Red Hat Enterprise Linux 7

ネットワークガイド

RHEL 7 でネットワーク、ネットワークインターフェイス、およびネットワークサービスの設定および管理

Last Updated: 2023-09-21

Red Hat Enterprise Linux 7 ネットワークガイド

RHEL 7 でネットワーク、ネットワークインターフェイス、およびネットワークサービスの設定および管理

Marc Muehlfeld
Red Hat Customer Content Services
mmuehlfeld@redhat.com

Ioanna Gkioka
Red Hat Customer Content Services

Mirek Jahoda
Red Hat Customer Content Services

Jana Heves
Red Hat Customer Content Services

Stephen Wadeley
Red Hat Customer Content Services

Christian Huffman
Red Hat Customer Content Services

法律上の通知

概要

Red Hat Enterprise Linux 7 ネットワークガイドでは、Red Hat Enterprise Linux におけるネットワークインターフェイス、ネットワーク、およびネットワークサービスの設定および管理に関する情報を説明します。本ガイドは、Linux およびネットワークに関する基本的知識があるシステム管理者を対象としています。 専門知識を深めるには、Red Hat システム管理 I (RH124) トレーニングコースの受講を推奨します。

目次

パート I. 作業開始前の準備

ここでは、Red Hat Enterprise Linux のネットワークサービスの基本概念の概要を説明します。

第1章 ネットワークトピックの概要

1.1. IP ネットワークと非 IP ネットワークの比較

ネットワークとは、ファイル、プリンター、アプリケーション、インターネット接続など、情報とリソースを共有して通信できる、相互接続されたデバイスのシステムです。これらの各デバイスには、プロトコルと呼ばれる一連の規則を使用して 2 つ以上のデバイス間でメッセージを送受信する固有のインターネットプロトコル (IP) アドレスがあります。

ネットワーク通信のカテゴリー

IP ネットワーク

インターネットプロトコルアドレスを介して通信するネットワーク。IP ネットワークは、インターネットおよびほとんどの内部ネットワークに実装されています。イーサネット、ケーブルモデム、DSL モデム、ダイヤルアップモデム、無線ネットワーク、VPN 接続などがその代表的な例です。

非 IP ネットワーク

トランスポート層ではなく下位層を介して通信するのに使用されるネットワーク。このネットワークはほとんど使用されないことに注意してください。13章*InfiniBand および RDMA ネットワークの設定*で説明されているように、InfiniBand は非 IP ネットワークです。

1.2. 静的 IP アドレス指定と動的 IP アドレス指定の比較

静的な IP アドレス指定

デバイスに静的 IP アドレスが割り当てられている場合は、そのアドレスを手動で変更しない限り、時間の経過とともに変わることはありません。静的 **IP** アドレス指定の使用が推奨されるのは、次のような場合です。

- **DNS** などのサーバーや認証サーバーのネットワークアドレスの整合性を確保する。
- 他のネットワークインフラストラクチャーから独立して動作する、帯域外管理デバイスを使用する。

「ネットワーク設定方法の選択」に列挙されるすべての設定ツールでは、静的な **IP** アドレスを手動で割り当てることができます。「nmcli を使用した静的イーサネット接続の追加および設定」で説明されている **nmcli** ツールも適しています。

自動設定および管理の詳細は、Red Hat Enterprise Linux 7 システム管理ガイドの『**OpenLMI**』を参照してください。『Red Hat Enterprise Linux 7 インストールガイド』では、ネットワーク設定の割り当てを自動化するのに使用できる **Kickstart** ファイルの使用方法を説明します。

動的な IP アドレス指定

デバイスに動的 IP アドレスが割り当てられている場合は、そのアドレスが時間の経過とともに変わります。このため、マシンを再起動すると IP アドレスが変わる可能性があるため、随時ネットワークに接続するデバイスに使用することが推奨されます。

動的 IP アドレスは、より柔軟で、設定と管理が簡単です。*ダイナミックホストコントロールプロトコル* (DHCP) は、ネットワーク設定をホストに動的に割り当てる従来の方法です。詳細は、「DHCP を使用する理由」を参照してください。また、「nmcli を使用した動的イーサネット接続の追加および設定」で説明されているように、**nmcli** ツールを使用することもできます。

注記

静的 IP アドレスまたは動的 IP アドレスをどのような場合に使用するかを定義する厳密な規則はありません。ユーザーのニーズ、設定、およびネットワーク環境によって異なります。

デフォルトでは、**NetworkManager** は、**DHCP** クライアントである **dhclient** を呼び出します。

1.3. DHCP クライアントの動作の設定

DHCP (Dynamic Host Configuration Protocol) クライアントは、クライアントがネットワークに接続するたびに、動的 IP アドレスと対応する設定情報を DHCP サーバーに要求します。

NetworkManager はデフォルトで **DHCP** クライアントである **dhclient** を呼び出すことに注意してください。

IP アドレスの要求

DHCP 接続が開始すると、dhcp クライアントは **DHCP** サーバーから IP アドレスを要求します。dhcp クライアントがこの要求を完了するのを待つ時間は、デフォルトで 60 秒です。**nmcli** ツールを使用して **ipv4.dhcp-timeout** プロパティーを設定するか、**/etc/sysconfig/network-scripts/ifcfg-*ifname*** ファイルの **IPV4_DHCP_TIMEOUT** オプションを設定できます。たとえば、**nmcli** を使用します。

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

この間にアドレスを取得できないと、IPv4 設定は失敗します。接続全体が失敗する場合もあり、これは **ipv4.may-fail** プロパティーにより異なります。

- **ipv4.may-fail** が **yes**（デフォルト）に設定されている場合、接続の状態は IPv6 設定に依存します。

  1. IPv6 設定が有効であり、これが成功すると、接続はアクティブになりますが、IPv4 設定は再試行できません。

  2. IPv6 設定が無効であるか、または設定されていないと、接続は失敗します。

- **ipv4.may-fail** が **no** に設定されている場合、接続は非アクティブになります。この場合は、以下のようになります。

  1. 接続の **autoconnect** プロパティーが有効になっている場合、**NetworkManager** は、**autoconnect-retries** プロパティーに設定された回数だけ接続のアクティブ化を再試行します。デフォルトでは 4 回です。

  2. それでも接続が dhcp アドレスを取得できないと、自動アクティベーションは失敗します。

     5 分後に自動接続プロセスが再開されて、dhcp クライアントが dhcp サーバーからのアドレスの取得を再試行することに注意してください。

リース更新の要求

dhcp アドレスを取得し、IP アドレスのリースを更新できない場合、dhcp クライアントは 2 分ごとに 3 下位再起動して、dhcp サーバーからリースを取得しようとします。毎回、リースを取得するために **ipv4.dhcp-timeout** プロパティーを秒単位（デフォルトは 60）で設定して設定されます。試行時に応答を得ると、プロセスは停止し、リースが更新されます。

3 回失敗した後:

- **ipv4.may-fail** が **yes**（デフォルト）に設定され、IPv6 が正常に設定されている場合、接続はアクティブになり、dhcp クライアントは 2 分ごとに再起動します。
- **ipv4.may-fail** を **no** に設定すると、接続は非アクティブになります。この場合、接続の **autoconnect** プロパティーが有効になっている場合、接続は最初からアクティブになります。

1.3.1. DHCPv4 の永続化

起動時とリース更新プロセス時に DHCPv4 を永続化するには、**ipv4.dhcp-timeout** プロパティーを 32 ビットの整数(MAXINT32)の最大値 **2147483647** または **infinity** 値に設定します。

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

その結果、**NetworkManager** は、成功するまで DHCP サーバーからのリースの取得または更新の試行を停止しません。

リース更新プロセス中にのみ DHCP の永続的な動作を保証するには、**/etc/sysconfig/network-scripts/ifcfg-*enp1s0*** 設定ファイルまたは **nmcli** を使用して、**IPADDR** プロパティーに静的 IP を手動で追加します。

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

IP アドレスのリース期限が切れると、静的 IP は、設定済みあるいは一部設定済みの IP 状態を保持します。IP アドレスを持つことはできますが、インターネットには接続されていません。dhcp クライアントが 2 分ごとに再起動することを確認してください。

1.4. ワイヤレス規制ドメインの設定

Red Hat Enterprise Linux では、crda パッケージに、特定地区のワイヤレス規制ルールをカーネルに提供する Central Regulatory Domain Agent が含まれています。これは特定の udev スクリプトで使用するもので、udev スクリプトをデバッグする場合以外は手動で実行しないでください。カーネルは、新しい規制ドメインの変更にあたり、udev イベントを送信することで crda を実行します。規制ドメインの変更は、Linux ワイヤレスサブシステム (IEEE-802.11) により起こります。このサブシステムは、regregation **.bin** ファイルを使用して規制データベース情報を維持します。

setregdomain ユーティリティーは、システムの規制ドメインを設定します。**Setregdomain** は引数を取らず、通常は管理者が手動で呼び出すのではなく、udev などのシステムスクリプトを介して呼び出されます。国コードの検索に失敗した場合、システム管理者は **/etc/sysconfig/regdomain** ファイルで **COUNTRY** 環境変数を定義できます。

規制ドメインの詳細は、次の man ページを参照してください。

- **setregdomain (1)** man ページ：国コードに基づいて規制ドメインを設定します。
- **crda (8)** man ページ - 特定の ISO または IEC 3166 alpha2 のワイヤレス規制ドメインをカーネルに送信します。
- **regulatory.bin (5)** man ページ - Linux ワイヤレス規制データベースを表示します。
- man ページの **iw (8)** - ワイヤレスデバイスおよびその設定を表示または操作します。

1.5. NETCONSOLEの設定

ディスクへのログの記録に失敗した場合や、シリアルコンソールを使用できない場合は、カーネルデバッグの使用が必要になる場合があります。**netconsole** カーネルモジュールを使用すると、ネットワークを介して別のコンピューターにカーネルメッセージをログに記録できます。

netconsole を使用できるようにするには、ネットワークに適切に設定された **rsyslog** サーバーが必要です。

手順1.1 netconsole 用 rsyslog サーバーの設定

1. **/etc/rsyslog.conf** ファイルの **MODULES** セクションで次の行のコメントを解除して、514/udp ポートをリッスンし、ネットワークからメッセージを受信するように **rsyslogd** デーモンを設定します。

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

2. **rsyslogd** サービスを再起動して、変更を適用します。

```
]# systemctl restart rsyslog
```

3. **rsyslogd** が 514/udp ポートでリッスンしていることを確認します。

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

netstat -l 出力の **0.0.0.0:syslog** および **[::]:syslog** 値は、**rsyslogd** が **/etc/services** ファイルで定義されたデフォルトの **netconsole** ポートでリッスンしていることを意味します。

```
]$ 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 は、initscripts パッケージの一部である **/etc/sysconfig/netconsole** ファイルを使用して設定されます。このパッケージはデフォルトでインストールされ、**netconsole** サービスも提供します。

送信元マシンを設定する場合は、次の手順に従ってください。

手順1.2 送信元マシンの設定

1. **/etc/sysconfig/netconsole** ファイルの **SYSLOGADDR** 変数の値を、**syslogd** サーバーの IP アドレスに一致するように設定します。以下に例を示します。

```
SYSLOGADDR=192.168.0.1
```

2. 変更を有効にするために **netconsole** サービスを再起動します。

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

3. システムを再起動した後に **netconsole.service** を実行できるようにします。

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

4. クライアントからの **netconsole** メッセージを **/var/log/messages** ファイル（デフォルト）または **rsyslog.conf** で指定されたファイルで表示します。

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

注記

デフォルトでは、**rsyslogd** および **netconsole.service** はポート 514 を使用します。別のポートを使用するには、**/etc/rsyslog.conf** の以下の行を、必要なポート番号に変更します。

```
$UDPServerRun <PORT>
```

送信マシンで、**/etc/sysconfig/netconsole** ファイルで以下の行のコメントを解除して編集します。

```
SYSLOGPORT=514
```

netconsole 設定およびトラブルシューティングのヒントに関する詳細は、Netconsole Kernel Documentation を参照してください。

1.6. SYSCTL によるネットワークカーネル調整パラメーターの使用

sysctl ユーティリティーで特定のカーネルチューナブルを使用すると、実行中のシステムでネットワーク設定を調整し、ネットワークパフォーマンスに直接影響を与えることができます。

ネットワーク設定を変更するには、**sysctl** コマンドを使用します。システムの再起動後も持続する永続的な変更の場合は、**/etc/sysctl.conf** ファイルに行を追加します。

利用可能なすべての **sysctl** パラメーターの一覧を表示するには、**root** で以下を入力します。

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

sysctl を使用したネットワークカーネルパラメーターの詳細は、システム管理者のガイド の複数のインターフェイスでの PTP の使用 を参照してください。

ネットワークカーネルパラメーターの詳細は、カーネル管理ガイドのNetwork Interface Tunables を参照してください。

1.7. NCAT ユーティリティーを使用したデータの管理

ncat ネットワークユーティリティーは、Red Hat Enterprise Linux 7 の **netcat** に代わるものです。**ncat** は、他のアプリケーションやユーザーにネットワーク接続を提供する信頼できるバックエンドツールです。これはコマンドラインからデータを読み取り、書き込みを行い、通信に Transmission Control Protocol (TCP)、User Datagram Protocol (UDP)、Stream Control Transmission Protocol (SCTP)、Unix ソケットを使用します。**ncat** は、**IPv4** と **IPv6** の両方を処理し、接続を開き、パケットを送信し、ポートスキャンを実行し、**SSL**、接続ブローカーなどの高レベルの機能をサポートします。

同じオプションを使用して、**nc** コマンドを **ncat** として入力することもできます。**ncat** オプションの詳細は、『移行計画ガイドの新しいネットワーキングユーティリティー(ncat)および『ncat(1)』 の 』man ページを参照してください。

ncat のインストール

ncat パッケージをインストールするには、**root** で以下のコマンドを実行します。

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

ncat ユースケースの簡単な例

例1.1 クライアントとサーバーとの間の通信の有効化

1. TCP ポート *8080* で接続をリッスンするように、クライアントマシンを設定します。

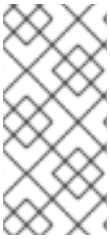
   ```
   ~]$ ncat -l 8080
   ```

2. サーバーマシンで、クライアントの IP アドレスを指定し、同じポート番号を使用します。

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

   接続のいずれの側でもメッセージを送信でき、ローカルマシンとリモートマシンの両方に表示されます。

3. **Ctrl+D** を押して、TCP 接続を閉じます。



注記

UDP ポートを確認するには、**-u** オプションを指定して同じ **nc** コマンドを使用します。以下に例を示します。

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

例1.2 ファイルの送信

前の例で説明したように情報を画面に表示するのではなく、すべての情報をファイルに送信できます。たとえば、TCP ポート *8080* を介してクライアントからサーバーにファイルを送信するには、次の手順を実行します。

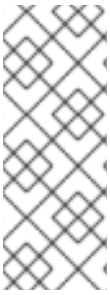
1. クライアントマシンで、ファイルをサーバーマシンに転送する特定のポートをリッスンするには、次のコマンドを実行します。

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

2. サーバーマシンで、クライアントの IP アドレス、ポート、および転送するファイルを指定します。

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

ファイルが転送されると、接続は自動的に閉じます。



注記

他の対象にファイルを転送することもできます。

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

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

例1.3 HTTP プロキシーサーバーの作成

localhost ポート *8080* に HTTP プロキシーサーバーを作成するには、以下を実行します。

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

例1.4 ポートのスキャン

開いているポートを表示するには、**-z** オプションを使用してスキャンするポートの範囲を指定します。

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

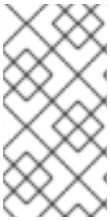
例1.5 SSL を使用した安全なクライアントサーバー通信の設定

サーバーに **SSL** を設定します。

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

クライアントマシンで、次のコマンドを実行します。

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



注記

SSL 接続の真に機密性を確保するには、サーバーには **--ssl-cert** オプションおよび **--ssl-key** オプションが必要で、クライアントには **--ssl-verify** オプションおよび **--ssl-trustfile** オプションが必要です。**OpenSSL** の詳細は、『セキュリティーガイドの OpenSSL の使用 セクションを参照して』ください。

その他の例は、『ncat(1)』 の man ページを参照してください。

パート II. IP ネットワークの管理

ここでは、Red Hat Enterprise Linux でネットワークを設定および管理する方法について詳しく説明します。

第2章 NETWORKMANAGER の使用

2.1. NETWORKMANAGER の概要

Red Hat Enterprise Linux 7 では、**NetworkManager** がデフォルトのネットワークサービスを提供します。これは動的ネットワーク制御および設定デーモンで、ネットワークデバイスと接続が利用可能なときにアクティブな状態を維持します。従来の **ifcfg** タイプの設定ファイルは引き続きサポートされます。詳細は、「ネットワークスクリプトによる NetworkManager の使用」を参照してください。

2.1.1. NetworkManager を使用する利点

NetworkManager を使用する主な利点は、次の通りです。

- ネットワーク管理の容易化： **NetworkManager** は、ネットワーク接続が機能するようにします。システムにネットワーク設定がなく、ネットワークデバイスがあることを検出すると、**NetworkManager** は一時的な接続を作成して接続を提供します。
- ユーザーへの接続設定が容易になります。**NetworkManager** は 、**GUI**、**nmtui**、**nmcli** など、さまざまなツールで管理を提供します。「NetworkManager のツール」を参照してください。
- 柔軟な設定に対応します。たとえば、WiFi インターフェイスを設定すると、**NetworkManager** は利用可能な wifi ネットワークをスキャンして表示します。インターフェイスを選択でき、**NetworkManager** は、再起動プロセス後の自動接続を提供するのに必要な認証情報を表示します。**NetworkManager** は、ネットワークエイリアス、IP アドレス、静的ルート、DNS 情報、VPN 接続、および接続固有のパラメーターを多数設定できます。設定オプションは、必要に応じて修正できます。
- ネットワーク設定と状態についてアプリケーションによるクエリーと制御を可能にする、D-Bus を介した API を提供します。この方法により、アプリケーションは D-BUS を介してネットワークを確認し、制御できます。たとえば、**Web コンソールインターフェイスは、Web** ブラウザーを介してサーバーを監視および設定し、**NetworkManager** D-BUS インターフェイスを使用してネットワークを設定します。
- 再起動プロセス後もデバイスの状態を維持し、再起動中に管理モードに設定されているインターフェイスを引き継ぎます。
- 明示的にマネージド外として設定されていないが、ユーザーまたは他のネットワークサービスによって手動で制御されているデバイスを処理します。

2.2. NETWORKMANAGER のインストール

NetworkManager は、Red Hat Enterprise Linux にデフォルトでインストールされます。そうでない場合は、**root** で以下を入力します。

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

ユーザーの権限および権限の取得に関する詳細情報は、『Red Hat Enterprise Linux システム管理者のガイド』を参照してください。

2.3. NETWORKMANAGER のステータスの確認

NetworkManager が実行しているかどうかを確認するには、次のコマンドを実行します。

```
~]$ 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 が実行されていない場合、**systemctl status** コマンドは **Active: inactive (dead)** と表示されることに注意してください。

2.4. NETWORKMANAGER の開始

NetworkManager を起動するには、次のコマンドを実行します。

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

システムの起動時に NetworkManager を自動的に有効にするには、次のコマンドを実行します。

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

サービスの起動、停止、および管理に関する詳細情報は、『Red Hat Enterprise Linux システム管理者のガイド』を参照してください。

2.5. NETWORKMANAGER のツール

表2.1 NetworkManager のツールとアプリケーションの概要

アプリケーションおよびツール	説明
nmcli	コマンドラインツール。ユーザーとスクリプトが NetworkManager と対話できるようにします。nmcli は、サーバーなどの GUI がないシステムで、NetworkManager のすべての側面を制御することができることに注意してください。GUI ツールのようにさらに高度な機能を提供します。
nmtui	NetworkManager用の単純な curses ベースのテキストユーザーインターフェイス(TUI)
nm-connection-editor	ボンドの設定や接続のチーミングなど、control-center ユーティリティーで処理されていない特定のタスク用のグラフィカルユーザーインターフェイスツール。NetworkManager が保存したネットワーク接続を追加、削除、および変更できます。これを起動するには、端末に nm-connection-editor と入力します。 `~]$ nm-connection-editor`
control-center	GNOME Shell が提供し、デスクトップユーザーが利用可能なユーザーインターフェイス。これには、ネットワーク設定ツールが含まれます。これを起動するには、**Super** キーを押してアクティビティーの概要に入り、Network と **入力** して Enter を押します。Network 設定ツールが表示されます。

アプリケーションおよびツール	説明
ネットワーク接続アイコン	GNOME Shell が提供するグラフィカルユーザーインターフェイスツールで、**NetworkManager** が報告するネットワーク接続の状態を表します。アイコンには複数の状態があり、現在使用中の接続の種類を視覚的に表示します。

2.6. ネットワークスクリプトによる NETWORKMANAGER の使用

本セクションは、スクリプトの実行方法と、ネットワークスクリプトでカスタムコマンドを使用する方法を説明します。

ネットワークスクリプト という用語は、**/etc/init.d/network** スクリプトと、それが呼び出すその他のインストール済みスクリプトを指します。**NetworkManager** はデフォルトのネットワークサービスを提供しますが、スクリプトと **NetworkManager** は並行して実行され、連携することができます。Red Hat は、それを最初にテストすることを推奨します。

ネットワークスクリプトの実行

systemctl コマンド ***のみ*** を使用してネットワークスクリプトを実行します。

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

systemctl ユーティリティーは既存の環境変数をクリアし、正しい実行を確保します。

Red Hat Enterprise Linux 7 では、 **NetworkManager** が最初に起動し、**/etc/init.d/network** が NetworkManager をチェックして、**NetworkManager** の接続の改ざんを回避します。**NetworkManager** は、sysconfig 設定ファイルを使用するプライマリーアプリケーションであり、**/etc/init.d/network** はセカンダリーとなることを目的としています。

/etc/init.d/network スクリプトは、以下を実行します。

1. 手動で - **systemctl** コマンド **start|stop|restart network** のいずれかを使用して、

   または

2. ネットワークサービスが有効な場合は、起動およびシャットダウン時 - **systemctl enable network** コマンドの結果として。

これは手動のプロセスで、起動後に発生するイベントに反応しません。また、ユーザーは **ifup** スクリプトおよび **ifdown** スクリプトを手動で呼び出すこともできます。

注記

initscripts の技術的制限により、**systemctl reload network.service** コマンドは機能しません。ネットワークサービスに新しい設定を適用するには、**restart** コマンドを使用します。

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

これにより、新しい設定を読み込むために、ネットワークインターフェイスカード (NIC) をすべて無効にして有効にします。詳細は、Red Hat ナレッジベースソリューション Reload and force-reload options for network service を参照してください。

ネットワークスクリプトのカスタムコマンドを使用

/sbin/ifup-local スクリプト、**ifdown-pre-local** スクリプト、および **ifdown-local** スクリプトのカスタムコマンドは、これらのデバイスが **/etc/init.d/network** サービスによって制御されている場合にのみ実行されます。**ifup-local** ファイルは、デフォルトでは存在しません。必要に応じて、**/sbin/** ディレクトリーの下に作成します。

ifup-local スクリプトは initscripts によってのみ読み取りでき、**NetworkManager** は読み取りできません。**NetworkManager** を使用してカスタムスクリプトを実行するには、**dispatcher.d/** ディレクトリーの下に作成します。「dispatcher スクリプトの実行」を参照してください。

重要

initscripts パッケージに関連する rpms に含まれるファイルの変更は推奨されません。ファイルを変更した場合は、Red Hat サポートの対象外となります。

カスタムタスクは、ネットワーク接続がアップまたはダウンしたときに、古い **ネットワークスクリプト** と **NetworkManager** の両方を使用して実行できます。**NetworkManager** が有効な場合は、**ifup** スクリプトおよび **ifdown** スクリプトにより、**NetworkManager** が問題のインターフェイスを管理するかどうかを **NetworkManager** に尋ねます。これは、**ifcfg** ファイルの 「DEVICE=」 行にあります。

NetworkManager が管理するデバイス：

ifup の呼び出し

ifup を呼び出して、デバイスが **NetworkManager** で **管理** されている場合は、以下の 2 つのオプションがあります。

- デバイスがまだ接続されてい **ない** 場合は、**NetworkManager** に接続を開始するように要求します。
- **デバイスがすでに 接続している 場合は、何もする必要がありません。**

ifdown の呼び出し

ifdown を呼び出して、デバイスが NetworkManager により 管理 されます。

- **ifdown により、NetworkManager が接続を終了するように求められます。**

NetworkManager が管理していないデバイス：

ifup または ifdown のいずれかを呼び出すと、このスクリプトは、NetworkManager が存在してから使用した、古い NetworkManager 以外のメカニズムを使用して接続を開始します。

dispatcher スクリプトの実行

NetworkManager は、追加のカスタムスクリプトを実行して、接続の状態に基づいてサービスを開始または停止する方法を提供します。デフォルトでは、/etc/NetworkManager/dispatcher.d/ ディレクトリーが存在し、NetworkManager はアルファベット順にそこにあるスクリプトを実行します。各スクリプトは、root が所有する実行可能ファイルであり、ファイル 所有 者に対してのみ 書き込み権限 を持っている必要があります。NetworkManager の dispatcher スクリプト実行の詳細は、Red Hat ナレッジベースソリューションethtool コマンドを適用するように NetworkManager の dispatcher スクリプトを記述するを参照してください。

2.7. SYSCONFIG ファイルによる NETWORKMANAGER の使用

/etc/sysconfig/ ディレクトリーは、設定ファイルおよびスクリプトの場所です。ほとんどのネットワーク設定情報がここに保存されます。ただし、VPN、モバイルブロードバンド、および PPPoE の設定を除き、/etc/NetworkManager/ サブディレクトリーに保存されます。たとえば、インターフェイス固有の情報は、/etc/sysconfig/network-scripts/ ディレクトリーの ifcfg ファイルに保存されます。

グローバル設定には、/etc/sysconfig/network ファイルを使用します。VPN、モバイルブロードバンド、および PPPoE 接続に関する情報は /etc/NetworkManager/system-connections/ に保存されます。

Red Hat Enterprise Linux 7 では、ifcfg ファイルを編集しても、NetworkManager は自動的に変更を認識しないため、変更を通知する必要があります。NetworkManager プロファイル設定を更新するツールのいずれかを使用すると、NetworkManager は、そのプロファイルを使用して再接続するまでこれらの変更を実装しません。たとえば、エディターを使用して設定ファイルを変更した場合、NetworkManager は設定ファイルを再度読み込む必要があります。

これを確認するには、root で と入力して、すべての接続プロファイルをリロードします。

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

または、変更したファイル ifcfg-*ifname* を 1 つだけ 再読み込みするには、次のコマンドを実行します。

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

上記のコマンドを使用して複数のファイル名を指定できることに注意してください。

nmcli などのツールを使用して変更した変更はリロードする必要はありませんが、関連するインターフェイスをダウンしてから再度起動する必要があります。

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

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

nmcli の詳細は、「nmcli を使用する IP ネットワークの設定」 を参照してください。

NetworkManager は、ネットワークスクリプトをトリガーしません。ただし、ネットワークスクリプトは、ifup コマンドの使用時に実行中の場合に NetworkManager をトリガーしようとします。ネットワークスクリプトの詳細は、「ネットワークスクリプトによる NetworkManager の使用」を参照してください。

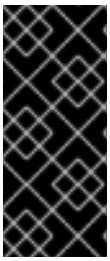
ifup スクリプトは汎用スクリプトで、いくつかのことを実行し、ifup-*device_name*、ifup-wireless、ifup-ppp などのインターフェイス固有のスクリプトを呼び出します。ユーザーが ifup enp1s0 を手動で実行すると、以下を行います。

1. ifup が /etc/sysconfig/network-scripts/ifcfg-enp1s0 というファイルを探します。
2. ifcfg ファイルが存在する場合、ifup はそのファイル内の TYPE キーを検索し、呼び出すタイプ固有のスクリプトを特定します。
3. ifup は、TYPE に基づいて ifup-wireless または ifup-*device_name* を呼び出します。
4. タイプ固有のスクリプトがタイプ固有のセットアップを実行します。
5. タイプ固有のスクリプトにより、共通関数が DHCP や静的セットアップなどの IP関連の

タスクを実行できます。

/etc/init.d/network は起動時にすべての ifcfg ファイルを読み取り、ONBOOT=yes になっている各ファイルについて、NetworkManager がすでに ifcfg ファイルから DEVICE を起動しているかどうかを確認します。NetworkManager がそのデバイスを起動するか、またはすでに起動している場合は、そのファイルに対しては何も実行されず、次の ONBOOT=yes ファイルもチェックされます。NetworkManager がまだそのデバイスを起動していない場合は、initscripts は従来の動作を続行し、その ifcfg ファイルの ifup を呼び出します。

その結果、NetworkManager または initscripts のいずれかで、ONBOOT=yes が含まれる ifcfg ファイルがシステム起動時に開始されることが予想されます。これにより、 NetworkManager が処理しないレガシーネットワークタイプ(ISDN、analog dial-up modems など)や、NetworkManager でまだサポートされていない新しいアプリケーションは、NetworkManager が処理できない場合でも initscripts によって正常に起動されます。



重要

スクリプトは ifcfg-* を文字通り行うため、バックアップファイルは /etc ディレクトリー内のどこか、またはライブファイルと同じ場所に保存しないことが推奨されます。.old、.orig、.rpmnew、.rpmorig、.rpmsave の拡張機能のみが除外されます。

sysconfig ファイルの使用方法は、「ifcfg ファイルを使用した IP ネットワークの設定」 および『ifcfg(8)』 の man ページを参照してください。

2.8. 関連情報

- man (1) man ページ：man ページとその検索方法が説明されています。
- NetworkManager (8) man ページ - ネットワーク管理デーモンを説明しています。
- NetworkManager.conf (5) man ページ - NetworkManager 設定ファイルが説明されています。
- /usr/share/doc/initscripts-*version*/sysconfig.txt: 従来のネットワークサービスが理解できるように ifcfg 設定ファイルとそのディレクティブについて説明しています。

- /usr/share/doc/initscripts-*version*/examples/networking/: 設定ファイルのサンプルが含まれるディレクトリー。

- ifcfg (8) man ページ - ifcfg コマンドについて簡単に説明しています。

第3章 IP ネットワークの設定

システム管理者は、NetworkManager を使用してネットワークインターフェイスを設定できます。

3.1. ネットワーク設定方法の選択

- NetworkManager を使用してネットワークインターフェイスを設定するには、以下のいずれかのツールを使用します。
  - テキスト形式のユーザーインターフェイスツール nmtui。詳細は、「nmtui を使用した IP ネットワークの設定」を参照してください。
  - コマンドラインツール nmcli詳細は、「nmcli を使用する IP ネットワークの設定」を参照してください。
  - グラフィカルユーザーインターフェイスツール GNOME GUI詳細は、「 GNOME GUI を使用した IP ネットワークの設定 」を参照してください。
- NetworkManager を使用 せず にネットワークインターフェイスを設定するには、以下を行います。
  - ifcfg ファイルを手動で編集します。詳細は、「ifcfg ファイルを使用した IP ネットワークの設定」を参照してください。
  - ip コマンドを使用します。IP アドレスをインターフェイスに割り当てるのに使用できますが、変更は再起動をまたぐ永続的なものではありません。詳細は、「ip コマンドを使用した IP ネットワークの設定」を参照してください。
- root ファイルシステムがローカルでは ない 場合にネットワークを設定するには、以下の手順を行います。

- カーネルのコマンドラインを使用する。詳細は、「カーネルコマンドラインから IP ネットワークの設定」を参照してください。

3.2. NMTUI を使用した IP ネットワークの設定

システム管理者は、NetworkManager のツール nmtui を使用してネットワークインターフェイスを設定できます。「NetworkManager のツール」を参照してください。

この手順では、テキストユーザーインターフェイスツール nmtui を使用してネットワークを設定する方法を説明します。

前提条件

- 端末ウィンドウで nmtui ツールを使用する。NetworkManager-tui パッケージに含まれていますが、デフォルトでは NetworkManager と一緒にインストールされません。NetworkManager-tui をインストールするには、次のコマンドを実行します。

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

- NetworkManager が実行していることを確認するには、「NetworkManager のステータスの確認」 を参照してください。

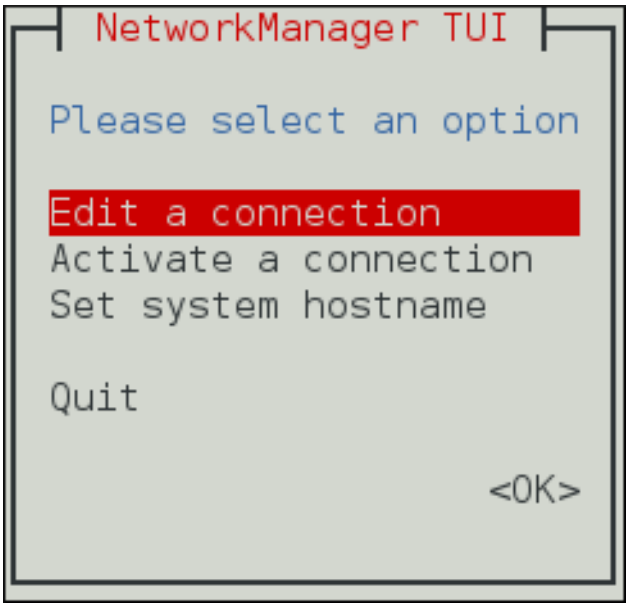
手順

1. nmtui ツールを起動します。

```
~]$ nmtui
```

テキストユーザーインターフェイスが表示されます。

図3.1 NetworkManager のテキスト形式ユーザーインターフェイスの開始メニュー



[D]

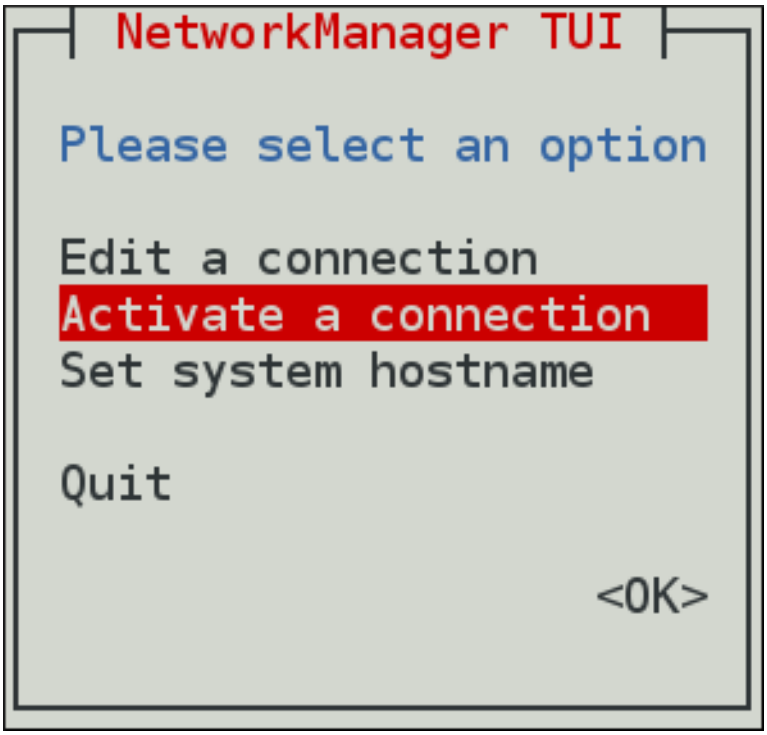
2. 移動するには、矢印キーを使用するか、Tab を押して順方向に進み、Shift+Tab を押してオプションを再度実行します。Enter を押してオプションを選択します。Space バーは、チェックボックスのステータスを切り替えます。

すでにアクティブな接続に加えた変更を適用するには、接続を再アクティブすることが必要です。この場合は、以下の手順を実施します。

手順

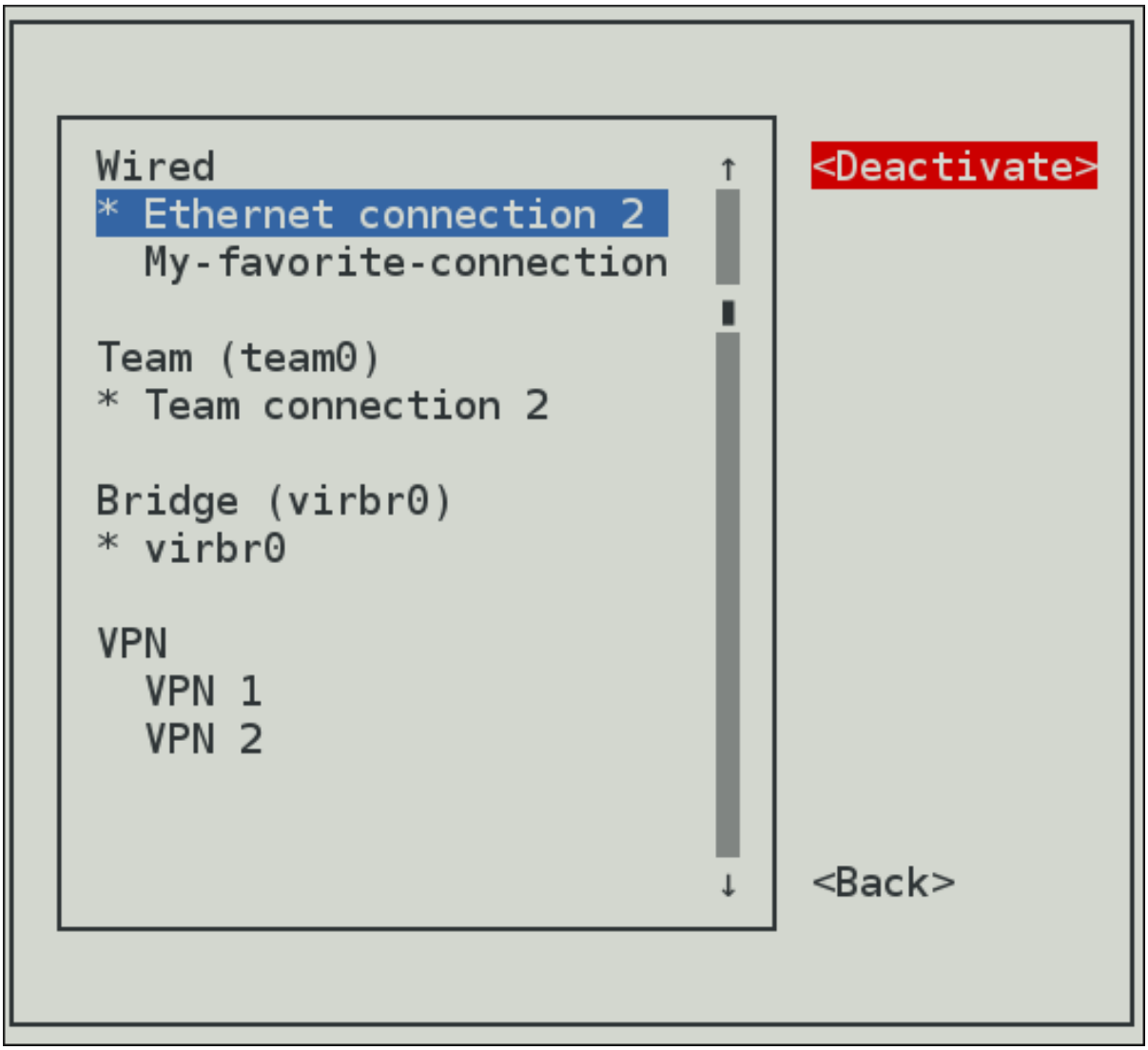
1. Activate a connection メニューエントリーを選択します。

図3.2 接続のアクティブ化



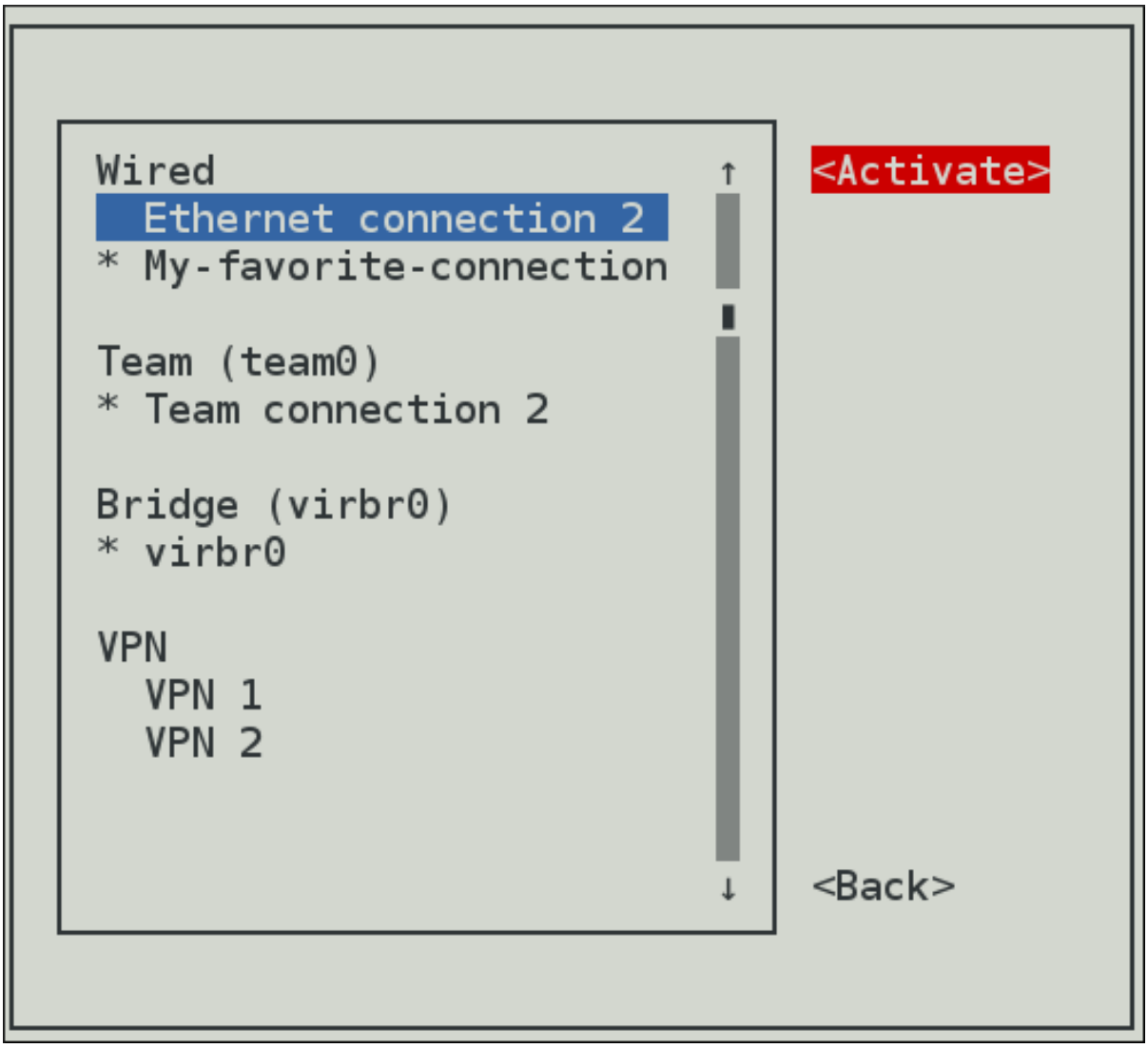
2. **修正した接続を選択します。右側の Deactivate ボタンをクリックします。**

図3.3 修正した接続の非アクティブ化



3. **接続を再度選択し、Activate ボタンをクリックします。**

図3.4 修正した接続の再アクティブ化



以下のコマンドも利用できます。

- nmtui edit *connection-name*

  接続名が指定されていない場合、選択メニューが表示されます。接続名が指定され、正しく特定されると、関連する Edit connection 画面が表示されます。

- nmtui connect *connection-name*

  接続名が指定されていない場合、選択メニューが表示されます。接続名を指定して正しく特定

されると、関連する接続がアクティブになります。無効なコマンドがあると、使用方法に関するメッセージが表示されます。

nmtui は、すべての種類の接続に対応しているわけではないことに注意してください。特に、VPN、WPA Enterprise を使用したワイヤレスネットワーク接続、802.1X を使用したイーサネット接続は編集できません。

3.3. NMCLI を使用する IP ネットワークの設定

nmcli (NetworkManager コマンドラインインターフェイス)コマンドラインユーティリティーは、NetworkManager を制御し、ネットワークステータスの報告に使用されます。これは、nm-applet またはその他のグラフィカルクライアントの代わりに使用できます。「NetworkManager のツール」を参照してください。nmcli は、ネットワーク接続の作成、表示、編集、削除、有効化、非アクティブ化、ネットワークデバイスのステータスの制御と表示に使用されます。

nmcli ユーティリティーは、NetworkManager を制御するためにユーザーとスクリプトの両方で使用できます。

- サーバー、ヘッドレスマシン、および端末の場合は、nmcli を使用して、GUI を使用せずに NetworkManager を直接制御することができます。これには、ネットワーク接続の作成、編集、開始、および停止やネットワークステータスの表示が含まれます。

- スクリプトの場合、nmcli は簡潔な出力形式をサポートします。これはスクリプト処理に適しています。この場合、ネットワーク接続を手動で管理するのではなく、ネットワーク設定の整合性を維持するために用いられます。

nmcli コマンドの基本的な形式は次のとおりです。ここでの OBJECT は、一般的 な、ネットワーク、ラジオ、接続、デバイス、エージェント、および 監視 のいずれかのオプションになります。

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

コマンドには、このいずれかの接頭辞を使用できます。たとえば、nmcli con help、nmcli c help、nmcli connection help は、同じ出力を生成します。

OPTIONS で便利なオプションは以下のとおりです。

-t (terse)

このモードは、コンピューターのスクリプト処理に使用される場合があり、値だけを表示する簡潔な出力を確認できます。

例3.1 簡潔な出力の表示

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

-f (field)

このオプションでは、どのフィールドを出力に表示できるかを指定します。たとえば、NAME、UUID、TYPE、AUTOCONNECT、ACTIVE、DEVICE、STATE です。フィールドは、1 つまたは複数使用できます。複数のフィールドを使用する場合は、フィールドを区切るコンマの後に空白を入力しないでください。

例3.2 出力内のフィールドの指定

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

また、次のようなスクリプトの記述に適しています。

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

-p (pretty)

このオプションにより、nmcli は人間が判読できる出力を生成します。たとえば、値を揃え、ヘッダーを表示します。

例3.3 pretty モードで出力の表示

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

-h (help)

ヘルプ情報を表示します。

nmcli ツールには、コンテキスト依存ヘルプが組み込まれています。

nmcli help

このコマンドでは、その後のコマンドで使用される利用可能なオプションおよびオブジェクト名のリストが表示されます。

nmcli *object* help

このコマンドでは、指定したオブジェクトに関する利用可能なアクションのリストが表示されます。以下に例を示します。

```
nmcli c help
```

3.3.1. nmcli の簡単な選択例

例3.4 NetworkManager の全体ステータスの確認

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

簡潔モードの場合は、以下のようになります。

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

例3.5 NetworkManager のロギングの状態の表示

```
~]$ 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,CONCHE
CK,DC
B,DISPATCH
```

例3.6 すべての接続を表示

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

例3.7 現在アクティブな接続のみを表示

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

例3.8 NetworkManager が認識するデバイスとその状態のみの表示

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

nmcli コマンドの以下の省略形を使用することもできます。

表3.1 nmcli コマンドの省略形の例

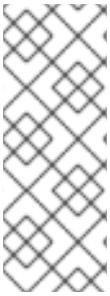
nmcli コマンド	省略形
nmcli general status	nmcli g
nmcli general logging	nmcli g log
nmcli connection show	nmcli con show
nmcli connection show --active	nmcli con show -a
nmcli device status	nmcli dev

その他の例は、『nmcli-examples(5)』 の man ページを参照してください。

3.3.2. nmcli を使用したネットワークインターフェイスの起動および停止

nmcli ツールを使用すると、コントローラーを含むネットワークインターフェイスを起動および停止できます。以下に例を示します。

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



注記

nmcli connection down コマンドは、デバイスをさらに自動アクティブ化しないようにすることなく、デバイスからの接続を非アクティブにします。nmcli device disconnect コマンドは、デバイスを切断し、手動で介入することなく、デバイスが自動的に接続を自動的にアクティベートしないようにします。

3.3.3. nmcli オプションについて

nmcli の重要なプロパティーオプションを以下に示します。『nmcli(1)』 の man ページの包括的な一覧を参照してください。

connection.type

接続の種類です。設定可能な値は、adsl、bond、bond-slave、bridge、bridge-slave、bluetooth、cdma、ethernet、gsm、infiniband、olpc-mesh、team、team-slave、vlan、wifi、wimax です。各接続タイプには、タイプ固有のコマンドオプションがあります。TYPE_SPECIFIC_OPTIONS の一覧は、『nmcli(1)』 の man ページで確認できます。以下に例を示します。

- gsm 接続では、アクセスポイント名を apn に指定する必要があります。

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

- wifi デバイスには、ssid に指定されるサービスセット ID が必要です。

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

connection.interface-name

接続に関連するデバイス名。

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

connection.id

接続プロファイルに使用される名前。接続名を指定しないと、次のように接続名が生成されます。

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

connection.id は接続 *プロファイル* の名前です。デバイスを示すインターフェイス名 (wlp61s0、ens3、em1)と混同しないようにしてください。なお、ユーザーはインターフェイスと同じ名前を接続に付けることができますが、これは別のものです。1 つのデバイスに複数の接続プロファイルを利用することもできます。これは、モバイルデバイスの場合や異なるデバイス間でネットワークケーブルを切り替える場合に非常に便利です。必要に応じて、設定を編集するのではなく、異なるプロファイルを作成してインターフェイスに適用します。id オプションも接続プロファイル名を参照します。

show、up、down などの nmcli コマンドで最も重要なオプションは次のとおりです。

id

ユーザーが接続プロファイルに割り当てる識別用文字列。nmcli connection コマンドで、ID を使用して接続を指定できます。コマンド出力の NAME フィールドには、必ず接続 ID が表示されます。con-name が参照するのと同じ接続プロファイル名が参照されます。

uuid

システムが接続プロファイルに割り当てる一意の識別用文字列。nmcli connection コマンドで uuid を使用して、接続を特定できます。

3.3.4. nmcli インタラクティブ接続エディターの使用

nmcli ツールには、インタラクティブな接続エディターがあります。使用するには、次のコマンドを実行します。

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

表示されたリストから有効な接続の種類を入力するよう求められます。接続の種類を入力すると、nmcli プロンプトが表示されます。接続の種類に精通している場合は、nmcli con edit コマンドに有効な接続 タイプ オプションを追加して、nmcli プロンプトに直接取り込むことができます。既存の接続プロファイルの編集には、次の形式になります:

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

新しい接続プロファイルを編集します。

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

有効なコマンドの一覧を表示するには、nmcli プロンプトで help と入力します。describe コマンドを使用して、設定とそのプロパティーの説明を取得します(

```
describe setting.property
```

)。以下に例を示します。

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

3.3.5. nmcli による接続プロファイルの作成および修正

接続プロファイルには、データソースへの接続に必要な接続プロパティー情報が含まれています。

nmcli を使用して NetworkManager に新しいプロファイルを 作成 するには、以下のコマンドを実行します。

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

nmcli c add では、2 種類のパラメーターを使用できます。

プロパティー名

接続を内部的に記述するために NetworkManager が使用する名前。最も重要なものを以下に示します。

- 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"
```

プロパティーとその設定の詳細は、man ページの nm-settings (5) を参照してください。

エイリアス名

内部的にプロパティーに翻訳された、人間が理解可能な名前。最も一般的なものを以下に示します。

- タイプ(connection.type プロパティー)

```
nmcli c add type bond
```

- ifname (connection.interface-name プロパティー)

```
nmcli c add ifname enp1s0
```

- **con-name (connection.id プロパティー)**

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

以前のバージョンの nmcli では、接続を作成するには、エイリアス を使用する必要がありました。たとえば、ifname enp1s0 および con-name My Connection などです。次の形式のコマンドを使用できます。

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

より新しいバージョンでは、プロパティー名 と エイリアス の両方を同じ意味で使用できます。以下は、すべて有効であり、同等です。

```
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
```

引数は、接続の種類によって異なります。type 引数のみがすべての接続タイプに必須であり、ifname は ボンディング、チーム、ブリッジ、および vlan を除くすべてのタイプで必須です。

type (*type_name*)

接続の種類です。以下に例を示します。

```
nmcli c add type bond
```

–

ifname (*interface_name*)

接続のバインド先となるインターフェイスです。例を以下に示します。

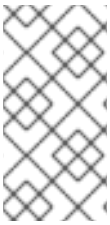
```
nmcli c add ifname interface_name type ethernet
```

接続プロファイルの 1 つまたは複数のプロパティーを 修正する には、以下のコマンドを使用します。

```
nmcli c modify
```

たとえば、connection.id を My Connection から My favorite connection に変更し、connection.interface-name を enp1s0 に変更するには、以下のコマンドを実行します。

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



注記

プロパティー名 を使用することが推奨されます。エイリアス は、互換性の理由でのみ使用されます。

また、イーサネット MTU を 1600 に設定するには、以下のようにサイズを変更します。

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

nmcli を使用して接続を修正した後に変更を適用するには、以下のコマンドを入力して接続を再度アクティブ化します。

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

以下に例を示します。

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

3.3.6. nmcli を使用したネットワーク接続

現在利用可能なネットワーク接続をリスト表示するには、以下を実行します。

```
~]$ 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
```

出力の NAME フィールドは常に接続 ID （名前）を示すことに注意してください。これはインターフェイス名と同じように見えますが、異なるものです。上記の 2 つ目の接続では、NAME フィールドの ens3 は、ユーザーがインターフェイスに適用されるプロファイルに指定した接続 ID です。 ens3.最後の接続では、ユーザーは接続 ID MyWiFi をインターフェイスに割り当てています。 wlp61s0.

イーサネット接続を追加すると、設定プロファイルが作成され、それがデバイスに割り当てられます。新規プロファイルを作成する前に、以下のように利用可能なデバイスを確認します。

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

3.3.7. nmcli を使用した動的イーサネット接続の追加および設定

動的イーサネット接続を追加する

動的 IP 設定でイーサネット設定プロファイルを追加するには、DHCP がネットワーク設定を割り当てるのを許可します。

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

たとえば、*my-office* という名前の動的接続を作成するには、以下のコマンドを実行します。

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

イーサネット接続を開くには、以下のコマンドを実行します。

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

デバイスおよび接続のステータスを確認します。

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

動的イーサネット接続の設定

ホストから DHCP サーバーに送信したホスト名を変更するには、dhcp-hostname プロパティーを修正します。

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

ホストから DHCP サーバーに送信した IPv4 クライアント ID を変更するには、dhcp-client-id プロパティーを修正します。

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

IPv6 には dhcp-client-id プロパティーがなく、dhclient は IPv6 の識別子を作成します。詳細は、man ページの dhclient (8) を参照してください。

DHCP サーバーでホストに送信された DNS サーバーを無視するには、ignore-auto-dns プロパティーを変更します。

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

プロパティーとその設定の詳細は、man ページの nm-settings (5) を参照してください。

例3.9 インタラクティブエディターを使用した動的イーサネット接続の設定

インタラクティブエディターを使用して動的イーサネット接続を設定するには、次のコマンドを実行します。

```
~]$ 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
~]$
```

デフォルトの動作では、接続プロファイルが永続的に保存されます。必要な場合は、save

temporary コマンドで、次回の再起動時まで、プロファイルをメモリーにのみ保持できます。

3.3.8. nmcli を使用した静的イーサネット接続の追加および設定

静的イーサネット接続の追加

静的 IPv4 設定でイーサネット接続を追加するには、

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

IPv6 アドレスとゲートウェイ情報を ip6 および gw6 オプションを使用して追加できます。

たとえば、IPv4 アドレスとゲートウェイのみを使用して静的イーサネット接続を作成するには、次のコマンドを実行します。

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

必要に応じて、デバイスの IPv6 アドレスとゲートウェイを指定します。

```
~]$ 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.
```

2 つの IPv4 DNS サーバーアドレスを設定するには、次のコマンドを実行します。

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

これにより、以前に設定された DNS サーバーが置き換えられることに注意してください。2 つの IPv6 DNS サーバーアドレスを設定するには、次のコマンドを実行します。

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

これにより、以前に設定された DNS サーバーが置き換えられることに注意してください。または、以前のセットに DNS サーバーを追加するには、+ 接頭辞を使用します。

```
~]$ 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"
```

新しいイーサネット接続を開くには、以下のコマンドを実行します。

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

デバイスおよび接続のステータスを確認します。

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

新規に設定した接続の詳細情報を表示するには、以下のコマンドを実行します。

```
~]$ 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 オプションを使用すると、出力にタイトルバナーとセクション区切りが追加されます。

例3.10 インタラクティブエディターを使用した静的イーサネット接続の設定

インタラクティブエディターを使用した静的イーサネット接続を設定するには、以下のコマンドを実行します。

```
~]$ 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
~]$
```

デフォルトの動作では、接続プロファイルが永続的に保存されます。必要な場合は、save temporary コマンドで、次回の再起動時まで、プロファイルをメモリーにのみ保持できます。

NetworkManager は、内部パラメーター connection.autoconnect を yes に設定します。NetworkManager は、設定を /etc/sysconfig/network-scripts/ifcfg-my-office に書き込みます。ここで、対応する BOOTPROTO は none に、ONBOOT は yes に設定されます。

ifcfg ファイルへの手動の変更は、インターフェイスが次に起動するまで NetworkManager では認識されません。設定ファイルの使用方法については、「sysconfig ファイルによる NetworkManager の使用」、「ifcfg ファイルを使用した IP ネットワークの設定」を参照してください。

3.3.9. nmcli を使用してプロファイルを特定のデバイスにロック

特定のインターフェイスデバイスにプロファイルをロックするには、

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

で行います。すべての互換性のあるイーサネットインターフェイスで利用可能なプロファイルを指定するには、

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

を使用します。特定のインターフェイスを設定しなくても、ifname 引数を使用する必要があることに注意してください。ワイルドカード文字 * を使用して、プロファイルと互換性のあるデバイスを使用できることを指定します。

プロファイルを特定の MAC アドレスにロックするには、次のコマンドを実行します。

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

3.3.10. nmcli を使用する Wi-Fi 接続の追加

利用可能な Wi-Fi アクセスポイントを表示するには、次のコマンドを実行します。

```
~]$ 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
```

静的 IP 設定で Wi-Fi 接続プロファイルを作成し、DNS アドレスの自動割り当てを許可するには、以下を実行します。

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

WPA2 パスワード (例: 「caffeine」) を設定するには、以下のコマンドを実行します。

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

パスワードのセキュリティーに関する情報は、『Red Hat Enterprise Linux 7 セキュリティーガイド』を参照してください。

Wi-Fi 状態を変更するには、以下のコマンドを実行します。

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

nmcli を使用した特定プロパティーの変更

特定のプロパティー(mtu など)を確認するには、以下を実行します。

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

設定のプロパティーを変更するには、次のコマンドを実行します。

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

変更を確認するには、次のコマンドを実行します。

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

NetworkManager は、設定の 802 -3-ethernet および 802- 11-wireless、および 設定のプロパティーとしての mtu などのパラメーターを参照します。プロパティーとその設定の詳細は、man ページの nm-settings (5) を参照してください。

3.3.11. 特定のデバイスを無視するように NetworkManager を設定

デフォルトでは、NetworkManager は lo (loopback)デバイス以外のすべてのデバイスを管理します。ただし、特定のデバイスを unmanaged に設定して、NetworkManager がこのデバイスを無視するように設定できます。この設定では、スクリプトなどを使用して、このデバイスを手動で管理できます。

3.3.11.1. NetworkManager で、デバイスを Unmanaged (管理外) として永続的に設定

インターフェイス名、MAC アドレス、デバイスタイプなど、複数の基準に基づいてデバイスを 管理対象外 として設定できます。この手順では、NetworkManager で、enp1s0 インターフェイスを unmanaged として永続的に設定する方法を説明します。

一時的にネットワークデバイスを 管理対象外 として設定する場合は、「NetworkManager でデバイスの非管理として一時的に設定」 を参照してください。

手順

1. オプション：デバイスの一覧を表示して、**unmanaged** に設定するデバイスを特定します。

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

2. 以下の内容で **/etc/NetworkManager/conf.d/99-unmanaged-devices.conf** ファイルを作成します。

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

複数のデバイスを **unmanaged** に設定するには、***unmanaged-devices*** パラメーターのエントリーをセミコロンで区切ります。

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

3. **NetworkManager** サービスを再読み込みします。

```
# systemctl reload NetworkManager
```

検証手順

- デバイスのリストを表示します。

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

enp1s0 デバイスの横にある **unmanaged** 状態は、**NetworkManager** がこのデバイスを管理していないことを示します。

関連情報

デバイスを非管理対象と、対応する構文に設定するのに使用できる基準の一覧は、man ページの NetworkManager.conf(5) の 『Device List Format』 セクションを参照してください。

3.3.11.2. NetworkManager でデバイスの非管理として一時的に設定

インターフェイス名、MAC アドレス、デバイスタイプなど、複数の基準に基づいてデバイスを 管理対象外 として設定できます。この手順では、NetworkManager で、enp1s0 インターフェイスを unmanaged として一時的に設定する方法を説明します。

この方法は、たとえば、テスト目的で使用します。ネットワークデバイスを 管理対象外 として永続的に設定するには、「NetworkManager で、デバイスを Unmanaged (管理外) として永続的に設定」 を参照してください。

手順

1. オプション：デバイスの一覧を表示して、unmanaged に設定するデバイスを特定します。

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

2. enp1s0 デバイスを unmanaged 状態に設定します。

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

検証手順

- デバイスのリストを表示します。

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

enp1s0 デバイスの横にある unmanaged 状態は、NetworkManager がこのデバイスを管理していないことを示します。

関連情報

デバイスを非管理対象と、対応する構文に設定するのに使用できる基準の一覧は、man ページの NetworkManager.conf(5) の 『Device List Format』 セクションを参照してください。

3.4. GNOME GUI を使用した IP ネットワークの設定

Red Hat Enterprise Linux 7 では、NetworkManager には独自のグラフィカルユーザーインターフェイス(GUI)がありません。デスクトップ右上のネットワーク接続アイコンは GNOME Shell の一部として提供され、ネットワーク設定ツールは、有線、無線、vpn 接続をサポートする新しい GNOME control-center GUI の一部として提供されます。nm-connection-editor は、GUI 設定の主なツールです。control-center の機能に加えて、ボンド、チーム、ブリッジ接続の設定など、GNOME control-center によって提供されない機能も適用されます。本セクションでは、以下を使用してネットワークインターフェイスを設定できます。

- GNOME control-center アプリケーション
- GNOME nm-connection-editor アプリケーション

3.4.1. control-center GUI を使用したネットワーク接続

control-center アプリケーションの ネットワーク 設定ウィンドウにアクセスするには、次の 2 つの方法があります。

- Super キーを押してアクティビティーの概要に入り、Settings と 入力 して Enter を押します。次に、左側の Network タブを選択すると、Network 設定ツールが表示されます。「control-center を使用した新しい接続の設定」に進みます。
- 画面右上にある GNOME シェルのネットワーク接続アイコンをクリックして、メニューを開く。

図3.5 control-center アプリケーションを使用したネットワーク設定

[D]

GNOME シェルのネットワーク接続アイコンをクリックすると、以下が表示されます。

- 現在接続しているカテゴリー別のネットワーク一覧(有線 や Wi-Fiなど)。
- NetworkManager が検出した Available Networks の全一覧。
- 設定済みの仮想プライベートネットワーク (VPN) への接続オプション。

  および

- ネットワーク設定 メニューエントリーを 選択するオプション。

ネットワークに接続されていれば、その接続名の左側に 黒い点 が表示されます。

Network Settings をクリックすると、Network 設定ツールが表示されます。「control-center を使

用した新しい接続の設定」に進みます。

3.4.2. GUI を使用した新規接続の設定および既存接続の編集

システム管理者は、ネットワーク接続を設定できます。これにより、ユーザーがインターフェイスの設定を適用または変更できます。それには、次のいずれかの方法を使用できます。

- GNOME control-center アプリケーション
- GNOME nm-connection-editor アプリケーション

3.4.2.1. control-center を使用した新規接続の設定および既存接続の編集

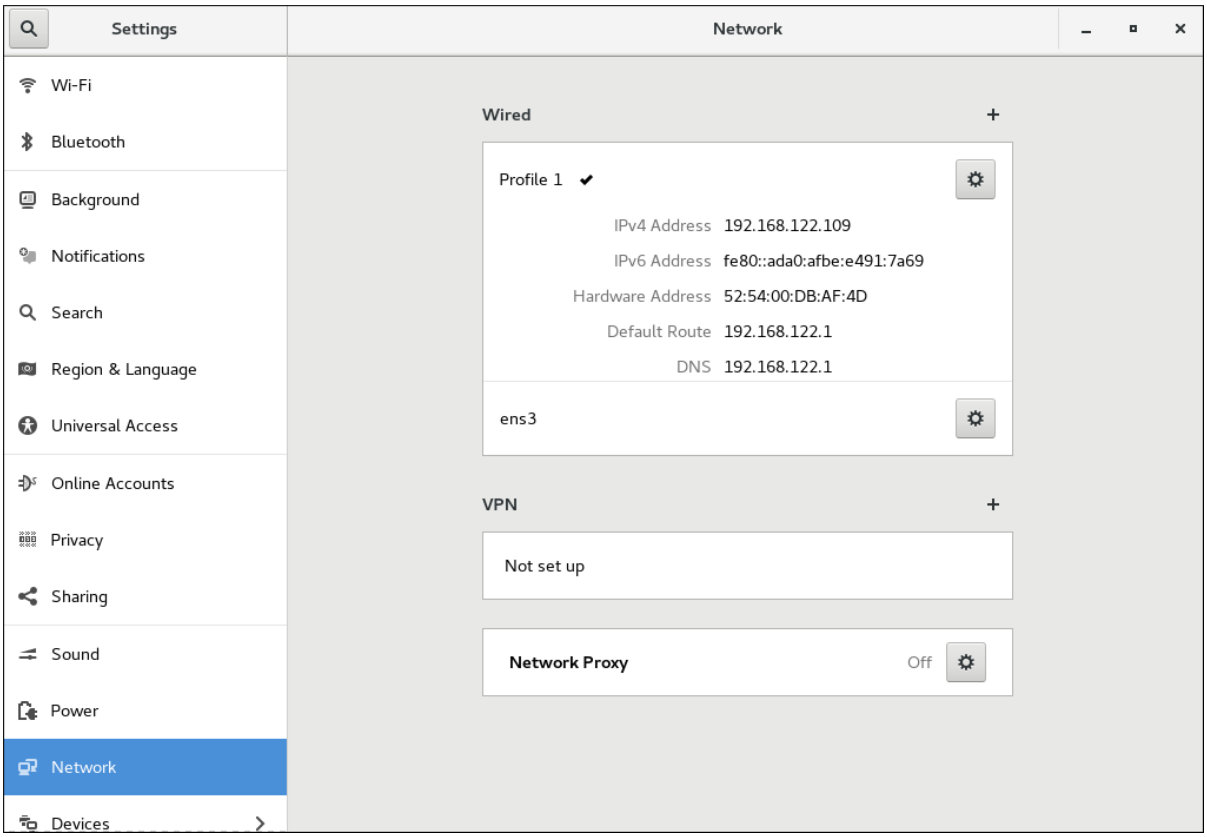
GNOME control-center アプリケーションを使用してネットワーク接続を作成および設定できます。

control-center を使用した新しい接続の設定

control-center アプリケーションを使用して、新しい有線、ワイヤレス、vpn 接続を設定するには、以下の手順を実行します。

1. Super キーを押してアクティビティーの概要に入り、Settings と 入力 して Enter を押します。次に、左側の Network タブを選択します。右側のメニューに Network 設定ツールが表示されます。

図3.6 ネットワーク設定ウィンドウの表示



2. 新しい接続を追加するには、プラスボタンをクリックします。

   設定するには、以下のコマンドを実行します。

   - 有線接続 を設定するには、有線 エントリーの横にある + （プラス）ボタンをクリックして、「GUI を使用した有線 (イーサーネット) 接続の設定」 に進みます。

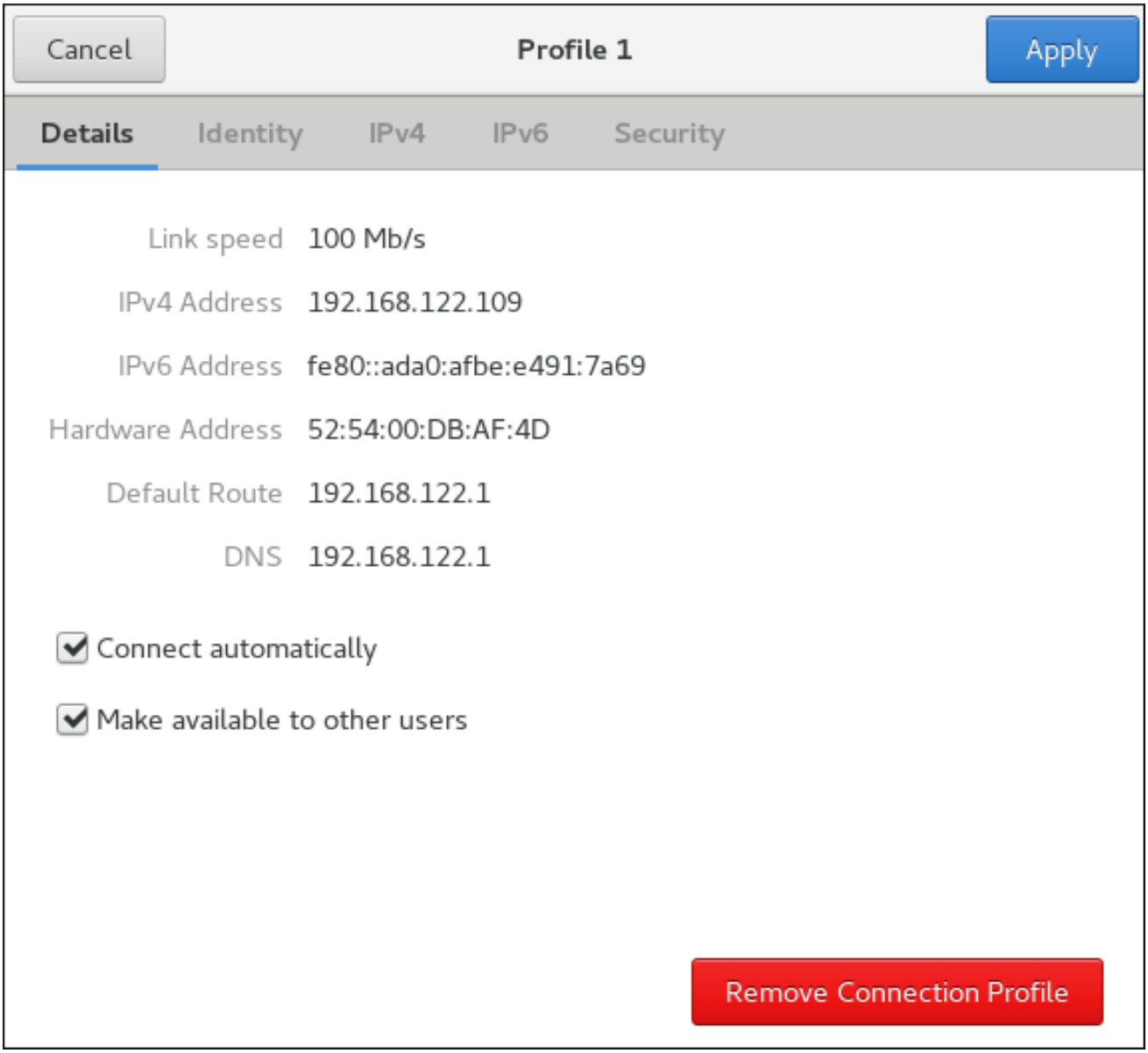
   - VPN 接続 の場合は、VPN エントリーの横にあるプラスボタンをクリックして、に進みます。 「control-center を使用した VPN 接続の確立」

   Wi-Fi 接続 の場合は、Settings メニューの Wi-fi エントリーをクリックして、に進みます。 「GUI を使用した Wi-Fi 接続の設定」

control-center を使用した既存の接続の編集

Network settings ウィンドウで既存の接続プロファイルの歯車アイコンをクリックすると、Details ウィンドウが開き、そこから IP アドレス、DNS、ルーティング設定などのほとんどのネットワーク設定タスクを実行できます。

図3.7 ネットワーク接続詳細ウィンドウを使用したネットワークの設定



[D]

追加または設定するすべての接続で、NetworkManager を選択して、利用可能な接続時にそのネットワークに自動的に接続できます。これを行うには、Connect automatically を選択し、NetworkManager が利用可能であることを検出すると、NetworkManager が接続に自動接続するようにします。NetworkManager が自動的に接続しないようにする場合は、チェックボックスの選択を解除します。この場合、接続するには手動でネットワーク接続のアイコンを選択する必要があります。

他のユーザーにも利用可能にするには、他のユーザーにも 利用可能にする のチェックボックスを

選択します。

接続の変更後に変更を適用するには、接続ウィンドウの右上隅にある Apply ボタンをクリックします。

Remove Connection Profile の赤いボックスをクリックすると、接続を削除できます。

3.4.2.2. nm-connection-editor を使用した新規接続の設定および既存接続の編集

nm-connection-editor GUI アプリケーションを使用すると、control-center が提供する以外の追加機能を使用して、任意の接続を設定できます。さらに、nm-connection-editor は、ボンディング、ブリッジ、VLAN、チーム接続の設定など、GNOME control-center で提供されない機能を適用します。

nm-connection-editor を使用した新しい接続の設定

nm-connection-editor を使用して新しい接続タイプを追加するには、以下を実行します。

手順

1. 端末に nm-connection-editor と入力します。

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

ネットワーク接続 ウィンドウが 表示されます。

2. + (プラス) ボタンをクリックして、接続タイプを選択します。

図3.8 nm-connection-editor を使用した接続タイプの追加

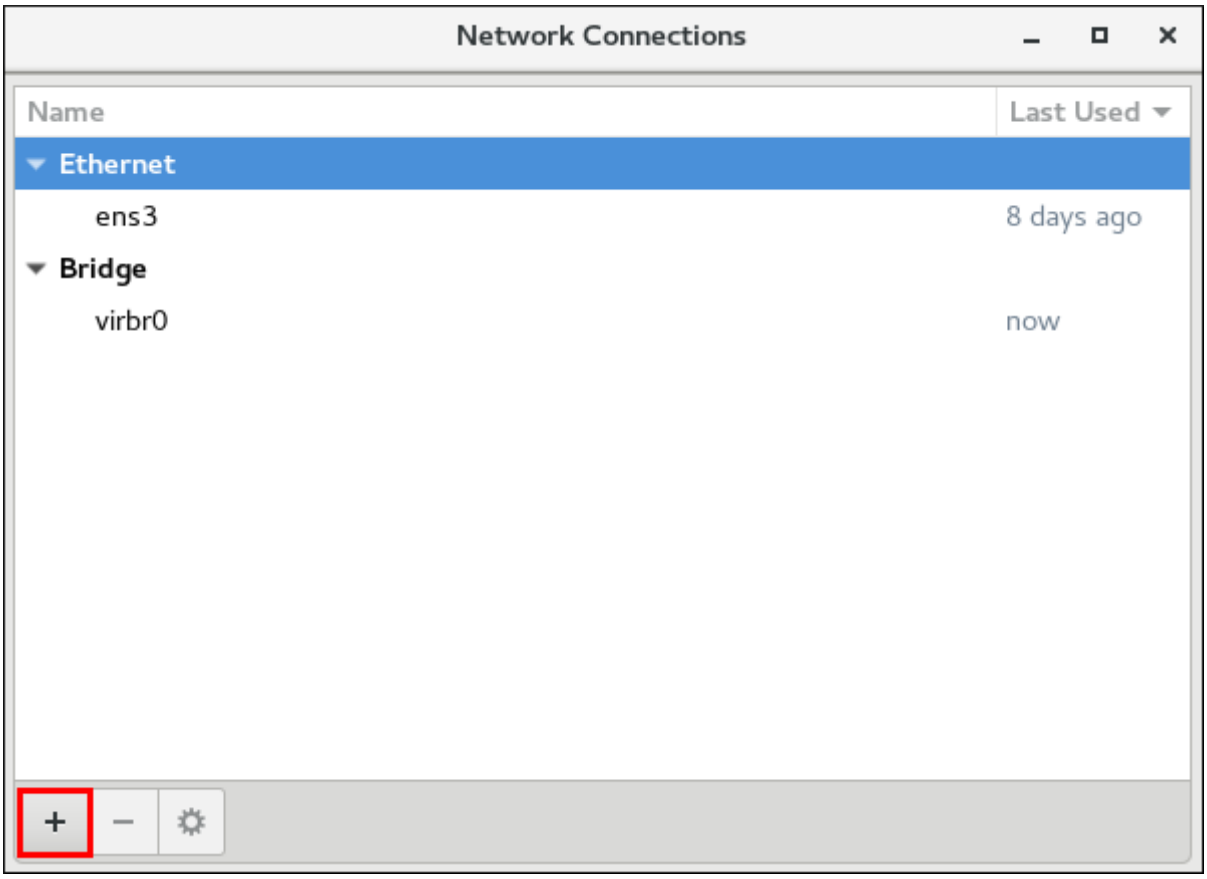


図3.9 nm-connection-editor を使用した接続タイプの選択

作成および設定するには、以下のコマンドを実行します。

- Bond 接続 の場合は、Bond エントリーをクリックして 「ボンド接続の確立」 に進みます。
- ブリッジ接続 の場合は、Bridge エントリーをクリックして 「GUI を使用したブリッジ接続の確立」 に進みます。
- VLAN 接続 の場合は、VLAN エントリーをクリックして 「VLAN 接続の確立」 に進みます。または、
- チーム接続 で、Team エントリーをクリックして 「GUI を使用したネットワークチームの作成」 に進みます。

nm-connection-editor を使用した既存接続の編集

既存の接続タイプについては、ネットワーク接続ダイアログの歯車アイコンをクリックします。「nm-connection-editor を使用した新しい接続の設定」 を参照してください。

3.4.3. nm-connection-editor を使用した一般的な設定オプション

nm-connection-editor ユーティリティーを使用する場合は、以下の手順に従って、ほとんどの接続タイプ（イーサネット、wifi、モバイルブロードバンド、DSL）に共通する設定オプションが 5 つあります。

手順

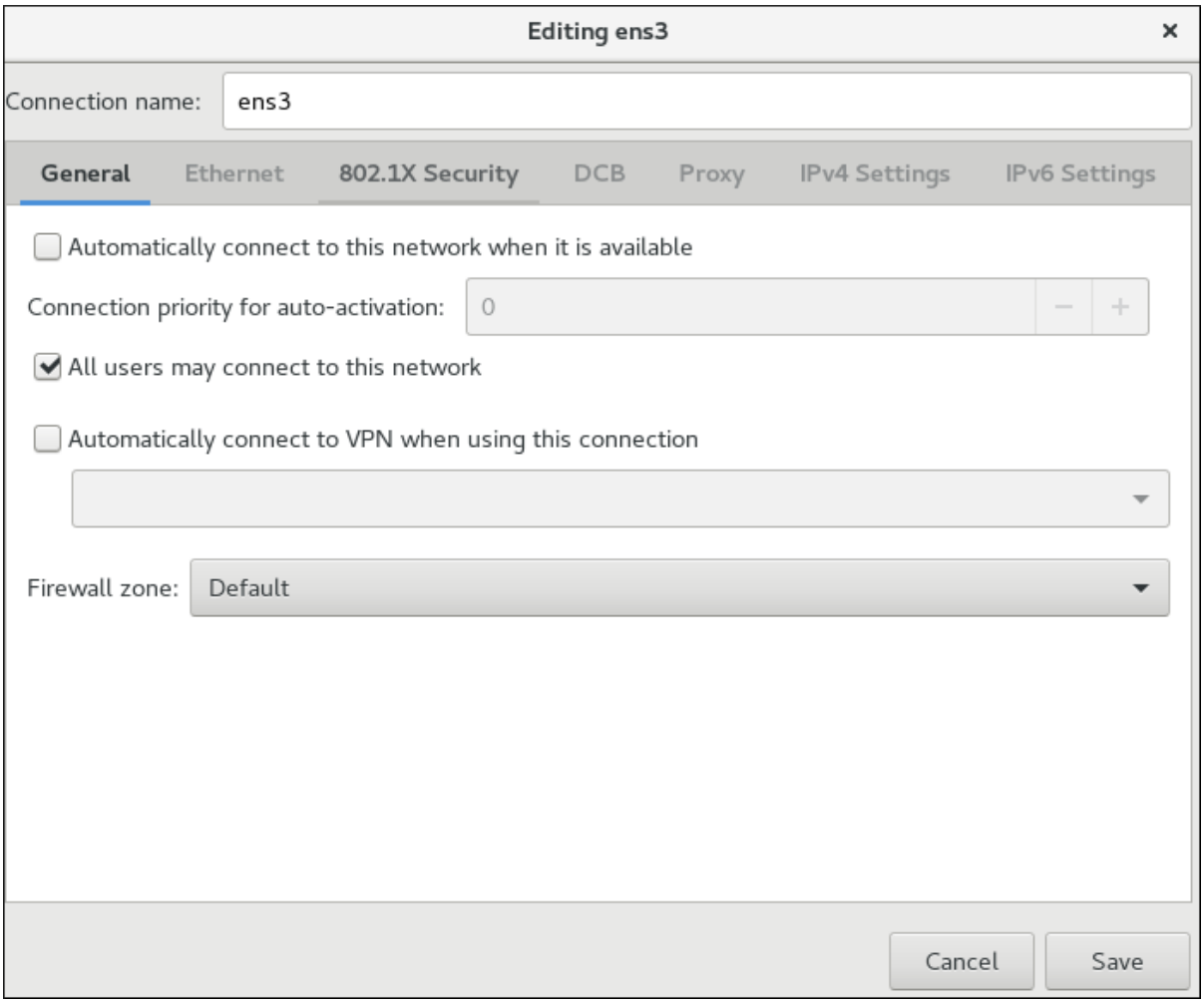
1. 端末に nm-connection-editor と入力します。

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

ネットワーク接続 ウィンドウが 表示されます。+ (プラス) ボタンをクリックして接続タイプを選択するか、歯車アイコンをクリックして、既存の接続を編集します。

2. Editing ダイアログで General タブを選択します。

図3.10 nm-connection-editor における設定オプション



- 接続名: ネットワーク接続のわかりやすい名前を入力します。この名前は、Network ウィンドウのメニューでこの接続を一覧表示するために使用されます。

- Connection priority for auto-activation - 接続が自動接続に設定されていると、番号がアクティブになります（デフォルトでは0 ）。数値が大きいほど優先度が高くなります。

- Automatically connect to this network when it is available - このボックスを選択すると、NetworkManager が利用可能なときにこの接続に自動接続します。詳細は、「control-center を使用した既存の接続の編集」を参照してください。

- All users can connect to this network - このボックスを選択すると、システム上のすべてのユーザーが 利用できる接続が作成されます。この設定を変更するには、root 権限が必要になる場合があります。詳細は、「GUI を使用したシステム全体およびプライベート接続プロファイルの管理」を参照してください。

- Automatically connect to VPN when using this connection - このボックスを選択すると、NetworkManager が利用可能なときに VPN 接続に自動接続します。ドロップダウンメニューから VPN を選択します。

- ファイアウォールゾーン - ドロップダウンメニューからファイアウォールゾーンを選択します。ファイアウォールゾーンに関する詳細情報は、『Red Hat Enterprise Linux 7 セキュリティーガイド』を参照してください。

注記

VPN 接続タイプでは、上記の設定オプションのうち 3 つだけが利用できます(接続名、すべてのユーザーはこのネットワークおよび ファイアウォールゾーンに接続 できます)。

3.4.4. GUI を使用してネットワークに自動的に接続

追加または設定するすべての接続で、利用可能なときに NetworkManager がそのネットワークに自動的に接続しようとするかどうかを選択できます。以下のいずれかの方法を使用できます。

- GNOME control-center アプリケーション

- GNOME nm-connection-editor アプリケーション

3.4.4.1. control-center を使用したネットワークの自動接続

control-center を使用して、自動的にネットワークに接続できます。

手順

1. Super キーを押してアクティビティーの概要に入り、Settings と 入力 して Enter を押します。次に、左側の Network タブを選択します。ネットワーク設定ツールが右側のメニューに表示されます。「control-center を使用した新しい接続の設定」を参照してください。

2. 右側のメニューから、ネットワークインターフェイスを選択します。

3. 右側のメニューにある接続プロファイルの歯車アイコンをクリックします。Network の詳細ウィンドウが表示されます。

4. Details メニューエントリーを選択します。「control-center を使用した既存の接続の編集」 を参照してください。

5. Connect automatically を選択し、NetworkManager が利用可能であることを検知すると、NetworkManager が接続に自動接続するようになります。NetworkManager が自動的に接続しないようにする場合は、チェックボックスの選択を解除します。この場合、接続するには手動でネットワーク接続のアイコンを選択する必要があります。

3.4.4.2. nm-connection-editor を使用したネットワークの自動接続

GNOME nm-connection-editor アプリケーションを使用して、ネットワークに自動的に接続することもできます。これを行うには、「nm-connection-editor を使用した一般的な設定オプション」 に記載の手順に従って、General タブの Automatically connect to this network when it is available チェックボックスにチェックを入れます。

3.4.5. GUI を使用したシステム全体およびプライベート接続プロファイルの管理

NetworkManager は、すべての *接続プロファイル* を保存します。プロファイルは、インターフェイスに適用できる名前の付いたコレクションです。NetworkManager は、システム全体の使用用の接続プロファイル(*システム接続*)とすべての *ユーザー接続* プロファイルを保存します。接続プロファイルへのアクセスは、NetworkManager によって保存されるパーミッションによって制御されます。接続設定の permissions プロパティーの詳細は、man ページの nm-settings (5) を参照してください。 以下のグラフィカルユーザーインターフェイスツールを使用して、接続プロファイルへのアクセスを制御できます。

- nm-connection-editor アプリケーション
- control-center アプリケーション

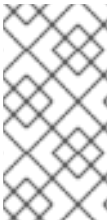
3.4.5.1. nm-connection-editor を使用した接続プロファイルの権限の管理

システムの全ユーザーが利用できる接続を作成するには、「nm-connection-editor を使用した一般的な設定オプション」 に記載される手順に従って、General タブの All users may connect to this network チェックボックスをオンにします。

3.4.5.2. control-center を使用した接続プロファイルの権限の管理

他のユーザーが利用できる接続を確立するには、「control-center を使用した既存の接続の編集」に記載されている手順に従って、GNOME control-center Network settings Details ウィンドウで Make available to other users チェックボックスを選択します。

逆に、他のユーザーにも利用可能にする チェックボックスの選択を解除して、システム全体ではなく、ユーザー固有のものにします。



注記

システムのポリシーによっては、接続をユーザー固有またはシステム全体に変更するために、システムで root 権限が必要となる場合があります。

NetworkManager のデフォルトポリシーでは、すべてのユーザーがシステム全体の接続を作成および変更できます。起動時に利用可能とするプロファイルはユーザーがログインするまで見えないため、これらをプライベートにすることはできません。たとえば、ユーザーが接続プロファイル user-em2 を作成し、自動的 に接続する チェックボックスを選択して いるが、他のユーザーにも利用可能に する選択を行わない 場合、この接続は起動時に利用できなくなります。

接続とネットワークを制限するには 2 つのオプションがあり、これらは個別または合わせて使用できます。

- 他のユーザーにも利用可能にする チェックボックスの選択を解除します。これにより、この変更を行ったユーザーのみが 接続を変更および使用可能に変更します。

- polkit フレームワークを使用して、ユーザーごとに一般的なネットワーク操作のパーミッションを制限します。

この 2 つのオプションを合わせると、ネットワークに関して詳細なセキュリティーと制御が可能になります。 polkit の詳細については、polkit (8) の man ページを参照してください。

VPN 接続は Wi-Fi やイーサネット接続よりもプライベートなものという前提なので、これは 常にユーザーごとのプライベートとして作成されることに留意してください。

3.4.6. GUI を使用した有線 (イーサーネット) 接続の設定

GUI を使用して有線接続を設定する方法は 2 つあります。

- control-center アプリケーション
- nm-connection-editor アプリケーション

3.4.6.1. control-center を使用した有線接続の設定

手順

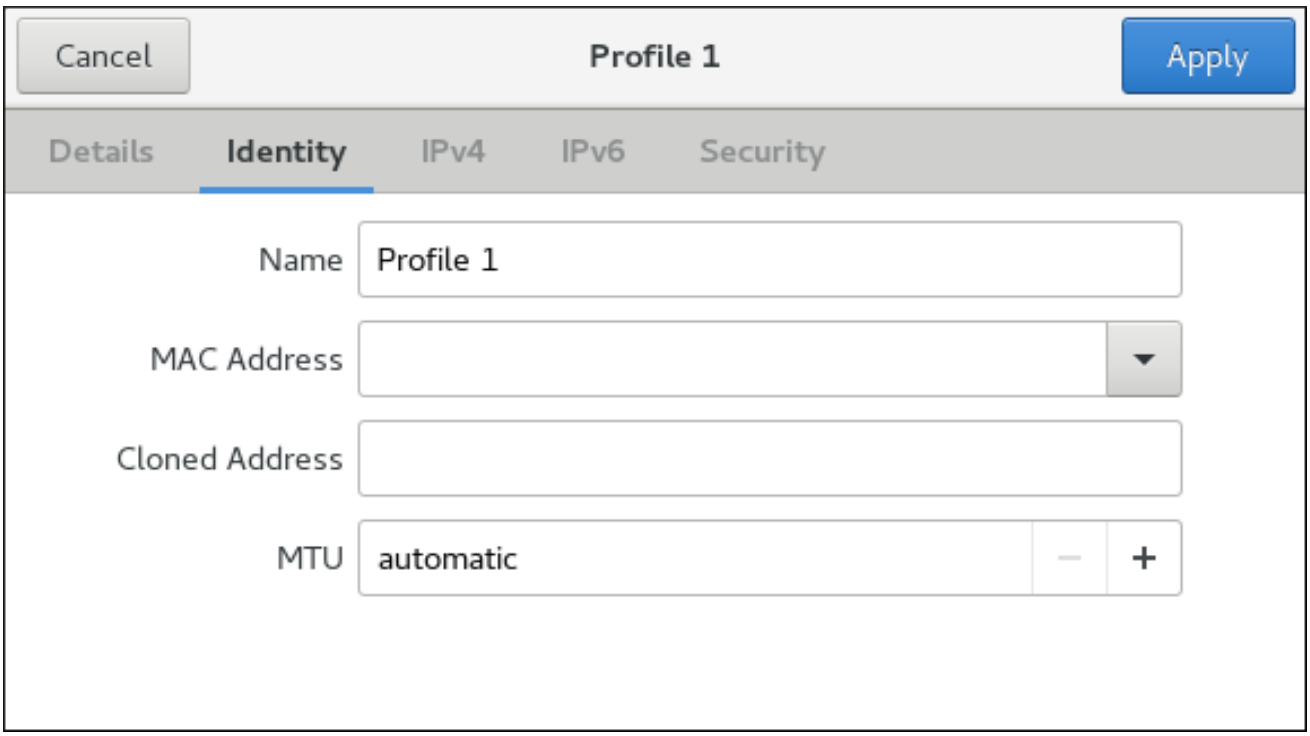
1. Super キーを押してアクティビティーの概要に入り、Settings と 入力 して Enter を押します。次に、左側の Network メニューエントリーを選択すると、Network 設定ツールが表示されます。「control-center を使用した新しい接続の設定」 を参照してください。
2. Wired ネットワークインターフェイスがまだ強調表示されていない場合は、選択します。

   システムは、デフォルトで、1 つの有線 *接続プロファイル* (Wired)を作成し、設定します。プロファイルは、インターフェイスに適用できる名前の付いたコレクションです。必要に応じて、1 つのインターフェイスに対して複数のプロファイルを作成して適用できます。デフォルトのプロファイルは削除できませんが、設定は変更できます。
3. 歯車アイコンをクリックして、デフォルトの Wired プロファイルを編集します。

基本設定オプション

Wired ダイアログでは、Identity メニューエントリーを選択すると、以下の設定が表示されます。

図3.11 有線接続に関する基本設定オプション



- 名前: ネットワーク接続のわかりやすい名前を入力します。この名前は、Network ウィンドウのメニューでこの接続を一覧表示するために使用されます。

- MAC アドレス - このプロファイルを適用する必要があるインターフェイスの MAC アドレスを選択します。

- Cloned Address - 必要な場合は、使用する別の MAC アドレスを入力します。

- MTU: 必要な場合は、使用する特定の *最大伝送単位* (MTU)を入力します。MTU 値は、リンク層が転送する最大パケットサイズをバイト単位で表したものです。この値のデフォルトは 1500 で、通常は指定したり変更したりする必要はありません。

有線設定の追加作成

編集ダイアログで、既存の接続をさらに設定できます。

設定するには、以下のコマンドを実行します。

-

接続の IPv4 を設定するには、IPv4 メニューエントリーをクリックして、に進みます。「IPv4 設定の設定」

または

- 接続の IPv6 を設定するには、IPv6 メニューエントリーをクリックして、「IPv6 セッティングの設定」 に進みます。

- Port-based Network Access Control (PNAC) にアクセスし、802.1X セキュリティー メニューエントリーをクリックして、「802.1X セキュリティーの設定」 に進みます。

新しい (または変更した) 有線接続の保存

有線接続の編集が終わったら、適用 ボタンをクリックしてカスタマイズした設定を保存します。編集中にプロファイルが使用されていた場合は、接続を再起動して、NetworkManager に変更を適用します。プロファイルがオフだった場合は、これをオンにするか、ネットワーク接続アイコンメニューで選択します。新規および変更後の接続を使用することに関する詳細情報は、「control-center GUI を使用したネットワーク接続」を参照してください。

有線接続の新規作成

新しい有線接続プロファイルを作成するには、+ (プラス) ボタンをクリックします。「control-center を使用した新しい接続の設定」を参照してください。

プラスボタンをクリックして新しい接続を追加する場合は、NetworkManager により、その接続用の新しい設定ファイルが作成され、既存の接続の編集に使用するのと同じダイアログが表示されます。「control-center を使用した既存の接続の編集」 を参照してください。これらのダイアログの違いは、既存の接続プロファイルに Details メニューエントリーがあることです。

3.4.6.2. nm-connection-editor を使用した有線接続の設定

nm-connection-editor GUI アプリケーションは、control-center GUI アプリケーションよりも多くの設定オプションを提供します。nm-connection-editor を使用して有線接続を設定するには、以下を行います。

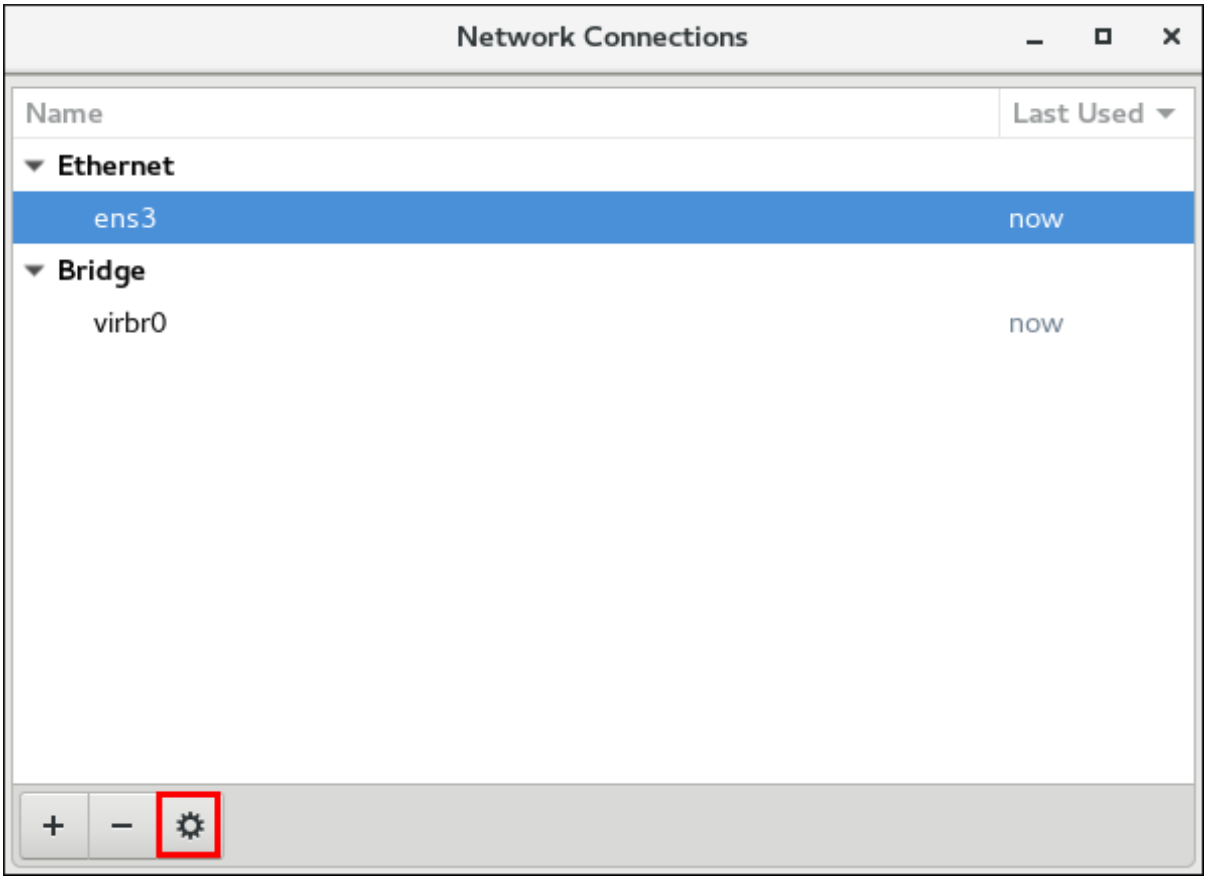
1. 端末に nm-connection-editor を入力します。

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

ネットワーク接続 ウィンドウが 表示されます。

2. 編集するイーサネット接続を選択して、歯車アイコンをクリックします。

図3.12 有線接続を編集します。



編集 ダイアログが表示されます。

- ネットワークに自動的に接続し、接続を制限するには、General タブをクリックします。「nm-connection-editor を使用した一般的な設定オプション」 を参照してください。

- ネットワーク設定を設定するには、Ethernet タブをクリックします。??? を参照してください。

- 有線接続に 802.1X セキュリティーを設定するには、802.1X セキュリティー タブをクリックします。??? を参照してください。

- IPV4 を設定するには、IPV4 Settings タブをクリックします。詳細は、「nm-

connection-editor を使用した IPV4 の方法の設定」 を参照してください。

- IPV6 設定を設定するには、IPV6 Settings タブをクリックします。「IPv6 セッティングの設定」 を参照してください。

3.4.7. GUI を使用した Wi-Fi 接続の設定

本セクションでは、NetworkManager を使用して、アクセスポイントへの Wi-Fi （ワイヤレス または 802.11*a/b/g/n*とも呼ばれます）接続を設定する方法を説明します。アクセスポイントは、ワイヤレスデバイスをネットワークに接続できるデバイスです。

(3G などの) モバイルブロードバンドの設定方法は、「GUI を使用したモバイルブロードバンド接続の設定」を参照してください。

利用可能なアクセスポイントにすばやく接続

手順

1. ネットワーク接続アイコンをクリックして、ネットワーク接続アイコンのメニューをアクティブにします。「control-center GUI を使用したネットワーク接続」を参照してください。
2. Wi-Fi ネットワークの一覧で、アクセスポイントの *Service Set Identifier* (SSID)を見つけます。
3. ネットワークの SSID をクリックします。鍵の記号は、アクセスポイントが認証を要求することを示します。アクセスポイントがセキュリティーで保護されている場合は、認証キーまたはパスワードの入力を求めるダイアログが表示されます。

   NetworkManager は、アクセスポイントで使用されるセキュリティーの種類を自動検出しようとします。複数の可能性がある場合、NetworkManager はセキュリティータイプを推測し、Wi-Fi セキュリティー ドロップダウンメニューで表示します。

   - WPA-PSK セキュリティー (パスフレーズ付きの WPA) の場合は、選択する必要がありません。

- WPA Enterprise (802.1X) の場合は、セキュリティーを自動検出できないため、セキュリティーを選択する必要があります。

  良くわからない場合は、順番に、それぞれのタイプに接続してみてください。

4. Password フィールドにキーまたはパスフレーズを入力します。40 ビットの WEP キーまたは 128 ビットの WPA キーは、必要な長さがないと無効になります。選択したセキュリティータイプに必要な長さのキーを入力するまで、Connect ボタンは非アクティブのままになります。ワイヤレスセキュリティーの詳細は、「802.1X セキュリティーの設定」 を参照してください。

NetworkManager が正常にアクセスポイントに接続すると、ネットワーク接続アイコンがワイヤレス接続のシグナル強度を示すグラフィカルインジケーターに変わります。

また、自動作成されたアクセスポイント接続の設定を、あたかも自分で追加したように編集することもできます。Network ウィンドウの Wi-Fi ページには History ボタンがあります。このボタンをクリックすると、接続を試みたすべての接続が一覧表示されます。「既存の Wi-Fi 接続の編集」 を参照

非表示 Wi-Fi ネットワークへの接続

すべてのアクセスポイントにはそれら自体の識別のために *サービスセット識別子* (SSID) があります。ただし、アクセスポイントは、その SSID をブロードキャストしないように設定できます。この場合、非 表示であり、NetworkManager の Available ネットワークのリストには表示されません。ただし、その SSID と認証方法と秘密情報が分かっていれば、SSID を非表示としているワイヤレスアクセスポイントに接続することは可能です。非表示のワイヤレスネットワークに接続するには、以下を行います。

手順

1. Super キーを押してアクティビティーの概要に入り、Settings と 入力 して Enter を押します。次に、左側の Wi-Fi メニューエントリーを選択します。

2. Connect to Hidden Network を選択します。2 つの選択肢があります。

- 以前非表示にしたネットワークに接続したことがある場合は、以下を行います。

1. ネットワークを選択するには、Connection ドロップダウンを使用します。

2. Connect をクリックします。

- そうでない場合は、以下の手順に従ってください。

1. Connection ドロップダウンを New のままにします。

2. 非表示のネットワークの SSID を入力します。

3. その Wi-Fi セキュリティー 方法を選択します。

4. 正しい認証の秘密を入力します。

5. Connect をクリックします。

ワイヤレスセキュリティー設定の詳細は、「802.1X セキュリティーの設定」 を参照してください。

新しい Wi-Fi 接続の設定

手順

1. Settings の Wi-Fi メニューエントリーを選択します。

2. 接続する Wi-Fi 接続名 (デフォルトでは SSID と同じ) をクリックします。

- SSID が範囲にない場合は、「非表示 Wi-Fi ネットワークへの接続」を参照してくだ

さい。

- SSID が範囲内にある場合は、右側のメニューで Wi-Fi 接続プロファイルをクリックします。鍵の記号は、鍵やパスワードが必要であることを示します。要求されたら、認証の詳細を入力してください。

既存の Wi-Fi 接続の編集

以前接続しようとした、または接続できた既存の接続を編集できます。

手順

1. Super キーを押してアクティビティーの概要に入り、Settings と 入力 して Enter を押します。

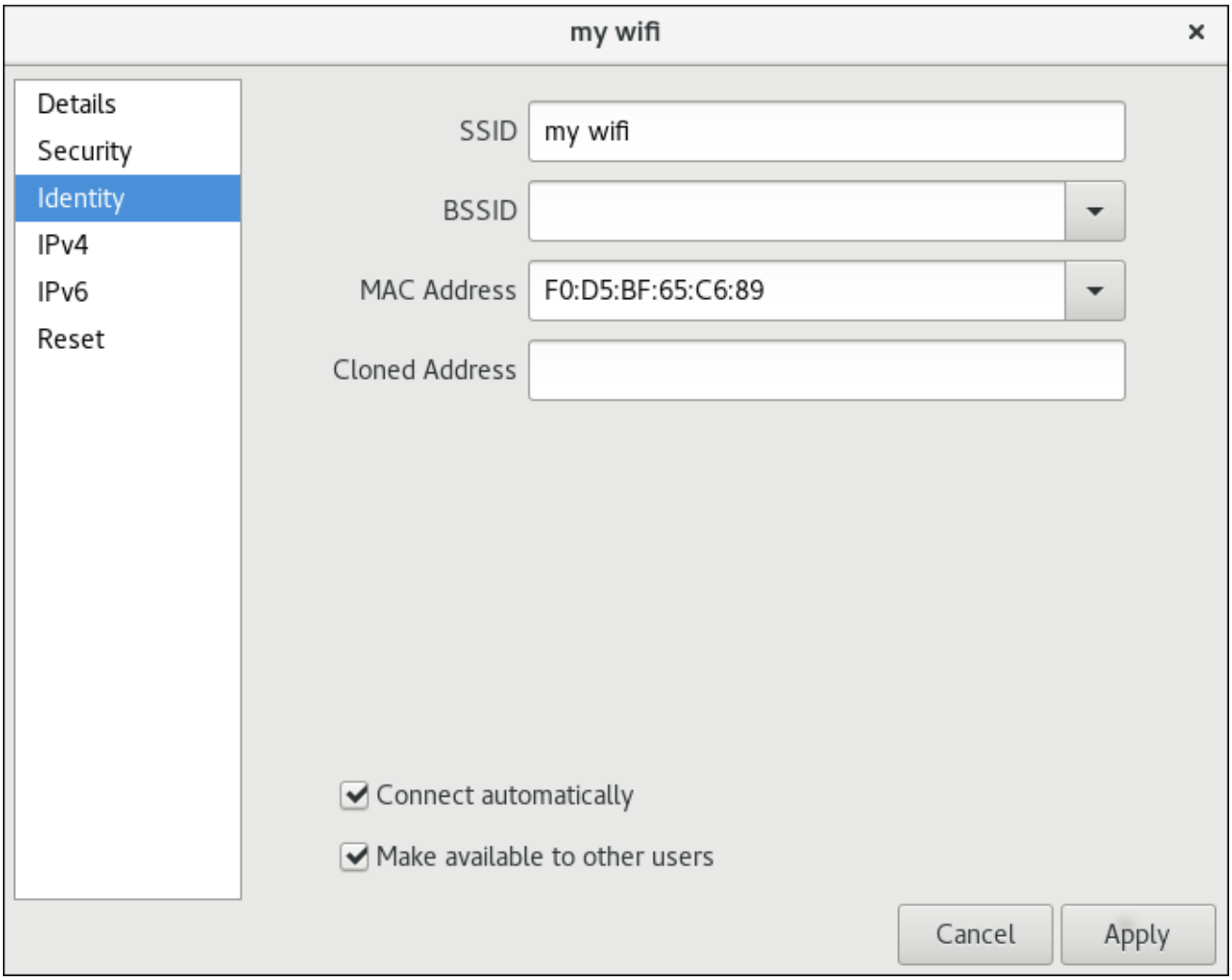
2. 左側のメニューエントリーから Wi-Fi を選択します。

3. 編集する Wi-Fi 接続名の右側にある歯車アイコンを選択すると、接続の編集ダイアログが表示されます。現在ネットワークが範囲に含まれていない場合は、履歴 をクリックして過去の接続を表示することに注意してください。Details ウィンドウには、接続の詳細が表示されます。

Wi-Fi 接続に関する基本設定オプション

Wi-Fi 接続の設定を編集するには、接続の編集ダイアログから Identity を選択します。以下の設定が可能です。

図3.13 Wi-Fi 接続に関する基本設定オプション



SSID

アクセスポイント (AP) の *サービスセット識別子* (SSID) です。

BSSID

BSSID(*Basic Service Set Identifier*)は、インフラストラクチャー モードで接続する特定のワイヤレスアクセスポイントの MAC *アドレス（ハードウェアアドレス* とも呼ばれます）です。このフィールドはデフォルトで空白になっており、BSSID を指定せずに SSID によってワイヤレスアクセスポイントに接続できます。BSSID を指定している場合は、システムによる特定のアクセスポイントのみへの関連付けが強制的に実行されます。

アドホックネットワークの場合、アドホックネットワークの作成時に、mac80211 サブシステムがランダムに BSSID を生成します。NetworkManagerでは表示されません。

MAC アドレス

MAC アドレスを選択します。これは、Wi-Fi が使用する *ハードウェアアドレス* とも呼ばれます。

単一システムには、1 つまたは複数のワイヤレスネットワークアダプターを接続することができます。したがって、MAC アドレス フィールドを使用すると、特定のワイヤレスアダプターを特定の接続（または接続）に関連付けることができます。

クローンしたアドレス

実際のハードウェアアドレスの代わりに使用する、クローンした MAC アドレスです。必要でない限り、空白のままにします。

以下の設定は、ほとんどの接続の種類に共通のものです。

- Connect automatically - このボックスを選択すると、NetworkManager が利用可能なときにこの接続に自動接続します。詳細は、「control-center を使用した既存の接続の編集」を参照してください。

- Make available to other users - このボックスを選択して、システム上のすべてのユーザーが利用できる接続を作成します。この設定を変更するには、root 権限が必要になる場合があります。詳細は、「GUI を使用したシステム全体およびプライベート接続プロファイルの管理」を参照してください。

Wi-Fi 接続の追加設定

編集ダイアログで、既存の接続をさらに設定できます。

設定するには、以下のコマンドを実行します。

- ワイヤレス接続の セキュリティー認証 では、セキュリティー をクリックし て、「802.1X セキュリティーの設定」 に進みます。

- 接続の IPv4 を設定するには、IPv4 をクリックして、に進みます。 「IPv4 設定の設定」

  または

- 接続の IPv6 を設定するには、IPv6 をクリックして、「IPv6 セッティングの設定」 に進

みます。

新しい (または変更した) 接続の保存

ワイヤレス接続の編集が終わったら、適用 ボタンをクリックして設定を保存します。設定が適切であれば、ネットワーク接続のアイコンメニューからこの変更した接続を選択することで接続できます。ネットワークの選択および接続については、「control-center GUI を使用したネットワーク接続」を参照してください。

3.4.8. GUI を使用した VPN 接続の設定

Libreswan が提供する IPsec は、VPN を作成するのに推奨される方法です。Libreswan は、VPN 用のオープンソースのユーザー空間 IPsec 実装です。コマンドラインを使用した IPsec VPN の設定は、『Red Hat Enterprise Linux 7 セキュリティーガイド を参照してください』。

3.4.8.1. control-center を使用した VPN 接続の確立

Libreswan が提供する IPsec は、Red Hat Enterprise Linux 7 で VPN を作成するのに推奨される方法です。詳細は、「GUI を使用した VPN 接続の設定」 を参照してください。

下記の GNOME グラフィカルユーザーインターフェイスツールには、NetworkManager-libreswan-gnome パッケージが必要です。パッケージをインストールするには、root で以下のコマンドを実行します。

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

Red Hat Enterprise Linux に新しいパッケージをインストールする方法の詳細は、『Red Hat Enterprise Linux システム管理者のガイド』を参照してください。

仮想プライベートネットワーク (VPN) を確立すると、使用中の LAN (ローカルエリアネットワーク) と別のリモートの LAN との間で通信ができるようになります。これは、インターネットなどの中間ネットワークにトンネルを設定して行います。設定している VPN トンネルは、通常、認証と暗号化を使用します。安全なトンネルを使用して VPN 接続を正常に確立した後は、ユーザーが送信するパケットに対して、VPN ルーターまたはゲートウェイが以下のアクションを実行します。

1. ルーティングおよび認証目的で *認証*ヘッダー を追加します。

2. パケットデータを暗号化します。

3. カプセル化セキュリティーペイロード (ESP) プロトコルに従ってデータをパケットに囲みます。ESP は暗号化解除および処理の指示を設定します。

受信側の VPN ルーターはヘッダー情報を開いてデータを暗号化解読し、それを目的地 (ネットワーク上のワークステーションまたは他のノード) に送信します。ネットワーク対ネットワークの接続を使用すると、ローカルネットワーク上の受信側ノードはすでに暗号化解読されいてすぐに処理ができる状態のパケットを受信します。このため、ネットワーク対ネットワークの VPN 接続での暗号化と暗号化解除のプロセスは、クライアントに透過的になっています。

VPN は認証と暗号化で複数のレイヤーを使用するため、複数のリモートノードを統合してひとつのイントラネットとして作動させる上で安全かつ効果的な手段となります。

新しい IPsec VPN 接続の追加

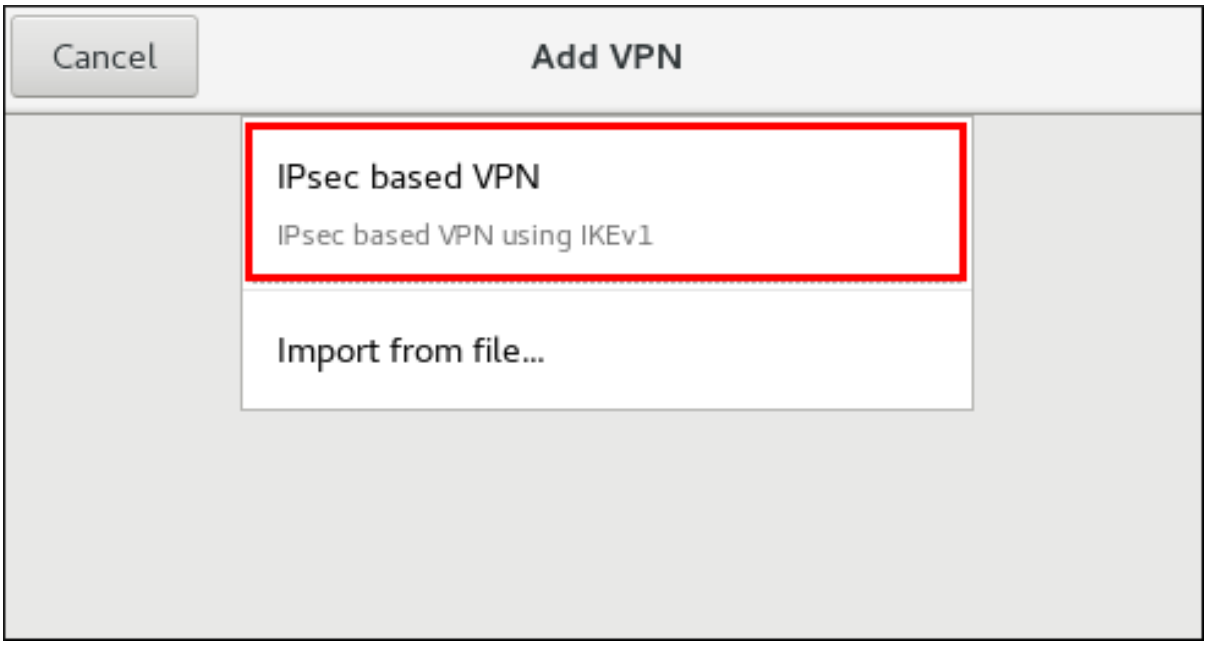
手順

1. Super キーを押してアクティビティーの概要に入り、Settings と 入力 して Enter を押します。次に Network メニューエントリーを選択すると、Network 設定ツールが表示されます。「control-center を使用した新しい接続の設定」 を参照してください。

2. VPN エントリーで + (プラス) ボタンをクリックします。

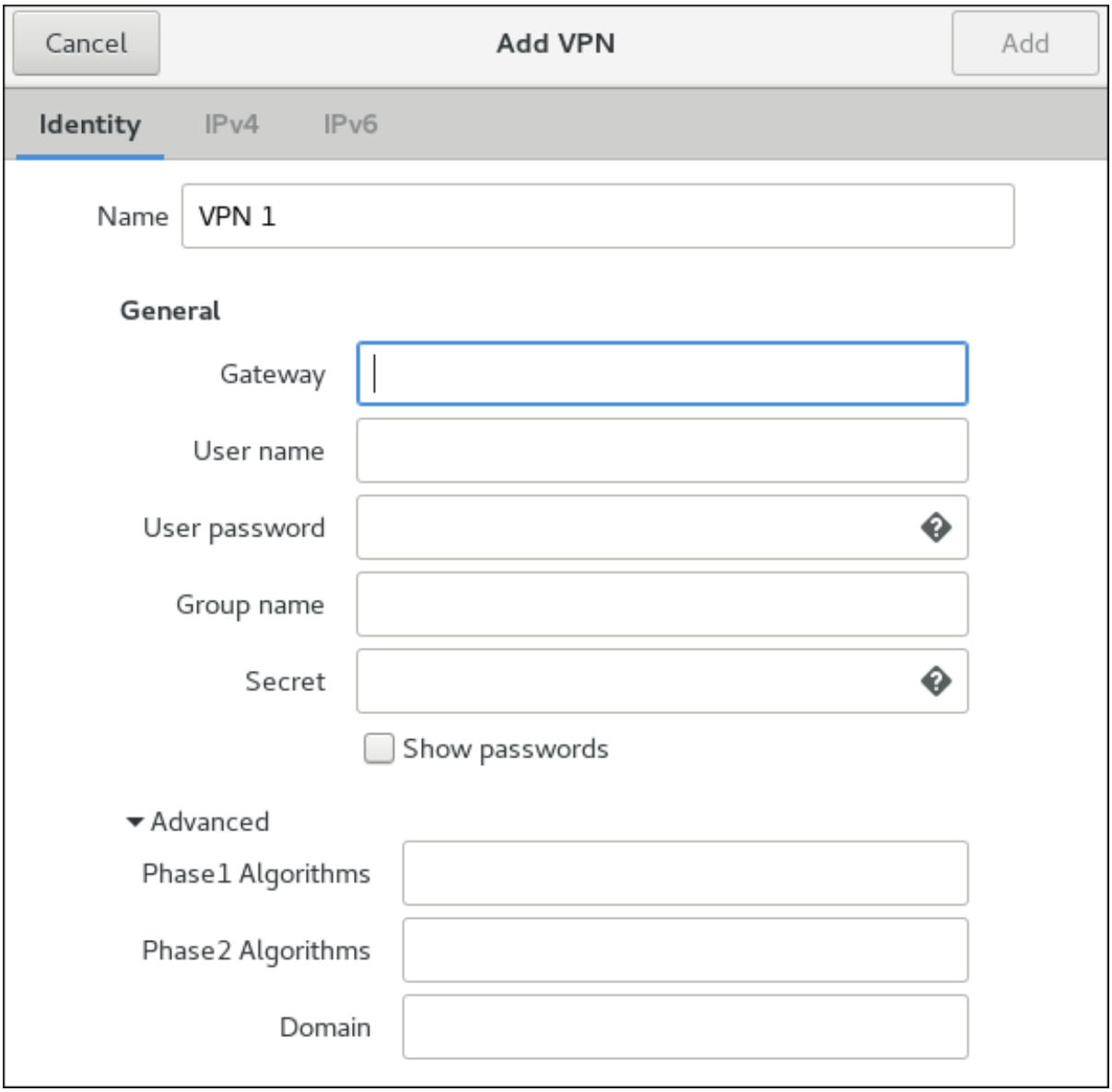
3. Add VPN ウィンドウが表示されます。　手動で設定する場合は、IPsec ベースの VPN を選択します。

図3.14 IPsec モードの VPN の設定



4. **Identity 設定フォームでは、General セクションおよび Advanced セクションにフィールドを指定できます。**

図3.15 全般オプションおよび高度セクション



- **General** セクションで、以下を指定できます。

ゲートウェイ

リモート VPN ゲートウェイの名前または IP アドレス。

ユーザー名

必要な場合は、認証のために VPN ユーザーに関連付けられたパスワードを入力します。

ユーザーパスワード

必要に応じて、認証のために VPN ユーザーの ID に関連付けられているパスワードを入力します。

グループ名

リモートゲートウェイで設定された VPN グループ名です。空欄の場合は、デフォルトのアグレッシブモードではなく IKEv1 メインモードが使われます。

シークレット

ユーザー認証の前に暗号化を初期化するのに使われる、事前共有キーです。必要に応じて、グループ名に関連付けられたパスワードを入力します。

- Advanced セクションでは、以下の設定が可能です。

フェーズ 1 アルゴリズム

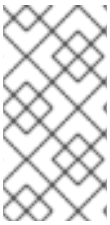
必要な場合は、暗号化チャンネルの認証および設定で使用するアルゴリズムを入力します。

フェーズ 2 アルゴリズム

必要な場合は、IPsec ネゴシエーションに使用するアルゴリズムを入力します。

ドメイン

必要な場合は、ドメイン名を入力します。



注記

NetworkManager を使用せずに IPsec VPN を設定する場合は、「GUI を使用した VPN 接続の設定」 を参照してください。

既存の VPN 接続を編集する

手順

1. Super キーを押してアクティビティーの概要に入り、Settings と 入力 して Enter を押します。次に Network メニューエントリーを選択すると、Network 設定ツールが表示されます。「control-center を使用した新しい接続の設定」 を参照してください。

2. 編集する VPN 接続を選択し、歯車アイコンをクリックして、General セクションおよび Advanced セクションを編集します。「control-center を使用した VPN 接続の確立」 を参照してください。

新規 (または修正した) 接続を保存して他の設定を行う

新しい VPN 接続の編集が終わったら、保存 ボタンをクリックしてカスタマイズした設定を保存します。編集中にプロファイルが使用されていた場合には、接続の電源を入れ直して、NetworkManager が変更を適用するようにします。プロファイルがオフだった場合は、これをオンにするか、ネットワーク接続アイコンメニューで選択します。新規および変更後の接続を使用することに関する詳細情報は、「control-center GUI を使用したネットワーク接続」を参照してください。

既存の接続をさらに設定するには、Network ウィンドウでその接続を選択し、Configure をクリックして Editing ダイアログに戻ります。

そして、以下のいずれかの設定をします。

- IPv4 の設定は、IPv4 のセッティング タブをクリックして、「IPv4 設定の設定」 に進みます。

3.4.8.2. nm-connection-editor を使用した VPN 接続の設定

nm-connection-editor を使用して VPN 接続を追加し、設定することもできます。以下の手順を行います。

手順

1. 端末に nm-connection-editor と入力します。ネットワーク接続 ウィンドウが表示されます。詳細は 「nm-connection-editor を使用した一般的な設定オプション」 を参照してください。
2. + (プラス) ボタンをクリックします。Choose a Connection Type メニューが開きます。
3. VPN メニューエントリーから、IPsec ベースの VPN オプションを選択します。
4. Create をクリックして Editing ダイアログを開き、「新しい IPsec VPN 接続の追加」 に進んで General セクションおよび Advanced セクションを編集します。

3.4.9. GUI を使用したモバイルブロードバンド接続の設定

NetworkManager のモバイルブロードバンド接続機能を使用して、以下の *2G* および *3G* サービスに接続できます。

- 2G: *GPRS* (*General Packet Radio Service*)、*EDGE* (*Enhanced Data Rates for GSM Evolution*)、または CDMA (Code Division Multiple Access)。

- 3G: *UMTS* (*Universal Mobile Telecommunications System*)、*HSPA* (*High Speed Packet Access*)、または EVDO (EVolution Data-Only)。

接続を作成するには、使用中のシステムがすでに発見して認識しているモバイルブロードバンドのデバイス (モデム) をコンピューターが備えている必要があります。そのようなデバイスはコンピューターに内蔵されている場合 (多くのノートブックやネットブック) と、外付けまたは内蔵のハードウェアとして提供されている場合があります。たとえば、PC カードや USB モデム、ドングル、モデムとして機能する携帯電話などです。

3.4.9.1. nm-connection-editor を使用したモバイルブロードバンド接続の設定

GNOME nm-connection-editor を使用して、モバイルブロードバンド接続を設定できます。

新しいモバイルブロードバンド接続の追加

手順

1. 端末に nm-connection-editor と入力します。ネットワーク接続 ウィンドウが表示されます。詳細は 「nm-connection-editor を使用した一般的な設定オプション」 を参照してください。
2. + (プラス) ボタンをクリックします。Choose a Connection Type メニューが開きます。
3. Mobile Broadband メニューエントリーを選択します。
4. Create をクリックして、Set up a Mobile Broadband Connection アシスタントを開きます。
5. Create a connection for this mobile broadband device で、接続で使用する 2G または 3G 対応デバイスを選択します。ドロップダウンメニューが使用できない場合は、システムがモバイルブロードバンドの機能を持ったデバイスを検出できなかったことを示します。この場合、Cancel をクリックし、モバイルブロードバンド対応のデバイスが接続されており、コンピューターによって認識されていることを確認してから、この手順を再試行してください。続行 ボタンをクリックします。
6. 一覧からサービスプロバイダーが置かれている国を選択し、Continue ボタンをクリックします。

7. リストからプロバイダーを選択するか、手動で入力します。続行 ボタンをクリックします。

8. ドロップダウンメニューから請求プランを選んで、*Access Point Name* (APN) が正しいか確認します。続行 ボタンをクリックします。

9. 設定を確認して、適用 ボタンをクリックします。

10. 「モバイルブロードバンドタブの設定」を参照して、モバイルブロードバンド特有の設定を編集します。

既存のモバイルブロードバンド接続を編集する

手順

1. 端末に nm-connection-editor と入力します。ネットワーク接続 ウィンドウが 表示されます。

2. Mobile Broadband タブを選択します。

3. 編集する接続を選択し、歯車アイコンをクリックします。詳細は、「nm-connection-editor を使用した一般的な設定オプション」を参照してください。

4. 「モバイルブロードバンドタブの設定」を参照して、モバイルブロードバンド特有の設定を編集します。

モバイルブロードバンドタブの設定

アシスタントを使用して新しいモバイルブロードバンド接続をすでに追加している場合（手順は「新しいモバイルブロードバンド接続の追加」 を参照）、ホームネットワークが利用できない場合はローミングを無効にするか、ネットワーク ID を割り当て、接続を使用するときに特定の技術(3G や 2G など)を優先するように NetworkManager に指示できます。

Number

GSM ベースのモバイルブロードバンドネットワークでの PPP 接続を確立するためにダイアルする番号です。このフィールドは、ブロードバンドデバイスの初期インストールの際に自動設定さ

れている場合があります。通常、このフィールドは空白のままにして、代わりに APN を入力できます。

Username

ネットワークでの認証に使用するユーザー名を記入します。一部のプロバイダーは、ユーザー名を提供しないことや、ネットワーク接続の時点でユーザー名を受け付けたりすることがあります。

Password

ネットワークで認証に使用するパスワードを記入します。一部のプロバイダーはパスワードを提供しなかったり、またはすべてのパスワードを受け付けたりします。

APN

GSM ベースのネットワークとの接続を確立するために使用する *Access Point Name* (APN) を記入します。これは以下の項目を決定するので、正しい APN を記入することが重要になります。

- ネットワーク使用量についてユーザーが請求される方法
- ユーザーがインターネット、イントラネット、サブネットワークにアクセスできるかどうか。

ネットワーク ID

ネットワーク ID を入力すると、NetworkManager は、デバイスが特定のネットワークにのみ登録するように強制します。これにより、ローミングを直接に制御できない時に接続がローミングしないようにします。

Type

any: デフォルト値の Any では、モデムは最速のネットワークを選択します。

3G (UMTS/HSPA): 接続に 3G ネットワーク技術のみを使用するように強制します。

2G (GPRS/EDGE): 接続に 2G ネットワーク技術のみを使用するように強制します。

prefer 3G (UMTS/HSPA) - HSPA や UMTS などの 3G 技術を使用して最初に接続を試み、障害時にのみ GPRS または EDGE にフォールバックします。

prefer 2G (GPRS/EDGE) - まず、GPRS や EDGE などの 2G 技術を使用して接続しようとし、障害時にのみ HSPA または UMTS にフォールバックします。

ホームネットワークが使用できない場合にローミングを許可

ホームネットワークからローミングに移行せずに NetworkManager が接続を終了するようにするには、このボックスをオフにします。ボックスがオンになっていると、NetworkManager は、ホームネットワークからローミングに移行して適切な接続を維持しようとします。その逆も同様です。

PIN

デバイスの *SIM* (*Subscriber Identity Module*)が PIN (Person*Identification Number*)でロックされている場合は、*PIN* を入力して、NetworkManager がデバイスのロックを解除できるようにします。デバイスを使用するために PIN が必要な場合は、NetworkManager は SIM のロックを解除する必要があります。

CDMA および EVDO のオプションは少なくなります。APN、Network ID、または Type オプションは含まれません。

新規 (または修正した) 接続を保存して他の設定を行う

モバイルブロードバンド接続の編集が終わったら、適用 ボタンをクリックしてカスタマイズした設定を保存します。編集中にプロファイルが使用されていた場合には、接続の電源を入れ直して、NetworkManager が変更を適用するようにします。プロファイルがオフだった場合は、これをオンにするか、ネットワーク接続アイコンメニューで選択します。新規および変更後の接続を使用することに関する詳細情報は、「control-center GUI を使用したネットワーク接続」を参照してください。

既存の接続をさらに設定するには、ネットワーク 接続 ウィンドウでその接続を選択し、編集 をクリックして 編集 ダイアログに戻ります。

そして、以下のいずれかの設定をします。

- **ポイントツーポイント を設定する には、PPP 設定 タブをクリックして、「PPP (ポイントツーポイント) セッティングの設定」 に進みます。**
-

IPv4 の設定は、IPv4 のセッティング タブをクリックして 「IPv4 設定の設定」 に進みます。または、

- IPv6 の設定は、IPv6 のセッティング タブをクリックして、「IPv6 セッティングの設定」 に進みます。

3.4.10. GUI を使用して DSL 接続の設定

このセクションでは、個人ユーザーや SOHO インストールでよくある DSL モデムルーターの外部の組み合わせではなく、ホスト内に DSL カードが組み込まれているインストールについて説明します。

3.4.10.1. nm-connection-editor を使用した DSL 接続の設定

GNOME nm-connection-editor を使用して DSL 接続を設定できます。

新しい DSL 接続の追加

手順

1. 端末に nm-connection-editor と入力します。ネットワーク接続 ウィンドウが表示されます。詳細は 「nm-connection-editor を使用した一般的な設定オプション」 を参照してください。
2. + (プラス) ボタンをクリックします。
3. 接続の種類の選択 リストが表示されます。
4. DSL を選択し、Create ボタンをクリックします。
5. DSL Connection *1*の編集 ウィンドウが表示されます。

既存の DSL 接続を編集する

手順

1.

端末に nm-connection-editor と入力します。ネットワーク接続 ウィンドウが 表示されます。

2. 編集する接続を選択し、歯車アイコンをクリックします。詳細は、「nm-connection-editor を使用した一般的な設定オプション」を参照してください。

DSL タブの設定

Username

サービスプロバイダー認証で使用するユーザー名を入力します。

サービス

サービスプロバイダーからの指示がない限り、空白のままにします。

Password

サービスプロバイダーから提供されたパスワードを入力します。

新規 (または修正した) 接続を保存して他の設定を行う

DSL 接続の編集が終わったら、適用 ボタンをクリックしてカスタマイズした設定を保存します。編集中にプロファイルが使用されていた場合には、接続の電源を入れ直して、NetworkManager が変更を適用するようにします。プロファイルがオフだった場合は、これをオンにするか、ネットワーク接続アイコンメニューで選択します。新規および変更後の接続を使用することに関する詳細情報は、「control-center GUI を使用したネットワーク接続」を参照してください。

既存の接続をさらに設定するには、ネットワーク 接続 ウィンドウでその接続を選択し、編集 をクリックして 編集 ダイアログに戻ります。

設定するには、以下のコマンドを実行します。

- MAC アドレスおよび MTU を設定するには、有線 タブをクリックして、「基本設定オプション 」 に進みます。
- ポイントツーポイント を設定する には、PPP 設定 タブをクリックして、「PPP (ポイントツーポイント) セッティングの設定」 に進みます。

- IPv4 の設定は、IPv4 のセッティング タブをクリックして、「IPv4 設定の設定」 に進みます。

3.5. IFCFG ファイルを使用した IP ネットワークの設定

システム管理者は、ifcfg ファイルを編集して、ネットワークインターフェイスを手動で設定できます。

インターフェイス設定 (ifcfg) ファイルは、個々のネットワークデバイスのソフトウェアインターフェイスを制御します。これは、システムの起動時に、このファイルを使用して、どのインターフェイスを起動するかと、どのように設定するかを決定します。通常、これらのファイルの名前は ifcfg-*name* です。接尾辞 *name* は、設定ファイルが制御するデバイスの名前を指します。通常、ifcfg ファイルの接尾辞は、設定ファイル自体の DEVICE ディレクティブが指定する文字列と同じです。

ifcfg ファイルを使用した静的ネットワーク設定によるインタフェースの設定

たとえば、ifcfg ファイルを使用して静的ネットワークでインターフェイスを設定するには、enp1s0 という名前のインターフェイスに対して、/etc/sysconfig/network-scripts/ ディレクトリーに ifcfg-enp1s0 という名前のファイルを作成します。これには以下が含まれます。

- IPv4 設定の場合

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

- IPv6 設定の場合

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

これは ipcalc によって自動的に計算されるため、ネットワークまたはブロードキャストアドレスを指定する必要はありません。

IPv6 の ifcfg 設定オプションの詳細は、『nm-settings-ifcfg-rh(5)』 の man ページを参照してください。

重要

Red Hat Enterprise Linux 7 では、*11章ネットワークデバイス命名における一貫性*で説明されるように、ネットワークインターフェイスの命名規則が変更されています。HWADDR ディレクティブを使用してハードウェアまたは MAC アドレスを指定すると、デバイスの命名手順に影響が及ぶ可能性があります。

ifcfg ファイルを使用した動的ネットワーク設定によるインタフェースの設定

ifcfg ファイルを使用して動的ネットワーク設定で *em1* という名前のインターフェイスを設定するには、次のコマンドを実行します。

1. /etc/sysconfig/network-scripts/ ディレクトリーに、以下のような ifcfg-em1 という名前のファイルを作成します。

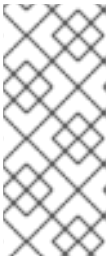
```
DEVICE=em1
BOOTPROTO=dhcp
ONBOOT=yes
```

2. DHCP サーバーに別のホスト名を送信するようにインターフェイスを設定するには、ifcfg ファイルに以下の行を追加します。

```
DHCP_HOSTNAME=hostname
```

DHCP サーバーに別の完全修飾ドメイン名(FQDN)を送信するようにインターフェイスを設定するには、ifcfg ファイルに以下の行を追加します。

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



注記

指定した ifcfg ファイルでは、DHCP_HOSTNAME または DHCP_FQDN のいずれかのディレクティブ 1 つのみを使用してください。DHCP_HOSTNAME と DHCP_FQDN の両方が指定されている場合は、後者のみが使用されます。

3. **特定の DNS サーバーを使用するようにインターフェイスを設定するには、ifcfg ファイルに以下の行を追加します。**

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

***ip-address* は、DNS サーバーのアドレスです。これにより、ネットワークサービスが指定された DNS サーバーで /etc/resolv.conf を更新します。DNS サーバーアドレスは 1 つだけ必要です。もう 1 つのアドレスは任意です。**

4. **ifcfg ファイルで静的ルートを設定するには、「ifcfg ファイルでの静的ルートの設定」 を参照してください。**

デフォルトでは、インターフェイス設定ファイルで *BOOTPROTO* を dhcp に設定してアドレスを自動的に取得するようにプロファイルが設定されている場合、NetworkManager は DHCP クライアントである dhclient を呼び出します。DHCP が必要な場合は、インターフェイス上のすべてのインターネットプロトコル(IPv4 および IPv6)に対して dhclient のインスタンスが開始されます。NetworkManager が実行していない場合や、インターフェイスを管理していない場合は、必要に応じて従来のネットワークサービスが dhclient のインスタンスを呼び出します。動的な IP アドレスの詳細は、「静的 IP アドレス指定と動的 IP アドレス指定の比較」を参照してください。

5. **設定を適用するには、以下を行います。**

   a. **更新した接続ファイルを再読み込みします。**

```
# nmcli connection reload
```

   b. **接続を再度有効にします。**

```
# nmcli connection up connection_name
```

3.5.1. ifcfg ファイルを使用したシステム全体およびプライベート接続プロファイルの管理

パーミッションは、ifcfg ファイルの USERS ディレクティブに対応します。USERS ディレクティブが存在しない場合、ネットワークプロファイルはすべてのユーザーが利用できます。たとえば、ifcfg ファイルの以下のコマンドは、リストされているユーザーだけが接続を利用できるようにします。

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

また、USERCTL ディレクティブを設定して、デバイスを管理できます。

- yes を設定すると、root 以外のユーザーはこのデバイスを制御できます。
- を設定し ない と、root 以外のユーザーがこのデバイスを制御することはでき ません。

3.6. IP コマンドを使用した IP ネットワークの設定

システム管理者は、ip コマンドを使用してネットワークインターフェイスを設定できますが、再起動後も変更は維持されません。システムを再起動すると、変更が失われます。

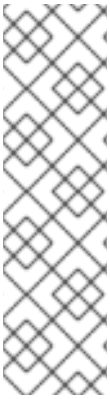
ip ユーティリティーのコマンドは、アップストリームのパッケージ名の後に iproute2 と呼ばれることがありますが、man ip (8) ページに記載されています。Red Hat Enterprise Linux 7 におけるパッケージ名は、iproute となります。必要に応じて、以下のようにバージョン番号を確認して、ip ユーティリティーがインストールされていることを確認できます。

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

ip コマンドを使用すると、NetworkManager と並行してアドレスとルートを追加および削除することができます。これにより、アドレスとルートが維持され、nmcli、nmtui、control-center、D-Bus API で認識されます。

インターフェイスを停止するには、以下のコマンドを実行します。

```
ip link set ifname down
```



注記

ip link set *ifname* コマンドは、ネットワークインターフェイスを IFF_UP 状態に設定し、カーネルのスコープから有効にします。これは、initscripts の ifup *ifname* コマンドまたは NetworkManager のデバイスのアクティブ化状態とは異なります。実際、NetworkManager は、インターフェイスが現在切断されている場合でも常にインターフェイスを設定します。nmcli ツールを使用してデバイスを切断しても、IFF_UP フラグは削除されません。このようにして、NetworkManager はキャリアの状態に関する通知を受け取ります。

ip ユーティリティーは、ifconfig ユーティリティーに代わるものであることに注意してください。これは、(ifconfigが提供する) net-tools パッケージが InfiniBand アドレスに対応していないためです。

利用可能な OBJECT の詳細は、ip help コマンドを使用します。たとえば、ip link help および ip addr help などです。

注記

コマンド ラインで指定された IP コマンドは、システム再起動後に維持されません。永続性が必要な場合は、設定ファイル(ifcfg ファイル)を利用するか、スクリプトにコマンドを追加します。

コマンドラインと設定ファイルの使用例は、nmtui および nmcli の例の後に含まれていますが、NetworkManager へのグラフィカルユーザーインターフェイス(control-center、および nm-connection-editor)のいずれかの使用を説明する前。

ip ユーティリティーを使用すると、以下の形式のインターフェイスに IP アドレスを割り当てることができます。

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

ip コマンドを使って静的アドレスを割り当てる

IP アドレスをインターフェイスに割り当てるには、以下を実行します。

```
~]# 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
```

その他の例およびコマンドオプションは、man ページの ip-address (8) を参照してください。

ip コマンドを使って複数のアドレスを設定する

ip ユーティリティーは、同じインターフェイスへの複数のアドレスの割り当てをサポートしているため、複数のアドレスを同じインターフェイスにバインドする alias インターフェイスメソッドを使用する必要がなくなりました。アドレスを割り当てる ip コマンドは、複数のアドレスを割り当てるために複数回繰り返すことができます。以下に例を示します。

```
~]# 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 ユーティリティーのコマンドの詳細は、man ページの ip (8) を参照してください。

注記

コマンド ラインで指定された IP コマンドは、システム再起動後に維持され ません。

3.7. カーネルコマンドラインから IP ネットワークの設定

インターフェイスから iSCSI ターゲットの root ファイルシステムに接続すると、インストール済みシステムでネットワーク設定が設定されません。この問題を解決するために、以下を行います。

1. dracut ユーティリティーをインストールします。dracut の使用に関する詳細は、『Red Hat Enterprise Linux System Administrator's Guide』を参照してください。

2. カーネルコマンドラインの ip オプションを使用して、設定を行います。

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

- DHCP: DHCP 設定
- dhpc6: DHCP IPv6 設定
- auto6 - 自動 IPv6 設定
- で、any: カーネルで利用可能なプロトコル（デフォルト）
- none、off - 自動設定なし、静的ネットワーク設定

以下に例を示します。

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

3. ネームサーバーの設定を定義します。

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

dracut ユーティリティーはネットワーク接続を設定し、/etc/sysconfig/network-scripts/ ファイルにコピーできる新しい ifcfg ファイルを生成します。

3.8. IGMP で IP マルチキャストの有効化

IGMP (Internet Group Management Protocol) を使用すると、管理者は、ネットワーク間、ホスト間、およびルーター間のマルチキャストトラフィックへのルーティングおよびサブスクリプションを管理できるようになります。Red Hat Enterprise Linux のカーネルは IGMPv3 に対応しています。

マルチキャスト情報を表示するには、以下のように ip maddr show サブコマンドを使用します。

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

または、ip link show コマンド出力で MULTICAST 文字列を見つけます。以下に例を示します。

```
~]$ 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
```

デバイスでマルチキャストを無効にし、*br0* デバイスで無効になっていることを確認するには、以下を行います。

```
~]# 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
```

MULTICAST 文字列がない場合は、マルチキャストが無効になっていることを示します。

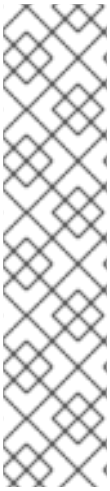
***br0* デバイスでマルチキャストを有効にし、それが有効になっていることを確認するには、以下を行います。**

```
~]# 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
```

詳細は、ip Command Cheat Sheet for Red Hat Enterprise Linux の記事および ip (8) man ページを参照してください。

マルチキャスト用にサブスクライブしている IGMP と IP アドレスの現行バージョンを確認するには、/proc/net/igmp ファイルを参照してください。

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



注記

デフォルトでは、IGMP は firewalld で有効になっていません。ゾーンで IGMP を有効にするには、以下を行います。

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

詳細は、Red Hat Enterprise Linux セキュリティーガイドのファイアウォールの使用の章を参照してください。

3.9. 関連情報

インストールされているドキュメント

- 『ip(8)』 man ページ： ip ユーティリティーのコマンド構文を説明しています。
- 『nmcli(1)』 の man ページ： NetworkManager のコマンドラインツールを説明しています。
- 『nmcli-examples(5)』 man ページ： nmcli コマンドの例を説明します。
- 『nm-settings(5)』 man ページ： NetworkManager のプロパティーとその設定を説明しています。
- 『nm-settings-ifcfg-rh(5)』 man ページ： ifcfg-rh 設定プラグインを説明しています。

オンラインドキュメント

『Red Hat Enterprise Linux 7 セキュリティーガイド』

IPsec ベースの VPN とその設定について説明します。DNSSEC を使用した認証済み DNS クエリーの使用を説明します。

『RFC 1518』: Classless Inter-Domain Routing (CIDR)

可変長サブネットを含む CIDR アドレス割り当ておよび集約戦略を説明しています。

『RFC 1918』: Address Allocation for Private Internets

は、プライベート使用用に予約されている IPv4 アドレスの範囲を説明します。

『RFC 3330』: Special-Use IPv4 Addresses

Internet Assigned Numbers Authority (IANA)によって割り当てられたグローバルおよび他の特殊な IPv4 アドレスブロックを説明します。

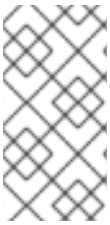
第4章 静的ルートおよびデフォルトゲートウェイの設定

本章は、静的ルートおよびデフォルトゲートウェイの設定を説明します。

4.1. ルーティングおよびゲートウェイの概要

ルーティングは、別のシステムへのネットワークパスをシステムが見つけられるようにするメカニズムです。ルーティングは、しばしば、ルーティング専用のネットワーク上で、デバイスにより処理されます (ただし、デバイスはルーティングを行うように設定できます)。したがって、Red Hat Enterprise Linux サーバーまたはクライアントで静的ルートを設定する必要がない場合もしばしばあります。例外は、暗号化された VPN トンネルを通過する必要があるトラフィックや、コストやセキュリティー上の理由から、特定のルートを通過する必要があるトラフィックが含まれます。ホストのルーティングテーブルには、ネットワークに直接接続しているルートが自動的に追加されます。このルートは、ネットワークインターフェイスが 「起動」 していると調べられます。リモートネットワークまたはホストに到達するには、システムには、トラフィックが送られるゲートウェイのアドレスが指定されている必要があります。

ホストのインターフェイスが DHCP によって設定されている場合、通常はアップストリームネットワークまたはインターネットにつながるゲートウェイのアドレスが割り当てられます。このゲートウェイは、通常はデフォルトのゲートウェイと呼ばれます。これは、システムがこれ以上のルールを認識していない場合 (ルーティングテーブルに存在しない場合)、使用するゲートウェイとなるためです。ネットワーク管理者は、ネットワーク内の最初または最後のホスト IP アドレスをゲートウェイアドレスとして使用します（例：192.168.10 .1 または 192.168.10.254 ）。ネットワーク自体を表すアドレス（この例では 192 . 168.10.0）と、サブネットのブロードキャストアドレス（この例では 192. 168.10.255）と混同しないようにしてください。デフォルトゲートウェイは、従来のネットワークルーターです。デフォルトゲートウェイは、ローカルネットワーク宛ではなく、ルーティングテーブルで優先ルートが指定されていないすべてのトラフィックに適用されます。



注記

専門知識を深めるには、Red Hat システム管理 I (RH124) トレーニングコースの受講を推奨します。

4.2. NMCLI を使った静的ルートの設定

nmcli ツールを使用して静的ルートを設定するには、以下のいずれかを使用します。

- nmcli コマンドライン
- nmcli インタラクティブ接続エディター

例4.1 nmcli を使った静的ルートの設定

コマンドラインを使用して、既存のイーサネット接続に静的ルートを設定するには、以下のコマンドを実行します。

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

これにより、192 .168.22. 0/24 サブネットへのトラフィックが 10.10.10.1 のゲートウェイに転送されます。

例4.2 nmcli をエディター使用した静的ルートの設定

インタラクティブエディターを使用したイーサネット接続に静的ルートを設定するには、以下のコマンドを実行します。

```
~]$ 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. GUI を使用した静的ルートの設定

静的ルートを設定するには、設定する接続の IPv4 または IPv6 設定ウィンドウを開きます。詳細は「control-center GUI を使用したネットワーク接続」を参照してください。

ルート

Address - リモートネットワーク、サブネット、またはホストの IP アドレスを入力します。

ネットマスク - 上記で入力した IP アドレスのネットマスクまたは接頭辞長。

Gateway - 上記で入力したリモートネットワーク、サブネット、またはホストにつながるゲートウェイの IP アドレス。

メトリック: このルートに渡す優先値であるネットワークコスト。数値が低い方が優先されます。

自動

Automatic が ON の場合、RA または DHCP からのルートが使用されますが、追加の静的ルートを追加することもできます。OFF の場合は、定義した静的ルートのみが使用されます。

Use this connection only for resources on its network (この接続はネットワーク上のリソースのためだけに使用)

このチェックボックスを選択すると、この接続はデフォルトルートになりません。よくある例としては、本社への接続が VPN トンネルや専用線で、インターネット向けトラフィックにこの接続を使用しない場合が挙げられます。このオプションを選択すると、この接続で自動的に学習したルートを使用することが明確なトラフィックか、手動で入力したトラフィックのみがこの接続を経由します。

4.4. IP コマンドでの静的ルートの設定

システム管理者は、ip route コマンドを使用して静的ルートを設定できます。

IP ルーティングテーブルを表示するには、ip route コマンドを使用します。以下に例を示します。

```
~]$ 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 コマンドは、以下の形式を取ります。オプションおよび形式の詳細は、

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

の ip-route (8) man ページを参照してください。

静的ルートをホストアドレス、つまり 1 つの IP アドレスに追加するには、以下のコマンドを実行します。

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

ここでの *192. 0.2.1* は、ドット付き 10 進表記のホストの IP アドレスで、10.0.0.1 はネクストホップアドレスで、*interface* はネクストホップにつながる終了インターフェイスです。

ネットワークに静的ルートを追加するには（つまり IP アドレスの範囲を表す IP アドレス）、以下を実行します。

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

192.0.2.0 はドット付き 10 進表記の宛先ネットワークの IP アドレスで、*/24* はネットワーク接頭辞です。ネットワーク接頭辞は、サブネットマスク内の有効なビット数です。ネットワークアドレスにスラッシュ、ネットワーク接頭辞長を続けるこの形式は、*classless inter-domain routing* (CIDR) 表記と呼ばれることもあります。

割り当てた静的ルートを削除するには、以下のコマンドを実行します。

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

ip route を使用してルーティングテーブルに加えた変更は、システムを再起動しても維持されません。静的ルートを永続的に設定するには、そのインターフェイスの /etc/sysconfig/network-scripts/ ディレクトリーに route-interface ファイルを作成して設定します。たとえば、*enp1s0* インターフェイスの静的ルートは、/etc/sysconfig/network-scripts/route-*enp1s0* ファイルに保存されます。route-interface ファイルに加えた変更は、ネットワークサービスまたはインターフェイスを再起動するまで反映されません。route-*interface* ファイルには、2 つの形式があります。

- ip コマンド引数については、「IP コマンド引数形式を使用した静的ルート」 を参照してください。

および

- network/netmask ディレクティブは、「Network または Netmask のディレクティブ形式を使用した静的ルート」を参照してください。

ip route コマンドの詳細は、ip-route (8) man ページを参照してください。

4.5. IFCFG ファイルでの静的ルートの設定

コマンドプロンプトで ip コマンドを使用して設定した静的ルートは、システムがシャットダウンまたは再起動されると失われます。システムの再起動後に静的ルートを永続化するように設定するには、/etc/sysconfig/network-scripts/ ディレクトリーのインターフェイスごとの設定ファイルに配置する必要があります。ファイル名は、route-*interface* の形式にする必要があります。設定ファイルで使用するコマンドの種類は 2 つあります。

IP コマンド引数形式を使用した静的ルート

インターフェイスごとの設定ファイル（例： /etc/sysconfig/network-scripts/route-enp1s0 ）で必要な場合は、最初の行でデフォルトゲートウェイへのルートを定義します。これは、ゲートウェイが DHCP 経由で設定されておらず、/etc/sysconfig/network ファイルでグローバルに設定されていない場合のみ必要です。

```
default via 192.168.1.1 dev interface
```

ここでの *192.168.1.1* は、デフォルトゲートウェイの IP アドレスです。*interface* は、デフォルトゲートウェイに接続されている、または到達可能なインターフェイスになります。dev オプションは省略できます。これはオプションです。この設定は、/etc/sysconfig/network ファイルの設定よりも優先されることに注意してください。

リモートネットワークへのルートが必要な場合は、静的ルートは以下のように指定できます。各行は、個別のルートとして解析されます。

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

ここでの *10.10.10.0/24* は、リモートもしくは宛先ネットワークのネットワークアドレスおよび接頭辞長です。アドレス *192.168.1.1* は、リモートネットワークにつながる IP アドレスです。*ネクストホップアドレス* の方が好ましいですが、出口インターフェイスのアドレスでも機能します。「ネクストホッ

プ」とは、ゲートウェイやルーターなどリンクのリモート側を意味します。dev オプションを使用して、終了 *インターフェイス* を指定できますが、必須ではありません。必要に応じて静的ルートを追加します。

以下は、ip コマンド引数形式を使用した route-*interface* ファイルの例です。デフォルトゲートウェイは 192.168.0.1 です。 enp1s0 リースされた行または WAN 接続は 192.168.0.10 で利用できます。2 つの静的ルートは、10.10.10.0/ 24 ネットワークおよび 172.16.1.10/32 ホストに到達するためのものです。

```
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
```

上記の例では、ローカルの 192.168.0.0/24 ネットワークに送信されるパケットは、そのネットワークに接続されているインターフェイスに送信されます。10.10.10.0/ 24 ネットワークおよび 172.16.1.10/32 ホストに送信されるパケットは 192. 168.0.10 に転送されます。既知でないリモートネットワークに向かうパケットはデフォルトゲートウェイを使用するので、デフォルトルートが適切でない場合は、静的ルートはリモートネットワークもしくはホスト用のみに設定すべきです。ここでのリモートとは、システムに直接繋がれていないネットワークやホストを指します。

IPv6 設定の場合、ip ルート形式の route6-*interface* ファイルの例：

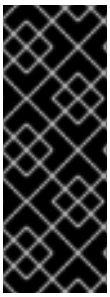
```
2001:db8:1::/48 via 2001:db8::1 metric 2048
2001:db8:2::/48
```

出口インターフェイスの指定は、オプションです。特定のインターフェイスからトラフィックを強制的に締め出したい場合は、これが便利です。たとえば、VPN の場合、リモートネットワークへのトラフィックが通過するように強制できます。 tun0 インターフェイスが宛先ネットワークとは別のサブネットにある場合でも、インターフェイス。

ip route 形式を使用して、送信元アドレスを指定できます。以下に例を示します。

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

複数のルーティングテーブルを指定する既存のポリシーベースのルーティング設定を定義するには、「ポリシールーティングについて」を参照してください。



重要

デフォルトゲートウェイがすでに DHCP によって割り当てられており、同じメトリックを持つ同じゲートウェイが設定ファイルで指定されている場合、起動時またはインターフェイスの起動時にエラーが発生します。"RTNETLINK answers: File exists" というエラーメッセージが表示される可能性があります。このエラーは無視できます。

Network または Netmask のディレクティブ形式を使用した静的ルート

route-*interface* ファイルに network/netmask ディレクティブ形式を使用することもできます。以下は、ネットワーク/ネットマスク形式のテンプレートで、後に説明が続きます。

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

- ADDRESS0=*10.10.10.* 0 は、到達するリモートネットワークまたはホストのネットワークアドレスです。
- NETMASK0=*255 .255.255* .0 は、ADDRESS0= 10.10.10.0 で定義されているネットワークアドレスのネットマスクです。
- GATEWAY0=*192 .168.1.1*は、ADDRESS0= 10.10.10.0 に到達するために使用できるデフォルトゲートウェイまたは IP アドレスです。

以下は、network/netmask ディレクティブ形式を使用した route-*interface* ファイルの例です。デフォルトゲートウェイは 192.168.0.1 ですが、リースされた行または WAN 接続は 192.168.0 .10 で利用できます。2 つの静的ルートは、10.10.10.0/24 および 172.16.1.0/24 ネットワークに到達するためのものです。

```
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
```

後に続く静的ルートは、順番に番号付けされる必要があり、いずれの値もスキップしてはいけません。たとえば、ADDRESS0、ADDRESS1、ADDRESS2 などです。

デフォルトでは、パケットをあるインターフェイスから別のインターフェイスに転送することや、

同じインターフェイスから外部に送信することは、セキュリティー上の理由から無効になっています。

これにより、外部トラフィック用のルーターとしてシステムが動作しているのを防ぐことができます。接続の共有や VPN サーバーの設定など、外部トラフィックをルーティングするのにシステムがルーティングする必要がある場合は、IP 転送を有効にする必要があります。詳細は『Red Hat Enterprise Linux 7 Security Guide』を参照してください。

4.5.1. ポリシールーティングについて

Policy-routing はソースルーティングとも呼ばれ、より柔軟なルーティング設定のためのメカニズムです。ルーティングの決定は、通常、パッケージの宛先 IP アドレスに基づいて行われます。ポリシールーティング により、ソース IP アドレス、ソースポート、プロトコルタイプなどの他のルーティングプロパティーに基づいてルートをより柔軟に選択できます。ルーティングテーブルは、ネットワークに関するルート情報を保存します。これらは、/etc/iproute2/rt_tables ファイルに設定できる数値または名前で識別されます。デフォルトのテーブルは 254 で識別されます。policy-routing を使用すると、ルールも必要になります。ルールは、ルーティングテーブルを選択するために使用され、パケットの特定のプロパティーに基づいています。

initscript の場合、ルーティングテーブルは、テーブルの引数から設定できるルートのプロパティーになります。ip route 形式は、複数のルーティングテーブルを指定する既存のポリシーベースのルーティング設定を定義するために使用できます。

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

initscripts でルーティングルールを指定するには、IPv4 の場合は /etc/sysconfig/network-scripts/rule-*enp1s0* ファイル、または IPv6 の場合は /etc/sysconfig/network-scripts/rule6-*enp1s0* ファイルに編集します。

NetworkManager はポリシールーティングをサポートしますが、ルールはまだサポートされていません。ルールは、カスタムスクリプトを実行して設定する必要があります。手動の静的なルート 1 つに対して、ルーティングテーブルを 1 つ選択できます。

- IPv4の場合は ipv4.route-table

  および

- IPv6 の場合は ipv6.route-table

特定のテーブルへのルートを設定することで、DHCP、autoconf6、DHCP6 からのすべてのルートがそのテーブルに配置されます。また、アドレスをすでに設定しているサブネットのルートはすべて、対応するルーティングテーブルに追加します。たとえば、*192.168.1.10/24* アドレスを設定する場合は、ipv4.route-table に *192.168.1.0/24* サブネットを追加します。

policy-routing ルールの詳細は、man ページの ip-rule (8) を参照してください。ルーティングテーブルは、man ページの ip-route (8) を参照してください。

4.6. デフォルトゲートウェイの設定

デフォルトゲートウェイは、最初に /etc/sysconfig/network ファイルを解析し、「up」 にあるインターフェイスのネットワークインターフェイス ifcfg ファイルを解析するネットワークスクリプトによって決定されます。ifcfg ファイルは数値順に解析され、最後に読み取られる GATEWAY ディレクティブはルーティングテーブルのデフォルトルートを作成するために使用されます。

デフォルトルートは、GATEWAY ディレクティブで、グローバルまたはインターフェイス固有の設定ファイルのいずれかで指定できます。ただし、Red Hat Enterprise Linux では、グローバルな /etc/sysconfig/network ファイルの使用は非推奨となり、ゲートウェイの指定はインターフェイスごとの設定ファイルでのみ行う必要があります。

NetworkManager がモバイルホストが管理している動的ネットワーク環境では、ゲートウェイ情報はインターフェイス固有である可能性が高く、DHCP によって割り当てられるのが最適です。NetworkManager のゲートウェイへのアクセスに使用する終了インターフェイスの選択に影響を与える必要がある特別なケースでは、デフォルトゲートウェイに送られないインターフェイスに対して ifcfg ファイルで DEFROUTE=no コマンドを利用します。

第5章 ネットワーク接続設定の設定

本章では、ネットワーク接続設定のさまざまな設定について説明し、NetworkManager を使用して設定する方法を説明します。

5.1. 802.3 リンクセッティングの設定

以下の設定パラメーターを修正して、イーサネット接続の 802.3 リンク設定を設定できます。

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

802.3 リンクセッティングを、以下の 3 つの主要モードに設定できます。

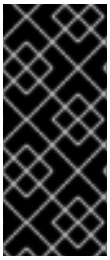
- リンクネゴシエーションを無視する
- オートネゴシエーションを強制的にアクティブ化する
- リンク設定 speed および duplex を手動で設定する

リンクネゴシエーションを無視する

この場合、NetworkManager はイーサネット接続のリンク設定を無視し、デバイス上の設定を維持します。

リンクネゴシエーションを無視するには、次のようにパラメーターを設定します。

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



重要

auto-negotiate パラメーターが no に設定されていても、speed および duplex の値が設定されていない場合、これはオートネゴシエーションが無効になっているわけではありません。

オートネゴシエーションを強制的にアクティブ化する

この場合、NetworkManager はデバイスでオートネゴシエーションを強制的に実行します。

オートネゴシエーションを強制的にアクティブ化するには、以下のようにオプションを設定します。

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

リンクの speed および duplex を手動で設定する

この場合は、リンクの speed および duplex 設定を手動で設定できます。

リンク設定 speed および duplex を手動で設定するには、上記のパラメーターを以下のように設定します。

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

重要

必ず speed と duplex の値の両方を設定してください。設定しないと、NetworkManager はリンク設定を更新しません。

システム管理者は、次のいずれかの方法で 802.3 リンク設定を設定できます。

- **nmcli ツール**
- **nm-connection-editor ユーティリティー**

nmcli ツールを使用した 802.3 リンク設定の設定

手順

1. ***enp1s0* デバイス用に、新規イーサネット接続を作成します。**
2. **希望の 802.3 リンクセッティングを設定します。詳細は、を参照してください。** **「802.3 リンクセッティングの設定」**

   たとえば、speed オプション *100 Mbit/s* および duplex を *full* に手動で設定するには、次のコマンドを実行します。

```
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-editor による 802.3 リンク設定の設定

手順

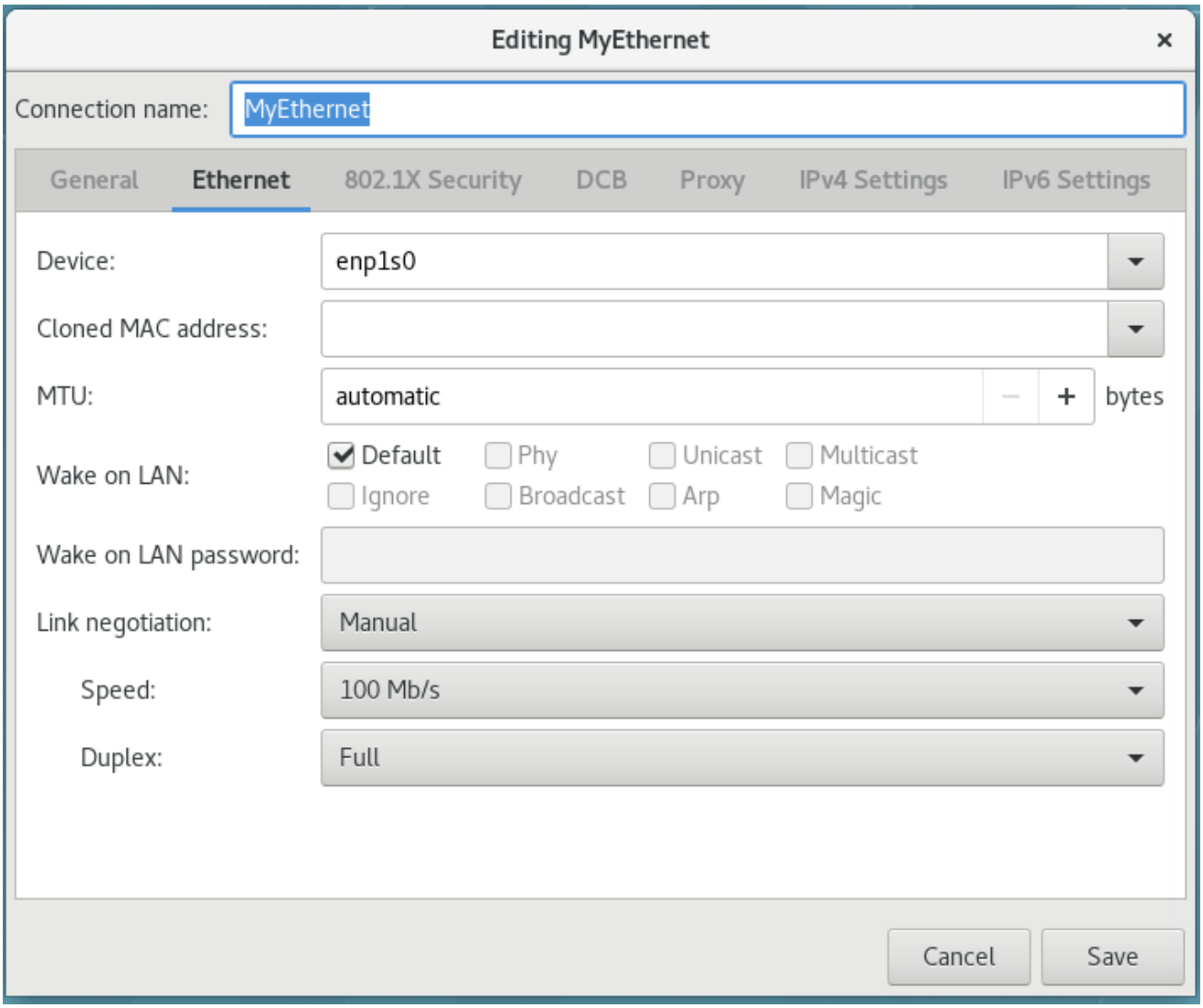
1. 端末に **nm-connection-editor** と入力します。

2. 編集するイーサネット接続を選択し、歯車アイコンをクリックして、編集ダイアログに移動します。詳細は、「nm-connection-editor を使用した一般的な設定オプション」を参照してください。

3. 希望のリンクネゴシエーションを選択します。

   - **ignore:** リンク設定はスキップされます（デフォルト）。
   - **Automatic:** デバイスに対してリンクオートネゴシエーションが強制されます。
   - **Manual: Speed** オプションおよび **Duplex** オプションを指定して、リンクネゴシエーションを強制することができます。

図5.1 nm-connection-editor を使用した 802.3 リンクセッティングの設定



5.2. 802.1X セキュリティーの設定

802.1X セキュリティーとは、*ポートベースのネットワークアクセス制御* (PNAC) 用の IEEE 基準の名前です。これは、*WPA Enterprise* とも呼ばれます。802.1X セキュリティーは、物理ネットワークから *論理ネットワーク* へのアクセスを制御する手段です。論理ネットワークに参加するクライアントはすべて、正しい 802.1X 認証方法を使用して、ルーターなどのサーバーで認証を行う必要があります。

802.1X セキュリティーは、ほとんどの場合、ワイヤレスネットワーク (WLAN) のセキュリティー保護に関連付けられていますが、ネットワーク (LAN) に物理的にアクセスする侵入者が侵入するのを防ぐためにも使用できます。

以前は、DHCP サーバーは、許可されていないユーザーに IP アドレスをリースしないように設定されていましたが、さまざまな理由から、これは実用的ではなく、安全ではないため、推奨されなくなり

ました。代わりに、802.1X セキュリティーを使用して、ポートベースの認証を通じて、論理的に安全なネットワークを確保します。

802.1X は、WLAN と LAN のアクセス制御のためのフレームワークを提供して、EAP (Extensible Authentication Protocol) タイプの 1 つを運搬するエンベロープとして機能します。EAP のタイプとは、ネットワーク上でセキュリティーの達成方法を定義するプロトコルです。

5.2.1. nmcli を使用した Wi-Fi 用の 802.1X セキュリティーの設定

手順

1. 認証された key-mgmt （キー管理）プロトコルを設定します。セキュアな wifi 接続の鍵設定メカニズムを設定します。プロパティーの詳細は 『nm-settings(5)』 の man ページを参照してください。

2. 802-1x 認証設定の設定TLS (Transport Layer Security) 認証については、 「TLS の設定」を参照してください。

表5.1 802-1x 認証設定

802-1x 認証設定	名前
802-1x.identity	アイデンティティー
802-1x.ca-cert	CA 証明書
802-1x.client-cert	ユーザー証明書
802-1x.private-key	秘密鍵
802-1x.private-key-password	秘密鍵のパスワード

たとえば、EAP-TLS 認証メソッドを使用して WPA2 Enterprise を設定するには、以下の設定を適用します。

```
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. nmcli を使用した有線用の 802.1X セキュリティーの設定

nmcli ツールを使用して 有線 接続を設定するには、ワイヤレス 接続と同じ手順に従います(802-11-wireless.ssid および 802-11-wireless-security.key-mgmt 設定を除く)。

5.2.3. GUI を使用した Wi-Fi 用の 802.1X セキュリティーの設定

手順

1. Network ウィンドウを開きます(「control-center GUI を使用したネットワーク接続」を参照してください)。

2. 右側のメニューから ワイヤレス ネットワークインターフェイスを選択します。必要に応じて、電源ボタンを ON に設定し、ハードウェアスイッチがオンであることを確認します。

3. 802.1X セキュリティーを設定する新規接続の接続名を選択するか、既存の接続プロファイルのギアのアイコンをクリックします。新規接続の場合、必要な認証手順を完了して接続を完了させてからギアのアイコンをクリックします。

4. Security を選択します。

   以下の設定オプションが利用できます。

   Security

   none: Wi-Fi 接続を暗号化しません。

   WEP 40/128-bit Key - IEEE 802.11 標準からの Wired Equivalent Privacy (WEP)。共有キー (PSK) を 1 つ使用します。

WEP 128 ビットパスフレーズ - パスフレーズの MD5 ハッシュを使用して WEP キーを取得します。

LEAP - Cisco Systems の Lightweight Extensible Authentication Protocol。

動的 WEP (802.1X) - WEP キーは動的に変更されます。用途「TLS の設定」

WPA & WPA2 Personal - IEEE 802.11i 標準からの Wi-Fi Protected Access (WPA)。WEP の代替。802.11i-2004 規格の Wi-Fi Protected Access II (WPA2)。個人モードは、事前共有キー (WPA-PSK) を使用します。

WPA & WPA2 Enterprise - IEEE 802.1X ネットワークアクセス制御を提供するために RADIUS 認証サーバーで使用する WPA。用途「TLS の設定」

Password

認証プロセスで使用するパスワードを入力します。

5. ドロップダウンメニューから、LEAP、Dynamic WEP (802.1X)、または WPA & WPA2 Enterprise のセキュリティー方法のいずれかを選択します。

Security ドロップダウンメニューでの選択に対応する *拡張認証プロトコル* (EAP)タイプは、「TLS の設定」 を参照してください。

5.2.4. nm-connection-editor を使用した有線用の 802.1X セキュリティーの設定

手順

1. 端末に nm-connection-editor を入力します。

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

- ネットワーク接続 ウィンドウが 表示されます。

2. 編集するイーサネット接続を選択し、歯車アイコンをクリックしてください。「nm-connection-editor を使用した有線接続の設定」を参照してください。

3. セキュリティー を 選択 し、電源ボタンを ON に設定して、設定を有効にします。

4. 以下のいずれかの認証方法を選択します。

   - *Transport Layer Security* には TLS を選択し、「TLS の設定」 に進みます。
   - FAST を選択して *Flexible Authentication through Secure Tunneling* を選択し、「Tunneled TLS の設定」 に進みます。
   - *Tunneled Transport Layer Security* (TTLS または EAP-TTLS として知られる)には Tunneled TLS を選択し、「Tunneled TLS の設定」 に進みます。
   - *Protected Extensible Authentication Protocol* には Protected EAP (PEAP) を選択し、「Protected EAP (PEAP) の設定」 に進みます。

TLS の設定

トランスポート層セキュリティー (TLS) では、クライアントとサーバーは、TLS プロトコルを使用した相互認証が行われます。サーバーはデジタル証明書を維持していることを示し、クライアントはクライアント側の証明書を使用して自身の ID を証明することで、キー情報が交換されます。認証が完了すると、TLS トンネルの使用は終了します。その代わりにクライアントとサーバーは交換したキーで、AES、TKIP、WEP のいずれかを使用してデータを暗号化します。

認証を希望する全クライアントに証明書が配布される必要があるということは、EAP-TLS 認証のメソッドが非常に強力であることを意味しますが、セットアップはより複雑になります。TLS セキュリティーを使用すると、証明書を管理する公開鍵インフラストラクチャー (PKI) のオーバーヘッドが必要になります。TLS セキュリティーを使用する利点は、パスワードが危険にさらされても (W)LAN へのアクセスが許可されないことです。侵入者は、認証するクライアントのプライベートキーにもアクセスを必要とします。

NetworkManager は、対応している TLS のバージョンを決定しません。NetworkManager は、

ユーザーが入力するパラメーターを収集し、手順を処理するデーモン wpa_supplicant に渡します。このデーモンは、OpenSSL を使用して TLS トンネルを確立します。OpenSSL 自体は、SSL/TLS プロトコルバージョンを処理します。両端が対応する一番高いバージョンが使用されます。

TLS を設定するには、「nm-connection-editor を使用した有線用の 802.1X セキュリティーの設定」 で説明されている手順に従います。以下の設定が可能です。

アイデンティティー

このサーバーの識別子を入力します。

ユーザー証明書

個人用 X.509 証明書ファイルをブラウズして選択します。これは、*Distinguished Encoding Rules* (DER) または *Privacy Enhanced Mail* (PEM) でエンコードされたものです。

CA 証明書

X.509 *認証局* 証明書ファイルをブラウズして選択します。これは、*Distinguished Encoding Rules* (DER) または *Privacy Enhanced Mail* (PEM) でエンコードされたものです。

秘密鍵

*プライベートキー*をブラウズして選択します。これは、*Distinguished Encoding Rules* (DER)、*Privacy Enhanced Mail* (PEM)、または *Personal Information Exchange Syntax Standard* (PKCS #12) でエンコードされたものです。

秘密鍵のパスワード

Private key フィールドに秘密鍵のパスワードを入力します。Show password を選択して、入力時にパスワードを表示します。

FAST の設定

FAST を設定するには、「nm-connection-editor を使用した有線用の 802.1X セキュリティーの設定」 で説明されている手順に従います。以下の設定が可能です。

Anonymous Identity

このサーバーの識別子を入力します。

PAC プロビジョニング

チェックボックスを選択してから Anonymous、Authenticated、および both から選択し ます。

PAC file

クリックしてブラウズし、*protected access credential* (PAC) ファイルを選択します。

Inner authentication

GTC - Generic Token Card。

MSCHAPv2 - Microsoft Challenge Handshake Authentication Protocol version 2.

Username

認証プロセスで使用するユーザー名を入力します。

Password

認証プロセスで使用するパスワードを入力します。

Tunneled TLS の設定

Tunneled TLS を設定するには、「nm-connection-editor を使用した有線用の 802.1X セキュリティーの設定」 で説明されている手順に従います。以下の設定が可能です。

Anonymous identity

この値は、非暗号化 ID として使用されます。

CA 証明書

クリックしてブラウズし、認証局 (CA) の証明書を選択します。

Inner authentication

PAP - パスワード認証プロトコル。

MSCHAP - チャレンジハンドシェイク認証プロトコル

MSCHAPv2 - Microsoft Challenge Handshake Authentication Protocol version 2.

CHAP - チャレンジハンドシェイク認証プロトコル

Username

認証プロセスで使用するユーザー名を入力します。

Password

認証プロセスで使用するパスワードを入力します。

Protected EAP (PEAP) の設定

Protected EAP (PEAP)を設定するには、「nm-connection-editor を使用した有線用の 802.1X セキュリティーの設定」 に記載されている手順に従います。以下の設定が可能です。

Anonymous Identity

この値は、非暗号化 ID として使用されます。

CA 証明書

クリックしてブラウズし、認証局 (CA) の証明書を選択します。

PEAP version

使用する、保護された EAP のバージョン。Automatic、0、1 のいずれか。

Inner authentication

MSCHAPv2 - Microsoft Challenge Handshake Authentication Protocol version 2.

MD5 - メッセージダイジェスト 5、暗号ハッシュ関数。

GTC - Generic Token Card。

Username

認証プロセスで使用するユーザー名を入力します。

Password

認証プロセスで使用するパスワードを入力します。

5.3. WPA_SUPPLICANT および NETWORKMANAGERでの MACSEC の使用

Media Access Control Security (MACsec、IEEE 802.1AE)は、LAN 内のすべてのトラフィックを GCM-AES-128 アルゴリズムで暗号化および認証します。MACsec は、IP だけでなく、ARP (Address Resolution Protocol)、ND (Neighbor Discovery)、または DHCP も保護できます。IPsec はネットワーク層（レイヤー 3）および SSL または TLS 上でアプリケーション層（レイヤー 7）で動作しますが、MACsec はデータリンク層（レイヤー 2）で動作します。MACsec と他のネットワーク層のセキュリティープロトコルを組み合わせて、これらの標準が提供するさまざまなセキュリティー機能を利用します。

事前に共有されている CAK/CKN (Connectivity Association Key/CAK Name)ペアを使用して認証を実行するスイッチを使用して MACsec を有効にするには、以下を実行します。

手順

1. **CAK/CKN ペアを作成します。たとえば、次のコマンドにより、16 バイトの鍵が 16 進数表記で生成されます。**

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

2. **wpa_supplicant.conf 設定ファイルを作成し、次の行を追加します。**

```
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
}
```

前の手順の値を使用して、**wpa_supplicant.conf** 設定ファイルの **mka_cak** 行と **mka_ckn** 行を完了します。

詳細は、**man** ページの **wpa_supplicant.conf (5)** を参照してください。

3. ***wlp61s0*** を使用してネットワークに接続している場合は、以下のコマンドを使用して **wpa_supplicant** を起動します。

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

Red Hat では、**wpa_supplicant.conf** ファイルを作成および編集する代わりに、**nmcli** コマンドを使用して、前述の手順と同じように **wpa_supplicant** を設定することを推奨します。以下の例では、すでに **16** バイトの **16** 進数 **CAK ($MKA_CAK)**と **32** バイトの **16** 進数 **CKN ($MKA_CKN)**があることを前提としています。

```
~]# 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+
```

この手順の後に、***macsec0*** デバイスを設定してネットワーク設定に使用する必要があります。

詳細は、**What's new in MACsec: setting up MACsec using wpa_supplicant and (optionally) NetworkManager**の記事を参照してください。さらに、**MACsec** ネットワークのアーキテクチャー、ユースケースシナリオ、設定例の詳細は、**MACsec: a different solution to encrypt network traffic** の記事を参照してください。

5.4. IPV4 設定の設定

control-center を使用した IPv4 設定の設定

手順

1. Super キーを押してアクティビティーの概要に入り、Settings と 入力 して Enter を押します。次に、左側の Network タブを選択すると、Network 設定ツールが表示されます。「control-center を使用した新しい接続の設定」に進みます。

2. 編集する接続を選択し、歯車アイコンをクリックします。編集 ダイアログが表示されます。

3. IPv4 メニューエントリーをクリックします。

IPv4 メニューエントリーでは、ネットワークへの接続に使用する方法を設定し、必要に応じて IP アドレス、DNS、およびルート情報を入力します。IPv4 メニューエントリーは、有線、ワイヤレス、モバイルブロードバンド、VPN、DSL のいずれかを作成して変更するときに利用できます。

DHCP を使用して DHCP サーバーから動的 IP アドレスを取得する場合は、アドレス を Automatic (DHCP) に設定するだけです。

静的ルートを設定する必要がある場合は、「GUI を使用した静的ルートの設定」を参照してください。

nm-connection-editor を使用した IPV4 の方法の設定

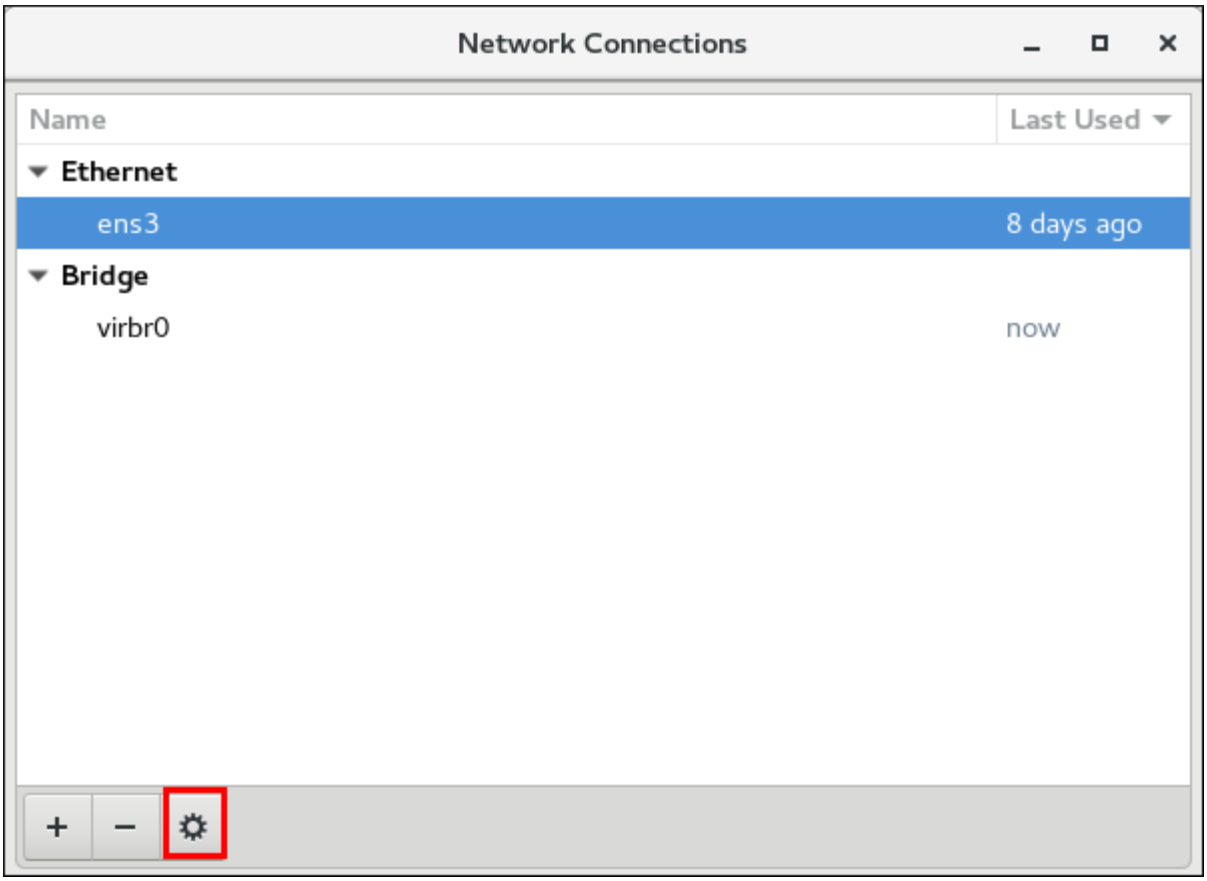
nm-connection-editor を使用して、接続設定を編集および設定できます。この手順では、IPv4 設定方法を説明します。

手順

1. 端末に nm-connection-editor と入力します。

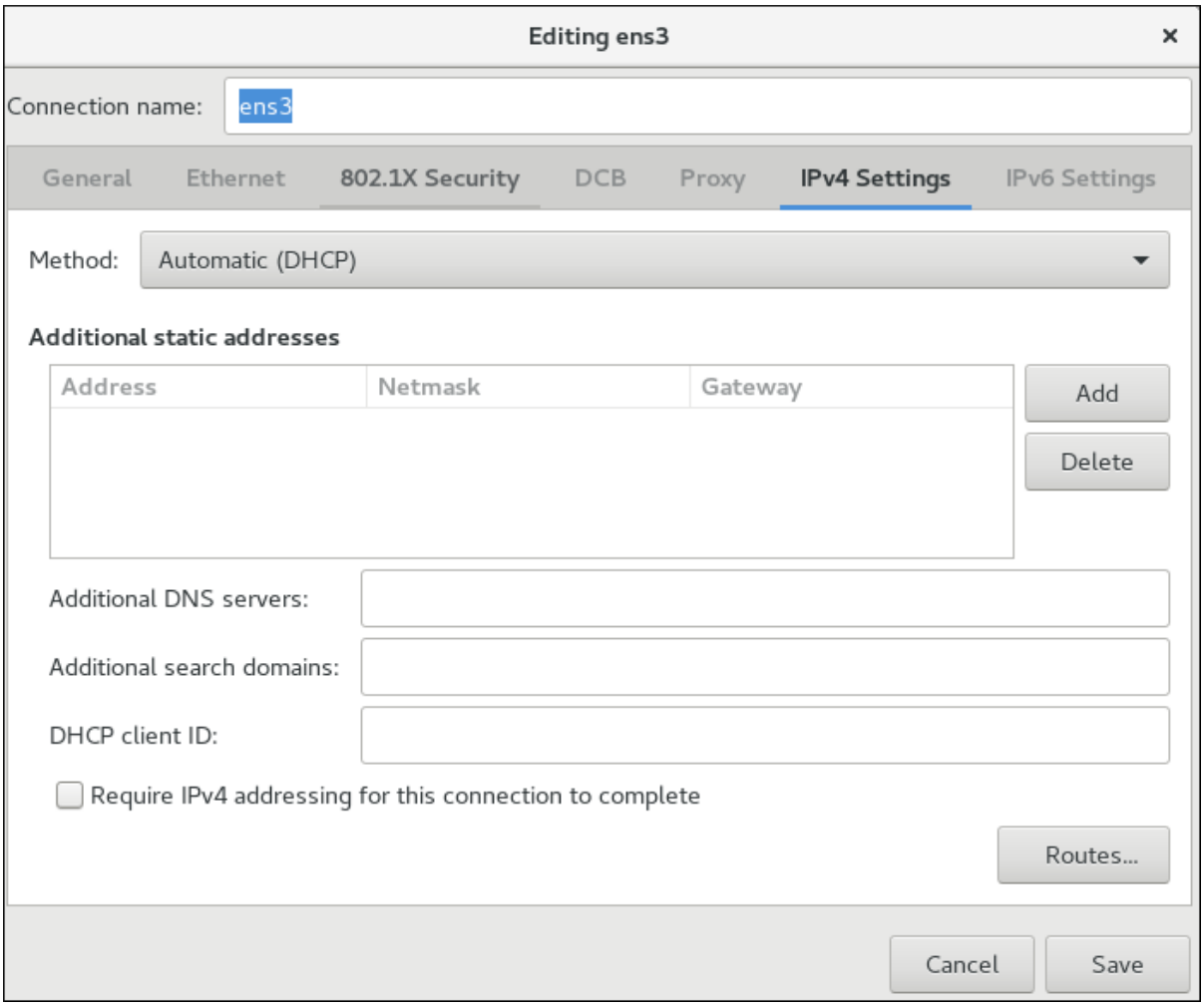
2. 既存の接続タイプの場合は、歯車アイコンをクリックします。

図5.2 接続の編集



3. **IPv4 Settings** をクリックします。

図5.3 IPv4 設定の設定



接続の種類別で利用可能な IPv4 方式

Method ドロップダウンメニューをクリックすると、設定している接続のタイプに応じて、以下の IPv4 接続方法のいずれかを選択できます。関連のある接続の種類に応じてすべての方式がここにリスト表示されています。

有線、ワイヤレス、DSL 接続の方式

Automatic (DHCP) - 接続しているネットワークが IP アドレスの割り当てに DHCP サーバーを使用する場合は、このオプションを選択します。DHCP クライアント ID フィールドに入力する必要はありません。

自動(DHCP)アドレスのみ - 接続しているネットワークが IP アドレスの割り当てに DHCP サーバーを使用しているが、DNS サーバーを手動で割り当てたい場合は、このオプションを選択します。

Manual: IP アドレスを手動で割り当てる場合は、このオプションを選択します。

Link-Local Only - 接続しているネットワークに DHCP サーバーがなく、IP アドレスを手動で割り当てない場合は、このオプションを選択します。接頭辞 169.254/16 が付いたランダムなアドレスが 『RFC 3927』 に従って割り当てられます。

他のコンピューターと共有 - 設定しているインターフェイスがインターネットまたは WAN 接続を共有する場合は、このオプションを選択します。このインターフェイスには、10.42 .x.1/24 範囲のアドレスが割り当てられ、DHCP サーバーおよび DNS サーバーが起動し、ネットワーク *アドレス変換* (NAT)を使用してシステム上のデフォルトのネットワーク接続にインターフェイスが接続されています。

Disabled - この接続では IPv4 は無効になっています。

モバイルブロードバンド接続の方式

自動(PPP) - 接続しているネットワークが IP アドレスと DNS サーバーを自動的に割り当てる場合は、このオプションを選択します。

自動(PPP)アドレスのみ - 接続しているネットワークが IP アドレスを自動的に割り当てても DNS サーバーを手動で指定する場合は、このオプションを選択します。

VPN 接続の方式

自動(VPN) - 接続しているネットワークが IP アドレスと DNS サーバーを自動的に割り当てる場合は、このオプションを選択します。

自動(VPN)アドレスのみ - 接続しているネットワークが IP アドレスを自動的に割り当てても DNS サーバーを手動で指定する場合は、このオプションを選択します。

DSL 接続の方式

自動(PPPoE) - 接続しているネットワークが IP アドレスと DNS サーバーを自動的に割り当てる場合は、このオプションを選択します。

自動(PPPoE)アドレスのみ - 接続しているネットワークが IP アドレスを自動的に割り当てても DNS サーバーを手動で指定する場合は、このオプションを選択します。

DHCP を使用して DHCP サーバーから動的 IP アドレスを取得する場合は、method を Automatic (DHCP) に設定するだけです。

静的ルートを設定する必要がある場合は、Routes ボタンをクリックします。設定オプションの詳細は、「GUI を使用した静的ルートの設定」 を参照してください。

5.5. IPV6 セッティングの設定

IPv6 を設定するには、「IPv4 設定の設定」 に記載されている手順に従い、IPv6 メニューエントリーをクリックします。

メソッド

ignore: この接続の IPv6 設定を無視する場合は、このオプションを選択します。

Automatic: *SLAAC* を使用して、ハードウェアアドレスおよび *ルーター広告* (RA)に基づいてステートレスの自動設定を作成するには、このオプションを選択します。

自動、アドレスのみ - 接続しているネットワークが *ルーター広告* (RA)を使用して自動ステートレス設定を作成するが、DNS サーバーを手動で割り当てたい場合は、このオプションを選択します。

自動、DHCP のみ - RA を使用しないが、DHCPv6 からの情報を直接要求してステートフルな設定を作成する場合は、このオプションを選択します。

Manual: IP アドレスを手動で割り当てる場合は、このオプションを選択します。

Link-Local Only - 接続しているネットワークに DHCP サーバーがなく、IP アドレスを手動で割り当てない場合は、このオプションを選択します。接頭辞 FE80::0 の付いたランダムなアドレスが 『RFC 4862』 に従って割り当てられます。

アドレス

DNS サーバー: DNS サーバーのコンマ区切りリストを入力します。

ドメインの検索: ドメインコントローラーのコンマ区切りリストを入力します。

静的ルートを設定する必要がある場合は、Routes ボタンをクリックします。設定オプションの詳細は、「GUI を使用した静的ルートの設定」 を参照してください。

5.6. PPP (ポイントツーポイント) セッティングの設定

Authentication Methods

多くの場合、プロバイダーの PPP サーバーは許可されたすべての認証方法をサポートしています。接続に失敗する場合は、PPP サーバーの設定に応じて一部の認証方法のサポートを無効にする必要があります。

MPPE (ポイントツーポイント暗号化) を使用

Microsoft のポイントツーポイント暗号化プロトコル (『RFC 3078』)。

BSD データ圧縮を許可する

PPP BSD 圧縮プロトコル (『RFC 1977』)。

Deflate データ圧縮を許可する

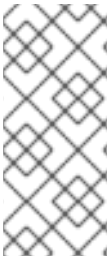
PPP Deflate プロトコル (『RFC 1979』)。

TCP ヘッダー圧縮を使用

低スピードシリアルリンク用に TCP/IP ヘッダーを圧縮します (『RFC 1144』)。

PPP echo のパケットを送信

ループバックテスト用の LCP Echo 要求および Echo 応答コード (『RFC 1661』)。



注記

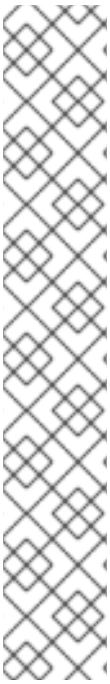
NetworkManager の PPP サポートは任意であるため、PPP 設定を設定するには、NetworkManager-ppp パッケージがすでにインストールされていることを確認してください。

第6章 ホスト名の設定

6.1. ホスト名について

ホスト 名には、static、pretty、および transient の 3 つのクラスがあります。

「静的」 ホスト名は従来のホスト 名 で、ユーザーが選択でき、/etc/hostname ファイルに保存されます。「一時的な」 ホスト 名 は、カーネルによって維持される動的なホスト名です。デフォルトでは static ホスト名に初期化され、その値は「localhost」になります。実行時に DHCP または mDNS で変更できます。「pretty」 ホスト 名 は、ユーザーに表示するためのフリーフォームの UTF8 ホスト名です。



注記

ホスト名は、最大 64 文字の長さの自由形式の文字列にすることができます。ただし、Red Hat では、static および transient の両方の名前が host.example.com などの DNS 内のマシンに使用される *完全修飾ドメイン名* (FQDN)と一致することを推奨しています。また、静的および一時的な名前は、厳密な要件ではない場合でも、7 ビットの ASCII 小文字の下位文字のみで設定され、スペースやドットはなく、DNS ドメイン名ラベルで許可されている形式に制限することが推奨されます。ただし、これは厳密な要件ではありません。従来の仕様ではアンダースコアは禁止されているので、この使用も推奨されません。

hostnamectl ツールは、Static および transient のホスト名が、a-z、A-Z、0 「-」 9、-、「_」、および で設定されます 「。」 唯一の場合は、ドットで開始または終了しないようにし、それぞれに続く 2 つのドットをたどらないようにします。また、最大 64 文字までの長さも強制されます。

6.1.1. 推奨される命名プラクティス

Internet Corporation for Assigned Names and Numbers (ICANN)は、以前に未登録のトップレベルドメイン(.yourcompany等)を公開レジスターに追加することがあります。このため、Red Hat では、プライベートネットワーク上であっても委任されていないドメイン名を使用しないことを強く推奨しています。その結果、ネットワークリソースは利用できなくなります。また、委任されていないドメイン名を使うと、DNSSEC の実装および維持がより困難になります。これは、ドメイン名の競合が DNSSEC 検証の有効化に手動の設定ペナルティーを必要とするためです。この問題の詳細は、ドメイン名の衝突に関する ICANN のよくある質問を参照してください。

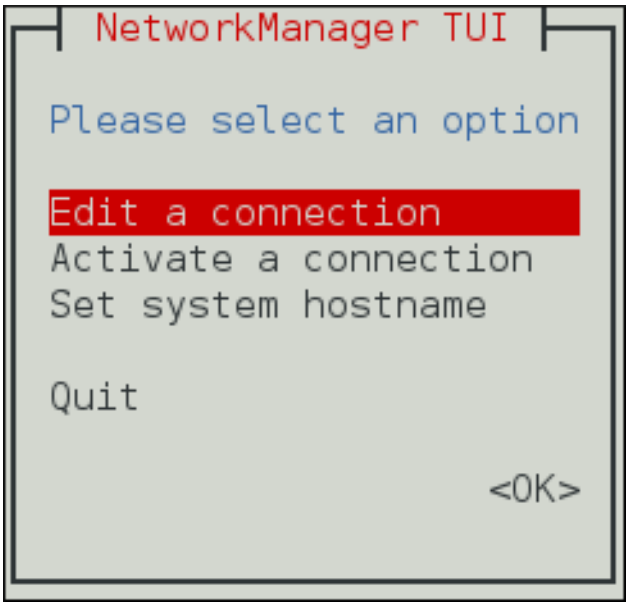
6.2. テキスト形式のユーザーインターフェイス NMTUI を使用したホスト名の設定

テキスト形式のユーザーインターフェイスツール nmtui を使用すると、ターミナルのウィンドウでホスト名を設定できます。このツールを起動するには、以下のコマンドを実行します。

```
~]$ nmtui
```

テキストユーザーインターフェイスが表示されます。無効なコマンドがあると、使用方法に関するメッセージが表示されます。

図6.1 NetworkManager のテキスト形式ユーザーインターフェイスの開始メニュー



[D]

移動するには、矢印キーを使用するか、Tab を押して順方向に進み、Shift+Tab を押してオプションを再度実行します。Enter を押してオプションを選択します。Space バーは、チェックボックスのステータスを切り替えます。

nmtui のインストール方法は、「nmtui を使用した IP ネットワークの設定」 を参照してください。

NetworkManager のテキストユーザーインターフェイスツール nmtui を使用すると、/etc/hostname ファイルで静的ホスト名をクエリーして設定できます。

重要

Red Hat Enterprise Linux では、NetworkManager は systemd-hostnamed サービスを使用して、/etc/hostname ファイルに保存される静的ホスト名の読み取りおよび書き込みを行います。このため、/etc/hostname ファイルに対する手動の変更が NetworkManager によって自動的に取得されなくなりました。システムのホスト名は、hostnamectl ユーティリティーで変更する必要があります。また、/etc/sysconfig/network ファイルで HOSTNAME 変数の使用が非推奨になりました。

6.3. HOSTNAMECTL を使用したホスト名の設定

hostnamectl ツールは、特定のシステム上で使用中の 3 つのクラスのホスト名を管理するためのものです。

6.3.1. 全ホスト名の表示

現行のホスト名をすべて表示するには、以下のコマンドを実行します。

```
~]$ hostnamectl status
```

オプションが指定されていない場合、status オプションはデフォルトで暗示されています。

6.3.2. 全ホスト名の設定

システム上のすべてのホスト名を設定するには、root で以下のコマンドを実行します。

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

このコマンドは、pretty、static、および transient のホスト名を同様に変更します。static および transient ホスト名は、pretty ホスト名のシンプルな形式です。スペースは 「-」 に置き換えられ、特殊文字は削除されます。

6.3.3. 特定のホスト名の設定

特定のホスト名を設定するには、関連するオプションを指定して、root で以下のコマンドを入力します。

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

ここでの *オプション* は、-- pretty、-- static、および --transient のいずれかです。

--static オプションまたは --transient オプションが --pretty オプションとともに使用すると、静的および一時的なホスト名は Pretty ホスト名の形式を簡素化されます。スペースは 「-」 に置き換えられ、特殊文字は削除されます。--pretty オプションを使用しなければ、簡素化されることはありません。

pretty ホスト名を設定する際、ホスト名に空白や単一引用符が含まれているのであれば、適切な引用符を用いてください。以下に例を示します。

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

6.3.4. 特定のホスト名の削除

特定のホスト名を削除してデフォルトに戻すには、root で以下のコマンドを関連するオプションと共に実行します。

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

"" は引用符付きの 空の文字列で、*オプション* は --pretty、--static、および --transient のいずれかです。

6.3.5. ホスト名のリモートでの変更

リモートシステムで hostnamectl コマンドを実行するには、以下のように -H, --host オプションを使用します。

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

ここでの *hostname* は、設定対象となるリモートホストです。*username* はオプション選択になります。hostnamectl ツールは SSH を使用してリモートシステムに接続します。

6.4. NMCLI を使用したホスト名の設定

NetworkManager ツールの nmcli を使用して、/etc/hostname ファイルで静的ホスト名をクエリーして設定できます。

静的ホスト名にクエリーを実行するには、以下のコマンドを発行します。

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

静的ホスト名を *my-server* に設定するには、root で以下のコマンドを実行します。

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

6.5. 関連情報

- hostnamectl (1) man ページ - コマンドおよびコマンドオプションを含む hostnamectl について説明しています。
- hostname (1) man ページ - hostname コマンドおよび domainname コマンドについて説明しています。
- hostname (5) man ページ - ホスト名ファイル、その内容、および使用について説明します。
- hostname (7) man ページ - ホスト名の解決の説明が記載されています。
- machine-info (5) man ページ - ローカルマシン情報ファイルおよびそれに含まれる環境変数について説明しています。
- machine-id (5) man ページ - ローカルマシン ID 設定ファイルが説明されています。
- systemd-hostnamed.service (8) man ページ - hostnamectl が使用する systemd-hostnamed システムサービスについて説明しています。

第7章 ネットワークボンディングの設定

Red Hat Enterprise Linux 7 では、管理者が複数のネットワークインターフェイスを単一のチャンネルにまとめること (ボンディング) ができます。このチャンネルボンディングにより、複数のネットワークインターフェイスが 1 つとして機能できるようになり、また同時に帯域幅が増加し、冗長性を提供します。

警告

ネットワークスイッチを使わずにケーブルの直接接続を使用すると、ボンディングはサポートされません。本章で説明されているフェイルオーバーメカニズムは、ネットワークスイッチがないと予想どおりに機能しません。詳細についてはナレッジベースの記事『ボンディングは、クロスオーバーケーブルを使用したダイレクトコレクションをサポートしますか ?』を参照してください。

注記

active-backup、balance-tlb および balance-alb の各モードはスイッチの特定の設定を必要としません。他のボンディングモードでは、スイッチがリンクを集約するように設定する必要があります。たとえば、Cisco スイッチでは Modes 0、2、および 3 に EtherChannel を必要としますが、Mode 4 には LACP と EtherChannel が必要となります。スイッチで提供されるドキュメントを参照し、https://www.kernel.org/doc/Documentation/networking/bonding.txt を参照してください。

7.1. コントローラーおよびポートインターフェイスのデフォルト動作の理解

NetworkManager デーモンを使用してボンディングされたポートインターフェイスを制御する場合、特に障害検索時には、以下の点に留意してください。

1. コントローラーインターフェイスを起動しても、ポートインターフェイスは自動的に起動しない。

2. ポートインターフェイスを起動すると、コントローラーインターフェイスは毎回、起動する。

3. コントローラーインターフェイスを停止すると、ポートインターフェイスも停止する。

4. ポートのないコントローラーは静的 IP 接続を開始できる。

5. コントローラーにポートがない場合は、DHCP 接続の開始時にポートを待機します。

6. DHCP 接続でポートを待機中のコントローラーは、キャリアを含むポートが追加されると完了する。

7. DHCP 接続でポートを待機中のコントローラーは、キャリアをともなわないポートが追加されると待機を継続します。

7.2. テキスト形式のユーザーインターフェイス NMTUI を使ったボンディングの設定

テキスト形式のユーザーインターフェイスツール nmtui を使用すると、ターミナルのウィンドウでボンディングを設定できます。このツールを起動するには、以下のコマンドを実行します。

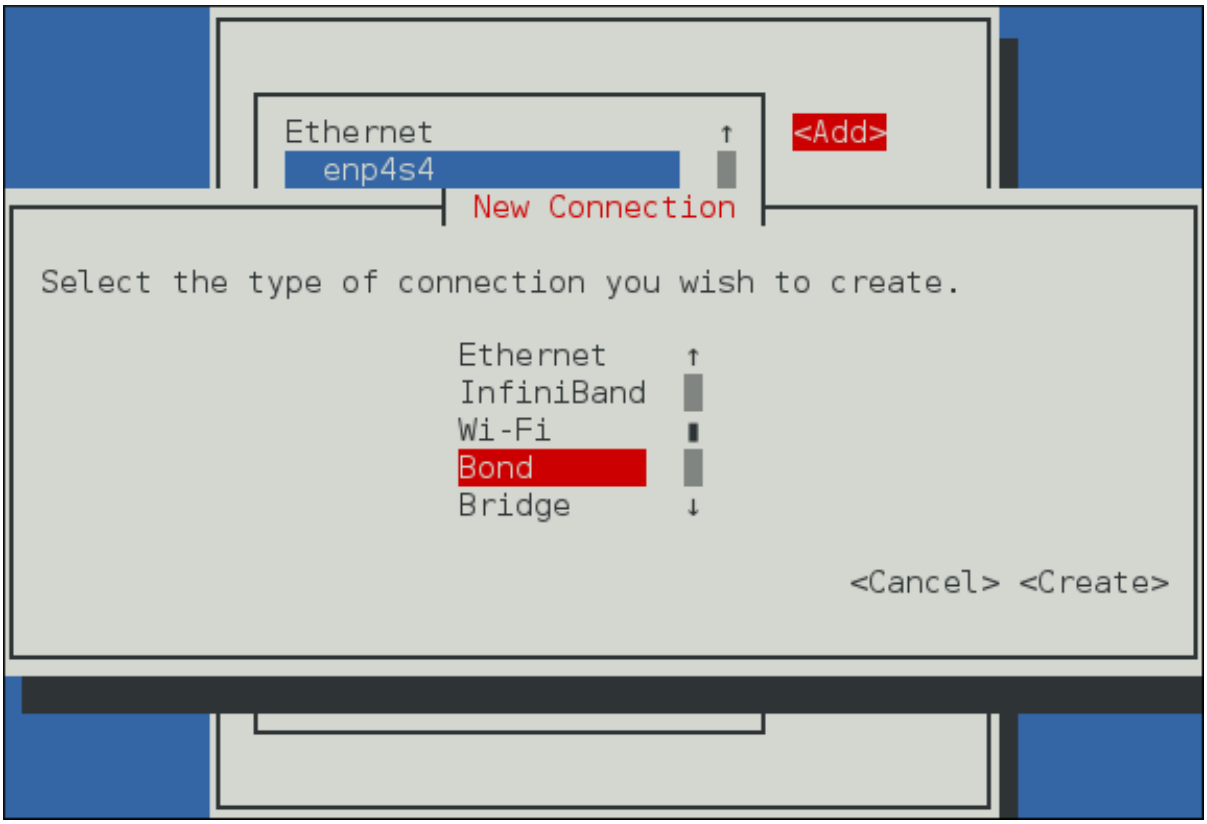
```
~]$ nmtui
```

テキストユーザーインターフェイスが表示されます。無効なコマンドがあると、使用方法に関するメッセージが表示されます。

移動するには、矢印キーを使用するか、Tab を押して順方向に進み、Shift+Tab を押してオプションを再度実行します。Enter を押してオプションを選択します。Space バーは、チェックボックスのステータスを切り替えます。

1. メニューから 接続の編集 を選択します。Add を選択すると、New Connection 画面が開きます。

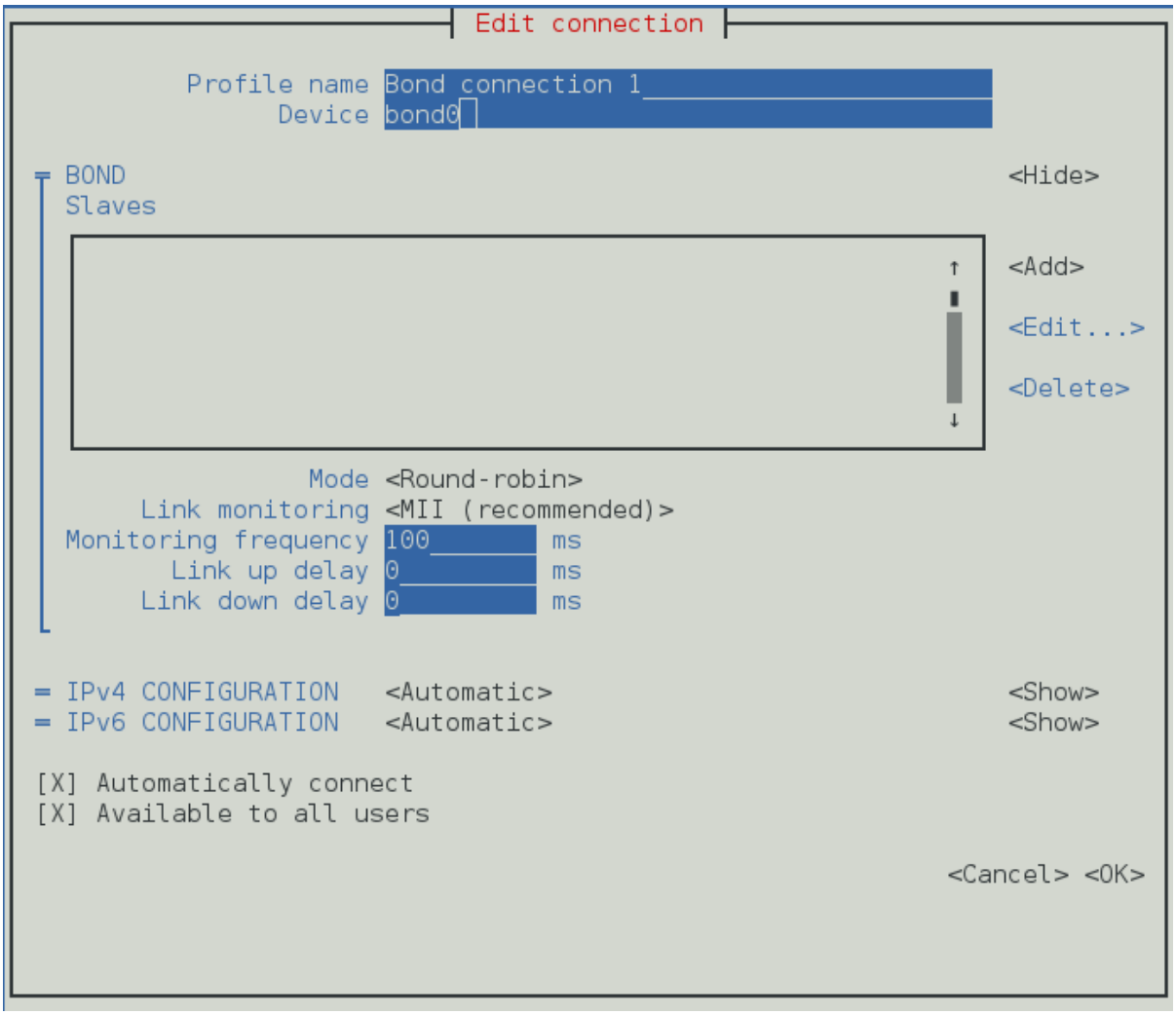
図7.1 NetworkManager テキスト形式のユーザーインターフェイスのボンド接続追加メニュー



[D]

2. **Bond を選択してから Create を選択します。ボンドの 接続の編集 画面が開きます。**

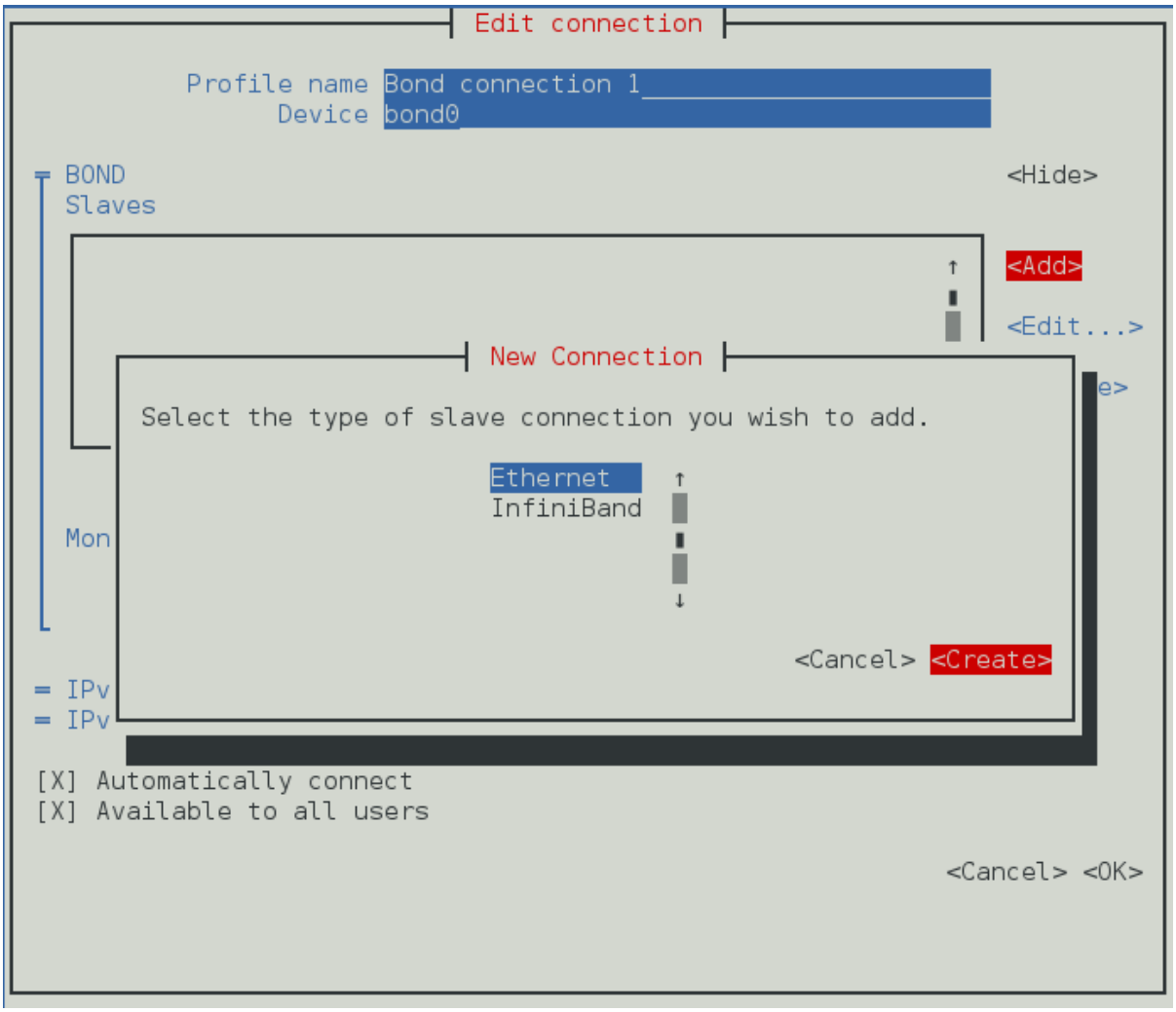
図7.2 NetworkManager テキスト形式ユーザーインターフェイスでボンド接続を設定するメニュー



[D]

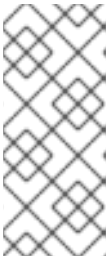
3. この時点で、ボンドにポートインターフェイスを追加する必要があります。これらを追加するには、Add を選択します。これにより、New Connection 画面が開きます。Connection のタイプを選択したら、Create ボタンを選択します。

図7.3 NetworkManager テキスト形式ユーザーインターフェイスで新規ボンドスレーブを設定するメニュー



[D]

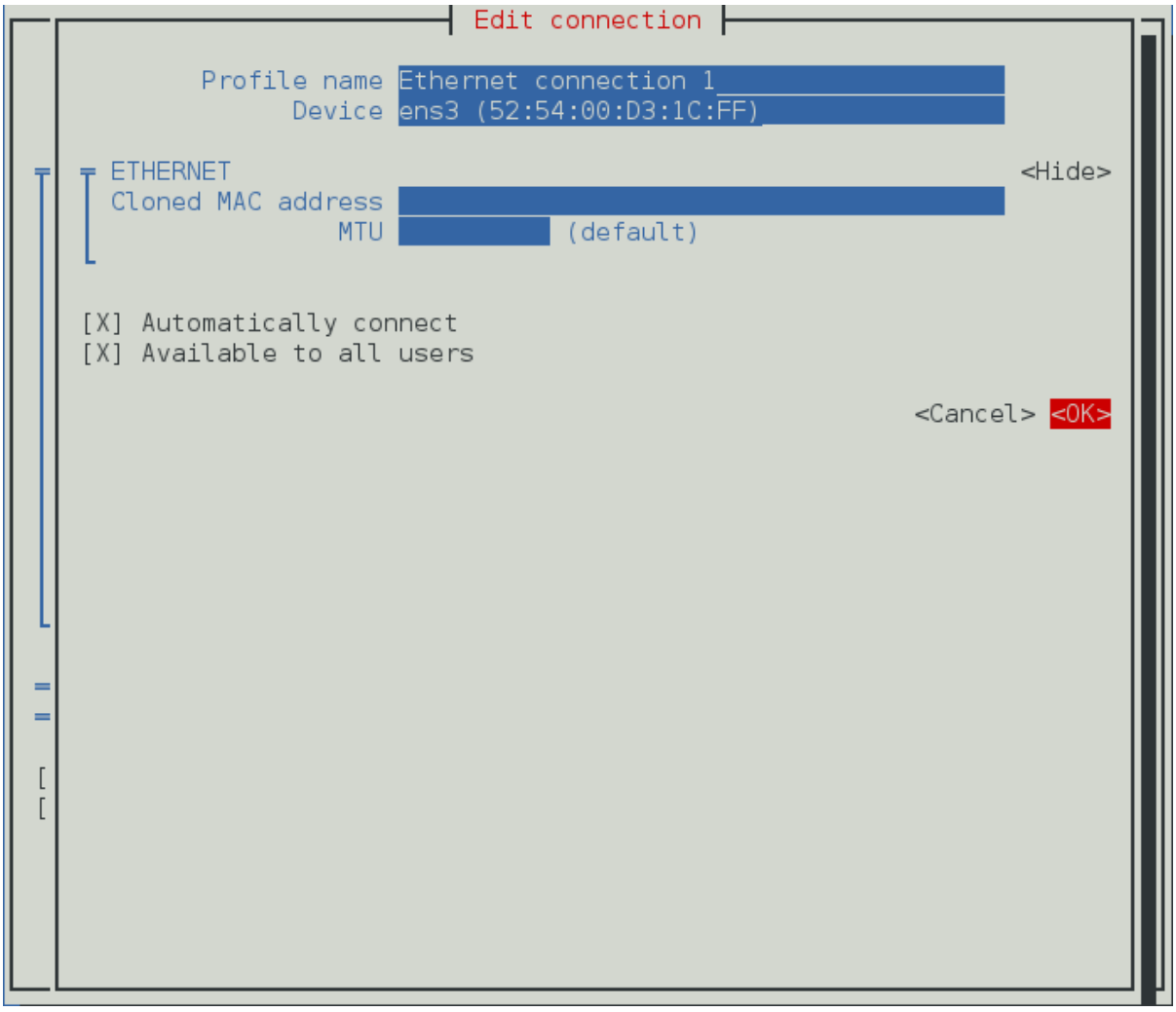
4. ポートの 接続の編集 画面が表示されます。Device セクションに、必要なポートのデバイス名または MAC アドレスを入力します。必要な場合は、イーサネット ラベルの右側にある Show を選択して、ボンドの MAC アドレスとして使用するクローンの MAC アドレスを入力します。OK ボタンを選択してポートを保存します。



注記

MAC アドレスなしでデバイスを指定すると、Edit Connection ウィンドウが再読み込みされると Device セクションが自動的に入力されますが、デバイスが正常に見つかった場合にのみデバイスセクションが自動的に設定されます。

図7.4 NetworkManager テキスト形式ユーザーインターフェイスでボンドスレーブ接続を設定するメニュー

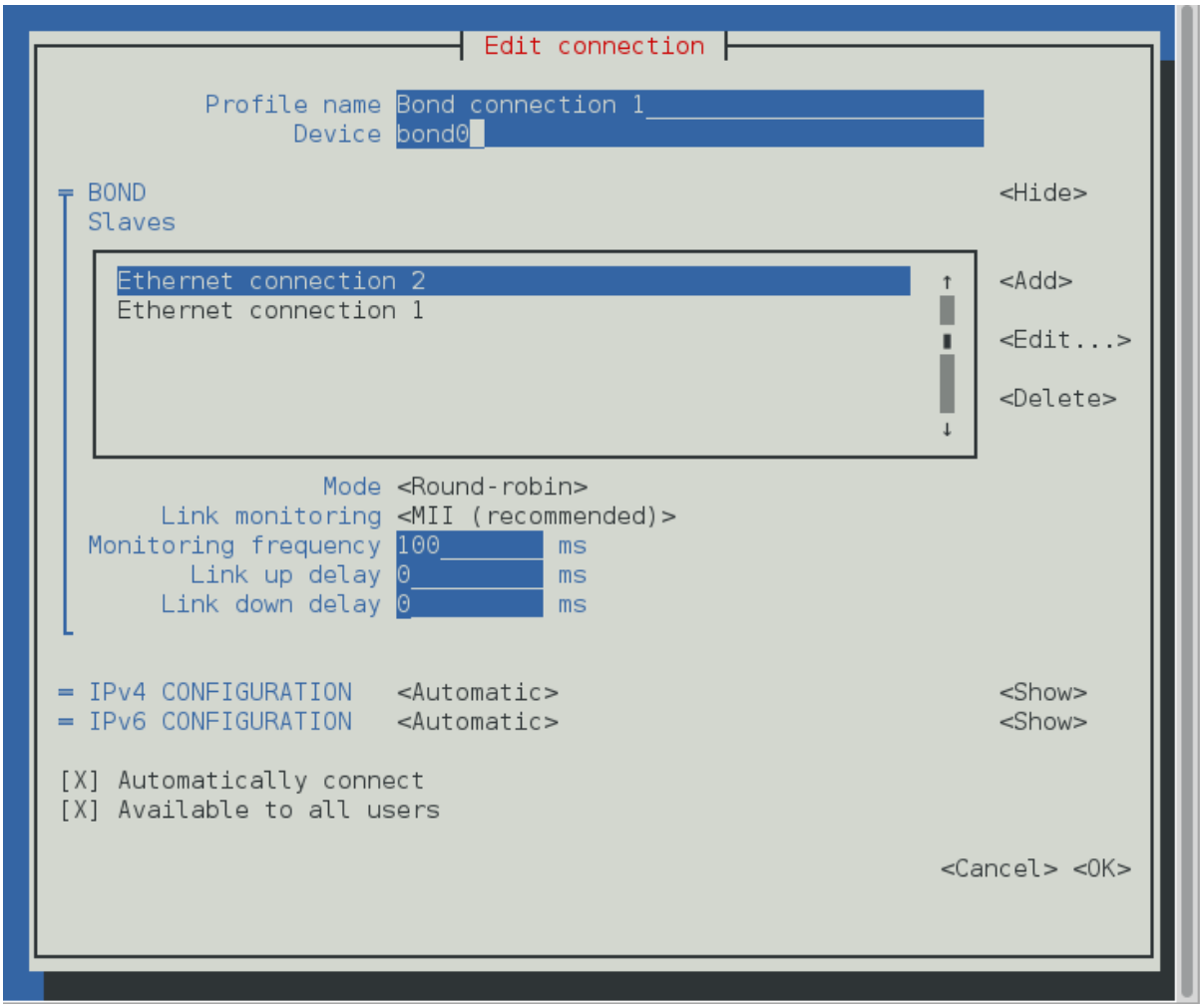


[D]

5. **Slaves セクションにボンディングポートの名前が表示されます。さらにポート接続を追加する場合は、上記のステップを繰り返します。**

6. **設定を確認してから、OK ボタンを選択します。**

図7.5 NetworkManager のテキスト形式ユーザーインターフェイスでボンドを完了



[D]

ボンド用語の定義は、「Bond タブの設定」を参照してください。

nmtui のインストール方法は、「nmtui を使用した IP ネットワークの設定」 を参照してください。

7.3. NETWORKMANAGER のコマンドラインツール NMCLI を使用したネットワークボンディング

注記

nmcli の概要は、「nmcli を使用する IP ネットワークの設定」 を参照してください。

nmcli ツールを使用して ボンド 接続を作成するには、以下のコマンドを実行します。

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

-

ボンドに **con-name** を指定しなかったため、接続名はインターフェイス名の前に種類を追加したものであったことに注意してください。

NetworkManager は、カーネルが提供するほとんどのボンディングオプションに対応します。以下に例を示します。

```
~]$ nmcli con add type bond ifname mybond0 bond.options "mode=balance-rr,miimon=100"
Connection 'bond-mybond0' (5f739690-47e8-444b-9620-1895316a28ba) successfully added.
```

ポートインターフェイスを追加するには、次の手順を実行します。

1. 新しいコネクションを作成します。詳細は「**nmcli による接続プロファイルの作成および修正**」を参照してください。

2. コントローラープロパティーを ボンド インターフェイス名、またはコントローラー接続の名前に設定します。

```
~]$ nmcli con add type ethernet ifname ens3 master mybond0
Connection 'bond-slave-ens3' (220f99c6-ee0a-42a1-820e-454cbabc2618) successfully added.
```

新しい ポート インターフェイスを追加するには、新しいインターフェイスで直前のコマンドを繰り返します。以下に例を示します。

```
~]$ nmcli con add type ethernet ifname ens7 master mybond0
Connection 'bond-slave-ens7' (ecc24c75-1c89-401f-90c8-9706531e0231) successfully added.
```

ポートをアクティブにするには、以下のコマンドを実行します。

```
~]$ 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)
```

ポートをアクティブ化すると、コントローラー接続も開始されます。詳細は、「コントローラーおよびポートインターフェイスのデフォルト動作の理解」 を参照してください。この場合、コントローラー接続を手動でアクティブ化する必要はありません。

接続を解除せずに、実行時にボンディングの **active_slave** オプションおよび **primary** オプションを変更できます。たとえば、**active_slave** オプションを変更するには、以下のコマンドを実行します。

```
~]$ nmcli dev mod bond0 +bond.options "active_slave=ens7"
Connection successfully reapplied to device 'bond0'.
```

または、**primary** オプションを変更する場合は、以下を実行します。

```
~]$ nmcli dev mod bond0 +bond.options "primary=ens3"
Connection successfully reapplied to device 'bond0'.
```

注記

active_slave オプションで設定したインターフェイスが、直ちにアクティブポートになります。一方、ボンドの **primary** オプションでは、新たなポートが追加された時またはアクティブポートに障害が発生した時にカーネルが自動的に選択するアクティブポートを指定します。

7.4. コマンドラインインターフェイス (CLI) の使用

ボンドは、ボンディングカーネルモジュールと、*チャンネルボンディング* インターフェイス と呼ばれる特別なネットワークインターフェイスを使用して作成されます。

7.4.1. ボンディングカーネルモジュールがインストールされているかの確認

Red Hat Enterprise Linux 7 では、ボンディングモジュールはデフォルトでは読み込まれません。**root** で以下のコマンドを実行して、モジュールを読み込むことができます。

```
~]# modprobe --first-time bonding
```

このアクティベーションは、システム再起動後は維持されません。永続的なモジュールの読み込みに関する詳細は、『Red Hat Enterprise Linux カーネル管理ガイド』を参照してください。**BONDING_OPTS** ディレクティブを使用した正しい設定ファイルを指定すると、ボンディングモジュールは必要に応じて読み込まれるため、別個に読み込む必要はありません。

モジュールについての情報を表示するには、以下のコマンドを実行します。

-

```
~]$ modinfo bonding
```

コマンドオプションについては、**modprobe (8)** の **man** ページを参照してください。

7.4.2. チャンネルボンディングインターフェイスの作成

チャンネルボンディングインターフェイスを作成するには、**/etc/sysconfig/network-scripts/** ディレクトリーに **ifcfg-bond*N*** という名前のファイルを作成し、***N*** をそのインターフェイスの番号（例： **0**）に置き換えます。

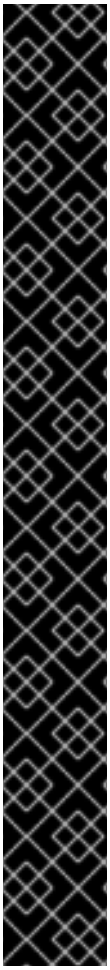
ファイルのコンテンツは、イーサネットインターフェイスなどボンディングされるインターフェイスのものであればどの設定ファイルでもそれをベースとすることができます。**DEVICE** ディレクティブは ボンディング***N***で、***N*** はインターフェイスの番号に置き換え、**TYPE=Bond** があることです。さらに、**BONDING_MASTER=yes** を設定します。

例7.1 ifcfg-bond0 インターフェイス設定ファイルの例

チャンネルボンディングインターフェイスの例は以下のようになります。

```
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 ディレクティブは、**NetworkManager** で接続プロファイルに名前を付けるのに役立ちます。**ONBOOT** は、起動時 (一般的にはデバイスの自動接続時) にプロファイルを起動するかどうかを示しています。



重要

ボンディングカーネルモジュールのパラメーターは、ifcfg-bond*N* インターフェイスファイルの BONDING_OPTS="*bonding parameters*" ディレクティブにスペース区切りリストとして指定する必要があります。ボンディングデバイスのオプションは、/etc/modprobe.d/*bonding*.conf または非推奨の /etc/modprobe.conf ファイルで指定し ない でください。

max_bonds パラメーターはインターフェイス固有ではないため、BONDING_OPTS ディレクティブで ifcfg-bond*N* ファイルを使用する場合は設定しないでください。このディレクティブにより、ネットワークスクリプトが必要に応じてボンドインターフェイスを作成するためです。

ボンディングモジュールの設定に関する指示およびアドバイスとボンディングパラメーターのリストについては、「チャンネルボンディングの使用」を参照してください。

NM_CONTROLLED="no" 設定が存在しない場合は、NetworkManager がこの設定ファイルの設定を上書きする可能性があることに注意してください。

7.4.3. ポートインターフェイスの作成

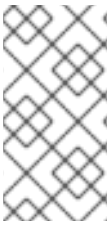
チャンネルボンディングインターフェイスはコントローラー(マスターとも呼ばれます)で、ボンディングされるインターフェイスはポート(スレーブ)と呼ばれます。チャネルボンディングインターフェイスを作成した後に、ボンディングされるネットワークインターフェイスを設定するには、そのポートの設定ファイルに MASTER および SLAVE ディレクティブを追加します。各ポートインターフェイスの設定ファイルは、ほぼ同一となる可能性があります。

例7.2 ポートインターフェイス設定ファイルの例

たとえば、2 つのイーサネットインターフェイス enp1s0 と enp2s0 がチャンネルボンディングされている場合、両方のインターフェイスの例を以下に示します。

```
DEVICE=device_name
NAME=bond0-slave
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
NM_CONTROLLED="no"
```

この例では、*device_name* をインターフェイス名で置換します。インターフェイスが **ONBOOT=yes** となっているプロファイルまたは設定ファイルが複数ある場合は、それらが相互に競合し、ボンドポートの代わりに単純な **TYPE=Ethernet** プロファイルがアクティブ化される可能性があることに注意してください。



注記

NM_CONTROLLED="no" 設定が存在しない場合は、**NetworkManager** がこの設定ファイルの設定を上書きする可能性があることに注意してください。

7.4.4. チャンネルボンディングのアクティブ化

ボンドをアクティブにするには、すべてのポートを有効にします。**root** で以下のコマンドを実行します。

```
~]# 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)
```

すでに 「**up**」 となっているインターフェイスのインターフェイスファイルを編集する場合は、以下のようにまず最初にそれらを **down** にします。

```
ifdown device_name
```

これが完了した後、すべてのポートを有効にすることで、ボンドが有効になります (「**down**」 に設定されていないことが前提)。

NetworkManager が変更を認識させるには、**root** で変更したすべてのインターフェイスに対してコマンドを実行します。

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

または、すべてのインターフェイスをリロードします。

```
~]# nmcli con reload
```

デフォルトの動作では、NetworkManager は変更を認識せず、古い設定データを引き続き使用します。これは、NetworkManager.conf ファイルの monitor-connection-files オプションで設定されます。詳細は、NetworkManager.conf (5) man ページを参照してください。

ボンドインターフェイスのステータスを表示するには、以下のコマンドを実行します。

```
~]# 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. 複数のボンド作成

Red Hat Enterprise Linux では、各ボンドに BONDING_OPTS ディレクティブを含むチャンネルボンディングインターフェイスが作成されます。この設定方法を使うと、複数のボンディングデバイスに異なる設定をすることが可能になります。複数のチャンネルボンディングインターフェイスを作成するには、以下の手順に従います。

- BONDING_OPTS ディレクティブで複数の ifcfg-bond*N* ファイルを作成します。このディレクティブにより、ネットワークスクリプトが必要に応じてボンドインターフェイスを作成します。
- ボンディングされるインターフェイス設定ファイルを作成するか、または既存のインターフェイス設定ファイルを編集し、SLAVE ディレクティブを追加します。
- ボンディングされるポートインターフェイスであるインターフェイスを MASTER ディレクティブでチャネルボンディングインターフェイスに割り当てます。

例7.3 複数の ifcfg-bondN インターフェイス設定ファイルの例

以下は、チャンネルボンディングインターフェイス設定ファイルの例です。

```
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"
```

この例では、*N* をインターフェイスの番号に置き換えます。たとえば、2 つのボンディングを作成するには、ifcfg-bond0 と ifcfg-bond1 の設定ファイルを適切な IP アドレスで作成します。

例7.2「ポートインターフェイス設定ファイルの例」 に従ってボンディングされるインターフェイスを作成し、MASTER=bond*N* ディレクティブを使って必要に応じてボンドインターフェイスに割り当てます。たとえば、上記の例から、ボンドごとに 2 つのインターフェイスが必要な場合は、2 つのボンドで 4 つのインターフェイス設定ファイルを作成し、最初の 2 つを MASTER=bond*0* を使用して割り当て、次の 2 つは MASTER=bond*1* を使用して割り当てます。

7.5. 冗長性についてネットワーク設定ボンディングの確認

ネットワークの冗長性は、特定のシステムの障害を防止または回復するために、バックアップ目的でデバイスが使用される場合のプロセスです。次の手順では、冗長性でボンディングのネットワーク設定を確認する方法を説明します。

手順

1. ボンドインターフェイスから、宛先の IP を ping します。以下に例を示します。

```
~]# ping -I bond0 DSTADDR
```

2. どのインターフェイスが アクティブ モードであるかを表示します。

```
~]# cat /sys/class/net/bond0/bonding/active_slave
enp1s0
```

enp1s0 は、アクティブな ポートインターフェイスです。

3. アクティブな ポートインターフェイスをダウンに設定します。

```
~]# ip link set enp1s0 down
```

4. バックアップ インターフェイスが起動しているかどうかを確認します。

```
~]# cat /sys/class/net/bond0/bonding/active_slave
enp2s0
```

enp2s0 が アクティブな ポートインターフェイスになりました。

5. ボンドインターフェイスから、宛先の IP を ping できるかどうかを確認します。

```
~]# ping -I bond0 DSTADDR
```

7.6. スイッチにおけるボンディングモードおよび必要な設定の概要

以下の表は、ボンディングモードに応じて、アップストリームのスイッチに適用する必要がある設定を示しています。

表7.1 ボンディングモードに依存するスイッチ設定

ボンディングモード	スイッチの設定

ボンディングモード	スイッチの設定
0 - balance-rr	(LACP がネゴシエートされたものではなく) 静的な Etherchannel を有効にする必要があります。
1 - active-backup	自動ポートが必要です。
2 - balance-xor	(LACP がネゴシエートされたものではなく) 静的な Etherchannel を有効にする必要があります。
3 - broadcast	(LACP がネゴシエートされたものではなく) 静的な Etherchannel を有効にする必要があります。
4 - 802.3ad	LACP がネゴシエートされた Etherchannel が有効になっている必要があります。
5 - balance-tlb	自動ポートが必要です。
6 - balance-alb	自動ポートが必要です。

スイッチの設定は、スイッチのドキュメントを参照してください。

7.7. チャンネルボンディングの使用

パフォーマンスを強化するには、利用可能なモジュールオプションを調節して、最適な組み合わせを確認します。特に、miimon パラメーターまたは arp_interval パラメーターおよび arp_ip_target パラメーターに注意してください。利用可能なオプションリストと使用しているボンディングされたインターフェイスに最適なオプションを迅速に決定する方法については、「ボンディングモジュールのディレクティブ」を参照してください。

7.7.1. ボンディングモジュールのディレクティブ

ボンディングインターフェイス設定ファイル(ifcfg-bond0 など)の BONDING_OPTS="*bonding parameters*" ディレクティブに追加する前に、ボンディングされたインターフェイスでどのチャンネルボンディングモジュールパラメーター が最適かをテストすることが推奨されます。ボンディングされたインターフェイスのパラメーターは、sysfs ファイルシステムのファイルを操作することで、ボンディングモジュールをアンロード（およびリロード）せずに設定できます。

sysfs は、カーネルオブジェクトをディレクトリー、ファイル、シンボリックリンクとして表現する仮想ファイルシステムです。sysfs は、カーネルオブジェクトに関する情報をクエリーするのに使用でき、通常のファイルシステムコマンドを使用してこれらのオブジェクトを操作することもできます。sysfs 仮想ファイルシステムは、/sys/ ディレクトリーにマウントされます。すべてのボンディングインターフェイスは、/sys/class/net/ ディレクトリー配下のファイルと対話して操作することで動的に設定できます。

ボンディングインターフェイスに最適なパラメーターを決定するには、「チャンネルボンディングインターフェイスの作成」 の手順に従って、ifcfg-bond0 などのチャンネルボンディングインターフェイスファイルを作成します。bond0 にボンディングされる各インターフェイスの設定ファイルに SLAVE=yes および MASTER=bond0 ディレクティブを挿入します。これが完了すると、パラメーターの確認に進むことができます。

まず、root で ifup bond*N*を実行して、作成したボンディング を開きます。

```
~]# ifup bond0
```

ifcfg-bond0 ボンディングインターフェイスファイルを正しく作成した場合は、root で ip link show を実行した場合の出力に bond0 が表示されます。

```
~]# 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
```

アップになっていないボンドも含めてすべての既存のボンドを表示するには、以下を実行します。

```
~]$ cat /sys/class/net/bonding_masters
bond0
```

/sys/class/net/bond*N*/bonding/ ディレクトリーにあるファイルを操作すると、各ボンドを個別に設定できます。まず、設定するボンドをダウンにします。

```
~]# ifdown bond0
```

たとえば、bond0 で 1 秒間隔で MII 監視を有効にするには、root で以下のコマンドを実行します。

```
~]# echo 1000 > /sys/class/net/bond0/bonding/miimon
```

bond0 を *balance-alb* モードに設定するには、以下のいずれかを実行します。

```
~]# echo 6 > /sys/class/net/bond0/bonding/mode
```

またはモード名を使用します。

```
~]# echo balance-alb > /sys/class/net/bond0/bonding/mode
```

該当するボンディングのオプションを設定したら、ifup bond*N* を実行して、これを起動してテストできます。オプションを変更する場合は、インターフェイスをダウンさせ、sysfs を使用してパラメーターを変更し、これを元に戻して再テストします。

ボンディングに最適なパラメーターセットを決定したら、設定するボンディングインターフェイスの /etc/sysconfig/network-scripts/ifcfg-bond*N* ファイルの *BONDING_OPTS=* ディレクティブに、これらのパラメーターをスペース区切りリストとして追加します。ボンディングが開始されるたびに(*ONBOOT=yes* ディレクティブが設定されている場合はブートシーケンス中のシステムなど)、 *BONDING_OPTS* で指定したボンディングオプションがそのボンディングに対して有効になります。

以下では、多くの一般的なチャンネルボンディングパラメーターの名前とそれらの機能の詳細をリスト表示しています。詳細は、modinfo bonding の出力にある各 parm の簡単な説明を参照してください。詳細は、https://www.kernel.org/doc/Documentation/networking/bonding.txtを参照してください。

ボンディングインターフェイスパラメーター

ad_select=*value*

使用する 802.3ad アグリゲーションの選択論理を指定します。以下の値が使用できます。

- stable または 0 - デフォルト設定。アクティブなアグリゲーターは、最大のアグリゲーション帯域幅によって選択されます。アクティブなアグリゲーターの再選択は、すべてのポートがダウンとなるか、ポートがなくなった時にのみ行われます。
- bandwidth または 1: アクティブなアグリゲーターは、最大集約帯域幅によって選択されます。アクティブなアグリゲーターの再選択は以下の場合に行われます。
  - ボンドにポートが追加される、またはボンドからポートが削除される。

- ポートのリンク状態が変更される。
- ポートの 802.3ad 関連状態が変更される。
- ボンドの管理状態が有効になる。

- count または 2: アクティブなアグリゲーターは、ポートの最大数で選択されます。上記の 帯域幅 設定については、再選択が行われます。

帯域幅 と カウント 選択ポリシーにより、アクティブなアグリゲーターの部分的な障害が発生した場合に 802.3ad アグリゲーションのフェイルオーバーが可能になります。これにより、帯域幅またはポート数の可用性が最も高いアグリゲーターが常にアクティブになります。

arp_interval=*time_in_milliseconds*

ARP 監視が発生する頻度を指定します（ミリ秒単位）。

重要

arp_interval および arp_ip_target パラメーターの両方を指定するか、miimon パラメーターを指定する必要があります。指定されないと、リンクが失敗した場合にネットワークパフォーマンスが低下する恐れがあります。

mode =0 または mode = 2 (2 つの負荷分散モード)でこの設定を使用する場合は、NIC 全体にパケットを均等に分散するようにネットワークスイッチを設定する必要があります。この方法の詳細については、https://www.kernel.org/doc/Documentation/networking/bonding.txt を参照してください。

デフォルトでは、値は 0 に設定され、無効にされます。

arp_ip_target=*ip_address*[,*ip_address_2*,...*ip_address_16*]

arp_interval パラメーターが有効な場合に ARP 要求のターゲット IP アドレスを指定します。最大 16 の IP アドレスはコンマ区切りリストで指定できます。

arp_validate=*value*

ARP プローブのソース/ディストリビューションを検証します。デフォルトは none です。その他の有効な値は、active、backup、および all です。

downdelay=*time_in_milliseconds*

リンクを無効にする前に、リンクの失敗後に待機する時間を指定します (ミリ秒単位)。値は、miimon パラメーターで指定された値の倍数である必要があります。デフォルトでは、値は 0 に設定され、無効にされます。

fail_over_mac=*value*

アクティブ-バックアップモードが、割り当て時にすべてのポートを同一 MAC アドレスに設定する (従来の動作) か、有効な場合は、選択されたポリシーに従って、ボンドの MAC アドレスの特別な処理を実行するかを指定します。可能な値は次のとおりです。

- none または 0 - デフォルト設定。この設定により fail_over_mac が無効になり、割り当て時にボンディングがアクティブ-バックアップボンドのすべてのポートを同じ MAC アドレスに設定します。

- active または 1 - 「active」 fail_over_mac ポリシーは、ボンドの MAC アドレスは常に現在アクティブなポートの MAC アドレスである必要があることを示します。ポートの MAC アドレスは変更されませんが、代わりにフェイルオーバー中にボンドの MAC アドレスが変更されます。

  このポリシーは、MAC アドレスを変更できないデバイスや、(ARP 監視を妨害する) 自身のソース MAC を持つ着信ブロードキャストを拒否するデバイスに便利なものです。このポリシーのマイナス面は、ネットワーク上のすべてのデバイスが余計な ARP によって更新される必要があるという点です。通常の方法では、スイッチが着信トラフィックを嗅ぎ付けて ARP テーブルを更新します。余計な ARP が失われると、通信が中断される可能性があります。

  このポリシーを MII モニターと合わせて使用すると、実際に送受信可能になる前にリンクを有効にするデバイスが特に余計な ARP を失いやすくなります。また、適切な updelay 設定が必要になる可能性があります。

- follow または 2 - 「follow」 fail_over_mac ポリシーにより、ボンドの MAC アドレスが正常に選択されます（通常、最初のポートの MAC アドレスがボンディングに追加されます）。ただし、2 番目以降のポートはこの MAC アドレスに設定されず、バックアップのロールを果たします。つまり、ポートはフェイルオーバー時にボンドの MAC アドレスでプログラミングされます (また、それまでアクティブだったポートが新たにアクティブになったポートの MAC アドレスを受け取ります)。

  このポリシーは、複数ポートが同一 MAC アドレスでプログラミングされる際に、混乱したりパフォーマンスペナルティーを受けるマルチポートデバイスに便利なものです。

lacp_rate=*value*

リンクパートナーが 802.3ad モードで LACPDU パケットを送信するレートを指定します。以下の値が使用できます。

- slow または 0 - デフォルト設定。パートナーが 30 秒ごとに LACPDU を送信するよう指定します。
- fast または 1 - パートナーが 1 秒ごとに LACPDU を送信するように指定します。

miimon=*time_in_milliseconds*

MII リンク監視が発生する頻度を指定します (ミリ秒単位)。MII は NIC がアクティブであることを検証するために使用されるため、これは高可用性が必要な場合に役立ちます。特定の NIC のドライバーが MII ツールに対応していることを確認するには、root で以下のコマンドを入力します。

```
~]# ethtool interface_name | grep "Link detected:"
```

このコマンドでは、***interface_name*** を、ボンドインターフェイスではなく、enp1s0 などのデバイスインターフェイスの名前に置き換えます。MII が対応している場合は、コマンドは以下を返します。

```
Link detected: yes
```

高可用性のためにボンディングされたインターフェイスを使用する場合、各 NIC のモジュール

は MII に対応していなければなりません。値を 0（デフォルト）に設定すると、この機能はオフになります。この設定を設定する際に、このパラメーターのスタート地点は 100 です。

重要

arp_interval および arp_ip_target パラメーターの両方を指定するか、miimon パラメーターを指定する必要があります。指定されないと、リンクが失敗した場合にネットワークパフォーマンスが低下する恐れがあります。

mode=*value*

ボンディングポリシーの指定が可能になります。*value* は、次のいずれかになります。

- balance-rr または 0 - 耐障害性および負荷分散のラウンドロビンポリシーを設定します。利用可能な最初のインターフェイスからそれぞれのボンディングされたポートインターフェイスで送受信が順次行われます。

- active-backup または 1 - 耐障害性のためのアクティブバックアップポリシーを設定します。利用可能になった最初のボンディングされたポートインターフェイスにより送受信が行われます。別のボンディングされたポートインターフェイスは、アクティブなボンディングされたポートインターフェイスに障害が発生した場合にのみ使用されます。

- balance-xor または 2 - 送信は、選択したハッシュポリシーに基づきます。デフォルトでは、送信元と宛先の MAC アドレスの XOR にポートインターフェイス数のモジュロを掛けてハッシュを導出します。このモードでは、宛先が特定のピアになっているトラフィックは常に同一インターフェイスで送信されます。宛先は MAC アドレスで決められるので、この方法は同一リンクまたはローカルネットワーク上にあるピアが宛先のトラフィックに最適なものです。トラフィックが単一ルーターを通過する必要がある場合は、このトラフィックバランスのモードは最適ではなくなります。

- broadcast または 3 - 耐障害性を確保するためにブロードキャストポリシーを設定します。すべての送信は、すべてのポートインターフェイスで行われます。

- 802.3ad または 4 - IEEE 802.3ad 動的リンクアグリゲーションポリシーを設定します。同一の速度とデュプレックス設定を共有するアグリゲーショングループを作成します。アクティブなアグリゲーターのすべてのポートで送受信を行います。802.3ad に対応するスイッチが必要です。

- balance-tlb または 5 - 耐障害性と負荷分散のために TLB (Transmit Load

Balancing)ポリシーを設定します。発信トラフィックは、各ポートインターフェイスの現在の負荷に従って分散されます。受信トラフィックは、現在のポートにより受信されます。受信ポートに障害が発生すると、障害が発生したポートの MAC アドレスを別のポートが引き継ぎます。このモードは、カーネルボンディングモジュールが認識しているローカルアドレスにのみ、適したものになります。このため、仮想マシンのブリッジの背後では使用できません。

- balance-alb または 6 - 耐障害性と負荷分散のための Adaptive Load Balancing (ALB)ポリシーを設定します。IPv4 トラフィック用の送受信負荷分散が含まれます。ARP ネゴシエーションにより、受信負荷分散を実現します。このモードは、カーネルボンディングモジュールが認識しているローカルアドレスにのみ、適したものになります。このため、仮想マシンのブリッジの背後では使用できません。

アップストリームスイッチで必要な設定の詳細は、「スイッチにおけるボンディングモードおよび必要な設定の概要」を参照してください。

primary=*interface_name*

プライマリーデバイスのインターフェイス名（例： enp1s0 ）を指定します。プライマリー デバイスは、最初に使用されるボンディングインターフェイスであり、失敗しない限り破棄されません。この設定が特に役立つのは、ボンディングインターフェイスの NIC の 1 つが高速なため、大規模な負荷に対応できる場合です。

この設定は、ボンディングインターフェイスが active-backup モードである場合にのみ有効です。詳細は、https://www.kernel.org/doc/Documentation/networking/bonding.txt を参照してください。

primary_reselect=*value*

プライマリーポートに対して再選択ポリシーを指定します。これは、アクティブなポートの障害やプライマリーポートの回復が発生した場合に、どのようにプライマリーポートが選択されてアクティブなポートになるかという点に影響します。このパラメーターは、プライマリポートと他のポートとの間で何度も切り替えが発生しないようにするためのものです。可能な値は次のとおりです。

- always または 0 （デフォルト）：プライマリーポートは、復旧するたびにアクティブなポートになります。

- better または 1 - プライマリーポートの速度とデュプレックスが、現在のアクティブなポートの速度とデュプレックスよりも良い場合、プライマリーポートは有効になるとアク

ティブなポートになります。

- failure または 2 - 現在のアクティブなポートが失敗し、プライマリーポートが稼働している場合に限り、プライマリーポートはアクティブなポートになります。

primary_reselect 設定は無視されます。

- アクティブなポートがない場合は、回復する最初のポートがアクティブなポートになります。

- プライマリーポートがボンドに初めて割り当てられると、そのポートが必ずアクティブなポートになります。

sysfs を介して primary_reselect ポリシーを変更すると、新しいポリシーに従って直ちに最適なアクティブなポートが選択されます。これにより、状況によってはアクティブなポートに変更が生じる場合があります。

resend_igmp=*range*

フェイルオーバーイベント後に発行される IGMP メンバーシップレポートの数を指定します。1 つのメンバーシップレポート がフェイルオーバーの直後に発行され、以降のパケットは 200ms (ミリ秒) の間隔で送信されます。

有効な範囲は 0 から 255 です。デフォルト値は 1 です。値が 0 の場合は、フェイルオーバーイベントへの応答で IGMP メンバーシップレポートが発行されなくなります。

このオプションは、フェイルオーバーが IGMP トラフィックをあるポートから別のポートに切り替えることができる balance-rr (mode 0)、active-backup (mode 1)、balance-tlb (mode 5)および balance-alb (mode 6)のボンディングモードに役立ちます。したがって、新しく選択されたポートを介して着信 IGMP トラフィックをスイッチで転送するには、新しい IGMP レポートを発行する必要があります。

updelay=*time_in_milliseconds*

リンクを有効にする前の待機時間を指定します (ミリ秒単位)。値は、miimon パラメーターで指定された値の倍数である必要があります。デフォルトでは、値は 0 に設定され、無効にされま

す。

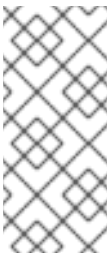
use_carrier=*number*

リンク状態を決定するために、miimon が MII/ETHTOOL ioctls または netif_carrier_ok() を使用するかどうかを指定します。netif_carrier_ok() 関数は、デバイスドライバーに依存して netif_carrier_*on/off* でその状態を維持します。ほとんどのデバイスドライバーがこの機能に対応しています。

MII/ETHTOOL ioctls ツールは、カーネル内の非推奨の呼び出しシーケンスを活用します。ただし、デバイスドライバーが netif_carrier_*on/off* に対応していない場合も設定可能です。

有効な値は以下のとおりです。

- 1 : デフォルト設定。netif_carrier_ok() の使用を有効にします。
- 0 : MII/ETHTOOL ioctls の使用を有効にします。



注記

リンクがアップしているべきではない時にリンクがアップであると主張する場合は、ネットワークデバイスドライバーが netif_carrier_*on/off* に対応していない可能性があります。

xmit_hash_policy=*value*

balance-xor および 802.3ad モードでポートを選択するために使用される送信ハッシュポリシーを選択します。可能な値は次のとおりです。

- 0 または layer2 - デフォルト設定。このパラメーターは、ハードウェア MAC アドレスの XOR を使用してハッシュを生成します。使用する式は以下のとおりです。

  (*source_MAC_address* **XOR** *destination_MAC*) **MODULO** *slave_count*

  このアルゴリズムは、すべてのトラフィックを同じポートの特定のネットワークピアに割り振り、802.3ad に対応します。

- 1 または layer3+4 - 上層プロトコル情報（利用可能な場合）を使用してハッシュを生成します。これにより、特定のネットワークピアへのトラフィックが複数のポートに及ぶようにできますが、単一の接続では複数のポートに及びません。

  断片化された TCP および UDP パケットに使用される公式は、以下のとおりです:

  ```
  ((source_port XOR dest_port) XOR
    ((source_IP XOR dest_IP) AND 0xffff)
      MODULO slave_count
  ```

  断片化された TCP または UDP パケットおよびその他の IP プロトコルトラフィックの場合、送信元および宛先ポート情報は省略されます。非IP トラフィックの場合、式は layer2 送信ハッシュポリシーと同じです。

  このポリシーの目的は、特に PFC2 付きの Cisco スイッチや Foundry および IBM 製品など一部のスイッチの動作を真似ることです。

  このポリシーで使用されるアルゴリズムは、802.3ad に対応していません。

- 2 または layer2+3 - layer2 および layer3 プロトコル情報の組み合わせを使用して、ハッシュを生成します。

  ハードウェア MAC アドレスと IP アドレスの XOR を使用してハッシュを生成します。式は以下のとおりです。

  ```
  (((source_IP XOR dest_IP) AND 0xffff) XOR
    ( source_MAC XOR destination_MAC ))
      MODULO slave_count
  ```

  このアルゴリズムは、すべてのトラフィックを同じポートの特定のネットワークピアに割り振ります。非IP トラフィックの場合、式は layer2 送信ハッシュポリシーと同じになります。

  このポリシーの目的は、特に layer3 ゲートウェイデバイスが大半の宛先に到達する必要がある環境において、layer2 単独の場合より分散されたトラフィックを提供することです。

このアルゴリズムは、802.3ad に対応しています。

7.8. GUI を使用したボンディング接続の作成

GNOME control-center ユーティリティーを使用して、NetworkManager に、2 つ以上の有線または InfiniBand 接続からボンドを作成するように指示できます。最初にボンディングする接続を作成する必要はありません。それは、ボンディングを設定するためのプロセスで設定できます。設定プロセスを完了するには、利用可能なインターフェイスの MAC アドレスが必要です。

7.8.1. ボンド接続の確立

手順7.1 nm-connection-editor を使用して新規ボンド接続を追加する

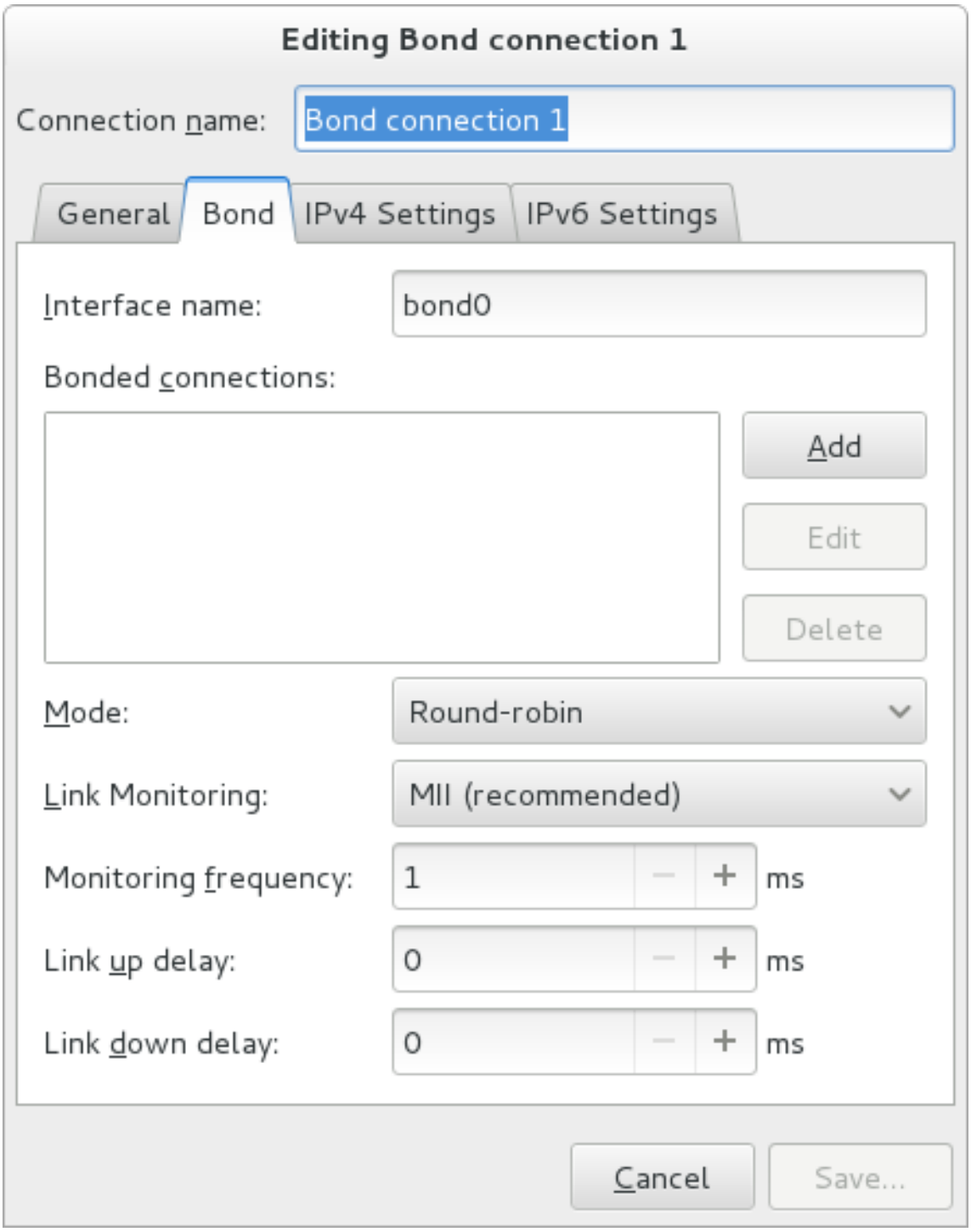
新規ボンド接続を作成するには、以下のステップに従います。

1. 端末に nm-connection-editor と入力します。

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

2. Add ボタンをクリックします。Choose a Connection Type ウィンドウが表示されます。Bond を選択し、Create を クリックします。ボンディング接続 *1*の編集 ウィンドウが表示されます。

図7.6 NetworkManager グラフィカルユーザーインターフェイスの Bond 追加メニュー



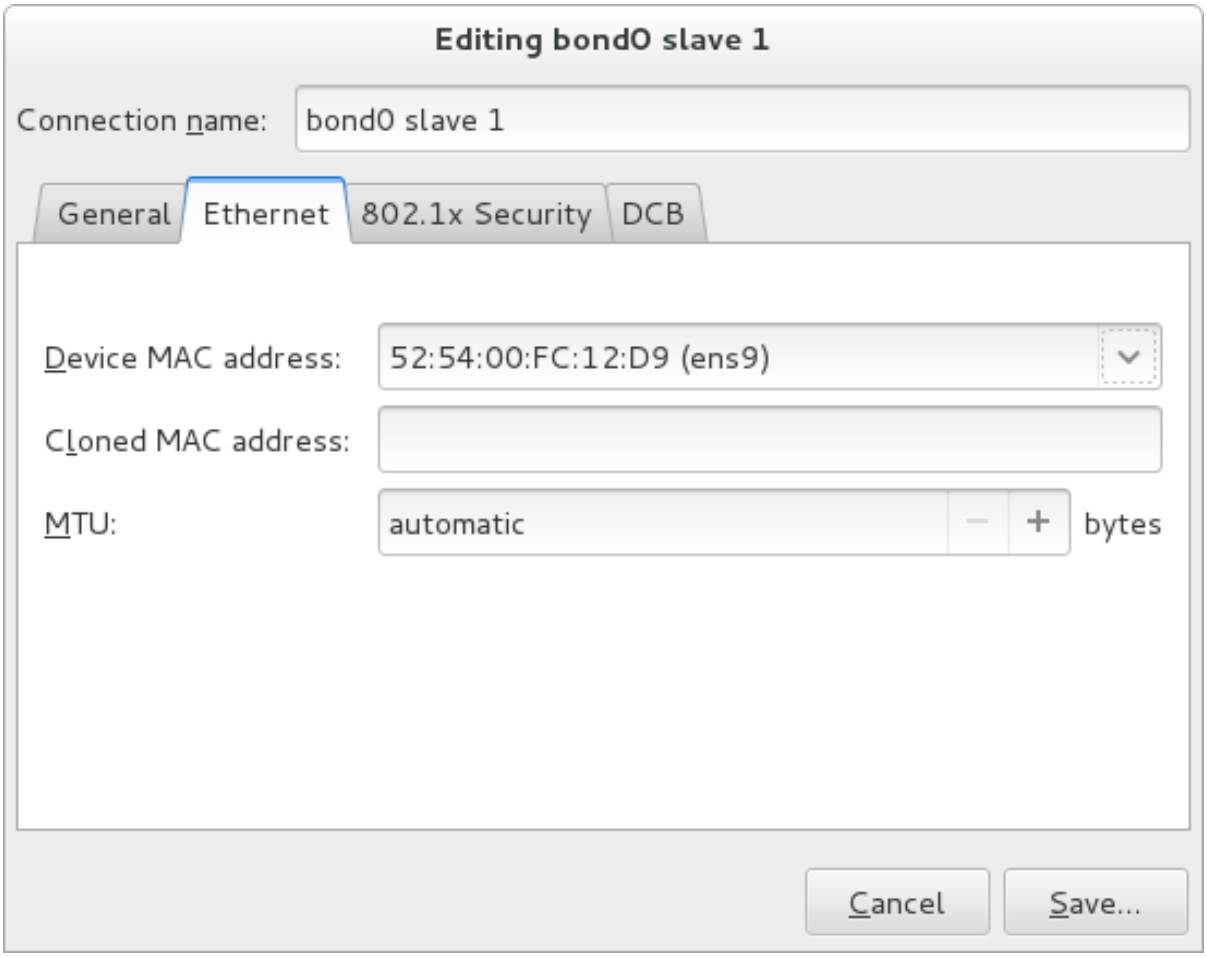
[D]

3. **Bond タブで 追加 をクリックし、ボンド接続で使用するインターフェイスのタイプを選択します。Create ボタンをクリックします。ポートタイプを選択するダイアログが表示されるのは、最初のポートを作成する時のみです。その後は、すべてのポートに同じタイプが自動的に使われます。**

4. **bond0 スレーブ 1 の編集 ウィンドウが表示されます。デバイス MAC アドレス ドロップダウンメニューを使用して、ボンディングするインターフェイスの MAC アドレスを選択します。最初のポートの MAC アドレスがボンドインターフェイス用の MAC アドレスとして使用さ**

れます。必要な場合は、ボンドの MAC アドレスとして使用するクローンした MAC アドレスを入力します。Save ボタンをクリックします。

図7.7 NetworkManager グラフィカルユーザーインターフェイスのボンド接続追加メニュー



[D]

5. ボンディングされたポートの名前が Bond connections ウィンドウに表示されます。Add ボタンをクリックして、さらにポート接続を追加します。

6. 設定を確認してから Save ボタンをクリックします。

7. 以下の「Bond タブの設定」を参照して、ボンド固有の設定を編集します。

手順7.2 既存のボンド接続を編集する

既存のボンド接続を編集するには以下のステップに従います。

1. 端末に nm-connection-editor と入力します。

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

2. 編集する接続を選択し、Edit ボタンをクリックします。

3. General タブを選択します。

4. 接続名、自動接続の動作、および可用性のセッティングを設定します。

   編集 ダイアログの 5 つの設定は、すべての接続の種類に共通です。全般 タブ：

   - 接続名: ネットワーク接続のわかりやすい名前を入力します。この名前は、Network ウィンドウのメニューでこの接続を一覧表示するために使用されます。
   - Automatically connect to this network when it is available - このボックスを選択すると、NetworkManager が利用可能なときにこの接続に自動接続します。詳細は、「control-center を使用した既存の接続の編集」を参照してください。
   - All users can connect to this network - このボックスを選択すると、システム上のすべてのユーザーが 利用できる接続が作成されます。この設定を変更するには、root 権限が必要になる場合があります。詳細は、「GUI を使用したシステム全体およびプライベート接続プロファイルの管理」を参照してください。
   - Automatically connect to VPN when using this connection - このボックスを選択すると、NetworkManager が利用可能なときに VPN 接続に自動接続します。ドロップダウンメニューから VPN を選択します。
   - ファイアウォールゾーン - ドロップダウンメニューからファイアウォールゾーンを選択します。ファイアウォールゾーンに関する詳細情報は、『Red Hat Enterprise Linux 7 セキュリティーガイド』を参照してください。

5. 以下の「Bond タブの設定」を参照して、ボンド固有の設定を編集します。

新規 (または修正した) 接続を保存して他の設定を行う

ボンド接続の編集が終わったら、保存 ボタンをクリックしてカスタマイズした設定を保存します。

そして、以下のいずれかの設定をします。

- IPv4 の設定は、IPv4 のセッティング タブをクリックしてに進みます。 「IPv4 設定の設定」

  または

- IPv6 の設定は、IPv6 のセッティング タブをクリックして、「IPv6 セッティングの設定」に進みます。

7.8.1.1. Bond タブの設定

新規のボンド接続をすでに追加している場合（手順は 手順7.1「nm-connection-editor を使用して新規ボンド接続を追加する」 を参照）、Bond タブを編集して、負荷分散モードと、ポート接続の失敗を検出するのに使用するリンク監視のタイプを設定できます。

Mode

ボンドを設定するポート接続でのトラフィック共有に使われるモード。デフォルトは ラウンドロビン です。802.3ad などの他のロード共有モードは、ドロップダウンリストで選択できます。

リンク監視

ネットワークトラフィックを伝送するポートの能力を監視する方法。

以下の負荷分散モードは、モード ドロップダウンリストから選択できます。

ラウンドロビン

耐障害性とロードバランシングにラウンドロビンポリシーを設定します。利用可能な最初のインターフェイスからそれぞれのボンディングされたポートインターフェイスで送受信が順次行われます。このモードは、仮想マシンのブリッジの背後では追加のスイッチ設定がないと機能しない可能性があります。

アクティブバックアップ

耐障害性のためアクティブなバックアップポリシーを設定します。利用可能になった最初のボンディングされたポートインターフェイスにより送受信が行われます。別のボンディングされたポートインターフェイスは、アクティブなボンディングされたポートインターフェイスに障害が発生した場合にのみ使用されます。これは、InfiniBand デバイスのボンドで利用可能な唯一のモードです。

XOR

XOR (排他的理論和) を設定します。送受信は選択されたハッシュポリシーに基づいて行われます。デフォルトでは、送信元と宛先の MAC アドレスの XOR にポートインターフェイス数のモジュロを掛けてハッシュを導出します。このモードでは、宛先が特定のピアになっているトラフィックは常に同一インターフェイスで送信されます。宛先は MAC アドレスで決められるので、この方法は同一リンクまたはローカルネットワーク上にあるピアが宛先のトラフィックに最適なものです。トラフィックが単一ルーターを通過する必要がある場合は、このトラフィックバランスのモードは最適ではなくなります。

ブロードキャスト

耐障害性にブロードキャストポリシーを設定します。すべての送信は、すべてのポートインターフェイスで行われます。このモードは、仮想マシンのブリッジの背後では追加のスイッチ設定がないと機能しない可能性があります。

802.3ad

IEEE 802.3ad 動的リンクアグリゲーションポリシーを設定します。同一の速度とデュプレックス設定を共有するアグリゲーショングループを作成します。アクティブなアグリゲーターのすべてのポートで送受信を行います。802.3ad に準拠するネットワークスイッチが必要です。

適応送信のロードバランシング

耐障害性とロードバランシングのための適応型送信ロードバランシング (TLB) ポリシーを設定します。発信トラフィックは、各ポートインターフェイスの現在の負荷に従って分散されます。受信トラフィックは、現在のポートにより受信されます。受信ポートに障害が発生すると、障害が発生したポートの MAC アドレスを別のポートが引き継ぎます。このモードは、カーネルボンディングモジュールが認識しているローカルアドレスにのみ、適したものになります。このため、仮想マシンのブリッジの背後では使用できません。

適応ロードバランス

耐障害性とロードバランシングに適応型ロードバランシング (ALB) ポリシーを設定します。IPv4 トラフィック用の送受信負荷分散が含まれます。ARP ネゴシエーションにより、受信負荷分散を実現します。このモードは、カーネルボンディングモジュールが認識しているローカルアドレスにのみ、適したものになります。このため、仮想マシンのブリッジの背後では使用できません。

以下のリンク監視のタイプは、Link Monitoring ドロップダウンリストから選択できます。ボンディングされたインターフェイスでどのチャンネルボンディングのモジュールパラメーターが最適な動作をするかテストするとよいでしょう。

MII (Media Independent Interface)

インターフェイスのキャリア波の状態を監視します。これには、ドライバーをクエリーするか、MII レジスターに直接クエリーするか、ethtool を使用してデバイスにクエリーを実行します。利用可能な 3 つのオプションは以下のとおりです。

監視周期

ドライバーもしくは MII レジスターへのクエリーの間隔時間 (ミリ秒単位)

接続遅延

up とレポートされたリンクの使用を試みるまでの待機時間 (ミリ秒単位)。この遅延は、リンクが 「up」 として報告される直後に、一部の Gratuitous ARP 要求が期間内に失われる場合に使用できます。これが発生するのは、たとえばスイッチ初期化などの間です。

接断遅延

これまでアクティブだったリンクが「down」とレポートされた際に、別のリンクに変更するまでの待ち時間 (ミリ秒単位)。アタッチされたスイッチがバックアップモードに変更するまで比較的長い時間がかかる場合に、この遅延は使用できます。

ARP

アドレス解決プロトコル(ARP)は、1 つ以上のピアをプローブし、リンク層接続がどのように機能しているかを判断するために使用されます。これは、送信開始時間および最終受信時間を提供しているデバイスドライバーに依存しています。

以下の 2 つのオプションがあります。

監視周期

ARP リクエストを送信する間隔（ミリ秒単位）。

ARP ターゲット

ARP 要求を送信する IP アドレスのコンマ区切りリスト。

7.9. 関連情報

インストールされているドキュメント

- nmcli (1) man ページ - NetworkManager のコマンドラインツールを説明しています。
- nmcli-examples (5) man ページ - nmcli コマンドの例を説明します。
- nm-settings (5) man ページ - NetworkManager 接続の設定およびパラメーターが説明されています。

オンラインドキュメント

『Red Hat Enterprise Linux システム管理者のガイド』

カーネルモジュール機能の使用方法を説明しています。

https://access.redhat.com/site/node/28421/Configuring_VLAN_devices_over_a_bonded_interface

ボンディングされたインターフェイスでの VLAN デバイスの設定に関する Red Hat ナレッジベースの記事です。

第8章 ネットワークチーミングの設定

8.1. ネットワークチーミングについて

ネットワークリンクを結合させてより高いスループットの論理リンクや冗長性を提供する手段には、*チャンネルボンディング*、*イーサネットボンディング*、*ポートトランキング*、*チャンネルチーミング*、*NIC チーミング*、または *リンクアグリゲーション* など多くの名前が付けられています。当初 Linux カーネルで実装されたこの概念は、広く ボンディング と呼ばれます。この概念の新しい実装の呼び方として、*ネットワークチーミング* という用語が選択されました。既存のボンディングドライバーは影響を受けません。ネットワークチーミングは Red Hat Enterprise Linux 7 のボンディングの代替メカニズムとして提供されるもので、これに置き換わるものではありません。

注記

モード 4 Link Aggregation Control Protocol (LACP) チーミングモードについては、リンクを集約するようにスイッチを設定する必要があります。詳細は https://www.kernel.org/doc/Documentation/networking/bonding.txt を参照してください。

ネットワークチーミング (またはチーム) は、高速でパケットフローを処理する小型のカーネルドライバーおよびその他すべてのユーザースペースタスクを実行する様々なユーザースペースのアプリケーションを提供するという、異なる方法でこの概念を実装するように設計されています。このドライバーには 「Team Netlink API」 と呼ばれる *アプリケーションプログラミングインターフェイス* (API) が備わっており、これが Netlink 通信を実装します。ユーザースペースのアプリケーションは、この API を使用してドライバーと通信できます。「lib」と呼ばれるライブラリーは、Team Netlink 通信と RT Netlink メッセージのユーザースペースラッピングを行うために提供されています。libteam ライブラリーを使用するアプリケーションデーモン teamd も利用可能です。teamd の 1 つのインスタンスが、チームドライバーの 1 つのインスタンスを制御できます。このデーモンは、「ランナー」と呼ばれる新たなコードを使用することで、ラウンドロビンなどの負荷分散やアクティブバックアップ論理を実装します。このようにコードを分離することで、ネットワークチーミングの実装は負荷分散および冗長性要件に対して容易に拡張可能およびスケーラブルなソリューションを提供します。たとえば、カスタムランナーは、teamd を介して新しいロジックを実装するために比較的簡単に記述でき、 teamd も任意であるため、libteam を使用するために独自のアプリケーションを作成することができます。

teamdctl ユーティリティーは、D-bus を使用して teamd の実行中のインスタンスを制御するために使用できます。teamdctl は、teamd D-Bus API に関する D-Bus ラッパーを提供します。デフォルトでは、teamd は Unix Domain Sockets を使用してリッスンし、通信しますが、引き続き D-Bus を監視します。これは、teamd が D-Bus が存在しない環境またはまだ読み込まれていない環境で使用できるようにするためです。たとえば、teamd リンクで起動すると、D-Bus がまだ読み込まれません。ランタイム時に teamdctl ユーティリティーを使用すると、設定の読み取り、リンク監視の状態の確認と変更、ポートの状態の確認と変更、ポートの追加と削除、アクティブ状態とバックアップ状態間のポート変更を行うことができます。

Team Netlink API は、Netlink メッセージを使ってユーザースペースのアプリケーションと通信します。libteam ユーザー空間ライブラリーは API と直接対話しませんが、libnl または teamnl を使用し

てドライバー API と対話します。

要約すると、カーネルで実行中のチームドライバーのインスタンスは、直接設定、制御されることはありません。すべての設定は、teamd アプリケーションなどのユーザー空間アプリケーションを使用して行われます。アプリケーションはその後、カーネルドライバーのパートに適切に指示します。

注記

ネットワークチーミングのコンテキストでは、ポート という用語は スレーブ としても知られています。teamd を 直接使用する場合は、port が推奨されます。一方、NetworkManager を使用してチームを作成するインターフェイスを参照する場合は、スレーブ が使用されます。

8.2. コントローラーおよびポートインターフェイスのデフォルト動作の理解

NetworkManager デーモンを使用してチーミングされたポートインターフェイスを制御する場合、特に障害検索時には、以下の点に留意してください。

1. コントローラーインターフェイスを起動しても、ポートインターフェイスは自動的に起動しない。

2. ポートインターフェイスを起動すると、コントローラーインターフェイスは毎回、起動する。

3. コントローラーインターフェイスを停止すると、ポートインターフェイスも停止する。

4. ポートのないコントローラーは静的 IP 接続を開始できる。

5. コントローラーにポートがない場合は、DHCP 接続の開始時にポートを待機します。

6. DHCP 接続でポートを待機中のコントローラーは、キャリアを含むポートが追加されると完了する。

7. DHCP 接続でポートを待機中のコントローラーは、キャリアをともなわないポートが追加されると待機を継続します。

警告

ネットワークスイッチを使わずにケーブルの直接接続を使用すると、チーミングはサポートされません。本章で説明されているフェイルオーバーメカニズムは、ネットワークスイッチがないと予想どおりに機能しません。詳細についてはナレッジベースの記事『ボンディングは、クロスオーバーケーブルを使用したダイレクトコレクションをサポートしますか ?』を参照してください。

8.3. ネットワークチーミングとボンディングの比較

表8.1 ボンディングおよびチームにおける機能の比較

機能	ボンディング	チーム
ブロードキャスト Tx ポリシー	はい	はい
ラウンドロビン Tx ポリシー	はい	はい
アクティブバックアップ Tx ポリシー	はい	○
LACP (802.3ad) への対応	あり (アクティブのみ)	○
ハッシュベースの Tx ポリシー	○	○
ユーザーによるハッシュ機能の設定	いいえ	○
Tx 負荷分散への対応 (TLB)	○	○
LACP ハッシュポートの選択	○	はい
LACP サポートの負荷分散	いいえ	○
Ethtool リンク監視	○	○

機能	ボンディング	チーム
ARP リンク監視	○	○
NS/NA (IPv6) リンク監視	いいえ	はい
ポート アップ/ダウン 遅延	はい	はい
ポート優先度および持続性 (「プライマリー」オプション強化)	いいえ	はい
ポートごとの個別リンク監視のセットアップ	いいえ	はい
複数のリンク監視セットアップ	限定的	はい
ロックなし Tx/Rx パス	なし (rwlock)	あり (RCU)
VLAN への対応	○	はい
ユーザースペースランタイム制御	限定的	完全
ユーザー空間での論理	いいえ	○
拡張性	困難	容易
モジュラー設計	いいえ	○
パフォーマンスのオーバーヘッド	低	非常に低い
D-Bus インターフェイス	いいえ	はい
複数デバイススタッキング	はい	はい
LLDP を使った zero config	いいえ	(計画中)
NetworkManager への対応	○	はい

8.4. ネットワークチーミングデーモンおよびランナーについて

チームデーモン teamd は、libteam を使用してチームドライバーのインスタンスを制御します。このチームドライバーのインスタンスは、ハードウェアドライバーのインスタンスを追加してネットワークリンクの「チーム」を形成します。チームドライバーがネットワークインターフェイスを提示します。team0 たとえば、カーネルの他の部分まで。チームドライバーのインスタンスによって作成されたインターフェイスの名前は、以下のようになります。 team0,team1ドキュメントに記載されているため、同様。これは分かりやすくするためのもので、他の名前を使っても構いません。チーミングのすべてのメソッドに共通するロジックは、teamd によって実装されます。ラウンドロビンなど、異なる負荷

共有とバックアップメソッドに固有の機能は、「runners」 と呼ばれる別のコードのユニットによって実装されます。「ランナー」という用語がこれらのコードユニットの呼称に選ばれたのは、「モジュール」や「モード」といった言葉がカーネルとの関係ですでに特別な意味を持っているためです。ユーザーは JSON 形式の設定ファイルでランナーを指定し、インスタンスの作成時にコードが teamd のインスタンスにコンパイルされます。ランナーのコードは作成時に teamd のインスタンスにコンパイルされるため、ランナーはプラグインではありません。必要に応じて、teamd のプラグインとしてコードを作成できます。

本ガイド執筆時点では、以下のランナーが利用可能です。

- broadcast (データは全ポートで送信されます)
- round-robin (データは全ポートで順番に送信されます)
- active-backup (1 つのポートまたはリンクが使用され、他はバックアップとして維持されます)
- loadbalance (アクティブ Tx 負荷分散と BPF ベースの Tx ポートセレクターを使用)
- lacp (802.3ad リンクアグリゲーション制御プロトコルを実装)

さらに、以下のリンク監視が利用可能です。

- ethtool (Libteam lib は ethtool を使用してリンク状態の変更を監視します)。設定ファイルで他のリンク監視が指定されていなければ、これがデフォルトになります。
- arp_ping (arp_ping ユーティリティーは、ARP パケットを使用して遠端のハードウェアアドレスの存在を監視するために使用されます。)
- nsna_ping (IPv6 近隣検出プロトコルからの近隣広告と近隣要望は、近隣のインターフェイスの存在を監視するために使用されます)

コードには、特定のリンク監視が特定のランナーで使用されないようにするための制限はありませんが、lacp ランナーを使用する場合は、ethtool が唯一の推奨されるリンク監視です。

8.5. ネットワークチーミングデーモンのインストール

ネットワークチーミングデーモン teamd は、デフォルトではインストールされません。teamd をインストールするには、root で以下のコマンドを実行します。

```
~]# yum install teamd
```

8.6. ボンドのチーム変換

bond2team ツールを使用して、既存のボンディング設定ファイルをチーム設定ファイルに変換できます。ifcfg 形式のボンド設定ファイルを、 ifcfg または JSON 形式のチーム設定ファイルに変換できます。ファイアウォールルール、エイリアスインターフェイス、および元のインターフェイス名に関連付けられる可能性のあるものはすべて、名前変更後に破損する可能性があることに注意してください。これは、このツールは ifcfg ファイルのみを変更し、それ以外は何も変更しないためです。

コマンド形式の例を見るには、以下のコマンドを実行します。

```
~]$ bond2team --examples
```

/tmp/bond2team.XXXXXX/ で始まるディレクトリーに新規ファイルが作成されます。XXXXXX はランダムな文字列です。新しい設定ファイルを作成したら、古いボンディングファイルをバックアップフォルダーに移動し、新しいファイルを /etc/sysconfig/network-scripts/ ディレクトリーに移動します。

例8.1 ボンドのチーム変換

現在の bond0 設定をチーム ifcfg に変換するには、root でコマンドを実行します。

```
~]# /usr/bin/bond2team --master bond0
```

これで bond0 という名前が保持されることに注意してください。新しい名前を使用して設定を保存するには、以下のように --rename を使用します。

```
~]# /usr/bin/bond2team --master bond0 --rename team0
```

ifcfg ファイルではなく JSON 形式のファイルを出力する --json オプションを追加します。

JSON 形式の例については、teamd.conf (5) の man ページを参照してください。

例8.2 ボンドをチームに変換してファイルパスを指定する手順

現在の bond0 設定をチーム ifcfg に変換し、ifcfg ファイルへのパスを手動で指定するには、root でコマンドを実行します。

```
~]# /usr/bin/bond2team --master bond0 --configdir /path/to/ifcfg-file
```

ifcfg ファイルではなく JSON 形式のファイルを出力する --json オプションを追加します。

例8.3 Bond2team を使ってチーム設定を作成する手順

bond2team ツールにボンディングパラメーターの一覧を指定して、チーム設定を作成することもできます。以下に例を示します。

```
~]# /usr/bin/bond2team --bonding_opts "mode=1 miimon=500"
```

以下のように、コマンドラインにポートを提供することもできます。

```
~]# /usr/bin/bond2team --bonding_opts "mode=1 miimon=500 primary=enp1s0 \
 primary_reselect-0" --port enp1s0 --port enp2s0 --port enp3s0 --port enp4s0
```

詳細は、bond2team (1) man ページを参照してください。ボンディングパラメーターの説明については、「チャンネルボンディングの使用」 を参照してください。

8.7. ネットワークチームでポートとして使用するインターフェイスの選択

利用可能なインターフェイスを表示するには、以下のコマンドを実行します。

```
~]$ 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
```

利用可能なインターフェイスから使用するネットワークチームに最適なものを選択し、「ネットワークチーム設定方式の選択」に進みます。

8.8. ネットワークチーム設定方式の選択

NetworkManager のテキストユーザーインターフェイスツール nmtui を使用してネットワークチームを設定するに は、に進みます。 「テキスト形式のユーザーインターフェイス nmtui でネットワークチームを設定する手順」

コマンドラインツールnmcli を使用してネットワークチームを作成するに は、「nmcli を使用したネットワークチーミングの設定」 に進みます。

チームデーモン であるteamd を使用してネットワークチームを作成するに は、「teamd を使用したネットワークチームの作成」 に進みます。

設定ファイルを使用してネットワークチームを作成するには、「ifcfg ファイルを使用したネットワークチームの作成」に進みます。

グラフィカルユーザーインターフェイスを使用してネットワークチームを設定するには、「GUI を使用したネットワークチームの作成」に進みます。

8.9. テキスト形式のユーザーインターフェイス NMTUI でネットワークチームを設定する手順

テキスト形式のユーザーインターフェイスツール nmtui を使用すると、ターミナルのウィンドウでチーミングを設定できます。このツールを起動するには、以下のコマンドを実行します。

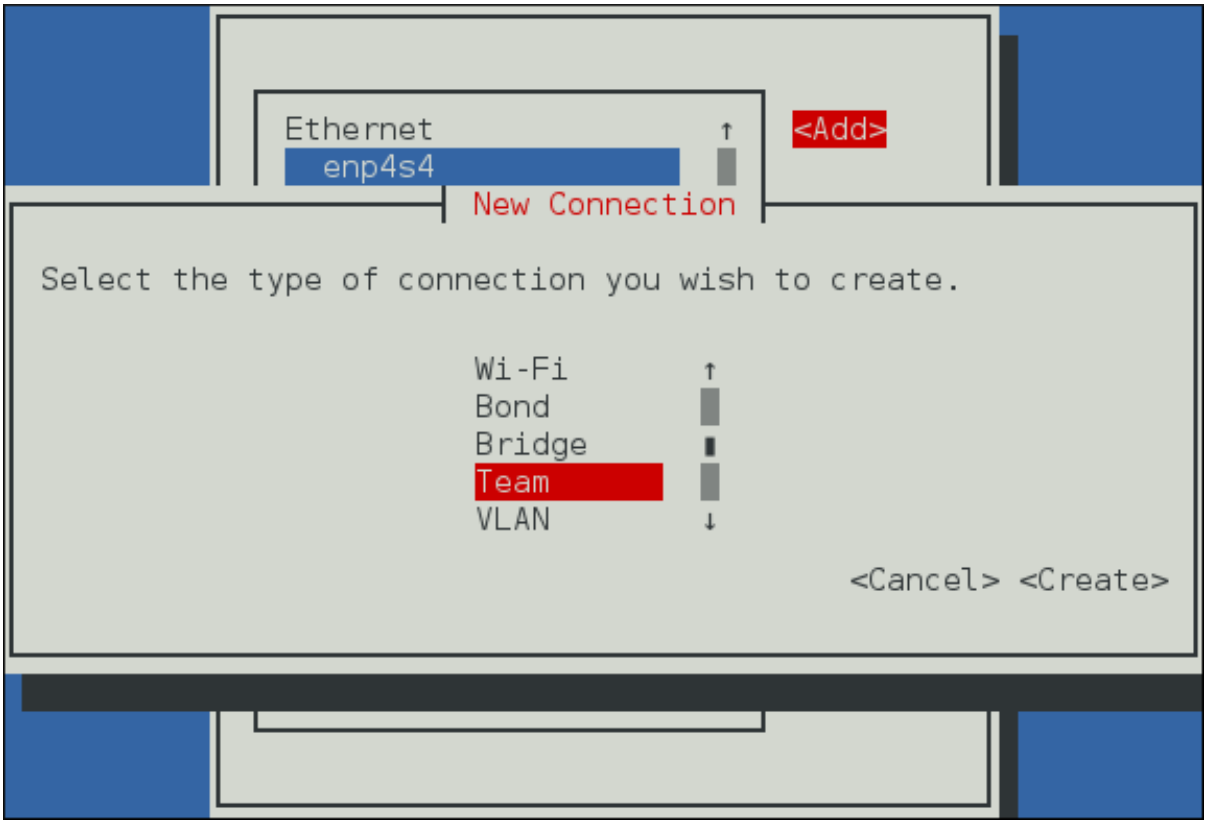
```
~]$ nmtui
```

テキストユーザーインターフェイスが表示されます。無効なコマンドがあると、使用方法に関するメッセージが表示されます。

移動するには、矢印キーを使用するか、Tab を押して順方向に進み、Shift+Tab を押してオプションを再度実行します。Enter を押してオプションを選択します。Space バーは、チェックボックスのステータスを切り替えます。

1. メニューから 接続の編集 を選択します。Add を選択すると、New Connection 画面が開きます。

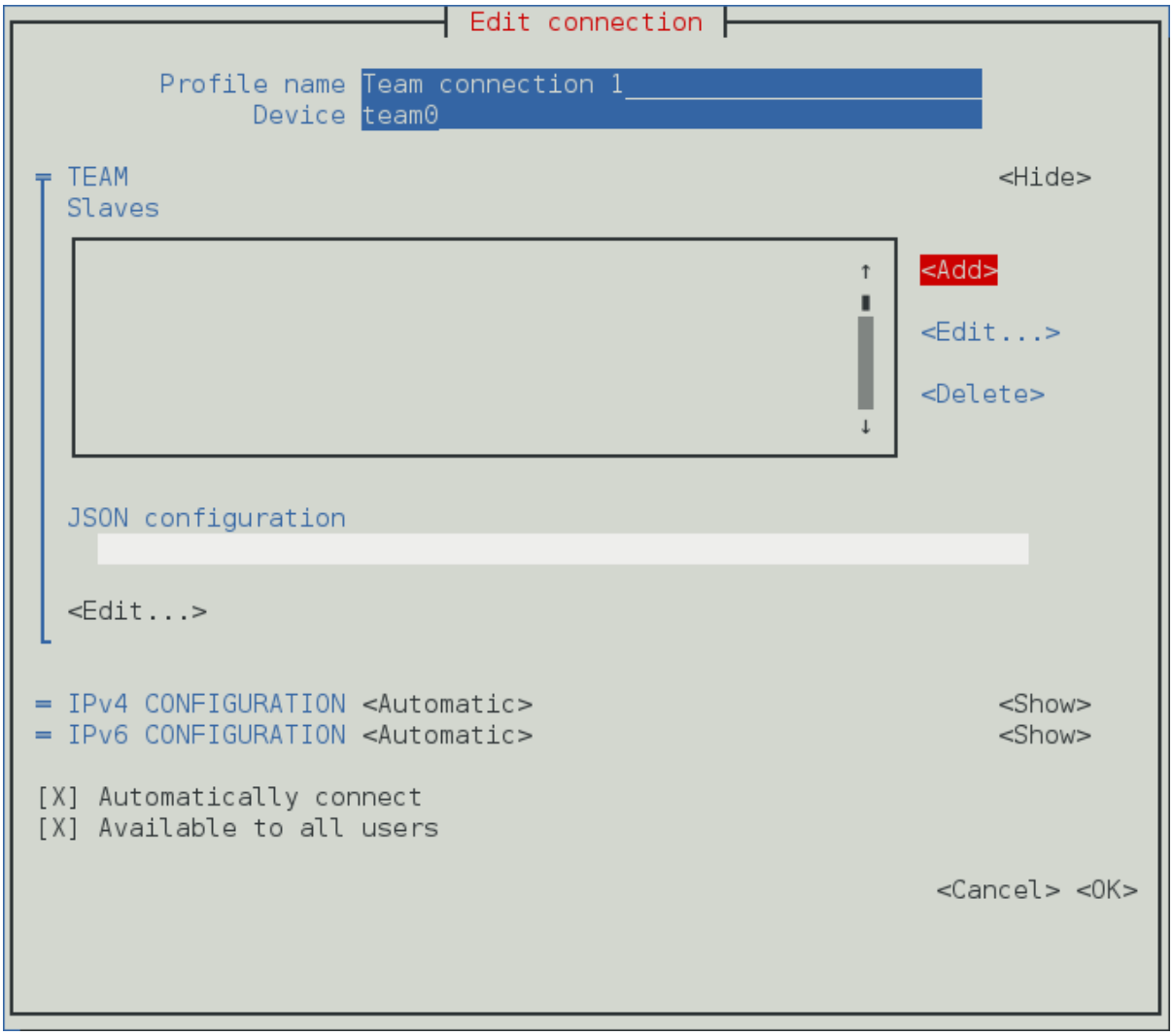
図8.1 NetworkManager テキスト形式のユーザーインターフェイスのチーム接続追加メニュー



[D]

2. チーム を選択すると、接続の編集 画面が開きます。

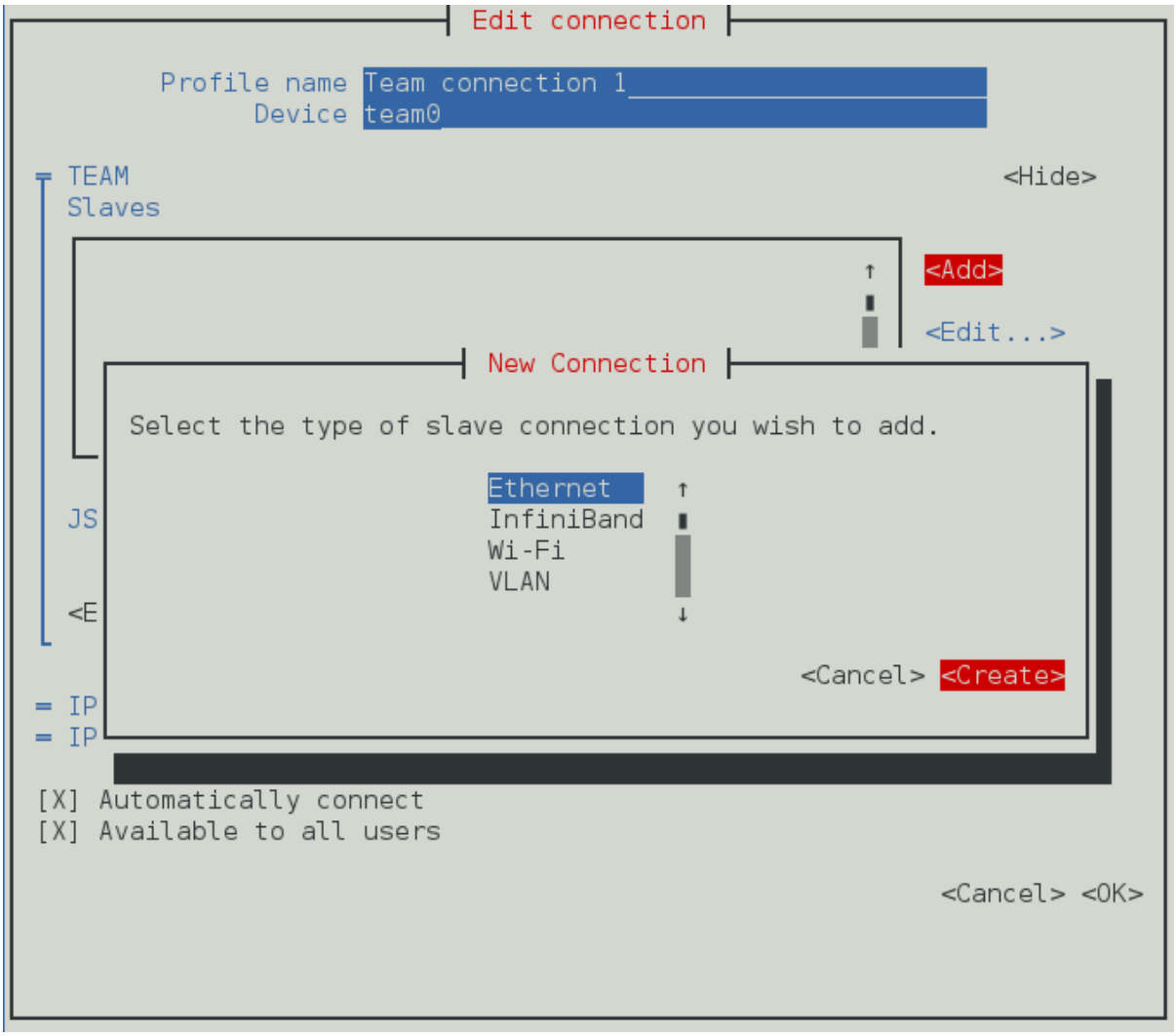
図8.2 NetworkManager テキスト形式ユーザーインターフェイスでチーム接続を設定するメニュー



[D]

3. チームにポートインターフェイスを追加するには 追加 を選択すると、新規接続 画面が開きます。Connection のタイプを選択したら、Create ボタンを選択して、チームの 接続の編集 表示を表示します。

図8.3 NetworkManager テキスト形式ユーザーインターフェイスで新規チームポートインターフェイス接続を設定するメニュー



[D]

4. **Device セクションに、必要なポートのデバイス名または MAC アドレスを入力します。必要な場合は、イーサネット ラベルの右側にある Show を選択して、チームの MAC アドレスとして使用するクローンの MAC アドレスを入力します。OK ボタンを選択します。**

注記

MAC アドレスなしでデバイスを指定すると、Edit Connection ウィンドウが再読み込みされると Device セクションが自動的に入力されますが、デバイスが正常に見つかった場合にのみデバイスセクションが自動的に設定されます。

図8.4 NetworkManager テキスト形式ユーザーインターフェイスでチームのポートインターフェイス接続を設定するメニュー

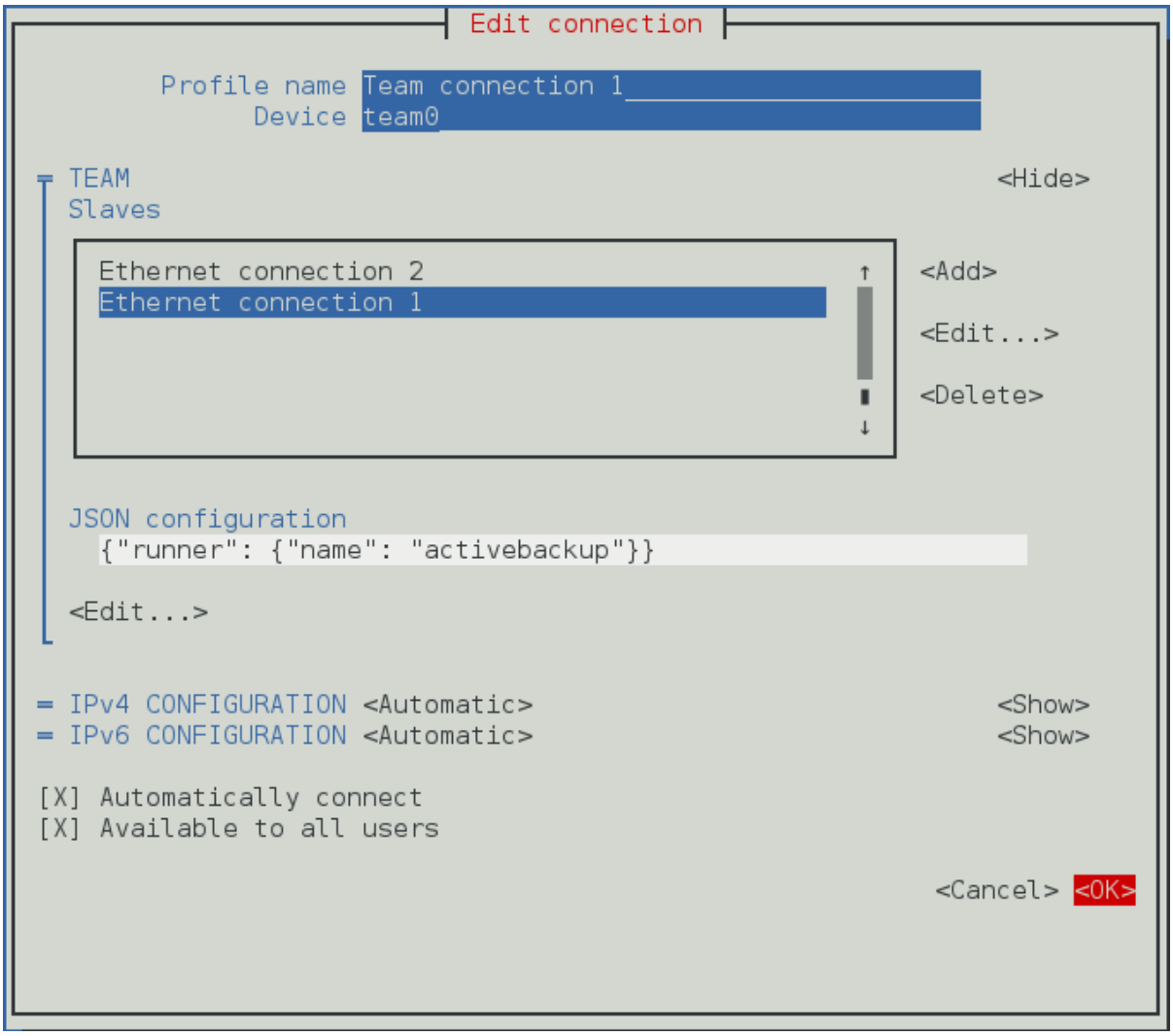
[D]

5. **Slaves セクションにチームポートの名前が表示されます。さらにポート接続を追加する場合は、上記のステップを繰り返します。**

6. **カスタムポート設定を適用する場合は、JSON 設定 セクションの下にある Edit ボタンを選択します。これにより vim コンソールが起動し、変更を適用できます。vim からの変更の書き込みが完了したら、JSON 設定 の下に表示される JSON 文字列が意図したものと一致することを確認します。**

7. **設定を確認してから、OK ボタンを選択します。**

図8.5 NetworkManager テキスト形式ユーザーインターフェイスでチーム接続を設定するメニュー



[D]

JSON 文字列の例については、「teamd ランナーの設定」を参照してください。nmtui を使用したチームまたはポート設定には、文字列例の関連セクションのみを使用する必要があることに注意してください。JSON 文字列の一部として「Device」を指定しないでください。たとえば、チームの JSON 設定フィールドでは、「device」の後から「port」の前までの JSON 文字列を使用してください。ポートに関連するすべての JSON 文字列のみをポート設定フィールドに追加する必要があります。

nmtui のインストール方法は、「nmtui を使用した IP ネットワークの設定」 を参照してください。

8.10. コマンドラインを使用したネットワークチームの設定

8.10.1. nmcli を使用したネットワークチーミングの設定

システムで利用可能な接続を表示するには、以下のコマンドを実行します。

```
~]$ 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  --
```

システムで利用可能なデバイスを表示するには、以下のコマンドを実行します。

```
~]$ nmcli device status
DEVICE     TYPE     STATE                      CONNECTION
virbr0     bridge   connected                  virbr0
ens3       ethernet  connected                  ens3
```

新しいチームインターフェイスを、*ServerA* という名前で作成するには、以下のコマンドを実行します。

```
~]$ nmcli connection add type team ifname ServerA
Connection 'team-ServerA' (b954c62f-5fdd-4339-97b0-40efac734c50) successfully added.
```

NetworkManager は内部パラメーター connection.autoconnect を yes に設定します。IP アドレスが指定されていないため、ipv4.method は auto に設定されます。NetworkManager は、設定ファイルを /etc/sysconfig/network-scripts/ifcfg-team-ServerA に書き込みます。ここで、対応する ONBOOT は yes に、BOOTPROTO は dhcp に設定されます。

ifcfg ファイルへの手動の変更は、インターフェイスが次に起動するまで NetworkManager では認識されません。設定ファイルの使用方法については、「sysconfig ファイルによる NetworkManager の使用」を参照してください。

割り当てた別の値を表示するには、以下のコマンドを実行します。

```
~]$ 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 設定ファイルが指定されていないので、デフォルト値が適用されます。チーム JSON パラ

メーターとそのデフォルト値の詳細は、teamd.conf (5) man ページを参照してください。名前は、インターフェイス名の前に種類を追加したものになっていることに留意してください。別の方法では、以下のように con-name オプションで名前を指定します。

```
~]$ nmcli connection add type team con-name Team0 ifname ServerB
Connection 'Team0' (5f7160a1-09f6-4204-8ff0-6d96a91218a7) successfully added.
```

設定したチームインターフェイスを表示するには、以下のコマンドを実行します。

```
~]$ 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
```

チームに割り当てられた名前を変更するには、以下の形式でコマンドを入力します。

```
nmcli con mod old-team-name connection.id new-team-name
```

すでに存在するチームのチーム設定ファイルを読み込むには、

```
nmcli connection modify team-name team.config JSON-config
```

: JSON 文字列としてチーム設定を指定するか、設定を含むファイル名を指定します。ファイル名には、パスを含めることができます。いずれの場合も、*team.config* プロパティーに保存される内容は JSON 文字列です。JSON 文字列の場合は、文字列を単一引用符で囲み、文字列全体をコマンドラインにペーストします。

team.config プロパティーを確認するには、以下を実行します。

```
nmcli con show team-name | grep team.config
```

team.config プロパティーを設定すると、それに応じてその他のチームプロパティーがすべて更新されます。

対応する JSON 文字列を直接変更せずに、特定のチームオプションを公開して設定するより柔軟な方法も可能になります。これは、利用可能なその他のチームプロパティーを使用して、必要な値に 1 つずつ関連するチームオプションを設定できます。その結果、新しい値に一致するように team.config プロパティーが更新されます。

たとえば、1 つまたは複数のリンク監視を指定できるようにする team.link-watchers プロパティーを設定するには、

```
nmcli connection modify team-name team.link-watchers "name=ethtool delay-up=5,
name=nsna_ping target-host=target.host"
```

の形式でコマンドを入力します。必要な link-watchers はコンマで区切り、同じ link-watcher に属する属性はスペースで区切られます。

team.runner および team.link-watchers プロパティーを設定するには、次の形式でコマンドを入力します。

```
nmcli connection modify team-name team.runner activebackup team.link-watchers
"name=ethtool delay-up=5, name=nsna_ping target-host=target.host"
```

これは、team.config プロパティーを対応する JSON 文字列に設定するのと同じです。

```
nmcli connection modify team-name team.config '{"runner": {"name": "activebackup"},
"link_watch": [{"name": "ethtool", "delay_up": 5},{"name": "nsna_ping", "target_host ":
"target.host"}]'
```

Team0-port1 という名前の Team0 にインターフェイス *enp1s0* を追加するには、以下のコマンドを実行します。

```
~]$ 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.
```

同様に *Team0-port2* の名前で別のインターフェイス *enp2s0* を追加するには、以下のコマンドを実行します。

```
~]$ 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 は、イーサネットポートのみをサポートします。

チームを有効にするには、以下のように最初にポートをアップにする必要があります。

```
~]$ 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)
```

以下のようにポートをアクティブ化することで、チームインターフェイスがアップになっていることを確認できます。

```
~]$ 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
```

あるいは、以下のようにチームを有効にするコマンドを実行します。

```
~]$ nmcli connection up Team0
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/4)
```

nmcliの概要は、「nmcli を使用する IP ネットワークの設定」 を参照してください。

8.10.2. teamd を使用したネットワークチームの作成

注記

teamd を使って作成された設定には永続性がありません。このため、「nmcli を使用したネットワークチーミングの設定」または「ifcfg ファイルを使用したネットワークチームの作成」で定義されているステップを使ってチームを作成する必要がある場合があります。

ネットワークチームを作成するには、ポートまたはリンクのチームに対するインターフェイスとして作動する仮想インターフェイスの設定ファイルが JSON 形式で必要になります。簡単な方法は、サンプル設定ファイルをコピーし、root 権限でエディターを使用してファイルを編集することです。利用可能な設定例をリスト表示するには、以下のコマンドを実行します。

```
~]$ 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.conf など、含まれているファイルのいずれかを表示するには、以下のコマンドを実行します。

```
~]$ 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 設定ファイルを保存する作業設定ディレクトリーを作成します。たとえば通常ユーザーの場合、以下の形式のコマンドを実行します。

```
~]$ mkdir ~/teamd_working_configs
```

選択したファイルを作業ディレクトリーにコピーし、必要に応じて編集します。以下の形式のコマンドを使用できます。

```
~]$ cp /usr/share/doc/teamd-*/example_configs/activebackup_ethtool_1.conf \
~/teamd_working_configs/activebackup_ethtool_1.conf
```

ネットワークチームのポートとして使用するインターフェイスを変更する場合など、使用中の環境に適合するようにファイルを編集するには、以下のように編集するファイルを開きます。

```
~]$ vi ~/teamd_working_configs/activebackup_ethtool_1.conf
```

必要な変更を加えて、ファイルを保存します。vi (1) の man ページでは、vi エディターの使用についてのヘルプを参照するか、お好みのエディターを使用してください。

チーム内でポートとして使用するインターフェイスをチームデバイスに追加する際には、それがアクティブになっていない、つまり「down」になっている必要があることに注意してください。インターフェイスのステータスを確認するには、以下のコマンドを実行します。

```
~]$ 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
```

この例では、使用する予定のインターフェイスはどちらも「UP」になっていることが分かります。

インターフェイスを停止するには、root で以下の形式のコマンドを実行します。

```
~]# ip link set down em1
```

必要に応じて各インターフェイスでこれを繰り返します。

設定ファイルに基づいてチームインターフェイスを作成するには、root ユーザーとして作業設定ディレクトリー（この例では*teamd_working_configs* ）に移動します。

```
~]# cd /home/userteamd_working_configs
```

それから、以下の形式のコマンドを実行します。

```
~]# 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 オプションはデバッグメッセージ用のオプションであり、-f オプションは、読み込む設定ファイルを指定することです。-d オプションは、起動後にプロセスがデーモンとして実行されるようにします。その他のオプションについては、teamd (8) man ページを参照してください。

チームのステータスを確認するには、root で以下のコマンドを実行します。

```
~]# 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
```

アドレスをネットワークチームインターフェイスに適用するには、以下を実行します。team0root で以下の形式でコマンドを実行します。

```
~]# ip addr add 192.168.23.2/24 dev team0
```

チームインターフェイスの IP アドレスを確認するには、以下のコマンドを実行します。

```
~]$ 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
```

チームインターフェイスをアクティブにするか、を起動するには、root で以下の形式のコマンドを実行します。「」

```
~]# ip link set dev team0 up
```

チームインターフェイスを一時的に非アクティブ化するか、を 「停止」 するには、root で以下の形式のコマンドを実行します。

```
~]# ip link set dev team0 down
```

チームデーモンのインスタンスを終了するか、または強制終了するには、root ユーザーとして以下の形式のコマンドを実行します。

```
~]# teamd -t team0 -k
```

-k オプションは、デバイスに関連付けられたデーモンのインスタンスを指定することです。 team0 強制終了されます。その他のオプションについては、teamd (8) man ページを参照してください。

teamd のコマンドラインオプションのヘルプは、次のコマンドを発行します。

```
~]$ teamd -h
```

また、teamd (8) の man ページを参照してください。

8.10.3. ifcfg ファイルを使用したネットワークチームの作成

ifcfg ファイルを使用してネットワークチームを作成するには、以下のように /etc/sysconfig/network-scripts/ ディレクトリーにファイルを作成します。

```
DEVICE=team0
DEVICETYPE=Team
ONBOOT=yes
BOOTPROTO=none
```

```
IPADDR=192.168.11.1
PREFIX=24
TEAM_CONFIG='{"runner": {"name": "activebackup"}, "link_watch": {"name": "ethtool"}}'
```

これでチームへのインターフェイスが作成されます。つまり、これが マスター になります。

メンバーとなるポートを作成するには、以下を実行します。 team0/etc/sysconfig/network-scripts/ ディレクトリーに以下のように 1 つ以上のファイルを作成します。

```
DEVICE=enp1s0
HWADDR=D4:85:64:01:46:9E
DEVICETYPE=TeamPort
ONBOOT=yes
TEAM_MASTER=team0
TEAM_PORT_CONFIG='{"prio": 100}'
```

必要に応じてポートインターフェイスを同様に追加します。追加する各ポート (ネットワークデバイス) に応じて、DEVICE と HWADDR のフィールドを変更します。ポートの優先度が prio によって指定されていない場合、デフォルトで 0 になります。-32,767 から + 32,767 までの範囲の負の値および正の値を受け入れます。

HWADDR ディレクティブを使用してハードウェアまたは MAC アドレスを指定すると、デバイスの命名手順に影響します。これについては、11章*ネットワークデバイス命名における一貫性*で説明します。

ネットワークチームを開くには、root で以下のコマンドを実行します。

```
~]# ifup team0
```

ネットワークチームを表示するには、以下のコマンドを実行します。

```
~]$ ip link show
```

8.10.4. iputils を使用したネットワークチームへのポートの追加

ポートを追加するには、以下を実行します。 em1 ネットワークチームへ team0ip ユーティリティーを使用して、root で以下のコマンドを実行します。

```
~]# ip link set dev em1 down
~]# ip link set dev em1 master team0
```

必要に応じてさらにポートを追加します。チームドライバーが自動的にポートをアップにします。

8.10.5. teamnl を使用したチームのポートのリスト表示

teamnl ユーティリティーを使用して、ネットワークチーム内のポートを表示または一覧表示するには、root で以下のコマンドを実行します。

```
~]# teamnl team0 ports
em2: up 100 fullduplex
em1: up 100 fullduplex
```

8.10.6. teamnl を使用したチームのオプション設定

teamnl ユーティリティーを使用して、現在利用可能なすべてのオプションを表示または一覧表示するには、root で以下のコマンドを実行します。

```
~]# teamnl team0 options
```

チームがアクティブなバックアップモードを使用するように設定するには、root で以下のコマンドを実行します。

```
~]# teamnl team0 setoption mode activebackup
```

8.10.7. iputils を使用したネットワークチームへのアドレス追加

チームへのアドレスの追加 team0ip ユーティリティーを使用して、root で以下のコマンドを実行します。

```
~]# ip addr add 192.168.252.2/24 dev team0
```

8.10.8. iputils を使用したネットワークチームへのインターフェイスの有効化

ネットワークチームへインターフェイスをアクティブまたは 「開く」 には、以下を行います。team0ip ユーティリティーを使用して、root で以下のコマンドを実行します。

```
~]# ip link set team0 up
```

8.10.9. teamnl を使用したチームのアクティブポートオプション表示

teamnl ユーティリティーを使用して、ネットワークチームで activeport オプションを表示または一覧表示するには、root で以下のコマンドを実行します。

```
~]# teamnl team0 getoption activeport
0
```

8.10.10. teamnl を使用したチームのアクティブポートオプション設定

teamnl ユーティリティーを使用して、ネットワークチームで activeport オプションを設定するには、root で以下のコマンドを実行します。

```
~]# teamnl team0 setoption activeport 5
```

チームポートオプションの変更を確認するには、root で以下のコマンドを実行します。

```
~]# teamnl team0 getoption activeport
5
```

8.11. TEAMDCTL を使用した TEAMD の制御

実行中の teamd のインスタンスに対して統計や設定情報をクエリーしたり、変更を加えたりするには、制御ツール teamdctl を使用します。

チームの現在の状態を表示するには、以下を実行します。 team0root で以下のコマンドを入力します。

```
~]# teamdctl team0 state view
```

さらに詳細な出力を表示するには、以下のコマンドを実行します。

```
~]#
```

```
~]# teamdctl team0 state view -v
```

完全な状態のダンプを JSON 形式 (マシン処理に便利) で表示 team0以下のコマンドを使用します。

```
~]# teamdctl team0 state dump
```

JSON 形式の設定ダンプの場合 team0以下のコマンドを使用します。

```
~]# teamdctl team0 config dump
```

ポートの設定を表示するには、 em1これはチームの一部です team0以下のコマンドを入力します。

```
~]# teamdctl team0 port config dump em1
```

8.11.1. ネットワークチームへのポートの追加

ポートを追加するには、以下を実行します。 em1 ネットワークチームへ team0root で以下のコマンドを発行します。

```
~]# teamdctl team0 port add em1
```

重要

teamdctl を直接使用してポートを追加する場合は、ポートを down に設定する必要があります。そうしないと、teamdctl team0 port add em1 コマンドが失敗します。

8.11.2. ネットワークチームからのポートの削除

インターフェイスを削除するには、以下を行います。 **em1** ネットワークチームから **team0root** で以下のコマンドを発行します。

```
~]# teamdctl team0 port remove em1
```

8.11.3. ネットワークチームのポートに対するスティッキー設定の適用

teamdctl コマンドを使用してスティッキー設定を適用し、特定のポートが利用可能なときにアクティブなリンクとして使用されるようにすることができます。

前提条件

- すでにネットワークインターフェイスのチームを作成している。その結果、ポート（**em1**）の設定を更新します。

手順

JSON 形式の設定をポートに適用するには、以下を実行します。 **em1** ネットワークチームにおいて **team0**以下のコマンドを実行します。

1. スティッキー設定を更新します **em1:**

```
~]# teamdctl team0 port config update em1 '{ "prio": 100, "sticky": true }'
```

2. **Remove em1:**

```
~]# teamdctl team0 port remove em1
```

3. 追加 **em1** もう一度スティッキー設定が有効になるようにします。

```
~]# teamdctl team0 port add em1
```

古い設定は上書きされ、省略されたオプションはデフォルト値にリセットされることに注意してください。他のチームデーモン制御ツールコマンド例は、**teamdctl (8)** の **man** ページを参照してください。

8.11.4. ネットワークチーム内のポート設定表示

ポートの設定をコピーするには、以下を実行します。 em1 ネットワークチームにおいて team0root で以下のコマンドを発行します。

```
~]# teamdctl team0 port config dump em1
```

これでポート設定が JSON 形式で標準出力にダンプされます。

8.12. 冗長性についてネットワーク設定チーミングの確認

ネットワークの冗長性は、特定のシステムの障害を防止または回復するために、バックアップ目的でデバイスが使用される場合のプロセスです。次の手順では、冗長性でチーミングのネットワーク設定を確認する方法を説明します。

手順

1. チームインターフェイスから、宛先の IP を ping します。以下に例を示します。

```
~]# ping -I team0 DSTADDR
```

2. どのインターフェイスが アクティブ モードであるかを表示します。

```
~]# 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 **は アクティブ なインターフェイスです。**

3. **ホストからネットワークケーブルを一時的に削除します。**

注記

ソフトウェアユーティリティーを使用してリンク障害イベントを適切にテストする方法はありません。ip や nmcli などの接続を非アクティブにするツールでは、ポート設定の変更を処理するドライバーの機能のみが表示され、実際のリンク障害イベントは表示されません。

4. **バックアップ インターフェイスが起動しているかどうかを確認します。**

```
~]# 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 **が アクティブな インターフェイスになりました。**

5. **チームインターフェイスから、宛先の IP を ping できるかどうかを確認します。**

```
~]# ping -I team0 DSTADDR
```

8.13. TEAMD ランナーの設定

ランナーとは、デーモンのインスタンスが作成される際に、チームデーモンにコンパイルされるコードのユニットです。teamd ランナーについては、「ネットワークチーミングデーモンおよびランナーについて」 を参照してください。

8.13.1. ブロードキャストランナーの設定

ブロードキャストランナーを設定するには、root でエディターを使用して、以下をチームの JSON 形式設定ファイルに追加します。

```
{
 "device": "team0",
 "runner": {"name": "broadcast"},
 "ports": {"em1": {}, "em2": {}}
}
```

詳細は、man ページの teamd.conf (5) を参照してください。

8.13.2. ランダムランナーの設定

ランダムランナーは、ラウンドロビンランナーと同様の動作をします。

ランダムなランナーを設定するには、root でエディターを使用して、以下をチームの JSON 形式設定ファイルに追加します。

```
{
 "device": "team0",
 "runner": {"name": "random"},
 "ports": {"em1": {}, "em2": {}}
}
```

詳細は、man ページの teamd.conf (5) を参照してください。

8.13.3. ラウンドロビンランナーの設定

ラウンドロビンランナーを設定するには、root でエディターを使用して、以下をチームの JSON 形式設定ファイルに追加します。

```
{
 "device": "team0",
 "runner": {"name": "roundrobin"},
 "ports": {"em1": {}, "em2": {}}
}
```

これが、ラウンドロビンの非常に基本的な設定になります。

詳細は、man ページの teamd.conf (5) を参照してください。

8.13.4. アクティブバックアップランナーの設定

アクティブバックアップランナーは、リンク監視すべてを使用してチーム内のリンクのステータスを判断できます。以下のいずれかの例を JSON 形式の設定ファイルに追加できます。

```
{
  "device": "team0",
  "runner": {
    "name": "activebackup"
  },
  "link_watch": {
    "name": "ethtool"
  },
  "ports": {
    "em1": {
      "prio": -10,
      "sticky": true
    },
    "em2": {
      "prio": 100
    }
  }
}
```

この設定例では、ethtool でアクティブバックアップランナーをリンク監視として使用します。

ポート em2 に高い優先度が設定されています。スティッキーフラグにより、 em1 がアクティブになると、リンクが起動している限りアクティブな状態が維持されます。

```
{
  "device": "team0",
  "runner": {
    "name": "activebackup"
  },
  "link_watch": {
    "name": "ethtool"
  },
  "ports": {
    "em1": {
      "prio": -10,
      "sticky": true,
      "queue_id": 4
    },
    "em2": {
      "prio": 100
    }
  }
}
```

この設定例では、4 のキュー ID を追加します。ethtool でアクティブバックアップランナーをリンク監視として使用します。ポート em2 に高い優先度が設定されています。ただし、スティッキーフラグにより、 em1 がアクティブになると、リンクが起動している限りアクティブな状態が維持されます。

ethtool をリンク監視として使用するアクティブバックアップランナーを設定し、遅延を適用するには、root でエディターを使用して、以下をチームの JSON 形式設定ファイルに追加します。

```
{
  "device": "team0",
  "runner": {
    "name": "activebackup"
  },
  "link_watch": {
    "name": "ethtool",
    "delay_up": 2500,
    "delay_down": 1000
  },
  "ports": {
    "em1": {
      "prio": -10,
      "sticky": true
    },
    "em2": {
      "prio": 100
```

```
    }
  }
}
```

この設定例では、ethtool でアクティブバックアップランナーをリンク監視として使用します。ポート em2 に高い優先度が設定されています。ただし、スティッキーフラグにより、 em1 がアクティブになると、リンクが起動している限りアクティブな状態が維持されます。リンク変更はランナーに即座に反映されませんが、遅延は適用されます。

詳細は、man ページの teamd.conf (5) を参照してください。

8.13.5. 負荷分散ランナーの設定

このランナーは、アクティブとパッシブという 2 つのタイプの負荷分散に使用できます。アクティブモードでは、最近のトラフィックの統計値を使用して、トラフィックをできるだけ均一に共有することで、持続的なトラフィックの再分散が図られます。パッシブモードでは、トラフィックのストリームが利用可能なリンクにランダムに分配されます。この方法では処理オーバーヘッドが低くなることから、速度面で有利になります。大量のトラフィックアプリケーションでは、トラフィックは通常、利用可能なリンク間でランダムに分散される複数のストリームで設定されるため、これが推奨されます。このように、負荷共有は teamd による介入なしに行われます。

パッシブ送信(Tx)負荷分散用に負荷分散ランナーを設定するには、root でエディターを使用して、以下をチームの JSON 形式設定ファイルに追加します。

```
{
 "device": "team0",
 "runner": {
  "name": "loadbalance",
  "tx_hash": ["eth", "ipv4", "ipv6"]
 },
 "ports": {"em1": {}, "em2": {}}
}
```

ハッシュベースのパッシブ送信 (Tx) 負荷分散の設定

アクティブ送信(Tx)負荷分散用に負荷分散ランナーを設定するには、root でエディターを使用して、以下をチームの JSON 形式設定ファイルに追加します。

```
{
  "device": "team0",
  "runner": {
    "name": "loadbalance",
    "tx_hash": ["eth", "ipv4", "ipv6"],
    "tx_balancer": {
      "name": "basic"
```

```
    }
  },
  "ports": {"em1": {}, "em2": {}}
}
```

基本的ロードバランサーを使用したアクティブ送信 (Tx) 負荷分散の設定

詳細は、man ページの teamd.conf (5) を参照してください。

8.13.6. LACP (802.3ad) ランナーの設定

ethtool をリンク監視として使用する LACP ランナーを設定するには、root でエディターを使用して、以下をチームの JSON 形式設定ファイルに追加します。

```
{
  "device": "team0",
  "runner": {
     "name": "lacp",
     "active": true,
     "fast_rate": true,
     "tx_hash": ["eth", "ipv4", "ipv6"]
  },
    "link_watch": {"name": "ethtool"},
    "ports": {"em1": {}, "em2": {}}
}
```

接続先が *link aggregation control protocol* (LACP) に対応している場合の接続の設定になります。LACP ランナーは ethtool を使用してリンクのステータスを監視する必要があります。ethtool だけがリンク監視に使用できます。たとえば、arp_ping の場合、リンクは起動しません。この理由は、リンクが最初に確立される必要があり、その後でのみ、ARP を含むパケットが送信可能となるためです。ethtool を使用すると、各リンク層を個別に監視するため、これを防ぐことができます。

このランナーでは、負荷分散ランナーを使用した場合と同様の方法でアクティブ負荷分散が可能になります。アクティブ送信 (Tx) 負荷分散を有効にするには、以下のセクションを追加します。

```
"tx_balancer": {
      "name": "basic"
}
```

詳細は、man ページの teamd.conf (5) を参照してください。

8.13.7. リンクのステータス監視の設定

リンクのステータス監視は以下の方法があります。メソッドのいずれかを実装するには、root 権限でエディターを使用して、JSON 形式の文字列をチームの JSON 形式の設定ファイルに追加します。

8.13.7.1. リンクのステータス監視用の Ethtool 設定

次のリンクと通知を受けるランナー間のミリ秒で既存の遅延を追加または編集するには、以下のようにセクションを追加または編集します。

```
"link_watch": {
        "name": "ethtool",
        "delay_up": 2500
}
```

リンクがダウンになってからそれがランナーに通知されるまでの既存の遅延 (ミリ秒単位) を編集する、または追加するには、以下のセクションを追加もしくは以下のように編集します。

```
"link_watch": {
        "name": "ethtool",
        "delay_down": 1000
}
```

8.13.7.2. リンクのステータス監視用の ARP Ping の設定

チームデーモン teamd は、リンクがアップかどうかを判断するために、ARP REQUEST をリンクのリモート側のアドレスに送信します。使用される方法は arping ユーティリティーと同じですが、そのユーティリティーを使用しません。

以下のような JSON 形式の新規設定を含むファイルを準備します。

```
{
        "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
          }
       }
}
```

この設定では、arp_ping をリンク監視として使用します。missed_max オプションは、逃したリプライの最大許容数 (ARP 応答など) の制限値です。これは interval オプションとともに選択して、リンクがダウンだと報告されるまでの合計回数を決定します。

JSON 設定を含んでいるファイルからチームポート em2JSON 設定が含まれるファイルから、root で以下のコマンドを実行します。

```
~]# teamdctl port config update em2 JSON-config-file
```

古い設定は上書きされ、省略されたオプションはデフォルト値にリセットされることに注意してください。他のチームデーモン制御ツールコマンド例は、teamdctl (8) の man ページを参照してください。

8.13.7.3. リンクのステータス監視用の IPv6 NA/NS の設定

```
{
  "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 パケットの送信間隔を設定するには、以下のセクションを追加もしくは以下のように編集します。

```
"link_watch": {
    "name": "nsna_ping",
    "interval": 200
}
```

ここでの値は、ミリ秒単位の正の数になります。これは missed_max オプションとともに選択して、リンクがダウンだと報告されるまでの合計回数を決定します。

リンクがダウンだと報告されるまでの、実行されなかった NS/NA 返信パケットの最大数を設定するには、以下のセクションを追加もしくは以下のように編集します。

```
"link_watch": {
    "name": "nsna_ping",
    "missed_max": 15
}
```

実行されなかった NS/NA 返信パケットの最大数。この数を超えると、リンクがオフラインになっていると報告されます。missed_max オプションは、逃したリプライの最大許容数 (ARP 応答など) の制限値です。これは interval オプションとともに選択して、リンクがダウンだと報告されるまでの合計回数を決定します。

NS/NA パケットの IPv6 アドレスターゲットアドレスに解決されるホスト名を設定するには、以下のセクションを追加または編集します。

```
"link_watch": {
    "name": "nsna_ping",
    "target_host": "MyStorage"
}
```

「target_host」 オプションには、NS/NA パケットのターゲットアドレスとして使用される IPv6 アドレスに変換されるホスト名が含まれます。IPv6 アドレスは、ホスト名の代わりに使用できます。

詳細は、man ページの teamd.conf (5) を参照してください。

8.13.8. ポート選択上書きの設定

フレームを送信する物理的なポートは、通常、チームドライバーのカーネル部分が選択するもので、ユーザーまたはシステム管理者とは関係がありません。出力ポートは、選択したチームモード (teamd ランナー)のポリシーを使用して選択されます。ただし場合によっては、送信トラフィックの特定クラスを、特定の物理的インターフェイスに向けて、やや複雑なポリシーを実装することが役に立つこともあります。デフォルトでは、チームドライバーはマルチキューを認識し、ドライバーが初期化されると 16 のキューが作成されます。キューが増減する必要がある場合は、Netlink 属性 tx_queues を使用してチームドライバーのインスタンスの作成時にこの値を変更できます。

ポートのキュー ID は、以下のようにポート設定オプション queue_id で設定できます。

```
{
 "queue_id": 3
}
```

これらのキュー ID を tc ユーティリティーと組み合わせて使用して、マルチキューキュー識別子とフィルターを設定し、特定のポートデバイスで特定のトラフィックを送信するようにフィルターを設定できます。たとえば、上記の設定を使用し、192.168.1.100 にバインドされているすべてのトラフィックを強制的に使用させる場合 enp1s0 チームで出力デバイスとして、root で以下の形式のコマンドを実行します。

```
~]# 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
```

トラフィックを特定ポートにバインドするためにランナー選択論理を上書きするこのメカニズムは、すべてランナーに使用できます。

8.13.9. BPF ベースの Tx ポートセレクターの設定

負荷分散および LACP ランナーは、パケットのハッシュを使ってネットワークトラフィックのフローを分類します。ハッシュの計算メカニズムは、*Berkeley Packet Filter* (BPF) コードに基づいています。BPF コードは、送信パケットのポリシー判断の作成ではなく、ハッシュ生成のために使用されます。ハッシュの長さは 8 ビットで、256 バリアントになります。つまり、多くの異なる *ソケットバッファー* (SKB) は同じハッシュを持つことが可能で、このため同一リンクでトラフィックを渡すことになります。短いハッシュを使うと、複数のリンクに負荷を分散する目的でトラフィックを異なるストリームにすばやく分類できます。静的モードでは、トラフィックをどのポートに送信するかを判断するためだけにハッシュが使用されます。アクティブモードでは、ランナーは継続的にハッシュを異なるポートに割り当て、完全な負荷分散を試みます。

パケット Tx ハッシュの計算には、以下の断片化されたタイプまたは文字列が使用できます。

- eth: ソースおよび宛先の MAC アドレスを使用します。

- VLAN: VLAN ID を使用します。
- ipv4: ソースおよび宛先の IPv4 アドレスを使用します。
- ipv6 - ソースおよび宛先 IPv6 アドレスを使用します。
- ip: ソースおよび宛先の IPv4 および IPv6 アドレスを使用します。
- L3: ソースおよび宛先の IPv4 および IPv6 アドレスを使用します。
- TCP: ソースおよび宛先の TCP ポートを使用します。
- UDP: ソースおよび宛先の UDP ポートを使用します。
- SCTP: ソースおよび宛先の SCTP ポートを使用します。
- L4: ソースおよび宛先の TCP および UDP および SCTP ポートを使用します。

これらの文字列は、負荷分散ランナーに行を追加することによって使用できます。

```
"tx_hash": ["eth", "ipv4", "ipv6"]
```

例は「負荷分散ランナーの設定」を参照してください。

8.14. GUI を使用したネットワークチームの作成

8.14.1. チーム接続の確立

nm-connection-editor を使用して、NetworkManager に、2 つ以上の Wired 接続または InfiniBand 接続からチームを作成できます。接続が最初にチーミングされている必要はありません。

チームを設定するプロセスの一部として設定することが可能です。設定プロセスを完了するには、利用可能なインターフェイスの MAC アドレスが必要です。

手順8.1 nm-connection-editor を使用して新規チーム接続を追加する

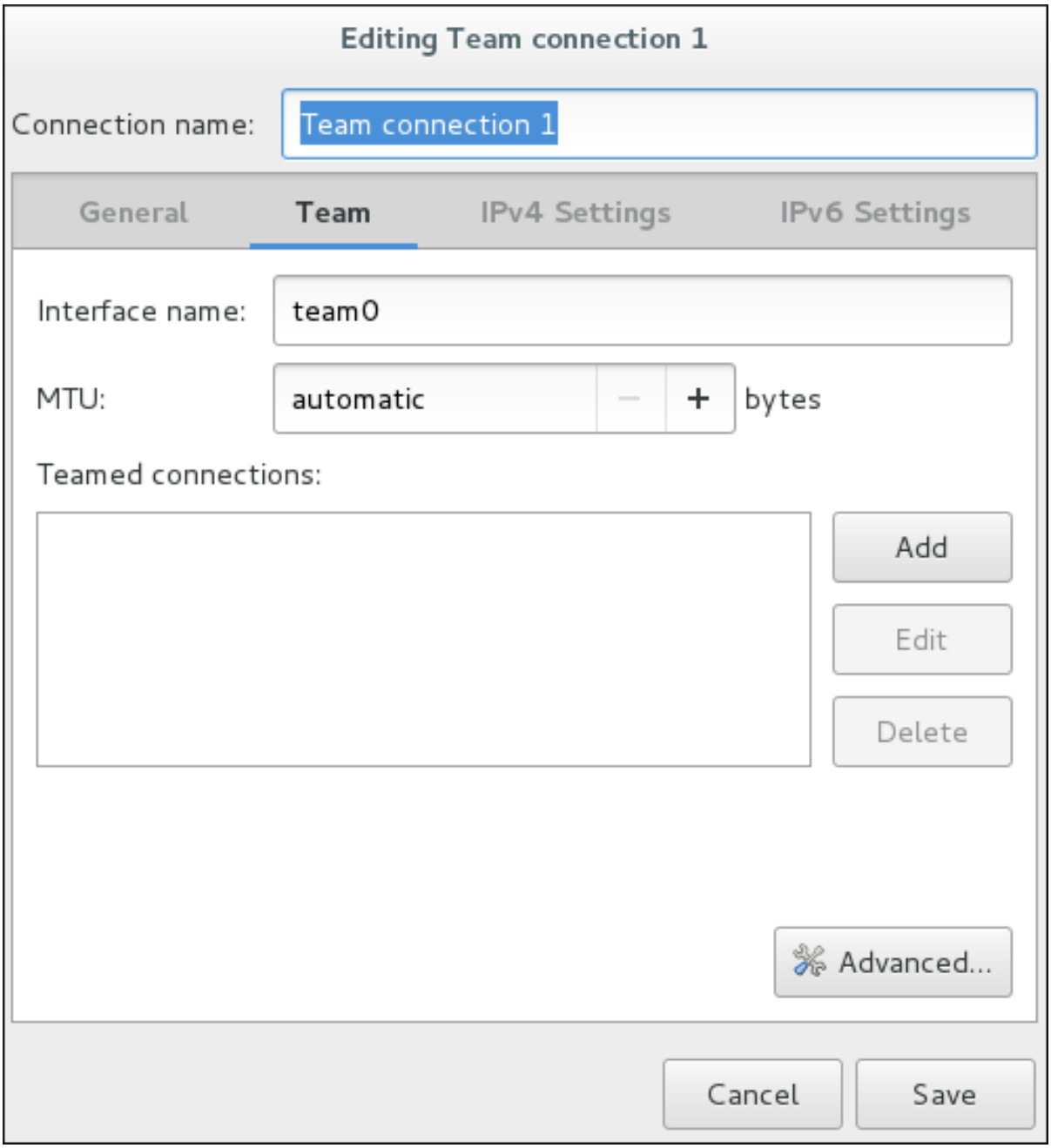
新規チーム接続を追加するには、以下のステップに従います。

1. 端末に **nm-connection-editor** と入力します。

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

2. **Add** ボタンをクリックします。**Choose a Connection Type** ウィンドウが表示されます。**Team** を選択し、**Create** を クリックします。チーム接続 *1*の編集 ウィンドウが表示されます。

図8.6 NetworkManager グラフィカルユーザーインターフェイスの追加メニュー

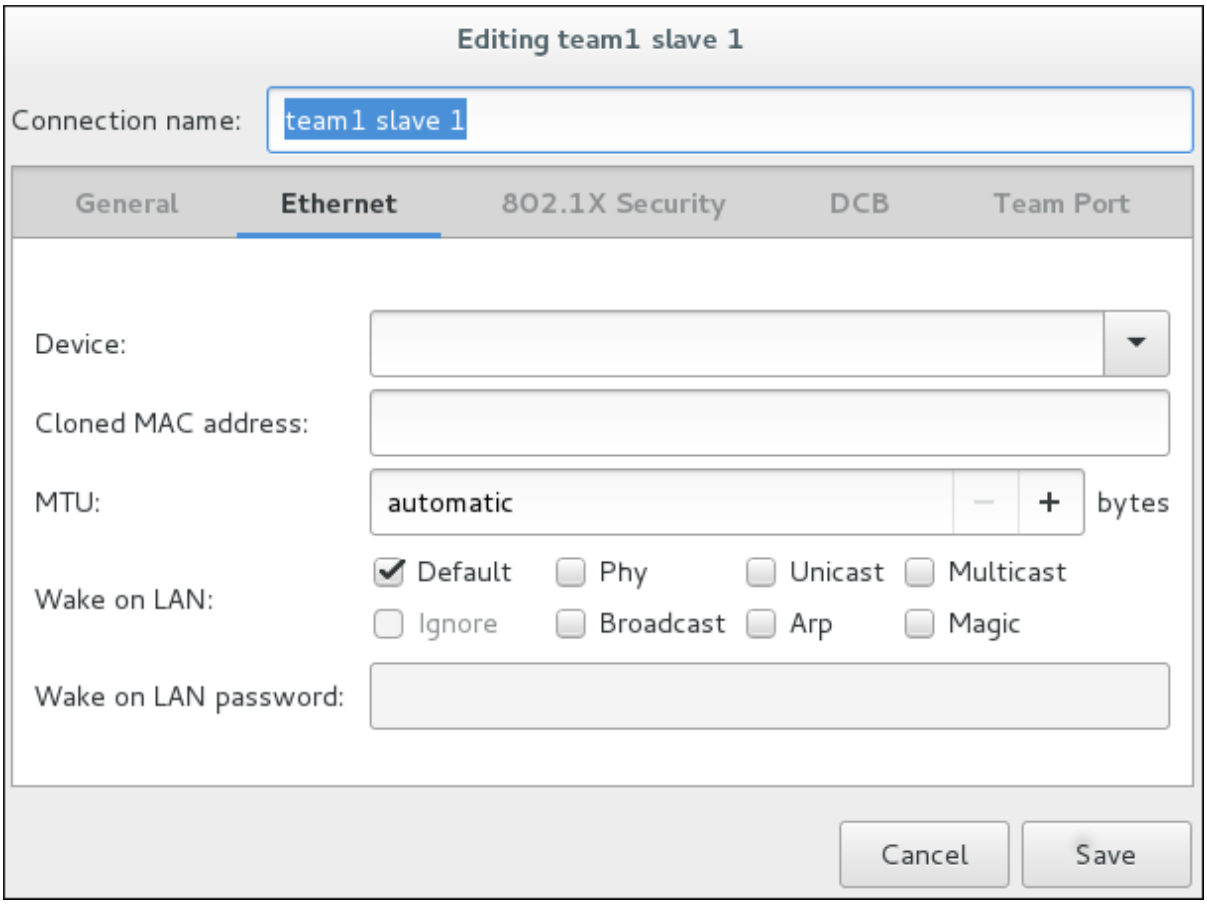


[D]

3. **チーム タブで 追加 をクリックし、チーム接続で使用するインターフェイスのタイプを選択します。Create ボタンをクリックします。ポートタイプを選択するダイアログが表示されるのは、最初のポートを作成する時のみです。その後は、すべてのポートに同じタイプが自動的に使われます。**

4. **team0 スレーブ 1 の編集 ウィンドウが表示されます。**

図8.7 NetworkManager グラフィカルユーザーインターフェイスの スレーブ 接続追加



[D]

5. カスタムポート設定を適用する場合は、**Team Port** タブをクリックして **JSON** 設定文字列を入力するか、ファイルからインポートします。

6. **Save** ボタンをクリックします。

7. チーミングされたポートの名前が **Team connections** ウィンドウに表示されます。**Add** ボタンをクリックして、さらにポート接続を追加します。

8. 設定を確認してから **Save** ボタンをクリックします。

9. チーム固有の設定は、以下の「チームタブの設定」を参照してください。

手順8.2 既存のチーム接続を編集する

既存のチーム接続を編集するには以下の手順に従います。

1. 端末に nm-connection-editor と入力します。

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

2. 編集する接続を選択し、Edit ボタンをクリックします。

3. General タブを選択します。

4. 編集 ダイアログの 5 つの設定は、ほとんどの接続の種類に共通です。General タブを参照してください。

- 接続名: ネットワーク接続のわかりやすい名前を入力します。この名前は、Network ウィンドウのメニューでこの接続を一覧表示するために使用されます。
- Connection priority for auto-activation - 接続が自動接続に設定されていると、番号がアクティブになります（デフォルトでは0 ）。数値が大きいほど優先度が高くなります。
- Automatically connect to this network when it is available - このボックスを選択すると、NetworkManager が利用可能なときにこの接続に自動接続します。詳細は、「control-center を使用した既存の接続の編集」を参照してください。
- All users can connect to this network - このボックスを選択すると、システム上のすべてのユーザーが 利用できる接続が作成されます。この設定を変更するには、root 権限が必要になる場合があります。詳細は、「GUI を使用したシステム全体およびプライベート接続プロファイルの管理」を参照してください。
- Automatically connect to VPN when using this connection - このボックスを選択すると、NetworkManager が利用可能なときに VPN 接続に自動接続します。ドロップダウンメニューから VPN を選択します。
- ファイアウォールゾーン - ドロップダウンメニューからファイアウォールゾーンを選択します。ファイアウォールゾーンに関する詳細情報は、『Red Hat Enterprise Linux 7 セキュリティーガイド』を参照してください。

5. チーム固有の設定は、以下の「チームタブの設定」を参照してください。

新規 (または修正した) 接続を保存して他の設定を行う

チーム接続の編集が終わったら、保存 ボタンをクリックしてカスタマイズした設定を保存します。

そして、以下のいずれかの設定をします。

- IPv4 の設定は、IPv4 のセッティング タブをクリックしてに進みます。「IPv4 設定の設定」

  または

- IPv6 の設定は、IPv6 のセッティング タブをクリックして、「IPv6 セッティングの設定」に進みます。

8.14.1.1. チームタブの設定

すでに新規チーム接続が追加されている場合は、カスタムの JSON 設定文字列をテキストボックスに入力するか、設定ファイルをインポートできます。Save をクリックして、JSON 設定をチームインターフェイスに適用します。

JSON 文字列の例については、「teamd ランナーの設定」を参照してください。

新規チームの追加方法は 手順8.1「nm-connection-editor を使用して新規チーム接続を追加する」を参照してください。

8.15. 関連情報

インストールされているドキュメント

- teamd (8) man ページ - teamd サービスについて説明しています。
- teamdctl (8) man ページ - teamd 制御ツールについて説明しています。

- teamd.conf (5) man ページ - teamd 設定ファイルが説明されています。
- teamnl (8) man ページ - teamd Netlink ライブラリーについて説明しています。
- bond2team (1) man ページ：ボンディングオプションをチームに変換するツールを説明しています。

オンラインドキュメント

http://www.w3schools.com/js/js_json_syntax.asp

JSON 構文についての説明です。

第9章 ネットワークブリッジングの設定

ネットワークブリッジは、MAC アドレスに基づいてネットワーク間のトラフィックを転送するリンク層デバイスです。転送の決定は、MAC アドレスのテーブルに基づいて行われ、このテーブルはネットワークトラフィックをリッスンして、どのホストがどのネットワーク接続しているかをネットワークブリッジが学習することで構築されます。Linux ホスト内では、ソフトウェアブリッジを使ってハードウェアをエミュレートすることができます。たとえば、仮想化アプリケーション内で NIC を 1 つ以上の仮想 NIC と共有するなどです。

アドホック または *インフラストラクチャー* モードで稼働している Wi-Fi ネットワーク上では、ブリッジは確立できないことに注意してください。IEEE 802.11 標準が、通信時間の効率性のために Wi-Fi で 3 アドレスフレームの使用を指定するためです。

9.1. テキスト形式のユーザーインターフェイス NMTUI によるブリッジングの設定

テキスト形式のユーザーインターフェイスツール nmtui を使用すると、端末ウィンドウでブリッジングを設定できます。このツールを起動するには、以下のコマンドを実行します。

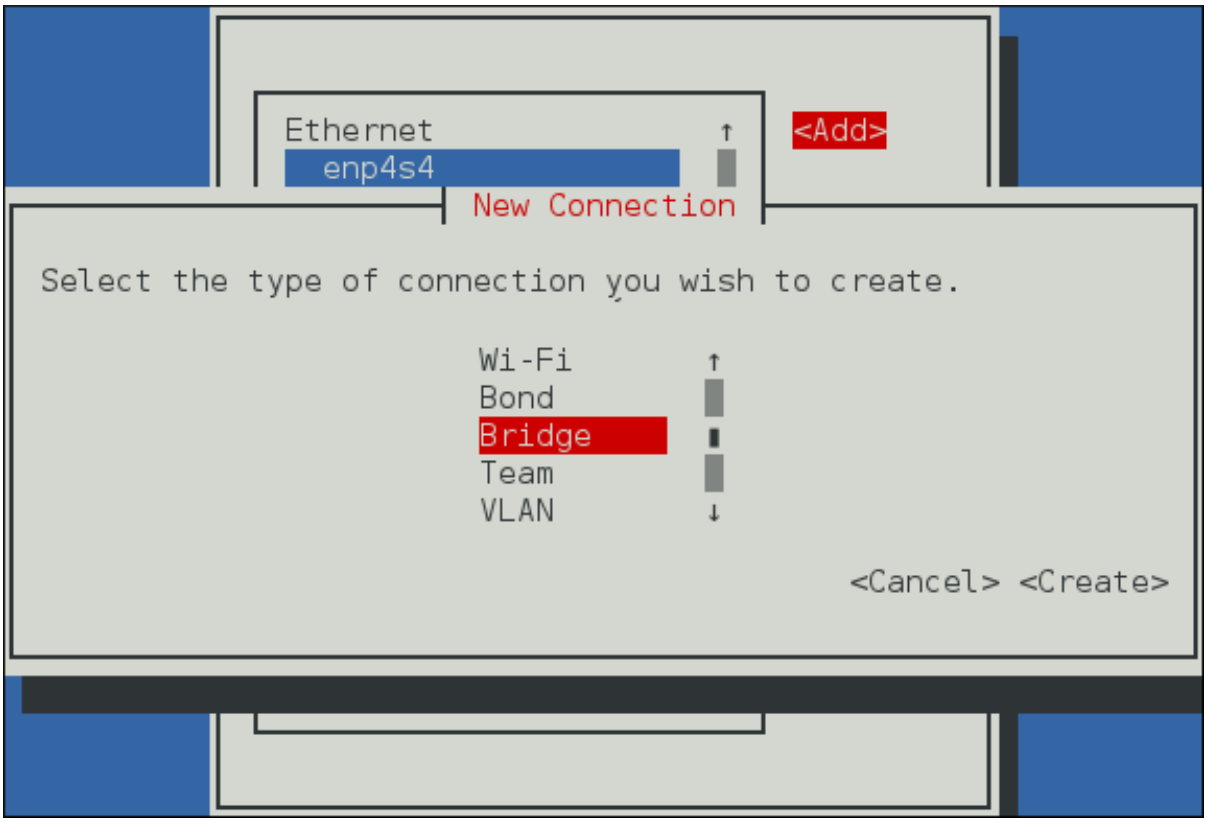
```
~]$ nmtui
```

テキストユーザーインターフェイスが表示されます。無効なコマンドがあると、使用方法に関するメッセージが表示されます。

移動するには、矢印キーを使用するか、Tab を押して順方向に進み、Shift+Tab を押してオプションを再度実行します。Enter を押してオプションを選択します。Space バーは、チェックボックスのステータスを切り替えます。

1. メニューから 接続の編集 を選択します。Add を選択すると、New Connection 画面が開きます。

図9.1 NetworkManager テキスト形式のユーザーインターフェイスのブリッジ接続追加メニュー

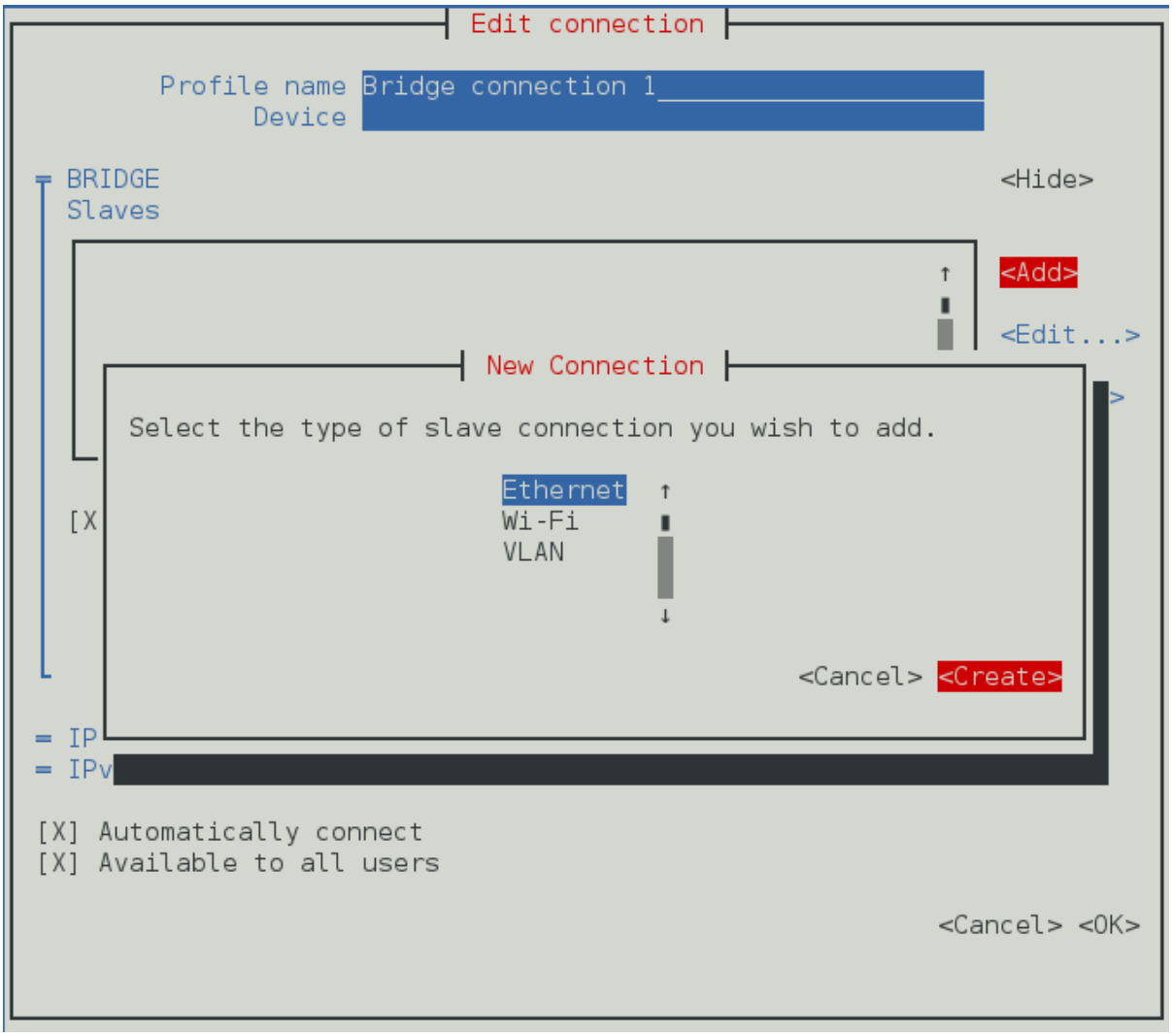


[D]

2. **Bridge を選択すると、接続の編集 画面が開きます。**

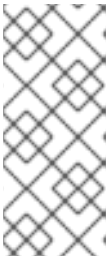
3. **ブリッジにポートインターフェイスを追加するには 追加 を選択すると、新規接続 画面が開きます。Connection のタイプを選択したら、Create ボタンを選択して、ブリッジの Edit Connection 表示を表示します。**

図9.2 NetworkManager テキスト形式ユーザーインターフェイスで新規ブリッジ スレーブ 接続の追加メニュー



[D]

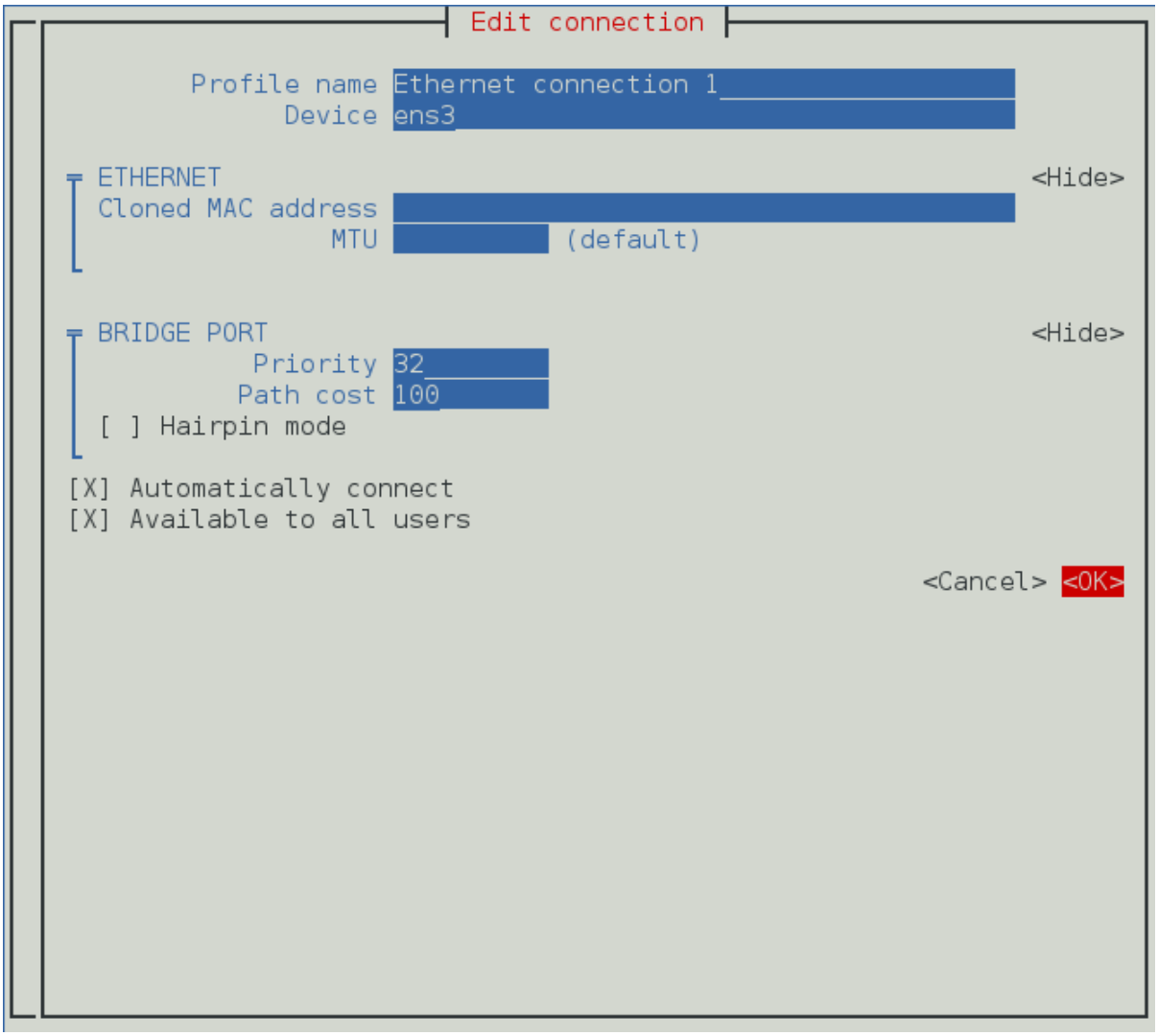
4. **Device セクションに、必要なポートのデバイス名または MAC アドレスを入力します。必要に応じて、イーサネット ラベルの右側にある Show を選択して、ブリッジの MAC アドレスとして使用するクローンの MAC アドレスを入力します。OK ボタンを選択します。**



注記

MAC アドレスなしでデバイスを指定すると、Edit Connection ウィンドウが再読み込みされると Device セクションが自動的に入力されますが、デバイスが正常に見つかった場合にのみデバイスセクションが自動的に設定されます。

図9.3 NetworkManager テキスト形式ユーザーインターフェイスでブリッジ スレーブ を設定するメニュー

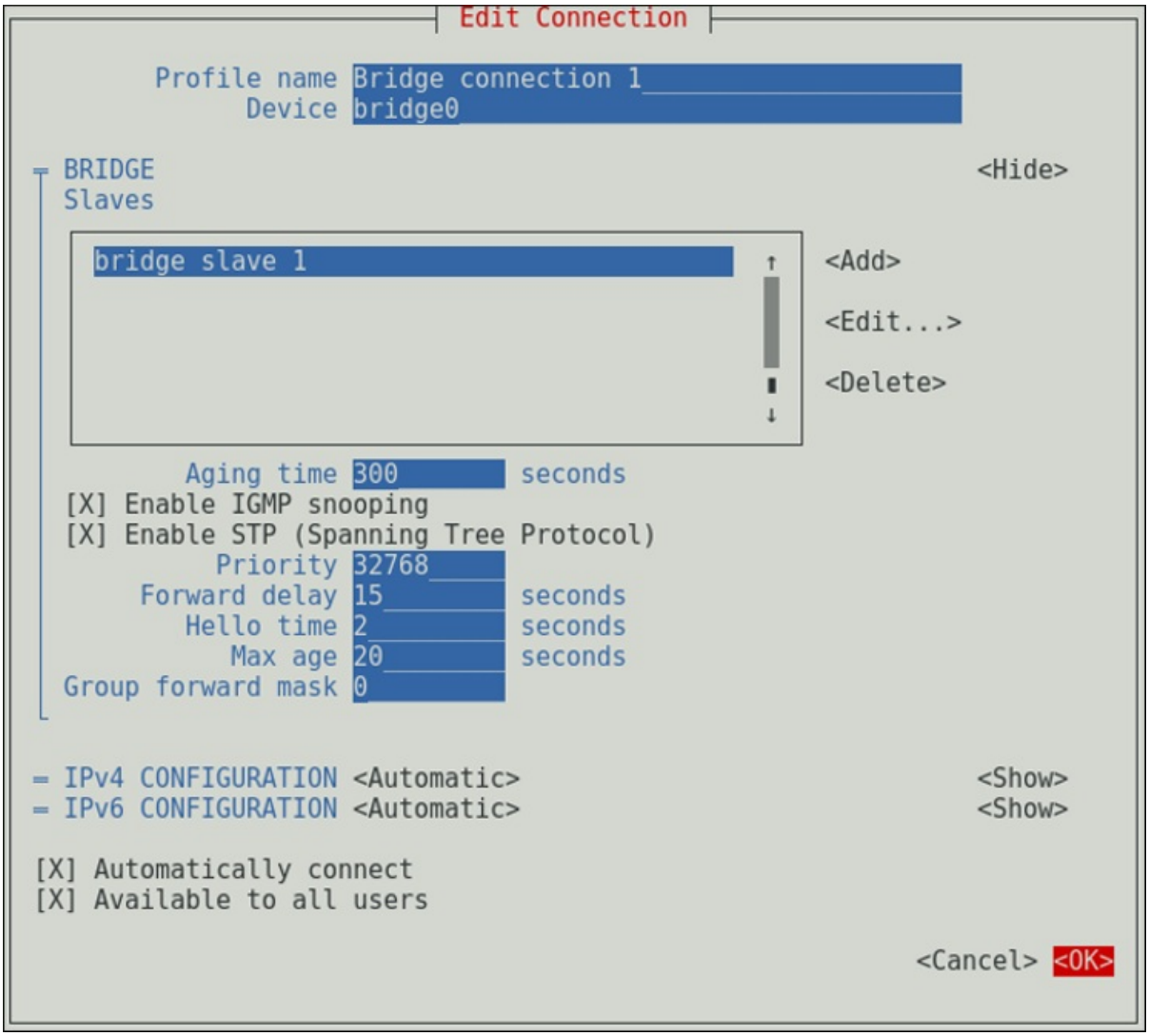


[D]

5. **Slaves セクションにブリッジポートの名前が表示されます。さらにポート接続を追加する場合は、上記のステップを繰り返します。**

6. **設定を確認してから、OK ボタンを選択します。**

図9.4 NetworkManager テキスト形式ユーザーインターフェイスでブリッジを設定するメニュー



[D]

ブリッジ用語の定義は、「ブリッジタブを設定する」を参照してください。

nmtui のインストール方法は、「nmtui を使用した IP ネットワークの設定」 を参照してください。

9.2. NETWORKMANAGER のコマンドラインツール NMCLI の使用

ブリッジを作成するには、次のコマンドを実行します。 bridge-br0root で以下のコマンドを発行します。

```
~]# nmcli con add type bridge ifname br0
Connection 'bridge-br0' (6ad5bba6-98a0-4f20-839d-c997ba7668ad) successfully added.
```

インターフェイス名が指定されない場合、名前はデフォルトで bridge,bridge-1,bridge-2などになります。

接続を表示するには、以下のコマンドを実行します。

```
~]$ 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
```

デフォルトでは、*スパニングツリープロトコル* (STP) が有効になっています。使用される値は、IEEE 802.1D-1998 標準からのものです。このブリッジの STP を無効にするには、root で以下のコマンドを実行します。

```
~]# nmcli con modify bridge-br0 bridge.stp no
```

このブリッジの 802.1D STP を再度有効にするには、root で以下のコマンドを実行します。

```
~]# nmcli con modify bridge-br0 bridge.stp yes
```

802.1D STP のデフォルトのブリッジ優先度は 32768 です。root ブリッジ選択では、少ない値が選ばれます。たとえば、優先度が 28672 のブリッジは、優先度が 32768 （デフォルト）のブリッジよりも root ブリッジとして選択されます。デフォルト値以外の値のブリッジを作成するには、以下のコマンドを実行します。

```
~]$ nmcli con add type bridge ifname br5 stp yes priority 28672
Connection 'bridge-br5' (86b83ad3-b466-4795-aeb6-4a66eb1856c7) successfully added.
```

許可される値は 0 から 65535 です。

既存ブリッジのブリッジ優先度をデフォルト値以外の値に変更するには、以下の形式のコマンドを実行します。

```
~]$ nmcli connection modify bridge-br5 bridge.priority 36864
```

許可される値は 0 から 65535 です。

範囲内のグループアドレスを転送するブリッジ接続を 01 :80:C2: 00:00:00 から 01:80:C2:00:00:0F

に設定するには、**group-forward-mask** プロパティーを変更します。このプロパティーは、**16** ビットのマスクです。各ビットは、転送する必要がある上記の範囲内のグループアドレスに対応します。以下に例を示します。

```
~]$ nmcli connection modify bridge-br5 bridge.group-forward-mask 8
```

重要

group-forward-mask プロパティーは、**0**、**1**、**2** ビットを **1** に設定することはできません。これは、スパニングツリープロトコル(**STP**)、リンクアグリゲーション制御プロトコル(**LACP**)、およびイーサネット **MAC** 一時停止フレームに使用されるためです。

ブリッジの設定を表示するには、以下のコマンドを実行します。

```
~]$ nmcli -f bridge con show bridge-br0
```

802.1D STP のその他のオプションは、**man** ページの **nmcli (1)** の **bridge** セクションに記載されています。

インターフェイスを追加または割り当てる場合、たとえば **enp1s0**をブリッジ **bridge-br0**に割り当てるには、以下のコマンドを実行します。

```
~]$ nmcli con add type ethernet ifname enp1s0 master bridge-br0
Connection 'bridge-slave-enp1s0' (70ffae80-7428-4d9c-8cbd-2e35de72476e) successfully
added.
```

既存の接続をブリッジに割り当てるには、以下の手順に従ってください。

1. そのコントローラーとポートタイプのプロパティーを変更します。たとえば、***vlan100*** という名前の既存の **VLAN** 接続を割り当てるには、次のコマンドを実行します。

   ```
   ~]$ nmcli connection modify vlan100 master bridge-br0 slave-type bridge
   ```

2. 接続を再度アクティブにして、変更を適用します。

```
~]$ nmcli connection up vlan100
```

インタラクティブモードを使用して値を変更するには、以下のコマンドを実行します。

```
~]$ nmcli connection edit bridge-br0
```

nmcli プロンプトが表示されます。

```
nmcli> set bridge.priority 4096
nmcli> save
Connection 'bridge-br0' (79cf6a3e-0310-4a78-b759-bda1cc3eef8d) successfully saved.
nmcli> quit
```

nmcli の概要は、「nmcli を使用する IP ネットワークの設定」 を参照してください。

9.3. コマンドラインインターフェイス (CLI) の使用

9.3.1. ブリッジングカーネルモジュールがインストールされているかの確認

Red Hat Enterprise Linux 7 では、ブリッジングモジュールはデフォルトで読み込まれています。必要に応じて、root で以下のコマンドを実行して、モジュールがロードされていることを確認することができます。

```
~]# modprobe --first-time bridge
modprobe: ERROR: could not insert 'bridge': Module already in kernel
```

モジュールについての情報を表示するには、以下のコマンドを実行します。

```
~]$ modinfo bridge
```

コマンドオプションについては、modprobe (8) の man ページを参照してください。

9.3.2. ネットワークブリッジの作成

ネットワークブリッジを作成するには、/etc/sysconfig/network-scripts/ ディレクトリーに ifcfg-br*N* という名前のファイルを作成し、*N* を 0 などのインターフェイスの番号に置き換えます。

ファイルのコンテンツは、イーサネットインターフェイスなどブリッジされるインターフェイスがどのようなタイプでも類似したものになります。相違点は、以下のとおりです。

- DEVICE ディレクティブは、br*N* 形式の引数としてインターフェイス名を指定しています。*N* はインターフェイスの数に置き換えます。
- TYPE ディレクティブには、引数 Bridge が指定されています。このディレクティブは、デバイスタイプと、引数が大文字/小文字を区別するかを決定します。
- ブリッジインターフェイス設定ファイルには IP アドレスが割り当てられますが、物理インターフェイスの設定ファイルには MAC アドレスのみが必要です（以下を参照）。
- 追加のディレクティブ DELAY=0 が加えられ、ブリッジがトラフィックを監視し、ホストの位置を学習し、フィルタリング機能の基になる MAC アドレステーブルを構築する間に、ブリッジが待機することを回避します。ルーティングループが可能でない場合は、デフォルトの 15 秒遅延は不要です。

例9.1 ifcfg-br0 インターフェイス設定ファイルの例

以下は、静的 IP アドレスを使用したブリッジインターフェイスの設定ファイルの例になります。

```
DEVICE=br0
TYPE=Bridge
IPADDR=192.168.1.1
PREFIX=24
BOOTPROTO=none
ONBOOT=yes
DELAY=0
```

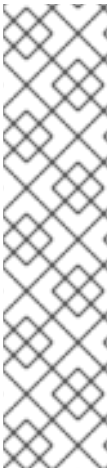
ブリッジを完成するには、別のインターフェイスを作成するか既存のインターフェイスを修正して、これをブリッジインターフェイスに向けます。

例9.2 ifcfg-enp1s0 インターフェイス設定ファイルの例

以下の例は、イーサネットインターフェイス設定ファイルをブリッジインターフェイスに向けたものです。/etc/sysconfig/network-scripts/ifcfg-*device_name* で物理インターフェイスを設定します。*device_name* はインターフェイスの名前です。

```
DEVICE=device_name
TYPE=Ethernet
HWADDR=AA:BB:CC:DD:EE:FF
BOOTPROTO=none
ONBOOT=yes
BRIDGE=br0
```

オプションで、NAME ディレクティブを使って名前を指定することもできます。名前が指定されていない場合、NetworkManager プラグインである ifcfg-rh は、「Type Interface」 の形式で接続プロファイルの名前を作成します。この例では、ブリッジの名前が Bridge br0 であることを意味します。または、NAME=bridge-br0 が ifcfg-br0 ファイルに追加されると、接続プロファイルの名前は bridge-br0 になります。



注記

DEVICE ディレクティブでは、デバイスの種類を判断しないため、ほとんどすべてのインターフェイス名を使用できます。TYPE=Ethernet は必須ではありません。TYPE ディレクティブが設定されていない場合、(名前が明確に異なるインターフェイス設定ファイルと合致していなければ) そのデバイスはイーサネットデバイスとして扱われます。

ディレクティブでは、大文字と小文字は区別されます。

HWADDR ディレクティブを使用してハードウェアまたは MAC アドレスを指定すると、11章ネットワークデバイス命名における一貫性 で説明されているようにデバイスの命名手順に影響します。

警告

リモートホスト上でブリッジ設定をしていて、そのホストへの接続に設定中の物理 NIC を使用している場合、この先に進む前に接続が切断された場合の影響を検討してください。サービスを再起動する際には接続が失われ、エラーが発生すると接続を再確立することができない場合があります。コンソールもしくは帯域外のアクセスが推奨されます。

新規または最近設定したインターフェイスを開くには、

```
ifup device
```

の形式で root でコマンドを実行します。このコマンドは、NetworkManager が実行されているかどうかを検出し、nmcli con load *UUID* を呼び出して、nmcli con up *UUID* を呼び出します。

または、すべてのインターフェイスを再読み込みするには、root で以下のコマンドを実行します。

```
~]# systemctl restart network
```

このコマンドは、ネットワークサービスを停止し、ネットワークサービスを起動してから、ONBOOT=yes の ifcfg ファイルすべてに対して ifup を呼び出します。

注記

デフォルトの動作では、NetworkManager は ifcfg ファイルへの変更を認識せず、インターフェイスが次に起動するまで古い設定データを引き続き使用します。これは、NetworkManager.conf ファイルの monitor-connection-files オプションで設定されます。詳細は、NetworkManager.conf (5) man ページを参照してください。

9.3.3. ボンドを使ったネットワークブリッジ

ボンディングされた 2 つ以上のイーサネットインターフェイスで形成されたネットワークブリッジの例を示します。ボンディングインターフェイスの設定ファイルを理解していない場合は、「チャンネルボンディングインターフェイスの作成」を参照してください。

ボンディングする 2 つ以上のイーサネットインターフェイスの設定ファイルを、以下のように作成または編集します。

```
DEVICE=interface_name
TYPE=Ethernet
SLAVE=yes
MASTER=bond0
BOOTPROTO=none
HWADDR=AA:BB:CC:DD:EE:FF
```

注記

interface_name をインターフェイス名として使用することは一般的ですが、ほとんどすべての名前を使用することができます。

以下のように、インターフェイス設定ファイル /etc/sysconfig/network-scripts/ifcfg-bond0 を作成または編集します。

```
DEVICE=bond0
ONBOOT=yes
BONDING_OPTS='mode=1 miimon=100'
BRIDGE=brbond0
```

ボンディングモジュールの設定に関する指示およびアドバイスとボンディングパラメーターのリストについては、「チャンネルボンディングの使用」を参照してください。

インターフェイス設定ファイル /etc/sysconfig/network-scripts/ifcfg-brbond0 を以下のように作成または編集します。

```
DEVICE=brbond0
ONBOOT=yes
TYPE=Bridge
IPADDR=192.168.1.1
PREFIX=24
```

MASTER=bond0 ディレクティブを持つ 2 つ以上のインターフェイス設定ファイルができました。これらは、DEVICE=bond0 ディレクティブを含む /etc/sysconfig/network-scripts/ifcfg-bond0 という名前の設定ファイルを参照します。この ifcfg-bond0 は、/etc/sysconfig/network-scripts/ifcfg-brbond0 設定ファイルを参照します。これには IP アドレスが含まれ、ホスト内の仮想ネットワークへのインターフェイスとして機能します。

新規または最近設定したインターフェイスを開くには、

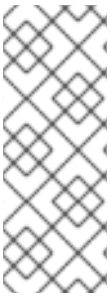
```
ifup device
```

の形式で root でコマンドを実行します。このコマンドは、NetworkManager が実行されているかどうかを検出し、nmcli con load *UUID* を呼び出して、nmcli con up *UUID* を呼び出します。

または、すべてのインターフェイスを再読み込みするには、root で以下のコマンドを実行します。

```
~]# systemctl restart network
```

このコマンドは、ネットワークサービスを停止し、ネットワークサービスを起動してから、ONBOOT=yes の ifcfg ファイルすべてに対して ifup を呼び出します。



注記

デフォルトの動作では、NetworkManager は ifcfg ファイルへの変更を認識せず、インターフェイスが次に起動するまで古い設定データを引き続き使用します。これは、NetworkManager.conf ファイルの monitor-connection-files オプションで設定されます。詳細は、NetworkManager.conf (5) man ページを参照してください。

9.4. GUI を使ったネットワークブリッジングの設定

ブリッジインターフェイスを開始すると、NetworkManager は少なくとも 1 つのポートが 「転送」状態になるのを待ってから、DHCP や IPv6 自動設定などのネットワーク依存 IP 設定を開始します。ポートまたはポートが接続されるか、またはパケットの転送を開始する前に、静的 IP アドレス処理を続行できます。

9.4.1. GUI を使用したブリッジ接続の確立

手順9.1 nm-connection-editor を使用して新規ブリッジ接続の追加

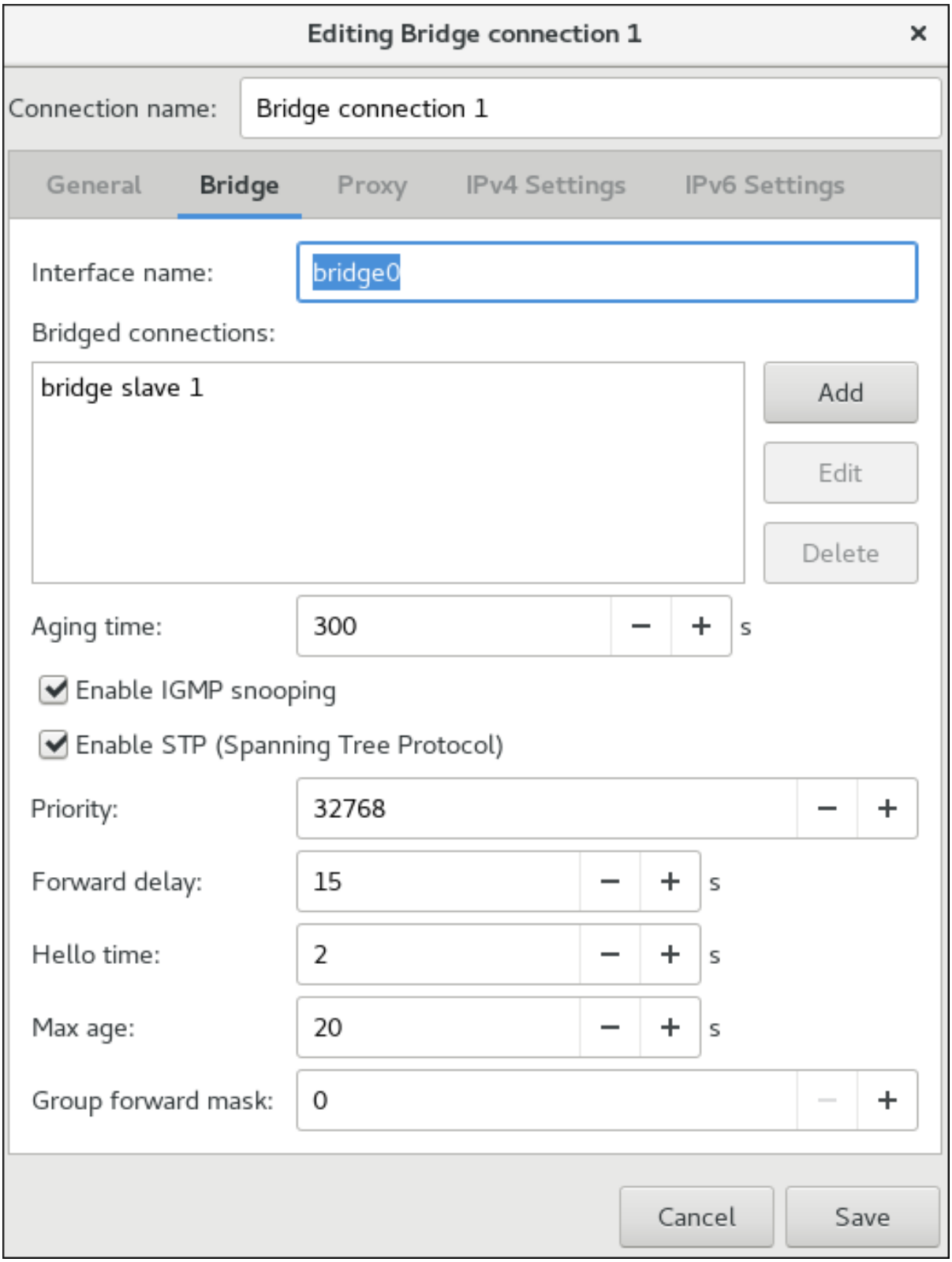
新規ブリッジ接続を作成するには、以下のステップに従います。

1. 端末に nm-connection-editor と入力します。

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

2. **Add ボタンをクリックします。Choose a Connection Type ウィンドウが表示されます。Bridge を選択し、Create を クリックします。ブリッジ 接続 *1*の編集 ウィンドウが表示されます。**

図9.5 ブリッジ接続 1 の編集



[D]

3. 以下の 手順9.3「ブリッジにポートインターフェイスを追加する」 を参照してポートデバイスを追加します。

手順9.2 既存のブリッジ接続を編集する

1. 端末に **nm-connection-editor** と入力します。

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

2. 編集する **Bridge** 接続を選択します。

3. **Edit** ボタンをクリックします。

接続名、自動接続の動作、可用性の設定

編集 ダイアログの 5 つの設定は、すべての接続の種類に共通です。全般 タブ：

- **接続名: ネットワーク接続のわかりやすい名前を入力します。この名前は、Network ウィンドウのメニューでこの接続を一覧表示するために使用されます。**

- **Automatically connect to this network when it is available - このボックスを選択すると、NetworkManager が利用可能なときにこの接続に自動接続します。詳細は、「control-center を使用した既存の接続の編集」を参照してください。**

- **All users can connect to this network - このボックスを選択すると、システム上のすべてのユーザーが 利用できる接続が作成されます。この設定を変更するには、root 権限が必要になる場合があります。詳細は、「GUI を使用したシステム全体およびプライベート接続プロファイルの管理」を参照してください。**

- **Automatically connect to VPN when using this connection - このボックスを選択すると、NetworkManager が利用可能なときに VPN 接続に自動接続します。ドロップダウンメニューから VPN を選択します。**

- **ファイアウォールゾーン - ドロップダウンメニューからファイアウォールゾーンを選択します。ファイアウォールゾーンに関する詳細情報は、『Red Hat Enterprise Linux 7 セキュリティーガイド』を参照してください。**

9.4.1.1. ブリッジタブを設定する

インターフェイス名

ブリッジへのインターフェイス名。

ブリッジ接続

1 つ以上のポートインターフェイス。

エージング時間

MAC アドレスが MAC アドレス転送データベースに保持される時間 (秒単位)。

IGMP スヌーピングのを有効化

必要に応じて、チェックボックスをオンにして、デバイスで IGMP スヌーピングを有効にします。

STP (スパニングツリープロトコル) を有効化

必要に応じて、チェックボックスを選択して STP を有効にします。

優先度

ブリッジの優先度。優先度の一番低いブリッジが root ブリッジに選ばれます。

転送遅延

転送状態に入るまでのリスニングと状態確認の両方に費やされる秒数。デフォルトは 15 秒です。

Hello タイム

ブリッジプロトコルデータ単位 (BPDU) で設定情報を送信する間隔 (秒単位)。

最大エージ

BPDU カラの設定情報を保存する最大秒数。この値は Hello タイムを 2 倍したものに 1 を加えたものになりますが、転送遅延を 2 倍したものから 1 を引いたものよりも少なくなる必要があります。

グループ転送マスク

このプロパティーは、グループアドレスの転送を許可するグループアドレスのマスクです。ほ

とんどの場合、01 :80:C2: 00:00:00 から 01:80:C2:00:00:0F の範囲のグループアドレスは、ブリッジデバイスによって転送されません。このプロパティーは、16 ビットのマスクで、それぞれ上記の範囲内のグループアドレスに対応しており、転送する必要があります。グループ転送マスク プロパティーは、スパニングツリープロトコル(STP)、リンクアグリゲーション制御プロトコル(LACP)、およびイーサネット MAC 一時停止フレームに使用されるため、0、1 、2 ビットを 1 に設定できないことに注意してください。

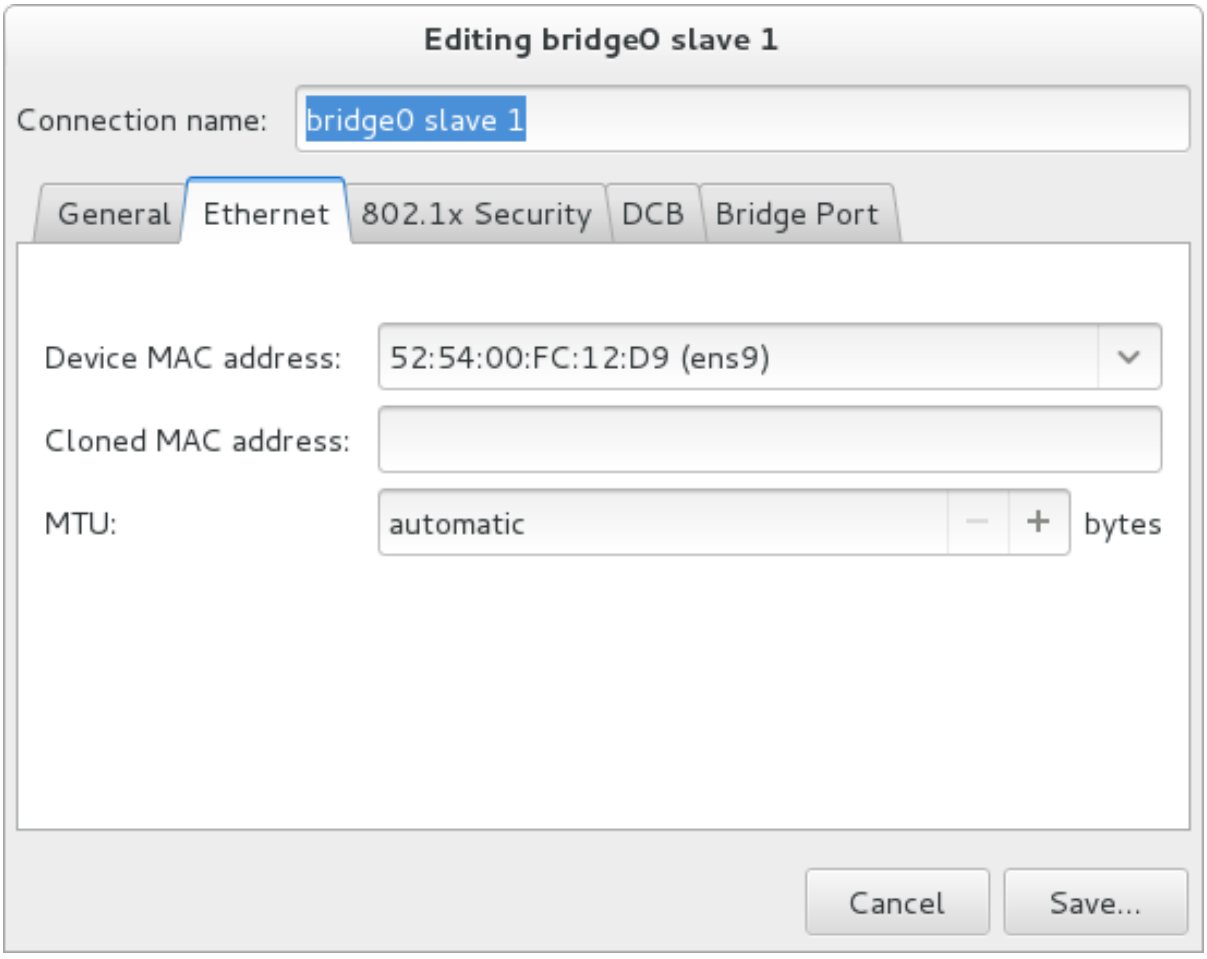
手順9.3 ブリッジにポートインターフェイスを追加する

1. ブリッジにポートを追加するには、ブリッジ 接続 *1*の編集 ウィンドウで Bridge タブを選択します。必要な場合は、手順9.2「既存のブリッジ接続を編集する」の手順に従ってこのウィンドウを開きます。

2. Add をクリックします。接続の種類の選択 メニューが表示されます。

3. リストから作成する接続の種類を選択します。Create をクリックします。選択した接続タイプに該当するウィンドウが表示されます。

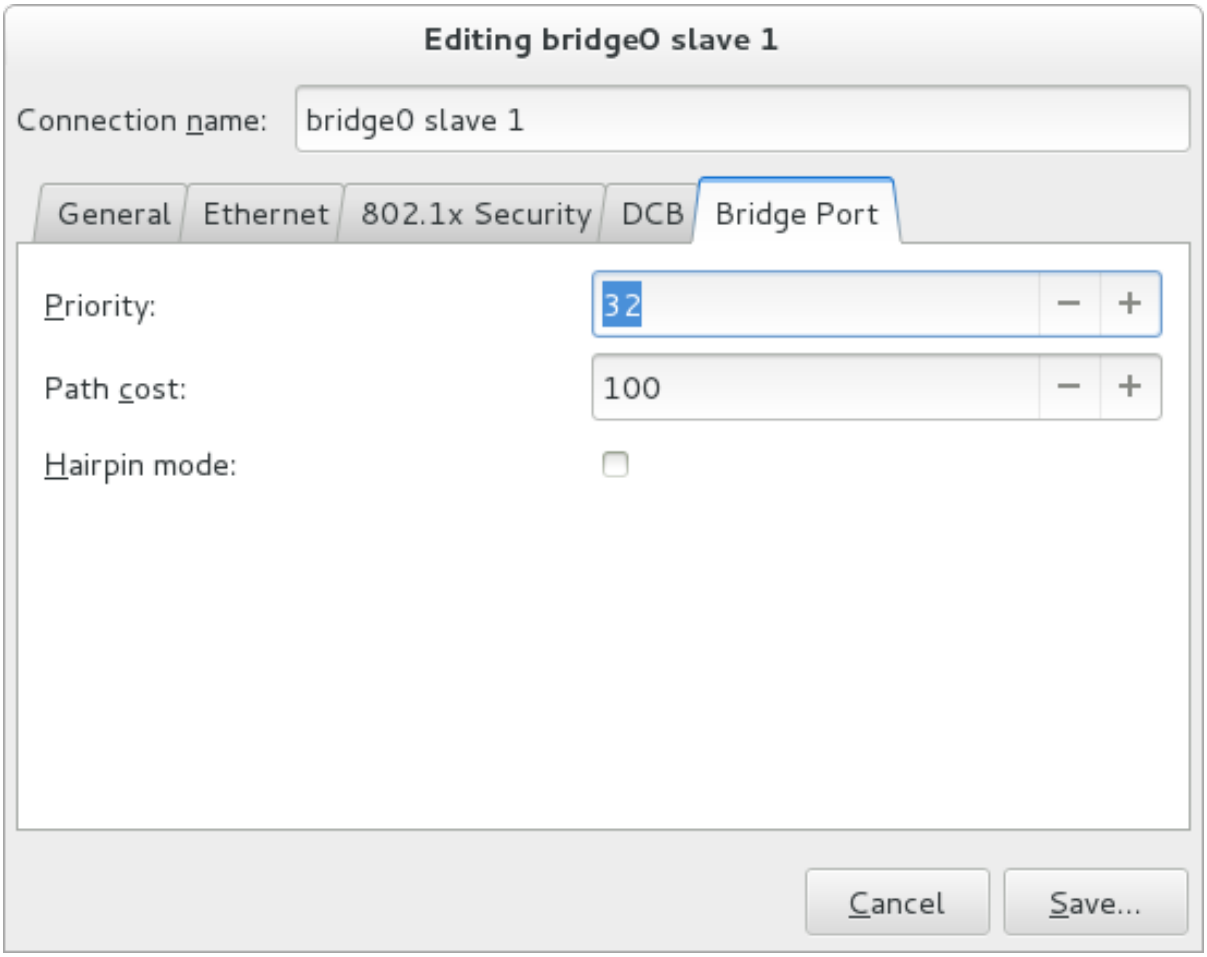
図9.6 NetworkManager グラフィカルユーザーインターフェイスのブリッジ接続追加



[D]

4. Bridge Port タブを選択します。必要に応じて Priority and Path cost を設定します。ブリッジポートの STP 優先度は Linux カーネルで制限されていることに注意してください。標準では 0 から 255 の範囲が許可されますが、Linux では 0 から 63 までしか許可されません。この場合、デフォルトは 32 です。

図9.7 NetworkManager グラフィカルユーザーインターフェイスでのブリッジポートタブ



[D]

5. 必要な場合は、Hairpin mode チェックボックスを選択して、外部処理用のフレームの転送を有効にします。これは、*仮想イーサネットポートアグリゲーター* (VEPA) モードとも呼ばれます。

そして、以下のいずれかの設定をします。

- イーサネットポートの場合は Ethernet タブをクリックして「基本設定オプション」に進みます。
- ボンディングポートの場合は、Bond タブをクリックして「Bond タブの設定」に進みます。または、

- チームポートの場合は Team タブをクリックして 「チームタブの設定」 に進みます。

- VLAN ポートの場合は VLAN タブをクリックして 「VLAN タブの設定」 に進みます。

新規 (または修正した) 接続を保存して他の設定を行う

新しいブリッジ接続の編集が終わったら、保存 ボタンをクリックしてカスタマイズした設定を保存します。編集中にプロファイルが使用されていた場合には、接続の電源を入れ直して、NetworkManager が変更を適用するようにします。プロファイルがオフだった場合は、これをオンにするか、ネットワーク接続アイコンメニューで選択します。新規および変更後の接続を使用することに関する詳細情報は、「control-center GUI を使用したネットワーク接続」を参照してください。

既存の接続をさらに設定するには、Network ウィンドウでその接続を選択し、Options をクリックして Editing ダイアログに戻ります。

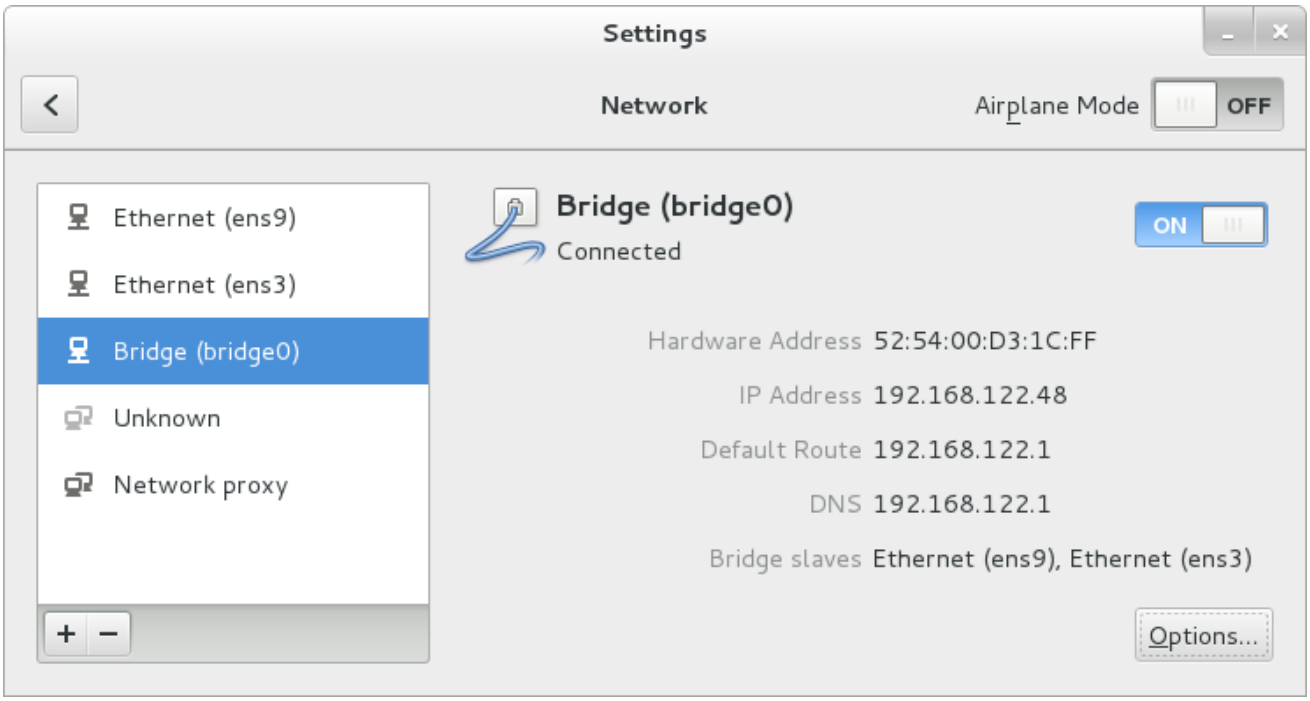
そして、以下のいずれかの設定をします。

- IPv4 の設定は、IPv4 のセッティング タブをクリックして 「IPv4 設定の設定」 に進みます。または、

- IPv6 の設定は、IPv6 のセッティング タブをクリックして、「IPv6 セッティングの設定」 に進みます。

保存が完了すると、ブリッジはポートとともにネットワーク設定ツール画面に表示されます。

図9.8 NetworkManager グラフィカルユーザーインターフェイスでのブリッジ



[D]

9.5. IPROUTE を使用したイーサネットブリッジの設定

bridge-utils の代わりに、iproute パッケージを使用できます。優先度、コスト、状態 などのブリッジポートオプションを設定できます。

ip ユーティリティーを使用して、ブリッジデバイスに割り当てられたインターフェイス *enp1s0* にポートオプションを設定するには、root で以下のコマンドを実行します。

```
~]# ip link set enp1s0 type bridge_slave option
```

ip ユーティリティーを使用して利用可能なオプションを選択するには、root で以下のコマンドを実行します。

```
~]# 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} ]
```

ポートオプションの詳細は、man ページの ip-link (8) を参照してください。

9.6. 関連情報

- nmcli (1) man ページ - NetworkManager のコマンドラインツールを説明しています。
- nmcli-examples (5) man ページ - nmcli コマンドの例を説明します。
- nm-settings (5) man ページ - NetworkManager 接続の設定およびパラメーターが説明されています。
- ip-link (8) man ページ - ブリッジポートオプションの説明。

第10章 802.1Q VLAN タグの設定

VLAN を作成するには、*親インターフェイス* と呼ばれるインターフェイスの上に別のインターフェイスを作成します。VLAN インターフェイスは、パケットがインターフェイスを通過する際に VLAN ID でタグ付けし、返信パケットの場合はタグを外します。VLAN インターフェイスは他のインターフェイスと同様に設定することができます。親インターフェイスはイーサネットインターフェイスである必要はありません。802.1Q VLAN タグインターフェイスは、ブリッジ、ボンド、およびチームインターフェイスの上に作成することができますが、以下の点に注意してください。

- ボンド上に VLAN を作成した場合は、ボンドにポートがあり、VLAN インターフェイスをアクティブにする前にそれらが 「up」 になっていることが重要です。ポートのないボンドに VLAN インターフェイスを追加しても機能しません。

- VLAN 仮想デバイスは、親の新規 MAC アドレスに一致するように MAC アドレスを変更できないため、fail_over_mac=follow オプションが指定されたボンディングで VLAN ポートを設定することはできません。この場合、トラフィックは間違ったソースの MAC アドレスで送信されます。

- VLAN のタグ付けがされたパケットをネットワークスイッチ経由で送信するには、スイッチを適切に設定する必要があります。たとえば、複数の VLAN からタグ付けされたパケットを受け付けるには、Cisco スイッチ上のポートは 1 つの VLAN に割り当てられているか、トランクポートになるように設定されている必要があります。一部ベンダーのスイッチでは、トランクポートが *ネイティブ VLAN* のタグ付けされていないフレームを処理することが許されます。一部のデバイスでは、*ネイティブ VLAN* を有効または無効にすることができますが、他のデバイスでは、デフォルトでは無効になっています。この相違が原因となり、異なる 2 つのスイッチ間で *ネイティブ VLAN* の誤設定が生じ、セキュリティーリスクが発生する可能があります。以下に例を示します。

  あるスイッチは *ネイティブ VLAN 1* を使用し、他のスイッチは *ネイティブ VLAN 10* を使用するとします。タグが挿入されずにフレームの通過が許可されると、攻撃者は VLAN 間をジャンプすることができます。この一般的なネットワーク侵入方法は、*VLAN ホッピング* とも呼ばれています。

  セキュリティーリスクを最小限に抑えるためには、インターフェイスを以下のように設定します。

  スイッチ

  - 必要でない限り、トランクポートを無効にする。

- トランクポートが必要な場合は、タグ付けされていないフレームが許可されないように *ネイティブ VLAN* を無効にする。

Red Hat Enterprise Linux サーバー

- nftables ユーティリティーまたは ebtables ユーティリティーを使用して、入力フィルターリングでタグ付けされていないフレームをドロップします。

- 古いネットワークインターフェイスカードやループバックインターフェイス、Wimax カードや InfiniBand デバイスのなかには、*VLAN 非対応* といって、VLAN をサポートできないものもあります。これは通常、これらのデバイスがタグ付けされたパケットに関連する VLAN ヘッダーや大きい MTU サイズに対応できないためです。

注記

Red Hat では、VLAN 上へのボンディング設定をサポートしていません。詳細については、Red Hat ナレッジベースの記事 『VLAN 上にポートインターフェイスとしてボンディングを設定することは有効か』 を参照してください。

10.1. VLAN インターフェイス設定方式の選択

- NetworkManager のテキストユーザーインターフェイスツール nmtui を使用して VLAN インターフェイスを設定するに は、に進みます。 「テキスト形式のユーザーインターフェイス nmtui を使った 802.1Q VLAN タグの設定」

- NetworkManager のコマンドラインツール nmcli を使用して VLAN インターフェイスを設定するに は、に進みます。 「コマンドラインツール nmcli を使った 802.1Q VLAN タグの設定」

- ネットワークインターフェイスを手動で設定するには、「コマンドラインを使用した 802.1Q VLAN タグの設定」を参照してください。

- グラフィカルユーザーインターフェイスツールを使ってネットワークを設定するには、「GUI を使用した 802.1Q VLAN タグの設定」に進みます。

10.2. テキスト形式のユーザーインターフェイス NMTUI を使った 802.1Q VLAN タグの設定

テキスト形式のユーザーインターフェイスツール nmtui を使用すると、ターミナルのウィンドウで 802.1Q VLAN を設定できます。このツールを起動するには、以下のコマンドを実行します。

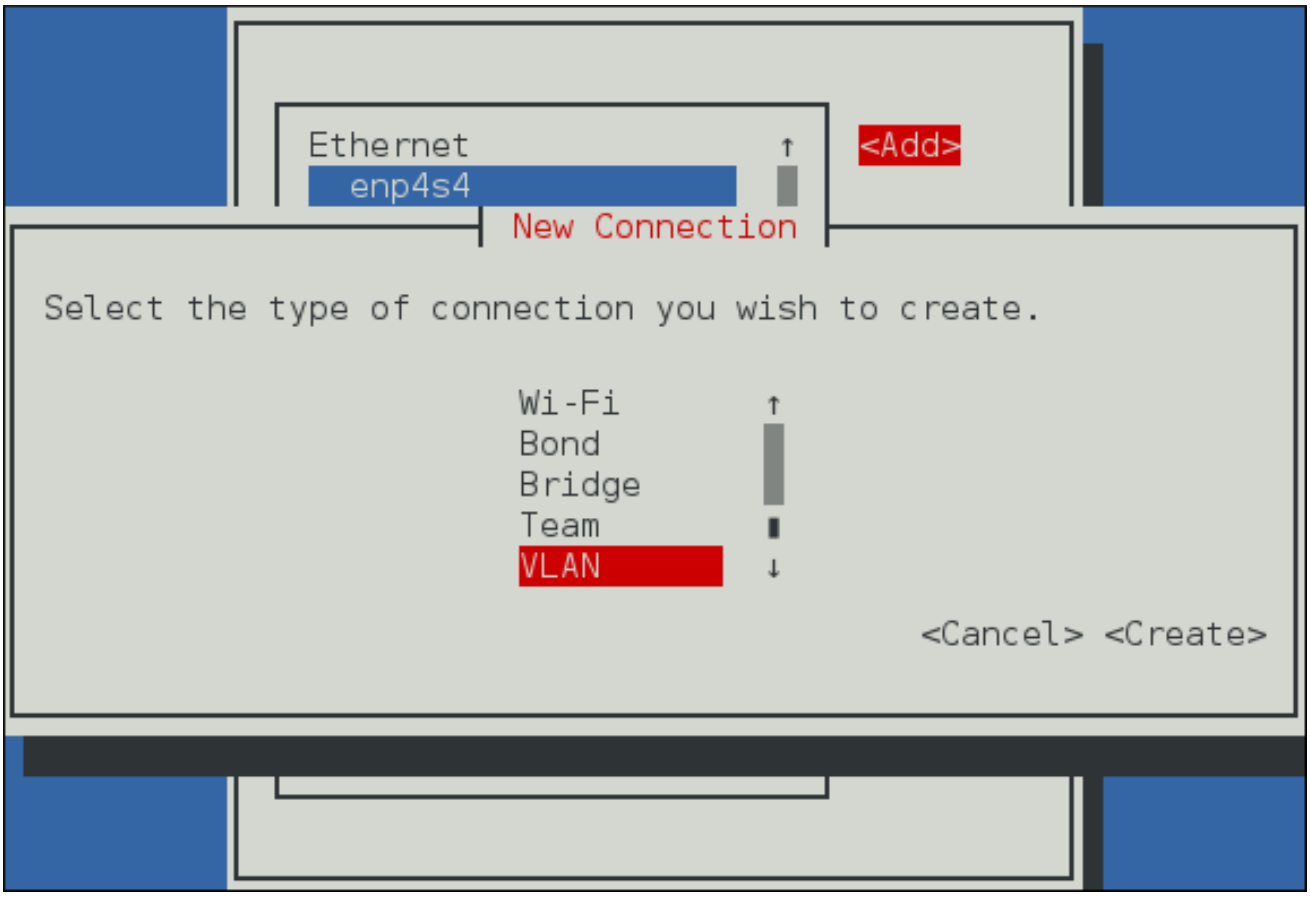
```
~]$ nmtui
```

テキストユーザーインターフェイスが表示されます。無効なコマンドがあると、使用方法に関するメッセージが表示されます。

移動するには、矢印キーを使用するか、Tab を押して順方向に進み、Shift+Tab を押してオプションを再度実行します。Enter を押してオプションを選択します。Space バーは、チェックボックスのステータスを切り替えます。

メニューから 接続の編集 を選択します。Add を選択すると、New Connection 画面が開きます。

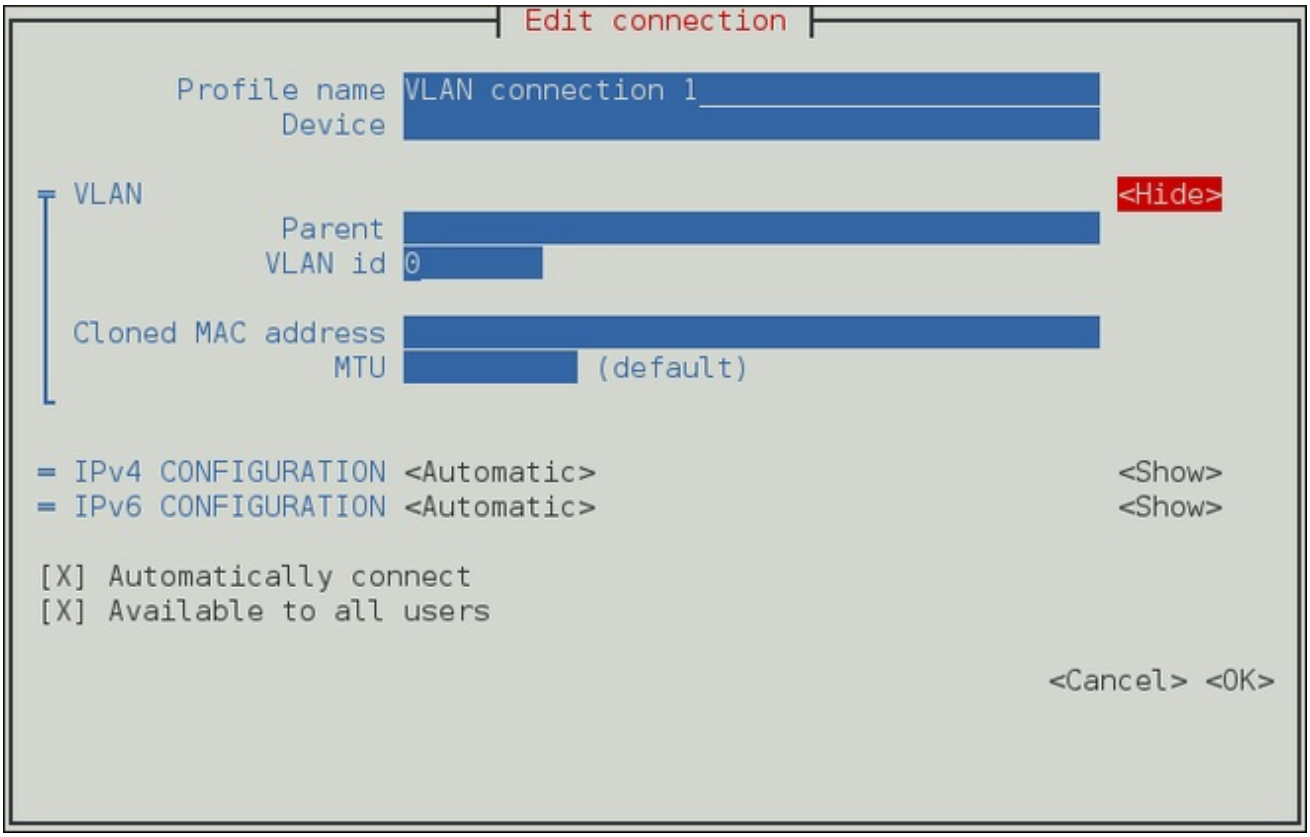
図10.1 NetworkManager テキスト形式のユーザーインターフェイスの VLAN 接続追加メニュー



[D]

VLAN を選択すると、接続の編集 画面が開きます。画面のプロンプトに従って設定を完了します。

図10.2 NetworkManager テキスト形式ユーザーインターフェイスで VLAN 接続を設定するメニュー



[D]

VLAN 用語の定義は、「VLAN タブの設定」を参照してください。

nmtui のインストール方法は、「nmtui を使用した IP ネットワークの設定」 を参照してください。

10.3. コマンドラインツール NMCLI を使った 802.1Q VLAN タグの設定

システム上で利用可能なインターフェイスを表示するには、以下のコマンドを実行します。

```
~]$ 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
```

出力の NAME フィールドは常に接続 ID を表すことに留意してください。これはインターフェイス名と同じように見えますが、異なるものです。nmcli connection コマンドで ID を使用して接続を識別できます。firewalld などの他のアプリケーションで DEVICE 名を使用します。

VLAN インターフェイス *VLAN10* および ID 10 のイーサネットインターフェイス *enp1s0* で 802.1Q VLAN インターフェイスを作成するには、以下のコマンドを実行します。

■

```
~]$ nmcli con add type vlan ifname VLAN10 dev enp1s0 id 10
Connection 'vlan-VLAN10' (37750b4a-8ef5-40e6-be9b-4fb21a4b6d17) successfully added.
```

VLAN インターフェイスに **con-name** を指定しなかったため、接続名がインターフェイス名の前に種類を追加したものとなっていることに留意してください。別の方法では、以下のように **con-name** オプションで名前を指定します。

```
~]$ nmcli con add type vlan con-name VLAN12 dev enp1s0 id 12
Connection 'VLAN12' (b796c16a-9f5f-441c-835c-f594d40e6533) successfully added.
```

VLAN インターフェイスにアドレスを割り当てる

同じ **nmcli** コマンドを使用して、他のインターフェイスと同様に静的および動的インターフェイスアドレスを割り当てることができます。

たとえば、静的 **IPv4** アドレスとゲートウェイを持つ **VLAN** インターフェイスを作成するコマンドは以下のようになります。

```
~]$ nmcli con add type vlan con-name VLAN20 dev enp1s0 id 20 ip4 10.10.10.10/24 \
gw4 10.10.10.254
```

動的アドレスを割り当てる **VLAN** インターフェイスを作成するには、以下のコマンドを実行します。

```
~]$ nmcli con add type vlan con-name VLAN30 dev enp1s0 id 30
```

nmcli コマンドを使用してインターフェイスを設定する例は、「nmcli を使用したネットワーク接続」 を参照してください。

作成した **VLAN** インターフェイスを表示するには、以下のコマンドを実行します。

```
~]$ 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
```

新規に設定した接続の詳細情報を表示するには、以下のコマンドを実行します。

```
~]$ 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 コマンドのその他のオプションは、nmcli (1) man ページの VLAN セクションに記載されています。man ページでは、VLAN が作成されるデバイスは、parent device と呼ばれています。上記の例では、デバイスはインターフェイス名 enp1s0 で指定されており、接続 UUID または MAC アドレスで指定することもできます。

イーサネットインターフェイス *np2s0* 上で、イングレス優先度マッピングを使用して 802.1Q VLAN 接続プロファイル (名前 VLAN1 ID 13 で、以下のコマンドを実行します。

```
~]$ nmcli con add type vlan con-name VLAN1 dev enp2s0 id 13 ingress "2:3,3:5"
```

上記で作成された VLAN に関連するすべてのパラメーターを表示するには、以下のコマンドを実行します。

```
~]$ nmcli connection show vlan-VLAN10
```

MTU を変更するには、以下のコマンドを実行します。

```
~]$ nmcli connection modify vlan-VLAN10 802.mtu 1496
```

MTU 設定は、ネットワーク層パケットの最大サイズを決定します。リンク層フレームが送信可能なペイロードの最大サイズは、ネットワーク層 MTU を制限します。通常のイーサネットフレームの場合、1500 バイトの MTU になります。VLAN 設定の際には、802.1Q タグを受け入れるためにリンク層ヘッダーのサイズが 4 バイト拡大されるので、MTU を変更する必要はありません。

執筆時点では、connection.interface-name と vlan.interface-name は同じでなければなりません（設定されている場合）。したがって、nmcli のインタラクティブモードを使用して同時に変更する必要があります。VLAN 接続名を変更するには、以下のコマンドを実行します。

```
~]$ nmcli con edit vlan-VLAN10
nmcli> set vlan.interface-name superVLAN
nmcli> set connection.interface-name superVLAN
nmcli> save
nmcli> quit
```

nmcli ユーティリティーを使用すると、802.1Q コードの動作を変更する ioctl フラグを設定および

消去できます。次の VLAN フラグが NetworkManager でサポートされています。

- 0x01 - 出力パケットヘッダーを並び替えます。
- 0x02 - GVRP プロトコルを使用します。
- 0x04 - インターフェイスとそのマスターのバインディングを外します。

VLAN の状態は、親もしくはマスターインターフェイス (VLAN が作成されているインターフェイスもしくはデバイス) に同期されます。親インターフェイスが「down」の管理状態に設定されていると、関連するすべての VLAN もダウンに設定され、ルートもすべてルーティングテーブルからフラッシュされます。フラグ 0x04 は、*loose binding* モードを有効にします。このモードでは、動作状態のみが親から関連付けられた VLAN に渡されますが、VLAN デバイスの状態は変更されません。

VLAN フラグを設定するには、以下のコマンドを実行します。

```
~]$ nmcli connection modify vlan-VLAN10 vlan.flags 1
```

nmcli の概要は、「nmcli を使用する IP ネットワークの設定」 を参照してください。

10.4. コマンドラインを使用した 802.1Q VLAN タグの設定

Red Hat Enterprise Linux 7 では、8021q モジュールはデフォルトでロードされています。必要に応じて、root で以下のコマンドを実行して、モジュールがロードされていることを確認することができます。

```
~]# modprobe --first-time 8021q
modprobe: ERROR: could not insert '8021q': Module already in kernel
```

モジュールについての情報を表示するには、以下のコマンドを実行します。

```
~]$ modinfo 8021q
```

コマンドオプションについては、modprobe (8) の man ページを参照してください。

10.4.1. ifcfg ファイルを使用した 802.1q VLAN タグの設定

1. **/etc/sysconfig/network-scripts/ifcfg-*device_name* で親インターフェイスを設定します。*device_name* はインターフェイスの名前です。**

```
DEVICE=interface_name
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
```

2. **/etc/sysconfig/network-scripts/ ディレクトリーで VLAN インターフェイスを設定します。設定ファイル名は、親インターフェイスに . 文字と VLAN ID 番号を加えたものにする必要があります。たとえば、VLAN ID が 192 で、親インターフェイスが *enp1s0* の場合、設定ファイル名は ifcfg-enp1s0.192 になります。**

```
DEVICE=enp1s0.192
BOOTPROTO=none
ONBOOT=yes
IPADDR=192.168.1.1
PREFIX=24
NETWORK=192.168.1.0
VLAN=yes
```

同じインターフェイス enp1s0 で 2 つ目の VLAN (VLAN ID 193 など)を設定する必要がある場合は、VLAN 設定の詳細で *enp1s0*.193 という名前の新しいファイルを追加します。

3. **変更を反映させるには、ネットワークサービスを再起動します。root で以下のコマンドを実行します。**

```
~]# systemctl restart network
```

10.4.2. ip コマンドを使用した 802.1Q VLAN タグの設定

イーサネットインターフェイス *enp1s0* に 802.1Q VLAN インターフェイスを作成し、名前が *VLAN8* および ID 8 の場合は、root で以下のコマンドを実行します。

```
~]# ip link add link enp1s0 name enp1s0.8 type vlan id 8
```

VLAN を表示するには、以下のコマンドを実行します。

```
~]$ 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 ユーティリティーは、VLAN ID が 0x で始まる場合、および 8 進数値(0 の場合)として、VLAN ID を 16 進数として解釈することに注意してください。つまり、10 進数の値が 22 の VLAN ID を割り当てるには、ゼロを追加しないでください。

VLAN を削除するには、root で以下のコマンドを実行します。

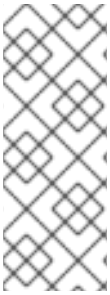
```
~]# ip link delete enp1s0.8
```

複数の VLAN に属する複数のインターフェイスを使用するには、物理インターフェイス ***enp1s0*** に、適切な VLAN ID を持つローカルの ***enp1s0.1*** と ***enp1s0.2*** を作成します。

```
~]# 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
```

物理デバイスでネットワークスニファーを実行すると、***enp1s0*** に VLAN デバイスが設定されていなくても、物理デバイスに到達するタグ付きフレームをキャプチャーできます。以下に例を示します。

```
tcpdump -nnei enp1s0 -vvv
```



注記

ip コマンドを使用して作成された VLAN インターフェイスは、システムがシャットダウンまたは再起動されると失われます。システム再起動後に VLAN インターフェイスを永続化するように設定するには、ifcfg ファイルを使用します。「ifcfg ファイルを使用した 802.1q VLAN タグの設定」 を参照

10.5. GUI を使用した 802.1Q VLAN タグの設定

10.5.1. VLAN 接続の確立

nm-connection-editor を使用して、既存のインターフェイスを親インターフェイスとして使用して VLAN を作成できます。 VLAN デバイスが自動的に作成されるのは、親インターフェイスが自動的に接続するよう設定されている場合のみであることに注意してください。

手順10.1 nm-connection-editor を使用して新規 VLAN 接続を追加する

1. 端末に nm-connection-editor と入力します。

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

2. Add ボタンをクリックします。Choose a Connection Type ウィンドウが表示されます。VLAN を選択し、Create を クリックします。VLAN 接続 *1*の編集 ウィンドウが表示されます。

3. VLAN タブで、VLAN 接続に使用する親インターフェイスをドロップダウンリストから選択します。

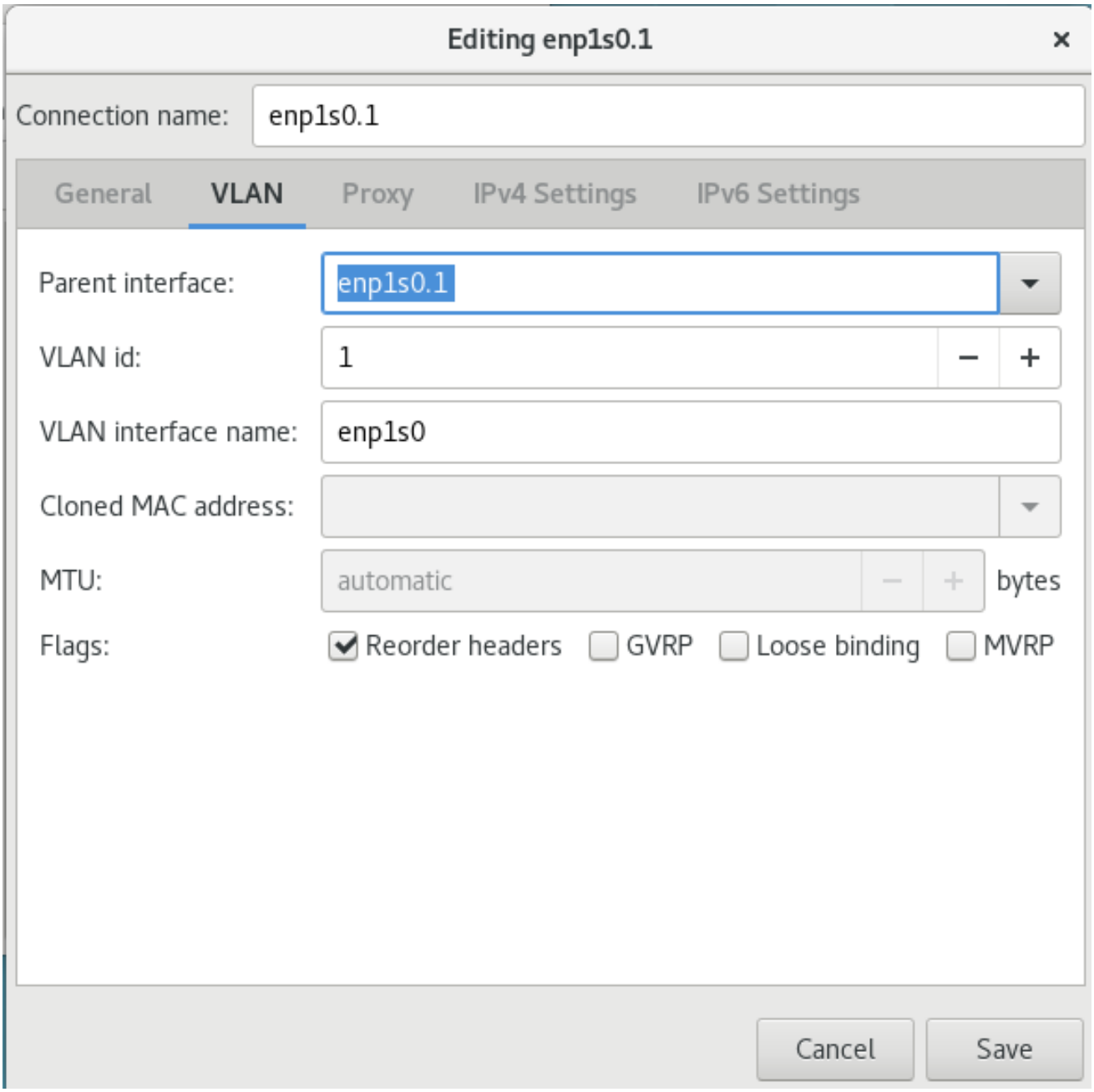
4. VLAN ID を入力します。

5. VLAN インターフェイス名を入力します。これは、作成される VLAN インターフェイスの名前です。たとえば、enp1s0.1 または vlan2 などです。（通常、これは親インターフェイス名に 「.」 と VLAN ID を加えたか、「vlan」 に VLAN ID を加えたものになります。）

6. 設定を確認してから Save ボタンをクリックします。

7. VLAN 固有の設定を編集するには、「VLAN タブの設定」を参照してください。

図10.3 nm-connection-editor を使用して新規 VLAN 接続を追加する



[D]

手順10.2 既存の VLAN 接続を編集する

既存の VLAN 接続を編集するには以下の手順に従います。

1. **端末に nm-connection-editor と入力します。**

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

2. 編集する接続を選択し、Edit ボタンをクリックします。

3. General タブを選択します。

4. 接続名、自動接続の動作、および可用性のセッティングを設定します。

   編集 ダイアログの これらの設定は、すべての接続の種類に共通です。

   - 接続名: ネットワーク接続のわかりやすい名前を入力します。この名前は、ネットワーク ウィンドウの VLAN セクションでこの接続を一覧表示するために使用されます。

   - Automatically connect to this network when it is available - このボックスを選択すると、NetworkManager が利用可能なときにこの接続に自動接続します。詳細は、「control-center を使用した既存の接続の編集」を参照してください。

   - Available to all users - このボックスを選択して、システム上のすべてのユーザーが利用できる接続を作成します。この設定を変更するには、root 権限が必要になる場合があります。詳細は、「GUI を使用したシステム全体およびプライベート接続プロファイルの管理」を参照してください。

5. VLAN 固有の設定を編集するには、「VLAN タブの設定」を参照してください。

新規 (または修正した) 接続を保存して他の設定を行う

VLAN 接続の編集が終わったら、保存 ボタンをクリックしてカスタマイズした設定を保存します。

そして、以下のいずれかの設定をします。

- IPv4 の設定は、IPv4 のセッティング タブをクリックして、「IPv4 設定の設定」 に進みます。

  または

-

IPv6 の設定は、IPv6 のセッティング タブをクリックして、「IPv6 セッティングの設定」に進みます。

10.5.1.1. VLAN タブの設定

新しい VLAN 接続をすでに追加している場合(手順10.1「nm-connection-editor を使用して新規 VLAN 接続を追加する」 を参照)、VLAN タブを編集して親インターフェイスと VLAN ID を設定できます。

親インターフェイス

ドロップダウンリストから以前に設定したインターフェイスを選択できます。

VLAN ID

VLAN ネットワークのトラフィックのタグ付けに使用する ID 番号。

VLAN インターフェイス名

作成される VLAN インターフェイスの名前。たとえば、enp1s0.1 または vlan2 などです。

クローンした MAC アドレス

VLAN インターフェイスの特定に使用する別の MAC アドレスをオプションで設定します。このアドレスを使って、この VLAN 上で送信されたパケットのソース MAC アドレスを変更することができます。

MTU

VLAN 接続で送信されるパケットに使用する最大転送単位 (MTU) のサイズをオプションで設定します。

10.6. IP コマンドを使用したボンドおよびブリッジ上での VLAN の使用

ボンドおよびブリッジ上で VLAN を使用するには、以下の手順を実施します。

1. root としてボンドデバイスを追加します。

```
# 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. ボンドデバイス上に VLAN を設定します。

```
# ip link add link bond0 name bond0.2 type vlan id 2
# ip link set bond0.2 up
```

3. ブリッジデバイスを追加し、そこに VLAN をアタッチします。

```
# ip link add br0 type bridge
# ip link set bond0.2 master br0
# ip link set br0 up
```

10.7. NETWORKMANAGER のコマンドラインツール NMCLI の使用した、ボンドおよびブリッジの VLAN

ボンドおよびブリッジ上で VLAN を使用するには、以下の手順を実施します。

1. ボンドデバイスを追加します。

```
~]$ nmcli connection add type bond con-name Bond0 ifname bond0 bond.options
"mode=active-backup,miimon=100" ipv4.method disabled ipv6.method ignore
```

この場合、ボンド接続は、VLAN の下位インターフェイスとしてのみ機能し、IP アドレスを取得しません。したがって、**ipv4.method disabled** パラメーターおよび **ipv6.method ignore** パラメーターがコマンドラインに追加されました。

2. ボンドデバイスにポートを追加します。

```
~]$ 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. **ブリッジデバイスを追加します。**

```
~]$ nmcli connection add type bridge con-name Bridge0 ifname br0 ipv4.method manual
ipv4.addresses 192.0.2.1/24
```

4. **ブリッジデバイスに割り当てられたボンドに、VLAN インターフェイスを追加します。**

```
~]$ nmcli connection add type vlan con-name Vlan2 ifname bond0.2 dev bond0 id 2 master
br0 slave-type bridge
```

5. **作成した接続を表示します。**

```
~]$ 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 スイッチポートモードの設定

Red Hat Enterprise Linux マシンは、多くの場合はルーターとして使用され、ネットワークインターフェイスで高度な VLAN 設定を可能にします。イーサネットインターフェイスがスイッチに接続し、物理インターフェイスを介して VLAN を実行している場合は、switchport モード を設定する必要があります。Red Hat Enterprise Linux サーバーまたはワークステーションは、通常は 1 つの VLAN にのみ接続されているため、switchport mode access を適切なものにし、デフォルト設定にします。

特定のシナリオでは、複数のタグ付き VLAN が同じ物理リンク、つまり、スイッチと Red Hat

Enterprise Linux マシンとの間のイーサネットを使用するため、両端で switchport mode trunk を設定する必要があります。

たとえば、Red Hat Enterprise Linux マシンをルーターとして使用する場合、マシンは、ルーターの背後にあるさまざまな VLAN から同じ物理イーサーネットを介して、スイッチにタグ付きパケットを転送し、その VLAN 間の分離を維持する必要があります。

「コマンドラインツール nmcli を使った 802.1Q VLAN タグの設定」 などで説明されている設定では、Cisco スイッチポートモードトランク を使用します。インターフェイスに IP アドレスのみを設定する場合は、Cisco スイッチポートモードによるアクセス を使用します。

10.9. 関連情報

- ip-link (8) man ページ - ip ユーティリティーのネットワークデバイス設定コマンドが説明されています。
- nmcli (1) man ページ - NetworkManager のコマンドラインツールを説明しています。
- nmcli-examples (5) man ページ - nmcli コマンドの例を説明します。
- nm-settings (5) man ページ - NetworkManager 接続の設定およびパラメーターが説明されています。
- nm-settings-ifcfg-rh (5) man ページ - /etc/sysconfig/network-scripts/ifcfg-* ファイルの ifcfg-rh 設定が説明されています。

第11章 ネットワークデバイス命名における一貫性

Red Hat Enterprise Linux は、ネットワークインターフェイス用に一貫した予想可能なネットワークデバイス命名の方法を提供します。この機能では、インターフェイスの位置判定と区別が容易になるようにシステム上のネットワークインターフェイス名を変更します。

従来の Linux のネットワークインターフェイスには次のように列挙されます。 eth[0123...]s0ただし、この名前がシャーシの実際のラベルに対応しているとは限りません。複数のネットワークアダプターを使用する最新のサーバープラットフォームでは、このインターフェイスの非決定論的および反直感的な命名が行われています。これは、マザーボードに組み込まれたネットワークアダプター (*Lan-on-Motherboard* または *LOM*) とアドイン (シングルおよびマルチのポート) アダプターの両方に影響します。

Red Hat Enterprise Linux では、udev は多くの異なる命名スキームをサポートしています。デフォルトでは、ファームウェア、トポロジー、および場所情報に基づいて固定名が割り当てられます。これには、名前が完全に自動的かつ予想可能であり、ハードウェアが追加もしくは削除されても (再列挙がなされず) 固定のままであり、またハードウェアが壊れた場合にシームレスに交代可能であるという利点があります。マイナス面は、従来使用されていた eth または wla といった名前と比べて読みにくい場合があるという点です。以下に例を示します。enp5s0。

警告

Red Hat は、一貫性のあるデバイス命名が無効になっているシステムをサポートしていません。詳細は、Is it safe to set net.ifnames=0? を参照してください。

11.1. 命名スキームの序列

デフォルトでは、systemd は以下のポリシーを使用してサポート対象の命名スキームを適用するインターフェイスに名前を付けます。

- スキーム 1: ファームウェアまたは BIOS が提供するオンボードデバイスのインデックス番号を組み込んだ名前（例： eno1）は、ファームウェアまたは BIOS からの情報が適用可能かつ利用可能な場合に適用されます。そうでない場合は、スキーム 2 にフォールバックします。

- スキーム 2: ファームウェアまたは BIOS が提供する PCI Express ホットプラグスロットインデックス番号（例： ens1）を組み込んだ名前は、ファームウェアまたは BIOS からの情報

が適用可能かつ利用可能な場合に適用されます。そうでない場合は、スキーム 3 にフォールバックします。

- スキーム 3: ハードウェアのコネクターの物理的な場所を組み込む名前（例： enp2s0）は、該当する場合に適用されます。そうでない場合は、スキーム 5 にフォールバックします。

- スキーム 4: インターフェイスの MAC アドレス（例： enx78e7d1ea46da）を組み込む名前はデフォルトで使用されていませんが、ユーザーが選択した場合は利用できます。

- スキーム 5: 従来の予測不可能なカーネル命名スキームは、他のすべての方法が失敗した場合に使用されます（例： eth0）。

上記のポリシーがデフォルトになります。システムで biosdevname が有効になっている場合は、これが使用されます。biosdevname を有効にするには、Dell システムの場合を除き、biosdevname= 1 をカーネルコマンドラインパラメーターとして渡す必要があります。ここで、biosdevname がデフォルトで使用されます。ユーザーがカーネルデバイスの名前を変更する udev ルールを追加している場合は、これらのルールが優先されます。

11.2. デバイスの名前変更ステップについて

デバイス命名の詳細なステップは以下のとおりです。

1. /usr/lib/udev/rules.d/60-net.rules 内のルールは、udev ヘルパーユーティリティー /lib/udev/rename_device に、すべての / etc/sysconfig/network-scripts/ifcfg-*接尾辞* ファイルを検索するように指示します。インターフェイスの MAC アドレスに一致する HWADDR エントリーを持つ ifcfg ファイルが見つかると、インターフェイスの名前を DEVICE ディレクティブにより ifcfg ファイルで指定した名前に変更します。

2. /usr/lib/udev/rules.d/71-biosdevname.rules 内のルールが biosdevname に対して、前の手順で名前が変更されておらず、biosdevname がインストールされていて、biosdevname = 0 が起動コマンドラインでカーネルコマンドラインでカーネルコマンドラインでカーネルコマンドとして指定されていない場合に、命名ポリシーに従ってインターフェイスの名前を変更するように指示します。

3. /lib/udev/rules.d/75-net-description.rules 内のルールが udev に対して、ネットワークインターフェイスデバイスを調べて内部 udev デバイスプロパティー値

ID_NET_NAME_ONBOARD、ID_NET_NAME_SLOT、ID_NET_NAME_PATH、ID_NET_NAME_MAC を入力するように指示します。一部のデバイスプロパティーは未定義である可能性があることに注意してください。

4. /usr/lib/udev/rules.d/80-net-name-slot.rules 内のルールは、手順 1 または 2 で名前が変更されておらず、カーネルパラメーター net.ifnames=0 が与えられなかった場合に、udev にインターフェイスの名前を変更するように指示します(ID_NET_NAME_ONBOARD、ID_NET_NAME_SLOT、ID_NET_NAME_PATH)。ひとつ目が設定されていない場合、リストの次にあるものが適用されます。いずれのものも設定されていない場合は、インターフェイスの名前は変更されません。

ステップ 3 と 4 は、「命名スキームの序列」にある命名スキーム 1、2、3 とオプションでスキーム 4 を実行しています。ステップ 2 については、「biosdevname を使用した一貫性のあるネットワークデバイスの命名」でより詳細に説明しています。

11.3. 予想可能なネットワークインターフェイスデバイスの命名について

名前には、インターフェイスのタイプに応じた 2 文字の接頭辞が付けられます。

1. en はイーサネット
2. wl はワイヤレス LAN (WLAN)、
3. ww はワイヤレスワイドエリアネットワーク(WWAN)です。

名前には、以下のタイプがあります。

o<*index*>

オンボードデバイスの索引番号

s<*slot*>[f<*function*>][d<*dev_id*>]

ホットプラグスロットの索引番号。すべての多機能 PCI デバイスには、関数 0 デバイスを含むデバイス名 の [f <*function*>] 番号が含まれます。

x<*MAC*>

MAC アドレス

[P<*domain*>]p<*bus*>s<*slot*>[f<*function*>][d<*dev_id*>]

PCI の地理的な場所。PCI の地理的な場所では、[P<*domain*>] 番号は、値が *0* でない場合にのみ言及されます。以下に例を示します。

ID_NET_NAME_PATH=P1enp5s0

[P<*domain*>]p<*bus*>s<*slot*>[f<*function*>][u<*port*>][..][c<*config*>][i<*interface*>]

USB ポート番号のチェーン。USB デバイスの場合は、ハブのポート番号の完全なチェーンで設定されます。名前が 15 文字を超えると、名前はエクスポートされません。チェーンに複数の USB デバイスがある場合は、USB 設定記述子 (c1)および USB インターフェイス記述子 (i0)のデフォルト値が抑制されます。

11.4. SYSTEM Z 上の LINUX で利用可能なネットワークデバイスの命名スキーム

System z 上の Linux インスタンス内のネットワークインターフェイスに予測可能なデバイス名を作成するには、bus-ID 識別子を使用します。bus-ID は、s390 チャンネルサブシステム内でデバイスを特定します。bus ID は、Linux インスタンスのスコープ内でデバイスを特定します。CCW デバイスの場合、バス ID は、先頭に 0.n のデバイスのデバイス番号です。ここで、n はサブチャンネルセット ID です。たとえば、0.1.0ab1 です。

デバイスタイプがイーサネットのネットワークインターフェイスは、以下のように命名されます。

enccw*0.0.1234*

デバイスタイプが SLIP の CTC ネットワークデバイスは、以下のように命名されます。

slccw*0.0.1234*

znetconf -c コマンドまたは **lscss -a** コマンドを使用して、利用可能なネットワークデバイスとその **bus-ID** を表示します。

表11.1 System z 上の Linux のデバイス名タイプ

形式	説明
enccw*bus-ID*	デバイスタイプがイーサネット
slccw*bus-ID*	デバイスタイプが SLIP の CTC ネットワークデバイス

11.5. VLAN インターフェイスの命名スキーム

従来は、VLAN インターフェイス名には ***interface-name.VLAN-ID*** という形式が使われていました。VLAN-ID の範囲は 0 から 4096 までです。最大 4 文字で、インターフェイス名の合計値は 15 文字に制限されます。インターフェイス名の最大の長さはカーネルヘッダーによって定義され、これにはグローバル制限があり、すべてのアプリケーションに影響します。

Red Hat Enterprise Linux 7 では、4 つの命名規則が VLAN インターフェイス名でサポートされています。

VLAN plus VLAN ID

vlan という単語に VLAN ID を加えます。例: vlan0005

パディングなしの VLAN + VLAN ID

vlan という語に VLAN ID （先頭にゼロを付けない）を追加します。例: vlan5

デバイス名および VLAN ID

親インターフェイス名に VLAN ID を追加します。例: enp1s0.0005

デバイス名および VLAN ID (パディングなし)

親インターフェイス名に VLAN ID を追加 (先頭にゼロを付けない) します。例: enp1s0.5

11.6. BIOSDEVNAME を使用した一貫性のあるネットワークデバイスの命名

biosdevname udev ヘルパーユーティリティーで実装されるこの機能は、すべての組み込みネットワークインターフェイス、PCI カードネットワークインターフェイス、および Virtual Function ネットワークインターフェイスの名前を既存のものから変更します。 eth[0123...] から、表 11.2「biosdevname の命名規則」 に示されている新たな命名規則に変更します。システムが Dell システムでない場合や、 「機能の有効化および無効化」 で説明されているように biosdevname が明示的に有効になっていない場合は、systemd 命名スキームが優先されます。

表11.2 biosdevname の命名規則

デバイス	旧式の名前	新しい名前
組み込み型ネットワークインターフェイス (LOM)	**eth[0123...]**	**em[1234...]**[a]
PCI カードネットワークインターフェイス	**eth[0123...]**	**p<*slot*>p<*ethernet port*>**[b]
仮想機能	**eth[0123...]**	**p<*slot*>p<*ethernet port*>_<*virtual interface*>**[c]

[a] 新しい列挙は **1** から始まります。

[b] 以下に例を示します。**p3p4**

[c] 以下に例を示します。**p3p4_1**

11.6.1. システム要件

biosdevname プログラムは、システムの BIOS からの情報、特に SMBIOS に含まれる タイプ 9 (System Slot)および タイプ 41 （オンボードデバイス拡張情報）フィールドからの情報を使用します。システムの BIOS に SMBIOS のバージョン 2.6 もしくはそれ以降がなければ、新しい命名規則は使用されません。ほとんどの旧型のハードウェアは、必要な SMBIOS バージョンとフィールド情報がある BIOS を欠いているため、この機能をサポートしていません。BIOS または SMBIOS バージョンの詳細については、ご使用のハードウェアの製造元にご連絡ください。

この機能を有効にするには、biosdevname パッケージもインストールする必要があります。 これをインストールするには、root で以下のコマンドを実行します。

```
~]# yum install biosdevname
```

11.6.2. 機能の有効化および無効化

この機能を無効にするには、インストール中およびインストール後の両方で、以下のオプションをブートコマンドラインに渡します。

```
biosdevname=0
```

この機能を有効にするには、インストール中およびインストール後の両方で、以下のオプションをブートコマンドラインに渡します。

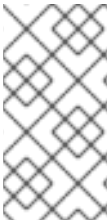
```
biosdevname=1
```

システムが最小要件を満たしている場合を除き、このオプションは無視され、システムは本章の最初に説明したように、**systemd** 命名スキームを使用します。

biosdevname インストールオプションが指定されている場合、システムの有効期間中にブートオプションとして留まる必要があります。

11.7. 管理者向け注意点

多くのシステムカスタム化ファイルはネットワークインターフェイス名を含んでいる場合があるので、システムを旧式規則から新しい規則に移す際には、更新が必要となります。新しい命名規則を使用する場合は、カスタム **iptables** ルール、スクリプトの変更、その他の同様の設定ファイルなど、エリアでネットワークインターフェイス名を更新する必要があります。また、インストールに対してこの変更を有効にするには、**ksdevice** パラメーターでデバイス名を使用する既存の キック スタートファイルを変更する必要があります。これらの キックスタート ファイルは、ネットワークデバイスの **MAC** アドレスまたはネットワークデバイスの新しい名前を使用するように更新する必要があります。



注記

インターフェイス名の最大の長さはカーネルヘッダーによって定義され、これにはグローバル制限があり、すべてのアプリケーションに影響します。

11.8. ネットワークデバイス名の選択に関する制御

デバイスの命名は、以下の方法で制御できます。

ネットワークインターフェイスデバイスの特定

HWADDR ディレクティブを使用して ifcfg ファイルに MAC アドレスを設定すると、udev で識別できます。名前は、DEVICE ディレクティブが指定した文字列から取得されます。この文字列は、慣例により ifcfg の接尾辞と同じです。たとえば、ifcfg-*enp1s0* です。

biosdevnameをオンまたはオフにします。

biosdevname が提供する名前が使用されます(biosdevname が名前を決定できる場合)。

systemd-udev 命名スキームをオンまたはオフにします。

systemd-udev が提供する名前が使用されます(systemd-udev が名前を決定できる場合)。

11.9. ネットワークデバイス命名におけるトラブルシューティング

「デバイスの名前変更ステップについて」にあるように、適用可能な場合は、予測可能なインターフェイス名が各インターフェイスに割り当てられます。udev が使用する名前の一覧を表示するには、root で以下の形式のコマンドを実行します。

```
~]# udevadm info /sys/class/net/ifname | grep ID_NET_NAME
```

ここでの *ifname* は、以下のコマンドでリストを表示されるインターフェイスのいずれかになります。

```
~]$ ls /sys/class/net/
```

利用可能な名前の 1 つは、「デバイスの名前変更ステップについて」 で説明されているルールに従って udev により適用されます。

- /usr/lib/udev/rules.d/60-net.rules - from initscripts,
- /usr/lib/udev/rules.d/71-biosdevname.rules - from biosdevname,

- /usr/lib/udev/rules.d/80-net-name-slot.rules - from systemd

上記のルールファイルの一覧からは、インターフェイスの命名が initscripts または biosdevname を介して行われる場合は、常に udev のネイティブ命名よりも優先されます。ただし、initscripts の名前変更が行われず、biosdevname が無効になっている場合は、インターフェイス名を変更するには、80-net-name-slot.rules を /usr から /etc にコピーし、ファイルを適切に編集します。つまり、特定の順番で使われるようにスキームをコメントアウトまたは配置します。

例11.1 カーネルネームスペースから命名されているインターフェイス (eth[0,1,2...]) と udev が正常に名前を変更したインターフェイスが混在している場合

スキームが混在しているとは、ハードウェアによってはカーネルによって udev に提供される情報で使用可能なものがないため、名前を特定できなかったり、udev に提供された情報が非一意のデバイス ID などに適していないことを意味します。後者の方が一般的で、ifcfg ファイルでカスタム命名スキームを使用するか、80-net-name-slot.rules を編集して使用する udev スキームを変更してください。

例11.2 /var/log/messages または systemd ジャーナルで、各インターフェイスの名前変更が 2 回実行されている場合

ifcfg ファイルにエンコードされているが、initrd イメージを再生成していない命名スキームのシステムでは、この問題が発生する可能性があります。インターフェイス名は、最初に initrd にある間、初期起動時に(biosdevname、udev、またはカーネルコマンドラインの dracut パラメーターを介して)割り当てられます。そして、実際の rootfs に切り替えた後、名前変更は 2 回行われ、60-net.rules を処理するため、udev が生成する /usr/lib/udev/rename_device バイナリーにより、新しいインターフェイス名が決定されます。このメッセージは無視しても問題ありません。

例11.3 ifcfg ファイル内の ethX 名の命名スキームが機能しない場合

Red Hat Enterprise Linux 7 では、非常に特殊な状況を除き、一貫性のある ethX 命名規則を適用することはできません。

インターフェイスを特定の名前に設定する udev ルールは、要求された名前が他のインターフェイスによってすでに使用されていると失敗します。これには、/usr/lib/udev/rules.d/60-net.rules ファイルによって提供される機能が含まれます。

起動時にカーネルがすべてのネットワークデバイスを把握する際には、eth*X* 命名規則が使われ

ます。eth*X* 名は再起動ごとに変わるので、予測が不可能です。したがって、udev を使用してインターフェイスの名前を予測可能な名前に変更しようとしたり、カーネルが提供する予測不可能な eth*X* 名の順序を変更しようとすると失敗します。

eth*X* 名を使用すると、以下のシナリオで適切に機能します。

- システムにネットワークインターフェイスが 1 つしかない。
- Red Hat Enterprise Linux 7 ゲスト仮想マシンの virtio NIC に使われた場合。仮想化の導入および管理ガイドの KVM 準仮想化 (virtio) ドライバーおよびネットワーク設定の章を参照してください。

例11.4 net.ifnames=0 を設定した結果 enpXxX 名に一貫性がない場合

systemd 予測可能なインターフェイス命名(net.ifnames)と biosdevname 命名スキームの両方が無効になっている場合、ネットワークインターフェイスは引き続き、カーネルによって最初に指定された予測不能で一貫性のない eth*X* 名を使用します。

起動時にカーネルがすべてのネットワークデバイスを把握する際には、必ず enpXxX 命名規則が使われます。並列化により、カーネルインターフェイスの列挙の順序は、再起動後も異なります。Red Hat Enterprise Linux は、systemd の予測可能なインターフェイス命名スキームまたは biosdevname 命名スキームのいずれかに依存して、カーネル予測できない eth*X* インターフェイスの名前を、再起動後も常に一貫性を保ちます。

ネットワークアダプターの命名規則の詳細は、Red Hat カスタマーポータルでナレッジセンターサポートの記事Is it safe to set net.ifnames=0 in RHEL7?を参照してください。Red Hat カスタマーポータルのナレッジセンターのサポートに関する記事

例11.5 イーサネットインターフェイスの接頭辞の制限

選択する接頭辞は、次の要件を満たす必要があります。

- ASCII 文字で設定される。

- 英数字の文字列である。
- 16 文字より短い。
- eth、eno、ens、em など、ネットワークインターフェイスの命名に使用される周知の接頭辞と競合しません。

11.10. 関連情報

インストールされているドキュメント

- udev (7) man ページ - Linux 動的デバイス管理デーモン udevd を説明しています。
- systemd (1) man ページ - systemd システムおよびサービスマネージャーを説明しています。
- biosdevname (1) man ページ - デバイスの BIOS 指定名を取得するユーティリティーが説明されています。

オンラインドキュメント

- IBM Knowledge Center の文書 SC34-2710-00『Device Drivers, Features, and Commands on Red Hat Enterprise Linux 7』には、IBM System z デバイスおよび付属品の「Predictable network device names」に関する情報が含まれます。

第12章 代替ルートを定義するためのポリシーベースのルーティングの設定

デフォルトでは、RHEL のカーネルは、ルーティングテーブルを使用して宛先アドレスに基づいてネットワークパケットを転送する場所を決定します。ポリシーベースのルーティングにより、複雑なルーティングシナリオを設定できます。たとえば、送信元アドレス、パケットメタデータ、プロトコルなどのさまざまな基準に基づいてパケットをルーティングできます。

本セクションでは、NetworkManager を使用してポリシーベースのルーティングを設定する方法を説明します。

注記

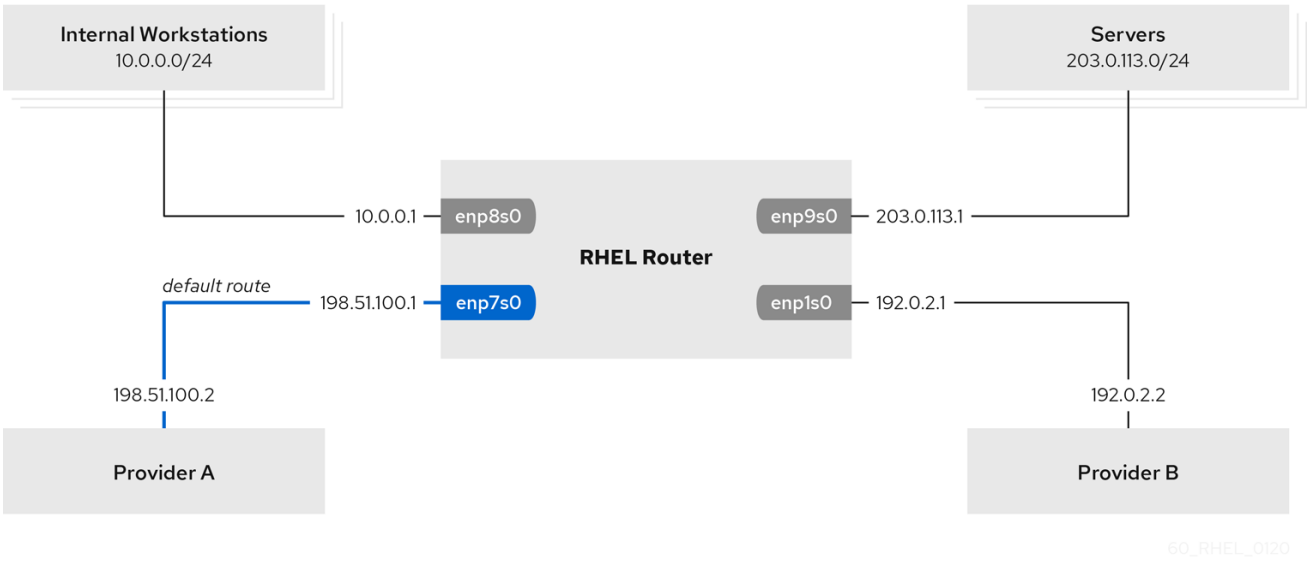
NetworkManager を使用するシステムでは、nmcli ユーティリティーのみがルーティングルールの設定と、特定のテーブルへのルートの割り当てをサポートします。

12.1. 特定のサブネットから異なるデフォルトゲートウェイへのトラフィックのルーティング

本セクションでは、デフォルトで、デフォルトルートを使用して、すべてのトラフィックをインターネットプロバイダー A にルーティングするルーターとして RHEL を設定する方法を説明します。RHEL は、ポリシーベースのルーティングを使用して、内部ワークステーションのサブネットから受信したトラフィックをプロバイダー B にルーティングします。

この手順では、次のネットワークトポロジーを想定しています。

図12.1 接続のアクティブ化



前提条件

- この手順で設定する RHEL ルーターには、4 つのネットワークインターフェイスがあります。
  - enp7s0 インターフェイスはプロバイダー A のネットワークに接続されています。プロバイダーのネットワークのゲートウェイ IP は 198.51.100.2 で、ネットワークは /30 ネットワークマスクを使用します。
  - enp1s0 インターフェイスはプロバイダー B のネットワークに接続されています。プロバイダーのネットワークのゲートウェイ IP は 192. 0.2.2 で、ネットワークは /30 ネットワークマスクを使用します。
  - enp8s0 インターフェイスは、内部ワークステーションで 10.0.0.0 /24 サブネットに接続されています。
  - enp9s0 インターフェイスは、会社のサーバーを使用して 203.0 .113.0/24 サブネットに接続されています。
- 内部ワークステーションのサブネット内のホストは、デフォルトゲートウェイとして 10.0.0.1 を使用します。この手順では、この IP アドレスをルーターの enp8s0 ネットワークインターフェイスに割り当てます。
- サーバーサブネット内のホストは、デフォルトゲートウェイとして 203.0.113.1 を使用します。この手順では、この IP アドレスをルーターの enp9s0 ネットワークインターフェイスに割り当てます。
- firewalld サービスは有効でアクティブです（これがデフォルトです）。

手順

1. プロバイダー A へのネットワークインターフェイスを設定します。

```
# 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 コマンドは、NetworkManager 接続プロファイルを作成します。次のリストでは、コマンドのオプションを説明します。

- **type ethernet: 接続タイプがイーサネットであることを定義します。**
- **con-name *connection_name*: プロファイルの名前を設定します。混乱を避けるために、わかりやすい名前を使用してください。**
- **ifname *network_device*: ネットワークインターフェイスを設定します。**
- **ipv4.method manual: 静的 IP アドレスを設定できるようにします。**
- **ipv4.addresses *IP_address*/*subnet_mask*: IPv4 アドレスとサブネットマスクを設定します。**
- **ipv4.gateway *IP_address*: デフォルトゲートウェイアドレスを設定します。**
- **ipv4.dns *IP_of_DNS_server*: DNS サーバーの IPv4 アドレスを設定します。**
- **connection.zone *firewalld_zone*: 定義された firewalld ゾーンにネットワークインターフェイスを割り当てます。firewalld は、external ゾーンに割り当てられたマスカレードインターフェイスを自動的に有効にすることに注意してください。**

2. **プロバイダー B へのネットワークインターフェイスを設定します。**

```
# 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
```

このコマンドは、デフォルトゲートウェイを設定する *ipv4 .gateway の代わりに ipv4. routes* パラメーターを使用します。これは、この接続のデフォルトゲートウェイをデフォルトとは異なるルーティングテーブル(5000)に割り当てるために必要です。NetworkManager は、接続がアクティブになると、この新しいルーティングテーブルを自動的に作成します。

注記

nmcli ユーティリティーは、*ipv4*.gateway のデフォルトゲートウェイに 0.0.0.0/0 を使用することに対応していません。この問題を回避するには、0.0.0. 0/1 と 128. 0.0.0/1 の両方のサブネットに個別のルートを作成します。これは、完全な IPv4 アドレス空間に対応します。

3. **内部ワークステーションサブネットへのネットワークインターフェイスを設定します。**

```
# 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
```

このコマンドは、*ipv4.routes* パラメーターを使用して、ID が 5000 のルーティングテーブルに静的ルートを追加します。10.0.0. 0/24 サブネットのこの静的ルートは、プロバイダー B (192.0.2.1)へのローカルネットワークインターフェイスの IP を次のホップとして使用します。

さらに、このコマンドは *ipv4.routing-rules* パラメーターを使用して、優先度 5 のルーティングルールを追加します。このルーティングルールは、トラフィックを 10.0.0.0/24 サブネットからテーブル 5000 にルーティングします。値が小さいほど優先度が高くなります。

***ipv4.routing-rules* パラメーターの構文は ip route add コマンドと同じですが、*ipv4.routing-rules* は常に優先度を指定する必要があります。**

4. **サーバーサブネットへのネットワークインターフェイスを設定します。**

```
# nmcli connection add type ethernet con-name Servers ifname enp9s0 ipv4.method manual
ipv4.addresses 203.0.113.1/24 connection.zone trusted
```

検証手順

1. **内部ワークステーションサブネットの RHEL ホストで、以下を行います。**
    1. **traceroute パッケージをインストールします。**

```
# yum install traceroute
```

2. **traceroute ユーティリティーを使用して、インターネット上のホストへのルートを表示します。**

```
# 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
 ...
```

コマンドの出力には、ルーターがプロバイダー B のネットワークである 192.0.2.1 を介してパケットを送信することが表示されます。

2. **サーバーのサブネットの RHEL ホストで、以下を行います。**

1. **traceroute パッケージをインストールします。**

```
# yum install traceroute
```

2. **traceroute ユーティリティーを使用して、インターネット上のホストへのルートを表示します。**

```
# 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
 ...
```

コマンドの出力には、ルーターがプロバイダー A のネットワークである 198.51.100.2 を介してパケットを送信することが表示されます。

トラブルシューティングの手順

RHEL ルーターで以下を行います。

1. **ルールのリストを表示します。**

```
# 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. **テーブル 5000 のルートを表示します。**

```
# 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. **ファイアウォールゾーンに割り当てられているインターフェイスを表示します。**

```
# firewall-cmd --get-active-zones
external
  interfaces: enp1s0 enp7s0
trusted
  interfaces: enp8s0 enp9s0
```

4. **外部 ゾーンでマスカレードが有効になっていることを確認します。**

```
# firewall-cmd --info-zone=external
external (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0 enp7s0
  sources:
  services: ssh
  ports:
  protocols:
  masquerade: yes
  ...
```

関連情報

- **nmcli connection add コマンドに設定できる *ipv4.** パラメーターの詳細は、nm-settings(5) man ページの 『IPv4 settings』 セクションを参照してください。**

- **nmcli connection add コマンドに設定できる *connection.** パラメーターの詳細は、nm-settings(5) man ページの 『Connection settings』 セクションを参照してください。**

- **nmcli を使用した NetworkManager 接続の管理の詳細は、nmcli(1) man ページの 『Connection management commands』 セクションを参照してください。**

パート III. INFINIBAND および RDMA ネットワーク

このパートでは、RDMA、InfiniBand、および InfiniBand 上での IP ネットワーク接続の設定を説明します。

第13章 INFINIBAND および RDMA ネットワークの設定

13.1. INFINIBAND および RDMA のテクノロジーについて

InfiniBand とは、2 つの異なることを指します。1 つ目は、InfiniBand ネットワーク用の物理的リンク層プロトコルです。2 つ目は、InfiniBand Verbs API と呼ばれる高レベルのプログラミング API です。InfiniBand Verbs API は、 *remote direct memory access* (RDMA) テクノロジーの実装になります。

RDMA は、いずれのコンピューターのオペレーティングシステムを呼び出さずに、あるコンピューターのメモリーから、別のコンピューターのメモリーへの直接アクセスします。この技術により、CPU の使用量が低いまま、スループットが高く、レイテンシーが低いネットワークが可能になります。これは、並行のコンピューターのクラスターが大量にある場合に特に役に立ちます。

一般的な IP データ転送では、マシン A のアプリケーション X はマシン B のアプリケーション Y にいくつかのデータを送信します。 転送の一環として、マシン B のカーネルは最初にデータを受け取り、パケットヘッダーをデコードし、データがアプリケーション Y に属することを確認し、アプリケーション Y がカーネルに対して読み取りシステムコールを実行するのを待ちます。その後、カーネル自体の内部メモリー領域からアプリケーション Y が提供するバッファーにデータを手動でコピーする必要があります。 このプロセスは、ほとんどのネットワークトラフィックがシステムのメインメモリーバス全体に少なくとも 2 回コピーされる必要があることを意味します（ホストアダプターが DMA を使用してデータをカーネル提供のメモリーバッファーに配置してから 1 回、カーネルがデータをアプリケーションのメモリーバッファーに移動すると再び）。また、コンピューターは、カーネルコンテキストとアプリケーション Y コンテキストの間で切り換えるために複数のコンテキストスイッチを実行する必要があることを意味します。このため、ネットワークトラフィックが高速で流れている場合は、システムにかかる CPU 負荷が非常に高くなり、その他のタスクが遅くなる可能性があります。

RDMA 通信は通常の IP 通信とは異なります。これは、通信プロセスでカーネルの介入を回避し、その過程でネットワーク通信の処理に通常必要となる CPU オーバーヘッドを大幅に削減するためです。RDMA プロトコルを使用すると、ネットワークからいつパケットが届くか、そのパケットをどのアプリケーションが受信すべきか、アプリケーションのメモリースペースのどこにそのパケットが送られるべきかを、マシン内のホストアダプターが判別できます。パケットをカーネルに送信して処理を行い、それからユーザーのアプリケーションメモリーにコピーする代わりに、パケットのコンテンツを直接アプリケーションのバッファーに配置すると、その他の介入は不要になります。ただし、ほとんどの IP ネットワークアプリケーションが構築されている標準の Berkeley Sockets API を使用して実行することはできません。そのため、独自の API、InfiniBand Verbs API、およびアプリケーションは、RDMA テクノロジーを直接使用する前にこの API に移植する必要があります。

Red Hat Enterprise Linux 7 は、InfiniBand ハードウェアおよび InfiniBand Verbs API の両方をサポートします。さらには、InfiniBand Verbs API を、InfiniBand 以外のハードウェアで使用できるようにする追加技術もあります。

- Internet Wide Area RDMA Protocol (iWARP)

iWARP は、インターネットプロトコル (IP) ネットワークでの効果的なデータ転送に対してリモートダイレクトメモリーアクセス (RDMA) を実装するコンピューターのネットワークプロトコルです。

- RoCE (RDMA over Converged Ethernet) プロトコル (後に IBoE (InfiniBand over Ethernet) に名前が変更)

  RoCE は、イーサネットネットワークでリモートダイレクトメモリーアクセス (RDMA) を許可するネットワークプロトコルです。

前提条件

iWARP と RoCE の技術はいずれも、通常の IP ネットワークリンク層が基礎となるテクノロジーであるため、設定の大半は実際には *3章IP ネットワークの設定* で説明されています。 ほとんどの場合、IP ネットワーク機能が適切に設定されると、RDMA 機能はすべて自動で、ハードウェアに適したドライバーがインストールされている限り表示されます。 カーネルドライバーは、常に Red Hat が提供する各カーネルに含まれていますが、マシンのインストール時に InfiniBand パッケージグループが選択されなかった場合は、ユーザー領域ドライバーを手動でインストールする必要があります。

Red Hat Enterprise Linux 7.4 以降、すべての RDMA ユーザー領域ドライバーは rdma-core パッケージに統合されました。サポートされているすべての iWARP、RoCE、または InfiniBand ユーザースペースドライバーをインストールするには、root で次のコマンドを実行します。

```
~]# yum install libibverbs
```

Priority Flow Control (PFC)および mlx4 ベースのカードを使用している場合は、/etc/modprobe.d/mlx4.conf を編集して、カードがプラグインするイーサネットスイッチ上の「no-drop」 サービスに設定されたパケット優先度をドライバーに指示し、initramfs を再構築して変更したファイルを追加します。新しい mlx5 ベースのカードは、スイッチと PFC 設定を自動ネゴシエートし、「no-drop」 優先度を通知するモジュールオプションは必要としません。

Mellanox カードを設定して、イーサネットモードでポートのいずれかまたは両方を使用するには、「Mellanox カードのイーサネット用の設定」を参照してください。

(InfiniBand インストールで通常はインストールされる RDMA パッケージに加えて) これらのドライバーパッケージがインストールされれば、ユーザーは通常の RDMA アプリケーションの大半を利用して、アダプターで RDMA プロトコル通信が行われていることをテスト、確認できます。 ただし、Red Hat Enterprise Linux 7 に含まれているすべてのプログラムが適切に iWARP や RoCE/IBoE デバイスをサポートするわけではありません。 これは特に、iWARP 上の接続確立プロトコルが実際の

InfiniBand リンク層接続とは異なるためです。 問題のプログラムが librdmacm 接続管理ライブラリーを使用する場合は、iWARP と InfiniBand の違いをサイレントに処理し、プログラムは機能します。 アプリケーションが独自の接続管理を実行しようとすると、iWARP を明確にサポートする必要があり、これが行われない場合は機能しません。

13.2. ROCE を使用したデータ転送

RoCE (RDMA over Converged Ethernet) は、イーサネットネットワークでリモートダイレクトメモリーアクセス (RDMA) を有効にするネットワークプロトコルです。RoCE には 2 つのバージョン (RoCE v1 および RoCE v2) があり、使用されるネットワークアダプターにより異なります。

RoCE v1

RoCE v1 プロトコルは、同じイーサネットブロードキャストドメインの任意の 2 つのホスト間の通信を可能にするイーサネットタイプ 0x8915 を持つイーサネットリンク層プロトコルです。RoCE v1 は、ConnectX-3 ネットワークアダプターを使用する場合の RDMA Connection Manager (RDMA_CM) のデフォルトバージョンです。

RoCE v2

RoCE v2 プロトコルは、UDP over IPv4 または UDP over IPv6 プロトコルのいずれかに存在します。UDP 宛先ポート番号 4791 は RoCE v2 用に予約されています。Red Hat Enterprise Linux 7.5 以降、RoCE v2 は、ネットワークアダプター ConnectX-3 Pro、ConnectX-4、ConnectX-4 Lx、および ConnectX-5 を使用する場合は、RDMA_CM のデフォルトバージョンです。ハードウェアは RoCE v1 および RoCE v2 の両方をサポートします。

RDMA Connection Manager (RDMA_CM) は、データを転送するために、クライアントとサーバーとの間の信頼できる接続を設定するために使用されます。RDMA_CM は、接続を確立するために RDMA トランスポートに依存しないインターフェイスを提供します。通信は、特定の RDMA デバイスで行われ、データ転送はメッセージベースとなります。

前提条件

RDMA_CM セッションには、以下のいずれかが必要です。

- クライアントおよびサーバーで、同じ RoCE モードをサポートします。
- クライアントは RoCE v1 をサポートし、サーバーは RoCE v2 をサポートします。

クライアントが接続のモードを決定するため、次のような状況が考えられます。

成功した接続:

クライアントが、使用されるネットワークカードおよびドライバーに従って、RoCE v1 モードまたは RoCE v2 モードにある場合、対応するサーバーは同じバージョンを使用して接続を作成する必要があります。また、接続は、クライアントが RoCE v1 モードで、サーバーが RoCE v2 モードの場合に限り成功します。

失敗した接続:

クライアントが RoCE v2 にあり、対応するサーバーが RoCE v1 の場合は、接続は確立されません。この場合は、対応するサーバーのドライバーまたはネットワークアダプターを更新します。「RoCE を使用したデータ転送」を参照してください。

表13.1 RDMA_CM を使用する RoCE バージョンのデフォルト

クライアント	サーバー	デフォルト設定
RoCE v1	RoCE v1	接続
RoCE v1	RoCE v2	接続
RoCE v2	RoCE v2	接続
RoCE v2	RoCE v1	接続なし

クライアントの RoCE v2 と、サーバーの RoCE v1 は互換性がありません。この問題を解決するには、サーバーとクライアントで、RoCE v1 で通信することを強制します。これは、RoCE v2 がサポートするハードウェアが RoCE v1 を使用するように強制することを意味します。

手順13.1 ハードウェアがすでに Roce v2 で実行している場合にデフォルトの RoCE モードへの変更

1. /sys/kernel/config/rdma_cm ディレクトリーに移動し、RoCE モードにします。

```
~]# cd /sys/kernel/config/rdma_cm
```

2. イーサネットネットワークデバイスを指定して ibstat コマンドを入力して、ステータスを表示します。たとえば、*mlx5_0* の場合は以下のようになります。

```
~]$ 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*** デバイスにディレクトリーを作成します。

```
~]# mkdir mlx5_0
```

4. ツリー形式で、default_roce_mode ファイルで RoCE モードを表示します。

```
~]# 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. デフォルトの RoCE モードを変更します。

```
~]# echo "RoCE v1" > /sys/kernel/config/rdma_cm/mlx5_0/ports/1/default_roce_mode
```

6. 変更を表示します。

```
~]$ cat /sys/kernel/config/rdma_cm/mlx5_0/ports/1/default_roce_mode
  RoCE v1
```

13.3. SOFT-ROCE の設定

RoCE は、ハードウェアおよびソフトウェアに実装できます。Soft-RoCE は、RDMA トランスポートのソフトウェア実装です。

前提条件

Red Hat Enterprise Linux 7.4 以降、Soft-RoCE ドライバーはカーネルに統合されました。ユーザー領域のドライバーも rdma-core パッケージに統合されました。Soft-RoCE は RXE としても知られています。RXE を起動、停止、設定するには、rxe_cfg スクリプトを使用します。rxe_cfg のオプションを表示するには、rxe_cfg help を入力します。

手順13.2 Soft-RoCE の設定

1. root ユーザーとして以下のコマンドを入力し、RXE の現在の設定ステータスを表示します。

```
~]# 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 カーネルモジュールを読み込み、RXE を起動するには、root で次のコマンドを実行します。

```
~]# 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
```

必要に応じて、RXE カーネルモジュールが読み込まれていることを確認するには、以下を実行します。

```
~]# 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.

イーサーネットインターフェイスに新しい RXE デバイスを追加する前に、対応するインターフェイスを開いて、有効な IP アドレスを割り当てます。新しい RXE デバイス (たとえば *igb_1*) を追加するには、以下のコマンドを実行します。

```
~]# 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 列の rxe0 は、*igb_1* デバイスで rxe が有効であることを示しています。

4.

RXE デバイスのステータスを確認するには、ibv_devices コマンドを使用します。

```
~]# ibv_devices
 device                 node GUID
 ------              ----------------
 mlx4_0              0002c90300b3cff0
 rxe0                a2369ffffe018294
```

または、**ibstat** に詳細なステータスを入力します。

```
~]# 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 デバイスの削除

RXE デバイスを削除する場合は、以下のコマンドを実行します。

```
~]# rxe_cfg remove igb_1
```

RXE デバイスの接続性の確認

以下の例は、サーバーおよびクライアントで、**RXE** デバイスの接続を確認する方法を説明します。

例13.1 サーバーで RXE デバイスの接続性の確認

```
~]$ 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
```

例13.2 クライアントで RXE デバイスの接続性の確認

```
~]$ 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 および RDMA に関連するソフトウェアパッケージ

RDMA アプリケーションは Berkeley Sockets ベースのアプリケーションと、通常の IP ネットワークとは異なるため、IP ネットワークで使用されるほとんどのアプリケーションは、RDMA ネットワークで直接使用できません。Red Hat Enterprise Linux 7 には、RDMA ネットワーク管理、テストおよびデバッグ、高レベルソフトウェア開発 API、およびパフォーマンス分析を目的とした多くの異なるソフトウェアパッケージが同梱されています。

これらのネットワークを使用するには、以下のパッケージのいくつか、もしくはすべてをインストールする必要が あります (以下のリストはすべてをカバーしたものではありませんが、RDMA 関連の最重要パッケージは記載されています)。

必須パッケージ

- rdma: RDMA スタックのカーネル初期化を行います。
- libibverbs: InfiniBand Verbs API を提供します。
- opensm: サブネットマネージャー (ファブリックでアクティブなサブネットマネージャーがない場合、1 台のマシンでのみ必要)。
- user space driver for installed hardware: 1 つ以上: infinipath-psm、libcxgb3、libcxgb4、libehca、libipathverbs、libmthca、libmlx4、libmlx5、libnes

、libocrdma。libehca は IBM Power Systems サーバーでのみ利用できます。

推奨パッケージ

- librdmacm、librdmacm-utils、ibacm: InfiniBand、iWARP、および RoCE の違いを認識する接続管理ライブラリーで、これらすべてのタイプのハードウェアで適切に接続を開くことができます。ネットワーク稼働を検証する簡単なテストプログラム。ライブラリーと一体化して大規模クラスターでのリモートホスト解決を迅速化するキャッシングデーモン。

- libibverbs-utils: インストール済みハードウェアのクエリーを行い、ファブリックでの通信を検証する簡単な verb ベースのプログラム。

- infiniband-diags および ibutils: InfiniBand ファブリック管理用に便利なデバッグツールを多く提供します。iWARP や RoCE 上では、これらのツールの機能は非常に限定されたものになります。これは、ほとんどのツールは Verbs API 層ではなく、InfiniBand リンク層で機能するためです。

- perftest および qperf: さまざまなタイプの RDMA 通信のパフォーマンステストアプリケーション。

オプションパッケージ

以下のパッケージは Optional チャンネルで入手できます。Optional チャンネルからパッケージをインストールする前に、対象範囲の詳細を参照してください。Optional チャンネルのサブスクライブに関する情報は、Red Hat ナレッジベースソリューションの記事Red Hat Subscription Management (RHSM) を使用して、オプショナルチャンネル、サブチャンネル、-devel パッケージにアクセスするで説明されています。

- dapl、dapl-devel、および dapl-utils: Verbs API とは異なる RDMA 用の API を提供します。これらのパッケージには、ランタイムコンポーネントと開発コンポーネントの両方があり

ます。

- openmpi、mvapich2、および mvapich2-psm: RDMA 通信を使用できる MPI スタック。これらのスタックに書き込みを行うユーザースペースのアプリケーションは、必ずしも RDMA 通信が実行中であることを認識しません。

13.5. BASE RDMA サブシステムの設定

rdma サービスの起動は自動的に実行されます。 InfiniBand か iWARP か RoCE/IBoE かに関係なく、RDMA 対応ハードウェアが検出されると、udev は systemd に rdma サービスを開始するように指示します。

```
~]# 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
```

ユーザーは rdma サービスを有効にする必要はありませんが、常に強制する場合は可能です。これを行うには、root で以下のコマンドを入力します。

```
~]# systemctl enable rdma
```

13.5.1. rdma.conf ファイルの設定

rdma サービスは /etc/rdma/rdma.conf を読み込んで、管理者がデフォルトで読み込む必要のあるカーネルレベルおよびユーザーレベルの RDMA プロトコルを確認します。 このファイルを編集すると、様々なドライバーを有効、無効にできます。

以下のドライバーが有効、無効にできます。

- IPoIB: IP ネットワークエミュレーション層で、InfiniBand ネットワーク上での IP アプリケーションの実行を可能にします。

- SRP: SCSI 要求プロトコルです。 これにより、マシンはローカルハードディスクであるかのように、マシンの SRP プロトコルを使用してエクスポートされるリモートドライブまたは

ドライブアレイをマウントできます。

- SRPT: これは SRP プロトコルのターゲットモードまたはサーバーモードです。 これは、他のマシンにドライブもしくはドライブアレイをエクスポートするために必要なカーネルサポートを読み込みます。その他のマシンは、これらをまるでローカル上にあるかのようにマウントします。詳細は、targetd および targetcli パッケージのドキュメントを参照してください。

- ISER: Linux カーネルの iSCSI 層全般用の低レベルのドライバーで、iSCSI デバイスに InfiniBand ネットワークでのトランスポートを提供します。

- RDS: Linux カーネル内の Reliable Datagram Service です。 Red Hat Enterprise Linux 7 カーネル内では有効にされないため、読み込むことができません。

13.5.2. 70-persistent-ipoib.rules の使用

rdma パッケージは、/etc/udev.d/rules.d/70-persistent-ipoib.rules ファイルを提供します。この udev ルールファイルは、IPoIB デバイスの名前をデフォルト名(ib0 や ib1など)からよりわかりやすい名前に変更するために使用されます。デバイス名を変更するには、このファイルの編集が必要になります。まず、名前を変更する デバイスの GUID アドレスを見つけます。

```
~]$ 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
```

リンク/ インフニバンドの直後に、IPoIB インターフェイスの 20 バイトのハードウェアアドレスになります。新規の名前作成に必要なのは、上記で太字表示となっている最後の 8 バイトのみです。ユーザーは好みに合わせて自由に命名スキームを作成できます。たとえば、mlx4 デバイスが ib0 サブネットファブリックに接続されている場合は、mlx4 _ib0 などの *device_fabric* 命名規則を使用します。推奨されない唯一の方法は、ib0 や ib1 などの標準名を使用することです。これらは、カーネルに割り当てられた自動名と競合する可能性があるためです。次に、ルールファイルにエントリーを追加します。ルールファイルの既存の例をコピーし、ATTR{address} エントリーの 8 バイトを、名前を変更するデバイスから強調表示した 8 バイトに置き換え、NAME フィールドで使用する新しい名前を入力します。

13.5.3. ユーザーのメモリーロック制限解除

RDMA 通信では、コンピューター内の物理的メモリーを固定 (つまり、コンピューター全体で利用可能なメモリーが不足した場合に、カーネルはそのメモリーをページングファイルにスワップすること

ができない) する必要があります。メモリーの固定は通常、非常に限られた権限が必要となる操作です。root 以外のユーザーが大規模な RDMA アプリケーションを実行できるようにするには、root 以外のユーザーがシステムのピン留めを許可されているメモリー量を増やす必要がある場合があります。これを行うには、/etc/security/limits.d/ ディレクトリーに、以下のような内容でファイルを追加します。

```
~]$ more /etc/security/limits.d/rdma.conf
# configuration for rdma tuning
*       soft    memlock         unlimited
*       hard    memlock         unlimited
# rdma tuning end
```

13.5.4. Mellanox カードのイーサネット用の設定

Mellanox の特定のハードウェアは、InfiniBand モードまたは Ethernet モードで実行できます。このようなカードのデフォルトは通常 InfiniBand です。カードを Ethernet モードに設定できます。現在、ConnectX ファミリーハードウェア(mlx5 または mlx4 のいずれかを使用する)でのみモードを設定することがサポートされています。

Mellanox mlx5 カードを設定するには、mstflint パッケージの mstconfig プログラムを使用します。詳細は、Red Hat カスタマーポータルでナレッジベースの記事Configuring Mellanox mlx5 cards in Red Hat Enterprise Linux 7を参照してください。

Mellanox mlx4 カードを設定するには、ナレッジベースの記事 で説明されているように、mstconfig を使用してカードにポートタイプを 設定します。mstconfig がカードに対応していない場合は、/etc/rdma/mlx4.conf ファイルを編集し、そのファイルの手順に従って RoCE/IBoE の使用に適したポートタイプを設定します。この場合、initramfs を再構築して、更新されたポート設定が initramfs にコピーされていることを確認する必要もあります。

ポートタイプが設定されたら、1 つまたは両方のポートがイーサネットに設定されており、mstconfig を使用してポートタイプを設定しなかった場合には、ログに以下のメッセージが表示される可能性があります。

```
mlx4_core 0000:05:00.0: Requested port type for port 1 is not supported on this HCA
```

これは正常なことで、操作に影響は出ません。ポートタイプを設定するスクリプトには、ドライバーがいつポート 2 を内部で要求されたタイプに切り替えたかを知るすべがありません。このため、スクリプトがポート 2 の切り替えを要求してから切り替え操作が完了するまでの間は、ポート 1 を異なるタイプに設定しようとしても拒否されます。スクリプトはコマンドが成功するか、ポートの切り替えが完了されないことを示すタイムアウトになるまで設定しようとします。

13.5.5. リモートの Linux SRP ターゲットへの接続

SCSI RDMA Protocol (SRP) は、システムが、別のシステムに接続されている SCSI デバイスにアクセスするために RDMA を使用できるようにするネットワークプロトコルです。SRP イニシエーターが、SRP ターゲット側で SRP ターゲットに接続するようにするには、そのイニシエーターで使用するホストチャンネルアダプター (HCA) ポートに、アクセスコントロールリスト (ACL) エントリーを追加する必要があります。

HCA ポートの ACL ID は固有ではありません。ACL ID は、HCA の GID 形式により異なります。同じドライバーを使用する HCA (ib_qib など)は異なる形式の GID を持つことができます。また、ACL ID は、接続要求を開始する方法により異なります。

リモートの Linux SRP ターゲットへの接続: 概要

1. **ターゲット側の準備:**

   a. **ストレージのバックエンドを作成します。たとえば、*/dev/sdc1* パーティションを取得します。**

   ```
   /> /backstores/block create vol1 /dev/sdc1
   ```

   b. **SRP ターゲットを作成します。**

   ```
   /> /srpt create 0xfe80000000000000001175000077dd7e
   ```

   c. **ステップ A で作成したバックエンドに基づいた LUN を作成します。**

   ```
   /> /srpt/ib.fe80000000000000001175000077dd7e/luns create /backstores/block/vol1
   ```

   d. **リモートの SRP クライアントに Node ACL を作成します。**

   ```
   /> /srpt/ib.fe80000000000000001175000077dd7e/acls create
   0x7edd77000075110000 1175000077d708
   ```

ノード ACL は srp_daemon と ibsrpdm では異なることに注意してください。

2. クライアント側で srp_daemon または ibsrpdm を使用して SRP 接続を開始します。

```
[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. オプション:lsscsi や dmesg など、さまざまなツールとの SRP 接続を検証することが推奨されます。

手順13.3 srp_daemon または ibsrpdm を使用して、リモートの Linux SRP ターゲットへの接続

1. ターゲットで ibstat コマンドを使用して、State および Port GUID の値を確認します。HCA は Active 状態である必要があります。ACL ID は、Port GUID に基づいています。

```
[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. SRP ターゲット ID を取得します。この ID は、HCA ポートの GUID に基づきます。SRP ターゲットのバックエンドとして専用のディスクパーティションが必要になります（例：

/dev/sdc1 ）。次のコマンドは、デフォルトの接頭辞を fe80 に置き換え、コロンを削除し、その残りの文字列に、新しい接頭辞を追加します。

```
[root@target]# ibstatus | grep '<default-gid>' | sed -e 's/<default-gid>://' -e 's/://g' | grep
001175000077dd7e
fe80000000000000001175000077dd7e
```

3. **targetcli ツールを使用してブロックデバイスに LUN vol1 を作成し、SRP ターゲットを作成し、LUN をエクスポートします。**

```
[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. **イニシエーターで ibstat コマンドを使用して、状態が Active かどうかを確認し、ポート GUID を決定します。**

```
[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. **以下のコマンドを使用して、リモートの SRP ターゲットに接続せずにスキャンします。ターゲットの GUID は、イニシエーターがリモートターゲットを検出したことを示します。ID 文字列は、リモートターゲットが Linux ソフトウェアターゲット(ib_srpt.ko)であることを示します。**

```
[root@initiator]# srp_daemon -a -o
IO Unit Info:
    port LID:        0001
    port GID:        fe8000000000000000001175000077dd7e
    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 接続を確認するには、lsscsi コマンドを使用して SCSI デバイスを一覧表示し、イニシエーターがターゲットに接続する前後に lsscsi 出力を比較します。**

```
[root@initiator]# lsscsi
[0:0:10:0]   disk    IBM-ESXS ST9146803SS      B53C  /dev/sda
```

7. **イニシエーターポートに有効な ACL を設定せずにリモートターゲットに接続するには、srp_daemon または ibsrpdm に以下のコマンドを使用します。**

```
[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 の出力には、SRP 接続操作が失敗した理由が表示されます。後のステップでは、ターゲット側で dmesg コマンドを使用して状況をクリアします。**

```
[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 が失敗したため、lsscsi コマンドの出力は前の手順の出力と同じです。**

```
[root@initiator]# lsscsi
[0:0:10:0]   disk    IBM-ESXS ST9146803SS      B53C  /dev/sda
```

10. **ターゲット側で dmesg (ib_srpt.ko)を使用すると、LOGIN が失敗した理由について説明しています。また、出力には、srp_daemon:0x7edd77000075110000l175000077d708 が提供する有効な ACL ID が含まれます。**

```
[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=0xfe80000000000000:0x1175000077dd7e)
[ 1200.322207] ib_srpt Rejected login because no ACL has been configured yet for initiator
0x7edd77000075110000l175000077d708.
```

11. **targetcli ツールを使用して、有効な ACL を追加します。**

```
[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.fe8000000000000000l175000077dd7e/acls create
0x7edd77000075110000l175000077d708
Created Node ACL for ib.7edd77000075110000l175000077d708
Created mapped LUN 0.
```

12. **SRP LOGIN 操作を確認します。**

    a. **srp_daemon がログインを再試行できるように 60 秒待機します。**

    ```
    [root@initiator]# sleep 60
    ```

b.

SRP LOGIN 操作を確認します。

```
[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 ターゲット検出のカーネルログに、以下を使用します。

```
[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. サブネットマネージャーの設定

13.6.1. 必要性の判断

ほとんどの InfiniBand スイッチには、組み込み型サブネットマネージャーが備わっています。 ただし、スイッチファームウェア内のものよりも最新のサブネットマネージャーが必要な場合や、スイッチマネージャーが許可するよりも完全な制御が必要な場合は、Red Hat Enterprise Linux 7 には opensm サブネットマネージャーが含まれています。 InfiniBand ネットワークが機能するには、そのネットワークにサブネットマネージャーが備わっている 必要があります。 スイッチがないマシン 2 台で簡単なネットワークを設定し、カードが背中合わせに差し込まれている場合でも、カード上にリンクが現れるにはサブネットマネージャーが必要になります。サブネットマネージャーが複数あることもあり、その場合は 1 つがコントローラーとして動作し、その他はこのコントローラーに障害が発生した際にポートとして引き継ぐことになります。

13.6.2. opensm の主要な設定ファイルの設定

opensm プログラムは、主要な設定ファイルを /etc/rdma/opensm.conf に保持します。ユーザーはこのファイルをいつでも編集でき、変更内容はアップグレードで維持されます。 このファイル自体に

も、オプションについての詳しい説明があります。ただし、GUID をバインドするための最も一般的な編集と、実行する PRIORITY の設定には、opensm.conf ファイルは編集されず、/etc/sysconfig/opensm ファイルを編集することが強く推奨されます。ベースの /etc/rdma/opensm.conf ファイルに編集がない場合は、opensm パッケージがアップグレードされるたびにアップグレードされます。このファイルには新たなオプションが定期的に追加されるので、この方法だと現行の設定を最新のものに容易に維持できます。opensm.conf ファイルが変更された場合は、アップグレード時に新しいオプションを編集されたファイルにマージする必要がある場合があります。

13.6.3. opensm スタートアップオプションの設定

/etc/sysconfig/opensm ファイルのオプションは、サブネットマネージャーが実際にどのように起動するか、起動するサブネットマネージャーのコピー数を制御します。たとえば、各ポートが物理的に別個のネットワークに差し込まれているデュアルポートの InfiniBand カードは、各ポートで実行中のサブネットマネージャーのコピーを必要とします。opensm サブネットマネージャーは、アプリケーションのインスタンスごとに 1 つのサブネットのみを管理し、管理する必要のあるサブネットごとに 1 回起動する必要があります。さらに、複数の opensm サーバーがある場合は、どのサーバーがポートとコントローラーであるかを制御する優先度を各サーバーに設定します。

/etc/sysconfig/opensm ファイルは、サブネットマネージャーの優先度を設定し、サブネットマネージャーがバインドする GUID を制御する簡単な方法を提供するために使用されます。/etc/sysconfig/opensm ファイル自体には、オプションを広範囲に説明しています。ユーザーは、opensm のフェイルオーバーおよびマルチファブリック操作を有効にするためには、ファイル自体の指示だけを読んでください。

13.6.4. P_Key 定義の作成

デフォルトでは、opensm.conf は /etc/rdma/partitions.conf ファイルを検索し、ファブリックで作成するパーティションの一覧を取得します。すべてのファブリックには 0x7fff サブネットが含まれ、すべてのスイッチとすべてのホストがそのファブリックに属している必要があります。これに加えて他のパーティションを作成することが可能で、すべてのスイッチやホストがこれらの新たなパーティションのメンバーである必要はありません。これにより管理者は、InfiniBand ファブリック上のイーサネットの VLAN に似たサブネットを作成できます。パーティションが 40 Gbps などの特定のスピードで定義されていて、ネットワーク上に 40 Gbps を実行できないホストがある場合、そのホストはスピードに合致できないので、パーミッションがあってもこのパーミッションに参加することはできません。このため、パーティションのスピードは、そのパーティションに参加を許可されているホストのなかで一番遅いものに設定することが推奨されます。ホストのいずれかのサブネット用により高速のパーティションが必要な場合は、高速パーティションを別個に作成します。

以下のパーティションファイルでは、デフォルトの 0x7ff パーティションは 10 Gbps のスピードになり、0x0002 のパーティションは 40 Gbps のスピードになります。

```
~]$ 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 の有効化

ユーザーは、インストール時にデフォルトで有効になっていないため、opensm サービスを有効にする必要があります。root で以下のコマンドを発行します。

```
~]# systemctl enable opensm
```

13.7. 初期の INFINIBAND RDMA 操作のテスト

注記

このセクションは、InfiniBand デバイスにのみ適用されます。iWARP および RoCE/IBoE デバイスは IP ベースのデバイスであるため、IPoIB が設定され、デバイスに IP アドレスが設定されると、ユーザーは RDMA 操作のテストのセクションに進んでください。

rdma サービスが有効になり、（必要な場合） opensm サービスが有効になり、特定のハードウェアに適切なユーザー空間ライブラリーがインストールされると、ユーザー空間の rdma 操作が可能になります。libibverbs-utils パッケージの簡単なテストプログラムが、RDMA が適切に機能しているかどうかの判断に役立ちます。ibv_devices プログラムは、システムに存在するデバイスを示し、ibv_devinfo コマンドは各デバイスに関する詳細情報を提供します。以下に例を示します。

```
~]$ 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 および ibstat コマンドは、若干異なる情報を出力します（たとえば、ポート MTU は ibv_devinfo にあり、ibstat 出力には存在せず、ポート GUID は ibstat 出力に存在しますが

ibv_devinfo 出力には存在しません）、いくつかの名前が異なります(Base など)。 ibstat 出力の *ローカル識別子* (LID)は、ibv_devinfoの port_lid 出力と同じです）

infiniband-diags パッケージからの ibping などの単純な ping プログラムを使用して、RDMA の接続性をテストできます。ibping プログラムはクライアントサーバーモデルを使用します。まず、あるマシンで ibping サーバーを起動し、次に別のマシンでクライアントとして ibping を実行し、ibping サーバーに接続するように指示する必要があります。ベースの RDMA 機能をテストするため、サーバーの指定には、IP アドレスの代わりに RDMA 固有のアドレス解決方法を使用する必要があります。

サーバーマシンでは、ユーザーは ibv_devinfo および ibstat コマンドを使用して、テストするポートの port_lid （または Base lid）およびポート GUID を出力できます（上記のインターフェイスのポート 1 と仮定し、port_lid/Base LID が 2 で、Port GUID が 0xf4521403007bcba1であると仮定します）。次に、必要なオプションで ibping を開始して、特にテストするカードとポートにバインドし、ibping をサーバーモードで実行する必要があります。-? または --help を渡すと、ibping に利用可能なオプションが表示されますが、-S オプションまたは --Server オプションのいずれかが必要になります。また、特定のカードとポートにバインドするには、-C または --Ca および -P または --Port のいずれかが必要です。このインスタンスのポートはネットワークポート番号を表示せず、マルチポートカードを使用する際のカード上の物理的ポート番号を表示します。たとえば、マルチポートカードの 2 番目のポートを使用して RDMA ファブリックへの接続をテストするには、ibping にカード上のポート 2 にバインドするように指示する必要があります。単一ポートカードを使用している場合、もしくはカード上の最初のポートをテストする場合は、このオプションは不要です。以下に例を示します。

```
~]$ ibping -S -C mlx4_1 -P 1
```

次に、クライアントマシンに切り替え、ibping を実行します。サーバー ibping プログラムがバインドされているポートのポート GUID、またはサーバーの ibping プログラムがバインドされているポートの *ローカル識別子* (LID)をメモします。また、サーバーでバインドされているカードとポートと同じネットワークに、クライアントマシンのどのカードとポートが物理的に接続されているかをメモします。たとえば、サーバー上の最初のカードの 2 番目のポートにバインドされていて、そのポートが 2 番目の RDMA ファブリックに接続されている場合、クライアント上でその 2 番目のファブリックに接続する必要のあるカードとポートを指定します。これらがわかったら、ibping プログラムをクライアントとして実行し、サーバー上で収集されたポート LID または GUID を接続先のアドレスとして使用してサーバーに接続します。以下に例を示します。

```
~]$ 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
```

または

```
~]$ 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
```

この出力は、エンドツーエンドの RDMA 通信がユーザースペースのアプリケーションで機能していることを証明しています。

以下のエラーが表示される場合もあります。

```
~]$ ibv_devinfo
libibverbs: Warning: no userspace device-specific driver found for
/sys/class/infiniband_verbs/uverbs0
No IB devices found
```

このエラーは、必要なユーザースペースのライブラリーがインストールされていないことを示しています。管理者は「InfiniBand および RDMA に関連するソフトウェアパッケージ」に掲載されているユーザースペースのライブラリー (該当ハードウェアに適したもの) のいずれかをインストールする必要があります。まれに、ユーザーがドライバーまたは libibverbs に間違ったアーキテクチャータイプをインストールすると、これが発生する可能性があります。たとえば、libibverbs が arch x86_64 で、libmlx4 がインストールされているが i686 タイプの場合は、このエラーが生じる可能性があります。

注記

多くのサンプルアプリケーションでは、サーバーとクライアント間の通信を開く際に、LID ではなくホスト名もしくはアドレスの使用が好まれます。これらのアプリケーションでは、エンドツーエンドの RDMA 通信をテストする前に IPoIB をセットアップする必要があります。ibping アプリケーションは、アドレス指定形式として単純な LID を受け入れるという点で非意味です。これにより、テストシナリオから IPoIB アドレス指定の問題の可能性をなくし、単純な RDMA 通信が機能しているかどうかを切り離すための簡単なテストが可能になります。

13.8. IPOIB の設定

13.8.1. IPoIB のロールについて

「IP ネットワークと非 IP ネットワークの比較」 で説明したように、ほとんどのネットワークは IP ネットワークです。InfiniBand は違います。IPoIB のロールは、InfiniBand RDMA ネットワークの上に IP ネットワークエミュレーション層を提供することです。これにより、既存のアプリケーションが変更無しで InfiniBand ネットワーク上で実行できるようになります。しかし、これらのアプリケーションのパフォーマンスは、RDMA 通信をネイティブで使用するように作成されたアプリケーションの場合よりもかなり低いものになります。ほとんどのネットワークには最大級のパフォーマンスを必要とするアプリケーションと、低パフォーマンスを受け入れられるアプリケーションの 2 種類があります。後者は通常、RDMA 通信のような新しい通信方法を使用するように更新されないため、IPoIB はこれらのアプリケーション用に引き続き利用可能となっています。

iWARP ネットワークと RoCE/IBoE ネットワークはいずれも実際には IP リンク層の上に RDMA を階層化した IP ネットワークであるため、IPoIB は必要ありません。このため、カーネルは iWARP や

RoCE/IBoE RDMA デバイスの上に IPoIB デバイスを作成することは拒否します。

13.8.2. IPoIB 通信モードについて

IPoIB デバイスは、datagram または connected のモードで実行するよう設定できます。違いは、通信先のマシンで IPoIB 層が開こうとする queue pair のタイプです。datagram モードでは、信頼できない未接続の queue pair が開かれます。connected モードでは、信頼性のある接続済みの queue pair が開かれます。

datagram モードの使用時には、信頼性のない未接続の queue pair タイプが InfiniBand リンク層の MTU よりも大きいパケットを許可しません。IPoIB レイヤーは、送信中の IP パケット上に 4 バイトの IPoIB ヘッダーを追加します。これにより、IPoIB MTU は InfiniBand リンク層の MTU よりも 4 バイト小さくなければなりません。InfiniBand リンク層の MTU は通常 2048 バイトなので、datagram モードでの一般的な IPoIB デバイス MTU は 2044 バイトになります。

connected モード使用時には、信頼性のある接続済み queue pair タイプが InfiniBand リンク層 MTU よりも大きいメッセージを許可し、ホストアダプターが各末端でパケットのセグメント化と再構築を処理します。このため、connected モード時に InfiniBand アダプターが送信する IPoIB メッセージにはサイズ制限が課せられません。ただし、IP パケットには 16 ビットサイズフィールドしかないため、最大バイト数として 65535 に制限されます。現実には、このサイズに適合しなくてはならない様々な TCP/IP ヘッダーも勘案する必要があるので、実際に許可される MTU はこれよりも小さくなります。その結果、必要なすべての TCP ヘッダーに十分なスペースを確保するために、Connected モードの IPoIB MTU は 65520 に制限されます。

connected モードオプションでのパフォーマンスは通常高いものとなりますが、消費するカーネルメモリーも多くなります。ほとんどのシステムでは、メモリー消費量よりもパフォーマンスの方が重視されるため、より一般的に使用されるのは connected モードになります。

しかしシステムが connected モードで設定されていても、マルチキャストトラフィックは依然として datagram モードで送信する必要があり (InfiniBand スイッチとファブリックは connected モードではマルチキャストトラフィックを通過させることができません)、さらに connected モードで設定されていないホストとの通信時には datagram モードにフォールバックする必要があります。マルチキャストデータを送信するプログラムを実行する際は、これらのプログラムがインターフェイス上で最大 MTU でマルチキャストデータの送信を試みるため、インターフェイスを datagram 操作用に設定するか、マルチキャストアプリケーションが送信パケットサイズを datagram のパケットサイズに収まるように制限する必要があります。

13.8.3. IPoIB ハードウェアアドレスについて

IPoIB デバイスには、20 バイトのハードウェアアドレスがあります。非推奨のユーティリティー ifconfig はすべての 20 バイトを読み取りできず、IPoIB デバイスの正しいハードウェアアドレスを見つけるために使用しないでください。iproute パッケージの ip ユーティリティーが適切に機能します。

IPoIB ハードウェアアドレスの最初の 4 バイトは、フラグと queue pair 番号です。次の 8 バイトはサブネットの接頭辞です。IPoIB デバイスが最初に作成されると、デフォルトのサブネット接頭辞は 0xfe:80:00:00:00:00:00:00 になり ます。このデバイスはサブネットマネージャーと連絡するまで、このデフォルトのサブネット接頭辞 (0xfe80000000000000) を使用し、その時点でサブネット接頭辞をサブネットマネージャーが設定したものに再設定します。最後の 8 バイトは、IPoIB デバイスの接続先となる InfiniBand ポートの GUID アドレスです。最初の 4 バイトと次の 8 バイトは時々変化するため、IPoIB インターフェイスのハードウェアアドレスを指定する際には、これを使ったり適合対象としたりすることはありません。セクション 「70-persistent-ipoib.rules の使用」 では、デバイスマッチングを確実に行えるように、udev ルールファイルの ATTR{address} フィールドの最初の 12 バイトを除外してアドレスを取得する方法を説明します。IPoIB インターフェイスを設定する際、設定ファイルの HWADDR フィールドには、すべての 20 バイトを含めることができます。ただし、最後の 8 バイトは通常、設定ファイルによって指定されたハードウェアにマッチしてこれを見つけるのに使用されます。ただし、TYPE=InfiniBand エントリーがデバイス設定ファイルで正しくスペルされず、ifup-ib が IPoIB インターフェイスを開くために使用される実際のスクリプトではない場合、設定で指定されたハードウェアを検出できないシステムに関するエラーが発生します。IPoIB インターフェイスの場合、設定ファイルの TYPE= フィールドは InfiniBand または infiniband のいずれかである必要があります（エントリーは大文字と小文字を区別しますが、スクリプトはこれらの 2 つの特定のスペルを受け入れます）。

13.8.4. InfiniBand P_Key サブネットについて

InfiniBand ファブリックは、異なる P_Key サブネットを使用することで、仮想サブネットに論理的にセグメント化できます。これは、イーサネットインターフェイス上で VLAN を使用することに非常に似ています。すべてのスイッチとホストはデフォルトの P_Key サブネットのメンバーである必要がありますが、管理者は追加のサブネットを作成し、それらのサブネットのメンバーをファブリック内のホストまたはスイッチのサブセットに制限できます。P_Key サブネットは、ホストがこれを使用する前にサブネットマネージャーが定義する必要があります。opensm サブネットマネージャーを使用して P_Key サブネットを定義する方法については、「P_Key 定義の作成」 セクションを参照してください。IPoIB インターフェイスの場合は、P_Key サブネットが作成されると、それらの P_Key サブネット専用の IPoIB 設定ファイルを追加で作成できます。イーサネットデバイスで VLAN インターフェイスと同様に、各 IPoIB インターフェイスは同じリンクを共有し、P_Key 値が異なる他の IPoIB インターフェイスと完全に異なるファブリックにあるかのように動作します。

IPoIB P_Key インターフェイスには特別な要件があります。すべての IPoIB P_Keyの範囲は 0x0000 から 0x7fff で、高いビット 0x8000 は、P_Key のメンバーシップが部分的なメンバーシップではなく完全なメンバーシップであることを示します。Linux カーネルの IPoIB ドライバーは、P_Key サブネットの完全なメンバーシップしかサポートしないため、Linux が接続可能なサブネットでは、P_Key 番号が高いビットが常に設定されます。つまり、Linux コンピューターが P_Key 0x0002 に参加すると、そのコンピューターの実際の P_Key 番号は 0x8002 になり、P_Key 0x0002 のメンバーであることが示唆されます。このため、セクション 「P_Key 定義の作成」 にあるように opensm partitions.conf ファイルに P_Key 定義を作成する場合は、0x8000 なしで P_Key 値を指定する必要がありますが、Linux クライアントで P_Key IPoIB インターフェイスを定義する場合は、ベースの P_Key 値に 0x8000 の値を追加します。

13.8.5. テキスト形式のユーザーインターフェイス nmtui による InfiniBand の設定

テキスト形式のユーザーインターフェイスツール nmtui を使用すると、ターミナルのウィンドウで InfiniBand を設定できます。このツールを起動するには、以下のコマンドを実行します。

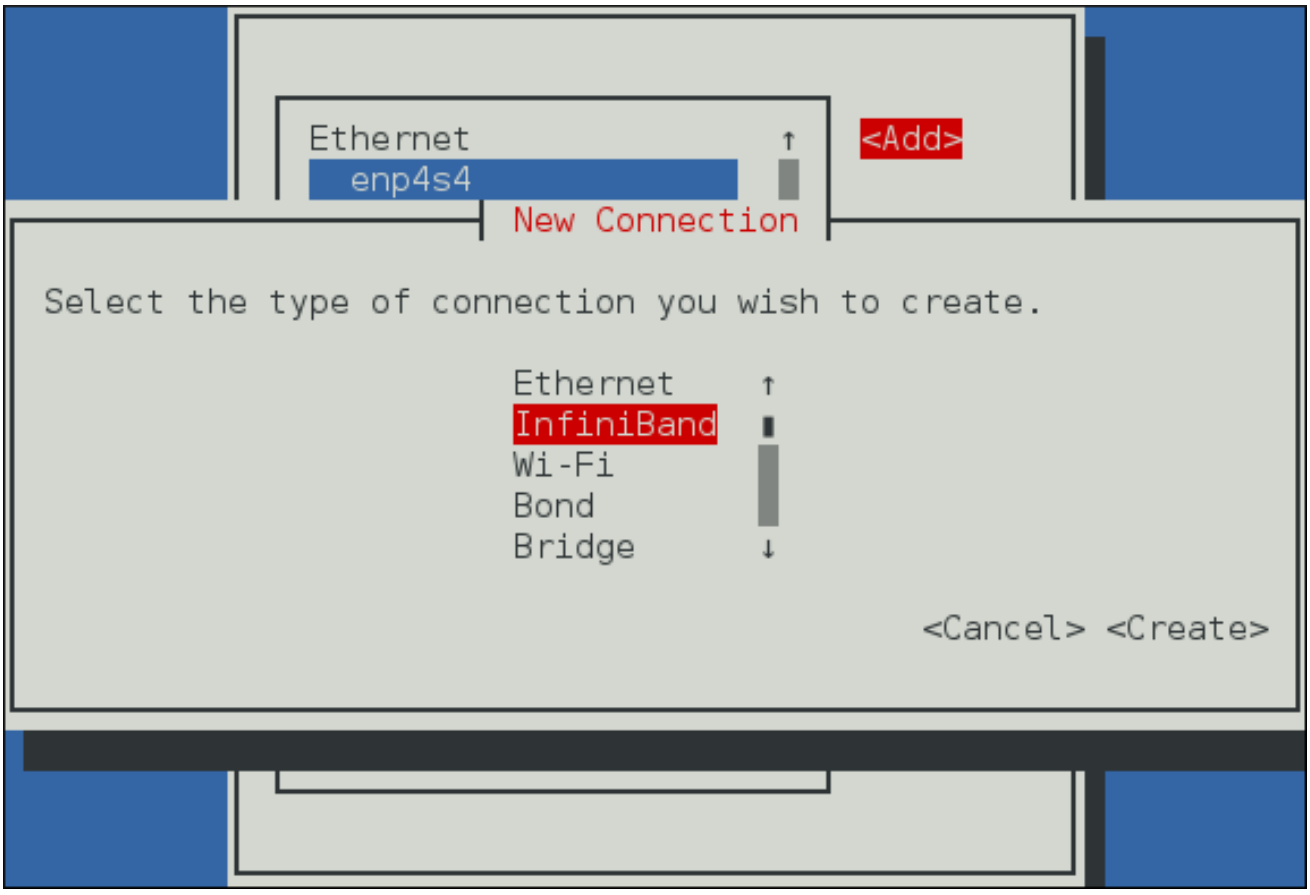
```
~]$ nmtui
```

テキストユーザーインターフェイスが表示されます。無効なコマンドがあると、使用方法に関するメッセージが表示されます。

移動するには、矢印キーを使用するか、Tab を押して順方向に進み、Shift+Tab を押してオプションを再度実行します。Enter を押してオプションを選択します。Space バーは、チェックボックスのステータスを切り替えます。

メニューから 接続の編集 を選択します。Add を選択すると、New Connection 画面が開きます。

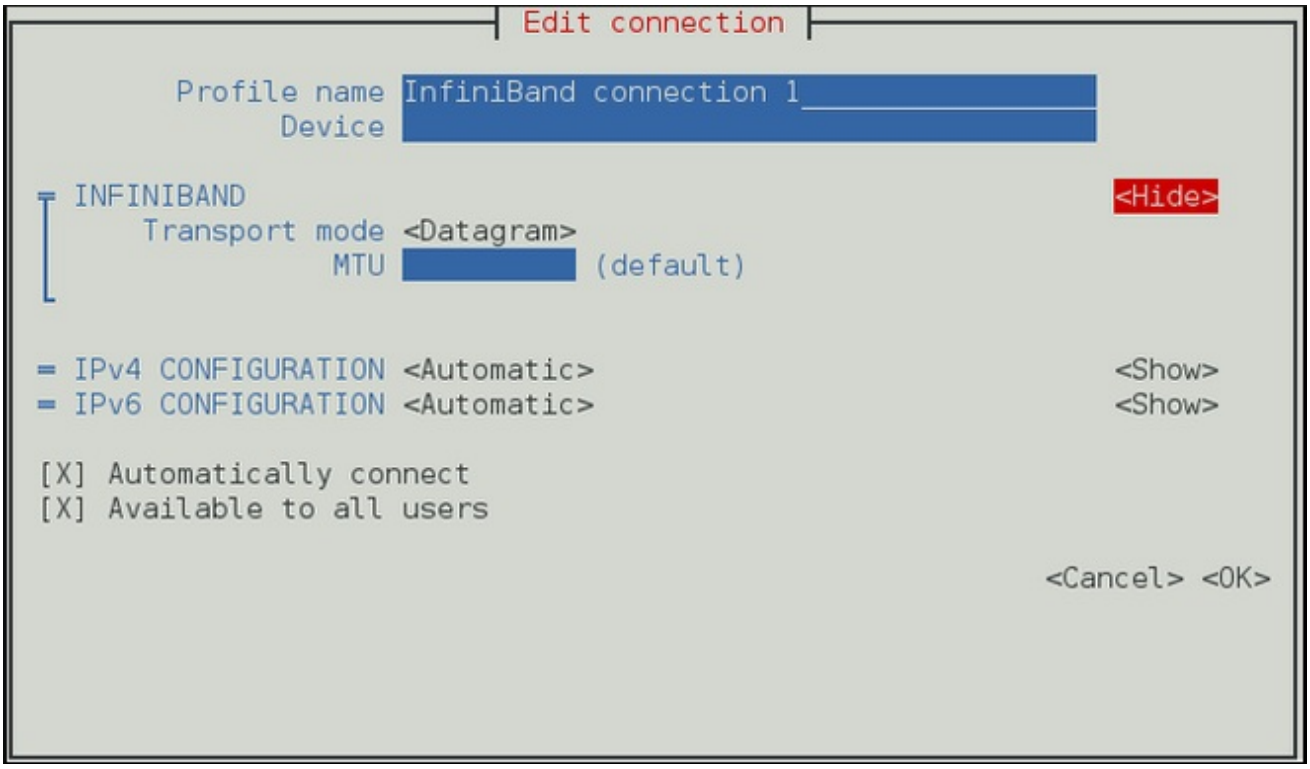
図13.1 NetworkManager テキスト形式のユーザーインターフェイスの InfiniBand 接続追加メニュー



[D]

InfiniBand を選択すると、接続の編集 画面が開きます。画面のプロンプトに従って設定を完了します。

図13.2 NetworkManager テキスト形式ユーザーインターフェイスで InfiniBand 接続を設定するメニュー



[D]

InfiniBand 用語の定義については、「InfiniBand タブの設定」を参照してください。

nmtui のインストール方法は、「nmtui を使用した IP ネットワークの設定」 を参照してください。

13.8.6. コマンドラインツール nmcli での IPoIB の設定

まず、デフォルトの IPoIB デバイスの名前変更が必要かどうかを決定し、その場合は、「70-persistent-ipoib.rules の使用」 セクションの手順に従って、udev の名前ルールを使用してデバイスの名前を変更します。以下のように ib_ipoib カーネルモジュールを削除してからリロードすることで、再起動せずに IPoIB インターフェイスの名前を強制的に変更できます。

```
~]$ rmmod ib_ipoib
~]$ modprobe ib_ipoib
```

デバイスに必要な名前が付けられたら、nmcli ツールを使用して IPoIB インターフェイスを作成します。以下に 2 とおりの方法を示します。

例13.3 2 つの別々のコマンドによる IPoIB の作成および修正

```
~]$ 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:f4:52:14:03:00:7b:cb:a3
nmcli> save
Connection 'mlx4_ib3' (8029a0d7-8b05-49ff-a826-2a6d722025cc) successfully updated.
nmcli> quit
```

または、以下のように **nmcli c add** および **nmcli c modify** を 1 つのコマンドで実行することができます。

例13.4 1 つのコマンドによる IPoIB の作成および修正

```
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:f4:52:14:03:00:7b:cb:a3
```

この時点で、**mlx4_ib0** という名前の **IPoIB** インターフェイスが作成され、**Connect** モードを使用するように設定され、最大接続モード **MTU**、**IPv4** および **IPv6** の **DHCP** を使用するように設定されています。**IPoIB** インターフェイスをクラスタートラフィックに使用し、イーサネットインターフェイスをクラスター以外の通信に使用する場合は、デフォルトルートと **IPoIB** インターフェイス上のデフォルトネームサーバーの無効化が必要になる可能性があります。これは、以下のように実行できます。

```
~]$ 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 インターフェイスが必要な場合は、以下のように nmcli を使用して作成します。

```
~]$ 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. コマンドラインを使用した IPoIB の設定

まず、デフォルトの IPoIB デバイスの名前変更が必要かどうかを決定し、その場合は、「70-persistent-ipoib.rules の使用」 セクションの手順に従って、udev の名前ルールを使用してデバイスの名前を変更します。以下のように ib_ipoib カーネルモジュールを削除してからリロードすることで、再起動せずに IPoIB インターフェイスの名前を強制的に変更できます。

```
~]$ rmmod ib_ipoib
~]$ modprobe ib_ipoib
```

デバイスに必要な名前が付けられたら、管理者は希望するエディターで ifcfg ファイルを作成し、デバイスを制御できます。静的 IPv4 アドレス指定を使用する通常の IPoIB 設定ファイルは以下のように

なります。

```
~]$ 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 フィールドは、udev の名前ルールで作成したカスタム名と一致している必要があります。NAME エントリーはデバイス名に合致する必要はありません。GUI 接続エディターが起動していれば、NAME フィールドは現在使用している接続に使用されている名前になります。InfiniBand オプションが正常に処理されるには、TYPE フィールドは InfiniBand である必要があります。CONNECTED_MODE は yes または no です。yes は接続モードを使用し、通信にデータグラムモードを使用します（セクション 「IPoIB 通信モードについて」を参照）。

P_Key インターフェイスの場合、これは一般的な設定ファイルです。

```
~]$ 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
```

すべての P_Key インターフェイスファイルでは、PHYSDEV ディレクティブが必要で、親デバイス名である必要があります。PKEY ディレクティブは yes に設定し、PKEY_ID はインターフェイスの数(0x8000 メンバーシップビットの追加の有無なし)に設定する必要があります。ただし、デバイス名は、以下のように論理 OR 演算子を使用して、0x8000 メンバーシップビットと組み合わせた PKEY_ID の 4 桁の 16 進数表記である必要があります。

```
NAME=${PHYSDEV}.$((0x8000 | $PKEY_ID))
```

デフォルトでは、ファイルの PKEY_ID は 10 進数として処理され、16 進数に変換され、論理 OR 演算子 0x8000 を使用してデバイスに適した名前に到達しますが、ユーザーは数字に標準の 0x 接頭辞を追加して PKEY_ID を 16 進数で指定できます。

13.8.8. IPoIB 設定後の RDMA ネットワークテスト

IPoIB を設定したら、IP アドレスを使用して RDMA デバイスを指定できます。IP アドレスとホスト名を使用してマシンを指定することが一般的であるため、ほとんどの RDMA アプリケーションはこれを優先的または場合によっては、接続するリモートマシンまたはローカルデバイスを指定する方法になります。

IPoIB 層の機能をテストするには、標準の IP ネットワークテストツールを使用して IPoIB デバイスの IP アドレスを提供できます。たとえば、IPoIB デバイスの IP アドレス間の ping コマンドが機能するようになりました。

Red Hat Enterprise Linux には qperf および perftest という、2 つの異なる RDMA パフォーマンステストパッケージが含まれています。これらのうちのどちらかを使って RDMA ネットワークをさらにテストできます。

ただし、perftest パッケージの一部であるアプリケーション、または qperf アプリケーションを使用する場合は、アドレス解決に特別な注意があります。リモートホストは IPoIB デバイスの IP アドレスまたはホスト名を使用して指定されますが、テストアプリケーションは実際に別の RDMA インターフェイスを介して接続できます。これは、ホスト名または IP アドレスを RDMA アドレスに変換するプロセスにより、2 つのマシン間の有効な RDMA アドレスペアを許可するためです。クライアントが複数の方法でサーバーに接続できる場合、プログラムは指定したパスに問題があれば、別のパスを選択することができます。たとえば、同じ InfiniBand サブネットに接続された各マシンに 2 つのポートがあり、各マシンの 2 番目のポートの IP アドレスが指定されている場合は、プログラムは各マシンの最初のポートが有効な接続方法であり、代わりにそれらを使用する可能性があります。この場合、「初期の InfiniBand RDMA 操作のテスト」 の ibping で行ったように、どの perftest プログラムでもコマンドラインオプションを使ってどのカードやポートをバインドするかを指示することで、テストが必要な特定のポートで確実にテストが行われるようになります。qperf では、ポートへのバインディング方法は若干異なります。qperf プログラムは 1 台のマシンでサーバーとして機能し、すべてのデバイス(RDMA 以外のデバイスを含む)をリッスンします。クライアントは、サーバーの有効な IP アドレスまたはホスト名を使用して qperf に接続できます。qperf は、最初にデータ接続を開き、クライアントのコマンドラインで指定された IP アドレスまたはホスト名で要求されたテストを実行しますが、そのアドレスに問題がある場合、qperf はクライアントとサーバー間の任意の有効なパスでテストの実行を試みます。このため、特定のリンクで qperf を強制的にテストするには、qperf クライアントに対して -loc_id オプションおよび -rem_id オプションを使用して、テストを特定のリンクで強制的に実行します。

13.8.9. GUI を使った IPoIB の設定

グラフィカルツールを使用して InfiniBand 接続を設定するには、nm-connection-editorを使用します。

手順13.4 nm-connection-editor を使用して新規 InfiniBand 接続を追加する

1. 端末に nm-connection-editor と入力します。

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

2. Add ボタンをクリックします。Choose a Connection Type ウィンドウが表示されます。InfiniBand を選択し、Create を クリックします。InfiniBand 接続 *1*の編集 ウィンドウが表示されます。

3. InfiniBand タブで、InfiniBand 接続に使用するトランスポートモードをドロップダウンリストから選択します。

4. InfiniBand MAC アドレスを入力します。

5. 設定を確認してから Save ボタンをクリックします。

6. 「InfiniBand タブの設定」を参照して、InfiniBand 固有のセッティングを編集します。

手順13.5 既存の InfiniBand 接続を編集する

既存の InfiniBand 接続を編集するには以下の手順に従います。

1. 端末に nm-connection-editor と入力します。

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

2. 編集する接続を選択し、Edit ボタンをクリックします。

3. General タブを選択します。

4. 接続名、自動接続の動作、および可用性のセッティングを設定します。

   編集 ダイアログの 5 つの設定は、すべての接続の種類に共通です。全般 タブ：

   - 接続名: ネットワーク接続のわかりやすい名前を入力します。この名前は、Network ウィンドウのメニューでこの接続を一覧表示するために使用されます。
   - Automatically connect to this network when it is available - このボックスを選択すると、NetworkManager が利用可能なときにこの接続に自動接続します。詳細は、「control-center を使用した既存の接続の編集」を参照してください。
   - All users can connect to this network - このボックスを選択すると、システム上のすべてのユーザーが 利用できる接続が作成されます。この設定を変更するには、root 権限が必要になる場合があります。詳細は、「GUI を使用したシステム全体およびプライベート接続プロファイルの管理」を参照してください。
   - Automatically connect to VPN when using this connection - このボックスを選択すると、NetworkManager が利用可能なときに VPN 接続に自動接続します。ドロップダウンメニューから VPN を選択します。
   - ファイアウォールゾーン - ドロップダウンメニューからファイアウォールゾーンを選択します。ファイアウォールゾーンに関する詳細情報は、『Red Hat Enterprise Linux 7 セキュリティーガイド』を参照してください。

5. 「InfiniBand タブの設定」を参照して、InfiniBand 固有のセッティングを編集します。

新規 (または修正した) 接続を保存して他の設定を行う

InfiniBand 接続の編集が終わったら、保存 ボタンをクリックしてカスタマイズした設定を保存します。

そして、以下のいずれかの設定をします。

- IPv4 の設定は、IPv4 のセッティング タブをクリックしてに進みます。 「IPv4 設定の設定」

  または

- IPv6 の設定は、IPv6 のセッティング タブをクリックして、「IPv6 セッティングの設定」に進みます。

13.8.9.1. InfiniBand タブの設定

新しい InfiniBand 接続をすでに追加している場合(手順13.4「nm-connection-editor を使用して新規 InfiniBand 接続を追加する」 を参照)、InfiniBand タブを編集して親インターフェイスと InfiniBand ID を設定できます。

トランスポートモード

ドロップダウンリストから、Datagram または Connected モードを選択できます。他の IPoIB ネットワークで使用しているモードと同じものを選びます。

Device MAC アドレス

InfiniBand ネットワークのトラフィックで使用される InfiniBand 対応デバイスの MAC アドレスです。InfiniBand ハードウェアがインストールされていれば、このハードウェアのアドレス フィールドは事前に記入されます。

MTU

InfiniBand 接続で送信されるパケットに使用する最大転送単位 (MTU) のサイズをオプションで設定します。

13.8.10. 関連情報

インストールされているドキュメント

- /usr/share/doc/initscripts-*version*/sysconfig.txt: 設定ファイルとそのディレクティブについて説明しています。

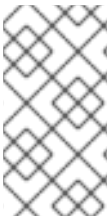
オンラインドキュメント

https://www.kernel.org/doc/Documentation/infiniband/ipoib.txt

IPoIB ドライバーの説明。関連する RFC への参照が含まれます。

パート IV. サーバー

このパートでは、ネットワークで通常必要とされるサーバーの設定方法を説明します。



注記

Web ブラウザーを使用してサーバーを監視および管理する方法は 『RHEL 7 で Web コンソールを使用したシステムの管理』 を参照してください。

第14章 DHCP サーバー

DHCP (Dynamic Host Configuration Protocol: 動的ホスト設定プロトコル) は、クライアントマシンに TCP/IP 情報を自動的に割り当てるネットワークプロトコルです。各 DHCP クライアントは、一元的に配置された DHCP サーバーに接続し、そのクライアントのネットワーク設定(IP アドレス、ゲートウェイ、DNS サーバーなど)を返します。

14.1. DHCP を使用する理由

DHCP は、クライアントネットワークインターフェイスの自動設定に便利です。クライアントシステムを設定する場合は、IP アドレス、ネットマスク、ゲートウェイ、または DNS サーバーを指定する代わりに、DHCP を選択できます。クライアントはこの情報を DHCP サーバーから取得します。また、DHCP は、多数のシステムの IP アドレスを変更する場合にも役立ちます。すべてのシステムを設定する代わりに、サーバー上の新しい IP アドレスのセット用に 1 つの設定ファイルを編集するだけです。組織の DNS サーバーが変更されると、DHCP クライアントではなく、DHCP サーバーで変更が行われます。ネットワークを再起動するかクライアントを再起動すれば、変更が反映されます。

機能している DHCP サーバーをネットワークに適切に接続している場合は、ラップトップおよびその他のモバイルコンピューターユーザーがこれらのデバイスをオフィスからオフィスに移動できます。

DNS および DHCP サーバーの管理者、およびプロビジョニングアプリケーションは、組織で使用されるホスト名の形式に同意する必要があることに注意してください。ホスト名の形式に関する詳細は、「推奨される命名プラクティス」を参照してください。

14.2. DHCP サーバーの設定

dhcp パッケージには、*Internet Systems Consortium* (ISC) DHCP サーバーが含まれています。root で パッケージをインストールします。

```
~]# yum install dhcp
```

dhcp パッケージをインストールすると、ファイル /etc/dhcp/dhcpd.conf が作成されます。これは、単なる空の設定ファイルです。root で以下のコマンドを実行します。

```
~]# cat /etc/dhcp/dhcpd.conf
#
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.example
#   see dhcpd.conf(5) man page
#
```

設定ファイルの例は、/usr/share/doc/dhcp-*version*;/dhcpd.conf.example にあります。/etc/dhcp/dhcpd.conf の設定に役立つこのファイルを使用する必要があります。これについては、以下を参照してください。

また、DHCP は /var/lib/dhcpd/dhcpd.leases ファイルを使用してクライアントのリースデータベースを保存します。詳細は、「リースデータベース」を参照してください。

14.2.1. 設定ファイル

DHCP サーバーの設定の最初の手順は、クライアントのネットワーク情報を保存する設定ファイルを作成することです。このファイルを使用して、クライアントシステムに対するオプションを宣言します。

設定ファイルには追加のタブや空白行が含まれているため、簡単に書式を整えることができます。キーワードは大文字と小文字を区別せず、ハッシュ記号(#)で始まる行はコメントとみなされます。

設定ファイルのステートメントには、次のような 2 つのタイプがあります。

- パラメーター: タスクの実行方法、タスクを実行するかどうか、クライアントに送信するネットワーク設定のオプションを規定します。
- 宣言 - ネットワークトポロジーの記述、クライアントの記述、クライアントのアドレス指定、宣言グループへのパラメーターグループの適用を行います。

キーワードオプションから始まるパラメーターは、*オプション* と呼ばれます。これらのオプションは DHCP オプションを制御します。一方、パラメーターはオプションではない値を設定するか、DHCP サーバーの動作を制御します。

中括弧で囲まれたセクションの前に宣言されたパラメーター（オプションを含む）は、グローバルパラメーターとみなされます。グローバルパラメーターは、これ以降のすべてのセクションに適用されます。

重要

設定ファイルが変更された場合、systemctl restart dhcpd コマンドを使用して DHCP デーモンを再起動するまで変更が反映されません。

注記

毎回 DHCP 設定ファイルを変更してサービスを再起動する代わりに、omshell コマンドを使用すると、DHCP サーバーへの接続、クエリー、設定の変更をインタラクティブに行うことができます。omshell を使用すると、サーバーの実行中にすべての変更を加えることができます。omshell の詳細は、omshell の man ページを参照してください。

例14.1「サブネットの宣言」 では、routers、subnet-mask、domain-search、domain-name-servers、および time-offset オプションがその中に宣言された ホスト ステートメントに使用されます。

提供されるすべてのサブネットと、DHCP サーバーが接続されるすべてのサブネットについて、subnet 宣言が 1 つ必要です。これは、アドレスがそのサブネットにあることを認識する方法を DHCP デーモンに指示します。subnet 宣言は、そのサブネットに動的に割り当てられるアドレスがない場合でも、各サブネットに必要です。

この例では、サブネット内のすべての DHCP クライアントと宣言された 範囲 にグローバルオプションがあります。クライアントには、範囲 内の IP アドレスが割り当てられます。

例14.1 サブネットの宣言

```
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;
}
```

サブネット内のシステムに動的 IP アドレスをリースする DHCP サーバーを設定するには、例14.2「Range パラメーター」 の例の値を変更します。これにより、クライアントのデフォルトのリース時間、最大リース時間、ネットワークの設定値を宣言します。以下の例では、範囲 192.168 .1.10 および 192.168.1.100 の IP アドレスをクライアントシステムに割り当てます。

例14.2 Range パラメーター

```
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;
}
```

ネットワークインターフェイスカードの MAC アドレスに基づいてクライアントに IP アドレスを割り当てるには、host 宣言内の hardware ethernet パラメーターを使用します。例14.3「DHCP を使用した静的 IP アドレス」 で示されているように、host apex 宣言は、MAC アドレス 00:A0:78:8E:9E:AA のネットワークインターフェイスカードが常に IP アドレス 192.168.1.4 を受信することを指定します。

オプションのパラメーターである host-name を使用すると、クライアントにホスト名を割り当てることができる点にも注目して下さい。

例14.3 DHCP を使用した静的 IP アドレス

```
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 7 は、InfiniBand IPoIB インターフェイスへの静的 IP アドレスの割り当てをサポートしています。ただし、これらのインターフェイスには通常のハードウェアイーサネットアドレスがないため、IPoIB インターフェイスに一意の ID を指定する別の方法を使用する必要があります。標準は、dhcp-client-identifier= construct オプションを使用して IPoIB インターフェイスの dhcp-client-identifier フィールドを指定することです。DHCP サーバーホストコンストラクトは、host スタンザごとに最大 1 つのハードウェアイーサネットと 1 つの dhcp-client-identifier エントリーをサポートします。ただし、複数の固定アドレスエントリーが存在する可能性があり、DHCP サーバーは、DHCP 要求が受信されたネットワークに適したアドレスに自動的に応答します。

例14.4 複数インターフェイスにおける DHCP を使用した静的 IP アドレス

たとえば、マシンが 2 つの InfiniBand インターフェイスと各物理インターフェイスの P_Key インターフェイスとイーサネット接続がある場合、次の静的 IP コンストラクトを使用してこの設定を提供できます。

```
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;
}
```

デバイスに適した dhcp-client-identifier を見つけるには、通常、接頭辞 ff:00:00:00:00:00:02:00:00:02:c9:00 を使用してから IPoIB インターフェイスの最後の 8 バイトを追加します(IPoIB インターフェイスが置かれている InfiniBand ポートの 8 バイトの GUID にもなります)。旧式のコントローラーのなかには、この接頭辞が適切でないものもあります。その場合は、DHCP サーバーで tcpdump を使用して受信 IPoIB DHCP 要求を取得し、そのキャプチャーから適切な dhcp-client-identifier を収集することが推奨されます。以下に例を示します。

```
]$ 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
```

上記のダンプでは、Client-ID フィールドが表示されています。hardware-type 255 は ID の最初の ff: に対応します。ID の残りの部分は、DHCP 設定ファイルに表示されるように、そのまま引用符で囲まれます。

同じ物理ネットワークを共有するすべてのサブネットは、例14.5「Shared-network 宣言」 のように shared-network 宣言内で宣言される必要があります。shared-network 内では、囲まれた subnet 宣言の外にあるパラメーターは、グローバルパラメーターと見なされます。shared-network に割り当てられた名前は、ネットワークの説明的なタイトルである必要があります。たとえば、「test-lab」 というタイトルを使用して、テストラボ環境のすべてのサブネットを記述します。

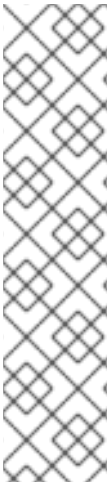
例14.5 Shared-network 宣言

```
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;
   }
}
```

例14.6「Group 宣言」 で説明したように、group 宣言を使用して、宣言のグループにグローバルパラメーターを適用します。たとえば、共有ネットワーク、サブネット、ホストをグループ化することができます。

例14.6 Group 宣言

```
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;
  }
}
```



注記

提供されている設定ファイルのサンプルをベースとして使用し、それにカスタムの設定オプションを追加できます。このファイルを適切な場所にコピーするには、root で以下のコマンドを実行します。

```
~]# cp /usr/share/doc/dhcp-version_number/dhcpd.conf.example /etc/dhcp/dhcpd.conf
```

... *version_number* は DHCP バージョン番号になります。

option ステートメントの完全なリストとそれらの機能については、dhcp-options (5) の man ページを参照してください。

14.2.2. リースデータベース

DHCP サーバーでは、/var/lib/dhcpd/dhcpd.leases ファイルが DHCP クライアントのリースデータベースを保存します。このファイルは変更しないでください。最近割り当てられた各 IP アドレスの DHCP リース情報は、自動的にリースデータベースに保存されます。この情報には、リースの長さ、IP アドレスが割り当てられているユーザー、リースの開始日と終了日、リースの取得に使用されたネットワークインターフェイスカードの MAC アドレスが含まれます。

リースデータベースの時刻はすべて、現地時間でなく協定世界時 (UTC) を使用します。

リースデータベースは、サイズが大きくなり過ぎるのを避けるために適宜再作成されます。最初に、すべての既知のリースは一時的なリースデータベースに保存されます。dhcpd.leases ファイルの名前が dhcpd.leases~ に変更され、一時リースデータベースが dhcpd.leases に書き込まれます。

DHCP デーモンは、リースデータベースがバックアップファイルに名前が変更された後、新しいファイルが書き込まれる前に、リースデータベースの名前がバックアップファイルに変更された後に、システムがクラッシュする可能性があります。この場合、dhcpd.leases ファイルは存在しませんが、サービスを起動する必要があります。この際、新規のリースファイルを作成しないでください。作成すると、それまでのリースはすべて失われ、多くの問題が発生します。正しい解決策は、dhcpd.leases~ バックアップファイルの名前を dhcpd.leases に変更し、デーモンを起動することです。

14.2.3. サーバーの起動と停止

重要

DHCP サーバーを初めて起動すると、dhcpd.leases ファイルが存在しないと失敗します。コマンド touch /var/lib/dhcpd/dhcpd.leases を使用して、ファイルが存在しない場合は作成できます。同じサーバーが DNS サーバーとして BIND も実行している場合は、named サービスを起動すると dhcpd.leases ファイルが自動的にチェックされるため、この手順は必要ありません。

これまで稼働していたシステムで、新規のリースファイルを作成しないでください。作成すると、それまでのリースはすべて失われ、多くの問題が発生します。正しい解決策は、dhcpd.leases~ バックアップファイルの名前を dhcpd.leases に変更し、デーモンを起動することです。

DHCP サービスを起動するには、次のコマンドを使用します。

```
systemctl start dhcpd.service
```

DHCP サーバーを停止するには、以下を入力します。

```
systemctl stop dhcpd.service
```

デフォルトでは、DHCP サービスは起動時に開始されません。ブート時にデーモンが自動的に起動するように設定する方法については、『Red Hat Enterprise Linux システム管理者のガイド』 を参照してください。

システムに複数のネットワークインターフェイスが接続されており、DHCP サーバーはいずれかのインターフェイスでのみ DHCP 要求をリッスンするようにするには、DHCP サーバーがそのデバイスでのみリッスンするように設定します。DHCP デーモンは、/etc/dhcp/dhcpd.conf ファイルで subnet 宣言を見つけるインターフェイスのみをリッスンします。

これは、ネットワークカードが 2 つあるファイアウォールマシンで役立ちます。インターネットへの IP アドレスを取得するために、1 つのネットワークカードを DHCP クライアントとして設定できます。他のネットワークカードは、ファイアウォールの背後にある内部ネットワークの DHCP サーバーとして使用できます。内部ネットワークに接続されたネットワークカードのみを指定するとユーザーはインターネット経由でデーモンに接続できないため、システムをよりセキュアにすることができます。

コマンドラインオプションを指定するには、root ユーザーとして dhcpd.service ファイルをコピーして編集します。例を示します。

```
~]# cp /usr/lib/systemd/system/dhcpd.service /etc/systemd/system/
~]# vi /etc/systemd/system/dhcpd.service
```

セクション **[Service]:**

```
ExecStart=/usr/sbin/dhcpd -f -cf /etc/dhcp/dhcpd.conf -user dhcpd -group dhcpd --no-pid
your_interface_name(s)
```

の行を編集します。次に、root ユーザーとしてサービスを再起動します。

```
~]# systemctl --system daemon-reload
~]# systemctl restart dhcpd
```

/etc/systemd/system /dhcpd.service ユニットファイルの [Service] セクションにある ExecStart=/usr/sbin /dhcpd にコマンドラインオプションを追加することができます。使用可能なオプションには、以下のものがあります。

- -p *portnum* - dhcpd がリッスンする UDP ポート番号を指定します。デフォルト値はポート 67 です。DHCP サーバーは、指定された UDP ポートよりも番号が 1 つある DHCP クライアントに応答を送信します。たとえば、デフォルトのポート 67 を使用する場合、サーバーはポート 67 でリクエストをリッスンし、ポート 68 にあるクライアントに応答します。ここでポートが指定され、DHCP リレーエージェントが使用される場合は、DHCP リレーエージェントがリッスンするのと同じポートを指定する必要があります。詳細は、「DHCP リレーエージェント」を参照してください。

- -f: フォアグラウンドプロセスとしてデーモンを実行します。これは主にデバッグ用に使用されます。

- -d: DHCP サーバーデーモンを標準エラー記述子に記録します。これは主にデバッグ用に使用されます。これが指定されていない場合、ログは /var/log/messages に書き込まれます。

- -cf *filename* : 設定ファイルの場所を指定します。デフォルトの場所は /etc/dhcp/dhcpd.conf です。

- -LF *filename* : リースデータベースファイルの場所を指定します。リースデータベースファイルがすでに存在する場合は、DHCP サーバーが起動するたびに同じファイルが使用されることが非常に重要です。このオプションは、実稼働環境以外のマシンでデバッグする目的にのみ使用することが強く推奨されます。デフォルトの場所は /var/lib/dhcpd/dhcpd.leases です。

- -q - デーモンの起動時に、著作権メッセージ全体を出力しません。

14.3. DHCP リレーエージェント

DHCP リレーエージェント(dhcrelay)は、DHCP サーバーのないサブネットから、他のサブネット上の 1 つ以上の DHCP サーバーに DHCP および BOOTP 要求のリレーを有効にします。

DHCP クライアントが情報を要求すると、DHCP リレーエージェントは起動時に指定された DHCP サーバーの一覧に要求を転送します。DHCP サーバーが応答を返すと、応答は元の要求を送信したネットワーク上でブロードキャストまたはユニキャストになります。

IPv4 用の DHCP リレーエージェントであるdhcrelay は、インターフェイスが /etc/sysconfig/dhcrelay で INTERFACES ディレクティブを使って指定されていない限り、すべてのインターフェイスで DHCPv4 および BOOTP 要求をリッスンします。「dhcrelay の DHCPv4 および BOOTP リレーエージェントとしての設定」を参照してください。IPv6 用の DHCP リレーエージェントであるdhcrelay6 には、このデフォルトの動作がないため、DHCPv6 要求をリッスンするインターフェイスを指定する必要があります。「dhcrelay を DHCPv6 リレーエージェントとして設定」を参照してください。

dhcrelay は、DHCPv4 および BOOTP リレーエージェント（デフォルト）として、または DHCPv6 リレーエージェント(-6 引数を使用)として実行できます。使用方法のメッセージを表示するには、dhcrelay -h コマンドを実行します。

14.3.1. dhcrelay の DHCPv4 および BOOTP リレーエージェントとしての設定

DHCPv4 および BOOTP モードで dhcrelay を実行するには、リクエストの転送先となるサーバーを指定します。root ユーザーとして dhcrelay.service ファイルをコピーして編集します。

```
~]# cp /lib/systemd/system/dhcrelay.service /etc/systemd/system/
~]# vi /etc/systemd/system/dhcrelay.service
```

セクション [Service] の ExecStart オプションを編集し、行の最後に 1 つ以上のサーバー IPv4 アドレスを追加します。以下に例を示します。

```
ExecStart=/usr/sbin/dhcrelay -d --no-pid 192.168.1.1
```

DHCP リレーエージェントが DHCP 要求をリッスンするインターフェイスも指定する場合は、-i 引数を指定して ExecStart オプションに追加します（そうしないと、すべてのインターフェイスでリッスンします）。以下に例を示します。他のオプションについては、dhcrelay (8) man ページを参照してください。

```
ExecStart=/usr/sbin/dhcrelay -d --no-pid 192.168.1.1 -i em1
```

加えられた変更を有効にするには、root ユーザーとしてサービスを再起動します。

```
~]# systemctl --system daemon-reload
~]# systemctl restart dhcrelay
```

14.3.2. dhcrelay を DHCPv6 リレーエージェントとして設定

DHCPv6 モードで dhcrelay を実行するには、-6 引数を追加し、（クライアントまたは他のリレーエージェントからクエリーを受信する）「『下層インターフェイス』」（クライアントと他のリレーエージェントからクエリーを転送する）を指定します。dhcrelay.service を dhcrelay6.service にコピーし、root ユーザーとして編集します。

```
~]# cp /lib/systemd/system/dhcrelay.service /etc/systemd/system/dhcrelay6.service
~]# vi /etc/systemd/system/dhcrelay6.service
```

[Service] add -6 引数で ExecStart オプションを編集し、「下層インターフェイスと上層インターフェイスインターフェイス」 を追加します（例：

```
ExecStart=/usr/sbin/dhcrelay -d --no-pid -6 -l em1 -u em2
```

）。他のオプションについては、dhcrelay (8) の man ページを参照してください。「」

加えられた変更を有効にするには、root ユーザーとしてサービスを再起動します。

```
~]# systemctl --system daemon-reload
~]# systemctl restart dhcrelay6
```

14.4. マルチホーム DHCP サーバーの設定

マルチホーム DHCP サーバーは、複数のネットワーク、つまり複数のサブネットを提供します。このセクションの例では、複数のネットワークを提供するように DHCP サーバーを設定する方法、リッスンするネットワークインターフェイスを選択する方法、およびネットワークを移動するシステムのネットワーク設定を定義する方法について詳しく説明します。

変更を行う前に、既存の /etc/dhcp/dhcpd.conf ファイルをバックアップします。

DHCP デーモンは、/etc/dhcp/dhcpd.conf ファイルで subnet 宣言を見つけるインターフェイスのみをリッスンします。

以下は、2 つのネットワークインターフェイスを持つサーバーの基本的な /etc/dhcp/dhcpd.conf ファイルです。enp1s0 10.0.0.0/24 ネットワーク enp2s0 172.16.0.0/24 ネットワークで。複数の subnet 宣言を使用すると、複数のネットワークに異なる設定を定義できます。

```
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 サーバーが提供しているネットワークごとに、subnet 宣言が必要です。複数のサブネットには複数の subnet 宣言が必要です。DHCP サーバーに subnet 宣言の範囲内のネットワークインターフェイスがない場合、DHCP サーバーはそのネットワークを提供しません。

subnet 宣言が 1 つしかなく、そのサブネットの範囲内にネットワークインターフェイスがない場合、DHCP デーモンは起動に失敗し、以下のようなエラーが /var/log/messages に記録されます。

```
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*;

option subnet-mask オプションは、サブネットマスクを定義し、subnet 宣言の netmask 値を上書きします。簡単なケースでは、サブネットとネットマスクの値は同じです。

option routers *10.0.0.1*;

option routers オプションは、サブネットのデフォルトゲートウェイを定義します。これは、システムが異なるサブネット上の内部ネットワーク、さらには外部ネットワークに届くために必要です。

range *10.0.0.5 10.0.0. 15*;

range オプションは、利用可能な IP アドレスのプールを指定します。システムには、指定された IP アドレスの範囲からアドレスが割り当てられます。

詳細は、dhcpd.conf (5) の man ページを参照してください。

警告

DHCP サーバーが IP 範囲からの IP アドレスを別の物理 Ethernet セグメントにした場合に設定の間違いを回避するため、共有ネットワーク宣言にこれ以上サブネットを含めいないようにしてください。

14.4.1. ホストの設定

変更を行う前に、既存の /etc/sysconfig/dhcpd ファイルおよび /etc/dhcp/dhcpd.conf ファイルをバックアップします。

複数ネットワークに対する単一システムの設定

以下の /etc/dhcp/dhcpd.conf の例では、2 つのサブネットを作成し、接続先のネットワークに応じて、同じシステムの IP アドレスを設定します。

```
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*

host 宣言は、IP アドレスなどの単一システムの特定のパラメーターを定義します。複数のホストに特定のパラメーターを設定するには、複数の host 宣言を使用します。

ほとんどの DHCP クライアントは host 宣言の名前を無視し、この名前は他の host 宣言に固有である限り、任意の名前にすることができます。複数のネットワークに同じシステムを設定するには、host 宣言ごとに異なる名前を使用します。そうしないと、DHCP デーモンが起動に失敗します。システムは、host 宣言の名前ではなく、hardware ethernet オプションで識別されます。

hardware ethernet *00:1A:6B:6A:2E:0B*;

hardware ethernet オプションは、システムを識別します。このアドレスを見つけるには、ip link コマンドを実行します。

fixed-address *10.0.0.20*;

fixed-address オプションは、hardware ethernet オプションで指定したシステムに有効な IP アドレスを割り当てます。このアドレスは、range オプションで指定した IP アドレスプール外でなければなりません。

option ステートメントがセミコロンで終了しない場合、DHCP デーモンは起動に失敗し、以下のようなエラーが /var/log/messages に記録されます。

```
/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
```

複数のネットワークインターフェイスを持つシステムの設定

以下の host 宣言は、複数のネットワークインターフェイスを持つ単一のシステムを設定し、各インターフェイスが同じ IP アドレスを受け取るようにします。両方のネットワークインターフェイスが同じネットワークに同時に接続されている場合には、この設定は機能しません。

```
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;
}
```

この例では、interface0 は最初のネットワークインターフェイスで、interface1 は 2 番目のインターフェイスです。異なる hardware ethernet オプションは、各インターフェイスを識別します。

このようなシステムが別のネットワークに接続する場合は、host 宣言をさらに追加します。以下の点に留意してください。

- ホストが接続しているネットワークに有効な fixed-address を割り当てます。
- host 宣言の名前を一意にします。

host 宣言で指定された名前が一意でない場合、DHCP デーモンは起動に失敗し、以下のようなエラーが /var/log/messages に記録されます。

```
dhcpd: /etc/dhcp/dhcpd.conf line 31: host interface0: already exists
dhcpd: }
dhcpd: ^
```

```
dhcpd: Configuration file errors encountered -- exiting
```

このエラーは、/etc/dhcp/dhcpd.conf に複数の host interface0 宣言が定義されているために生じました。

14.5. IPV6 の DHCP (DHCPV6)

ISC DHCP には、DHCPv6サーバー、クライアント、リレーエージェント機能を備えた 4.x リリース以降、IPv6 (DHCPv6)のサポートが含まれています。エージェントは IPv4 と IPv6 の両方をサポートしますが、エージェントは一度に管理できるのは 1 つのプロトコルのみです。デュアルサポートについては、IPv4 および IPv6 に対して個別に起動する必要があります。たとえば、それぞれの設定ファイル /etc/dhcp/dhcpd.conf および /etc/dhcp/dhcpd6.conf を編集して DHCPv4 と DHCPv6 の両方を設定してから、以下のコマンドを実行します。

```
~]# systemctl start dhcpd
~]# systemctl start dhcpd6
```

DHCPv6 サーバーの設定ファイルは、/etc/dhcp/dhcpd6.conf にあります。

サーバー設定ファイルの例は、/usr/share/doc/dhcp-*version*/dhcpd6.conf.example にあります。

単純な DHCPv6 サーバー設定ファイルは以下のようになります。

```
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";
}
```

ネットワークインターフェイスカードの MAC アドレスに基づいてクライアントに fixed-address を割り当てるには、hardware ethernet パラメーターを使用します。

```
host otherclient {
        hardware ethernet 01:00:80:a2:55:67;
        fixed-address6 3ffe:501:ffff:100::4321;
 }
```

IPv6 の shared-network および group 宣言の設定オプションは、IPV4 と同じです。詳細は、例 14.5「Shared-network 宣言」および 例14.6「Group 宣言」で示す例を参照してください。

14.6. IPV6 ルーター用 RADVD デーモンの設定

ルーター広告デーモン(radvd)は、IPv6 ステートレス自動設定に必要なルーター広告メッセージを送信します。これらの広告を元に、ユーザーはアドレス、設定、ルートを自動的に設定し、デフォルトのルーターを選択することができます。radvd デーモンを設定するには、以下を実行します。

1. radvd デーモンをインストールします。

```
~]# sudo yum install radvd
```

2. /etc/radvd.conf ファイルを設定します。以下に例を示します。

```
interface enp1s0
{
AdvSendAdvert on;
MinRtrAdvInterval 30;
MaxRtrAdvInterval 100;
prefix 2001:db8:1:0::/64
{
AdvOnLink on;
AdvAutonomous on;
AdvRouterAddr off;
};

};
```

注記

DNS リゾルバーをルーター広告とともに追加するには、 /etc/radvd.conf ファイルに RDNSS <ip> <ip> <ip> { }; オプションを追加します。サブネットに DHCPv6 サービスを設定するには、AdvManagedFlag を *on* に設定すると、ルーター広告を使用してクライアントが IPv6 サービスが利用可能な場合に IPv6 アドレスを自動的に取得できるようにします。DHCPv6 サービスの設定に関する詳細は、「IPv6 の DHCP (DHCPv6)」を参照してください。

3. **radvd デーモンを有効にします。**

```
~]# sudo systemctl enable radvd.service
```

4. **すぐに radvd デーモンを起動します。**

```
~]# sudo systemctl start radvd.service
```

ルーター広告パッケージの内容と、radvd デーモンによって送信された設定値を表示するには、radvdump コマンドを使用します。

```
~]# 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 デーモンの詳細は、radvd (8)、 radvd.conf (5)、radvdump (8) の man ページを参照してください。

14.7. DHCPV6 と RADVD の比較

IPv4 の動的ホスト設定は、主に DHCPv4 に適用されます。ただし、IPv6 では、以下のオプションを使用できます。

- 手動
- radvd デーモンの使用
- DHCPv6 サーバーの使用

手動

手動アドレス指定はいつでも利用できます。IPv6 アドレスをシステムに割り当てるには、「nmcli を使用したネットワーク接続」、「テキスト形式のユーザーインターフェイス nmtui を使ったボンディングの設定」、「ip コマンドを使用した IP ネットワークの設定」で説明しているツールを使用できます。

radvd デーモンの使用

標準仕様に準拠した IPv6 ネットワークは、ルーター通知を提供する必要があるため、ルーター通知デーモン (radvd) を実行して IPv6 設定オプションを適用できます。ルーター通知は、実際に物理 LAN でローカルに使用できる接頭辞に関するオンリンク情報を提供します。ルーター通知のほかに、手動 IPv6 設定、ルーター通知による自動 IPv6 設定、または動的ホスト設定プロトコル (DHCPv6) のいずれかを選択できます。radvd デーモンの設定に関する詳細は、「IPv6 ルーター用 radvd デーモンの設定」を参照してください。

DHCPv6 サーバーの使用

アドレス管理を中央管理している場合は、DHCPv6 サーバーを設定できます。DHCPv6 の可用性は、ルーター通知パケットのフラグにより通知されます。

表14.1 DHCPv6 と radvd の比較

DHCPv6	radvd
プライバシーを保護するために、ランダムなアドレスを保証します。	デフォルトゲートウェイに関する情報を入力します。

DHCPv6	radvd
クライアントに、さらなるネットワーク設定オプションを送信します。たとえば、ネットワークタイムプロトコル (NTP) サーバー、セッション開始プロトコル (SIP) サーバー、プレブート実行環境 (iPXE) 設定などです。	
MAC アドレスを IPv6 アドレスにマップします。	

注記

ネットワークを正しく設定するには、ルーター通知のみがデフォルトゲートウェイに関する情報を提供するため、radvd と組み合わせて DHCPv6 を使用します。

14.8. 関連情報

- dhcpd (8) man ページ： DHCP デーモンがどのように機能するかが説明されています。
- dhcpd.conf (5) man ページ： DHCP 設定ファイルの設定方法を説明し、例がいくつか含まれています。
- dhcpd.leases (5) man ページ：リースの永続データベースについて説明しています。
- dhcp-options (5) man ページ： dhcpd.conf で DHCP オプションを宣言する構文を説明します。以下に例を示します。
- dhcrelay (8) man ページ： DHCP リレーエージェントとその設定オプションを説明します。
- /usr/share/doc/dhcp-*version*/: DHCP サービスの現行バージョンに関するサンプルファイル、README ファイル、およびリリースノートが含まれています。

第15章 DNS サーバー

DNS（ドメイン名システム）は、ホスト名をそれぞれの IP アドレスに関連付けるために使用される分散データベースシステムです。ユーザーにとっては、ネットワーク上のマシンを名前で参照できるという利点があります。システム管理者は、ネームサーバー とも呼ばれる DNS *サーバーを使用すると*、*名前* ベースのクエリーに影響を与えずにホストの IP アドレスを変更できます。DNS データベースの使用は、IP アドレスをドメイン名に解決するためのだけでなく、DNSSEC がデプロイされるため、その使用は広範囲に及ぶようになりました。

15.1. DNS の概要

DNS は通常、特定のドメインに対して権限を持つ 1 つ以上の集中サーバーを使用して実装されます。クライアントホストがネームサーバーから情報を要求すると、ネームサーバーは通常ポート 53 に接続します。その後、ネームサーバーは要求された名前の解決を試行します。ネームサーバーが再帰ネームサーバーとして設定されていて権限のある回答がない場合、または以前のクエリーからキャッシュされた回答がない場合は、*ルートネームサーバー* と呼ばれる他のネームサーバーにクエリーを行い、問題の名前に対して権限のあるネームサーバーがどれか判断します。その後、要求した名前を取得するために権限のあるネームサーバーにクエリーを行います。再帰が無効となっており権限のあるネームサーバーとして設定されているものは、クライアント向けの検索を行いません。

15.1.1. ネームサーバーゾーン

DNS サーバーでは、すべての情報は *リソースレコード* (RR)と呼ばれる基本的なデータ要素に保存されます。リソースレコードは、RFC 1034 で定義されています。ドメイン名は、ツリー構造に構造化されています。階層の各レベルはピリオド（）で区切られ ます。例：で示されるルートドメイン 。 は DNS ツリーのルートで、レベル 0 になります。*トップレベルドメイン(TLD)と呼ばれるドメイン名 com は、ルートドメイン* (.)の子であるため、階層の最初のレベルになります。ドメイン名 example.com は、階層の第 2 レベルにあります。

例15.1 シンプルなリソースレコード

以下は、シンプルな *リソースレコード* (RR) の例になります。

```
example.com.      86400    IN        A          192.0.2.1
```

ドメイン名 example.com は、RR の *所有者* です。値 86400 は存続 *時間* (TTL)です。IN 文字(「インターネットシステムの を」 意味する)は、RR の *クラス* を示します。A の文字は、RR の *タイプ*を示します（この例ではホストアドレス）。ホストアドレス 192.0.2.1 は、この RR の最終セクションに含まれるデータです。この一行の例が 1 つの PR になります。タイプ、所有者、クラスが同一の RR のセットは、*リソースレコードセット* (RRSet) と呼ばれます。

ゾーンは *ゾーンファイル* を使用して権威ネームサーバー上で定義されます。これには、各ゾーンの

リソース記録の定義が含まれます。ゾーンファイルは、ファイルへの変更が行われる *プライマリーネームサーバー* (別名 *マスターネームサーバー*)、プライマリーネームサーバーからゾーン定義を受け取る *セカンダリーネームサーバー* (別名 *スレーブネームサーバー*) に保存されています。プライマリーネームサーバー、セカンダリーネームサーバーともゾーンに対し権威があり、クライアントには同一に見えます。設定により、どのネームサーバーも複数ゾーンに対してプライマリーサーバーまたはセカンダリーサーバーとして同時に機能することができます。

DNS および DHCP サーバーの管理者、およびプロビジョニングアプリケーションは、組織で使用されるホスト名の形式に同意する必要があることに注意してください。ホスト名の形式に関する詳細は、「推奨される命名プラクティス」を参照してください。

15.1.2. ネームサーバーの種類

ネームサーバーの設定タイプは 2 つ あります。

権威

権威ネームサーバーはゾーンの一部であるリソースレコードに対してのみ回答します。このカテゴリーには、プライマリー (マスター) ネームサーバーとセカンダリー (スレーブ) ネームサーバーの両方が含まれます。

recursive

再帰ネームサーバーは解決サービスを行いますが、どのゾーンに対しても権威ではありません。すべての解決への回答は一定期間はメモリーにキャッシュされ、取得したリソースレコードで指定されます。

ネームサーバーは権威的と同時に再帰的になることが可能ですが、これらの設定タイプを組み合わせることは推奨されません。権威サーバーが機能するには、これらが常にすべてのクライアントに利用可能となる必要があります。一方で、再帰的ルックアップは権威ある応答よりはるかに時間がかかるため、再帰的なサーバーは限られた数のクライアントにのみ利用可能とすべきです。それ以外の場合は、DDoS 攻撃 (分散型サービス拒否攻撃) の可能性が高まります。

15.1.3. ネームサーバーとしての BIND

BIND は一連の DNS 関連プログラムで設定されています。これには、と いう名前 のネームサーバー、rndc と呼ばれる管理ユーティリティー、dig と呼ばれるデバッグツールが含まれます。Red Hat Enterprise Linux でサービスを実行する方法の詳細は、『Red Hat Enterprise Linux システム管理者のガイド』 を参照してください。

15.2. BIND

このセクションでは、Red Hat Enterprise Linux に含まれる DNS サーバーである BIND (Berkeley Internet Name Domain)について説明します。ここでは、その設定ファイルの構造にフォーカスし、ローカルとリモートの両方での管理方法を記述しています。

15.2.1. 空白ゾーン

BIND は、再帰サーバーが不要なクエリーを処理できないインターネットサーバーに送信しないように、多くの 「空のゾーン」 を設定します（遅延を作成し、クエリーするクライアントに SERVFAIL 応答を作成します）。これらの空白ゾーンにより、遅延なしで権威のある NXDOMAIN 応答が返されます。設定オプション empty-zones-enable は、空のゾーンを作成するかどうかを制御します。一方、使用するデフォルトの接頭辞の一覧から 1 つ以上の空のゾーンを無効にするだけでなく、disable-empty-zone オプションを使用することができます。

『RFC 1918』 接頭辞用に作成された空のゾーンの数が引き上げられ、BIND 9.9 以降のユーザーは、empty-zones-enable が指定されていない場合（デフォルトは yes ）と明示的に yesに設定されている場合の両方で、『RFC 1918』 の空のゾーンが表示されます。

15.2.2. named サービスの設定

named サービスは起動時に、表15.1「named サービスの設定ファイル」 に記載されているファイルから設定を読み込みます。

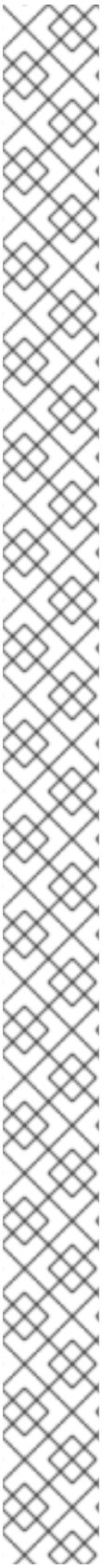
表15.1 named サービスの設定ファイル

パス	説明
/etc/named.conf	主要設定ファイル。
/etc/named/	主要設定ファイル内に含まれている設定ファイル用の補助ディレクトリー。

設定ファイルは、中括弧({ および })を開いて閉じることで囲むネストされたオプションを持つステートメントのコレクションで設定されています。ファイルを編集する際には、構文エラーを行わないように注意する必要があります。そうしないと、named サービスは起動しません。一般的な /etc/named.conf ファイルは、以下のように整理されています。

```
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;
};
```



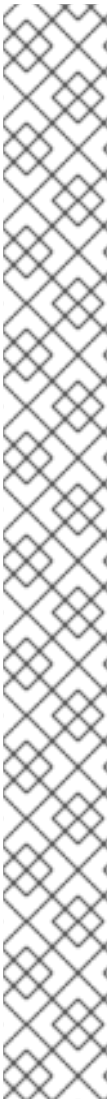
注記

bind-chroot パッケージをインストールしている場合、BIND サービスは chroot 環境で実行されます。その場合、初期化スクリプトは mount --bind コマンドを使用して上記の設定ファイルをマウントするため、この環境外で設定を管理できます。自動的にマウントされるため、/var/named/chroot/ ディレクトリーに何もコピーする必要はありません。chroot 環境で実行される場合に BIND 設定ファイルを特別な注意する必要がないため、メンテナーンスが簡素化されます。chroot 環境で BIND を実行した場合と同じように、すべてを編成できます。

/var/named/chroot/ の下の対応するマウントポイントディレクトリーが空の場合、以下のディレクトリーは /var/named/chroot/ ディレクトリーに自動的にマウントされます。

- **/etc/named**
- **/etc/pki/dnssec-keys**
- **/run/named**
- **/var/named**
- **/usr/lib64/bind または /usr/lib/bind （アーキテクチャーに依存します）。**

ターゲットファイルが /var/named/chroot/ に存在しない場合は、以下のファイルもマウントされます。



- /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

重要

chroot 環境でマウントされたファイルを編集するには、バックアップコピーを作成してから、元のファイルを編集する必要があります。別の方法では、「edit-a-copy」モードを無効にしてエディターを使います。たとえば、chroot 環境で実行されているときに BIND の設定ファイル /etc/named.conf を編集し、root で以下のコマンドを実行します。

```
~]# vim -c "set backupcopy=yes" /etc/named.conf
```

15.2.2.1. chroot 環境で BIND をインストールする

BIND を chroot 環境で実行するようにインストールするには、root で以下のコマンドを実行します。

```
~]# yum install bind-chroot
```

named-chroot サービスを有効にするには、最初に以下のコマンドを実行して named サービスが実行されているかどうかを確認します。

```
~]$ systemctl status named
```

実行中の場合は、無効にする必要があります。

named を無効にするには、root で以下のコマンドを実行します。

```
~]# systemctl stop named
```

```
~]# systemctl disable named
```

次に、named-chroot サービスを有効にするには、root で以下のコマンドを実行します。

```
~]# systemctl enable named-chroot
```

```
~]# systemctl start named-chroot
```

named-chroot サービスのステータスを確認するには、root で以下のコマンドを実行します。

```
~]# systemctl status named-chroot
```

15.2.2.2. 一般的なステートメントのタイプ

以下のタイプのステートメントは、/etc/named.conf で一般的に使用されます。

acl

acl (Access Control List) (アクセス制御リスト) ステートメントにより、ホストのグループを定義できるようになるため、それらのホストはネームサーバーへのアクセスを許可/拒否できるようになります。以下の形式を取ります。

```
acl acl-name {
  match-element;
  ...
};
```

acl-name **ステートメント名はアクセス制御リストの名前です。*match-element*** **オプションは、通常個別の IP アドレス(10 .0.1. 1)または *Classless Inter-Domain Routing* (CIDR)ネットワーク表記です（例：10.0.1.0/24）。定義済みのキーワードのリストは、表15.2「事前定義のアクセス制御リスト」を参照してください。**

表15.2 事前定義のアクセス制御リスト

キーワード	説明
any	**すべての IP アドレスにマッチします。**
localhost	**ローカルシステムが使用している IP アドレスにマッチします。**
localnets	**ローカルシステムが接続しているネットワーク上の IP アドレスにマッチします。**
none	**どの IP アドレスにも一致しません。**

acl ステートメントは、特に オプション などの他のステートメントと併用できます。例15.2「options と併用して acl を使用する」 2 つのアクセス制御リスト(black-hats と red-hats)を定義し、red-hats に通常のアクセスを付与する間に black-hats をブラックリストに追加します。

例15.2 options と併用して acl を使用する

```
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 ステートメントでは、/etc/named.conf にファイルを含めることができ、機密データを制限付きパーミッションが設定された別のファイルに配置できます。以下の形式を取ります。

```
include "file-name"
```

***file-name* ステートメント名はファイルへの絶対パスとなります。**

例15.3 ファイルを /etc/named.conf に含める

```
include "/etc/named.rfc1912.zones";
```

options

options ステートメントでは、グローバルサーバー設定オプションや他のステートメントのデフォルトを設定できます。名前付き 作業ディレクトリーの場所、許可されるクエリーのタイプなどを指定できます。以下の形式を取ります。

```
options {
  option;
  ...
};
```

よく使用される *option* ディレクティブのリストは、表15.3「一般的な設定オプション」を参照してください。

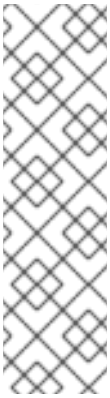
表15.3 一般的な設定オプション

オプション	説明
allow-query	**権限のあるリソースレコード用のネームサーバーにクエリーを許可されるホストを指定します。アクセス制御リスト、IP アドレスのコレクション、または CIDR 表記のネットワークのコレクションを受け入れます。デフォルトではすべてのホストが許可されています。**

オプション	説明
allow-query-cache	再帰クエリーなど権限の必要ないデータ用のネームサーバーにクエリーを許可されるホストを指定します。デフォルトでは、localhost と localnets のみが許可されます。
blackhole	ネームサーバーへのクエリーを許可されないホストを指定します。このオプションは、特定のホストやネットワークがサーバーに要求を集中させる時には使用すべきではありません。デフォルトのオプションは none です。
directory	named サービスの作業ディレクトリーを指定します。デフォルトのオプションは /var/named/ です。
disable-empty-zone	使用されるデフォルトの接頭辞リストから空白ゾーンを無効にするために使用します。options ステートメントおよび view ステートメントで指定することができます。複数回の使用が可能です。
dnssec-enable	DNSSEC 関連のリソースレコードを返すかどうかを指定します。デフォルトのオプションは yes です。
dnssec-validation	リソースレコードが DNSSEC を通じて本物であることを証明するかどうかを指定します。デフォルトのオプションは yes です。
empty-zones-enable	空白ゾーンを作成するかどうかを制御します。options ステートメントでのみ指定可能です。
forwarders	解決のために要求を転送するネームサーバーの有効な IP アドレス一覧を指定します。

オプション	説明
forward	forwarders ディレクティブの動作を指定します。以下のオプションを取ります。 • first: サーバーは、独自の名前の解決を試行する前に、forwarders ディレクティブにリストされているネームサーバーをクエリーします。 • only: forwarders ディレクティブに一覧表示されるネームサーバーをクエリーできない場合、サーバーは自身で名前の解決を試行しません。
listen-on	クエリーをリッスンする IPv4 ネットワークインターフェイスを指定します。ゲートウェイとしても機能する DNS サーバーでは、このオプションを使用して、1 つのネットワークからのみ発信されたクエリーに応答できます。デフォルトでは、すべての IPv4 インターフェイスが使用されます。
listen-on-v6	クエリーをリッスンする IPv6 ネットワークインターフェイスを指定します。ゲートウェイとしても機能する DNS サーバーでは、このオプションを使用して、1 つのネットワークからのみ発信されたクエリーに応答できます。デフォルトでは、すべての IPv6 インターフェイスが使用されます。
max-cache-size	サーバー用キャッシュとして使用されるメモリーの最大容量を指定します。最大値に到達すると、その限度を超過しないようにサーバーは記録が早期期限切れになるようにします。複数表示を持つサーバーでは、この制限は各表示のキャッシュ毎に別々に適用されます。デフォルトのオプションは 32M です。

オプション	説明
notify	あるゾーンが更新された時にセカンダリーネームサーバーに通知するかどうかを指定します。以下のオプションを取ります。 • **yes:** サーバーはすべてのセカンダリーネームサーバーに通知します。 • **no:** サーバーはどのセカンダリーネームサーバーにも通知し ません。 • **master-only:** サーバーはゾーンに対してのみプライマリーサーバーに通知します。 • **explicit -** サーバーは、ゾーンステートメント内の **also-notify** リストで指定されているセカンダリーサーバーにのみ通知します。
pid-file	named サービスによって作成されたプロセス ID ファイルの場所を指定します。
recursion	再帰的なサーバーとして動作するかどうかを指定します。デフォルトのオプションは **yes** です。
statistics-file	統計ファイルの代替の場所を指定します。/var/named/named.stats ファイルがデフォルトで使用されます。



注記

named がランタイムデータに使用するディレクトリーは、BIND のデフォルトの場所である /var /run/named/ から新しい場所 /run/named/ に移動しました。その結果、PID ファイルはデフォルトの場所 /var/run/named/named.pid から新しい場所 /run/named/named.pid に移動しました。さらに、session-key ファイルは /run/named/session.key に移動しました。これらの場所は、options セクションのステートメントで指定する必要があります。例15.4「オプションステートメントの使用」を参照してください。

重要

分散型サービス拒否(DDoS)攻撃を防ぐために、allow-query-cache オプションを使用して、クライアントの特定サブセットのみの再帰 DNS サービスを制限することが推奨されます。

利用可能なオプションの完全なリストは、「インストールされているドキュメント」 で参照されている 『BIND 9 Administrator Reference Manual』 および named.conf の man ページを参照してください。

例15.4 オプションステートメントの使用

```
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";
};
```

zone

zone ステートメントでは、設定ファイルやゾーン固有のオプションなど、ゾーンの特性を定義でき、グローバル options ステートメントを上書きするのに使用できます。以下の形式を取ります。

```
zone zone-name [zone-class] {
```

```
    option;
    ...
};
```

zone-name 属性はゾーンの名前で、zone-class はゾーンの オプション です。option は、表15.4「Zone ステートメントで一般的に使用されるオプション」で説明されているように zone ステートメントオプションになります。

zone-name 属性は、/var/named/ ディレクトリーにある対応するゾーンファイル内で使用される $ORIGIN ディレクティブに割り当てられるデフォルト値であるため、特に重要です。named デーモンは、ゾーンの名前を、ゾーンファイルにリストされている非完全修飾ドメイン名に追加します。たとえば、zone ステートメントが example.com の名前空間を定義する場合は、example.com を *zone-name* として使用して、example.com ゾーンファイル内のホスト名の最後に配置されるようにします。

ゾーンファイルの詳細は、「ゾーンファイルの編集」を参照してください。

表15.4 Zone ステートメントで一般的に使用されるオプション

オプション	説明
allow-query	このゾーンに関する情報要求が出来るクライアントを指定します。このオプションはグローバル allow-query オプションを上書きします。デフォルトではすべてのクエリー要求が許可されます。
allow-transfer	ゾーン情報の転送要求を許可されるセカンダリーサーバーを指定します。デフォルトでは、すべての転送要求が許可されています。
allow-update	自身のゾーン内で動的な情報更新を許可されるホストを指定します。デフォルトオプションでは、すべての動的更新要求は拒否されます。 ホストがゾーンについての情報を更新可能とするには注意が必要です。サーバーが信頼できるネットワークにありない限り、このオプションで IP アドレスを設定しないでください。代わりに、「Transaction SIGnatures トランザクション署名 (TSIG)」の説明に従って TSIG キーを使用します。

オプション	説明
file	ゾーンの設定データが含まれる named 作業ディレクトリー内のファイルの名前を指定します。
masters	権威ゾーン情報を要求する IP アドレスを指定します。このオプションは、ゾーンが type slave として定義されている場合にのみ使用されます。
notify	あるゾーンが更新された時にセカンダリーネームサーバーに通知するかどうかを指定します。以下のオプションを取ります。 • yes: サーバーはすべてのセカンダリーネームサーバーに通知します。 • no: サーバーはセカンダリーネームサーバーに通知しません。 • master-only: サーバーはゾーンに対してのみプライマリーサーバーに通知します。 • explicit: サーバーは、ゾーンステートメント内の also-notify リストで指定したセカンダリーサーバーのみを通知します。

オプション	説明
type	ゾーンのタイプを指定します。以下のオプションを取ります。 • delegation-only: COM、NET、ORG などのインフラストラクチャーゾーンの委譲ステータスを強制します。明示的な委譲または暗黙的な委譲なしで受信される回答は、NXDOMAIN として扱われます。このオプションは、再帰的あるいはキャッシング実装で使用される TLD もしくは root のゾーンのファイルにのみ適用されます。 • forward: このゾーンに関する情報へのすべての要求を他のネームサーバーに転送します。 • hint: ゾーンが不明な場合にクエリーを解決するルートネームサーバーをポイントする特別な種類のゾーン。hint ゾーンでは、デフォルト以外の設定は必要ありません。 • master: このゾーンに対してネームサーバーを権威として指定します。ゾーンの設定ファイルがシステムに存在する場合は、ゾーンを master として設定する必要があります。 • slave: このゾーンに対してネームサーバーをセカンダリーサーバーとして指定します。プライマリーサーバーは masters ディレクティブで指定します。

プライマリーまたはセカンダリーネームサーバーの /etc/named.conf ファイルに対するほとんどの変更には、zone ステートメントの追加、変更、または削除が必要で、通常は zone ステートメントオプションのサブセットのみが、ネームサーバーが効率的に機能させるために必要です。

例15.5「プライマリーネームサーバー用の Zone ステートメント」 では、ゾーンは example.com として識別され、タイプは master に設定され、named サービスは /var/named/example.com.zone ファイルを読み取るように指示します。また、セカンダリーネームサーバー(192.168.0.2)のみがゾーンを転送することを許可します。

例15.5 プライマリーネームサーバー用の Zone ステートメント

```
zone "example.com" IN {
  type master;
  file "example.com.zone";
  allow-transfer { 192.168.0.2; };
};
```

セカンダリーサーバーの zone ステートメントは、若干異なります。type は スレーブ に設定されており、masters ディレクティブはプライマリーサーバー の IP アドレスの名前を指示します。

例15.6「セカンダリーネームサーバー用の Zone ステートメント」 では、named サービスは、IP アドレス 192. 168.0.1 のプライマリーサーバーに example.com ゾーンに関する情報をクエリーするように設定されています。受信した情報は /var/named/slaves/example.com.zone ファイルに保存されます。すべてのセカンダリーゾーンを /var/named/slaves/ ディレクトリーに置く必要があることに注意してください。そうしないと、サービスはゾーンの転送に失敗します。

例15.6 セカンダリーネームサーバー用の Zone ステートメント

```
zone "example.com" {
  type slave;
  file "slaves/example.com.zone";
  masters { 192.168.0.1; };
};
```

15.2.2.3. その他のステートメントタイプ

以下のタイプのステートメントは、一般的に /etc/named.conf で使用されていません。

controls

controls ステートメントにより、named サービスの管理に rndc コマンドを使用するために必要なさまざまなセキュリティー要件を設定できます。

rndc ユーティリティーとその使用方法は、「rndc ユーティリティーの使用」 を参照してください。

key

key ステートメントでは、名前で特定のキーを定義できます。キーは、セキュアな更新や rndc コマンドの使用など、さまざまなアクションを認証するために使用されます。以下の 2 つのオプションが key と合わせて使用されます。

- algorithm *algorithm-name* - 使用するアルゴリズムのタイプ(hmac-md5など)。
- secret "*key-value*": 暗号化キー。

rndc ユーティリティーとその使用方法は、「rndc ユーティリティーの使用」 を参照してください。

logging

logging ステートメントでは、*channels* と呼ばれる複数の種類のログを使用できます。ステートメント内で channel オプションを使用すると、独自のファイル名 (file)、サイズ制限 (size)、バージョン番号 (version)、および重要度 (severity) でログのカスタマイズされたタイプを作成できます。カスタマイズされたチャネルが定義されると、カテゴリー オプションを使用してチャネルを分類し、named サービスの再起動時にロギングを開始します。

デフォルトでは、named は標準メッセージを rsyslog デーモンに送信し、それが /var/log/messages に配置されます。default_syslog （情報ロギングメッセージを処理）や default_debug （特にデバッグメッセージを処理する）など、さまざまな重大度レベルで BIND に組み込まれています。default と呼ばれる デフォルト のカテゴリーは、組み込みチャネルを使用して、特別な設定なしで通常のロギングを行います。

ロギングプロセスのカスタマイズは詳細なプロセスとなるため、本章の範囲外になります。カスタム BIND ログ作成の詳細は「インストールされているドキュメント」で参照されている BIND 9 Administrator Reference Manual (『BIND 9 管理者リファレンスマニュアル』) を参照してください。

server

server ステートメントにより、named サービスがリモートのネームサーバーに応答する方法に影響を与えるオプションを指定できます（特に通知およびゾーン転送に関するもの）。

transfer-format オプションは、各メッセージと共に送信されるリソースレコードの数を制御します。1 つの 応答(1 つ のリソースレコードのみ)または many-answers （複数のリソースレコード）のいずれかになります。many-answers オプションはより効率的ですが、古いバージョンの BIND ではサポートされていないことに注意してください。

ステートメントを使うと、安全な DNS (DNSSEC) に使用される各種パブリックキーを指定できるようになります。

trusted-keys ステートメントでは、DNSSEC （セキュアな DNS ）に使用されるソート済み公開鍵を指定できます。このトピックの詳細については、「DNSSEC (DNS Security Extensions)」を参照してください。

match-clients

view ステートメントでは、ホストがネームサーバーをクエリーするネットワークに応じて、特別なビューを作成できます。これにより、他のホストが全く異なる情報を受け取る間、ゾーンに関する応答が 1 つの応答を受け取ることができます。また、信頼されないホスト以外のホストでは他のゾーンに対するクエリーしか実行できません。

view はその名前が一意になっていれば、複数のものを使用できます。match-clients オプションを使用すると、特定のビューに適用される IP アドレスを指定できます。options ステートメントがビュー内で使用されると、設定済みのグローバルオプションが上書きされます。最後に、ほとんどの view ステートメントには、match-clients リストに適用される複数の zone ステートメントが含まれます。

view ステートメントが特定のクライアントの IP アドレスに一致する最初のステートメントが使用されるため、view ステートメントが記載されている順序が重要である点に注意してください。このトピックの詳細については、「複数表示」を参照してください。

15.2.2.4. コメントタグ

ステートメントの他に、/etc/named.conf ファイルにコメントを含めることもできます。コメントは named サービスでは無視されますが、ユーザーに追加情報を提供する際に役に立ちます。以下に有効なコメントタグを示します。

//

// 文字の後のテキストはすべてコメントとみなされます。以下に例を示します。

```
notify yes;  // notify all secondary nameservers
```

#

文字の後のテキストはすべてコメントとみなされます。以下に例を示します。

```
notify yes;  # notify all secondary nameservers
```

/* and */

/* と */ で囲まれたテキストのブロックはコメントとみなされます。以下に例を示します。

```
notify yes;  /* notify all secondary nameservers */
```

15.2.3. ゾーンファイルの編集

「ネームサーバーゾーン」で要約したように、ゾーンファイルにはネームスペースの情報が含まれています。これらは、デフォルトで /var/ named / にある named 作業ディレクトリーに保存されます。各ゾーンファイルは、zone ステートメントの file オプションに従って名前が付けられます。通常、ドメインに関連する方法で、example.com.zone などのゾーンデータを含むゾーンデータとしてファイルを識別します。

表15.5 named サービスのゾーンファイル

パス	説明
/var/named/	named サービスの作業ディレクトリー。ネームサーバーにはこのディレクトリーに書き込む許可が ありません。
/var/named/slaves/	セカンダリーゾーンのディレクトリーです。このディレクトリーは named サービスによって書き込み可能です。
/var/named/dynamic/	動的 DNS (DDNS)ゾーンや管理された DNSSEC キーなどの他のファイルのディレクトリー。このディレクトリーは named サービスによって書き込み可能です。
/var/named/data/	様々な統計とデバッギングファイル用のディレクトリーです。このディレクトリーは named サービスによって書き込み可能です。

ゾーンファイルはディレクティブとリソースの記録で設定されています。ディレクティブはネームサーバーに対してタスクを実行するか、または特別なセッティングをゾーンに適用するように指示し、リソースレコードはゾーンのパラメーターを定義して識別子を個々のホストに割り当てます。ディレクティブはオプションですが、リソースレコードはゾーンにネームサービスを提供するために必須です。

ディレクティブとリソースレコードはすべて、個別の行で記入します。

15.2.3.1. 一般的なディレクティブ

ディレクティブはドル記号記号($)で始まり、その後にディレクティブ名が続き、通常はファイルの上部に表示されます。通常、ファイルの最上部に現れます。以下のディレクティブは一般的にゾーンファイルで使用されます。

$INCLUDE

$INCLUDE ディレクティブを使用すると、表示される場所に別のファイルを追加して、他のゾーン設定を別のゾーンファイルに保存できるようにします。

> **例15.7 $INCLUDE ディレクティブの使用**
>
> ```
> $INCLUDE /var/named/penguin.example.com
> ```

$ORIGIN

$ORIGIN ディレクティブを使用すると、ホスト名のみを持つレコードなど、ドメイン名を非修飾レコードに追加できます。デフォルトではゾーン名が使用されるため、/etc/named.conf にゾーンが指定されている場合は、このディレクティブの使用は必要ありません。

例15.8「$ORIGIN ディレクティブの使用」 では、リソースレコード内で使用され、末尾がピリオド(. 文字)で終わらない名前には example.com が追加されます。

> **例15.8 $ORIGIN ディレクティブの使用**
>
> ```
> $ORIGIN example.com.
> ```

$TTL

$TTL ディレクティブを使用すると、ゾーンのデフォルトの *Time to Live* (TTL)値（つまり、

ゾーンレコードの有効期間）を設定できます。各リソースレコードはそれ自身の TTL 値を含むことができるため、それがこのディレクティブを上書きします。

この値を増加させるとリモートのネームサーバーはより長い期間でゾーン情報をキャッシュ化できるようになり、ゾーンへのクエリー回数が減少し、リソースレコード変更の伝達に必要な時間を延長させることができます。

例15.9 $TTL ディレクティブの使用

```
$TTL 1D
```

15.2.3.2. 一般的なリソースレコード

以下のリソースレコードは一般的にゾーンファイル内で使用されます。

A

Address レコードは、名前に割り当てられる IP アドレスを指定します。以下の形式を取ります。

```
hostname IN A IP-address
```

hostname の値ない場合、レコードは最後に指定された *hostname* を指します。

例15.10「リソースレコードの使用」 では、server1.example.com の要求は 10.0. 1.3 または 10.0.1. 5 を指し ています。

例15.10 リソースレコードの使用

```
server1  IN  A  10.0.1.3
         IN  A  10.0.1.5
```

CNAME

Canonical Name (別名) レコードはある名前を別の名前にマッピングします。このため、このタイプのレコードは、*エイリアスレコード*と呼ばれることもあります。以下の形式を取ります。

-

> *alias-name* IN CNAME *real-name*

CNAME レコードは、Web サーバーの www など、一般的な命名スキームを使用するサービスを参照するために最も一般的に使用されます。しかし、それらの使用については複数の制限があります。

- CNAME レコードは他の CNAME レコードを指してはいけません。これは主に無限のループの可能性を避けるためです。
- CNAME レコードには他のリソースレコードタイプ (A、NS、MX など) を含めないでください。唯一の例外は、ゾーンが署名されている時の DNSSEC 関連のレコード (RRSIG、NSEC など) です。
- ホストの完全修飾型ドメイン名 (FQDN) を指す他のリソースレコード (NS、MX、PTR) は CNAME レコードを指してはいけません。

例15.11「CNAME リソースレコードの使用」 では、A レコードはホスト名を IP アドレスにバインドし、CNAME レコードは一般的に使用される www ホスト名を指します。

例15.11 CNAME リソースレコードの使用

```
server1  IN  A      10.0.1.5
www      IN  CNAME  server1
```

MX

Mail Exchange レコードは、このゾーンで制御されている特定のネームスペースに送信されるメールの行き先を指定します。以下の形式を取ります。

> IN MX *preference-value email-server-name*

email-server-name は完全修飾型ドメイン名 (FQDN) です。***preference-value*** によってネームスペースのメールサーバーの数値ランキングが可能になり、一部のメールシステムに他のシステムよりも優先度を与えます。最も低い ***preference-value*** を持つ MX リソースレコードが他の値よりも優先されます。しかし複数メールサーバーが同じ値を持つ可能性があり、その場合はメールトラフィックをサーバー間で均等に分配することになります。

例15.12「MX リソースレコードの使用」 では、example.com ドメイン宛のメールを受信す

る場合は、最初の mail.example.com メールサーバーが mail2. example.com メールサーバーよりも優先されます。

例15.12 MX リソースレコードの使用

```
example.com.  IN  MX  10  mail.example.com.
              IN  MX  20  mail2.example.com.
```

NS

Nameserver レコードはある特定のゾーン用に正当なネームサーバーを表明します。以下の形式を取ります。

```
IN NS nameserver-name
```

nameserver-name は完全修飾型ドメイン名 (FQDN) である必要があります。ドメインに対して 2 つのネームサーバーが正当だとしてリスト表示されている時には、これらのネームサーバーがセカンダリーネームサーバーであるか、またはその 1 つがプライマリーサーバーであるかどうかは重要でありません。両方とも正当と考慮されます。

例15.13 NS リソースレコードの使用

```
IN  NS  dns1.example.com.
IN  NS  dns2.example.com.
```

PTR

Pointer レコードはネームスペースの別の部分を指します。以下の形式を取ります。

```
last-IP-digit IN PTR FQDN-of-system
```

last-IP-digit ディレクティブは IP アドレスの最後の番号で、***FQDN-of-system*** は完全修飾ドメイン名(FQDN)です。

PTR レコードは、主に逆引き名前解決に使用されます。IP アドレスは特定の名前を参照するためです。PTR レコードの使用例は、「逆引き名前解決ゾーンファイル」 を参照してください。

SOA

Start of Authority レコードはネームスペースについての信頼できる重要な情報をネームサーバーに表明します。ディレクティブの後に配置されていて、ゾーンファイルでは最初のリソースレコードです。以下の形式を取ります。

```
@ IN SOA primary-name-server hostmaster-email (
    serial-number
    time-to-refresh
    time-to-retry
    time-to-expire
    minimum-TTL )
```

ディレクティブは以下の通りです。

- @ 記号は、**$ORIGIN** ディレクティブ(**$ORIGIN** ディレクティブが設定されていない場合はゾーン名)を、この **SOA** リソースレコードで定義されている名前空間として配置します。
- ***primary-name-server*** ディレクティブは、このドメインの正式なプライマリーネームサーバーのホスト名です。
- ***hostmaster-email*** ディレクティブは、ネームスペースに関して連絡する相手のメールです。
- ***serial-number*** ディレクティブは、**named** サービスがゾーンを再ロードする時間であることを示すためにゾーンファイルが変更されるたびに増加する数値です。
- ***time-to-refresh*** ディレクティブは、ゾーンに対して変更がなされたかどうかをプライマリーネームサーバーに尋ねるまで待機する時間の長さを決定するためにセカンダリーネームサーバーが使用する数値です。
- ***time-to-retry*** ディレクティブは、プライマリーネームサーバーが応答しない事態にリフレッシュ要求を出すまで待機する時間の長さを決定するためにセカンダリーネームサーバーによって使用される数値です。***time-to-expire*** ディレクティブ内で指定された時間が経過するまでに、プライマリーネームサーバーがリフレッシュ要求に応答しない場合は、セカンダリーサーバーはそのネームスペースに関する要求での権威としての応答を停止します。
- **BIND 4** と **8** では、***minimum-TTL*** ディレクティブは他のネームサーバーがゾーンの情報をキャッシュ化する時間の長さになります。**BIND 9** では、これは否定的な回答が

キャッシュ化される時間の長さを定義します。負の回答のキャッシュは、最大 3 時間 (3H) に設定できます。

BIND の設定時には、すべての時間は秒で指定されます。ただし、秒以外の時間単位を指定する場合は、分(M)、時間(H)、日(D)、および週(W)など、省略形を使用することができます。表 15.6「秒表示とその他の時間単位」は、秒単位で、同等の時間 (秒単位) を別の形式で示しています。

表15.6 秒表示とその他の時間単位

秒	他の時間単位
60	1M
1800	30M
3600	1H
10800	3H
21600	6H
43200	12H
86400	1D
259200	3D
604800	1W
31536000	365D

例15.14 SOA リソースレコードの使用

```
@  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. コメントタグ

リソースレコードとディレクティブの他にも、ゾーンファイルもコメントを格納することができます。コメントは named サービスでは無視されますが、ユーザーに追加情報を提供する際に役に立ちます。セミコロンの後の行末までのテキストはすべてコメントとみなされます。以下に例を示します。

```
604800  ; expire after 1 week
```

15.2.3.4. 使用例

下記の例は、ゾーンファイルの基本的使用法を示したものです。

15.2.3.4.1. 単純なゾーンファイル

例15.15「単純なゾーンファイル」 標準のディレクティブと SOA 値の使用を示しています。

例15.15 単純なゾーンファイル

```
$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.
;
;
```

この例では、権威ネームサーバーは dns 1.example.com および dns2.example.com として設定され、A レコードを使用してそれぞれ 10. 0.1.1 および 10.0.1.2 の IP アドレスにそれぞれ関連付けられます。

MX レコードで設定されたメールサーバーは、A レコードを介して mail と mail2 を指しています。これらの名前は末尾のピリオドで終了しないため、$ORIGIN ドメインは後に置かれ、それらを mail.example.com および mail2.example.com に拡張します。

www.example.com (WW)などの標準名で利用可能なサービスは、CNAME レコードを使用して適切なサーバーを参照します。

このゾーンファイルは、以下のように zone ステートメントが /etc/named.conf にて サービス と呼ばれます。

```
zone "example.com" IN {
  type master;
  file "example.com.zone";
  allow-update { none; };
};
```

15.2.3.4.2. 逆引き名前解決ゾーンファイル

逆引き名前解決ゾーンファイルは、特定の namespace の IP アドレスを完全修飾ドメイン名 (FQDN)に変換するために使用されます。これは標準のゾーンファイルと非常に似ていますが、PTR リソースレコードは、例15.16「逆引き名前解決ゾーンファイル」 に示すように、IP アドレスを完全修飾ドメイン名にリンクするために使用されます。

例15.16 逆引き名前解決ゾーンファイル

```
$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.
```

この例では、10. 0.1.1 から 10.0.1 .6 までの IP アドレスは、対応する完全修飾ドメイン名を指しています。

このゾーンファイルは、以下のような /etc/named.conf ファイルの zone ステートメントを使用してサービスを呼び出します。

```
zone "1.0.10.in-addr.arpa" IN {
  type master;
  file "example.com.rr.zone";
  allow-update { none; };
};
```

ゾーン名を除き、この例と標準の zone ステートメントにはほとんど違いがありません。逆引き名前解決ゾーンには、IP アドレスの最初の 3 つのブロックを逆方向にし、その後に .in-addr.arpa が必要であることに注意してください。これにより、逆引き名前解決ゾーンファイルで使用される IP 番号の単一ブロックをゾーンに関連付けることができます。

15.2.4. rndc ユーティリティーの使用

rndc ユーティリティーは、ローカルおよびリモートマシンの両方から named サービスを管理できるコマンドラインツールです。以下のような使用法になります。

```
rndc [option...] command [command-option]
```

15.2.4.1. ユーティリティーの設定

サービスへの不正アクセスを防ぐには、named が選択したポートをリッスンするように設定する必要があります（デフォルトでは 953）。また、サービスと rndc ユーティリティーの両方で同じキーを使用する必要があります。

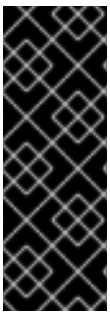
表15.7 関連ファイル

パス	説明
/etc/named.conf	named サービスのデフォルトの設定ファイルです。
/etc/rndc.conf	rndc ユーティリティーのデフォルト設定ファイル。
/etc/rndc.key	デフォルトキーの場所

rndc 設定は /etc/rndc.conf にあります。ファイルが存在しない場合、ユーティリティーは /etc/rndc.key にあるキーを使用します。これは、rndc-confgen -a コマンドを使用してインストールプロセス時に自動的に生成されたものです。

named サービスは、「その他のステートメントタイプ」 で説明されている ように、/etc/named.conf 設定ファイルの control ステートメントを使用して設定されます。このステートメントが存在しない限り、ループバックアドレス(127.0.0.1)からの接続のみが許可され、/etc/rndc.key にあるキーが使用されます。

このトピックの詳細は、man ページと 『BIND 9 Administrator Reference Manual』 にある「関連情報」を参照してください。



重要

非特権ユーザーが制御コマンドをサービスに送信しないようにするには、root のみが /etc/rndc.key ファイルを読み取れるようにします。

```
~]# chmod o-rwx /etc/rndc.key
```

15.2.4.2. サービスステータスの確認

named サービスの現在の状態を確認するには、次のコマンドを使用します。

```
~]# 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. 設定とゾーンのリロード

設定ファイルとゾーンの両方をリロードするには、シェルプロンプトで以下を入力します。

```
~]# rndc reload
server reload successful
```

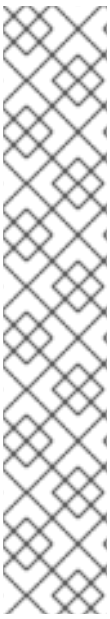
これがゾーンをリロードすると同時に以前にキャッシュ化した応答を維持するため、すべての保存済みの名前解決を消失することなくゾーンファイルを変更できます。

単一のゾーンを再読み込みするには、**reload** コマンドの後にその名前を指定します。以下に例を示します。

```
~]# rndc reload localhost
zone reload up-to-date
```

最後に、設定ファイルと、新規に追加されたゾーンのみをリロードするには、以下を入力します。

```
~]# rndc reconfig
```



注記

動的 **DNS (DDNS)**を使用するゾーンを手動で修正する場合は、**freeze** コマンドを最初に実行してください。

```
~]# rndc freeze localhost
```

完了したら、**thaw** コマンドを実行して **DDNS** を再度許可し、ゾーンをリロードします。

```
~]# rndc thaw localhost
The zone reload and thaw was successful.
```

15.2.4.4. ゾーンキーの更新

DNSSEC キーを更新してゾーンに署名するには、sign コマンドを使用します。以下に例を示します。

```
~]# rndc sign localhost
```

上記のコマンドでゾーンに署名するには、zone ステートメントで auto-dnssec オプションを maintain に設定する必要があります。以下に例を示します。

```
zone "localhost" IN {
  type master;
  file "named.localhost";
  allow-update { none; };
  auto-dnssec maintain;
};
```

15.2.4.5. DNSSEC 検証の有効化

DNSSEC 検証を有効にするには、root で以下のコマンドを実行します。

```
~]# rndc validation on
```

同様に、このオプションを無効にするには、以下を入力します。

```
~]# rndc validation off
```

/etc/named.conf でこの オプション を設定する方法は、「一般的なステートメントのタイプ」 で説明されている options ステートメントを参照してください。

『Red Hat Enterprise Linux 7 セキュリティーガイド』には、DNSSEC に関する詳細なセクションがあります。

15.2.4.6. クエリーロギングの有効化

クエリーロギングを有効にする場合は、root で以下のコマンドを発行します。

```
~]# rndc querylog
```

現在の設定を確認するには、「サービスステータスの確認」 で説明されているように status コマ

ンドを使用します。

15.2.5. dig ユーティリティーの使用

dig ユーティリティーは、DNS ルックアップの実行とネームサーバー設定のデバッグを可能にするコマンドラインツールです。これは通常、以下のように使用されます。

```
dig [@server] [option...] name type
```

type に使用する一般的な値のリストは、「一般的なリソースレコード」を参照してください。

15.2.5.1. ネームサーバーのルックアップ

特定のドメイン用にネームサーバーをルックアップするには、以下の形式でコマンドを使用します。

```
dig name NS
```

例15.17「ネームサーバールックアップのサンプル」 では、**dig** ユーティリティーは **example.com** のネームサーバーを表示するために使用されます。

例15.17 ネームサーバールックアップのサンプル

```
~]$ 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 アドレスのルックアップ

特定のドメインに割り当てられた IP アドレスを検索するには、以下の形式でコマンドを使用します。

```
dig name A
```

例15.18「IP アドレス検索のサンプル」 では、dig ユーティリティーを使用して example.com の IP アドレスを表示します。

例15.18 IP アドレス検索のサンプル

```
~]$ 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. ホスト名の検索

特定の IP アドレスのホスト名を検索するには、以下の形式でコマンドを使用します。

```
dig -x address
```

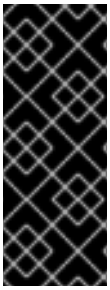
例15.19「ホスト名検索のサンプル」 では、dig ユーティリティーを使用して 192.0.32.10 に割り当てられたホスト名を表示します。

例15.19 ホスト名検索のサンプル

```
~]$ 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. BIND の高度な機能

ほとんどの BIND 実装は、named サービスのみを使用して名前解決サービスを提供したり、特定のドメインの権限として機能することだけです。ただし、BIND バージョン 9 には、より安全で効率的な DNS サービスを可能にする多くの高度な機能があります。



重要

DNSSEC、TSIG、IXFR (増分ゾーン転送) などの高度な機能の使用を試みる前に、特に BIND の古いバージョンや BIND 以外のサーバーを使用している場合は、その特定の機能がネットワーク環境内のすべてのネームサーバーでサポートされていることを確認してください。

ここに記載されているすべての機能は「インストールされているドキュメント」で参照されている

『BIND 9 Administrator Reference Manual』でより詳細に説明されています。

15.2.6.1. 複数表示

オプションとして、リクエストが発信されたネットワークに応じて異なる情報をクライアントに提供することができます。これは主に、ローカルネットワーク外のクライアントからの機密 DNS エントリーを拒否するために使用され、ローカルネットワーク内のクライアントからのクエリーを許可します。

複数のビューを設定するには、view ステートメントを /etc/named.conf 設定ファイルに追加します。match-clients オプションを使用して IP アドレス全体またはネットワーク全体を照合し、特別なオプションとゾーンデータを指定します。

15.2.6.2. IXFR (Incremental Zone Transfers 差分ゾーン転送)

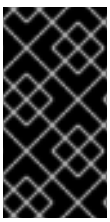
Incremental Zone Transfers 差分ゾーン転送 (*IXFR*) により、セカンダリーネームサーバーはプライマリーネームサーバー上で修正されたゾーンの更新部分だけをダウンロードすることができます。標準の転送プロセスに比較すると、これが通知と更新のプロセスを格段に効率的にします。

IXFR は、動的な更新を使用してプライマリーゾーンレコードに変更を加える時にのみ使用可能なことに注意して下さい。ゾーンファイルを手動の編集で変更する場合は、*Automatic Zone Transfer 自動ゾーン転送* (*AXFR*) が使用されます。

15.2.6.3. Transaction SIGnatures トランザクション署名 (TSIG)

Transaction SIGnatures トランザクション署名 (TSIG) は、転送を許可する前に共有の秘密鍵がプライマリーとセカンダリーの両方のサーバー上に存在することを確認します。これにより、攻撃者はゾーンを転送するため、IP アドレスベースの転送認証方法が強化されます。これは、攻撃者がゾーンを転送するために IP アドレスにアクセスする必要はありませんが、秘密鍵を知る必要もあります。

バージョン 9 以降は、BIND は *TKEY* もサポートします。これはゾーン転送を認証するもう 1 つの共有秘密鍵メソッドです。



重要

セキュアでないネットワーク上で通信する場合には、IP アドレスベースの認証のみに依存しないでください。

15.2.6.4. DNSSEC (DNS Security Extensions)

DNSSEC(*Domain Name System Security Extensions*)は、DNS データの発信元認証、認証された存在拒否、およびデータの整合性を提供します。特定のドメインがセキュアとしてマークされている場合、検証に失敗したリソースレコードごとに SERVFAIL 応答が返されます。

DNSSEC 署名のドメインまたは DNSSEC 対応リゾルバーをデバッグするには、「dig ユーティリティーの使用」 で説明されているように、dig ユーティリティーを使用できます。便利なオプションは +dnssec (DNSSEC OK ビット)、+cd (応答の検証に再帰的ネームサーバー)、および +bufsize=512 (パケットサイズを 512B に変更してファイアウォール経由で取得する。

15.2.6.5. インターネットプロトコルバージョン 6 (IPv6)

表15.3「一般的な設定オプション」で説明されているように、*Internet Protocol version 6* (*IPv6*) は AAAA リソースレコードと listen-on-v6 ディレクティブを使用してサポートされます。

15.2.7. 回避すべき一般的な間違い

ネームサーバー設定時にユーザーが一般的な間違いを回避するためのアドバイスリストを以下に示します。

セミコロンと弓形括弧の正しい使用

/etc/named.conf ファイルのセミコロンまたは一致しない中括弧を使用すると、named サービスが起動しないようにすることができます。

期間(. 文字)を正しく使用

ゾーンファイル内では、ドメイン名の末尾のピリオドは完全修飾型ドメイン名を示します。省略した場合、named サービスはゾーンの名前または $ORIGIN の値を追加して完了します。

ゾーンファイルを編集する時のシリアル番号増加

シリアル番号が増加していない場合、プライマリーネームサーバーは正しくて新しい情報を持ちますが、セカンダリーネームサーバーは決して変更を通知されません。そのため、そのゾーンのデータをリフレッシュする試みをしません。

ファイヤーウォールの設定

ファイアウォールが named サービスから他のネームサーバーへの接続をブロックしている場合、ファイアウォール設定を変更することが推奨されます。

警告

DNS クエリーに固定 UDP ソースポートを使用すると、攻撃者がキャッシュポイズニング攻撃をより簡単に実施できるセキュリティー脆弱性が発生する可能性があります。これを防ぐために、デフォルトでは DNS はランダムな一時ポートから送信します。ランダムな UDP ソースポートからの発信クエリーを許可するようにファイアウォールを設定します。デフォルトでは、1024 から 65535 の範囲が使用されます。

15.2.8. 関連情報

以下の資料は、BIND に関するその他のリソースを提供します。

15.2.8.1. インストールされているドキュメント

BIND は、多種多様なトピックを網羅した広範囲に及ぶインストール済みのドキュメントを特徴としています。各ドキュメントはその議題のディレクトリー内に配置されています。以下の各項目について、*version* の部分をシステム上にインストールしてある bind パッケージのバージョンに入れ替えてください。

/usr/share/doc/bind-*version*/

最新のドキュメンテーションを格納しているメインのディレクトリーです。ここには、HTML と PDF 形式で『BIND 9 Administrator Reference Manual』を収納しており、BIND のリソース要件、異種タイプのネームサーバーの設定方法、ロードバランシングの実行方法、および他の高度なトピックを説明しています。

/usr/share/doc/bind-*version*/sample/etc/

名前付き 設定ファイルのサンプルを含むディレクトリー。

rndc(8)

rndc ネームサーバー制御ユーティリティーの man ページ。その使用方法に関するドキュメントが含まれています。

named(8)

と いう名前 のインターネットドメイン名サーバーの man ページです。これには、BIND ネームサーバーデーモンの制御に使用できる各種の引数が記載されています。

lwresd(8)

軽量リゾルバーデーモン lwresd の man ページには、デーモンとその使用方法が記載されています。

named.conf(5)

の man ページには、named 設定ファイル内で利用可能なオプションの包括的な一覧が記載されています。

rndc.conf(5)

man ページには、rndc 設定ファイル内で利用可能なオプションの包括的なリストが記載されています。

15.2.8.2. オンラインリソース

https://access.redhat.com/site/articles/770133

Red Hat Enterprise Linux 6 と比較しての違いを含む、chroot 環境で BIND を実行するという Red Hat ナレッジベースアーティクルです。

https://access.redhat.com/documentation/ja-JP/Red_Hat_Enterprise_Linux/7/html/Security_Guide/

『Red Hat Enterprise Linux 7 セキュリティーガイド』には、DNSSEC に関する詳細なセクションがあります。

https://www.icann.org/namecollision

『Name Collision Resources and Information』

第16章 SQUID キャッシュプロキシーサーバーの設定

Squid は、コンテンツをキャッシュして帯域幅を削減し、Web ページをより迅速に読み込むプロキシーサーバーです。本章では、HTTP、HTTPS、FTP のプロトコルのプロキシーとして Squid を設定する方法と、アクセスの認証および制限を説明します。

16.1. 認証なしで SQUID をキャッシュプロキシーとして設定

本セクションでは、認証なしでキャッシュプロキシーとして Squid の基本設定を説明します。以下の手順では、IP 範囲に基づいてプロキシーへのアクセスを制限します。

前提条件

- この手順では、/etc/squid/squid.conf ファイルが squid パッケージで提供されることを前提としています。このファイルを編集した場合は、ファイルを削除して、パッケージを再インストールしている。

手順

1. squid パッケージをインストールします。

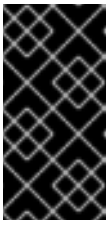
```
# yum install squid
```

2. /etc/squid/squid.conf ファイルを編集します。

   a. localnet アクセス制御リスト(ACL)を、プロキシーを使用できる IP 範囲と一致するように変更します。

```
acl localnet src 192.0.2.0/24
acl localnet 2001:db8::/32
```

デフォルトでは、/etc/squid/squid.conf ファイルには、localnet ACL で指定されたすべての IP 範囲のプロキシーを使用できる http_access allow localnet ルールが含まれます。http_access allow localnet ルールの前に、すべての localnet ACL を指定する必要があります。



重要

お使いの環境に一致しない既存の acl localnet エントリーをすべて削除します。

b.

以下の ACL はデフォルト設定に存在し、HTTPS プロトコルを使用するポートとして 443 を定義します。

```
acl SSL_ports port 443
```

ユーザーが他のポートでも HTTPS プロトコルを使用できるようにするには、ポートごとに ACL を追加します。

```
acl SSL_ports port port_number
```

c.

Squid が接続を確立できるポートに設定する acl Safe_ports ルールの一覧を更新します。たとえば、プロキシーを使用するクライアントがポート 21 (FTP)、80 (HTTP)、443 (HTTPS) のリソースにのみアクセスできるようにするには、その設定の以下の acl Safe_ports ステートメントのみを保持します。

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

デフォルトでは、設定には、Safe_ports ACL で定義されていないポートへのアクセス拒否を定義する http_access deny !Safe_ports ルールが含まれます。

d.

***cache_dir* パラメーターにキャッシュの種類、キャッシュディレクトリーへのパス、キャッシュサイズ、さらにキャッシュの種類固有の設定を設定します。**

```
cache_dir ufs /var/spool/squid 10000 16 256
```

この設定により、以下が可能になります。

- **Squid は、ufs キャッシュタイプを使用します。**

- Squid はキャッシュを /var/spool/squid/ ディレクトリーに保存します。
- キャッシュのサイズが最大 10000 MB になります。
- Squid は、16 個の レベル 1 サブディレクトリーを /var/spool/squid/ ディレクトリーに作成します。
- Squid は、レベル 1 の各ディレクトリーに 256 個のサブディレクトリーを作成します。

cache_dir ディレクティブを設定しない場合、Squid はキャッシュをメモリーに保存します。

3. *cache_dir* パラメーターに /var/spool/squid/ 以外のキャッシュディレクトリーを設定する場合は、以下を行います。

a. キャッシュディレクトリーを作成します。

```
# mkdir -p path_to_cache_directory
```

b. キャッシュディレクトリーの権限を設定します。

```
# chown squid:squid path_to_cache_directory
```

c. SELinux を Enforcing モードで実行する場合は、squid_cache_t コンテキストをキャッシュディレクトリーに設定します。

```
# semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"
# restorecon -Rv path_to_cache_directory
```

semanage ユーティリティーがシステムで利用できない場合は、policycoreutils-python-utils パッケージをインストールします。

4. ファイアウォールで 3128 ポートを開きます。

```
# firewall-cmd --permanent --add-port=3128/tcp
# firewall-cmd --reload
```

5. squid サービスを開始します。

```
# systemctl start squid
```

6. システムの起動時に squid サービスが自動的に起動するようにします。

```
# systemctl enable squid
```

検証手順

プロキシーが適切に機能していることを確認するには、curl ユーティリティーを使用して Web ページをダウンロードします。

```
# curl -O -L "https://www.redhat.com/index.html" -x "proxy.example.com:3128"
```

curl でエラーが表示されず、index.html ファイルが現在のディレクトリーにダウンロードされている場合は、プロキシーが機能します。

16.2. LDAP 認証を使用したキャッシュプロキシーとしての SQUID の設定

本セクションでは、LDAP を使用してユーザーを認証するキャッシングプロキシーとしての Squid の基本設定を説明します。この手順では、認証されたユーザーのみがプロキシーを使用できるように設定します。

前提条件

- この手順では、/etc/squid/squid.conf ファイルが squid パッケージで提供されることを前提としています。このファイルを編集した場合は、ファイルを削除して、パッケージを再インストールしている。

- uid=proxy_user,cn=users,cn=accounts,dc=example,dc=com などのサービスユーザーが LDAP ディレクトリーに存在します。Squid はこのアカウントを使用して認証ユーザーを検索します。認証ユーザーが存在する場合、Squid はこのユーザーをディレクトリーにバインドして、認証を確認します。

手順

1. **squid パッケージをインストールします。**

```
# yum install squid
```

2. **/etc/squid/squid.conf ファイルを編集します。**

a. **basic_ldap_auth ヘルパーユーティリティーを設定するには、/etc/squid/squid.conf の上部に以下の設定エントリーを追加します。**

```
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
```

以下は、上記の例で basic_ldap_auth ヘルパーユーティリティーに渡されるパラメーターを説明します。

- ***-b base_DN* は LDAP 検索ベースを設定します。**
- ***-d proxy_service_user_DN* は、Squid がディレクトリー内の認証ユーザーを検索するために使用するアカウントの識別名(DN)を設定します。**
- ***-w path_to_password_file* は、プロキシーサービスユーザーのパスワードが含まれるファイルへのパスを設定します。パスワードファイルを使用すると、オペレーティングシステムのプロセスリストにパスワードが表示されなくなります。**
- ***-f LDAP_filter* は、LDAP 検索フィルターを指定します。Squid は、%s 変数を、認証ユーザーが提供するユーザー名に置き換えます。**

この例の (& (objectClass=person) (uid=%s)) フィルターは、ユーザー名が *uid* 属性に設定された値と一致する必要があり、ディレクトリーエントリーに person オブジェクトクラスが含まれることを定義します。

- ***-ZZ* は、STARTTLS コマンドを使用して LDAP プロトコルで TLS 暗号化接続を**

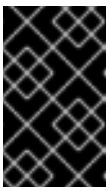
強制します。以下の状況で *-ZZ* を省略します。

- LDAP サーバーは、暗号化された接続にを対応しません。
- URL に指定されたポートは、LDAPS プロトコルを使用します。

- *-H LDAP_URL* パラメーターは、プロトコル、ホスト名、IP アドレス、および LDAP サーバーのポートを URL 形式で指定します。

b. 以下の ACL およびルールを追加して、Squid で、認証されたユーザーのみがプロキシーを使用できるように設定します。

```
acl ldap-auth proxy_auth REQUIRED
http_access allow ldap-auth
```



重要

これらの設定は、http_access deny all ルールの前に指定します。

c. 次のルールを削除して、localnet ACL で指定された IP 範囲のプロキシー認証の回避を無効にします。

```
http_access allow localnet
```

d. 以下の ACL はデフォルト設定に存在し、HTTPS プロトコルを使用するポートとして 443 を定義します。

```
acl SSL_ports port 443
```

ユーザーが他のポートでも HTTPS プロトコルを使用できるようにするには、ポートごとに ACL を追加します。

```
acl SSL_ports port port_number
```

e. Squid が接続を確立できるポートに設定する acl Safe_ports ルールの一覧を更新し

ます。たとえば、プロキシーを使用するクライアントがポート 21 (FTP)、80 (HTTP)、443 (HTTPS) のリソースにのみアクセスできるようにするには、その設定の以下の acl Safe_ports ステートメントのみを保持します。

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

デフォルトでは、設定には、Safe_ports ACL で定義されていないポートへのアクセス拒否を定義する http_access deny !Safe_ports ルールが含まれます。

f. *cache_dir* パラメーターにキャッシュの種類、キャッシュディレクトリーへのパス、キャッシュサイズ、さらにキャッシュの種類固有の設定を設定します。

```
cache_dir ufs /var/spool/squid 10000 16 256
```

この設定により、以下が可能になります。

- Squid は、ufs キャッシュタイプを使用します。
- Squid はキャッシュを /var/spool/squid/ ディレクトリーに保存します。
- キャッシュのサイズが最大 10000 MB になります。
- Squid は、16 個の レベル 1 サブディレクトリーを /var/spool/squid/ ディレクトリーに作成します。
- Squid は、レベル 1 の各ディレクトリーに 256 個のサブディレクトリーを作成します。

cache_dir ディレクティブを設定しない場合、Squid はキャッシュをメモリーに保存します。

3. *cache_dir* パラメーターに /var/spool/squid/ 以外のキャッシュディレクトリーを設定する場合は、以下を行います。

a.
キャッシュディレクトリーを作成します。

```
# mkdir -p path_to_cache_directory
```

b.
キャッシュディレクトリーの権限を設定します。

```
# chown squid:squid path_to_cache_directory
```

c.
SELinux を Enforcing モードで実行する場合は、squid_cache_t コンテキストをキャッシュディレクトリーに設定します。

```
# semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"
# restorecon -Rv path_to_cache_directory
```

semanage ユーティリティーがシステムで利用できない場合は、policycoreutils-python-utils パッケージをインストールします。

4.
LDAP サービスユーザーのパスワードを /etc/squid/ldap_password ファイルに保存し、ファイルに適切なパーミッションを設定します。

```
# echo "password" > /etc/squid/ldap_password
# chown root:squid /etc/squid/ldap_password
# chmod 640 /etc/squid/ldap_password
```

5.
ファイアウォールで 3128 ポートを開きます。

```
# firewall-cmd --permanent --add-port=3128/tcp
# firewall-cmd --reload
```

6.
squid サービスを開始します。

```
# systemctl start squid
```

7.
システムの起動時に squid サービスが自動的に起動するようにします。

```
# systemctl enable squid
```

検証手順

プロキシーが適切に機能していることを確認するには、**curl** ユーティリティーを使用して **Web** ページをダウンロードします。

```
# curl -O -L "https://www.redhat.com/index.html" -x
"user_name:password@proxy.example.com:3128"
```

curl でエラーが表示されず、**index.html** ファイルが現在のディレクトリーにダウンロードされている場合は、プロキシーが機能します。

トラブルシューティングの手順

ヘルパーユーティリティーが正しく機能していることを確認するには、以下の手順を行います。

1. ***auth_param*** パラメーターで使用したのと同じ設定でヘルパーユーティリティーを手動で起動します。

```
# /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. 有効なユーザー名とパスワードを入力し、**Enter** を押します。

```
user_name password
```

ヘルパーユーティリティーが **OK** を返すと、認証に成功しました。

16.3. KERBEROS 認証を使用したキャッシュプロキシーとしての SQUID の設定

本セクションでは、**Kerberos** を使用して **Active Directory (AD)** にユーザーを認証するキャッシングプロキシーとしての **Squid** 基本設定を説明します。この手順では、認証されたユーザーのみがプロキシーを使用できるように設定します。

前提条件

- この手順では、**/etc/squid/squid.conf** ファイルが **squid** パッケージで提供されることを前提としています。このファイルを編集した場合は、ファイルを削除して、パッケージを再インストールしている。

- **Squid をインストールするサーバーが、AD ドメインのメンバーである。詳細は、『Red Hat Enterprise Linux 7 システム管理者のガイド』の『Samba をドメインメンバーとしてセットアップ』を参照してください。**

手順

1. **以下のパッケージをインストールします。**

```
# yum install squid krb5-workstation
```

2. **AD ドメイン管理者として認証します。**

```
# kinit administrator@AD.EXAMPLE.COM
```

3. **Squid 用の keytab を作成し、/etc/squid/HTTP.keytab ファイルに保存します。**

```
# export KRB5_KTNAME=FILE:/etc/squid/HTTP.keytab
# net ads keytab CREATE -U administrator
```

4. **HTTP サービスプリンシパルをキータブに追加します。**

```
# net ads keytab ADD HTTP -U administrator
```

5. **keytab ファイルの所有者を squid ユーザーに設定します。**

```
# chown squid /etc/squid/HTTP.keytab
```

6. **必要に応じて、キータブファイルに、プロキシーサーバーの完全修飾ドメイン名(FQDN)の HTTP サービスプリンシパルが含まれていることを確認します。**

```
#  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 ファイルを編集します。**

a. **negotiate_kerberos_auth ヘルパーユーティリティーを設定するには、/etc/squid/squid.conf の上部に以下の設定エントリーを追加します。**

```
auth_param negotiate program /usr/lib64/squid/negotiate_kerberos_auth -k
/etc/squid/HTTP.keytab -s HTTP/proxy.ad.example.com@AD.EXAMPLE.COM
```

以下は、上記の例で negotiate_kerberos_auth ヘルパーユーティリティーに渡されるパラメーターを説明します。

- ***-k file* は、キータブファイルへのパスを設定します。squid ユーザーには、このファイルの読み取り権限が必要なことに注意してください。**
- ***-s HTTP/host_name@kerberos_realm* は、Squid が使用する Kerberos プリンシパルを設定します。**

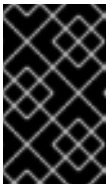
必要に応じて、以下のパラメーターのいずれかまたは両方をヘルパーユーティリティーに渡すことによりロギングを有効にできます。

- ***-i* は、認証ユーザーなどの情報メッセージをログに記録します。**
- ***-d* は、デバッグロギングを有効にします。**

Squid は、ヘルパーユーティリティーから /var/log/squid/cache.log ファイルにデバッグ情報をログに記録します。

b. **以下の ACL およびルールを追加して、Squid で、認証されたユーザーのみがプロキシーを使用できるように設定します。**

```
acl kerb-auth proxy_auth REQUIRED
http_access allow kerb-auth
```



重要

これらの設定は、http_access deny all ルールの前に指定します。

c.

次のルールを削除して、localnet ACL で指定された IP 範囲のプロキシー認証の回避を無効にします。

```
http_access allow localnet
```

d.

以下の ACL はデフォルト設定に存在し、HTTPS プロトコルを使用するポートとして 443 を定義します。

```
acl SSL_ports port 443
```

ユーザーが他のポートでも HTTPS プロトコルを使用できるようにするには、ポートごとに ACL を追加します。

```
acl SSL_ports port port_number
```

e.

Squid が接続を確立できるポートに設定する acl Safe_ports ルールの一覧を更新します。たとえば、プロキシーを使用するクライアントがポート 21 (FTP)、80 (HTTP)、443 (HTTPS) のリソースにのみアクセスできるようにするには、その設定の以下の acl Safe_ports ステートメントのみを保持します。

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

デフォルトでは、設定には、Safe_ports ACL で定義されていないポートへのアクセス拒否を定義する http_access deny !Safe_ports ルールが含まれます。

f.

***cache_dir* パラメーターにキャッシュの種類、キャッシュディレクトリーへのパス、キャッシュサイズ、さらにキャッシュの種類固有の設定を設定します。**

```
cache_dir ufs /var/spool/squid 10000 16 256
```

この設定により、以下が可能になります。

- Squid は、ufs キャッシュタイプを使用します。
- Squid はキャッシュを /var/spool/squid/ ディレクトリーに保存します。
- キャッシュのサイズが最大 10000 MB になります。
- Squid は、16 個の レベル 1 サブディレクトリーを /var/spool/squid/ ディレクトリーに作成します。
- Squid は、レベル 1 の各ディレクトリーに 256 個のサブディレクトリーを作成します。

cache_dir ディレクティブを設定しない場合、Squid はキャッシュをメモリーに保存します。

8. *cache_dir* パラメーターに /var/spool/squid/ 以外のキャッシュディレクトリーを設定する場合は、以下を行います。

a. キャッシュディレクトリーを作成します。

```
# mkdir -p path_to_cache_directory
```

b. キャッシュディレクトリーの権限を設定します。

```
# chown squid:squid path_to_cache_directory
```

c. SELinux を Enforcing モードで実行する場合は、squid_cache_t コンテキストをキャッシュディレクトリーに設定します。

```
# semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"
# restorecon -Rv path_to_cache_directory
```

semanage ユーティリティーがシステムで利用できない場合は、policycoreutils-python-utils パッケージをインストールします。

9. **ファイアウォールで 3128 ポートを開きます。**

```
# firewall-cmd --permanent --add-port=3128/tcp
# firewall-cmd --reload
```

10. **squid サービスを開始します。**

```
# systemctl start squid
```

11. **システムの起動時に squid サービスが自動的に起動するようにします。**

```
# systemctl enable squid
```

検証手順

プロキシーが適切に機能していることを確認するには、curl ユーティリティーを使用して Web ページをダウンロードします。

```
# curl -O -L "https://www.redhat.com/index.html" --proxy-negotiate -u : -x
"proxy.ad.example.com:3128"
```

curl でエラーが表示されず、index.html ファイルが現在のディレクトリーに存在する場合、プロキシーは機能します。

トラブルシューティングの手順

Kerberos 認証を手動でテストするには、以下を行います。

1. **AD アカウントの Kerberos チケットを取得します。**

```
# kinit user@AD.EXAMPLE.COM
```

2. **必要に応じて、キーを表示します。**

```
# klist
```

3. **negotiate_kerberos_auth_test ユーティリティーを使用して認証をテストします。**

```
# /usr/lib64/squid/negotiate_kerberos_auth_test proxy.ad.example.com
```

ヘルパーユーティリティーがトークンを返すと、認証に成功しました。

```
Token: YIIFtAYGKwYBBQUCoIIFqDC...
```

16.4. SQUID でのドメインブラックリストの設定

多くの場合、管理者は特定のドメインへのアクセスをブロックする必要があります。本セクションでは、Squid でドメインブラックリストを設定する方法を説明します。

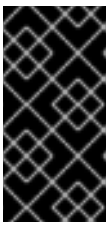
前提条件

- **Squid が設定され、ユーザーはプロキシーを使用できます。**

手順

1. **/etc/squid/squid.conf ファイルを編集し、以下の設定を追加します。**

```
acl domain_blacklist dstdomain "/etc/squid/domain_blacklist.txt"
http_access deny all domain_blacklist
```

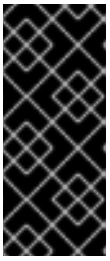


重要

ユーザーまたはクライアントへのアクセスを許可する最初の http_access allow ステートメントの前に、これらのエントリーを追加します。

2. **/etc/squid/domain_blacklist.txt ファイルを作成し、ブロックするドメインを追加します。たとえば、サブドメインを含む example.com へのアクセスをブロックし、example.net をブロックするには、以下を追加します。**

```
.example.com
example.net
```



重要

squid 設定の /etc/squid/domain_blacklist.txt ファイルを参照している場合は、このファイルを空にすることはできません。このファイルが空の場合、Squid は起動できません。

3. **squid サービスを再起動します。**

```
# systemctl restart squid
```

16.5. 特定のポートまたは IP アドレスでリッスンするように SQUID サービスの設定

デフォルトでは、Squid プロキシーサービスは、すべてのネットワークインターフェイスの 3128 ポートをリッスンします。本セクションでは、ポートを変更し、Squid が特定の IP アドレスをリッスンするように設定する方法を説明します。

前提条件

- **Squid がインストールされています。**

手順

1. **/etc/squid/squid.conf ファイルを編集します。**

   - **Squid サービスがリッスンするポートを設定するには、*http_port* パラメーターにポート番号を設定します。たとえば、ポートを 8080 に設定するには、以下を設定します。**

     ```
     http_port 8080
     ```

   - **Squid サービスがリッスンする IP アドレスを設定するには、*http_port* パラメーターに IP アドレスとポート番号を設定します。たとえば、Squid がポート 3128 の IP アドレス 192.0.2.1 でのみリッスンするように設定するには、以下を設定します。**

     ```
     http_port 192.0.2.1:3128
     ```

   複数の *http_port* パラメーターを設定ファイルに追加して、Squid が複数のポートおよび IP アドレスでリッスンするように設定します。

   -

```
http_port 192.0.2.1:3128
http_port 192.0.2.1:8080
```

2. **Squid が別のポートをデフォルトとして使用するように設定した場合(3128):**

   a. **ファイアウォールのポートを開きます。**

   ```
   # firewall-cmd --permanent --add-port=port_number/tcp
   # firewall-cmd --reload
   ```

   b. **Enforcing モードで SELinux を実行する場合は、ポートを squid_port_t ポートタイプ定義に割り当てます。**

   ```
   # semanage port -a -t squid_port_t -p tcp port_number
   ```

   semanage ユーティリティーがシステムで利用できない場合は、policycoreutils-python-utils パッケージをインストールします。

3. **squid サービスを再起動します。**

```
# systemctl restart squid
```

16.6. 関連情報

- **/etc /squid/squid.conf ファイルで設定できる設定パラメーターの一覧と詳細な説明は、/usr/share/doc/squid- <*version*> /squid.conf.documented ファイルを参照してください。**

付録A ネットワーク設定に関する RED HAT CUSTOMER PORTAL LABS

Red Hat Customer Portal Labs のツールは、パフォーマンスの向上、問題のトラブルシューティング、セキュリティー問題の特定、設定の最適化に役立てていただくために開発しました。この付録では、ネットワーク設定に関連する Red Hat Customer Portal Labs の概要を説明します。すべての Red Hat Customer Portal Labs は https://access.redhat.com/labs/ からご利用いただくことができます。

BRIDGE CONFIGURATION

Bridge Configuration は、Red Hat Enterprise Linux 5.4 以降を使用する KVM 等のアプリケーションに、ブリッジ接続のネットワークインターフェイスを設定するために作成されました。

NETWORK BONDING HELPER

管理者は、Network Bonding Helper を使用して、ボンディングカーネルモジュールおよびボンディングネットワークインターフェイスを使用して、複数のネットワークインターフェイスコントローラーを 1 つのチャンネルにまとめることができます。

Network Bonding Helper を使用すると、2 つ以上のネットワークインターフェイスが 1 つのボンディングインターフェイスとして機能できるようになります。

PACKET CAPTURE SYNTAX GENERATOR

Packet capture syntax generator は、ネットワークパケットを取得するのに役立ちます。

Packet capture syntax generator を使用して、インターフェイスを選択して情報をコンソールに出力する tcpdump コマンドを生成します。コマンドを入力するには、root アクセスが必要です。

付録B 改訂履歴

改訂 0.10-06 Tue 03 Mar 2020 Marc Muehlfeld
『代替ルートを定義するためのポリシーベースのルーティングの設定』 セクションを追加しました。

改訂 0.10-05 Fri 22 Nov 2019 Marc Muehlfeld
『Squid キャッシュプロキシーサーバーの設定』の章を書き直しました。

改訂 0.10-04 Tue 06 Aug 2019 Marc Muehlfeld
7.7 GA 公開用バージョン

改訂 0.10-03 Thu 22 Mar 2018 Ioanna Gkioka
7.5 GA 公開用バージョン

改訂 0.10-02 Mon 14 Aug 2017 Ioanna Gkioka
さまざまな更新が追加された非同期リリース

改訂 0.10-01 Tue 25 Jul 2017 Mirek Jahoda
7.4 GA 公開用バージョン

改訂 0.9-30 Tue 18 Oct 2016 Mirek Jahoda
7.3 GA リリースのバージョン

改訂 0.9-25 Wed 11 Nov 2015 Jana Heves
7.2 GA リリース向けのバージョン

改訂 0.9-15 Tue 17 Feb 2015 Christian Huffman
7.1 GA 公開用バージョン

改訂 0.9-14 Fri Dec 05 2014 Christian Huffman
nmtui および NetworkManager GUI のセクションを更新

改訂 0.9-12 Wed Nov 05 2014 Stephen Wadeley
『IP ネットワーク』、『802.1Q VLAN tagging』、および『チーミング』を更新。

改訂 0.9-11 Tues Oct 21 2014 Stephen Wadeley
『ボンディング』、『ブリッジング』、および『チーミング』を更新。

改訂 0.9-9 Tue Sep 2 2014 Stephen Wadeley
『ボンディング』および『ネットワークデバイスの命名における一貫性』を更新。

改訂 0.9-8 Tue July 8 2014 Stephen Wadeley
Red Hat Enterprise Linux 7.0 GA 公開用のネットワークガイド

改訂 0-0 Wed Dec 12 2012 Stephen Wadeley
Red Hat Enterprise Linux 7 ネットワークガイドの作成

B.1. 承認

索引