 협상합니다. 가장 높은 버전 양쪽에서 지원을 받습니다.

TLS 설정을 구성하려면 5.2.4절. “nm-connection-editor를 사용하여 Wired에 대해 802.1X 보안 설정” 에 설명된 절차를 따르십시오. 다음 구성 설정을 사용할 수 있습니다.

ID

이 서버의 ID를 제공합니다.

사용자 인증서

Distinguished Encoding Rules (DER) 또는 *Privacy Enhanced Mail* (PEM)으로 인코딩된 개인 X.509 인증서 파일을 찾아서 선택하고 선택합니다.

CA 인증서

Distinguished Encoding Rules (DER) 또는 *Privacy Enhanced Mail* (PEM)으로 인코딩된 X.509 인증 *기관 인증서* 파일을 찾아서 선택하고 선택합니다.

개인 키

Distinguished Encoding Rules (DER), *Privacy Enhanced Mail* (PEM) 또는 *Personal Information Exchange Syntax Standard* (PKCS #12)로 인코딩된 *개인 키* 파일을 찾아서 선택하십시오.

개인 키 암호

개인 키 필드에 개인 키의 암호를 입력합니다. **Enter password**(암호 표시)를 선택하여 암호를 입력한 대로 표시합니다.

FAST 설정 구성

FAST 설정을 구성하려면 5.2.4절. “nm-connection-editor를 사용하여 Wired에 대해 802.1X 보안 설정” 에 설명된 절차를 따르십시오. 다음 구성 설정을 사용할 수 있습니다.

익명 **ID**

이 서버의 **ID**를 제공합니다.

PAC 프로비저닝

활성화하려는 확인란을 선택한 다음 **Anonymous,Authenticated** 및 **Both** 를 선택합니다.

PAC 파일

를 클릭하여**PAC(** *보호된 액세스 자격 증명* **)** 파일을 찾아 선택합니다.

내부 인증

GTC - 일반 토큰 카드.

MSCHAPv2 - Microsoft Challenge Handshake Authentication Protocol 버전 **2.**

사용자 이름

인증 프로세스에서 사용할 사용자 이름을 입력합니다.

암호

인증 프로세스에서 사용할 암호를 입력합니다.

터널 **TLS** 설정 구성

터널 **TLS** 설정을 구성하려면 5.2.4절. “nm-connection-editor를 사용하여 Wired에 대해 802.1X 보안 설정” 에 설명된 절차를 따르십시오. 다음 구성 설정을 사용할 수 있습니다.

익명 **ID**

이 값은 암호화되지 않은 ID로 사용됩니다.

CA 인증서

를 클릭하여 인증 기관의 인증서를 찾아 선택합니다.

내부 인증

PAP - 암호 인증 프로토콜.

MSCHAP - Challenge Handshake Authentication Protocol.

MSCHAPv2 - Microsoft Challenge Handshake Authentication Protocol 버전 2.

CHAP - 과제 처리 인증 프로토콜.

사용자 이름

인증 프로세스에서 사용할 사용자 이름을 입력합니다.

암호

인증 프로세스에서 사용할 암호를 입력합니다.

보호된 EAP(PEAP) 설정 구성

보호된 EAP(PEAP) 설정을 구성하려면 5.2.4절. “nm-connection-editor를 사용하여 Wired에 대해 802.1X 보안 설정” 에 설명된 절차를 따르십시오. 다음 구성 설정을 사용할 수 있습니다.

익명 ID

이 값은 암호화되지 않은 ID로 사용됩니다.

CA 인증서

를 클릭하여 인증 기관의 인증서를 찾아 선택합니다.

PEAP 버전

사용할 **EAP**의 버전입니다. 자동, **0** 또는 **1.**

내부 인증

MSCHAPv2 - Microsoft Challenge Handshake Authentication Protocol 버전 **2.**

MD5 - 암호화 해시 기능인 메시지 다이제스트 **5.**

GTC - 일반 토큰 카드**.**

사용자 이름

인증 프로세스에서 사용할 사용자 이름을 입력합니다**.**

암호

인증 프로세스에서 사용할 암호를 입력합니다**.**

5.3. WPA_SUPPLICANT 및 NETWORKMANAGER에서 MACSEC 사용

Media Access Control Security (MACsec, IEEE 802.1AE)는 **GCM-AES-128** 알고리즘을 사용하여 **LAN**의 모든 트래픽을 암호화하고 인증합니다**. MACsec** 은 **IP** 뿐만 아니라 **ARP(Address Resolution Protocol), ND(Nighbor Discovery)** 또는 **DHCP** 도 보호할 수 있습니다**. IPsec** 은 애플리케이션 계층**(**계층 **7)**의 네트워크 계층**(**계층 **3** 및 **SSL** 또는 **TLS)**에서 작동하지만 **MACsec** 은 데이터 링크 계층**(**계층 **2)**에서 작동합니다**.** 다른 네트워킹 계층을 위해 **MACsec** 과 보안 프로토콜을 결합하여 이러한 표준이 제공하는 다양한 보안 기능을 활용할 수 있습니다**.**

사전 공유 연결 키**/CAK** 이름**(CAK/CKN)** 쌍을 사용하여 인증을 수행하는 스위치에서 **MACsec** 을 활성화하려면 다음을 수행합니다**.**

절차

1. **CAK/CKN** 쌍을 만듭니다**.** 예를 들어 다음 명령은 **16**바이트 키를 **16**진수 표기법으로 생성합니다**.**

```
-
```

```
~]$ dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2 "%02x"'
```

2. **wpa_supplicant.conf** 구성 파일을 생성하고 여기에 다음 행을 추가합니다.

```
ctrl_interface=/var/run/wpa_supplicant
eapol_version=3
ap_scan=0
fast_reauth=1

network={
    key_mgmt=NONE
    eapol_flags=0
    macsec_policy=1

    mka_cak=0011... # 16 bytes hexadecimal
    mka_ckn=2233... # 32 bytes hexadecimal
}
```

이전 단계의 값을 사용하여 **wpa _supplicant.conf** 구성 파일에서 **mka_cak** 및 **mka_ ckn** 행을 완료합니다.

자세한 내용은 **wpa_supplicant.conf(5)** 매뉴얼 페이지를 참조하십시오.

3. ***wlp61s0*** 을 사용하여 네트워크에 연결하는 경우 다음 명령을 사용하여 **wpa_supplicant** 를 시작합니다.

```
~]# wpa_supplicant -i wlp61s0 -Dmacsec_linux -c wpa_supplicant.conf
```

wpa_supplicant.conf 파일을 만들고 편집하는 대신**, Red Hat**은 **nmcli** 명령을 사용하여 이전 단계와 동일하게 **wpa_supplicant** 를 구성하는 것이 좋습니다**.** 다음 예제에서는 **16**바이트 **16**진수 **CAK($MKA_CAK)** 및 **32**바이트 **16**진수 **CKN($MKA_CKN)**이 이미 있다고 가정합니다**.**

```
~]# nmcli connection add type macsec \
  con-name test-macsec+ ifname macsec0 \
  connection.autoconnect no \
  macsec.parent wlp61s0 macsec.mode psk \
  macsec.mka-cak $MKA_CAK \
  macsec.mka-cak-flags 0 \
  macsec.mka-ckn $MKA_CKN

~]# nmcli connection up test-macsec+
```

이 단계 후에 *macsec0* 장치를 구성하고 네트워킹에 사용해야 합니다.

자세한 내용은 MACsec의 새로운 기능: wpa_supplicant를 사용하여 MACsec 설정 및 (선택 사항) NetworkManager 문서를 참조하십시오. 또한 MACsec 네트워크의 아키텍처에 대한 자세한 내용은 MACsec: 네트워크 트래픽 문서를 암호화하는 다른 솔루션 , 사용 사례 시나리오 및 구성 예제를 참조하십시오.

5.4. IPV4 설정 구성

제어 센터를 사용하여 IPv4 설정 구성

절차

1. Super 키를 눌러 Activities Overview(활동 개요)를 입력하고 Settings (설정)를 입력한 다음 Enter 키를 누릅니다. 그런 다음 왼쪽에서 Network (네트워크) 탭을 선택하면 Network settings(네트워크 설정) 도구가 표시됩니다. "제어 센터를 사용하여 새 연결 구성" 진행하십시오.

2. 편집할 연결을 선택하고 톱니바퀴 아이콘을 클릭합니다. Editing(편집) 대화 상자가 표시됩니다.

3. IPv4 메뉴 항목을 클릭합니다.

IPv4 메뉴 항목을 사용하면 네트워크에 연결하고 IP 주소,DNS 및 필요에 따라 경로 정보를 입력하도록 사용하는 방법을 구성할 수 있습니다. IPv4 메뉴 항목은 다음 연결 유형 중 하나를 만들고 수정할 때 사용할 수 있습니다: 유선, 무선, 모바일 광대역, VPN 또는 DSL.

DHCP를 사용하여 DHCP 서버에서 동적 IP 주소를 가져오는 경우 주소를 DHCP (자동) 로 설정할 수 있습니다.

정적 경로를 구성해야 하는 경우 4.3절. "GUI를 사용하여 정적 경로 구성" 을 참조하십시오.

nm-connection-editor를 사용하여 IPV4의 메서드 설정

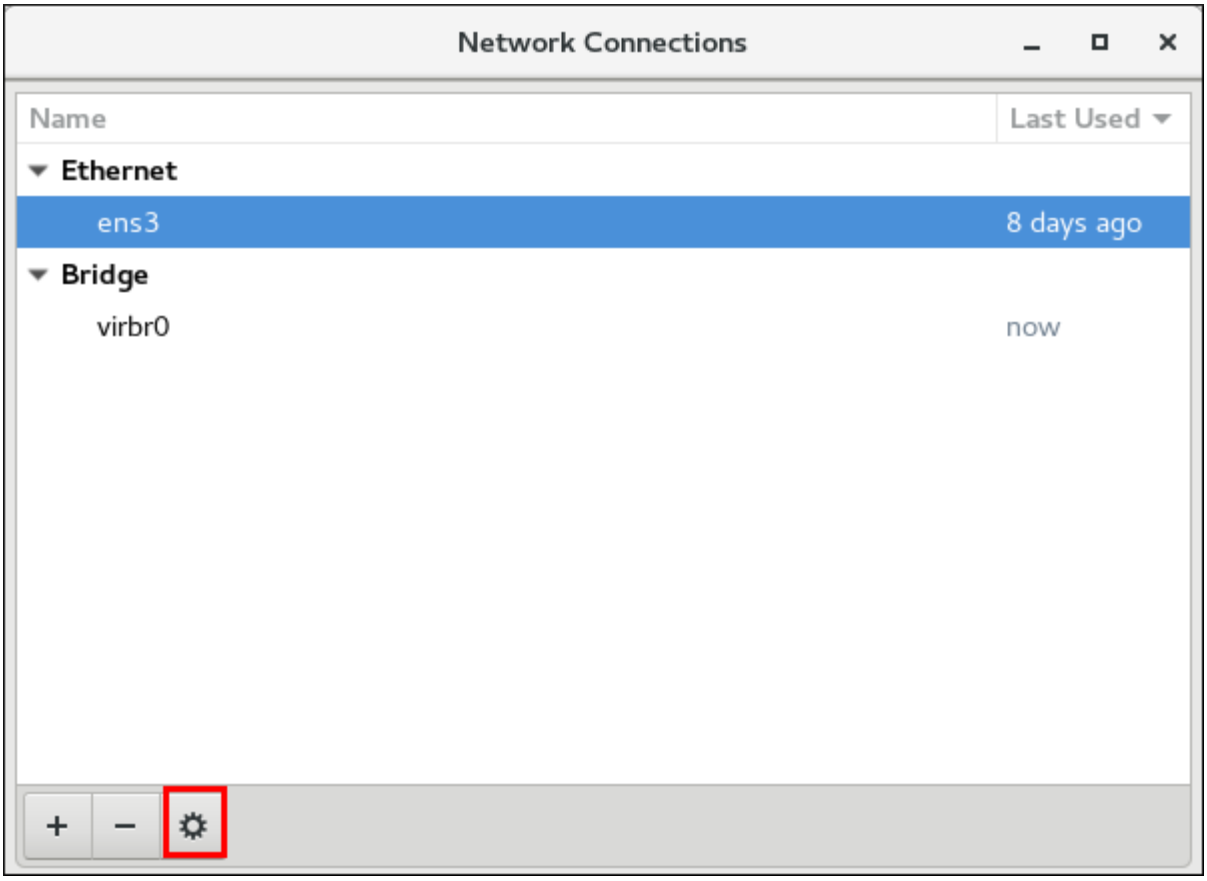
nm-connection-editor 를 사용하여 연결 설정을 편집하고 구성할 수 있습니다. 다음 절차에서는 IPv4 설정을 구성하는 방법을 설명합니다.

절차

1. 터미널에서 **nm-connection-editor** 를 입력합니다.

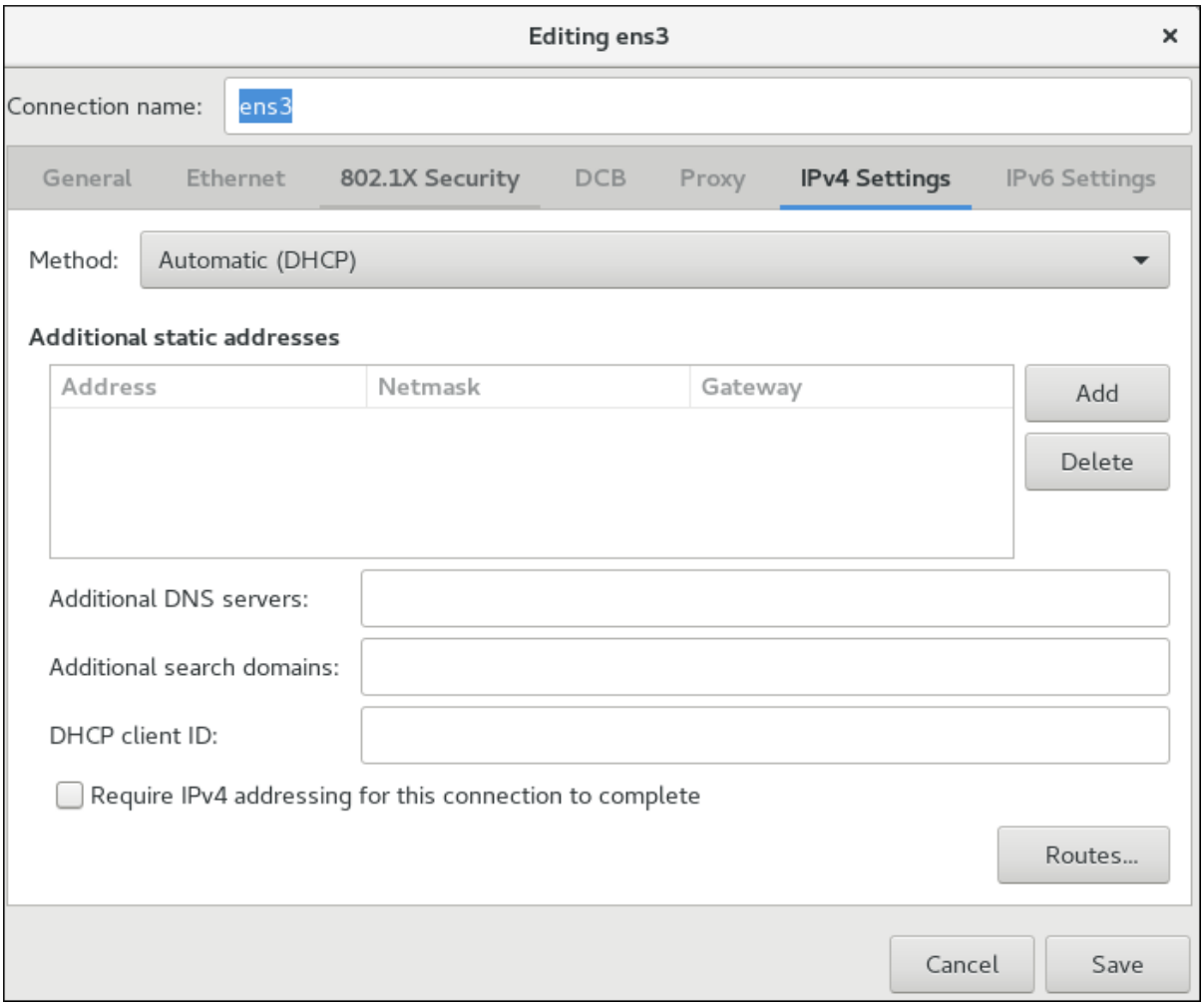
2. 기존 연결 유형의 경우 기어 **wheel** 아이콘을 클릭합니다.

그림 5.2. 연결 편집



3. **IPv4 Settings(IPv4 설정)** 를 클릭합니다.

그림 5.3. IPv4 설정 구성



연결 유형별 사용 가능한 IPv4 메서드

구성 중인 연결 유형에 따라 **Method** (메서드) 드롭다운 메뉴를 클릭하면 다음 **IPv4** 연결 방법 중 하나를 선택할 수 있습니다. 모든 메서드는 다음과 관련된 연결 유형 또는 유형에 따라 여기에 나열됩니다.

유선, 무선 및 DSL 연결 방법

자동(DHCP) - 연결된 네트워크에서 **DHCP** 서버를 사용하여 **IP** 주소를 할당하는 경우 이 옵션을 선택합니다. **DHCP** 클라이언트 **ID** 필드를 입력하지 않아도 됩니다.

주소만 자동(DHCP) 주소만 - 연결 중인 네트워크에서 **DHCP** 서버를 사용하여 **IP** 주소를 할당하지만 **DNS** 서버를 수동으로 할당하려는 경우 이 옵션을 선택합니다.

수동 - **IP** 주소를 수동으로 할당하려면 이 옵션을 선택합니다.

링크-로컬 전용 - 연결하는 네트워크에 DHCP 서버가 없고 IP 주소를 수동으로 할당하지 않으려는 경우 이 옵션을 선택합니다. 임의의 주소는 접두사 169.254/16 을 사용하여 *RFC 3927* 에 따라 할당됩니다.

다른 컴퓨터에 공유 - 구성 중인 인터페이스가 인터넷 또는 WAN 연결을 공유하도록 하려면 이 옵션을 선택합니다. 인터페이스에는 10.42.x.1/24 범위의 주소가 할당되고 DHCP 서버와 DNS 서버가 시작되며 인터페이스는 네트워크 *주소 변환(NAT)이 있는 시스템의 기본 네트워크* 연결에 연결됩니다.

Disabled - 이 연결에 대해 IPv4 가 비활성화되어 있습니다.

모바일 광대역 연결 방법

자동(PPP) - 연결하는 네트워크가 IP 주소와 DNS 서버를 자동으로 할당하는 경우 이 옵션을 선택합니다.

주소만 자동(PPP) - 연결 중인 네트워크에서 IP 주소를 자동으로 할당하지만 DNS 서버를 수동으로 지정하려는 경우 이 옵션을 선택합니다.

VPN 연결 방법

자동(VPN) - 연결하는 네트워크가 IP 주소와 DNS 서버를 자동으로 할당하는 경우 이 옵션을 선택합니다.

자동(VPN) 주소만 - 연결하는 네트워크가 IP 주소를 자동으로 할당하지만 DNS 서버를 수동으로 지정하는 경우 이 옵션을 선택합니다.

DSL 연결 방법

자동(PPPoE) - 연결 중인 네트워크가 IP 주소와 DNS 서버를 자동으로 할당하는 경우 이 옵션을 선택합니다.

주소만 자동(PPPoE) - 연결 중인 네트워크에서 IP 주소를 자동으로 할당하지만 DNS 서버를 수동으로 지정하려는 경우 이 옵션을 선택합니다.

DHCP를 사용하여 DHCP 서버에서 동적 IP 주소를 가져오는 경우 Method 를 Automatic(DHCP) 으로 설정하면 됩니다.

정적 경로를 구성해야 하는 경우 **Routes** (경로) 버튼을 클릭하고 구성 옵션에 대한 자세한 내용은 4.3 절. "GUI를 사용하여 정적 경로 구성" 를 참조하십시오.

5.5. IPV6 설정 구성

IPv6 설정을 구성하려면 5.4절. "IPv4 설정 구성" 에 설명된 절차를 따르고 **IPv6** 메뉴 항목을 클릭합니다.

방식

무시 - 이 연결에 대한 **IPv6** 설정을 무시하려면 이 옵션을 선택합니다.

자동 - ***SLAAC*** 를 사용하여 하드웨어 주소 및 *라우터 알림* (RA)에 따라 자동 상태 비저장 구성을 생성하도록 이 옵션을 선택합니다.

자동, 주소만 - 연결 중인 네트워크에서 *라우터 알림* (RA)을 사용하여 자동 상태 비저장 구성을 생성하지만 **DNS** 서버를 수동으로 할당하려는 경우 이 옵션을 선택합니다.

Automatic, DHCP only - RA를 사용하지 않도록 이 옵션을 선택하고 **DHCPv6** 의 정보를 직접 요청하여 상태 저장 구성을 생성합니다.

수동 - IP 주소를 수동으로 할당하려면 이 옵션을 선택합니다.

링크-로컬 전용 - 연결하는 네트워크에 **DHCP** 서버가 없고 **IP** 주소를 수동으로 할당하지 않으려는 경우 이 옵션을 선택합니다. 임의의 주소는 *RFC 4862* 에 따라 **FE80::0** 접두사가 할당됩니다.

주소

DNS 서버 - 쉼표로 구분된 **DNS** 서버 목록을 입력합니다.

검색 도메인 - 쉼표로 구분된 도메인 컨트롤러 목록을 입력합니다.

정적 경로를 구성해야 하는 경우 **Routes** (경로) 버튼을 클릭하고 구성 옵션에 대한 자세한 내용은 4.3 절. "GUI를 사용하여 정적 경로 구성" 를 참조하십시오.

5.6. PPP (POINT-TO-POINT) 설정 구성

인증 방법

대부분의 경우 공급자의 PPP 서버는 허용되는 모든 인증 방법을 지원합니다. 연결에 실패하는 경우 사용자는 PPP 서버 구성에 따라 일부 메서드에 대한 지원을 비활성화해야 합니다.

점대점 암호화 (MPPE) 사용

Microsoft 지점 간 암호화 프로토콜(*RFC 3078*).

BSD 데이터 압축 허용

PPP BSD 압축 프로토콜(*RFC 1977*).

Deflate 데이터 압축 허용

PPP Deflate Protocol(*RFC 1979*).

TCP 헤더 압축 사용

낮은 직렬 링크(RF*1144*)에 대한 TCP/IP 헤더 압축.

PPP 에코 패킷 전송

루프백 테스트를 위한 LCP 에코 요청 및 에코 응답 코드(*RFC 1661*).

참고

NetworkManager 에서 PPP 지원은 선택 사항이므로 PPP 설정을 구성하려면 NetworkManager-ppp 패키지가 이미 설치되어 있는지 확인합니다.

6장. 호스트 이름 구성

6.1. 호스트 이름 이해

호스트 이름 클래스에는 정적, 예, 임시의 세 가지 클래스가 있습니다.

"정적" 호스트 이름은 사용자가 선택할 수 있는 기존 호스트 이름이며 **/etc/hostname** 파일에 저장됩니다. "임시" 호스트 이름은 커널에서 유지 관리하는 동적 호스트 이름입니다. 기본적으로 정적 호스트 이름으로 초기화되며, 기본값은 "**localhost**" 입니다. 런타임 시 **DHCP** 또는 **mDNS** 에 의해 변경될 수 있습니다. "좋은" 호스트 이름은 사용자에게 프레젠테이션할 수 있는 자유 형식 **UTF8** 호스트 이름입니다.

참고

호스트 이름은 최대 **64**자까지 자유 형식 문자열일 수 있습니다. 그러나 정적 및 일시적인 이름 모두 **host.example.com** 과 같이 **DNS** 의 시스템에 사용되는 *정규화된 도메인 이름* **(FQDN)**과 일치하도록 권장합니다. 또한 정적 및 일시적인 이름은 **7**비트 **ASCII** 소문자, 공백이나 점 없음으로만 구성되며 엄격한 요구 사항은 아니지만 **DNS** 도메인 이름 레이블에 허용된 형식으로 제한하는 것이 좋습니다. 이전 사양에서는 밑줄을 허용하지 않으므로 사용하지 않는 것이 좋습니다.

hostnamectl 도구는 다음을 적용합니다. **a-z,A- Z,0-9,"-", "_"** 및 "." 로 구성되는 정적 및 일시적인 호스트 이름은 점으로 시작 또는 종료하지 않으며 즉시 두 개의 점이 없어야 합니다. 크기 제한은 **64**자입니다.

6.1.1. 권장 명명 관행

IANN(Internet Corporation for Assigned Names and Numbers)이 이전에 등록되지 않은 최상위 도메인(예: **.yourcompany**)을 공개 등록에 추가하는 경우가 있습니다. 따라서 사설 네트워크에서도 사용자에게 위임되지 않은 도메인 이름을 사용하지 않는 것이 좋습니다. 이로 인해 도메인 이름이 네트워크 구성에 따라 다르게 확인될 수 있습니다. 결과적으로 네트워크 리소스를 사용할 수 없게 될 수 있습니다. 도메인 이름 충돌을 수행하려면 **DNSSEC** 검증을 활성화하는 수동 구성이 필요하므로 **DNSSEC**를 배포하고 유지 관리하기가 더 어렵습니다. 이 문제에 대한 자세한 내용은 도메인 이름 충돌에 대한 **FAQ** 를 참조하십시오.

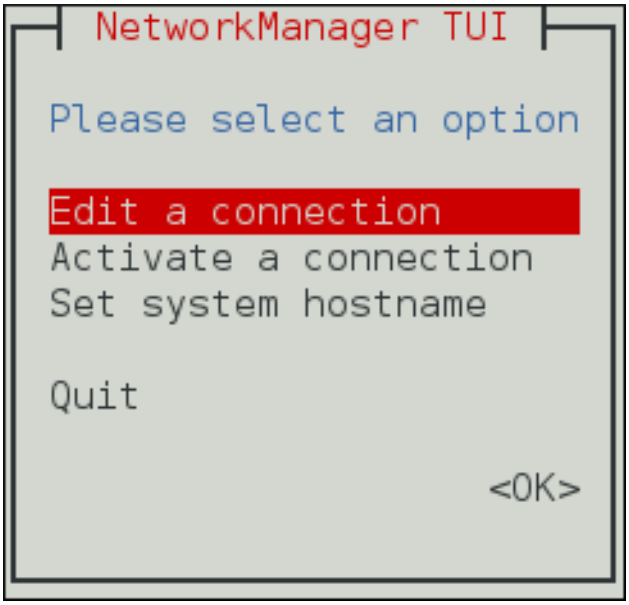
6.2. 텍스트 사용자 인터페이스를 사용하여 호스트 이름 구성, NMTUI

텍스트 사용자 인터페이스 도구 **nmtui** 는 터미널 창에서 호스트 이름을 구성하는 데 사용할 수 있습니다. 다음 명령을 실행하여 도구를 시작합니다.

```
~]$ nmtui
```

텍스트 사용자 인터페이스가 나타납니다. 잘못된 명령은 사용 메시지를 인쇄합니다.

그림 6.1. NetworkManager 텍스트 사용자 인터페이스 시작 메뉴



[D]

탐색하려면 화살표 키를 사용하거나 탭을 눌러 앞으로 이동하고 **Shift+Tab** 을 눌러 옵션을 다시 이동합니다. **Enter** 를 눌러 옵션을 선택합니다. **Space** 표시줄에서 확인란의 상태를 전환합니다.

nmtui 설치에 대한 자세한 내용은 3.2절. “nmtui로 IP 네트워킹 구성” 을 참조하십시오.

NetworkManager 텍스트 사용자 인터페이스 도구 **nmtui** 를 사용하여 **/etc/hostname** 파일에 정적 호스트 이름을 쿼리하고 설정할 수 있습니다.

중요

Red Hat Enterprise Linux에서 **NetworkManager** 는 **systemd-hostnamed** 서비스를 사용하여 **/etc/hostname** 파일에 저장된 정적 호스트 이름을 읽고 씁니다. 이로 인해 **NetworkManager** 에서 **/etc/hostname** 파일에 대한 수동 수정이 더 이상 자동으로 처리되지 않습니다. **hostnamectl** 유틸리티를 통해 시스템 호스트 이름을 변경해야 합니다. 또한 **/etc/sysconfig/network** 파일의 **HOSTNAME** 변수 사용은 더 이상 사용되지 않습니다.

6.3. HOSTNAMECTL을 사용하여 호스트 이름 구성

지정된 시스템에서 사용 중인 호스트 이름의 세 가지 별도 클래스를 관리하기 위해 **hostnamectl** 도구가 제공됩니다.

6.3.1. 모든 호스트 이름 보기

현재 호스트 이름을 모두 보려면 다음 명령을 입력합니다.

```
~]$ hostnamectl status
```

옵션을 지정하지 않으면 **status** 옵션이 기본적으로 반영됩니다.

6.3.2. 모든 호스트 이름 설정

시스템에서 모든 호스트 이름을 설정하려면 **root** 로 다음 명령을 입력합니다:

```
~]# hostnamectl set-hostname name
```

이렇게 하면 깔끔하고 정적이며 일시적인 호스트 이름이 모두 변경됩니다. 정적 호스트 이름 및 일시적인 호스트 이름은 비교적 간단하게 호스트 이름의 양식을 작성할 수 있습니다. 공백이 “-” 로 바뀌고 특수 문자가 제거됩니다.

6.3.3. 일반 호스트 이름 설정

특정 호스트 이름을 설정하려면 관련 옵션을 사용하여 **root** 로 다음 명령을 입력합니다.

```
~]# hostnamectl set-hostname name [option...]
```

여기서 ***option*** 은 **--pretty, -- static, --transient** 중 하나 이상입니다.

static 또는 **-- transient** 옵션을 **--pretty** 옵션과 함께 사용하는 경우 정적 호스트 이름과 일시적인 호스트 이름이 상당히 단순화됩니다. 공백이 “-” 로 바뀌고 특수 문자가 제거됩니다. **pretty** 옵션을 지정하지 않으면 간소화가 발생하지 않습니다.

호스트 이름을 지정할 때는 호스트 이름에 공백이나 작은따옴표가 포함된 경우 적절한 따옴표를 사용해야 합니다. 예를 들어 다음과 같습니다.

```
~]# hostnamectl set-hostname "Stephen's notebook" --pretty
```

6.3.4. 일반 호스트 이름 지우기

특정 호스트 이름을 지우고 기본값으로 되돌리려면 관련 옵션을 사용하여 **root** 로 다음 명령을 입력합니다.

```
~]# hostnamectl set-hostname "" [option...]
```

여기서 *""* 는 따옴표가 지정된 빈 문자열이며 여기서 *옵션은* **--pretty, -- static, -- transient** 중 하나 이상입니다.

6.3.5. 원격 호스트 이름 변경

원격 시스템에서 **hostnamectl** 명령을 실행하려면 다음과 같이 **-H, --host** 옵션을 사용합니다.

```
~]# hostnamectl set-hostname -H [username]@hostname
```

여기서 ***hostname*** 은 구성할 원격 호스트입니다. *사용자 이름은* 선택 사항입니다. **hostnamectl** 도구는 **SSH** 를 사용하여 원격 시스템에 연결합니다.

6.4. NMCLI를 사용하여 호스트 이름 구성

NetworkManager 도구 **nmcli** 를 사용하여 **/etc/hostname** 파일에 정적 호스트 이름을 쿼리하고 설정할 수 있습니다.

정적 호스트 이름을 쿼리하려면 다음 명령을 실행합니다.

```
~]$ nmcli general hostname
```

정적 호스트 이름을 ***my-server*** 로 설정하려면 **root** 로 다음 명령을 실행합니다:

```
~]# nmcli general hostname my-server
```

6.5. 추가 리소스

- **hostnamectl(1)** 도움말 페이지 **-** 명령 및 명령 옵션을 포함하는 **hostnamectl** 을 설명합니다.

- **hostname(1)** 도움말 페이지 **- hostname** 및 **domainname** 명령에 대한 설명이 포함되어 있습니다.

- **hostname(5)** 도움말 페이지 **-** 호스트 이름 파일**,** 해당 콘텐츠 및 사용에 대한 설명이 포함되어 있습니다.

- **hostname(7)** 도움말 페이지 **-** 호스트 이름 확인에 대한 설명이 포함되어 있습니다.

- **machine-info(5)** 도움말 페이지 **-** 로컬 시스템 정보 파일과 포함된 환경 변수를 설명합니다.

- **machine-id(5)** 도움말 페이지 **-** 로컬 시스템 **ID** 구성 파일을 설명합니다.

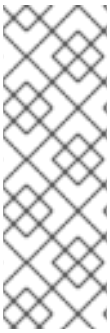
- **systemd-hostnamed.service(8)** 도움말 페이지 **- hostnamectl** 에서 사용하는 **systemd-hostnamed** 시스템 서비스를 설명합니다.

7장. 네트워크 연결 설정

Red Hat Enterprise Linux 7을 사용하면 관리자가 여러 네트워크 인터페이스를 결합한 단일 채널로 연결할 수 있습니다. 채널 본딩을 통해 두 개 이상의 네트워크 인터페이스가 하나로 작동하고 동시에 대역폭을 늘리고 이중화를 제공할 수 있습니다.

주의

네트워크 스위치 없이 직접 케이블 연결을 사용하는 것은 본딩에 대해 지원되지 않습니다. 여기에 설명된 장애 조치 메커니즘은 네트워크 스위치가 없으면 예상대로 작동하지 않습니다. 자세한 내용은 **Red Hat** 지식베이스 문서를 *참조하십시오*.



참고

active-backup, balance-tlb 및 **balance-alb** 모드에는 스위치의 특정 구성이 필요하지 않습니다. 다른 본딩 모드에는 링크를 집계하도록 스위치를 구성해야 합니다. 예를 들어 **Cisco** 스위치에는 모드 **0, 2** 및 **3**용 **EtherChannel**이 필요하지만 모드 **4 LACP** 및 **EtherChannel**의 경우 필요합니다. 스위치와 함께 제공된 설명서를 확인하고 https://www.kernel.org/doc/Documentation/networking/bonding.txt를 참조하십시오.

7.1. 기본 컨트롤러 동작 및 포트 인터페이스 이해

NetworkManager 데몬을 사용하여 결합된 포트 인터페이스를 제어할 때나 결함을 찾을 때 특히 다음에 유의해야 합니다.

1. 컨트롤러 인터페이스를 시작하면 포트 인터페이스가 자동으로 시작되지 않습니다.
2. 포트 인터페이스를 시작하면 항상 컨트롤러 인터페이스가 시작됩니다.
3. 컨트롤러 인터페이스를 중지하면 포트 인터페이스도 중지됩니다.
4. 포트가 없는 컨트롤러는 고정 **IP** 연결을 시작할 수 있습니다.

5.
포트 없는 컨트롤러는 **DHCP** 연결을 시작할 때 포트를 대기합니다.

6.
포트를 기다리는 **DHCP** 연결이 있는 컨트롤러는 캐리어가 있는 포트를 추가하면 완료됩니다.

7.
포트를 기다리는 **DHCP** 연결이 있는 컨트롤러는 캐리어가 없는 포트를 추가하면 계속 대기합니다.

7.2. 텍스트 사용자 인터페이스 NMTUI를 사용하여 본딩 구성

텍스트 사용자 인터페이스 도구 **nmtui** 는 터미널 창에서 본딩을 구성하는 데 사용할 수 있습니다. 다음 명령을 실행하여 도구를 시작합니다.

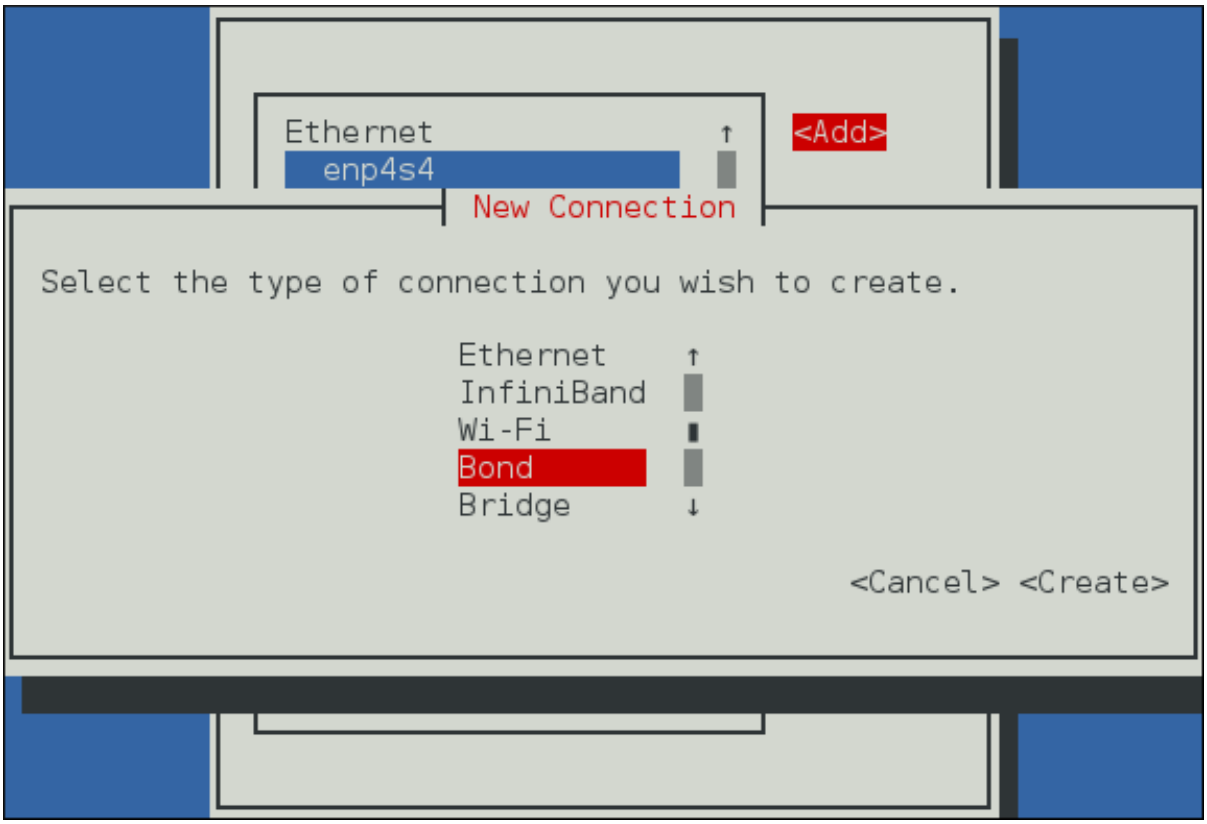
```
~]$ nmtui
```

텍스트 사용자 인터페이스가 나타납니다. 잘못된 명령은 사용 메시지를 인쇄합니다.

탐색하려면 화살표 키를 사용하거나 탭을 눌러 앞으로 이동하고 **Shift+Tab** 을 눌러 옵션을 다시 이동합니다. **Enter** 를 눌러 옵션을 선택합니다. **Space** 표시줄에서 확인란의 상태를 전환합니다.

1.
시작 메뉴에서 **Edit a connection** (연결 편집)을 선택합니다. **Add** 를 선택하면 **New Connection** (새 연결) 화면이 열립니다.

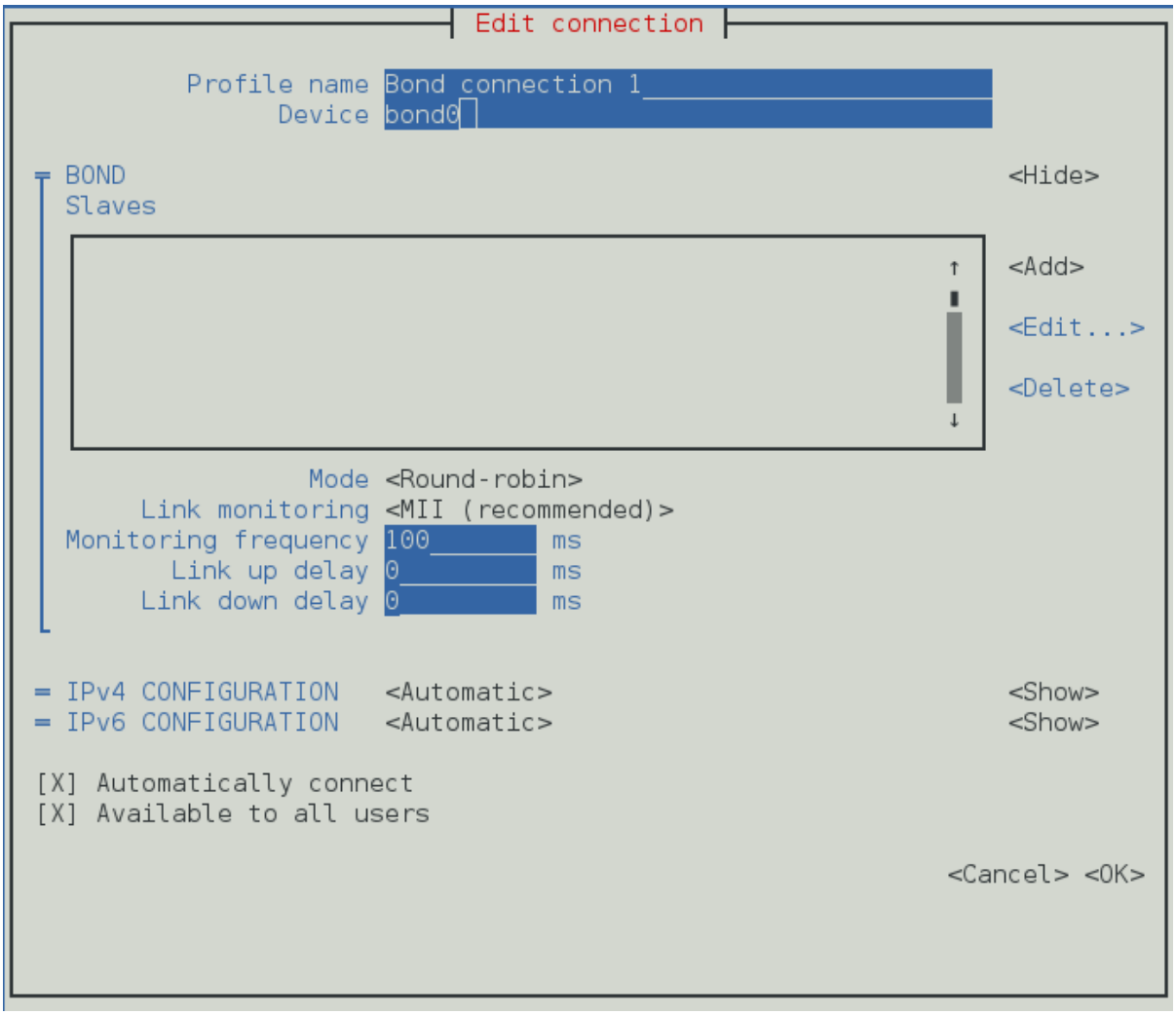
그림 7.1. NetworkManager 텍스트 사용자 인터페이스 본딩 연결 메뉴 추가



[D]

2. **Bond** 를 선택한 다음 **Create** (생성)를 선택합니다. 본딩의 연결 편집 화면이 열립니다.

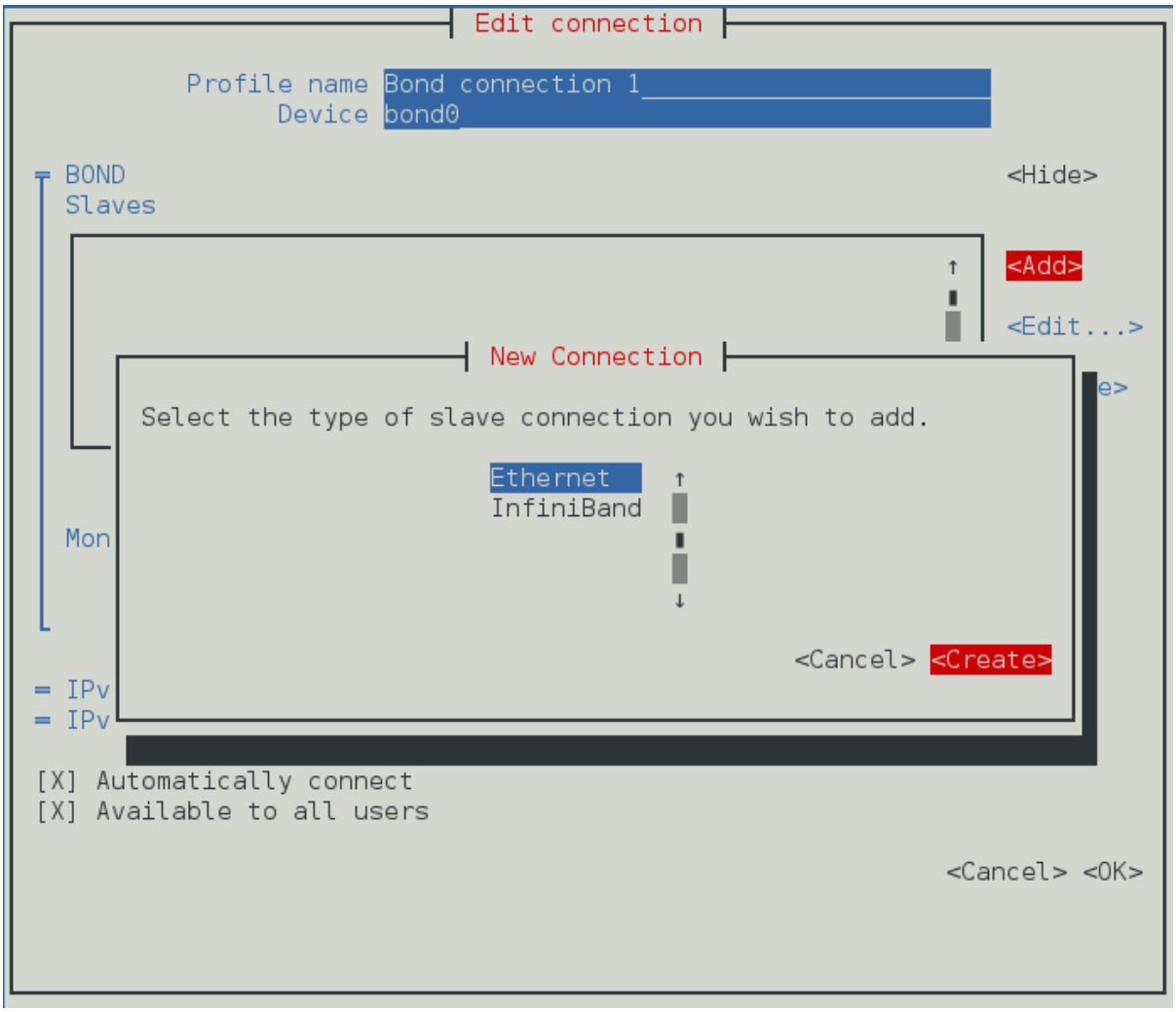
그림 7.2. NetworkManager 텍스트 사용자 인터페이스 본딩 연결 메뉴 구성



[D]

3. **이 시점에서 포트 인터페이스를 bond에 추가해야 합니다. 이러한 선택 추가 를 추가하려면 새 연결 화면이 열립니다. 연결 유형을 선택한 후 Create (만들기) 단추를 선택합니다.**

그림 7.3. NetworkManager 텍스트 사용자 인터페이스 새 본딩 연결 설정



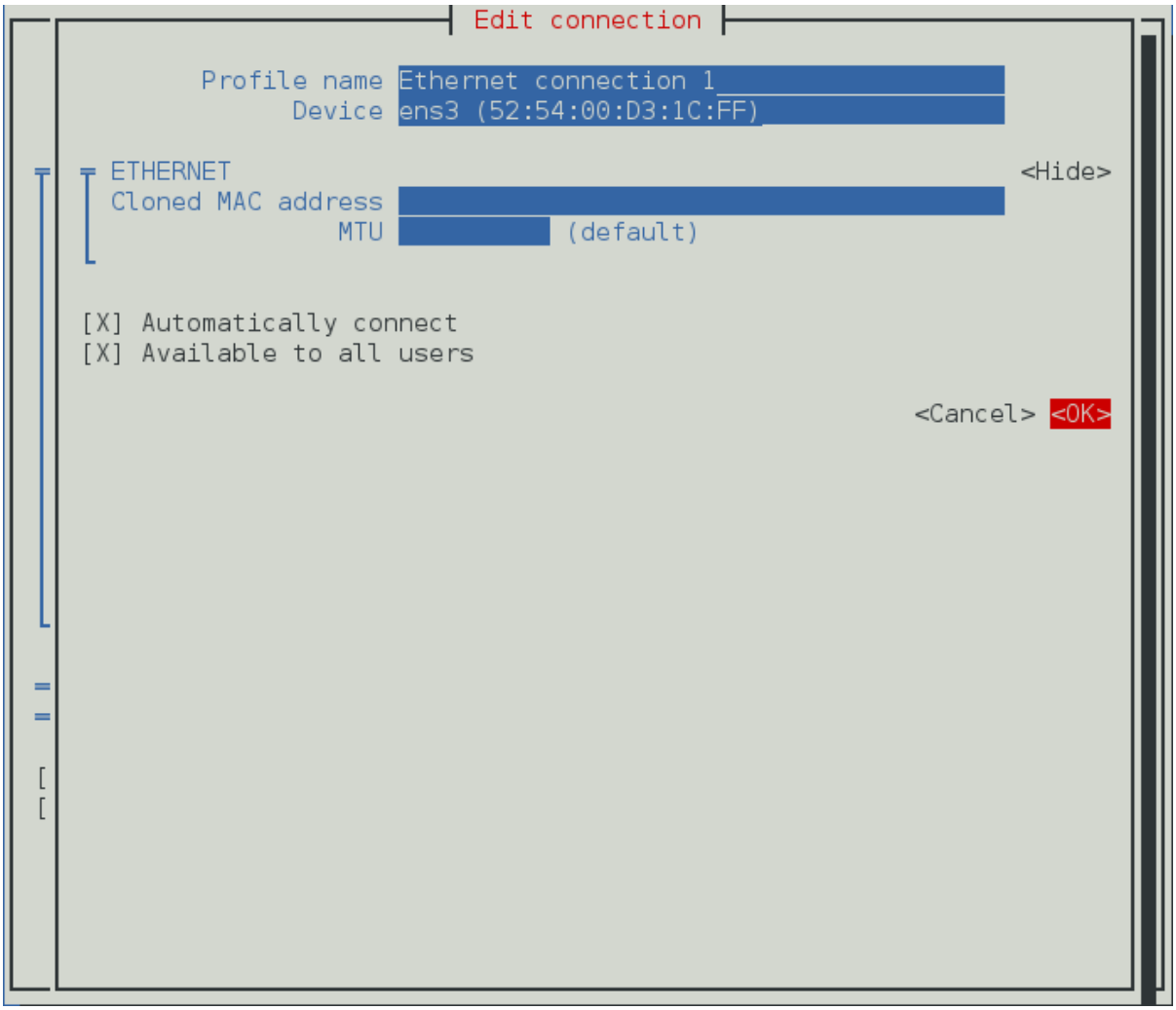
[D]

4. 포트의 연결 편집 표시가 나타납니다. 장치 섹션에 필요한 포트의 장치 이름 또는 **MAC** 주소를 입력합니다. 필요한 경우 이더넷 레이블의 오른쪽에 표시를 선택하여 본딩의 **MAC** 주소로 사용할 복제 **MAC** 주소를 입력합니다. **OK** 버튼을 선택하여 포트를 저장합니다.

참고

MAC 주소 없이 장치를 지정하면 장치 섹션이 자동으로 입력되며 **Edit Connection**(연결 편집) 창이 다시 로드되지만 장치를 성공적으로 찾은 경우에만 가능합니다.

그림 7.4. NetworkManager 텍스트 사용자 인터페이스 본딩 연결 메뉴 구성

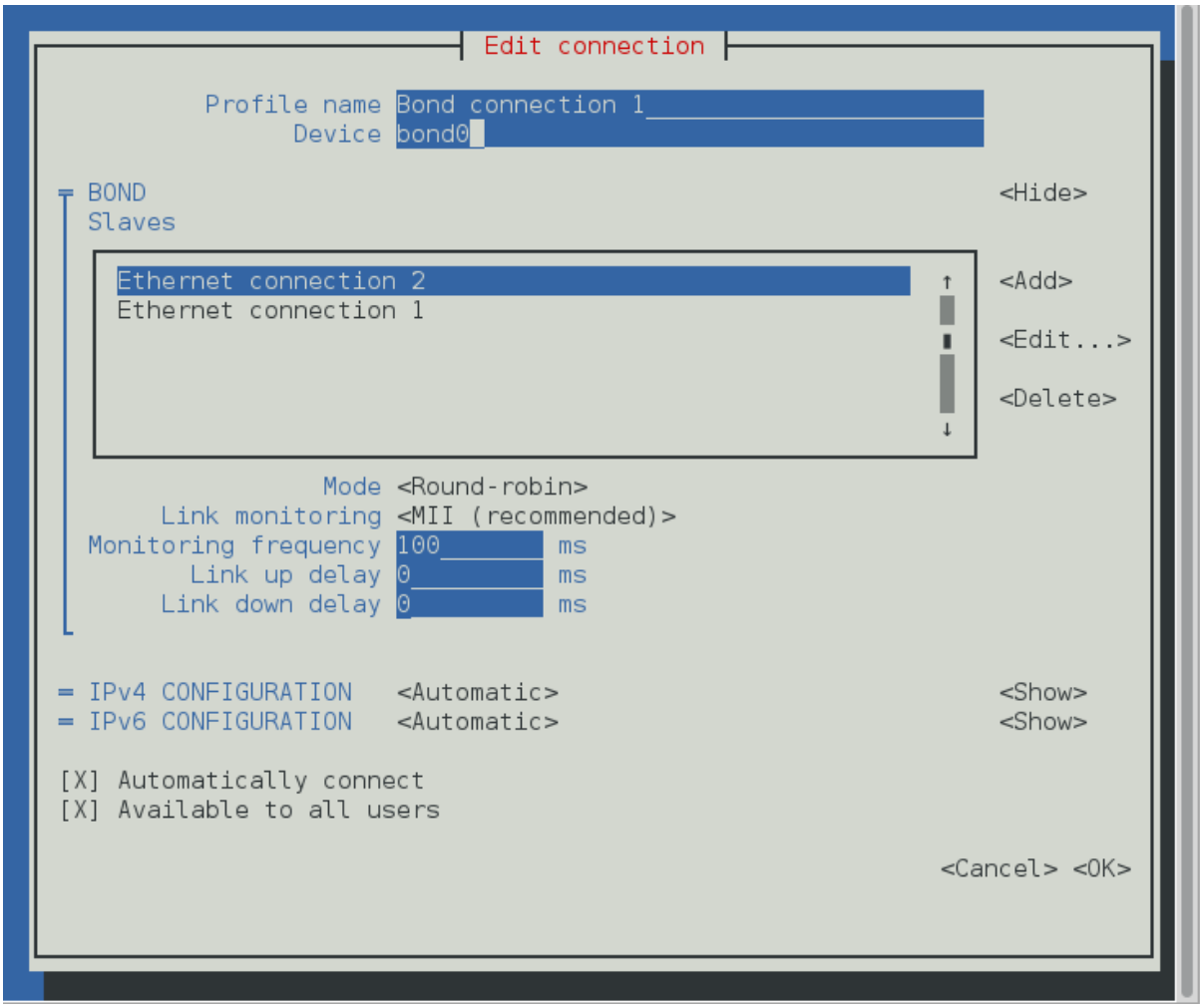


[D]

5. 본딩 포트의 이름이 **Slaves** 섹션에 표시됩니다. 위의 단계를 반복하여 추가 포트 연결을 추가합니다.

6. **OK(확인)** 버튼을 선택하기 전에 설정을 검토하고 확인합니다.

그림 7.5. NetworkManager 텍스트 사용자 인터페이스 완료된 본딩

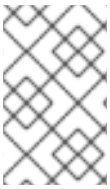


[D]

본드 용어에 대한 정의는 7.8.1.1절. “본딩 탭 구성” 을 참조하십시오.

nmtui 설치에 대한 자세한 내용은 3.2절. “nmtui로 IP 네트워킹 구성” 을 참조하십시오.

7.3. NETWORKMANAGER 명령줄 도구를 사용하여 네트워크 연결 도구 NMCLI



참고

nmcli 에 대한 소개는 3.3절. “nmcli로 IP 네트워킹 구성” 을 참조하십시오.

nmcli 툴로 본딩 연결을 생성하려면 다음 명령을 실행합니다.

```
~]$ nmcli con add type bond ifname mybond0
Connection 'bond-mybond0' (5f739690-47e8-444b-9620-1895316a28ba) successfully added.
```

본딩에 대해 **Con -name** 을 지정하지 않아 연결 이름은 유형 앞에 따라 인터페이스 이름에서 파생되었습니다.

NetworkManager 는 커널에서 제공하는 대부분의 본딩 옵션을 지원합니다. 예를 들어 다음과 같습니다.

```
~]$ nmcli con add type bond ifname mybond0 bond.options "mode=balance-rr,miimon=100"
Connection 'bond-mybond0' (5f739690-47e8-444b-9620-1895316a28ba) successfully added.
```

포트 인터페이스를 추가하려면 다음을 수행합니다.

1. 새 연결을 만들려면 **3.3.5절. "nmcli를 사용하여 연결 프로필 생성 및 수정"** 을 참조하십시오. 자세한 내용은 을 참조하십시오.

2. **controller** 속성을 본딩 인터페이스 이름 또는 컨트롤러 연결 이름으로 설정합니다.

```
~]$ nmcli con add type ethernet ifname ens3 master mybond0
Connection 'bond-slave-ens3' (220f99c6-ee0a-42a1-820e-454cbabc2618) successfully added.
```

새 포트 인터페이스를 추가하려면 새 인터페이스를 사용하여 이전 명령을 반복합니다. 예를 들어 다음과 같습니다.

```
~]$ nmcli con add type ethernet ifname ens7 master mybond0
Connection 'bond-slave-ens7' (ecc24c75-1c89-401f-90c8-9706531e0231) successfully added.
```

포트를 활성화하려면 다음과 같이 명령을 실행합니다.

```
~]$ nmcli con up bond-slave-ens7
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/14)
```

```
~]$ nmcli con up bond-slave-ens3
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/15)
```

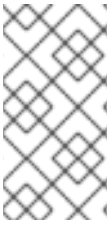
포트를 활성화하면 컨트롤러 연결도 시작됩니다. 자세한 내용은 7.1절. “기본 컨트롤러 동작 및 포트 인터페이스 이해” 를 참조하십시오. 이 경우 컨트롤러 연결을 수동으로 활성화할 필요가 없습니다.

연결을 비활성화하지 않고 런타임에 **active_slave** 옵션과 본딩의 기본 옵션을 변경할 수 있습니다. 예를 들어 **active_slave** 옵션을 변경하려면 다음 명령을 실행합니다.

```
~]$ nmcli dev mod bond0 +bond.options "active_slave=ens7"
Connection successfully reapplied to device 'bond0'.
```

또는 기본 옵션을 변경합니다.

```
~]$ nmcli dev mod bond0 +bond.options "primary=ens3"
Connection successfully reapplied to device 'bond0'.
```



참고

active_slave 옵션은 현재 활성 포트를 설정하지만 본딩의 기본 옵션은 새 포트를 추가하거나 활성 포트의 실패가 발생할 때 커널에서 자동으로 선택할 활성 포트를 지정합니다.

7.4. 명령줄 인터페이스(CLI) 사용

본딩 커널 모듈과 *채널 본딩* 인터페이스라는 특수 네트워크 인터페이스를 사용하여 본딩이 생성됩니다.

7.4.1. 본딩 커널 모듈이 설치되었는지 확인

Red Hat Enterprise Linux 7에서는 기본적으로 **bonding** 모듈이 로드되지 않습니다. **root** 로 다음 명령을 실행하여 모듈을 로드할 수 있습니다 :

```
~]# modprobe --first-time bonding
```

이 활성화는 시스템을 재시작해도 유지되지 않습니다. 영구적인 모듈 로드에 대한 설명은 ***Red Hat Enterprise Linux 커널 관리 가이드*** 를 참조하십시오. **BONDING_OPTS** 지시문을 사용하여 올바른 구성 파일을 제공하면 필요에 따라 **bonding** 모듈이 로드되므로 별도로 로드할 필요가 없습니다.

모듈에 대한 정보를 표시하려면 다음 명령을 실행합니다.

```
~]$ modinfo bonding
```

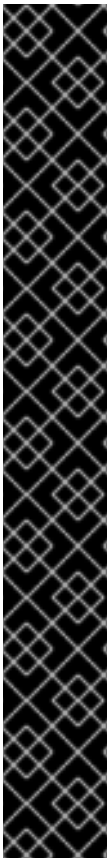
자세한 명령 옵션은 **modprobe(8)** 도움말 페이지를 참조하십시오.

7.4.2. 채널 연결 인터페이스 만들기

채널 본딩 인터페이스를 만들려면 **/etc/sysconfig/network-scripts/** 디렉터리에 **ifcfg- bond*N*이라는 *파일을 만들고 **N** 을 인터페이스의 번호로 바꿉니다(예: 0).*

*파일의 내용은 이더넷 인터페이스와 같이 결합되는 인터페이스 유형에 대한 구성 파일을 기반으로 할 수 있습니다. 중요한 차이점은 **DEVICE** 지시문은 **bondN**이고 **N** 을 인터페이스의 번호로, **TYPE=Bond**라는 것입니다. 또한 **set BONDING_MASTER=yes.***

> *예 7.1. **ifcfg-bond0** 인터페이스 구성 파일 예*
>
> *채널 본딩 인터페이스의 예.*
>
> ```
> DEVICE=bond0
> NAME=bond0
> TYPE=Bond
> BONDING_MASTER=yes
> IPADDR=192.168.1.1
> PREFIX=24
> ONBOOT=yes
> BOOTPROTO=none
> BONDING_OPTS="bonding parameters separated by spaces"
> NM_CONTROLLED="no"
> ```
>
> ***NAME** 지시문은 **NetworkManager** 에서 연결 프로필 이름을 지정하는 데 유용합니다. **ONBOOT** 는 부팅 시 프로필을 시작해야 하는지 여부(또는 장치를 자동 연결할 때)를 나타냅니다.*



중요

bonding kernel 모듈의 매개 변수는 **BONDING_OPTS="본딩 매개 변수" ifcfg-bondN** 인터페이스 파일의 지시어를 공백으로 구분된 목록으로 지정해야 합니다. **/etc/modprobe.d/ bonding.conf**에서본딩 장치 또는 더 이상 사용되지 않는 **/etc/modprobe.conf** 파일에 옵션을 지정하지 마십시오.

max_bonds 매개변수는 인터페이스별로 지정되지 않으며 **ifcfg-bondN** 파일을 **BONDING_OPTS** 지시문과 함께 사용하면 안 됩니다. 이 지시문으로 인해 네트워크 스크립트가 필요에 따라 본딩 인터페이스를 만들 수 있습니다.

bonding 모듈을 구성하고 본딩 매개 변수 목록을 확인하는 방법에 대한 자세한 지침과 지침은 *7.7절. “채널 연결 사용”* 에서 참조하십시오.

NM_CONTROLLED="no" 설정이 없으면 **NetworkManager**가 이 구성 파일의 설정을 재정의할 수 있습니다.

7.4.3. 포트 인터페이스 만들기

채널 본딩 인터페이스는 컨트롤러(마스터라고도 함)이며 결합할 인터페이스를 포트(슬레이브)라고 합니다. 채널 본딩 인터페이스를 만든 후에는 **MASTER** 및 **SLAVE** 지시문을 포트의 구성 파일에 추가하여 함께 바인딩되는 네트워크 인터페이스를 구성해야 합니다. 각 포트 인터페이스에 대한 구성 파일은 거의 같을 수 있습니다.

예 7.2. 포트 인터페이스 구성 파일 예

예를 들어 두 개의 이더넷 인터페이스가 채널 본딩, **enp1s0** 및 **enp2s0** 인 경우 둘 다 다음 예와 같을 수 있습니다.

```
DEVICE=device_name
NAME=bond0-slave
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
NM_CONTROLLED="no"
```

이 예에서 **device_name** 을 인터페이스 이름으로 교체합니다. 인터페이스에 대해 **ONBOOT=yes** 와 함께 둘 이상의 프로필 또는 구성 파일이 있는 경우 본딩 포트 대신 일반 **TYPE=Ethernet** 프로필이 활성화될 수 있습니다.

참고

***NM_CONTROLLED="no"** 설정이 없으면 **NetworkManager**가 이 구성 파일의 설정을 재정의할 수 있습니다.*

7.4.4. 채널 연결 활성화

*본딩을 활성화하려면 모든 포트를 엽니다. **root**로 다음 명령을 실행합니다.*

```
~]# ifup ifcfg-enp1s0
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/7)
```

```
~]# ifup ifcfg-enp2s0
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/8)
```

현재 실행 중인 인터페이스의 인터페이스 파일을 편집하면 다음과 같이 먼저 아래로 "설정합니다".

```
ifdown device_name
```

그런 다음 완료되면 모든 포트를 열어 본딩을 엽니다(설정되지 않은 경우). ""

***NetworkManager** 에서 변경 사항을 인식할 수 있도록 **root** 로 변경된 모든 인터페이스에 대해 명령을 실행합니다:*

```
~]# nmcli con load /etc/sysconfig/network-scripts/ifcfg-device
```

또는 모든 인터페이스를 다시 로드하려면 다음을 실행합니다.

```
~]# nmcli con reload
```

*기본 동작은 **NetworkManager** 가 변경 사항을 인식하지 못하고 이전 구성 데이터를 계속 사용하는 것입니다. 이는 **NetworkManager.conf** 파일의 **monitor-connection-files** 옵션으로 설정합니다. 자세한 내용은 **NetworkManager.conf(5)** 매뉴얼 페이지를 참조하십시오.*

본딩 인터페이스의 상태를 보려면 다음 명령을 실행합니다.

```
~]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp1s0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master bond0 state UP mode DEFAULT qlen 1000
    link/ether 52:54:00:e9:ce:d2 brd ff:ff:ff:ff:ff:ff
3: enp2s0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master bond0 state UP mode DEFAULT qlen 1000
    link/ether 52:54:00:38:a6:4c brd ff:ff:ff:ff:ff:ff
4: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP mode DEFAULT
    link/ether 52:54:00:38:a6:4c brd ff:ff:ff:ff:ff:ff
```

7.4.5. 여러 본딩 생성

Red Hat Enterprise Linux에서는 각 본딩에 대해 BONDING_OPTS 지시문을 포함하여 채널 본딩 인터페이스가 생성됩니다. 이 구성 방법은 여러 개의 본딩 장치에 다른 구성이 있을 수 있도록 사용됩니다. 여러 채널 본딩 인터페이스를 생성하려면 다음과 같이 진행하십시오.

- BONDING_OPTS 지시문을 사용하여 여러 ifcfg-bondN 파일을 만듭니다. 이 지시어를 사용하면 네트워크 스크립트가 필요에 따라 본딩 인터페이스를 만듭니다.
- 본딩할 기존 인터페이스 구성 파일을 만들거나 편집하고 SLAVE 지시문을 포함합니다.
- MASTER 지시문을 통해 본딩될 인터페이스인 포트 인터페이스를 채널 본딩 인터페이스에 할당합니다.

예 7.3. 여러 ifcfg-bondN 인터페이스 구성 파일의 예

다음은 채널 본딩 인터페이스 구성 파일의 예입니다.

```
DEVICE=bondN
NAME=bondN
TYPE=Bond
BONDING_MASTER=yes
```

```
IPADDR=192.168.1.1
PREFIX=24
ONBOOT=yes
BOOTPROTO=none
BONDING_OPTS="bonding parameters separated by spaces"
```

이 예에서 **N** 을 본딩 인터페이스의 번호로 바꿉니다. 예를 들어, 두 개의 본딩을 만들려면 **ifcfg-bond0** 및 **ifcfg-bond 1** 의 두 개의 구성 파일을 만들고 적절한 **IP** 주소를 만듭니다.

예 7.2. "포트 인터페이스 구성 파일 예" 에 따라 결합된 인터페이스를 생성하고 **MASTER=bondN** 지시문을 사용하여 필요에 따라 본딩 인터페이스에 할당합니다. 예를 들어 위의 예에서 본딩당 두 개의 인터페이스가 필요한 경우 두 개의 본딩에서 네 개의 인터페이스 구성 파일을 만들고 **MASTER=bond0**을 사용하여 처음 두 개를 할당하고 **MASTER=bond 1** 을 사용하여 다음 두 개를 할당합니다.

7.5. 중복을 위한 네트워크 구성 연결 확인

네트워크 중복은 장치가 백업 목적으로 사용되어 특정 시스템의 장애를 방지하거나 복구하는 프로세스입니다. 다음 절차에서는 중복성에서 본딩을 위해 네트워크 구성을 확인하는 방법을 설명합니다.

절차

1. 본딩 인터페이스에서 대상 **IP**를 **ping**합니다. 예를 들어 다음과 같습니다.

```
~]# ping -I bond0 DSTADDR
```

2. 활성 모드에 있는 인터페이스를 확인합니다.

```
~]# cat /sys/class/net/bond0/bonding/active_slave
enp1s0
```

enp1s0 은 활성 포트 인터페이스입니다.

3. 활성 포트 인터페이스를 아래의 아래로 설정합니다.

```
~]# ip link set enp1s0 down
```

4. 백업 인터페이스가 작동 중인지 확인합니다.

```
~]# cat /sys/class/net/bond0/bonding/active_slave
enp2s0
```

enp2s0 은 이제 활성 포트 인터페이스입니다.

5. 본딩 인터페이스에서 대상 **IP**를 계속 **ping**할 수 있는지 확인합니다.

```
~]# ping -I bond0 DSTADDR
```

7.6. 연결 모드 및 스위치의 필수 설정 개요

다음 테이블에서는 본딩 모드에 따라 업스트림 스위치에 적용해야 하는 필수 구성을 설명합니다.

표 7.1. 연결 모드에 따라 구성 설정

본딩 모드	스위치의 구성
0 - **balance-rr**	정적 Etherchannel 활성화 필요 (LACP 협상 안 함)
1 - **active-backup**	자율 포트 필요
2 - **balance-xor**	정적 Etherchannel 활성화 필요 (LACP 협상 안 함)
3 - **broadcast**	정적 Etherchannel 활성화 필요 (LACP 협상 안 함)
4 - **802.3ad**	LACP 협상 Etherchannel 사용 필요
5 - **balance-tlb**	자율 포트 필요
6 - **balance-alb**	자율 포트 필요

스위치에서 이러한 설정을 구성하려면 스위치 설명서를 참조하십시오.

7.7. 채널 연결 사용

성능을 향상하려면 사용 가능한 모듈 옵션을 조정하여 가장 잘 작동하는 조합을 확인합니다. **miimon** 또는 **arp_interval** 및 **arp_ ip_target** 매개변수에 특히 주의하십시오. 사용 가능한 옵션 목록 및 연결된 인터페이스에 가장 적합한 옵션 목록을 신속하게 확인하는 방법은 7.7.1절. “본딩 모듈 지시문” 을 참조하십시오.

7.7.1. 본딩 모듈 지시문

BONDING_OPTS=" BONDING_OPTS=" 본딩 매개 변수 "연결 인터페이스 구성 파일의 지시문(예: **ifcfg-bond 0)**에 추가하기 전에 결합된 인터페이스에 가장 적합한 채널 본딩 모듈매개 변수를 테스트하는 것이 좋습니다. 본딩된 인터페이스에 대한 매개 변수는 **sysfs** 파일 시스템에서 파일을 조작하여 **bonding** 모듈을 언로드하거나 다시 로드하지 않고도 구성할 수 있습니다.

sysfs 는 커널 객체를 디렉토리, 파일 및 심볼릭 링크로 표시하는 가상 파일 시스템입니다. **sysfs** 는 커널 오브젝트에 대한 정보를 쿼리하는 데 사용할 수 있으며 일반 파일 시스템 명령을 사용하여 이러한 개체를 조작할 수도 있습니다. **sysfs** 가상 파일 시스템은 **/sys/** 디렉터리에 마운트됩니다. 모든 본딩 인터페이스는 **/sys/class/net/** 디렉토리에서 파일과 상호 작용하고 조작하여 동적으로 구성할 수 있습니다.

연결 인터페이스에 가장 적합한 매개 변수를 확인하려면 7.4.2절. “채널 연결 인터페이스 만들기” 의 지침에 따라 **ifcfg-bond0** 과 같은 채널 본딩 인터페이스 파일을 만듭니다. **SLAVE=yes** 및 **MASTER=bond0** 지시문을 **bond0** 에 결합하는 각 인터페이스의 구성 파일에 삽입합니다. 이 작업을 마치면 매개 변수 테스트를 진행할 수 있습니다.

먼저 **ifup** 본딩 **N**을 루트로 실행하여 만든 본딩 을 엽니다.

```
~]# ifup bond0
```

ifcfg-bond0 본딩 인터페이스 파일을 올바르게 만든 경우 실행 중인 **ip link**의 출력에 표시된 **bond0** 이 **root** 로 표시됩니다.

```
~]# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp1s0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
master bond0 state UP mode DEFAULT qlen 1000
    link/ether 52:54:00:e9:ce:d2 brd ff:ff:ff:ff:ff:ff
3: enp2s0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
```

```
master bond0 state UP mode DEFAULT qlen 1000
    link/ether 52:54:00:38:a6:4c brd ff:ff:ff:ff:ff:ff
4: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP mode DEFAULT
    link/ether 52:54:00:38:a6:4c brd ff:ff:ff:ff:ff:ff
```

기존의 본딩이 모두 표시되지 않더라도 기존 본딩을 모두 보려면 다음을 실행합니다.

```
~]$ cat /sys/class/net/bonding_masters
bond0
```

/sys/class/net/bondN/bonding/ 디렉토리에 있는 파일을 조작하여 각 본딩을 개별적으로 구성할 수 있습니다. 먼저 구성 중인 본딩을 제거해야 합니다.

```
~]# ifdown bond0
```

예를 들어 **1**초 간격으로 **bond0**에서 **MII** 모니터링을 활성화하려면 **root** 로 실행합니다.

```
~]# echo 1000 > /sys/class/net/bond0/bonding/miimon
```

balance-alb 모드에 대해 **bond0**을 구성하려면 다음 중 하나를 실행합니다.

```
~]# echo 6 > /sys/class/net/bond0/bonding/mode
```

또는 모드 이름 사용:

```
~]# echo balance-alb > /sys/class/net/bond0/bonding/mode
```

문제가 있는 본딩에 대한 옵션을 구성한 후 **ifup bondN** 을 실행하여 테스트하고 테스트할 수 있습니다. 옵션을 변경하려면 인터페이스를 작동 중단하고, **sysfs** 를 사용하여 매개 변수를 수정하고, 다시 작동시킨 후 다시 테스트합니다.

본딩에 가장 적합한 매개변수 세트를 결정한 후 해당 매개변수를 공백으로 구분된 목록으로 **/etc/sysconfig/network-scripts/ifcfg-bondN** 파일에 구성하려는 본딩 인터페이스의 **BONDING_OPTS=** 지시문에 추가합니다. 본딩이 가동될 때마다(예: **ONBOOT=yes** 지시문이 설정된 경우 부팅 시퀀스 중 시스템에 의해) **BONDING_OPTS** 에 지정된 본딩 옵션이 해당 연결에 적용됩니다.

다음 목록은 보다 일반적인 채널 본딩 매개 변수 중 많은 이름과 함께 수행하는 작업에 대한 설명을 제공합니다. 자세한 내용은 **modinfo bonding** 출력의 각 구문에 대한 간략한 설명을 참조하거나 자세한 내

용은 https://www.kernel.org/doc/Documentation/networking/bonding.txthttps://www.kernel.org/doc/Documentation/networking/bonding.txt 참조하십시오.

본딩 인터페이스 매개변수

ad_select=value

*사용할 **802.3ad** 집계 선택 논리를 지정합니다. 가능한 값은 다음과 같습니다.*

- *__stable__ 또는 __0__ - 기본 설정. 활성 집계 대역폭은 가장 큰 집계 대역폭에 의해 선택됩니다. 활성 집계기의 재선택은 활성 집계기의 모든 포트가 다운되거나 활성 수집기에 포트가 없는 경우에만 수행됩니다.*
- *__bandwidth__ 또는 __1__ - 활성 집계가 가장 큰 집계 대역폭에 의해 선택됩니다. 다음과 같은 경우 재선택이 수행됩니다.*
  - *본딩에 포트가 추가되거나 제거됩니다.*
  - *포트의 링크 상태가 변경됩니다.*
  - *모든 포트의 **802.3ad** 연결 상태가 변경됩니다.*
  - *본드의 관리 상태가 **up**으로 변경됩니다.*
- *__count__ 또는 __2__ - 활성 집계기는 가장 많은 포트 수에 의해 선택됩니다. 위의 대역폭 설정에 설명된 대로 재선택이 수행됩니다.*

*대역폭 및 개수 선택 정책에서는 활성 집계의 부분 장애가 발생할 때 **802.3ad** 집계에 대한 장애 조치(__failover__)를 허용합니다. 이를 통해 대역폭 또는 포트 수에서 항상 가용성이 가장 높은 가용성을 유지하는 데 도움이 됩니다.*

arp_interval=time_in_milliseconds

ARP 모니터링이 발생하는 빈도를 밀리초 단위로 지정합니다.

중요

arp_interval 및 arp_ ip_target 매개 변수가 지정되거나 miimon 매개 변수가 지정되어야 합니다. 이렇게 하지 않으면 링크가 실패하는 경우 네트워크 성능이 저하될 수 있습니다.

mode=0 또는 mode =2 (두 로드 밸런싱 모드)에서 이 설정을 사용하는 경우 NIC 간에 패킷을 균등하게 분산하도록 네트워크 스위치를 구성해야 합니다. 이 작업을 수행하는 방법에 대한 자세한 내용은 https://www.kernel.org/doc/Documentation/networking/bonding.txthttps://www.kernel.org/doc/Documentation/networking/bonding.txt을 참조하십시오.

이 값은 기본적으로 0 로 설정되어 있어 이를 비활성화합니다.

arp_ip_target=ip_address[,ip_address_2,...ip_address_16]

arp_interval 매개 변수가 활성화될 때 ARP 요청의 대상 IP 주소를 지정합니다. 최대 16개의 IP 주소를 쉼표로 구분된 목록으로 지정할 수 있습니다.

arp_validate=value

ARP 프로브의 소스/배포를 검증합니다. 기본값은 none 입니다. 기타 유효한 값은 active, backup 및 all 입니다.

downdelay=time_in_milliseconds

링크를 비활성화하기 전에 링크 실패 후 대기하는 시간(밀리초)을 지정합니다. 값은 miimon 매개 변수에 지정된 값의 배수여야 합니다. 이 값은 기본적으로 0 로 설정되어 있어 이를 비활성화합니다.

fail_over_mac=value

active-backup 모드가 모든 포트를 할당 시점(기존 동작)에서 동일한 MAC 주소로 설정하거나, 활성화된 경우 선택한 정책에 따라 본딩의 MAC 주소를 특별한 처리하도록 지정합니다. 가능한 값은 다음과 같습니다.

- *none 또는 0 - 기본 설정. 이 설정은 fail_over_mac 를 비활성화하고 본딩을 통해 active-backup 본딩의 모든 포트를 할당 시점에서 동일한 MAC 주소로 설정합니다.*

- *active 또는 1 - "활성" fail_over_mac 정책은 본딩의 MAC 주소가 항상 현재 활성 포트의 MAC 주소임을 나타냅니다. 포트의 MAC 주소는 변경되지 않습니다. 대신 장애 조치 중에 본딩의 MAC 주소가 변경됩니다.*

  이 정책은 MAC 주소를 변경할 수 없는 장치 또는 자체 소스 MAC로 들어오는 브로드캐스트를 거부하는 장치에 유용합니다(ARP 모니터를 방해함). 이 정책의 단점은 들어오는 트래픽 스누핑을 전환하여 ARP 표를 업데이트하는 일반적인 방법과 달리 네트워크의 모든 장치를 무료 ARP로 업데이트해야 한다는 것입니다. 무료 ARP가 손실되면 통신이 중단될 수 있습니다.

  이 정책이 MII 모니터와 함께 사용되는 경우 실제로 전송 및 수신되기 전에 링크를 요청하는 장치는 특히 무료 ARP 손실에 취약하며 적절한 updelay 설정이 필요할 수 있습니다.

- *follow 또는 2 - "다음" fail_over_mac 정책으로 인해 본딩의 MAC 주소가 정상적으로 선택됩니다(일반적으로 본딩에 추가된 첫 번째 포트의 MAC 주소). 그러나 백업 역할에 있는 동안 두 번째 및 후속 포트는 이 MAC 주소로 설정되지 않습니다. 포트는 장애 조치 시 본딩의 MAC 주소로 프로그래밍됩니다(이전의 활성 포트는 새로 활성 포트의 MAC 주소를 수신합니다).*

  이 정책은 여러 포트가 동일한 MAC 주소로 프로그래밍되는 경우 혼동되거나 성능 저하가 발생하는 다중 포트 장치에 유용합니다.

lacp_rate=value

802.3ad 모드에서 링크 파트너가 LACPDU 패킷을 전송해야 하는 속도를 지정합니다. 가능한 값은 다음과 같습니다.

- *slow 또는 0 - 기본 설정. 이는 파트너가 30초마다 LACPDU를 전송하도록 지정합니다.*

- *fast 또는 1 - 파트너가 1초마다 LACPDU를 전송하도록 지정합니다.*

miimon=time_in_milliseconds

MII 링크 모니터링이 발생하는 빈도(밀리초)를 지정합니다. MII가 NIC가 활성 상태인지 확인하는 데 사용되므로 고가용성이 필요한 경우 유용합니다. 특정 NIC의 드라이버가 MII 툴을 지원하는지 확인하려면 root로 다음 명령을 입력합니다.

```
~]# ethtool interface_name | grep "Link detected:"
```

이 명령에서 interface_name 을 본딩 인터페이스가 아닌 enp1s0 과 같은 장치 인터페이스 이름으로 바꿉니다. MII가 지원되는 경우 명령은 다음을 반환합니다.

```
Link detected: yes
```

고가용성을 위해 결합된 인터페이스를 사용하는 경우 각 NIC의 모듈은 MII를 지원해야 합니다. 값을 0 (기본값)로 설정하면 이 기능이 꺼집니다. 이 설정을 구성할 때 이 매개변수의 시작 지점은 100 입니다.

중요

arp_interval 및 arp_ ip_target 매개 변수가 지정되거나 miimon 매개 변수가 지정되어야 합니다. 이렇게 하지 않으면 링크가 실패하는 경우 네트워크 성능이 저하될 수 있습니다.

mode=value

본딩 정책을 지정할 수 있습니다. 값은 다음 중 하나일 수 있습니다.

- *balance-rr 또는 0 - 내결함성 및 로드 밸런싱을 위한 라운드 로빈 정책을 설정합니다. 전송은 첫 번째로 사용 가능한 결합된 포트 인터페이스에서 수신 및 순차적으로 전송됩니다.*

- *active-backup 또는 1 - 내결함성을 위한 active-backup 정책을 설정합니다. 첫 번째 사용 가능한 결합된 포트 인터페이스를 통해 전송이 수신되고 전송됩니다. 다른 본딩된 포트 인터페이스는 활성 본딩 포트 인터페이스가 실패한 경우에만 사용됩니다.*

- *balance-xor 또는 2 - 전송은 선택한 해시 정책을 기반으로 합니다. 기본값은 소스 및 대상 MAC 주소의 XOR에 포트 인터페이스 수를 곱한 해시를 파생하는 것입니다. 이 모드에서는 특정 피어를 향하는 트래픽이 항상 동일한 인터페이스를 통해 전송됩니다. 대상이 결정되*

기 때문에 이 방법은 동일한 링크 또는 로컬 네트워크에 있는 피어로의 트래픽에 가장 적합합니다. 트래픽이 단일 라우터를 통과해야 하는 경우 이 트래픽 분산 모드는 차선책이 됩니다.

- **broadcast** 또는 **3** - 내결함성에 대한 브로드캐스트 정책을 설정합니다. 모든 전송은 모든 포트 인터페이스에서 전송됩니다.

- **802.3ad** 또는 **4** - **IEEE 802.3ad** 동적 링크 집계 정책을 설정합니다. 동일한 속도 및 이중화된 설정을 공유하는 집계 그룹을 생성합니다. 활성 수집기의 모든 포트에서 전송 및 수신됩니다. **802.3ad** 호환 가능한 스위치 필요.

- **balance-tlb** 또는 **5** - 내결함성 및 로드 밸런싱을 위해 **Transmit Load Balancing(TLB)** 정책을 설정합니다. 발신 트래픽은 각 포트 인터페이스의 현재 부하에 따라 배포됩니다. 현재 포트에서 들어오는 트래픽을 수신합니다. 수신 포트가 실패하면 다른 포트가 실패한 포트의 **MAC** 주소를 넘겨받습니다. 이 모드는 커널 본딩 모듈에 알려진 로컬 주소에만 적합하므로 가상 시스템과 브리지 뒤에 사용할 수 없습니다.

- **balance-alb** 또는 **6** - 내결함성 및 로드 밸런싱을 위한 적응형 로드 밸런싱**(ALB)** 정책을 설정합니다. **IPv4** 트래픽에 대한 전송 및 부하 분산 수신을 포함합니다. **ARP** 협상을 통해 부하 분산을 달성합니다. 이 모드는 커널 본딩 모듈에 알려진 로컬 주소에만 적합하므로 가상 시스템과 브리지 뒤에 사용할 수 없습니다.

업스트림 스위치의 필수 설정에 대한 자세한 내용은 7.6절. “연결 모드 및 스위치의 필수 설정 개요”을 참조하십시오.

primary=interface_name

기본 장치의 인터페이스 이름(예: **enp1s0**)을 지정합니다. 기본 장치는 사용할 본딩 인터페이스 중 첫 번째이며 실패하는 경우가 아니면 문제가 되지 않습니다. 이 설정은 본딩 인터페이스에서 하나의 **NIC**가 더 빠르기 때문에 더 큰 부하를 처리할 수 있을 때 특히 유용합니다.

이 설정은 연결 인터페이스가 **active-backup** 모드에 있는 경우에만 유효합니다. 자세한 내용은 **https://www.kernel.org/doc/Documentation/networking/bonding.txt**
https://www.kernel.org/doc/Documentation/networking/bonding.txt을 참조하십시오.

primary_reselect=value

기본 포트에 대한 재선택 정책을 지정합니다. 이는 활성 포트가 실패하거나 기본 포트의 복구가 발생할 때 기본 포트가 활성 포트가 되도록 선택하는 방법에 영향을 미칩니다. 이 매개변수는 기본 포

트와 기타 포트 간 플립을 방지하도록 설계되었습니다. 가능한 값은 다음과 같습니다.

- **always** 또는 **0** (기본값) - 백업될 때마다 기본 포트가 활성 포트가 됩니다.
- **better** 또는 **1** - 백업 시 기본 포트가 활성 포트가 됩니다. 기본 포트의 속도와 **duplex**가 현재 활성 포트의 속도 및 중복보다 더 나은 경우
- **failure** 또는 **2** - 현재 활성 포트가 실패하고 기본 포트가 작동 중인 경우에만 기본 포트가 활성화됩니다.

primary_reselect 설정은 다음 두 가지 경우 무시됩니다.

- 활성 포트가 없으면 복구할 첫 번째 포트가 활성 포트가 됩니다.
- 처음에 본딩에 할당되면 기본 포트는 항상 활성 포트가 됩니다.

primary_reselect 정책을 **sysfs** 로 변경하면 새 정책에 따라 최상의 활성 포트가 즉시 선택됩니다. 상황에 따라 활성 포트가 변경될 수도 있고 변경되지 않을 수 있습니다.

resend_igmp=range

장애 조치(**failover**) 이벤트 이후에 발행할 **IGMP** 멤버십 보고서 수를 지정합니다. 하나의 멤버십 보고서는 장애 조치(**failover**) 직후 발행되며 이후의 패킷은 각 **200ms** 간격으로 전송됩니다.

유효한 범위는 **0** 에서 **255** 입니다. 기본값은 **1** 입니다. 값이 **0** 이면 장애 조치(**failover**) 이벤트에 대한 응답으로 **IGMP** 멤버십 보고서가 발행되지 않습니다.

이 옵션은 연결 모드 **balance-rr (mode 0), active-backup (mode 1), balance-tlb (mode 5)** 및 **balance-alb (mode 6))**에 유용하며, 이 경우 장애 조치를 통해 **IGMP** 트래픽을 한 포트에서 다른 포트로 전환할 수 있습니다. 따라서 새로운 **IGMP** 보고서를 발행하여 스위치가 새로 선택한 포트로 들어오는 **IGMP** 트래픽을 전달하도록 해야 합니다.

updelay=time_in_milliseconds

*링크를 활성화하기 전에 대기하는 시간(밀리초)을 지정합니다. 값은 **miimon** 매개 변수에 지정된 값의 배수여야 합니다. 이 값은 기본적으로 **0** 로 설정되어 있어 이를 비활성화합니다.*

use_carrier=number

miimon 이 MII/ETHTOOL ioctls 또는 netif_carrier_ok() 를 사용하여 링크 상태를 확인해야 하는지 여부를 지정합니다. netif_carrier_ok() 기능은 장치 드라이버를 사용하여 netif_carrier_on/off 를 사용하여 상태를 유지 관리합니다. 대부분의 장치 드라이버는 이 기능을 지원합니다.

MII/ETHTOOL ioctls 툴은 커널 내에서 더 이상 사용되지 않는 호출 시퀀스를 사용합니다. 그러나 장치 드라이버에서 netif_carrier_on/off 를 지원하지 않는 경우 이는 여전히 구성할 수 있습니다.

유효한 값은 다음과 같습니다.

- *1 - 기본 설정. netif_carrier_ok() 의 사용을 활성화합니다.*
- *0 - MII/ETHTOOL ioctls 사용을 활성화합니다.*

참고

본딩 인터페이스가 링크를 사용할 수 없을 때 작동한다고 주장하는 경우 네트워크 장치 드라이버가 netif_carrier_on/off 를 지원하지 않을 수 있습니다.

xmit_hash_policy=value

balance-xor 및 802.3ad 모드에서 포트 선택에 사용되는 전송 해시 정책을 선택합니다. 가능한 값은 다음과 같습니다.

- *0 또는 layer2 - 기본 설정. 이 매개 변수는 하드웨어 MAC 주소의 XOR을 사용하여 해시를 생성합니다. 사용되는 공식은 다음과 같습니다.*

  ```
  (source_MAC_address XOR destination_MAC) MODULO slave_count
  ```

  이 알고리즘은 모든 트래픽을 동일한 포트에 특정 네트워크 피어에 배치하며 802.3ad 준수입니다.

- *1 또는 layer3+4 - 상위 계층 프로토콜 정보(사용 가능한 경우)를 사용하여 해시를 생성합니다. 이렇게 하면 단일 연결이 여러 포트에 걸쳐 있지 않지만 특정 네트워크 피어에 대한 트래픽이 여러 포트에 걸쳐 있을 수 있습니다.*

  사용된 TCP 및 UDP 패킷의 공식은 다음과 같습니다.

  ```
  ((source_port XOR dest_port) XOR
    ((source_IP XOR dest_IP) AND 0xffff)
      MODULO slave_count
  ```

  조각화된 TCP 또는 UDP 패킷 및 기타 모든 IP 프로토콜 트래픽의 경우 소스 및 대상 포트 정보가 생략됩니다. 비IP 트래픽의 경우 수식은 layer2 전송 해시 정책과 동일합니다.

  이 정책은 특정 스위치의 동작을 모방하려고 합니다. 특히 Cisco는 PFC2로 전환하고 일부 중추적 및 IBM 제품을 사용합니다.

  이 정책에서 사용하는 알고리즘은 802.3ad와 호환되지 않습니다.

- *2 또는 layer2+3 - layer2 및 layer3 프로토콜 정보를 조합하여 해시를 생성합니다.*

  하드웨어 MAC 주소 및 IP 주소의 XOR을 사용하여 해시를 생성합니다. 공식은 다음과 같습니다.

  ```
  (((source_IP XOR dest_IP) AND 0xffff) XOR
    ( source_MAC XOR destination_MAC ))
      MODULO slave_count
  ```

  이 알고리즘은 모든 트래픽을 동일한 포트에 특정 네트워크 피어에 배치합니다. 비IP 트래픽의 경우 수식은 layer2 전송 해시 정책과 동일합니다.

  이 정책은 특히 대부분의 대상에 도달하기 위해 계층3 게이트웨이 장치가 필요한 환경에서 계층2보다 더 균형 있는 트래픽 배포를 제공하기 위한 것입니다.

  이 알고리즘은 802.3ad와 호환됩니다.

7.8. GUI를 사용하여 본딩 연결 만들기

GNOME 제어 센터 유틸리티를 사용하여 NetworkManager 에 지시하여 두 개 이상의 Wired 또는 InfiniBand 연결에서 본딩을 생성할 수 있습니다. 먼저 본딩할 연결을 만들 필요가 없습니다. 본딩을 구성하기 위해 프로세스의 일부로 구성할 수 있습니다. 구성 프로세스를 완료하려면 인터페이스의 MAC 주소를 사용할 수 있어야 합니다.

7.8.1. 본딩 연결 설정

절차 7.1. 새로운 Bond Connection_ Using nm-connection-editor 추가

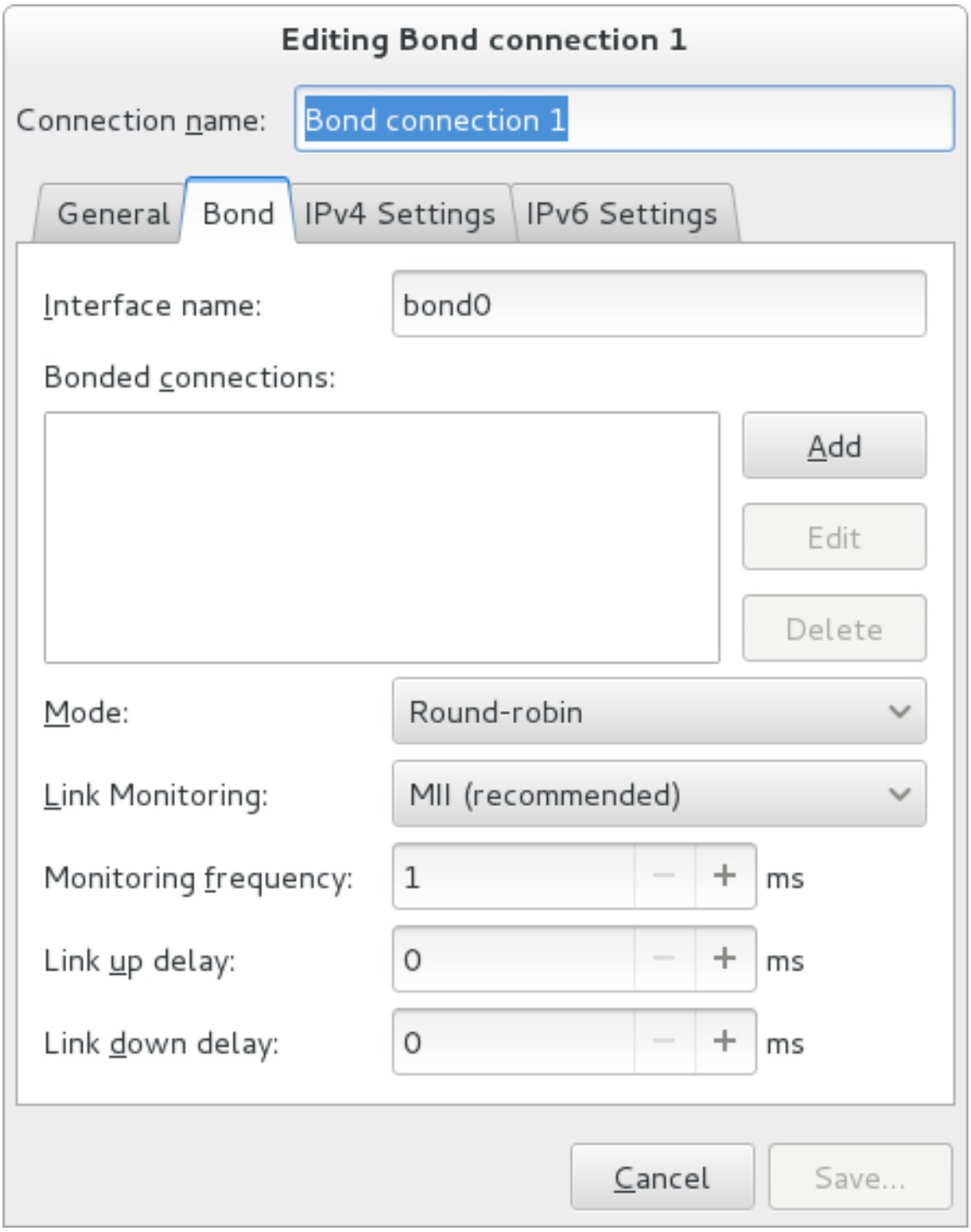
새 본딩 연결을 생성하려면 다음 단계를 수행하십시오.

1. 터미널에서 nm-connection-editor 를 입력합니다.

```
~]$ nm-connection-editor
```

2. Add(추가) 단추를 클릭합니다. Choose a Connection Type(연결 유형 선택) 창이 표시됩니다. Bond 를 선택하고 Create(생성)를 클릭합니다. Editing Bond connection 1(본딩 연결 편집) 1 창이 표시됩니다.

그림 7.6. NetworkManager 그래픽 사용자 인터페이스 추가 본딩 메뉴



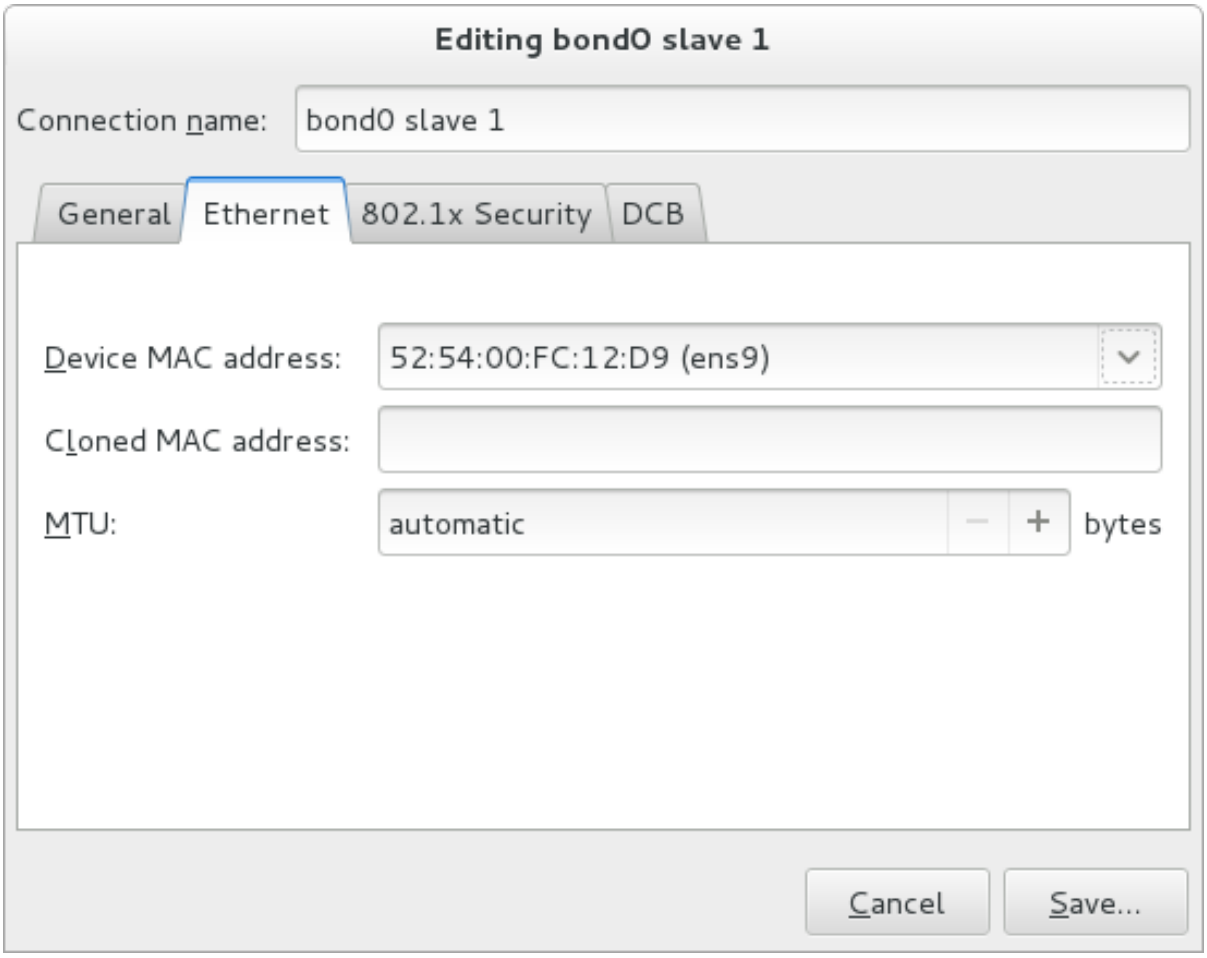
[D]

3. ***Bond** 탭에서 **Add**(추가)를 클릭하고 본딩 연결에 사용할 인터페이스 유형을 선택합니다. **Create**(만들기) 단추를 클릭합니다. 포트 유형을 선택하는 대화 상자가 첫 번째 포트를 만들 때만 나타납니다. 그런 다음 모든 추가 포트에 대해 동일한 유형을 자동으로 사용합니다.*

4. ***Editing bond0 slave 1**(**bond0** 슬레이브 **1** 편집) 창이 표시됩니다. 장치 **MAC** 주소 드롭다운 메뉴를 사용하여 결합할 인터페이스의 **MAC** 주소를 선택합니다. 첫 번째 포트의 **MAC** 주소는*

*본딩 인터페이스의 **MAC** 주소로 사용됩니다. 필요한 경우 본딩의 **MAC** 주소로 사용할 복제본 **MAC** 주소를 입력합니다. **Save**(저장) 버튼을 클릭합니다.*

*그림 7.7. **NetworkManager** 그래픽 사용자 인터페이스 **Add a Bond Connection** 메뉴*



[D]

5. *본딩된 포트의 이름이 **Bonded** 연결 창에 표시됩니다. **Add**(추가) 버튼을 클릭하여 추가 포트 연결을 추가합니다.*

6. *설정을 검토 및 확인한 다음 **Save** (저장) 단추를 클릭합니다.*

7. *아래의 7.8.1.1절. “본딩 탭 구성” 을 참조하여 본딩별 설정을 편집합니다.*

*절차 **7.2.** 기존 본딩 연결 편집*

다음 단계에 따라 기존 본딩 연결을 편집합니다.

1. *터미널에서 **nm-connection-editor** 를 입력합니다.*

```
~]$ nm-connection-editor
```

2. 편집할 연결을 선택하고 **Edit** (편집) 단추를 클릭합니다.

3. **General**(일반) 탭을 선택합니다.

4. 연결 이름, 자동 연결 동작 및 가용성 설정을 구성합니다.

편집 대화 상자의 **5**가지 설정은 모든 연결 유형에 공통적입니다. **General**(일반) 탭을 참조하십시오.

- 연결 이름 **-** 네트워크 연결에 대한 설명이 포함된 이름을 입력합니다. 이 이름은 **Network** (네트워크) 창의 메뉴에 이 연결을 나열하는 데 사용됩니다.

- 사용 가능한 경우 이 네트워크에 자동으로 연결 **-** 사용 가능한 경우 **NetworkManager** 가 이 연결에 자동으로 연결되도록 하려면 이 상자를 선택합니다. 자세한 내용은 "제어 센터를 사용하여 기존 연결 편집"을 참조하십시오.

- 모든 사용자가 이 네트워크에 연결할 수 있습니다 **-** 이 상자를 선택하여 시스템의 모든 사용자가 사용할 수 있는 연결을 만듭니다. 이 설정을 변경하려면 **root** 권한이 필요할 수 있습니다. 자세한 내용은 **3.4.5**절. "**GUI**를 사용하여 시스템 전체 및 개인 연결 프로파일 관리" 을 참조하십시오.

- 이 연결을 사용할 때 **VPN**에 자동으로 연결 **- NetworkManager** 가 **VPN** 연결에 자동으로 연결되도록 하려면 이 상자를 선택합니다. 드롭다운 메뉴에서 **VPN**을 선택합니다.

- 방화벽 영역 **-** 드롭다운 메뉴에서 방화벽 영역을 선택합니다. 방화벽 영역에 대한 자세한 내용은 **Red Hat Enterprise Linux 7** 보안 가이드 를 참조하십시오.

5. 아래의 **7.8.1.1**절. "본딩 탭 구성" 을 참조하여 본딩별 설정을 편집합니다.

새 연결 (또는 수정된) 연결 저장 및 추가 설정 만들기

본딩 연결 편집을 마치면 **Save** (저장) 버튼을 클릭하여 사용자 지정 구성을 저장합니다.

그런 다음 다음을 구성하려면 다음을 수행합니다.

- 연결의 **IPv4** 설정, **IPv4** 설정 탭을 클릭하고 계속 진행합니다. 5.4절. "IPv4 설정 구성"

  또는

- 연결의 **IPv6** 설정, **IPv6** 설정 탭을 클릭하고 5.5절. "IPv6 설정 구성" 진행합니다.

7.8.1.1. 본딩 탭 구성

새 본딩 연결을 이미 추가한 경우(명령에 대해서는 절차 7.1. "새로운 **Bond Connection_ Using nm-connection-editor** 추가" 참조) **Bond** 탭을 편집하여 부하 공유 모드 및 포트 연결 실패를 감지하는 데 사용할 링크 모니터링 유형을 설정할 수 있습니다.

모드

본딩을 구성하는 포트 연결을 통해 트래픽을 공유하는 데 사용되는 모드입니다. 기본값은 **Round-robin** 입니다. **802.3ad** 와 같은 기타 부하 공유 모드는 드롭다운 목록을 통해 선택할 수 있습니다.

링크 모니터링

네트워크 트래픽을 전송하는 포트 기능을 모니터링하는 방법입니다.

다음과 같은 로드 공유 모드는 모드 드롭다운 목록에서 선택할 수 있습니다.

라운드 로빈

내결함성 및 로드 밸런싱에 대한 라운드 로빈 정책을 설정합니다. 전송은 첫 번째로 사용 가능한 결합된 포트 인터페이스에서 수신 및 순차적으로 전송됩니다. 이 모드는 추가 스위치 구성 없이 가상 시스템을 사용하는 브리지 뒤에 작동하지 않을 수 있습니다.

활성 백업

내결함성에 대한 **active-backup** 정책을 설정합니다. 첫 번째 사용 가능한 결합된 포트 인터페이스를 통해 전송이 수신되고 전송됩니다. 다른 본딩된 포트 인터페이스는 활성 본딩 포트 인터페이스가 실패한 경우에만 사용됩니다. **InfiniBand** 장치 본딩에 사용할 수 있는 유일한 모드입니다.

XOR

XOR(특정 또는) 정책을 설정합니다. 전송은 선택한 해시 정책을 기반으로 합니다. 기본값은 소스 및 대상 **MAC** 주소의 **XOR**에 포트 인터페이스 수를 곱한 해시를 파생하는 것입니다. 이 모드에서는 특정 피어를 향하는 트래픽이 항상 동일한 인터페이스를 통해 전송됩니다. 대상이 결정되기 때문에 이 방법은 동일한 링크 또는 로컬 네트워크에 있는 피어로의 트래픽에 가장 적합합니다. 트래픽이 단일 라우터를 통과해야 하는 경우 이 트래픽 분산 모드는 차선책이 됩니다.

브로드캐스트

내결함성을 위한 브로드캐스트 정책을 설정합니다. 모든 전송은 모든 포트 인터페이스에서 전송됩니다. 이 모드는 추가 스위치 구성 없이 가상 시스템을 사용하는 브리지 뒤에 작동하지 않을 수 있습니다.

802.3ad

IEEE 802.3ad 동적 링크 집계 정책을 설정합니다. 동일한 속도 및 이중화된 설정을 공유하는 집계 그룹을 생성합니다. 활성 수집기의 모든 포트에서 전송 및 수신됩니다. **802.3ad** 호환 네트워크 스위치가 필요합니다.

조정 전송 로드 밸런싱

내결함성 및 로드 밸런싱을 위한 적응형 **Transmit Load Balancing(TLB)** 정책을 설정합니다. 발신 트래픽은 각 포트 인터페이스의 현재 부하에 따라 배포됩니다. 현재 포트에서 들어오는 트래픽을 수신합니다. 수신 포트가 실패하면 다른 포트가 실패한 포트의 **MAC** 주소를 넘겨받습니다. 이 모드는 커널 본딩 모듈에 알려진 로컬 주소에만 적합하므로 가상 시스템과 브리지 뒤에 사용할 수 없습니다.

자동 조정 로드 밸런싱

내결함성 및 로드 밸런싱에 대한 적응형 로드 밸런싱(**ALB**) 정책을 설정합니다. **IPv4** 트래픽에 대한 전송 및 부하 분산 수신을 포함합니다. **ARP** 협상을 통해 부하 분산을 달성합니다. 이 모드는 커널 본딩 모듈에 알려진 로컬 주소에만 적합하므로 가상 시스템과 브리지 뒤에 사용할 수 없습니다.

다음 링크 모니터링 유형은 **Link Monitoring** (링크 모니터링) 드롭다운 목록에서 선택할 수 있습니다. 본딩된 인터페이스에 가장 적합한 채널 본딩 모듈 매개 변수를 테스트하는 것이 좋습니다.

MII(Media Independent Interface)

인터페이스의 캐리어 트렌드 상태가 모니터링됩니다. 이 작업은 **MII** 레지스터를 직접 쿼리하거나 **ethtool** 을 사용하여 장치를 쿼리하여 드라이버를 쿼리하여 수행할 수 있습니다. 다음 세 가지 옵션

을 사용할 수 있습니다.

모니터링 주기

드라이버 또는 **MII** 레지스터를 쿼리하는 시간 간격(밀리초)입니다.

링크 업 지연

보고된 링크를 사용하기 전에 대기하는 시간(밀리초)입니다. 이 지연은 링크가 "**up**" 으로 보고되는 즉시 불필요한 **ARP** 요청이 손실된 경우 사용할 수 있습니다. 이러한 문제는 전환 초기화 중에 발생할 수 있습니다.

링크 다운 지연

이전에 활성 링크를 "**down**" 으로 보고한 경우 다른 링크로 변경하기 전에 대기하는 시간(밀리초)입니다. 연결된 스위치가 백업 모드로 변경되는 데 상대적으로 오랜 시간이 걸리는 경우 이 지연을 사용할 수 있습니다.

ARP

ARP(주소 해결 프로토콜)는 링크 계층 연결이 작동하는 정도를 확인하기 위해 하나 이상의 피어를 프로브하는 데 사용됩니다. 전송 시작 시간과 마지막 수신 시간을 제공하는 장치 드라이버에 따라 다릅니다.

두 가지 옵션을 사용할 수 있습니다.

모니터링 주기

ARP 요청을 보내는 시간 간격(밀리초)입니다.

ARP 대상

ARP 요청을 보낼 쉼표로 구분된 **IP** 주소 목록입니다.

7.9. 추가 리소스

설치된 문서

- **nmcli(1)** 도움말 페이지 - **NetworkManager** 의 명령줄 도구를 설명합니다.

- ***nmcli-examples(5)** 도움말 페이지 - **nmcli** 명령 예제 표시.*
- ***nm-settings(5)** 도움말 페이지 - **NetworkManager** 연결의 설정 및 매개 변수에 대한 설명.*

온라인 문서

***Red Hat Enterprise Linux** 시스템 관리자 가이드*

커널 모듈 기능 사용에 대해 설명합니다.

https://access.redhat.com/site/node/28421/Configuring_VLAN_devices_over_a_bonded_interface

*결합된 인터페이스를 통해 **VLAN** 장치 구성에 대한 **Red Hat** 지식베이스 문서.*

8장. 네트워크 티밍 구성

8.1. 네트워크 티밍 이해

논리적 링크를 더 높은 처리량으로 제공하거나 네트워크 링크를 결합하거나 중복성을 제공하기 위해는 채널 본딩,이더넷 본딩,포트 트렁킹, 채널 티밍,NIC티밍 또는 링크 집계 와 같은 여러 이름으로 알려져 있습니다. 이 개념은 **Linux** 커널에 원래 구현된 바와 같이 널리 본딩 이라고 합니다. 네트워크 티밍 이라는 용어는 이 개념의 새로운 구현을 참조하도록 선택되었습니다. 기존의 본딩 드라이버는 영향을 받지 않으며, 네트워크 티밍은 대안으로 제공되며 **Red Hat Enterprise Linux 7**의 본딩을 대체하지 않습니다.

참고

모드 **4 LACP(Link Aggregation Control Protocol)** 티밍 모드에 대해서는 링크를 집계하도록 스위치를 구성해야 합니다. 자세한 내용은 **https://www.kernel.org/doc/Documentation/networking/bonding.txt**를 참조하십시오.

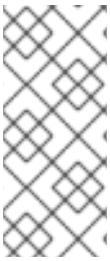
네트워크 티밍 또는 팀은 패킷 흐름의 빠른 처리를 구현하는 소형 커널 드라이버와 사용자 공간에서 다른 모든 작업을 수행하는 다양한 사용자 공간 애플리케이션을 제공하여 다양한 방식으로 개념을 구현하도록 설계되었습니다. 드라이버에는 **Netlink** 통신을 구현하는 **"Team NetlinkAPI**라는 **API(Application Programming Interface)**가" 있습니다. 사용자 공간 애플리케이션은 이 **API**를 사용하여 드라이버와 통신할 수 있습니다. **"lib"** 라고 하는 라이브러리가 **Team Netlink** 통신 및 **RT Netlink** 메시지의 사용자 공간을 래핑하는 데 제공되었습니다. **libteam** 라이브러리를 사용하는 애플리케이션 데몬 **teamd** 도 사용할 수 있습니다. **teamd** 의 인스턴스 한 개가 팀 드라이버의 인스턴스 하나를 제어할 수 있습니다. 데몬은 "러너" 라는 추가 코드를 사용하여 라운드 로빈과 같은 부하 분산 및 활성 백업 논리를 구현합니다. 이러한 방식으로 코드를 분리함으로써 네트워크 티밍 구현은 부하 분산 및 중복 요구 사항에 맞게 쉽게 확장 가능하고 확장 가능한 솔루션을 제공합니다. 예를 들어, **teamd** 를 통해 새 논리를 구현하도록 사용자 지정 러너를 비교적 쉽게 작성할 수 있으며 **teamd** 도 선택 사항이므로 사용자가 **libteam** 을 사용하도록 자체 애플리케이션을 작성할 수 있습니다.

teamdctl 유틸리티는 **D-bus**를 사용하여 **teamd** 의 실행 중인 인스턴스를 제어할 수 있습니다. **teamdctl** 은 **teamd D-Bus API**를 중심으로 **D-Bus** 래퍼를 제공합니다. 기본적으로 **teamd** 는 **Unix** 도메인 소켓을 사용하여 수신 대기하고 통신하지만 **D-Bus**를 모니터링합니다. 이는 **teamd** 가 **D-Bus**가 없거나 아직 로드되지 않은 환경에서 사용할 수 있도록 하기 위한 것입니다. 예를 들어 티밍된 링크를 통해 부팅하는 경우 **D-Bus**가 아직 로드되지 않습니다. 런타임 중에 **teamdctl** 유틸리티를 사용하여 구성, **link-watchers**의 상태, 포트 상태를 확인 및 변경하고, 포트를 추가 및 제거하고, 활성 및 백업 상태 간에 포트를 변경할 수 있습니다.

팀 **Netlink API**는 **Netlink** 메시지를 사용하여 사용자 공간 애플리케이션과 통신합니다. **libteam** 사용자 공간 라이브러리는 **API**와 직접 상호 작용하지 않지만 **libnl** 또는 **teamnl** 을 사용하여 드라이버 **API**와 상호 작용합니다.

요약하기 위해 커널에서 실행되는 팀 드라이버의 인스턴스는 구성하거나 직접 제어하지 않습니다. 모든 구성은 **teamd** 애플리케이션과 같은 사용자 공간 애플리케이션의 지원을 통해 수행됩니다. 그런 다음

애플리케이션은 커널 드라이버 부분을 적절하게 지시합니다.



참고

네트워크 티밍의 컨텍스트에서 **port** 라는 용어는 슬레이브 라고도 합니다. 슬레이브 를 사용하면 직접 **teamd** 를 사용하는 경우 포트가 선호되지만 **NetworkManager** 를 사용하여 팀을 생성하는 인터페이스를 참조할 때 포트가 사용됩니다.

8.2. 기본 컨트롤러 동작 및 포트 인터페이스 이해

NetworkManager 데몬을 사용하여 티밍된 포트 인터페이스를 제어할 때나 결함을 찾을 때 특히 다음에 유의해야 합니다.

1. 컨트롤러 인터페이스를 시작하면 포트 인터페이스가 자동으로 시작되지 않습니다.

2. 포트 인터페이스를 시작하면 항상 컨트롤러 인터페이스가 시작됩니다.

3. 컨트롤러 인터페이스를 중지하면 포트 인터페이스도 중지됩니다.

4. 포트가 없는 컨트롤러는 고정 **IP** 연결을 시작할 수 있습니다.

5. 포트 없는 컨트롤러는 **DHCP** 연결을 시작할 때 포트를 대기합니다.

6. 포트를 기다리는 **DHCP** 연결이 있는 컨트롤러는 캐리어가 있는 포트를 추가하면 완료됩니다.

7. 포트를 기다리는 **DHCP** 연결이 있는 컨트롤러는 캐리어가 없는 포트를 추가하면 계속 대기합니다.

주의

*네트워크 스위치 없이 직접 케이블 연결을 사용하는 것은 팀 구성에는 지원되지 않습니다. 여기에 설명된 장애 조치 메커니즘은 네트워크 스위치가 없으면 예상대로 작동하지 않습니다. 자세한 내용은 **Red Hat** 지식베이스 문서에서 교차 케이블을 사용한 직접 연결로 본딩이 지원되지 않는 이유를 참조하십시오.*

8.3. 네트워크 티밍과 본딩 비교

표 8.1. 연결 및 팀의 기능 비교

기능	본딩	팀
브로드캐스트 Tx 정책	있음	있음
라운드 로빈 Tx 정책	있음	있음
active-backup Tx 정책	있음	있음
LACP(802.3ad) 지원	예 (활성 전용)	있음
해시 기반 Tx 정책	있음	있음
사용자가 해시 함수를 설정할 수 있습니다	없음	있음
TX 로드 밸런싱 지원 (TLB)	있음	있음
LACP 해시 포트 선택	있음	있음
LACP 지원에 대한 로드 밸런싱	없음	있음
Ethtool 링크 모니터링	있음	있음
ARP 링크 모니터링	있음	있음
NS/NA(IPv6) 링크 모니터링	없음	있음
포트 업/다운 지연	있음	있음

기능	본딩	팀
포트 우선 순위 및 고정 ("기본" 옵션 개선)	없음	있음
별도의 포트별 링크 모니터링 설정	없음	있음
다중 링크 모니터링 설정	제한됨	있음
잠금 없는 Tx/Rx 경로	아니요(rwlock)	예(RCU)
VLAN 지원	있음	있음
사용자 공간 런타임 제어	제한됨	full
사용자 공간의 논리	없음	있음
확장성	하드	쉬운
모듈식 설계	없음	있음
성능 오버헤드	낮음	매우 낮음
D-Bus 인터페이스	없음	있음
다중 장치 스택	있음	있음
LLDP를 사용한 제로 구성	없음	(계획)
NetworkManager 지원	있음	있음

8.4. 네트워크 티밍 데몬과 "실행자" 이해

***teamd** 팀 데몬인 **teamd** 는 **libteam** 을 사용하여 팀 드라이버의 인스턴스 하나를 제어합니다. 이 팀 드라이버 인스턴스는 하드웨어 장치 드라이버의 인스턴스를 추가하여 네트워크 링크 "팀을" 구성합니다. 팀 드라이버에서 네트워크 인터페이스를 제공합니다. **team0** 예를 들어 커널의 다른 부분에 대해 다음을 수행합니다. 팀 드라이버 인스턴스에서 만든 인터페이스에는 다음과 같은 이름이 지정됩니다. **team0, team1,** 설명서에 나와 있습니다. 이 방법은 이해하기 쉽고 다른 이름을 사용할 수 있습니다. 티밍의 모든 메서드에 공통된 논리는 **teamd** 에 의해 구현됩니다. 라운드 로빈과 같이 다양한 부하 공유 및 백업 방법에 고유한 해당 함수는 "러너" 라고 하는 별도의 코드 단위로 구현됩니다. "모듈" 및 "모드" 와 같은 단어는 이미 커널과 관련하여 특정 의미를 가지고 있기 때문에 "**runner**" 라는 단어가 이러한 코드 단위를 참조하도록 선택되었습니다. 사용자는 **JSON** 형식 구성 파일의 러너를 지정하고, 인스턴스가 생성될 때 코드는 **teamd** 의 인스턴스로 컴파일됩니다. 러너의 코드가 생성될 때 **teamd** 의 인스턴스로 컴파일되므로 러너가 플러그인이 아닙니다. 필요성이 발생할 경우 **teamd** 용 플러그인으로 코드를 만들 수 있습니다.*

다음 러너는 쓰기 시 사용할 수 있습니다.

- 브로드캐스트(데이터는 모든 포트로 전송됨)
- 라운드 로빈 (데이터는 모든 포트에서 차례로 전송됨)
- **active-backup** (한 개의 포트 또는 링크가 사용되고 다른 포트는 백업으로 유지됨)
- 부하 분산(활성 **Tx** 부하 분산 및 **BPF** 기반 **Tx** 포트 선택기 사용)
- **LACP(802.3ad** 링크 집계 제어 프로토콜 구현)

다음 링크-감사도 사용할 수 있습니다.

- **ethtool (Libteam lib**는 **ethtool** 을 사용하여 링크 상태 변경 사항을 감시). 구성 파일에 다른 **link-watcher**가 지정되지 않은 경우 기본값입니다.
- **arp_ping (Arp_ping** 유틸리티는 **ARP** 패킷을 사용하여 **far-end** 하드웨어 주소의 존재를 모니터링하는 데 사용됩니다.)
- **nsna_ping (IPv6 Neighbor Discovery** 프로토콜의 **Neighbor Advertisements** 및 **Neighbor Solicitation)**을 사용하여 인접 인터페이스의 존재를 모니터링합니다.

특정 **link-watcher**가 특정 러너와 함께 사용되지 않도록 하는 코드에는 제한이 없지만 **lacp** 러너를 사용할 때 **ethtool** 은 권장되는 유일한 링크-감사자입니다.

8.5. 네트워크 티밍 데몬 설치

team d 네트워킹 티밍 데몬은 기본적으로 설치되지 않습니다. **teamd** 를 설치하려면 **root** 로 다음 명령을 실행합니다:

```
~]# yum install teamd
```

8.6. 본딩을 팀으로 변환

bond2team 도구를 사용하여 기존 본딩 구성 파일을 팀 구성 파일로 변환할 수 있습니다. **ifcfg** 형식의 본딩 구성 파일을 **ifcfg** 또는 **JSON** 형식의 팀 구성 파일로 변환할 수 있습니다. 방화벽 규칙, 별칭 인터페이스 및 원래 인터페이스 이름에 연결될 수 있는 항목은 변경 후에 중단될 수 있습니다. 도구는 **ifcfg** 파일만 변경하므로 그 외에는 아무것도 변경하지 않습니다.

명령 형식의 몇 가지 예를 보려면 다음 명령을 실행합니다.

```
~]$ bond2team --examples
```

이름이 **/tmp/bond2team.XXXXXX/**로 시작하는 디렉토리에 새 파일이 생성됩니다. 여기서 **XXXXXX**는 임의 문자열입니다. 새 구성 파일을 만든 후 이전 본딩 파일을 백업 폴더로 이동한 다음 새 파일을 **/etc/sysconfig/network-scripts/** 디렉터리로 이동합니다.

예 8.1. 본딩을 팀으로 변환

현재 **bond0** 구성을 팀 **ifcfg** 로 변환하려면 **root** 로 명령을 실행합니다.

```
~]# /usr/bin/bond2team --master bond0
```

그러면 이름 **bond0** 이 유지됩니다. 새 이름을 사용하여 구성을 저장하려면 다음과 같이 **--rename**을 사용합니다.

```
~]# /usr/bin/bond2team --master bond0 --rename team0
```

ifcfg 파일 대신 **JSON** 형식 파일을 출력하는 데 **--json** 옵션을 추가합니다. **JSON** 형식의 예는 **teamd.conf(5)** 도움말 페이지를 참조하십시오.

예 8.2. 본딩을 팀으로 변환하고 파일 경로를 지정합니다.

현재 **bond0** 구성을 팀 **ifcfg** 로 변환하고 경로를 수동으로 **ifcfg** 파일로 지정하려면 **root** 로 명령을 실행합니다:

```
~]# /usr/bin/bond2team --master bond0 --configdir /path/to/ifcfg-file
```

ifcfg 파일 대신 **JSON** 형식 파일을 출력하는 데 **--json** 옵션을 추가합니다.

예 8.3. Bond2team을 사용하여 팀 구성 만들기

bond2team 툴에 본딩 매개 변수 목록을 제공하여 팀 구성을 만들 수도 있습니다. 예를 들어 다음과 같습니다.

```
~]# /usr/bin/bond2team --bonding_opts "mode=1 miimon=500"
```

다음과 같이 명령행에서 포트를 제공할 수도 있습니다.

```
~]# /usr/bin/bond2team --bonding_opts "mode=1 miimon=500 primary=enp1s0 \
 primary_reselect-0" --port enp1s0 --port enp2s0 --port enp3s0 --port enp4s0
```

자세한 내용은 **bond2team(1)** 도움말 페이지를 참조하십시오. 본딩 매개변수에 대한 설명은 다음을 참조하십시오. **7.7절. "채널 연결 사용"**

8.7. 네트워크 팀의 포트로 사용할 인터페이스 선택

사용 가능한 인터페이스를 보려면 다음 명령을 실행합니다.

```
~]$ ip link show
1: lo:  <LOOPBACK,UP,LOWER_UP > mtu 65536 qdisc noqueue state UNKNOWN mode
DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: em1:  <BROADCAST,MULTICAST,UP,LOWER_UP > mtu 1500 qdisc pfifo_fast state UP
mode DEFAULT qlen 1000
    link/ether 52:54:00:6a:02:8a brd ff:ff:ff:ff:ff:ff
3: em2:  <BROADCAST,MULTICAST,UP,LOWER_UP > mtu 1500 qdisc pfifo_fast state UP
mode DEFAULT qlen 1000
link/ether 52:54:00:9b:6d:2a brd ff:ff:ff:ff:ff:ff
```

사용 가능한 인터페이스에서 네트워크 팀에 추가하는 데 적합한 인터페이스를 확인한 다음 계속 진행합니다. **8.8절. "네트워크 팀 구성 방법 선택"**

8.8. 네트워크 팀 구성 방법 선택

NetworkManager 의 텍스트 사용자 인터페이스 도구인 nmtui 를 사용하여 네트워크 팀을 구성하려면 다음을 진행합니다. 8.9절. “텍스트 사용자 인터페이스 nmtui를 사용하여 네트워크 팀 구성”

명령줄 도구nmcli 를 사용하여 네트워크 팀을 만들려면 8.10.1절. “nmcli를 사용하여 네트워크 티밍 구성” 로 진행합니다.

팀 데몬 teamd를 사용하여 네트워크 팀을 만들려면 8.10.2절. “teamd를 사용하여 네트워크 팀 만들기” 진행합니다.

구성 파일을 사용하여 네트워크 팀을 만들려면 8.10.3절. “ifcfg 파일을 사용하여 네트워크 팀 만들기” 로 이동합니다.

그래픽 사용자 인터페이스를 사용하여 네트워크 팀을 구성하려면 다음을 참조하십시오. 8.14절. “GUI 를 사용하여 네트워크 팀 만들기”

8.9. 텍스트 사용자 인터페이스 NMTUI를 사용하여 네트워크 팀 구성

텍스트 사용자 인터페이스 도구 nmtui 는 터미널 창에서 티밍을 구성하는 데 사용할 수 있습니다. 다음 명령을 실행하여 도구를 시작합니다.

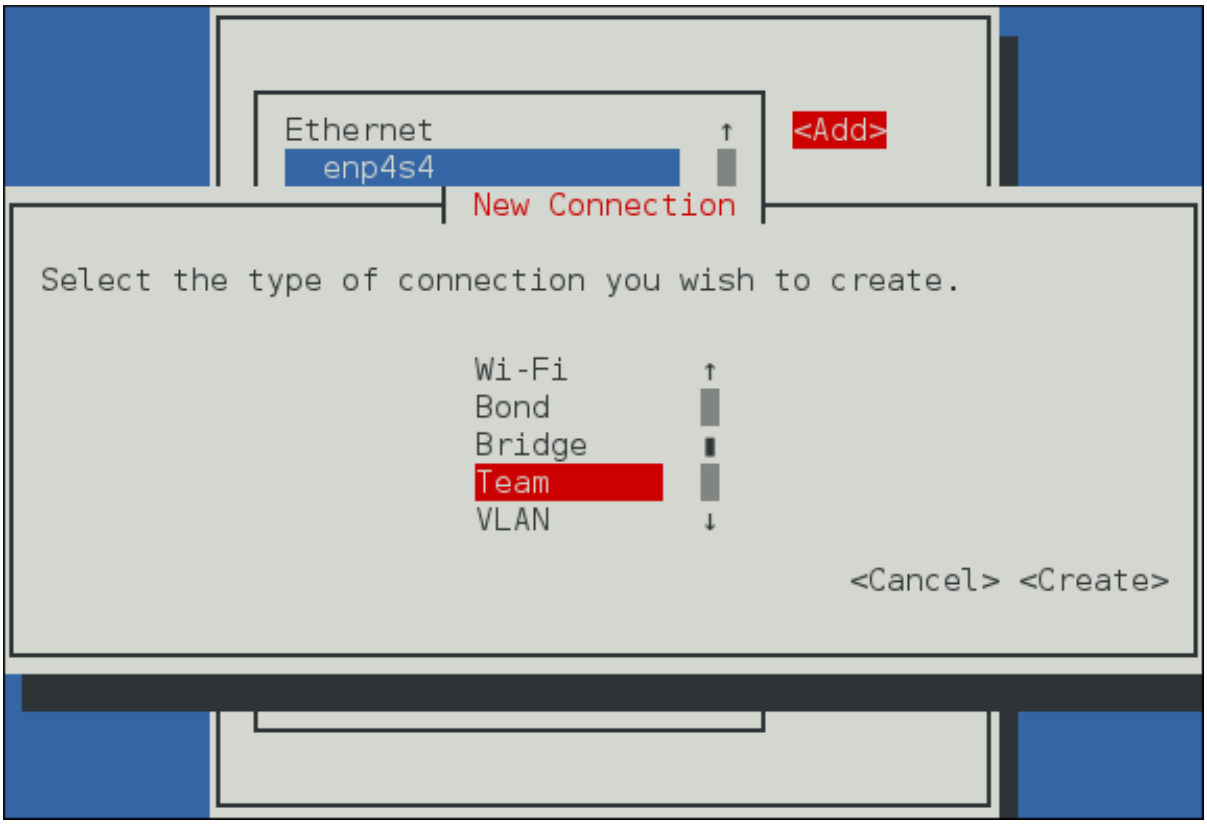
```
~]$ nmtui
```

텍스트 사용자 인터페이스가 나타납니다. 잘못된 명령은 사용 메시지를 인쇄합니다.

탐색하려면 화살표 키를 사용하거나 탭을 눌러 앞으로 이동하고 Shift+Tab 을 눌러 옵션을 다시 이동합니다. Enter 를 눌러 옵션을 선택합니다. Space 표시줄에서 확인란의 상태를 전환합니다.

1. 시작 메뉴에서 Edit a connection (연결 편집)을 선택합니다. Add 를 선택하면 New Connection (새 연결) 화면이 열립니다.

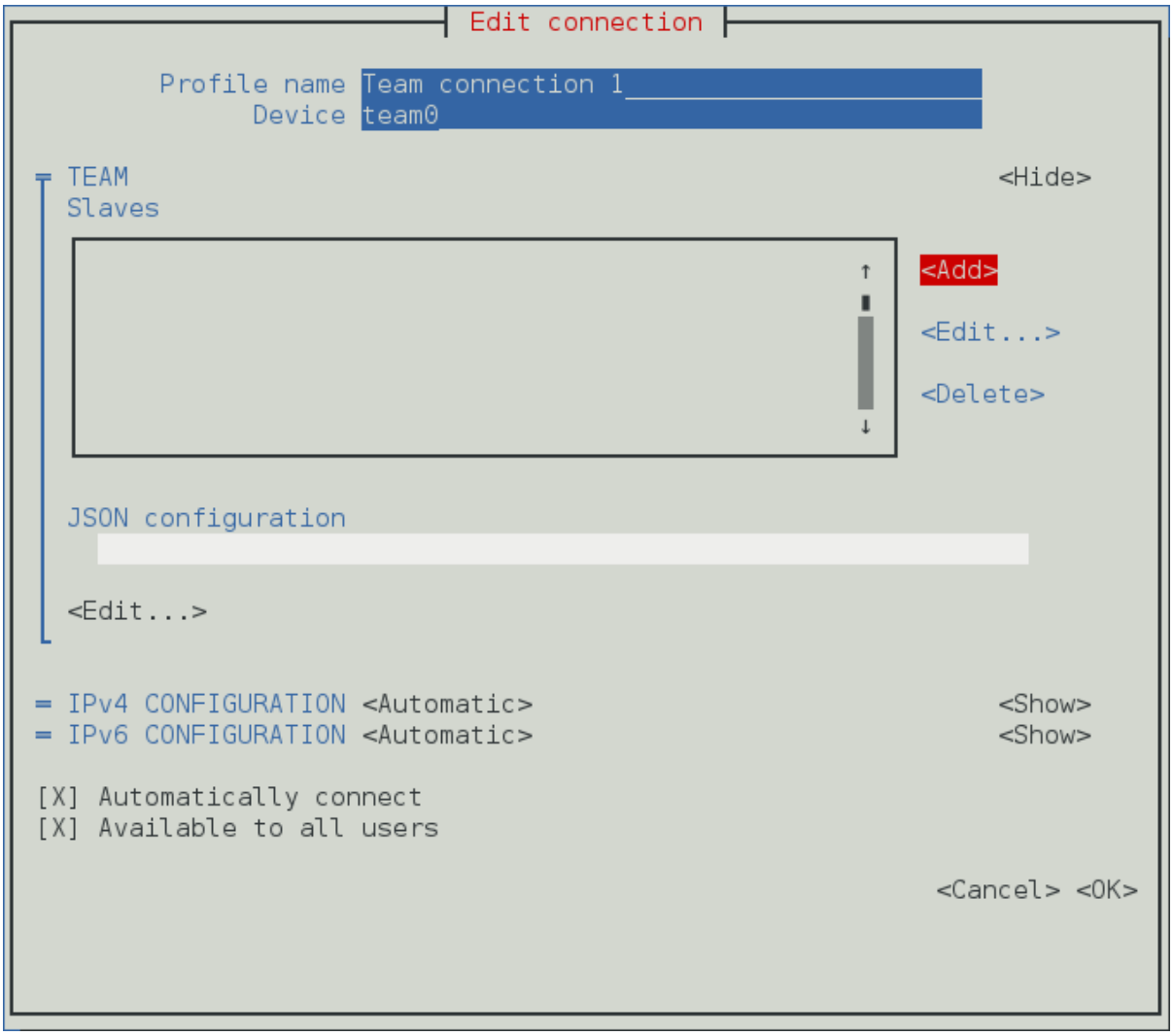
그림 8.1. NetworkManager 텍스트 사용자 인터페이스 Add a Team Connection 메뉴



[D]

2. ***Edit connection (연결 편집)*** *화면이 열리면서* ***Team (팀)****을 선택합니다.*

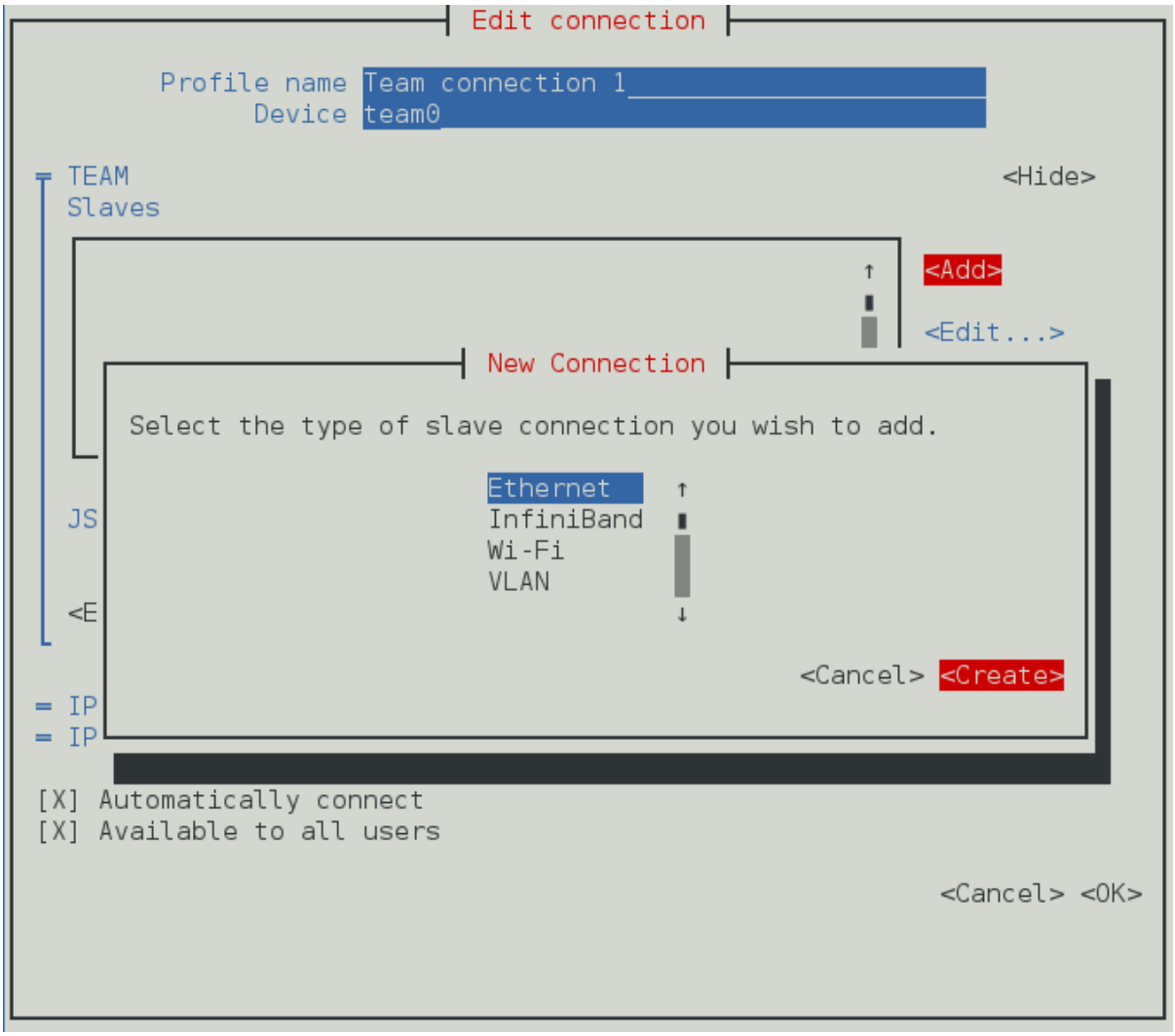
그림 8.2. NetworkManager 텍스트 사용자 인터페이스 팀 연결 구성 메뉴



[D]

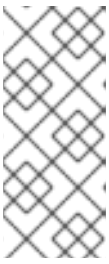
3. *팀에 포트 인터페이스를 추가하려면 **Add**(추가)를 선택합니다. **New Connection** (새 연결) 화면이 열립니다. 연결 유형을 선택한 후 **Create** (만들기) 버튼을 선택하여 팀의 **Edit Connection** (연결 편집) 디스플레이가 표시됩니다.*

그림 8.3. NetworkManager 텍스트 사용자 인터페이스 새 팀 포트 인터페이스 연결 메뉴



[D]

4. *장치 섹션에 필요한 포트의 장치 이름 또는 **MAC** 주소를 입력합니다. 필요한 경우 이더넷 레이블의 오른쪽에 표시를 선택하여 팀의 **MAC** 주소로 복제된 **MAC** 주소를 입력합니다. **OK**(확인) 버튼을 선택합니다.*



참고

***MAC** 주소 없이 장치를 지정하면 장치 섹션이 자동으로 입력되며 **Edit Connection**(연결 편집) 창이 다시 로드되지만 장치를 성공적으로 찾은 경우에만 가능합니다.*

그림 8.4. NetworkManager 텍스트 사용자 인터페이스 구성 팀의 포트 인터페이스 연결 메뉴

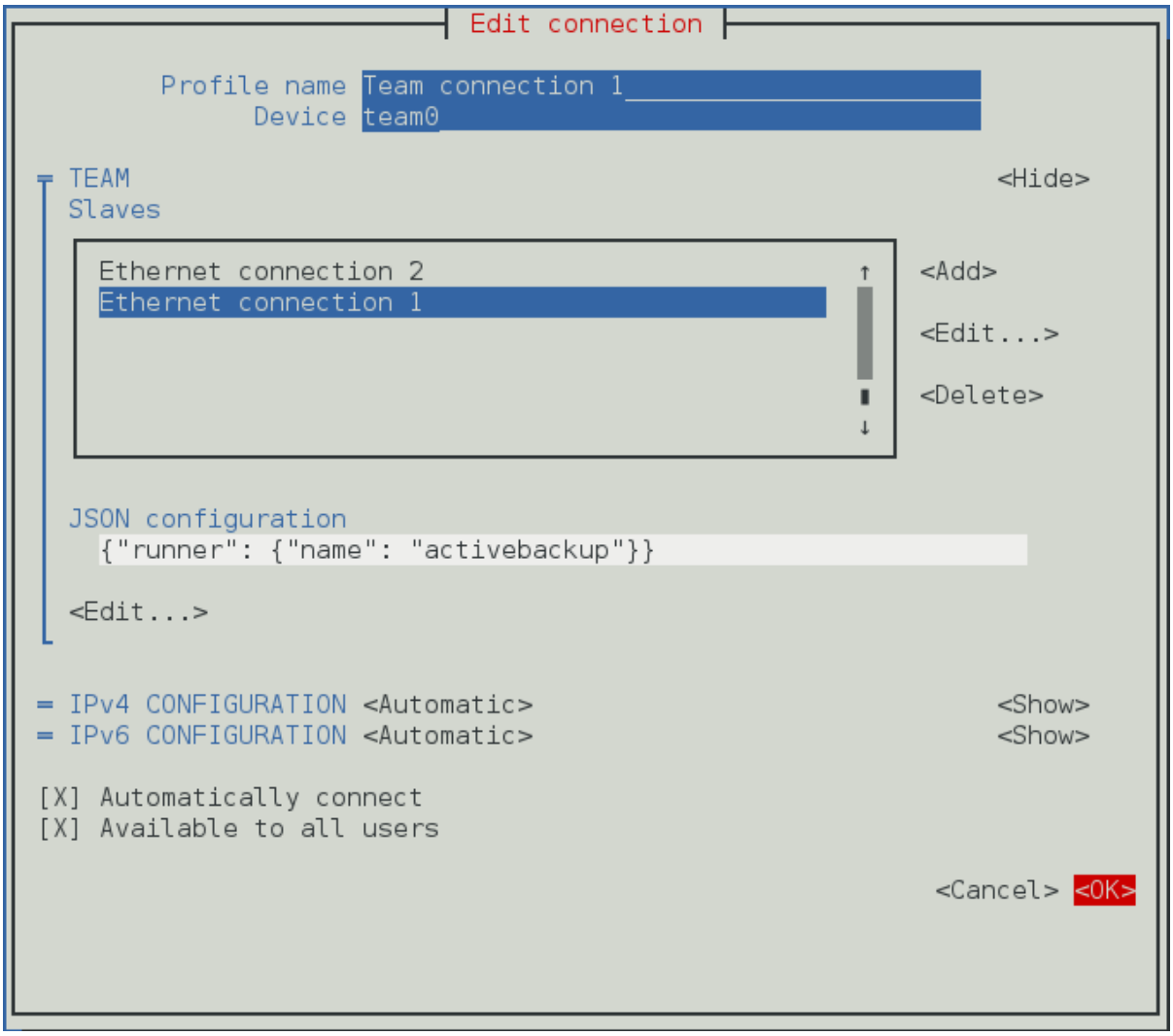
[D]

5. *티밍된 포트의 이름이 **Slaves** 섹션에 표시됩니다. 위의 단계를 반복하여 추가 포트 연결을 추가합니다.*

6. *사용자 지정 포트 설정을 적용할 경우 **JSON** 구성 섹션에서 **Edit(편집)** 버튼을 선택합니다. 그러면 변경 사항이 적용될 수 있는 **vim** 콘솔이 실행됩니다. **vim** 의 변경 사항 쓰기가 끝났으면 **JSON** 구성 아래의 표시된 **JSON** 문자열이 의도한 내용과 일치하는지 확인합니다.*

7. ***OK(확인)** 버튼을 선택하기 전에 설정을 검토하고 확인합니다.*

그림 8.5. NetworkManager 텍스트 사용자 인터페이스 팀 연결 구성 메뉴



[D]

JSON 문자열의 예는 8.13절. "teamd Runners 구성" 을 참조하십시오. 예제 문자열의 관련 섹션만 nmtui 를 사용하여 팀 또는 포트 구성에 사용해야 합니다. 장치를 JSON 문자열의 "일부로" 지정하지 마십시오. 예를 들어, "장치" 뒤의 JSON 문자열만 팀 JSON 구성 필드에서 "사용해야 합니다". 포트와 관련된 모든 JSON 문자열은 포트 구성 필드에만 추가해야 합니다.

nmtui 설치에 대한 자세한 내용은 3.2절. "nmtui로 IP 네트워킹 구성" 을 참조하십시오.

8.10. 명령줄을 사용하여 네트워크 팀 구성

8.10.1. nmcli를 사용하여 네트워크 티밍 구성

시스템에서 사용 가능한 연결을 보려면 다음을 수행합니다.

```
~]$ nmcli connection show
NAME  UUID                                  TYPE          DEVICE
enp2s0  0e8185a1-f0fd-4802-99fb-bedbb31c689b  802-3-ethernet  --
```

```
enp1s0  dfe1f57b-419d-4d1c-aaf5-245deab82487  802-3-ethernet  --
```

시스템에서 사용 가능한 장치를 보려면 다음을 수행합니다.

```
~]$ nmcli device status
DEVICE     TYPE     STATE                      CONNECTION
virbr0     bridge   connected                  virbr0
ens3       ethernet connected                  ens3
```

이름 **ServerA** 를 사용하여 새 팀 인터페이스를 만들려면 다음을 수행합니다.

```
~]$ nmcli connection add type team ifname ServerA
Connection 'team-ServerA' (b954c62f-5fdd-4339-97b0-40efac734c50) successfully added.
```

NetworkManager 는 내부 매개 변수 **connection.autoconnect** 를 **yes** 로 설정하고 **IP** 주소에 **ipv4.method** 를 **auto** 로 설정하지 않으므로 설정합니다. 또한 **NetworkManager** 는 해당 **ONBOOT**가 **yes** 로 설정되고 **BOOTPROTO**가 **dhcp** 로 설정되는 **/etc/sysconfig/network-scripts/ifcfg-team-ServerA** 에 구성 파일을 작성합니다.

나중에 인터페이스가 표시될 때까지 **NetworkManager** 는 **ifcfg** 파일에 대한 수동 변경 사항을 인식하지 못합니다. 설정 파일 사용에 대한 자세한 내용은 2.7절. “sysconfig 파일로 NetworkManager 사용” 을 참조하십시오.

할당된 다른 값을 보려면 다음을 수행합니다.

```
~]$ nmcli con show team-ServerA
connection.id:                          team-ServerA
connection.uuid:                        b954c62f-5fdd-4339-97b0-40efac734c50
connection.interface-name:              ServerA
connection.type:                        team
connection.autoconnect:                 yes
...
ipv4.method:                            auto
[output truncated]
```

JSON 구성 파일이 지정되지 않았으므로 기본값이 적용됩니다. 팀 **JSON** 매개 변수 및 기본값에 대한 자세한 내용은 **teamd.conf(5)** 도움말 페이지를 참조하십시오. 이름은 인터페이스 이름에서 형식 앞에 따라 파생되었습니다. 또는 다음과 같이 **con-name** 옵션을 사용하여 이름을 지정합니다.

```
~]$ nmcli connection add type team con-name Team0 ifname ServerB
Connection 'Team0' (5f7160a1-09f6-4204-8ff0-6d96a91218a7) successfully added.
```

방금 구성한 팀 인터페이스를 보려면 다음과 같이 명령을 입력합니다.

```
~]$ nmcli con show
NAME            UUID                                  TYPE           DEVICE
team-ServerA    b954c62f-5fdd-4339-97b0-40efac734c50  team           ServerA
enp2s0          0e8185a1-f0fd-4802-99fb-bedbb31c689b  802-3-ethernet --
enp1s0          dfe1f57b-419d-4d1c-aaf5-245deab82487  802-3-ethernet --
Team0           5f7160a1-09f6-4204-8ff0-6d96a91218a7  team           ServerB
```

팀에 할당된 이름을 변경하려면 다음 형식으로 명령을 입력합니다.

```
nmcli con mod old-team-name connection.id new-team-name
```

이미 존재하는 팀의 팀 구성 파일을 로드하려면 다음을 수행합니다.

```
nmcli connection modify team-name team.config JSON-config
```

팀 구성을 **JSON** 문자열로 지정하거나 구성이 포함된 파일 이름을 제공할 수 있습니다. 파일 이름에 경로가 포함될 수 있습니다. 두 경우 모두 **team.config** 속성에 저장된 내용은 **JSON** 문자열입니다. **JSON** 문자열의 경우 문자열 주변의 작은따옴표를 사용하고 전체 문자열을 명령줄에 붙여넣습니다.

team.config 속성을 검토하려면 다음을 수행합니다.

```
nmcli con show team-name | grep team.config
```

team.config 속성이 설정되면 다른 모든 팀 속성이 적절하게 업데이트됩니다.

또한 해당 **JSON** 문자열을 직접 수정하지 않고 특정 팀 옵션을 노출하고 설정하는 보다 유연한 방법을 사용할 수 있습니다. 다른 사용 가능한 팀 속성을 사용하여 관련 팀 옵션을 각각 필요한 값으로 설정하여

*이 작업을 수행할 수 있습니다. 결과적으로 **team.config** 속성이 새 값과 일치하도록 업데이트됩니다.*

*예를 들어 하나 이상의 **link -watchers**를 지정할 수 있는 **team.link -watchers** 속성을 설정하려면 다음 형식으로 명령을 입력합니다.*

```
nmcli connection modify team-name team.link-watchers "name=ethtool delay-up=5,
name=nsna_ping target-host=target.host"
```

*필요한 **link-watchers** 는 쉼표로 구분되고 동일한 링크-감사에 속하는 특성은 공백으로 구분됩니다.*

*team.runner 및 **team.link-watchers** 속성을 설정하려면 다음 형식으로 명령을 입력합니다.*

```
nmcli connection modify team-name team.runner activebackup team.link-watchers
"name=ethtool delay-up=5, name=nsna_ping target-host=target.host"
```

*이는 **team.config** 속성을 해당 **JSON** 문자열로 설정하는 것과 동일합니다.*

```
nmcli connection modify team-name team.config '{"runner": {"name": "activebackup"},
"link_watch": [{"name": "ethtool", "delay_up": 5},{"name": "nsna_ping", "target_host ":
"target.host"}]'
```

*이름이 **Team 0-port 1** 인 **Team0** 에 인터페이스를 추가하려면 다음과 같이 명령을 실행합니다.*

```
~]$ nmcli con add type ethernet con-name Team0-port1 ifname enp1s0 slave-type team master
Team0
Connection 'Team0-port1' (ccd87704-c866-459e-8fe7-01b06cf1cffc) successfully added.
```

*마찬가지로 이름이 **Team0-port 2**인 다른 인터페이스 **enp2 s0** 을 추가하려면 다음과 같이 명령을 실행합니다.*

```
~]$ nmcli con add type ethernet con-name Team0-port2 ifname enp2s0 slave-type team master
Team0
Connection 'Team0-port2' (a89ccff8-8202-411e-8ca6-2953b7db52dd) successfully added.
```

nmcli 는 이더넷 포트만 지원합니다.

팀을 열려면 다음과 같이 포트를 먼저 가져와야 합니다.

```
~]$ nmcli connection up Team0-port1
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/2)
```

```
~]$ nmcli connection up Team0-port2
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/3)
```

다음과 같이 포트를 활성화하여 팀 인터페이스가 작동되었는지 확인할 수 있습니다.

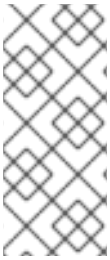
```
~]$ ip link
3:  Team0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
mode DEFAULT
    link/ether 52:54:00:76:6f:f0 brd ff:ff:ff:ff:ff:f
```

또는 명령을 실행하여 다음과 같이 팀을 엽니다.

```
~]$ nmcli connection up Team0
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/4)
```

nmcli에 대한 소개는 **3.3절. “nmcli로 IP 네트워킹 구성”** 를 참조하십시오.

8.10.2. teamd를 사용하여 네트워크 팀 만들기



참고

teamd를 사용하여 만든 구성은 영구적이 아니므로 **8.10.1절. “nmcli를 사용하여 네트워크 티밍 구성”** 또는 **8.10.3절. “ifcfg 파일을 사용하여 네트워크 팀 만들기”** 에 정의된 단계를 사용하여 팀을 만들어야 할 수 있습니다.

네트워크 팀을 생성하려면 포트 또는 링크 팀에 대한 인터페이스 역할을 하는 가상 인터페이스에 **JSON** 형식 구성 파일이 필요합니다. 빠른 방법은 예제 구성 파일을 복사한 다음 **root** 권한으로 실행되는

편집기를 사용하여 편집하는 것입니다. 사용 가능한 예제 구성을 나열하려면 다음 명령을 입력합니다.

```
~]$ ls /usr/share/doc/teamd-*/example_configs/
activebackup_arp_ping_1.conf  activebackup_multi_lw_1.conf   loadbalance_2.conf
activebackup_arp_ping_2.conf  activebackup_nsna_ping_1.conf  loadbalance_3.conf
activebackup_ethtool_1.conf   broadcast.conf                 random.conf
activebackup_ethtool_2.conf   lacp_1.conf                    roundrobin_2.conf
activebackup_ethtool_3.conf   loadbalance_1.conf             roundrobin.conf
```

activebackup_ethtool_1.conf와 같은 포함된 파일 중 하나를 보려면 다음 명령을 입력합니다.

```
~]$ cat /usr/share/doc/teamd-*/example_configs/activebackup_ethtool_1.conf
{
 "device": "team0",
 "runner": {"name": "activebackup"},
 "link_watch": {"name": "ethtool"},
 "ports": {
  "enp1s0": {
   "prio": -10,
   "sticky": true
  },
  "enp2s0": {
   "prio": 100
  }
 }
}
```

teamd 구성 파일을 저장할 작업 구성 디렉터리를 만듭니다. 예를 들어 일반 사용자로 다음 형식을 사용하여 명령을 입력합니다.

```
~]$ mkdir ~/teamd_working_configs
```

선택한 파일을 작업 디렉터리에 복사하고 필요에 따라 편집합니다. 예를 들어 다음 형식의 명령을 사용할 수 있습니다.

```
~]$ cp /usr/share/doc/teamd-*/example_configs/activebackup_ethtool_1.conf \
~/teamd_working_configs/activebackup_ethtool_1.conf
```

환경에 맞게 파일을 편집하려면 네트워크 팀의 포트로 사용할 인터페이스를 변경하려면 다음과 같이 편집할 파일을 엽니다.

```
~]$ vi ~/teamd_working_configs/activebackup_ethtool_1.conf
```

필요한 사항을 모두 변경하고 파일을 저장합니다. **vi** 편집기 사용 또는 선호하는 편집기를 사용하는 방법에 대한 도움말은 **vi (1)** 도움말 페이지를 참조하십시오.

팀 내 포트로 사용할 인터페이스는 반드시 활성 상태가 아니어야 합니다. 즉, 팀 장치에 추가할 때 반드시 “작동” 중단되어야 합니다. 해당 상태를 확인하려면 다음 명령을 실행합니다.

```
~]$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: em1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
DEFAULT qlen 1000
    link/ether 52:54:00:d5:f7:d4 brd ff:ff:ff:ff:ff:ff
3: em2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
DEFAULT qlen 1000
  link/ether 52:54:00:d8:04:70 brd ff:ff:ff:ff:ff:ff
```

*이 예에서 사용할 인터페이스는 모두 "**UP**" 입니다.*

*인터페이스를 종료하려면 다음 형식으로 **root** 로 명령을 실행합니다.*

```
~]# ip link set down em1
```

필요에 따라 각 인터페이스에 대해 반복합니다.

*구성 파일을 기반으로 팀 인터페이스를 생성하려면 **root** 사용자로 작업 구성 디렉터리(이 예에서는**teamd_working_configs**)로 변경합니다.*

```
~]# cd /home/userteamd_working_configs
```

그런 다음 다음 형식으로 명령을 실행합니다.

```
~]# teamd -g -f activebackup_ethtool_1.conf -d
Using team device "team0".
Using PID file "/var/run/teamd/team0.pid"
Using config file "/home/user/teamd_working_configs/activebackup_ethtool_1.conf"
```

*g 옵션은 디버그 메시지의 경우 **-f** 옵션은 로드할 구성 파일을 지정하는 것이며, **-d** 옵션은 시작 후 프로세스가 데몬으로 실행되도록 하는 것입니다. 다른 옵션은 **teamd(8)** 도움말 페이지를 참조하십시오.*

*팀 상태를 확인하려면 **root** 로 다음 명령을 실행합니다 :*

```
~]# teamdctl team0 state
setup:
  runner: activebackup
ports:
  em1
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
```

```
      link: up
  em2
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
runner:
  active port: em1
```

네트워크 팀 인터페이스에 주소를 적용하려면 다음을 수행합니다. **team0**에서 다음 형식으로 **root** 로 명령을 실행합니다.

```
~]# ip addr add 192.168.23.2/24 dev team0
```

팀 인터페이스의 **IP** 주소를 확인하려면 다음과 같이 명령을 실행합니다.

```
~]$ ip addr show team0
4: team0:  <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    link/ether 16:38:57:60:20:6f brd ff:ff:ff:ff:ff:ff
    inet 192.168.23.2/24 scope global team0
       valid_lft forever preferred_lft forever
    inet6 2620:52:0:221d:1438:57ff:fe60:206f/64 scope global dynamic
       valid_lft 2591880sec preferred_lft 604680sec
    inet6 fe80::1438:57ff:fe60:206f/64 scope link
       valid_lft forever preferred_lft forever
```

팀 인터페이스를 활성화하거나 시작하려면 다음 형식으로 **root** "로" 명령을 실행합니다.

```
~]# ip link set dev team0 up
```

팀 인터페이스를 일시적으로 비활성화하거나 차단하려면 다음 형식으로 **root** 로 "명령을" 실행합니다.

```
~]# ip link set dev team0 down
```

팀 데몬 인스턴스를 종료하거나 종료하려면 **root** 사용자로 다음 형식으로 명령을 실행합니다.

```
~]# teamd -t team0 -k
```

k 옵션은 장치와 연결된 데몬의 인스턴스를 지정하는 것입니다. **team0** 가 종료됩니다. 다른 옵션은 **teamd(8)** 도움말 페이지를 참조하십시오.

teamd 에 대한 명령줄 옵션에 대한 도움말을 보려면 다음 명령을 실행합니다.

```
~]$ teamd -h
```

또한 **teamd(8)** 도움말 페이지를 참조하십시오.

8.10.3. ifcfg 파일을 사용하여 네트워크 팀 만들기

ifcfg 파일을 사용하여 네트워킹 팀을 만들려면 다음과 같이 **/etc/sysconfig/network-scripts/** 디렉터리에 파일을 만듭니다.

```
DEVICE=team0
DEVICETYPE=Team
ONBOOT=yes
BOOTPROTO=none
IPADDR=192.168.11.1
PREFIX=24
TEAM_CONFIG='{"runner": {"name": "activebackup"}, "link_watch": {"name": "ethtool"}}'
```

이렇게 하면 팀에 인터페이스가 생성됩니다. 즉, 마스터입니다.

멤버가 될 포트를 만들려면 **team0**다음과 같이 **/etc/sysconfig/network-scripts/** 디렉토리에 하나 이상의 파일을 만듭니다.

```
DEVICE=enp1s0
HWADDR=D4:85:64:01:46:9E
DEVICETYPE=TeamPort
ONBOOT=yes
TEAM_MASTER=team0
TEAM_PORT_CONFIG='{"prio": 100}'
```

필요에 따라 위와 유사한 포트 인터페이스를 추가하여 추가 중인 포트(네트워크 장치)와 일치하도록 **DEVICE** 및 **HWADDR** 필드를 변경합니다. 포트 우선 순위를 **prio** 에 지정하지 않으면 기본값은 **0** 입니다.

범위에서 **-32,767**에서 **+32,767** 까지 음수 및 양의 값을 허용합니다.

HWADDR 지시문을 사용하여 하드웨어 또는 **MAC** 주소를 지정하면 장치 명명 절차에 영향을 미칩니다. 자세한 내용은 *11장. 일관된 네트워크 장치 이름 지정* 에서 참조하십시오.

네트워크 팀을 열려면 **root** 로 다음 명령을 실행합니다:

```
~]# ifup team0
```

네트워크 팀을 보려면 다음 명령을 실행합니다.

```
~]$ ip link show
```

8.10.4. iputils를 사용하여 네트워크 팀에 포트 추가

포트를 추가하려면 **em1** 네트워크 팀 **team0ip** 유틸리티를 사용하여 **root** 로 다음 명령을 실행합니다.

```
~]# ip link set dev em1 down
~]# ip link set dev em1 master team0
```

필요한 경우 다른 포트를 추가합니다. 팀 드라이버에서 포트를 자동으로 작동시킵니다.

8.10.5. teamnl을 사용하여 팀의 포트 나열

team nl 유틸리티를 사용하여 네트워크 팀 의 포트를 보거나 나열하려면 **root** 로 다음 명령을 실행합니다:

```
~]# teamnl team0 ports
em2: up 100 fullduplex
em1: up 100 fullduplex
```

8.10.6. teamnl을 사용하여 팀 옵션 구성

현재 사용 가능한 모든 옵션을 보거나 나열하려면 **teamnl** 유틸리티를 사용하여 **root** 로 다음 명령을 실행합니다:

```
~]# teamnl team0 options
```

활성 백업 모드를 사용하도록 팀을 구성하려면 **root** 로 다음 명령을 실행합니다.

```
~]# teamnl team0 setoption mode activebackup
```

8.10.7. iputils를 사용하여 네트워크 팀에 주소 추가

팀에 주소를 추가하려면 다음을 수행합니다. **team0ip** 유틸리티를 사용하여 **root** 로 다음 명령을 실행합니다.

```
~]# ip addr add 192.168.252.2/24 dev team0
```

8.10.8. iputils를 사용하여 네트워크 팀에 대한 인터페이스를 엽니다.

네트워크 팀에 대한 인터페이스를 활성화하거나 "열려" 는 경우 **team0ip** 유틸리티를 사용하여 **root** 로 다음 명령을 실행합니다.

```
~]# ip link set team0 up
```

8.10.9. teamnl을 사용하여 팀의 활성 포트 옵션 보기

team nl 유틸리티를 사용하여 네트워크 팀의 **activeport** 옵션을 보거나 나열하려면 **root** 로 다음 명령을 실행합니다 :

```
~]# teamnl team0 getoption activeport
0
```

8.10.10. teamnl을 사용하여 팀의 활성 포트 옵션 설정

team nl 유틸리티를 사용하여 네트워크 팀에서 **activeport** 옵션을 설정하려면 **root** 로 다음 명령을 실행합니다:

```
~]# teamnl team0 setoption activeport 5
```

팀 포트 옵션의 변경 사항을 확인하려면 **root** 로 다음 명령을 실행합니다.

```
~]# teamnl team0 getoption activeport
5
```

8.11. TEAMDCTL로 TEAMD 제어

통계 또는 구성 정보를 위해 **teamd** 의 실행 중인 인스턴스를 쿼리하거나 변경하려면 제어 도구 **teamdctl** 이 사용됩니다.

팀의 현재 팀 상태 보기 **team0, root** 로 다음 명령을 입력합니다:

```
~]# teamdctl team0 state view
```

자세한 출력을 보려면 다음을 수행하십시오.

```
~]# teamdctl team0 state view -v
```

JSON 형식의 전체 상태 덤프 (시스템 처리에 유용함)의 경우 **team0**다음 명령을 사용합니다.

```
~]# teamdctl team0 state dump
```

JSON 형식의 구성 덤프의 경우 **team0**다음 명령을 사용합니다.

```
~]# teamdctl team0 config dump
```

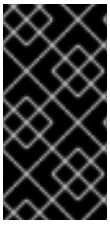
포트 구성을 보려면 다음을 수행합니다. **em1**이것은 팀의 일원입니다. **team0**다음 명령을 입력합니다.

```
~]# teamdctl team0 port config dump em1
```

8.11.1. 네트워크 팀에 포트 추가

포트를 추가하려면 **em1** 네트워크 팀 **team0root** 로 다음 명령을 실행합니다.

```
~]# teamdctl team0 port add em1
```



중요

teamdctl 을 사용하여 포트를 추가하는 경우 포트를 **down** 으로 설정해야 합니다. 그렇지 않으면 **teamdctl team0** 포트 추가 **em1** 명령이 실패합니다.

8.11.2. 네트워크 팀에서 포트 제거

인터페이스를 제거하려면 **em1** 네트워크 팀 **team0root** 로 다음 명령을 실행합니다.

```
~]# teamdctl team0 port remove em1
```

8.11.3. 네트워크 팀의 포트에 구성 적용

포트에 **JSON** 형식 구성을 적용하려면 다음을 수행합니다. **em1** 네트워크 팀 **team0**에서 다음 형식으로 **root** 로 명령을 실행합니다.

```
~]# teamdctl team0 port config update em1 JSON-config-string
```

여기서 **JSON-config-string** 은 **JSON** 형식의 텍스트 문자열로 구성입니다. 이렇게 하면 제공된 **JSON** 형식 문자열을 사용하여 포트 구성이 업데이트됩니다. 포트를 구성하는 유효한 **JSON** 문자열의 예는 다음과 같습니다.

```
{
  "prio": -10,
  "sticky": true
}
```

JSON 구성 문자열에서 작은따옴표를 사용하고 줄 바꿈을 생략합니다.

이전 구성은 덮어쓰기되며 생략된 옵션은 기본값으로 재설정됩니다. 팀 데몬 제어 도구 명령 예제는 teamdctl(8) 도움말 페이지를 참조하십시오.

8.11.4. 네트워크 팀의 포트 구성 보기

포트 구성을 복사하려면 다음을 수행합니다. em1 네트워크 팀 team0root 로 다음 명령을 실행합니다.

```
~]# teamdctl team0 port config dump em1
```

이렇게 하면 포트의 JSON 형식 구성이 표준 출력에 덤프됩니다.

8.12. 중복을 위한 네트워크 구성 티밍 확인

네트워크 중복은 장치가 백업 목적으로 사용되어 특정 시스템의 장애를 방지하거나 복구하는 프로세스입니다. 다음 절차에서는 중복성으로 티밍을 위해 네트워크 구성을 확인하는 방법을 설명합니다.

절차

1. 팀 인터페이스에서 대상 IP를 ping합니다. 예를 들어 다음과 같습니다.

```
~]# ping -I team0 DSTADDR
```

2. 활성 모드에 있는 인터페이스를 확인합니다.

```
~]#  teamdctl team0 state
setup:
  runner: activebackup
ports:
  enp1s0
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
  enp2s0
```

```
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
runner:
  active port: enp1s0
```

enp1s0 은 활성 인터페이스입니다.

3. 활성 포트 인터페이스를 아래의 아래로 설정합니다.

```
~]# ip link set enp1s0 down
```

4. 백업 인터페이스가 작동 중인지 확인합니다.

```
~]#  teamdctl team0 state
setup:
 runner: activebackup
ports:
 enp1s0
   link watches:
     link summary: down
     instance[link_watch_0]:
       name: ethtool
       link: down
       down count: 1
 enp2s0
   link watches:
     link summary: up
     instance[link_watch_0]:
       name: ethtool
       link: up
       down count: 0
runner:
 active port: enp2s0
```

enp2s0 은 이제 활성 인터페이스입니다.

5. 팀 인터페이스에서 대상 **IP**를 계속 **ping**할 수 있는지 확인합니다.

```
~]# ping -I team0 DSTADDR
```

8.13. TEAMD RUNNERS 구성

러너는 데몬 인스턴스가 생성될 때 팀 데몬으로 컴파일되는 코드 단위입니다. **teamd** 러너 소개는 8.4 절. “네트워크 티밍 데몬과 "실행자" 이해” 를 참조하십시오.

8.13.1. 브로드캐스트 Runner 구성

브로드캐스트 러너를 **root** 로 구성하려면 팀 **JSON** 형식 구성 파일에 다음을 추가합니다.

```
{
 "device": "team0",
 "runner": {"name": "broadcast"},
 "ports": {"em1": {}, "em2": {}}
}
```

자세한 내용은 **teamd.conf(5)** 매뉴얼 페이지를 참조하십시오.

8.13.2. 임의의 Runner 구성

임의의 러너는 라운드 로빈 러너와 유사하게 작동합니다.

임의의 러너를 **root** 로 구성하려면 팀 **JSON** 형식 구성 파일에 다음을 추가합니다.

```
{
 "device": "team0",
 "runner": {"name": "random"},
 "ports": {"em1": {}, "em2": {}}
}
```

자세한 내용은 **teamd.conf(5)** 매뉴얼 페이지를 참조하십시오.

8.13.3. 라운드 로빈 Runner 구성

root 로 라운드 로빈 러너를 구성하려면 팀 JSON 형식 구성 파일에 다음을 추가합니다.

```
{
 "device": "team0",
 "runner": {"name": "roundrobin"},
 "ports": {"em1": {}, "em2": {}}
}
```

라운드 로빈에 대한 매우 기본적인 구성입니다.

자세한 내용은 teamd.conf(5) 매뉴얼 페이지를 참조하십시오.

8.13.4. activebackup Runner 구성

활성 백업 러너는 모든 link-watchers를 사용하여 팀의 링크 상태를 확인할 수 있습니다. 다음 예제 중 하나를 팀 JSON 형식 구성 파일에 추가할 수 있습니다.

```
{
  "device": "team0",
  "runner": {
    "name": "activebackup"
  },
  "link_watch": {
    "name": "ethtool"
  },
  "ports": {
    "em1": {
      "prio": -10,
      "sticky": true
    },
    "em2": {
      "prio": 100
    }
  }
}
```

이 예제 구성에서는 ethtool 이 있는 active-backup 러너를 링크 감시자로 사용합니다. 포트 em2 우선 순위가 높습니다. sticky 플래그를 사용하면 다음을 보장할 수 있습니다. em1 가 활성 상태가 되고, 링크가 유지되는 한 활성 상태로 유지됩니다.

```
{
  "device": "team0",
  "runner": {
    "name": "activebackup"
  },
  "link_watch": {
    "name": "ethtool"
  },
  "ports": {
    "em1": {
      "prio": -10,
      "sticky": true,
      "queue_id": 4
    },
    "em2": {
      "prio": 100
    }
  }
}
```

이 예제 구성은 큐 ID 4 를 추가합니다. 링크 감시자로 ethtool 이 있는 active-backup 러너를 사용합니다. 포트 em2 우선 순위가 높습니다. 그러나 Sticky 플래그를 사용하면 이를 보장할 수 있습니다. em1 링크가 활성 상태로 유지되는 동안 계속 활성 상태가 됩니다.

ethtool 을 링크 감시자로 사용하고 지연을 적용하여 activebackup 러너를 구성하려면 root 로 편집기를 사용하여 다음을 팀 JSON 형식 구성 파일에 추가합니다.

```
{
  "device": "team0",
  "runner": {
    "name": "activebackup"
  },
  "link_watch": {
    "name": "ethtool",
    "delay_up": 2500,
    "delay_down": 1000
  },
  "ports": {
    "em1": {
      "prio": -10,
      "sticky": true
    },
    "em2": {
      "prio": 100
    }
  }
}
```

이 예제 구성에서는 ethtool 이 있는 active-backup 러너를 링크 감시자로 사용합니다. 포트 em2 우

선 순위가 높습니다. 그러나 **Sticky** 플래그를 사용하면 이를 보장할 수 있습니다. **em1** 활성화된 상태로 유지되고 링크가 계속 활성 상태로 유지됩니다. 링크 변경 사항은 러너에 즉시 전파되지 않지만 지연이 적용됩니다.

자세한 내용은 **teamd.conf(5)** 매뉴얼 페이지를 참조하십시오.

8.13.5. loadbalance Runner 구성

이 러너는 활성 및 수동의 두 가지 유형의 부하 분산에 사용할 수 있습니다. 액티브 모드에서는 최근 트래픽의 통계를 사용하여 최대한 균등하게 트래픽을 균등하게 공유하여 트래픽의 지속적인 재조정을 수행합니다. 패시브 모드에서 트래픽 스트림은 사용 가능한 링크 간에 임의로 분산됩니다. 이로 인해 처리 오버헤드가 줄어드는 이점이 있습니다. 볼륨 트래픽이 많은 애플리케이션에서는 일반적으로 트래픽이 사용 가능한 링크 간에 무작위로 분산되는 여러 스트림으로 구성되므로 선호됩니다. 이러한 방식으로 **teamd** 의 개입 없이 부하 공유를 수행합니다.

root 로 편집기를 사용하여 수동 전송(**Tx**) 로드 밸런싱을 위해 **loadbalance** 러너를 구성하려면 팀 **JSON** 형식 구성 파일에 다음을 추가합니다.

```
{
 "device": "team0",
 "runner": {
  "name": "loadbalance",
  "tx_hash": ["eth", "ipv4", "ipv6"]
 },
 "ports": {"em1": {}, "em2": {}}
}
```

해시 기반 수동 전송(**Tx**) 로드 밸런싱 구성.

root 로 편집기를 사용하여 **Tx**(활성 전송) 로드 밸런싱을 위해 **loadbalance** 러너를 구성하려면 팀 **JSON** 형식 구성 파일에 다음을 추가합니다.

```
{
  "device": "team0",
  "runner": {
    "name": "loadbalance",
    "tx_hash": ["eth", "ipv4", "ipv6"],
    "tx_balancer": {
      "name": "basic"
    }
  },
  "ports": {"em1": {}, "em2": {}}
}
```

기본 로드 밸런서를 사용한 활성 전송(Tx) 로드 밸런싱 구성.

자세한 내용은 teamd.conf(5) 매뉴얼 페이지를 참조하십시오.

8.13.6. LACP(802.3ad) Runner 구성

root 로 ethtool 을 링크 감시자로 사용하여 LACP 러너를 구성하려면 팀 JSON 형식 구성 파일에 다음을 추가합니다.

```
{
  "device": "team0",
  "runner": {
     "name": "lacp",
     "active": true,
     "fast_rate": true,
     "tx_hash": ["eth", "ipv4", "ipv6"]
  },
    "link_watch": {"name": "ethtool"},
    "ports": {"em1": {}, "em2": {}}
}
```

LACP(링크 집계 제어 프로토콜)에 연결할 수 있는 구성. LACP 러너는 ethtool 을 사용하여 링크 상태를 모니터링해야 합니다. arp_ping 의 경우와 같이 링크가 작동하지 않기 때문에 ethtool 만 링크 모니터링에 사용할 수 있습니다. 그 이유는 해당 링크를 먼저 설정해야 하며, 그 후에만 페킷 ARP가 포함된 후에 액세스할 수 있기 때문입니다. ethtool 을 사용하면 각 링크 계층을 개별적으로 모니터링하므로 이를 방지할 수 있습니다.

활성 부하 분산은 loadbalance 러너에 대해 수행되는 것과 동일한 방식으로 이 러너를 통해 가능합니다. 활성 전송(Tx) 로드 밸런싱을 활성화하려면 다음 섹션을 추가합니다.

```
"tx_balancer": {
      "name": "basic"
}
```

자세한 내용은 teamd.conf(5) 매뉴얼 페이지를 참조하십시오.

8.13.7. 링크 상태 모니터링 구성

다음 링크 상태 모니터링 방법을 사용할 수 있습니다. 메서드 중 하나를 구현하려면 루트 권한으로 실행되는 편집기를 사용하여 JSON 형식 문자열을 팀 JSON 형식 구성 파일에 추가합니다.

8.13.7.1. 링크 상태 모니터링을 위해 Ethtool 구성

들어오는 링크와 이에 대해 알림을 받은 러너 간에 기존 지연을 추가하거나 편집하려면 다음과 같이 섹션을 추가하거나 편집합니다.

```
"link_watch": {
        "name": "ethtool",
        "delay_up": 2500
}
```

다운되는 링크와 이에 대해 알림을 받은 러너 간에 기존 지연(밀리초)을 추가하거나 편집하려면 다음과 같이 섹션을 추가하거나 편집합니다.

```
"link_watch": {
        "name": "ethtool",
        "delay_down": 1000
}
```

8.13.7.2. 링크 상태 모니터링을 위한 ARP Ping 구성

팀 데몬 **teamd** 는 **ARP REQUEST**를 링크의 원격 끝에 있는 주소로 전송하여 링크가 작동 중인지 확인합니다. 사용되는 방법은 **arping** 유틸리티와 동일하지만 해당 유틸리티를 사용하지는 않습니다.

다음 예제와 유사한 **JSON** 형식으로 새 구성이 포함된 파일을 준비합니다.

```
{
        "device": "team0",
        "runner": {"name": "activebackup"},
        "link_watch":{
            "name": "arp_ping",
            "interval": 100,
            "missed_max": 30,
            "source_host": "192.168.23.2",
            "target_host": "192.168.23.1"
        },
          "ports": {
              "em1": {
                "prio": -10,
                "sticky": true
              },
```

```
            "em2": {
              "prio": 100
            }
        }
}
```

이 구성에서는 **arp_ping** 을 링크 감시자로 사용합니다. **missed_max** 옵션은 허용되는 최대 응답 수의 제한 값입니다(예:**ARP** 응답). 링크를 아래로 보고하기 전에 총 시간을 결정하려면 간격 옵션과 함께 선택해야 합니다.

팀 포트에 대한 새 구성을 로드하려면 다음을 수행합니다. **em2, JSON** 구성이 포함된 파일에서 **root** 로 다음 명령을 실행합니다.

```
~]# teamdctl port config update em2 JSON-config-file
```

이전 구성은 덮어쓰기되며 생략된 옵션은 기본값으로 재설정됩니다. 팀 데몬 제어 도구 명령 예제는 **teamdctl(8)** 도움말 페이지를 참조하십시오.

8.13.7.3. 링크 상태 모니터링을 위해 IPv6 NA/NS 구성

```
{
  "device": "team0",
    "runner": {"name": "activebackup"},
    "link_watch": {
    "name": "nsna_ping",
    "interval": 200,
    "missed_max": 15,
    "target_host": "fe80::210:18ff:feaa:bbcc"
    },
      "ports": {
        "em1": {
          "prio": -10,
          "sticky": true
        },
        "em2": {
          "prio": 100
        }
      }
}
```

NS/NA 패킷 전송 간격을 구성하려면 다음과 같이 섹션을 추가하거나 편집합니다.

```
"link_watch": {
    "name": "nsna_ping",
    "interval": 200
}
```

값은 밀리초 단위의 양수입니다. 링크를 **down**으로 보고하기 전에 총 시간을 확인하려면 **missed_max** 옵션과 함께 선택해야 합니다.

링크를 아래로 보고하기 전에 허용되는 최대 누락된 **NS/NA** 응답 패킷 수를 구성하려면 다음과 같이 섹션을 추가하거나 편집합니다.

```
"link_watch": {
    "name": "nsna_ping",
    "missed_max": 15
}
```

최대 누락된 **NS/NA** 응답 패킷 수. 이 번호를 초과하면 링크가 **down**으로 보고됩니다. **missed_max** 옵션은 허용되는 최대 응답 수의 제한 값입니다(예:**ARP** 응답). 링크를 아래로 보고하기 전에 총 시간을 결정하려면 간격 옵션과 함께 선택해야 합니다.

NS/NA 패킷의 **IPv6** 주소 주소로 확인되는 호스트 이름을 구성하려면 다음과 같이 섹션을 추가하거나 편집합니다.

```
"link_watch": {
    "name": "nsna_ping",
    "target_host": "MyStorage"
}
```

“target_host” 옵션에는 **NS/NA** 패킷의 대상 주소로 사용할 **IPv6** 주소로 변환할 호스트 이름이 포함되어 있습니다. **IPv6** 주소는 호스트 이름 대신 사용할 수 있습니다.

자세한 내용은 **teamd.conf(5)** 매뉴얼 페이지를 참조하십시오.

8.13.8. 포트 선택 덮어쓰기 구성

프레임을 전송하는 물리 포트는 일반적으로 팀 드라이버의 커널 부분에서 선택되며 사용자 또는 시스템 관리자와 관련이 없습니다. 출력 포트는 선택한 팀 모드(**teamd** 러너)의 정책을 사용하여 선택합니다. 그러나 때로는 나가는 트래픽의 특정 클래스를 특정 물리적 인터페이스로 안내하여 약간 더 복잡한 정책을 구현하는 것이 유용합니다. 기본적으로 팀 드라이버는 다중 대기열 인식이며 드라이버가 를 초기화할 때 **16**개 대기열이 생성됩니다. 더 많거나 적은 대기열이 필요한 경우 **Netlink** 특성 **tx_queues** 를 사용하여 팀 드라이버 인스턴스 생성 중에 이 값을 변경할 수 있습니다.

포트 구성 옵션 **queue _id**로 포트의 큐 **ID**를 다음과 같이 설정할 수 있습니다.

```
{
  "queue_id": 3
}
```

이러한 큐 **ID**는 **the tc** 유틸리티와 함께 사용하여 특정 포트 장치에서 전송할 특정 트래픽을 반향하도록 다중 대기열 대기열 규제 및 필터를 구성할 수 있습니다. 예를 들어 위의 구성을 사용하고 **192.168.1.100** 에 바인딩된 모든 트래픽을 강제로 사용하려는 경우 **enp1s0** 팀에서 출력 장치로 다음 형식으로 **root** 로 명령을 실행합니다.

```
~]# tc qdisc add dev team0 handle 1 root multiq
~]# tc filter add dev team0 protocol ip parent 1: prio 1 u32 match ip dst \
  192.168.1.100 action skbedit queue_mapping 3
```

특정 포트에 트래픽을 바인딩하기 위해 러너 선택 논리를 재정의하는 이 메커니즘은 모든 러너와 함께 사용할 수 있습니다.

8.13.9. BPF 기반 Tx 포트 선택기 구성

loadbalance 및 **LACP** 러너는 패킷 해시를 사용하여 네트워크 트래픽 흐름을 정렬합니다. 해시 계산 메커니즘은**BPF(Berkeley Packet Filter)** 코드를 기반으로 합니다. **BPF** 코드는 나가는 패킷에 대한 정책을 결정하는 것이 아니라 해시를 생성하는 데 사용됩니다. 해시 길이는 **256** 변형을 제공하는 **8**비트입니다. 이는 다양한 소켓 버퍼 **(SKB)**가 동일한 해시를 가질 수 있으므로 동일한 링크를 통해 트래픽을 전달할 수 있음을 의미합니다. 짧은 해시를 사용하면 여러 링크에 걸쳐 부하 분산을 위해 트래픽을 다른 스트림으로 빠르게 정렬할 수 있습니다. 정적 모드에서 해시는 트래픽을 보내야 하는 포트를 결정하는 데만 사용됩니다. 활성 모드에서 러너는 완벽한 균형을 달성하기 위해 다른 포트에 해시를 다시 할당합니다.

다음 내용 유형 또는 문자열을 패킷 **Tx** 해시 계산에 사용할 수 있습니다.

- **eth -** 소스 및 대상 **MAC** 주소 사용.
- **VLAN - VLAN ID**를 사용합니다.
- **ipv4 -** 소스 및 대상 **IPv4** 주소 사용.
- **ipv6 -** 소스 및 대상 **IPv6** 주소를 사용합니다.

- *IP - 소스 및 대상 IPv4 및 IPv6 주소를 사용합니다.*
- *l3 - 소스 및 대상 IPv4 및 IPv6 주소 사용.*
- *TCP - 소스 및 대상 TCP 포트를 사용합니다.*
- *UDP - 소스 및 대상 UDP 포트를 사용합니다.*
- *SCTP - 소스 및 대상 SCTP 포트를 사용합니다.*
- *l4 - 소스 및 대상 TCP 및 UDP 및 SCTP 포트를 사용합니다.*

이러한 문자열은 로드 밸런싱 러너에 다음 형식으로 행을 추가하여 사용할 수 있습니다.

```
"tx_hash": ["eth", "ipv4", "ipv6"]
```

예를 보려면 8.13.5절. “loadbalance Runner 구성” 을 참조하십시오.

8.14. GUI를 사용하여 네트워크 팀 만들기

8.14.1. 팀 연결 설정

nm-connection-editor 를 사용하여 NetworkManager 를 지시하여 두 개 이상의 Wired 또는 InfiniBand 연결에서 팀을 생성할 수 있습니다. 먼저 티밍할 연결을 만들 필요가 없습니다. 팀을 구성하기 위해 프로세스의 일부로 구성할 수 있습니다. 구성 프로세스를 완료하려면 인터페이스의 MAC 주소를 사용할 수 있어야 합니다.

절차 8.1. nm-connection-editor를 사용하여 새 팀 연결 추가

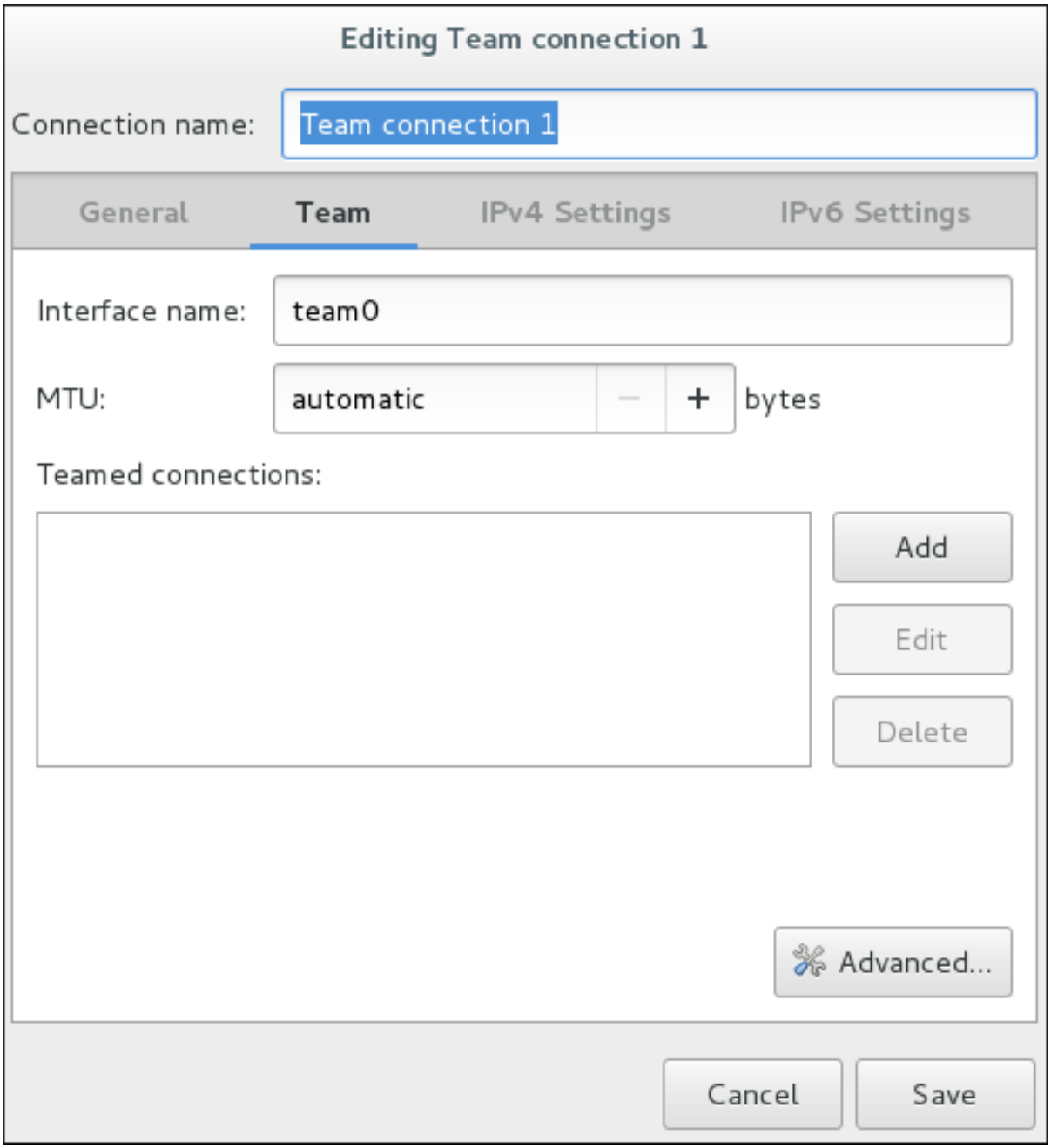
새 팀 연결을 추가하려면 아래 단계를 수행합니다.

1. *터미널에서 nm-connection-editor 를 입력합니다.*

```
~]$ nm-connection-editor
```

2. **Add**(추가) 단추를 클릭합니다. **Choose a Connection Type**(연결 유형 선택) 창이 표시됩니다. **Team**(팀)을 선택하고 **Create**(만들기)를 클릭합니다. **Editing Team connection 1** (팀 연결 편집 **1**) 창이 표시됩니다.

그림 **8.6. NetworkManager** 그래픽 사용자 인터페이스 메뉴 추가



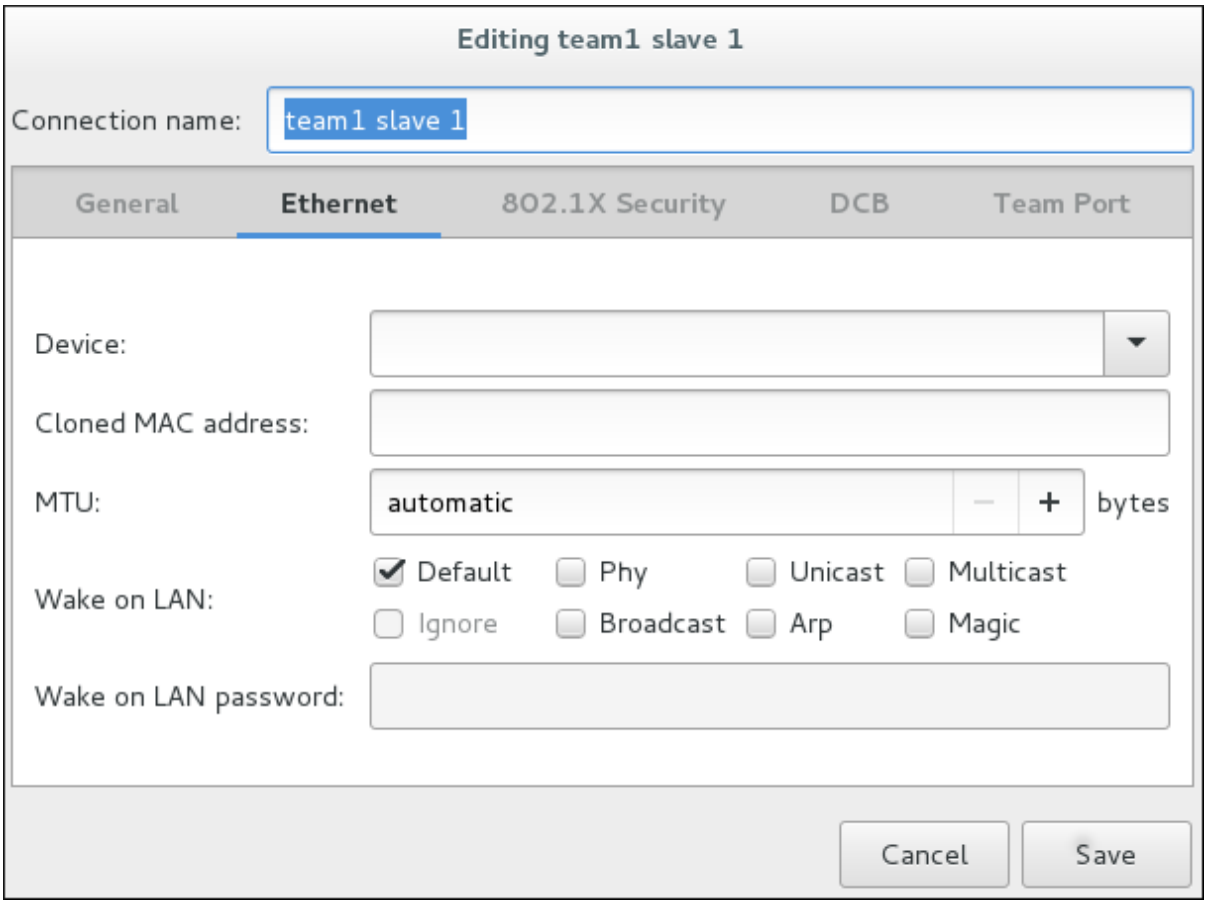
[D]

3. **Team**(팀) 탭에서 **Add**(추가)를 클릭하고 팀 연결에 사용할 인터페이스 유형을 선택합니다. **Create**(만들기) 단추를 클릭합니다. 포트 유형을 선택하는 대화 상자가 첫 번째 포트를 만들 때만

나타납니다. 그런 다음 모든 추가 포트에 대해 동일한 유형을 자동으로 사용합니다.

4. ***Editing team0 slave 1** 창이 표시됩니다.*

그림 8.7. NetworkManager 그래픽 사용자 인터페이스 추가 서브 연결 추가



[D]

5. *사용자 지정 포트 설정을 적용할 경우 **Team Port**(팀 포트) 탭을 클릭하고 **JSON** 구성 문자열을 입력하거나 파일에서 가져옵니다.*

6. ***Save**(저장) 버튼을 클릭합니다.*

7. *티밍된 포트의 이름이 티밍된 연결 창에 표시됩니다. **Add**(추가) 버튼을 클릭하여 추가 포트 연결을 추가합니다.*

8. *설정을 검토 및 확인한 다음 **Save** (저장) 단추를 클릭합니다.*

9. 아래에서 8.14.1.1절. “팀 탭 구성” 를 참조하여 팀별 설정을 편집합니다.

절차 8.2. 기존 팀 연결 편집

기존 팀 연결을 편집하려면 아래 단계를 수행합니다.

1. 터미널에서 **nm-connection-editor** 를 입력합니다.

```
~]$ nm-connection-editor
```

2. 편집할 연결을 선택하고 **Edit** (편집) 단추를 클릭합니다.

3. **General**(일반) 탭을 선택합니다.

4. **Editing** (편집) 대화 상자의 **5**가지 설정은 대부분의 연결 유형에 일반적으로 적용됩니다. 일반 탭을 참조하십시오.

- 연결 이름 **-** 네트워크 연결에 대한 설명이 포함된 이름을 입력합니다. 이 이름은 **Network** (네트워크) 창의 메뉴에 이 연결을 나열하는 데 사용됩니다.

- 자동 활성화의 연결 우선 순위 **-** 연결이 자동 연결로 설정되면 번호가 활성화됩니다(기본적으로 **0**). 숫자가 클수록 우선 순위가 높습니다.

- 사용 가능한 경우 이 네트워크에 자동으로 연결 **-** 사용 가능한 경우 **NetworkManager** 가 이 연결에 자동으로 연결되도록 하려면 이 상자를 선택합니다. 자세한 내용은 “제어 센터를 사용하여 기존 연결 편집”을 참조하십시오.

- 모든 사용자가 이 네트워크에 연결할 수 있습니다 **-** 이 상자를 선택하여 시스템의 모든 사용자가 사용할 수 있는 연결을 만듭니다. 이 설정을 변경하려면 **root** 권한이 필요할 수 있습니다. 자세한 내용은 3.4.5절. “GUI를 사용하여 시스템 전체 및 개인 연결 프로파일 관리” 을 참조하십시오.

- 이 연결을 사용할 때 **VPN**에 자동으로 연결 **-** **NetworkManager** 가 **VPN** 연결에 자동으

로 연결되도록 하려면 이 상자를 선택합니다. 드롭다운 메뉴에서 **VPN**을 선택합니다.

- 방화벽 영역 - 드롭다운 메뉴에서 방화벽 영역을 선택합니다. 방화벽 영역에 대한 자세한 내용은 *Red Hat Enterprise Linux 7 보안 가이드* 를 참조하십시오.

5. 아래에서 *8.14.1.1절. “팀 탭 구성”* 를 참조하여 팀별 설정을 편집합니다.

새 연결 (또는 수정된) 연결 저장 및 추가 설정 만들기

팀 연결 편집을 마치면 **Save** (저장) 버튼을 클릭하여 사용자 지정 구성을 저장합니다.

그런 다음 다음을 구성하려면 다음을 수행합니다.

- 연결의 **IPv4** 설정, **IPv4** 설정 탭을 클릭하고 계속 진행합니다. *5.4절. “IPv4 설정 구성”*

  또는

- 연결의 **IPv6** 설정, **IPv6** 설정 탭을 클릭하고 *5.5절. “IPv6 설정 구성”* 진행합니다.

8.14.1.1. 팀 탭 구성

새 팀 연결을 이미 추가한 경우 텍스트 상자에 사용자 지정 **JSON** 구성 문자열을 입력하거나 구성 파일을 가져올 수 있습니다. **Save** (저장)를 클릭하여 팀 인터페이스에 **JSON** 구성을 적용합니다.

JSON 문자열 예제는 참조하십시오. *8.13절. “teamd Runners 구성”*

새 팀을 추가하는 방법에 대한 지침은 *절차 8.1. “nm-connection-editor를 사용하여 새 팀 연결 추가”* 을 참조하십시오.

8.15. 추가 리소스

설치된 문서

- *teamd(8) 도움말 페이지 - teamd 서비스를 설명합니다.*
- *teamdctl(8) 도움말 페이지 - teamd 제어 툴을 설명합니다.*
- *teamd.conf(5) 도움말 페이지 - teamd 구성 파일을 설명합니다.*
- *teamnl(8) 도움말 페이지 - teamd Netlink 라이브러리를 설명합니다.*
- *bond2team(1) 도움말 페이지 - 본딩 옵션을 팀으로 변환하는 툴을 설명합니다.*

온라인 문서

http://www.w3schools.com/js/js_json_syntax.asp

JSON 구문에 대한 설명입니다.

9장. 네트워크 브리징 설정

네트워크 브리지는 MAC 주소를 기반으로 네트워크 간에 트래픽을 전달하는 링크 계층 장치입니다. 네트워크 트래픽을 수신하고 각 네트워크에 연결된 호스트를 학습하여 빌드하는 MAC 주소 테이블을 기반으로 전달 결정을 내립니다. 소프트웨어 브릿지는 Linux 호스트 내에서 사용할 수 있습니다. 예를 들어 NIC를 하나 이상의 가상 NIC와 공유하는 가상화 애플리케이션에서는 하드웨어 브리지를 에뮬레이션할 수 있습니다.

애드혹 또는 인프라 모드에서 작동하는 Wi-Fi 네트워크를 통해 브리지를 설정할 수 없습니다. 이는 항공 시간의 효율적인 사용을 위해 Wi-Fi에서 3개의 주소 프레임 사용을 지정하는 IEEEruntime 표준 때문입니다.

9.1. 텍스트 사용자 인터페이스 NMTUI를 사용하여 브리징 구성

텍스트 사용자 인터페이스 도구 nmtui 는 터미널 창에서 브리징을 구성하는 데 사용할 수 있습니다. 다음 명령을 실행하여 도구를 시작합니다.

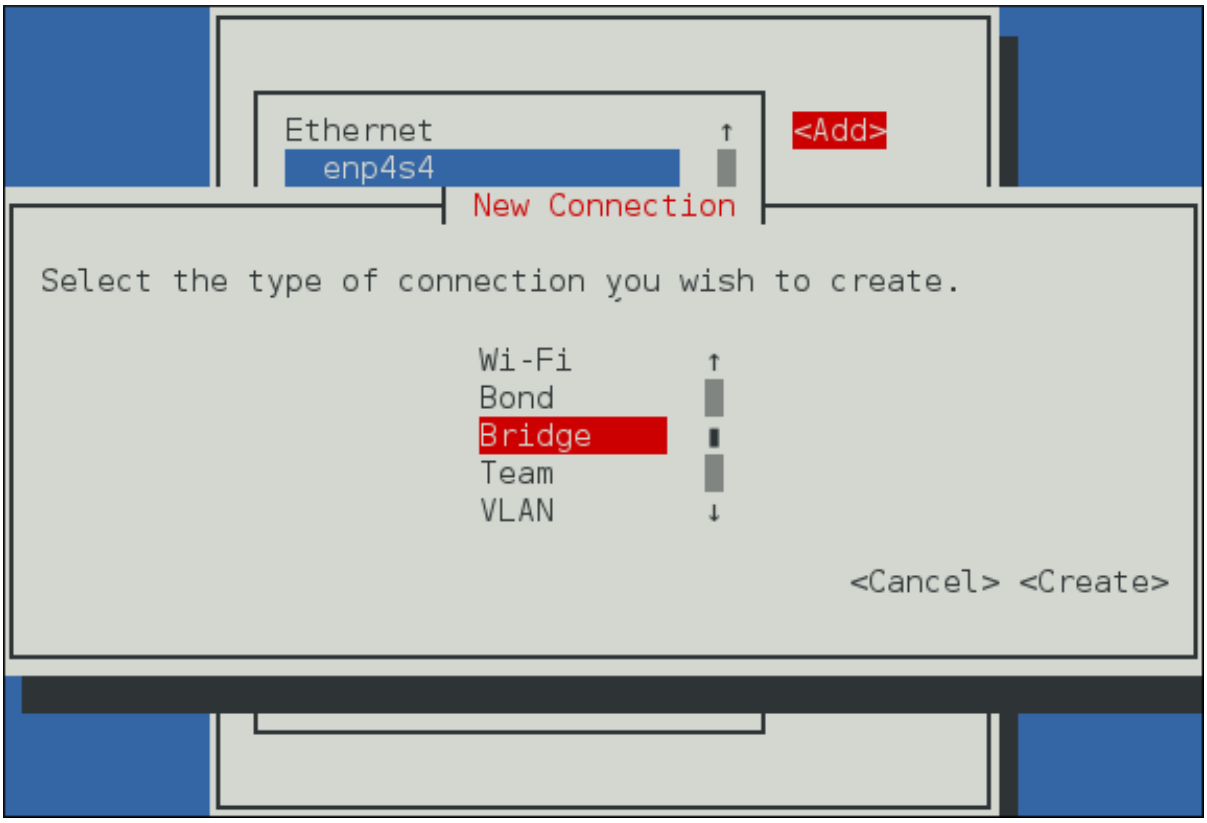
```
~]$ nmtui
```

텍스트 사용자 인터페이스가 나타납니다. 잘못된 명령은 사용 메시지를 인쇄합니다.

탐색하려면 화살표 키를 사용하거나 탭을 눌러 앞으로 이동하고 Shift+Tab 을 눌러 옵션을 다시 이동합니다. Enter 를 눌러 옵션을 선택합니다. Space 표시줄에서 확인란의 상태를 전환합니다.

1. 시작 메뉴에서 Edit a connection (연결 편집)을 선택합니다. Add 를 선택하면 New Connection (새 연결) 화면이 열립니다.

그림 9.1. NetworkManager 텍스트 사용자 인터페이스 Add a Bridge Connection 메뉴

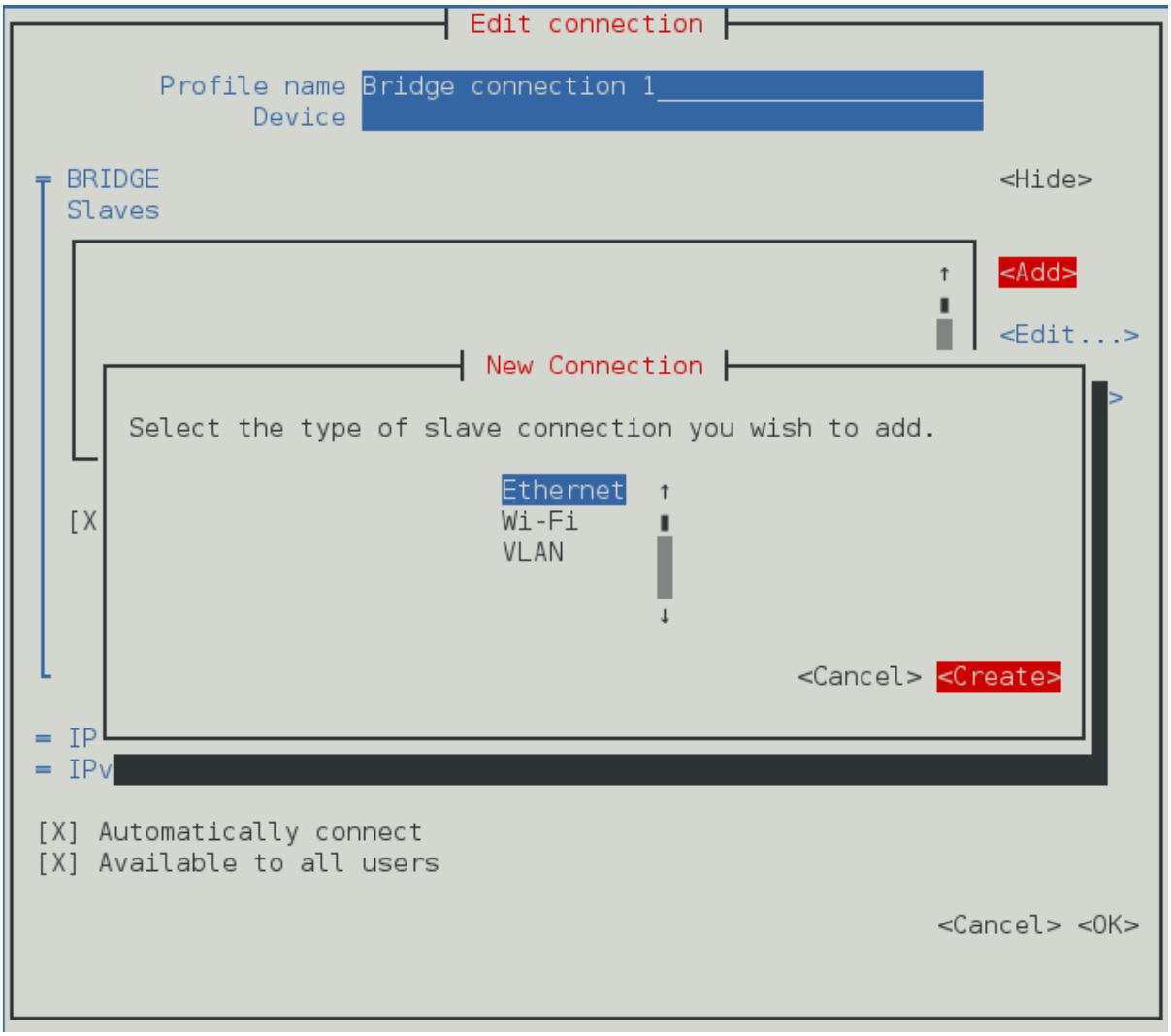


[D]

2. *브리지 를 선택하고 **Edit connection screen**(연결 편집) 화면이 열립니다.*

3. *브리지에 포트 인터페이스를 추가하려면 **Add** (추가)를 선택하면 **New Connection** (새 연결) 화면이 열립니다. 연결 유형을 선택한 후 **Create** (만들기) 버튼을 선택하여 브리지의 **Edit Connection** (연결 편집) 디스플레이가 표시됩니다.*

그림 9.2. NetworkManager 텍스트 사용자 인터페이스 추가 새로운 Bridge Slave Connection 메뉴



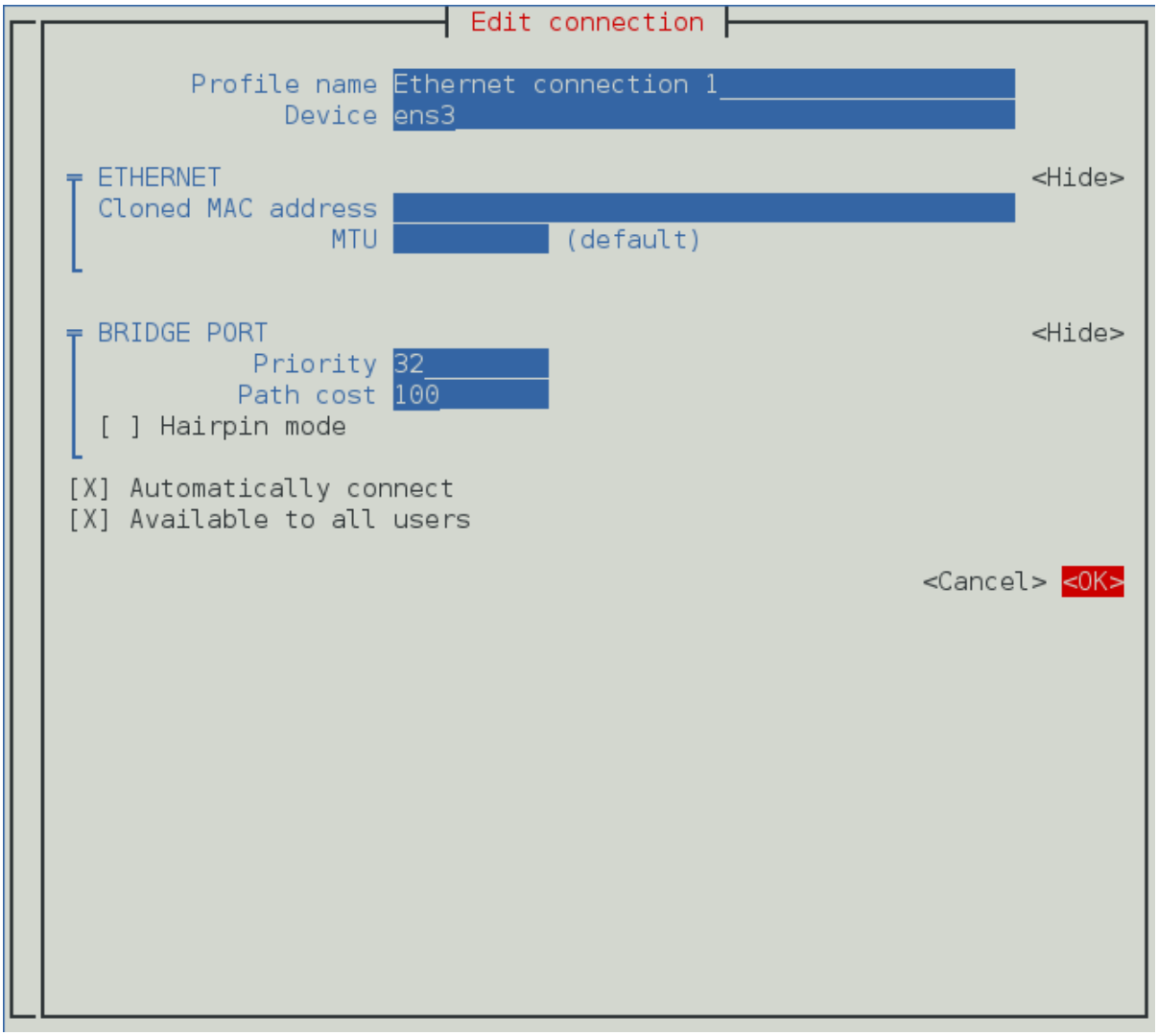
[D]

4. *장치 섹션에 필요한 포트의 장치 이름 또는 MAC 주소를 입력합니다. 필요한 경우 이더넷 레이블의 오른쪽에 표시를 선택하여 브리지의 MAC 주소로 복제된 MAC 주소를 입력합니다. OK(확인) 버튼을 선택합니다.*

참고

MAC 주소 없이 장치를 지정하면 장치 섹션이 자동으로 입력되며 Edit Connection(연결 편집) 창이 다시 로드되지만 장치를 성공적으로 찾은 경우에만 가능합니다.

그림 9.3. NetworkManager 텍스트 사용자 인터페이스 Configuring a Bridge Slave Connection 메뉴

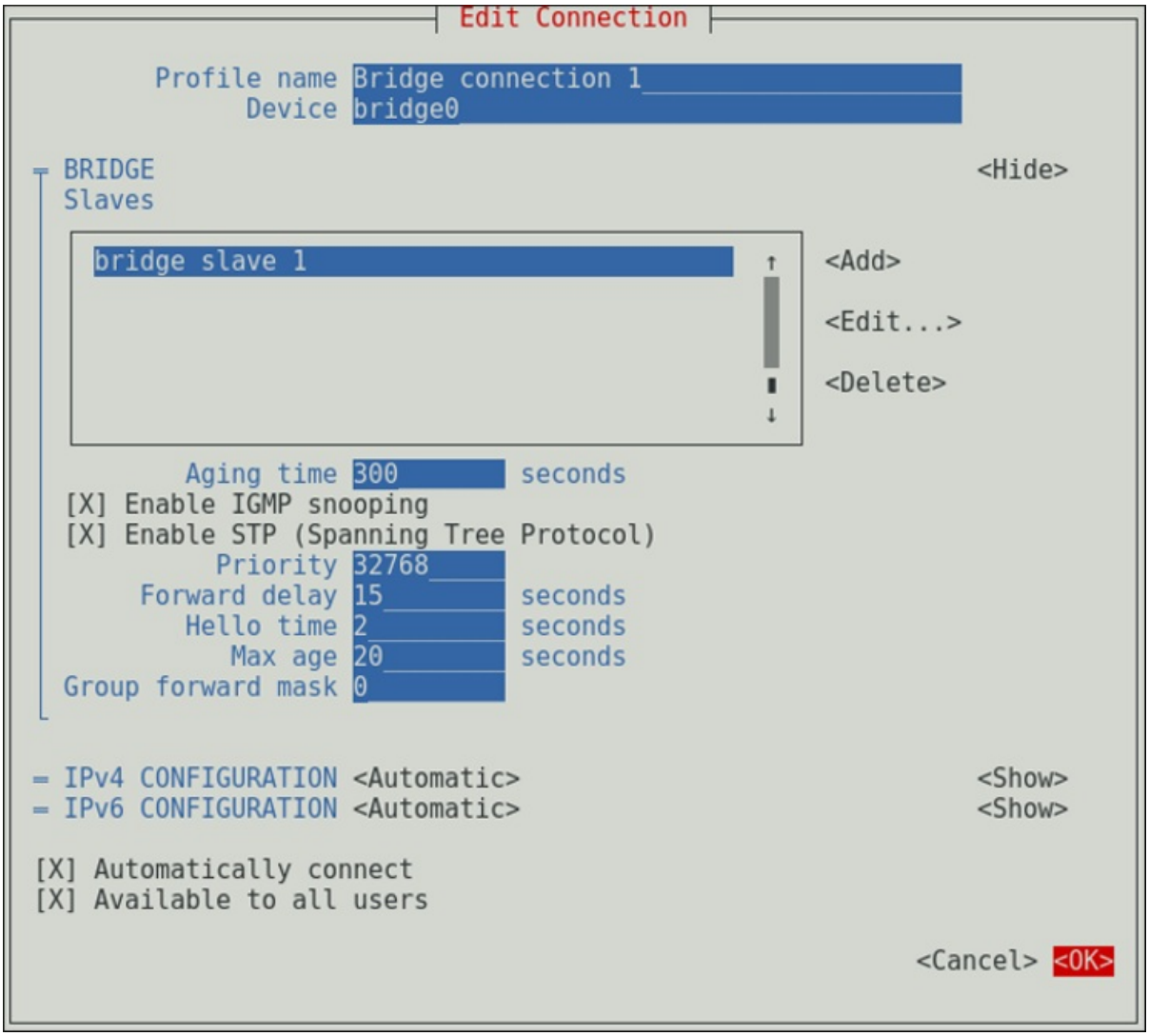


[D]

5. *브리지 포트의 이름이 **Slaves** 섹션에 표시됩니다. 위의 단계를 반복하여 추가 포트 연결을 추가합니다.*

6. ***OK(확인)** 버튼을 선택하기 전에 설정을 검토하고 확인합니다.*

그림 9.4. NetworkManager 텍스트 사용자 인터페이스 브리지 메뉴 구성



[D]

브리지 용어에 대한 정의는 9.4.1.1절. "브리지 탭 구성" 을 참조하십시오.

nmtui 설치에 대한 자세한 내용은 3.2절. "nmtui로 IP 네트워킹 구성" 을 참조하십시오.

9.2. NETWORKMANAGER 명령줄 도구인 NMCLI 사용

브리지를 만들려면 이름이 인 bridge-br0, 다음과 같이 명령을 실행합니다:

```
~]# nmcli con add type bridge ifname br0
Connection 'bridge-br0' (6ad5bba6-98a0-4f20-839d-c997ba7668ad) successfully added.
```

인터페이스 이름을 지정하지 않으면 기본값은 입니다. bridge, bridge-1, bridge-2, so on.

연결을 보려면 다음 명령을 실행합니다.

```
~]$ nmcli con show
NAME        UUID                                  TYPE           DEVICE
bridge-br0  79cf6a3e-0310-4a78-b759-bda1cc3eef8d  bridge         br0
enp1s0      4d5c449a-a6c5-451c-8206-3c9a4ec88bca  802-3-ethernet  enp1s0
```

스패닝 트리 프로토콜 (STP)은 기본적으로 활성화되어 있습니다. 사용된 값은 IEEE 802.1D-1998 표준에서 가져온 것입니다. 이 브릿지에 대해 STP 를 비활성화하려면 다음과 같이 명령을 실행합니다:

```
~]# nmcli con modify bridge-br0 bridge.stp no
```

이 브리지에 대해 802.1D STP 를 다시 활성화하려면 root 로 명령을 실행합니다:

```
~]# nmcli con modify bridge-br0 bridge.stp yes
```

802.1D STP 의 기본 브리지 우선 순위는 32768 입니다. 낮은 숫자는 루트 브리지 선택에서 선호됩니다. 예를 들어 우선 순위가 28672 인 브리지는 우선 순위 값이 32768 (기본값)인 브릿지에 우선하여 루트 브리지로 선택됩니다. 기본값이 아닌 값으로 브리지를 만들려면 다음과 같이 명령을 실행합니다.

```
~]$ nmcli con add type bridge ifname br5 stp yes priority 28672
Connection 'bridge-br5' (86b83ad3-b466-4795-aeb6-4a66eb1856c7) successfully added.
```

허용되는 값은 0 에서 65535 범위에 있습니다.

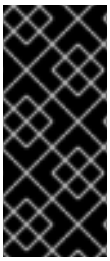
기존 브리지의 브리지 우선 순위를 기본값이 아닌 값으로 변경하려면 다음 형식으로 명령을 실행합니다.

```
~]$ nmcli connection modify bridge-br5 bridge.priority 36864
```

허용되는 값은 0 에서 65535 범위에 있습니다.

01:80:C2:00:00:00:00:00에서 01:80: C2:00:00: 0F 범위의 그룹 주소를 전달하도록 브리지 연결을 구성하려면 group-forward-mask 속성을 변경합니다. 이 속성은 16비트 마스크입니다. 각 비트는 전달되어야 하는 위의 범위의 그룹 주소에 해당합니다. 예를 들어 다음과 같습니다.

```
~]$ nmcli connection modify bridge-br5 bridge.group-forward-mask 8
```



중요

*해당 주소는 **STP(스패닝 트리 프로토콜), LACP(링크 집계 제어 프로토콜)** 및 이더넷 **MAC** 일시 중지 프레임에 사용되므로 **group-forward-mask** 속성에는 **1** 로 설정할 수 없습니다.*

브리지 설정을 보려면 다음 명령을 실행합니다.

```
~]$ nmcli -f bridge con show bridge-br0
```

***802.1D STP** 에 대한 추가 옵션은 **nmcli(1)** 도움말 페이지의 **bridge** 섹션에 나열되어 있습니다.*

*인터페이스를 추가하거나 할당하려면 다음을 수행합니다. **enp1s0**을 브리지로 보냅니다. **bridge-br0**다음과 같이 명령을 실행합니다.*

```
~]$ nmcli con add type ethernet ifname enp1s0 master bridge-br0
Connection 'bridge-slave-enp1s0' (70ffae80-7428-4d9c-8cbd-2e35de72476e) successfully added.
```

브리지에 기존 연결을 할당하려면 다음과 같이 진행합니다.

1. *컨트롤러 및 포트 유형 속성을 변경합니다. 예를 들어 **vlan100** 이라는 기존 **VLAN** 연결을 할당하려면 다음을 수행합니다.*

   ```
   ~]$ nmcli connection modify vlan100 master bridge-br0 slave-type bridge
   ```

2. *연결을 다시 활성화하여 변경 사항을 적용합니다.*

   ```
   ~]$ nmcli connection up vlan100
   ```

-

대화형 모드를 사용하여 값을 변경하려면 다음 명령을 실행합니다.

```
~]$ nmcli connection edit bridge-br0
```

nmcli 프롬프트에 배치됩니다.

```
nmcli> set bridge.priority 4096
nmcli> save
Connection 'bridge-br0' (79cf6a3e-0310-4a78-b759-bda1cc3eef8d) successfully saved.
nmcli> quit
```

nmcli 에 대한 소개는 **3.3절. “nmcli로 IP 네트워킹 구성”** 을 참조하십시오.

9.3. 명령줄 인터페이스(CLI) 사용

9.3.1. 커널 모듈 브리징이 설치되었는지 확인합니다.

Red Hat Enterprise Linux 7에서 **bridging** 모듈은 기본적으로 로드됩니다. 필요한 경우 다음 명령을 루트로 실행하여 모듈이 로드되었는지 확인할 수 있습니다:

```
~]# modprobe --first-time bridge
modprobe: ERROR: could not insert 'bridge': Module already in kernel
```

모듈에 대한 정보를 표시하려면 다음 명령을 실행합니다.

```
~]$ modinfo bridge
```

자세한 명령 옵션은 **modprobe(8)** 도움말 페이지를 참조하십시오.

9.3.2. 네트워크 브리지 만들기

네트워크 브리지를 만들려면 **/etc/sysconfig/network-scripts/** 디렉터리에 **ifcfg- brN**이라는 파일을 만들고 **N** 을 인터페이스의 번호로 바꿉니다(예: **0**).

파일의 내용은 이더넷 인터페이스와 같이 브리지되는 인터페이스 유형과 유사합니다. 이 예제의 차이점은 다음과 같습니다.

- ***DEVICE** 지시어에는 **brN** 형식의 인수로 인터페이스 이름이 부여됩니다. 여기서 **N** 은 인터페이스 번호로 교체됩니다.*

- ***TYPE** 지시어 에는 **Bridge** 인수가 지정됩니다. 이 지시문은 장치 유형을 결정하고 인수는 대소문자를 구분합니다.*

- *브리지 인터페이스 구성 파일에는 **IP** 주소가 제공되지만 실제 인터페이스 구성 파일에는 **MAC** 주소만 있어야 합니다(아래 참조).*

- *추가 지시문인 **DELAY=0** 은 브리지가 트래픽을 모니터링하는 동안 대기하지 못하게 하고, 호스트가 있는 위치를 학습하며, 필터링 결정에 기반한 **MAC** 주소 테이블을 구축하기 위해 추가됩니다. 라우팅 루프를 사용할 수 없는 경우 **15**초의 기본 지연이 필요하지 않습니다.*

*예 9.1. **ifcfg-br0** 인터페이스 구성 파일 예*

*다음은 고정 **IP** 주소를 사용하는 브리지 인터페이스 구성 파일의 예입니다.*

```
DEVICE=br0
TYPE=Bridge
IPADDR=192.168.1.1
PREFIX=24
BOOTPROTO=none
ONBOOT=yes
DELAY=0
```

다른 인터페이스를 완성하려면 다른 인터페이스가 생성되거나 기존 인터페이스가 수정되어 브리지 인터페이스를 가리킵니다.

*예 9.2. **ifcfg-enp1s0** 인터페이스 구성 파일 예*

*다음은 브리지 인터페이스를 가리키는 이더넷 인터페이스 구성 파일의 예입니다. **/etc/sysconfig/network-scripts/ifcfg-device_name** 에서 실제 인터페이스를 구성합니다. 여기서*

device_name 은 인터페이스 이름입니다.

```
DEVICE=device_name
TYPE=Ethernet
HWADDR=AA:BB:CC:DD:EE:FF
BOOTPROTO=none
ONBOOT=yes
BRIDGE=br0
```

선택적으로 NAME 지시문을 사용하여 이름을 지정합니다. 이름을 지정하지 않으면 NetworkManager 플러그인인 ifcfg-rh 가 연결 프로필의 이름을 "Type Interface" 형식으로 만듭니다. 이 예제에서 브리지의 이름은 Bridge br0 입니다. 또는 NAME=bridge-br0 이 ifcfg-br0 파일에 추가되면 연결 프로필의 이름은 bridge-br0 입니다.

참고

DEVICE 지시문의 경우 장치 유형을 결정하지 않으므로 거의 모든 인터페이스 이름을 사용할 수 있습니다. TYPE=Ethernet 은 엄격하게 필요하지 않습니다. TYPE 지시문이 설정되지 않은 경우 장치는 이더넷 장치로 처리됩니다(이름이 다른 인터페이스 구성 파일과 명시적으로 일치하지 않는 한).

지시문은 대소문자를 구분합니다.

HWADDR 지시문을 사용하여 하드웨어 또는 MAC 주소를 지정하면 11장. 일관된 네트워크 장치 이름 지정 에 설명된 장치 명명 절차에 영향을 미칩니다.

주의

원격 호스트에서 브리징을 구성하고 구성 중인 물리적 NIC를 통해 해당 호스트에 연결된 경우 계속하기 전에 연결 손실의 영향을 고려해야 합니다. 서비스를 다시 시작할 때 연결이 끊어지고 오류가 발생한 경우 연결을 다시 시작하지 못할 수 있습니다. 콘솔 또는 대역 외 액세스가 권장됩니다.

새 인터페이스 또는 최근 구성된 인터페이스를 열려면 다음 형식으로 root 로 명령을 실행합니다.

```
ifup device
```

*이 명령은 **NetworkManager** 가 실행 중인지 감지하고 **nmcli con load UUID**를 호출한 다음 **nmcli con up UUID** 를 호출합니다.*

*또는 모든 인터페이스를 다시 로드하려면 **root** 로 다음 명령을 실행합니다:*

```
~]# systemctl restart network
```

*이 명령은 네트워크 서비스를 중지하고 네트워크 서비스를 시작한 다음 **ONBOOT=yes** 를 사용하여 모든 **ifcfg** 파일의 **ifup** 을 호출합니다.*

참고

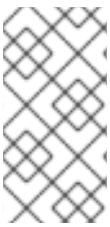
*기본 동작은 **NetworkManager** 가 **ifcfg** 파일에 대한 변경 사항을 인식하지 않고 다음에 인터페이스가 표시될 때까지 이전 구성 데이터를 계속 사용하는 것입니다. 이는 **NetworkManager.conf** 파일의 **monitor-connection-files** 옵션으로 설정합니다. 자세한 내용은 **NetworkManager.conf(5)** 매뉴얼 페이지를 참조하십시오.*

9.3.3. 본드를 통한 네트워크 브리지

*두 개 이상의 본딩 이더넷 인터페이스로 구성된 네트워크 브릿지의 예로 가상화 환경의 다른 공통 애플리케이션이므로 이제 이 인터페이스가 제공됩니다. 본딩된 인터페이스의 구성 파일에 익숙하지 않은 경우 를 참조하십시오. **7.4.2절. “채널 연결 인터페이스 만들기”***

다음과 같이 결합할 두 개 이상의 이더넷 인터페이스 구성 파일을 생성하거나 편집합니다.

```
DEVICE=interface_name
TYPE=Ethernet
SLAVE=yes
MASTER=bond0
BOOTPROTO=none
HWADDR=AA:BB:CC:DD:EE:FF
```



참고

***interface_name** 을 인터페이스 이름으로 사용하는 것이 일반적이지만 거의 모든 이름을 사용할 수 있습니다.*

다음과 같이 인터페이스 구성 파일 **/etc/sysconfig/network-scripts/ifcfg-bond0** 를 생성하거나 편집합니다.

```
DEVICE=bond0
ONBOOT=yes
BONDING_OPTS='mode=1 miimon=100'
BRIDGE=brbond0
```

bonding 모듈을 구성하고 본딩 매개 변수 목록을 확인하는 방법에 대한 자세한 지침과 지침은 *7.7절. “채널 연결 사용”* 에서 참조하십시오.

다음과 같이 하나의 인터페이스 구성 파일 **/etc/sysconfig/network-scripts/ifcfg-brbond0** 을 생성하거나 편집합니다.

```
DEVICE=brbond0
ONBOOT=yes
TYPE=Bridge
IPADDR=192.168.1.1
PREFIX=24
```

이제 **MASTER=bond0** 지시문을 사용하는 인터페이스 구성 파일이 두 개 이상 있습니다. 해당 파일은 **DEVICE=bond0** 지시문이 포함된 **/etc/sysconfig/network-scripts/ifcfg-bond0** 라는 구성 파일을 가리킵니다. 이 **ifcfg-bond0** 은 차례로 **IP** 주소가 포함된 **/etc/sysconfig/network-scripts/ifcfg-brbond0** 구성 파일을 가리키며 호스트 내부의 가상 네트워크에 대한 인터페이스 역할을 합니다.

새 인터페이스 또는 최근 구성된 인터페이스를 열려면 다음 형식으로 **root** 로 명령을 실행합니다.

```
ifup device
```

이 명령은 **NetworkManager** 가 실행 중인지 감지하고 **nmcli con load UUID**를 호출한 다음 **nmcli con up UUID** 를 호출합니다.

또는 모든 인터페이스를 다시 로드하려면 **root** 로 다음 명령을 실행합니다:

```
~]# systemctl restart network
```

이 명령은 네트워크 서비스를 중지하고 네트워크 서비스를 시작한 다음 **ONBOOT=yes** 를 사용하여 모든 **ifcfg** 파일의 **ifup** 을 호출합니다.

참고

기본 동작은 **NetworkManager** 가 **ifcfg** 파일에 대한 변경 사항을 인식하지 않고 다음에 인터페이스가 표시될 때까지 이전 구성 데이터를 계속 사용하는 것입니다. 이는 **NetworkManager.conf** 파일의 **monitor-connection-files** 옵션으로 설정합니다. 자세한 내용은 **NetworkManager.conf(5)** 매뉴얼 페이지를 참조하십시오.

9.4. GUI를 사용하여 네트워크 브리징 구성

브리지 인터페이스를 시작할 때 **NetworkManager** 는 **DHCP** 또는 **IPv6** 자동 구성과 같은 네트워크 종속 **IP** 구성을 시작하기 전에 하나 이상의 포트가 "전달" 상태가 될 때까지 기다립니다. 고정 **IP** 주소 지정은 포트 또는 포트가 연결되어 있거나 패킷을 전달하기 시작하기 전에 진행할 수 있습니다.

9.4.1. GUI를 사용하여 브리지 연결 설정

절차 **9.1. nm-connection-editor**를 사용하여 새 브리지 연결 추가

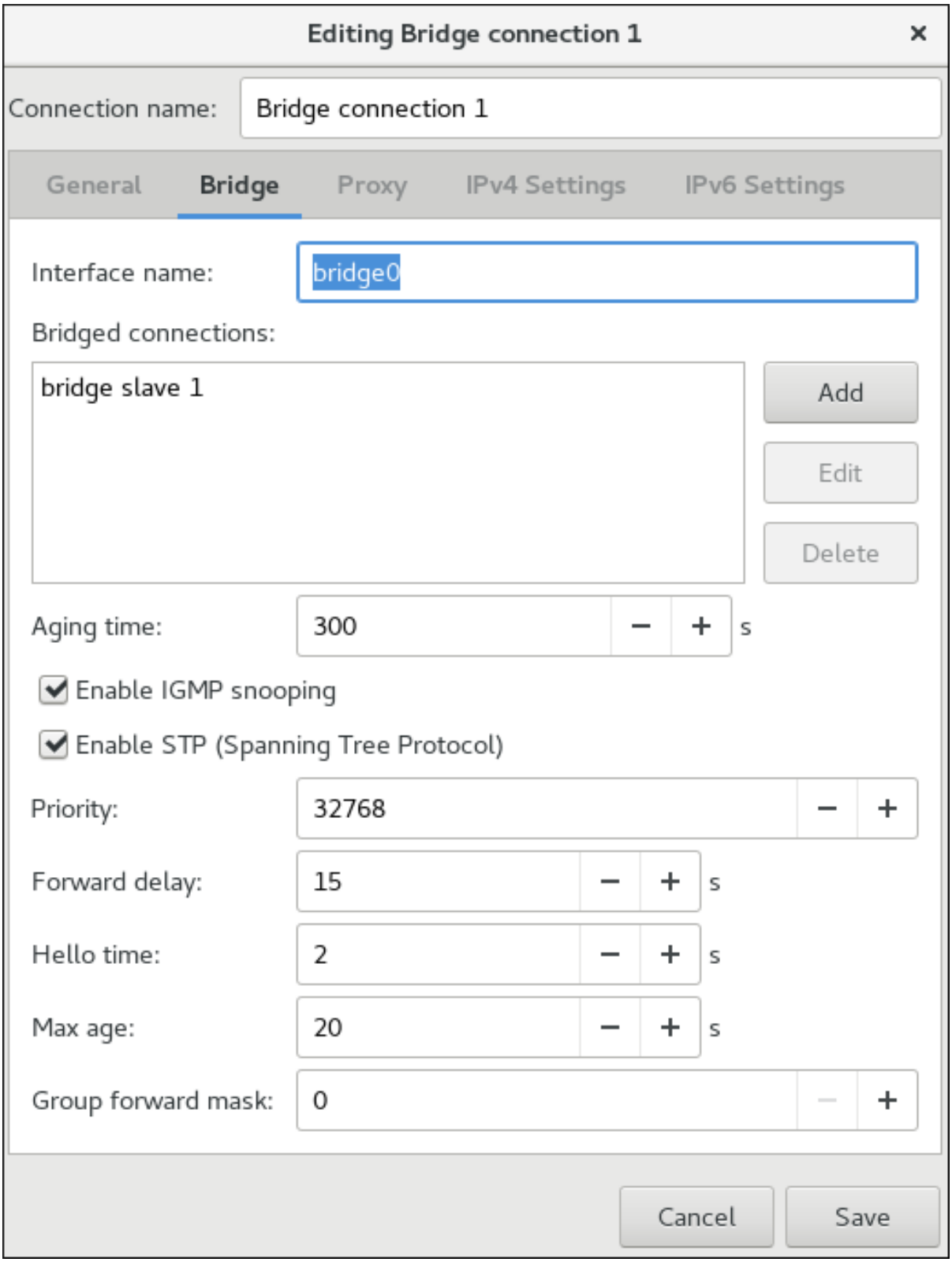
다음 지침에 따라 새 브리지 연결을 만듭니다.

1. 터미널에서 **nm-connection-editor** 를 입력합니다.

```
~]$ nm-connection-editor
```

2. **Add(추가)** 단추를 클릭합니다. **Choose a Connection Type(연결 유형 선택)** 창이 표시됩니다. **Bridge** 를 선택하고 **Create(만들기)**를 클릭합니다. **Editing Bridge connection 1** 창이 표시됩니다.

그림 9.5. 브릿지 연결 1 편집



[D]

3. *아래 절차 9.3. "브릿지에 포트 인터페이스 추가" 를 참조하여 포트 장치를 추가합니다.*

절차 9.2. 기존 브리지 연결 편집

1. 터미널에서 **nm-connection-editor** 를 입력합니다.

```
~]$ nm-connection-editor
```

2. 편집할 브릿지 연결을 선택합니다.

3. **Edit**(편집) 버튼을 클릭합니다.

연결 이름, 자동 연결 동작 및 가용성 설정 구성

편집 대화 상자의 **5**가지 설정은 모든 연결 유형에 공통적입니다. **General**(일반) 탭을 참조하십시오.

- 연결 이름 **-** 네트워크 연결에 대한 설명이 포함된 이름을 입력합니다. 이 이름은 **Network** (네트워크) 창의 메뉴에 이 연결을 나열하는 데 사용됩니다.

- 사용 가능한 경우 이 네트워크에 자동으로 연결 **-** 사용 가능한 경우 **NetworkManager** 가 이 연결에 자동으로 연결되도록 하려면 이 상자를 선택합니다. 자세한 내용은 *"제어 센터를 사용하여 기존 연결 편집"*을 참조하십시오.

- 모든 사용자가 이 네트워크에 연결할 수 있습니다 **-** 이 상자를 선택하여 시스템의 모든 사용자가 사용할 수 있는 연결을 만듭니다. 이 설정을 변경하려면 **root** 권한이 필요할 수 있습니다. 자세한 내용은 **3.4.5**절. *"GUI*를 사용하여 시스템 전체 및 개인 연결 프로파일 관리*"* 을 참조하십시오.

- 이 연결을 사용할 때 **VPN**에 자동으로 연결 **- NetworkManager** 가 **VPN** 연결에 자동으로 연결되도록 하려면 이 상자를 선택합니다. 드롭다운 메뉴에서 **VPN**을 선택합니다.

- **Firewall Zone** (방화벽 영역) **-** 드롭다운 메뉴에서 **Firewall Zone**(방화벽 영역)을 선택합니다. 방화벽 영역에 대한 자세한 내용은 **Red Hat Enterprise Linux 7** 보안 가이드 를 참조하십시오.

9.4.1.1. 브리지 탭 구성

인터페이스 이름

브리지에 대한 인터페이스 이름.

브리지된 연결

하나 이상의 포트 인터페이스입니다.

유효 기간

***MAC** 주소가 **MAC** 주소 전달 데이터베이스에 유지되는 시간(초)입니다.*

***IGMP** 스누핑 활성화*

*필요한 경우 확인란을 선택하여 장치에서 **IGMP** 스누핑을 활성화합니다.*

***STP (Spanning Tree Protocol)** 활성화*

*필요한 경우 확인란을 선택하여 **STP** 를 활성화합니다.*

priority

브리지 우선 순위입니다. 우선 순위가 가장 낮은 브리지가 루트 브리지로 선택됩니다.

전송 지연

*전달 상태를 시작하기 전에 **Listening** 및 **Learning** 상태를 모두 소비한 시간(초)입니다. 기본값은 **15**초입니다.*

***Hello** 타임*

*브리지 프로토콜 데이터 단위(**BPDU**)로 구성 정보를 보내는 시간 간격(초)입니다.*

최대 경과 시간

***BPDU**에서 구성 정보를 저장하는 최대 시간(초)입니다. 이 값은 **Hello Time + 1**보다 두 배이지만 **Forwarding delay** 빼 **1**보다 두 배 작아야 합니다.*

그룹 앞으로 마스크

*이 속성은 그룹 주소를 전달할 수 있는 그룹 주소 마스크입니다. 대부분의 경우 **01:80:C2:00:00:00:00**부터 **01:80: C2:00:00:0F** 범위의 그룹 주소는 브리지 장치에 의해 전달되지 않습니다. 이 속성은 **16**비트 마스크로, 각각 위 범위의 그룹 주소에 해당하며 전달되어야 합니다. 이러한*

주소는 **STP**(스패닝 트리 프로토콜), **LACP**(링크 집계 제어 프로토콜) 및 이더넷 **MAC** 일시 중지 프레임에 사용되므로 그룹 전달 마스크 속성에는 **0**,**1** ,**2** 비트를 **1**로 설정할 수 없습니다.

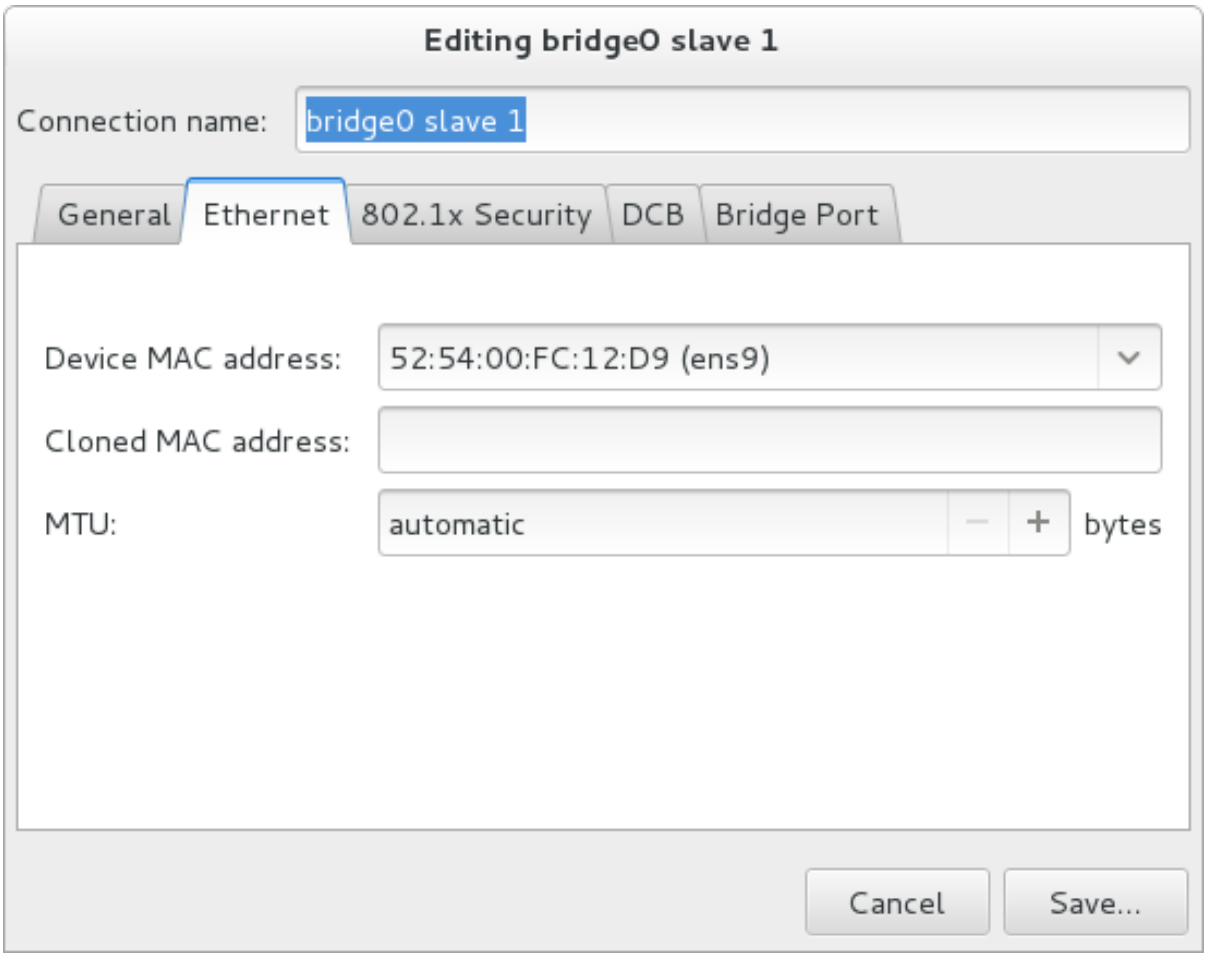
절차 9.3. 브릿지에 포트 인터페이스 추가

1. 브리지에 포트를 추가하려면 **Editing Bridge connection 1**창에서 **Bridge** 탭을 선택합니다. 필요한 경우 **절차 9.2.** "기존 브리지 연결 편집" 의 절차에 따라 이 창을 엽니다.

2. **Add**(추가)를 클릭합니다. **Choose a Connection Type**(연결 유형 선택) 메뉴가 표시됩니다.

3. 목록에서 만들 연결 유형을 선택합니다. 생성을 클릭합니다. 선택한 연결 유형에 적합한 창이 표시됩니다.

그림 9.6. NetworkManager 그래픽 사용자 인터페이스 추가 브릿지 연결

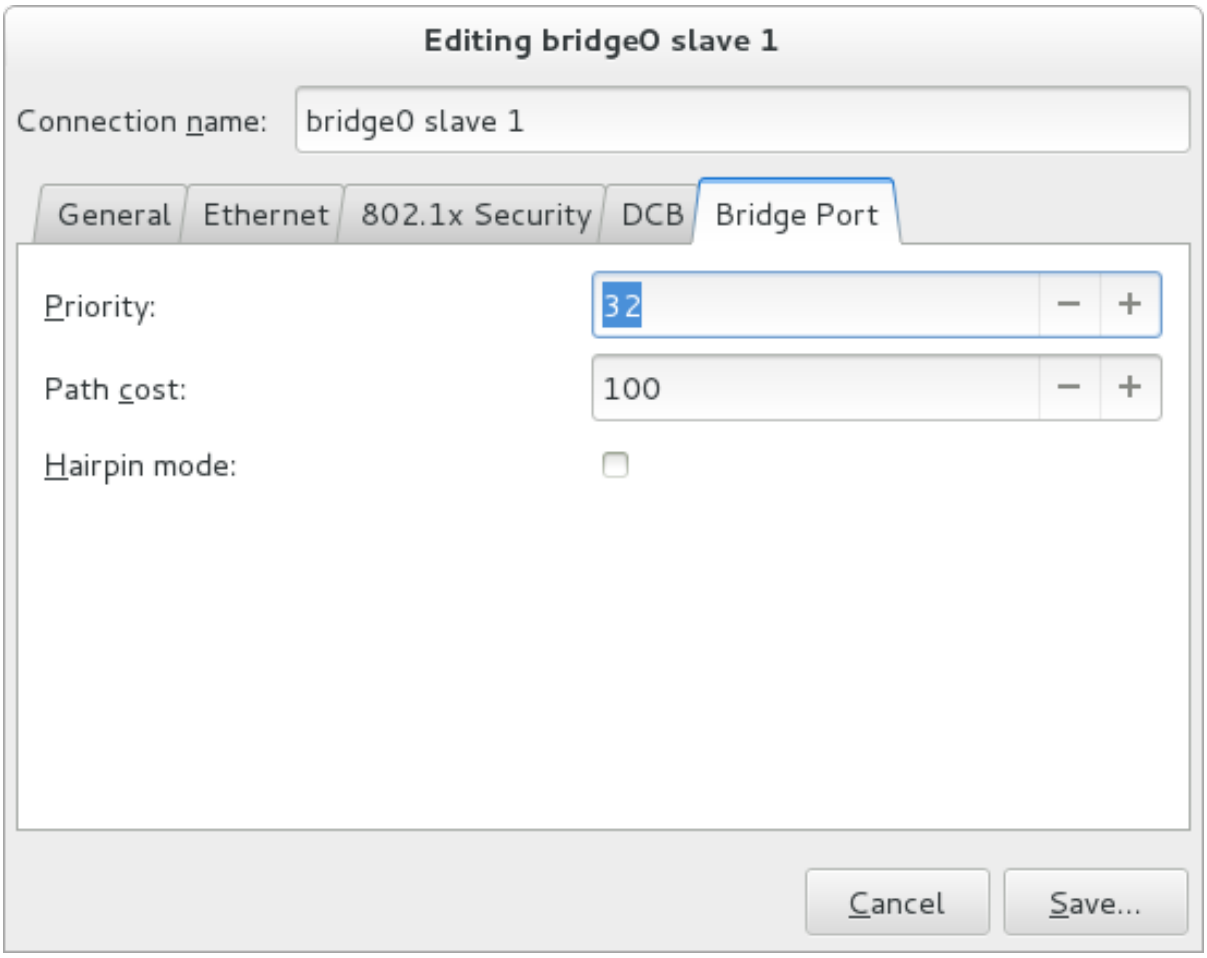


[D]

4. **Bridge Port**(브리지 포트) 탭을 선택합니다. 필요에 따라 우선 순위 및 경로 비용 구성. 브리지 포트의 **STP** 우선 순위는 **Linux** 커널에 의해 제한됩니다. 표준에서는 **0** 에서 **255** 범위의 범

위를 허용하지만 **Linux**는 **0 ~63**만 허용합니다. 이 경우 기본값은 **32** 입니다.

그림 **9.7. NetworkManager** 그래픽 사용자 인터페이스 브리지 포트 탭



[D]

5. 필요한 경우 **Hairpin** 모드 확인란을 선택하여 외부 처리용 프레임 전달을 활성화합니다. **VEPA(Virtual Ethernet port aggregator)** 모드라고도 합니다.

그런 다음 다음을 구성하려면 다음을 수행합니다.

- 이더넷 포트를 클릭하고 이더넷 탭을 클릭하고 "기본 설정 옵션 " 또는 으로 이동합니다.
- **Bond** 포트를 클릭하고 **Bond** 탭을 클릭하고 **7.8.1.1**절. "본딩 탭 구성" 또는 으로 이동합니다.
- 팀 포트를 클릭하고 팀 탭을 클릭하고 **8.14.1.1**절. "팀 탭 구성" 또는 으로 이동합니다.

- *VLAN 포트를 클릭하고 VLAN 탭을 클릭하고 10.5.1.1절. “VLAN 탭 구성” 또는 으로 이동 합니다.*

새 연결 (또는 수정된) 연결 저장 및 추가 설정 만들기

새 브리지 연결 편집을 마치면 Save (저장) 버튼을 클릭하여 사용자 지정 구성을 저장합니다. 프로필이 편집하는 동안 사용 중인 경우 NetworkManager 에서 변경 사항을 적용하도록 연결의 전원을 켭니다. 프로필이 OFF(꺼짐)인 경우 네트워크 연결 아이콘의 메뉴에서 ON(켜짐)으로 설정하거나 선택합니다. 새로운 연결 또는 변경된 연결 사용에 대한 정보는 3.4.1절. “제어 센터 GUI를 사용하여 네트워크에 연결” 을 참조하십시오.

네트워크 창에서 기존 연결을 추가로 선택하고 Options (옵션) 를 클릭하여 편집 대화 상자로 돌아갈 수 있습니다.

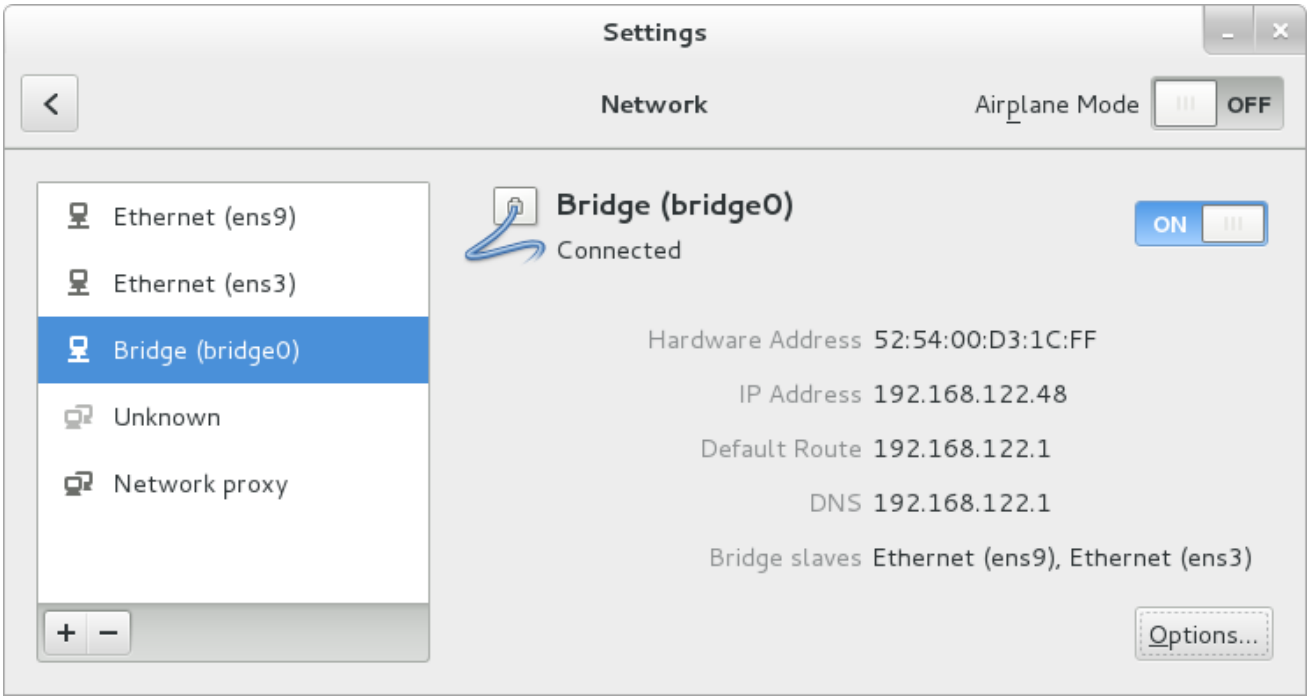
그런 다음 다음을 구성하려면 다음을 수행합니다.

- *연결의 IPv4 설정, IPv4 설정 탭을 클릭하고 5.4절. “IPv4 설정 구성”, 또는;*

- *연결의 IPv6 설정, IPv6 설정 탭을 클릭하고 5.5절. “IPv6 설정 구성” 진행합니다.*

저장한 후 브리지가 네트워크 설정 도구에 표시되고 각 포트가 화면에 표시됩니다.

그림 9.8. 브리지를 사용한 NetworkManager 그래픽 사용자 인터페이스



[D]

9.5. IPROUTE를 사용하여 이더넷 브리지 구성

iproute 패키지를 **bridge-utils** 대신 사용할 수 있습니다. 이를 통해 우선 순위,비용 또는 상태와 같은 브리지 포트 옵션을 설정할 수 있습니다.

ip 유틸리티를 사용하여 브리지 장치에 할당된 인터페이스 **enp1s0** 의 포트 옵션을 설정하려면 **root** 로 다음 명령을 실행합니다.

```
~]# ip link set enp1s0 type bridge_slave option
```

사용 가능한 옵션을 선택하려면 **ip** 유틸리티를 사용하여 **root** 로 다음 명령을 실행합니다:

```
~]# ip link help bridge_slave
    Usage: ... bridge_slave [ state STATE ] [ priority PRIO ] [cost COST ]
                    [ guard {on | off} ]
                    [ hairpin {on | off} ]
                    [ fastleave {on | off} ]
                    [ root_block {on | off} ]
                    [ learning {on | off} ]
                    [ flood {on | off} ]
```

포트 옵션에 대한 자세한 내용은 **ip-link(8)** 도움말 페이지를 참조하십시오.

9.6. 추가 리소스

- **nmcli(1)** 도움말 페이지 - **NetworkManager** 의 명령줄 도구를 설명합니다.
- **nmcli-examples(5)** 도움말 페이지 - **nmcli** 명령 예제 표시.
- **nm-settings(5)** 도움말 페이지 - **NetworkManager** 연결의 설정 및 매개 변수에 대한 설명.
- **ip-link(8)** 도움말 페이지 - 브리지 포트 옵션에 대한 설명입니다.

10장. 802.1Q VLAN 태깅 구성

VLAN을 만들기 위해 인터페이스를 상위 인터페이스라고 하는 다른 인터페이스 상단에 만듭니다 . VLAN 인터페이스는 인터페이스를 통과할 때 VLAN ID로 패킷 태그를 지정하며, 패킷이 태그되지 않습니다. VLAN 인터페이스는 다른 인터페이스와 유사하게 구성할 수 있습니다. 상위 인터페이스가 이더넷 인터페이스일 필요는 없습니다. 802.1Q VLAN 태그 지정 인터페이스는 브리지, 본딩 및 팀 인터페이스에 만들 수 있지만 몇 가지 유의 사항이 있습니다.

- *VLAN over bonds의 경우 VLAN 인터페이스를 열기 전에 본딩에 포트와 포트가 있어야 합니다. “” 포트 없이 본딩에 VLAN 인터페이스를 추가할 수 없습니다.*

- *VLAN 가상 장치는 상위 MAC 주소와 일치하도록 MAC 주소를 변경할 수 없기 때문에 fail_over_mac=follow 옵션을 사용하여 VLAN 포트를 구성할 수 없습니다. 이러한 경우 트래픽이 현재 잘못된 소스 MAC 주소와 함께 전송됩니다.*

- *네트워크 스위치를 통해 VLAN 태그가 지정된 패킷을 전송하려면 스위치를 올바르게 구성해야 합니다. 예를 들어 Cisco 스위치의 포트는 하나의 VLAN에 할당하거나 여러 VLAN에서 태그가 지정된 패킷을 허용하도록 트렁크 포트로 구성해야 합니다. 일부 벤더 스위치에서는 기본 VLAN 의 태그되지 않은 프레임을 트렁크 포트에서 처리할 수 있습니다. 일부 장치를 사용하면 기본 VLAN 을 활성화하거나 비활성화할 수 있으며, 다른 장치에는 기본적으로 비활성화되어 있습니다. 이러한 차이로 인해 두 스위치 간에 기본 VLAN 의 구성이 잘못되어 보안 위험이 발생할 수 있습니다. 예를 들어 다음과 같습니다.*

  한 스위치는 기본 VLAN 1 을 사용하고 다른 스위치는 기본 VLAN 10 을 사용합니다. 프레임을 삽입하지 않고 전달할 수 있는 경우 공격자가 VLAN을 건너뛸 수 있습니다. 이 일반적인 네트워크 도입 기법을 VLAN 홉핑이라고도 합니다.

  보안 위험을 최소화하려면 다음과 같이 인터페이스를 구성합니다.

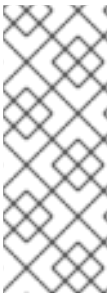
  스위치

  - *필요하지 않은 경우 트렁크 포트를 비활성화합니다.*
  - *트렁크 포트가 필요한 경우 태그가 지정되지 않도록 기본 VLAN 을 비활성화합니다.*

Red Hat Enterprise Linux 서버

- nftables or ebtables 유틸리티를 사용하여 수신 필터링에서 태그가 지정되지 않은 프레임을 삭제합니다.

- 일부 오래된 네트워크 인터페이스 카드, 루프백 인터페이스, Wimax 카드 및 일부 InfiniBand 장치는 VLAN 챌린지로 알려져 있으므로 VLAN을 지원할 수 없습니다. 이는 일반적으로 장치가 VLAN 헤더와 태그된 패킷과 연결된 더 큰 MTU 크기를 처리할 수 없기 때문입니다.



참고

VLAN 상단에 있는 본딩은 Red Hat에서 지원하지 않습니다. 자세한 내용은 VLAN 상단에 있는 포트 인터페이스로 본딩을 구성하는 것이 유효한 구성인 Red Hat 지식베이스 문서(Whether bond on top of VLAN as port interfaces is a valid configuration?)를 참조하십시오.

10.1. VLAN 인터페이스 구성 방법 선택

- NetworkManager 의 텍스트 사용자 인터페이스 도구인 nmtui 를 사용하여 VLAN 인터페이스를 구성하려면 다음을 진행합니다. 10.2절. “텍스트 사용자 인터페이스 nmtui를 사용하여 802.1Q VLAN 태깅 구성”

- NetworkManager 의 명령줄 도구 nmcli 를 사용하여 VLAN 인터페이스를 구성하려면 다음을 진행합니다. 10.3절. “명령줄 도구 nmcli를 사용하여 802.1Q VLAN 태그 지정 구성”

- 네트워크 인터페이스를 수동으로 구성하려면 10.4절. “명령줄을 사용하여 802.1Q VLAN 태그 지정 구성” 을 참조하십시오.

- 그래픽 사용자 인터페이스 도구를 사용하여 네트워크를 구성하려면 다음을 진행합니다. 10.5절. “GUI를 사용하여 802.1Q VLAN 태그 지정 설정”

10.2. 텍스트 사용자 인터페이스 NMTUI를 사용하여 802.1Q VLAN 태깅 구성

텍스트 사용자 인터페이스 도구 **nmtui** 는 터미널 창에서 **802.1Q VLAN**을 구성하는 데 사용할 수 있습니다. 다음 명령을 실행하여 도구를 시작합니다.

```
~]$ nmtui
```

텍스트 사용자 인터페이스가 나타납니다. 잘못된 명령은 사용 메시지를 인쇄합니다.

탐색하려면 화살표 키를 사용하거나 탭을 눌러 앞으로 이동하고 **Shift+Tab** 을 눌러 옵션을 다시 이동합니다. **Enter** 를 눌러 옵션을 선택합니다. **Space** 표시줄에서 확인란의 상태를 전환합니다.

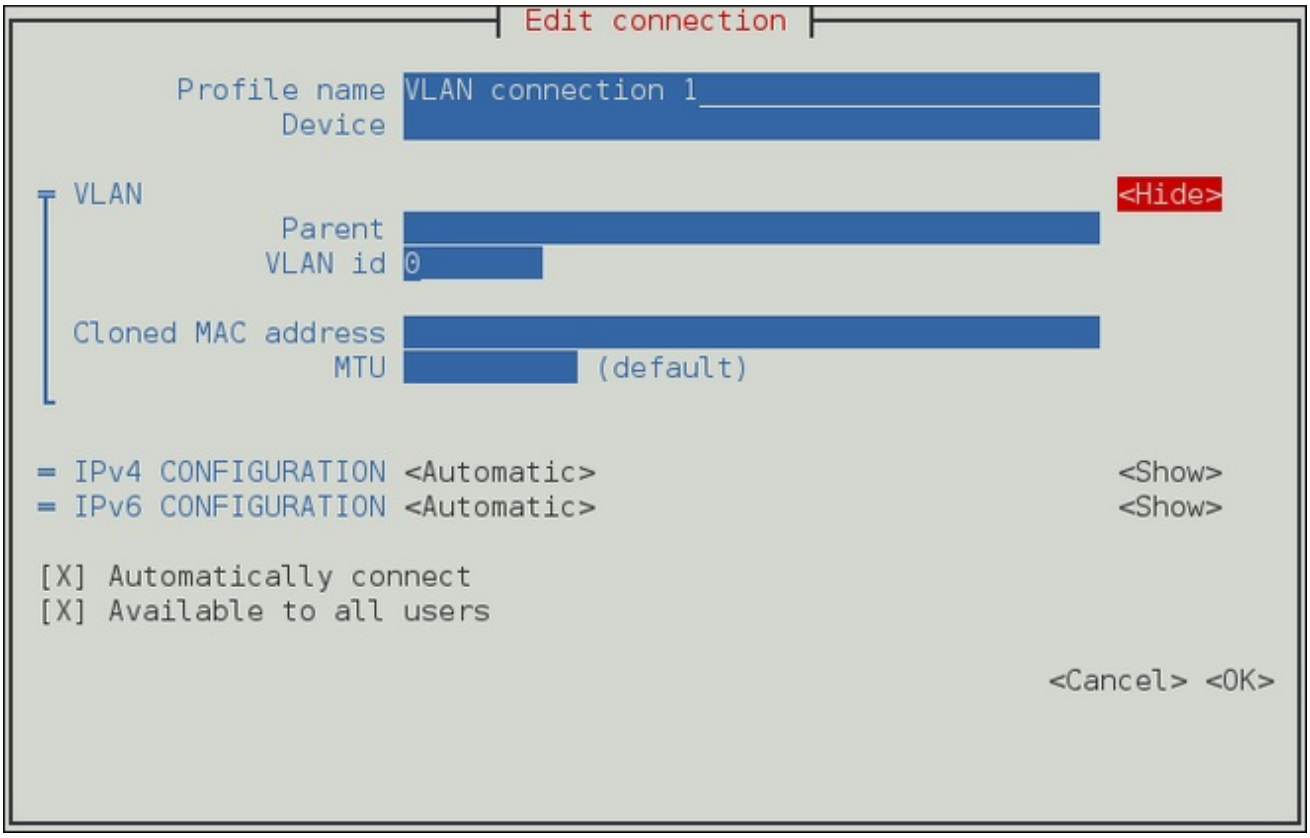
시작 메뉴에서 **Edit a connection** (연결 편집)을 선택합니다. **Add** 를 선택하면 **New Connection** (새 연결) 화면이 열립니다.

그림 10.1. NetworkManager 텍스트 사용자 인터페이스 VLAN 연결 메뉴 추가

[D]

VLAN 을 선택하고 **Edit connection** (연결 편집) 화면이 열립니다. 화면의 메시지에 따라 구성을 완료합니다.

그림 10.2. VLAN 연결 구성 NetworkManager 텍스트 사용자 인터페이스



[D]

VLAN 용어 정의는 10.5.1.1절. “VLAN 탭 구성” 를 참조하십시오.

nmtui 설치에 대한 자세한 내용은 3.2절. “nmtui로 IP 네트워킹 구성” 을 참조하십시오.

10.3. 명령줄 도구 NMCLI를 사용하여 802.1Q VLAN 태그 지정 구성

시스템에서 사용 가능한 인터페이스를 보려면 다음과 같이 명령을 실행합니다.

```
~]$ nmcli con show
NAME        UUID                                  TYPE            DEVICE
System enp2s0  9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  802-3-ethernet  enp2s0
System enp1s0  5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  802-3-ethernet  enp1s0
```

출력의 NAME 필드는 항상 연결 ID를 나타냅니다. 이 이름은 인터페이스 이름이 아니며, 인터페이스 이름이 같을 수도 있습니다. nmcli connection 명령에서 ID를 사용하여 연결을 식별할 수 있습니다. DEVICE 이름을 firewalld 와 같은 다른 애플리케이션에 사용합니다.

VLAN 인터페이스 VLAN 10 및 ID 10 을 사용하여 이더넷 인터페이스 enp1s0 에서 802.1Q VLAN 인터페이스를 생성하려면 다음과 같이 명령을 실행합니다.

```
~]$ nmcli con add type vlan ifname VLAN10 dev enp1s0 id 10
Connection 'vlan-VLAN10' (37750b4a-8ef5-40e6-be9b-4fb21a4b6d17) successfully added.
```

VLAN 인터페이스에 대해 Con-name 이 지정되지 않았기 때문에 유형 앞에 따라 이름이 인터페이스 이름에서 파생되었습니다. 또는 다음과 같이 con-name 옵션을 사용하여 이름을 지정합니다.

```
~]$ nmcli con add type vlan con-name VLAN12 dev enp1s0 id 12
Connection 'VLAN12' (b796c16a-9f5f-441c-835c-f594d40e6533) successfully added.
```

VLAN 인터페이스에 주소 할당

동일한 nmcli 명령을 사용하여 다른 인터페이스와 마찬가지로 정적 및 동적 인터페이스 주소를 할당할 수 있습니다.

예를 들어, 정적 IPv4 주소 및 게이트웨이를 사용하여 VLAN 인터페이스를 생성하는 명령은 다음과 같습니다.

```
~]$ nmcli con add type vlan con-name VLAN20 dev enp1s0 id 20 ip4 10.10.10.10/24 \
gw4 10.10.10.254
```

동적으로 할당된 VLAN 인터페이스를 생성하려면 다음과 같이 명령을 실행합니다.

```
~]$ nmcli con add type vlan con-name VLAN30 dev enp1s0 id 30
```

nmcli 명령을 사용하여 인터페이스를 구성하는 방법은 3.3.6절. “nmcli를 사용하여 네트워크에 연결”를 참조하십시오.

생성된 VLAN 인터페이스를 검토하려면 다음과 같이 명령을 실행합니다.

```
~]$ nmcli con show
NAME        UUID                                  TYPE           DEVICE
VLAN12      4129a37d-4feb-4be5-ac17-14a193821755  vlan           enp1s0.12
System enp2s0  9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  802-3-ethernet  enp2s0
System enp1s0  5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  802-3-ethernet  enp1s0
vlan-VLAN10    1be91581-11c2-461a-b40d-893d42fed4f4  vlan           VLAN10
```

새로 구성된 연결에 대한 자세한 정보를 보려면 다음과 같이 명령을 실행합니다.

```
~]$ nmcli -p con show VLAN12
===============================================================================
                Connection profile details (VLAN12)
===============================================================================
connection.id:                          VLAN12
connection.uuid:                        4129a37d-4feb-4be5-ac17-14a193821755
connection.interface-name:              --
connection.type:                        vlan
connection.autoconnect:                 yes
…
-------------------------------------------------------------------------------
802-3-ethernet.port:                    --
802-3-ethernet.speed:                   0
802-3-ethernet.duplex:                  --
802-3-ethernet.auto-negotiate:          yes
802-3-ethernet.mac-address:             --
802-3-ethernet.cloned-mac-address:      --
802-3-ethernet.mac-address-blacklist:
802-3-ethernet.mtu:                     auto
…
vlan.interface-name:                    --
vlan.parent:                            enp1s0
vlan.id:                                12
vlan.flags:                             0 (NONE)
vlan.ingress-priority-map:
vlan.egress-priority-map:
-------------------------------------------------------------------------------
===============================================================================
    Activate connection details (4129a37d-4feb-4be5-ac17-14a193821755)
===============================================================================
GENERAL.NAME:                           VLAN12
GENERAL.UUID:                           4129a37d-4feb-4be5-ac17-14a193821755
GENERAL.DEVICES:                        enp1s0.12
GENERAL.STATE:                          activating
[output truncated]
```

VLAN 명령의 추가 옵션은 **nmcli(1)** 도움말 페이지의 **VLAN** 섹션에 나열되어 있습니다. 도움말에서 **VLAN**이 생성된 장치를 상위 장치라고 합니다. 위의 예에서 장치는 인터페이스 이름 **enp1s0** 으로 지정되었으며 연결 **UUID** 또는 **MAC** 주소로 지정할 수도 있습니다.

*이름이 인 이더넷 인터페이스 **enp2s0** 에서 수신 우선 순위 매핑을 사용하여 **802.1Q VLAN** 연결 프로파일을 생성하려면 다음을 수행합니다. **VLAN1 ID 13** 은 다음과 같이 명령을 실행합니다.*

```
~]$ nmcli con add type vlan con-name VLAN1 dev enp2s0 id 13 ingress "2:3,3:5"
```

*위에서 만든 **VLAN**과 관련된 모든 매개 변수를 보려면 다음과 같이 명령을 실행합니다.*

```
~]$ nmcli connection show vlan-VLAN10
```

***MTU**를 변경하려면 다음과 같이 명령을 실행합니다.*

```
~]$ nmcli connection modify vlan-VLAN10 802.mtu 1496
```

***MTU** 설정은 네트워크 계층 패킷의 최대 크기를 결정합니다. 링크 계층 프레임이 전송할 수 있는 페이로드의 최대 크기는 네트워크 계층 **MTU**를 제한합니다. 표준 이더넷 프레임의 경우 **1500**바이트의 **MTU**를 나타냅니다. **802.1Q** 태그를 수용하기 위해 링크 계층 헤더 크기가 **4**바이트만큼 증가하므로 **VLAN**을 설정할 때 **MTU**를 변경할 필요가 없습니다.*

*작성 시 **connection.interface-name** 및 **vlan.interface-name** 은 동일해야 합니다(설정된 경우). 따라서 **nmcli** 의 대화형 모드를 사용하여 동시에 변경해야 합니다. **VLAN** 연결 이름을 변경하려면 다음과 같이 명령을 실행합니다.*

```
~]$ nmcli con edit vlan-VLAN10
nmcli> set vlan.interface-name superVLAN
nmcli> set connection.interface-name superVLAN
nmcli> save
nmcli> quit
```

***nmcli** 유틸리티를 사용하여 **802.1Q** 코드 작동 방식을 변경하는 **ioctl** 플래그를 설정하고 삭제할 수 있습니다. **NetworkManager** 에서 지원하는 **VLAN** 플래그는 다음과 같습니다.*

- ***0x01 -** 출력 패킷 헤더의 순서 변경*

- *0x02 - GVRP 프로토콜 사용*

- *0x04 - 인터페이스 및 마스터의 느슨한 바인딩*

VLAN의 상태는 상위 또는 마스터 인터페이스(VLAN이 생성된 인터페이스 또는 장치)의 상태와 동기화됩니다. 상위 인터페이스가 "down" 관리 상태로 설정되면 연결된 모든 VLAN이 down으로 설정되고 모든 경로가 라우팅 테이블에서 플러시됩니다. 플래그 0x04 에서는 상위에서 연결된 VLAN으로만 운영 상태가 전달되는 느슨한 바인딩 모드를 사용할 수 있지만 VLAN 장치 상태는 변경되지 않습니다.

VLAN 플래그를 설정하려면 다음과 같이 명령을 실행합니다.

```
~]$ nmcli connection modify vlan-VLAN10 vlan.flags 1
```

nmcli 에 대한 소개는 3.3절. "nmcli로 IP 네트워킹 구성" 을 참조하십시오.

10.4. 명령줄을 사용하여 802.1Q VLAN 태그 지정 구성

Red Hat Enterprise Linux 7에서 8021q 모듈은 기본적으로 로드됩니다. 필요한 경우 다음 명령을 루트로 실행하여 모듈이 로드되었는지 확인할 수 있습니다:

```
~]# modprobe --first-time 8021q
modprobe: ERROR: could not insert '8021q': Module already in kernel
```

모듈에 대한 정보를 표시하려면 다음 명령을 실행합니다.

```
~]$ modinfo 8021q
```

자세한 명령 옵션은 modprobe(8) 도움말 페이지를 참조하십시오.

10.4.1. ifcfg 파일을 사용하여 802.1Q VLAN 태그 지정 설정

1. */etc/sysconfig/network-scripts/ifcfg-device_name 에서 상위 인터페이스를 구성합니다. 여기서 device_name 은 인터페이스 이름입니다.*

   ```
   DEVICE=interface_name
   ```

```
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
```

2. */etc/sysconfig/network-scripts/* 디렉터리에서 **VLAN** 인터페이스 구성을 구성합니다. 구성 파일 이름은 상위 인터페이스와 **.** 문자 및 **VLAN ID** 번호여야 합니다. 예를 들어 **VLAN ID**가 **192** 이고 상위 인터페이스가 **enp1s0** 인 경우 구성 파일 이름은 **ifcfg-enp1s0.192:**

```
DEVICE=enp1s0.192
BOOTPROTO=none
ONBOOT=yes
IPADDR=192.168.1.1
PREFIX=24
NETWORK=192.168.1.0
VLAN=yes
```

동일한 인터페이스에서 **VLAN ID 193**와 같은 두 번째 **VLAN**을 구성해야 하는 경우 **VLAN** 구성 세부 정보를 사용하여 **enp1s0.193** 이라는 이름으로 새 파일을 추가합니다.

3. 변경 사항을 적용하려면 네트워킹 서비스를 다시 시작합니다. **root** 로서 다음 명령을 실행합니다.

```
~]# systemctl restart network
```

10.4.2. ip 명령을 사용하여 802.1Q VLAN 태그 지정 설정

이름이 **VLAN 8** 및 **ID 8** 인 이더넷 인터페이스 **enp1s0** 에서 **802.1Q VLAN** 인터페이스를 만들려면 다음과 같이 **root** 로 명령을 실행합니다.

```
~]# ip link add link enp1s0 name enp1s0.8 type vlan id 8
```

VLAN을 보려면 다음 명령을 실행합니다.

```
~]$ ip -d link show enp1s0.8
4: enp1s0.8@enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP mode DEFAULT
    link/ether 52:54:00:ce:5f:6c brd ff:ff:ff:ff:ff:ff promiscuity 0
    vlan protocol 802.1Q id 8 <REORDER_HDR>
```

ip 유틸리티는 앞에 **0x**가 있고 선행 **0** 인 경우 **8**진수 값으로 **VLAN ID**를 **16**진수 값으로 해석합니다 **.** 즉, **10**진수 값이 **22** 인 **VLAN ID**를 할당하려면 앞에 **0**을 추가하지 않아야 합니다.

VLAN을 제거하려면 다음과 같이 **root** 로 명령을 실행합니다.

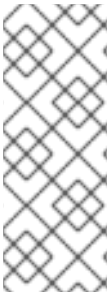
```
~]# ip link delete enp1s0.8
```

여러 **VLAN**에 속하는 여러 인터페이스를 사용하려면 실제 인터페이스 **enp1s0.1**에 적절한 **VLAN ID** 를 사용하여 로컬 **enp1s0.1** 및 **enp1s0.2** 를 생성합니다.

```
~]# ip link add link enp1s0 name enp1s0.1 type vlan id 1
   ip link set dev enp1s0.1 up
~]# ip link add link enp1s0 name enp1s0.2 type vlan id 2
   ip link set dev enp1s0.2 up
```

실제 장치에서 네트워크 스니퍼를 실행하면 **enp1s0** 상단에 **VLAN** 장치가 구성되어 있지 않아도 실제 장치에 도달할 태그가 지정된 프레임을 캡처할 수 있습니다. 예를 들어 다음과 같습니다.

```
tcpdump -nnei enp1s0 -vvv
```



참고

시스템이 종료되거나 다시 시작되면 명령 프롬프트에서 **ip** 명령을 사용하여 생성된 **VLAN** 인터페이스가 손실됩니다. 시스템을 다시 시작한 후 **VLAN** 인터페이스를 영구적으로 구성하려면 **ifcfg** 파일을 사용합니다. 보기 10.4.1절. “ifcfg 파일을 사용하여 802.1Q VLAN 태그 지정 설정”

10.5. GUI를 사용하여 802.1Q VLAN 태그 지정 설정

10.5.1. VLAN 연결 설정

nm-connection-editor 를 사용하여 기존 인터페이스를 상위 인터페이스로 사용하여 **VLAN**을 생성할 수 있습니다. **VLAN** 장치는 상위 인터페이스가 자동으로 연결되도록 설정된 경우에만 자동으로 생성됩니다.

절차 10.1. nm-connection-editor를 사용하여 새 VLAN 연결 추가

1. *터미널에서 **nm-connection-editor** 를 입력합니다.*

```
~]$ nm-connection-editor
```

2. ***Add**(추가) 단추를 클릭합니다. **Choose a Connection Type**(연결 유형 선택) 창이 표시됩니다. **VLAN** 을 선택하고 **Create**(생성)를 클릭합니다. **VLAN** 연결 1편집 창이 표시됩니다.*

3. ***VLAN** 탭의 **VLAN** 연결에 사용할 드롭다운 목록에서 상위 인터페이스를 선택합니다.*

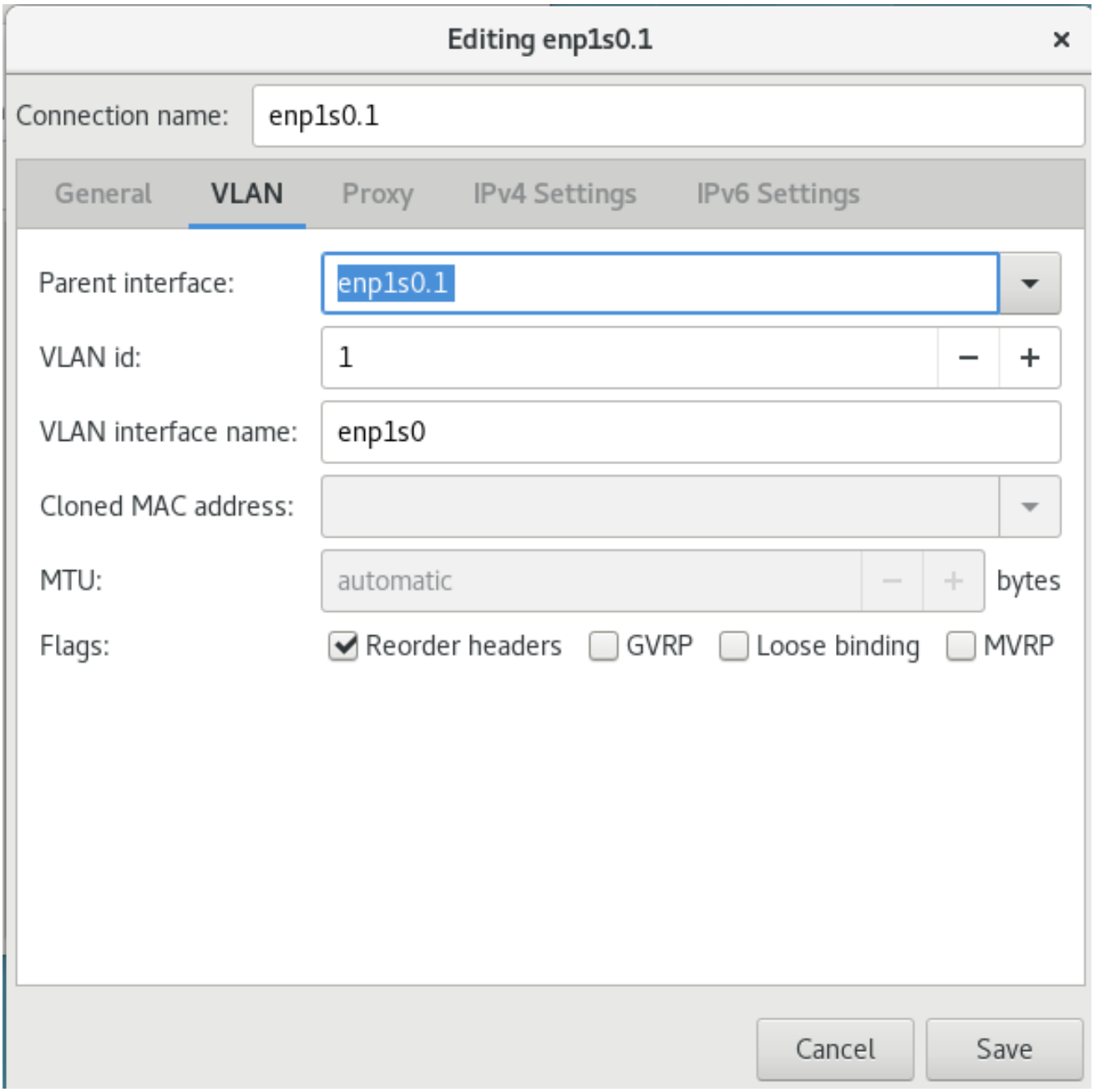
4. ***VLAN ID** 입력*

5. ***VLAN** 인터페이스 이름을 입력합니다. 이 이름은 생성될 **VLAN** 인터페이스의 이름입니다. 예를 들면 **enp1s0.1** 또는 **vlan2** 입니다. (일반적으로 상위 인터페이스 이름 **+, VLAN ID** 또는 **“vlan” “ 및” VLAN ID** 중 하나입니다.)*

6. *설정을 검토 및 확인한 다음 **Save** (저장) 단추를 클릭합니다.*

7. ***VLAN**별 설정을 편집하려면 **10.5.1.1절. “VLAN 탭 구성”** 에서 참조하십시오.*

그림 10.3. nm-connection-editor를 사용하여 새 VLAN 연결 추가



[D]

절차 10.2. 기존 VLAN 연결 편집

다음 단계에 따라 기존 VLAN 연결을 편집합니다.

1. *터미널에서 nm-connection-editor 를 입력합니다.*

```
~]$ nm-connection-editor
```

2. 편집할 연결을 선택하고 **Edit** (편집) 단추를 클릭합니다.

3. **General**(일반) 탭을 선택합니다.

4. 연결 이름, 자동 연결 동작 및 가용성 설정을 구성합니다.

   Editing(편집) 대화 상자 의 다음 설정은 모든 연결 유형에 일반적입니다.

   - 연결 이름 - 네트워크 연결에 대한 설명이 포함된 이름을 입력합니다. 이 이름은 네트워크 창의 **VLAN** 섹션에서 이 연결을 나열하는 데 사용됩니다.

   - 사용 가능한 경우 이 네트워크에 자동으로 연결 - 사용 가능한 경우 **NetworkManager** 가 이 연결에 자동으로 연결되도록 하려면 이 상자를 선택합니다. 자세한 내용은 *"제어 센터를 사용하여 기존 연결 편집"* 을 참조하십시오.

   - 모든 사용자가 사용 가능 - 이 상자를 선택하여 시스템의 모든 사용자가 사용할 수 있는 연결을 만듭니다. 이 설정을 변경하려면 **root** 권한이 필요할 수 있습니다. 자세한 내용은 **3.4.5**절. *"GUI*를 사용하여 시스템 전체 및 개인 연결 프로파일 관리*"* 을 참조하십시오.

5. **VLAN**별 설정을 편집하려면 **10.5.1.1**절. *"VLAN* 탭 구성*"* 에서 참조하십시오.

새 연결 (또는 수정된) 연결 저장 및 추가 설정 만들기

VLAN 연결 편집을 마치면 **Save** (저장) 버튼을 클릭하여 사용자 지정 구성을 저장합니다.

그런 다음 다음을 구성하려면 다음을 수행합니다.

- 연결의 **IPv4** 설정을 클릭하고 **IPv4 Settings**(**IPv4** 설정) 탭을 클릭하고 **5.4**절. *"IPv4* 설정 구성*"* 진행합니다.

  또는

- 연결의 IPv6 설정, IPv6 설정 탭을 클릭하고 5.5절. “IPv6 설정 구성” 진행합니다.

10.5.1.1. VLAN 탭 구성

이미 새 VLAN 연결을 추가한 경우(자세한 내용은 절차 10.1. “nm-connection-editor를 사용하여 새 VLAN 연결 추가” 참조), 상위 인터페이스와 VLAN ID를 설정하기 위해 VLAN 탭을 편집할 수 있습니다.

상위 인터페이스

이전에 구성된 인터페이스는 드롭다운 목록에서 선택할 수 있습니다.

VLAN ID

VLAN 네트워크 트래픽을 태그하는 데 사용할 식별 번호입니다.

VLAN 인터페이스 이름

생성될 VLAN 인터페이스의 이름입니다. 예를 들면 enp1s0.1 또는 vlan2 입니다.

복제된 MAC 주소

선택적으로 VLAN 인터페이스를 식별하는 데 사용할 대체 MAC 주소를 설정합니다. 이 VLAN에서 전송된 패킷의 소스 MAC 주소를 변경하는 데 사용할 수 있습니다.

MTU

선택적으로 VLAN 연결을 통해 패킷을 보낼 최대 전송 단위(MTU) 크기를 설정합니다.

10.6. IP 명령을 사용하여 BOND 및 브리지의 VLAN

본딩 및 브리지를 통해 VLAN을 사용하려면 다음과 같이 진행합니다.

1. 본드 장치를 루트로 추가 :

```
# ip link add bond0 type bond
# ip link set bond0 type bond miimon 100 mode active-backup
# ip link set em1 down
# ip link set em1 master bond0
```

```
# ip link set em2 down
# ip link set em2 master bond0
# ip link set bond0 up
```

2. *본딩 장치에서 **VLAN**을 설정합니다.*

```
# ip link add link bond0 name bond0.2 type vlan id 2
# ip link set bond0.2 up
```

3. *브리지 장치를 추가하고 **VLAN**을 연결합니다.*

```
# ip link add br0 type bridge
# ip link set bond0.2 master br0
# ip link set br0 up
```

10.7. NETWORKMANAGER 명령줄 도구를 사용하여 본딩 및 브리지의 VLAN, NMCLI

*본딩 및 브리지를 통해 **VLAN**을 사용하려면 다음과 같이 진행합니다.*

1. *본딩 장치를 추가합니다.*

```
~]$ nmcli connection add type bond con-name Bond0 ifname bond0 bond.options
"mode=active-backup,miimon=100" ipv4.method disabled ipv6.method ignore
```

*이 경우 본딩 연결은 **VLAN**의 "더 낮은 인터페이스"로만 제공되며 **IP** 주소를 가져오지 않습니다. 따라서 **ipv4.method** 비활성화 및 **ipv6.method ignore** 매개 변수가 명령줄에 추가되었습니다.*

2. *본딩 장치에 포트를 추가합니다.*

```
~]$ nmcli connection add type ethernet con-name Slave1 ifname em1 master bond0 slave-
type bond
~]$ nmcli connection add type ethernet con-name Slave2 ifname em2 master bond0 slave-
type bond
```

3.
브리지 장치를 추가합니다.

```
~]$ nmcli connection add type bridge con-name Bridge0 ifname br0 ip4 192.0.2.1/24
```

4.
*브리지 장치에 할당된 본딩 상단에 **VLAN** 인터페이스를 추가합니다.*

```
~]$ nmcli connection add type vlan con-name Vlan2 ifname bond0.2 dev bond0 id 2 master
br0 slave-type bridge
```

5.
생성된 연결을 확인합니다.

```
~]$ nmcli connection show
NAME     UUID                                  TYPE           DEVICE
Bond0    f05806fa-72c3-4803-8743-2377f0c10bed  bond           bond0
Bridge0  22d3c0de-d79a-4779-80eb-10718c2bed61  bridge         br0
Slave1   e59e13cb-d749-4df2-aee6-de3bfaec698c  802-3-ethernet  em1
Slave2   25361a76-6b3c-4ae5-9073-005be5ab8b1c  802-3-ethernet  em2
Vlan2    e2333426-eea4-4f5d-a589-336f032ec822  vlan           bond0.2
```

10.8. VLAN 스위치 모드 구성

***Red Hat Enterprise Linux** 시스템은 주로 라우터로 사용되며 네트워크 인터페이스에서 고급 **VLAN** 구성을 활성화합니다. 이더넷 인터페이스가 스위치에 연결되어 있고 실제 인터페이스를 통해 **VLAN**이 실행 중인 경우 **switchport** 모드를 설정해야 합니다. **Red Hat Enterprise Linux** 서버 또는 워크스테이션은 일반적으로 하나의 **VLAN**에만 연결되어 **switchport** 모드 액세스 와 기본 설정입니다.*

*특정 시나리오에서 태그가 지정된 여러 **VLAN**은 스위치와 **Red Hat Enterprise Linux** 시스템 간에 이더넷인 동일한 실제 링크를 사용합니다. 이 경우 두 끝 끝에 **switchport** 모드 트렁크 가 있어야 합니다.*

*예를 들어 **Red Hat Enterprise Linux** 시스템을 라우터로 사용하는 경우 시스템에서 라우터 뒤의 다양한 **VLAN** 패킷을 동일한 실제 이더넷을 통한 스위치로 전달하고 해당 **VLAN** 간의 분리를 유지해야 합니*

다.

예를 들어 10.3절. “명령줄 도구 nmcli를 사용하여 802.1Q VLAN 태그 지정 구성” 의 설정을 사용하여 Cisco switchport 모드 트렁크 를 사용합니다. 인터페이스에서 IP 주소만 설정하면 Cisco switchport 모드 액세스를 사용하십시오.

10.9. 추가 리소스

- ip-link(8) 도움말 페이지 - ip 유틸리티의 네트워크 장치 구성 명령을 설명합니다.
- nmcli(1) 도움말 페이지 - NetworkManager 의 명령줄 도구를 설명합니다.
- nmcli-examples(5) 도움말 페이지 - nmcli 명령 예제 표시.
- nm-settings(5) 도움말 페이지 - NetworkManager 연결의 설정 및 매개 변수에 대한 설명.
- nm-settings-ifcfg-rh(5) 도움말 페이지 - /etc/sysconfig/network-scripts/ifcfg-* 파일의 ifcfg- rh 설정 설명.

11장. 일관된 네트워크 장치 이름 지정

Red Hat Enterprise Linux는 네트워크 인터페이스에 대한 일관되고 예측 가능한 네트워크 장치 명명 방법을 제공합니다. 이러한 기능은 인터페이스를 더 쉽게 찾고 구분하기 위해 시스템의 네트워크 인터페이스 이름을 변경합니다.

전통적으로 **Linux**의 네트워크 인터페이스는 다음과 같습니다. **eth[0123...]s0**그러나 이러한 이름이 새 시의 실제 레이블과 반드시 일치하지는 않습니다. 다중 네트워크 어댑터가 있는 최신 서버 플랫폼은 이러한 인터페이스를 결정적이지 않고 반감적인 명명에 직면할 수 있습니다. 이는 마더보드(**Lan-on-Motherboard** 또는 **LOM**) 및 추가 기능(단일 및 멀티포트) 어댑터에 내장된 네트워크 어댑터 모두에 영향을 미칩니다.

Red Hat Enterprise Linux에서 **udev** 는 다양한 명명 체계를 지원합니다. 기본값은 펌웨어, 토폴로지 및 위치 정보를 기반으로 고정 이름을 할당하는 것입니다. 이렇게 하면 이름이 완전히 자동이고 완전히 예측 가능하며 하드웨어가 추가 또는 제거되어도 수정되고(재정화되지 않음) 손상된 하드웨어를 원활하게 교체할 수 있다는 이점이 있습니다. 단점은 때때로 읽기가 더 어려울 수 있다는 것입니다. **eth** 또는 **wla** 일반적으로 사용되는 이름. 예를 들어 다음과 같습니다. **enp5s0**.

주의

일관된 네트워크 장치 이름 지정을 비활성화하지 마십시오. 이 이름은 시스템을 사용할 수 있기 때문입니다 **ethX** 스타일 이름(여기서 **X**는 특정 인터페이스에 해당하는 고유한 번호이며 부팅 프로세스 중 네트워크 인터페이스의 이름이 다를 수 있습니다. 자세한 내용은 **11.10**절. “네트워크 장치 이름 지정 문제 해결” 의 내용을 참조하십시오.

11.1. SCHEMES 계층 이름 지정

기본적으로 **systemd** 는 지원되는 명명 체계를 적용하기 위해 다음 정책을 사용하여 인터페이스 이름을 지정합니다.

- 스키마 **1:** 펌웨어 또는 **BIOS**가 제공하는 온보드 장치(예: **eno1**)에 대한 색인 번호를 포함하는 이름은 펌웨어 또는 **BIOS**의 해당 정보가 적용 가능하고 사용 가능한 경우 적용되며, 그렇지 않으면 스키마 **2**로 대체합니다.

- 스키마 **2:** 펌웨어 또는 **BIOS**가 제공하는 **PCI Express** 핫플러그 슬롯 인덱스 번호(예: **ens1**)는펌웨어 또는 **BIOS**의 정보가 적용 가능하며 사용 가능한 경우 스키마 **3**으로 대체되는 이름이 적

용됩니다.

- *체계 **3:** 하드웨어 커넥터의 물리적 위치를 통합한 이름(예: **enp2s0**)은해당하는 경우 적용되며, 다른 모든 경우에는 스키마 **5**로 직접 대체합니다.*

- *스키마 **4:** 인터페이스의 **MAC** 주소(예: **enx78e7d1ea46da**)를 포함하는 이름은 기본적으로 사용되지 않지만 사용자가 선택하는 경우 사용할 수 있습니다.*

- *스키마 **5:** 다른 모든 메서드가 실패하는 경우 기존의 예측 불가능한 커널 명명 스키마가 사용됩니다(예: **enp1s0**).*

*위에 설명된 절차인 이 정책은 기본값입니다. 시스템에 **biosdevname** 이 활성화된 경우 사용됩니다. **biosdevname** 을 활성화하려면 **Dell** 시스템의 경우와 달리 **biosdevname=1** 을 커널 명령줄 매개 변수로 전달해야 합니다. 여기서 **biosdevname** 은 기본적으로 설치되는 한 사용됩니다. 사용자가 커널 장치 이름을 변경하는 **udev** 규칙을 추가한 경우 해당 규칙이 우선합니다.*

11.2. 장치 이름 변경 절차 이해

장치 이름 절차는 다음과 같습니다.

1. */usr/lib/udev/rules.d/60-net.rules 의 규칙은 udev 도우미 유틸리티 /lib/udev/rename_device 를 지시하여 모든 /etc/sysconfig/network-scripts/ifcfg-suffix 파일을 찾습니다. 인터페이스의 MAC 주소와 일치하는 HWADDR 항목이 있는 ifcfg 파일을 찾으면 DEVICE 지시문을 통해 ifcfg 파일에 지정된 이름으로 인터페이스 이름을 바꿉니다.*

2. */usr/lib/udev/rules.d/71-biosdevname.rules 의 규칙은 이름 지정 정책에 따라 인터페이스 이름을 바꾸도록 biosdev name에 지시합니다. biosdevname 이 이전 단계에서 이름이 변경되지 않은 경우 biosdevname=0 은 부팅 명령줄에 커널 명령으로 제공되지 않았습니다.*

3. */lib/udev/rules.d/75-net-description.rules 는 udev 에 네트워크 인터페이스 장치를 검사하여 내부 udev 장치 속성 값 ID_NET_NAME_ONBOARD, ID_NET_NAME_SLOT, ID_NET_NAME_PATH, ID_NET_NAME_NAME_MAC를 입력합니다. 일부 장치 속성은 정의되지 않았을 수 있습니다.*

4.

/usr/lib/udev/rules.d/80-net-name-slot.rules 의 규칙은 udev 에 인터페이스 이름을 바꾸도록 지시합니다. 이 규칙은 1단계 또는 2단계에서 이름이 변경되지 않은 경우 커널 매개 변수 net.ifnames=0 은 다음 우선 순위에 따라 지정되지 않았습니다. ID_NET_NAME_ONBOARD, ID_NET_NAME_SLOT, ID_NET_NAME_PATH. 값이 설정되지 않은 경우 목록의 다음 항목으로 이동합니다. 이 중 아무것도 설정되지 않은 경우 인터페이스의 이름은 변경되지 않습니다.

3단계와 4단계는 11.1절. “Schemes 계층 이름 지정” 에 설명된 명명 체계 1, 2, 3 및 4를 구현하고 있습니다. 2단계에 대한 자세한 내용은 11.6절. “biosdevname을 사용하여 일관된 네트워크 장치 명명” 에서 참조하십시오.

11.3. PREDICTABLE NETWORK INTERFACE 장치 이름 이해

이름에는 인터페이스 유형에 따라 2자 접두사가 있습니다.

1. 이더넷 의 경우 EN
2. 무선 LAN(WLAN) 을 위한 WL
3. 무선 광역 네트워크(WWAN) 용 WW.

이름에는 다음과 같은 유형이 있습니다.

O<index>

온보드 장치 인덱스 번호

s<slot>[f<function>][d<dev_id>]

핫플러그 슬롯 인덱스 번호. 모든 다기능 PCI 장치는 기능 0 장치를 포함하여 장치 이름에 [f<함수>] 번호를 전달합니다.

x<MAC>

MAC 주소

[P<domain>]p<bus>s<slot>[f<function>][d<dev_id>]

PCI 지리적 위치. PCI 지리적 위치에서 [P<domain>] 번호는 값이 0 이 아닌 경우에만 언급됩니다. 예를 들어 다음과 같습니다.

ID_NET_NAME_PATH=P1enp5s0

[P<domain>]p<bus>s<slot>[f<function>][u<port>][..][c<config>][i<interface>]

USB 포트 번호 체인. USB 장치의 경우 허브의 포트 번호 전체 체인이 구성됩니다. 이름이 최대 15자보다 길면 이름을 내보내지 않습니다. 체인에 여러 개의 USB 장치가 있는 경우 USB 구성 설명자 (c1) 및 USB 인터페이스 설명자 (i0) 의 기본값이 비활성화됩니다.

11.4. SYSTEM Z에서 LINUX에 사용 가능한 네트워크 장치용 이름 지정 스키마

bus-ID를 사용하여 **System z** 인스턴스의 **Linux**에서 네트워크 인터페이스에 대한 예측 가능한 장치 이름을 만듭니다. **bus-ID**는 **s390** 채널 하위 시스템에서 장치를 식별합니다. 버스 **ID**는 **Linux** 인스턴스의 범위 내에서 장치를 식별합니다. **CCW** 장치의 경우 버스 **ID**는 선행 **0.n**이 있는 장치의 장치 번호입니다. 여기서 **n** 은 하위 채널 세트 **ID**입니다. 예를 들면 **0.1.0ab1** 입니다.

장치 유형 이더넷의 네트워크 인터페이스 이름은 다음과 같습니다.

```
enccw0.0.1234
```

장치 유형 **SLIP**의 **CTC** 네트워크 장치의 이름은 다음과 같습니다.

```
slccw0.0.1234
```

znetconf -c 명령 또는 lscss -a 명령을 사용하여 사용 가능한 네트워크 장치 및 해당 버스 ID를 표시합니다.

표 11.1. System z에서 Linux 용 장치 이름 유형

포맷	Description
enccw*bus-ID*	장치 유형 이더넷
slccw*bus-ID*	장치 유형 SLIP의 CTC 네트워크 장치

11.5. VLAN 인터페이스 이름 지정

전통적으로 형식의 VLAN 인터페이스 이름: interface-name.VLAN-ID 가 사용됩니다. VLAN-ID 범위는 0 에서 4096 이며, 최대 4자이며 전체 인터페이스 이름은 15자로 제한됩니다. 최대 인터페이스 이름 길이는 커널 헤더에 의해 정의되며 모든 애플리케이션에 영향을 주는 전역 제한입니다.

Red Hat Enterprise Linux 7에서는 VLAN 인터페이스 이름에 대한 네 가지 명명 규칙이 지원됩니다.

VLAN 및 VLAN ID

vlan 및 VLAN ID를 더한 단어. 예: vlan0005

추가하지 않고 VLAN 및 VLAN ID

추가 선행 0을 사용하여 패딩 없이 vlan 및 VLAN ID를 추가합니다. 예: vlan5

장치 이름 및 VLAN ID

상위 인터페이스의 이름과 VLAN ID입니다. 예: enp1s0.0005

패딩 없이 장치 이름 및 VLAN ID

상위 인터페이스의 이름과 선행 0을 추가하여 패딩하지 않고 VLAN ID를 추가합니다. 예: enp1s0.5

11.6. BIOSDEVNAME을 사용하여 일관된 네트워크 장치 명명

biosdevname udev 도우미 유틸리티를 통해 구현된 이 기능은 기존의 모든 임베디드 네트워크 인터

페이스, PCI 카드 네트워크 인터페이스 및 가상 기능 네트워크 인터페이스의 이름을 변경합니다. eth[0123...] 표 11.2. "biosdevname 명명 규칙" 에 표시된 대로 새로운 명명 규칙에 대해 설명합니다. 시스템이 Dell 시스템이거나 11.6.2절. "기능 활성화 및 비활성화" 설명된 대로 biosdevname 이 명시적으로 활성화되어 있지 않으면 systemd 명명 스키마가 우선합니다.

표 11.2. biosdevname 명명 규칙

장치	이전 이름	새 이름
임베디드 네트워크 인터페이스 (LOM)	**eth[0123...]**	**em[1234...]**[a]
PCI 카드 네트워크 인터페이스	**eth[0123...]**	**p<*slot*>p<*이더넷 포트*>**[b]
가상 기능	**eth[0123...]**	**p<*slot*>p<*이더넷 포트*>_<*가상 인터페이스*>**[c]

[a] 새 열거는 **1** 에서 시작됩니다.

[b] 예를 들어 다음과 같습니다. **p3p4**

[c] 예를 들어 다음과 같습니다. **p3p4_1**

11.6.1. 시스템 요구 사항

biosdevname 프로그램은 시스템 BIOS의 정보, 특히 9(시스템 슬롯) 유형 및 SMBIOS 내에 포함된 41 (온보드 장치 확장 정보) 필드를 사용합니다. 시스템의 BIOS에 SMBIOS 버전 2.6 이상이 있고 이 데이터가 없으면 새로운 명명 규칙이 사용되지 않습니다. 대부분의 오래된 하드웨어는 올바른 SMBIOS 버전 및 필드 정보가 있는 BIOS가 부족하기 때문에 이 기능을 지원하지 않습니다. BIOS 또는 SMBIOS 버전 정보는 하드웨어 벤더에 문의하십시오.

이 기능을 적용하려면 biosdevname 패키지도 설치해야 합니다. 설치하려면 root 로 다음 명령을 실행합니다.

```
~]# yum install biosdevname
```

11.6.2. 기능 활성화 및 비활성화

이 기능을 비활성화하려면 설치 중 및 설치 후 모두 부팅 명령줄에 다음 옵션을 전달합니다.

```
biosdevname=0
```

이 기능을 활성화하려면 설치 중 및 설치 후 모두 부팅 명령줄에 다음 옵션을 전달합니다.

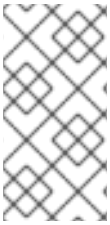
```
biosdevname=1
```

시스템이 최소 요구 사항을 충족하지 않는 한 이 옵션은 무시되고 시스템이 장의 시작 부분에 설명된 대로 **systemd** 명명 체계를 사용합니다.

biosdevname 설치 옵션을 지정하면 시스템 수명 동안 부팅 옵션으로 유지되어야 합니다.

11.7. 관리자의 참고

많은 시스템 사용자 지정 파일에는 네트워크 인터페이스 이름이 포함될 수 있으므로 시스템을 이전 규칙에서 새 규칙으로 이동하는 경우 업데이트가 필요합니다. 새로운 명명 규칙을 사용하는 경우 사용자 지정 **iptables** 규칙, **irqbalance** 스크립트 및 기타 유사한 구성 파일과 같은 영역에서 네트워크 인터페이스 이름도 업데이트해야 합니다. 또한 설치에 대해 이 변경을 활성화하려면 **ksdevice** 매개 변수를 통해 장치 이름을 사용하는 기존 **kickstart** 파일을 수정해야 합니다. 네트워크 장치의 **MAC** 주소 또는 네트워크 장치의 새 이름을 사용하려면 이러한 킥스타트 파일을 업데이트해야 합니다.



참고

최대 인터페이스 이름 길이는 커널 헤더에 의해 정의되며 모든 애플리케이션에 영향을 주는 전역 제한입니다.

11.8. 네트워크 장치 이름 선택 제어

장치 이름 지정은 다음과 같은 방식으로 제어할 수 있습니다.

네트워크 인터페이스 장치 식별

HWADDR 지시문을 사용하여 **ifcfg** 파일에서 **MAC** 주소를 설정하면 **udev** 로 식별할 수 있습니다. 이름은 **DEVICE** 지시문에서 지정한 문자열에서 가져오며, 규칙은 **ifcfg** 접미사와 같습니다. 예를 들어 **ifcfg-enp1s0** 입니다.

biosdevname을 켜거나 끄면

biosdevname 에서 제공하는 이름이 사용됩니다(**biosdevname** 이 **1**을 결정할 수 있는 경우).

systemd-udev 명명 스키마를 켜거나 끄면

systemd-udev 에서 제공하는 이름이 사용됩니다(dsd -udev 에서 하나만 결정할 수 있는 경우).

11.9. 일관된 네트워크 장치 명명 비활성화

일관된 네트워크 장치 이름 지정을 비활성화하려면 특수 시나리오에만 권장됩니다. 자세한 내용은 11 장. 일관된 네트워크 장치 이름 지정 및 11.10절. "네트워크 장치 이름 지정 문제 해결" 을 참조하십시오.

일관된 네트워크 장치 이름 지정을 비활성화하려면 다음 중 하나에서 선택하십시오.

- *기본 정책에 대한 udev 규칙 파일을 "마스크링"하여 고정 이름 할당을 비활성화합니다. 이 작업은 /dev/null 에 대한 심볼릭 링크를 만들어 수행할 수 있습니다. 따라서 예측할 수 없는 커널 이름이 사용됩니다. root로 다음 명령을 입력합니다.*

```
~]# ln -s /dev/null /etc/udev/rules.d/80-net-name-slot.rules
```

- *예를 들어 internet0,dmz0 또는 lan0 인터페이스 이름을 지정하여 고유한 수동 명명 체계를 만듭니다. 이렇게 하려면 고유한 udev 규칙 파일을 만들고 장치에 대한 NAME 속성을 설정합니다. 예를 들어 /etc/udev/rules.d/70-my-net-names.rules에 이름을 지정하여 기본 정책 파일 위에 새 파일을 주문해야 합니다.*

- *기본 정책 파일을 변경하여 다른 명명 체계를 선택합니다. 예를 들어 기본적으로 MAC 주소 뒤에 모든 인터페이스의 이름을 지정합니다. root로 다음과 같이 기본 정책 파일을 복사합니다.*

```
~]# cp /usr/lib/udev/rules.d/80-net-name-slot.rules /etc/udev/rules.d/80-net-name-slot.rules
```

/etc/udev/rules.d/ 디렉터리에서 파일을 편집하고 필요에 따라 행을 변경합니다.

- */etc/default/grub 파일을 열고 GRUB_CMDLINE_LINUX 변수를 찾습니다.*

참고

GRUB_CMDLINE_LINUX 는 커널 명령줄에 추가된 항목이 포함된 변수입니다. 시스템 설정에 따라 이미 추가 구성이 포함되어 있을 수 있습니다.

net.ifnames=0 및 **biosdevname=0** 을 커널 매개변수 값으로 **GRUB_CMDLINE_LINUX** 변수에 추가합니다.

```
~]# cat /etc/default/grub
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="rd.lvm.lv=rhel_7/swap rd.luks.uuid=luks-cc387312-6da6-469a-8e49-b40cd58ad67a crashkernel=auto  vconsole.keymap=us vconsole.font=latarcyrheb-sun16 rd.lvm.lv=rhel_7/root rhgb quiet net.ifnames=0 biosdevname=0"
GRUB_DISABLE_RECOVERY="true"
```

grub2-mkconfig 명령을 실행하여 **/boot/grub2/grub.cfg** 파일을 다시 빌드합니다.

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

참고

UEFI를 사용하여 부팅된 시스템의 경우:

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

현재 장치 이름을 봅니다. 예를 들면 **eno1:**

```
~]# nmcli connection show
NAME   UUID                                  TYPE           DEVICE
Wired  63cba8b2-60f7-4317-bc80-949e800a23cb  802-3-ethernet  eno1
```

장치 이름을 **enp1s0** 으로 수정하고 시스템을 재부팅합니다.

```
~]# nmcli connection modify Wired connection.interface-name enp1s0
```

```
~]# reboot
```

grubby 유틸리티는 **grub** 부트 로더의 구성 파일을 업데이트하고 표시하는 데 사용됩니다. 자세한 내용은 **grubby(8)** 도움말 페이지를 참조하십시오. **GRUB 2** 사용에 대한 자세한 내용은 **Red Hat Enterprise Linux** 시스템 관리자 가이드를 참조하십시오.

11.10. 네트워크 장치 이름 지정 문제 해결

11.2절. "장치 이름 변경 절차 이해" 에 설명된 절차에 따라 각 인터페이스에 대해 예측 가능한 인터페이스 이름이 할당됩니다(해당하는 경우). **udev** 가 사용할 수 있는 이름 목록을 보려면 **root** 로 다음 형식으로 명령을 실행합니다:

```
~]# udevadm info /sys/class/net/ifname | grep ID_NET_NAME
```

여기서 **ifname** 은 다음 명령으로 나열된 인터페이스 중 하나입니다.

```
~]$ ls /sys/class/net/
```

가능한 이름 중 하나는 11.2절. "장치 이름 변경 절차 이해" 에 설명된 규칙에 따라 **udev** 에 의해 적용되며 여기에 요약됩니다.

- **/usr/lib/udev/rules.d/60-net.rules - initscripts**에서
- **/usr/lib/udev/rules.d/71-biosdevname.rules - biosdevname,**

- */usr/lib/udev/rules.d/80-net-name-slot.rules - from systemd*

*위의 규칙 파일 목록에서 인터페이스 이름 지정을 **initscripts** 또는 **biosdevname** 을 통해 수행하면 항상 **udev** 네이티브 명명보다 우선합니다. 그러나 **initscripts** 재배치가 수행되지 않고 **biosdevname** 이 비활성화되면, **80-net-name-slot.rules** 를 **/usr**에서 **/ etc** 로 복사하고 파일을 적절하게 편집합니다. 즉, 특정 순서로 사용할 체계를 주석 처리하거나 정렬해야 합니다.*

> *예 11.1. 일부 인터페이스에는 커널 네임 스페이스 (eth[0,1,2...])의 이름이 있지만 다른 인터페이스는 **udev**에 의해 성공적으로 이름이 지정됩니다.*
>
> *혼합된 스키마는 일부 하드웨어의 경우 **udev**에 커널에서 제공하는 사용 가능한 정보가 없으므로 어떠한 이름이나 **udev** 에 제공된 정보가 적합하지 않음(예: 비회용 장치 **ID**) 입니다. 후자의 경우 더 일반적이며 해결 방법은 **ifcfg** 파일에서 사용자 지정 명명 체계를 사용하거나 **80-net-name-slot.rules**를 편집하여 사용 중인 **udev** 체계를 변경하는 것입니다.*

> *예 11.2. **/var/log/messages** 또는 **systemd** 저널에서 각 인터페이스에 대해 재지정을 완료할 수 있습니다.*
>
> ***ifcfg** 파일에 인코딩된 명명 스키마가 있지만 다시 생성된 **initrd** 이미지가 없는 시스템은 이 문제가 발생할 수 있습니다. 인터페이스 이름은 **initrd** 에 있는 동안 초기 부팅 중에 처음에 커널 명령행의 **biosdevname** 또는 **udev** 또는 **dracut** 매개 변수를 통해 할당됩니다. 그런 다음 실제 **rootfs** 로 전환한 후 새 인터페이스 이름이 두 번째로 수행되고 새 인터페이스 이름이 **60-net.rules** 처리로 **udev** 에서 생성된 **/usr/lib/udev/rename_device** 바이너리에 의해 결정됩니다. 이러한 메시지는 무시해도 됩니다.*

> *예 11.3. **ifcfg** 파일에 **ethX** 이름이 있는 **Naming Scheme** 사용 중 작동하지 않음*
>
> ***Red Hat Enterprise Linux**는 매우 구체적인 상황을 제외하고는 **ethX** 명명 규칙을 일관되게 적용하는 방법을 제공하지 않습니다.*
>
> *인터페이스를 특정 이름으로 설정하는 **udev** 규칙은 요청된 이름이 이미 다른 인터페이스에서 사용 중인 경우 실패합니다. 여기에는 **/usr/lib/udev/rules.d/60-net.rules** 파일에서 제공하는 기능이 포함됩니다.*
>
> *커널은 네트워크 장치를 열거할 때 부팅 시 **ethX** 명명 규칙을 사용합니다. **ethX** 이름은 다양한 재부팅 시 일관되지 않으므로 예측할 수 없습니다. 그 결과 **udev** 를 사용하여 인터페이스를 예측 가능한 이름으로 바꾸거나 커널에서 지정한 예측 불가능한 **ethX** 이름을 다시 정렬하려고 합니다.*

다음 시나리오에서 **ethX** 이름을 사용하는 것이 올바르게 작동합니다.

- 시스템에는 하나의 네트워크 인터페이스만 있습니다.
- **Red Hat Enterprise Linux 7** 가상 시스템 게스트의 **virtio NIC**에 사용하는 경우. 가상화 배포 및 관리 가이드의 **KVM** 반가상화(**virtio**) 드라이버 및 네트워크 구성 장을 참조하십시오.

예 11.4. 일관되지 않은 **enpXxX Names**에서 **net.ifnames=0** 결과 설정

systemd 예측 가능한 인터페이스 명명(**net.ifnames**)과 **biosdevname** 명명 스키마가 모두 비활성화되면 네트워크 인터페이스는 원래 커널에서 제공한 예측 가능하고 잠재적으로 일치하지 않는 **ethX** 이름을 계속 사용합니다.

커널은 네트워크 장치를 열거할 때 부팅 시 항상 **enpXxX** 명명 규칙을 사용합니다. 병렬화로 인해 커널 인터페이스의 순서는 재부팅 시마다 다를 것으로 예상됩니다. **Red Hat Enterprise Linux**는 **systemd** 예측 가능한 인터페이스 명명 체계 또는 **biosdevname** 명명 체계를 사용하여 커널 예측 불가 **ethX** 인터페이스의 이름을 재부팅할 때마다 항상 일관되게 유지되는 이름으로 변경할 수 있습니다.

네트워크 어댑터 명명 규칙에 대한 자세한 내용은 **RHEL7**에서 **net.ifnames=0**을 설정하는 것이 안전합니까? **Red Hat** 고객 포털의 지식 센터 지원 문서.

예 11.5. 이더넷 인터페이스의 접두사에 대한 제한 사항

선택하는 접두사는 다음 요구 사항을 충족해야 합니다.

- **ASCII** 문자로 구성됩니다.
- 영숫자 문자열입니다.
- 이 문자는 **16**자보다 짧습니다.

- eth,eno,ens, em 과 같은 네트워크 인터페이스 이름 지정에 사용되는 잘 알려진 다른 접두사와 충돌하지 않습니다.

11.11. 추가 리소스

설치된 문서

- udev(7) 도움말 페이지 - Linux 동적 장치 관리 데몬, udevd 를 설명합니다.
- systemd(1) 도움말 페이지 - systemd 시스템 및 서비스 관리자를 설명합니다.
- biosdevname(1) 도움말 페이지 - 장치의 BIOS-given 이름을 가져오는 유틸리티를 설명합니다.

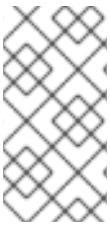
온라인 문서

- IBM Knowledge Center Publication SC34-2710-00 장치 드라이버, 기능 및 명령에 IBM System z 장치 및 연결의 “예측 가능 네트워크 장치 이름에” 대한 정보가 포함되어 있습니다.

12장. 대체 경로를 정의하도록 정책 기반 라우팅 구성

기본적으로 **RHEL**의 커널은 라우팅 테이블을 사용하여 대상 주소를 기반으로 네트워크 패킷을 전달할 위치를 결정합니다. 정책 기반 라우팅을 사용하면 복잡한 라우팅 시나리오를 구성할 수 있습니다. 예를 들어 소스 주소, 패킷 메타데이터 또는 프로토콜과 같은 다양한 기준에 따라 패킷을 라우팅할 수 있습니다.

이 섹션에서는 **NetworkManager**를 사용하여 정책 기반 라우팅을 구성하는 방법을 설명합니다.



참고

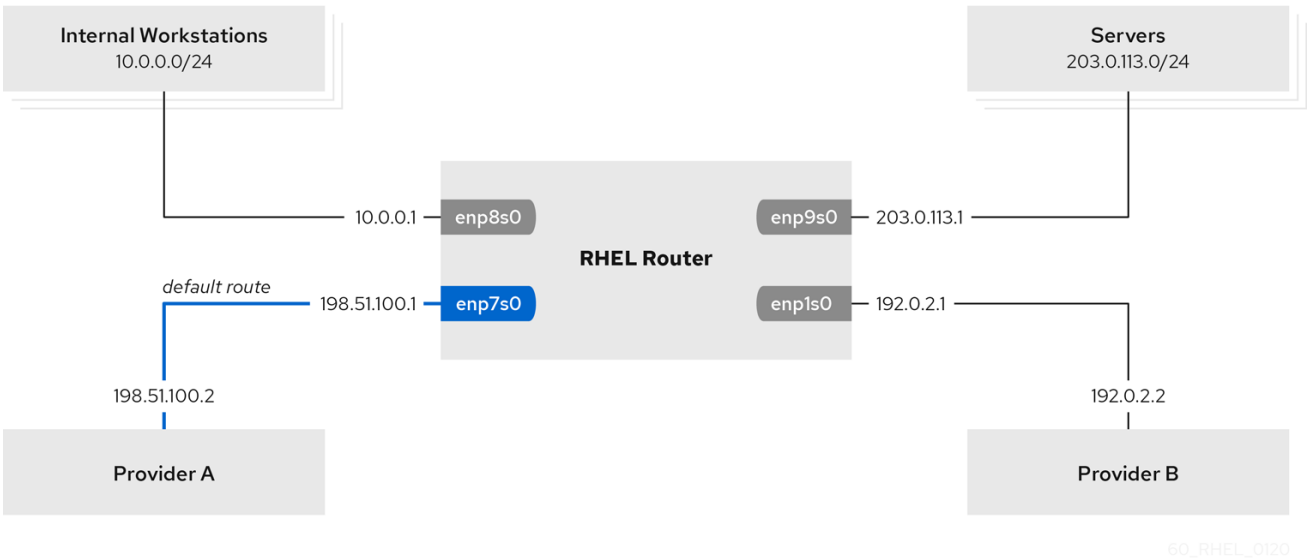
NetworkManager를 사용하는 시스템에서는 **nmcli** 유틸리티만 라우팅 규칙을 설정하고 특정 테이블에 경로를 할당하는 기능을 지원합니다.

12.1. 특정 서브넷에서 다른 기본 게이트웨이로 트래픽 라우팅

이 섹션에서는 기본 경로를 사용하여 모든 트래픽을 인터넷 공급자 **A**로 라우팅하는 라우터로 **RHEL**을 구성하는 방법을 설명합니다. 정책 기반 라우팅을 사용하여 **RHEL**은 내부 워크스테이션 서브넷에서 공급자 **B**로 수신되는 트래픽을 라우팅합니다.

이 절차에서는 다음과 같은 네트워크 토폴로지를 가정합니다.

그림 12.1. 연결 활성화



사전 요구 사항

- 절차에서 설정하려는 **RHEL** 라우터에는 네 개의 네트워크 인터페이스가 있습니다.

- *enp7s0 인터페이스는 프로바이더 A의 네트워크에 연결됩니다. 프로바이더 네트워크의 게이트웨이 IP는 198.51.100.2 이고 네트워크는 /30 네트워크 마스크를 사용합니다.*
- *enp1s0 인터페이스는 프로바이더 B의 네트워크에 연결됩니다. 프로바이더 네트워크의 게이트웨이 IP는 192.0.2.2 이고 네트워크는 /30 네트워크 마스크를 사용합니다.*
- *enp8s0 인터페이스는 내부 워크스테이션을 사용하여 10.0.0.0/24 서브넷에 연결됩니다.*
- *enp9s0 인터페이스는 회사 서버가 있는 203.0.113.0/24 서브넷에 연결됩니다.*

- *내부 워크스테이션 서브넷의 호스트는 기본 게이트웨이로 10.0.0.1 을 사용합니다. 이 절차에서는 이 IP 주소를 라우터의 enp8s0 네트워크 인터페이스에 할당합니다.*
- *서버 서브넷의 호스트는 기본 게이트웨이로 203.0.113.1 을 사용합니다. 이 절차에서는 이 IP 주소를 라우터의 enp9s0 네트워크 인터페이스에 할당합니다.*
- *firewalld 서비스는 활성화되어 활성 상태이며 기본값입니다.*

절차

1. *공급자 A에 네트워크 인터페이스를 구성합니다.*

```
# nmcli connection add type ethernet con-name Provider-A ifname enp7s0 ipv4.method
manual ipv4.addresses 198.51.100.1/30 ipv4.gateway 198.51.100.2 ipv4.dns 198.51.100.200
connection.zone external
```

nmcli connection add 명령은 NetworkManager 연결 프로필을 생성합니다. 다음 목록에서는 명령의 옵션을 설명합니다.

- *ethernet 유형: 연결 유형이 이더넷임을 정의합니다.*
- *con-name connection_name: 프로필의 이름을 설정합니다. 의미 있는 이름을 사용하여 혼동을 피하십시오.*

- *ifname network_device: 네트워크 인터페이스를 설정합니다.*
- *ipv4.method manual: 정적 IP 주소를 구성할 수 있습니다.*
- *ipv4.addresses IP_address/subnet_mask: IPv4 주소 및 서브넷 마스크를 설정합니다.*
- *ipv4.gateway IP_address: 기본 게이트웨이 주소를 설정합니다.*
- *ipv4.dns IP_of_DNS_server: DNS 서버의 IPv4 주소를 설정합니다.*
- *connection.zone firewalld_zone: 네트워크 인터페이스를 정의된 firewalld 영역에 할당합니다. firewalld 는 자동으로 외부 영역에 할당된 마스커레이딩 인터페이스를 활성화합니다.*

2. *네트워크 인터페이스를 공급자 B로 구성합니다.*

```
# nmcli connection add type ethernet con-name Provider-B ifname enp1s0 ipv4.method
manual ipv4.addresses 192.0.2.1/30 ipv4.routes "0.0.0.0/1 192.0.2.2 table=5000, 128.0.0.0/1
192.0.2.2 table=5000" connection.zone external
```

이 명령은 ipv4. gateway 대신 ipv4. routes 매개 변수를 사용하여 기본 게이트웨이를 설정합니다. 이는 이 연결에 대한 기본 게이트웨이를 기본값과 다른 라우팅 테이블(5000)에 할당하는 데 필요합니다. NetworkManager는 연결이 활성화되면 이 새 라우팅 테이블을 자동으로 생성합니다.

참고

nmcli 유틸리티는 ipv4.gateway 의 기본 게이트웨이로 0.0.0.0/0 사용을 지원하지 않습니다. 이 문제를 해결하기 위해 명령은 0.0.0.0/1 및 128.0.0.0/1 서브넷 모두에 대해 별도의 경로를 생성합니다. 이 서브넷은 전체 IPv4 주소 공간도 포함합니다.

3. 네트워크 인터페이스를 내부 워크스테이션 서브넷에 구성합니다.

```
# nmcli connection add type ethernet con-name Internal-Workstations ifname enp8s0 ipv4.method manual ipv4.addresses 10.0.0.1/24 ipv4.routes "10.0.0.0/24 src=192.0.2.1 table=5000" ipv4.routing-rules "priority 5 from 10.0.0.0/24 table 5000" connection.zone trusted
```

이 명령은 **ipv4.routes** 매개 변수를 사용하여 **ID 5000** 이 있는 라우팅 테이블에 정적 경로를 추가합니다. **10.0.0.0/24** 서브넷의 정적 경로는 로컬 네트워크 인터페이스의 **IP**를 프로바이더 **B(192.0.2.1)**에 다음 홉으로사용합니다.

또한 이 명령은 **ipv4.routing-rules** 매개변수를 사용하여 **10.0.0.0/24** 서브넷의 트래픽을 테이블 **5000** 으로 라우팅하는 우선 순위 **5** 인 라우팅 규칙을 추가합니다. 낮은 값은 우선 순위가 높습니다.

ipv4.routing-rules 매개변수의 구문은 **ip route add** 명령에서와 동일합니다. 단, **ipv4.routing-rules** 는 항상 우선 순위를 지정해야 한다는 점을 제외하고는 다음과 같습니다.

4. 네트워크 인터페이스를 서버 서브넷에 구성합니다.

```
# nmcli connection add type ethernet con-name Servers ifname enp9s0 ipv4.method manual ipv4.addresses 203.0.113.1/24 connection.zone trusted
```

검증 단계

1. 내부 워크스테이션 서브넷의 **RHEL** 호스트에서 다음을 수행합니다.

   1. **traceroute** 패키지를 설치합니다.

   ```
   # yum install traceroute
   ```

   2. **traceroute** 유틸리티를 사용하여 인터넷에 있는 호스트에 대한 경로를 표시합니다.

   ```
   # traceroute redhat.com
   traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
    1  10.0.0.1 (10.0.0.1)     0.337 ms  0.260 ms  0.223 ms
    2  192.0.2.1 (192.0.2.1)   0.884 ms  1.066 ms  1.248 ms
    ...
   ```

*명령의 출력에 라우터가 공급자 **B**의 네트워크인 **192.0.2.1** 에 패킷을 전송한다는 내용이 표시됩니다.*

2. *서버 서브넷의 **RHEL** 호스트에서 다음을 수행합니다.*

   1. ***traceroute** 패키지를 설치합니다.*

      ```
      # yum install traceroute
      ```

   2. ***traceroute** 유틸리티를 사용하여 인터넷에 있는 호스트에 대한 경로를 표시합니다.*

      ```
      # traceroute redhat.com
      traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
       1  203.0.113.1 (203.0.113.1)    2.179 ms  2.073 ms  1.944 ms
       2  198.51.100.2 (198.51.100.2)  1.868 ms  1.798 ms  1.549 ms
       ...
      ```

      *명령 출력에 라우터가 공급자 **A**의 네트워크인 **198.51.100.2** 를 통해 패킷을 전송한다는 내용이 표시됩니다.*

문제 해결 단계

***RHEL** 라우터에서 다음을 수행합니다.*

1. *규칙 목록을 표시합니다.*

   ```
   # ip rule list
   0: from all lookup local
   5: from 10.0.0.0/24 lookup 5000
   32766: from all lookup main
   32767: from all lookup default
   ```

2. *표 **5000** 의 경로를 표시합니다.*

   ```
   # ip route list table 5000
   0.0.0.0/1 via 192.0.2.2 dev enp1s0 proto static metric 100
   10.0.0.0/24 dev enp8s0 proto static scope link src 192.0.2.1 metric 102
   128.0.0.0/1 via 192.0.2.2 dev enp1s0 proto static metric 100
   ```

3. 방화벽 영역에 할당되는 인터페이스를 표시합니다.

```
# firewall-cmd --get-active-zones
external
  interfaces: enp1s0 enp7s0
trusted
  interfaces: enp8s0 enp9s0
```

4. 외부 영역에 마스커레이딩이 활성화되었는지 확인합니다.

```
# firewall-cmd --info-zone=external
external (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0 enp7s0
  sources:
  services: ssh
  ports:
  protocols:
  masquerade: yes
  ...
```

추가 리소스

- **ipv4.*** 매개변수에 대한 자세한 내용은 **nmcli connection add** 명령에 설정하면 **nm-settings(5)** 도움말 페이지의 **IPv4** 설정 섹션을 참조하십시오.
- 연결에 대한 자세한 내용은 **nmcli connection add** 명령에 설정하면 **nm-settings(5)** 도움말 페이지의 연결 설정 섹션을 참조하십시오.
- **nmcli** 를 사용하여 **NetworkManager** 연결 관리에 대한 자세한 내용은 **nmcli(1)** 도움말 페이지의 연결 관리 명령 섹션을 참조하십시오.

III 부. INFINIBAND 및 RDMA 네트워킹

이 부분에서는 InfiniBand 네트워크 연결을 통해 RDMA, InfiniBand 및 IP를 설정하는 방법에 대해 설명합니다.

13장. INFINIBAND 및 RDMA 네트워크 구성

13.1. INFINIBAND 및 RDMA 기술 이해

InfiniBand는 다음과 같은 두 가지 요소를 나타냅니다. 첫 번째는 InfiniBand 네트워크의 물리적 링크 계층 프로토콜입니다. 두 번째는 InfiniBand Verbs API라는 상위 수준 프로그래밍 API입니다. InfiniBand Verbs API는RDMA(Remote Direct Memory Access) 기술의 구현입니다.

RDMA는 컴퓨터의 운영 체제를 포함하지 않고 한 컴퓨터의 메모리에서 다른 컴퓨터의 메모리로 직접 액세스할 수 있도록 합니다. 이 기술을 사용하면 CPU 사용률이 짧은 처리량이 높고 대기 시간이 짧은 네트워킹을 사용할 수 있습니다. 특히 대규모 병렬 컴퓨터 클러스터에서 유용합니다.

일반적인 IP 데이터 전송에서 시스템 A의 애플리케이션 X는 일부 데이터를 시스템 B의 애플리케이션 Y로 보냅니다. 전송의 일환으로, 시스템 B의 커널은 먼저 데이터를 수신하고 패킷 헤더를 디코딩하고, 애플리케이션 Y에 속하는지 확인하고, 애플리케이션 Y가 읽기 syscall을 커널에 수행할 때까지 기다린 다음, 커널의 자체 내부 메모리 공간의 데이터를 애플리케이션 Y가 제공하는 버퍼로 수동으로 복사해야 합니다. 이 프로세스는 대부분의 네트워크 트래픽이 최소 두 번(호스트 어댑터에서 DMA를 사용하여 데이터를 커널 제공 메모리 버퍼에 넣은 경우)을 통해 복사해야 하며, 커널이 데이터를 애플리케이션의 메모리 버퍼로 이동할 때에도 컴퓨터가 커널 컨텍스트와 애플리케이션 Y 컨텍스트 간에 전환하기 위해 다수의 컨텍스트 스위치를 실행해야 함을 의미합니다. 이러한 두 가지 모두 네트워크 트래픽이 매우 빠른 속도로 이동하고 다른 작업을 속도 저하시킬 수 있는 경우 시스템에 매우 높은 CPU 부하를 부과합니다.

RDMA 통신은 통신 프로세스의 커널 개입을 우회하기 때문에 일반 IP 통신과는 다르며, 프로세스에서는 네트워크 통신을 처리하는 데 일반적으로 필요한 CPU 오버헤드를 크게 줄입니다. RDMA 프로토콜을 통해 시스템의 호스트 어댑터는 패킷이 네트워크에서 들어오는 시기, 해당 패킷을 수신해야 하는 애플리케이션 및 애플리케이션의 메모리 공간에 들어야 하는 위치를 알 수 있습니다. 패킷을 커널로 전송하여 사용자 애플리케이션 메모리에 복사한 다음 추가 조작 없이도 패킷의 콘텐츠를 애플리케이션 버퍼에 직접 배치합니다. 그러나 대부분의 IP 네트워킹 애플리케이션이 빌드된 표준 Berkeley Sockets API를 사용하여 수행할 수 없으므로 RDMA 기술을 직접 사용할 수 있으려면 자체 API, InfiniBand Verbs API 및 애플리케이션을 이 API에 이식해야 합니다.

Red Hat Enterprise Linux 7은 InfiniBand 하드웨어와 InfiniBand Verbs API를 모두 지원합니다. 또한 InfiniBand Verbs API를 비InfiniBand 하드웨어에서 활용할 수 있는 두 가지 추가 지원 기술이 있습니다.

- iWARP(Internet Wide Area RDMA Protocol)

  iWARP는 IP(인터넷 프로토콜) 네트워크를 통한 효율적인 데이터 전송을 위해 RDMA(Remote Direct Memory Access)를 구현하는 컴퓨터 네트워킹 프로토콜입니다.

-

RoCE(RDMA over Converged Ethernet) 프로토콜은 나중에 IBoE(InfiniBand over Ethernet)로 이름이 변경되었습니다.

RoCE는 이더넷 네트워크를 통해 RDMA(Remote Direct Memory Access)를 허용하는 네트워크 프로토콜입니다.

사전 요구 사항

iWARP 및 RoCE 기술은 모두 기본 기술로 일반 IP 네트워크 링크 계층을 있으므로 대부분의 구성은 실제로 3장. IP 네트워킹 구성 다룹니다. 대부분의 경우 IP 네트워킹 기능이 올바르게 구성되면 RDMA 기능이 모두 자동이며 하드웨어에 적합한 드라이버가 설치되는 한 표시됩니다. 커널 드라이버는 각 커널에 항상 포함되어 있지만 시스템 설치 시 InfiniBand 패키지 그룹을 선택하지 않은 경우 사용자 공간 드라이버를 수동으로 설치해야 합니다.

Red Hat Enterprise Linux 7.4부터 모든 RDMA 사용자 공간 드라이버가 rdma-core 패키지에 병합됩니다. 지원되는 모든 iWARP, RoCE 또는 InfiniBand 사용자 공간 드라이버를 설치하려면 root 로 다음을 입력하십시오.

```
~]# yum install libibverbs
```

PFC(우선 순위 흐름 제어) 및 mlx4 기반 카드를 사용하는 경우 /etc/modprobe.d/mlx4.conf 를 편집하여 이더넷 스위치의 "no-drop" 서비스에 대해 패킷 우선 순위가 구성된 드라이버에 지시하여 카드가 연결되고 initramfs 를 다시 빌드하여 수정된 파일을 포함합니다. 최신 mlx5 기반 카드는 스위치로 PFC 설정을 자동 협상하며 "드롭" 되지 않은 우선 순위 또는 우선 순위를 알려주는 모듈 옵션이 필요하지 않습니다.

이더넷 모드에서 하나 또는 두 포트를 사용하도록 Mellanox 카드를 설정하려면 13.5.4절. "이더넷 운영을 위한 Mellanox 카드 구성" 을 참조하십시오.

이러한 드라이버 패키지가 설치된 경우(일반적으로 InfiniBand 설치를 위해 일반 RDMA 패키지 설치 시) 사용자는 대부분의 일반 RDMA 애플리케이션을 활용하여 어댑터에서 발생하는 RDMA 프로토콜 통신을 테스트하고 확인할 수 있어야 합니다. 그러나 Red Hat Enterprise Linux 7에 포함된 일부 프로그램은 iWARP 또는 RoCE/IBoE 장치를 적절하게 지원하지 않습니다. 특히 iWARP의 연결 설정 프로토콜은 실제 InfiniBand 링크 계층 연결과 다르기 때문입니다. 해당 프로그램이 librdmacm 연결 관리 라이브러리를 사용하는 경우 iWARP와 InfiniBand의 차이점을 자동으로 처리하고 프로그램이 작동해야 합니다. 애플리케이션에서 자체 연결 관리를 수행하려고 하면 특별히 iWARP를 지원하거나 작동하지 않는 것이 좋습니다.

13.2. ROCE를 사용하여 데이터 전송

RoCE(RDMA over Converged Ethernet)는 이더넷 네트워크를 통해 RDMA(Remote Direct Memory Access)를 활성화하는 네트워크 프로토콜입니다. RoCE 버전인 RoCE v1 및 RoCE v2에는 사용된 네트워크 어댑터에 따라 두 가지 RoCE 버전이 있습니다.

RoCE v1

RoCE v1 프로토콜은 이더넷 브로드캐스트 도메인에 있는 두 호스트 간의 통신을 활성화하는 이더넷 링크 계층 프로토콜로, 이더넷 0x8915 입니다. RoCE v1은 ConnectX-3 네트워크 어댑터를 사용할 때 RDMA Connection Manager(RDMA_CM)의 기본 버전입니다.

RoCE v2

RoCE v2 프로토콜은 IPv4를 통한 UDP 또는 IPv6 프로토콜을 통해 UDP에 있습니다. UDP 대상 포트 번호 4791 은 RoCE v2용으로 예약되어 있습니다. Red Hat Enterprise Linux 7.5 이후 RoCE v2는 ConnectX-3 Pro, ConnectX-4, ConnectX-4 Lx 및 ConnectX-5 네트워크 어댑터를 사용할 때 RDMA_CM의 기본 버전입니다. 하드웨어는 RoCE v1 및 RoCE v2 를 모두 지원합니다.

RDMA 연결 관리자(RDMA_CM)는 데이터 전송을 위한 클라이언트와 서버 간에 안정적인 연결을 설정하는 데 사용됩니다. RDMA_CM은 연결을 설정하기 위한 RDMA 전송 중립 인터페이스를 제공합니다. 통신은 특정 RDMA 장치를 통해 이루어지며 데이터 전송은 메시지 기반입니다.

사전 요구 사항

RDMA_CM 세션에는 다음 중 하나가 필요합니다.

- 클라이언트와 서버 모두 동일한 RoCE 모드를 지원합니다.
- 클라이언트는 RoCE v1 및 서버 RoCE v2를 지원합니다.

클라이언트가 연결 모드를 결정하므로 다음 사례가 가능합니다.

성공적인 연결:

클라이언트가 RoCE v1 또는 RoCE v2 모드에 있는 경우 사용된 네트워크 카드와 드라이버에 따라 해당 서버에는 연결을 생성하기 위해 동일한 버전이 있어야 합니다. 또한 클라이언트가 RoCE v1 및 RoCE v2 모드에 있는 경우 연결이 성공합니다.

실패한 연결:

*클라이언트가 **RoCE v2**에 있고 해당 서버가 **RoCE v1**인 경우 연결을 설정할 수 없습니다. 이 경우 해당 서버의 드라이버 또는 네트워크 어댑터를 업데이트합니다. 13.2절. "RoCE를 사용하여 데이터 전송"*

*표 **13.1. RDMA_CM**을 사용한 **RoCE** 버전 기본값*

클라이언트	서버	기본 설정
RoCE v1	RoCE v1	연결
RoCE v1	RoCE v2	연결
RoCE v2	RoCE v2	연결
RoCE v2	RoCE v1	연결 없음

*서버의 클라이언트 및 **RoCE v1**의 **RoCE v2**는 호환되지 않습니다. 이 문제를 해결하려면 서버 및 클라이언트 측 환경이 **RoCE v1**을 통해 통신하도록 강제 시행합니다. 즉, **RoCE v2**를 지원하는 하드웨어가 **RoCE v1**을 사용하도록 강제 시행됩니다.*

*절차 **13.1. Roce v2**에서 실행 중인 하드웨어가 이미 실행 중인 경우 기본 **RoCE** 모드 변경*

1. *RoCE 모드를 et하도록 /sys/kernel/config/rdma_cm 디렉터리로 변경합니다.*

```
~]# cd /sys/kernel/config/rdma_cm
```

2. *상태를 표시할 이더넷 네트워크 장치에 ibstat 명령을 입력합니다. 예를 들어 mlx5_0:*

```
~]$ ibstat mlx5_0
  CA 'mlx5_0'
      CA type: MT4115
      Number of ports: 1
      Firmware version: 12.17.1010
      Hardware version: 0
      Node GUID: 0x248a0703004bf0a4
      System image GUID: 0x248a0703004bf0a4
      Port 1:
          State: Active
          Physical state: LinkUp
          Rate: 40
          Base lid: 0
          LMC: 0
          SM lid: 0
```

```
        Capability mask: 0x04010000
        Port GUID: 0x268a07fffe4bf0a4
        Link layer: Ethernet
```

3. **mlx5_0 장치의 디렉터리를 생성합니다.**

```
~]# mkdir mlx5_0
```

4. **default_roce_mode 파일에 RoCE 모드를 트리 형식으로 표시합니다.**

```
~]# cd mlx5_0
```

```
~]$ tree
└── ports
    └── 1
        ├── default_roce_mode
        └── default_roce_tos
```

```
~]$ cat /sys/kernel/config/rdma_cm/mlx5_0/ports/1/default_roce_mode
  RoCE v2
```

5. **기본 RoCE 모드를 변경합니다.**

```
~]# echo "RoCE v1" > /sys/kernel/config/rdma_cm/mlx5_0/ports/1/default_roce_mode
```

6.

변경 사항을 확인합니다.

```
~]$ cat /sys/kernel/config/rdma_cm/mlx5_0/ports/1/default_roce_mode
  RoCE v1
```

13.3. SOFT-ROCE 구성

RoCE는 하드웨어와 소프트웨어에 모두 구현할 수 있습니다. soft-RoCE는 RDMA 전송의 소프트웨어 구현입니다.

사전 요구 사항

Red Hat Enterprise Linux 7.4부터 soft-RoCE 드라이버가 커널에 이미 병합되어 있습니다. user-space 드라이버도 rdma-core 패키지에 병합됩니다. soft-RoCE는 RXE라고도 합니다. RXE를 시작, 중지 및 구성하려면 rxe_cfg 스크립트를 사용합니다. rxe_cfg 옵션을 보려면 rxe _cfg help 를 입력합니다.

절차 13.2. soft-RoCE 구성

1.

root 사용자로 다음 명령을 입력하여 RXE의 현재 구성 상태를 표시합니다.

```
~]# rxe_cfg
 rdma_rxe module not loaded
 Name      Link  Driver   Speed   NMTU  IPv4_addr  RDEV  RMTU
 igb_1     yes   igb
 mlx4_1    no    mlx4_en
 mlx4_2    no    mlx4_en
```

2.

RXE 커널 모듈을 로드하고 RXE를 시작하려면 root 로 를 입력합니다.

```
~]# rxe_cfg start
 Name      Link  Driver   Speed   NMTU  IPv4_addr  RDEV  RMTU
 igb_1     yes   igb
 mlx4_1    no    mlx4_en
 mlx4_2    no    mlx4_en
```

선택적으로 RXE 커널 모듈이 로드되었는지 확인하려면 다음을 입력합니다.

```
~]# lsmod |grep rdma_rxe
 rdma_rxe            111129  0
```

```
ip6_udp_tunnel        12755  1 rdma_rxe
udp_tunnel            14423  1 rdma_rxe
ib_core              236827  15
rdma_cm,ib_cm,iw_cm,rpcrdma,mlx4_ib,ib_srp,ib_ucm,ib_iser,ib_srpt,ib_umad,ib_uve
rbs,rdma_rxe,rdma_ucm,ib_ipoib,ib_isert
```

3.

*이더넷 인터페이스를 통해 새 **RXE** 장치를 추가하기 전에 해당 인터페이스를 열어야 하며 유효한 **IP** 주소가 할당됩니다. 새 **RXE** 장치를 추가하려면 (예: igb_1:)*

```
~]# rxe_cfg add igb_1
```

```
~]# rxe_cfg status
Name      Link  Driver   Speed   NMTU  IPv4_addr  RDEV  RMTU
igb_1     yes   igb                    rxe0  1024  (3)
mlx4_1    no    mlx4_en
mlx4_2    no    mlx4_en
```

*RDEV 열의 **rxe0**은 **rxe**가 **igb_1** 장치에 대해 활성화되었음을 나타냅니다.*

4.

*RXE 장치의 상태를 확인하려면 **ibv_devices** 명령을 사용합니다.*

```
~]# ibv_devices
device              node GUID
------          ----------------
mlx4_0              0002c90300b3cff0
rxe0            a2369ffffe018294
```

*또는 자세한 상태를 위해 **ibstat** 를 입력하십시오.*

```
~]# ibstat rxe0
CA 'rxe0'
   CA type:
   Number of ports: 1
   Firmware version:
   Hardware version:
   Node GUID: 0xa2369ffffe018294
   System image GUID: 0x0000000000000000
   Port 1:
      State: Active
      Physical state: LinkUp
      Rate: 2.5
      Base lid: 0
      LMC: 0
```

```
SM lid: 0
Capability mask: 0x00890000
Port GUID: 0xa2369ffffe018294
Link layer: Ethernet
```

RXE 장치 제거

RXE 장치를 제거하려면 다음을 입력합니다.

```
~]# rxe_cfg remove igb_1
```

RXE 장치의 연결 확인

다음 예제에서는 서버 및 클라이언트측에서 RXE 장치의 연결을 확인하는 방법을 보여줍니다.

예 13.1. 서버 사이드에서 RXE 장치의 연결 확인

```
~]$ ibv_rc_pingpong -d rxe0 -g 0
 local address:  LID 0x0000, QPN 0x000012, PSN 0xe2965f, GID fe80::290:faff:fe29:486a
 remote address: LID 0x0000, QPN 0x000011, PSN 0x4bf206, GID fe80::290:faff:fe29:470a
8192000 bytes in 0.05 seconds = 1244.06 Mbit/sec
1000 iters in 0.05 seconds = 52.68 usec/iter
```

예 13.2. 클라이언트 사이드에서 RXE 장치의 연결 확인

```
~]$ ibv_rc_pingpong -d rxe0 -g 0 172.31.40.4
 local address:  LID 0x0000, QPN 0x000011, PSN 0x4bf206, GID fe80::290:faff:fe29:470a
 remote address: LID 0x0000, QPN 0x000012, PSN 0xe2965f, GID fe80::290:faff:fe29:486a
8192000 bytes in 0.05 seconds = 1245.72 Mbit/sec
1000 iters in 0.05 seconds = 52.61 usec/iter
```

13.4. INFINIBAND 및 RDMA 관련 소프트웨어 패키지

RDMA 애플리케이션은 Berkeley Sockets 기반 애플리케이션과 일반 IP 네트워킹과 매우 다르기 때문에 IP 네트워크에서 사용되는 대부분의 애플리케이션은 RDMA 네트워크에서 직접 사용할 수 없습니다. Red Hat Enterprise Linux 7은 RDMA 네트워크 관리, 테스트 및 디버깅, 높은 수준의 소프트웨어 개발 API 및 성능 분석을 위한 다양한 소프트웨어 패키지를 제공합니다.

이러한 네트워크를 활용하려면 이러한 패키지 중 일부 또는 전체를 설치해야 합니다(이 목록은 포괄적이지 않지만 RDMA와 관련된 가장 중요한 패키지를 다룹니다).

필수 패키지:

- RDMA - RDMA 스택의 커널 초기화 담당.
- libibverbs - InfiniBand Verbs API를 제공합니다.
- opensm - 서브넷 관리자(한 시스템에만 필요하며 패브릭에 활성화된 서브넷 관리자가 없는 경우에만).
- 설치된 하드웨어의 사용자 공간 드라이버 - infinipath-psm,libcxgb3,libcxgb4,libehca,libmthca, libmthca,libmlx4,libmlx5,lib ocrdma 중 하나 이상. libehca 는 IBM Power Systems 서버에서만 사용할 수 있습니다.

권장 패키지 :

- librdmacm,librdmacm-utils 및 ibacm - InfiniBand, iWARP 및 RoCE 간의 차이점을 인지하는 연결 관리 라이브러리, 이러한 모든 하드웨어 유형 간의 연결을 올바르게 열고 네트워크 작동을 확인하기 위한 몇 가지 간단한 테스트 프로그램 및 라이브러리와 통합된 캐싱 데몬이 대규모 클러스터에서 원격 호스트 확인을 더 빠르게 만들 수 있도록 합니다.

- *libibverbs-utils - 설치된 하드웨어를 쿼리하고 패브릭에 대한 통신을 확인하기 위한 간단한 Verbs 기반 프로그램.*

- *InfiniBand -diags 및 ibutils - InfiniBand 패브릭 관리를 위한 여러 유용한 디버깅 툴을 제공합니다. 이러한 툴은 대부분 Verbs API 계층이 아닌 InfiniBand 링크 계층에서 작동하므로 iWARP 또는 RoCE에 매우 제한된 기능만 제공합니다.*

- *perftest 및 qperf - 다양한 유형의 RDMA 통신에 대한 성능 테스트 애플리케이션.*

선택적 패키지:

이러한 패키지는 선택적 채널에서 사용할 수 있습니다. 선택적 채널에서 패키지를 설치하기 전에 지원 범위 세부 정보를 참조하십시오. 선택적 채널에 가입하는 방법에 대한 정보는 Red Hat Knowledgebase 솔루션에서 Optional 및 Supplementary 채널에 액세스하는 방법을 참조하십시오.

- *dapl,dapl-devel 및 dapl-utils - Verbs API와 다른 API를 제공합니다. 이러한 패키지에는 런타임 구성 요소와 개발 구성 요소가 있습니다.*

- *openmpi,mvapich2 및 mvapich2-psm - RDMA 통신을 사용할 수 있는 MPI 스택. 이러한 스택에 작성하는 사용자 공간 애플리케이션이 RDMA 통신이 수행되고 있음을 반드시 인식할 필요는 없습니다.*

13.5. 기본 RDMA 하위 시스템 구성

rdma 서비스의 시작은 자동으로 수행됩니다. RDMA 가능 하드웨어(InfiniBand 또는 iWARP 또는 RoCE/IBoE)가 감지되면 udev 에서 systemd 에 rdma 서비스를 시작하도록 지시합니다.

```
~]# systemctl status rdma
● rdma.service - Initialize the iWARP/InfiniBand/RDMA stack in the kernel
   Loaded: loaded (/usr/lib/systemd/system/rdma.service; disabled; vendor preset: disabled)
```

```
Active: inactive (dead)
  Docs: file:/etc/rdma/rdma.conf
```

사용자는 **rdma** 서비스를 활성화 할 필요가 없지만 항상 강제로 실행하려는 경우 사용할 수 있습니다. 이렇게 하려면 **root**로 다음 명령을 입력합니다.

```
~]# systemctl enable rdma
```

13.5.1. rdma.conf 파일 설정

The rdma 서비스는 **/etc/rdma/rdma.conf** 를 읽고 관리자가 기본적으로 로드하려는 커널 수준 및 사용자 수준 **RDMA** 프로토콜을 찾습니다. 사용자는 이 파일을 편집하여 다양한 드라이버를 켜거나 꺼야 합니다.

활성화 및 비활성화할 수 있는 다양한 드라이버는 다음과 같습니다.

- **IPoIB - IP** 애플리케이션을 **InfiniBand** 네트워크를 통해 실행할 수 있는 **IP** 네트워크 에뮬레이션 계층입니다.
- **SRP - SCSI** 요청 프로토콜입니다. 이를 통해 머신은 시스템의 **SRP** 프로토콜을 통해 내보낸 원격 드라이브 또는 드라이브 어레이를 로컬 하드 디스크와 동일하게 마운트할 수 있습니다.
- **SRPT - SRP** 프로토콜의 대상 모드 또는 서버 모드입니다. 이렇게 하면 다른 시스템이 시스템의 로컬 것처럼 마운트할 수 있도록 드라이브 또는 드라이브 어레이를 내보내는 데 필요한 커널 지원이 로드됩니다. 장치를 실제로 내보내기 전에 대상 모드 지원을 추가로 구성해야 합니다. 자세한 내용은 **targetd** 및 **targetcli** 패키지의 설명서를 참조하십시오.
- **ISER - iSCSI** 장치의 **InfiniBand** 네트워크에서 전송을 제공하는 **Linux** 커널의 일반 **iSCSI** 계층의 하위 수준 드라이버입니다.
- **RDS - Linux** 커널에서 신뢰할 수 있는 데이터그램 서비스입니다. **Red Hat Enterprise Linux 7** 커널에서는 이 기능이 활성화되어 있지 않으므로 로드할 수 없습니다.

13.5.2. 70-persistent-ipoib.rules 사용

The rdma 패키지는 /etc/udev.d/rules.d/70-persistent-ipoib.rules 파일을 제공합니다. 이 udev 규칙 파일은 기본 이름(예: ib0 및 ib 1)에서 더설명적인 이름으로 IPoIB 장치의 이름을 변경하는 데 사용됩니다. 사용자는 이 파일을 편집하여 장치 이름을 로 지정하는 방식을 변경해야 합니다. 먼저 이름을 변경할 장치의 GUID 주소를 확인합니다.

```
~]$ ip link show ib0
8: ib0: >BROADCAST,MULTICAST,UP,LOWER_UP< mtu 65520 qdisc pfifo_fast state UP mode
DEFAULT qlen 256
    link/infiniband 80:00:02:00:fe:80:00:00:00:00:00:00:00:f4:52:14:03:00:7b:cb:a1 brd
00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:00:ff:ff:ff:ff
```

링크/infiniband 직후는 IPoIB 인터페이스의 20바이트 하드웨어 주소입니다. 위의 굵은 글꼴로 표시된 주소의 마지막 8바이트는 새 이름을 만드는 데 필요한 모든 바이트입니다. 사용자는 자신에게 맞는 명명 체계를 구성할 수 있습니다. 예를 들어 mlx 4 장치가 ib0 서브넷 패브릭에 연결된 경우 mlx4_ib0 과 같은 device_fabric 명명 규칙을 사용하십시오. 권장되지 않는 유일한 방법은 자동 이름이 할당된 커널과 충돌할 수 있으므로 ib0 또는 ib1 과 같은 표준 이름을 사용하는 것입니다. 다음으로 규칙 파일에 항목을 추가합니다. 규칙 파일에 있는 기존 예제를 복사하고 ATTR{address} 항목의 8바이트를 이름을 변경할 장치에서 강조 표시된 8바이트로 교체한 다음 NAME(이름) 필드에 사용할 새 이름을 입력합니다.

13.5.3. 사용자의 memlock 제한 완화

RDMA 통신에서는 컴퓨터의 실제 메모리를 고정해야 합니다(전체 컴퓨터가 사용 가능한 메모리에서 부족함을 시작하는 경우 커널이 페이징 파일로 해당 메모리를 스왑할 수 없음을 의미). 고정 메모리는 일반적으로 매우 권한 있는 작업입니다. root 이외의 사용자가 큰 RDMA 애플리케이션을 실행할 수 있도록 하려면root 가 아닌 사용자가 시스템에 고정할 수 있는 메모리 양을 늘려야 할 수 있습니다. 이 작업은 다음과 같은 내용이 있는 /etc/security/limits.d/ 디렉터리에 파일을 추가하여 수행됩니다.

```
~]$ more /etc/security/limits.d/rdma.conf
# configuration for rdma tuning
*       soft    memlock         unlimited
*       hard    memlock         unlimited
# rdma tuning end
```

13.5.4. 이더넷 운영을 위한 Mellanox 카드 구성

Mellanox의 특정 하드웨어는 InfiniBand 또는 이더넷 모드에서 실행할 수 있습니다. 이 카드의 기본값은 InfiniBand입니다. 사용자는 카드를 이더넷 모드로 설정할 수 있습니다. 현재 ConnectX 제품군 하드웨어에서만 모드를 설정할 수 있습니다(mlx 5 또는 mlx 4 드라이버를 사용).

Mellanox mlx5 카드를 구성하려면 mst flint 패키지에서 mst config 프로그램을 사용합니다. 자세한 내용은 Red Hat Customer Portal의 Red Hat Enterprise Linux 7 Knowledge Base 문서의 Mellanox mlx5 카드 구성 을 참조하십시오.

Mellanox mlx4 카드를 구성하려면 mstconfig 를 사용하여 기술 자료 문서에 설명된 대로 카드의 포트 유형을 설정합니다. mstconfig 가 카드를 지원하지 않는 경우 /etc/rdma/mlx4.conf 파일을 편집하고 해당 파일의 지침에 따라 RoCE/IBoE 사용에 맞게 포트 유형을 적절하게 설정합니다. 이 경우 업데이트된 포트 설정이 initramfs 에 복사되었는지 확인하려면 initramfs 를 다시 빌드해야 합니다.

포트 유형을 설정한 후 하나 또는 두 포트가 이더넷으로 설정되어 있고 mstconfig 가 포트 유형을 설정하는 데 사용되지 않은 경우 사용자는 로그에서 이 메시지를 표시할 수 있습니다.

```
mlx4_core 0000:05:00.0: Requested port type for port 1 is not supported on this HCA
```

이는 정상이며 운영에 영향을 주지 않습니다. 포트 유형을 설정하는 스크립트에는 드라이버가 내부적으로 포트 2의 전환을 완료한 시점을 알 수 없으며, 스크립트가 해당 스위치가 완료될 때까지 포트 2를 실행할 때 포트 1을 다른 유형으로 설정하려고 하면 거부됩니다. 스크립트는 명령이 성공할 때까지 또는 포트 스위치가 완료되지 않았음을 나타내는 시간 초과가 전달될 때까지 다시 시도합니다.

13.5.5. 원격 Linux SRP 대상에 연결

SCSI RDMA Protocol(SRP)는 시스템이 RDMA를 사용하여 다른 시스템에 연결된 SCSI 장치에 액세스할 수 있는 네트워크 프로토콜입니다. SRP 이니시에이터가 SRP 대상측에서 SRP 대상을 연결할 수 있도록 하려면 이니시에이터에 사용된 호스트 채널 어댑터(HCA) 포트에 대한 ACL(액세스 제어 목록) 항목을 추가해야 합니다.

HCA 포트의 ACL ID는 고유하지 않습니다. ACL ID는 HCA의 GID 형식에 따라 다릅니다. 동일한 드라이버를 사용하는 HCA(예: ib_qib)는 GID 형식이 다를 수 있습니다. 또한 ACL ID는 연결 요청을 시작하는 방법에 따라 달라집니다.

원격 Linux SRP 대상에 연결: 상위 수준 개요

1. *대상 측면을 준비합니다.*

   a. *스토리지 백엔드 생성. 예를 들어 /dev/sdc1 파티션을 가져옵니다.*

   ```
   /> /backstores/block create vol1 /dev/sdc1
   ```

   b. *SRP 대상을 생성합니다.*

   ```
   /> /srpt create 0xfe8000000000000001175000077dd7e
   ```

c.

*단계에서 생성된 백엔드를 기반으로 **LUN**을 생성합니다.*

```
/> /srpt/ib.fe80000000000000001175000077dd7e/luns create /backstores/block/vol1
```

d.

*원격 **SRP** 클라이언트에 대한 노드 **ACL**을 생성합니다.*

```
/> /srpt/ib.fe80000000000000001175000077dd7e/acls create
0x7edd77000075110001175000077d708
```

***Node ACL**은 **srp_daemon** 및 **ibsrp dm** 에 대해 다릅니다.*

2.

*클라이언트 측의 경우 **srp_daemon** 또는 **ibsrpdm** 로 **SRP** 연결을 시작합니다.*

```
[root@initiator]# srp_daemon -e -n -i qib0 -p 1 -R 60 &
```

```
[root@initiator]# ibsrpdm -c -d /dev/infiniband/umad0 > /sys/class/infiniband_srp/srp-
qib0-1/add_target
```

3.

*선택사항입니다. **lsscsi** 또는 **dmesg** 와 같은 다양한 도구를 사용하여 **SRP** 연결을 확인하는 것이 좋습니다.*

*절차 **13.3. srp_daemon** 또는 **ibsrpdm**을 사용하여 원격 **Linux SRP** 대상에 연결*

1.

*대상에 **ibstat** 명령을 사용하여 상태 및 포트 **GUID** 값을 확인합니다. **HCA**는 **Active** 상태여야 합니다. **ACL ID**는 **Port GUID** 를 기반으로 합니다:*

```
[root@target]# ibstat
CA 'qib0'
 CA type: InfiniPath_QLE7342
 Number of ports: 1
 Firmware version:
```

```
Hardware version: 2
Node GUID: 0x001175000077dd7e
System image GUID: 0x001175000077dd7e
Port 1:
 State: Active
 Physical state: LinkUp
 Rate: 40
 Base lid: 1
 LMC: 0
 SM lid: 1
 Capability mask: 0x0769086a
 Port GUID: 0x001175000077dd7e
 Link layer: InfiniBand
```

2. ***HCA 포트 GUID를 기반으로 하는 SRP 대상 ID를 가져옵니다. 전용 디스크 파티션이 SRP 대상의 백엔드로 필요합니다(예: /dev/sdc1). 다음 명령은 기본 접두사 fe80을 교체하고 콜론을 제거하고 새 접두사를 문자열의 나머지 부분에 추가합니다.***

```
[root@target]# ibstatus | grep '<default-gid>' | sed -e 's/<default-gid>://' -e 's/://g' | grep
001175000077dd7e
fe80000000000000001175000077dd7e
```

3. ***targetcli 도구를 사용하여 블록 장치에 LUN vol1을 생성하고 SRP 대상을 생성하고 LUN을 내보냅니다.***

```
[root@target]# targetcli


/> /backstores/block create vol1 /dev/sdc1
Created block storage object vol1 using /dev/sdc1.
/> /srpt create 0xfe80000000000000001175000077dd7e
Created target ib.fe80000000000000001175000077dd7e.
/> /srpt/ib.fe80000000000000001175000077dd7e/luns create /backstores/block/vol1
Created LUN 0.
/> ls /
o- / ..... [...]
  o- backstores ..... [...]
  | o- block ..... [Storage Objects: 1]
  | | o- vol1 ..... [/dev/sdc1 (77.8GiB) write-thru activated]
  | o- fileio ..... [Storage Objects: 0]
  | o- pscsi ..... [Storage Objects: 0]
  | o- ramdisk ..... [Storage Objects: 0]
  o- iscsi ..... [Targets: 0]
  o- loopback ..... [Targets: 0]
  o- srpt ..... [Targets: 1]
    o- ib.fe80000000000000001175000077dd7e ..... [no-gen-acls]
      o- acls ..... [ACLs: 0]
      o- luns ..... [LUNs: 1]
        o- lun0 ..... [block/vol1 (/dev/sdc1)]
/>
```

4.

*이니시에이터에서 **ibstat** 명령을 사용하여 상태가 **Active** 인지 확인하고 **Port GUID:***

```
[root@initiator]# ibstat
CA 'qib0'
 CA type: InfiniPath_QLE7342
 Number of ports: 1
 Firmware version:
 Hardware version: 2
 Node GUID: 0x001175000077d708
 System image GUID: 0x001175000077d708
 Port 1:
  State: Active
  Physical state: LinkUp
  Rate: 40
  Base lid: 2
  LMC: 0
  SM lid: 1
  Capability mask: 0x07690868
  Port GUID: 0x001175000077d708
  Link layer: InfiniBand
```

5.

*원격 **SRP** 대상에 연결하지 않고 스캔하려면 다음 명령을 사용합니다. 대상 **GUID**는 이니시에이터가 원격 타겟을 찾았음을 보여줍니다. **ID** 문자열은 원격 타겟이 **Linux** 소프트웨어 대상 **(ib_srpt.ko)**임을 보여줍니다.*

```
[root@initiator]# srp_daemon -a -o
IO Unit Info:
    port LID:        0001
    port GID:        fe8000000000000001175000077dd7e
    change ID:       0001
    max controllers: 0x10

    controller[  1]
        GUID:      001175000077dd7e
        vendor ID: 000011
        device ID: 007322
        IO class : 0100
        ID:        Linux SRP target
        service entries: 1
            service[  0]: 001175000077dd7e / SRP.T10:001175000077dd7e
```

6.

__SRP__ 연결을 확인하려면 __lsscsi__ 명령을 사용하여 __SCSI__ 장치를 나열하고 이니시에이터가 대상에 연결되기 전후에 __lsscsi__ 출력을 비교합니다.

```
[root@initiator]# lsscsi
[0:0:10:0]   disk    IBM-ESXS ST9146803SS      B53C  /dev/sda
```

7.

*실패할 것으로 예상되는 이니시에이터 포트에 유효한 **ACL**을 구성하지 않고 원격 대상에 연결하려면 **srp_daemon** 또는 **ibsrp dm:**에 대해 다음 명령을 사용합니다.*

```
[root@initiator]# srp_daemon -e -n -i qib0 -p 1 -R 60 &
[1] 4184
```

```
[root@initiator]# ibsrpdm -c -d /dev/infiniband/umad0 > /sys/class/infiniband_srp/srp-qib0-1/add_target
```

8. ***dmesg** 의 출력은 **SRP** 연결 작업이 실패한 이유를 보여줍니다. 이후 단계에서 대상측의 **dmesg** 명령을 사용하여 상황을 명확하게 만듭니다.*

```
[root@initiator]# dmesg -c
[ 1230.059652] scsi host5: ib_srp: REJ received
[ 1230.059659] scsi host5: ib_srp: SRP LOGIN from
fe80:0000:0000:0000:0011:7500:0077:d708 to fe80:0000:0000:0000:0011:7500:0077:dd7e
REJECTED, reason 0x00010006
[ 1230.073792] scsi host5: ib_srp: Connection 0/2 failed
[ 1230.078848] scsi host5: ib_srp: Sending CM DREQ failed
```

9. ***LOGIN** 실패로 인해 **lsscsi** 명령의 출력은 이전 단계와 동일합니다.*

```
[root@initiator]# lsscsi
[0:0:10:0]   disk    IBM-ESXS ST9146803SS      B53C  /dev/sda
```

10. *대상측(**ib_srpt.ko**)에서 **dmesg** 를 사용하면 **LOGIN**이 실패한 이유에 대한 설명이 제공됩니다. 또한 출력에는 **srp_daemon** 에서 제공하는 유효한 **ACL ID**가 포함되어 있습니다 : **0x7edd770000751100001175000077d708.***

```
[root@target]# dmesg
[ 1200.303001] ib_srpt Received SRP_LOGIN_REQ with i_port_id
0x7edd770000751100:0x1175000077d708, t_port_id
0x1175000077dd7e:0x1175000077dd7e and it_iu_len 260 on port 1
(guid=0xfe8000000000000:0x1175000077dd7e)
[ 1200.322207] ib_srpt Rejected login because no ACL has been configured yet for initiator
0x7edd770000751100001175000077d708.
```

11. ***targetcli** 도구를 사용하여 유효한 **ACL**을 추가합니다.*

```
[root@target]# targetcli
targetcli shell version 2.1.fb41
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.
```

```
/> /srpt/ib.fe80000000000000001175000077dd7e/acls create
0x7edd7700007511000001175000077d708
Created Node ACL for ib.7edd7700007511000001175000077d708
Created mapped LUN 0.
```

12. ***SRP** 로그인 작업을 확인합니다.*

    a. ***srp_daemon** 이 다시 로그인할 수 있도록 **60**초 동안 기다립니다.*

    ```
    [root@initiator]# sleep 60
    ```

    b. ***SRP** 로그인 작업을 확인합니다.*

    ```
    [root@initiator]# lsscsi
    [0:0:10:0]   disk    IBM-ESXS ST9146803SS      B53C  /dev/sda
    [7:0:0:0] disk LIO-ORG vol1 4.0 /dev/sdb
    ```

    c. ***SRP** 대상 검색의 커널 로그의 경우 다음을 사용합니다.*

    ```
    [root@initiator]# dmesg -c
    [ 1354.182072] scsi host7: SRP.T10:001175000077DD7E
    [ 1354.187258] scsi 7:0:0:0: Direct-Access     LIO-ORG  vol1            4.0  PQ: 0 ANSI: 5
    [ 1354.208688] scsi 7:0:0:0: alua: supports implicit and explicit TPGS
    [ 1354.215698] scsi 7:0:0:0: alua: port group 00 rel port 01
    [ 1354.221409] scsi 7:0:0:0: alua: port group 00 state A non-preferred supports TOlUSNA
    [ 1354.229147] scsi 7:0:0:0: alua: Attached
    [ 1354.233402] sd 7:0:0:0: Attached scsi generic sg1 type 0
    [ 1354.233694] sd 7:0:0:0: [sdb] 163258368 512-byte logical blocks: (83.5 GB/77.8 GiB)
    [ 1354.235127] sd 7:0:0:0: [sdb] Write Protect is off
    [ 1354.235128] sd 7:0:0:0: [sdb] Mode Sense: 43 00 00 08
    [ 1354.235550] sd 7:0:0:0: [sdb] Write cache: disabled, read cache: enabled, doesn't
    support DPO or FUA
    [ 1354.255491] sd 7:0:0:0: [sdb] Attached SCSI disk
    [ 1354.265233] scsi host7: ib_srp: new target: id_ext 001175000077dd7e ioc_guid
    001175000077dd7e pkey ffff service_id 001175000077dd7e sgid
    fe80:0000:0000:0000:0011:7500:0077:d708 dgid
    fe80:0000:0000:0000:0011:7500:0077:dd7e
    xyx
    ```

13.6. 서브넷 관리자 구성

13.6.1. 필요 확인

*대부분의 **InfiniBand** 스위치는 임베디드 서브넷 관리자와 함께 제공됩니다. 그러나 최신 서브넷 관리*

*자가 스위치 펌웨어에 있는 것보다 더 많은 서브넷 관리자가 필요하거나, 스위치 관리자가 허용하는 것보다 더 완전한 제어가 필요한 경우 **Red Hat Enterprise Linux 7**에는 **opensm** 서브넷 관리자가 포함됩니다. 모든 **InfiniBand** 네트워크에는 네트워크가 작동하려면 서브넷 관리자가 실행되고 있어야 합니다. 이는 스위치가 없는 두 시스템으로 간단한 네트워크를 만들고 카드를 다시 연결한 경우에도 카드의 링크에서 작동하려면 서브넷 관리자가 필요합니다. 둘 이상 있을 수 있습니다. 이 경우 하나는 컨트롤러 역할을 하며, 다른 서브넷 관리자가 컨트롤러 서브넷 관리자가 실패하면 다른 서브넷 관리자가 포트 역할을 합니다.*

13.6.2. opensm 기본 구성 파일 구성

***opensm** 프로그램은 기본 구성 파일을 **/etc/rdma/opensm.conf** 에 유지합니다. 사용자는 언제든지 이 파일을 편집할 수 있으며 편집은 업그레이드에 유지됩니다. 파일 자체에는 옵션에 대한 광범위한 문서가 있습니다. 그러나 가장 일반적인 편집 두 개에 필요한 경우 **GUID**를 바인딩하도록 설정하고 로 실행할 경우 **opensm.conf** 파일이 편집되지 않고 **/etc/sysconfig/opensm** 을 편집하는 것이 좋습니다. 기본 **/etc/rdma/opensm.conf** 파일을 편집하지 않으면 **opensm** 패키지를 업그레이드할 때마다 업그레이드됩니다. 이 파일에 새로운 옵션을 정기적으로 추가하면 현재 구성을 최신 상태로 유지하는 것이 더 쉬워집니다. **opensm.conf** 파일이 변경된 경우 업그레이드 시 편집된 파일에 새 옵션을 병합해야 할 수 있습니다.*

13.6.3. opensm 시작 옵션 구성

***/etc/sysconfig/opensm** 파일의 옵션은 서브넷 관리자가 실제로 시작되는 방법과 서브넷 관리자의 복사본 수를 제어합니다. 예를 들어, 각 포트가 물리적으로 분리된 네트워크에 연결된 이중 포트 **InfiniBand** 카드는 각 포트에서 실행 중인 서브넷 관리자의 사본이 필요합니다. **opensm** 서브넷 관리자는 애플리케이션 인스턴스당 하나의 서브넷만 관리하며 관리해야 하는 각 서브넷에 대해 한 번 시작해야 합니다. 또한, 둘 이상의 **opensm** 서버가 있는 경우 각 서버의 우선순위를 설정하여 포트와 컨트롤러여야 하는 포트를 제어합니다.*

***/etc/sysconfig/opensm** 파일은 서브넷 관리자의 우선 순위를 설정하고 서브넷 관리자가 바인딩하는 **GUID**를 제어하는 간단한 수단을 제공하는 데 사용됩니다. **/etc/sysconfig/opensm** 파일 자체에는 옵션에 대한 광범위한 설명이 있습니다. 사용자는 **opensm** 의 장애 조치 및 다중 파일 운영을 가능하게 하기 위해 파일 자체의 지침을 읽고 따라야 합니다.*

13.6.4. P_Key 정의 생성

*기본적으로 **opensm.conf** 는 패브릭에 만들 파티션 목록을 가져오는 **/etc/rdma/partitions.conf** 파일을 찾습니다. 모든 패브릭에는 **0x7fff** 서브넷이 포함되어야 하며 모든 스위치와 모든 호스트가 해당 패브릭에 속해야 합니다. 그 외에도 다른 파티션을 만들 수 있으며 모든 호스트와 모든 스위치가 이러한 추가 파티션의 멤버일 필요는 없습니다. 이를 통해 관리자는 **InfiniBand** 패브릭에서 이더넷의 **VLAN**과 유사한 서브넷을 만들 수 있습니다. 지정된 속도(예: **40Gbps**)로 파티션을 정의하고 네트워크에서 **40Gbps**를 수행할 수 없는 경우 해당 호스트는 속도 요구 사항과 일치할 수 없으므로 파티션 속도가 가장 느린 속도로 파티션 속도를 설정할 수 없기 때문에 파티션에 참여할 수 있는 권한이 있더라도 해당 호스트가 파티션에 참여할 수 없습니다. 일부 호스트 하위 집합에 대해 더 빠른 파티션이 필요한 경우 더 빠른 속도로 다른 파티션을 생성합니다.*

*다음 파티션 파일의 경우 기본 **0x7fff** 파티션이 **10Gbps**로 단축되고 속도가 **40Gbps**인 **0x0002** 의 파티션이 생성됩니다.*

```
~]$ more /etc/rdma/partitions.conf
# For reference:
# IPv4 IANA reserved multicast addresses:
#   http://www.iana.org/assignments/multicast-addresses/multicast-addresses.txt
# IPv6 IANA reserved multicast addresses:
#   http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xml
#
# mtu =
#   1 = 256
#   2 = 512
#   3 = 1024
#   4 = 2048
#   5 = 4096
#
# rate =
#   2  =   2.5 GBit/s
#   3  =  10   GBit/s
#   4  =  30   GBit/s
#   5  =   5   GBit/s
#   6  =  20   GBit/s
#   7  =  40   GBit/s
#   8  =  60   GBit/s
#   9  =  80   GBit/s
#   10 = 120   GBit/s

Default=0x7fff, rate=3, mtu=4, scope=2, defmember=full:
        ALL, ALL_SWITCHES=full;
Default=0x7fff, ipoib, rate=3, mtu=4, scope=2:
        mgid=ff12:401b::ffff:ffff       # IPv4 Broadcast address
        mgid=ff12:401b::1               # IPv4 All Hosts group
        mgid=ff12:401b::2               # IPv4 All Routers group
        mgid=ff12:401b::16              # IPv4 IGMP group
        mgid=ff12:401b::fb              # IPv4 mDNS group
        mgid=ff12:401b::fc              # IPv4 Multicast Link Local Name Resolution group
        mgid=ff12:401b::101             # IPv4 NTP group
        mgid=ff12:401b::202             # IPv4 Sun RPC
        mgid=ff12:601b::1               # IPv6 All Hosts group
        mgid=ff12:601b::2               # IPv6 All Routers group
        mgid=ff12:601b::16              # IPv6 MLDv2-capable Routers group
        mgid=ff12:601b::fb              # IPv6 mDNS group
        mgid=ff12:601b::101             # IPv6 NTP group
        mgid=ff12:601b::202             # IPv6 Sun RPC group
        mgid=ff12:601b::1:3             # IPv6 Multicast Link Local Name Resolution group
        ALL=full, ALL_SWITCHES=full;

ib0_2=0x0002, rate=7, mtu=4, scope=2, defmember=full:
        ALL, ALL_SWITCHES=full;
ib0_2=0x0002, ipoib, rate=7, mtu=4, scope=2:
        mgid=ff12:401b::ffff:ffff       # IPv4 Broadcast address
        mgid=ff12:401b::1               # IPv4 All Hosts group
        mgid=ff12:401b::2               # IPv4 All Routers group
        mgid=ff12:401b::16              # IPv4 IGMP group
```

```
mgid=ff12:401b::fb          # IPv4 mDNS group
mgid=ff12:401b::fc          # IPv4 Multicast Link Local Name Resolution group
mgid=ff12:401b::101          # IPv4 NTP group
mgid=ff12:401b::202          # IPv4 Sun RPC
mgid=ff12:601b::1           # IPv6 All Hosts group
mgid=ff12:601b::2           # IPv6 All Routers group
mgid=ff12:601b::16          # IPv6 MLDv2-capable Routers group
mgid=ff12:601b::fb          # IPv6 mDNS group
mgid=ff12:601b::101          # IPv6 NTP group
mgid=ff12:601b::202          # IPv6 Sun RPC group
mgid=ff12:601b::1:3         # IPv6 Multicast Link Local Name Resolution group
ALL=full, ALL_SWITCHES=full;
```

13.6.5. opensm 활성화

사용자는 을 설치할 때 기본적으로 활성화되지 않으므로 **opensm** 서비스를 활성화해야 합니다. **root** 로 다음 명령을 실행합니다:

```
~]# systemctl enable opensm
```

13.7. 초기 INFINIBAND RDMA 작업 테스트

참고

이 섹션은 **InfiniBand** 장치에만 적용됩니다. **iWARP** 및 **RoCE/IBoE** 장치는 **IP** 기반 장치이므로, **IPoIB**가 구성되고 장치에 **IP** 주소가 있는 경우 **RDMA** 작업을 테스트하는 데 대한 섹션을 진행해야 합니다.

rdma 서비스가 활성화되고 **opensm** 서비스(필요한 경우)가 활성화되고 특정 하드웨어에 대한 적절한 사용자 공간 라이브러리가 설치되면 사용자 **space rdma** 작업이 가능해야 합니다. **libibverbs-utils** 패키지의 간단한 테스트 프로그램은 **RDMA** 작업이 제대로 작동하는지 확인하는 데 유용합니다. **ibv_devices** 프로그램은 시스템에 있는 장치를 표시하고 **ibv_devinfo** 명령은 각 장치에 대한 자세한 정보를 제공합니다. 예를 들어 다음과 같습니다.

```
~]$ ibv_devices
    device                 node GUID
    ------              ----------------
    mlx4_0              0002c903003178f0
    mlx4_1              f4521403007bcba0
~]$ ibv_devinfo -d mlx4_1
hca_id: mlx4_1
```

```
        transport:                InfiniBand (0)
        fw_ver:                   2.30.8000
        node_guid:                f452:1403:007b:cba0
        sys_image_guid:           f452:1403:007b:cba3
        vendor_id:                0x02c9
        vendor_part_id:           4099
        hw_ver:                   0x0
        board_id:                 MT_1090120019
        phys_port_cnt:            2
                port:   1
                        state:            PORT_ACTIVE (4)
                        max_mtu:          4096 (5)
                        active_mtu:       2048 (4)
                        sm_lid:           2
                        port_lid:         2
                        port_lmc:         0x01
                        link_layer:       InfiniBand

                port:   2
                        state:            PORT_ACTIVE (4)
                        max_mtu:          4096 (5)
                        active_mtu:       4096 (5)
                        sm_lid:           0
                        port_lid:         0
                        port_lmc:         0x00
                        link_layer:       Ethernet
~]$ ibstat mlx4_1
CA 'mlx4_1'
        CA type: MT4099
        Number of ports: 2
        Firmware version: 2.30.8000
        Hardware version: 0
        Node GUID: 0xf4521403007bcba0
        System image GUID: 0xf4521403007bcba3
        Port 1:
                State: Active
                Physical state: LinkUp
                Rate: 56
                Base lid: 2
                LMC: 1
                SM lid: 2
                Capability mask: 0x0251486a
                Port GUID: 0xf4521403007bcba1
                Link layer: InfiniBand
        Port 2:
                State: Active
                Physical state: LinkUp
                Rate: 40
                Base lid: 0
                LMC: 0
                SM lid: 0
                Capability mask: 0x04010000
                Port GUID: 0xf65214fffe7bcba2
                Link layer: Ethernet
```

ibv_devinfo 및 ibstat 명령은 약간 다른 정보를 출력합니다(예: port MTU는 ibv_devinfo 에 있지만 ibstat 출력에는 없음) Port GUID는 ibstat 출력에 존재하지만 ibv_devinfo 출력에는 존재하지 않으며 몇 가지 항목은 다르게 지정됩니다(예:). ibstat 출력의 기본 로컬 식별자 (LID)는 ib v _devinfo의 port_ lid출력과 동일합니다.

infiniband-diags 패키지에서 ibping 과 같은 간단한 ping 프로그램을 사용하여 RDMA 연결을 테스트할 수 있습니다. ibping 프로그램은 클라이언트-서버 모델을 사용합니다. 먼저 한 시스템에서 ibping 서버를 시작한 다음 다른 시스템에서 ibping 을 클라이언트로 실행하고 ibping 서버에 연결하라고 알려야 합니다. 기본 RDMA 기능을 테스트하려고 하므로 서버를 지정하기 위해 IP 주소 대신 RDMA 특정 주소 확인 방법을 사용해야 합니다.

서버 시스템에서 사용자는 ibv_devinfo 및 ib stat 명령을 사용하여 테스트할 포트의 port_lid (또는 Base lid)를 출력할 수 있습니다(위 인터페이스의 포트 1이라고 가정함, port_lid/Base LID 는 2 이고 Port GUID는 0xf4521407bcba1)입니다. 그런 다음 테스트할 카드와 포트에 특별히 바인딩하는 데 필요한 옵션으로 ibping 을 시작하고 ibping 을 서버 모드에서 실행해야 합니다. 를 전달하여 ibping 또는 --help 를 볼 수 있지만, 이 경우 -S 또는 --Server 옵션이 필요하고 특정 카드와 포트에 바인딩하려면 -C 또는 --Ca 및 - P 또는 -- Port 가 필요합니다. 참고: 이 인스턴스의 포트는 네트워크 포트 번호를 나타내지 않지만 다중 포트 카드를 사용할 때 카드의 실제 포트 번호를 나타냅니다. 예를 들어 다중 포트 카드의 두 번째 포트를 사용하여 RDMA 패브릭에 대한 연결을 테스트하려면 ibping 에 카드의 포트 2 에 바인딩하도록 지시해야 합니다. 단일 포트 카드를 사용하거나 카드에서 첫 번째 포트를 테스트할 때는 이 옵션이 필요하지 않습니다. 예를 들어 다음과 같습니다.

```
~]$ ibping -S -C mlx4_1 -P 1
```

그런 다음 클라이언트 시스템으로 변경하고 ibping 을 실행합니다. 서버 ibping 프로그램이 바인딩되어 있는 포트 GUID 포트 또는 서버 ibping 프로그램이 바인딩되는 포트의LID(로컬 식별자) 를 기록해 둡니다. 또한 클라이언트 시스템의 어떤 카드 및 포트가 서버에 바인딩된 카드 및 포트와 동일한 네트워크에 물리적으로 연결되어 있는지 확인합니다. 예를 들어 서버에 있는 첫 번째 카드의 두 번째 포트가 바인딩되고 해당 포트가 보조 RDMA 패브릭에 연결된 경우 클라이언트의 경우 해당 보조 패브릭에 연결하는 데 필요한 카드와 포트를 지정합니다. 이러한 항목을 파악한 후 ibping 프로그램을 클라이언트로 실행하고 에서 연결할 주소로 서버에서 수집된 LID 또는 GUID 포트를 사용하여 서버에 연결합니다. 예를 들어 다음과 같습니다.

```
~]$ ibping -c 10000 -f -C mlx4_0 -P 1 -L 2
--- rdma-host.example.com.(none) (Lid 2) ibping statistics ---
10000 packets transmitted, 10000 received, 0% packet loss, time 816 ms
rtt min/avg/max = 0.032/0.081/0.446 ms
```

또는

```
~]$ ibping -c 10000 -f -C mlx4_0 -P 1 -G 0xf4521403007bcba1 \
```

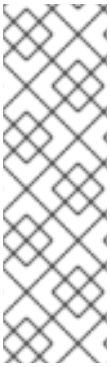
```
--- rdma-host.example.com.(none) (Lid 2) ibping statistics ---
10000 packets transmitted, 10000 received, 0% packet loss, time 769 ms
rtt min/avg/max = 0.027/0.076/0.278 ms
```

이 결과는 사용자 공간 애플리케이션에서 최종 RDMA 통신이 작동하는지 확인합니다.

다음과 같은 오류가 발생할 수 있습니다.

```
~]$ ibv_devinfo
libibverbs: Warning: no userspace device-specific driver found for
/sys/class/infiniband_verbs/uverbs0
No IB devices found
```

이 오류는 필요한 사용자 공간 라이브러리가 설치되지 않았음을 나타냅니다. 관리자는 섹션 13.4절. "InfiniBand 및 RDMA 관련 소프트웨어 패키지" 에 나열된 사용자 공간 라이브러리(하드웨어에 적합) 중 하나를 설치해야 합니다. 드문 경우지만 사용자가 드라이버 또는 libibverbs 에 대해 잘못된 아키텍처 유형을 설치하는 경우 이러한 상황이 발생할 수 있습니다. 예를 들어 libibverbs 가 arch x86_64 이고 libmlx4 가 설치되어 있지만 i686 유형이 설치된 경우 이 오류는 발생할 수 있습니다.



참고

많은 샘플 애플리케이션에서는 LID 대신 호스트 이름 또는 주소를 사용하여 서버와 클라이언트 간의 통신을 여는 것을 선호합니다. 이러한 애플리케이션의 경우 엔드 투 엔드 RDMA 통신을 테스트하기 전에 IPoIB를 설정해야 합니다. ibping 애플리케이션은 간단한 LID를 주소 지정 형태로 허용할 예정이므로 테스트 시나리오에서 IPoIB 주소 지정에 발생할 수 있는 문제를 없애므로 단순한 RDMA 통신이 작동하는지 여부를 보다 격리된 보기를 제공할 수 있습니다.

13.8. IPOIB 구성

13.8.1. IPoIB의 역할 이해

1.1절. "IP와 IP가 아닌 네트워크 비교" 에서 언급했듯이 대부분의 네트워크는 IP 네트워크입니다. InfiniBand는 그렇지 않습니다. IPoIB의 역할은 InfiniBand RDMA 네트워크 위에 IP 네트워크 에뮬레이션 계층을 제공하는 것입니다. 이를 통해 기존 애플리케이션을 수정하지 않은 InfiniBand 네트워크를 통해 실행할 수 있습니다. 그러나 이러한 애플리케이션의 성능은 기본적으로 RDMA 통신을 사용하도록 작성된 경우보다 훨씬 낮습니다. 대부분의 InfiniBand 네트워크에는 네트워크에서 가능한 모든 성능을 확보해야 하는 몇 가지 애플리케이션 세트가 있으며, RDMA 통신을 사용하도록 애플리케이션을 수정할 필요가 없는 경우 성능이 저하되는 일부 다른 애플리케이션도 있으므로, IPoIB는 네트워크에서 중요한 애플리케이션이 실행되도록 허용하는 것입니다.

iWARP 및 RoCE/IBoE 네트워크는 실제로 IP 링크 계층 위에 RDMA 계층이 있는 IP 네트워크이므로 IPoIB가 필요하지 않습니다. 결과적으로 커널은 iWARP 또는 RoCE/IBoE RDMA 장치에 IPoIB 장치 생성

을 거부합니다.

13.8.2. IPoIB 통신 모드 이해

IPoIB 장치는 데이터그램 또는 연결 모드로 실행되도록 구성할 수 있습니다. 차이점은 IPoIB 계층이 통신의 다른 끝에 있는 시스템으로 열려고 시도하는 대기열 쌍의 유형에 있습니다. 데이터그램 모드의 경우 불안정하고 연결이 끊긴 큐 쌍이 열립니다. 연결 모드의 경우 신뢰할 수 있고 연결된 대기열 쌍이 열립니다.

데이터그램 모드를 사용하는 경우 불안정하고 연결이 끊긴 큐 쌍 유형에서는 InfiniBand 링크 계층의 MTU보다 큰 패킷을 허용하지 않습니다. IPoIB 계층은 전송 중인 IP 패킷 상단에 4바이트 IP oIB 헤더를 추가합니다. 따라서 IPoIB MTU는 InfiniBand 링크 계층 MTU보다 4바이트 작아야 합니다. 2048는 일반적인 InfiniBand 링크 계층 MTU이므로 데이터그램 모드의 일반적인 IPoIB 장치 MTU는 2044입니다.

연결 모드를 사용하는 경우 신뢰할 수 있는 연결된 대기열 쌍 유형을 사용하면 InfiniBand 링크 계층 MTU보다 큰 메시지를 허용하고 호스트 어댑터는 패킷 분할을 처리하고 각 끝에서 다시 조합합니다. 결과적으로 연결된 모드에서 InfiniBand 어댑터에서 보낼 수 있는 IPoIB 메시지의 크기에 따른 크기 제한이 없습니다. 그러나 IP 패킷에 16비트 크기 필드만 있고 최대 바이트 수로 65535 로 제한된다는 제한이 여전히 있습니다. 허용되는 최대 MTU는 실제로 해당 크기에도 부합해야 하는 다양한 TCP/IP 헤더에 대해 설명해야 하므로 실제로보다 작습니다. 결과적으로 연결된 모드의 IPoIB MTU는 필요한 모든 TCP 헤더에 사용할 공간이 충분한지 확인하기 위해 65520 으로 제한됩니다.

연결 모드 옵션은 일반적으로 성능이 뛰어나지만 커널 메모리도 더 많이 사용됩니다. 대부분의 시스템은 메모리 소비보다 성능이 더 중요하기 때문에 연결 모드는 가장 일반적으로 사용되는 모드입니다.

그러나 연결된 모드를 위해 시스템이 구성된 경우 데이터그램 모드에서 멀티캐스트 트래픽을 보내야 하며(Infiniband 스위치 및 패브릭이 연결된 모드에서 멀티캐스트 트래픽을 전달할 수 없음) 연결 모드로 구성되지 않은 호스트와 통신할 때 데이터그램 모드로 대체됩니다. 관리자는 멀티 캐스트 데이터를 전송하는 프로그램을 실행하려는 경우 해당 프로그램이 인터페이스에서 최대 MTU까지 멀티캐스트 데이터를 보내려고하는 경우 데이터그램 작업의 인터페이스를 구성하거나 데이터그램 크기가 지정된 패킷에 맞는 크기로 패킷 전송 크기를 제한하도록 멀티캐스트 애플리케이션을 구성하는 방법을 찾아야 한다는 점을 알고 있어야 합니다.

13.8.3. IPoIB 하드웨어 주소 이해

IPoIB 장치에는 20바이트 하드웨어 주소가 있습니다. 더 이상 사용되지 않는 유틸리티 ifconfig 는 20바이트를 모두 읽을 수 없으며 IPoIB 장치에 대한 올바른 하드웨어 주소를 찾는 데 사용되지 않아야 합니다. ip route 패키지의 ip 유틸리티가 올바르게 작동합니다.

IPoIB 하드웨어 주소의 처음 4바이트는 플래그와 대기열 쌍 번호입니다. 다음 8바이트는 서브넷 접두사입니다. IPoIB 장치가 처음 생성되면 기본 서브넷 접두사 0xfe:80:00:00:00:00:00:00 이 됩니다. 해당 장치

는 서브넷 관리자와 연결할 때까지 기본 서브넷 접두사(0xfe80000000000000)를 사용합니다. 이 경우 서브넷 관리자가 구성한 설정과 일치하도록 서브넷 접두사를 재설정합니다. 마지막 8바이트는 IPoIB 장치가 연결된 InfiniBand 포트의 GUID 주소입니다. 처음 4바이트와 다음 8바이트 모두 수시로 변경될 수 있기 때문에 IPoIB 인터페이스에 대한 하드웨어 주소를 지정할 때 사용되지 않거나 일치하지 않습니다. 섹션 13.5.2절. "70-persistent-ipoib.rules 사용" 은 장치 일치가 안정적으로 수행되도록 ATTR{address} 필드를 udev 규칙 파일에 두어 주소를 파생하는 방법을 설명합니다. IPoIB 인터페이스를 구성할 때 구성 파일의 HWADDR 필드에는 모든 20바이트를 포함할 수 있지만, 실제로 일치하는 마지막 8바이트만 구성 파일에서 지정한 하드웨어를 찾는 데 사용됩니다. 그러나 TYPE=InfiniBand 항목이 장치 구성 파일에서 올바르게 철자 지정되지 않고 ifup-ib 가 IPoIB 인터페이스를 여는 데 사용되는 실제 스크립트가 아닌 경우 구성에서 지정한 하드웨어를 찾을 수 없는 시스템에 대한 오류가 발생합니다. IPoIB 인터페이스의 경우 구성 파일의 TYPE= 필드는 InfiniBand 또는 infiniband 여야 합니다. 항목이 대소문자를 구분하지만 스크립트는 이러한 두 개의 특정 철자를 수락합니다.

13.8.4. InfiniBand P_Key 서브넷 이해

InfiniBand 패브릭은 다양한 P_Key 서브넷을 사용하여 가상 서브넷으로 논리적으로 분할할 수 있습니다. 이는 이더넷 인터페이스에서 VLAN을 사용하는 것과 매우 유사합니다. 모든 스위치와 호스트는 기본 P_Key 서브넷의 멤버여야 하지만 관리자가 추가 서브넷을 생성하고 해당 서브넷의 구성원을 패브릭에 있는 호스트 또는 스위치의 하위 집합으로 제한할 수 있습니다. 호스트에서 사용하려면 서브넷 관리자가 P_Key 서브넷을 정의해야 합니다. opensm 서브넷 관리자를 사용하여 P_Key 서브넷을 정의하는 방법에 대한 자세한 내용은 13.6.4절. "P_Key 정의 생성" 섹션을 참조하십시오. IPoIB 인터페이스의 경우, P_Key 서브넷이 생성되면 P_Key 서브넷을 위한 IPoIB 구성 파일을 추가로 만들 수 있습니다. 이더넷 장치의 VLAN 인터페이스와 마찬가지로 각 IPoIB 인터페이스는 동일한 링크를 공유하지만 P_Key 값이 서로 다른 IPoIB 인터페이스와 완전히 다른 패브릭과 같이 작동합니다.

IPoIB P_Key 인터페이스의 이름에 대한 특별한 요구 사항이 있습니다. 모든 IPoIB P_Key범위 0x0000 에서 0x7fff, 상위 비트 0x8000 은 P_Key 의 멤버가 부분 멤버십 대신 전체 멤버십임을 나타냅니다. Linux 커널의 IPoIB 드라이버는 P_Key 서브넷의 전체 멤버십만 지원하므로 Linux가 연결할 수 있는 서브넷의 경우 상위 비트의 P_Key 번호가 항상 설정됩니다. 즉, Linux 컴퓨터가 P_Key 0x0002 에 조인하면 한 번 가입한 실제 P_Key 번호는 0x8002 가 됩니다. P_Key 0x0002 의 전체 구성원임을 알 수 있습니다. 이러한 이유로 13.6.4절. "P_Key 정의 생성" 절에 표시된 대로 opensm partitions.conf 파일에 P_Key 정의를 생성하는 경우 0x8000 없이 P_Key 값을 지정해야 하지만 Linux 클라이언트에서 P_Key IPoIB 인터페이스를 정의할 때는 기본 P_Key 값에 0x8000 값을 추가합니다.

13.8.5. 텍스트 사용자 인터페이스 nmtui를 사용하여 InfiniBand 구성

텍스트 사용자 인터페이스 도구 nmtui 는 터미널 창에서 InfiniBand를 구성하는 데 사용할 수 있습니다. 다음 명령을 실행하여 도구를 시작합니다.

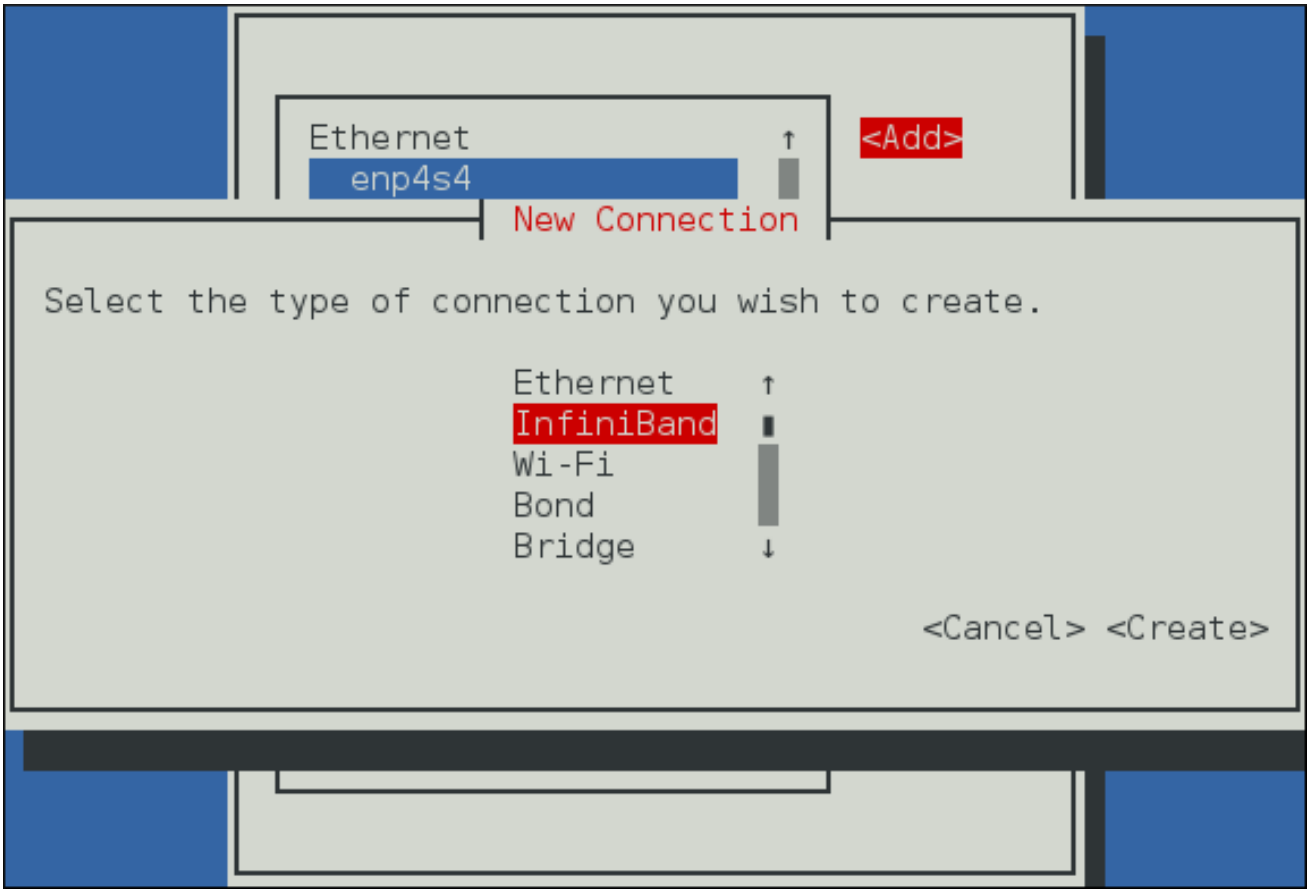
```
~]$ nmtui
```

텍스트 사용자 인터페이스가 나타납니다. 잘못된 명령은 사용 메시지를 인쇄합니다.

탐색하려면 화살표 키를 사용하거나 탭을 눌러 앞으로 이동하고 Shift+Tab 을 눌러 옵션을 다시 이동합니다. Enter 를 눌러 옵션을 선택합니다. Space 표시줄에서 확인란의 상태를 전환합니다.

*시작 메뉴에서 **Edit a connection** (연결 편집)을 선택합니다. **Add** 를 선택하면 **New Connection** (새 연결) 화면이 열립니다.*

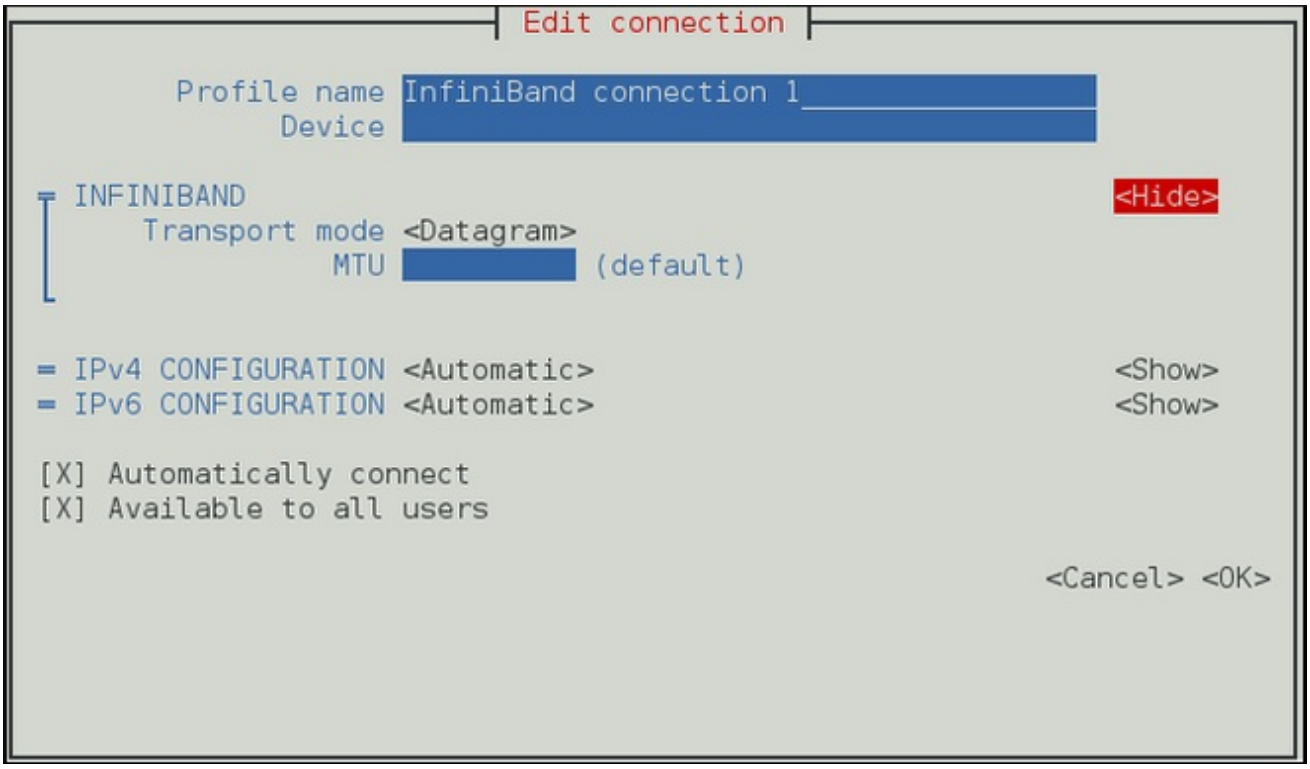
*그림 **13.1. NetworkManager** 텍스트 사용자 인터페이스 **Add an InfiniBand Connection** 메뉴*



[D]

***InfiniBand** 를 선택합니다. 연결 편집 화면이 열립니다. 화면의 메시지에 따라 구성을 완료합니다.*

그림 13.2. NetworkManager 텍스트 사용자 인터페이스 InfiniBand 연결 구성 메뉴



[D]

InfiniBand 용어에 대한 정의는 13.8.9.1절. “InfiniBand 탭 구성” 을 참조하십시오.

nmtui 설치에 대한 자세한 내용은 3.2절. “nmtui로 IP 네트워킹 구성” 을 참조하십시오.

13.8.6. 명령줄 도구 nmcli를 사용하여 IPoIB 구성

먼저 기본 IPoIB 장치의 이름을 변경해야 하는지 확인하고 13.5.2절. “70-persistent-ipoib.rules 사용” 섹션의 지침에 따라 udev 변경 규칙을 사용하여 장치 이름을 바꿉니다. 사용자는 ib_ipoib 커널 모듈을 제거한 다음 다음과 같이 다시 로드하여 재부팅을 수행하지 않고 IPoIB 인터페이스의 이름을 강제 변경할 수 있습니다.

```
~]$ rmmod ib_ipoib
~]$ modprobe ib_ipoib
```

장치 이름이 필요한 경우 nmcli 툴을 사용하여 IPoIB 인터페이스를 만듭니다. 다음 예제에서는 다음 두 가지 방법을 보여줍니다.

예 13.3. 두 개의 개별 명령에서 IPoIB 생성 및 수정.

```
~]$ nmcli con add type infiniband con-name mlx4_ib0 ifname mlx4_ib0 transport-mode connected
mtu 65520
Connection 'mlx4_ib0' (8029a0d7-8b05-49ff-a826-2a6d722025cc) successfully added.
~]$ nmcli con edit mlx4_ib0

===| nmcli interactive connection editor |===

Editing existing 'infiniband' connection: 'mlx4_ib0'

Type 'help' or '?' for available commands.
Type 'describe [>setting<.>prop<]' for detailed property description.

You may edit the following settings: connection, infiniband, ipv4, ipv6
nmcli> set infiniband.mac-address
80:00:02:00:fe:80:00:00:00:00:00:00:00:f4:52:14:03:00:7b:cb:a3
nmcli> save
Connection 'mlx4_ib3' (8029a0d7-8b05-49ff-a826-2a6d722025cc) successfully updated.
nmcli> quit
```

또는 다음과 같이 한 명령에서 **nmcli c add** 및 **nmcli c modify** 을 실행할 수 있습니다.

예 13.4. 한 명령에서 **IPoIB** 생성 및 수정.

```
nmcli con add type infiniband con-name mlx4_ib0 ifname mlx4_ib0 transport-mode connected mtu
65520  infiniband.mac-address 80:00:02:00:fe:80:00:00:00:00:00:00:00:f4:52:14:03:00:7b:cb:a3
```

이 시점에서 **mlx4_ib0** 라는 **IPoIB** 인터페이스가 생성되어 최대 연결 모드 **MTU, IPv4** 및 **IPv 6** 용 **DHCP** 를 사용하여 연결 모드를 사용하도록 설정되었습니다. 클러스터 트래픽에 **IPoIB** 인터페이스를 사용하고 클러스터 외부에서 이더넷 인터페이스를 사용하는 경우 기본 경로를 비활성화하고 **IPoIB** 인터페이스에서 기본 이름 서버가 필요할 수 있습니다. 이 작업은 다음과 같이 수행할 수 있습니다.

```
~]$ nmcli con edit mlx4_ib0

===| nmcli interactive connection editor |===

Editing existing 'infiniband' connection: 'mlx4_ib0'
```

```
Type 'help' or '?' for available commands.
Type 'describe [>setting<.>prop<]' for detailed property description.

You may edit the following settings: connection, infiniband, ipv4, ipv6
nmcli> set ipv4.ignore-auto-dns yes
nmcli> set ipv4.ignore-auto-routes yes
nmcli> set ipv4.never-default true
nmcli> set ipv6.ignore-auto-dns yes
nmcli> set ipv6.ignore-auto-routes yes
nmcli> set ipv6.never-default true
nmcli> save
Connection 'mlx4_ib0' (8029a0d7-8b05-49ff-a826-2a6d722025cc) successfully updated.
nmcli> quit
```

P_Key 인터페이스가 필요한 경우 다음과 같이 **nmcli** 를 사용하여 하나를 생성합니다.

```
~]$ nmcli con add type infiniband con-name mlx4_ib0.8002 ifname mlx4_ib0.8002 parent mlx4_ib0 p-key 0x8002
Connection 'mlx4_ib0.8002' (4a9f5509-7bd9-4e89-87e9-77751a1c54b4) successfully added.
~]$ nmcli con modify mlx4_ib0.8002 infiniband.mtu 65520 infiniband.transport-mode connected ipv4.ignore-auto-dns yes ipv4.ignore-auto-routes yes ipv4.never-default true ipv6.ignore-auto-dns yes ipv6.ignore-auto-routes yes ipv6.never-default true
```

13.8.7. 명령줄을 사용하여 IPoIB 구성

먼저 기본 **IPoIB** 장치의 이름을 변경해야 하는지 확인하고 13.5.2절. “70-persistent-ipoib.rules 사용” 섹션의 지침에 따라 **udev** 변경 규칙을 사용하여 장치 이름을 바꿉니다. 사용자는 **ib_ipoib** 커널 모듈을 제거한 다음 다음과 같이 다시 로드하여 재부팅을 수행하지 않고 **IPoIB** 인터페이스의 이름을 강제 변경할 수 있습니다.

```
~]$ rmmod ib_ipoib
~]$ modprobe ib_ipoib
```

장치 이름이 필요한 경우 관리자는 선호하는 편집기에서 **ifcfg** 파일을 만들어 장치를 제어할 수 있습니다. 정적 **IPv4** 주소 지정을 사용하는 일반적인 **IPoIB** 구성 파일은 다음과 같습니다.

```
~]$ more ifcfg-mlx4_ib0
DEVICE=mlx4_ib0
TYPE=InfiniBand
```

```
ONBOOT=yes
HWADDR=80:00:00:4c:fe:80:00:00:00:00:00:00:00:f4:52:14:03:00:7b:cb:a1
BOOTPROTO=none
IPADDR=172.31.0.254
PREFIX=24
NETWORK=172.31.0.0
BROADCAST=172.31.0.255
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
MTU=65520
CONNECTED_MODE=yes
NAME=mlx4_ib0
```

DEVICE 필드는 udev 변경 규칙에서 생성된 사용자 정의 이름과 일치해야 합니다. NAME 항목은 장치 이름과 일치하지 않아도 됩니다. GUI 연결 편집기가 시작되면 NAME(이름) 필드는 이 연결의 이름을 사용자에게 제공하는 데 사용됩니다. InfiniBand 옵션을 올바르게 처리하려면 TYPE 필드는 InfiniBand 여야 합니다. CONNECTED_MODE는 yes 또는 no 이며 yes 는 연결 모드를 사용하고 통신에 데이터그램 모드를 사용하지 않습니다 (13.8.2절. “IPoIB 통신 모드 이해”섹션 참조).

P_Key 인터페이스의 경우 일반적인 구성 파일입니다.

```
~]$ more ifcfg-mlx4_ib0.8002
DEVICE=mlx4_ib0.8002
PHYSDEV=mlx4_ib0
PKEY=yes
PKEY_ID=2
TYPE=InfiniBand
ONBOOT=yes
HWADDR=80:00:00:4c:fe:80:00:00:00:00:00:00:00:f4:52:14:03:00:7b:cb:a1
BOOTPROTO=none
IPADDR=172.31.2.254
PREFIX=24
NETWORK=172.31.2.0
BROADCAST=172.31.2.255
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
MTU=65520
CONNECTED_MODE=yes
NAME=mlx4_ib0.8002
```

모든 P_Key 인터페이스 파일의 경우 PHYSDEV 지시문이 필요하며 상위 장치의 이름이어야 합니다. PKEY 지시문은 yes 로 설정해야 하며, PKEY_ID 는 인터페이스 수여야 합니다(에 0x8000 멤버십 비트 추가 또는 미포함). 그러나 장치 이름은 다음과 같이 논리 OR 연산자를 사용하여 0x8000 멤버십 비트와 결합된 PKEY_ID 의 4자리 16진수여야 합니다.

```
NAME=${PHYSDEV}.$((0x8000 | $PKEY_ID))
```

기본적으로 파일의 PKEY_ID 는 10진수로 처리되고 16진수로 변환된 다음 0x8000 이 있는 논리 OR

연산자를 사용하여 장치에 적절한 이름을 가져오지만 표준 **0x** 접두사를 앞에 추가하여 **16**진수에 **PKEY_ID** 를 지정할 수 있습니다.

13.8.8. IPoIB가 구성된 후 RDMA 네트워크 테스트

IPoIB가 구성되면 **IP** 주소를 사용하여 **RDMA** 장치를 지정할 수 있습니다. **IP** 주소와 호스트 이름을 사용하여 시스템을 지정하는 유비쿼터스 특성으로 인해 대부분의 **RDMA** 애플리케이션은 이를 선호하거나 경우에 따라 연결할 원격 시스템 또는 로컬 장치를 지정하는 방식으로만 사용합니다.

IPoIB 계층의 기능을 테스트하려면 모든 표준 **IP** 네트워크 테스트 도구를 사용하고 테스트할 **IP oIB** 장치의 **IP** 주소를 제공할 수 있습니다. 예를 들어, **IPoIB** 장치의 **IP** 주소 간 **ping** 명령이 작동해야 합니다.

Red Hat Enterprise Linux에는 **qperf** 및 **perftest** 라는 두 가지 **RDMA** 성능 테스트 패키지가 포함되어 있습니다. 이 중 하나를 사용하여 **RDMA** 네트워크의 성능을 추가로 테스트할 수 있습니다.

그러나 **perftest** 패키지의 일부인 애플리케이션을 사용하거나 **qperf** 애플리케이션을 사용하는 경우 주소 확인에 대한 특별한 참고 사항이 있습니다. **IPoIB** 장치의 **IP** 주소 또는 호스트 이름을 사용하여 원격 호스트를 지정하더라도 테스트 애플리케이션이 다른 **RDMA** 인터페이스를 통해 실제로 연결할 수 있습니다. 그 이유는 호스트 이름 또는 **IP** 주소에서 **RDMA** 주소로 변환하는 프로세스를 통해 두 시스템 간에 유효한 **RDMA** 주소 쌍을 사용할 수 있기 때문입니다. 클라이언트가 서버에 연결할 수 있는 여러 가지 방법이 있는 경우 지정된 경로에 문제가 있는 경우 프로그램이 다른 경로를 사용하도록 선택할 수 있습니다. 예를 들어, 각 시스템에 동일한 **InfiniBand** 서브넷에 연결된 두 개의 포트가 있고 각 시스템에서 두 번째 포트에 대한 **IP** 주소가 제공되는 경우 프로그램이 각 시스템에서 첫 번째 포트를 찾고 대신 사용합니다. 이 경우, **perftest** 프로그램에 대한 명령줄 옵션을 사용하여 테스트할 특정 포트를 테스트해야 하는 특정 포트에서 테스트를 수행하도록 하기 위해 *13.7절. “초기 InfiniBand RDMA 작업 테스트”* 에서 **ibping** 과 같이 바인딩할 카드와 포트를 알려주는 데 사용할 수 있습니다. **qperf** 의 경우 포트에 바인딩하는 방법은 약간 다릅니다. **qperf** 프로그램은 하나의 시스템에서 서버로 작동하며 모든 장치(**RDMA** 장치 포함)에서 수신 대기합니다. 클라이언트는 서버의 유효한 **IP** 주소 또는 호스트 이름을 사용하여 **qperf** 에 연결할 수 있습니다. **q perf** 는 먼저 데이터 연결을 열고 클라이언트 명령줄에 지정된 **IP** 주소 또는 호스트 이름을 통해 요청된 테스트를 실행하지만 해당 주소를 사용하는 데 문제가 있는 경우 **qperf** 는 클라이언트와 서버 간에 유효한 경로에서 테스트를 실행하려고 시도합니다. 이러한 이유로 **qperf** 가 특정 링크에서 테스트하도록 하려면 **-loc_id** 및 **-rem_id** 옵션을 **qperf** 클라이언트에 사용하여 특정 링크에서 테스트를 강제 실행합니다.

13.8.9. GUI를 사용하여 IPoIB 구성

그래픽 도구를 사용하여 **InfiniBand** 연결을 구성하려면 **nm-connection-editor**를 사용합니다.

절차 **13.4. nm-connection-editor**를 사용하여 새 **InfiniBand** 연결 추가

1. 터미널에서 **nm-connection-editor** 를 입력합니다.

```
~]$ nm-connection-editor
```

2. **Add(추가)** 단추를 클릭합니다. **Choose a Connection Type(연결 유형 선택)** 창이 표시됩니다. **InfiniBand** 를 선택하고 **Create(생성)**를 클릭합니다. **Editing InfiniBand connection 1** 창이 표시됩니다.

3. **InfiniBand** 탭의 **InfiniBand** 연결에 사용할 드롭다운 목록에서 전송 모드를 선택합니다.

4. **InfiniBand MAC** 주소를 입력합니다.

5. 설정을 검토 및 확인한 다음 **Save (저장)** 단추를 클릭합니다.

6. 13.8.9.1절. “InfiniBand 탭 구성” 을 참조하여 **InfiniBand**별 설정을 편집합니다.

절차 **13.5.** 기존 **InfiniBand** 연결 편집

다음 단계에 따라 기존 **InfiniBand** 연결을 편집합니다.

1. 터미널에서 **nm-connection-editor** 를 입력합니다.

```
~]$ nm-connection-editor
```

2. 편집할 연결을 선택하고 **Edit (편집)** 단추를 클릭합니다.

3. **General(일반)** 탭을 선택합니다.

4. 연결 이름, 자동 연결 동작 및 가용성 설정을 구성합니다.

편집 대화 상자의 **5**가지 설정은 모든 연결 유형에 공통적입니다. **General(일반)** 탭을 참조하십시오.

- *연결 이름 - 네트워크 연결에 대한 설명이 포함된 이름을 입력합니다. 이 이름은 **Network** (네트워크) 창의 메뉴에 이 연결을 나열하는 데 사용됩니다.*

- *사용 가능한 경우 이 네트워크에 자동으로 연결 - 사용 가능한 경우 **NetworkManager** 가 이 연결에 자동으로 연결되도록 하려면 이 상자를 선택합니다. 자세한 내용은 "제어 센터를 사용하여 기존 연결 편집"을 참조하십시오.*

- *모든 사용자가 이 네트워크에 연결할 수 있습니다 - 이 상자를 선택하여 시스템의 모든 사용자가 사용할 수 있는 연결을 만듭니다. 이 설정을 변경하려면 **root** 권한이 필요할 수 있습니다. 자세한 내용은 **3.4.5절. "GUI를 사용하여 시스템 전체 및 개인 연결 프로파일 관리"** 을 참조하십시오.*

- *이 연결을 사용할 때 **VPN**에 자동으로 연결 - **NetworkManager** 가 **VPN** 연결에 자동으로 연결되도록 하려면 이 상자를 선택합니다. 드롭다운 메뉴에서 **VPN**을 선택합니다.*

- *방화벽 영역 - 드롭다운 메뉴에서 방화벽 영역을 선택합니다. 방화벽 영역에 대한 자세한 내용은 **Red Hat Enterprise Linux 7** 보안 가이드 를 참조하십시오.*

5. *13.8.9.1절. "InfiniBand 탭 구성" 을 참조하여 **InfiniBand**별 설정을 편집합니다.*

새 연결 (또는 수정된) 연결 저장 및 추가 설정 만들기

*InfiniBand 연결 편집을 마치면 **Save** (저장) 버튼을 클릭하여 사용자 지정 구성을 저장합니다.*

그런 다음 다음을 구성하려면 다음을 수행합니다.

- *연결의 **IPv4** 설정, **IPv4** 설정 탭을 클릭하고 계속 진행합니다. **5.4절. "IPv4 설정 구성"***

  또는

- *연결의 **IPv6** 설정, **IPv6** 설정 탭을 클릭하고 **5.5절. "IPv6 설정 구성"** 진행합니다.*

13.8.9.1. InfiniBand 탭 구성

새로운 InfiniBand 연결을 이미 추가한 경우(명령은 절차 13.4. “nm-connection-editor를 사용하여 새 InfiniBand 연결 추가” 참조), 상위 인터페이스와 InfiniBand ID를 설정하기 위해 InfiniBand 탭을 편집할 수 있습니다.

전송 모드

데이터그램 또는 연결 모드는 드롭다운 목록에서 선택할 수 있습니다. 나머지 IPoIB 네트워크에서 사용 중인 것과 동일한 모드를 선택합니다.

장치 MAC 주소

InfiniBand 네트워크 트래픽에 사용할 InfiniBand 가능 장치의 MAC 주소입니다. 이 하드웨어 주소 필드는 InfiniBand 하드웨어를 설치한 경우 미리 채워집니다.

MTU

선택적으로 InfiniBand 연결을 통해 보내지는 패킷에 사용할 MTU(최대 전송 단위) 크기를 설정합니다.

13.8.10. 추가 리소스

설치된 문서

- /usr/share/doc/initscripts-버전/sysconfig.txt - 구성 파일과 해당 지시문을 설명합니다.

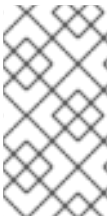
온라인 문서

https://www.kernel.org/doc/Documentation/infiniband/ipoib.txt

IPoIB 드라이버에 대한 설명입니다. 관련 RFC 참조 포함.

IV 부. 서버

이 부분에서는 네트워킹에 일반적으로 필요한 서버를 설정하는 방법에 대해 설명합니다.



참고

웹 브라우저를 통해 서버를 모니터링하고 관리하려면 RHEL 7 웹 콘솔을 사용하여 시스템 관리를 참조하십시오.

14장. DHCP 서버

DHCP(Dynamic Host Configuration Protocol)는 TCP/IP 정보를 클라이언트 시스템에 자동으로 할당하는 네트워크 프로토콜입니다. 각 DHCP 클라이언트는 중앙에 있는 DHCP 서버에 연결하여 해당 클라이언트의 네트워크 구성(IP 주소, 게이트웨이 및 DNS 서버 포함)을 반환합니다.

14.1. DHCP를 사용하는 이유는 무엇입니까?

DHCP 는 클라이언트 네트워크 인터페이스의 자동 구성에 유용합니다. 클라이언트 시스템을 구성할 때 IP 주소, 넷마스크, 게이트웨이 또는 DNS 서버를 지정하는 대신 DHCP 를 선택할 수 있습니다. 클라이언트는 DHCP 서버에서 이 정보를 검색합니다. DHCP 는 여러 시스템의 IP 주소를 변경하려는 경우에도 유용합니다. 모든 시스템을 재구성하는 대신 새 IP 주소 집합에 대해 서버에서 하나의 구성 파일만 편집할 수 있습니다. 조직의 DNS 서버가 변경되면 DHCP 클라이언트 가 아닌 DHCP 서버에서 변경 사항이 발생합니다. 네트워크를 다시 시작하거나 클라이언트를 재부팅하면 변경 사항이 적용됩니다.

조직에 기능적인 DHCP 서버가 네트워크에 올바르게 연결되어 있는 경우 노트북과 기타 모바일 컴퓨터 사용자가 이러한 장치를 사무실로 이동할 수 있습니다.

DNS 및 DHCP 서버 관리자뿐만 아니라 프로비저닝 애플리케이션도 조직에서 사용되는 호스트 이름 형식에 동의해야 합니다. 호스트 이름 형식에 대한 자세한 내용은 6.1.1절. “권장 명명 관행” 을 참조하십시오.

14.2. DHCP 서버 구성

dhcp 패키지에는ISC(인터넷 시스템) DHCP 서버가 포함되어 있습니다. 패키지를 root 로 설치합니다. :

```
~]# yum install dhcp
```

dhcp 패키지를 설치하면 단순히 빈 구성 파일인 /etc/dhcp/dhcpd.conf 파일이 생성됩니다. root로 다음 명령을 실행합니다.

```
~]# cat /etc/dhcp/dhcpd.conf
#
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.example
#   see dhcpd.conf(5) man page
#
```

예제 구성 파일은 /usr/share/doc/dhcp-버전;/dhcpd.conf.example 에서 찾을 수 있습니다. 이 파일을 사용하여 아래에 자세히 설명된 /etc/dhcp/dhcpd.conf 를 구성하는 데 도움이 됩니다.

또한 **DHCP** 는 **/var/lib/dhcpd/dhcpd.leases** 파일을 사용하여 클라이언트 리스를 저장합니다. 자세한 내용은 **14.2.2절. "리스 데이터베이스"**을 참조하십시오.

14.2.1. 설정 파일

DHCP 서버 구성의 첫 번째 단계는 클라이언트에 대한 네트워크 정보를 저장하는 구성 파일을 생성하는 것입니다. 이 파일을 사용하여 클라이언트 시스템에 대한 옵션을 선언합니다.

구성 파일에는 더 쉬운 형식을 위해 추가 탭 또는 빈 행이 포함될 수 있습니다. 키워드는 대소문자를 구분하지 않으며 해시 기호(#)로 시작하는 행은 주석으로간주됩니다.

구성 파일에는 다음 두 가지 유형의 설명이 있습니다.

- **parameters -** 작업을 수행하는 방법, 작업 수행 여부 또는 클라이언트에 보낼 네트워크 구성 옵션을 지정합니다.

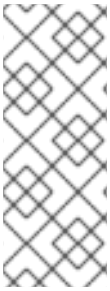
- 선언 - 네트워크의 토폴로지를 설명하고, 클라이언트를 설명하고, 클라이언트에 주소를 제공하거나, 매개 변수 그룹을 선언 그룹에 적용합니다.

키워드 옵션으로 시작하는 매개 변수를 옵션 이라고 합니다. 이러한 옵션은 **DHCP** 옵션을 제어합니다. 반면 매개 변수는 선택 사항이 아닌 값을 구성하거나 **DHCP** 서버가 작동하는 방식을 제어합니다.

중괄호({ })로 묶은 섹션 앞에 선언된 매개 변수(옵션 포함)는 전역 매개 변수로 간주됩니다. 전역 매개 변수는 아래의 모든 섹션에 적용됩니다.

중요

구성 파일이 변경되면 **systemctl restart dhcpd** 명령으로 **DHCP** 데몬을 다시 시작할 때까지 변경 사항이 적용되지 않습니다.



참고

DHCP 구성 파일을 변경하고 매번 서비스를 다시 시작하는 대신, **omshell** 명령을 사용하면 **DHCP** 서버의 구성을 연결하고 쿼리하고, 변경할 수 있는 대화식 방법을 제공합니다. **omshell** 을 사용하면 서버가 실행되는 동안 모든 변경을 수행할 수 있습니다. 자세한 내용은 **omshell** 도움말 페이지를 참조하십시오.

예 14.1. “서브넷 선언” 에서는 아래에 선언된 모든 호스트 문에 라우터,**subnet-mask,domain-search, domain -name-servers** 및 **time-offset** 옵션이 사용됩니다.

제공할 모든 서브넷과 **DHCP** 서버가 연결된 모든 서브넷에 대해 **DHCP** 데몬에 주소가 해당 서브넷에 있음을 인식하는 방법을 알려주는 하나의 서브넷 선언이 있어야 합니다. 주소를 해당 서브넷에 동적으로 할당하지 않아도 각 서브넷에 서브넷 선언이 필요합니다.

이 예에서는 서브넷의 모든 **DHCP** 클라이언트와 선언된 범위의 글로벌 옵션이 있습니다. 클라이언트에는 범위 내의 **IP** 주소가 할당됩니다.

예 14.1. 서브넷 선언

```
subnet 192.168.1.0 netmask 255.255.255.0 {
        option routers                  192.168.1.254;
        option subnet-mask              255.255.255.0;
        option domain-search            "example.com";
        option domain-name-servers      192.168.1.1;
        option time-offset              -18000;     # Eastern Standard Time
 range 192.168.1.10 192.168.1.100;
}
```

동적 **IP** 주소를 서브넷 내의 시스템으로 리스하는 **DHCP** 서버를 구성하려면 예 14.2. “범위 매개 변수” 에서 예제 값을 수정합니다. 클라이언트에 대한 기본 리스 시간, 최대 리스 시간 및 네트워크 구성 값을 선언합니다. 이 예에서는 **192.168.1.10** 및 **192.168.1. 100** 범위의 **IP** 주소를 클라이언트 시스템에 할당합니다.

예 14.2. 범위 매개 변수

```
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.254;
option domain-name-servers 192.168.1.1, 192.168.1.2;
option domain-search "example.com";
```

```
subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.10 192.168.1.100;
}
```

네트워크 인터페이스 카드의 **MAC** 주소를 기반으로 클라이언트에 **IP** 주소를 할당하려면 호스트 선언 내에서 **hardware ethernet** 매개 변수를 사용합니다. 예 14.3. “DHCP를 사용하여 고정 IP 주소” 에서 시연한 대로 호스트 **apex** 선언은 **MAC** 주소 **00:A0:78:8E:9E:AA** 가 항상 **IP** 주소 **192.168.1.4** 를 수신하도록 지정합니다.

선택적 매개 변수 **host-name** 을 사용하여 호스트 이름을 클라이언트에 할당할 수도 있습니다.

예 14.3. DHCP를 사용하여 고정 IP 주소

```
host apex {
  option host-name "apex.example.com";
  hardware ethernet 00:A0:78:8E:9E:AA;
  fixed-address 192.168.1.4;
}
```

Red Hat Enterprise Linux 7은 **InfiniBand IP oIB** 인터페이스에 고정 **IP** 주소 할당을 지원합니다. 그러나 이러한 인터페이스에는 일반 하드웨어 이더넷 주소가 없으므로 **IPoIB** 인터페이스의 고유 식별자를 지정하는 다른 방법을 사용해야 합니다. 표준에서는 옵션 **dhcp-client-identifier= construct**를 사용하여 **IPoIB** 인터페이스의 **dhcp-client-identifier** 필드를 지정하는 것입니다. **DHCP** 서버 호스트 구성은 최대 하나의 하드웨어 이더넷과 호스트 스탠자당 하나의 **dhcp-client-identifier** 항목을 지원합니다. 그러나 고정 주소 항목이 두 개 이상 있을 수 있으며 **DHCP** 서버는 **DHCP** 요청을 수신한 네트워크에 적합한 주소로 자동으로 응답합니다.

예 14.4. 여러 인터페이스에서 DHCP를 사용하여 고정 IP 주소

컴퓨터에 복잡한 구성(예: **InfiniBand** 인터페이스 두 개 및 각 물리적 인터페이스 및 이더넷 연결)의 **P_Key** 인터페이스가 있는 경우 다음 정적 **IP** 구문을 사용하여 이 구성을 제공할 수 있습니다.

```
Host apex.0 {
   option host-name “apex.example.com”;
   hardware ethernet 00:A0:78:8E:9E:AA;
   option dhcp-client-identifier=ff:00:00:00:00:00:02:00:00:02:c9:00:00:02:c9:03:00:31:7b:11;
   fixed-address 172.31.0.50,172.31.2.50,172.31.1.50,172.31.3.50;
}

host apex.1 {
   option host-name “apex.example.com”;
   hardware ethernet 00:A0:78:8E:9E:AB;
   option dhcp-client-identifier=ff:00:00:00:00:00:02:00:00:02:c9:00:00:02:c9:03:00:31:7b:12;
   fixed-address 172.31.0.50,172.31.2.50,172.31.1.50,172.31.3.50;
}
```

장치에 적합한 **dhcp-client-identifier** 를 찾으려면 일반적으로 접두사 **ff:00:00:00:00:00:00:00:00:00:02:c9:00** 을 사용하여 **IPoIB** 인터페이스의 **InfiniBand** 포트 **8**바이트 **GUID**로 표시될 수 있습니다. 일부 이전 컨트롤러에서 이 접두사가 올바르지 않습니다. 이 경우 **DHCP** 서버에서 **tcpdump** 를 사용하여 들어오는 **IPoIB DHCP** 요청을 캡처하고 해당 캡처에서 올바른 **dhcp-client-identifier** 를 수집하는 것이 좋습니다. 예를 들어 다음과 같습니다.

```
]$ tcpdump -vv -i mlx4_ib0
tcpdump: listening on mlx4_ib0, link-type LINUX_SLL (Linux cooked), capture size 65535
bytes
23:42:44.131447 IP (tos 0x10, ttl 128, id 0, offset 0, flags [none], proto UDP (17), length 328)
    0.0.0.0.bootpc > 255.255.255.255.bootps: [udp sum ok] BOOTP/DHCP, Request, length
300, htype 32, hlen 0, xid 0x975cb024, Flags [Broadcast] (0x8000)
        Vendor-rfc1048 Extensions
          Magic Cookie 0x63825363
          DHCP-Message Option 53, length 1: Discover
          Hostname Option 12, length 10: "rdma-qe-03"
          Parameter-Request Option 55, length 18:
            Subnet-Mask, BR, Time-Zone, Classless-Static-Route
            Domain-Name, Domain-Name-Server, Hostname, YD
            YS, NTP, MTU, Option 119
            Default-Gateway, Classless-Static-Route, Classless-Static-Route-Microsoft, Static-
Route
            Option 252, NTP
          Client-ID Option 61, length 20: hardware-type 255,
00:00:00:00:00:02:00:00:02:c9:00:00:02:c9:02:00:21:ac:c1
```

위의 덤프에는 **Client-ID** 필드가 표시됩니다. **hardware-type 255** 는 **ID**의 **initial ff:** 에 해당하고 나머지 **ID**는 **DHCP** 구성 파일에 표시되어야 하므로 정확히 인용됩니다.

동일한 물리적 네트워크를 공유하는 모든 서브넷은 예 14.5. “공유 네트워크 선언” 에 표시된 대로 공유 네트워크 선언 내에서 선언해야 합니다. **shared-network** 내의 매개 변수는 묶인 서브넷 선언 외부의 매개 변수는 전역 매개 변수로 간주됩니다. **shared-network** 에 할당된 이름은 테스트 랩 환경에서 모든 서브넷을 설명하는 “**test-lab**” 제목과 같이 네트워크에 대한 설명 제목이어야 합니다.

예 14.5. 공유 네트워크 선언

```
shared-network name {
   option domain-search          "test.redhat.com";
   option domain-name-servers     ns1.redhat.com, ns2.redhat.com;
   option routers                 192.168.0.254;
   #more parameters for EXAMPLE shared-network
   subnet 192.168.1.0 netmask 255.255.252.0 {
      #parameters for subnet
      range 192.168.1.1 192.168.1.254;
   }
```

```
    subnet 192.168.2.0 netmask 255.255.252.0 {
      #parameters for subnet
      range 192.168.2.1 192.168.2.254;
    }
}
```

예 14.6. "그룹 선언" 에서 시연한 대로 그룹 선언은 선언 그룹에 전역 매개 변수를 적용하는 데 사용됩니다. 예를 들어 공유 네트워크, 서브넷 및 호스트를 그룹화할 수 있습니다.

예 14.6. 그룹 선언

```
group {
  option routers                192.168.1.254;
  option subnet-mask            255.255.255.0;
  option domain-search          "example.com";
  option domain-name-servers    192.168.1.1;
  option time-offset            -18000;     # Eastern Standard Time
  host apex {
    option host-name "apex.example.com";
    hardware ethernet 00:A0:78:8E:9E:AA;
    fixed-address 192.168.1.4;
  }
  host raleigh {
    option host-name "raleigh.example.com";
    hardware ethernet 00:A1:DD:74:C3:F2;
    fixed-address 192.168.1.6;
  }
}
```

참고

*제공된 예제 구성 파일을 시작점으로 사용하고 사용자 지정 구성 옵션을 추가할 수 있습니다. 이 파일을 적절한 위치에 복사하려면 **root** 로 다음 명령을 사용합니다.*

```
~]# cp /usr/share/doc/dhcp-version_number/dhcpd.conf.example /etc/dhcp/dhcpd.conf
```

*여기서 **version_number** 는 **DHCP** 버전 번호입니다.*

*전체 옵션 문 목록과 해당 명령문의 내용은 **dhcp-options(5)** 도움말 페이지를 참조하십시오.*

14.2.2. 리스 데이터베이스

DHCP 서버에서 **/var/lib/dhcpd/dhcpd.leases** 파일은 **DHCP** 클라이언트 리스 데이터베이스를 저장합니다. 이 파일을 변경하지 마십시오. 최근에 할당된 각 **IP** 주소에 대한 **DHCP** 임대 정보는 리스 데이터베이스에 자동으로 저장됩니다. 이 정보에는 리스의 길이, **IP** 주소가 할당된 경우 리스를 위한 시작 및 종료 날짜, 리스 검색에 사용된 네트워크 인터페이스 카드의 **MAC** 주소가 포함됩니다.

lease 데이터베이스의 모든 시간은 현지 시간이 아닌 **UTC(Coordinated Universal Time)**입니다.

리스 데이터베이스는 수시로 다시 생성되므로 너무 크지 않습니다. 먼저 알려진 모든 리스가 임시 리스 데이터베이스에 저장됩니다. **dhcpd.leases** 파일의 이름이 **dhcpd.leases~** 로 바뀌고, 임시 리스가 **dhcpd.leases** 에 작성됩니다.

DHCP 데몬을 종료하거나 리스 데이터베이스 이름을 백업 파일로 변경한 후 새 파일을 작성하기 전에 시스템이 충돌할 수 있습니다. 이 경우 **dhcpd.leases** 파일이 존재하지 않지만 서비스를 시작하는 데 필요합니다. 새 리스 파일을 만들지 마십시오. 그렇게 하면 모든 오래된 리스가 손실되어 많은 문제가 발생합니다. 올바른 솔루션은 **dhcpd.leases~** 백업 파일의 이름을 **dhcpd.leases** 로 변경한 다음 데몬을 시작하는 것입니다.

14.2.3. 서버 시작 및 중지

중요

DHCP 서버가 처음 시작되면 **dhcpd.leases** 파일이 존재하지 않는 한 실패합니다. **/var/lib/dhcpd/dhcpd.leases** 명령을 사용하여 파일이 없는 경우 만들 수 있습니다. 동일한 서버가 **DNS** 서버로 **BIND**를 실행하는 경우 **named** 서비스를 시작하면 **dhcpd.leases** 파일을 자동으로 검사하므로 이 단계가 필요하지 않습니다.

이전에 실행 중인 시스템에 새 리스 파일을 만들지 마십시오. 그렇게 하면 모든 오래된 리스가 손실되어 많은 문제가 발생합니다. 올바른 솔루션은 **dhcpd.leases~** 백업 파일의 이름을 **dhcpd.leases** 로 변경한 다음 데몬을 시작하는 것입니다.

DHCP 서비스를 시작하려면 다음 명령을 사용합니다.

```
systemctl start dhcpd.service
```

DHCP 서버를 중지하려면 다음을 입력합니다.

```
systemctl stop dhcpd.service
```

기본적으로 **DHCP** 서비스는 부팅 시 시작되지 않습니다. 부팅 시 자동으로 시작되도록 데몬을 구성하는 방법에 대한 자세한 내용은 **Red Hat Enterprise Linux** 시스템 관리자 가이드 을 참조하십시오.

둘 이상의 네트워크 인터페이스가 시스템에 연결되어 있지만 **DHCP** 서버에서 인터페이스 중 하나에서 **DHCP** 요청만 수신 대기해야 하는 경우 해당 장치에서만 수신 대기하도록 **DHCP** 서버를 구성합니다. **DHCP** 데몬은 **/etc/dhcp/dhcpd.conf** 파일에서 서브넷 선언을 찾는 인터페이스에서만 수신 대기합니다.

이는 두 개의 네트워크 카드가 있는 방화벽 시스템에 유용합니다. 인터넷의 **IP** 주소를 검색하도록 하나의 네트워크 카드를 **DHCP** 클라이언트로 구성할 수 있습니다. 다른 네트워크 카드는 방화벽 뒤의 내부 네트워크의 **DHCP** 서버로 사용할 수 있습니다. 내부 네트워크에 연결된 네트워크 카드만 지정하면 사용자가 인터넷을 통해 데몬에 연결할 수 없기 때문에 시스템의 보안이 향상됩니다.

명령줄 옵션을 지정하려면 **dhcpd.service** 파일을 **root** 사용자로 복사한 다음 편집합니다. 예를 들면 다음과 같습니다.

```
~]# cp /usr/lib/systemd/system/dhcpd.service /etc/systemd/system/
~]# vi /etc/systemd/system/dhcpd.service
```

[Service] 섹션에서 행을 편집합니다.

```
ExecStart=/usr/sbin/dhcpd -f -cf /etc/dhcp/dhcpd.conf -user dhcpd -group dhcpd --no-pid
your_interface_name(s)
```

그런 다음 **root** 사용자로 서비스를 다시 시작합니다.

```
~]# systemctl --system daemon-reload
~]# systemctl restart dhcpd
```

명령줄 옵션은 섹션 **[Service]** 아래의 **/etc /systemd/system/dhcpd.service** 유닛 파일에 **ExecStart=/usr/sbin /dhcpd** 에 추가할 수 있습니다. 여기에는 다음이 포함됩니다.

- **-P portnum - dhcpd** 가 수신해야 하는 **UDP** 포트 번호를 지정합니다. 기본값은 포트 **67**입니다. **DHCP** 서버는 지정된 **UDP** 포트보다 **1** 큰 포트 번호에서 **DHCP** 클라이언트에 응답을 전송합니다. 예를 들어 기본 포트 **67**을 사용하는 경우 서버는 포트 **67**에서 요청을 수신 대기하고 포트 **68**에서 클라이언트에 응답합니다. 여기서 포트를 지정하고 **DHCP** 릴레이 에이전트를 사용하는 경우 **DHCP** 릴레이 에이전트가 수신해야 하는 포트와 동일한 포트를 지정해야 합니다. 자세한 내용은 **14.3**절. “**DHCP** 릴레이 에이전트” 을 참조하십시오.

- **-f -** 데몬을 포그라운드 프로세스로 실행합니다. 대부분은 디버깅에 사용됩니다.

- *-d - DHCP 서버 데몬을 표준 오류 설명자에 기록합니다. 대부분은 디버깅에 사용됩니다. 지정하지 않으면 로그가 /var/log/messages 에 기록됩니다.*

- *- CF filename - 구성 파일의 위치를 지정합니다. 기본 위치는 /etc/dhcp/dhcpd.conf 입니다.*

- *- LF filename - 리스 데이터베이스 파일의 위치를 지정합니다. 리스 데이터베이스 파일이 이미 있는 경우 DHCP 서버를 시작할 때마다 동일한 파일을 사용하는 것이 매우 중요합니다. 이 옵션은 프로덕션이 아닌 머신의 디버깅 목적으로만 사용하는 것이 좋습니다. 기본 위치는 /var/lib/dhcpd/dhcpd.leases 입니다.*

- *-q - 데몬을 시작할 때 전체 저작권 메시지를 출력하지 마십시오.*

14.3. DHCP 릴레이 에이전트

DHCP 릴레이 에이전트(dhcrelay)를 사용하면 DHCP 서버가 없는 서브넷의 DHCP 및 BOOTP 요청을 다른 서브넷의 하나 이상의 DHCP 서버로 릴레이할 수 있습니다.

DHCP 클라이언트에서 정보를 요청하면 DHCP 릴레이 에이전트는 DHCP 릴레이 에이전트를 시작할 때 지정된 DHCP 서버 목록으로 요청을 전달합니다. DHCP 서버에서 응답을 반환하면 원래 요청을 보낸 네트워크에서 응답이 브로드캐스트 또는 유니캐스트됩니다.

IPv4 용 DHCP 릴레이 에이전트인dhcrelay 는 INTERFACES 지시문을 사용하여 /etc/sysconfig/dhcrelay 에 인터페이스가 지정되지 않은 한 모든 인터페이스에서 DHCPv4 및 BOOTP 요청을 수신 대기합니다. 14.3.1절. “dhcrelay를 DHCPv4 및 BOOTP 릴레이 에이전트로 구성” 의 내용을 참조하십시오. IPv6 용 DHCP 릴레이 에이전트dhcrelay6 에는 DHCPv6 요청을 수신 대기하는 기본 동작 및 인터페이스가 지정되지 않아야 합니다. 14.3.2절. “dhcrelay를 DHCPv6 릴레이 에이전트로 구성” 의 내용을 참조하십시오.

dhcrelay 는 DHCPv4 및 BOOTP 릴레이 에이전트(기본적으로) 또는 DHCPv6 릴레이 에이전트(-6 인수 사용)로 실행할 수 있습니다. 사용법 메시지를 보려면 dhcrelay -h 명령을 실행합니다.

14.3.1. dhcrelay를 DHCPv4 및 BOOTP 릴레이 에이전트로 구성

DHCPv4 및 BOOTP 모드에서 dhcrelay 를 실행하려면 요청을 전달해야 하는 서버를 지정합니다. dhcrelay.service 파일을 root 사용자로 복사한 다음 편집합니다.

```
~]# cp /lib/systemd/system/dhcrelay.service /etc/systemd/system/
~]# vi /etc/systemd/system/dhcrelay.service
```

*[Service] 섹션에서 **ExecStart** 옵션을 편집하고 행 끝에 하나 이상의 서버 **IPv4** 주소를 추가합니다. 예를 들면 다음과 같습니다.*

```
ExecStart=/usr/sbin/dhcrelay -d --no-pid 192.168.1.1
```

*DHCP 릴레이 에이전트가 DHCP 요청을 수신 대기하는 인터페이스도 지정하려면 **-i** 인수(모든 인터페이스에서 수신 대기할 예정)를 사용하여 **ExecStart** 옵션에 추가합니다.*

```
ExecStart=/usr/sbin/dhcrelay -d --no-pid 192.168.1.1 -i em1
```

*다른 옵션은 **dhcrelay(8)** 도움말 페이지를 참조하십시오.*

*변경 사항을 활성화하려면 **root** 사용자로 서비스를 다시 시작합니다.*

```
~]# systemctl --system daemon-reload
~]# systemctl restart dhcrelay
```

14.3.2. dhcrelay를 DHCPv6 릴레이 에이전트로 구성

*DHCPv6 모드에서 **dhcrelay** 를 실행하려면 **-6** 인수를 추가하고 "하위 인터페이스" (클라이언트 또는 다른 릴레이 에이전트에서 수신될 쿼리)와 "상위 인터페이스" (클라이언트 및 기타 릴레이 에이전트의 쿼리가 전달되어야 하는)를 지정합니다. **dhcrelay.service** 를 **dhcrelay6.service** 에 복사하고 **root** 사용자로 편집합니다.*

```
~]# cp /lib/systemd/system/dhcrelay.service /etc/systemd/system/dhcrelay6.service
~]# vi /etc/systemd/system/dhcrelay6.service
```

*섹션 **[Service]**의 **ExecStart** 옵션을 편집하고 "하위 인터페이스와 '상위 인터페이스 인터페이스를' 추가합니다. 예를 들면" 다음과 같습니다.*

```
ExecStart=/usr/sbin/dhcrelay -d --no-pid -6 -l em1 -u em2
```

*다른 옵션은 **dhcrelay(8)** 도움말 페이지를 참조하십시오.*

변경 사항을 활성화하려면 **root** 사용자로 서비스를 다시 시작합니다.

```
~]# systemctl --system daemon-reload
~]# systemctl restart dhcrelay6
```

14.4. 멀티홈 DHCP 서버 구성

다중 홈 **DHCP** 서버는 여러 개의 서브넷인 여러 네트워크를 제공합니다. 이 섹션의 예제에서는 여러 네트워크를 제공하도록 **DHCP** 서버를 구성하고, 수신 대기할 네트워크 인터페이스를 선택하고, 네트워크를 이동하는 시스템의 네트워크 설정을 정의하는 방법을 자세히 설명합니다.

변경하기 전에 기존 **/etc/dhcp/dhcpd.conf** 파일을 백업합니다.

DHCP 데몬은 **/etc/dhcp/dhcpd.conf** 파일에서 서브넷 선언을 찾는 인터페이스에서만 수신 대기합니다.

다음은 두 개의 네트워크 인터페이스가 있는 서버용 기본 **/etc/dhcp/dhcpd.conf** 파일입니다. **enp1s0** 10.0.0.0/24 네트워크에서 **enp2s0** 172.16.0.0/24 네트워크에서. 여러 서브넷 선언을 통해 여러 네트워크에 대해 다양한 설정을 정의할 수 있습니다.

```
default-lease-time 600;
max-lease-time 7200;
subnet 10.0.0.0 netmask 255.255.255.0 {
 option subnet-mask 255.255.255.0;
 option routers 10.0.0.1;
 range 10.0.0.5 10.0.0.15;
}
subnet 172.16.0.0 netmask 255.255.255.0 {
 option subnet-mask 255.255.255.0;
 option routers 172.16.0.1;
 range 172.16.0.5 172.16.0.15;
}
```

subnet 10.0.0.0 netmask 255.255.255.0;

DHCP 서버에서 제공하는 모든 네트워크에 서브넷 선언이 필요합니다. 여러 서브넷에는 여러 서브넷 선언이 필요합니다. **DHCP** 서버에 서브넷 선언 범위에 네트워크 인터페이스가 없는 경우 **DHCP** 서버에서 해당 네트워크를 제공하지 않습니다.

서브넷 선언이 하나뿐이고 해당 서브넷 범위에 네트워크 인터페이스가 없는 경우 **DHCP** 데몬이

시작되지 않으며 /var/log/messages 에 다음과 같은 오류가 기록됩니다.

```
dhcpd: No subnet declaration for enp1s0 (0.0.0.0).
dhcpd: ** Ignoring requests on enp1s0.  If this is not what
dhcpd:    you want, please write a subnet declaration
dhcpd:    in your dhcpd.conf file for the network segment
dhcpd:    to which interface enp2s0 is attached. **
dhcpd:
dhcpd:
dhcpd: Not configured to listen on any interfaces!
```

option subnet-mask 255.255.255.0;

옵션 subnet-mask 옵션은 서브넷 마스크를 정의하고 서브넷 선언에서 넷마스크 값을 재정의합니다. 간단한 경우 서브넷 및 넷마스크 값이 동일합니다.

option routers 10.0.0.1;

옵션 Router 옵션은 서브넷의 기본 게이트웨이를 정의합니다. 시스템이 다른 서브넷의 내부 네트워크와 외부 네트워크에 연결하는 데 필요합니다.

range 10.0.0.5 10.0.0.15;

range 옵션은 사용 가능한 IP 주소 풀을 지정합니다. 시스템에는 지정된 IP 주소 범위의 주소가 할당됩니다.

자세한 내용은 dhcpd.conf(5) 도움말 페이지를 참조하십시오.

주의

DHCP 서버가 하나의 IP 범위에서 다른 물리적 이더넷 세그먼트로 IP 주소를 제공할 때 잘못된 설정을 방지하려면 추가 서브넷을 공유 네트워크 선언에 묶지 않도록 하십시오.

14.4.1. 호스트 설정

변경하기 전에 기존 /etc/sysconfig/dhcpd 및 /etc/ dhcp/dhcpd.conf 파일을 백업하십시오.

여러 네트워크에 대한 단일 시스템 구성

다음 /etc/dhcp/dhcpd.conf 예제에서는 두 개의 서브넷을 만들고 연결되는 네트워크에 따라 동일한 시스템의 IP 주소를 구성합니다.

```
default-lease-time 600;
max-lease-time 7200;
subnet 10.0.0.0 netmask 255.255.255.0 {
 option subnet-mask 255.255.255.0;
 option routers 10.0.0.1;
 range 10.0.0.5 10.0.0.15;
}
subnet 172.16.0.0 netmask 255.255.255.0 {
 option subnet-mask 255.255.255.0;
 option routers 172.16.0.1;
 range 172.16.0.5 172.16.0.15;
}
host example0 {
 hardware ethernet 00:1A:6B:6A:2E:0B;
 fixed-address 10.0.0.20;
}
host example1 {
 hardware ethernet 00:1A:6B:6A:2E:0B;
 fixed-address 172.16.0.20;
}
```

host example0

호스트 선언은 단일 시스템에 대한 특정 매개 변수(예: IP 주소)를 정의합니다. 여러 호스트에 대한 특정 매개 변수를 구성하려면 여러 호스트 선언을 사용합니다.

대부분의 DHCP 클라이언트는 호스트 선언에서 이름을 무시합니다. 따라서 이 이름은 다른 호스트 선언에 고유한 경우 모든 항목이 될 수 있습니다. 여러 네트워크에 동일한 시스템을 구성하려면 각 호스트 선언에 대해 다른 이름을 사용합니다. 그러지 않으면 DHCP 데몬이 시작되지 않습니다. 시스템은 호스트 선언의 이름이 아닌 하드웨어 이더넷 옵션으로 식별됩니다.

hardware ethernet 00:1A:6B:6A:2E:0B;

하드웨어 이더넷 옵션은 시스템을 식별합니다. 이 주소를 찾으려면 ip link 명령을 실행합니다.

fixed-address 10.0.0.20;

fixed-address 옵션은 하드웨어 이더넷 옵션에 지정된 시스템에 유효한 IP 주소를 할당합니다.

*이 주소는 범위 옵션으로 지정된 **IP** 주소 풀 외부에 있어야 합니다.*

***option** 문이 세미콜론으로 끝나지 않으면 **DHCP** 데몬이 시작되지 않으며 **/var/log/messages** 에 다음과 같은 오류가 기록됩니다.*

```
/etc/dhcp/dhcpd.conf line 20: semicolon expected.
dhcpd: }
dhcpd: ^
dhcpd: /etc/dhcp/dhcpd.conf line 38: unexpected end of file
dhcpd:
dhcpd: ^
dhcpd: Configuration file errors encountered -- exiting
```

다중 네트워크 인터페이스를 사용하여 시스템 구성

*다음 **host** 선언은 각 인터페이스가 동일한 **IP** 주소를 수신하도록 여러 네트워크 인터페이스가 있는 단일 시스템을 구성합니다. 두 네트워크 인터페이스가 동일한 네트워크에 동시에 연결된 경우 이 구성이 작동하지 않습니다.*

```
host interface0 {
 hardware ethernet 00:1a:6b:6a:2e:0b;
 fixed-address 10.0.0.18;
}
host interface1 {
 hardware ethernet 00:1A:6B:6A:27:3A;
 fixed-address 10.0.0.18;
}
```

*이 예에서 **interface0** 은 첫 번째 네트워크 인터페이스이며, **interface1** 은 두 번째 인터페이스입니다. 서로 다른 하드웨어 이더넷 옵션은 각 인터페이스를 식별합니다.*

이러한 시스템이 다른 네트워크에 연결되는 경우 호스트 선언을 더 추가하고 다음을 기억하십시오.

- *호스트가 연결되는 네트워크에 유효한 **fixed-address** 를 할당합니다.*
- *호스트 선언에서 이름을 고유하게 만듭니다.*

*호스트 선언에 지정된 이름이 고유하지 않은 경우 **DHCP** 데몬이 시작되지 않으며 다음과 같은 오류가 **/var/log/messages** 에 기록됩니다:*

```
dhcpd: /etc/dhcp/dhcpd.conf line 31: host interface0: already exists
dhcpd: }
dhcpd: ^
dhcpd: Configuration file errors encountered -- exiting
```

*이 오류는 **/etc/dhcp/dhcpd.conf** 에 여러 개의 호스트 **interface0** 선언이 정의되어 있기 때문에 발생했습니다.*

14.5. IPV6(DHCPV6)용 DHCP

*ISC DHCP 에는**DHCPv 6** 서버, 클라이언트 및 릴레이 에이전트 기능이 포함된 **4.x** 릴리스 이후 **IPv 6(DHCPv 6)** 지원이 포함됩니다. 에이전트는 **IPv4** 및 **IPv 6** 를 모두 지원하지만 에이전트는 한 번에 하나의 프로토콜만 관리할 수 있습니다. 이중 지원의 경우 **IPv4** 및 **IPv 6** 에 대해 별도로 시작해야 합니다. 예를 들어, 해당 구성 파일 **/etc/dhcp/dhcpd.conf** 및 **/etc/dhcp/dhcpd 6.conf** 를 편집하여 **DHCPv 4** 및 **DHCPv 6**을 둘 다 구성한 다음 다음 명령을 실행합니다.*

```
~]# systemctl start dhcpd
~]# systemctl start dhcpd6
```

*DHCPv6 서버 구성 파일은 **/etc/dhcp/dhcpd6.conf** 에 있습니다.*

*예제 서버 구성 파일은 **/usr/share/doc/dhcp-버전/dhcpd6.conf.example** 에서 확인할 수 있습니다.*

*간단한 **DHCPv6** 서버 구성 파일은 다음과 같습니다.*

```
subnet6 2001:db8:0:1::/64 {
      range6 2001:db8:0:1::129 2001:db8:0:1::254;
      option dhcp6.name-servers fec0:0:0:1::1;
      option dhcp6.domain-search "domain.example";
}
```

*네트워크 인터페이스 카드의 **MAC** 주소를 기반으로 고정 주소를 클라이언트에 할당하려면 **hardware ethernet** 매개변수를 사용합니다.*

```
host otherclient {
      hardware ethernet 01:00:80:a2:55:67;
      fixed-address6 3ffe:501:ffff:100::4321;
 }
```

shared-network 및 IPv6에 대한 그룹 선언의 구성 옵션은 IPV4와 동일합니다. 자세한 내용은 예 14.5. "공유 네트워크 선언" 및 예 14.6. "그룹 선언" 에 표시된 예제를 참조하십시오.

14.6. IPV6 라우터용 RADVD 데몬 구성

라우터 광고 데몬(radvd)은 IPv6 상태 비저장 자동 구성에 필요한 라우터 광고 메시지를 전송합니다. 이를 통해 사용자는 해당 주소, 설정, 라우트를 자동으로 구성하고 이러한 알림을 기반으로 기본 라우터를 선택할 수 있습니다. radvd 데몬을 구성하려면 다음을 수행합니다.

1. radvd 데몬을 설치합니다.

```
~]# sudo yum install radvd
```

2. /etc/radvd.conf 파일을 설정합니다. 예를 들어 다음과 같습니다.

```
interface enp1s0
{
AdvSendAdvert on;
MinRtrAdvInterval 30;
MaxRtrAdvInterval 100;
prefix 2001:db8:1:0::/64
{
AdvOnLink on;
AdvAutonomous on;
AdvRouterAddr off;
};

};
```

참고

라우터 알림과 함께 **DNS** 확인자를 추가로 알리려면 **/etc/radvd.conf** 파일에 **RDNSS <ip> <ip> { };** 옵션을 추가합니다. 서브넷에 대한 **DHCPv6** 서비스를 구성하려면 **AdvManagedFlag**를 **on**으로 설정하면 라우터 알림을 통해 **DHCPv6** 서비스를 사용할 수 있을 때 클라이언트가 **IPv6** 주소를 자동으로 가져올 수 있습니다. **DHCPv6** 서비스 구성에 대한 자세한 내용은 을 참조하십시오. **14.5절. "IPv6(DHCPv6)용 DHCP"**

3. **radvd** 데몬을 활성화합니다.

```
~]# sudo systemctl enable radvd.service
```

4. **radvd** 데몬을 즉시 시작합니다.

```
~]# sudo systemctl start radvd.service
```

라우터 광고 패키지의 내용과 **radvd** 데몬에서 보낸 구성 값을 표시하려면 **radvd ump** 명령을 사용합니다.

```
~]# radvdump
Router advertisement from fe80::280:c8ff:feb9:cef9 (hoplimit 255)
        AdvCurHopLimit: 64
        AdvManagedFlag: off
        AdvOtherConfigFlag: off
        AdvHomeAgentFlag: off
        AdvReachableTime: 0
        AdvRetransTimer: 0
        Prefix 2002:0102:0304:f101::/64
                AdvValidLifetime: 30
                AdvPreferredLifetime: 20
                AdvOnLink: off
                AdvAutonomous: on
                AdvRouterAddr: on
        Prefix 2001:0db8:100:f101::/64
                AdvValidLifetime: 2592000
                AdvPreferredLifetime: 604800
                AdvOnLink: on
```

```
        AdvAutonomous: on
        AdvRouterAddr: on
    AdvSourceLLAddress: 00 80 12 34 56 78
```

radvd 데몬에 대한 자세한 내용은 radvd (8), radvd .conf(5), radvdump(8) 도움말 페이지를 참조하십시오.

14.7. DHCPV6와 RADVD 비교

IPv4의 동적 호스트 구성은 DHCPv4와 함께 주로 적용됩니다. 그러나 IPv6의 경우 다음 옵션을 사용할 수 있습니다.

- *수동*
- *radvd 데몬 사용*
- *DHCPv6 서버 사용*

수동

수동 주소 지정은 항상 사용할 수 있습니다. 3.3.6절. "nmcli를 사용하여 네트워크에 연결", 7.2절. "텍스트 사용자 인터페이스 nmtui를 사용하여 본딩 구성", 3.6절. "ip 명령을 사용하여 IP 네트워킹 구성" 에 설명된 도구를 사용하여 시스템에 IPv6 주소를 할당할 수 있습니다.

radvd 데몬 사용

표준 규격 IPv6 네트워크는 라우터 알림을 제공해야 하므로 IPv6 구성 옵션을 라우터 광고 데몬 (radvd)을 실행 중인 IPv6 구성 옵션을 적용할 수 있습니다. 라우터 알림은 실제 LAN에서 실제로 사용 가능한 접두사에 대한 링크 정보를 제공합니다. 라우터 알림 상단에는 라우터 알림을 통해 수동 IPv6 구성, 라우터 알림을 통해 자동 IPv6 구성 또는 DHCPv6)(Dynamic Host Configuration Protocol)를 선택할 수 있습니다. radvd 데몬 구성에 대한 자세한 내용은 14.6절. "IPv6 라우터용 radvd 데몬 구성" 을 참조하십시오.

DHCPv6 서버 사용

주소 관리가 중앙 집중식으로 관리되는 경우 사용자는 DHCPv6 서버를 설정할 수 있습니다. DHCPv6의 가용성은 라우터 광고 패킷의 플래그로 발표됩니다.

표 14.1. DHCPv6와 radvd 비교

DHCPv6	radvd
개인 정보 보호를 위한 임의 주소 보장.	기본 게이트웨이에 대한 정보를 제공합니다.
추가 네트워크 구성 옵션을 클라이언트에 보냅니다. 예를 들어 **NTP(Network Time Protocol)** 서버, **SIP(Session Initiation Protocol)** 서버, **iPXE(Preboot Execution Environment)** 구성입니다.	
MAC 주소를 **IPv6** 주소에 매핑합니다.	

참고

네트워크를 올바르게 구성하려면 라우터 알림만 기본 게이트웨이에 정보를 제공하므로 **radvd**와 함께 **DHCPv6**을 사용합니다.

14.8. 추가 리소스

- **dhcpd(8)** 도움말 페이지 - **DHCP** 데몬이 작동하는 방식을 설명합니다.
- **dhcpd.conf(5)** 도움말 페이지 - **DHCP** 구성 파일을 구성하는 방법을 설명합니다. 몇 가지 예제가 포함되어 있습니다.
- **dhcpd.leases(5)** 도움말 페이지 - 리스의 영구 데이터베이스를 설명합니다.
- **DHCP-options(5)** 도움말 페이지 - **dhcpd.conf**에서 **DHCP** 옵션을 선언하는 구문을 설명합니다. 몇 가지 예제가 포함되어 있습니다.
- **dhcrelay(8)** 도움말 페이지 - **DHCP** 릴레이 에이전트와 해당 구성 옵션을 설명합니다.
- **/usr/share/doc/dhcp-**버전**/** - 현재 **DHCP** 서비스 버전에 대한 예제 파일, **README** 파일 및 릴리스 노트를 포함합니다.

15장. DNS 서버

DNS (Domain Name System)는 호스트 이름을 해당 IP 주소와 연결하는 데 사용되는 분산 데이터베이스 시스템입니다. 사용자의 경우 네트워크상의 시스템을 숫자 네트워크 주소보다 일반적으로 기억하기 쉬운 이름으로 참조할 수 있다는 이점이 있습니다. 이름 서버라고도 하는 DNS 서버를 사용하면 이름 기반 쿼리에 영향을 주지 않고 호스트의 IP 주소를 변경할 수 있습니다. DNS 데이터베이스를 사용하는 것은 도메인 이름에 대한 IP 주소를 확인하는 것뿐만 아니라 DNSSEC가 배포됨에 따라 그 사용이 보다 광범위하게 사용되고 있습니다.

15.1. DNS 소개

DNS 는 일반적으로 특정 도메인에 대해 권한이 있는 하나 이상의 중앙 집중식 서버를 사용하여 구현됩니다. 클라이언트 호스트가 이름 서버에서 정보를 요청하면 일반적으로 포트 53에 연결됩니다. 그런 다음 이름 서버에서 요청된 이름을 확인하려고 시도합니다. 이름 서버가 재귀적 이름 서버로 구성되어 있고 신뢰할 수 있는 답변이 없거나 이전 쿼리에서 캐시된 응답이 없는 경우 다른 이름 서버를 쿼리하여 루트 이름 서버 라고 하는 이름 서버를 쿼리한 다음 요청된 이름을 가져오도록 쿼리합니다. 재귀가 비활성화된 상태에서만 권한 있는 것으로 구성된 이름 서버는 클라이언트를 대신하여 조회를 수행하지 않습니다.

15.1.1. 이름 서버 영역

DNS 서버에서 모든 정보는RR(리소스 레코드)이라는 기본 데이터 요소에 저장됩니다. 리소스 레코드는 RFC 1034 에 정의됩니다. 도메인 이름은 트리 구조로 구성됩니다. 계층 구조의 각 수준은 마침표(.)로 나뉩니다. 예를 들어 다음과 같습니다. 로 표시된 루트 도메인 은 DNS 트리의 루트이며, 수준 0에 있습니다. 최상위 도메인(TLD)이라고 하는 도메인 이름 com 은 계층 구조의 첫 번째 수준 이므로 루트 도메인(.)의 하위입니다. 도메인 이름 example.com 은 계층 구조의 두 번째 수준에 있습니다.

> 예 15.1. 간단한 리소스 레코드
>
> 간단한 리소스 레코드 (RR)의 예.
>
> ```
> example.com.      86400   IN      A       192.0.2.1
> ```
>
> 도메인 이름 example.com 은 RR의 소유자 입니다. 86400 값은TTL(Time to live)입니다. 문자 IN “(인터넷 시스템”)은 RR 클래스 를 나타냅니다. 문자 A 는 RR의 유형을 나타냅니다(이 예에서 호스트 주소). 호스트 주소 192.0.2.1 은 이 RR의 마지막 섹션에 포함된 데이터입니다. 이 한 줄 예제는 RR입니다. 동일한 유형, 소유자 및 클래스가 있는 RR 집합을 리소스 레코드 집합 (RRSet)이라고 합니다.

영역은 영역 파일을 사용하여 권한 있는 이름 서버에 정의되며 각 영역 의 리소스 레코드 정의가 포함됩니다. 영역 파일은 기본 이름 서버 (마스터 이름 서버라고도 함)에 저장되며, 여기에서 변경 사항이 파일의 변경 사항 및 기본 이름 서버에서 영역 정의를 수신하는 보조 이름 서버(컨트롤러 이름 서버라고도 함)에 저장됩니다. 기본 이름 서버와 보조 이름 서버는 둘 다 영역에 대해 권한이 있으며 클라이언트와 동일

하게 보입니다. 구성에 따라 모든 이름 서버는 동시에 여러 영역에 대한 기본 또는 보조 서버로도 사용할 수 있습니다.

DNS 및 **DHCP** 서버 관리자뿐만 아니라 프로비저닝 애플리케이션도 조직에서 사용되는 호스트 이름 형식에 동의해야 합니다. 호스트 이름 형식에 대한 자세한 내용은 **6.1.1**절. "권장 명명 관행" 을 참조하십시오.

15.1.2. 이름 서버 유형

이름 서버 구성 유형은 다음 두 가지가 있습니다.

권한

권한 있는 이름 서버는 해당 영역에만 포함된 리소스 레코드에 응답합니다. 이 범주에는 기본(마스터) 및 보조(슬레이브) 이름 서버가 모두 포함됩니다.

재귀

재귀적 이름 서버에서는 확인 서비스를 제공하지만 어떤 영역에 대해 권한이 없습니다. 모든 해상도에 대한 답변은 검색된 리소스 레코드로 지정된 고정된 기간 동안 메모리에 캐시됩니다.

이름 서버는 권한 있고 재귀적일 수 있지만 구성 유형을 결합하지 않는 것이 좋습니다. 작업을 수행하려면 항상 권한이 있는 서버를 모든 클라이언트에서 사용할 수 있어야 합니다. 반면, 재귀 조회는 신뢰할 수 있는 응답보다 훨씬 시간이 걸리기 때문에 제한된 수의 클라이언트에서만 재귀적 서버를 사용할 수 있어야 합니다. 그렇지 않으면 **DDoS**(서비스 거부) 공격을 받을 가능성이 높습니다.

15.1.3. Name 서버로서의 BIND

BIND는 **DNS** 관련 프로그램으로 구성됩니다. 이라는 이름 서버, **rndc** 라는 관리 유틸리티, **dig** 라는 디버깅 툴이 포함되어 있습니다. **Red Hat Enterprise Linux**에서 서비스 실행 방법에 대한 자세한 내용은 **Red Hat Enterprise Linux** 시스템 관리자 를 참조하십시오.

15.2. BIND

이 섹션에서는 **Red Hat Enterprise Linux**에 포함된 **DNS** 서버인 **BIND (Berkeley Internet Name Domain)**에 대해 설명합니다. 구성 파일의 구조에 초점을 맞추고 로컬 및 원격으로 관리하는 방법을 설명합니다.

15.2.1. 빈 영역

BIND 에서는 재귀적 서버가 처리할 수 없는 인터넷 서버에 불필요한 쿼리를 보내지 않도록 다수의 "빈 영역을" 구성합니다(예: 지연을 생성하고 이를 쿼리하는 클라이언트에 SERVFAIL 응답을 생성). 이러한 빈 영역은 즉시 및 신뢰할 수 있는 NXDOMAIN 응답이 대신 반환됩니다. 설정 옵션 empty-zones-enable 은 빈 영역이 생성되는지 여부를 제어하지만, 옵션을 disable-empty-zone 을 사용하여 사용할 기본 접두사 목록에서 하나 이상의 빈 영역을 비활성화할 수 있습니다.

RFC 1918 접두사에 대해 생성된 빈 영역 수가 증가했으며 BIND 9.9 이상의 사용자는 empty-zones-enable 이 지정되지 않은 경우(기본값: yes) 및 명시적으로 yes 로 설정된 경우 RFC 1918 빈 영역을 볼 수 있습니다.

15.2.2. 이름이 지정된 서비스 구성

명명된 서비스가 시작되면 표 15.1. "명명된 서비스 구성 파일" 에 설명된 대로 파일에서 구성을 읽습니다.

표 15.1. 명명된 서비스 구성 파일

경로	Description
/etc/named.conf	기본 구성 파일.
/etc/named/	기본 구성 파일에 포함된 구성 파일의 보조 디렉토리입니다.

구성 파일은 중괄호를 열고 닫는 등 중첩된 옵션이 포함된 문 컬렉션으로 구성됩니다({ 및 }). 파일을 편집할 때 구문 오류가 발생하지 않도록 주의해야 합니다. 그렇지 않으면 named 서비스가 시작되지 않습니다. 일반적인 /etc/named.conf 파일은 다음과 같이 구성됩니다.

```
statement-1 ["statement-1-name"] [statement-1-class] {
  option-1;
  option-2;
  option-N;
};
statement-2 ["statement-2-name"] [statement-2-class] {
  option-1;
  option-2;
  option-N;
};
statement-N ["statement-N-name"] [statement-N-class] {
  option-1;
  option-2;
  option-N;
};
```

참고

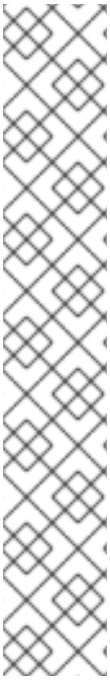
bind-chroot 패키지를 설치한 경우 BIND 서비스가 chroot 환경에서 실행됩니다. 이 경우 초기화 스크립트는 mount --bind 명령을 사용하여 이 환경 외부의 구성을 관리할 수 있도록 위의 구성 파일을 마운트합니다. 자동으로 마운트되므로 아무 것도 /var/named/chroot/ 디렉토리에 복사할 필요가 없습니다. 이렇게 하면 chroot 환경에서 실행되는 경우 BIND 구성 파일을 특별하게 관리할 필요가 없으므로 유지 관리가 간소화됩니다. chroot 환경에서 실행되지 않는 BIND 와 함께 모든 것을 구성할 수 있습니다.

/var/named/chroot/ 아래의 해당 마운트 지점 디렉토리가 비어 있으면 다음 디렉토리가 /var/named/chroot/ 디렉토리에 자동으로 마운트됩니다.

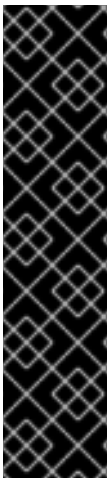
- */etc/named*
- */etc/pki/dnssec-keys*
- */run/named*
- */var/named*
- */usr/lib64/bind 또는 /usr/lib/bind (아키텍처 종속).*

대상 파일이 /var/named/chroot/ 에 없는 경우 다음 파일도 마운트됩니다:

- */etc/named.conf*
- */etc/rndc.conf*
- */etc/rndc.key*



- */etc/named.rfc1912.zones*
- */etc/named.dnssec.keys*
- */etc/named.iscdlv.key*
- */etc/named.root.key*



중요

chroot 환경에 마운트된 파일을 편집하려면 백업 복사본을 만든 다음 원본 파일을 편집해야 합니다. 또는 "edit-a-copy" 모드가 비활성화된 편집기를 사용합니다. 예를 들어 chroot 환경에서 실행되는 동안 Vim을 사용하여 BIND의 구성 파일 /etc/named.conf 를 편집하려면 root 로 다음 명령을 실행합니다.

```
~]# vim -c "set backupcopy=yes" /etc/named.conf
```

15.2.2.1. chroot 환경에 BIND 설치

chroot 환경에서 실행되도록 BIND 를 설치하려면 root 로 다음 명령을 실행합니다.

```
~]# yum install bind-chroot
```

named-chroot 서비스를 활성화하려면 먼저 다음 명령을 실행하여 명명된 서비스가 실행 중인지 확인합니다.

```
~]$ systemctl status named
```

실행 중인 경우 비활성화해야 합니다.

이름을 비활성화하려면 **root** 로 다음 명령을 실행합니다:

```
~]# systemctl stop named
```

```
~]# systemctl disable named
```

그런 다음 **named-chroot** 서비스를 활성화하려면 **root** 로 다음 명령을 실행합니다.

```
~]# systemctl enable named-chroot
```

```
~]# systemctl start named-chroot
```

named-chroot 서비스의 상태를 확인하려면 **root** 로 다음 명령을 실행합니다:

```
~]# systemctl status named-chroot
```

15.2.2.2. 일반적인 설명 유형

다음 유형의 설명은 **/etc/named.conf** 에서 일반적으로 사용됩니다:

acl

acl (Access Control List) 문을 사용하면 이름 서버에 대한 액세스를 허용하거나 거부할 수 있도록 호스트 그룹을 정의할 수 있습니다. 다음 형식을 취합니다.

```
acl acl-name {
  match-element;
  ...
};
```

acl-name 문 이름은 액세스 제어 목록의 이름이며 **match-element** 옵션은 일반적으로 개별 **IP** 주소(예: **10.0.1.1**) 또는**CIDR(Classless Inter-Domain Routing)** 네트워크 표기법(예: **10.0.1.0/24**)입

니다. 이미 정의된 키워드 목록은 표 15.2. "사전 정의된 액세스 제어 목록" 를 참조하십시오.

표 15.2. 사전 정의된 액세스 제어 목록

키워드	Description
Any	모든 IP 주소와 일치합니다.
localhost	로컬 시스템에서 사용 중인 모든 IP 주소와 일치합니다.
localnets	로컬 시스템이 연결된 모든 네트워크의 IP 주소와 일치합니다.
none	IP 주소와 일치하지 않습니다.

acl 문은 옵션과 같은 다른 문과 함께 특히 유용할 수 있습니다.예 15.2. "Conjunction with Options에서 acl 사용" 는 두 가지 액세스 제어 목록인 **Black-hats** 및 **red-hats** 를 정의하고 차단 목록에 차단형을 추가하고 빨간색**-hats**의 일반 액세스를 부여합니다.

예 15.2. Conjunction with Options에서 acl 사용

```
acl black-hats {
  10.0.2.0/24;
  192.168.0.0/24;
  1234:5678::9abc/24;
};
acl red-hats {
  10.0.1.0/24;
};
options {
  blackhole { black-hats; };
  allow-query { red-hats; };
  allow-query-cache { red-hats; };
};
```

include

include 문을 사용하면 **/etc/named.conf** 에 파일을 포함할 수 있으므로 잠재적으로 민감한 데이터를 제한된 권한이 있는 별도의 파일에 배치할 수 있습니다. 다음 형식을 취합니다.

```
include "file-name"
```

file-name 문 이름은 파일의 절대 경로입니다.

예 15.3. /etc/named.conf에 파일 포함

```
include "/etc/named.rfc1912.zones";
```

옵션

options 문을 사용하면 글로벌 서버 구성 옵션을 정의하고 다른 문에 대한 기본값을 설정할 수 있습니다. 지정된 작업 디렉터리의 위치, 허용된 쿼리 유형 등을 지정하는 데 사용할 수 있습니다. 다음 형식을 취합니다.

```
options {
  option;
  ...
};
```

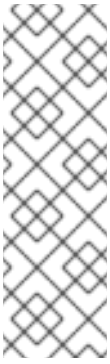
자주 사용되는 옵션 지시문 목록은 아래의 표 15.3. “일반적으로 사용되는 구성 옵션”를 참조하십시오.

표 15.3. 일반적으로 사용되는 구성 옵션

옵션	Description
allow-query	권한 있는 리소스 레코드에 대한 이름 서버를 쿼리할 수 있는 호스트를 지정합니다. **CIDR** 표기법의 액세스 제어 목록, **IP** 주소 컬렉션 또는 네트워크를 허용합니다. 모든 호스트는 기본적으로 허용됩니다.
allow-query-cache	재귀 쿼리와 같이 신뢰할 수 없는 데이터에 대해 이름 서버를 쿼리할 수 있는 호스트를 지정합니다. 기본적으로 **localhost** 및 **localnet**만 허용됩니다.
blackhole	이름 서버를 쿼리할 수 없는 호스트를 지정합니다. 이 옵션은 특정 호스트 또는 네트워크가 서버에 요청으로 급증할 때 사용해야 합니다. 기본 옵션은 **none** 입니다.
디렉토리	**named** 서비스의 작업 디렉터리를 지정합니다. 기본 옵션은 **/var/named/** 입니다.
disable-empty-zone	사용되는 기본 접두사 목록에서 하나 이상의 빈 영역을 비활성화하는 데 사용됩니다. **options** 문 및 뷰 문에서도 지정할 수 있습니다. 여러 번 사용할 수 있습니다.
dnssec-enable	**DNSSEC** 관련 리소스 레코드를 반환할지 여부를 지정합니다. 기본 옵션은 **yes** 입니다.

옵션	Description
dnssec-validation	**DNSSEC**를 통해 리소스 레코드를 인증할지 여부를 지정합니다. 기본 옵션은 **yes** 입니다.
empty-zones-enable	빈 영역이 생성되는지 여부를 제어합니다. **options** 문에서만 지정할 수 있습니다.
forwarders	확인을 위해 요청을 전달해야 하는 이름 서버의 유효한 **IP** 주소 목록을 지정합니다.
forward	**forwarders** 지시문의 동작을 지정합니다. 다음 옵션을 허용합니다. • **first** - 서버에서 이름 자체 확인을 시도하기 전에 **forwarders** 지시문에 나열된 이름 서버를 쿼리합니다. • **only** - **forwarders** 지시문에 나열된 이름 서버를 쿼리할 수 없으면 서버에서 이름 자체 확인을 시도하지 않습니다.
listen-on	쿼리를 수신 대기할 **IPv4** 네트워크 인터페이스를 지정합니다. 게이트웨이 역할을 하는 **DNS** 서버에서는 이 옵션을 사용하여 단일 네트워크에서만 발생하는 쿼리에 응답할 수 있습니다. 모든 **IPv4** 인터페이스는 기본적으로 사용됩니다.
listen-on-v6	쿼리를 수신 대기할 **IPv6** 네트워크 인터페이스를 지정합니다. 게이트웨이 역할을 하는 **DNS** 서버에서는 이 옵션을 사용하여 단일 네트워크에서만 발생하는 쿼리에 응답할 수 있습니다. 모든 **IPv6** 인터페이스는 기본적으로 사용됩니다.
max-cache-size	서버 캐시에 사용할 최대 메모리 양을 지정합니다. 제한에 도달하면 서버에서 레코드가 조기 만료되도록 하여 제한이 초과되지 않도록 합니다. 여러 뷰가 있는 서버에서 제한은 각 뷰의 캐시에 별도로 적용됩니다. 기본 옵션은 **32M** 입니다.

옵션	Description
알림	영역이 업데이트될 때 보조 이름 서버에 알릴지 여부를 지정합니다. 다음 옵션을 허용합니다. • 예 - 서버는 모든 보조 이름 서버에 알립니다. • 아니요 - 서버가 보조 이름 서버에 알리지 않습니다. • master-only - 서버가 영역에 대해 기본 서버에만 알립니다. • 명시적 - 서버는 zone 문 내에 지정된 보조 서버만 알립니다.
pid-file	named 서비스에서 생성한 프로세스 ID 파일의 위치를 지정합니다.
재귀	재귀적 서버로 사용할지 여부를 지정합니다. 기본 옵션은 yes 입니다.
statistics-file	통계 파일의 대체 위치를 지정합니다. 기본적으로 /var/named/named.stats 파일이 사용됩니다.



참고

에서 런타임 데이터에 대해 명명된 디렉터리가 BIND 기본 위치인 /var/run/named/ 에서 새 위치 /run/named/ 로 이동되었습니다. 그 결과 PID 파일이 기본 위치 /var/run/named/named.pid 에서 새 위치 /run/named/named.pid 로 이동되었습니다. 또한 session-key 파일이 /run/named/session.key 로 이동되었습니다. 이러한 위치는 options 섹션의 문에 의해 지정해야 합니다. 예 15.4. “옵션 문 사용” 의 내용을 참조하십시오.

중요

DDoS(분산 서비스 거부) 공격을 방지하려면 **allow-query-cache** 옵션을 사용하여 클라이언트의 특정 하위 집합에 대해서만 재귀적 **DNS** 서비스를 제한하는 것이 좋습니다.

사용 가능한 옵션의 전체 목록은 15.2.8.1절. "설치된 문서" 에서 참조한 **BIND 9** 관리자 참조 설명서와 **named.conf** 도움말 페이지를 참조하십시오.

예 15.4. 옵션 문 사용

```
options {
  allow-query       { localhost; };
  listen-on port    53 { 127.0.0.1; };
  listen-on-v6 port 53 { ::1; };
  max-cache-size    256M;
  directory         "/var/named";
  statistics-file   "/var/named/data/named_stats.txt";

  recursion         yes;
  dnssec-enable     yes;
  dnssec-validation yes;

  pid-file          "/run/named/named.pid";
  session-keyfile   "/run/named/session.key";
};
```

영역

zone 문을 사용하면 구성 파일 및 영역별 옵션의 위치와 같은 영역의 특성을 정의할 수 있으며 글로벌 옵션 문을 재정의하는 데 사용할 수 있습니다. 다음 형식을 취합니다.

```
zone zone-name [zone-class] {
  option;
  ...
};
```

zone-name 특성은 영역 이름이고 **zone-class** 는 영역의 선택적 클래스이며 옵션은 표 15.4. "일반적으로 영역 문에 사용되는 옵션" 에 설명된 **zone** 문 옵션입니다.

zone-name 특성은 **/var/named/** 디렉터리에 있는 해당 영역 파일 내에서 사용되는 **$ORIGIN** 지시문에 할당된 기본값이므로 특히 중요합니다. 명명된 데몬은 영역 파일에 나열된 정규화된 도메인 이름에 영역 이름을 추가합니다. 예를 들어 **zone** 문에서 **example .com** 의 네임스페이스를 정의하는 경

우 **example.com**이 **example.com** 영역 파일 내에 있는 호스트 이름의 끝에 배치되도록 **example.com** 을 **zone-name** 으로 사용합니다.

영역 파일에 대한 자세한 내용은 15.2.3절. "영역 파일 편집" 을 참조하십시오.

표 **15.4.** 일반적으로 영역 문에 사용되는 옵션

옵션	Description
allow-query	이 영역에 대한 정보를 요청할 수 있는 클라이언트를 지정합니다. 이 옵션은 글로벌 **allow-query** 옵션을 재정의합니다. 모든 쿼리 요청은 기본적으로 허용됩니다.
allow-transfer	영역의 정보 전송을 요청할 수 있는 보조 서버를 지정합니다. 모든 전송 요청은 기본적으로 허용됩니다.
allow-update	영역의 정보를 동적으로 업데이트할 수 있는 호스트를 지정합니다. 기본 옵션은 모든 동적 업데이트 요청을 거부하는 것입니다. 호스트에서 해당 영역에 대한 정보를 업데이트할 수 있도록 하는 경우 주의해야 합니다. 서버가 신뢰할 수 있는 네트워크에 있지 않은 한 이 옵션에서 **IP** 주소를 설정하지 마십시오. 대신 15.2.6.3절. "트랜잭션 SIGnatures(TSIG)" 에 설명된 **TSIG** 키를 사용합니다.
파일	영역의 구성 데이터를 포함하는 명명된 작업 디렉터리의 파일 이름을 지정합니다.
masters	권한 있는 영역 정보를 요청할 **IP** 주소를 지정합니다. 이 옵션은 영역이 유형 슬레이브 로 정의된 경우에만 사용됩니다.

옵션	Description
알림	영역이 업데이트될 때 보조 이름 서버에 알릴지 여부를 지정합니다. 다음 옵션을 허용합니다. • 예 - 서버는 모든 보조 이름 서버에 알립니다. • 아니요 - 서버가 보조 이름 서버에 알리지 않습니다. • master-only - 서버가 영역에 대해 기본 서버에만 알립니다. • 명시적 - 서버는 zone 문 내에 지정된 보조 서버만 알립니다.

옵션	Description
type	영역 유형을 지정합니다. 다음 옵션을 허용합니다. • 위임 전용 - COM, NET 또는 ORG와 같은 인프라 영역의 위임 상태를 적용합니다. 명시적 또는 암시적 위임 없이 수신되는 모든 응답은 NXDOMAIN 으로 처리됩니다. 이 옵션은 재귀 또는 캐싱 구현에 사용되는 TLD(Top-Level Domain) 또는 루트 영역 파일에서만 적용할 수 있습니다. • forward - 이 영역에 대한 모든 정보 요청을 다른 이름 서버로 전달합니다. • 힌트 - 영역을 별도로 알 수 없을 때 쿼리를 확인하는 루트 이름 서버를 가리키는 데 사용되는 특수 영역 유형입니다. 힌트 영역을 사용하여 기본값 이외의 구성이 필요하지 않습니다. • master - 이 영역에 대한 인가로 이름 서버를 지정합니다. 영역의 구성 파일이 시스템에 있는 경우 영역을 마스터 로 설정해야 합니다. • slave - 이름 서버를 이 영역의 보조 서버로 설계합니다. 기본 서버는 masters 지시문에 지정되어 있습니다.

기본 또는 보조 이름 서버의 /etc/named.conf 파일에 대한 대부분의 변경 사항에는 영역 명령문 추가, 수정 또는 삭제가 포함되며 일반적으로 이름 서버가 효율적으로 작동하려면 영역 명령문 옵션의 작은 하위 집합만 필요합니다.

예 15.5. "기본 이름 서버에 대한 영역 설명" 에서 영역이 example.com 으로 식별되고 유형은 master 로 설정되며 named 서비스는 /var/named/example.com.zone 파일을 읽도록 지시됩니다. 또한 보조 이름 서버(192.168.0.2)만 영역을 전송할 수있습니다.

예 15.5. 기본 이름 서버에 대한 영역 설명

```
zone "example.com" IN {
  type master;
  file "example.com.zone";
  allow-transfer { 192.168.0.2; };
};
```

보조 서버의 **zone** 문은 약간 다릅니다. 유형은 슬레이브 로 설정되며 **masters** 지시문은 기본 서버의 **IP** 주소 이름을 알리는 것입니다.

예 15.6. "보조 이름 서버에 대한 영역 설명" 에서 명명된 서비스는 **example.com** 영역에 대한 정보를 위해 **192.168.0.1 IP** 주소에 있는 기본 서버를 쿼리하도록 구성됩니다. 그런 다음, 수신된 정보는 **/var/named/slaves/example.com.zone** 파일에 저장됩니다. **/var/named/slaves/** 디렉터리의 모든 보조 영역을 배치해야 합니다. 그러지 않으면 서비스가 영역을 전송하지 못합니다.

예 15.6. 보조 이름 서버에 대한 영역 설명

```
zone "example.com" {
  type slave;
  file "slaves/example.com.zone";
  masters { 192.168.0.1; };
};
```

15.2.2.3. 기타 설명 유형

다음 유형의 설명은 **/etc/named.conf** 에서 덜 일반적으로 사용됩니다:

컨트롤

controls 문을 사용하면 **rndc** 명령을 사용하여 명명된 서비스를 관리하는 데 필요한 다양한 보안 요구 사항을 구성할 수 있습니다.

rndc 유틸리티 및 사용에 대한 자세한 내용은 *15.2.4절. "rndc 유틸리티 사용"* 을 참조하십시오.

키

key 문을 사용하면 이름으로 특정 키를 정의할 수 있습니다. 키는 보안 업데이트 또는 **rndc** 명령 사용과 같은 다양한 작업을 인증하는 데 사용됩니다. 두 가지 옵션이 키와 함께 사용됩니다 :

- *algorithm-name - 사용할 알고리즘 유형(예: hmac-md5).*

- *secret "key-value" - 암호화된 키 키.*

rndc 유틸리티 및 사용에 대한 자세한 내용은 15.2.4절. “rndc 유틸리티 사용” 을 참조하십시오.

로깅

logging 문을 사용하면 여러 유형의 로그를 사용할 수 있으므로 채널 이라고 합니다. 문 내에서 channel 옵션을 사용하면 자체 파일 이름(파일), 크기 제한(크기), 버전 번호(버전),중요도(심각도)를 사용하여 사용자 지정 유형의 로그를 구성할 수 있습니다. 사용자 지정 채널이 정의되면 category 옵션을 사용하여 채널을 분류하고 named 서비스가 다시 시작될 때 로깅을 시작합니다.

기본적으로 named 는 /var/log/messages 에 파일을 배치하는 rsyslog 데몬에 표준 메시지를 보냅니다. 몇 가지 표준 채널은 default_syslog(정보 로깅 메시지 처리) 및 default_ debug (디버그 메시지 처리)와 같은 다양한 심각도 수준으로 BIND에 빌드됩니다. default 라는 기본 카테고리는 기본 제공 채널을 사용하여 특별한 구성 없이 일반 로깅을 수행합니다.

로깅 프로세스 사용자 지정은 매우 자세한 프로세스일 수 있으며 이 장의 범위를 벗어납니다. 사용자 지정 BIND 로그 생성에 대한 자세한 내용은 15.2.8.1절. “설치된 문서” 에서 참조한 BIND 9 관리자 참조 설명서를 참조하십시오.

server

server 문을 사용하면 named 서비스가 특히 알림 및 영역 전송과 관련하여 원격 이름 서버에 응답하는 방법에 영향을 주는 옵션을 지정할 수 있습니다.

transfer-format 옵션은 각 메시지와 함께 전송되는 리소스 레코드 수를 제어합니다. 하나의 응답 (리소스 레코드 1개만) 또는 여러 답변(여러 리소스 레코드)일 수 있습니다. 많은 답변 옵션이 더 효율적이지만 이전 버전의 BIND에서는 지원되지 않습니다.

trusted-keys

trusted-keys 문을 사용하면 보안 DNS (DNSSEC)에 사용되는 정렬된 공개 키를 지정할 수 있습니다. 이 항목에 대한 자세한 내용은 15.2.6.4절. “DNSSEC(DNS 보안 확장)” 을 참조하십시오.

보기

view 문을 사용하면 이름 서버가 있는 네트워크에 따라 특수 보기를 생성할 수 있습니다. 이를 통해 일부 호스트는 영역과 관련된 하나의 답변을 받고 다른 호스트는 완전히 다른 정보를 수신할 수 있습니다. 또는 특정 영역은 신뢰할 수 있는 특정 호스트에서만 사용할 수 있지만 신뢰할 수 없는 호스트는 다른 영역에 대한 쿼리만 수행할 수 있습니다.

이름이 고유하면 여러 뷰를 사용할 수 있습니다. match-clients 옵션을 사용하면 특정 보기에 적용되는 IP 주소를 지정할 수 있습니다. options 문이 보기 내에서 사용되는 경우 이미 구성된 글로벌 옵션을 재정의합니다. 마지막으로 대부분의 뷰 문에는 match-clients 목록에 적용되는 여러 개의 zone 문이 포함되어 있습니다.

특정 클라이언트의 IP 주소와 일치하는 첫 번째 문이 사용되므로 view 문이 나열되는 순서가 중요합니다. 이 항목에 대한 자세한 내용은 15.2.6.1절. "다중 보기" 을 참조하십시오.

15.2.2.4. 코멘트 태그

또한 문에 /etc/named.conf 파일에 주석을 포함할 수도 있습니다. 주석은 명명된 서비스에서 무시되지만 사용자에게 추가 정보를 제공할 때 유용할 수 있습니다. 다음은 유효한 주석 태그입니다.

//

행 끝부분까지 TPM 문자 뒤에 있는 텍스트는 주석으로 간주됩니다. 예를 들어 다음과 같습니다.

```
notify yes;  // notify all secondary nameservers
```

#

행 끝에 # 문자 뒤에 있는 텍스트는 주석으로 간주됩니다. 예를 들어 다음과 같습니다.

```
notify yes;  # notify all secondary nameservers
```

/ 및 */*

/ 및 */ 로 묶은 모든 텍스트 블록은 주석으로 간주됩니다. 예를 들어 다음과 같습니다.*

```
notify yes;  /* notify all secondary nameservers */
```

15.2.3. 영역 파일 편집

15.1.1절. “이름 서버 영역” 에 설명된 대로 영역 파일에는 네임스페이스에 대한 정보가 포함되어 있습니다. 기본적으로 /var/named/ 에 있는 명명된 작업 디렉터리에 저장됩니다. 각 영역 파일의 이름은 zone 문에 있는 파일 옵션에 따라 이름이 지정되며, 일반적으로 에서 도메인과 관련이 있는 방식으로 파일을 포함하는 방식(예: example.com.zone)으로 지정됩니다.

표 15.5. 명명된 서비스 영역 파일

경로	Description
/var/named/	named 서비스의 작업 디렉터리입니다. 이름 서버는 이 디렉토리에 쓸 수 없습니다.
/var/named/slaves/	보조 영역의 디렉터리입니다. 이 디렉터리는 named 서비스에서 쓸 수 있습니다.
/var/named/dynamic/	동적 DNS (DDNS) 영역 또는 관리형 DNSSEC 키와 같은 기타 파일의 디렉터리입니다. 이 디렉터리는 named 서비스에서 쓸 수 있습니다.
/var/named/data/	다양한 통계 및 디버깅 파일을 위한 디렉터리입니다. 이 디렉터리는 named 서비스에서 쓸 수 있습니다.

영역 파일은 지시문과 리소스 레코드로 구성됩니다. 지시문을 사용하면 이름 서버에 작업을 수행하거나 특정 설정을 영역에 적용하고 리소스 레코드는 영역의 매개 변수를 정의하고 개별 호스트에 ID를 할당합니다. 지시문은 선택 사항이지만, 영역에 이름 서비스를 제공하려면 리소스 레코드가 필요합니다.

모든 지시문과 리소스 레코드는 개별 줄에 입력해야 합니다.

15.2.3.1. 일반 지시문

지시문은 달러 기호 문자($)로 시작하고 그 뒤에 지시문 이름이 오며 일반적으로 파일 맨 위에 표시됩니다. 다음 지시문은 영역 파일에서 일반적으로 사용됩니다.

$INCLUDE

$INCLUDE 지시문을 사용하면 다른 영역 설정을 별도의 영역 파일에 저장할 수 있도록 표시되는 위치에 다른 파일을 포함할 수 있습니다.

-

> **예 15.7. $INCLUDE 지시문 사용**
>
> ```
> $INCLUDE /var/named/penguin.example.com
> ```

$ORIGIN

$ORIGIN 지시문을 사용하면 호스트 이름을 사용하는 항목과 같이 정규화되지 않은 레코드에 도메인 이름을 추가할 수 있습니다. 영역 이름이 기본적으로 사용되므로 영역을 **/etc/named.conf** 에 지정하면 이 지시문을 사용할 필요가 없습니다.

예 15.8. "$ORIGIN 지시문 사용" 에서 후행 마침표(. 문자)로 끝나지 않는 리소스 레코드에 사용된 모든 이름은 **example.com** 에 추가됩니다.

> **예 15.8. $ORIGIN 지시문 사용**
>
> ```
> $ORIGIN example.com.
> ```

$TTL

$TTL 지시문을 사용하면 영역의 기본 **TTL(Time to Live)** 값을 설정할 수 있습니다. 즉, 영역 레코드의 유효 기간은 얼마입니까. 각 리소스 레코드에는 고유한 **TTL** 값이 포함될 수 있으며, 이 지시문을 재정의합니다.

이 값을 늘리면 원격 이름 서버가 더 오랜 기간 동안 영역 정보를 캐시하여 영역의 쿼리 수를 줄이고 리소스 레코드 변경 사항을 전파하는 데 필요한 시간을 늘릴 수 있습니다.

> **예 15.9. $TTL 지시문 사용**
>
> ```
> $TTL 1D
> ```

15.2.3.2. 일반적인 리소스 레코드

다음 리소스 레코드는 영역 파일에서 일반적으로 사용됩니다.

A

Address 레코드는 이름에 할당할 **IP** 주소를 지정합니다. 다음 형식을 취합니다.

```
hostname IN A IP-address
```

hostname 값이 생략되면 레코드는 마지막으로 지정된 **hostname** 을 가리킵니다.

예 15.10. "A 리소스 레코드 사용" 에서 **server1.example.com** 에 대한 요청은 **10.0.1.3** 또는 **10.0.1. 5**을 가리킵니다.

예 15.10. A 리소스 레코드 사용

```
server1  IN  A  10.0.1.3
         IN  A  10.0.1.5
```

CNAME

Canonical Name 레코드는 한 이름을 다른 이름에 매핑합니다. 이 때문에 이러한 유형의 레코드를 별칭 레코드 라고 합니다. 다음 형식을 취합니다.

```
alias-name IN CNAME real-name
```

CNAME 레코드는 웹 서버의 **www** 와 같은 일반적인 명명 체계를 사용하는 서비스를 가리키는 데 가장 일반적으로 사용됩니다. 그러나 사용법에 대한 여러 제한 사항이 있습니다.

- **CNAME** 레코드는 다른 **CNAME** 레코드를 가리켜서는 안 됩니다. 이것은 주로 가능한 무한 루프를 피하기 위한 것입니다.
- **CNAME** 레코드에는 다른 리소스 레코드 유형(예: **A, NS, MX** 등)이 없어야 합니다. 유일한 예외는 영역이 서명될 때 **DNSSEC** 관련 레코드(**RRSIG, NSEC** 등)입니다.
- 호스트(**NS, MX, PTR**)의 **FQDN**(정규화된 도메인 이름)을 가리키는 기타 리소스 레코드는 **CNAME** 레코드를 가리키지 않아야 합니다.

예 15.11. "CNAME 리소스 레코드 사용" 에서 **A** 레코드는 호스트 이름을 **IP** 주소에 바인딩하지만 **CNAME** 레코드는 일반적으로 사용되는 **www** 호스트 이름을 가리킵니다.

예 15.11. CNAME 리소스 레코드 사용

```
server1  IN  A      10.0.1.5
www      IN  CNAME  server1
```

MX

메일 교환 레코드는 특정 네임스페이스로 전송된 메일을 이 영역에서 제어해야 하는 위치를 지정합니다. 다음 형식을 취합니다.

```
IN MX preference-value email-server-name
```

email-server-name 은 정규화된 도메인 이름(FQDN)입니다. **preference-value** 를 사용하면 네임스페이스에 대한 이메일 서버의 숫자 순위를 사용할 수 있으므로 일부 이메일 시스템보다 우선 순위를 지정할 수 있습니다. **preference-value** 가 가장 낮은 **MX** 리소스 레코드가 다른 항목보다 우선합니다. 그러나 여러 이메일 서버가 동일한 값을 보유하여 이메일 트래픽을 균등하게 분산할 수 있습니다.

예 15.12. "MX 리소스 레코드 사용" 에서 첫 번째 **mail.example.com** 이메일 서버는 **example.com** 도메인으로 향하는 이메일을 수신할 때 **mail2.example.com** 이메일 서버로 선호됩니다.

예 15.12. MX 리소스 레코드 사용

```
example.com.  IN  MX  10  mail.example.com.
              IN  MX  20  mail2.example.com.
```

NS

Nameserver 레코드는 특정 영역에 대한 권한 있는 이름 서버를 보고합니다. 다음 형식을 취합니다.

```
IN NS nameserver-name
```

nameserver-name 은 정규화된 도메인 이름(FQDN)이어야 합니다. 두 이름 서버가 도메인에 대한 권한 있는 것으로 나열되면 이러한 이름 서버가 보조 이름 서버인지 아니면 기본 서버인지는 중요하지 않습니다. 이 두 가지는 여전히 권위 있는 것으로 간주됩니다.

예 15.13. NS 리소스 레코드 사용

```
IN  NS  dns1.example.com.
IN  NS  dns2.example.com.
```

PTR

포인터 레코드는 네임스페이스의 다른 부분을 가리킵니다. 다음 형식을 취합니다.

```
last-IP-digit IN PTR FQDN-of-system
```

last-IP-digit 지시문은 **IP** 주소의 마지막 숫자이며 **FQDN-of-system** 은 정규화된 도메인 이름 **(FQDN)**입니다.

PTR 레코드는 **IP** 주소를 다시 특정 이름으로 가리키므로 주로 역방향 이름 확인에 사용됩니다. 사용 중인 **PTR** 레코드의 예는 **15.2.3.4.2절. "역방향 이름 확인 영역 파일"** 을 참조하십시오.

SOA

Start of Authority record에서 네임스페이스에 대한 중요한 신뢰할 수 있는 정보를 이름 서버에 발표합니다. 지시문 뒤에 있는 이는 영역 파일의 첫 번째 리소스 레코드입니다. 다음 형식을 취합니다.

```
@  IN  SOA  primary-name-server hostmaster-email (
     serial-number
     time-to-refresh
     time-to-retry
     time-to-expire
     minimum-TTL )
```

지시문은 다음과 같습니다.

- @ 기호는 **$ORIGIN** 지시문(또는 **$ORIGIN** 지시문이 설정되지 않은 경우 영역의 이름)을 이 **SOA** 리소스 레코드에서 정의하는 네임스페이스로 배치합니다.
- **primary-name-server** 지시문은 이 도메인에 대해 권한이 있는 기본 이름 서버의 호스트 이름입니다.
- **hostmaster-email** 지시문은 네임스페이스에 연결할 사용자의 이메일입니다.

- **serial-number** 지시문은 영역 파일이 변경될 때마다 숫자 값이 증가하여 명명된 서비스가 영역을 다시 로드할 때입니다.

- **time-to-refresh** 지시문은 숫자 값 보조 이름 서버에서 해당 영역을 변경한 경우 기본 이름 서버에 요청하기 전에 대기하는 시간을 결정하는 데 사용합니다.

- **time-to-try** 지시문은 기본 이름 서버가 응답하지 않는 경우 새로 고침 요청을 발행하기 전에 대기하는 시간을 결정하는 데 사용되는 숫자 값입니다. **time -to-expire** 지시문에 지정된 시간 전에 기본 서버에서 새로 고침 요청에 응답하지 않으면 보조 서버에서 해당 네임스페이스에 대한 요청으로 응답을 중지합니다.

- **BIND 4 및 8**에서 **minimum-TTL** 지시문은 다른 이름 서버가 영역의 정보를 캐시하는 시간입니다. **BIND 9**에서는 에 부정적인 응답이 캐시되는 기간을 정의합니다. 음수 응답 캐싱은 최대 **3**시간**(3H)**으로 설정할 수 있습니다.

BIND를 구성할 때 모든 시간은 초 단위로 지정됩니다. 그러나 분**(M)**, 시간**(h)**, 일**(D)** 및 주**(W)**와 같이 시간 단위를 지정할 때 약어(약어)를 사용할 수 있습니다. 표 **15.6.** “다른 시간 단위와 비교한 초”은 시간(초)과 이에 해당하는 시간을 다른 형식으로 표시합니다.

표 **15.6.** 다른 시간 단위와 비교한 초

초	기타 시간 단위
60	**1M**
1800	**30M**
3600	**1H**
10800	**3H**
21600	**6H**
43200	**12H**
86400	**1D**
259200	**3D**
604800	**1W**

초	기타 시간 단위
31536000	365D

예 15.14. SOA 리소스 레코드 사용

```
@  IN  SOA  dns1.example.com.  hostmaster.example.com. (
        2001062501  ; serial
        21600       ; refresh after 6 hours
        3600        ; retry after 1 hour
        604800      ; expire after 1 week
        86400 )     ; minimum TTL of 1 day
```

15.2.3.3. 코멘트 태그

리소스 레코드 및 지시문 외에도 영역 파일에 주석도 포함될 수 있습니다. 주석은 명명된 서비스에서 무시되지만 사용자에게 추가 정보를 제공할 때 유용할 수 있습니다. 행 끝에 세미콜론 문자 뒤에 있는 텍스트는 주석으로 간주됩니다. 예를 들어 다음과 같습니다.

```
604800  ; expire after 1 week
```

15.2.3.4. 사용 예

다음 예제에서는 영역 파일의 기본 사용법을 보여줍니다.

15.2.3.4.1. 간단한 영역 파일

예 15.15. "간단한 영역 파일" 표준 지시문과 SOA 값의 사용을 보여줍니다.

예 15.15. 간단한 영역 파일

```
$ORIGIN example.com.
$TTL 86400
@       IN  SOA  dns1.example.com.  hostmaster.example.com. (
            2001062501  ; serial
            21600       ; refresh after 6 hours
            3600        ; retry after 1 hour
            604800      ; expire after 1 week
            86400 )     ; minimum TTL of 1 day
;
;
        IN  NS     dns1.example.com.
```

```
        IN  NS    dns2.example.com.
dns1    IN  A     10.0.1.1
        IN  AAAA  aaaa:bbbb::1
dns2    IN  A     10.0.1.2
        IN  AAAA  aaaa:bbbb::2
;
;
@       IN  MX    10  mail.example.com.
        IN  MX    20  mail2.example.com.
mail    IN  A     10.0.1.5
        IN  AAAA  aaaa:bbbb::5
mail2   IN  A     10.0.1.6
        IN  AAAA  aaaa:bbbb::6
;
;
; This sample zone file illustrates sharing the same IP addresses
; for multiple services:
;
services  IN  A     10.0.1.10
        IN  AAAA  aaaa:bbbb::10
        IN  A     10.0.1.11
        IN  AAAA  aaaa:bbbb::11

ftp     IN  CNAME  services.example.com.
www     IN  CNAME  services.example.com.
;
;
```

*이 예에서 권한 있는 이름 서버는 **dns1.example.com** 및 **dns2.example.com** 으로 설정되며 **A** 레코드를 사용하여 각각 **10.0.1.1** 및 **10.0.1.2 IP** 주소에 연결됩니다.*

*MX 레코드로 구성된 이메일 서버는 **A** 레코드를 통해 메일 과 **mail2** 를 가리킵니다. 이러한 이름은 후행 마침표로 종료되지 않으므로 **$ORIGIN** 도메인이 배치되고 **mail.example. com** 및 **mail2.example.com** 으로 확장됩니다.*

*WWW(www.example.com (WW)와 같은 표준 이름에서 사용할 수 있는 서비스는 **CNAME** 레코드를 사용하여 적절한 서버를 가리키도록 합니다.*

*이 영역 파일은 다음과 유사한 **/etc/named.conf** 에 **zone** 문을 사용하여 서비스로 호출됩니다.*

```
zone "example.com" IN {
  type master;
  file "example.com.zone";
  allow-update { none; };
};
```

15.2.3.4.2. 역방향 이름 확인 영역 파일

역방향 이름 확인 영역 파일은 특정 네임스페이스의 IP 주소를 FQDN(정규화된 도메인 이름)으로 변환하는 데 사용됩니다. 예 15.16. "역방향 이름 확인 영역 파일" 에 표시된 대로 fully qualified domain name에 표시된 대로 PTR 리소스 레코드를 사용하여 IP 주소를 정규화된 도메인 이름에 연결하는 것을 제외하고 표준 영역 파일과 매우 유사합니다.

예 15.16. 역방향 이름 확인 영역 파일

```
$ORIGIN 1.0.10.in-addr.arpa.
$TTL 86400
@  IN  SOA  dns1.example.com.  hostmaster.example.com. (
        2001062501  ; serial
        21600       ; refresh after 6 hours
        3600        ; retry after 1 hour
        604800      ; expire after 1 week
        86400 )     ; minimum TTL of 1 day
;
@  IN  NS   dns1.example.com.
;
1  IN  PTR  dns1.example.com.
2  IN  PTR  dns2.example.com.
;
5  IN  PTR  server1.example.com.
6  IN  PTR  server2.example.com.
;
3  IN  PTR  ftp.example.com.
4  IN  PTR  ftp.example.com.
```

이 예에서는 IP 주소 10.0.1.1~ 10.0.1.6 이 해당 정규화된 도메인 이름을 가리킵니다.

이 영역 파일은 다음과 유사한 /etc/named.conf 파일에 zone 문을 사용하여 서비스로 호출됩니다.

```
zone "1.0.10.in-addr.arpa" IN {
  type master;
  file "example.com.rr.zone";
  allow-update { none; };
};
```

영역 이름을 제외하고 이 예제와 표준 zone 문에는 큰 차이가 없습니다. 역방향 이름 확인 영역에는 역방향 IP 주소의 처음 세 블록과 .in-addr.arpa 가 필요합니다. 이를 통해 역방향 이름 확인 영역 파일에 사용되는 단일 IP 번호 블록을 영역과 연결할 수 있습니다.

15.2.4. rndc 유틸리티 사용

rndc 유틸리티는 로컬 및 원격 시스템에서 명명된 서비스를 관리할 수 있는 명령줄 도구입니다. 사용법은 다음과 같습니다.

```
rndc [option...] command [command-option]
```

15.2.4.1. 유틸리티 구성

서비스에 대한 무단 액세스를 방지하려면 **named** 를 구성하여 선택한 포트(기본적으로**953**)에서 수신하도록 구성해야 하며, 서비스와 **rndc** 유틸리티에서 동일한 키를 사용해야 합니다.

표 15.7. 관련 파일

경로	Description
/etc/named.conf	**named** 서비스의 기본 구성 파일.
/etc/rndc.conf	**rndc** 유틸리티의 기본 구성 파일.
/etc/rndc.key	기본 키 위치입니다.

rndc 구성은 **/etc/rndc.conf** 에 있습니다. 파일이 없는 경우 유틸리티는 **/etc/rndc.key**에 있는 키를 사용합니다. 이 키는 **rndc-confgen -a** 명령을 사용하여 설치 프로세스 중에 자동으로 생성되었습니다.

명명된 서비스는 15.2.2.3절. “기타 설명 유형” 에 설명된 대로 **/etc/named.conf** 구성 파일의 **control** 문을 사용하여 구성됩니다. 이 문이 없으면 루프백 주소(**127.0.0.1**)의 연결만 허용되고 **/etc/rndc.key** 에 있는 키가 사용됩니다.

이 항목에 대한 자세한 내용은 15.2.8절. “추가 리소스” 에 나열된 도움말 페이지와 **BIND 9** 관리자 참조 설명서를 참조하십시오.

중요

권한이 없는 사용자가 서비스에 제어 명령을 보내지 못하게 하려면 **root** 만 **/etc/rndc.key** 파일을 읽을 수 있는지 확인하십시오.

```
~]# chmod o-rwx /etc/rndc.key
```

15.2.4.2. 서비스 상태 확인

named 서비스의 현재 상태를 확인하려면 다음 명령을 사용합니다.

```
~]# rndc status
version: 9.7.0-P2-RedHat-9.7.0-5.P2.el6
CPUs found: 1
worker threads: 1
number of zones: 16
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
query logging is OFF
recursive clients: 0/0/1000
tcp clients: 0/100
server is up and running
```

15.2.4.3. 구성 및 영역 다시 로드

구성 파일과 영역을 모두 다시 로드하려면 쉘 프롬프트에서 다음을 입력합니다.

```
~]# rndc reload
server reload successful
```

이렇게 하면 이전에 캐시된 모든 응답을 유지하면서 영역이 다시 로드되므로 저장된 모든 이름 확인을 손실하지 않고 영역 파일을 변경할 수 있습니다.

단일 영역을 다시 로드하려면 다시 로드 명령 다음에 이름을 지정합니다. 예를 들면 다음과 같습니다.

```
~]# rndc reload localhost
zone reload up-to-date
```

마지막으로 구성 파일 및 새로 추가된 영역만 다시 로드하려면 다음을 입력합니다.

```
~]# rndc reconfig
```

참고

DDNS(동적 DNS)를 사용하는 영역을 수동으로 수정하려는 경우 먼저 중단 명령을 실행해야 합니다.

```
~]# rndc freeze localhost
```

완료되면 thaw 명령을 실행하여 DDNS 를 다시 허용하고 영역을 다시 로드합니다.

```
~]# rndc thaw localhost
The zone reload and thaw was successful.
```

15.2.4.4. 영역 키 업데이트

DNSSEC 키를 업데이트하고 영역에 서명하려면 sign 명령을 사용합니다. 예를 들어 다음과 같습니다.

```
~]# rndc sign localhost
```

위 명령을 사용하여 영역에 서명하려면 zone 문에 유지 관리 하도록 auto-dnssec 옵션을 설정해야 합니다. 예를 들어 다음과 같습니다.

```
zone "localhost" IN {
  type master;
  file "named.localhost";
  allow-update { none; };
  auto-dnssec maintain;
};
```

15.2.4.5. DNSSEC 검증 활성화

DNSSEC 검증을 활성화하려면 root 로 다음 명령을 실행합니다:

```
~]# rndc validation on
```

마찬가지로 이 옵션을 비활성화하려면 다음을 입력합니다.

```
~]# rndc validation off
```

/etc/named.conf 에서 이 옵션을 구성하는 방법에 대한 자세한 내용은 15.2.2.2절. “일반적인 설명 유

형” 에 설명된 옵션 설명을 참조하십시오.

Red Hat Enterprise Linux 7 보안 가이드 에는 DNSSEC에 대한 포괄적인 섹션이 있습니다.

15.2.4.6. 쿼리 로깅 활성화

쿼리 로깅이 현재 활성화된 경우 활성화(또는 비활성화)하려면 root 로 다음 명령을 실행합니다:

```
~]# rndc querylog
```

현재 설정을 확인하려면 15.2.4.2절. “서비스 상태 확인” 에 설명된 대로 status 명령을 사용합니다.

15.2.5. dig 유틸리티 사용

dig 유틸리티는 DNS 조회를 수행하고 이름 서버 구성을 디버깅할 수 있는 명령줄 도구입니다. 일반적인 사용법은 다음과 같습니다.

```
dig [@server] [option...] name type
```

유형에 사용할 공통 값 목록은 15.2.3.2절. “일반적인 리소스 레코드” 을 참조하십시오.

15.2.5.1. 이름 서버 검색

특정 도메인의 이름 서버를 찾으려면 다음 형식의 명령을 사용합니다.

```
dig name NS
```

예 15.17. “샘플 이름 서버 조회” 에서 dig 유틸리티는 example.com 의 네임서버를 표시하는 데 사용됩니다.

예 15.17. 샘플 이름 서버 조회

```
~]$ dig example.com NS

; <<>> DiG 9.7.1-P2-RedHat-9.7.1-2.P2.fc13 <<>> example.com NS
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57883
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0
```

```
;; QUESTION SECTION:
;example.com.                IN      NS

;; ANSWER SECTION:
example.com.         99374   IN      NS      a.iana-servers.net.
example.com.         99374   IN      NS      b.iana-servers.net.

;; Query time: 1 msec
;; SERVER: 10.34.255.7#53(10.34.255.7)
;; WHEN: Wed Aug 18 18:04:06 2010
;; MSG SIZE  rcvd: 77
```

15.2.5.2. IP 주소 검색

특정 도메인에 할당된 IP 주소를 조회하려면 다음 형식으로 명령을 사용합니다.

```
dig name A
```

예 15.18. "샘플 IP 주소 조회" 에서 dig 유틸리티는 example.com 의 IP 주소를 표시하는 데 사용됩니다.

예 15.18. 샘플 IP 주소 조회

```
~]$ dig example.com A

; <<>> DiG 9.7.1-P2-RedHat-9.7.1-2.P2.fc13 <<>> example.com A
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4849
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com.                IN      A

;; ANSWER SECTION:
example.com.         155606  IN      A       192.0.32.10

;; AUTHORITY SECTION:
example.com.         99175   IN      NS      a.iana-servers.net.
example.com.         99175   IN      NS      b.iana-servers.net.

;; Query time: 1 msec
;; SERVER: 10.34.255.7#53(10.34.255.7)
;; WHEN: Wed Aug 18 18:07:25 2010
;; MSG SIZE  rcvd: 93
```

15.2.5.3. 호스트 이름 조회

특정 IP 주소의 호스트 이름을 조회하려면 다음 형식으로 명령을 사용합니다.

```
dig -x address
```

예 15.19. "샘플 호스트 이름 조회" 에서 dig 유틸리티는 192.0.32.10 에 할당된 호스트 이름을 표시하는 데 사용됩니다.

예 15.19. 샘플 호스트 이름 조회

```
~]$ dig -x 192.0.32.10

; <<>> DiG 9.7.1-P2-RedHat-9.7.1-2.P2.fc13 <<>> -x 192.0.32.10
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29683
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 6

;; QUESTION SECTION:
;10.32.0.192.in-addr.arpa.      IN      PTR

;; ANSWER SECTION:
10.32.0.192.in-addr.arpa. 21600 IN      PTR     www.example.com.

;; AUTHORITY SECTION:
32.0.192.in-addr.arpa.  21600   IN      NS      b.iana-servers.org.
32.0.192.in-addr.arpa.  21600   IN      NS      c.iana-servers.net.
32.0.192.in-addr.arpa.  21600   IN      NS      d.iana-servers.net.
32.0.192.in-addr.arpa.  21600   IN      NS      ns.icann.org.
32.0.192.in-addr.arpa.  21600   IN      NS      a.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.net.     13688   IN      A       192.0.34.43
b.iana-servers.org.     5844    IN      A       193.0.0.236
b.iana-servers.org.     5844    IN      AAAA    2001:610:240:2::c100:ec
c.iana-servers.net.     12173   IN      A       139.91.1.10
c.iana-servers.net.     12173   IN      AAAA    2001:648:2c30::1:10
ns.icann.org.           12884   IN      A       192.0.34.126

;; Query time: 156 msec
;; SERVER: 10.34.255.7#53(10.34.255.7)
;; WHEN: Wed Aug 18 18:25:15 2010
;; MSG SIZE  rcvd: 310
```

15.2.6. BIND의 고급 기능

대부분의 BIND 구현에서는 이름 확인 서비스를 제공하거나 특정 도메인에 대한 기관 역할을 하는 데만 명명된 서비스를 사용합니다. 그러나 BIND 버전 9에는 보다 안전하고 효율적인 DNS 서비스를 허용하

는 여러 가지 고급 기능이 있습니다.

중요

DNSSEC, TSIG 또는 **IXFR**(상세 영역 전송)과 같은 고급 기능을 사용하기 전에 특히 이전 버전의 **BIND** 또는 비**BIND** 서버를 사용하는 경우 네트워크 환경의 모든 이름 서버에서 특정 기능을 지원하는지 확인합니다.

언급된 모든 기능은 15.2.8.1절. “설치된 문서” 에서 참조 한 **BIND 9** 관리자 참조 설명서에 더 자세히 설명되어 있습니다.

15.2.6.1. 다중 보기

선택적으로 요청이 시작되는 네트워크에 따라 클라이언트에 다른 정보를 제공할 수 있습니다. 이는 주로 로컬 네트워크 외부의 클라이언트에서 중요한 **DNS** 항목을 거부하는 동시에 로컬 네트워크 내부의 클라이언트의 쿼리를 허용하는 데 사용됩니다.

여러 뷰를 구성하려면 **view** 문을 **/etc/named.conf** 구성 파일에 추가합니다. **match-clients** 옵션을 사용하여 **IP** 주소 또는 전체 네트워크를 일치시키고 특수 옵션 및 영역 데이터를 제공합니다.

15.2.6.2. 증분 영역 전송(IXFR)

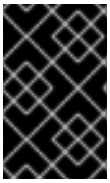
IXFR(incremental Zone Transfers)을 사용하면 보조 이름 서버에서 기본 이름 서버에서 수정된 영역의 업데이트된 부분만 다운로드할 수 있습니다. 표준 전송 프로세스에 비해 알림 및 업데이트 프로세스의 효율성이 훨씬 향상됩니다.

IXFR은 동적 업데이트를 사용하여 기본 영역 레코드를 변경할 때만 사용할 수 있습니다. 변경 작업을 위해 영역 파일을 수동으로 편집하는 경우**AXFR(Automatic Zone Transfer)**이 사용됩니다.

15.2.6.3. 트랜잭션 SIGnatures(TSIG)

TSIG(Transaction SIGnatures)는 전송을 허용하기 전에 공유 비밀 키가 기본 이름 서버와 보조 이름 서버에 모두 있는지 확인합니다. 그러면 공격자가 영역을 전송하기 위해 **IP** 주소에 액세스할 필요가 있을 뿐만 아니라 비밀 키를 알아야 하기 때문에 표준 **IP** 주소 기반 전송 방법이 강화됩니다.

BIND는 버전 **9**부터 영역 전송을 인증하는 또 다른 공유 비밀 키 방법인 **TKEY** 도 지원합니다.



중요

비보안 네트워크를 통해 통신할 때 IP 주소 기반 인증만 사용하지 마십시오.

15.2.6.4. DNSSEC(DNS 보안 확장)

DNSSEC(Domain Name System Security Extensions)는 DNS 데이터에 대한 출처 인증, 인증된 존재 거부 및 데이터 무결성을 제공합니다. 특정 도메인이 보안으로 표시되면 검증에 실패하는 각 리소스 레코드에 대해 SERVFAIL 응답이 반환됩니다.

DNSSEC 서명 도메인 또는 DNSSEC 인식 확인자를 디버깅하려면 15.2.5절. “dig 유틸리티 사용” 에 설명된 대로 dig 유틸리티를 사용할 수 있습니다. 유용한 옵션은 +dnssec (DNS OK 비트를 설정하여 DNSSEC 관련 리소스 레코드 요청), +cd (응답의 유효성을 검사하지 않음) 및 +bufsize=512 (일부 방화벽을 통과하기 위해 패킷 크기를 512B로 변경)입니다.

15.2.6.5. IPv6(Internet Protocol version 6)

IPv6 (Internet Protocol version 6)는 AAAA 리소스 레코드를 사용하여 지원되며 표 15.3. “일반적으로 사용되는 구성 옵션” 에 설명된 listen-on-v6 지시문이 지원됩니다.

15.2.7. 자주 발생하는 오류 발생 방지

다음은 이름 서버를 구성할 때 발생하는 일반적인 실수를 방지하는 방법에 대한 권장 사항입니다.

세미콜론 및 중괄 괄호 사용

/etc/named.conf 파일에서 생략된 세미콜론 또는 일치하지 않는 중괄호가 있으면 명명된 서비스가 시작되지 않을 수 있습니다.

마침표(. 문자)를 올바르게 사용합니다.

영역 파일에서 도메인 이름 끝에 있는 마침표는 정규화된 도메인 이름을 나타냅니다. 생략하면 named 서비스는 영역의 이름 또는 $ORIGIN 값을 추가하여 완료합니다.

영역 파일을 편집할 때 일련번호가 증가

일련 번호가 증가하지 않으면 기본 이름 서버에 올바른 새 정보가 포함되지만 보조 이름 서버에는 변경 내용이 표시되지 않으며 해당 영역의 데이터 새로 고침을 시도하지 않습니다.

방화벽 설정

방화벽이 명명된 서비스에서 다른 이름 서버로 의 연결을 차단하는 경우 방화벽 설정을 변경하는 것이 좋습니다.

주의

***DNS** 쿼리에 고정 **UDP** 소스 포트를 사용하는 것은 공격자가 캐시 스푸이싱 공격을 더 쉽게 수행할 수 있는 잠재적인 보안 취약점입니다. 이를 방지하기 위해 기본적으로 **DNS** 는 임의의 임시 포트에서 전송됩니다. 임의 **UDP** 소스 포트에서 나가는 쿼리를 허용하도록 방화벽을 구성합니다. 기본적으로 **1024 ~ 65535** 범위가 사용됩니다.*

15.2.8. 추가 리소스

*다음 정보 소스는 **BIND**와 관련된 추가 리소스를 제공합니다.*

15.2.8.1. 설치된 문서

***BIND**에는 각각 자체 주체 디렉터리에 배치된 다양한 주제를 다루는 설치된 전체 설명서가 있습니다. 아래 각 항목에 대해 **version** 을 시스템에 설치된 **bind** 패키지의 버전으로 바꿉니다.*

/usr/share/doc/bind-version/

*가장 최근 문서가 포함된 기본 디렉터리입니다. 디렉터리에는 **HTML** 및 **PDF** 형식의 **BIND 9** 관리자 참조 설명서 가 포함되어 있습니다. 이 설명서에는 **BIND** 리소스 요구 사항, 다양한 유형의 이름 서버 구성 방법, 부하 분산 방법 및 기타 고급 주제를 설명합니다.*

/usr/share/doc/bind-version/sample/etc/

명명된 구성 파일의 예제가 포함된 디렉터리입니다.

rndc(8)

사용에 대한 설명서를 포함하는 **rndc** 이름 서버 제어 유틸리티의 도움말 페이지.

named(8)

BIND 이름 서버 데몬을 제어하는 데 사용할 수 있는 정렬된 인수에 대한 문서가 들어 있는 라는 인터넷 도메인 이름 서버의 도움말 페이지.

lwresd(8)

경량 확인자 데몬 **lwresd**의 설명서가 데몬 및 그 사용량에 대한 설명서를 포함합니다.

named.conf(5)

명명된 구성 파일 내에서 사용할 수 있는 전체 옵션 목록이 있는 도움말 페이지.

rndc.conf(5)

rndc 구성 파일 내에서 사용 가능한 전체 옵션 목록이 있는 도움말 페이지.

15.2.8.2. 온라인 리소스

https://access.redhat.com/site/articles/770133

Red Hat Enterprise Linux 6과 비교하여 **chroot** 환경에서 **BIND**를 실행하는 방법에 대한 **Red Hat Knowledgebase** 문서입니다.

https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/

Red Hat Enterprise Linux 7 보안 가이드 에는 **DNSSEC**에 대한 포괄적인 섹션이 있습니다.

https://www.icann.org/namecollision

도메인 이름 충돌 시 **FAQ**.

16장. SQUID 캐싱 프록시 서버 구성

Squid는 콘텐츠를 캐시하여 대역폭과 부하 웹 페이지를 더 빠르게 줄이는 프록시 서버입니다. 이 장에서는 HTTP, HTTPS 및 FTP 프로토콜에 대한 프록시로 Squid를 설정하는 방법과 인증 및 액세스 제한에 대해 설명합니다.

16.1. 인증 없이 캐싱 프록시로 SQUID 설정

이 섹션에서는 인증 없이 캐싱 프록시로 Squid의 기본 구성에 대해 설명합니다. 이 절차에서는 IP 범위를 기반으로 프록시에 대한 액세스를 제한합니다.

사전 요구 사항

- 이 절차에서는 /etc/squid/squid.conf 파일이 squid 패키지에서 제공하는 것으로 가정합니다. 이전에 이 파일을 편집한 경우 파일을 제거하고 패키지를 다시 설치합니다.

절차

1. squid 패키지를 설치합니다.

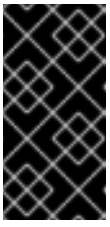
```
# yum install squid
```

2. /etc/squid/squid.conf 파일을 편집합니다.

   a. 프록시를 사용하도록 허용해야 하는 IP 범위와 일치하도록 localnet ACL(액세스 제어 목록)을 조정합니다.

```
acl localnet src 192.0.2.0/24
acl localnet 2001:db8::/32
```

기본적으로 /etc/squid/squid.conf 파일에는 http_access를 통해 localnet ACL에 지정된 모든 IP 범위에서 프록시를 사용할 수 있는 localnet 규칙이 있습니다. http_access에서 localnet 규칙을 허용하기 전에 모든 localnet ACL을 지정해야 합니다.



중요

사용자 환경과 일치하지 않는 기존 acl localnet 항목을 모두 제거합니다.

b.

*다음 **ACL**은 기본 구성에 있으며 **HTTPS** 프로토콜을 사용하는 포트로 **443** 을 정의합니다.*

```
acl SSL_ports port 443
```

*사용자가 다른 포트에서도 **HTTPS** 프로토콜을 사용할 수 있어야 하는 경우 다음 각 포트에 대한 **ACL**을 추가합니다.*

```
acl SSL_ports port port_number
```

c.

*연결을 설정할 수 있는 포트를 구성하도록 **aclSafe_ports** 규칙 목록을 업데이트합니다. 예를 들어 프록시를 사용하여 클라이언트가 포트 **21(FTP), 80(HTTP)** 및 **443(HTTPS)**의 리소스에만 액세스할 수 있도록 구성하려면 구성에서 다음 **aclSafe_ports** 문만 유지합니다.*

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

*기본적으로 이 구성에는 **Safe_ports ACL**에 정의되지 않은 포트에 대한 액세스 거부를 정의하는 **http_access deny !Safe _ports** 규칙이 포함됩니다.*

d.

***cache** 유형, 캐시 디렉터리의 경로, 캐시 크기 및 **cache _dir** 매개변수의 추가 캐시 유형별 설정을 구성합니다.*

```
cache_dir ufs /var/spool/squid 10000 16 256
```

다음 설정이 필요합니다.

- ***squid**는 **ufs** 캐시 유형을 사용합니다.*
- ***Squid**는 캐시를 **/var/spool/squid/** 디렉터리에 저장합니다.*
- *캐시는 최대 **10000 MB**까지 증가합니다.*
- ***squid**는 **/var/spool/squid/** 디렉토리에 **16 level-1** 하위 디렉토리를 만듭니다.*

- *Squid는 각 level-1 디렉토리에 256 개의 하위 디렉토리를 생성합니다.*

cache_dir 지시문을 설정하지 않으면 Squid는 캐시를 메모리에 저장합니다.

3. *cache _dir 매개변수에서 /var/spool/squid/ 와 다른 캐시 디렉토리를 설정하는 경우:*

   a. *캐시 디렉토리를 생성합니다.*

   ```
   # mkdir -p path_to_cache_directory
   ```

   b. *캐시 디렉토리에 대한 권한을 구성합니다.*

   ```
   # chown squid:squid path_to_cache_directory
   ```

   c. *강제 모드에서 SELinux를 실행하는 경우 캐시 디렉터리에 대한 squid_cache_t 컨텍스트를 설정합니다.*

   ```
   # semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"
   # restorecon -Rv path_to_cache_directory
   ```

   semanage 유틸리티를 시스템에서 사용할 수 없는 경우 policycoreutils-python-utils 패키지를 설치합니다.

4. *방화벽에서 3128 포트를 엽니다.*

   ```
   # firewall-cmd --permanent --add-port=3128/tcp
   # firewall-cmd --reload
   ```

5. *squid 서비스를 시작합니다.*

   ```
   # systemctl start squid
   ```

6. *시스템이 부팅될 때 squid 서비스가 자동으로 시작되도록 활성화합니다.*

```
# systemctl enable squid
```

검증 단계

프록시가 올바르게 작동하는지 확인하려면 **curl** 유틸리티를 사용하여 웹 페이지를 다운로드합니다.

```
# curl -O -L "https://www.redhat.com/index.html" -x "proxy.example.com:3128"
```

curl 에서 오류를 표시하지 않고 **index.html** 파일을 현재 디렉터리로 다운로드한 경우 프록시가 작동합니다.

16.2. LDAP 인증을 사용하여 캐싱 프록시로 SQUID 설정

이 섹션에서는 **LDAP**를 사용하여 사용자를 인증하는 캐싱 프록시로 **Squid**의 기본 구성에 대해 설명합니다. 이 절차에서는 인증된 사용자만 프록시를 사용할 수 있도록 구성됩니다.

사전 요구 사항

- 이 절차에서는 **/etc/squid/squid.conf** 파일이 **squid** 패키지에서 제공하는 것으로 가정합니다. 이전에 이 파일을 편집한 경우 파일을 제거하고 패키지를 다시 설치합니다.
- **uid=proxy_user,cn=users,cn=accounts,dc=example,dc=com** 과 같은 서비스 사용자가 **LDAP** 디렉터리에 있습니다. **Squid**는 이 계정을 인증 사용자를 검색하기 위해서만 사용합니다. 인증 사용자가 있는 경우 **Squid**는 이 사용자로 을 디렉터리에 바인딩하여 인증을 확인합니다.

절차

1. **squid** 패키지를 설치합니다.

```
# yum install squid
```

2. **/etc/squid/squid.conf** 파일을 편집합니다.

   a. **basic_ldap_auth** 도우미 유틸리티를 구성하려면 **/etc/squid/squid.conf**의 맨 위에 다음 설정 항목을 추가하십시오.

```
auth_param basic program /usr/lib64/squid/basic_ldap_auth -b
"cn=users,cn=accounts,dc=example,dc=com" -D
"uid=proxy_user,cn=users,cn=accounts,dc=example,dc=com" -W
```

```
/etc/squid/ldap_password -f "(&(objectClass=person)(uid=%s))" -ZZ -H
ldap://ldap_server.example.com:389
```

다음은 위의 예제에서 **basic_ldap_auth** 도우미 유틸리티에 전달된 매개변수를 설명합니다.

- **-B base_DN** 은 **LDAP** 검색 기반을 설정합니다.
- **-d proxy_service_user_DN** 은 디렉터리의 인증 사용자를 검색하는 데 **Squid** 사용 계정의 **DN**(고유 이름)을 설정합니다.
- **-w path_to_password_file** 은 프록시 서비스 사용자의 암호가 포함된 파일의 경로를 설정합니다. 암호 파일을 사용하면 운영 체제의 프로세스 목록에 암호가 표시되지 않습니다.
- **-f LDAP_filter** 는 **LDAP** 검색 필터를 지정합니다. **Squid**는 인증 사용자가 제공하는 사용자 이름으로 **%s** 변수를 대체합니다.

  예제의 **(&(objectClass=person)(uid=%s))** 필터는 사용자 이름이 **uid** 특성에 설정된 값과 일치해야 하고 디렉터리 항목에 **person** 오브젝트 클래스가 포함되어 있음을 정의합니다.
- **-ZZ** 는 **STARTTLS** 명령을 사용하여 **LDAP** 프로토콜에 **TLS** 암호화 연결을 적용합니다. 다음과 같은 상황에서 **-ZZ** 를 생략합니다.
  - **LDAP** 서버는 암호화된 연결을 지원하지 않습니다.
  - **URL**에 지정된 포트는 **LDAPS** 프로토콜을 사용합니다.
- **H LDAP_URL** 매개 변수는 프로토콜, 호스트 이름 또는 **IP** 주소, **LDAP** 서버의 포트를 **URL** 형식으로 지정합니다.

b. **Squid**가 인증된 사용자만 프록시를 사용하도록 허용하는 다음 **ACL** 및 규칙을 추가합니다.

```
acl ldap-auth proxy_auth REQUIRED
http_access allow ldap-auth
```

중요

***http_access**가 모든 규칙을 거부하기 전에 이러한 설정을 지정합니다.*

c.

*다음 규칙을 제거하여 **localnet ACL**에 지정된 **IP** 범위에서 프록시 인증을 우회하지 않도록 비활성화합니다.*

```
http_access allow localnet
```

d.

*다음 **ACL**은 기본 구성에 있으며 **HTTPS** 프로토콜을 사용하는 포트로 **443** 을 정의합니다.*

```
acl SSL_ports port 443
```

*사용자가 다른 포트에서도 **HTTPS** 프로토콜을 사용할 수 있어야 하는 경우 다음 각 포트에 대한 **ACL**을 추가합니다.*

```
acl SSL_ports port port_number
```

e.

*연결을 설정할 수 있는 포트를 구성하도록 **aclSafe_ports** 규칙 목록을 업데이트합니다. 예를 들어 프록시를 사용하여 클라이언트가 포트 **21(FTP), 80(HTTP)** 및 **443(HTTPS)**의 리소스에만 액세스할 수 있도록 구성하려면 구성에서 다음 **aclSafe_ports** 문만 유지합니다.*

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

*기본적으로 이 구성에는 **Safe_ports ACL**에 정의되지 않은 포트에 대한 액세스 거부를 정의하는 **http_access deny !Safe _ports** 규칙이 포함됩니다.*

f.

***cache** 유형, 캐시 디렉터리의 경로, 캐시 크기 및 **cache _dir** 매개변수의 추가 캐시 유형별 설정을 구성합니다.*

```
cache_dir ufs /var/spool/squid 10000 16 256
```

다음 설정이 필요합니다.

- squid는 ufs 캐시 유형을 사용합니다.
- Squid는 캐시를 /var/spool/squid/ 디렉터리에 저장합니다.
- 캐시는 최대 10000 MB까지 증가합니다.
- squid는 /var/spool/squid/ 디렉토리에 16 level-1 하위 디렉토리를 만듭니다.
- Squid는 각 level-1 디렉토리에 256 개의 하위 디렉토리를 생성합니다.

cache_dir 지시문을 설정하지 않으면 Squid는 캐시를 메모리에 저장합니다.

3. cache _dir 매개변수에서 /var/spool/squid/ 와 다른 캐시 디렉토리를 설정하는 경우:

a. 캐시 디렉토리를 생성합니다.

```
# mkdir -p path_to_cache_directory
```

b. 캐시 디렉토리에 대한 권한을 구성합니다.

```
# chown squid:squid path_to_cache_directory
```

c. 강제 모드에서 SELinux를 실행하는 경우 캐시 디렉터리에 대한 squid_cache_t 컨텍스트를 설정합니다.

```
# semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"
# restorecon -Rv path_to_cache_directory
```

semanage 유틸리티를 시스템에서 사용할 수 없는 경우 policycoreutils-python-utils 패키지를 설치합니다.

4. ***LDAP** 서비스 사용자의 암호를 **/etc/squid/ldap_password** 파일에 저장하고 파일에 적절한 권한을 설정합니다.*

```
# echo "password" > /etc/squid/ldap_password
# chown root:squid /etc/squid/ldap_password
# chmod 640 /etc/squid/ldap_password
```

5. *방화벽에서 **3128** 포트를 엽니다.*

```
# firewall-cmd --permanent --add-port=3128/tcp
# firewall-cmd --reload
```

6. ***squid** 서비스를 시작합니다.*

```
# systemctl start squid
```

7. *시스템이 부팅될 때 **squid** 서비스가 자동으로 시작되도록 활성화합니다.*

```
# systemctl enable squid
```

검증 단계

*프록시가 올바르게 작동하는지 확인하려면 **curl** 유틸리티를 사용하여 웹 페이지를 다운로드합니다.*

```
# curl -O -L "https://www.redhat.com/index.html" -x
"user_name:password@proxy.example.com:3128"
```

***curl** 에서 오류를 표시하지 않고 **index.html** 파일을 현재 디렉터리로 다운로드한 경우 프록시가 작동합니다.*

문제 해결 단계

도우미 유틸리티가 올바르게 작동하는지 확인하려면 다음을 수행하십시오.

1. ***auth_param** 매개변수에 사용한 것과 동일한 설정으로 도우미 유틸리티를 수동으로 시작합니다.*

```
# /usr/lib64/squid/basic_ldap_auth -b "cn=users,cn=accounts,dc=example,dc=com" -D
"uid=proxy_user,cn=users,cn=accounts,dc=example,dc=com" -W /etc/squid/ldap_password -
f "(&(objectClass=person)(uid=%s))" -ZZ -H ldap://ldap_server.example.com:389
```

2. 유효한 사용자 이름과 암호를 입력하고 **Enter** 키를 누릅니다.

```
user_name password
```

도우미 유틸리티에서 **OK** 를 반환하면 인증이 성공했습니다.

16.3. KERBEROS 인증을 사용하여 캐싱 프록시 설정

이 섹션에서는 **Kerberos**를 사용하여 사용자를 **AD(Active Directory)** 인증하는 캐싱 프록시로 **Squid**의 기본 구성에 대해 설명합니다. 이 절차에서는 인증된 사용자만 프록시를 사용할 수 있도록 구성됩니다.

사전 요구 사항

- 이 절차에서는 **/etc/squid/squid.conf** 파일이 **squid** 패키지에서 제공하는 것으로 가정합니다. 이전에 이 파일을 편집한 경우 파일을 제거하고 패키지를 다시 설치합니다.
- **Squid**를 설치할 서버는 **AD** 도메인의 멤버입니다. 자세한 내용은 **Red Hat Enterprise Linux 7** 시스템 관리자 가이드에서 **Samba**를 도메인 멤버로 설정을 참조하십시오.

절차

1. 다음 패키지를 설치합니다.

```
# yum install squid krb5-workstation
```

2. **AD** 도메인 관리자로 인증합니다.

```
# kinit administrator@AD.EXAMPLE.COM
```

3. **Squid**에 사용할 키탭을 생성하고 **/etc/squid/HTTP.keytab** 파일에 저장합니다.

```
# export KRB5_KTNAME=FILE:/etc/squid/HTTP.keytab
# net ads keytab CREATE -U administrator
```

4. **HTTP** 서비스 주체를 **keytab**에 추가합니다.

```
# net ads keytab ADD HTTP -U administrator
```

5. 키탭 파일의 소유자를 **squid** 사용자로 설정합니다.

```
# chown squid /etc/squid/HTTP.keytab
```

6. 필요한 경우 키탭 파일에 프록시 서버의 **FQDN**(정규화된 도메인 이름)에 대한 **HTTP** 서비스 주체가 포함되어 있는지 확인합니다.

```
#  klist -k /etc/squid/HTTP.keytab
Keytab name: FILE:/etc/squid/HTTP.keytab
KVNO Principal
---- --------------------------------------------------------------------------
...
   2 HTTP/proxy.ad.example.com@AD.EXAMPLE.COM
...
```

7. **/etc/squid/squid.conf** 파일을 편집합니다.

   a. **negotiate_kerberos_auth** 도우미 유틸리티를 구성하려면 **/etc/squid/squid.conf**의 맨 위에 다음 설정 항목을 추가하십시오.

```
auth_param negotiate program /usr/lib64/squid/negotiate_kerberos_auth -k
/etc/squid/HTTP.keytab -s HTTP/proxy.ad.example.com@AD.EXAMPLE.COM
```

   다음은 위의 예제에서 **negotiate_kerberos_auth** 도우미 유틸리티에 전달된 매개변수를 설명합니다.

   - **-K** 파일 은 키 탭 파일의 경로를 설정합니다. **squid** 사용자는 이 파일에 대한 읽기 권한이 있어야 합니다.
   - **-s HTTP/host_name@kerberos_realm** 은 **Squid**에서 사용하는 **Kerberos** 주체를 설정합니다.

   선택적으로 다음 매개변수 중 하나 또는 둘 다를 도우미 유틸리티에 전달하여 로깅을 활

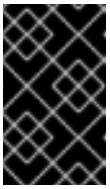
성화할 수 있습니다.

- ***-l**는 인증 사용자와 같은 정보 메시지를 기록합니다.*
- ***-d** 는 디버그 로깅을 활성화합니다.*

***squid**는 도우미 유틸리티의 디버깅 정보를 **/var/log/squid/cache.log** 파일에 기록합니다.*

b. ***Squid**가 인증된 사용자만 프록시를 사용하도록 허용하는 다음 **ACL** 및 규칙을 추가합니다.*

```
acl kerb-auth proxy_auth REQUIRED
http_access allow kerb-auth
```



중요

***http_access**가 모든 규칙을 거부하기 전에 이러한 설정을 지정합니다.*

c. *다음 규칙을 제거하여 **localnet ACL**에 지정된 **IP** 범위에서 프록시 인증을 우회하지 않도록 비활성화합니다.*

```
http_access allow localnet
```

d. *다음 **ACL**은 기본 구성에 있으며 **HTTPS** 프로토콜을 사용하는 포트로 **443** 을 정의합니다.*

```
acl SSL_ports port 443
```

*사용자가 다른 포트에서도 **HTTPS** 프로토콜을 사용할 수 있어야 하는 경우 다음 각 포트에 대한 **ACL**을 추가합니다.*

```
acl SSL_ports port port_number
```

e. *연결을 설정할 수 있는 포트를 구성하도록 **aclSafe_ports** 규칙 목록을 업데이트합니다.*

*예를 들어 프록시를 사용하여 클라이언트가 포트 **21(FTP), 80(HTTP)** 및 **443(HTTPS)**의 리소스에만 액세스할 수 있도록 구성하려면 구성에서 다음 **aclSafe_ports** 문만 유지합니다.*

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

*기본적으로 이 구성에는 **Safe_ports ACL**에 정의되지 않은 포트에 대한 액세스 거부를 정의하는 **http_access deny !Safe _ports** 규칙이 포함됩니다.*

f. *__cache__ 유형, 캐시 디렉터리의 경로, 캐시 크기 및 __cache _dir__ 매개변수의 추가 캐시 유형별 설정을 구성합니다.*

```
cache_dir ufs /var/spool/squid 10000 16 256
```

다음 설정이 필요합니다.

- *__squid__는 __ufs__ 캐시 유형을 사용합니다.*
- *__Squid__는 캐시를 __/var/spool/squid/__ 디렉터리에 저장합니다.*
- *캐시는 최대 __10000 MB__까지 증가합니다.*
- *__squid__는 __/var/spool/squid/__ 디렉토리에 __16 level-1__ 하위 디렉토리를 만듭니다.*
- *__Squid__는 각 __level-1__ 디렉토리에 __256__ 개의 하위 디렉토리를 생성합니다.*

__cache_dir__ 지시문을 설정하지 않으면 __Squid__는 캐시를 메모리에 저장합니다.

8. *__cache _dir__ 매개변수에서 __/var/spool/squid/__ 와 다른 캐시 디렉토리를 설정하는 경우:*

a. *캐시 디렉토리를 생성합니다.*

```
# mkdir -p path_to_cache_directory
```

b. 캐시 디렉토리에 대한 권한을 구성합니다.

```
# chown squid:squid path_to_cache_directory
```

c. 강제 모드에서 **SELinux**를 실행하는 경우 캐시 디렉터리에 대한 **squid_cache_t** 컨텍스트를 설정합니다.

```
# semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"
# restorecon -Rv path_to_cache_directory
```

semanage 유틸리티를 시스템에서 사용할 수 없는 경우 **policycoreutils-python-utils** 패키지를 설치합니다.

9. 방화벽에서 **3128** 포트를 엽니다.

```
# firewall-cmd --permanent --add-port=3128/tcp
# firewall-cmd --reload
```

10. **squid** 서비스를 시작합니다.

```
# systemctl start squid
```

11. 시스템이 부팅될 때 **squid** 서비스가 자동으로 시작되도록 활성화합니다.

```
# systemctl enable squid
```

검증 단계

프록시가 올바르게 작동하는지 확인하려면 **curl** 유틸리티를 사용하여 웹 페이지를 다운로드합니다.

```
# curl -O -L "https://www.redhat.com/index.html" --proxy-negotiate -u : -x
"proxy.ad.example.com:3128"
```

curl 에서 오류를 표시하지 않고 **index.html** 파일이 현재 디렉터리에 있는 경우 프록시가 작동합니다.

문제 해결 단계

***Kerberos** 인증을 수동으로 테스트하려면 다음을 수행합니다.*

1. ***AD** 계정에 대한 **Kerberos** 티켓을 받습니다.*

```
# kinit user@AD.EXAMPLE.COM
```

2. *선택적으로 티켓을 표시합니다.*

```
# klist
```

3. ***negotiate_kerberos_auth_test** 유틸리티를 사용하여 인증을 테스트합니다.*

```
# /usr/lib64/squid/negotiate_kerberos_auth_test proxy.ad.example.com
```

도우미 유틸리티에서 토큰을 반환하면 인증이 성공했습니다.

```
Token: YIIFtAYGKwYBBQUCoIIFqDC...
```

16.4. SQUID에서 도메인 블랙리스트 구성

*관리자는 특정 도메인에 대한 액세스를 차단하려는 경우가 많습니다. 이 섹션에서는 **Squid**에서 도메인 블랙리스트를 구성하는 방법을 설명합니다.*

사전 요구 사항

- ***Squid**가 구성되어 있으며 사용자는 프록시를 사용할 수 있습니다.*

절차

1. ***/etc/squid/squid.conf** 파일을 편집하고 다음 설정을 추가합니다.*

```
acl domain_blacklist dstdomain "/etc/squid/domain_blacklist.txt"
http_access deny all domain_blacklist
```

중요

사용자 또는 클라이언트에 대한 액세스를 허용하는 첫 번째 **http_access allow** 문 앞에 이러한 항목을 추가합니다.

2. **/etc/squid/domain_blacklist.txt** 파일을 만들고 차단할 도메인을 추가합니다. 예를 들어 하위 도메인을 포함한 **example.com** 에 대한 액세스를 차단하고 **example.net** 을 차단하려면 다음을 추가합니다.

```
.example.com
example.net
```

중요

squid 구성에서 **/etc/squid/domain_blacklist.txt** 파일을 참조하는 경우 이 파일은 비워둘 수 없습니다. 파일이 비어 있으면 **Squid**가 시작되지 않습니다.

3. **squid** 서비스를 다시 시작하십시오.

```
# systemctl restart squid
```

16.5. 특정 포트 또는 IP 주소에서 수신 대기하도록 SQUID 서비스 구성

기본적으로 **Squid** 프록시 서비스는 모든 네트워크 인터페이스의 **3128** 포트에서 수신 대기합니다. 이 섹션에서는 포트를 변경하고 특정 **IP** 주소에서 수신 대기하도록 **Squid**를 구성하는 방법에 대해 설명합니다.

사전 요구 사항

- **Squid**가 설치되어 있어야 합니다.

절차

1. **/etc/squid/squid.conf** 파일을 편집합니다.

   - **Squid** 서비스가 수신 대기하는 포트를 설정하려면 **http_port** 매개 변수에서 포트 번호를 설정합니다. 예를 들어 포트를 **8080** 으로 설정하려면 다음을 설정합니다.

```
http_port 8080
```

- **Squid** 서비스가 수신 대기하는 **IP** 주소를 구성하려면 **http_port** 매개 변수에서 **IP** 주소와 포트 번호를 설정합니다. 예를 들어 **Squid**가 포트 **3128** 의 **192.0.2.1 IP** 주소에서만 수신 대기하도록 구성하려면 다음을 설정합니다.

```
http_port 192.0.2.1:3128
```

여러 **http_port** 매개변수를 구성 파일에 추가하여 **Squid**가 여러 포트 및 **IP** 주소에서 수신 대기하도록 구성합니다.

```
http_port 192.0.2.1:3128
http_port 192.0.2.1:8080
```

2. **Squid**가 다른 포트를 기본값으로 사용하도록 구성한 경우**(3128):**

   a. 방화벽에서 포트를 엽니다.

   ```
   # firewall-cmd --permanent --add-port=port_number/tcp
   # firewall-cmd --reload
   ```

   b. 강제 모드에서 **SELinux**를 실행하는 경우 포트를 **squid_port_t** 포트 유형 정의에 할당합니다.

   ```
   # semanage port -a -t squid_port_t -p tcp port_number
   ```

   semanage 유틸리티를 시스템에서 사용할 수 없는 경우 **policycoreutils-python-utils** 패키지를 설치합니다.

3. **squid** 서비스를 다시 시작하십시오.

```
# systemctl restart squid
```

16.6. 추가 리소스

- 자세한 설명과 함께 **/etc/squid/doc/squid-<version>/squid.conf.documented** 파일을 **/etc/squid/squid.conf** 파일에 설정할 수 있습니다.

부록 A. RED HAT CUSTOMER PORTAL LABS 관련 네트워킹

Red Hat 고객 포털 랩은 성능 향상, 문제 해결, 보안 문제 식별 및 구성 최적화를 위해 설계된 툴입니다. 이 부록에서는 네트워킹과 관련된 Red Hat 고객 포털 랩 개요를 제공합니다. 모든 Red Hat 고객 포털 랩은 다음 위치에서 사용할 수 있습니다 https://access.redhat.com/labs/.

브릿지 설정

브리지 구성은 Red Hat Enterprise Linux 5.4 이상을 사용하는 KVM과 같은 애플리케이션에 대해 브리지된 네트워크 인터페이스를 구성하도록 설계되었습니다.

네트워크 연결 도우미

네트워크 본딩 도우미 를 사용하면 관리자가 bonding 커널 모듈과 본딩 네트워크 인터페이스를 사용하여 여러 네트워크 인터페이스 컨트롤러를 단일 채널로 연결할 수 있습니다.

네트워크 본딩 도우미 를 사용하여 두 개 이상의 네트워크 인터페이스가 하나의 본딩 인터페이스로 작동하도록 설정합니다.

패킷 캡처 구문 생성기

패킷 캡처 구문 생성기를 사용하면 네트워크 패킷을 캡처할 수 있습니다.

패킷 캡처 구문 생성기 를 사용하여 인터페이스를 선택한 tcpdump 명령을 생성한 다음 콘솔에 정보를 출력합니다. 명령을 입력하려면 root 액세스 권한이 필요합니다.

부록 B. 개정 내역

고침 0.10-06 **Tue 03 Mar 2020** **Marc Muehlfeld**
*정책 기반 라우팅 구성을 추가하여 대체 경로를 정의할 수*있습니다.

고침 0.10-05 **Fri 22 Nov 2019** **Marc Muehlfeld**
Squid Caching Proxy Server 구성 장을 다시 시작합니다.

고침 0.10-04 **Tue 06 Aug 2019** **Marc Muehlfeld**
7.7 GA 게시 버전.

고침 0.10-03 **Thu 22 Mar 2018** **Ioanna Gkioka**
7.5 GA 게시 버전.

고침 0.10-02 **Mon 14 Aug 2017** **Ioanna Gkioka**
잘못된 업데이트가 포함된 비동기 릴리스

고침 0.10-01 **Tue 25 Jul 2017** **Mirek Jahoda**
7.4 GA 게시 버전.

고침 0.9-30 **Tue 18 Oct 2016** **Mirek Jahoda**
7.3 GA 게시 버전.

고침 0.9-25 **Wed 11 Nov 2015** **Jana Heves**
7.2 GA 릴리스 버전.

고침 0.9-15 **Tue 17 Feb 2015** **Christian Huffman**
7.1 GA 릴리스 버전

고침 0.9-14 **Fri Dec 05 2014** **Christian Huffman**
nmtui 및 NetworkManager GUI 섹션을 업데이트했습니다.

고침 0.9-12 **Wed Nov 05 2014** **Stephen Wadeley**
개선된 *IP 네트워킹,802.1Q VLAN 태깅*및 *티밍*.

고침 0.9-11 **Tues Oct 21 2014** **Stephen Wadeley**
본딩,브리징 및 *티밍*개선.

고침 0.9-9 **Tue Sep 2 2014** **Stephen Wadeley**
*본딩*및 *일관성 네트워크 장치 명명*개선.

고침 0.9-8 **Tue July 8 2014** **Stephen Wadeley**
네트워킹 가이드의 Red Hat Enterprise Linux 7.0 GA 릴리스.

고침 0-0 **Wed Dec 12 2012** **Stephen Wadeley**
Red Hat Enterprise Linux 7 네트워킹 가이드 초기화.

B.1. 감사 인사

색인

Symbols

B

BIND

기능

디렉터리

설정

영역