



Red Hat Enterprise Linux 7

ネットワークガイド

Red Hat Enterprise Linux 7 におけるネットワークの設定と管理

Red Hat Enterprise Linux 7 ネットワークガイド

Red Hat Enterprise Linux 7 におけるネットワークの設定と管理

Mirek Jahoda

Red Hat Customer Content Services

mjahoda@redhat.com

Ioanna Gkioka

Red Hat Customer Content Services

igkioka@redhat.com

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

Copyright © 2010–2017 Red Hat, Inc.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

Red Hat Enterprise Linux 7 ネットワークガイドでは、Red Hat Enterprise Linux 7 におけるネットワークのインターフェースやネットワーク、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ています。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向けになっています。本ガイドは、Red Hat Enterprise Linux 6 導入ガイドをベースとしています。導入ガイドのネットワークに関する各章を基に、本ガイドは執筆されています。

目次

パート I. IP ネットワーク	5
第1章 RED HAT ENTERPRISE LINUX のネットワークについて	6
1.1. 本ガイドの構成	6
1.2. IP ネットワークと非 IP ネットワーク	6
1.3. NETWORKMANAGER について	6
1.4. NETWORKMANAGER のインストール	7
1.5. テキスト形式のユーザーインターフェース (NMTUI) を使ったネットワーク設定	8
1.6. NETWORKMANAGER の CLI (NMCLI) を使用したネットワーク設定	9
1.7. コマンドラインインターフェース (CLI) を使ったネットワーク設定	9
1.8. NETWORKMANAGER とネットワークスクリプト	10
1.9. SYSCONFIG ファイルを使ったネットワーク設定	11
1.10. ワイヤレス規制ドメインの設定	13
1.11. NETCONSOLE の設定	13
1.12. その他のリソース	15
第2章 IP ネットワークの設定	16
2.1. 静的および動的インターフェースの設定	16
2.2. ネットワーク設定ファイルの編集	33
2.3. GNOME グラフィカルユーザーインターフェースによる NETWORKMANAGER の使用	40
2.4. VPN 接続の確立	51
2.5. モバイルブロードバンド接続の確立	57
2.6. DSL 接続の確立	59
2.7. 接続セッティングの設定	60
2.8. その他のリソース	73
第3章 ホスト名の設定	74
3.1. ホスト名について	74
3.2. テキスト形式のユーザーインターフェース NMTUI を使ったホスト名の設定	74
3.3. HOSTNAMECTL を使ったホスト名の設定	75
3.4. NMCLI を使ったホスト名の設定	76
3.5. その他のリソース	77
第4章 ネットワークボンディングの設定	78
4.1. マスターおよびスレーブインターフェースのデフォルト動作について	78
4.2. テキスト形式のユーザーインターフェース NMTUI を使ったボンディングの設定	78
4.3. NETWORKMANAGER のコマンドラインツール NMCLI を使ったネットワークボンディング	83
4.4. コマンドラインインターフェース (CLI) の使用	85
4.5. チャンネルボンディングの使用	88
4.6. GUI を使用したボンディング接続の作成	95
4.7. その他のリソース	100
第5章 ネットワークチーミングの設定	102
5.1. ネットワークチーミングについて	102
5.2. マスターおよびスレーブインターフェースのデフォルト動作について	102
5.3. ネットワークチーミングとボンディングの比較	103
5.4. 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	105
5.5. ネットワークチーミングデーモンのインストール	105
5.6. ボンドのチーム変換	105
5.7. 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ースの選択	107
5.8. ネットワークチーム設定方式の選択	107
5.9. テキスト形式のユーザーインターフェース NMTUI でネットワークチームを設定する手順	107
5.10. コマンドラインを使用したネットワークチームの設定	112

5.11. TEAMDCTL を使用した TEAMD の制御	120
5.12. TEAMD ランナーの設定	121
5.13. GUI を使ったネットワークチームの作成	129
5.14. その他のリソース	132
第6章 ネットワークブリッジングの設定	134
6.1. テキスト形式のユーザーインターフェース NMTUI によるブリッジングの設定	134
6.2. NETWORKMANAGER のコマンドラインツール NMCLI の使用	138
6.3. コマンドラインインターフェース (CLI) の使用	140
6.4. GUI を使ったネットワークブリッジングの設定	144
6.5. その他のリソース	149
第7章 802.1Q VLAN タグの設定	150
7.1. VLAN インターフェース設定方式の選択	151
7.2. テキスト形式のユーザーインターフェース NMTUI を使った 802.1Q VLAN タグの設定	151
7.3. コマンドラインツール NMCLI を使った 802.1Q VLAN タグの設定	153
7.4. コマンドラインを使った 802.1Q VLAN タグの設定	156
7.5. GUI を使った 802.1Q VLAN タグの設定	157
7.6. 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	159
7.7. その他のリソース	160
第8章 ネットワークデバイス命名における一貫性	161
8.1. 命名スキームの序列	161
8.2. デバイスの名前変更ステップについて	162
8.3. 予想可能なネットワークインターフェースデバイスの命名について	162
8.4. 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	163
8.5. VLAN インターフェースの命名スキーム	163
8.6. BIOSDEVNAME を使った一貫性のあるネットワークデバイスの命名	164
8.7. 管理者向け注意点	165
8.8. ネットワークデバイス名の選択に関する制御	165
8.9. ネットワークデバイス命名における一貫性の無効化	166
8.10. 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	167
8.11. その他のリソース	168
パート II. INFINIBAND および RDMA ネットワーク	170
第9章 INFINIBAND および RDMA ネットワークの設定	171
9.1. INFINIBAND および RDMA のテクノロジーについて	171
9.2. INFINIBAND および RDMA に関連するソフトウェアパッケージ	172
9.3. BASE RDMA サブシステムの設定	173
9.4. サブネットマネジャーの設定	175
9.5. 初期の INFINIBAND RDMA 操作のテスト	177
9.6. IPOIB の設定	180
9.7. テキスト形式のユーザーインターフェース NMTUI による INFINIBAND の設定	182
9.8. コマンドラインツール NMCLI での IPOIB の設定	184
9.9. コマンドラインを使用した IPOIB の設定	186
9.10. IPOIB 設定後の RDMA ネットワークテスト	187
9.11. GUI を使った IPOIB の設定	188
9.12. その他のリソース	189
パート III. サーバー	191
第10章 DHCP サーバー	192
10.1. DHCP を使用する理由	192
10.2. DHCP サーバーの設定	192

10.3. DHCP リレーエージェント	199
10.4. マルチホーム DHCP サーバーの設定	200
10.5. IPV6 の DHCP (DHCPV6)	203
10.6. IPV6 ルーター用 RADVD デーモンの設定	204
10.7. その他のリソース	205
第11章 DNS サーバー	206
11.1. DNS の概要	206
11.2. BIND	207
第12章 SQUID	235
12.1. SQUID の概要	235
12.2. SQUID のインストールおよび実行	235
12.3. SQUID の設定	236
12.4. SQUID の認証	241
12.5. アクセス制限のための SQUID の使用	245
12.6. その他のリソース	247
付録A ネットワーク設定に関する RED HAT CUSTOMER PORTAL LABS	248
ブリッジの構成	248
NETWORK BONDING HELPER	248
PACKET CAPTURE SYNTAX GENERATOR	248
付録B 改訂履歴	249
索引	250

パート I. IP ネットワーク

このパートでは、Red Hat Enterprise Linux でのネットワーク設定方法について説明します。

第1章 RED HAT ENTERPRISE LINUX のネットワークについて

1.1. 本ガイドの構成

本ガイドにおける新たな内容は、概念やユースケースの説明などの導入用資料を設定タスクとは明確に区別して説明、配置されています。Red Hat Engineering Content Services では、ユーザーが必要な設定手順を迅速に見つけられる一方、ユーザーが必要とする適切なタスクを理解、判断できるように関連する説明および概念的資料を提供することを願っています。『Red Hat Enterprise Linux 6 導入ガイド』からの資料が再利用されている部分は見直しおよび変更がされており、可能な限り概念とタスクの分離が図られています。

各資料は、方法ではなく目的に沿ってグループ化されています。異なる方法を使って特定のタスクを完了する手順は、同じグループになっています。これは、特定のタスクや目的を完了する情報を見つけやすくする一方で、異なる方法も簡単に参照できるようにするためです。

各章における設定方法は、以下の順序で示されています。

- テキスト形式のユーザーインターフェースツールである **nmtui** を使用する方法
- **NetworkManager** のコマンドラインツールである **nmcli** を使用する方法
- 他のコマンドラインでの方法および設定ファイルを使用する方法
- グラフィカルユーザーインターフェース (GUI) を使用する方法。例: **nm-connection-editor** または **control-network** を使って **NetworkManager** に指示を出す。

コマンドラインはコマンドを発行できることから コマンドラインインターフェース (CLI) と呼ばれますが、コマンドラインはエディターの起動や、設定ファイルの作成および編集にも使用できます。そのため、**ip** コマンドの使用や **ifcfg** ファイルなどの設定ファイルの使用も合わせて説明されています。

1.2. IP ネットワークと非 IP ネットワーク

今日のネットワークのほとんどは、2つのカテゴリーに大別されます。そのひとつは IP ベースのネットワークです。インターネットプロトコルアドレスはインターネットおよび今日のほとんどの内部ネットワークの標準となっており、IP ベースのネットワークはすべて、インターネットプロトコルアドレス経由で通信しています。このカテゴリーに含まれるものには、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、ワイヤレスネットワーク、VPN 接続などがあります。

もうひとつは、非 IP ベースのネットワークです。これらは通常、非常に特殊なネットワークですが、ある 1 種類のネットワークが記述に値する使用量の増加を示しています。InfiniBand (インフィニバンド) がこれに当たります。InfiniBand は IP ネットワークではなく、IP ネットワークで通常使用される多くの機能や設定が InfiniBand では適用されません。本ガイドの「[9章 InfiniBand および RDMA ネットワークの設定](#)」では、InfiniBand の設定や管理における特定の要件と RDMA に対応しているデバイスのグループについて説明しています。



重要

ethX 命名規則を使用しようとした場合、Red Hat Enterprise Linux 7 では一貫性のある名前は提供されません。詳細は、「[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](#)」を参照してください。

1.3. NETWORKMANAGER について

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。これは動的な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間は、これらをアクティブな状態に保ちます。従来の **ifcfg** タイプの設定ファイルも、サポートが継続されます。詳細は、「[NetworkManager とネットワークスクリプト](#)」を参照してください。

表1.1 ネットワークツールおよび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	詳細
NetworkManager	デフォルトのネットワークデーモン
nmtui	NetworkManager 用のシンプルな curses ベースのテキストユーザーインターフェース (TUI)
nmcli	ユーザーとスクリプトが NetworkManager と対話できるようにするコマンドラインツール
control-center	GNOME Shell が提供するグラフィカルユーザーインターフェースツール
nm-connection-editor	control-center では処理されない特定のタスクに利用可能な GTK+ 3 アプリケーション

NetworkManager は、ネットワークエイリアスや **IP** アドレス、静的ルート、**DNS** 情報、VPN 接続に加えて接続固有のさまざまなパラメーターを設定できます。また、**NetworkManager** は D-Bus 経由の API を提供し、これによりアプリケーションはネットワーク設定や状態をクエリ、制御することができます。

さらに、**NetworkManager** は再起動プロセス後にデバイスの状態を維持し、再起動時に managed モードに設定されたインターフェースを引き継ぐようになりました。また、**NetworkManager** は、「unmanaged」と明示的に設定されていないが、ユーザーまたは別のネットワークサービスにより手動で制御されているデバイスを処理できるようになりました。

1.4. NETWORKMANAGER のインストール

NetworkManager は、デフォルトで Red Hat Enterprise Linux にインストールされています。確実にインストールする必要がある場合は、**root** ユーザーで以下のコマンドを入力します。

```
~]# 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 7 システム管理者のガイド](#)』を参照してください。

1.4.1. NetworkManager デーモン

NetworkManager デーモンは root 権限で実行し、デフォルトでブート時に起動するように設定されています。以下のコマンドを入力すると、**NetworkManager** デーモンが実行しているか確認できます。

```
~]$ 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 サービスが稼働していない場合は、**systemctl status** コマンドは **NetworkManager** を **Active: inactive (dead)** と報告します。現行セッションで起動するには、root ユーザーで以下のコマンドを入力します。

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

システム起動時に **NetworkManager** が毎回起動するにようするには、**systemctl enable** 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 7 システム管理者のガイド](#)』を参照してください。

1.4.2. NetworkManager との対話

ユーザーは、**NetworkManager** システムサービスと直接対話はしません。代わりに、グラフィカルおよびコマンドラインのユーザーインターフェースツールを使ってネットワーク設定タスクを実行します。Red Hat Enterprise Linux 7 では以下のツールが利用できます。

1. **NetworkManager** 用のシンプルな curses ベースのテキストユーザーインターフェース (TUI) である **nmtui** が利用可能です。
2. コマンドラインツールの **nmcli** を使うと、ユーザーとスクリプトは **NetworkManager** と対話できるようになります。サーバーのような GUI のないシステム上で **nmcli** を使用し、**NetworkManager** のすべての側面を制御できることに留意してください。つまり、GUI ツールと同等の機能を持ちます。
3. GNOME Shell は **NetworkManager** が報告するネットワーク接続状態のネットワークアイコンも通知スペースに表示します。この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示します。
4. デスクトップユーザーは、GNOME Shell が提供する **control-center** と呼ばれるグラフィカルユーザーインターフェースツールを利用できます。これに、ネットワーク設定ツールが含まれています。起動するには、**Super** キーを押してアクティビティ画面を開き、**Network** と入力して **ネットワーク 設定ツール** を選択します。
5. グラフィカルユーザーインターフェースツールの **nm-connection-editor** は、**control-center** では処理されない特定のタスクに利用できます。これを起動するには、ターミナルで **nm-connection-editor** と入力します。

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

1.5. テキスト形式のユーザーインターフェース (NMTUI) を使ったネットワーク設定

NetworkManager のテキスト形式のユーザーインターフェース (TUI) ツールである **nmtui** は、テキストインターフェースを提供し、**NetworkManager** を制御してネットワークを設定します。このツールは、**NetworkManager-tui** パッケージに含まれています。本ガイドの執筆時点では、デフォルトでは **NetworkManager** と共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、**root** で以下のコマンドを実行します。

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

NetworkManager が稼働しているかを確認する必要がある場合は、「[NetworkManager デーモン](#)」を参照してください。

nmtui を起動するには、以下のコマンドを実行します。

```
~]$ nmtui
```

テキスト形式のインターフェースが表示されます。移動するには矢印キーを使用するか、**Tab** を押して次に進むか **Shift+Tab** を押して前に戻ります。**Enter** を押してオプションを選びます。**Space** は、チェックボックスのステータスを切り替えます。

以下のコマンドが利用できます。

- **nmtui edit connection-name**

接続名を指定しないと、選択メニューが表示されます。接続名を指定して適切に認識されると、関連する **接続の編集** 画面が表示されます。

- **nmtui connect connection-name**

接続名を指定しないと、選択メニューが表示されます。接続名を指定して適切に認識されると、関連する接続がアクティブ化されます。無効なコマンドの場合は、使用法に関するメッセージがプリントされます。

本ガイドの執筆時点では、**nmtui** はすべての接続の種類をサポートしていません。特に、VPN や 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を編集することができません。

1.6. NETWORKMANAGER の CLI (NMCLI) を使用したネットワーク設定

NetworkManager のコマンドラインツールである **nmcli** は、**NetworkManager** を制御してネットワークを設定する方法を提供します。これは、**NetworkManager** とともにデフォルトでインストールされます。**NetworkManager** が稼働しているかを確認する必要がある場合は、「[NetworkManager デーモン](#)」を参照してください。

各タスクで **nmcli** ツールを使用する例については、可能な限り他のコマンドラインの使用法やグラフィカルユーザーインターフェースの説明の前に示されています。**nmcli** の導入については、「[NetworkManager コマンドラインツール \(nmcli\) の使用](#)」を参照してください。役立つ例については、**man nmcli-examples(7)** ページを確認してください。**nmcli c add** と **nmcli c modify** コマンドで利用可能なプロパティを確認するには **man nm-settings(5)** ページを使用します。

1.7. コマンドラインインターフェース (CLI) を使ったネットワーク設定

ip ユーティリティのコマンドはそのアップストリームのパッケージ名から **iproute2** と呼ばれることもあり、**man ip(8)** ページで説明されています。Red Hat Enterprise Linux 7 におけるパッケージ名は、**iproute** となります。必要な場合は、**ip** ユーティリティのバージョン番号を以下のようにチェックすることで、インストールされているかどうかを確認できます。

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

ip コマンドを使うと、**NetworkManager** と並行して、インターフェースにアドレスやルートを追加したり削除したりすることができます。**NetworkManager** はアドレスやルートを保持し、**nmcli** や **nmtui**、**control-center**、D-Bus API でこれらを認識します。

ip ユーティリティーが **ifconfig** ユーティリティーに代わるものであることに留意してください。これは、**ifconfig** を提供する **net-tools** パッケージが InfiniBand アドレスをサポートしないためです。コマンド **ip help** は、使用法に関するメッセージをプリントします。OBJECTS に関する特定のヘルプがあり、**ip link help** や **ip addr help** のように使います。



注記

コマンドライン上で発行された **ip** コマンドは、システム再起動後は維持されません。再起動後も維持する場合は、設定ファイル (**ifcfg** ファイル) を利用するか、コマンドをスクリプトに追加します。

コマンドラインを使用する例と設定ファイルの例は、各タスクの **nmtui** と **nmcli** の例の後に置かれています。ただし、**control-center** および **nm-connection-editor** といった **NetworkManager** へのグラフィカルユーザーインターフェースの使用法の前にあります。

1.8. NETWORKMANAGER とネットワークスクリプト

Red Hat Enterprise Linux の以前のリリースでは、ネットワークを設定するデフォルトの手段は ネットワークスクリプトを使用するものでした。ネットワークスクリプトという用語は、通常 **/etc/init.d/network/** スクリプトおよびこれが呼び出す他のインストール済みスクリプトを指します。ユーザーが提供するファイルは通常、設定とみなされますが、これらのスクリプトへの修正と解釈されることもあります。

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、Red Hat 開発者はスクリプトと **NetworkManager** が相互に協力できるようにしました。スクリプトに精通している管理者はそのままスクリプトを継続して使用できます。両方のシステムが並列して稼働し、協力できることが期待されています。以前のリリースからのほとんどのシェルスクリプトも機能し続けることが期待されています。Red Hat では、これらを最初にテストすることを推奨しています。

ネットワークスクリプトの実行

既存の環境変数すべてをクリアにし、クリーンな実行を確保する **systemctl** ユーティリティー *のみ* のスクリプトを実行します。コマンドは以下の形式をとります。

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

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** 接続の改ざんを防ぎます。**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とされており、**/etc/init.d/network** はフォールバックとなるセカンダリーとされています。

/etc/init.d/network スクリプトはイベント駆動型ではなく、以下のいずれかで実行されます。

1. 手動 (**systemctl** コマンドの **start|stop|restart network** のいずれかで)。
2. ネットワークサービスが有効となっている場合、起動時とシャットダウン時 (**systemctl enable network** コマンドによる)。

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。ユーザーは **ifup** および **ifdown** の各スクリプトを手動で呼び出すこともできます。

カスタムコマンドとネットワークスクリプト

`/sbin/ifup-local`、`ifdown-pre-local`、および `ifdown-local` の各スクリプト内のカスタムコマンドは、それらのデバイスが `/etc/init.d/network` サービスで制御されている時にのみ、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。必要であれば、`/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` だけが読み取り可能で、**NetworkManager** は読み取ることができません。**NetworkManager** を使っ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` スクリプトの説明については、「[dispatcher スクリプトの実行](#)」を参照してください。



注記

ユーザーが `initscripts` パッケージまたは関連する `rpm` に含まれるファイルを修正した場合、Red Hat はサポートを提供しません。

ネットワーク接続がオンラインおよびオフラインである時には、旧式のネットワークスクリプトと **NetworkManager** との両方の方法でカスタムタスクを実行する方法があります。**NetworkManager** が有効な場合は、`ifup` および `ifdown` スクリプトが **NetworkManager** に対して、問題となっているインターフェースを **NetworkManager** が管理しているかどうかを尋ねます。このインターフェースは、`ifcfg` ファイルの「`DEVICE=`」の行で見つかります。**NetworkManager** が該当するデバイスを管理しておらずデバイスが未接続の場合、`ifup` が **NetworkManager** に接続を開始するよう依頼します。

- **NetworkManager** がデバイスを管理していて、デバイスがすでに接続されている場合は、なにも実施されません。
- **NetworkManager** がデバイスを管理していない場合は、スクリプトは **NetworkManager** が存在する前からある古い **NetworkManager** 以外のメカニズムを使って接続を開始します。

`ifdown` を呼び出していて **NetworkManager** がデバイスを管理している場合、`ifdown` は **NetworkManager** に接続を切断するよう依頼します。

スクリプトは **NetworkManager** を動的にチェックするので、**NetworkManager** が実行していない場合、スクリプトは **NetworkManager** 以前のスクリプトベースの古いメカニズムにフォールバックします。

dispatcher スクリプトの実行

NetworkManager により、別のカスタムスクリプトを実行し、接続の状態に応じてサービスを開始/停止することができます。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。それぞれのスクリプトは、`root` が所有者である 実行可能ファイルで、ファイル所有者にのみ **書き込みのパーミッション** が設定され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。

1.9. SYSCONFIG ファイルを使ったネットワーク設定

`/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になります。ほとんどのネットワーク設定の情報は、ここに保存されます。例外は VPN、モバイルブロードバンド、および PPPoE 設定で、これらは `/etc/NetworkManager/` のサブディレクトリー内に保存されます。たとえば、インターフェース固有の情報は、`/etc/sysconfig/network-scripts/` ディレクトリー内の `ifcfg` ファイルに保存されます。

`/etc/sysconfig/network` はグローバル設定用のファイルです。VPN、モバイルブロードバンド、および PPPoE 接続の情報は、`/etc/NetworkManager/system-connections/` に保存されます。

Red Hat Enterprise Linux 7 では、`ifcfg` ファイルを変更しても、**NetworkManager** は自動的に変更を認識しないので、変更を通知する必要があります。**NetworkManager** のプロファイル設定を更新するツールを使用している場合、そのプロファイルを使用して接続するまでは、**NetworkManager** は変更を実行しません。たとえば、エディターを使用して設定ファイルが変更された場合、**NetworkManager** にその設定ファイルを再度読み込むように指示する必要があります。これを行うには、**root** で以下のコマンドを実行します。

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

上記のコマンドを実行すると、すべての接続プロファイルが読み込まれます。別の方法では、以下のコマンドを実行して、変更されたファイルである `ifcfg-ifname` をリロードします。

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

このコマンドは、複数のファイル名をとります。これらのコマンド実行には、**root** 権限が必要になります。ユーザー権限と権限の取得に関する詳細情報は、『[Red Hat Enterprise Linux 7 システム管理者のガイド](#)』および **su(1)** と **sudo(8)** の man ページを参照してください。

nmcli などのツールでなされた変更はリロードする必要がありませんが、関連するインターフェースをオフラインにしてから再度オンラインにする必要があります。これは、以下の形式のコマンドで実行できます。

```
nmcli dev disconnect interface-name
```

次に以下を実行します。

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

ifup コマンドの実行時に **NetworkManager** が稼働している場合、ネットワークスクリプトは **NetworkManager** の起動を試みますが、**NetworkManager** はネットワークスクリプトを起動することはありません。ネットワークスクリプトの詳細は、『[NetworkManager とネットワークスクリプト](#)』を参照してください。

ifup スクリプトは汎用スクリプトで、いくつかの動作を行った後に、**ifup-ethX** や **ifup-wireless**、**ifup-ppp** などのインターフェース固有のスクリプトを呼び出します。ユーザーが **ifup eth0** を手動で実行すると、以下のことが発生します。

1. **ifup** が `/etc/sysconfig/network-scripts/ifcfg-eth0` という名前のファイルを検索します。
2. `ifcfg` ファイルが存在する場合、**ifup** はそのファイルの **TYPE** キーを探して、どのタイプ固有のスクリプトを呼び出すか判断します。
3. **ifup** は **TYPE** に基づいて **ifup-wireless**、**ifup-eth**、または **ifup-XXX** のいずれかを呼び出します。
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. その後に、タイプ固有のスクリプトは、共通機能に **DHCP** や静的セットアップなどの **IP** 関連タスクの実行を許可します。

`/etc/init.d/network` は起動時にすべての `ifcfg` ファイルを読み込み、**ONBOOT=yes** となってい

るファイルすべてに関して、**NetworkManager** がすでに **ifcfg** ファイルの **DEVICE** を起動しているかどうかをチェックします。**NetworkManager** がデバイスを起動している最中、またはすでに起動し終わっている場合は、そのファイルについてはこれ以上なにも実行されず、次の **ONBOOT=yes** ファイルがチェックされます。**NetworkManager** がまだデバイスを起動していない場合は、**initscripts** は従来の動作を継続し、その **ifcfg** ファイルの **ifup** を呼び出します。

その結果、**ifcfg** ファイルで **ONBOOT=yes** となっているものはすべて、**NetworkManager** または **initscripts** でシステム起動時に開始されるはずですが。これにより、(ISDN やアナログダイヤルアップのモデムなどの) **NetworkManager** が処理しないレガシーネットワークタイプや **NetworkManager** のサポート対象となっていない新規アプリケーションは、**NetworkManager** がこれらを処理できなくても **initscripts** で正常に開始されることになります。



注記

ifcfg のバックアップファイルは、使用中のファイルとは別の場所に保存することが推奨されます。このスクリプトは、**.old**、**.orig**、**.rpmnew**、**.rpmorig**、および **.rpmsave** の拡張子を除き、例外なく **ifcfg-*** を実行します。**/etc** ディレクトリー内にはバックアップファイルを保存しないことが最善の方法になります。

1.10. 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 Central Regulatory Domain Agent が含まれ、特定地区のワイヤレス規制ルールがカーネルに提供されます。これは特定の **udev** スクリプトによって使われるので、**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。規制ドメインに新たな変更があると、カーネルは **udev** イベントを送信して **crda** 実行のトリガーとなります。Linux ワイヤレスサブシステム (IEEE-802.11) が、規制ドメイン変更のトリガーとなります。このサブシステムでは、**regulatory.bin** ファイルを使って規制データベース情報が維持されます。

お使いのシステムの規制ドメインは、**setregdomain** ユーティリティーによって設定されます。引数は不要で、管理者が手動で呼び出すのではなく、通常は **udev** 等のシステムスクリプトを通じて呼び出されます。国コードルックアップに失敗した場合、システム管理者は **/etc/sysconfig/regdomain** ファイルで **COUNTRY** 環境変数を定義することができます。

規制ドメインの詳細については、以下の **man** ページを参照してください。

- **setregdomain(1)** **man** ページ: 国コードを元にした規制ドメインの設定
- **crda(8)** **man** ページ: カーネルへの特定 ISO または IEC 3166 alpha2 のワイヤレス規制ドメインの送信
- **regulatory.bin(5)** **man** ページ: Linux ワイヤレス規制データベース
- **iw(8)** **man** ページ: ワイヤレスデバイスおよびその設定の表示または操作

1.11. NETCONSOLE の設定

netconsole カーネルモジュールにより、カーネルメッセージをネットワークを通じて他のコンピューターに記録することができます。これにより、ディスクへの記録に失敗した場合やシリアルコンソールの使用が不可能な場合に、カーネルのデバッグを行うことができます。

rsyslog サーバーがネットワーク上にあり、**rsyslogd** デーモンが 514/udp ポートをリッスンしている必要があります。514/udp ポートをリッスンしてネットワークからメッセージを受信するように **rsyslogd** を設定するには、**/etc/rsyslog.conf** の **MODULES** セクションで、以下の行をアンコメントします。

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

rsyslogd サービスを再起動して、変更を有効にします。

```
]# systemctl restart rsyslog
```

rsyslogd が 514/udp ポートをリッスンしていることを確認するには、以下のコマンドを使用します。

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

netstat -l により **0.0.0.0:syslog** および **[::]:syslog** の値が出力されていますが、これは **rsyslogd** が **/etc/services** ファイルで定義されるデフォルトの **netconsole** ポートを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
```

送信元マシンの設定

Red Hat Enterprise Linux 7 では、**netconsole** は **initscripts** パッケージに含まれ、ファイル **/etc/sysconfig/netconsole** を使って設定します。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には、**/etc/sysconfig/netconsole** ファイルの **SYSLOGADDR** 変数の値を、**syslogd** サーバーの IP アドレスに設定します。以下に例を示します。

```
SYSLOGADDR=192.168.0.1
```

netconsole サービスを再起動して、変更を有効にします。続いて、次の再起動後に再実行されるように **netconsole.service** を有効にします。

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

デフォルトでは、**rsyslogd** サーバーは、クライアントからの **netconsole** メッセージを **/var/log/messages** または **rsyslog.conf** で指定したファイルに書き込みます。



注記

別のポートを使用するように **rsyslogd** および **netconsole.service** を設定するには、**/etc/rsyslog.conf** の以下の行を希望するポート番号に変更します。

```
$UDPServerRun <PORT>
```

送信元マシンで、**/etc/sysconfig/netconsole** ファイルの以下の行をアンコメントして編集します。

```
SYSLOGPORT=514
```

netconsole 設定の詳細およびトラブルシューティングのヒントについては、[Netconsole カーネルのドキュメント](#) を参照してください。

1.12. その他のリソース

- **man(1)** man ページ: man ページについての説明とそれらの見つけ方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について説明されています。
- **NetworkManager.conf(5)** man ページ: **NetworkManager** 設定ファイルについて説明されています。
- **/usr/share/doc/initscripts-version/sysconfig.txt**: レガシーのネットワークサービスで理解される **ifcfg** 設定ファイルとそのディレクティブについて説明されています。
- **/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です。

第2章 IP ネットワークの設定

2.1. 静的および動的インターフェースの設定

静的アドレス指定や動的アドレス指定は、いつ使うべきでしょうか。この判断は主観的なもので、ユーザーのアクセスニーズや特定の要件によって異なります。通常は特定の判断よりも、ポリシーがあってそれを文書化し、一貫性を持って適用することの方が重要になります。従来の企業 LAN では、サーバーの数は他のホストよりも少ないことが多かったので、これは容易に判断できました。プロビジョニングとインストールツールがあれば新規ホストに静的設定を提供することは簡単で、それらのツールを使用することでワークフローと要件は変化します。以下の 2 つのセクションは、これらの判断プロセスを経験したことがないユーザーのための基本的なガイダンスになります。経験のあるシステム管理者は、ここで議論されるものとは異なるルールや要件のセットを持っていることでしょう。自動設定および管理についての詳細情報は、『[Red Hat Enterprise Linux 7 システム管理者のガイド](#)』の **OpenLMI** のセクション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 **kickstart** の使用方法を説明しており、これはネットワーク設定の自動割り当てにも使用できるものです。

2.1.1. 静的ネットワークインタフェース設定を使用する場合

静的 **IP** アドレス指定は、**DHCP** のような自動割り当て方式が失敗した場合にネットワークを確実に利用可能とするサーバーやデバイスに使用します。**DHCP**、**DNS**、および認証サーバーが典型的な例になります。帯域外管理デバイスは他のネットワークインフラストラクチャーにできるだけ依存せずに機能することが求められているので、これらのインターフェースも静的設定にする価値があります。

不可欠なホスト以外で静的 **IP** アドレス指定が望ましいものには、可能な場合、自動プロビジョニング方式を使用します。たとえば、**DHCP** サーバーは、毎回同じホストへ同じ **IP** アドレスを提供するように設定できます。この方法は、共有プリンターなどに使用できます。

「[ネットワーク設定方法の選択](#)」に記載の設定ツールはすべて、手動で静的 **IP** アドレスを割り当てることができます。**nmcli** ツールも、スクリプト化されたネットワーク設定割り当てに使用することができます。

2.1.2. 動的ネットワークインタフェース設定を使用する場合

IP アドレスおよび他のネットワーク情報を動的に割り当てる理由が特になければ、動的割り当て有効にして使用してください。手動設定のプランニングと文書化にかかる時間は、他の作業に当てることができます。*dynamic host control protocol* (**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式です。詳細情報は、「[DHCP を使用する理由](#)」を参照してください。

デフォルトでは、インターフェース設定ファイルの **BOOTPROTO** が **dhcp** に設定され、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と、**NetworkManager** は **DHCP** クライアントの **dhclient** を呼び出します。**DHCP** が必要な場合は、インターフェース上で、すべてのインターネットプロトコル (**IPv4** および **IPv6**) に **dhclient** のインスタンスが開始されます。**NetworkManager** が実行中ではない場合、もしくはこれがインターフェースを管理していない場合は、必要に応じてレガシー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。

2.1.3. ネットワーク設定方法の選択

- **NetworkManager** のテキストユーザーインターフェースツールである **nmtui** を使ってインターフェースを設定するには、「[テキスト形式のユーザーインターフェース \(nmtui\) の使用](#)」に進みます。
- **NetworkManager** のコマンドラインツールである **nmcli** を使ってインターフェースを設定するには、「[NetworkManager コマンドラインツール \(nmcli\) の使用](#)」に進みます。

- 手動でネットワークインターフェースを設定するには、「[ネットワーク設定ファイルの編集](#)」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ースツールを使ってネットワークを設定するには、「[GNOME グラフィカルユーザーインターフェースによる NetworkManager の使用](#)」に進みます。

2.1.4. テキスト形式のユーザーインターフェース (nmtui) の使用

テキスト形式のユーザーインターフェースツール **nmtui** を使うと、ターミナルのウィンドウでインターフェースを設定できます。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
```

テキスト形式のインターフェースが表示されます。無効なコマンドの場合は、使用法に関するメッセージがプリントされます。

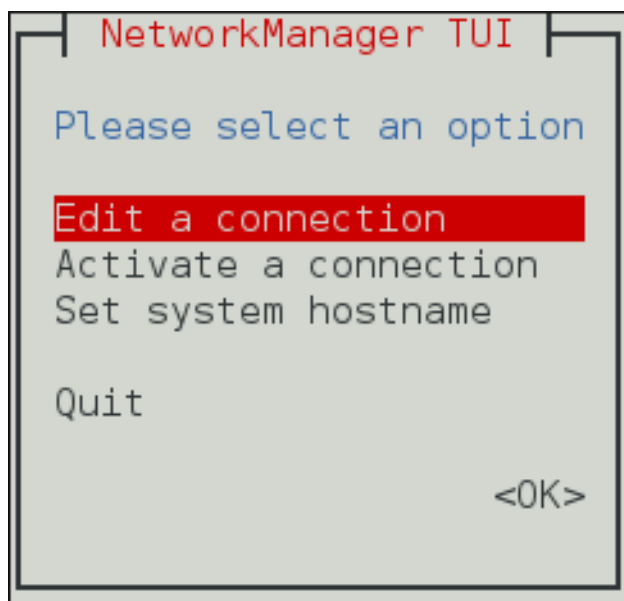


図2.1 NetworkManager のテキスト形式ユーザーインターフェースの開始メニュー

移動するには矢印キーを使用するか、**Tab** を押して次に進むか **Shift+Tab** を押して前に戻ります。**Enter** を押してオプションを選びます。**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、接続の再アクティブ化が必要です。この場合は、以下の手順を実施します。

1. 接続をアクティベートする メニューエントリーを選択します。

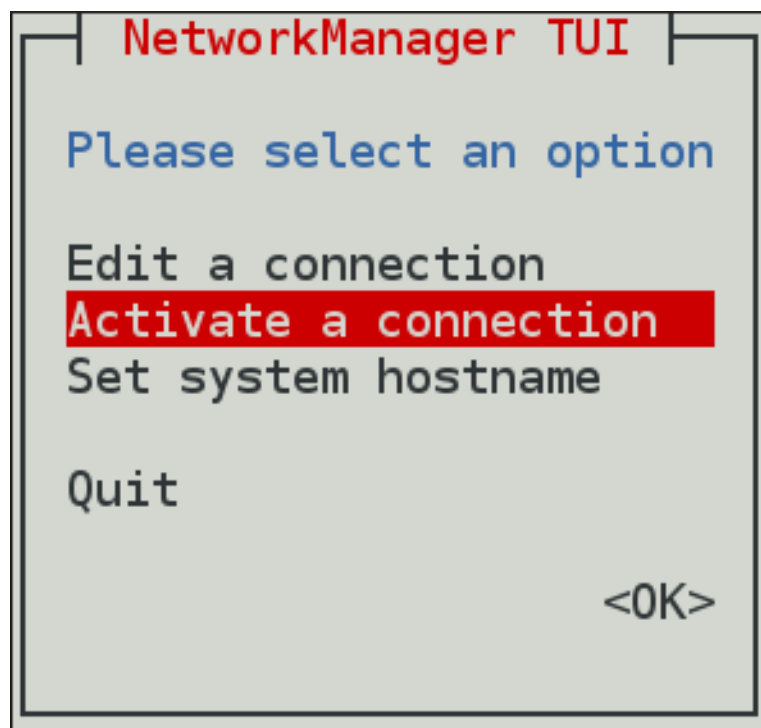


図2.2 接続のアクティブ化

- 修正した接続を選択します。右側で **解除** ボタンをクリックします。

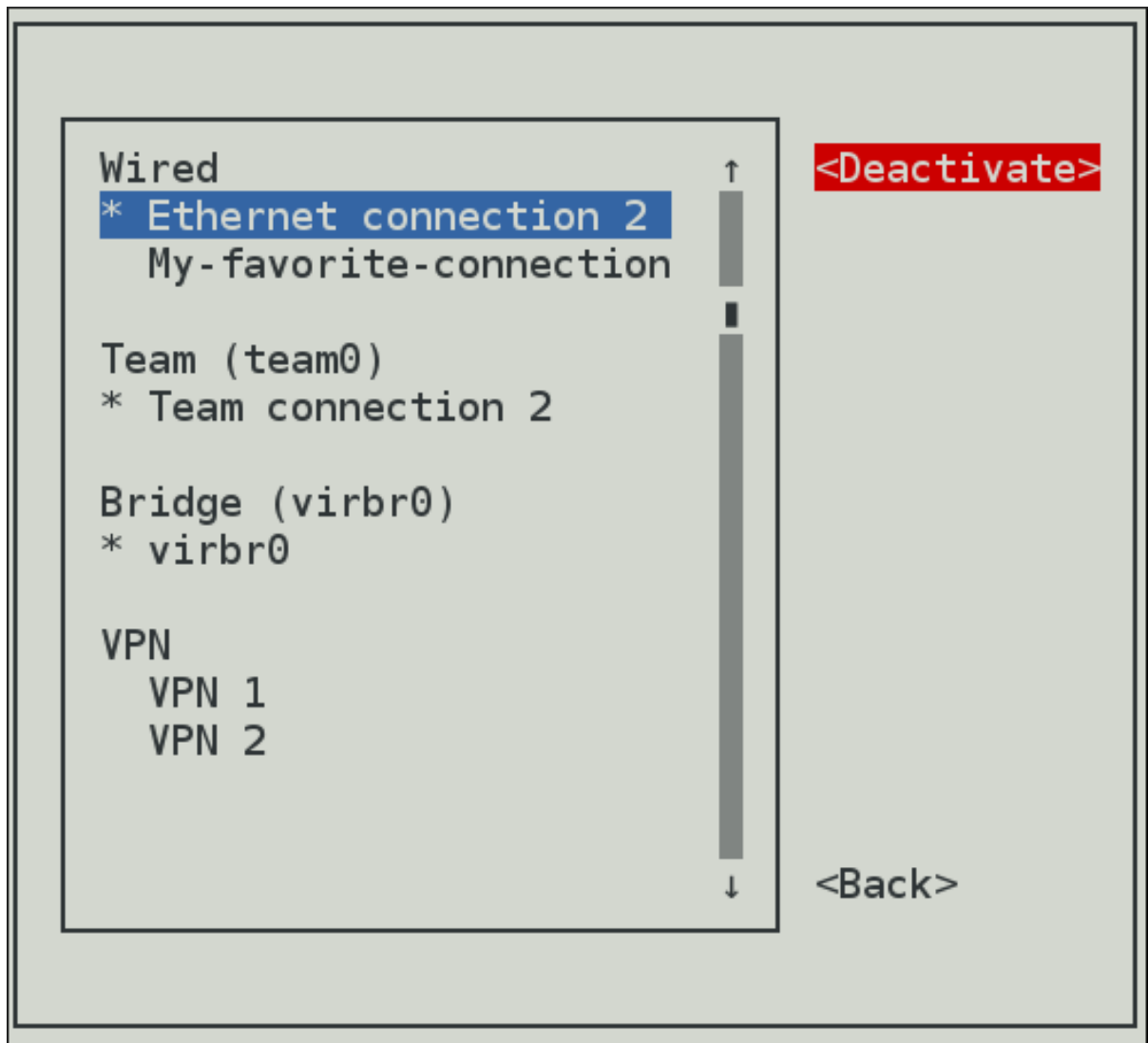


図2.3 修正した接続の非アクティブ化

3. もう一度接続を選択し、**アクティベート** ボタンをクリックします。

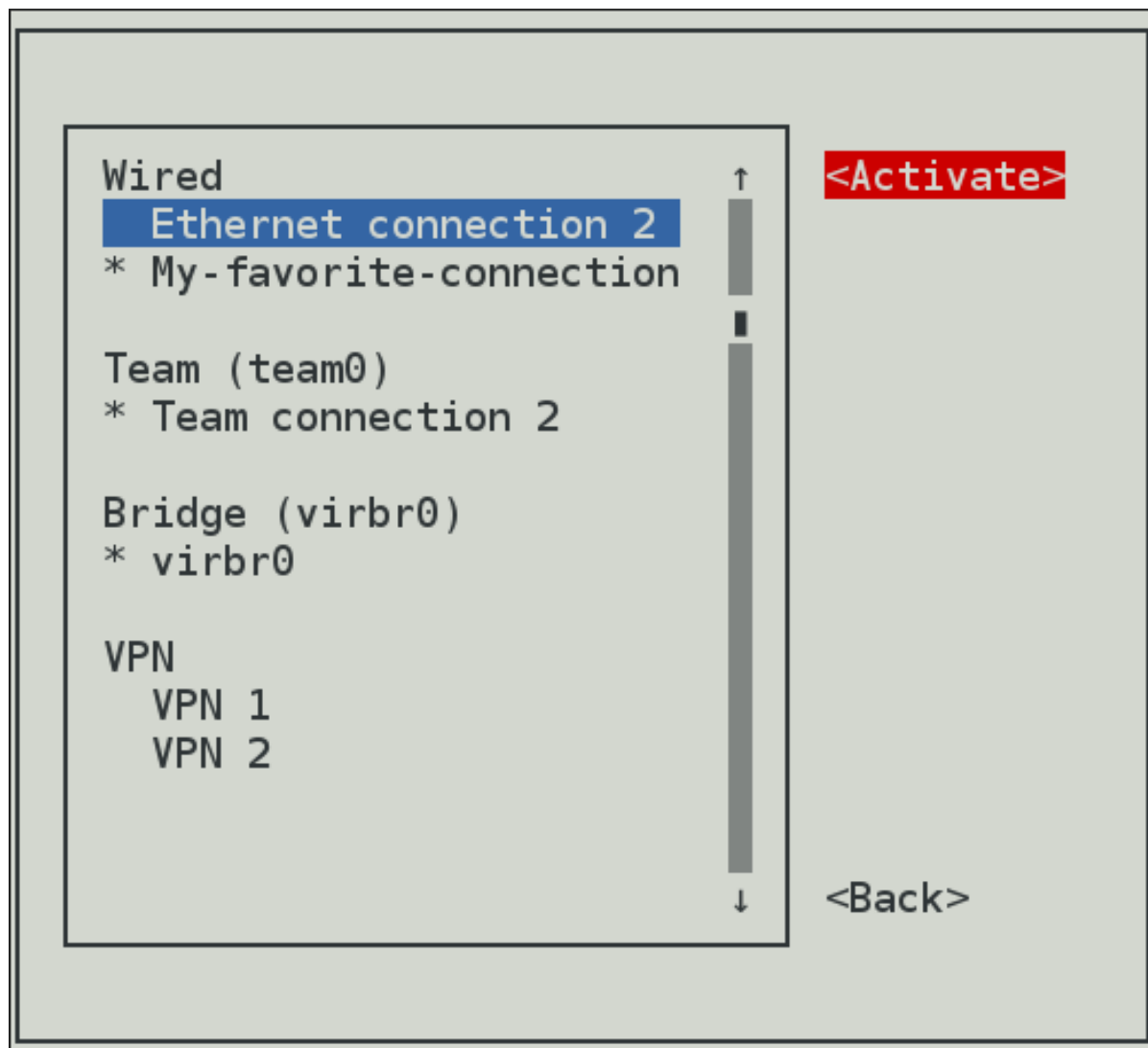


図2.4 修正した接続の再アクティブ化

`nmtui` のインストール方法については、「[テキスト形式のユーザーインターフェース \(nmtui\) を使ったネットワーク設定](#)」を参照してください。

2.1.5. NetworkManager コマンドラインツール (`nmcli`) の使用

`nmcli` (NetworkManager コマンドラインインターフェース) 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の状態を把握するのに使われます。このユーティリティーは、`nm-applet` またはその他のグラフィカルクライアントの代替として使うことができます。`nmcli` は、ネットワークデバイスの状態を制御および表示するのに加えて、ネットワーク接続を作成、表示、編集、削除、アクティブ化、および非アクティブ化するのに用いられます。

ユーザーおよびスクリプトの両方が、`nmcli` ユーティリティーを使用して **NetworkManager** を制御することができます。

- サーバー、ヘッドレスのマシン、およびターミナルについては、GUI を介さずに直接 `nmcli` を使って **NetworkManager** を制御し、ネットワーク接続を作成、編集、開始、および停止したり、ネットワークの状態を把握したりすることができます。
- スクリプトに関しては、`nmcli` はスクリプト処理に適した簡素出力フォーマットをサポートします。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以下のとおりです。

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

ここで OBJECT は、以下のオプションのいずれかです。**general**、**networking**、**radio**、**connection**、**device**、**agent**、および **monitor**。コマンド内では、これらのオプションを自由に短縮することができます。例: **nmcli con help**

すぐに役に立つ OPTIONS

-t (terse)

このモードは、コンピューター (スクリプト) 処理に適するように作られています。

-p (pretty)

このモードでは、**nmcli** により人間が理解可能な出力が生成されます。ヘッダーが表示され、値がそこに並べられます。

-h (help)

ヘルプ情報が表示されます。

nmcli ツールには、コンテキスト感知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
```

さまざまな **nmcli** コマンドの例 (簡単な説明)

『nmcli-examples(5)』 man ページには、役に立つ例が多数記載されています。以下に、そのいくつかを示します。

NetworkManager の全体的な状態:

```
nmcli general status
```

NetworkManager の現在のログ記録状態:

```
nmcli general logging
```

すべての接続:

```
nmcli connection show
```

現在アクティブな接続だけを表示するには、以下のように **--active** (または **-a**) オプションを追加します。

```
nmcli connection show --active
```

NetworkManager が認識するデバイスおよびその状態:

```
nmcli device status
```

nmcli を使用したインターフェースの開始および停止

nmcli ツールを使用すると、マスターを含めたネットワークインターフェースの開始および停止ができます。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 コマンドでは、デバイスからの接続は非アクティブ化されますが、その後デバイスが接続を自動的にアクティブ化することは禁じません。**nmcli device disconnect** コマンドでは、デバイスが切断され、手動の操作がない限りその後デバイスが接続を自動的にアクティブ化することはありません。

nmcli インタラクティブ接続エディター

nmcli ツールには、インタラクティブ接続エディターがあります。これを使用するには、以下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、有効な接続の **type** オプションを **nmcli con edit** コマンドに追加して、直接 **nmcli** プロンプト表示とすることができます。既存の接続プロファイルを編集する場合は、以下のような形式になります。

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

新規接続プロファイルの編集には、以下の形式を使用します。

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

有効なコマンド一覧を確認するには、**nmcli** プロンプトで **help** と入力します。設定およびプロパティの説明を確認するには、**describe** コマンドを使用します。

```
describe setting.property
```

例を示します。

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

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、データソースに接続するのに必要な接続プロパティ情報が含まれます。NetworkManager で使用する新規プロファイルを **作成する** には、以下のコマンドを使用します。

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

nmcli c add では、以下に示す異なる 2 つのタイプのパラメーターが使用可能です。

プロパティ名

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 eth0
```

- connection.id

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

属性およびその設定に関する詳細は、**nm-settings(5)** man ページを参照してください。

エイリアス名

内部的にプロパティに翻訳された、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。

- type (connection.type プロパティ)

```
nmcli c add type bond
```

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

```
nmcli c add ifname eth0
```

- con-name (connection.id プロパティ)

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

以前のバージョンの **nmcli** では、接続を作成するのに **エイリアス名** を使用する必要がありました。たとえば、**ifname** eth0 および **con-name** My Connection。以下の形式のコマンドが使用されました。

```
nmcli c add type ethernet ifname eth0 con-name "My Connection"
```

最新のバージョンでは、**プロパティ名** と **エイリアス名** の両方を、区別なく使用することができます。以下の例は、すべて有効で同じ結果が得られます。

```
nmcli c add type ethernet ifname eth0 con-name "My Connection"
ethernet.mtu 1600
```

```
nmcli c add connection.type ethernet ifname eth0 con-name "My Connection"
ethernet.mtu 1600
```

```
nmcli c add connection.type ethernet connection.interface-name eth0
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** 引数だけは、すべての接続の種類で必須です。また、**ifname** は **bond**、**team**、**bridge**、および **vlan** を除くすべての種類で必須です。

type (*type_name*)

接続の種類です。例:

```
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ースです。例:

```
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、**connection.interface-name** を **eth1** に変更するには、以下のようにコマンドを実行します。

```
nmcli c modify "My Connection" connection.id "My favorite connection"
connection.interface-name eth1
```



注記

一般的には、**プロパティ名** が使用されます。**エイリアス名** は、互換性の理由からしか使用されません。

また、イーサネットの MTU を 1600 に設定するには、**サイズ**を以下のように修正します。

```
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)
```

2.1.6. nmcli オプションについて

nmcli の重要なプロパティオプションを以下に示します。すべてのリストは、『nmcli(1)』man ページ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。それぞれの接続の種類には、種類に固有のコマンドオプションがあります。『nmcli(1)』man ページの **TYPE_SPECIFIC_OPTIONS** リストを参照してください。以下に例を示します。

- **gsm** 接続の場合は、**apn** でアクセスポイント名を指定する必要があります。

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

- **wifi** デバイスの場合は、**ssid** でサービスセットの識別子を指定する必要があります。

```
nmcli c add connection.type wifi ssid My_identifier
```

connection.interface-name

接続に関連するデバイス名です。

```
nmcli con add connection.interface-name eth0 type ethernet
```

connection.id

接続プロファイルに使用される名前です。接続名を指定しないと、以下のように生成されます。

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

connection.id は 接続プロファイル の名前で、デバイスを表すインターフェース名 (**wlan0**、**ens3**、**em1** など) 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ースにちなんで接続に名前を付けることができますが、これらは別のもので、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ができます。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。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、必要に応じてそれらをインターフェースに適用します。**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** 等の **nmcli** コマンドで最も重要なオプションを以下に示します。

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です。**nmcli connection** コマンド内で **id** を使用して、接続を特定することができます。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てる一意の識別用文字列です。**nmcli connection** コマンド内で **uuid** を使用して、接続を特定することができます。

2.1.7. 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一覧表示するには、以下のコマンドを実行します。

```
~]$ nmcli con show
```

NAME	UUID	TYPE
Auto Ethernet	9b7f2511-5432-40ae-b091-af2457dfd988	802-3-ethernet --
ens3	fb157a65-ad32-47ed-858c-102a48e064a2	802-3-ethernet
MyWiFi	91451385-4eb8-4080-8b82-720aab8328dd	802-11-wireless
wlan0		

出力の NAME フィールドは常に接続 ID (名前) を表す事に留意してください。これはインターフェー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 フィールドである **ens3** は、ユーザーがプロファイルに割り当て、インターフェース ens3 に適用される接続 ID です。最後の接続では、ユーザーは接続 ID **MyWiFi** をインターフェース wlan0 に割り当てています。

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	--

デバイスを **NetworkManager** が管理しない状態 (unmanaged) に設定するには、以下のコマンドを実行します。

```
$ nmcli device set ifname managed no
```

たとえば、**eth2** を unmanaged に設定するには、以下のコマンドを実行します。

```
$ nmcli device status
```

DEVICE	TYPE	STATE	CONNECTION
bond0	bond	connected	bond0
virbr0	bridge	connected	virbr0
eth1	ethernet	connected	bond-slave-eth1
eth2	ethernet	connected	bond-slave-eth2
eth0	ethernet	unmanaged	--

```
$ nmcli device set eth2 managed no
```

```
$ nmcli device status
```

DEVICE	TYPE	STATE	CONNECTION
bond0	bond	connected	bond0
virbr0	bridge	connected	virbr0

```
eth1      ethernet  connected  bond-slave-eth1
eth2      ethernet  unmanaged  --
eth0      ethernet  unmanaged  --
```



注記

デバイスを **unmanaged** に設定すると、**NetworkManager** はそのデバイスを制御しなくなります。ただし、デバイスは接続された状態を維持します。

動的イーサネット接続を追加する

動的 **IP** 設定のイーサネット設定プロファイルを追加して、**DHCP** がそのネットワーク設定を割り当てられるようにするには、以下の形式のコマンドを使用します。

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

NetworkManager が内部パラメーター **connection.autoconnect** を **yes** に設定します。また **NetworkManager** は設定を **/etc/sysconfig/network-scripts/ifcfg-my-office** にも書き出します。ここでは、ONBOOT ディレクティブが **yes** に設定されます。

ifcfg ファイルへの手動での変更は、当該インターフェースが次にオンラインになるまで **NetworkManager** には認識されないことに注意してください。設定ファイルの使用法に関しては、「[sysconfig ファイルを使ったネットワーク設定](#)」を参照してください。

イーサネット接続を有効にするには、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    --
```

ホストが **DHCP** サーバーに送信するホスト名を変更するには、以下のように **dhcp-hostname** プロパティを編集します。

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

ホストが **DHCP** に送信する **IPv4** クライアント ID を変更するには、以下のように **dhcp-client-id** プロパティを編集します。

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

IPv6 には **dhcp-client-id** プロパティがなく、**dhclient** が **IPv6** に識別子を作成します。詳細は、**dhclient(8)** man ページを参照してください。

DHCP サーバーがホストに送信する **DNS** サーバーを無視するようにするには、以下のように **ignore-auto-dns** プロパティを編集します。

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

属性およびその設定に関する詳細は、**nm-settings(5)** man ページを参照してください。

例2.1 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

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、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> 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、プロファイルを次の再起動時までメモリーにだけ保持することもできます。

静的イーサネット接続を追加する

静的 **IPv4** 設定のイーサネット接続を追加するには、以下の形式のコマンドを使用します。

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

ip6 および **gw6** のオプションを使用して **IPv6** アドレスとゲートウェイ情報を追加することもできます。

たとえば、**IPv4** アドレスおよびゲートウェイのみの静的イーサネット接続を作成するコマンドは、以下のようになります。

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

オプションで、デバイスに同時に **IPv6** アドレスとゲートウェイを指定する場合は、以下のようになります。

```
~]$ 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.
```

NetworkManager が内部パラメーター **ipv4.method** を **manual** に、**connection.autoconnect** を **yes** に設定します。また **NetworkManager** は設定を **/etc/sysconfig/network-scripts/ifcfg-my-office** にも書き出します。ここでは、対応する BOOTPROTO が **none** に、ONBOOT が **yes** に設定されます。

ifcfg ファイルへの手動での変更は、当該インターフェースが次にオンラインになるまで **NetworkManager** には認識されないことに注意してください。設定ファイルの使用方法に関しては、「[sysconfig ファイルを使ったネットワーク設定](#)」を参照してください。

2 つの **IPv4 DNS** サーバーアドレスを設定するには、以下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"
```

このコマンドにより、以前に設定された **DNS** サーバーが置換さ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、以下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"
```

このコマンドにより、以前に設定された **DNS** サーバーが置換されることに注意してください。別の方法では、以下のように **+** 接頭辞を使って、新たな **DNS** サーバーを以前のセットに追加します。

```
~]$ 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"
```

新規イーサネット接続を有効にするには、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[出力は省略されています]
```

-p, **--pretty** オプションを使用すると、出力にタイトルバナーが追加され、セクションが分けられます。

例2.2 インタラクティブエディターを使用して静的イーサネット接続を設定する

インタラクティブエディターを使用して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、プロファイルを次の再起動時までメモリーにだけ保持することもできます。

特定のデバイスにプロファイルをロックする

特定のデバイスにプロファイルをロックするために、ここまでのコマンド例ではインターフェース名が含まれています。例を示します。

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

互換性のあるイーサネットインターフェースすべてでプロファイルを利用可能とするには、以下のコマンドを実行します。

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

特定のインターフェースを指定しない場合でも、**ifname** 属性を使用する必要があることに注意してください。互換性のあるデバイスすべてでプロファイルを使用可能とするために、ワイルドカード記号 ***** を使用しています。

プロファイルを特定の MAC アドレスにロックするには、以下の形式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
```

Wi-Fi 接続を追加する

利用可能な Wi-Fi アクセスポイントを表示するには、以下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** 設定の Wi-Fi 接続プロファイルを作成し、自動 **DNS** アクセス割り当てを許可するには、以下のコマンドを実行します。

```
~]$ nmcli con add con-name MyCafe ifname wlan0 type wifi ssid MyCafe \
ip4 192.168.100.101/24 gw4 192.168.100.1
```

WPA2 パスワードをたとえば「caffeine」に設定するには、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に関する情報は、『[Red Hat Enterprise Linux 7 セキュリティガイド](#)』を参照してください。

Wi-Fi のステータスを変更するには、以下の形式のコマンドを実行します。

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

特定プロパティーの変更

特定のプロパティー、たとえば **mtu**、をチェックするには、以下のコマンドを実行します。

```
~]$ 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、以下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、以下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 **802-3-ethernet** や **802-11-wireless** といったパラメーターを設定として参照し、**mtu** を設定のプロパティーとして参照することに留意してください。プロパティーおよびそれらの設定に関する詳細情報は、**nm-settings(5)** man ページを参照してください。

2.1.8. nmcli を使った静的ルートの設定

nmcli ツールを使って静的ルートを設定するには、コマンドラインもしくはインタラクティブエディターモードを使用することができます。

例2.3 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、静的ルートを既存のイーサネット接続用に設定するには、以下のコマンドを実行します。

```
~]# nmcli connection modify eth0 +ipv4.routes "192.168.122.0/24
10.10.10.1"
```

これで **192.168.122.0/24** サブネットへのトラフィックが **10.10.10.1** のゲートウェイに向けられます。

例2.4 nmcli エディターを使って静的ルートを設定する

インタラクティブエディターを使用して、静的ルートをイーサネット接続用に設定するには、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.routes 192.168.122.0/24 10.10.10.1
nmcli>
nmcli> save persistent
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
~]$
```

2.2. ネットワーク設定ファイルの編集

2.2.1. ifcfg ファイルを使用したネットワークインターフェースの設定

インターフェース設定ファイルは、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ースを制御します。システムは、ブート時にこれらのファイルを使用して、どのインターフェースをアクティブにして、どのように設定するかを決定します。これらのファイルは、通常 **ifcfg-name** と命名されます。接尾辞 *name* は設定ファイルが制御するデバイス名を指します。**ifcfg** ファイルの接尾辞は慣習的に、設定ファイル自体で **DEVICE** ディレクティブが提供する文字列と同じものになります。

静的ネットワーク設定

たとえば、**ifcfg** ファイルを使って **eth0** という名前のインターフェースを静的ネットワークで設定するには、**/etc/sysconfig/network-scripts/** ディレクトリー内に以下のような内容で **ifcfg-eth0** という名前のファイルを作成します。

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
PREFIX=24
IPADDR=10.0.1.27
```

ネットワークまたはブロードキャストアドレスは、**ipcalc** が自動的に計算するので指定する必要はありません。



重要

Red Hat Enterprise Linux 7 では、「[8章 ネットワークデバイス命名における一貫性](#)」で説明するようにネットワークインターフェー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っ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出ます。

動的ネットワーク設定

たとえば、**ifcfg** ファイルを使って **em1** という名前のインターフェースを動的ネットワークで設定するには、**/etc/sysconfig/network-scripts/** ディレクトリー内に以下のような内容で **ifcfg-em1** という名前のファイルを作成します。

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

インターフェースが **DHCP** サーバーに異なるホスト名を送信するよう設定するには、**ifcfg** ファイルに以下の行を追加します。

```
DHCP_HOSTNAME=hostname
```

インターフェースが **DHCP** サーバーに異なる完全修飾ドメイン名 (FQDN) を送信するよう設定するには、**ifcfg** ファイルに以下の行を追加します。

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



注記

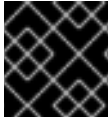
指定した **ifcfg** ファイルでは、**DHCP_HOSTNAME** または **DHCP_FQDN** ディレクティブのどちらか 1 つしか使用しないでください。**DHCP_HOSTNAME** と **DHCP_FQDN** の両方を指定した場合には、後者だけが使用されます。

インターフェースが特定の **DNS** サーバーを使用するよう設定するには、**ifcfg** ファイルに以下の行を追加します。

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

ここで、*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は指定した **DNS** サーバーで **/etc/resolv.conf** を更新します。指定する必要のある **DNS** サーバーのアドレスは 1 つだけで、その他はオプションです。

デフォルトでは、インターフェース設定ファイルの **BOOTPROTO** が **dhcp** に設定され、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と、**NetworkManager** は **DHCP** クライアントの **dhclient** を呼び出します。**DHCP** が必要な場合は、インターフェース上で、すべてのインターネットプロトコル (**IPv4** および **IPv6**) に **dhclient** のインスタンスが開始されます。**NetworkManager** が実行中ではない場合、もしくはこれがインターフェースを管理していない場合は、必要に応じてレガシー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。詳細は、「[動的ネットワークインターフェース設定を使用する場合](#)」を参照してください。



重要

設定を適用するには、**nmcli c reload** コマンドを入力する必要があります。

2.2.2. カーネルコマンドラインからのネットワークセッティングの設定

インターフェースから iSCSI ターゲット上のルートファイルシステムに接続する場合、インストールされたシステムにはネットワークセッティングが設定されません。この問題を回避するには、以下の手順を実施します。

1. **dracut** ユーティリティをインストールします。**dracut** 使用の詳細は、[『Red Hat Enterprise Linux 7 システム管理者のガイド』](#)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、設定を定義します。

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

- **dhcp**: DHCP 設定
- **dhcp6**: DHCP IPv6 設定
- **auto6**: 自動 IPv6 設定
- **on**, **any**: カーネルで利用可能な任意のプロトコル (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::eth0  
: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 ユーティリティによりネットワーク接続が設定され、新たな **ifcfg** ファイルが生成されます。このファイルを **/etc/sysconfig/network-scripts/** ファイルにコピーすることができます。

2.2.3. ip コマンドを使用したネットワークインターフェースの設定

ip ユーティリティを使うと、インターフェースに **IP** アドレスを割り当てることができます。コマンドは以下の形式になります。

```
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

インターフェースに **IP** アドレスを割り当てるには、**root** で以下のコマンドを実行します。

```
~]# ip address add 10.0.0.3/24 dev eth0
The address assignment of a specific device can be viewed as follows:
~]# ip addr show dev eth0
2: eth0: <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 eth0
        valid_lft 58682sec preferred_lft 58682sec
    inet6 fe80::f2de:f1ff:fe7b:6e5f/64 scope link
        valid_lft forever preferred_lft forever
```

ip-address(8) man ページでは、他の例やコマンドオプションが説明されています。

ip コマンドを使って複数のアドレスを設定する

ip ユーティリティでは同一インターフェースへの複数のアドレス割り当てをサポートしているので、エイリアスインターフェースを使って複数アドレスを同一インターフェースにバインドする必要がありません。**ip** コマンドによるアドレスの割り当ては、複数のアドレス割り当てのために繰り返すことができます。例を示します。

```
~]# ip address add 192.168.2.223/24 dev eth1
~]# ip address add 192.168.4.223/24 dev eth1
~]# ip addr
3: eth1: <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 eth1
    inet 192.168.4.223/24 scope global eth1
```

ip ユーティリティのコマンドは、**ip(8)** man ページで確認することができます。



注記

コマンドライン上で実行される **ip** コマンドは、システム再起動後は維持されません。

2.2.4. 静的ルートおよびデフォルトゲートウェイ

静的ルートは、デフォルトゲートウェイを通過すべきでない、通過してはいけないトラフィックのためのものです。ルーティングは通常、ネットワーク上にあるルーティング専用のデバイスによって処理されます (ただし、どのデバイスもルーティングを実行するように設定可能です)。このため多くの場合、Red Hat Enterprise Linux サーバーやクライアントで静的ルートを設定する必要はありません。例外は、暗号化 VPN トンネルの通過が必要なトラフィックや、費用やセキュリティのために特定のルートを通すべきトラフィックなどです。デフォルトゲートウェイは、ローカルネットワークが宛先となっておらず、ルーティングテーブルに優先ルートが指定されていないすべてのトラフィックのためのものです。デフォルトゲートウェイは従来、専用ネットワークルーターでした。



注記

専門知識のさらなる拡充を図るには、[Red Hat System Administration I \(RH124\)](#) トレーニングコースもあります。

静的ルートが必要な場合は、**ip route add** コマンドでルーティングテーブルに追加でき、削除したい時は **ip route del** コマンドを使います。頻繁に使用される **ip route** コマンドは、以下の形式で用います。

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

オプションおよび形式についての詳細は、**ip-route(8)** man ページを参照してください。

オプションなしで **ip route** コマンドを実行すると、**IP** ルーティングテーブルが表示されます。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 eth0 proto kernel scope link src 192.168.122.126
```

ホストアドレス、つまり単一 **IP** アドレスに静的ルートを追加するには、**root** で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 ifname]
```

ここでの **192.0.2.1** はドット形式 10 進法でのホストの **IP** アドレスに、**10.0.0.1** はネクストホップアドレスに、**ifname** はネクストホップに進む終了インターフェースになります。

ネットワーク、つまり **IP** アドレスの範囲を表す **IP** アドレスに静的ルートを追加するには、**root** で以下のコマンドを実行します。

```
ip route add 192.0.2.0/24 via 10.0.0.1 [dev ifname]
```

ここでの **192.0.2.0** はドット形式 10 進法での宛先ネットワークの **IP** アドレスに、**/24** はネットワークプレフィックスになります。ネットワークプレフィックスは、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、ネットワークプレフィックス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。

静的ルート設定はインターフェースごとに、**/etc/sysconfig/network-scripts/route-interface** ファイルに格納することができます。たとえば、eth0 インターフェースの静的ルートは、**/etc/sysconfig/network-scripts/route-eth0** ファイルに格納されます。**route-interface** ファイルには **ip** コマンド引数とネットワーク/ネットマスクディレクティブの 2 つの形式があり、これらは後で説明されます。

ip route コマンドに関する詳細情報は、**ip-route(8)** man ページを参照してください。

デフォルトゲートウェイを設定する

デフォルトゲートウェイは、ネットワークスクリプトにより決定されます。このスクリプトは、まず **/etc/sysconfig/network** ファイルを、その後に「up」の状態にあるインターフェースのネットワークインターフェースの **ifcfg** ファイルを解析します。**ifcfg** ファイルは数字の小さい順に解析され、最後に読み取られる **GATEWAY** ディレクティブが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。

つまりデフォルトルートは GATEWAY ディレクティブで指定することが可能で、グローバルインターフェー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 ではグローバルの `/etc/sysconfig/network` ファイルの使用は非推奨となっており、ゲートウェイの指定はインターフェースごとの設定ファイルでのみ行ってください。

NetworkManager がモバイルホストを管理しているという動的な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ース固有である可能性が高く、**DHCP** による割り当てに任せるのが最善です。**NetworkManager** においてゲートウェイに達する出口インターフェースの選択に影響を及ぼす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進まないインターフェースに **ifcfg** ファイルの **DEFROUTE=no** コマンドを利用します。

2.2.5. 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** コマンドを使って設定した静的ルートは、システムが終了したり再起動すると失われます。システム再起動後も維持される静的ルートを設定するには、`/etc/sysconfig/network-scripts/` ディレクトリー内のインターフェースごとの設定ファイルに格納する必要があります。ファイル名は、**route-ifname** という形式にします。設定ファイル内で使用するコマンドには、「[IP コマンド引数形式を使用した静的ルート](#)」で説明する **ip** コマンドと、「[ネットワーク/ネットマスクディレクティブの形式](#)」で説明する *Network/Netmask* 形式という 2 つのタイプがあります。

2.2.5.1. IP コマンド引数形式を使用した静的ルート

インターフェースごとの設定ファイル、たとえば `/etc/sysconfig/network-scripts/route-eth0` で必要な場合は、1 行目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、かつ `/etc/sysconfig/network` ファイルでグローバルに設定されていない場合にのみ必要となります。

```
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** アドレスになります。*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ースになります。**dev** オプションは省略可能なオプションです。この設定は、`/etc/sysconfig/network` ファイル内の設定に優先す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プレフィックス長です。アドレス `192.168.1.1` は、リモートネットワークに続く **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、出口インターフェー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** オプションを使うと出口インターフェース *interface* を指定できますが、これは必須ではありません。必要に応じた数の静的ルートを追加してください。

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、インターフェースは `eth0`、また専用回線または WAN 接続が **192.168.0.10** で利用可能です。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 eth0
10.10.10.0/24 via 192.168.0.10 dev eth0
172.16.1.10/32 via 192.168.0.10 dev eth0
```

上記の例では、ローカルの **192.168.0.0/24** ネットワークに向かう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ースに移動します。**10.10.10.0/24** ネットワークおよび **172.16.1.10/32** ホストに向かうパケットは、**192.168.0.10** に移動します。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。

出口インターフェースの指定は、オプションです。特定のインターフェー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、tun0 インターフェースが宛先ネットワークにつながる別のサブネットにあったとしても、リモートネットワークへのトラフィックを強制的にこのインターフェース経由とさせることができます。

ip route の形式を使用して、送信元アドレスを指定することができます。以下に例を示します。

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

あるいは、同じ形式で、既存のポリシーベースのルート設定を定義することもできます。この場合、複数のルーティングテーブルを指定します。以下に例を示します。

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



重要

DHCP がすでにデフォルトゲートウェイを割り当てており、設定ファイル内でこの同一ゲートウェイが同一メトリクスで指定されている場合、起動時またはインターフェースをアクティベートする際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が、これは無視して構いません。

2.2.5.2. ネットワーク/ネットマスクディレクティブの形式

ネットワーク/ネットマスクディレクティブの形式を **route-interface** ファイルに使用することも可能です。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** への到達に使用可能なデフォルトゲートウェイまたは **IP** アドレスです。

以下は、ネットワーク/ネットマスクディレクティブの形式を使用した **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、専用回線または WAN 接続が

192.168.0.10 で利用可能です。2 つの静的ルートは、**10.10.10.0/24** および **172.16.1.0/24** の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** というようになります。

デフォルトでは、パケットをあるインターフェースから別のインターフェースに転送することや、同じインターフェースから外部に送信することは、セキュリティ上の理由から無効になっています。これにより、システムが外部トラフィックのルーターとして機能するのを防ぐことができます。システムに外部トラフィックをルーティングさせる必要がある場合は (たとえば、接続を共有する場合や、VPN サーバーを設定する場合)、IP フォワーディングを有効にする必要があります。詳細は、『Red Hat Enterprise Linux 7 Security Guide』の『[Enabling Packet Forwarding](#)』を参照してください。

2.2.6. VPN の設定

Red Hat Enterprise Linux 7 で VPN を作成する推奨方法は、**Libreswan** が提供する IPsec になります。コマンドラインを使って IPsec VPN を設定する方法は、『[Red Hat Enterprise Linux 7 セキュリティーガイド](#)』で説明されています。

2.3. GNOME グラフィカルユーザーインターフェースによる NETWORKMANAGER の使用

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ース (GUI) はありません。GNOME シェルの一部としてデスクトップ画面の右上にネットワーク接続アイコンが表示されており、ネットワーク設定ツールが新たな GNOME **control-center** GUI の一部として提供されています。**nm-connection-editor** GUI は、ボンドやチーミング接続の設定など、GNOME **control-center** では提供されない機能に対応しています。

2.3.1. 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、2 つの方法があります。

- **Super** キーを押してアクティビティ画面を開き、**Network** と入力して **ネットワーク** 設定ツールを選択します。『[新規接続の設定と既存接続の編集](#)』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。

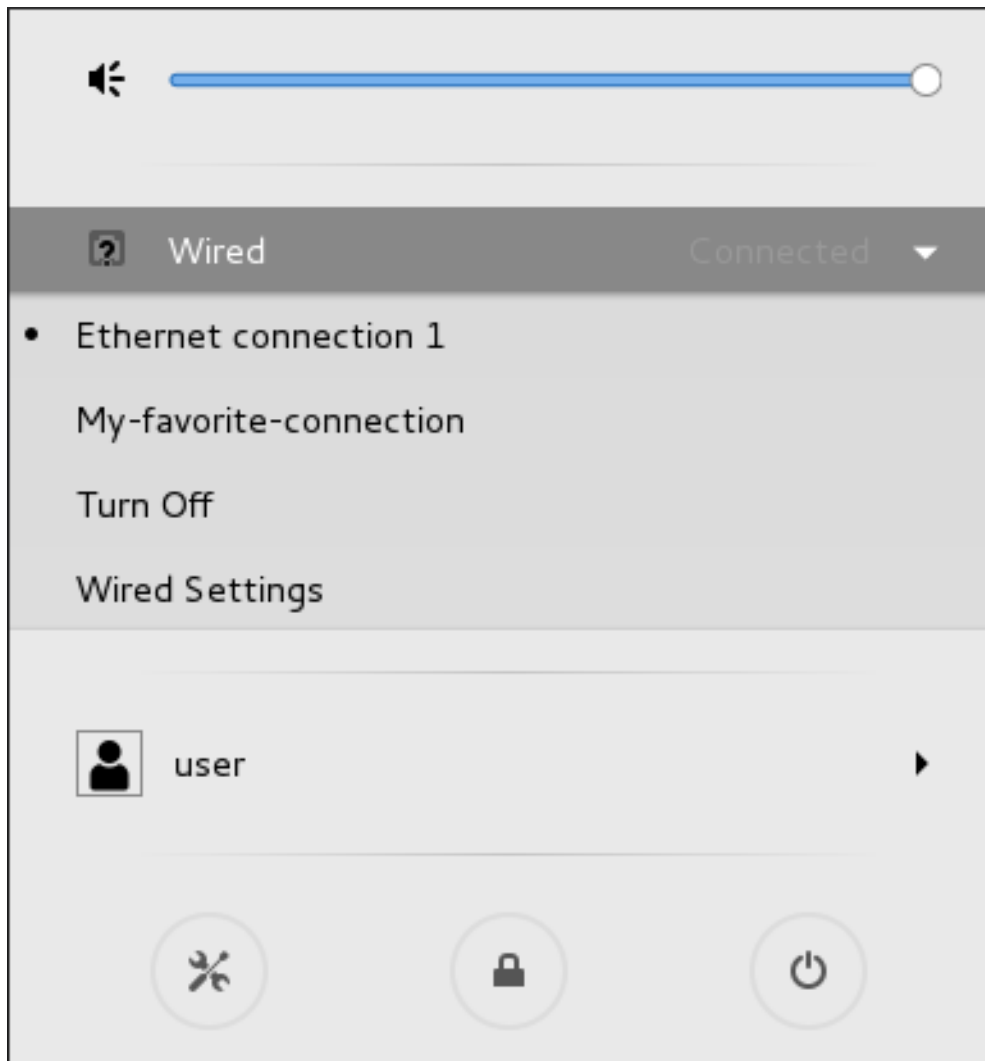


図2.5 GNOME でネットワークユーティリティを選択する

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別のネットワーク一覧 (有線 と **Wi-Fi** など)
- **NetworkManager** が検出した **Available Networks** (利用可能なネットワーク) の全一覧
- 設定済みの VPN (仮想プライベートネットワーク) への接続オプション
- **ネットワークの設定** メニューの選択オプション

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。

ネットワーク設定 をクリックすると、**ネットワーク 設定ツール**が表示されます。「[新規接続の設定と既存接続の編集](#)」に進みます。

2.3.2. 新規接続の設定と既存接続の編集

ネットワーク 設定ウィンドウでは、接続のステータス、種類およびインターフェース、**IP** アドレス、ルートの詳細などが表示されます。



図2.6 ネットワーク設定ウィンドウを使用したネットワークの設定

ネットワーク 設定ウィンドウの左側にはメニューがあり、利用可能なネットワークデバイスやインターフェースが表示されます。これには、VLAN やブリッジ、ボンド、チームなどのソフトウェアインターフェースが含まれます。右側には選択されたネットワークデバイスまたはインターフェースの **接続のプロファイル** が表示されます。プロファイルは、インターフェースに適用可能な設定の集合に名前をつけたものです。その下には新規ネットワーク接続を追加、削除するための+および-ボタンがあります。右側には丸いギアのアイコンがあり、これは選択されたネットワークデバイスまたは VPN 接続の詳細を編集するためのものです。新規接続を追加するには、+記号をクリックして **ネットワーク接続の追加** ウィンドウを開いて「**新規接続の設定**」に進みます。

既存接続の編集

ネットワーク 設定ウィンドウで既存接続のプロファイルのギアアイコンをクリックすると **ネットワーク** の詳細ウィンドウが開き、**IP** アドレスや **DNS**、ルート設定といったほとんどのネットワーク設定タスクが実行できます。



図2.7 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

接続の修正後に変更を適用するには、ボタンをクリックして オン から オフ に変更して非アクティブ化し、もう一度 オン に設定してデバイスを再度アクティブ化します。詳細は、「[新規接続の設定と既存接続の編集](#)」を参照してください。

2.3.2.1. 新規接続の設定

ネットワーク 設定ウィンドウでメニューの下にある+記号をクリックすると、ネットワーク接続の追加ウィンドウが開きます。ここでは、追加可能な接続の種類の一覧が表示されます。

以下のいずれかを選んで設定します。

- **VPN 接続** の場合は、**VPN** エントリーをクリックして「[VPN 接続の確立](#)」に進みます。
- **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「[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。
- **Team 接続** の場合は、**Team** エントリーをクリックして「[GUI を使ったネットワークチームの作成](#)」に進みます。

2.3.3. ネットワークへの自動接続

追加や設定を行うすべての接続で、ネットワークが利用可能な時に **NetworkManager** が自動的に接続を試行するかどうか選択することができます。

手順2.1 ネットワーク検出時に自動接続するよう **NetworkManager** を設定する

1. **Super** キーを押してアクティビティ画面を開き、**Network** と入力して ネットワーク 設定ツールを選択します。
2. 左側のメニューからネットワークインターフェースを選択します。
3. 右側の丸いギアのアイコンをクリックします。選択したインターフェースに関連付けられたプロファイルが1つの場合、ギアのアイコンは右下に表示されます。ネットワークの詳細ウィンドウが開きます。
4. 左側の **Identity** メニューエントリーを選択します。**Network** ウィンドウが identity ビューに切り替わります。
5. **自動接続する** を選択すると、その接続が利用可能であることを **NetworkManager** が検出すると、**NetworkManager** が自動接続するようになります。**NetworkManager** の自動接続を望まない場合は、チェックボックスを外します。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。

2.3.4. nm-connection-editor における共通の設定オプション

nm-connection-editor ユーティリティーを使用する場合、ほとんどの接続の種類 (イーサネット、Wi-Fi、モバイルブロードバンド、DSL) に共通な 5 つの設定オプションがあります。

1. ターミナルで **nm-connection-editor** と入力します。

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

2. **追加** ボタンをクリックします。**接続の種類を選択** ウィンドウが表示されます。



図2.8 接続の種類を選択

あるいは、既存の接続の種類の場合は、**ネットワーク接続** ダイアログで **編集** ボタンをクリックします。

3. **編集** ダイアログで **全般** タブを選択します。

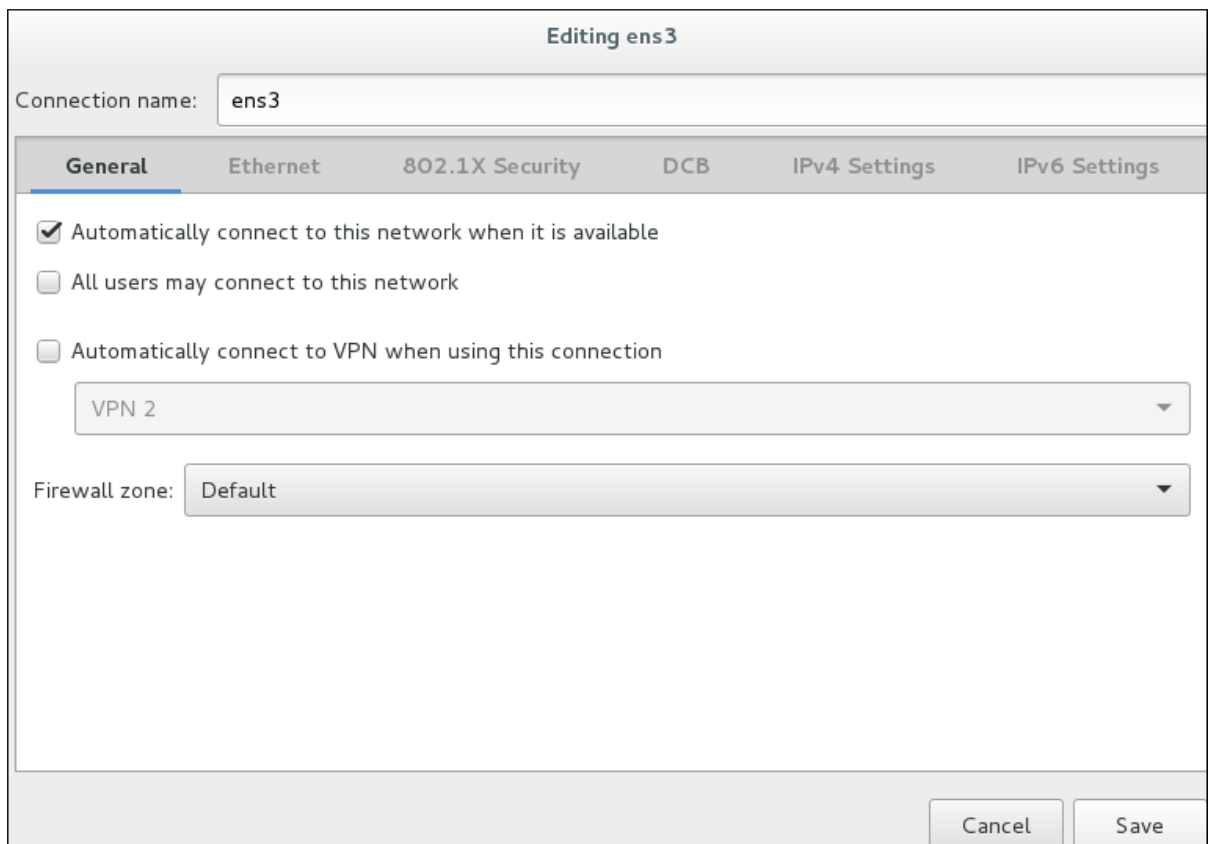


図2.9 nm-connection-editor における設定オプション

- **接続名:** ネットワーク接続の名前を入力します。この名前は **Network** ウィンドウメニューの接続名一覧に表示されます。

- **この接続が利用可能になったときは自動的に接続する:** このボックスにチェックを入れると、この接続が利用可能な時に **NetworkManager** が自動接続します。詳細については「[ネットワークへの自動接続](#)」を参照してください。
- **全ユーザーがこのネットワークに接続可能とする:** システム上のすべてのユーザーが接続可能とするには、このボックスにチェックを入れます。このセッティングを変更するには root 権限が必要になることがあります。詳細については「[システムワイドおよびプライベート接続プロファイル](#)」を参照してください。
- **この接続を使用したときは自動的に VPN に接続する:** このボックスにチェックを入れると、この接続が利用可能な時に **NetworkManager** が自動で選択された VPN に接続します。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 セキュリティガイド](#)』を参照してください。



注記

接続の種類が VPN の場合は、上記設定オプションのうち次に挙げる 3 つだけが利用可能です。接続名、全ユーザーがこのネットワークに接続可能とする、およびファイアウォールゾーン。

2.3.5. システムワイドおよびプライベート接続プロファイル

NetworkManager はすべての 接続プロファイル を保存します。プロファイルとは、インターフェースに適用可能な接続の集合に名前をつけたものです。**NetworkManager** はこれらの接続プロファイルをシステムワイド用 (システム接続) に、またすべての ユーザー接続 プロファイルも保存します。接続プロファイルへのアクセスはパーミッションで制御されており、これらは **NetworkManager** が保存しています。**connection** 設定の **permissions** プロパティに関する詳細情報は、**nm-settings(5)** man ページを参照してください。パーミッションは、**ifcfg** ファイル内の **USERS** ディレクティブに対応しています。**USERS** ディレクティブがない場合は、すべてのユーザーがそのネットワークプロファイルを使用できるようになります。たとえば、**ifcfg** ファイル内の以下のコマンドでは、記載されたユーザーのみがこの接続を使用できるようになります。

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

これはグラフィカルユーザーインターフェースツールを使って設定することもできます。**nm-connection-editor** では 全般 タブの 全ユーザーがこのネットワークに接続可能とする のチェックボックスがこれに相当し、GNOME **control-center** のネットワーク設定 Identity ウィンドウでは、他のユーザーにも利用可能にする がこれに相当します。

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ワイドの接続を作成、編集できます。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ので、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 **user** が接続プロファイル **user-em2** を作成し、**自動接続する** にチェックを入れて **他のユーザーにも利用可能にする** にチェックを入れないと、この接続は起動時に利用可能となりません。

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することができます。

- **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外します。これでこの接続は、この変更を行ったユーザーのみが編集、使用できるようになります。

- **polkit** フレームワークを使用して、ユーザーごとに全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と制御が可能になります。**polkit** に関する詳細情報は、**polkit(8)** 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。

手順2.2 control-center を使用してシステムワイドの接続からユーザー固有の接続に変更する、またはその逆

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ワイドに変更するために、システムで root 権限が必要となる場合があります。

1. **Super** キーを押してアクティビティ画面を開き、**Network** と入力して **ネットワーク 設定ツール** を選択します。
2. 左側のメニューからネットワークインターフェースを選択します。
3. 右側の丸いギアのアイコンをクリックします。選択したインターフェースに関連付けられたプロファイルが1つの場合、ギアのアイコンは右下に表示されます。**ネットワーク** の詳細ウィンドウが開きます。
4. 左側の **Identity** メニューエントリーを選択します。**Network** ウィンドウが identity ビューに切り替わります。
5. **他のユーザーにも利用可能にする** にチェックを入れると、**NetworkManager** はこの接続をシステムワイドで利用可能にします。

逆に **他のユーザーにも利用可能にする** のチェックを外すと、この接続はユーザー固有になります。

2.3.6. control-center を使用した有線 (イーサネット) 接続の設定

有線ネットワーク接続を設定するには、**Super** キーを押してアクティビティ画面を開き、**Network** と入力して **ネットワーク 設定ツール** を選択します。

左側のメニューからネットワークインターフェース **有線** を選択します。

システムはデフォルトで、**有線** という名前の単一の有線 **接続プロファイル** を作成、設定します。プロファイルとは、インターフェースに適用可能な設定の集合に名前をつけたものです。ひとつのインターフェースにつき複数のプロファイルを作成し、必要に応じてこれを適用することができます。デフォルトのプロファイルを削除することはできませんが、この設定を変更することはできます。デフォルトの**有線** プロファイルを編集するには、丸いギアのアイコンをクリックします。**プロファイルの追加** ボタンをクリックすると、新規の有線接続プロファイルを作成することができます。選択されたインターフェースに関連付けられている接続プロファイルは、右側のメニューに表示されます。

プロファイルの追加 ボタンをクリックして新規接続を追加すると、**NetworkManager** はその接続のための新しい設定ファイルを作成して、既存の接続の編集に使用されるものと同じダイアログを開きます。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は **詳細** と **リセット** メニューエントリーがあるという点です。結局は常に接続プロファイルを編集することになります。その接続が以前から存在していたか、それとも **プロファイルの追加** をクリックして **NetworkManager** が作成したものかという点のみが異なります。

基本設定オプション

有線 ダイアログにおいて、左側の **Identity** メニューエントリーを選択すると、以下のような設定画面が表示されます。

図2.10 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名前を入力します。この名前は **Network** ウィンドウメニューの接続名一覧に表示されます。
- **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ースの MAC アドレスを選択します。
- **クローンしたアドレス:** 必要な場合には、別の MAC アドレスを入力します。
- **MTU:** 必要に応じて、使用する特定の **最大転送単位 (MTU)** を入力します。MTU の値は、リンク層が転送する最大パケットサイズをバイト数で表したものです。この値のデフォルトは **1500** で、通常は指定や変更の必要はありません。
- **ファイアウォールゾーン:** 必要な場合は、適用する別の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 セキュリティガイド](#)』を参照してください。
- **自動接続する:** このボックスにチェックを入れると、この接続が利用可能な時に **NetworkManager** が自動接続します。詳細については「[ネットワークへの自動接続](#)」を参照してください。
- **他のユーザーにも利用可能にする:** 接続をこのシステム上のすべてのユーザーに利用可能とするには、このボックスにチェックを入れます。このセッティングを変更するには root 権限が必要

になることがあります。詳細については「[システムワイドおよびプライベート接続プロファイル](#)」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

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。編集集中に該当プロファイルが使用されていた場合、接続を再起動し、**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、「[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** ウィンドウ内でその接続を選択し、ギアのアイコンをクリックして編集ダイアログに戻ります。

以下のいずれかを選んで設定します。

- **ポートベースネットワークアクセスコントロール (PNAC)** を設定するには、**802.1X セキュリティ** タブをクリックして「[802.1X セキュリティの設定](#)」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 セッティングの設定](#)」に進みます。
- **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。

2.3.7. control-center を使用した Wi-Fi 接続の設定

このセクションでは、**NetworkManager** を使ってアクセスポイントに Wi-Fi (ワイヤレス または 802.11a/b/g/n と呼ばれる) 接続する設定方法を説明します。

(3G などの) モバイルブロードバンドの設定方法については、「[モバイルブロードバンド接続の確立](#)」を参照してください。

利用可能なアクセスポイントへのすばやい接続

利用可能なアクセスポイントに接続する最も簡単な方法は、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開き、**Wi-Fi** ネットワーク一覧内のアクセスポイントの **サービスセット識別子 (SSID)** を見つけてそれをクリックすることです。鍵の記号は、そのアクセスポイントで認識が必要であることを示しています。アクセスポイントが安全な場合は、認識キーまたはパスワード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われているセキュリティタイプの自動検出を試みます。複数の可能性がある場合、**NetworkManager** はセキュリティタイプを予想し、それを **Wi-Fi セキュリティ** ドロップダウンメニューに表示します。WPA-PSK セキュリティ (パスフレーズをとともなう WPA) の場合、選択は不要になります。WPA Enterprise (802.1X) は自動検出ができないので、セキュリティを明確に選択する必要があります。不明な場合は、それぞれのタイプに接続してみます。最後に **パスワード** フィールドにキーまたはパスフレーズを記入します。40-bit WEP や 128-bit WPA キーなどの特定のパスワードのタイプは、指定の長さに達していないと無効になります。選択したセキュリティタイプで要求される長さのキーを記入するまでは、**接続** ボタンは灰色のままです。無線セキュリティに関する情報については、「[Wi-Fi セキュリティの設定](#)」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無線接続信号の強度を示すグラフィカルインジケータに変わります。

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。**ネットワーク** ウィンドウの **Wi-Fi** ページには **履歴** 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。「[接続の編集、または新規接続の作成](#)」を参照してください。

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。しかし、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されている場合もあります。その際には **非表示** となり、**NetworkManager** 内の **使用可能** ネットワーク一覧に表示されないことになります。ただし、その SSID と認証方法と秘密情報が分かっている場合は、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、**Super** キーを押してアクティビティ画面を開き、**Network** と入力してネットワーク設定ツールを選択します。ネットワーク ウィンドウが表示されます。メニューから **Wi-Fi** を選択し、**非表示のネットワークに接続** を選ぶとダイアログが表示されます。これまでに非表示のネットワークに接続したことがある場合は **接続** ドロップダウンメニューを使って選択し、**接続** をクリックします。接続したことがない場合は、**接続** ドロップダウンメニューを **新規...** のままにして、非表示ネットワークの SSID を入力し、**Wi-Fi セキュリティー** を選択して、適切な認証情報を入力した後、**接続** をクリックします。

ワイヤレスセキュリティの設定に関する詳細情報については、「[Wi-Fi セキュリティーの設定](#)」を参照してください。

接続の編集、または新規接続の作成

これまでに接続を試行したことのある、または接続に成功したことのある既存の接続を編集するには、ネットワーク ダイアログの **Wi-Fi** ページを開いて、Wi-Fi 接続名の右側にあるギアアイコンを選択します。該当するネットワークが現在範囲にない場合は、**履歴** をクリックすると過去の接続が表示されます。ギアアイコンをクリックすると、編集する接続ダイアログが表示されます。**詳細** ウィンドウに接続詳細が表示されます。

SSID が範囲内にある新規接続を設定するには、まず ネットワーク ウィンドウを開いて **Wi-Fi** メニューエントリーを選択し、接続名 (デフォルトでは SSID と同じ) をクリックして、接続を試みます。SSID が範囲内にない場合の詳細については、「[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の手順は、以下のとおりです。

1. **Super** キーを押してアクティビティ画面を開き、**Network** と入力して **ネットワーク** 設定ツールを選択します。
2. 左側のメニューエントリーから **Wi-Fi** インターフェースを選択します。
3. 右側のメニューで接続する Wi-Fi 接続プロファイルをクリックします。鍵の記号は、キーまたはパスワードが必要であることを示しています。
4. 必要に応じて認証情報を入力します。

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、**ネットワーク** ページで **Wi-Fi** を選択し、その上で Wi-Fi 名の右側にあるギアアイコンをクリックします。**Identity** を選択すると、以下の設定が可能になります。

The screenshot shows a window titled "my wifi" with a close button (X) in the top right corner. On the left is a sidebar menu with the following items: "Details", "Security", "Identity" (which is highlighted with a blue background), "IPv4", "IPv6", and "Reset". The main area of the window contains the following fields and options:

- SSID**: A text input field containing the text "my wifi".
- BSSID**: A dropdown menu that is currently empty.
- MAC Address**: A dropdown menu containing the text "F0:D5:BF:65:C6:89".
- Cloned Address**: An empty text input field.
- At the bottom of the main area, there are two checked checkboxes:
 - ☒ Connect automatically
 - ☒ Make available to other users
- At the bottom right, there are two buttons: "Cancel" and "Apply".

図2.11 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 (基本的サービスセット識別子) (BSSID) は、インフラストラクチャー モードで接続している際の特定のワイヤレスアクセスポイントの MAC アドレスで、ハードウェアアドレスとも呼ばれます。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** でワイヤレスアクセスポイントに接続できます。BSSID が指定されると、システムは強制的に特定のアクセスポイントのみに関連付けられます。*

アドホックネットワークが生成される際にこのネットワーク用に **mac80211** サブシステムがランダムに **BSSID** を生成します。これは **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。そのため、**MAC アドレス** フィールドで、特定のワイヤレスアダプターと特定の接続 (単一または複数) の関連付けを可能にしています。

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。必要なければ、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。

- **自動接続する:** このボックスにチェックを入れると、この接続が利用可能な時に **NetworkManager** が自動接続します。詳細については「[ネットワークへの自動接続](#)」を参照してください。
- **他のユーザーにも利用可能にする:** 接続をこのシステム上のすべてのユーザーに利用可能とするには、このボックスにチェックを入れます。このセッティングを変更するには root 権限が必要になることがあります。詳細については「[システムワイドおよびプライベート接続プロファイル](#)」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

ワイヤレス接続の編集が終了したら、**適用** ボタンを押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、「[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** ウィンドウ内でその接続を選択し、ギアのアイコンをクリックして接続の詳細を表示します。

以下のいずれかを選んで設定します。

- ワイヤレス接続の **セキュリティ認証** を設定するには、**セキュリティ** をクリックします。「[Wi-Fi セキュリティの設定](#)」に進みます。
- 接続の **IPv4** を設定するには、**IPv4** をクリックします。「[IPv4 セッティングの設定](#)」に進みます。
- 接続の **IPv6** を設定するには、**IPv6** をクリックします。「[IPv6 セッティングの設定](#)」に進みます。

2.4. VPN 接続の確立

Red Hat Enterprise Linux 7 で VPN を作成する推奨方法は、**Libreswan** が提供する IPsec になります。以下で説明する GNOME グラフィカルユーザーインターフェースツールの使用には、**NetworkManager-libreswan-gnome** パッケージが必要になります。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 7 で新規パッケージをインストールする方法については、『[Red Hat Enterprise Linux 7 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仲介ネットワークにトンネルを設定することで実施されます。セットアップされる VPN トンネルは通常、認証および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構成します。

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て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。

手順2.3 control-center を使用して新規 VPN 接続を追加する

新規 VPN 接続を設定するには、**ネットワーク** ウィンドウを開いて、メニューの下にある+ボタンを選択します。

1. **Super** キーを押してアクティビティ画面を開き、**Network** と入力して **ネットワーク** 設定ツールを選択します。
2. ウィンドウの下部にある+ボタンをクリックします。

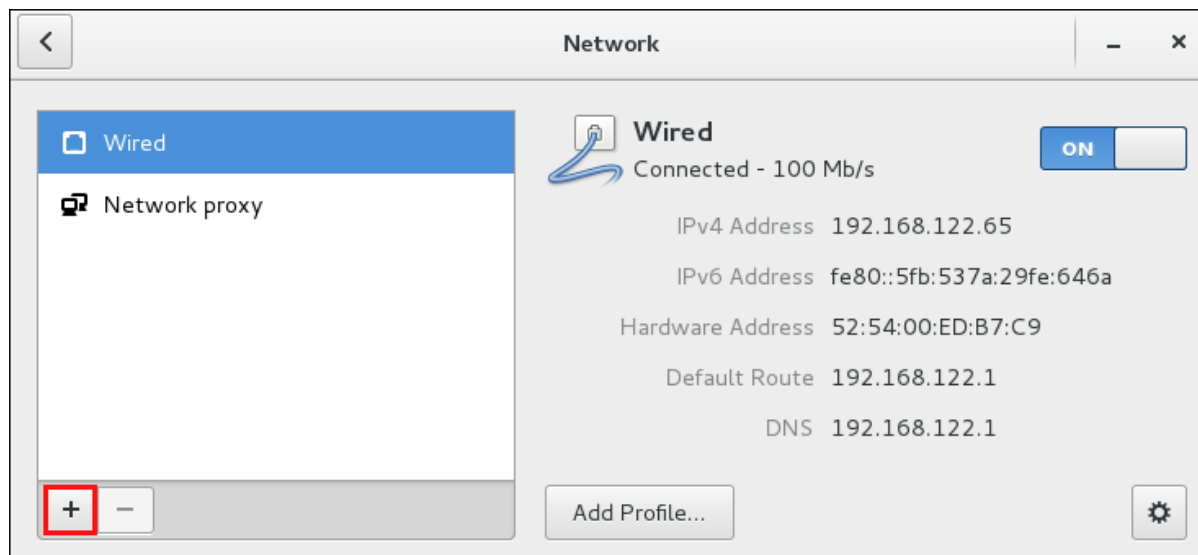


図2.12 接続の追加

3. 手動で設定するには、**IPsec** ベースの **VPN** を選択します。

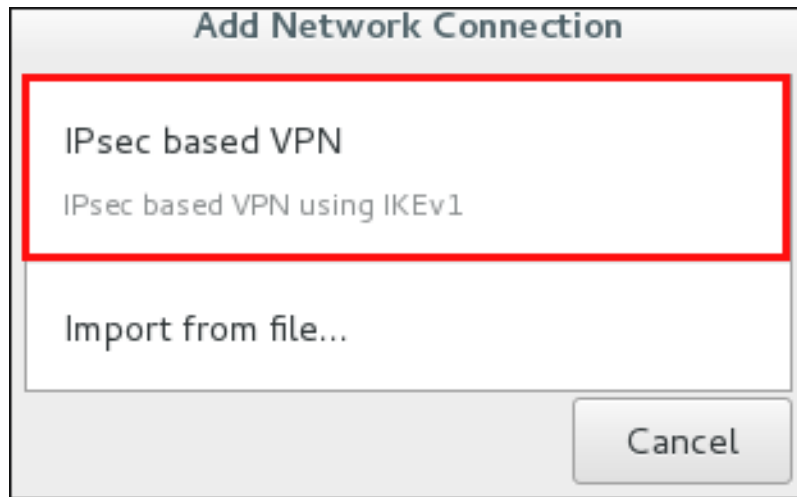


図2.13 IPsec モードの VPN の設定

4. **Identity** 設定フォームで、**全般** および **高度** セクションのフィールドを指定することができます。

Add Network Connection

Identity

- IPv4
- IPv6

Name: VPN 2

Firewall Zone: Default

☐ Make available to other users

General

Gateway:

User name:

User password:

Group name:

Secret:

☐ Show passwords

Advanced

Phase1 Algorithms:

Phase2 Algorithms:

Domain:

Cancel Add

図2.14 全般 および 高度 セクション

- **全般** セクションでは、以下のフィールドを指定することができます。

ゲートウェイ

リモート VPN ゲートウェイの名前もしくは **IP** アドレスです。

ユーザー名

必要の場合は、認証のために VPN ユーザーに関連付けられたユーザー名を入力します。

ユーザーパスワード

必要の場合は、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。

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ョンモードではなく 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。

- **高度** 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 ネゴシエーションで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。

手順2.4 既存の VPN 接続を編集する

既存の VPN 接続を設定するには、**ネットワーク** ウィンドウを開いて、リストにある接続名を選択します。

1. **Super** キーを押してアクティビティ画面を開き、**Network** と入力して **ネットワーク** 設定ツールを選択します。
2. 左側のメニューから編集する **VPN** 接続を選択します。
3. **編集** ボタンをクリックします。

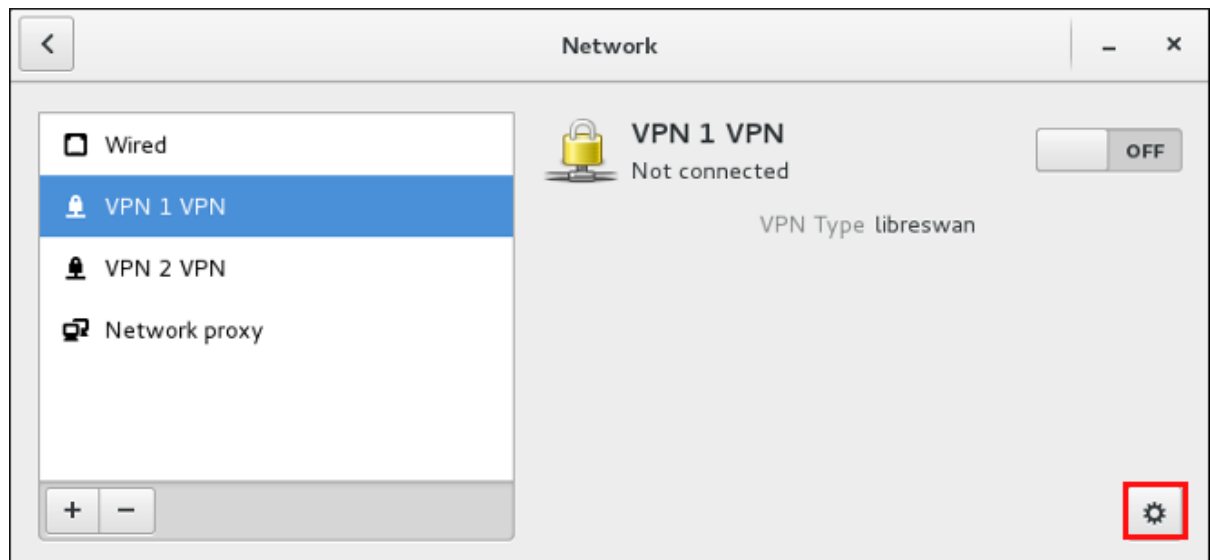


図2.15 既存 VPN 接続の編集

4. 左側の **Identity** メニューエントリーを選択し、**全般** セクションのフィールドを指定します。

VPN 1 VPN

Details
Identity
IPv4
IPv6
Reset

Name: VPN 1

Firewall Zone: Default

☐ Make available to other users

General

Gateway: ovpn.example.com

User name: jsmith

User password: [masked]

Group name: group_name

Secret: [masked]

☐ Show passwords

▶ Advanced

Cancel Apply

図2.16 既存 VPN 接続の設定

新規 (または修正した) 接続を保存して他の設定を行う

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。編集中に該当プロファイルが使用されていた場合、接続を切断してから再接続し、**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、「[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をするには、**ネットワーク接続** ウィンドウ内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。

以下のいずれかを選んで設定します。

- **IPv4** の設定は、**IPv4 のセッティング** タブをクリックして「[IPv4 セッティングの設定](#)」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。

2.5. モバイルブロードバンド接続の確立

NetworkManager のモバイルブロードバンド接続機能を使用すると、以下の 2G と 3G のサービスに接続することができます。

- 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、 dongle、モデムとして機能する携帯電話などです。

手順2.5 nm-connection-editor を使用して新規のモバイルブロードバンド接続を追加する

1. ターミナルで **nm-connection-editor** と入力します。

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

2. **追加** ボタンをクリックして **接続の種類を選択** メニューを開きます。
3. **モバイルブロードバンド** メニューエントリーを選びます。
4. **作成** をクリックして、**モバイルブロードバンド接続のセットアップ** を開きます。
5. **このモバイルブロードバンドデバイス用の接続を作成** の下で、その接続で使用する 2G または 3G に対応した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。この状況では、**キャンセル** をクリックして、モバイルブロードバンドの機能を持ったデバイスが接続されており、それがコンピューターに認識されていることを確認してください。それからこの手順を再試行します。**進む** ボタンをクリックします。
6. 使用するサービスプロバイダーの国をリストから選んで **進む** ボタンをクリックします。
7. プロバイダーをリストから選ぶか手動で入力し、**進む** 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。**進む** 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。

手順2.6 既存のモバイルブロードバンド接続を編集する

既存のモバイルブロードバンド接続を編集するには以下の手順に従います。

1. ターミナルで **nm-connection-editor** と入力します。

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

2. 編集する接続を選択して、**編集** ボタンをクリックします。詳細は、[「nm-connection-editor における共通の設定オプション」](#)を参照してください。
3. [「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。

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、**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。

以下のいずれかを選んで設定します。

- **ポイントツーポイント** を設定するには、**PPP の設定** 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 セッティングの設定」](#)に進みます。
- **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。

モバイルブロードバンドタブの設定

アシスタント (使用法は [手順2.5「nm-connection-editor を使用して新規のモバイルブロードバンド接続を追加する」](#)を参照) を使用して新規の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は、**モバイルブロードバンド** タブを編集して、ホームネットワークが使用不可能な場合はローミングを無効にしたり、ネットワーク ID を割り当てたり、接続使用時に特定の技術 (3G や 2G など) を優先するように **NetworkManager** に指示したりすることができます。

番号

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ヤルする番号です。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で残し、代わりに **APN** を記入します。

ユーザー名

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。

パスワード

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 を記入すると、**NetworkManager** は強制的にデバイスが特定のネットワークのみに登録されるようにします。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。

種別

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

3G (UMTS/HSPA): 接続が 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): 接続が 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): 最初に HSPA または UMTS などの 3G 技術を使用した接続を試み、失敗した後にのみ GPRS または EDGE にフォールバックします。

Prefer 2G (GPRS/EDGE): 最初に GPRS または EDGE などの 2G 技術を使用した接続を試み、失敗した後にのみ 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への移行ではなく、**NetworkManager** が接続を終了するようにするには、このボックスからチェックを外します。これにより、ローミング料金を回避できます。ボックスにチェックが入っていると、**NetworkManager** はホームネットワークからローミングに、またはその逆に切り替えることで接続を維持しようとします。

PIN 番号

デバイスの *SIM (Subscriber Identity Module (購読者識別モジュール))* が *PIN (Personal Identification Number (個人識別番号))* でロックされている場合は、その 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。どんな目的でもデバイスの使用に PIN を必要とする場合は、**NetworkManager** は SIM をロック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**、および **Type** にはオプションがありません。

2.6. DSL 接続の確立

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。

手順2.7 nm-connection-editor を使用して新規 DSL 接続を追加する

1. ターミナルで **nm-connection-editor** と入力します。

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

2. **追加** ボタンをクリックします。
3. **接続の種類を選択** が表示されます。
4. **DSL** を選択し、**作成** ボタンをクリックします。
5. **DSL 接続 1 の編集** ウィンドウが表示されます。

手順2.8 既存の DSL 接続を編集する

1. ターミナルで **nm-connection-editor** と入力します。

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

2. 編集する接続を選択して、**編集** ボタンをクリックします。詳細は、[「nm-connection-editor における共通の設定オプション」](#)を参照してください。

DSL タブの設定

ユーザー名

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。

パスワード

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、**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。

以下のいずれかを選んで設定します。

- **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 の設定** 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 セッティングの設定」](#)に進みます。

2.7. 接続セッティングの設定

このセクションでは、802.3 リンクセッティングのさまざまな設定、および **NetworkManager** を使ったその設定方法について説明します。

2.7.1. 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 リンクセッティングを設定することができます。

- **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することができます。

- 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
- **speed** および **duplex** リンクセッティングを手動で設定する

リンクネゴシエーションを無視する

このモードでは、**NetworkManager** はイーサネット接続のリンク設定を無視し、デバイスの既存設定を維持します。

リンクネゴシエーションを無視するには、以下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ければ、オートネゴシエーションは無効になりません。

オートネゴシエーションを強制的にアクティブ化する

このモードでは、**NetworkManager** はデバイスのオートネゴシエーションを強制的にアクティブ化します。

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

このモードでは、リンクの **speed** および **duplex** のセッティングを手動で設定することができます。

speed および **duplex** リンクセッティングを手動で設定するには、上述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** および **duplex** 両方の値を設定してください。そうしないと、**NetworkManager** はリンク設定を更新しません。

802.3 リンクセッティングを設定するには、以下のツールを使用することができます。

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

手順2.9 nmcli ツールを使用して 802.3 リンクセッティングを設定する

1. `eth0` 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、[「802.3 リンクセッティングの設定」](#)を参照してください。

たとえば、**speed** オプションを *100 Mbit/s* に、**duplex** オプションを *full* に、それぞれ手動で設定するには、以下のコマンドを実行します。

```
nmcli connection add con-name MyEthernet type ethernet ifname
eth0 \
  802-3-ethernet.auto-negotiate no \
  802-3-ethernet.speed 100 \
  802-3-ethernet.duplex full
```

手順2.10 nm-connection-editor を使用して 802.3 リンクセッティングを設定する

1. ターミナルで **nm-connection-editor** と入力します。

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

2. 編集するイーサネット接続を選択して、**編集** をクリックします。詳細は、[「nm-connection-editor における共通の設定オプション」](#)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。
 - **無視する**: リンク設定はスキップされます (デフォルト)。
 - **自動**: デバイスのリンクオートネゴシエーションを強制的にアクティブ化します。
 - **手動**: **Speed** 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的にアクティブ化することができます。

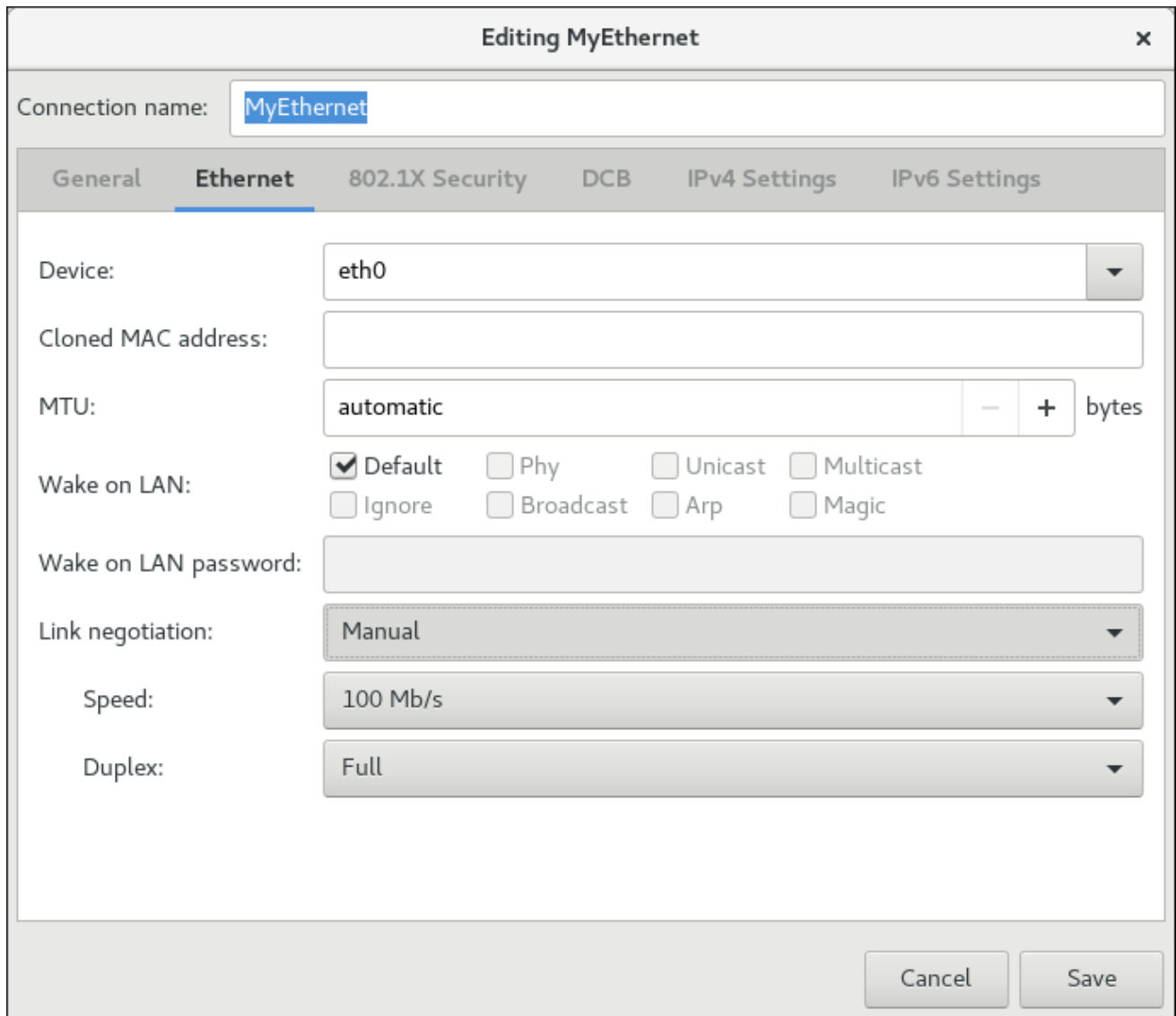


図2.17 nm-connection-editor を使用した 802.3 リンクセッティングの設定

2.7.2. 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。簡単に言うと、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 セキュリティーは通常、無線ネットワーク (WLAN) の保護に関連付けられますが、ネットワーク (LAN) への物理アクセスがある侵入者を阻止するためにも使用できます。以前は、DHCP サーバーは認証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、多くの理由でこれは非現実的で安全ではないため、今ではもう推奨されません。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介してネットワークを論理的に安全にしています。

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の達成方法を定義するプロトコルです。

GUI を使用した接続セッティング 802.1X セキュリティーの設定

有線もしくはワイヤレス接続に 802.1X セキュリティーを設定するには、ネットワーク ウィンドウを開き (「GUI を使用したネットワーク接続」を参照) 該当する以下の手順に従います。Super キーを押し

てアクティビティ画面を開き、**Network** と入力して **ネットワーク** 設定ツールを選択します。手順 2.11「有線接続の場合」または 手順2.12「ワイヤレス接続の場合」に進みます。

手順2.11 有線接続の場合

1. 左側のメニューから **有線** ネットワークインターフェースを選択します。
2. **プロファイルの追加** ボタンをクリックして 802.1X セキュリティーを設定する新規のネットワーク接続を追加するか、既存の接続を選択してギアのアイコンをクリックします。
3. **セキュリティ** を選択してからボタンを **オン** にして設定を有効にします。
4. 「**トランスポートレイヤーセキュリティ (TLS) セッティングの設定**」に進みます。

手順2.12 ワイヤレス接続の場合

1. 左側のメニューから **Wi-Fi** ネットワークインターフェースを選択します。必要に応じてボタンを **オン** にして、ハードウェアのスイッチがオンになっていることをチェックします。
2. 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。
3. **セキュリティ** を選択します。
4. ドロップダウンメニューから、**LEAP**、**動的 WEP (802.1X)**、**WPA & WPA2 Enterprise** いずれかのセキュリティ方法を選択します。
5. **セキュリティ** ドロップダウンメニューの選択肢がどの *extensible authentication protocol* (EAP) タイプに対応するかについての説明は、「**トランスポートレイヤーセキュリティ (TLS) セッティングの設定**」を参照してください。

nmcli ツールを使用した接続セッティング 802.1X セキュリティーの設定

nmcli ツールを使用して ワイヤレス 接続を設定するには、以下の手順を実施します。

1. 認証済みの **key-mgmt** (キーマネジメント) プロトコルを設定します。このプロトコルにより、セキュアな **Wi-Fi** 接続のための鍵のメカニズムが設定されます。プロパティの詳細については、『nm-settings(5)』 man ページを参照してください。
2. 802-1x 認証セッティングを設定します。トランスポートレイヤーセキュリティ (TLS) については、「**トランスポートレイヤーセキュリティ (TLS) セッティングの設定**」を参照してください。適切なプロパティに関する説明については、「**TLS の設定**」を参照してください。

表2.1 802-1x 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 認証設定	名前
802-1x.private-key	プライベートキー
802-1x.private-key-password	プライベートキーのパスワード

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 wlan0 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
```



注記

nmcli ツールを使用して **有線** 接続を設定するには、**802-11-wireless.ssid** および **802-11-wireless-security.key-mgmt** 設定を除き、**ワイヤレス** 接続と同じ手順を実施します。

2.7.2.1. トランスポートレイヤーセキュリティ (TLS) セットアップの設定

TLS (トランスポートレイヤーセキュリティ) では、クライアントとサーバーは TLS プロトコルを使用して相互に認証します。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 セキュリティを使用すると、証明書を管理するための PKI (パブリックキーインフラストラクチャー) のオーバーヘッドが必要になります。TLS セキュリティを使用する利点は、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、対応する 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集め、手順を処理するデーモンである **wpa_supplicant** にこれらを渡します。このデーモンは、OpenSSL を使って 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。このバージョンには、両方の末端でサポートされる一番高いバージョンが使用されます。

認証方法を選択する

以下のいずれかの認証方法を選択します。

- トランスポートレイヤーセキュリティを選択するには **TLS** を選択して「[TLS の設定](#)」に進みます。

- *Flexible Authentication via Secure Tunneling* を選択するには **FAST** を選択して「[FAST の設定](#)」に進みます。
- *Tunneled Transport Layer Security* (TTLS または EAP-TTLS と呼ぶ) を設定するには **トンネル化 TLS** を選択して「[トンネル化 TLS の設定](#)」に進みます。
- *Protected Extensible Authentication Protocol* を設定するには **保護つき EAP (PEAP)** を選択して「[保護つき EAP \(PEAP\) の設定](#)」に進みます。

2.7.2.2. TLS の設定

アイデンティティ

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ザして選択します。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ザして選択します。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。

プライベートキー

プライベートキーをブラウザして選択します。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。

プライベートキーのパスワード

プライベートキー フィールドのプライベートキー用のパスワードを入力します。パスワードを表示を選択すると、入力時にパスワードが表示されます。

2.7.2.3. FAST の設定

匿名の識別子

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、**匿名**、**認証**、**両方** のいずれかを選択します。

PAC ファイル

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

内部認証

GTC: Generic Token Card

MSCHAPv2: Microsoft チャレンジハンドシェイク認証プロトコルバージョン 2

ユーザー名

認証プロセスで使用するユーザー名を入力します。

パスワード

認証プロセスで使用するパスワードを入力します。

2.7.2.4. トンネル化 TLS の設定

匿名の識別子

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。

内部認証

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 チャレンジハンドシェイク認証プロトコルバージョン 2

CHAP: チャレンジ ハンドシェイク認証プロトコル

ユーザー名

認証プロセスで使用するユーザー名を入力します。

パスワード

認証プロセスで使用するパスワードを入力します。

2.7.2.5. 保護つき EAP (PEAP) の設定

匿名の識別子

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 バージョン

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

内部認証

MSCHAPv2: Microsoft チャレンジハンドシェイク認証プロトコルバージョン 2

MD5: メッセージダイジェスト 5、暗号化ハッシュ関数。

GTC: Generic Token Card

ユーザー名

認証プロセスで使用するユーザー名を入力します。

パスワード

認証プロセスで使用するパスワードを入力します。

2.7.3. Wi-Fi セキュリティーの設定

セキュリティ

なし: Wi-Fi 接続を暗号化しません。

WEP 40/128-bit キー: IEEE 802.11 標準からの Wired Equivalent Privacy (WEP)。単一の事前共有キー (PSK) を使用。

WEP 128-bit パスフレーズ: パスフレーズの MD5 ハッシュを使用して WEP キーを引き出します。

LEAP: Cisco Systems の Lightweight Extensible Authentication Protocol。

動的 WEP (802.1X): WEP キーが動的に変更されます。[「トランスポートレイヤーセキュリティ \(TLS\) セッティングの設定」](#) で使用します。

WPA & WPA2 Personal: IEEE 802.11i 標準からの Wi-Fi Protected Access (WPA)。WEP に代わるもの。802.11i-2004 標準からの Wi-Fi Protected Access II (WPA2)。パーソナルモードは、事前共有キー (WPA-PSK) を使用。

WPA & WPA2 Enterprise: RADUIS 認証サーバーで使用する WPA で、IEEE 802.1X ネットワークアクセス制御を提供します。[「トランスポートレイヤーセキュリティ \(TLS\) セッティングの設定」](#) で使用します。

パスワード

認証プロセスで使用するパスワードを入力します。

2.7.4. wpa_supplicant および NetworkManager による MACsec の使用

Media Access Control Security (MACsec) または IEEE 802.1AE) は、LAN にある全トラフィックを 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、アドレス解決プロトコル (ARP)、近隣検索 (ND)、または **DHCP** も保護することができます。**IPsec** はネットワーク層 (レイヤー 3) で機能し、**SSL** または **TLS** はトランスポート層 (レイヤー 4) で機能しますが、**MACsec** はデータリンク層 (レイヤー 2) で機能します。**MACsec** を他のネットワーキング層のセキュリティプロトコルと組み合わせて、それらの標準が提供する異なるセキュリティ機能を活用してください。

事前に共有された接続アソシエーションキー/CAK 名 (CAK/CKN) ペアを使用して認証を行うスイッチの **MACsec** を有効にするには、以下の手順を実施します。

1. CAK/CKN ペアを作成します。以下のコマンドにより、16 バイトのキーが 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、以下の行を追加します。


```
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
}
```

前の手順で作成した値を使用して、**wpa_supplicant.conf** 設定ファイルの **mka_cak** および **mka_ckn** 行を完成させます。

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

3. **eth0** を使っ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** を開始します。

```
~]# wpa_supplicant -i eth0 -Dmacsec_linux -c wpa_supplicant.conf
```

Red Hat では、**wpa_supplicant.conf** ファイルを作成して編集するのではなく、**nmcli** コマンドを使用して前の手順と同じように **wpa_supplicant** を設定することを推奨します。以下の例は、すでに 16 バイトの 16 進数 CAK (**\$MKA_CAK**) および 32 バイトの 16 進数 CKN (**\$MKA_CKN**) が作成されてい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 eth0 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。また、**MACsec** ネットワークのアーキテクチャー、ユースケースのシナリオ、設定例に関する詳しい情報は、「[MACsec: a different solution to encrypt network traffic](#)」の記事を参照してください。

2.7.5. PPP (ポイントツーポイント) セッティングの設定

認証方法

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

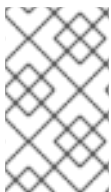
PPP Deflate プロトコル (『RFC 1979』)。

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 ではオプションとして PPP がサポートされるので、PPP セットアップを設定するには、事前に NetworkManager-ppp パッケージがインストールされている必要があります。

2.7.6. IPv4 セットアップの設定

IPv4 のセットアップ タブでは、ネットワークへの接続に使用する方式を設定することや、必要に応じて **IP** アドレス、ルートおよび **DNS** 情報を入力することができます。**IPv4 のセットアップ** タブは、次に挙げる接続の種類のいずれかを作成して修正する際に利用可能になります。有線、ワイヤレス、モバイルブロードバンド、VPN、DSL。**IPv6** アドレスを設定する必要がある場合は、「[IPv6 セットアップの設定](#)」を参照してください。静的ルートを設定する必要がある場合は、**ルート** ボタンをクリックして「[ルートの設定](#)」に進みます。

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、**方法** を **自動 (DHCP)** に設定するだけです。

方式の設定

接続の種類別で利用可能な IPv4 方式

方式 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種類によって以下の **IPv4** 接続方式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一覧表示されています。

方式

自動 (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** フィールドの記入は必要ありません。

自動 (DHCP) アドレス専用: 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。

ローカルへのリンク専用: 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たくない場合に、このオプションを選択します。『RFC 3927』に従って、接頭辞 **169.254/16** の付いたランダムなアドレスが割り当てられます。

他のコンピューターへ共有: 設定中のインターフェースがインターネットまたは WAN 接続の共有用である場合は、このオプションを選択します。インターフェースには **10.42.x.1/24** の範囲内のアドレスが割り当てられ、**DHCP** 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) で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ースが接続されます。

無効になっています: **IPv4** はこの接続で無効になります。

有線、ワイヤレス、DSL 接続の方式

手動: **IP** アドレス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。

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** サーバーを自動で割り当てる場合は、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** アドレスと **DNS** サーバーを自動で割り当てるものの、**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。

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** サーバーを自動で割り当てる場合は、このオプションを選択します。

自動 (VPN) アドレス専用: 接続しているネットワークが **IP** アドレスと **DNS** サーバーを自動で割り当てるものの、**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。

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** サーバーを自動で割り当てる場合は、このオプションを選択します。

自動 (PPPoE) アドレス専用: 接続しているネットワークが **IP** アドレスと **DNS** サーバーを自動で割り当てるものの、**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。

ネットワーク接続の静的ルート設定に関する詳細は、「[ルートの設定](#)」を参照してください。

2.7.7. IPv6 セットアップの設定

方式

無視する: この接続の **IPv6** セットアップを無視したい場合は、このオプションを選択します。

自動: **SLAAC** を使って、ハードウェアのアドレスおよび ルーターアドバタイズ (RA) に基づいて自動のステートレス設定を作成する場合は、このオプションを選択します。

自動、アドレスのみ: 接続中のネットワークが ルーターアドバタイズ (RA) を使用して自動のステートレス設定を作成するものの、**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 を使用しないで、**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。

手動: IP アドレス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。

ローカルへのリンク専用: 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たくない場合に、このオプションを選択します。『RFC 4862』に従って、接頭辞 **FE80::0** の付いたランダムなアドレスが割り当てられます。

アドレス

DNS サーバー: コンマ区切りの **DNS** サーバーのリストを入力します。

ドメインを検索: コンマで区切られたドメインコントローラーのリストを入力します。

ネットワーク接続の静的ルート設定に関する詳細は、「[ルートの設定](#)」を参照してください。

2.7.8. ルートの設定

ホストのルーティングテーブルは、ネットワークに直接接続されているルートで自動的に設定されます。このルートは、「up」の状態にあるネットワークインターフェースを確認することで学習されます。このセクションでは、VPN トンネルや専用線などの媒介ネットワークや接続を移動してアクセスできるネットワークやホストへの静的ルートの入力方法について説明します。リモートネットワークやホストにアクセスするために、トラフィックの送信先となるゲートウェイのアドレスがシステムに付与されます。

ホストのインターフェースが **DHCP** で設定される際には通常、アップストリームネットワークや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、システムに既知の別のすぐれたルートがない場合 (またルーティングテーブルにない場合) に使用するゲートウェイなので、通常これはデフォルトゲートウェイと呼ばれます。ネットワーク管理者は、ネットワークの最初もしくは最後のホスト **IP** アドレスをゲートウェイアドレスとして使用する場合があります。たとえば、**192.168.10.1** や **192.168.10.254** などです。ネットワーク自体を表すアドレスと混同しないようにしてください。この例では、**192.168.10.0** であったり、サブネットのブロードキャストアドレスとなる **192.168.10.255** がこれに当たります。

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** セットティングウィンドウを開きます。これに関しては、「[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

アドレス: 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プレフィックス長。

ゲートウェイ: 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** アドレスです。

メトリック: このルートに与える優先値であるネットワークコスト。高い値よりも低い値の方が望ましい。

自動

自動が **オン** になっている場合は、**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。これが **オフ** の場合は、ユーザーが定義した静的ルートのみが使用されます。

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、この接続がデフォルトルートになりません。よくある例は、ヘッドオフィスへの接続が 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たくない場合です。このオプションを選択すると、この接続で自動的に学習されたルートを使用することが明確なトラフィックか、手動で入力されたトラフィックのみがこの接続を経由します。

2.8. その他のリソース

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)』 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** コマンドの例を提供しています。
- 『nm-settings(5)』 man ページ: **NetworkManager** のプロパティおよびその設定を説明しています。
- 『nm-settings-ifcfg-rh(5)』 man ページ: **ifcfg-rh** 設定のプラグインを説明しています。

オンラインのドキュメント

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

IPsec ベースの VPN およびその設定について説明しています。DNSSEC を使用した認証 **DNS** クエリーの使用を説明しています。

『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

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

『RFC 3330』 : Special-Use IPv4 Addresses

Internet Assigned Numbers Authority (IANA) が割り当てたグローバルおよび他の特定 **IPv4** アドレスブロックを説明しています。

第3章 ホスト名の設定

3.1. ホスト名について

hostname には、static (静的)、pretty、transient (一時的) の 3 つのクラスがあります。

「static」ホスト名は従来の **hostname** で、ユーザーが選択することができ、**/etc/hostname** ファイルに保存されます。「transient」**hostname** 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。ランタイム時に **DHCP** または **mDNS** による変更が可能です。「pretty」**hostname** はユーザーに自由形式の UTF8 ホスト名を提示するものです。

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** のように **DNS** 内のマシンで使われている 完全修飾ドメイン名 (FQDN) に合致することを推奨しています。また、static および transient のホスト名は 7 ビットの ASCII 小文字のみで構成され、空白やドットを含めず、**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ています。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「.」のみで構成され、最初と最後がドットでないこと、また 2 つのドットが連続しないことを強制します。また、最大 64 文字までの長さも強制されます。

3.1.1. 推奨される命名プラクティス

ICANN (The Internet Corporation for Assigned Names and Numbers) は、(**.yourcompany** などの) トップレベルの未登録ドメインを公開登録簿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。これは、ネットワーク設定によっては異なる解決をしてしまうドメインネームになってしまう可能性があるからです。その結果、ネットワークリソースは利用不可能になってしまいます。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、ドメインネームが競合すると DNSSEC 検証に手動の設定が必要となるからです。この問題に関する詳細情報は、「[Name Collision Resources and Information](#)」を参照してください。

3.2. テキスト形式のユーザーインターフェース **NMTUI** を使ったホスト名の設定

テキスト形式のユーザーインターフェースツール **nmtui** を使うと、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
```

テキスト形式のインターフェースが表示されます。無効なコマンドの場合は、使用法に関するメッセージがプリントされます。

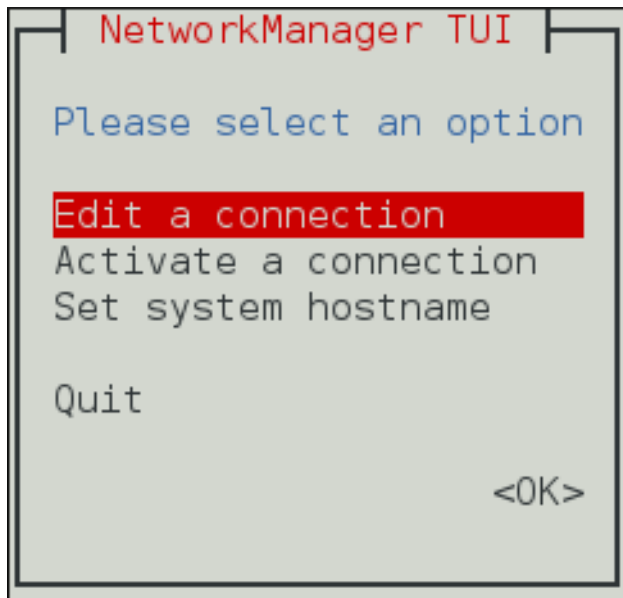


図3.1 NetworkManager のテキスト形式ユーザーインターフェースの開始メニュー

移動するには矢印キーを使用するか、**Tab** を押して次に進むか **Shift+Tab** を押して前に戻ります。**Enter** を押してオプションを選びます。**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については、「[テキスト形式のユーザーインターフェース \(nmtui\) を使ったネットワーク設定](#)」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ースツール **nmtui** を使用すると、**/etc/hostname** ファイル内の静的ホスト名にクエリーを実行し、設定することができます。



重要

Red Hat Enterprise Linux 7 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実行します。したがって、**/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。**hostnamectl** ユーティリティを使って、システムのホスト名を変更する必要があります。また、**/etc/sysconfig/network** ファイルでの **HOSTNAME** 変数の使用は非推奨になりました。

3.3. HOSTNAMECTL を使ったホスト名の設定

hostnamectl ツールは、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。

3.3.1. 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
```

オプションなしで実行された場合は、デフォルトで **status** オプションが暗示されます。

3.3.2. 全ホスト名の設定

システム上のホスト名すべてを設定するには、**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。空白は「-」で置換され、特殊文字は削除されます。

3.3.3. 特定のホスト名の設定

特定のホスト名を設定するには、**root** で以下のコマンドに関連するオプションと実行します。

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

ここでの *option* は、**--pretty**、**--static**、および **--transient** のいずれか 1 つまたは複数となります。

--static または **--transient** のオプションを **--pretty** オプションと併用した場合、static および transient のホスト名は簡素化された pretty ホスト名形式になります。空白は「-」で置換され、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。例を示します。

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

3.3.4. 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** で以下のコマンドに関連するオプションと実行します。

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

ここでの "" は空白の文字列の引用となり、*option* は **--pretty**、**--static**、および **--transient** のいずれか 1 つまたは複数となります。

3.3.5. ホスト名のリモートでの変更

リモートシステム上で **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** を使ってリモートシステムに接続します。

3.4. NMCLI を使ったホスト名の設定

NetworkManager のツール **nmcli** を使用すると、**/etc/hostname** ファイル内の静的ホスト名にクエリーを実行し、設定することができます。

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
```

3.5. その他のリソース

- **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** を説明しています。
- **hostname(1)** man ページ: **hostname** および **domainname** の各コマンドを説明しています。
- **hostname(5)** man ページ: ホスト名ファイル、そのコンテンツおよび使用法について説明しています。
- **hostname(7)** man ページ: ホスト名解決について説明しています。
- **machine-info(5)** man ページ: ローカルマシンの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 の設定ファイルについて説明しています。
- **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4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ー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ースが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。スイッチに付属の説明書と

<https://www.kernel.org/doc/Documentation/networking/bonding.txt> を参照してください。

4.1. マスターおよびスレーブインターフェースのデフォルト動作について

NetworkManager デーモンを使ってボンディングされたスレーブインターフェースを制御する際、特に障害検索時には、以下の点に留意してください。

1. マスターインターフェースを起動しても、スレーブインターフェースは自動的に起動されない。
2. スレーブインターフェースを起動すると、マスターインターフェースは毎回、自動的に起動される。
3. マスターインターフェースを停止すると、スレーブインターフェースも停止される。
4. マスターはスレーブなしで静的 **IP** 接続を開始できる。
5. マスターはスレーブなしの場合、**DHCP** 接続の開始時にスレーブを待機する。
6. **DHCP** 接続でスレーブを待機中のマスターは、キャリアをとまなうスレーブが追加されると完了する。
7. **DHCP** 接続でスレーブを待機中のマスターは、キャリアをとまなわないスレーブが追加されると待機を継続する。

4.2. テキスト形式のユーザーインターフェース **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ースツール **nmtui** を使うと、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
```

テキスト形式のインターフェースが表示されます。無効なコマンドの場合は、使用法に関するメッセージがプリントされます。

移動するには矢印キーを使用するか、**Tab** を押して次に進むか **Shift+Tab** を押して前に戻ります。**Enter** を押してオプションを選びます。**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。**追加** を選択すると **新規の接続** 画面が開きます。

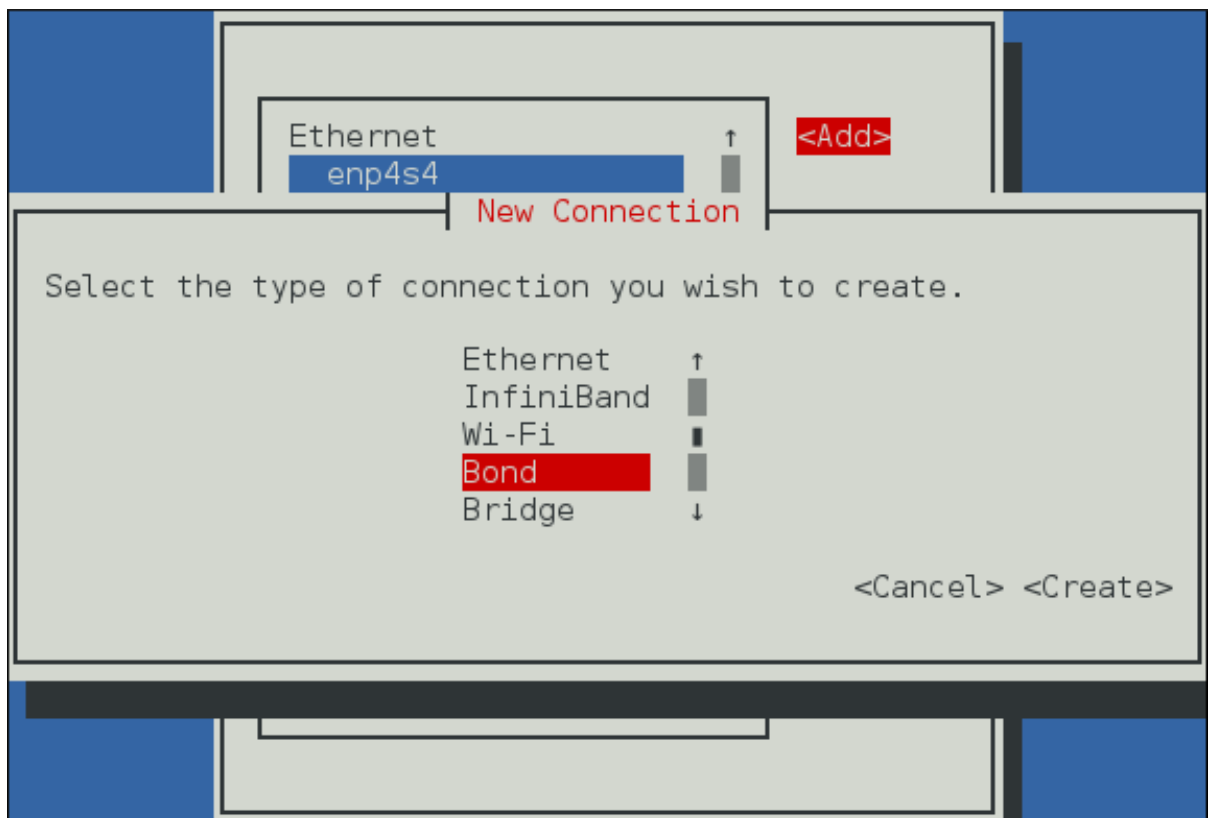


図4.1 NetworkManager テキスト形式のユーザーインターフェースのボンド接続追加メニュー

2. **ボンド** の後に **作成** を選択すると **接続の編集** 画面が開きます。

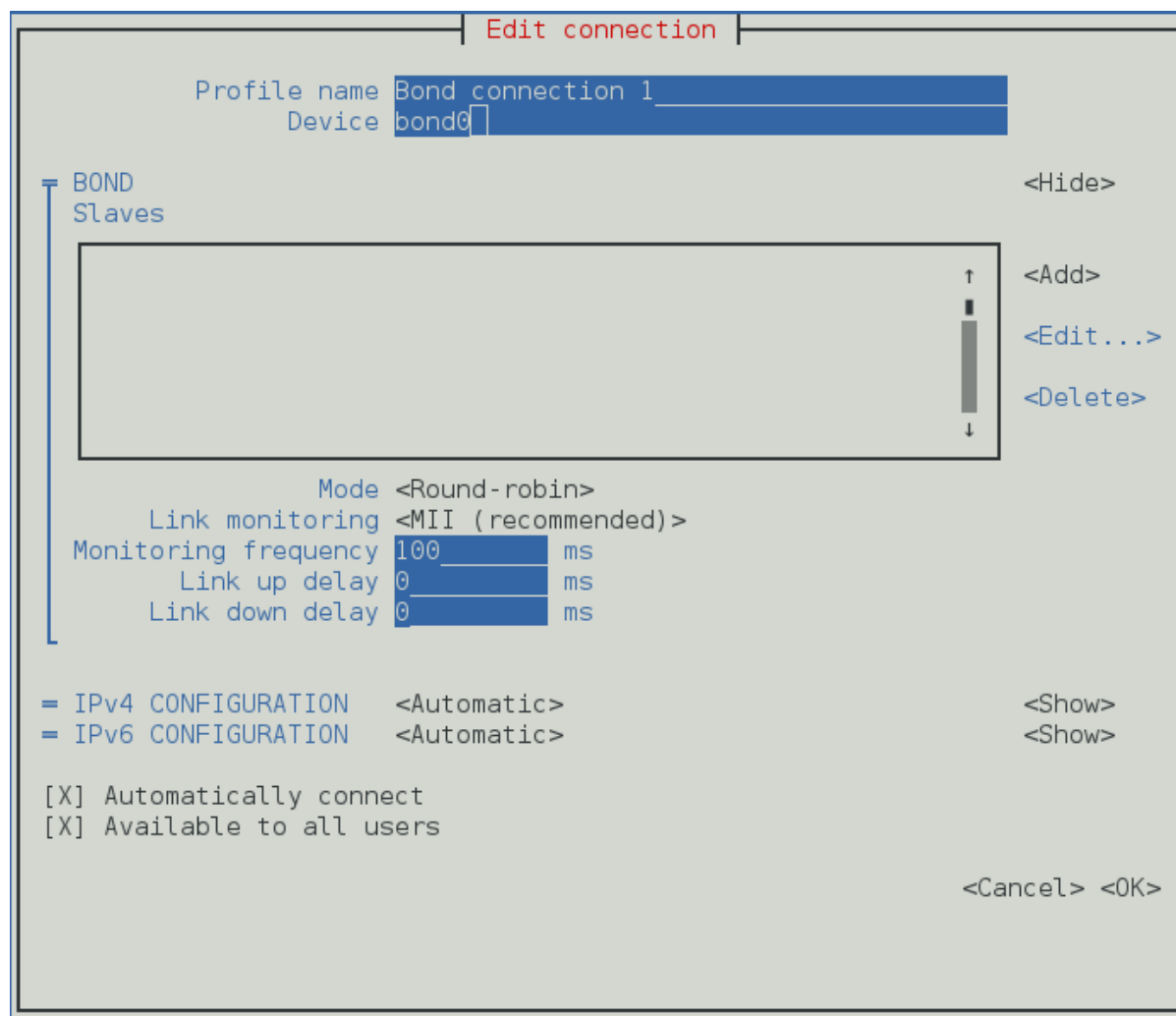


図4.2 NetworkManager テキスト形式ユーザーインターフェースでボンド接続を設定するメニュー

3. この時点で、ボンドにスレーブインターフェースを追加する必要があります。これを行うには **追加** を選択して **新規の接続** 画面を開きます。接続の種類を選んだら、**作成** ボタンを選択します。

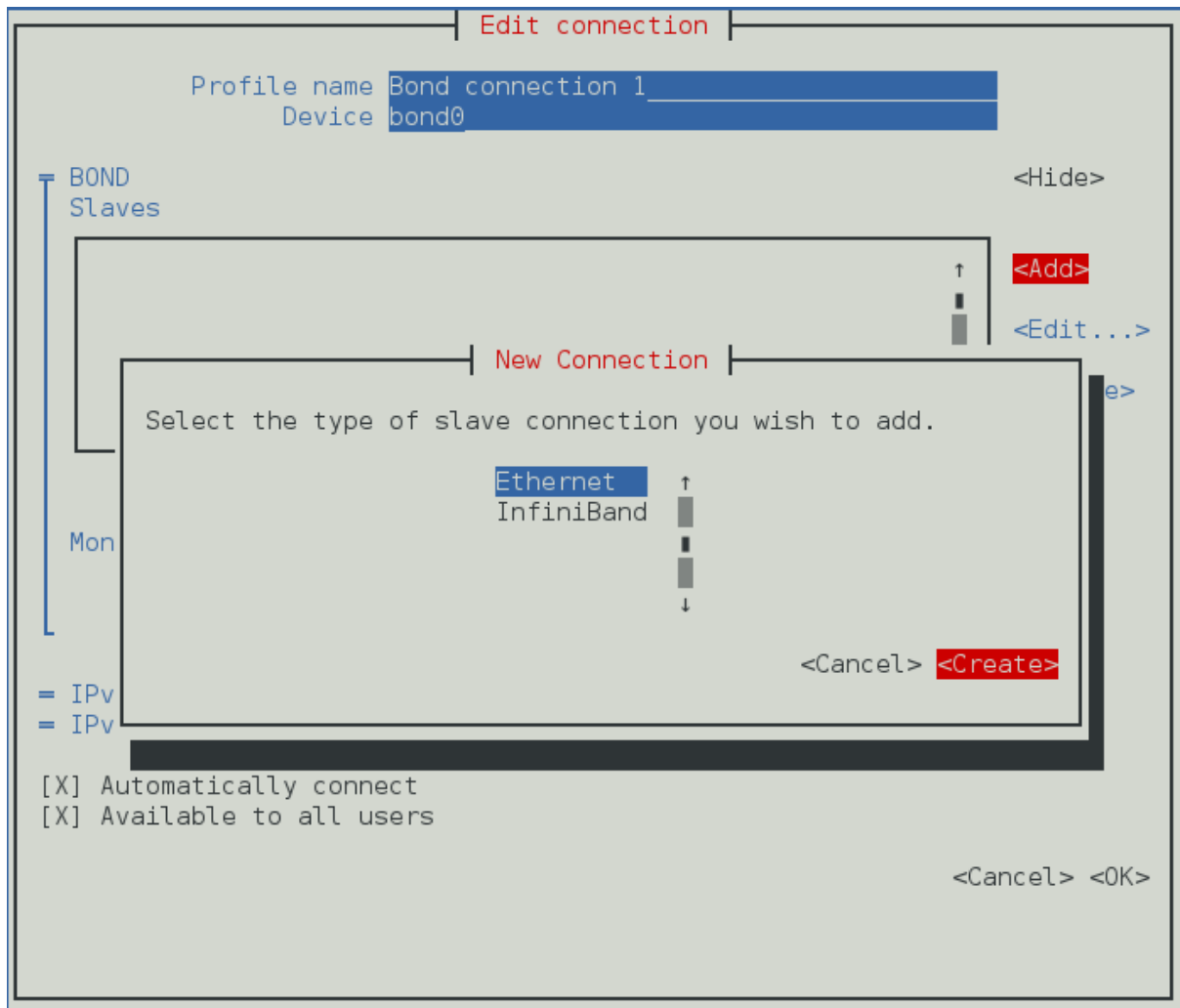


図4.3 NetworkManager テキスト形式ユーザーインターフェースで新規ボンDSLレーブを設定するメニュー

4. スレーブの **接続の編集** 画面が表示されます。**デバイス** セクションにスレーブのデバイス名もしくは MAC アドレスを入力します。必要であれば、**イーサネット** ラベルの右側にある **表示する** 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** ボタンを選択してスレーブを保存します。



注記

MAC アドレスなしでデバイスを指定すると、**接続の編集** ウィンドウがリロードされる際に **デバイス** セクションは自動的に設定されます。ただしこれは、デバイスが正常に発見された場合のみです。

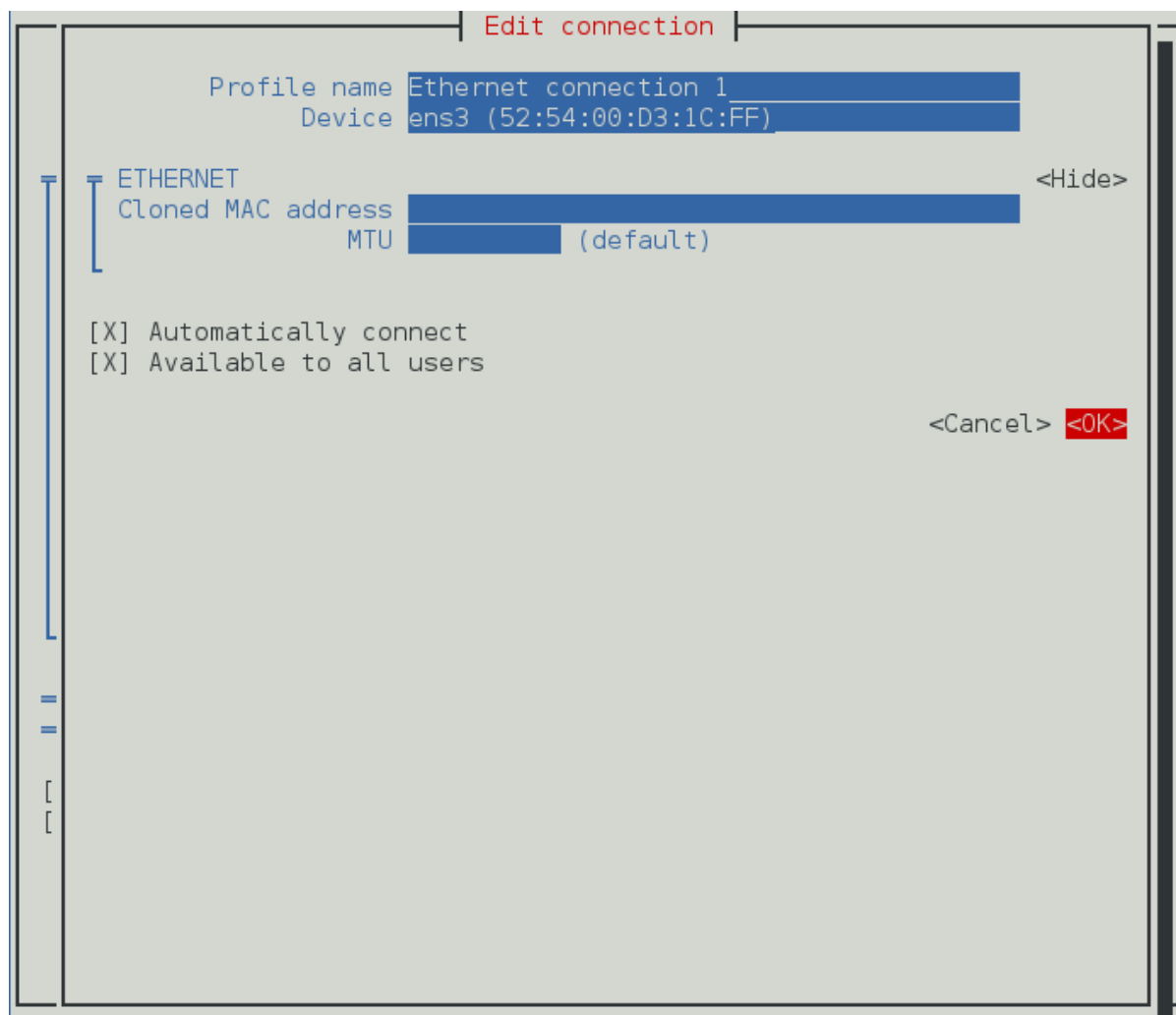


図4.4 NetworkManager テキスト形式ユーザーインターフェースでボンドスレーブ接続を設定するメニュー

5. スレーブ セクションにボンドスレーブの名前が表示されます。さらにスレーブ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。

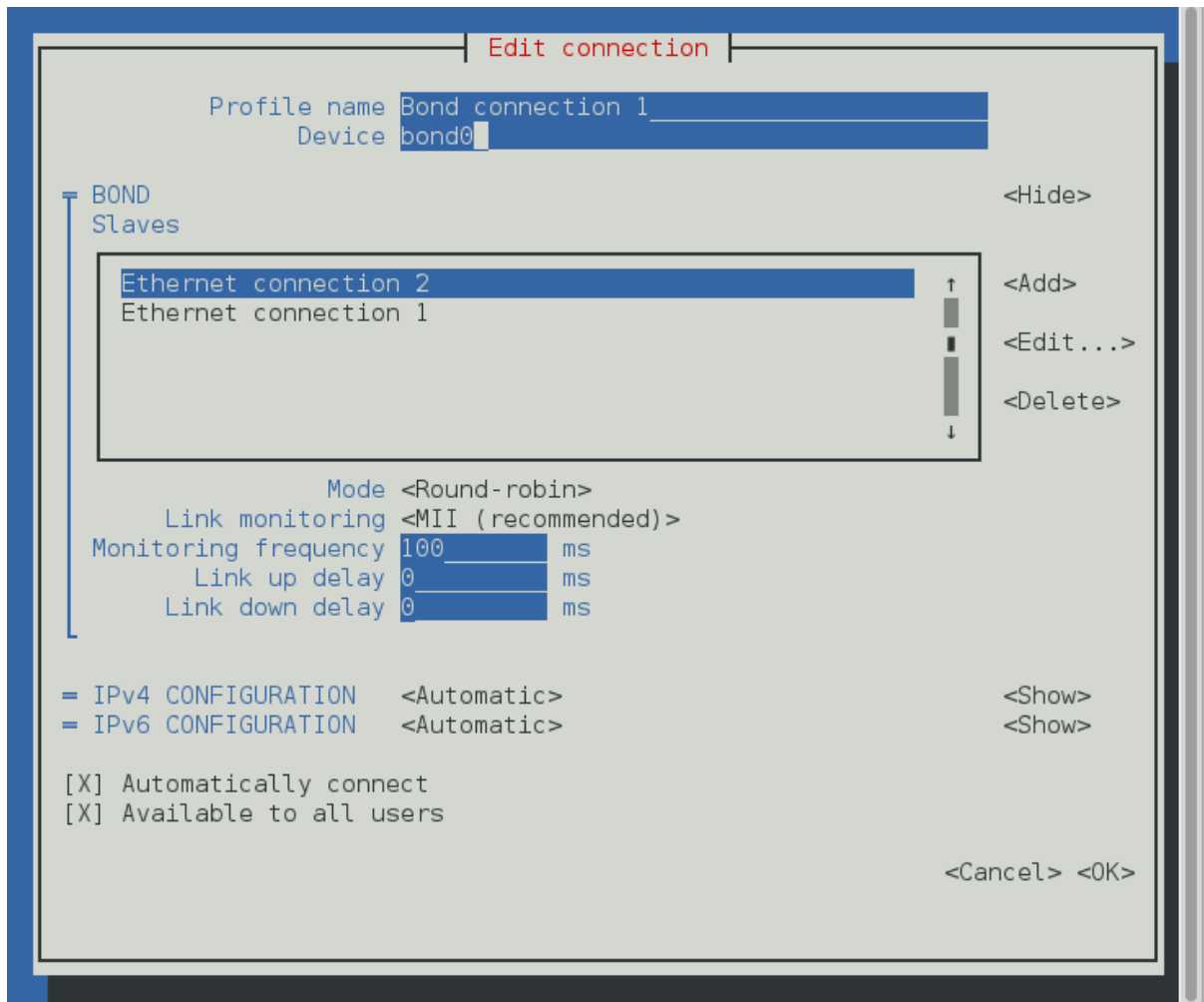


図4.5 NetworkManager のテキスト形式ユーザーインターフェースでボンドを完了

ボンド用語の定義については、「[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については、「[テキスト形式のユーザーインターフェース \(nmtui\) を使ったネットワーク設定](#)」を参照してください。

4.3. NETWORKMANAGER のコマンドラインツール NMCLI を使ったネットワークボンディング



注記

`nmcli` の導入については、「[NetworkManager コマンドラインツール \(nmcli\) の使用](#)」を参照してください。

`nmcli` ツールを使って 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、接続名がインターフェー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。

NetworkManager では、カーネルで提供されるほとんどのボンディングオプションがサポートされます。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.
```

スレーブ インターフェースを追加するには、以下の手順を実施します。

1. 新規接続を作成します。詳細は、[「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. マスターのプロパティを **ボンド** のインターフェース名、またはマスタ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.
```

新たなスレーブ インターフェースを追加するには、別のインターフェースで前の手順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.
```

スレーブをアクティブ化するには、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)
```

スレーブをアクティブ化すると、マスター接続も開始されます。詳細は、[「マスターおよびスレーブインターフェースのデフォルト動作について」](#)を参照してください。したがって、マスター接続を手動でアクティブ化する必要はありません。

接続を非アクティブ化することなく、ランタイム時にボンドの **active_slave** オプションおよび **primary** オプションを変更することができます。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、**primary** 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 "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ースが、直ちにアクティブスレーブになります。一方、ボンドの **primary** オプションでは、新たなスレーブが追加された時またはアクティブスレーブ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スレーブを指定します。

4.4. コマンドラインインターフェース (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ースと呼ばれる特殊なネットワークインターフェースを使用して作成されます。

4.4.1. 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このモジュールを読み込みます。

```
~]# 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。再起動後も維持されるモジュールの読み込みについては、『[Red Hat Enterprise Linux 7 システム管理者のガイド](#)』を参照してください。**BONDING_OPTS** ディレクティブを使用した適切な設定ファイルがあれば、ボンディングモジュールは必要に応じて読み込まれるので、別個に読み込む必要はないことに留意してください。

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)** man ページを参照してください。

4.4.2. チャンネルボンディングインターフェースの作成

チャンネルボンディングインターフェースを作成するには、**/etc/sysconfig/network-scripts/** ディレクトリーに **ifcfg-bondN** という名前のファイルを作成し、**N** をそのインターフェースの番号 **0** などに置き換えます。

ファイルのコンテンツは、イーサネットインターフェースなどボンディングされるインターフェー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。本質的な相違点は、**DEVICE** ディレクティブが **bondN** となり、この **N** をインターフェースの番号で置換し、さらに **TYPE=Bond** とすることです。また、**BONDING_MASTER=yes** と設定します。

例4.1 ifcfg-bond0 インターフェー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ー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"
```

NAME ディレクティブは、**NetworkManager** 内の接続プロファイルの命名時に便利なものです。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-bondN** インターフェースファイル内の **BONDING_OPTS="bonding parameters"** ディレクティブでスペース区切りの一覧として指定する必要があります。**/etc/modprobe.d/bonding.conf** または非推奨の **/etc/modprobe.conf** ファイルでボンディングデバイスのオプションを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ース固有ではなく、**ifcfg-bondN** ファイルの使用時に **BONDING_OPTS** ディレクティブで指定しないでください。このディレクティブを使うと、ネットワークスクリプトが必要に応じてボンディングインターフェースを作成することになります。

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一覧については、「[チャンネルボンディングの使用](#)」を参照してください。

4.4.3. スレーブインターフェースの作成

チャンネルボンディングインターフェースは「マスター」であり、ボンディングされるインターフェースは「スレーブ」と呼ばれます。チャンネルボンディングインターフェースを作成した後に、ボンディングされるネットワークインターフェースを設定するには、そのスレーブの設定ファイルに **MASTER** と **SLAVE** のディレクティブを追加する必要があります。各スレーブインターフェー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4.2 スレーブインターフェー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ース **eth0** と **eth1** をチャンネルボンディングする場合、それらは両方とも以下のようにすることができます。

```
DEVICE=ethN
NAME=bond0-slave
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```

この例では、*N* をインターフェースの番号で置換します。インターフェースが **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、それらが相互に競合し、ボンドスレーブではなく単純な **TYPE=Ethernet** プロファイルがアクティブ化される場合があることに注意してください。

4.4.4. 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化するには、すべてのスレーブを有効にします。**root** で以下のコマンドを実行します。

```
~]# ifup ifcfg-eth0
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/7)
```

```
~]# ifup ifcfg-eth1
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ースのインターフェー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 ethN
```

これが完了した後、すべてのスレーブ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 にこれらの変更を認識させるには、変更されたインターフェースすべてについて
root で以下の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ースをリロードします。

```
~]# nmcli con reload
```

デフォルトでは、**NetworkManager** は変更を認識せず、変更前の設定データの使用を継続します。こ
れは **NetworkManager.conf** ファイルの **monitor-connection-files** オプションで設定します。
詳細情報は、**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ー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: eth0: <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: eth1: <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
```

4.4.5. 複数のボンド作成

Red Hat Enterprise Linux 7 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ースが作成されます。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ースを作成するに

は、以下の手順に従います。

- **BONDING_OPTS** ディレクティブがある複数の **ifcfg-bondN** ファイルを作成します。このディレクティブを使うと、ネットワークスクリプトが必要に応じてボンディングインターフェースを作成するようになります。
- ボンディングされるインターフェース設定ファイルを作成、または既存のものを編集し、**SLAVE** ディレクティブをこれに含めます。
- ボンディングされるスレーブインターフェースを **MASTER** ディレクティブでチャンネルボンディングインターフェースに割り当てます。

例4.3 複数の ifcfg-bondN インターフェー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ー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* をインターフェースの番号に置き換えます。たとえば 2 つのボンドを作成するには、**ifcfg-bond0** と **ifcfg-bond1** という 2 つの設定ファイルを適切な **IP** アドレスとともに作成します。

例4.2「スレーブインターフェース設定ファイルの例」にあるようにボンディングされるインターフェースを作成し、**MASTER=bondN** ディレクティブを使って必要に応じてそれらをボンドインターフェースに割り当てます。たとえば上記の例では、ボンドあたり 2 つのインターフェースが必要だとすると、2 つのボンドに 4 つのインターフェース設定ファイルを作成し、最初の 2 つを **MASTER=bond0** を使って割り当て、残りの 2 つを **MASTER=bond1** を使って割り当てます。

4.5. 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**、**arp_interval**、**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一覧と使用しているボンディングされたインターフェー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。

4.5.1. ボンディングモジュールのディレクティブ

チャンネルボンディングのモジュールパラメーターをボンディングインターフェース設定ファイル (たとえば **ifcfg-bond0**) の **BONDING_OPTS="bonding parameters"** ディレクティブに追加する前に、どれがボンディングされたインターフェースに最適かをテストするとよいでしょう。ボンディングされたインターフェースのパラメーターは、**sysfs** ファイルシステム内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) することなく設定できます。

sysfs は仮想ファイルシステムであり、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示します。**sysfs** を使用すると、カーネルオブジェクトの情報をクエリーするこ

とが可能で、通常のファイルシステムのコマンドを使用してそれらのオブジェクトの操作もできます。**sysfs** 仮想ファイルシステムは、**/sys/** ディレクトリ下にマウントされます。ボンディングインターフェースはすべて、**/sys/class/net/** ディレクトリ下にあるファイルと対話したり、これらを実行することで動的に設定することができます。

使用中のボンディングインターフェー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ースの作成](#)」にある手順に従って **ifcfg-bond0** などのチャンネルボンディングインターフェースのファイルを作成します。**bond0** にボンディングされている各インターフェースの設定ファイルに **SLAVE=yes** および **MASTER=bond0** の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 bondN** を実行して作成したボンドを有効にします。

```
~]# ifup bond0
```

ifcfg-bond0 のボンディングインターフェースのファイルを正常に作成していれば、**root** で **ip link show** を実行した際の出力に **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: eth0: <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: eth1: <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/bondN/bonding/ ディレクトリにあるファイルを操作すると、各ボンドを個別に設定することができます。まず、設定するボンドをダウンにします。

```
~]# ifdown bond0
```

たとえば、**bond0** の MII 監視を 1 秒間隔で有効にするには、**root** で以下を実行します。

```
~]# 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
```

ボンドにオプションを設定後、**ifup bondM** を実行すると、そのボンドをアップにしてテストすることができます。オプションを変更する場合はインターフェースを停止して、**sysfs** を使用してそのパラメーターを修正後、有効に戻して再確認します。

ボンドに最適なパラメーターのセットを決定したら、設定しているボンディングインターフェースの **/etc/sysconfig/network-scripts/ifcfg-bondM** ファイルの **BONDING_OPTS=** ディレクティブにそれらのパラメーターを空白区切りのリストとして追加します。ボンドが有効な場合 (たとえば、**ONBOOT=yes** ディレクティブが設定されていて、ブートシーケンス中にシステムがボンドを有効にする場合) はいつでも、**BONDING_OPTS** で指定されているボンディングオプションはそのボンドに対して有効となります。

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一覧表示しています。詳細については、**modinfo bonding** 出力の各 **parm** の簡単な説明、または <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ースパラメーター

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。指定可能な値は、以下のとおりです。

- **stable** または **0**: デフォルト設定です。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、すべてのスレーブがダウンとなるか、スレーブ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以下の場合に行われます。
 - ボンドにスレーブが追加される、またはボンドからスレーブが削除される。
 - スレーブのリンク状態が変更される。
 - スレーブの 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、スレーブの最大数によって選択されます。再選択は、上記の **bandwidth** 設定の場合に行われます。

bandwidth および **count** の選択ポリシーは、アクティブアグリゲーターが部分的に失敗した場合に、802.3ad アグリゲーションのフェイルオーバーを可能にします。これにより、常にアクティブな状態のアグリゲーターを帯域幅もしくはスレーブ数における最大の利用可能数で維持します。

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** の両パラメーター、あるいは **miimon** パラメーターの指定は不可欠です。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、ネットワークスイッチは NIC 全体に均等にパケットを分散するよう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** に設定されており、ARP 監視を無効にします。

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の **ARP** 要求のターゲット **IP** アドレスを指定します。コンマ区切りのリストで最大 16 個の **IP** アドレスを指定できます。

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。他の有効な値は、**active**、**backup**、**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る値の倍数でなければなりません。デフォルトでは **0** に設定されており、待機時間を無効にします。

fail_over_mac=value

アクティブ-バックアップモードが、スレーブ指定時にすべてのスレーブを同一 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、MAC アドレスのボンドの特別な処理を実行するかを指定します。利用可能な値は以下のとおりです。

- **none** または **0**: デフォルト設定です。この設定では **fail_over_mac** が無効になり、スレーブ指定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スレーブを同一 MAC アドレスに設定するようにします。
- **active** または **1**: 「active」 **fail_over_mac** ポリシーでは、ボンドの MAC アドレスは常に現在アクティブなスレーブの MAC アドレスにすべきと指示しています。スレーブの 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** ポリシーでは、ボンドの MAC アドレスは通常どおり選択されます (通常、最初のスレーブの MAC アドレスがボンドに追加されます)。ただし、2 番目以降のスレーブはこの MAC アドレスに設定されず、バックアップの役割を果たします。つまり、スレーブ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スレーブが新たにアクティブになったスレーブの 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。指定可能な値は、以下のとおりです。

- **slow** または **0**: デフォルト設定です。パートナーが 30 秒ごとに LACPDU を送信するように指定します。
- **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* をボンダイインターフェースではなく **eth0** のようなデバイスインターフェースの名前で置換します。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ースを使用する場合、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。この設定を行う場合、このパラメーターの適切な開始ポイントは **100** です。

**重要**

arp_interval および **arp_ip_target** の両パラメーター、あるいは **miimon** パラメーターの指定は不可欠です。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、以下のいずれかになります。

- **balance-rr** または **0**: 耐障害性とロードバランシングのため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ースからそれぞれのボンディングされたスレーブインターフェー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なバックアップポリシーを設定します。利用可能な最初のボンディングされたスレーブインターフェースにより送受信が行われます。別のボンディングされたスレーブインターフェースは、アクティブなボンディングされたスレーブインターフェースが失敗した場合にのみ使用されます。
- **balance-xor** または **2**: 送受信は選択されたハッシュポリシーに基づきます。デフォルトでは、ハッシュはソースの XOR とスレーブインターフェース数による剰余で宛先 MAC アドレスを掛けて導き出します。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ースで送信されます。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ありません。

- **broadcast** または **3**: 耐障害性のための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、すべてのスレーブインターフェースで行われます。
- **802.3ad** または **4**: IEEE 802.3ad 動的リンクアグリゲーションの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スレーブ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**: 耐障害性とロードバランシングのための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、各スレーブインターフェー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、現在のスレーブにより受信されます。受信しているスレーブが失敗すると、別のスレーブが失敗したスレーブの MAC アドレスを引き継ぎます。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**: 耐障害性とロードバランシングのためアダプティブ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ロードバランシングが含まれます。**ARP** ネゴシエーションにより、受信ロードバランシングが可能です。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。

primary=interface_name

eth0 のようなプライマリーデバイスのインターフェース名を指定します。**primary** デバイスは、使用される最初のボンディングインターフェースであり、失敗しない限りは破棄されません。この設定が特に役立つのは、ボンディングインターフェースの 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ースが **active-backup** モードの場合にのみ有効です。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。

primary_reselect=value

プライマリースレーブに対して再選択ポリシーを指定します。これは、アクティブなスレーブの失敗やプライマリースレーブの回復が発生した場合に、どのようにプライマリースレーブが選択されてアクティブなスレーブになるかという点に影響します。このパラメーターは、プライマリースレーブと他のスレーブ間でのフリップフロップを防ぐように設計されています。指定可能な値は、以下のとおりです。

- **always** または **0** (デフォルト): プライマリースレーブは有効になるといつでもアクティブなスレーブになります。
- **better** または **1**: プライマリースレーブの速度とデュプレックスが、現在のアクティブなスレーブの速度とデュプレックスと比べて良い場合は、プライマリースレーブは有効になるとアクティブなスレーブになります。
- **failure** または **2**: 現在のアクティブなスレーブが失敗してプライマリースレーブが有効になる場合のみ、プライマリースレーブはアクティブなスレーブになります。

primary_reselect の設定は、以下の 2 つの場合では無視されます。

- アクティブなスレーブがない場合は、回復する最初のスレーブがアクティブなスレーブになります。

- 初めにプライマリースレーブがスレーブにされた場合は、それは常にアクティブなスレーブになります。

sysfs により **primary_reselect** ポリシーが変更されると、新しいポリシーに従って直ちに最良のアクティブなスレーブが選択されます。これにより、状況によってはアクティブなスレーブ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、デフォルト値は **1** です。値が **0** だと、フェイルオーバーイベント時に IGMP メンバーシップレポートが発行されません。

このオプションは、フェイルオーバーで IGMP トラフィックをあるスレーブから別のスレーブに切り替えられる **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で便利なものです。このため、新たに発行される IGMP レポートは、スイッチが着信 IGMP トラフィックを新たに選択されたスレーブに転送させるように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る値の倍数でなければなりません。デフォルトでは、値は **0** に設定されており、待機時間を無効にします。

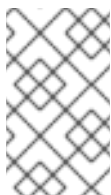
use_carrier=number

リンク状態を決定するために **miimon** が MII/ETHTOOL `ioctl`s または **netif_carrier_ok()** を使用するかどうか指定します。**netif_carrier_ok()** 機能は、デバイスドライバを使用して **netif_carrier_on/off** によりその状態を維持します。大半のデバイスドライバはこの機能に対応しています。

MIII/ETHTOOL `ioctl`s 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、これは使用しているデバイスドライバが **netif_carrier_on/off** に対応しない場合でも設定可能です。

有効な値は次のとおりです。

- **1**: デフォルト設定です。**netif_carrier_ok()** の使用を有効にします。
- **0**: MII/ETHTOOL `ioctl`s の使用を有効にします。



注記

リンクがアップになっているべきでない時にアップであるとボンディングインターフェースが主張した場合、使用しているネットワークデバイスドライバは **netif_carrier_on/off** に対応し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、スレーブを選択する時に使用する送信ハッシュポリシーを選択します。指定可能な値は、以下のとおりです。

- **0** または **layer2**: デフォルト設定です。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スレーブ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**: 上位レイヤープロトコルの情報を (利用可能な場合は) 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スレーブに及ぶようにできますが、単一の接続では複数のスレーブに及びません。

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 パケットと他のすべての **IP** プロトコルトラフィックについては、送信元および宛先ポート情報が省略されます。非 **IP** トラフィックに関しては、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スレーブ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一です。

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。

4.6. GUI を使用したボンディング接続の作成

nm-connection-editor を使って、**NetworkManager** に 2 つ以上の有線もしくは InfiniBand 接続からボンドを作成するよう指示することができます。接続が最初にボンディングされている必要はありません。ボンド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。この設定プロセスを完了するには、利用可能なインターフェースの MAC アドレスが必要です。

4.6.1. ボンド接続の確立

手順4.1 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. ターミナルで **nm-connection-editor** と入力します。

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

2. **追加** ボタンをクリックします。**接続の種類の選択** ウィンドウが表示されます。**ボン**ドを選択して **作成** をクリックします。**ボン**ド接続 **1** の**編集** ウィンドウが表示されます。

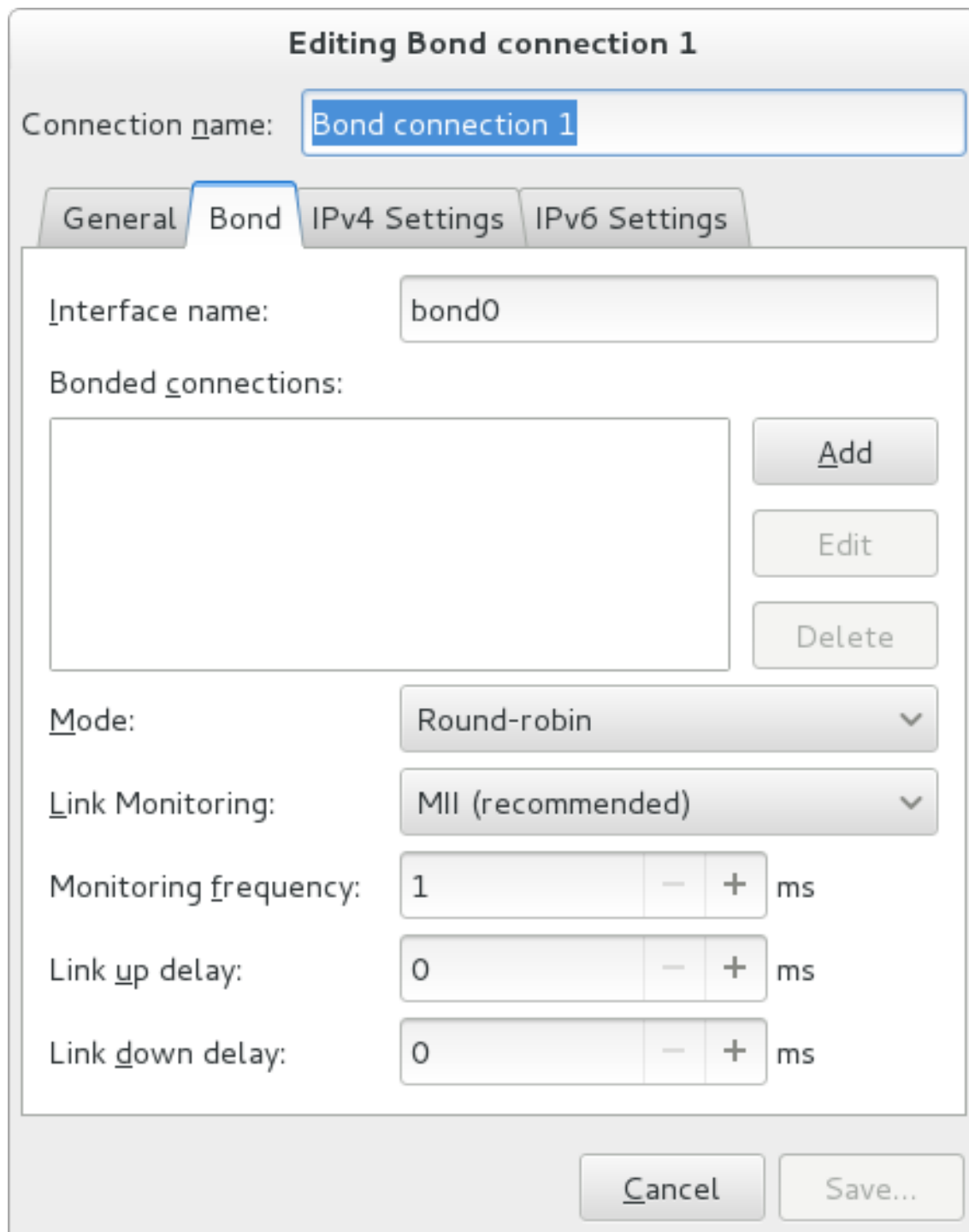


図4.6 NetworkManager グラフィカルユーザーインターフェースの Bond 追加メニュー

3. **Bond** タブで **追加** をクリックし、このボンド接続で使用するインターフェースのタイプを選択します。**作成** ボタンをクリックします。スレーブタイプを選択するダイアログが表示されるの

は、最初のスレーブを作成する時のみです。その後は、すべてのスレーブ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。デバイスの **MAC アドレス** ドロップダウンメニューでボンディングされるインターフェースの MAC アドレスを選択します。最初のスレーブの MAC アドレスがボンドインターフェース用の MAC アドレスとして使用されます。必要の場合は、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。**保存** ボタンをクリックします。

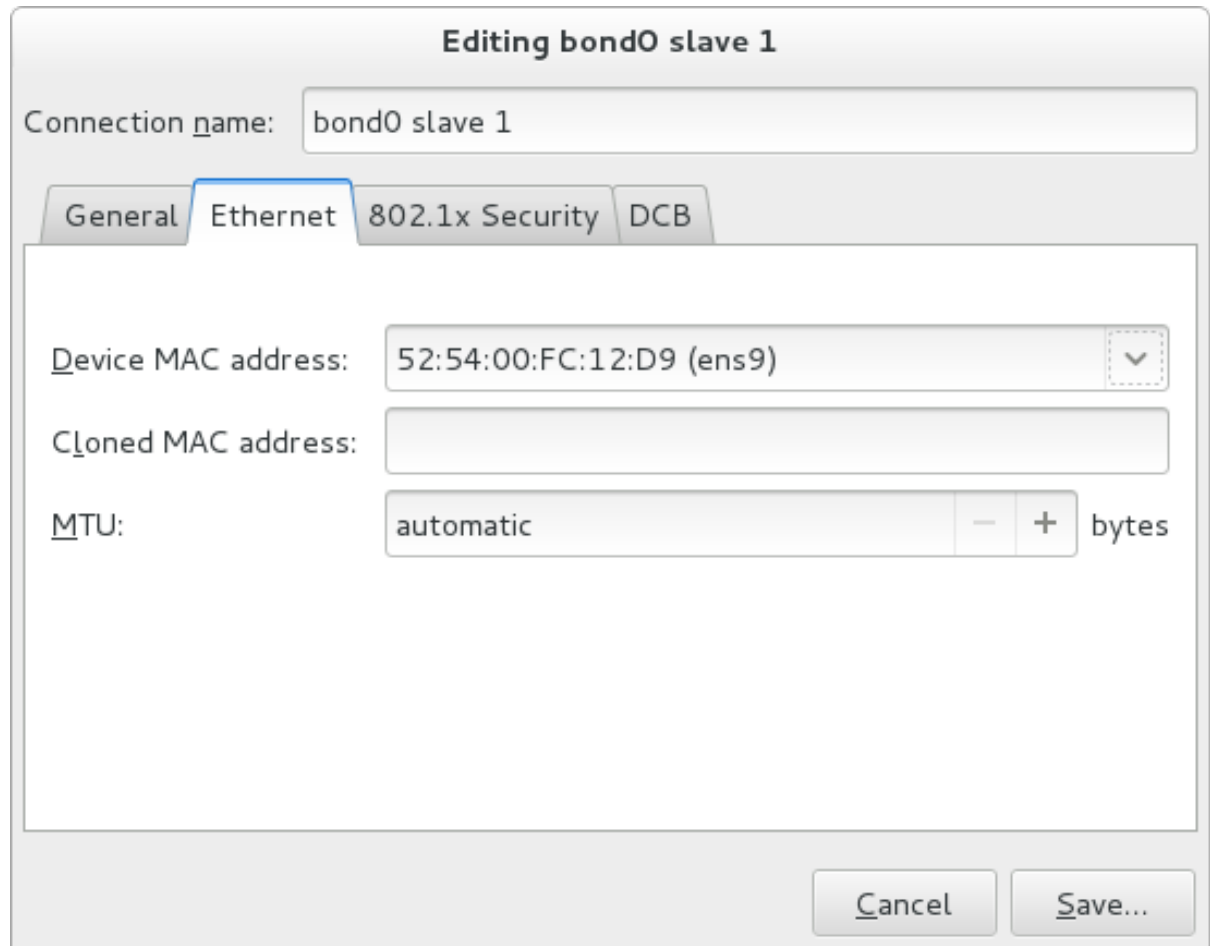


図4.7 NetworkManager グラフィカルユーザーインターフェースのボンド接続追加メニュー

5. ボンディングされたスレーブ名が **Bond 接続** ウィンドウに表示されます。**追加** ボタンをクリックしてさらにスレーブ接続を追加します。
6. 設定を確認してから **保存** ボタンをクリックします。
7. ボンド固有の設定については、「[Bond タブの設定](#)」を参照してください。

手順4.2 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. ターミナルで **nm-connection-editor** と入力します。

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

2. 編集する接続を選択して、**編集** ボタンをクリックします。

3. **全般** 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 **全般** タブにある 5 つの設定は、すべての接続の種類で共通のものです。

- **接続名**: ネットワーク接続の名前を入力します。この名前は **Network** ウィンドウメニューの接続名一覧に表示されます。
 - **この接続が利用可能になったときは自動的に接続する**: このボックスにチェックを入れると、この接続が利用可能な時に **NetworkManager** が自動接続します。詳細については「[ネットワークへの自動接続](#)」を参照してください。
 - **全ユーザーがこのネットワークに接続可能とする**: システム上のすべてのユーザーが接続可能とするには、このボックスにチェックを入れます。このセッティングを変更するには root 権限が必要になることがあります。詳細については「[システムワイドおよびプライベート接続プロファイル](#)」を参照してください。
 - **この接続を使用したときは自動的に VPN に接続する**: このボックスにチェックを入れると、この接続が利用可能な時に **NetworkManager** が自動で選択された VPN に接続します。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 セキュリティガイド](#)』を参照してください。
5. ボンド固有の設定については、「[Bond 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

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。

以下の設定が可能です。

- **IPv4** の設定は、**IPv4 のセッティング** タブをクリックして「[IPv4 セッティングの設定](#)」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。

4.6.1.1. 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 (手順に関しては、[手順4.1「nm-connection-editor を使用して新規ボンド接続を追加する」](#)を参照)、**ボンド** タブを編集して、負荷分散モードとスレーブ接続の障害検出に使用するリンク監視のタイプを設定できます。

モード

ボンドを構成するスレーブ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** などの他の負荷分散モードは、ドロップダウンリストから選択することができます。

リンク監視

ネットワークトラフィックを伝送するスレーブの能力を監視する方法。

以下の負荷分散モードが、モード の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。送受信は、ボンディングされた各スレーブインターフェースで、最初に利用可能になったインターフェースから順次行われます。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スレーブインターフェースにより送受信が行われます。別のボンディングされたスレーブインターフェースは、アクティブなボンディングされたスレーブインターフェースが失敗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、ハッシュはソースの XOR とスレーブインターフェース数による剰余で宛先 MAC アドレスを掛けて導き出します。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ースで送信されます。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。送受信はすべて、スレーブインターフェースで実行されます。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の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スレーブで送受信を行います。**802.3ad** に対応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、各スレーブインターフェー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、現在のスレーブにより受信されます。受信しているスレーブが失敗すると、別のスレーブが失敗したスレーブの MAC アドレスを引き継ぎます。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ロードバランシングが含まれます。**ARP** ネゴシエーションにより、受信ロードバランシングが可能です。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、**リンク監視** 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ー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)

インターフェースのキャリア波の状態を監視します。実行方法は、ドライバーへのクエリー、MII レジスターへの直接クエリー、**ethtool** を使ったデバイスへのクエリーがあります。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)。リンクが「up」とレポートされてからすぐに余計な **ARP** リクエストが失われた場合に、この遅延は使用できます。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 つ以上のピアにプローブしてリンク層接続の動作具合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時間 (ミリ秒単位)。

ARP ターゲット

ARP リクエスト送信先の **IP** アドレスのコンマ区切り。

4.7. その他のリソース

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** コマンドの例を提供しています。
- **nm-settings(5)** man ページ: **NetworkManager** 接続の設定およびパラメーターを説明しています。

オンラインのドキュメント

『Red Hat Enterprise Linux 7 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ースでの 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5章 ネットワークチームingの設定

5.1. ネットワークチームing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、ポートトラッキング、チャンネルチームing、NIC チームing、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。もともと Linux カーネルで実装されたこの概念は、広く **ボンディング** と呼ばれています。この概念の新しい実装の呼び方として、ネットワークチームing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は影響を受けません。ネットワークチームing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。

ネットワークチームing (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ース (API) が備わっており、これが Netlink 通信を実行します。ユーザースペースのアプリケーションは、この API を使ってドライバと通信できます。「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の **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、ネットワークチームing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、カスタムランナーを作成して **teamd** により新たな論理を実装するのも比較的簡単です。**teamd** の使用は必須ではないので、**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さえ可能です。

teamdctl ユーティリティにより、D-bus を使用して実行中の **teamd** インスタンスを制御することができます。**teamdctl** は、**teamd** D-Bus API を覆う 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、通信しますが、D-Bus の監視も継続します。これは、D-Bus が存在しない、またはまだ読み込まれていない環境でも **teamd** を使用可能とするためです。たとえば、**teamd** リンクで起動する際には、D-Bus はまだ読み込まれていません。ランタイム時に **teamdctl** ユーティリティを使うと、設定およびリンク監視の状態を読み取ることや、ポートの状態の確認および変更、ポートの追加および削除、ポートをアクティブおよびバックアップ状態間で切り替えることができます。

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。ユーザースペースのライブラリーである **libteam** はこの API と直接対話しませんが、**libnl** または **teamnl** を使ってドライバ 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。設定はすべて、**teamd** アプリケーションのようなユーザースペースのアプリケーションを使って行われます。アプリケーションはその後、カーネルドライバのパートに適切に指示します。



注記

ネットワークチームingにおいては、**ポート** の代わりに **スレーブ** という用語が使われることもあります。直接 **teamd** を用いている場合には、**ポート** を使うのが一般的です。一方、**NetworkManager** を用いている時にチームを形成するインターフェースを参照する場合には、**スレーブ** を使います。

5.2. マスターおよびスレーブインターフェースのデフォルト動作について

NetworkManager デーモンを使ってチームポートのインターフェースを制御する際、特に障害検索時には、以下の点に留意してください。

1. マスターインターフェースを起動しても、ポートインターフェースは自動的に起動されない。
2. ポートインターフェースを起動すると、マスターインターフェースは毎回、自動的に起動される。
3. マスターインターフェースを停止すると、ポートインターフェースも停止される。
4. マスターはポートなしで静的 **IP** 接続を開始できる。
5. マスターはポートなしの場合、**DHCP** 接続の開始時にポートを待機する。
6. **DHCP** 接続でポートを待機中のマスターは、キャリアをとまなう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マスターは、キャリアをとまなわないポートが追加されると待機を継続する。



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ネクションをサポートしますか?](#)』を参照してください。

5.3. ネットワークチーミングとボンディングの比較

表5.1 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	あり	あり
ラウンドロビン Tx ポリシー	あり	あり
アクティブバックアップ Tx ポリシー	あり	あり
LACP (802.3ad) サポート	あり (アクティブのみ)	あり
ハッシュベース Tx ポリシー	あり	あり
ユーザーによるハッシュ機能設定	なし	あり

機能	ボンディング	チーム
Tx 負荷分散サポート (TLB)	あり	あり
LACP ハッシュポート選択	あり	あり
LACP サポートの負荷分散	なし	あり
Ethtool リンク監視	あり	あり
ARP リンク監視	あり	あり
NS/NA (IPv6) リンク監視	なし	あり
ポート アップ/ダウン 遅延	あり	あり
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)	なし	あり
ポートごとの個別リンク監視のセットアップ	なし	あり
複数のリンク監視セットアップ	限定的	あり
ロックなし Tx/Rx パス	なし (rwlock)	あり (RCU)
VLAN サポート	あり	あり
ユーザースペースランタイム制御	限定的	完全
ユーザースペースでの論理	なし	あり
拡張性	困難	容易
モジュラー設計	なし	あり
パフォーマンスオーバーヘッド	低	非常に低い
D-Bus インターフェース	なし	あり
複数デバイススタッキング	あり	あり
LLDP を使った zero config	なし	計画中
NetworkManager サポート	あり	あり

5.4. 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

チームデーモンの **teamd** は、**libteam** を使ってチームドライバーのインスタンス 1 つを制御します。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。チームドライバーは、ネットワークインターフェース、たとえば **team0** をカーネルの他の部分に提示します。チームドライバーのインスタンスが作成したインスタンスには **team0** や **team1** などの名前が文書内で与えられます。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。チーミング方法のすべてに共通な論理は、**teamd** が実行します。ラウンドロビンのような異なる負荷分散やバックアップ方法に固有の機能は、「ランナー」と呼ばれる別のコードユニットによって実行されます。「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** 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その作成時に **teamd** のインスタンスにコンパイルされるので、ランナーはプラグインではありません。必要な際には、**teamd** 用のプラグインとしてコードを作成することは可能です。

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)。設定ファイルで他のリンク監視が指定されていないければ、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先方のハードウェアアドレスの存在を監視します。)
- **nsna_ping** (**IPv6** 近隣検索プロトコルからの近隣アドバタイズと近隣要請を使って近隣のインターフェースの存在を監視します。)

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ることに関する制限はありません。ただし、**lacp** ランナー使用時に推奨されるリンク監視は、**ethtool** のみです。

5.5. 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である **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
```

5.6. ボンドのチーム変換

bond2team ツールを使うと、既存のボンディング設定ファイルをチームの設定ファイルに変換することができます。**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することができます。ファイアウォールルールやエイリアスインターフェース、また元のインター

フェース名に関連づけられているものはすべて、名前変更後に機能しなくなる可能性があります。これは、このツールが **ifcfg** ファイル以外のものは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/ という名前のディレクトリーに新規ファイルが作成されます。ここでの **XXXXXX** は、ランダムな文字列です。新規設定ファイルを作成したら、既存のボンディングファイルをバックアップフォルダに移動し、新規ファイルを **/etc/sysconfig/network-scripts/** ディレクトリーに移動します。

例5.1 ボンドのチーム変換

現行の **bond0** 設定をチーム **ifcfg** に変換するには、**root** で以下のコマンドを実行します。

```
~]# /usr/bin/bond2team --master bond0
```

bond0 という名前が維持されることに留意してください。設定を新たな名前で保存する場合は、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には、**--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)** man ページを参照してください。

例5.2 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

現行の **bond0** 設定をチーム **ifcfg** に変換して、**ifcfg** ファイルへのパスを手動で指定するには、**root** で以下の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には、**--json** オプションを追加します。

例5.3 Bond2team を使ってチーム設定を作成する手順

チーム設定は、**bond2team** ツールにボンディングパラメーター一覧を提供して作成することもできます。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=eth1 \  
primary_reselect-0" --port eth1 --port eth2 --port eth3 --port eth4
```

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。

5.7. 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ースの選択

利用可能なインターフェースを表示するには、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ー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。

5.8. 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ースである **nmtui** を使用してネットワークチームを設定するには、「[テキスト形式のユーザーインターフェース nmtui でネットワークチームを設定する手順](#)」に進みます。

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ムingの設定](#)」に進みます。

チームデーモン **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ースを使用してネットワークチームを設定するには、「[GUI を使ったネットワークチームの作成](#)」に進みます。

5.9. テキスト形式のユーザーインターフェース 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ースツール **nmtui** を使うと、ターミナルのウィンドウでチームing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
```

テキスト形式のインターフェースが表示されます。無効なコマンドの場合は、使用法に関するメッセージがプリントされます。

移動するには矢印キーを使用するか、**Tab** を押して次に進むか **Shift+Tab** を押して前に戻ります。**Enter** を押してオプションを選びます。**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。**追加** を選択すると **新規の接続** 画面が開きます。

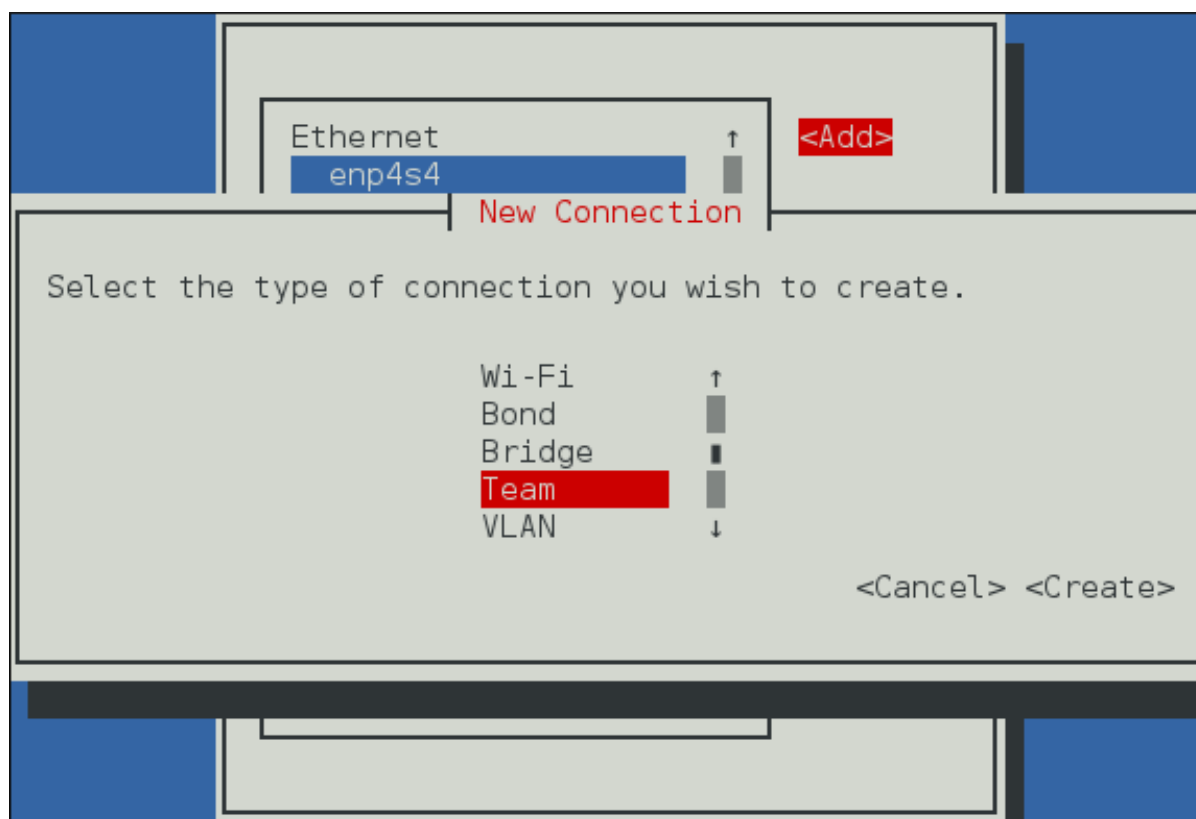


図5.1 NetworkManager テキスト形式のユーザーインターフェースのチーム接続追加メニュー

2. **team** を選択すると、**接続の編集** 画面が開きます。

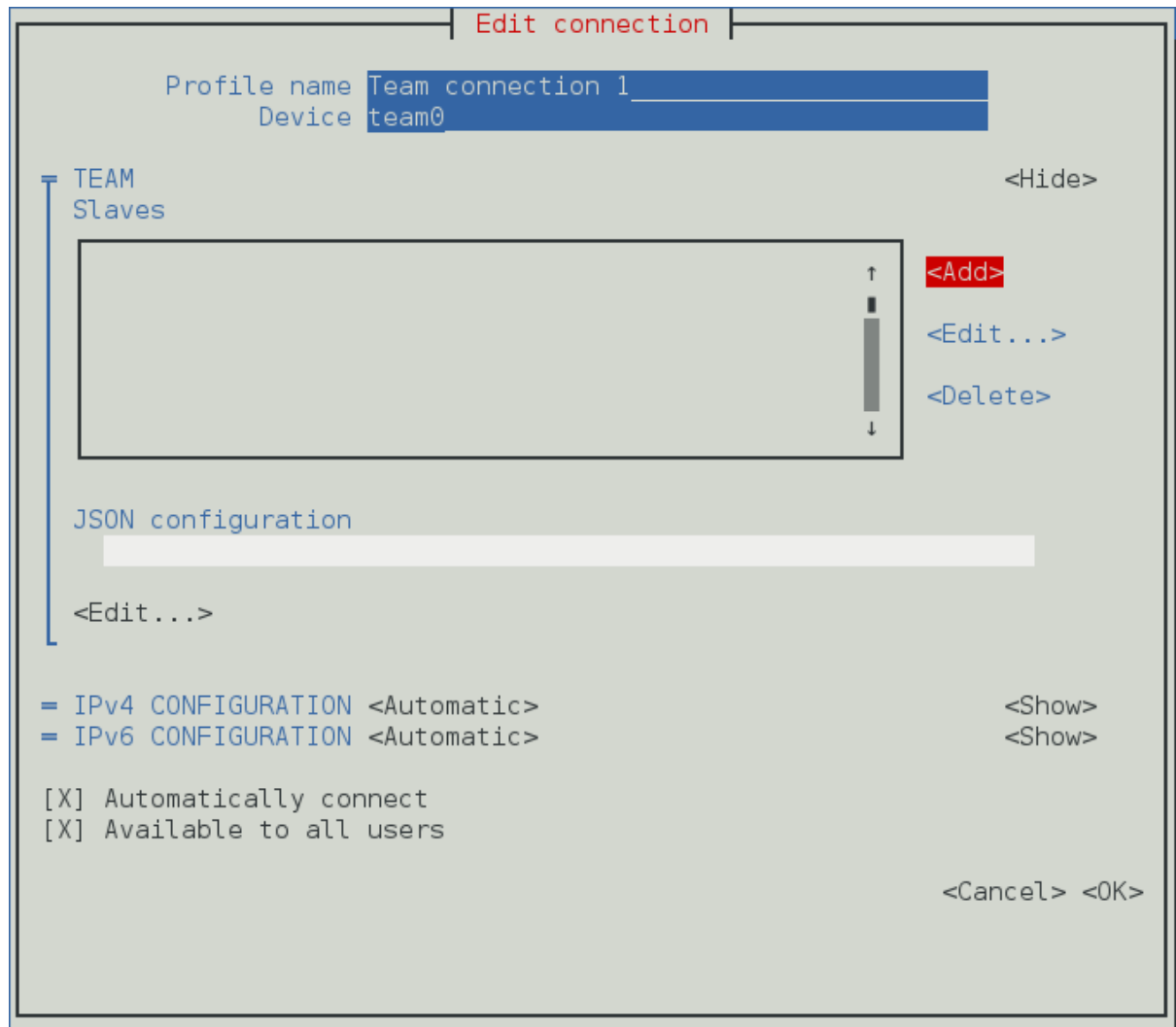


図5.2 NetworkManager テキスト形式ユーザーインターフェースでチーム接続を設定するメニュー

3. チームにポートインターフェースを追加するには **追加** を選択して **新規の接続** 画面を開きます。接続の種類を選んだら、**作成** ボタンを選択して、チームの **接続の編集** 画面を開きます。

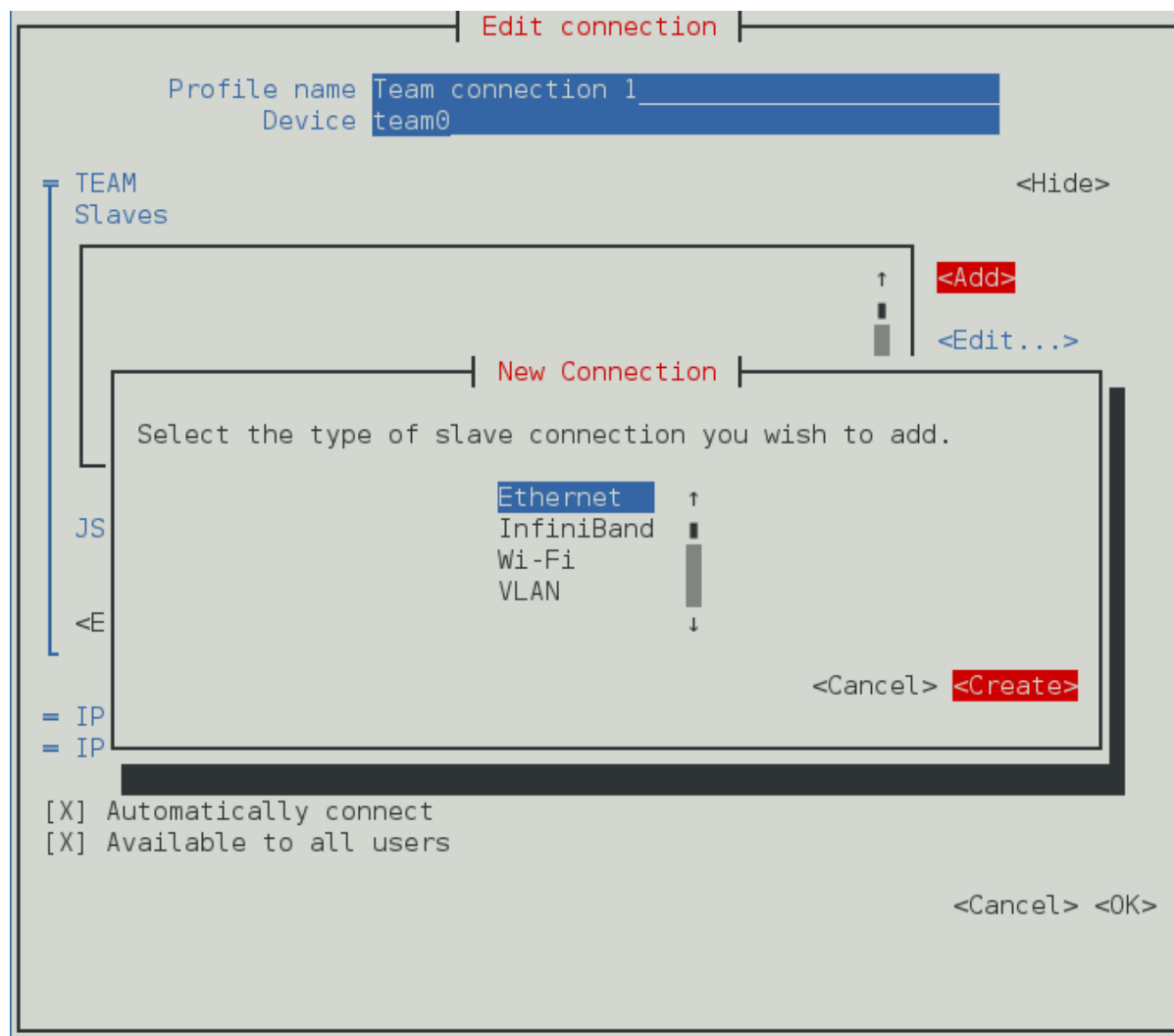


図5.3 NetworkManager テキスト形式ユーザーインターフェースで新規チームポートインターフェース接続を設定するメニュー

4. **デバイス** セクションに希望するスレーブのデバイス名もしくは MAC アドレスを入力します。必要であれば、**イーサネット** ラベルの右側にある **表示する** 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、**接続の編集** ウィンドウがリロードされる際に **デバイス** セクションは自動的に設定されます。ただしこれは、デバイスが正常に発見された場合のみです。

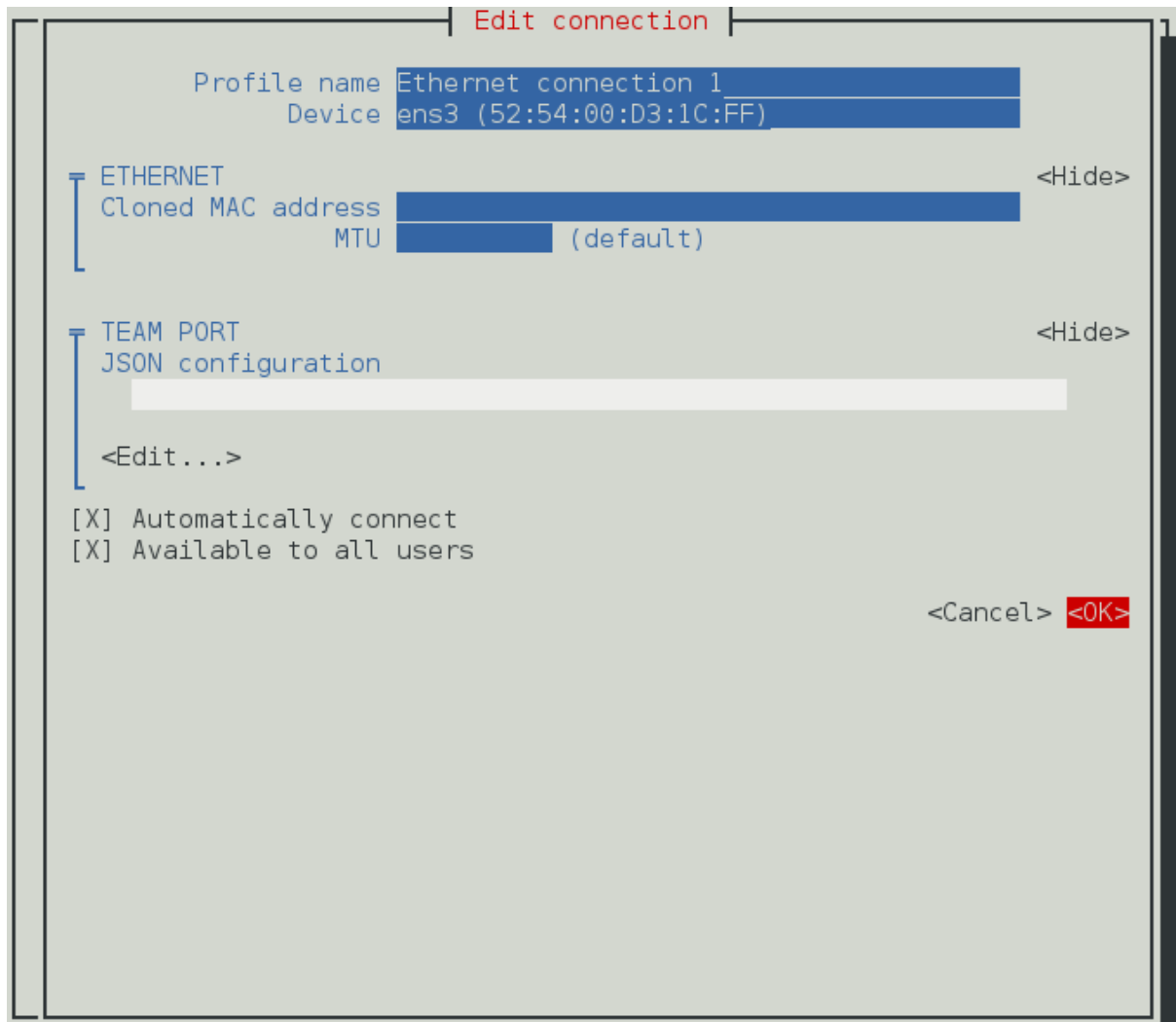


図5.4 NetworkManager テキスト形式ユーザーインターフェースでチームのポートインターフェース接続を設定するメニュー

5. スレーブ セクションにチームのスレーブ名が表示されます。さらにスレーブ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** セクションにある **編集** ボタンを選択します。**vim** コンソールが起動して変更が適用できます。**vim** で変更を保存した後、**JSON 設定** で表示されている JSON 文字列が意図したものになっているか確認します。
7. 設定を確認してから **OK** ボタンを選択します。

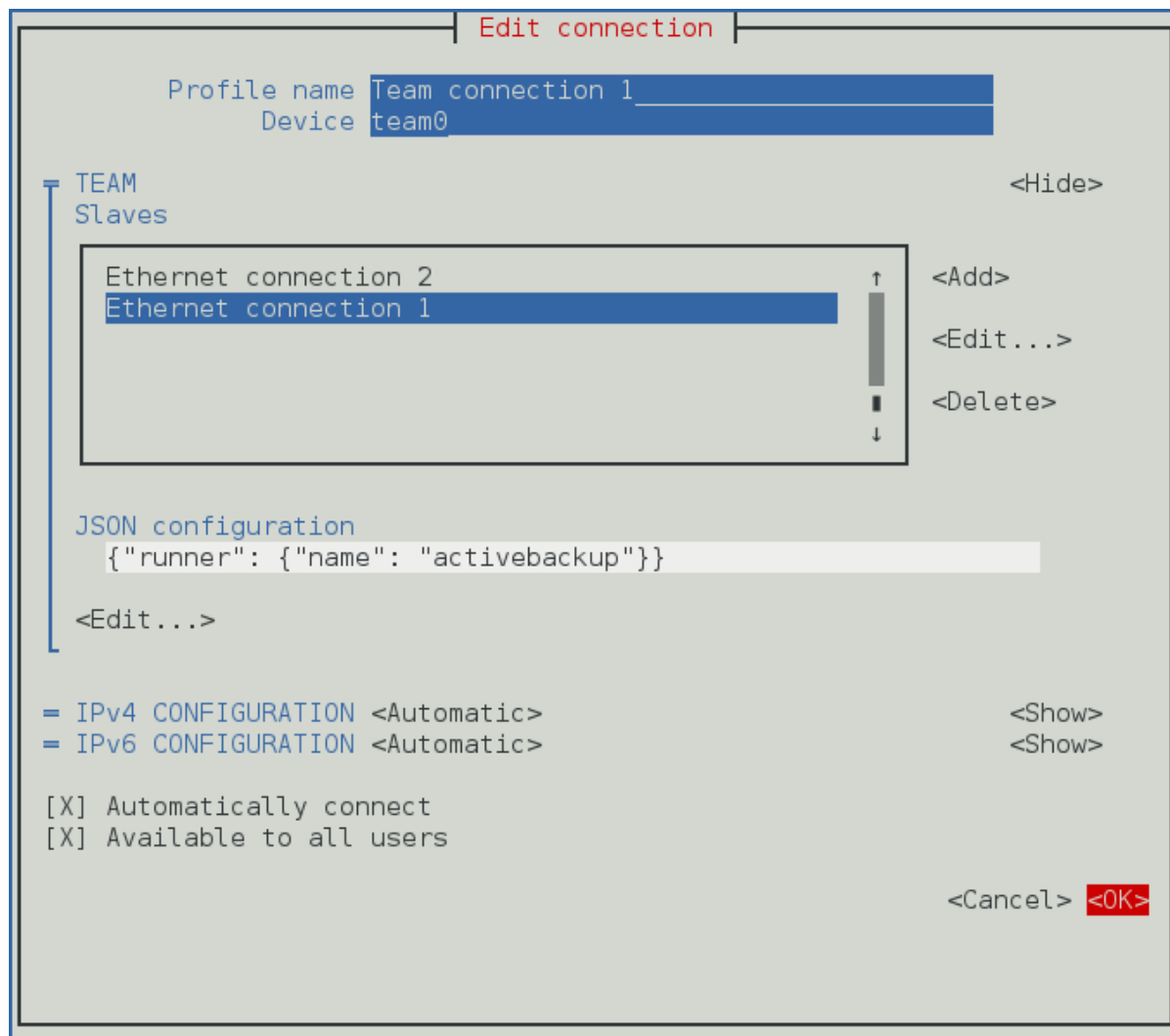


図5.5 NetworkManager テキスト形式ユーザーインターフェースでチーム接続を設定するメニュー

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で使用する文字列の例については、関連するセクションのみを使用するよう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については、「[テキスト形式のユーザーインターフェース \(nmtui\) を使ったネットワーク設定](#)」を参照してください。

5.10. コマンドラインを使用したネットワークチームの設定

5.10.1. nmcli を使用したネットワークチームingの設定

システム上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
eth1    0e8185a1-f0fd-4802-99fb-bedbb31c689b  802-3-ethernet  --
eth0    dfe1f57b-419d-4d1c-aaf5-245deab82487  802-3-ethernet  --
```

システム上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
```

ServerA という名前の新規チームインターフェースを作成するには、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 が内部パラメーター **connection.autoconnect** を **yes** に設定し、**IP** アドレスが与えられないので **ipv4.method** が **auto** に設定されます。また **NetworkManager** は **/etc/sysconfig/network-scripts/ifcfg-team-ServerA** に設定ファイルを書き込みます。ここでは、対応する ONBOOT が **yes** に、BOOTPROTO が **dhcp** に設定されます。

ifcfg ファイルへの手動での変更は、当該インターフェースが次にオンラインになるまで **NetworkManager** には認識されないことに注意してください。設定ファイルの使用方法に関しては、「[sysconfig ファイルを使ったネットワーク設定](#)」を参照してください。

割り当てられた他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[出力は省略されています]
```

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 パラメーターとそれらのデフォルト値についての詳細情報は、**teamd.conf(5)** man ページを参照してください。名前は、インターフェー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。あるいは、以下のように **con-name** オプションを使って名前を指定することもできます。

```
~]$ 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ースを表示するには、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
eth1        0e8185a1-f0fd-4802-99fb-bedbb31c689b  802-3-ethernet
--
eth0        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ている名前を変更するには、以下の形式のコマンドを実行します。

```
nmcli con mod old-team-name connection.id new-team-name
```

既存のチームのチーム設定ファイルを読み込むには、以下の形式のコマンドを実行します。

```
nmcli connection modify team-name team.config JSON-config
```

チーム設定は JSON 文字列として指定するか、設定を含んでいるファイル名を提供することができます。ファイル名にはパスを含めることができます。どちらの場合でも、***team.config*** プロパティに保存されるのは、JSON 文字列です。JSON 文字列の場合、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 プロパティを確認するには、以下の形式のコマンドを実行します。

```
nmcli con show team-name | grep team.config
```

Team0 に *Team0-port1* の名前でインターフェース *eth0* 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 eth0 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ース *eth1* を追加するには、以下のコマンドを実行します。

```
~]$ nmcli con add type team-slave con-name Team0-port2 ifname eth1 master  
Team0  
Connection 'Team0-port2' (a89ccff8-8202-411e-8ca6-2953b7db52dd)  
successfully added.
```

本ガイド執筆時点では、**nmcli** がサポートするのはイーサネットポートのみです。

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ー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 の導入については、「[NetworkManager コマンドラインツール \(nmcli\) の使用](#)」を参照してください。

5.10.2. 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ムing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ースとして作動する仮想インターフェー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、設定ファイルの例をコピーして、**root** 権限でエディターを使用してこれを編集するというものです。利用可能な設定例を一覧表示するには、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_nsn_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
```

ここに含まれるファイル、たとえば **activebackup_ethtool_1.conf** 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": {
    "eth1": {
      "prio": -10,
      "sticky": true
    },
    "eth2": {
      "prio": 100
    }
  }
}
```

teamd 設定ファイルを保存する作業設定ディレクトリを作成します。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することができます。

```
~]$ 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ー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** の使用方法については、**vi(1)** man ページを参照してください。お好みのエディターを使用しても構いません。

チーム内でポートとして使用するインターフェー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。インターフェー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ースはどちらも「UP」になっていることが分かります。

インターフェースをダウンにするには、**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ー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ースを作成するには、**root** ユーザーで作業設定ディレクトリ (この例では *teamd_working_configs*) に移動します。

```
~]# cd /home/user/teamd_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をオンにし、**-f** オプションは読み込む設定ファイルを指定します。**-d** オプションは、起動後に実行するプロセスをデーモン化します。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
```

ネットワークチームインターフェース team0 にアドレスを適用するには、**root** で以下の形式の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ースの 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ースをアクティブ化する、または「up」にするには、**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ースを一時的に非アクティブ化、つまり「down」にするには、**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強制終了、kill、するには、**root** で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 team0 に関連するデーモンのインスタンスを強制終了、kill、することを指定します。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でヘルプを表示するには、以下のコマンドを実行します。

```
~]$ teamd -h
```

また、**teamd(8)** man ページも参照してください。

5.10.3. ifcfg ファイルを使用したネットワークチームの作成

ifcfg ファイルを使ってネットワークチームを作成するには、以下のようなファイルを **/etc/sysconfig/network-scripts/** ディレクトリー内に作成します。

```
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ースが作成されます。つまり、これがマスターになります。

team0 のメンバーとなるポートを作成するには、以下のようなファイルを **/etc/sysconfig/network-scripts/** ディレクトリー内に作成します。

```
DEVICE=eth1
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ー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** で指定されない場合はデフォルトで **0** となり、**-32,767** から **+32,767** までの範囲の値 (プラスとマイナスの両方) を受け付けます。

HWADDR ディレクティブを使ってハードウェアまたは MAC アドレスを指定すると、デバイス命名手順に影響が出ます。これは「[8章 ネットワークデバイス命名における一貫性](#)」で説明しています。

ネットワークチームを有効にするには、**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
```

5.10.4. iputils を使用したネットワークチームへのポートの追加

ip ユーティリティを使ってポート em1 をネットワークチーム team0 に追加するには、**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。

5.10.5. teamnl を使用したチームのポートの一覧表示

teamnl ユーティリティを使ってネットワークチーム内のポートを一覧表示するには、**root** で以下のコマンドを実行します。

```
~]# teamnl team0 ports
em2: up 100 full duplex
em1: up 100 full duplex
```

5.10.6. teamnl を使用したチームのオプション設定

teamnl ユーティリティを使って現在利用可能なオプションすべてを一覧表示するには、**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
```

5.10.7. iputils を使用したネットワークチームへのアドレス追加

ip ユーティリティを使ってアドレスをチーム team0 に追加するには、**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
```

5.10.8. iputils を使用したネットワークチームへのインターフェースの有効化

ip ユーティリティを使ってネットワークチーム team0 へのインターフェースをアクティブ化または「有効」にするには、**root** で以下のコマンドを実行します。

```
~]# ip link set team0 up
```

5.10.9. 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ってネットワークチーム内の **activeport** オプションを一覧表示するには、**root** で以下のコマンドを実行します。

```
~]# teamnl team0 getoption activeport
0
```

5.10.10. 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ってネットワークチーム内の **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
```

5.11. TEAMDCTL を使用した TEAMD の制御

実行中の **teamd** のインスタンスに統計値や設定情報をクエリーする、または変更を加えるには、制御ツール **teamdctl** を使用します。

チーム **team0** の現在のステータスを表示するには、**root** で以下のコマンドを実行します。

```
~]# 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
```

team0 の完全な状態のダンプを JSON 形式 (マシン処理に便利) で表示するには、以下のコマンドを実行します。

```
~]# teamdctl team0 state dump
```

team0 の設定ダンプを JSON 形式で表示するには、以下のコマンドを実行します。

```
~]# teamdctl team0 config dump
```

チーム **team0** の一部であるポート **em1** の設定を表示するには、以下のコマンドを実行します。

```
~]# teamdctl team0 port config dump em1
```

5.11.1. ネットワークチームへのポートの追加

ポート **em1** をネットワークチーム **team0** に追加するには、**root** で以下のコマンドを実行します。

```
~]# teamdctl team0 port add em1
```



重要

teamdctl を直接使用してポートをスレーブとして設定する場合は、スレーブポートは **down** に設定する必要があります。そうでない場合には **teamdctl team0 port add em1** コマンドは失敗してしまいます。

5.11.2. ネットワークチームからのポートの削除

ポート **em1** をネットワークチーム **team0** から削除するには、**root** で以下のコマンドを実行します。

```
~]# teamdctl team0 port remove em1
```

5.11.3. ネットワークチーム内のポートへの設定適用

ネットワークチーム **team0** 内のポート **em1** に JSON 形式の設定を適用するには、**root** で以下の形式のコマンドを実行します。

```
~]# teamdctl team0 port config update em1 JSON-config-string
```

ここでの *JSON-config-string* は、JSON 形式での文字列による設定になります。これにより、提供された JSON 形式の文字列を使用しているポートの設定が更新されます。ポートを設定する有効な JSON 文字列の例は、以下のとおりです。

```
{
  "prio": -10,
  "sticky": true
}
```

JSON 設定文字列を単一引用符で囲み、改行は省略します。

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の制御ツールコマンド例については、**teamdctl(8)** man ページを参照してください。

5.11.4. ネットワークチーム内のポート設定表示

ネットワークチーム **team0** 内のポート **em1** の設定をコピーするには、**root** で以下のコマンドを実行します。

```
~]# 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。

5.12. 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、「[ネットワークチームingデーモンおよび「ランナー」について](#)」を参照してください。

5.12.1. 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": {}}
}
```

詳細情報は、**teamd.conf(5)** man ページを参照してください。

5.12.2. 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": {}}
}
```

詳細情報は、**teamd.conf(5)** man ページを参照してください。

5.12.3. 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。

詳細情報は、**teamd.conf(5)** man ページを参照してください。

5.12.4. 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することができます。以下のいずれかの例を JSON 形式の設定ファイルに追加することができます。

```
{
  "device": "team0",
  "runner": {
    "name": "activebackup"
  },
  "link_watch": {
    "name": "ethtool"
  },
}
```

```

    "ports": {
      "em1": {
        "prio": -10,
        "sticky": true
      },
      "em2": {
        "prio": 100
      }
    }
  }
}

```

上の設定例では、**ethtool** の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 の優先度が高くなります。sticky フラグにより、em1 がアクティブになると、リンクが有効な間はずっと、このポートはアクティブのままになります。

```

{
  "device": "team0",
  "runner": {
    "name": "activebackup"
  },
  "link_watch": {
    "name": "ethtool"
  },
  "ports": {
    "em1": {
      "prio": -10,
      "sticky": true,
      "queue_id": 4
    },
    "em2": {
      "prio": 100
    }
  }
}

```

上の設定例では、queue ID 4 が追加されます。**ethtool** の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 の優先度が高くなりますが、sticky フラグにより、em1 がアクティブになると、リンクが有効な間はずっと、このポートはアクティブのままになります。

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
    }
}
}

```

上の設定例では、**ethtool** の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 の優先度が高くなります。sticky フラグにより、em1 がアクティブになると、リンクが有効な間はずっと、このポートはアクティブのままになります。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。

詳細情報は、**teamd.conf(5)** man ページを参照してください。

5.12.5. 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。アクティブ文字列では、最近のトラフィックの統計値を使っ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。静的モードでは、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。トラフィックのボリュームが大きい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配される複数のストリームで構成されるため、この方法が好まれます。この方法により、**teamd** が介入することなく負荷共有が実施されます。

パッシブ送信 (Tx) 負荷分散機能向け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) 負荷分散機能向け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

詳細情報は、**teamd.conf(5)** man ページを参照してください。

5.12.6. 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** を使ってリンクのステータスを監視します。**ethtool** 以外のリンク監視方法は意味がありません。これはたとえば **arp_ping** の場合、リンクがアップにならないためです。この理由は、リンクは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** はリンク層を個別に監視するため、リンクが確立されていないために認識されないという事態を防ぎます。

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"
}
```

詳細情報は、**teamd.conf(5)** man ページを参照してください。

5.12.7. 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。これらのいずれかを実装するには、**root** でエディターを使用して、JSON 形式の文字列をチームの JSON 形式設定ファイルに追加します。

5.12.7.1. リンクのステータス監視用の Ethtool 設定

リンクがアップ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_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
  }

```

5.12.7.2. 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 要求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様ですが、このユーティリティーは使用しません。

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、実行されなかった返信 (たとえば ARP 返信) の最大許容数です。これは **interval** オプションとともに選択し、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 em2 用に新規設定を読み込むには、**root** で以下のコマンドを実行します。

```

~]# 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の制御ツールコマンド例については、**teamdctl(8)** man ページを参照してください。

5.12.7.3. 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ると報告されます。**missed_max** オプションは、実行されなかった返信 (たとえば ARP 返信) の最大許容数です。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** ターゲットアドレスを解決するホスト名を設定するには、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",
  "target_host": "MyStorage"
}

```

「target_host」オプションには、**IPv6** アドレスに変換され、NS/NA パケットのターゲットアドレスとして使用されるホスト名が含まれます。**IPv6** アドレスは、ホスト名の代わりとして使用することができます。

詳細情報は、**teamd.conf(5)** man ページを参照してください。

5.12.8. 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、選択されたチームモード (**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ースに向けて、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。キューの数を増減したい場合は、Netlink 属性 **tx_queues** を使って、チームドライバーのインスタンス作成中にこの値を変更することができます。

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** ユーティリティと合わせて使うとマルチキューのキュー規範を設定することができ、特定のポートデバイス上で特定のトラフィックが送信されるようにフィルターをかけることができます。たとえば、上記の設定を使用して **192.168.1.100** にバインドされたトラフィックすべてが出力デバイスとしてチームの **eth1** を使用するように強制するには、以下の形式のコマンドを **root** で実行します。

```
~]# 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。

5.12.9. BPF ベースの Tx ポートセレクトアーの設定

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 バリエーションになります。つまり、多くの異なる ソケットバッファ (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 MAC アドレスを使用。
- **vlan**: VLAN ID を使用。
- **ipv4**: ソースおよび宛先 **IPv4** アドレスを使用。
- **ipv6**: ソースおよび宛先 **IPv6** アドレスを使用。
- **ip**: **IPv4** と **IPv6** のソースおよび宛先アドレスを使用。
- **l3**: **IPv4** と **IPv6** のソースおよび宛先アドレスを使用。
- **tcp**: ソースおよび宛先 **TCP** ポートを使用。

- **udp**: ソースおよび宛先 **UDP** ポートを使用。
- **sctp**: ソースおよび宛先 **SCTP** ポートを使用。
- **14**: **TCP**、**UDP** および **SCTP** のソースおよび宛先ポートを使用。

これらの文字列は、以下の形式で負荷分散ランナーに行を追加して使用できます。

```
"tx_hash": ["eth", "ipv4", "ipv6"]
```

例については、「[負荷分散ランナーの設定](#)」を参照してください。

5.13. GUI を使ったネットワークチームの作成

5.13.1. チーム接続の確立

nm-connection-editor を使って、**NetworkManager** に 2 つ以上の有線もしくは InfiniBand 接続からチームを作成するよう指示することができます。接続が最初にチームing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。この設定プロセスを完了するには、利用可能なインターフェースの MAC アドレスが必要です。

手順5.1 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. ターミナルで **nm-connection-editor** と入力します。

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

2. **追加** ボタンをクリックします。**接続の種類を選択** ウィンドウが表示されます。**チーム** を選択して **作成** 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。

Editing Team connection 1

Connection name:

General **Team** IPv4 Settings IPv6 Settings

Interface name:

MTU: bytes

Teamed connections:

図5.6 NetworkManager グラフィカルユーザーインターフェースの Team 追加メニュー

3. **Team** タブで **追加** をクリックし、Team 接続で使用するインターフェースのタイプを選択します。**作成** 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。

Editing team1 slave 1

Connection name: team1 slave 1

General Ethernet 802.1X Security DCB Team Port

Device: [dropdown]

Cloned MAC address: [text]

MTU: automatic [minus] [plus] bytes

Wake on LAN: ☒ Default ☐ Phy ☐ Unicast ☐ Multicast
☐ Ignore ☐ Broadcast ☐ Arp ☐ Magic

Wake on LAN password: [text]

Cancel Save

図5.7 NetworkManager グラフィカルユーザーインターフェースのスレーブ接続追加

5. カスタムのポート設定を適用する場合は、**Team** ポート 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. **保存** ボタンをクリックします。
7. チームingされたポート名が **Team 接続** ウィンドウに表示されます。さらにポート接続を追加するには、**追加** ボタンをクリックしてします。
8. 設定を確認してから **保存** ボタンをクリックします。
9. チーム固有の設定については、「[チームタブの設定](#)」を参照してください。

手順5.2 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. ターミナルで **nm-connection-editor** と入力します。

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

2. 編集する接続を選択して、**編集** ボタンをクリックします。
3. **全般** タブを選択します。
4. **編集** ダイアログの **全般** タブにある 5 つの設定は、ほとんどの接続の種類で共通のものです。

- **接続名**: ネットワーク接続の名前を入力します。この名前は **Network** ウィンドウメニューの接続名一覧に表示されます。
- **この接続が利用可能になったときは自動的に接続する**: このボックスにチェックを入れると、この接続が利用可能な時に **NetworkManager** が自動接続します。詳細については「[ネットワークへの自動接続](#)」を参照してください。
- **全ユーザーがこのネットワークに接続可能とする**: システム上のすべてのユーザーが接続可能とするには、このボックスにチェックを入れます。この設定を変更するには root 権限が必要になることがあります。詳細については「[システムワイドおよびプライベート接続プロファイル](#)」を参照してください。
- **この接続を使用したときは自動的に VPN に接続する**: このボックスにチェックを入れると、この接続が利用可能な時に **NetworkManager** が自動で選択された VPN に接続します。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 セキュリティガイド](#)』を参照してください。

5. チーム固有の設定については、「[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、以下のいずれかを設定します。

- **IPv4** の設定は、**IPv4 の設定** タブをクリックして「[IPv4 セットアップの設定](#)」に進みます。
- または
- **IPv6** の設定は、**IPv6 の設定** タブをクリックして「[IPv6 セットアップの設定](#)」に進みます。

5.13.1.1. 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することができます。**保存** をクリックすると、JSON 設定がチームインターフェースに適用されます。

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については、[手順5.1「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。

5.14. その他のリソース

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** 設定ファイルについて説明しています。

- **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について説明しています。

オンラインのドキュメント

http://www.w3schools.com/js/js_json_syntax.asp

JSON 構文についての説明です。

第6章 ネットワークブリッジの設定

ネットワークブリッジは、ネットワーク間のトラフィックを MAC アドレスに基づいて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。

6.1. テキスト形式のユーザーインターフェース **NMTUI** によるブリッジの設定

テキスト形式のユーザーインターフェースツール **nmtui** を使うと、ターミナルのウィンドウでブリッジを設定できます。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
```

テキスト形式のインターフェースが表示されます。無効なコマンドの場合は、使用法に関するメッセージがプリントされます。

移動するには矢印キーを使用するか、**Tab** を押して次に進むか **Shift+Tab** を押して前に戻ります。**Enter** を押してオプションを選びます。**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。**追加** を選択すると **新規の接続** 画面が開きます。

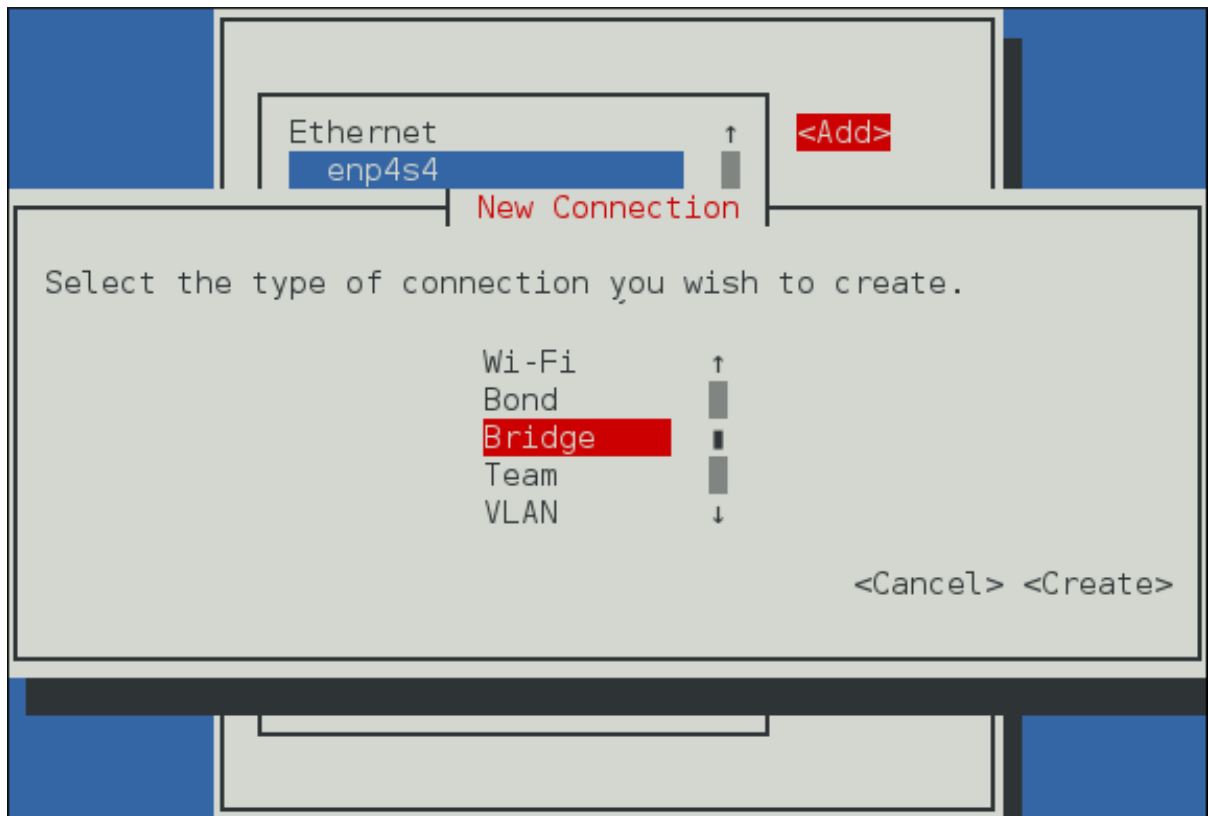


図6.1 NetworkManager テキスト形式のユーザーインターフェースのブリッジ接続追加メニュー

2. **ブリッジ** を選択すると、**接続の編集** 画面が開きます。
3. ブリッジにスレーブインターフェースを追加するには **追加** を選択して **新規の接続** 画面を開きます。接続の種類を選んだら、**作成** ボタンを選択して、ブリッジの **接続の編集** 画面を開きます。

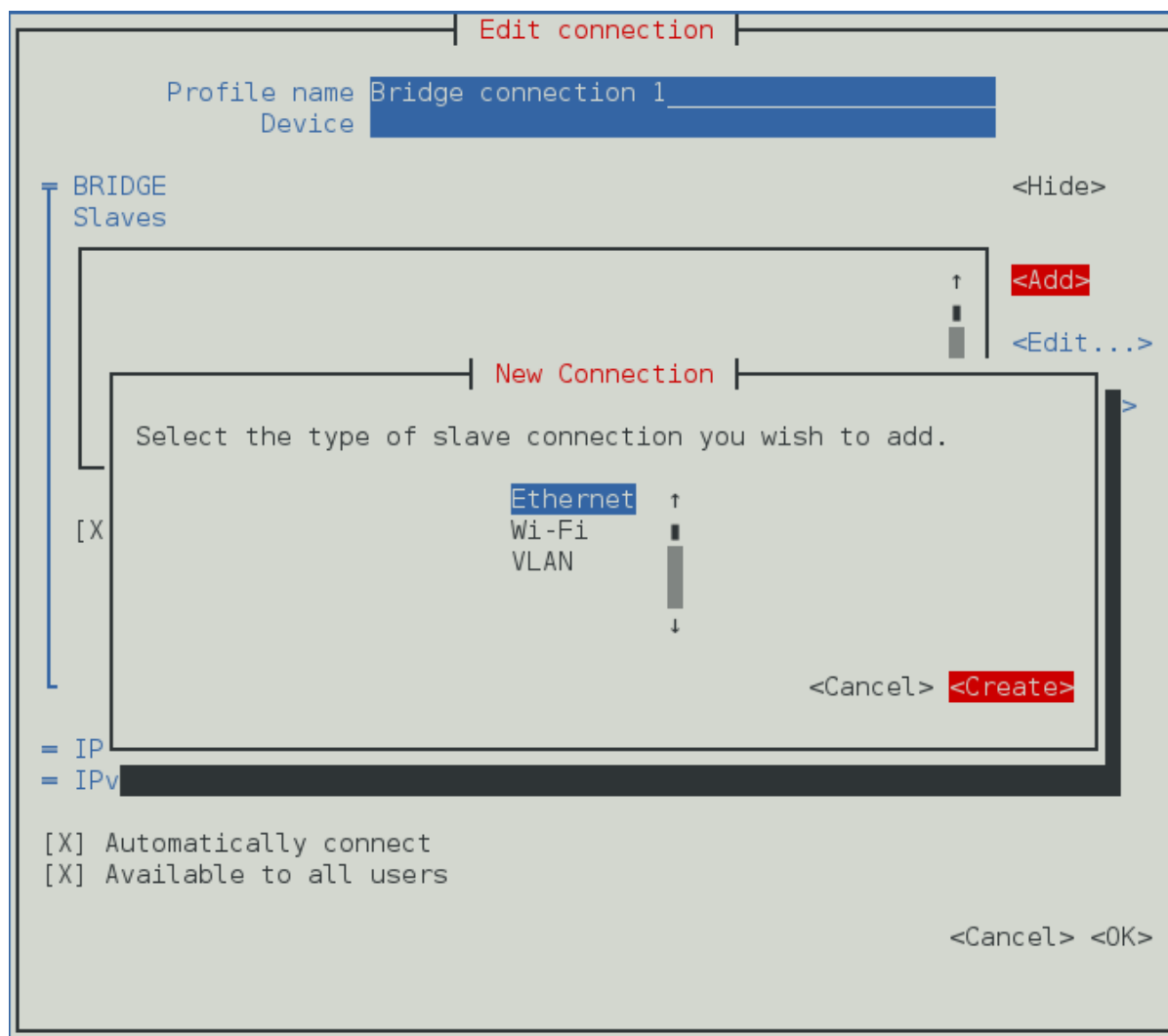


図6.2 NetworkManager テキスト形式ユーザーインターフェースで新規ブリッジスレーブを設定するメニュー

4. **デバイス** セクションに希望するスレーブのデバイス名もしくは MAC アドレスを入力します。必要であれば、**イーサネット** ラベルの右側にある **表示する** 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、**接続の編集** ウィンドウがリロードされる際に **デバイス** セクションは自動的に設定されます。ただしこれは、デバイスが正常に発見された場合のみです。

Edit connection

Profile name Ethernet connection 1

Device ens3

ETHERNET <Hide>

Cloned MAC address

MTU (default)

BRIDGE PORT <Hide>

Priority 32

Path cost 100

[] Hairpin mode

[X] Automatically connect

[X] Available to all users

<Cancel> <OK>

図6.3 NetworkManager テキスト形式ユーザーインターフェースでブリッジスレーブ接続を設定するメニュー

5. スレーブ セクションにブリッジのスレーブ名が表示されます。さらにスレーブ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。

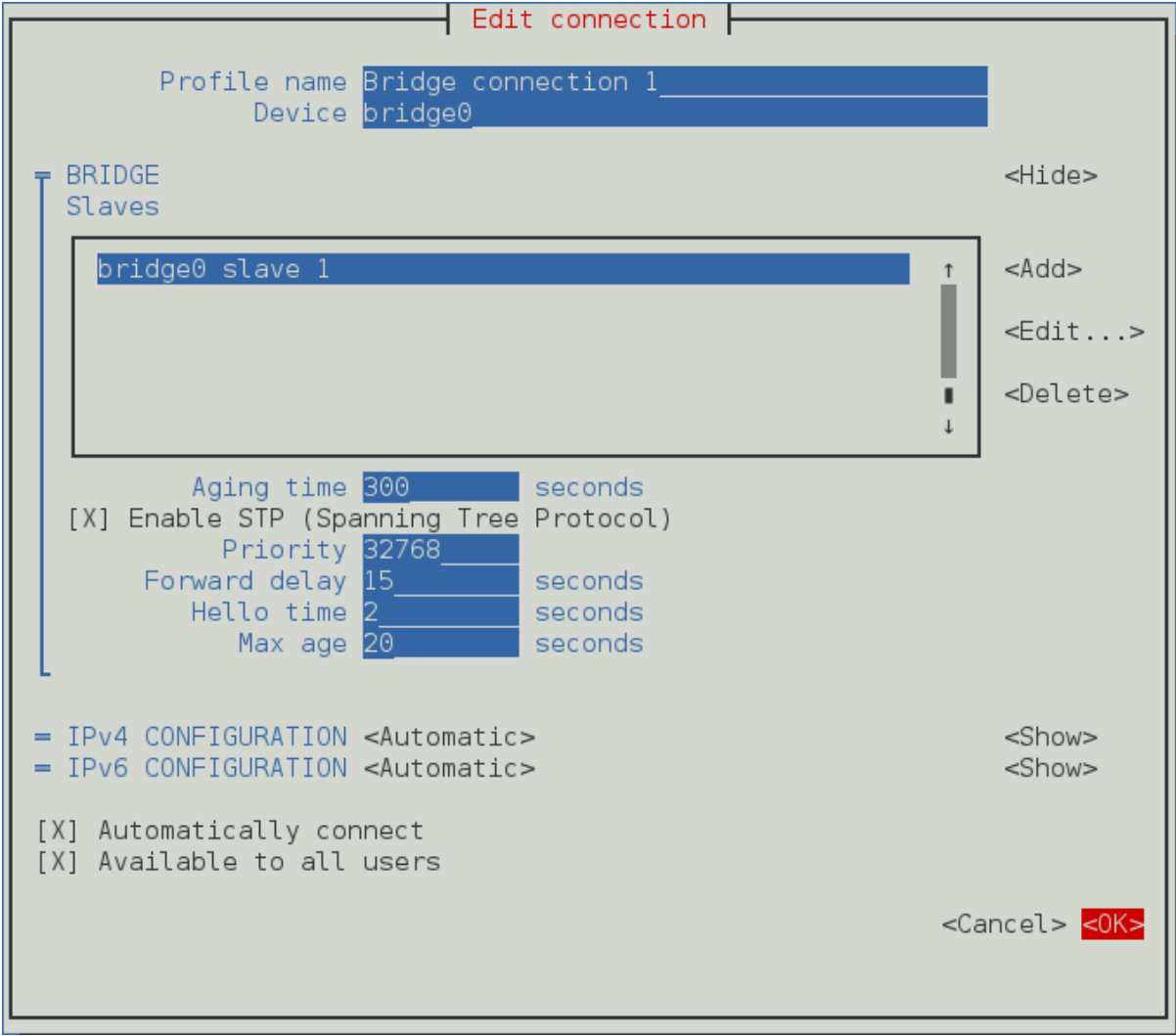


図6.4 NetworkManager テキスト形式ユーザーインターフェースでブリッジを設定するメニュー

ブリッジ用語の定義については、[「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については、[「テキスト形式のユーザーインターフェース \(nmtui\) を使ったネットワーク設定」](#)を参照してください。

6.2. NETWORKMANAGER のコマンドラインツール NMCLI の使用

bridge-br0 という名前のブリッジを作成するには、**root** で以下のコマンドを実行します。



```
~]# nmcli con add type bridge ifname br0
Connection 'bridge-br0' (6ad5bba6-98a0-4f20-839d-c997ba7668ad)
successfully added.
```

インターフェース名が指定されない場合は、デフォルトの bridge、bridge-1、bridge-2 といった名前が付けられます。

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
eth0 4d5c449a-a6c5-451c-8206-3c9a4ec88bca 802-3-ethernet eth0
```

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。このブリッジでの **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
```

このブリッジで再度 **802.1D STP** を有効にするには、**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 **32768** (デフォルト) よりも **28672** のブリッジが root ブリッジに選ばれます。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.
```

0 から **65535** までの値が使用できます。

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
```

0 から **65535** までの値が使用できます。

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 の他のオプションについては、**nmcli(1)** man ページのブリッジセクションに記載されています。

たとえばインターフェース **eth1** を **bridge-br0** に追加もしくはスレーブ化するには、以下のコマンドを実行します。

```
~]$ nmcli con add type ethernet ifname eth1 master bridge-br0
Connection 'bridge-slave-eth1' (70ffae80-7428-4d9c-8cbd-2e35de72476e)
successfully added.
```

本ガイド執筆時点では、**nmcli** がサポートするのはイーサネットのスレーブのみです。

インタラクティブモードを使っ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
```

これで **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 の導入については、「[NetworkManager コマンドラインツール \(nmcli\) の使用](#)」を参照してください。

6.3. コマンドラインインターフェース (CLI) の使用

6.3.1. 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。必要な場合は、**root** で以下のコマンドを実行して、モジュールを読み込みます。

```
~]# 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)** man ページを参照してください。

6.3.2. 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-brN** という名前のファイルを作成し、**N** をそのインターフェースの番号 **0** などに置き換えます。

ファイルのコンテンツは、イーサネットインターフェースなどブリッジされるインターフェー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** ディレクティブはインターフェース名を **brN** の形式で引数として与えられ、この **N** はインターフェースの番号に置き換えられます。
- **TYPE** ディレクティブには、引数 **Bridge** が与えられます。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ース設定ファイルには **IP** アドレスが与えられますが、物理インターフェース設定ファイルには **MAC** アドレスのみとします。(下記参照)
- 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6.1 ifcfg-br0 インターフェース設定ファイルの例

以下の例では、静的 **IP** アドレスを使ったブリッジインターフェースの設定ファイルを示しています。

```
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、別のインターフェースを作成するか既存のインターフェースを修正して、これをブリッジインターフェース向けます。

例6.2 ifcfg-ethX インターフェー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ース設定ファイルをブリッジインターフェースに向けたものです。`/etc/sysconfig/network-scripts/ifcfg-ethX` の物理インターフェースを設定します。ここでの *X* は、特定のインターフェースに対応する一意の番号です。

```
DEVICE=ethX
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。名前が指定されない場合は、**NetworkManager** プラグインである **ifcfg-rh** が、「Type Interface」という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、ブリッジが **Bridge br0** と命名されます。別の方法では、**NAME=bridge-br0** が **ifcfg-br0** ファイルに追加されると、接続プロファイル名は **bridge-br0** になります。

注記

DEVICE ディレクティブでは、インターフェース名はデバイスタイプを決定するわけではないので、ほぼいかなる名前でも使用できます。**TYPE=Ethernet** は、絶対に必要なわけではありません。**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ー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ってハードウェアまたは MAC アドレスを指定すると、デバイス命名手順に影響が出ます。これは「[8章 ネットワークデバイス命名における一貫性](#)」で説明しています。



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ースを有効にするには、**root** で以下の形式のコマンドを実行します。

ifup device

このコマンドは **NetworkManager** が稼働中かどうかを検出し、**nmcli con load UUID** を呼び出した後に **nmcli con up UUID** を呼び出します。

別の方法ではすべてのインターフェースをリロードします。**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てから再度、これを開始します。その後、**ONBOOT=yes** となっている ifcfg ファイルで **ifup** を呼び出します。



注記

デフォルトでは **NetworkManager** は、インターフェースが次回にアップになるまで ifcfg ファイルへの変更を認識せず、変更前の設定データの使用を継続します。これは **NetworkManager.conf** ファイルの **monitor-connection-files** オプションで設定します。詳細情報は、**NetworkManager.conf(5)** man ページを参照してください。

6.3.3. 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ースで形成されたネットワークブリッジの例を示します。仮想化環境では、この用例も一般的です。ボンディングされたインターフェースの設定ファイルに詳しくない場合は、[「チャンネルボンディングインターフェースの作成」](#)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ー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=ethX
TYPE=Ethernet
SLAVE=yes
MASTER=bond0
BOOTPROTO=none
HWADDR=AA:BB:CC:DD:EE:FF
```



注記

インターフェース名には **ethX** が使われることが多いですが、ほぼどんな名前を使用しても構いません。

インターフェース設定ファイル **/etc/sysconfig/network-scripts/ifcfg-bond0** を以下のように作成もしくは編集します。

```
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一覧については、「[チャンネルボンディングの使用](#)」を参照してください。

インターフェース設定ファイル **/etc/sysconfig/network-scripts/ifcfg-brbond0** を以下のように作成もしくは編集します。

```
DEVICE=brbond0
ONBOOT=yes
TYPE=Bridge
IPADDR=192.168.1.1
PREFIX=24
```

これで **MASTER=bond0** ディレクティブのある 2 つ以上のインターフェース設定ファイルができました。これらのファイルは、**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に向けられています。この **ifcfg-bond0** は、**/etc/sysconfig/network-scripts/ifcfg-brbond0** 設定ファイルに向けられています。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ースとして動作します。

新規または最近設定したインターフェースを有効にするには、**root** で以下の形式のコマンドを実行します。

```
ifup device
```

このコマンドは **NetworkManager** が稼働中かどうかを検出し、**nmcli con load UUID** を呼び出した後に **nmcli con up UUID** を呼び出します。

別の方法ではすべてのインターフェースをリロードします。**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てから再度、これを開始します。その後、**ONBOOT=yes** となっている **ifcfg** ファイルで **ifup** を呼び出します。



注記

デフォルトでは **NetworkManager** は、インターフェースが次回にアップになるまで **ifcfg** ファイルへの変更を認識せず、変更前の設定データの使用を続けます。これは **NetworkManager.conf** ファイルの **monitor-connection-files** オプションで設定します。詳細情報は、**NetworkManager.conf(5)** man ページを参照してください。

6.4. 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ースを開始する際に、**NetworkManager** は少なくとも 1 つのポートが「転送」状態に入るまでは、**DHCP** や **IPv6** といったネットワーク依存の **IP** 設定を開始しません。スレーブやポートが接続されたり、パケット転送を開始する前に、静的 **IP** アドレス指定の開始が許可されます。

6.4.1. ブリッジ接続の確立

手順6.1 nm-connection-editor を使用して新規ブリッジ接続を追加する

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. ターミナルで **nm-connection-editor** と入力します。

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

2. **追加** ボタンをクリックします。**接続の種類を選択** ウィンドウが表示されます。**ブリッジ** を選択して **作成** 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。

Editing Bridge connection 1

Connection name:

General **Bridge** IPv4 Settings IPv6 Settings

Interface name:

Bridged connections:

Aging time: s

☒ Enable STP (Spanning Tree Protocol)

Priority:

Forward delay: s

Hello time: s

Max age: s

図6.5 ブリッジ接続 1 の編集

- 手順6.3「ブリッジにスレーブインターフェースを追加する」を参照して、スレーブデバイスを追加します。

手順6.2 既存のブリッジ接続を編集する

- ターミナルで **nm-connection-editor** と入力します。

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

- 編集する **ブリッジ** 接続を選択します。

3. **編集** ボタンをクリックします。

接続名、自動接続の動作、可用性セッティングの設定

編集 ダイアログの **全般** タブにある 5 つの設定は、すべての接続の種類で共通のものです。

- **接続名**: ネットワーク接続の名前を入力します。この名前は **Network** ウィンドウメニューの接続名一覧に表示されます。
- **この接続が利用可能になったときは自動的に接続する**: このボックスにチェックを入れると、この接続が利用可能な時に **NetworkManager** が自動接続します。詳細については「[ネットワークへの自動接続](#)」を参照してください。
- **全ユーザーがこのネットワークに接続可能とする**: システム上のすべてのユーザーが接続可能とするには、このボックスにチェックを入れます。このセッティングを変更するには root 権限が必要になることがあります。詳細については「[システムワイドおよびプライベート接続プロファイル](#)」を参照してください。
- **この接続を使用したときは自動的に VPN に接続する**: このボックスにチェックを入れると、この接続が利用可能な時に **NetworkManager** が自動で選択された VPN に接続します。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 セキュリティガイド](#)』を参照してください。

6.4.1.1. ブリッジタブを設定する

インターフェース名

ブリッジへのインターフェース名。

ブリッジ接続

1 つ以上のスレーブインターフェース。

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)。

STP (スパニングツリープロトコル) を有効化

必要に応じてこのチェックボックスを選択し、**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。

手順6.3 ブリッジにスレーブインターフェースを追加する

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 **ブリッジ** タブを選択します。必要な場合は、[手順6.2「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. **追加** ボタンをクリックして **接続の種類を選択** メニューを表示します。
3. リストから作成する接続の種類を選択します。**作成** をクリックすると、選択した接続の種類に合わせたウィンドウが開きます。

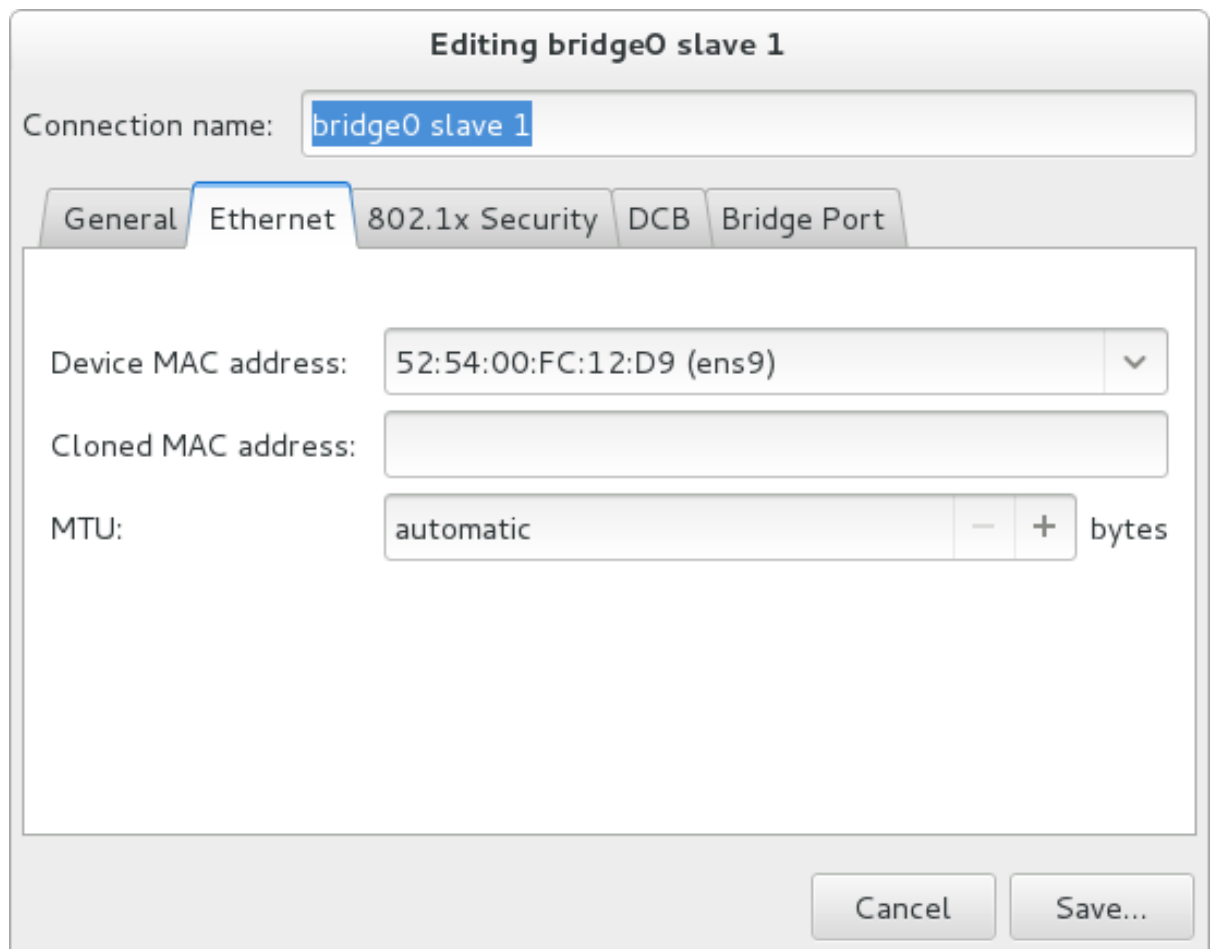


図6.6 NetworkManager グラフィカルユーザーインターフェースのブリッジ接続追加

4. **ブリッジポート** タブを選択します。必要に応じて **優先度** と **経路のコスト** 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。通常は 0 から 255 ままで設定できますが、Linux で利用可能なのは 0 から 63 までです。このケースでのデフォルト値は 32 です。

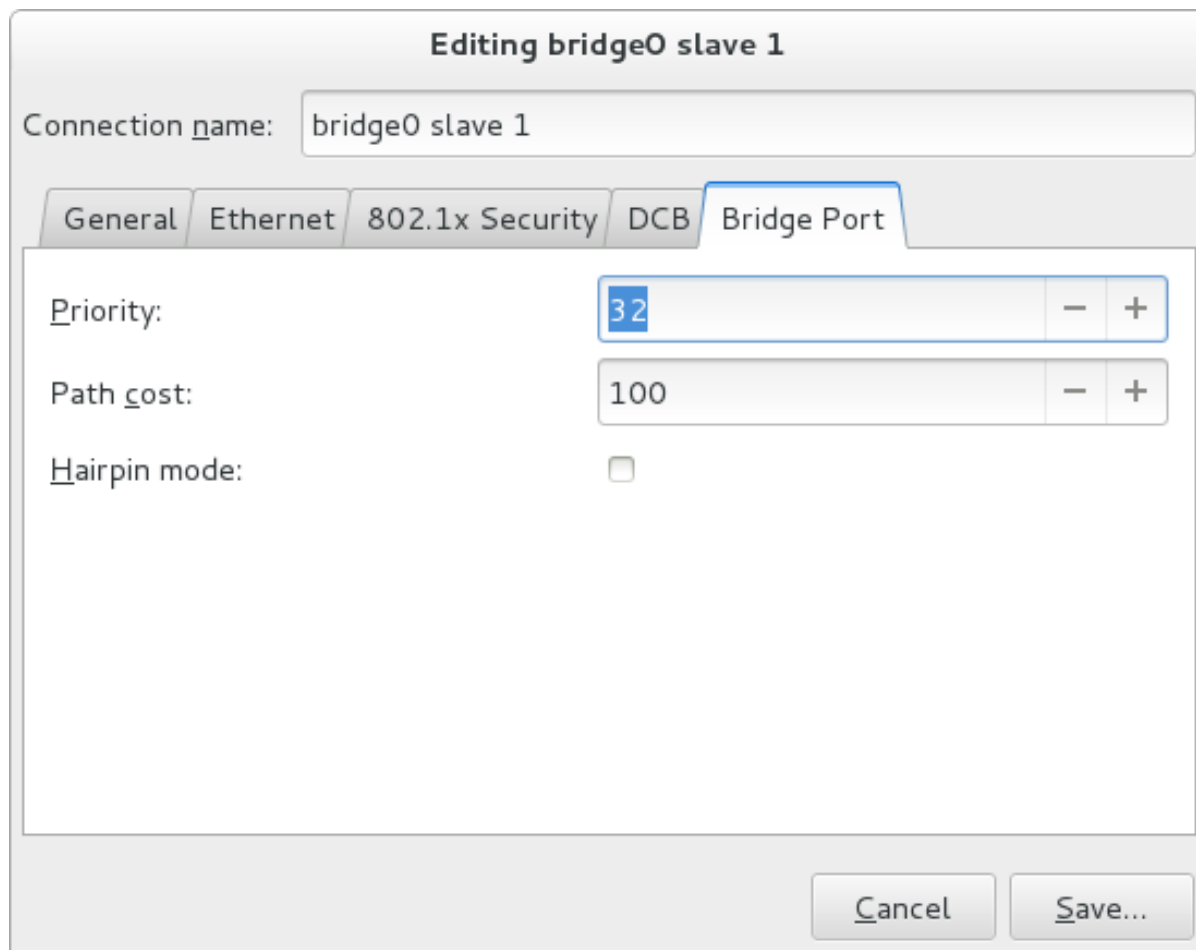


図6.7 NetworkManager グラフィカルユーザーインターフェースでのブリッジポートタブ

5. 必要の場合は **ヘアピンモード** のチェックボックスを選択し、外部処理用に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。

- イーサネットスレーブの場合は **Ethernet** タブをクリックして「[基本設定オプション](#)」に進みます。
- ボンドスレーブの場合は **Bond** タブをクリックして「[Bond タブの設定](#)」に進みます。
- チームスレーブの場合は **Team** タブをクリックして「[チームタブの設定](#)」に進みます。
- VLAN スレーブの場合は **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

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。編集中に該当プロファイルが使用されていた場合、接続を切断してから再接続し、**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、「[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をするには、**ネットワーク接続** ウィンドウ内でその接続を選択し、**オプション** 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。

- **IPv4** の設定は、**IPv4** のセッティング タブをクリックして「IPv4 セッティングの設定」に進みます。
- **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、ブリッジはスレーブとともにネットワーク設定ツール画面に表示されます。

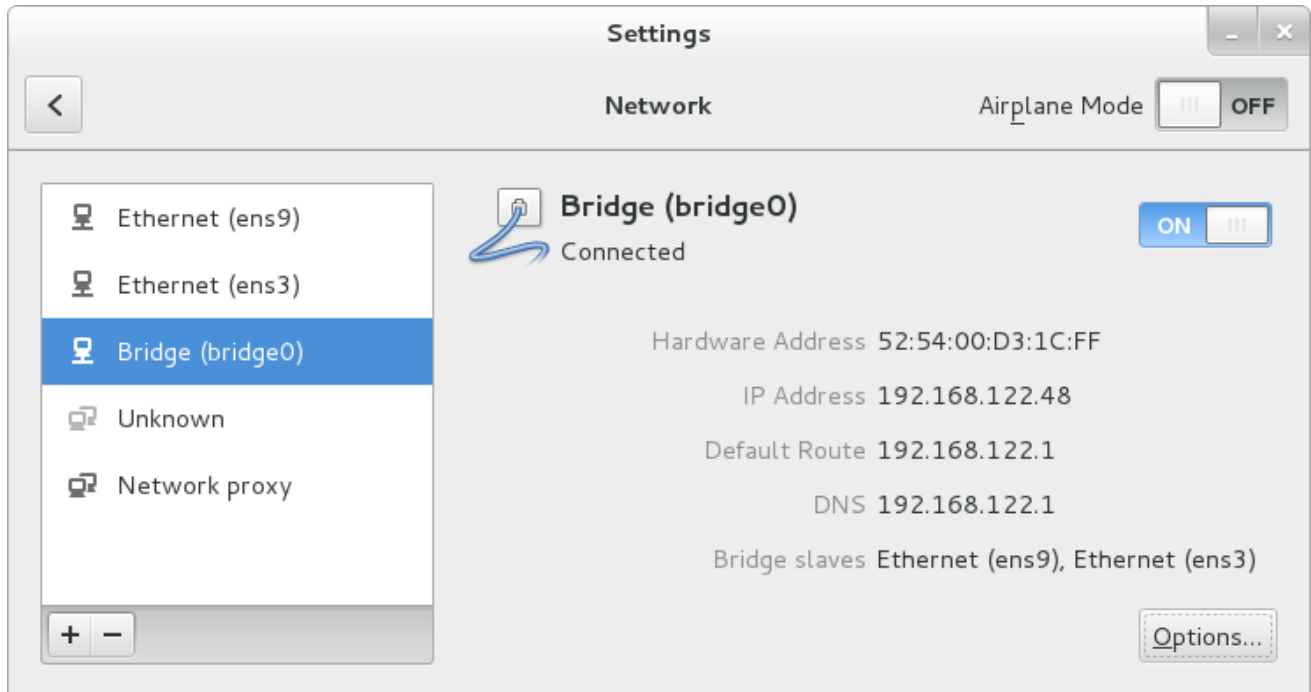


図6.8 NetworkManager グラフィカルユーザーインターフェースでのブリッジ

6.5. その他のリソース

- **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** コマンドの例を提供しています。
- **nm-settings(5)** man ページ: **NetworkManager** 接続の設定およびパラメーターを説明しています。

第7章 802.1Q VLAN タグの設定

VLAN を作成するには、**親インターフェース**と呼ばれるインターフェースの上に別のインターフェースを作成します。VLAN インターフェースは、パケットがインターフェースを通過する際に 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 インターフェースは他のインターフェースと同様に設定することができます。親インターフェースはイーサネットインターフェースである必要はありません。802.1Q VLAN タグインターフェースは、ブリッジ、ボンド、およびチームインターフェー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、ボンドにスレーブがあり、VLAN インターフェースをアクティブにする前にそれらが「アクティブ」になっていることが重要です。スレーブのないボンドに VLAN インターフェースを追加しても機能しません。
- VLAN スレーブはボンド上で **fail_over_mac=follow** オプションを使って設定することができません。これは、VLAN の仮想デバイスが MAC アドレスを変更して親の新 MAC アドレスに合致させることができないためです。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、セキュリティリスクが発生する可能性があります。以下にその例を示します。

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。

セキュリティリスクを最小限に抑えるためには、インターフェー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** または **ebtables** ユーティリティーを使用して、着信フレームのフィルタリングでタグ付けされていないフレームをドロップする。
- 古いネットワークインターフェースカードやループバックインターフェース、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『[Whether configuring bond on top of VLAN as slave interfaces is a valid configuration?](#)』を参照してください。

7.1. VLAN インターフェース設定方式の選択

- **NetworkManager** のテキストユーザーインターフェースである **nmtui** を使用して **VLAN インターフェースを設定するには**、[「テキスト形式のユーザーインターフェース nmtui を使った 802.1Q VLAN タグの設定」](#)に進みます。
- **NetworkManager** のコマンドラインツールである **nmcli** を使用して **VLAN インターフェースを設定するには**、[「コマンドラインツール nmcli を使った 802.1Q VLAN タグの設定」](#)に進みます。
- ネットワークインターフェースを手動で設定するには、[「コマンドラインを使った 802.1Q VLAN タグの設定」](#)に進みます。
- グラフィカルユーザーインターフェースツールを使ってネットワークを設定するには、[「GUI を使った 802.1Q VLAN タグの設定」](#)に進みます。

7.2. テキスト形式のユーザーインターフェース 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ースツール **nmtui** を使うと、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
```

テキスト形式のインターフェースが表示されます。無効なコマンドの場合は、使用法に関するメッセージがプリントされます。

移動するには矢印キーを使用するか、**Tab** を押して次に進むか **Shift+Tab** を押して前に戻ります。**Enter** を押してオプションを選びます。**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。**追加** を選択すると **新規の接続** 画面が開きます。

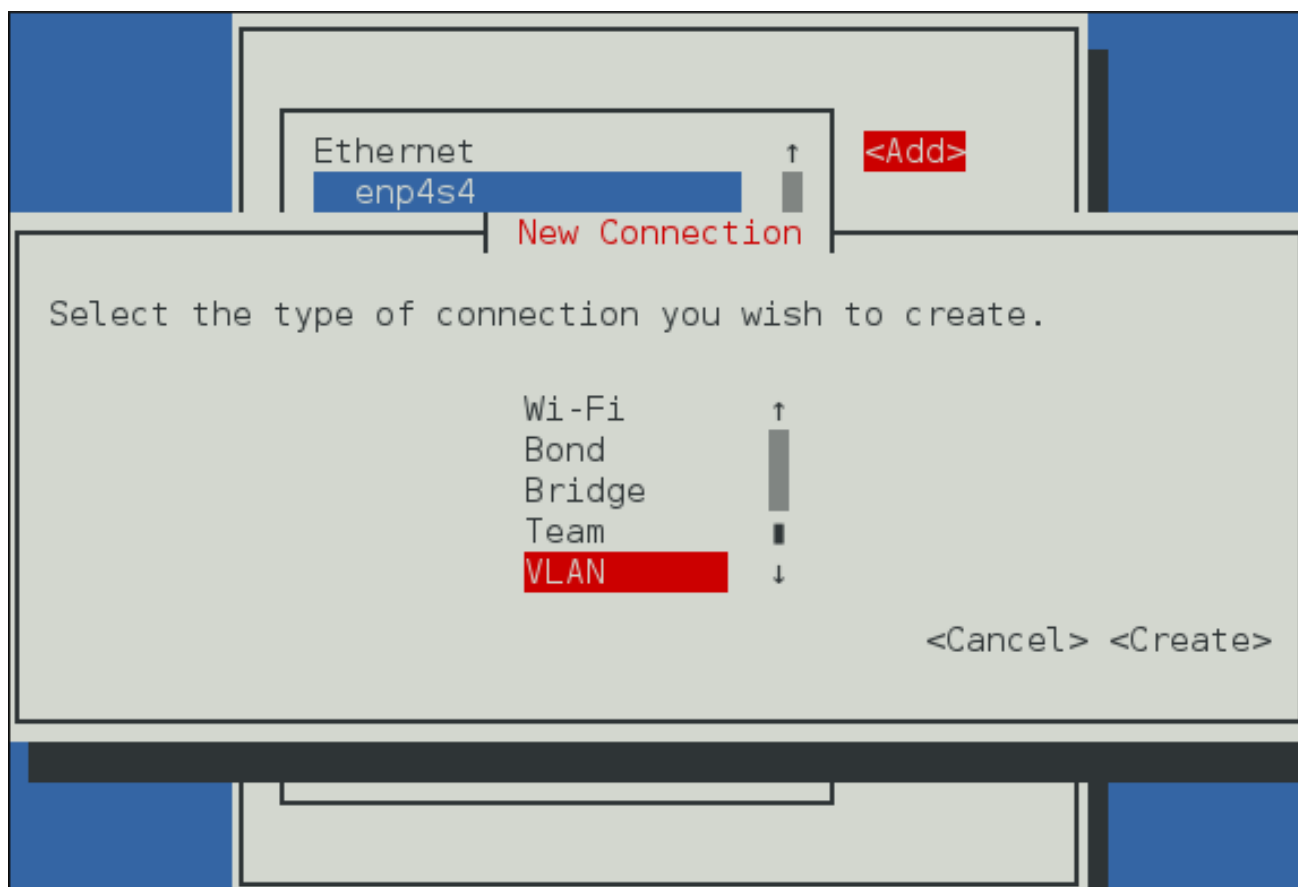


図7.1 NetworkManager テキスト形式のユーザーインターフェースの VLAN 接続追加メニュー

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。

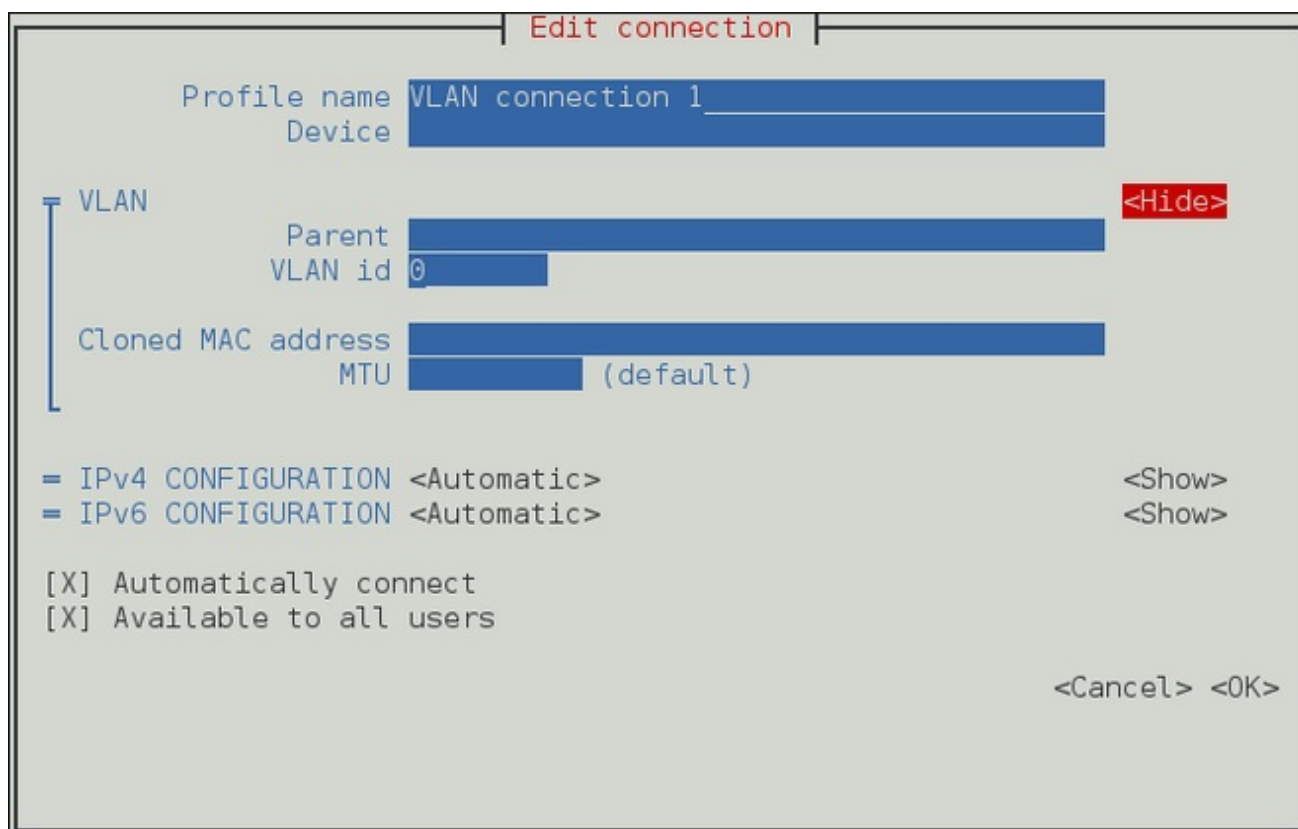


図7.2 NetworkManager テキスト形式ユーザーインターフェースで VLAN 接続を設定するメニュー

VLAN 用語の定義については、「[VLAN タグの設定](#)」を参照してください。

`nmcli` のインストール方法については、「[テキスト形式のユーザーインターフェース \(nmcli\) を使ったネットワーク設定](#)」を参照してください。

7.3. 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ースを表示するには、以下のコマンドを実行します。

```
~]$ nmcli con show
NAME                UUID                                TYPE
DEVICE
System eth1        9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  802-3-ethernet  eth1
System eth0        5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  802-3-ethernet  eth0
```

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ース名と同じように見えますが、異なるものです。`nmcli connection` コマンドでは ID を使用して接続を特定することができます。`firewalld` のような他のアプリケーションでは、デバイス名を使用します。

イーサネットインターフェース `eth0` 上で VLAN インターフェース `VLAN10` および ID **10** の 802.1Q VLAN インターフェースを作成するには、以下のコマンドを実行します。

```
~]$ nmcli con add type vlan ifname VLAN10 dev eth0 id 10
Connection 'vlan-VLAN10' (37750b4a-8ef5-40e6-be9b-4fb21a4b6d17)
successfully added.
```

VLAN インターフェースに **con-name** を指定しなかったため、名前がインターフェー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。あるいは、以下のように **con-name** オプションを使って名前を指定することもできます。

```
~]$ nmcli con add type vlan con-name VLAN12 dev eth0 id 12
Connection 'VLAN12' (b796c16a-9f5f-441c-835c-f594d40e6533) successfully
added.
```

VLAN インターフェースにアドレスを割り当てる

この `nmcli` コマンドを使って、他のインターフェースと同様に静的および動的インターフェー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** アドレスおよびゲートウェイの VLAN インターフェー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 eth0 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 インターフェースを作成するには、以下のコマンドを実行します。

```
~]$ nmcli con add type vlan con-name VLAN30 dev eth0 id 30
```

`nmcli` コマンドを使ったインスタンス設定例については、「[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 インターフェースを表示するには、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
eth0.12
System eth1         9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  802-3-ethernet  eth1
System eth0         5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  802-3-ethernet  eth0
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:                     eth0
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:                 eth0.12
GENERAL.STATE:                   activating[出力は省略されています]

```

VLAN コマンド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ース名である **eth0** で指定されており、これは接続の 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ース *eth1* 上で、イングレス優先度マッピングのインターフェース名 VLAN1 および ID **13** の 802.1Q VLAN インターフェース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 con add type vlan con-name VLAN1 dev eth2 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。

本ガイド執筆時点では、**connection.interface-name** と **vlan.interface-name** は、同一である必要があります (設定されている場合)。このため、**nmcli** のインタラクティブモードを使っ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 ユーティリティを使うと、802.1Q コードの作動方法を変更する **ioct1** フラグの設定および削除ができます。以下の 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 - インターフェー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ース (VLAN が作成されているインターフェースもしくはデバイス) に同期されます。親インターフェースが「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、作動状態のみが親から関連 VLAN に受け継がれ、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 の導入については、「[NetworkManager コマンドラインツール \(nmcli\) の使用](#)」を参照してください。

7.4. コマンドラインを使った 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読み込まれています。必要な場合は、**root** で以下のコマンドを実行して、モジュールを読み込みます。

```
~]# 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)** man ページを参照してください。

7.4.1. ifcfg ファイルを使用した 802.1q VLAN タグの設定

1. **/etc/sysconfig/network-scripts/ifcfg-ethX** の親インターフェースを以下のように設定します。X は特定のインターフェースに対応する一意の番号です。

```
DEVICE=ethX
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
```

2. **/etc/sysconfig/network-scripts/** ディレクトリーで VLAN インターフェースを設定します。設定ファイル名は、親インターフェースにピリオド . と VLAN ID 番号を加えたものにします。たとえば、VLAN ID が 192 で親インターフェースが *eth0* なら、設定ファイル名は **ifcfg-eth0.192** となります。

```
DEVICE=ethX.192
BOOTPROTO=none
ONBOOT=yes
IPADDR=192.168.1.1
PREFIX=24
NETWORK=192.168.1.0
VLAN=yes
```

たとえば、VLAN ID 193 で同一インターフェースの *eth0* 上に 2 つ目の VLAN を設定する必要がある場合は、VLAN 設定詳細で **eth0.193** という名前とした新規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
```

7.4.2. 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ース *eth0* 上で 名前が *VLAN8* および ID **8** の 802.1Q VLAN インターフェースを作成するには、**root** で以下のコマンドを実行します。

```
~]# ip link add link eth0 name eth0.8 type vlan id 8
```

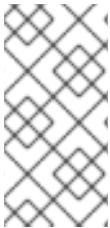

VLAN を表示するには、以下のコマンドを実行します。

```
~]$ ip -d link show eth0.8
4: eth0.8@eth0: <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 ユーティリティは VLAN ID が **0x** で始まっている場合は 16 進数として、**0** で始まっている場合は 8 進数として読み取ることに注意してください。つまり、10 進数の値 **22** の VLAN ID を割り当てるには、ゼロをつけてはいけないということになります。

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 eth0.8
```



注記

ip コマンドを使って作成された VLAN インターフェースは、システムが終了したり再起動すると失われます。システム再起動後も維持できるように VLAN インターフェースを設定するには、**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」](#)を参照してください。

7.5. GUI を使った 802.1Q VLAN タグの設定

7.5.1. VLAN 接続の確立

nm-connection-editor を使用して、既存のインターフェースを親インターフェースとして使い VLAN を作成することができます。VLAN デバイスが自動的に作成されるのは、親インターフェー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7.1 nm-connection-editor を使用して新規 VLAN 接続を追加する

1. ターミナルで **nm-connection-editor** と入力します。

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

2. **追加** ボタンをクリックします。**接続の種類を選択** ウィンドウが表示されます。**VLAN** を選択して **作成** 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ースをドロップダウンリストから選びます。
4. VLAN ID を入力します。
5. VLAN インターフェース名を入力します。これは、作成される VLAN インターフェースの名前です。たとえば、**eth0.1** や **vlan2** などです。(通常、この名前は親インターフェース名に「.」と VLAN ID を加えたものか、「**vlan**」と VLAN ID を加えたものになります。)
6. 設定を確認してから **保存** 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。

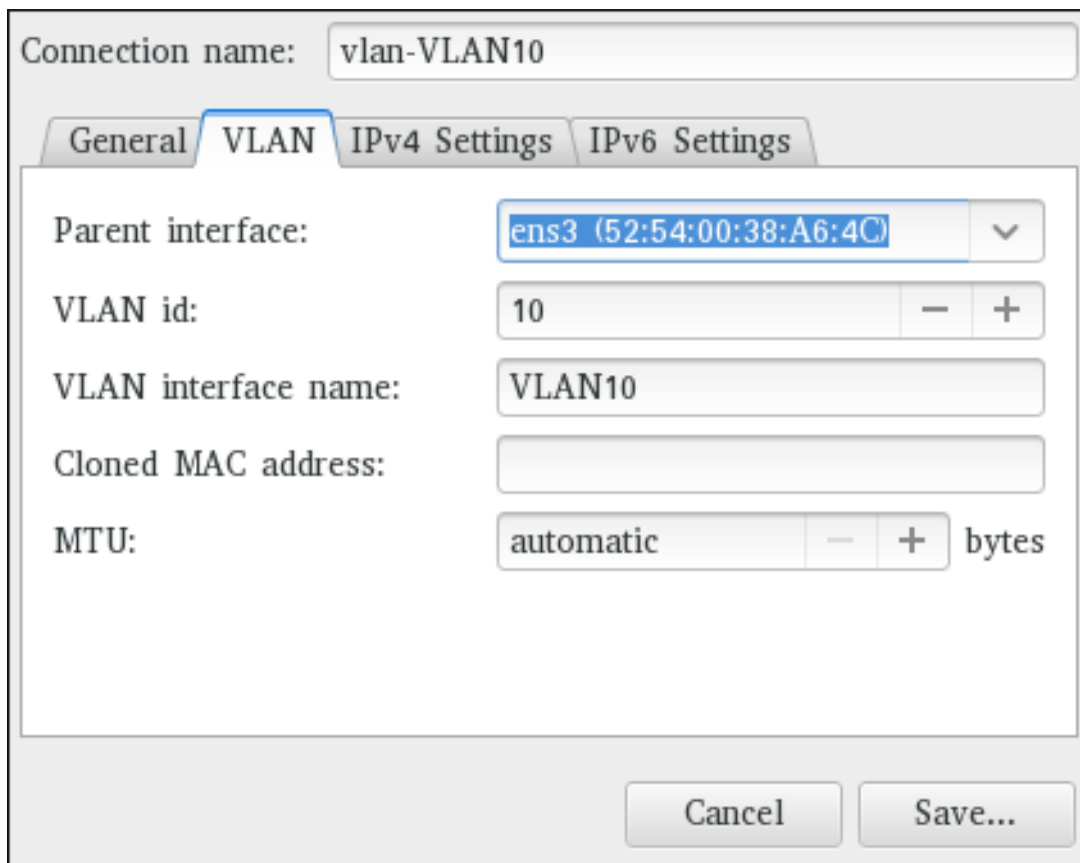


図7.3 nm-connection-editor を使用して新規 VLAN 接続を追加する

手順7.2 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. ターミナルで **nm-connection-editor** と入力します。

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

2. 編集する接続を選択して、**編集** ボタンをクリックします。
3. **全般** 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 ダイアログ内の以下の 3 つ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名前を入力します。この名前は **Network** ウィンドウメニューの **VLAN** セクションにある接続名一覧に表示されます。
- **この接続が利用可能になったときは自動的に接続する**: このボックスにチェックを入れると、この接続が利用可能な時に **NetworkManager** が自動接続します。詳細については「[ネットワークへの自動接続](#)」を参照してください。
- **全ユーザーがこのネットワークに接続可能とする**: 接続をこのシステム上のすべてのユーザーに利用可能とするには、このボックスにチェックを入れます。このセッティングを変更するには root 権限が必要になることがあります。詳細については「[システムワイドおよびプライベート接続プロファイル](#)」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、以下を設定します。

- **IPv4** の設定は、**IPv4 のセッティング** タブをクリックして「[IPv4 セッティングの設定](#)」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。

7.5.1.1. VLAN タブの設定

新規の VLAN 接続をすでに追加している場合 (手順に関しては、[手順7.1「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、親インターフェースと VLAN ID を設定できます。

親インターフェース

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ースを選択できます。

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 インターフェース名

作成される VLAN インターフェースの名前。たとえば、**eth0.1** や **vlan2** など。

クローンした MAC アドレス

VLAN インターフェー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。

7.6. 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** で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
```

7.7. その他のリソース

- **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について説明しています。
- **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** コマンドの例を提供しています。
- **nm-settings(5)** man ページ: **NetworkManager** 接続の設定およびパラメーターを説明しています。
- **nm-settings-ifcfg-rh(5)** man ページ: **/etc/sysconfig/network-scripts/ifcfg-*** ファイルの ifcfg-rh 設定を説明しています。

第8章 ネットワークデバイス命名における一貫性

Red Hat Enterprise Linux 7 では、ネットワークインターフェース用に一貫した予想可能なネットワークデバイスの命名方法を提供しています。この機能では、インターフェー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ース名を変更します。

従来の Linux のネットワークインターフェースには **eth[0123...]** などの数値が付けられました。しかし、これらの名前は必ずしもシャーシ上の実際のラベルと一致していません。複数のネットワークアダプターを持つ最近のサーバープラットフォームは、これらのインターフェースについて非断定的かつ非直感的な命名に遭遇する可能性があります。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 7 では、**udev** が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 **eth0** や **wlan0** 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。たとえば、**enp5s0** などの場合です。



警告

システムが **ethX** のような名前を使用することを許可してしまうので、一貫性のあるネットワークデバイス命名方法を無効にしないでください。ここで X は特定のインターフェースに対応する一意の数字ですが、起動プロセス中に別のネットワークインターフェース名になる場合があります。詳細は、「[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](#)」を参照してください。

8.1. 命名スキームの序列

デフォルトでは、**systemd** が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し、インターフェースに命名します。

- **スキーム 1:** ファームウェアや BIOS がオンボードデバイスについて提供する索引番号を組み入れた名前 (例: **eno1**) は、ファームウェアや BIOS からの情報が適用可能で利用可能な場合は、これが適用されます。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:** ファームウェアや BIOS が提供している PCI Express ホットプラグスロットの索引番号を組み込んだ名前 (例: **ens1**) は、ファームウェアや BIOS からの情報が適用可能で利用可能な場合は、これが適用されます。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場所を組み込んだ名前 (例: **enp2s0**) は、これが適用可能な場合は、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:** インターフェースの MAC アドレスを組み込んだ名前 (例: **enx78e7d1ea46da**) はデフォルトでは使用されませんが、ユーザーが選択すれば適用可能です。
- **スキーム 5:** 従来型の予測不可能なカーネル命名スキーム (例: **eth0**) は、他のすべての方法が失敗した場合に使用されます。

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、これが使用されます。Dell システム以外では、**biosdevname** を有効にするために **biosdevname=1** をカーネルコマンドラインのパラメーターとして渡す必要があります。Dell システムでは、**biosdevname** がインストールされていれば、これがデフォルトで使用されます。カーネルデバイス名を変更する **udev** ルールをユーザーが追加している場合、このルールが優先されます。

8.2. 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** 内のルールが、**udev** ヘルパーユーティリティーである **/lib/udev/rename_device** にすべての **/etc/sysconfig/network-scripts/ifcfg-suffix** ファイルを検索するよう指示します。インスタンスの MAC アドレスに合致する **HWADDR** のある **ifcfg** ファイルが見つかったら、**ifcfg** ファイルの **DEVICE** ディレクティブで与えられている名前にインターフェースの名前を変更します。
2. インターフェースの名前が上記のステップで変更されておらず、**biosdevname** がインストールされていて **biosdevname=0** がブートコマンドライン上でカーネルコマンドとして与えられていない場合、**/usr/lib/udev/rules.d/71-biosdevname.rules** 内のルールが **biosdevname** にインターフェースの名前を命名ポリシーに従って変更するよう指示します。
3. **/lib/udev/rules.d/75-net-description.rules** 内のルールが **udev** に、ネットワークインターフェースデバイスを調べて内部 **udev** デバイスのプロパティ値である **ID_NET_NAME_ONBOARD**、**ID_NET_NAME_SLOT**、**ID_NET_NAME_PATH**、**ID_NET_NAME_MAC** を満たすよう指示します。
4. 上記のステップ 1 および 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** が与えられていない場合、**/usr/lib/udev/rules.d/80-net-name-slot.rules** 内のルールが **udev** に、**ID_NET_NAME_ONBOARD**、**ID_NET_NAME_SLOT**、**ID_NET_NAME_PATH** の各優先度に従ってインターフェースの名前を変更するよう指示します。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、インターフェー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 を使っ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。

8.3. 予想可能なネットワークインターフェースデバイスの命名について

名前には、インターフェー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** はワイヤレス広域ネットワーク (WWAN)

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: 複数機能の PCI デバイスの名前にはすべて、[f<function>] 番号が含まれます。これには、機能 0 デバイスが含まれます。

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、値が 0 でない場合に限り、[P<domain>] が参照されます。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、ハブのポート番号の完全なチェーンで構成されます。名前が最大文字数の 15 を超えた場合は、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 設定記述子のデフォルト値 (c1) および USB インターフェース記述子のデフォルト値 (i0) は無視されます。

8.4. 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ー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、bus ID はデバイスのデバイス番号の始めに 0.n を付けたものになります。ここでの n は、サブチャンネル設定 IDになります。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ースは、以下のように命名されます。

enccw0.0.1234

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。

slccw0.0.1234

利用可能なネットワークデバイスとそれらの bus-ID を表示するには、znetconf -c または lscss -a のコマンドを使用します。

表8.1 System z 上の Linux のデバイス名タイプ

形式	詳細
enccwbus-ID	デバイスタイプがイーサネット
slccwbus-ID	デバイスタイプが SLIP の CTC ネットワークデバイス

8.5. VLAN インターフェースの命名スキーム

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** まですべて最大 4 文字となり、インターフェイス全体の名前は最大 15 文字までです。インターフェイス名の最大の長さはカーネルヘッダーで定義され、グローバルにすべてのアプリケーションに適用されます。

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 に VLAN ID を追加

vlan という語に VLAN ID を追加します。例: `vlan0005`

VLAN にパディングなしの VLAN ID を追加

vlan という語に先頭にゼロを付けない VLAN ID を追加します。例: `vlan5`

デバイス名に VLAN ID を追加

親インターフェイス名に VLAN ID を追加します。例: `eth0.0005`

デバイス名パディングなしの VLAN ID を追加

親インターフェイス名に先頭にゼロを付けない VLAN ID を追加します。例: `eth0.5`

8.6. BIOSDEVNAME を使っ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で実装されるこの機能は、すべての組み込み型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、および仮想機能ネットワークインターフェイスを、既存の **eth[0123...]** から [表8.2「biosdevname の命名規則」](#) に記載の新たな命名規則に変更します。システムが Dell システムか、「[機能の有効化および無効化](#)」にあるように **biosdevname** が明示的に有効となっている場合を除き、優先されるのは **systemd** 命名スキームであることに注意してください。

表8.2 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 in terface>^[c]
<div><div>[a] 新しい数値は 1 から始まります。</div><div>[b] 例: p3p4</div><div>[c] 例: p3p4_1</div></div>		

8.6.1. 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 内に収納されている **type 9** (システムスロット) フィールドと **type 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** パッケージがインストール済みである必要があります。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
```

8.6.2. 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
```

システムが最低要件を満たさない場合は、このオプションは無視され、システムは本章の始めに記載されている **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。

8.7. 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ー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、スクリプト変更 **irqbalance**、および他の同様の設定ファイルのエリアなどでもネットワークインターフェース名を更新する必要があります。また、インストール用にこの変更を有効にすると、**ksdevice** パラメータ経由でデバイス名を使用する既存の **kickstart** ファイルの変更も必要になります。これらの **kickstart** 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ために更新が必要になります。



注記

インターフェー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。

8.8. 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ースデバイスの特定

HWADDR ディレクティブを使って **ifcfg** ファイル内で MAC アドレスを設定すると、**udev** がこれを特定できるようになります。名前は **DEVICE** ディレクティブが提供する文字列から取られ、これは規則で **ifcfg** 接尾辞と同じになります。たとえば、**ifcfg-eth0** となります。

biosdevname の有効化または無効化

biosdevname が提供する名前を使用します (**biosdevname** が名前を決定できる場合)。

systemd-udev 命名スキームの有効化または無効化

systemd-udev が提供する名前を使用します (**systemd-udev** が名前を決定できる場合)。

8.9. ネットワークデバイス命名における一貫性の無効化

一貫性のあるネットワークデバイス命名方法を無効にすることは、特定の状況以外では推奨されません。詳細は、「[8章 ネットワークデバイス命名における一貫性](#)」および「[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](#)」を参照してください。

一貫性のあるネットワークデバイスの命名を無効にするには、以下のいずれかを選択します。

- デフォルトポリシー用の **udev** のルールファイルを「マスク化する」ことで、固定名の割り当てを無効にします。そのためには、**/dev/null** へのシンボリックリンクを作成します。これにより、予測不可能なカーネル名が使用されます。**root** で以下のコマンドを入力します。

```
~]# ln -s /dev/null /etc/udev/rules.d/80-net-name-slot.rules
```

- 独自の手動の命名スキームを作成します。たとえば、インターフェースを **internet0**、**dmz0**、または **lan0** と命名します。そのためには、独自の **udev** ルールファイルを作成し、デバイス用に **NAME** プロパティを設定します。新しいファイルがデフォルトのポリシーファイルの上に来るようにします。たとえば、これを **/etc/udev/rules.d/70-my-net-names.rules** という名前にします。
- 別の命名スキームを使用するようにデフォルトのポリシーファイルを変更します。たとえば、デフォルトで MAC アドレスをすべてのインターフェース名に使用するなどします。**root** で以下のようにデフォルトのポリシーファイルをコピーします。

```
~]# cp /usr/lib/udev/rules.d/80-net-name-slot.rules
/etc/udev/rules.d/80-net-name-slot.rules
```

/etc/udev/rules.d/ ディレクトリーにあるこのファイルを必要に応じて変更します。

- **/etc/default/grub/** ファイルを開き、**GRUB_CMDLINE_LINUX** 変数を探します。



注記

GRUB_CMDLINE_LINUX は、カーネルコマンドラインに追加されるエントリーを含む変数です。ご自分のシステム設定によっては、すでに追加の設定が含まれている場合があります。

net.ifnames=0 をカーネルパラメーターとして **GRUB_CMDLINE_LINUX** 変数に追加します。

```
GRUB_CMDLINE_LINUX="net.ifnames=0"
```

`/boot/grub2/grub.cfg` ファイルの GRUB 2 カーネルメニューエントリーをすべて更新するには、**root** で以下のコマンドを入力します。

```
~]# grubby --update-kernel=ALL --args=net.ifnames=0
```

grub ブートローダーの設定ファイルに関する情報を更新および表示するには、**grubby** ユーティリティーが使用されます。詳細は、**grubby(8)** man ページを参照してください。GRUB 2 での作業に関する詳細情報は、『[Red Hat Enterprise Linux 7 システム管理者のガイド](#)』を参照してください。

8.10. 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ース名が各インターフェースに割り当てられます。**udev** が使用する名前を一覧表示するには、**root** で以下の形式のコマンドを実行します。

```
~]# udevadm info /sys/class/net/iface | grep ID_NET_NAME
```

ここでの *iface* は、以下のコマンドで一覧を表示されるインターフェースのいずれかになります。

```
~]$ ls /sys/class/net/
```

「[デバイスの名前変更ステップについて](#)」にあるルールに従って、**udev** が使用可能な名前のいずれかを適用します。ルールを以下に要約します。

- `/usr/lib/udev/rules.d/60-net.rules` - `initscript` からのもの。
- `/usr/lib/udev/rules.d/71-biosdevname.rules` - `biosdevname` からのもの。
- `/usr/lib/udev/rules.d/80-net-name-slot.rules` - `systemd` からのもの。

上記のルールファイルを見ると、インターフェース命名が `initscript` もしくは `biosdevname` 経由で実行されると、これらは常に **udev** のネイティブ命名よりも優先されることが分かります。ただし、`initscript` による名前変更が行われない場合、または `biosdevname` が無効となっている場合にインターフェース名を変更するには、`80-net-name-slot.rules` を `/usr` から `/etc` にコピーして、このファイルを適切に編集することになります。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8.1 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ース (`eth[0,1,2...]`) と **udev** が正常に名前を変更したインターフェースが混在している場合

スキームが混在しているのは、ハードウェアのなかにはカーネルが **udev** に提供する情報で使用可能なものがないため命名できないか、**udev** に提供された情報が非一意のデバイス ID などのように適切でない場合のことが考えられます。後者の方が可能性が高く、この場合の解決法は、`ifcfg` ファイルでカスタマイズ命名スキームを使用するか、`80-net-name-slot.rules` を編集して使用される **udev** スキームを変更します。

例8.2 `/var/log/messages` または `systemd` ジャーナルで、各インターフェースの名前変更が 2 回実行されている場合

命名スキームがあるシステムで `ifcfg` ファイルでエンコードされているものの `initrd` イメージを再生成していないものは、この問題に直面する可能性があります。インターフェース名は当初 `initrd` にある間、初期ブート時に (`biosdevname`、**udev**、またはカーネルコマンドライン上の

dracut パラメータ経由で) 割り当てられます。そして実際の **rootfs** に切り替わった後、2 回目の名前変更が行われ、60-net.rules を処理するため、**udev** が生成する **/usr/lib/udev/rename_device** バイナリーによって新規インターフェー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8.3 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。

udev ルールはインターフェースに特定の名前を設定するので、指定した名前がすでに他のインターフェースで使用されている場合、設定に失敗します。これには、**/usr/lib/udev/rules.d/60-net.rules** ファイルで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、ethX 命名規則が使われます。ethx 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ース名を ethX 名に変更しようとしたり、カーネルにより与えられる予測が不可能な ethx 名の順序を変えようとした場合には、処理に失敗します。

以下の状況では、ethX 名の使用が正常に機能します。

- システムにネットワークインターフェースが 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8.4 net.ifnames=0 を設定した結果 ethX 名に一貫性がない場合

systemd による予測可能インターフェース命名 (**net.ifnames**) および **biosdevname** による命名スキームの両方が無効な場合、ネットワークインターフェースは初めにカーネルにより与えられた予測が不可能で一貫性が維持できない可能性のある ethX 名を使い続けます。

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 ethX 命名規則が使われます。並列処理が行われるため、カーネルがすべてのインターフェースを把握する順番は、再起動ごとに変わる可能性があります。Red Hat Enterprise Linux 7 では、**systemd** による予測可能なインターフェース命名スキームまたは **biosdevname** による命名スキームのどちらかを使って、カーネルにより与えられた予測が不可能な ethX インターフェースを、再起動後にも一貫性が維持できる予測可能な名前に変更します。

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「[RHEL7 で net.ifnames=0 を設定しても安全ですか?](#)」を参照してください。

8.11. その他のリソース

インストールされているドキュメント

- **udev(7)** man ページ: Linux の動的デバイス管理デーモン **udev** 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ジャーを説明しています。

- **biosdevname(1)** man ページ: BIOS 提供のデバイス名を獲得するユーティリティーについて説明しています。

オンラインのドキュメント

- 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。

パート II. 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について説明します。

第9章 INFINIBAND および RDMA ネットワークの設定

9.1. INFINIBAND および RDMA のテクノロジーについて

InfiniBand とは、2 つの全く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 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干渉を迂回し、その過程で通常はネットワーク通信の処理に必要な CPU オーバーヘッドを大幅に削減するためです。通常の IP データ送信では、マシン A 上のアプリケーション X がマシン B 上のアプリケーション Y にデータを送信します。この送信の一部として、マシン B 上のカーネルが最初にデータを受信し、パケットヘッダーの解読、データがアプリケーション Y に属することを判定、アプリケーション Y のウエイクアップ、アプリケーション Y によるカーネルへのシステムコールの読み込み待機を行います。その上でカーネルは、カーネル自体の内部メモリースペースからデータをアプリケーション Y が提供するバッファに手動でコピーする必要があります。つまりこのプロセスでは、ほとんどのネットワークトラフィックはシステムのメインメモリーバスで少なくとも 2 回にわたってコピーされる必要があるということになります (1 回目はホストアダプターが DMA を使用してデータをカーネルが提供するメモリーバッファにコピーし、2 回目は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にコピー)。またこのプロセスでは、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 のコンテキストを切り替えるために、何度もコンテキストを切り替える必要があります。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際には、システムに非常に高い CPU 負荷がかかります。

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に配置してその他の干渉は不要になります。こうすることで、高速ネットワーク通信のオーバーヘッドが大幅に削減されます。ただしこれは、ほとんどの IP ネットワークアプリケーションの基礎となっている標準 Berkeley Sockets API を使うと実行できません。このため、独自の API である InfiniBand Verbs API を提供し、アプリケーションをこの API に移植した後に、RDMA テクノロジーを直接使用できるようになります。

Red Hat Enterprise Linux 7 では InfiniBand ハードウェアと InfiniBand Verbs API の両方をサポートしています。さらに、InfiniBand Verbs API を InfiniBand ハードウェア以外で使用可能にするサポート対象の 2 つのテクノロジーがあります。それらは iWARP (Internet Wide Area RDMA Protocol) と RoCE/IBoE (RDMA over Converged Ethernet、これは後に InfiniBand over Ethernet に名前変更) です。これらは両方とも通常の IP ネットワークリンク層を基礎テクノロジーとしており、その設定の大半は実際に本ガイドの [2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、対象ハードウェアの適切なドライバーがインストールされていれば、RDMA 機能はすべて自動で現れます。カーネルドライバーは常に Red Hat が提供する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、ユーザースペースドライバーを手動でインストールする必要があります。

必要となるユーザースペースパッケージは以下のとおりです。

iWARP

Chelsio hardware: `libcxgb3` または `libcxgb4`。ハードウェアのバージョンによります。

RoCE/IBoE

Mellanox hardware: `libmlx4` または `libmlx5`。ハードウェアのバージョンによります。

さらに、`/etc/rdma/mlx4.conf` を編集して、RoCE/IBoE の使用に合ったポートタイプを適切に設定します。また、`/etc/modprobe.d/mlx4.conf` を編集して、カードがプラグインされている

イーサネットスイッチ上の「no-drop」サービスにどのパケット優先度を設定するかをドライバーに指示します。

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。

(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している場合、iWARP と InfiniBand の違いは表示されずに処理され、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、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。

9.2. 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や通常の **IP** ネットワークとは非常に異なるため、**IP** ネットワークで使用されてい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することができません。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**: サブネットマネジャー (ファブリックでアクティブなサブネットマネジャーがない場合、1 台のマシンでのみ必要)。
- **user space driver for installed hardware**: 以下のいずれかもしくは複数: infinipath-psm、libcxb3、libcxb4、libehca、libipathverbs、libmthca、libmlx4、libmlx5、libnes、および libocrdma。libehca は IBM Power Systems サーバーでのみ利用可能であることに注意してください。

推奨パッケージ

- **librdmacm**、**librdmacm-utils**、および **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

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**: 様々なタイプの RDMA 通信用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リプションに関する情報は、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、[オプションチャンネル](#)、[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 とは別の API を RDMA に提供します。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。

9.3. BASE RDMA サブシステムの設定

rdma サービスは自動で開始されます。InfiniBand でも iWARP でも RoCE/IBoE でも、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
```

ユーザーが **rdma** サービスを有効にする必要はありませんが、常時有効にしたい場合は、そうすることができます。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
```

9.3.1. 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みたいカーネルレベルとユーザーレベルの RDMA プロトコルを判別します。このファイルを編集すると、様々なドライバを有効、無効にできます。

以下のドライバが有効、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。マシン上の **SRP** プロトコル経由でエクスポートされたりリモートドライブまたはドライブアレイをまるでローカルのハードディスク上にあるかのようにマシンにマウントできるようになります。

- **SRPT: SRP** プロトコルのターゲットモードもしく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。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。デバイスが実際にエクスポートされる前に、ターゲットモードのサポートの設定が必要になります。詳細は、`targetd` および `targetcli` パッケージの資料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。

9.3.2. 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` rules ファイルは、IPoIB デバイスを (`ib0` や `ib1` などの) デフォルト名からより記述的な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:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
```

`link/infiniband` の直後には、IPoIB インターフェースの 20 バイトのハードウェアアドレスがあります。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することができます。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、`mlx4_ib0` のような `device_fabric` 命名規則を使用します。唯一推奨されないのは、`ib0` や `ib1` といった標準名を使用することです。これらの名前は、カーネルが割り当てる自動的な名前と競合する場合があるからです。次に、rules ファイルにエントリーを追加します。既存の例を rules ファイルからコピーして、**ATTR{address}** エントリー内の 8 バイトを名前を変更するデバイスからの強調表示された 8 バイトで置き換えます。また、新規の名前を **NAME** フィールドに記入します。

9.3.3. 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** 以外のユーザーによる 大型 RDMA アプリケーションの実行を可能にするには、非 **root** ユーザーがシステムで固定を許されるメモリー量を増やす必要があります。これを行うには、`/etc/security/limits.d/` ディレクトリー内に以下のようなコンテンツの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
```

9.3.4. 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 とイーサネットモードのいずれでも実行可能です。これらのカードは通常、InfiniBand がデフォルトとなっています。ユーザーがこれをイーサネットモードにすることもできます。現在このモードがサポートされているのは、ConnectX ファミリーのハードウェア (これは **mlx4** ドライバーを使用) のみです。このモードに設定するには、**/etc/rdma/mlx4.conf** ファイルにある指示に従い、当該ハードウェアに適切な PCI デバイス ID を見つけます。そして、そのデバイス ID と要求されるポートタイプを使用してファイル内に 1 行作成します。その後は **initramfs** を再構築して、更新されたポート設定が **initramfs** にコピーされたことを確認します。

ポートタイプを設定し、1 つもしくは両方のポートがイーサネットに設定されたら、ログに以下のメッセージが表示される場合があります。

```
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ています。

9.4. サブネットマネジャーの設定

9.4.1. 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ジャーが必要な場合、もしくはスイッチマネ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構成し、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ジャーが必要になります。サブネットマネジャーが複数あることもあり、その場合は 1 つがマスターとして動作し、その他はこのマスターが失敗した際にスレーブとして引き継ぐこととなります。

9.4.2. opensm マスター設定ファイルの設定

opensm プログラムは、マスター設定ファイルを **/etc/rdma/opensm.conf** に保存します。ユーザーはこのファイルをいつでも編集することができ、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 のバインドとそれと実行する PRIORITY という 2 つの最も一般的な編集については **opensm.conf** ファイルではなく **/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** ファイルが編集されない場合は、**opensm** パッケージのアップグレードに合わせてこのファイルも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規オプションを編集済みのファイルにマージする必要性がでてきます。

9.4.3. opensm スタートアップオプションの設定

/etc/sysconfig/opensm ファイル内のオプションは、サブネットマネジャーが実際に開始される方法や開始されるサブネットマネ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ジャーのコピーを必要とします。**opensm** サブネットマネジャーが管理するのはアプリケーション

ンのインスタンスにつき 1 つのサブネットのみで、管理する必要のある各サブネットで一度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、どのサーバーをマスターやスレーブにするかを管理する優先度を各サーバーで設定します。

/etc/sysconfig/opensm ファイルは、サブネットマネジャーの優先度を設定し、サブネットマネジャーがバインドされる 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** ファイル自体にもオプションの詳細な説明があります。**opensm** のフェイルオーバーとマルチファブリック操作を有効にするには、このファイルの指示に従うだけで可能です。

9.4.4. P_Key 定義の作成

デフォルトでは、**opensm.conf** は **/etc/rdma/partitions.conf** ファイルを検索してファブリックを作成するためのパーティション一覧を取得します。すべてのファブリックには **0x7fff** サブネットを含める必要があります。すべてのスイッチとホストは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することができます。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の場合、デフォルトの **0x7fff** パーティションは 10 Gbps のスピードになり、**0x0002** 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;

```

9.4.5. opensm の有効化

opensm サービスはインストール時にはデフォルトで有効となっていないため、これを有効にする必要があります。**root** で以下のコマンドを実行します。

```
~]# systemctl enable opensm
```

9.5. 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** ベースのデバイスなので、IPoIB の設定が完了しデバイスに **IP** アドレスが割り当てられてから、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** サービスが有効にされ、適切なユーザースペースのライブラリーが特定ハードウェア用にインストールされると、ユーザースペースの **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** プログラムはどのデバイスがシステム内にあるかを表示し、**ibv_devinfo** コマンドで各デバイスの詳細情報が表示されます。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** コマンドの出力はやや違ったものになり (ポート MTU が **ibv_devinfo** にはあるが **ibstat** 出力にはなし、またポート GUID が **ibstat** 出力にはあるが **ibv_devinfo** 出力にはなし、など) 名前も異なるものがいくつかあります (たとえば、**ibstat** 出力の *Base local identifier* (LID) は、**ibv_devinfo** の **port_lid** と同じもの)。

infiniband-diags パッケージからの **ibping** など、簡単な ping プログラムを使うと、RDMA の接続性をテストすることができます。**ibping** プログラムでは、クライアント-サーバーモデルが用いられます。1 つのマシン上で **ibping** サーバーを起動してから別のマシン上で **ibping** をクライアントとして稼働し、このクライアントに対して **ibping** サーバーに接続するよう命令します。ベースの RDMA 機能をテストするので、サーバーの指定には、**IP** アドレスではなく RDMA 固有のアドレス解決方法を使用する必要があります。

サーバーマシン上では、ユーザーは **ibv_devinfo** および **ibstat** のコマンドを使うと、テスト対象となるポートの **port_lid** (または Base lid) および Port GUID をプリントアウトできます (上記インターフェースの port 1 で **port_lid/Base LID** が 2、Port GUID が **0xf4521403007bcba1** と仮定する)。次に必要なオプションをつけて **ibping** を起動し、テスト対象のカードとポートをバインドするとともに、サーバーモードで **ibping** の指定を実行します。-? または --help オプションを **ibping** に渡すと、利用可能なオプションが表示されますが、このインスタンスでは -S または --Server オプションが必要です。特定のカードとポートをバインドするには、-C か --Ca と -P か --Port 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 への接続性をテストするには、**ibping** にカード上の port 2 にバインドするよう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。例を示します。

```
~]$ ibping -S -C mlx4_1 -P 1
```

それからクライアントマシンに切り替え、**ibping** を実行します。サーバーの **ibping** プログラムがバインドされているポートの port 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。これらが明らかになったら、クライアントとして **ibping** プログラムを実行し、サーバー上で収集された port 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。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 0xf4521403007bcb1 \
--- 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。ユーザーが間違ったアーキテクチャタイプ用のドライバーや **libibverbs** をインストールすると、稀にこのようなことが起こることもあります。たとえば、**libibverbs** のアーキテクチャが **x86_64** でインストールされた **libmlx4** のタイプが **i686** であった場合、このエラーが発生します。



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 LID を受け付けるという点で、独特のものです。これにより、テストのシナリオから IPoIB アドレス指定の問題の可能性を排除して簡単なテストにすることができ、単純な RDMA 通信が機能しているかどうかという切り分けられた結果が得られます。

9.6. IPOIB の設定

9.6.1. IPoIB の役割について

「[IP ネットワークと非 IP ネットワーク](#)」で言及したように、ほとんどのネットワークは **IP** ネットワークですが、InfiniBand は違います。IPoIB の役割は、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 InfiniBand ネットワークで修正することなく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作成されたアプリケーションの場合より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 および RoCE/IBoE のネットワークは実際には **IP** リンク層の上に RDMA 層のある **IP** ネットワークなので、IPoIB の必要はありません。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。

9.6.2. 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 層が、送信中の **IP** パケットに 4 バイトの IPoIB ヘッダーを追加します。このため、IPoIB MTU は InfiniBand リンク層の MTU より 4 バイト小さいものである必要があります。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 ビットのサイズのフィールドしかないので、最大バイト数は **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ため、実際に許可される MTU はこれよりも小さくなります。このため、必要となる **TCP** ヘッダーすべてに十分なスペースを確保するために、connected モードでの IPoIB MTU は最大 **65520** となります。

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ース上で最大 MTU でマルチキャストデータの送信を試みるため、インターフェースを 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。

9.6.3. 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 **ifconfig** ユーティリティでは 20 バイトすべては読み取れないため、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、これを使用しないでください。iproute パッケージの **ip** ユーティリティは正常に機能します。

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 デバイスは最初に作成される際に、デフォルトの **0xfe:80:00:00:00:00:00:00** というサブネット接頭辞が付けられます。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 インターフェー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。「[70-persistent-ipoib.rules の使用](#)」のセクションでは、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生成する方法を説明しています。こうすることで、デバイスの適合を確実にすることができます。IPoIB インターフェースを設定する場合は、設定ファイルの **HWADDR** フィールドに 20 バイトすべてを含めることができますが、適合に使用されるのは最後の 8 バイトのみで、これを使って設定ファイルで指定されたハードウェアを見つけます。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記入されておらず、**ifup-ib** が IPoIB インターフェースを有効にす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システムが見つけれられないというエラーが発生します。IPoIB インターフェースでは、設定ファイルの **TYPE=** フィールドは、**InfiniBand** か **infiniband** のどちら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特定の表記を受け入れます)。

9.6.4. 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** サブネットを使う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ース上で 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、管理者は新たなサブネットを作成して、これら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ネットに限定することができます。**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ジャーで定義される必要があります。**opensm** サブネットマネジャーを使って **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](#)」を参照してください。IPoIB インターフェースの場合は、**P_Key** サブネットが作成されると、それらの **P_Key** サブネットのみを対象とした IPoIB 設定ファイルを追加で作成することができます。各 IPoIB インターフェースはイーサネットデバイス上の VLAN インターフェースのように、同一リンクを共有するものの異なる **P_Key** の値を持つ IPoIB インターフェースとは完全に異なるファブリックにいるかのように動作します。

IPoIB **P_Key** インターフェー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、**0x8000** という高いビットは、**P_Key** のメンバーシップが部分的なもの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 ドライバーがサポートするのは **P_Key** サブネットの完全なメンバーシップのみなので、Linux が接続可能なサブネットでは常に高いビットの **P_Key** が設定されています。つまり、Linux コンピューターが **P_Key 0x0002** に加わると、そのコンピューターの実際の **P_Key** 番号は **0x8002** となり、**P_Key 0x0002** の完全なメンバーになったことを示します。このため、「[P_Key 定義の作成](#)」セクションの説明にあるように **opensm partitions.conf** ファイルで **P_Key** 定義を作成する際は、**0x8000** 以外で **P_Key** の値を指定する必要があります。ただし、Linux クライアント上で **P_Key** IPoIB インターフェースを定義する際には、ベースの **P_Key** の値に **0x8000** の値を追加します。

9.7. テキスト形式のユーザーインターフェース 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ースツール **nmtui** を使うと、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
```

テキスト形式のインターフェースが表示されます。無効なコマンドの場合は、使用法に関するメッセージがプリントされます。

移動するには矢印キーを使用するか、**Tab** を押して次に進むか **Shift+Tab** を押して前に戻ります。**Enter** を押してオプションを選びます。**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。追加 を選択すると **新規の接続** 画面が開きます。

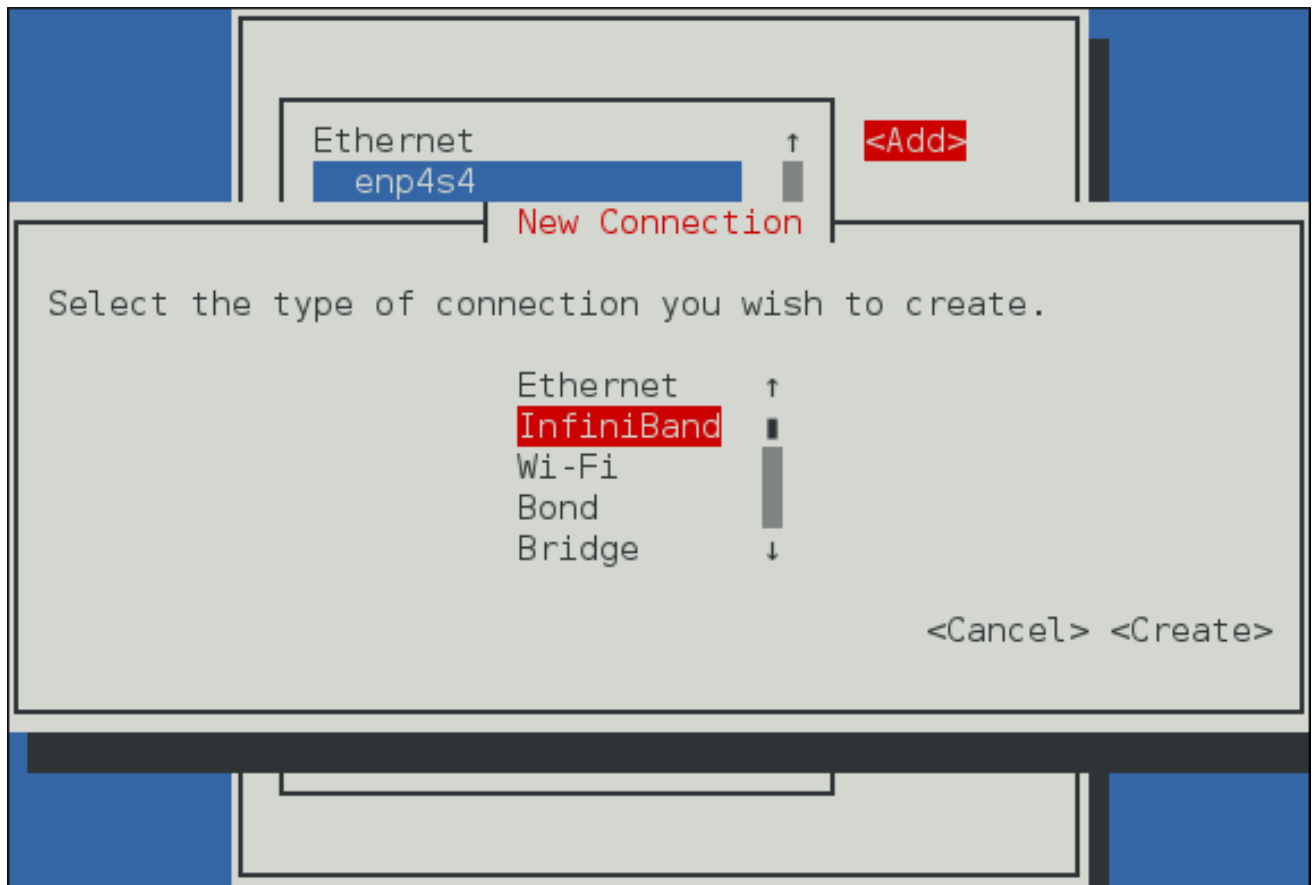


図9.1 NetworkManager テキスト形式のユーザーインターフェースの InfiniBand 接続追加メニュー

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。

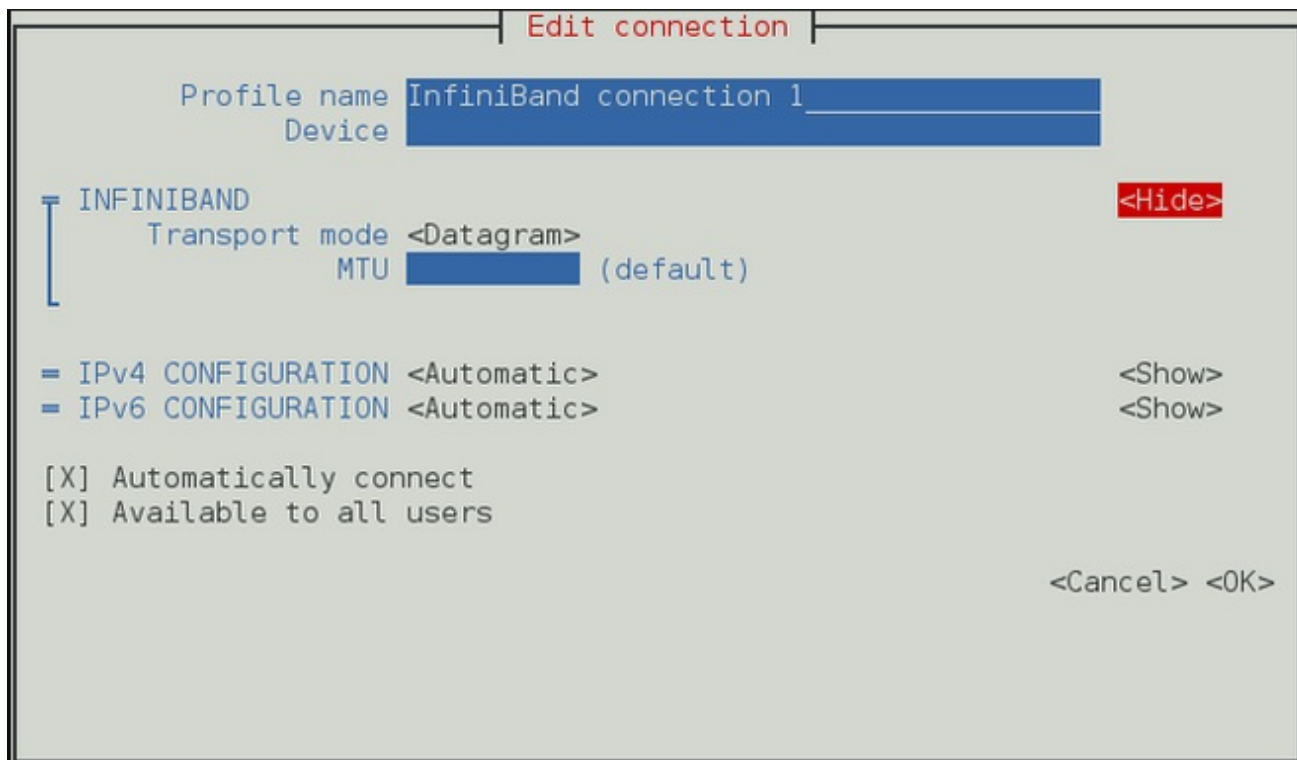


図9.2 NetworkManager テキスト形式ユーザーインターフェースで InfiniBand 接続を設定するメニュー

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については、[「テキスト形式のユーザーインターフェース \(nmtui\) を使ったネットワーク設定」](#)を参照してください。

9.8. 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判断します。必要な場合は、[「70-persistent-ipoib.rules の使用」](#)に記載の指示に従って、`udev` の名前変更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一旦削除してリロードすると、再起動せずに IPoIB インターフェースの名前変更ができます。

```
~]$ 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 インターフェースを作成します。以下に 2 とおりの方法を示します。

例9.1 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
```

あるいは、以下のように **nmcli c add** と **nmcli c modify** を1つのコマンドで実行することができます。

例9.2 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
```

これで **mlx4_ib0** という名前の IPoIB インターフェースが作成され、connected モードを使用するように設定されました。MTU は connected モードの最大値で、**IPv4** および **IPv6** に **DHCP** が使われます。IPoIB インターフェースをクラスタトラフィックに使用し、イーサネットインターフェー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 インターフェー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、以下のように実行することができます。

```
~]$ 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 インターフェースが必要な場合は、以下のように **nmcli** を使って作成します。

```
~]$ 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
```

9.9. 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判断します。必要な場合は、[「70-persistent-ipoib.rules の使用」](#)に記載の指示に従って、**udev** の名前変更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一旦削除してリロードすると、再起動せずに IPoIB インターフェースの名前変更ができます。

```
~]$ rmmod ib_ipoib
~]$ modprobe ib_ipoib
```

デバイスに必要な名前が付けられたら、管理者は好みのエディターで **ifcfg** ファイルを作成し、デバイスを制御することができます。静的 **IPv4** アドレス指定の通常の 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: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** の名前変更ルールで作成されたカスタム名に合致する必要があります。NAME エントリ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** か **no** で、**yes** の場合は connected モードが使用され、**no** の場合は datagram モードが使用されます ([「IPoIB 通信モードについて」](#) を参照)。

P_Key インターフェースの場合、以下のような設定ファイルになります。

```
~]$ 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: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** インターフェースファイルでは、**PHYSDEV** はディレクティブは必須となり、親デバイス名にする必要があります。**PKEY** ディレクティブは **yes** に、**PKEY_ID** はインターフェースの数 (**0x8000** メンバーシップビットを追加したものの有りまたはなし) に設定する必要があります。ただし、デバイス名は、**PKEY_ID** を 4 桁の 16 進数で表したものに論理和を使用して **0x8000** メンバーシップビットを合わせたものにします。以下のようにになります。

```
NAME=${PHYSDEV}.$((0x8000 | $PKEY_ID))
```

デフォルトでは、ファイル内の **PKEY_ID** は 10 進数として扱われ、16 進数に変換された後、論理和を用いて **0x8000** と合わされてデバイス用の適切な名前になります。ただしユーザーは、数字に標準 **0x** 接頭辞を追加して **PKEY_ID** を 16 進数で指定することもできます。

9.10. IPOIB 設定後の RDMA ネットワークテスト

IPoIB が設定されれば、**IP** アドレスを使用して RDMA デバイスの指定ができます。**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 アプリケーションでは、接続するリモートマシンまたはローカルデバイスの指定にこの方法が優先されます。場合によっては、この方法が唯一の指定方法になります。

IPoIB 層の機能のテストには、標準 **IP** ネットワークテストツールを使って IPoIB デバイスの **IP** アドレスを提供することができます。たとえば、IPoIB デバイスの **IP** アドレスの間で ping コマンドが機能させることができます。

Red Hat Enterprise Linux には **qperf** と **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することができます。

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 **IP** アドレスまたは IPoIB デバイスの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は異なる RDMA インターフェースを使って接続することが可能です。これは、ホスト名または **IP** アドレスを RDMA アドレスに変更するプロセスで、使用されるマシン 2 台の間で有効な RDMA アドレスのペアが許可されるためです。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ているマシンにそれぞれ 2 つのポートがあり、各マシンの 2 つ目のポートに **IP** アドレスが提供された場合、プログラムは各マシンの 1 つ目のポートを有効な接続方法と判断し、それらを使用することになります。この場合、「[初期の InfiniBand RDMA 操作のテスト](#)」の **ibping** で実行したように、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と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を実行することができます。**qperf** では、ポートへのバインディング方法は多少異なります。**qperf** プログラムは 1 台のマシン上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って **qperf** に接続できます。**qperf** はまず、データ接続の開封を試み、クライアントのコマンドラインで提供された **IP** アドレスもしくはホスト名で要求されたテストを実行しますが、このアドレスに問題がある場合は、**qperf** はクライアントとサーバー間にあるいずれかの有

効なパスでのテスト実行を試みます。このため、**qperf** が特定のリンク上でテストするには、**qperf** クライアントに **-loc_id** および **-rem_id** オプションを使用して特定リンク上でのテストを強制します。

9.11. GUI を使った IPOIB の設定

グラフィカルツールで InfiniBand 接続を設定するには、**nm-connection-editor** を使用します。

手順9.1 nm-connection-editor を使用して新規 InfiniBand 接続を追加する

1. ターミナルで **nm-connection-editor** と入力します。

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

2. **追加** ボタンをクリックします。**接続の種類を選択** ウィンドウが表示されます。**InfiniBand** を選択して **作成** 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びます。
4. InfiniBand MAC アドレスを入力します。
5. 設定を確認してから **保存** 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9.2 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. ターミナルで **nm-connection-editor** と入力します。

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

2. 編集する接続を選択して、**編集** ボタンをクリックします。
3. **全般** 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 **全般** タブにある 5 つの設定は、すべての接続の種類で共通のものです。

- **接続名**: ネットワーク接続の名前を入力します。この名前は **Network** ウィンドウメニューの接続名一覧に表示されます。
- **この接続が利用可能になったときは自動的に接続する**: このボックスにチェックを入れると、この接続が利用可能な時に **NetworkManager** が自動接続します。詳細については [「ネットワークへの自動接続」](#) を参照してください。
- **全ユーザーがこのネットワークに接続可能とする**: システム上のすべてのユーザーが接続可能とするには、このボックスにチェックを入れます。このセッティングを変更するには root 権限が必要になることがあります。詳細については [「システムワイドおよびプライベート接続プロファイル」](#) を参照してください。

- この接続を使用したときは自動的に **VPN に接続する**: このボックスにチェックを入れると、この接続が利用可能な時に **NetworkManager** が自動で選択された VPN に接続します。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。

以下のいずれかを選んで設定します。

- **IPv4** の設定は、**IPv4 のセッティング** タブをクリックして「[IPv4 セッティングの設定](#)」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。

9.11.1. InfiniBand タブの設定

新規の InfiniBand 接続をすでに追加している場合 (手順に関しては、[手順9.1 「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、親インターフェースと 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。

9.12. その他のリソース

インストールされているドキュメント

- `/usr/share/doc/initscripts-version/sysconfig.txt`: 設定ファイルおよびそれらのディレクティブについて説明しています。

オンラインのドキュメント

<https://www.kernel.org/doc/Documentation/infiniband/ipoib.txt>

IPoIB ドライバーの説明で、関連する RFC の参照先も含まれています。

パート III. 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について説明します。



注記

Web ブラウザーを使ってサーバーを監視および管理する方法については、『[Red Hat Enterprise Linux 7 Getting Started with Cockpit](#)』を参照してください。

第10章 DHCP サーバー

DHCP (Dynamic Host Configuration Protocol: 動的ホスト構成プロトコル) 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、中央に配置された **DHCP** サーバーに接続します。DHCP サーバーは、クライアントのネットワーク設定情報 (**IP** アドレス、ゲートウェイ、**DNS** サーバーなど) を返します。

10.1. 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ースの自動設定に便利なものです。クライアントシステムの設定時に、**IP** アドレスやネットマスク、ゲートウェイ、**DNS** サーバーを指定する代わりに **DHCP** を選択することができます。クライアントはこれらの情報を **DHCP** サーバーから取得します。多数のシステムの **IP** アドレスを変更する際にも **DHCP** は便利です。すべてのシステムを設定する代わりに、サーバーで新たな **IP** アドレスを 1 つの設定ファイルで編集することができます。組織の **DNS** サーバーが変更される際は、**DHCP** クライアントではなく、**DHCP** サーバー上で変更が実施されます。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正常に接続していれば、ノートパソコンや他のモバイルコンピュータのユーザーはそれらのデバイスのあるオフィスから別のオフィスに移動して使用できます。

DNS および **DHCP** サーバー、またプロビジョニングアプリケーションの管理者は、組織内で使用するホスト名の形式について合意する必要があることに留意してください。ホスト名の形式については、「[推奨される命名プラクティス](#)」を参照してください。

10.2. 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** でこの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、**/etc/dhcp/dhcpd.conf** ファイルが作成されます。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
#
```

サンプルの設定ファイルは、**/usr/share/doc/dhcp-version;/dhcpd.conf.example** にあります。**/etc/dhcp/dhcpd.conf** を設定する際に、このファイルを使用してください。詳細は以下で説明します。

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格納します。詳細は「[リースデータベース](#)」を参照してください。

10.2.1. 設定ファイル

DHCP サーバーを設定するには、最初にクライアントのネットワーク情報を保存している設定ファイルを作成します。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。

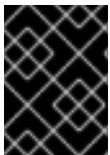
設定ファイルには追加のタブや空白行が含まれているため、簡単に書式を整えることができます。キーワードは大文字と小文字の区別がなく、行頭がハッシュ記号 (#) で始まる行はコメントとみなされます。

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。

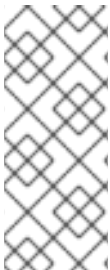
キーワードオプションから始まるパラメーターは、**オプション**と呼ばれます。オプションは、**DHCP** オプションを制御します。一方、パラメーターはオプションでない値を設定し、**DHCP** サーバーの動作を制御します。

中括弧 ({ }) で囲まれたセクションの前で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** コマンドを使って **DHCP** デーモンを再起動するまでは変更内容は反映されません。



注記

毎回 **DHCP** 設定ファイルを変更してサービスを再起動させる代わりに、**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、**DHCP** サーバーの実行中でも変更を行うことができます。**omshell** の詳細については、**omshell** の man ページを参照してください。

例10.1「サブネットの宣言」では、**routers**、**subnet-mask**、**domain-search**、**domain-name-servers**、および **time-offset** の各オプションは、それらの下で宣言されている **host** ステートメントに使用されます。

機能するサブネットと、**DHCP** サーバーが接続するサブネットではすべて、**subnet** 宣言が1つ必要になります。これは **DHCP** デーモンにアドレスがそのサブネット上にあることを認識する方法を指示します。**subnet** 宣言は、そのサブネットに動的に割り当てるアドレスがない場合でも、各サブネットに必要となります。

この例では、サブネット内のすべての **DHCP** クライアントに対するグローバルオプションと **range** が宣言されています。クライアントには **range** 内の **IP** アドレスが割り当てられています。

例10.1 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 10.2 「Range パラメーター」](#) の例の値を修正します。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。この例ではクライアントシステムに、**range 192.168.1.10 から 192.168.1.100** までの **IP** アドレスを割り当てます。

例10.2 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ースカードの 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10.3 「DHCP を使用した静的 IP アドレス」](#) の例では、**host apex** 宣言により、MAC アドレス **00:A0:78:8E:9E:AA** のネットワークインターフェースカードが常に **IP** アドレス **192.168.1.4** を受け取るように指定されます。

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のできる点にも留意してください。

例10.3 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 では、InfiniBand IPoIB インターフェースへの静的 **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ー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 インターフェースに一意の ID を指定する別の方法を使用する必要があります。標準では、オプションの **dhcp-client-identifier=** コンストラクトを使用して、IPoIB インターフェースの **dhcp-client-identifier** フィールドを指定します。**DHCP** サーバーのホストコンストラクトは、最大でホストの 1 節あたりハードウェアイーサネット 1 つと **dhcp-client-identifier** エントリー 1 つをサポートします。ただし、固定アドレスエントリーは複数ある場合があり、**DHCP** サーバーは、**DHCP** リクエストを受信したネットワークに適切なアドレスに自動的に応答します。

例10.4 複数インターフェースにおける DHCP を使用した静的 IP アドレス

たとえば 2 つの InfiniBand インターフェースと各物理的インターフェース上に **P_Key** インターフェースがあり、イーサネット接続もある場合など、マシンの設定が複雑な場合は、以下の静的 IP コンストラクトを使用することができます。

```
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;
}
```

デバイスに適切な **dhcp-client-identifier** をを見つけるには、通常、接頭辞 **ff:00:00:00:00:00:02:00:00:02:c9:00** を使用して、IPoIB インターフェースの最後の 8 バイトを追加します (これは、IPoIB インターフェースが接続されている InfiniBand ポートの 8 バイトの GUID と同じものです)。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、**DHCP** サーバーで **tcpdump** を使用し、着信 IPoIB **DHCP** リクエストを取得して、そこから適切な **dhcp-client-identifier** を収集することが推奨されます。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:** に対応しており、ID の残りの部分は **DHCP** 設定ファイルに表記するそのままの形で引用されています。

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10.5「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** 内でも囲まれた 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みなされます。**shared-network** に割り当てる名前は、そのネットワークの記述的なタイトルにします。たとえば、テストラボ環境のサブネットの場合は、「test-lab」というタイトルにします。

例10.5 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10.6「Group 宣言」](#)にあるように、**group** 宣言を使用すると、宣言のグループにグローバルパラメーターを適用できます。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10.6 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することができます。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
```

ここでの *version_number* は、**DHCP** バージョン番号になります。

オプションステートメントの全一覧とその機能については、**dhcp-options(5)** の man ページを参照してください。

10.2.2. 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格納しています。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、このリースデータベースに自動的に保存されます。この情報には、リース期間、**IP** アドレスの割り当て先、リースの開始日と終了日、リース取得に使用されるネットワークインターフェースカードの **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** ファイルの名前は **dhcpd.leases~** に変更され、一時的なリースデータベースが **dhcpd.leases** に書き込まれます。

リースデータベースがバックアップファイルに名前変更された後、新規ファイルが書き込まれる前に、**DHCPDHCP** デーモンが強制終了されるか、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。これを解決する方法は、**dhcpd.leases~** バックアップファイルの名前を **dhcpd.leases** に変更して、デーモンを起動することです。

10.2.3. サーバーの起動と停止

重要

DHCP サーバーの初回起動時には、**dhcpd.leases** ファイルが存在しないと起動に失敗します。ファイルが存在しない場合、**touch /var/lib/dhcpd/dhcpd.leases** コマンドを使用してファイルを作成してください。このサーバーが **DNS** サーバーとして **BIND** も実行している場合、このステップは不要です。**named** サービスを開始すると、自動的に **dhcpd.leases** ファイルを確認するためです。

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。これを解決する方法は、**dhcpd.leases~** バックアップファイルの名前を **dhcpd.leases** に変更して、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、以下のコマンドを実行します。

```
systemctl start dhcpd.service
```

DHCP サービスを停止するには、以下のコマンドを実行します。

```
systemctl stop dhcpd.service
```

デフォルトでは、**DHCP** サービスはブート時に起動しません。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 7 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ースが接続されており、**DHCP** サーバーがいずれか 1 つのインターフェースでのみ **DHCP** リクエスト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上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` ファイル内でサブネット宣言がなされているインターフェース上でのみ、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。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してインターネットの **IP** アドレスを取得します。もう 1 つ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** で **dhcpd.service** ファイルをコピーしてから編集します。例を示します。

```
~]# cp /usr/lib/systemd/system/dhcpd.service /etc/systemd/system/  
~]# vi /etc/systemd/system/dhcpd.service
```

[Service] セクションの下にある行を編集します。

```
ExecStart=/usr/sbin/dhcpd -f -cf /etc/dhcp/dhcpd.conf -user dhcpd -group  
dhcpd --no-pid your_interface_name(s)
```

その後に **root** でサービスを再起動します。

```
~]# systemctl --system daemon-reload  
~]# systemctl restart dhcpd
```

`/etc/systemd/system/dhcpd.service` ファイルの [Service] セクションの下では、`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 つ大きいポートにある **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。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、**DHCP** は **DHCP** リレーエージェントがリッスンするポート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**: 標準エラー記述子に **DHCP** サーバーデーモンを記録します。これは主にデバッグ用に使用されます。このオプションを指定しないと、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、**DHCP** サーバーを起動する度に、同じファイルが使用されるようにす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に関するメッセージ全体を表示しません。

10.3. DHCP リレーエージェント

DHCP リレーエージェント (**dhcrelay**) を使うと、**DHCP** サーバーがないサブネットから他のサブネットにある **DHCP** サーバーに **DHCP** および **BOOTP** リクエストのリレーができるようになります。

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、その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で送信されます。

IPv4 用の DHCP リレーエージェントである **dhcrelay** は、インターフェースが **/etc/sysconfig/dhcrelay** 内で **INTERFACES** ディレクティブを使って指定されている場合を除いて、**DHCPv4** および **BOOTP** リクエストをリッスンします。これについては、「[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はないので、**DHCPv6** リクエストをリッスンするインターフェースを指定する必要があります。これについては、「[dhcrelay の DHCPv6 リレーエージェントとしての設定](#)」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) として実行するか、**DHCPv6** リレーエージェント (**-6** 引数を使用) として実行することができます。使用法に関するメッセージを表示するには、**dhcrelay -h** コマンドを実行します。

10.3.1. dhcrelay の DHCPv4 および BOOTP リレーエージェントとしての設定

dhcrelay を **DHCPv4** および **BOOTP** モードで実行するには、リクエストを転送するサーバーを指定します。**root** で **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
```

[Service] セクション下にある **ExecStart** オプションを編集し、サーバー **IPv4** のアドレスを行末に追加します。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** リクエストをリッスンするインターフェースを指定するには、そのインターフェースを **-i** 引数をつけて **ExecStart** オプションに追加します (これがない場合は、すべてのインターフェースをリッスンすることになります)。例を示します。

```
ExecStart=/usr/sbin/dhcrelay -d --no-pid 192.168.1.1 -i em1
```

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。

変更を反映させるには、**root** でサービスを再起動します。

```
~]# systemctl --system daemon-reload
~]# systemctl restart dhcrelay
```

10.3.2. dhcrelay の DHCPv6 リレーエージェントとしての設定

dhcrelay を **DHCPv6** モードで実行するには、**-6** 引数をつけて (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) 「下方インターフェース」と (クライアントおよび他のリレーエージェントからのクエリーの転送先となる) 「上方インターフェース」を指定します。**root** で **dhcrelay.service** を **dhcrelay6.service** にコピーして、編集します。

```
~]# cp /lib/systemd/system/dhcrelay.service
/etc/systemd/system/dhcrelay6.service
~]# vi /etc/systemd/system/dhcrelay6.service
```

[Service] セクション下の **ExecStart** オプションを編集し、**-6** 引数をつけて「下方インターフェース」と「上方インターフェース」を追加します。例を示します。

```
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)** man ページを参照してください。

変更を反映させるには、**root** でサービスを再起動します。

```
~]# systemctl --system daemon-reload
~]# systemctl restart dhcrelay6
```

10.4. 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、すなわち複数のサブネットとして機能します。以下のセクションにあげる例では、**DHCP** サーバーが複数のネットワークで機能するように設定する方法、リッスンするネットワークインターフェースを選択する方法、ネットワークを移動するシステム用にネットワーク設定を定義する方法について詳述しています。

変更を行う前に、既存の **/etc/dhcp/dhcpd.conf** ファイルのバックアップを作成してください。

DHCP デーモンは、**/etc/dhcp/dhcpd.conf** ファイル内でサブネット宣言されているインターフェースでのみリッスンします。

以下は、**10.0.0.0/24** ネットワークの **eth0** と **172.16.0.0/24** ネットワークの **eth1** という 2 つのネットワークインターフェースを持つサーバー用の基本的な **/etc/dhcp/dhcpd.conf** ファイルです。複数の **subnet** 宣言により、複数のネットワークに対して異なる設定を定義することができます。

```
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;

subnet 宣言は、**DHCP** サーバーが機能するすべてのネットワークで必要です。複数のサブネットには、複数の **subnet** 宣言が必要です。**subnet** 宣言の範囲内に **DHCP** サーバーのネットワークインターフェースがない場合は、**DHCP** サーバーはそのネットワークでは機能しません。

subnet 宣言が1つしかなく、かつそのサブネットの範囲内にネットワークインターフェースがない場合には、**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 eth0 (0.0.0.0).
dhcpd: ** Ignoring requests on eth0.  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 eth1 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** 宣言内の **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、サブネット用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。指定された **IP** アドレスの範囲の中からアドレスがシステムに割り当てられます。

詳細情報は、**dhcpd.conf(5)** 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。

10.4.1. ホストの設定

変更を行う前に、既存の `/etc/sysconfig/dhcpd` および `/etc/dhcp/dhcpd.conf` ファイルのバックアップを作成してください。

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、接続するネットワークに応じて **IP** アドレスを同じシステム用に設定しています。

```
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、単一のシステム用に **IP** アドレスなどの特定のパラメーターを定義します。複数のホスト用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。

大半の **DHCP** クライアントは **host** 宣言内の名前を無視するため、他の **host** 宣言に対して一意である限りはどのような名前でも構いません。同じシステムを複数のネットワークに対して設定するには、各 **host** 宣言に異なる名前を使用します。異なる名前を使用しないと、**DHCP** デーモンは起動に失敗します。システムは **host** 宣言内の名前ではなく、**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。アドレスを確認するには、**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** オプションで指定され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** オプションで指定された **IP** アドレスプール外でなければなりません。

option ステートメントの最後にセミコロンがない場合、**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。

■

```
/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ー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ースを持つ単一のシステムを設定するため、各インターフェースは同一の **IP** アドレスを受け取ります。両方のネットワークインターフェー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** が最初のネットワークインターフェースで、**interface1** が2番目のインターフェースになります。異なる **hardware ethernet** オプションが各インターフェースを識別します。

このようなシステムを別のネットワークに接続するには、**host** 宣言をさらに追加します。ただし、以下の点に注意して下さい。

- ホストが接続されるネットワーク用の有効な **fixed-address** を割り当てます。
- **host** 宣言の名前は一意にします。

host 宣言で指定し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** に定義された **host interface0** 宣言が複数あるために生じたものです。

10.5. IPV6 の DHCP (DHCPV6)

ISC **DHCP** には、4.x リリース以降の **DHCPv6** サーバー、クライアント、およびリレーエージェント機能に対する **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ていますが、一度に管理できるのは1つのプロトコルのみです。両方をサポートするには、**IPv4** と **IPv6** に対して別個に起動する必要があります。たとえば、**DHCPv4** と **DHCPv6** についてそれぞれの設定ファイルである **/etc/dhcp/dhcpd.conf** と **/etc/dhcp/dhcpd6.conf** を編集して、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サンプルは、`/usr/share/doc/dhcp-version/dhcpd6.conf.example` にあります。

以下は、シンプルな **DHCPv6** サーバー設定ファイルの例です。

```
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";
}
```

10.6. 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**) により、IPv6 の Stateless Autoconfiguration に必要なルーター広告メッセージが送信されます。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、以下の手順を実施します。

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 eth0
{
    AdvSendAdvert on;
    MinRtrAdvInterval 30;
    MaxRtrAdvInterval 100;
    prefix 2001:db8:1:0::/64
    {
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr off;
    };
};
```

注記

ルーター広告と共に、さらに DNS リゾルバーを通知する場合は、`/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* に設定します。この設定により、DHCPv6 サービスが利用可能な場合、クライアントはルーター広告により IPv6 アドレスを自動的に取得することができます。DHCPv6 サービス設定の詳細については、「[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. 今すぐ **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および **radvd** デーモンにより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 デーモンの詳細については、**radvd(8)**、**radvd.conf(5)**、**radvdump(8)** man ページを参照してください。

10.7. その他のリソース

- **dhcpcd(8)** man ページ: **DHCP** デーモンの動作の仕組みを説明しています。
- **dhcpcd.conf(5)** man ページ: **DHCP** 設定ファイルの設定方法と例が含まれています。
- **dhcpcd.leases(5)** の man ページ: リースの永続的データベースについて説明しています。
- **dhcp-options(5)** の man ページ: **dhcpcd.conf** 内の **DHCP** オプション宣言の構文に関する説明と例が記載されています。
- **dhcrelay(8)** の man ページ: **DHCP** リレーエージェントとその設定オプションについて説明しています。
- **/usr/share/doc/dhcp-version/**: **DHCP** サービスの現行バージョン用のサンプルファイル、README ファイルおよびリリースノートが含まれています。

第11章 DNS サーバー

DNS (ドメインネームシステム) は、ホスト名とその **IP** アドレスの関連付けに使用される分散型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。これは、数字で表されるネットワークアドレスよりも通常は簡単なものです。システム管理者にとっては、**DNS**サーバー (ネームサーバーとも呼ぶ) を使用すると、名前ベースのクエリーに影響を与えることなくホスト用の **IP** アドレスを変更できるようになります。**DNS** データベースはドメイン名への **IP** アドレス解決だけでなく、DNSSEC が導入されるにつれてますます幅広く使用されるようになっていきます。

11.1. 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 つ以上の集中型サーバーを使用して実装されています。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。

11.1.1. ネームサーバーのゾーン

DNS サーバーでは、すべての情報が **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。この階層の各レベルは、ピリオド (.) で分けられます。たとえば、. で示される root ドメインは、**DNS** ツリーの根にあたり、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、root ドメイン (.) の子になるので、階層のレベル 1 になります。ドメイン名 **example.com** は、階層のレベル 2 になります。

例11.1 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、PR の **所有者** です。**86400** の値は、*time to live* (TTL) です。「Internet system」を意味する **IN** の文字は、PR の *class* を示しています。**A** の文字は PR の *type* を示しています (この例では、ホストアドレス)。ホストアドレスの **192.0.2.1** は、この PR の最後のセクションに含まれているデータです。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。ゾーンファイルには、各ゾーンのリソースレコードの定義が含まれています。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、またプロビジョニングアプリケーションの管理者は、組織内で使用するホスト名の形式について合意する必要があることに留意してください。ホスト名の形式については、「[推奨される命名プラクティス](#)」を参照してください。

11.1.2. ネームサーバーのタイプ

ネームサーバーの設定タイプには以下のように 2 つあります。

authoritative

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。

11.1.3. ネームサーバーとしての BIND

BIND は一連の DNS 関連プログラムで構成されています。これには **named** と呼ばれるネームサーバー、**rndc** という管理ユーティリティ、**dig** というデバッグツールが含まれています。Red Hat Enterprise Linux におけるサービスの実行方法については『[Red Hat Enterprise Linux 7 システム管理者のガイド](#)』を参照してください。

11.2. BIND

本セクションでは、Red Hat Enterprise Linux に含まれている DNS サーバーの **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。

11.2.1. 空白ゾーン

BIND は多くの「空白ゾーン」を設定し、再帰サーバーが不要なクエリーを、処理できない (このために遅延が発生したり、クエリーを送信したクライアントに SERVFAIL が返される) インターネットサーバーに送信することを防ぎます。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。設定オプションの **empty-zones-enable** は空白ゾーンを作成するかどうかを制御し、**disable-empty-zone** はさらに、使用されるデフォルトの接頭辞のリストから 1 つ以上の空白ゾーンを無効にします。

『[RFC 1918](#)』接頭辞用に作成された空白ゾーンの数が増加しており、**BIND 9.9** およびそれ以降のユーザーは、**empty-zones-enable** が未指定の場合 (デフォルトで **yes**) と明示的に **yes** に設定されている場合の両方で『[RFC 1918](#)』空白ゾーンを目にすることになります。

11.2.2. named サービスの設定

named サービスは起動時に、[表11.1「named サービスの設定ファイル」](#)に記載ファイルから設定を読み込みます。

表11.1 **named** サービスの設定ファイル

パス	説明
<code>/etc/named.conf</code>	主要設定ファイル
<code>/etc/named/</code>	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 (`{` と `}`)で囲まれた入り子オプションを持つステートメントの集合で構成されています。このファイルの編集時には、構文エラーを起こさないように注意してください。エラーがあると **named** サービスは起動しません。標準的な `/etc/named.conf` ファイルは以下のような構成になります。

```
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 パッケージがインストールされている場合、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** コマンドを使って上記の設定ファイルをマウントするので、この環境外で設定が管理できます。マウントは自動的に行われるので、**/var/named/chroot/** ディレクトリーにはなにもコピーする必要はありません。**chroot** 環境で **BIND** が実行されている場合は、この設定ファイルの管理については特別な作業が不要となり、維持が簡単になります。**chroot** 以外の環境で **BIND** を実行する際と同様の作業になります。

以下のディレクトリーは、**/var/named/chroot/** 下にある対応するマウントポイントディレクトリーが空の場合、**/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)

/var/named/chroot/ にターゲットファイルがない場合は、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場合は、バックアップコピーを作成し、その上でオリジナル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、BIND の設定ファイル **/etc/named.conf** を **chroot** 環境で実行中に Vim で編集するには、**root** で以下のコマンドを実行します。

```
~]# vim -c "set backupcopy=yes" /etc/named.conf
```

11.2.2.1. 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、以下のコマンドを実行して、まず **named** サービスが実行中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 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
```

その後に **named-chroot** サービスを有効にするために、**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
```

11.2.2.2. 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が一般的に **/etc/named.conf** で使用されます。

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。また、*match-element* オプションは通常、個別の **IP** アドレス (**10.0.1.1** など) または、CIDR (*Classless Inter-Domain Routing*) ネットワーク表記 (たとえば、**10.0.1.0/24**) です。定義済みキーワードの一覧については [表11.2「事前定義のアクセス制御リスト」](#) を参照してください。

表11.2 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する 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 アドレスにもマッチしません。

acl ステートメントは **options** などの他のステートメントとの併用すると特に便利です。例 11.2「**options** と併用して **acl** を使用する」では、**black-hats** と **red-hats** の2つのアクセス制御リストを定義し、**red-hats** に通常アクセスを許可する一方で **black-hats** をブラックリストに追加します。

例11.2 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 ステートメントにより、ファイルを **/etc/named.conf** 内に含めることができるので、機密性のあるデータを制限のあるパーミッションで別のファイルに配置することができます。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11.3 ファイルを **/etc/named.conf** に含める

```
include "/etc/named.rfc1912.zones";
```

options

options ステートメントにより、グローバルサーバー設定オプションを定義できるとともに他のス

テートメントのデフォルトを設定することもできます。これを使うと、**named** の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、その他多くを指定することができます。以下の形式を取ります。

```
options {
    option;
    ...
};
```

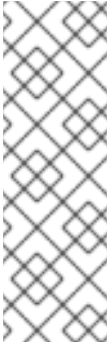
頻繁に使用される *option* ディレクティブの一覧については、以下の [表11.3「一般的な設定オプション」](#) を参照してください。

表11.3 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。これはアクセス制御リスト、 IP アドレスの集合、または CIDR 表記にあるネットワーク群などを受け付けます。デフォルトではすべてのホストが許可されています。
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 のみが許可されています。
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 サービス用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	DNSSEC を介してリソースが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	解決のためにリクエストの転送先とな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	<p>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受け付けます。</p> <ul style="list-style-type: none"> • first: サーバーは、それ自身で名前の解決を試行する前に forwarders ディレクティブに一覧表示されているネームサーバーにクエリーします。 • only: forwarders ディレクティブに一覧表示されたネームサーバーにクエリーすることができない時は、サーバーはそれ自身での名前解決を試行しません。
listen-on	<p>クエリーをリッスンする IPv4 ネットワークインターフェースを指定します。ゲートウェイとしても機能する DNS サーバー上では、このオプションを使用して単一ネットワークから発信するクエリーにのみ応答することができます。デフォルトで、すべての IPv4 インターフェースが使用されます。</p>
listen-on-v6	<p>クエリーをリッスンする IPv6 ネットワークインターフェースを指定します。ゲートウェイとしても機能する DNS サーバー上では、このオプションを使用して単一ネットワークから発信するクエリーにのみ応答することができます。デフォルトで、すべての IPv6 インターフェースが使用されます。</p>
max-cache-size	<p>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。</p>
notify	<p>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受け付けます。</p> <ul style="list-style-type: none"> • 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: サーバーはいずれの セカンダリーネームサーバーにも通知しません。 • master-only: サーバーはゾーン用のプライマリーサーバーにのみ通知します。 • explicit: サーバーは、ゾーンステートメント内の also-notify リストで指定されたセカンダリーサーバーにのみ通知します。
pid-file	<p>named サービスで作成されたプロセス ID ファイルの場所を指定します。</p>
recursion	<p>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。</p>

オプション	説明
statistics-file	統計ファイルの代替の場所を指定します。デフォルトでは、 /var/named/named.stats ファイルがデフォルトで使用されています。



注記

named がランタイムデータ用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/** から新たな場所の **/run/named/** に移りました。この結果、PID ファイルはデフォルトの **/var/run/named/named.pid** から新しい場所の **/run/named/named.pid** に移りました。また、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11.4「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

分散型サービス妨害 (DDoS) 攻撃を阻止するには、**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用にのみ再帰 DNS サービスを制限することが推奨されます。

利用可能な全オプションの一覧は、「[インストールされているドキュメント](#)」内で参照されている『BIND 9 Administrator Reference Manual』および **named.conf** man ページを参照してください。

例11.4 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 ステートメントは、その設定ファイルの場所やゾーン特有のオプションなどのゾーンの特性の定義を可能にし、グローバル **options** ステートメントの上書きに使用できます。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あり、`zone-class` はゾーンのオプションクラスであり、そして `option` は 表11.4 「Zone ステートメントで一般的に使用されるオプション」で説明してあるように `zone` ステートメントのオプションです。

`zone-name` 属性は `/var/named/` ディレクトリーに配置されている該当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たデフォルトの値なので、特に重要です。`named` デーモンはゾーンの名前を、ゾーンファイル内に一覧表示された非完全修飾型のドメイン名のいずれかに追記します。たとえば、`zone` ステートメントが `example.com` 用にネームスペースを定義する場合、`example.com` を `zone-name` として使用すると、それを `example.com` ゾーンファイル内のホスト名の末尾に配置することができます。

ゾーンファイルの詳細情報については、「[ゾーンファイルの編集](#)」を参照してください。

表11.4 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なければ、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) 」の説明にあるように TSIG キーを使用してください。
file	ゾーンの設定データを収納している named 作業ディレクトリー内のファイル名を指定します。
masters	信頼できるゾーン情報の要求元となる 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。

オプション	説明
notify	<p>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受け付けます。</p> <ul style="list-style-type: none"> • 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: サーバーはいずれの セカンダリーネームサーバーにも通知しません。 • master-only: サーバーはゾーン用のプライマリーサーバーにのみ通知します。 • explicit: サーバーは、ゾーンステートメント内の also-notify 一覧に指定してあるセカンダリーサーバーにのみ通知します。
type	<p>ゾーンのタイプを指定します。以下のオプションを受け付けます。</p> <ul style="list-style-type: none"> • delegation-only: COM、NET、ORG などのインフラストラクチャーゾーンの委任ステータスを強制します。明示的あるいは暗示的な委任のない受信回答は 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する 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: ゾーンが他の方法で認知されていない時にクエリーを解決する root ネームサーバーへポイントするために使用される特殊タイプのゾーンです。 • master: このゾーン用の権威としてネームサーバーを指定します。ゾーンの設定ファイルがシステム上に存在する場合は、ゾーンは master としてセットされる必要があります。 • slave: このゾーン用のスレーブとしてネームサーバーを指定します。マスターサーバーは masters ディレクティブ内に指定されています。

プライマリーまたはセカンダリーのネームサーバーの `/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、修正、または削除が含まれ、通常は **zone** ステートメントオプションの小さなサブセットのみが、ネームサーバーの効率的な機能のために必要となります。

例11.5「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** として識別されており、タイプは **master** にセットされて、**named** サービスは `/var/named/example.com.zone` ファイルを読み込むように指示されています。これはまた、ゾーンの転送にセカンダリーネームサーバー (**192.168.0.2**) のみを許可します。

例11.5 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** ステートメントは少し異なります。タイプは **slave** に設定され、**masters** ディレクティブは **named** にマスターサーバーの **IP** アドレスを伝えます。

例11.6「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、**192.168.0.1** IP アドレスにあるプライマリーサーバーに対して **example.com** ゾーンの情報についてクエリーをするように設定されています。受信した情報はその後、**/var/named/slaves/example.com.zone** ファイルに保存されます。すべてのスレーブゾーンを **/var/named/slaves/** ディレクトリーに配置する必要があることに注意してください。これを行わないと、サービスはゾーン転送に失敗します。

例11.6 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; };
};
```

11.2.2.3. 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、**/etc/named.conf** 内では通常多くは使用されません。

controls

controls ステートメントにより各種設定が可能になり、**named** サービスを管理するための **rndc** コマンドの使用に必要な様々なセキュリティ要件を設定できるようになります。

rndc ユーティリティーとその使用法の詳細については、「[rndc ユーティリティーの使用](#)」を参照してください。

key

key ステートメントを使うと、特定のキーを名前前で定義できるようになります。キーは、安全な更新や、あるいは、**rndc** コマンドの使用など各種動作を認証するために使用されます。2つのオプションが **key** と使用されます。

- **algorithm *algorithm-name***: 使用されるアルゴリズムのタイプ (たとえば、**hmac-md5**)。
- **secret "*key-value*"**: 暗号化キーです。

rndc ユーティリティーとその使用法の詳細については、「[rndc ユーティリティーの使用](#)」を参照してください。

logging

logging ステートメントを使うと チャンネル (*channels*) と呼ばれる複数のログタイプを使用できるようになります。**channel** オプションをステートメント内で使用すると、それ自体のファイル名 (**file**)、サイズ制限 (**size**)、バージョン番号 (**version**)、および重要度レベル (**severity**) などが

あるカスタマイズしたログのタイプを構築することができます。カスタムチャンネルが定義されると、**category** オプションの使用でチャンネルが分類化され、**named** サービスの再起動時にログインが開始されます。

デフォルトでは、**named** は標準メッセージを **rsyslog** デーモンに送信して、受信したデーモンはメッセージを **/var/log/messages** に配置します。数種類の標準チャンネルが各種重要度レベルで BIND に組み込まれています。それらの重要度レベルとして、**default_syslog** (情報ロギングメッセージを処理) と **default_debug** (特にデバッグメッセージを処理) などがあります。**default** と呼ばれるデフォルトカテゴリは、組み込み型チャン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 ログ作成の詳細については、「[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対しての反応の仕方に影響する、特に通知とゾーン転送に関して影響するオプションを指定できるようになります。

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。これは、**one-answer** (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** オプションがより効率的ですが、BIND の旧バージョンではサポートされていないことに注意して下さい。

trusted-keys

trusted-keys 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。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。

view

view ステートメントにより、ネームサーバーをクエリーしているホストが存在するネットワークに応じて特別な表示を作成できるようになります。これによって、一部のホストはゾーンに関して1つの応答を受信することができるようになり、他のホストは全く異なる情報を受信することができるようになります。別の観点から、一定のゾーンは特定の信頼されるホストにだけ利用可能であり、信頼できないホストは他のゾーンのクエリーしかできない可能性もあります。

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** オプションを使うと、特定の表示に適用する **IP** アドレスを指定することができます。**options** ステートメントが1つの表示内で使用された場合は、すでに設定済みのグローバルオプションを上書きします。最後に、ほとんどの **view** ステートメントは、**match-clients** 一覧に適用される複数の **zone** ステートメントを含んでいます。

view ステートメントが一覧表示される順序が重要であることに留意してください。特定クライアントの **IP** アドレスにマッチする最初のステートメントが使用されます。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。

11.2.2.4. コメントタグ

ステートメントのほかに、**/etc/named.conf** ファイルにはコメントも含まれています。コメントは **named** サービスには無視されますが、ユーザーに追加情報を提供する時に役に立ちます。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いずれもその行末までコメントとみなされます。例を示します。

```
notify yes; // notify all secondary nameservers
```

#

文字の後のテキストはいずれもその行末までコメントとみなされます。例を示します。

```
notify yes; # notify all secondary nameservers
```

/* と */

/* と */ によって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。例を示します。

```
notify yes; /* notify all secondary nameservers */
```

11.2.3. ゾーンファイルの編集

「[ネームサーバーのゾーン](#)」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。情報はデフォルトで `/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** ステートメント内の **file** オプションに従って命名されます。通常は、**example.com.zone** などのようにドメインに関連付けられ、ゾーンデータを含むファイルとして識別できる方法で命名されます。

表11.5 named サービスのゾーンファイル

パス	説明
<code>/var/named/</code>	named サービスの作業ディレクトリーです。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。
<code>/var/named/slaves/</code>	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 サービスによる書き込みが可能です。
<code>/var/named/dynamic/</code>	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 サービスによる書き込みが可能です。
<code>/var/named/data/</code>	様々な統計とデバッ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 サービスによる書き込みが可能です。

ゾーンファイルはディレクティブとリソースの記録で構成されています。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。

11.2.3.1. 一般的なディレクティブ

ディレクティブはドルマーク記号 (\$) で始まり、その後にディレクティブ名が続きます。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 ディレクティブにより、それが出現する場所にもう 1 つのファイルを含めることができるため、他のゾーンセッティングは別個のゾーンファイルに保存できるようになります。

例11.7 \$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うと、ホスト名だけの非完全修飾型の記録へドメイン名を追記できるようになります。デフォルトではゾーン名が使用されるので、**/etc/named.conf** 内で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不要です。

例11.8「**\$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る名前でピリオド (.) で終了していない名前はいずれも **example.com** が追記されます。

例11.8 \$ORIGIN ディレクティブの使用

```
$ORIGIN example.com.
```

\$TTL

\$TTL ディレクティブにより、ゾーン用のデフォルト *TTL* (Time to Live) 値をセットできるようになります。つまり、ゾーン記録が有効である時間の長さのセッティングです。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することができるようになり、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11.9 \$TTL ディレクティブの使用

```
$TTL 1D
```

11.2.3.2. 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11.10「リソースレコードの使用」では、**server1.example.com** 用の要求は、**10.0.1.3** または **10.0.1.5** を指しています。

例11.10 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 サーバー用の **www** のように、共通の命名基準を使用するサービスを指す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11.11「CNAME リソースレコードの使用」では、**A** レコードがホスト名を **IP** アドレスにバインドし、**CNAME** レコードが一般的に使用される **www** ホスト名をそれに向けています。

例11.11 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。最小の *preference-value* を持つ **MX** リソースレコードが他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11.12 「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受信時には最初の **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11.12 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一覧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はありません。両方とも正当と考慮されます。

例11.13 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** アドレスの末尾の番号です。*FQDN-of-system* は完全修飾型ドメイン名 (FQDN) になります。

PTR レコードは主に逆引き名前解決に使用されます。これは **IP** アドレスを特定の名前に向けます。**PTR** レコードの使用例については「[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。

- @ シンボルは **\$ORIGIN** ディレクティブ (または**\$ORIGIN** ディレクティブがセット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たネームスペースとして配置します。
- *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度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。否定的回答のキャッシュ化は最大で 3 時間に設定できます (**3H**)。

BIND の設定時には、すべての時間は秒で指定されます。しかし、秒以外の時間単位を指定するのに短縮形を使用することができます。たとえば、分 (**M**)、時間 (**H**)、日 (**D**)、および週 (**W**) です。[表 11.6「秒表示とその他の時間単位」](#)では時間数を秒で示し、さらにその同時間を他の形式で示しています。

表11.6 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

秒	他の時間単位
86400	1D
259200	3D
604800	1W
31536000	365D

例11.14 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
```

11.2.3.3. 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便利です。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。例を示します。

```
604800 ; expire after 1 week
```

11.2.3.4. 使用法の例

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。

11.2.3.4.1. 単純なゾーンファイル

例11.15「単純なゾーンファイル」では、標準のディレクティブと**SOA** 値の使用を提示しています。

例11.15 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.
```

```

    dns1      IN  NS      dns2.example.com.
              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 **dns1.example.com** と **dns2.example.com** として設定されており、これらはそれぞれ **A** レコードを使用して **10.0.1.1** と **10.0.1.2** の **IP** アドレスに結合されています。

MX レコードで設定されているメールサーバーは、**A** レコードを介して **mail** と **mail2** に向けられています。これらの名前はトレーリングピリオドで終了していないため、**\$ORIGIN** ドメインがその後に配置されており、それらを **mail.example.com** および **mail2.example.com** に広げています。

www.example.com (WWW) などの標準の名前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ビスを指すようにしてあります。

このゾーンファイルは、以下のような **zone** ステートメントが **/etc/named.conf** ファイルに追加される場合に使用されます。

```

zone "example.com" IN {
    type master;
    file "example.com.zone";
    allow-update { none; };
};
```

11.2.3.4.2. 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、特定のネームスペース内の **IP** アドレスを完全修飾型ドメイン名 (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に非常に似ていますが、[例 11.16 「逆引き名前解決ゾーンファイル」](#)にあるように **IP** アドレスを完全修飾型ドメイン名にリンクす

るために **PTR** リソースレコードが使われている点が異なります。

例11.16 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 **zone** ステートメントが **/etc/named.conf** ファイルに追加される場合に使用されます。

```
zone "1.0.10.in-addr.arpa" IN {
    type master;
    file "example.com.rr.zone";
    allow-update { none; };
};
```

ゾーン名以外は、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は、**IP** アドレスの最初の3つのブロックが逆になっていて、その後に **.in-addr.arpa** が続く必要があります。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** 番号のシングルブロックがそのゾーンと関連付けられます。

11.2.4. rndc ユーティリティーの使用

rndc ユーティリティーは、ローカルとリモートマシンの両方から **named** サービスの管理を可能にす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]
```

11.2.4.1. ユーティリティーの設定

サービスへの未承認のアクセスを防止するには、**named** は選択したポート (デフォルトでは **953**) をリッスンするように設定し、このサービスと**rndc** ユーティリティーの両方が同じキーを使用する必要があります。

表11.7 関連ファイル

パス	説明
<code>/etc/named.conf</code>	named サービス用のデフォルト設定ファイル
<code>/etc/rndc.conf</code>	rndc ユーティリティー用のデフォルト設定ファイル
<code>/etc/rndc.key</code>	デフォルトキーの場所

rndc 設定は `/etc/rndc.conf` に配置されています。ファイルが存在しない場合は、ユーティリティーは、**rndc-confgen -a** コマンドを使用してインストールプロセス中に自動的に生成されていて `/etc/rndc.key` にあるキーを使用します。

named サービスは、「[その他のステートメントタイプ](#)」の説明にある `/etc/named.conf` 設定ファイル内の **controls** 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ない場合、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ることになり、`/etc/rndc.key` にあるキーが使用されます。

このトピックの詳細については、「[その他のリソース](#)」にある『BIND 9 Administrator Reference Manual (管理者参照マニュアル)』と `man` ページを参照してください。



重要

権限のない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` ファイルの読み取りをできるようにします。

```
~]# chmod o-rwx /etc/rndc.key
```

11.2.4.2. サービスステータスの確認

named サービスの現在の状態をチェックするには、以下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
```

11.2.4.3. 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することができます。

単独ゾーンをリロードするには、**reload** コマンドの後にその名前を指定します。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
```

これが完了したら、**thaw** コマンドを実行して **DDNS** を再度有効にしてゾーンをリロードします。

```
~]# rndc thaw localhost
The zone reload and thaw was successful.
```

11.2.4.4. 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。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 ステートメント内で **auto-dnssec** オプションを **maintain** に設定する必要があることに注意してください。例を示します。

```
zone "localhost" IN {
    type master;
    file "named.localhost";
    allow-update { none; };
    auto-dnssec maintain;
};
```

11.2.4.5. 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 内でこのオプションを設定する方法については、「[一般的なステートメントのタイプ](#)」の **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。

11.2.4.6. 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 (すでに有効な場合は無効にする) には、**root** で以下のコマンドを実行します。

```
~]# 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](#)」にあるように **status** コマンドを使用します。

11.2.5. 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一覧は、「[一般的なリソースレコード](#)」を参照してください。

11.2.5.1. 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11.17「[ネームサーバールックアップのサンプル](#)」では、**dig** ユーティリティーが **example.com** 用のネームサーバーを表示するために使用されています。

例11.17 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
```

11.2.5.2. 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、以下の形式のコマンドを使用します。

```
dig name A
```

例11.18 「IP アドレス検索のサンプル」では、**dig** ユーティリティを使用して **example.com** の **IP** アドレスを表示しています。

例11.18 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
```

11.2.5.3. 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、以下の形式のコマンドを使用します。

```
dig -x address
```

例11.19「ホスト名検索のサンプル」では、**dig** ユーティリティを使用して **192.0.32.10** に割り当てられたホスト名を表示しています。

例11.19 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.

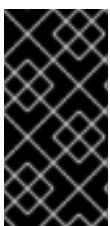
;; AUTHORITY SECTION:
32.0.192.in-addr.arpa. 21600 IN      NS       b.iana-servers.org.
32.0.192.in-addr.arpa. 21600 IN      NS       c.iana-servers.net.
32.0.192.in-addr.arpa. 21600 IN      NS       d.iana-servers.net.
32.0.192.in-addr.arpa. 21600 IN      NS       ns.icann.org.
32.0.192.in-addr.arpa. 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
```

11.2.6. BIND の高度な機能

ほとんどの BIND 実装では、**named** のみを使用して名前解決サービスを提供したり、特定ドメイン用の権威として機能させます。しかし、BIND バージョン 9 には数種の高度な機能が含まれており、より安全で効率的な **DNS** サービスを可能にしています。



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 (管理者参照マニュアル)』でより詳細に説明されています。

11.2.6.1. 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要注意 **DNS** エントリを拒否するために、そしてそれと同時にローカルネットワーク内のクライアントからのクエリを受け付けるために使用されます。

複数表示を設定するには、**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** アドレスかネットワーク全体とマッチするようにし、それらに特別オプションとゾーンデータを与えます。

11.2.6.2. 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マスタ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。

11.2.6.3. 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。攻撃者はゾーン転送のために **IP** アドレスにアクセスする必要があるだけでなく、秘密鍵を知る必要があるため、これにより標準の **IP** アドレスベースの転送認証メソッドが強化されます。

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

安全でないネットワーク上での通信時には、**IP** アドレスベース認証のみに頼らないでください。

11.2.6.4. DNSSEC (DNS Security Extensions)

Domain Name System Security Extensions ドメイン名システムセキュリティ拡張機能 (DNSSEC) は、**DNS** データの発信元認証、認証による存在否定、およびデータの整合性を提供します。特定のドメインが安全としてマークされている時、検証に失敗した各リソースレコードに **SERFVAIL** 応答が返されます。

DNSSEC 署名のドメイン、または DNSSEC 認識の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](#)」にある **dig** ユーティリティーを使用することができます。役に立つオプションとして、**+dnssec** (DNSSEC OK を設定することで、DNSSEC 関連のリソースレコードを要求)、**+cd** (応答を検証しないように再帰ネームサーバーに指示)、そして **+bufsize=512** (一部のファイヤーウォールを通過するためにパケットサイズを 512B に変更) があります。

11.2.6.5. インターネットプロトコルバージョン 6 (IPv6)

Internet Protocol version 6 インターネットプロトコルバージョン 6 (IPv6) は [表11.3「一般的な設定オプション」](#) に説明してあるように AAAA リソースレコードの使用、および **listen-on-v6** ディレクティブを介してサポートされています。

11.2.7. 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` ファイル内のセミコロンの欠如や、不一致な弓形括弧は **named** サービスの開始を阻止してしまいます。

ピリオド (. 記号) の正しい使用

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。これが欠如していると、**named** サービスはゾーン名、または **\$ORIGIN** の値を追記してそれを完結しようとします。

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

ファイヤーウォールが、**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は、ファイヤーウォールの設定変更が推奨されます。



警告

DNS クエリに固定 **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ようになるセキュリティ脆弱性につながる可能性があります。これを防止するには、デフォルトで **DNS** がランダム短期ポートから送信するようにします。ランダムな **UDP** ソースポートからの送信クエリを許可するようにファイヤー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。

11.2.8. その他のリソース

以下の情報資料は、BIND に関するその他のリソースを提供します。

11.2.8.1. 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。以下の各項目には、*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/

named 設定ファイルのサンプルが格納されているディレクトリーです。

rndc(8)

rndc ネームサーバー制御ユーティリティーの man ページで、使用法に関するドキュメンテーションが含まれています。

named(8)

インターネットドメインネームサーバー **named** の man ページには、BIND ネームサーバーデーモンの制御に使用可能な各種引数についてのドキュメントが含まれています。

lwresd(8)

軽量のリゾルバーデーモン **lwresd** の man ページには、このデーモンとその使用法についてのドキュメントが含まれています。

named.conf(5)

この 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総合的一覧があります。

rndc.conf(5)

この 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総合的一覧があります。

11.2.8.2. オンラインリソース

<https://access.redhat.com/site/articles/770133>

chroot 環境で BIND を実行することに関する Red Hat ナレッジベースアーティクルです。Red Hat Enterprise Linux 6 との違いも含まれています。

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12章 SQUID

本章は、Web クライアントの高性能プロキシキャッシュサーバーである **Squid** について取り扱います。本項では、**Squid** の設定方法、認証方法、**Squid** でアクセスをブロックする方法について説明しています。

12.1. SQUID の概要

Squid は、ページがより早く読み込まれるようにキャッシュして Web サイトの操作を最適化し、ユーザーが最も頻繁にアクセスするページのレスポンス時間を短縮するプロキシ Web サーバーのことで、**Squid** は、Hypertext Transport Protocol (HTTP)、File Transfer Protocol (FTP)、その他の有名なプロトコルにプロキシやキャッシュサービスを提供します。**Squid** は多くの場合、反復する要求をキャッシュして Web サーバーを加速したり、トラフィックのフィルタリングでセキュリティを保護したり、固有のページへのアクセスを制限したりするために使用されます。

Squid は FTP、gopher、ICAP、ICP、HTCP、HTTP データオブジェクトをサポートします。

Squid は以下で構成されます。

- 主要なサーバープログラム **Squid**
- オプションのカスタム処理および認証プログラム
- 管理およびクライアントツール

12.2. SQUID のインストールおよび実行

Red Hat Enterprise Linux では squid パッケージで **Squid** キャッシングプロキシが提供されます。**rpm -q squid** コマンドを実行して、squid パッケージがインストールされているかどうかを確認してください。インストールされていない場合には、**root** ユーザーとして以下のコマンドを入力してインストールしてください。

```
~]# yum install squid
```

root ユーザーとして **systemctl start squid** コマンドを実行して **Squid** を起動します。

```
~]# systemctl start squid
```

Squid は、マシン上にある全ネットワークインターフェースのポート 3128 (デフォルト) のリッスンを開始します。

systemctl status squid コマンドを実行して、**Squid** が実行されていることを確認します。出力例を以下に添付しています。

```
~]# systemctl status squid
● squid.service - Squid caching proxy
   Loaded: loaded (/usr/lib/systemd/system/squid.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2016-04-06 13:15:05 CEST; 2min 17s ago
 [output truncated]
```

ps -eZ | grep squid コマンドを実行して **Squid** のプロセスを表示します。

```
~]# ps -eZ | grep squid
unconfined_u:system_r:squid_t:s0 2522 ? 00:00:00 squid
unconfined_u:system_r:squid_t:s0 2524 ? 00:00:00 squid
unconfined_u:system_r:squid_t:s0 2526 ? 00:00:00 ncsa_auth
unconfined_u:system_r:squid_t:s0 2527 ? 00:00:00 ncsa_auth
unconfined_u:system_r:squid_t:s0 2528 ? 00:00:00 ncsa_auth
unconfined_u:system_r:squid_t:s0 2529 ? 00:00:00 ncsa_auth
unconfined_u:system_r:squid_t:s0 2530 ? 00:00:00 ncsa_auth
unconfined_u:system_r:squid_t:s0 2531 ? 00:00:00 unlinkd
```

コマンドライン環境から **Squid** の詳細にわたるパフォーマンス統計を取得するには、**Squid** サービスにアクセスして統計を取得可能な **squidclient** ツールを使用します。たとえば、一般的なパフォーマンス統計を取得するには、**Squid** サーバーで以下のコマンドを入力します。

```
~]# squidclient -p squid-port mgr:info
```

Squid を停止するには、以下のコマンドを発行します。

```
~]# systemctl stop squid
```

Squid のログファイル

Squid プロキシサーバーのログファイルは、**/var/log/squid/** ディレクトリに保存されています。プロキシされた要求の情報を格納するログファイルは **/var/log/squid/access.log** ファイルです。

12.3. SQUID の設定

Squid を設定するには、設定ファイルのディレクティブを調節します。**Squid** は通常、特定のネットワークの要件に合わせて、コマンドラインを使用して **/etc/squid/squid.conf** に配置されている **Squid** 設定ファイルを編集して設定します。この **squid.conf** ファイルには推奨の最小構成が含まれています。

12.3.1. 基本設定および /etc/squid/squid.conf

手順12.1 基本設定

1. 元の config ファイルをバックアップします。

```
mv /etc/squid/squid.conf /etc/squid/squid.conf.org
```

2. 以下の内容を使用して、新規の **/etc/squid/squid.conf** ファイルを作成します。**mynetwork** のアクセス制御リスト (ACL) の行を編集して、ローカルネットワークのソースネットワークを定義します。これは、クライアントのシステムが **Squid** サーバーをプロキシとして使用するネットワークのことです。



注記

Squid はファイルを最初から読み込んでいくので **/etc/squid/squid.conf** 設定ファイルのアイテムの順番は重要です。

```
acl mynetwork src xxx.xxx.xxx.0/24
```



```

http_access allow mynetwork

#defaults
acl localnet src 10.0.0.0/8
acl localnet src 172.16.0.0/12
acl localnet src 192.168.0.0/16
acl localnet src fc00::/7
acl localnet src fe80::/10
acl SSL_ports port 443
acl Safe_ports port 80
acl Safe_ports port 21
acl Safe_ports port 443
acl Safe_ports port 70
acl Safe_ports port 210
acl Safe_ports port 1025-65535
acl Safe_ports port 280
acl Safe_ports port 488
acl Safe_ports port 591
acl Safe_ports port 777
acl CONNECT method CONNECT
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localnet
http_access allow localhost
http_access deny all
http_port 3128
hierarchy_stoplist cgi-bin ?
coredump_dir /var/spool/squid
refresh_pattern ^ftp:      1440      20% 10080
refresh_pattern ^gopher:   1440      0%  1440
refresh_pattern -i (/cgi-bin/|\?) 0 0%  0
refresh_pattern .          0        20% 4320

```

3. サービスを起動して、ブート時に有効になるように設定します。

```

~]# systemctl enable squid
~]# systemctl start squid

```

4. ファイアウォールを有効化する場合には、**Squid** ポートを許可します。

```

~]# firewall-cmd --add-port=3128/tcp --permanent

```

5. Web ブラウザーがプロキシを使用するように設定します。これは、使用するブラウザーとバージョンにより左右されます。たとえば、バージョン 46.0.0 の **Firefox** を設定するには以下を実行してください。

手順12.2 プロキシを使用して Firefox を設定する

1. Firefox の右上端にあるメニューから **設定** を選択します。左のタブから **詳細** を選択して、上部のバーにあるタブから **ネットワーク** を選択します。
2. **接続** のセクションで、**接続設定** を開きます。

3. 開いた新規ウィンドウで、**手動でプロキシーを設定する** にチェックを入れて、**HTTP プロキシー** フィールドに接続先のプロキシーサーバーを入力します。固有のポートを入力する必要がある場合には、**ポート** フィールドに入力します。

`/etc/squid/squid.conf` に関する詳しい情報は **squid(8)** の man ページを参照してください。

12.3.2. HTTP プロキシーサーバーとしての **Squid** を設定する

手順12.3 HTTP プロキシーサーバーとしての **Squid** を設定する

1. `/etc/squid/squid.conf` ファイルの一番上に以下の行を追加してください。IP アドレスの例は実際の IP アドレスに置き換えてください。

```
cache_dir ufs /var/spool/squid 500 16 256
acl my_machine src 192.0.2.21 # Replace with your IP address
http_access allow my_machine
```

2. 以下のコマンドを使用してキャッシュディレクトリーを作成します。

```
~]# systemctl restart squid
```

Squid は、マシン上にある全ネットワークインターフェースのポート 3128 (デフォルト) のリッスンを開始するようになります。

3. **Firefox** などのブラウザーがマシンの IP アドレスとポート 3128 のホストで HTTP プロキシーサーバーとして **Squid** を使用するよう設定します。詳細は [手順12.2「プロキシーを使用して Firefox を設定する」](#) を参照してください。

12.3.2.1. HTTP ポートの設定

`http_port` ディレクティブを使用して、**Squid** がクライアントの接続をリッスンするポートを指定します。デフォルトの動作は、マシンで利用可能なすべてのインターフェースのポート 3128 をリッスンします。異なるインターフェースの異なるポートや、複数のインターフェースを強制的に **Squid** にリッスンさせることができます。

例12.1 HTTP ポートの指定

`/etc/squid/squid.conf` を開いて、適切な行を編集します。この例では、**Squid** はポート 8080 をリッスンするように設定します。

```
# Squid normally listens to port 3128
http_port 8080
```

Squid サーバーは、同時に複数のポートをリッスンすることができます。

例12.2 2 つ以上のポートの指定

以下の設定では、**Squid** はポート 8080 と 9090 の両方をリッスンします。

```
http_port 8080 9090
```



注記

以下を実行し、**Squid** サーバーを再起動して新規設定を適用するのは忘れないでください。

```
~]# systemctl restart squid
```

/etc/squid/squid.conf で IP アドレスおよびポートの組み合わせも指定してください。通常、マシンに複数のインターフェースがあり、**Squid** にローカルエリアネットワーク (LAN) に接続されているインターフェースのみをリッスンするようにする場合にはこのアプローチを使用します。

例12.3 IP アドレスの設定

以下のコマンドは **Squid** に対して IP アドレスが 192.0.2.25 のインターフェースのポート 3128 をリッスンするように指示を出します。

```
http_port 192.0.2.25:3128
```

さらに、**ホスト名** と **ポート** を使用して、**http_port** を指定できます。ホスト名は、**Squid** により IP アドレスに変換され、特定の IP アドレスのポート 8080 をリッスンします。

```
http_port myproxy.example.com:8080
```

http_port ディレクティブの別の特長として、別の行で複数の値を受け入れることができます。以下の行は、**Squid** が 3 つの異なる IP アドレスとポートの組み合わせをリッスンするようにトリガーします。一般的にこれは、異なる LAN にクライアントがあり、プロキシサーバーに異なるポートを使用するように設定されている場合に役立ちます。**/etc/squid/squid.conf** ファイルを以下のように編集します。

```
http_port 192.0.2.25:8080
http_port lan1.example.com:3128
http_port lan2.example.com:8081
```

12.3.2.2. ACL および HTTP のアクセス制御

アクセス制御リスト (ACL) は、アクセス制御の基本要素で、通常 **http_access** などの他のディレクティブと組み合わせて使用して、さまざまな **Squid** のコンポーネントと Web リソースへのアクセスを制御します。

例12.4 ドメイン名向けの ACL 構築

以下の例は、下記の一般的な指示を編集する方法を示しています。

```
acl example_site dstdomain example.com
```

example_site は任意の名前に置き換えて ACL に名前を指定してください。こおで使用する種別は **dstdomain** で、この値 (Web サイト) はドメイン名であると指定します。

```
acl FB dstdomain facebook.com
```

複数の Web サイトを対象とする ACL を構築する必要がある場合には、以下を行ってください。

- 単一の行に値を記述してください。

```
acl example_sites dstdomain example.com example.net example.org
```

- 値が大幅に増加する場合に向け、複数の行に分けて値を記述します。

```
acl example_sites dstdomain example.com example.net
acl example_sites dstdomain example.org
```

- 専用のファイルに値を投入して、**Squid** がそのファイルから値を読みこむように指示します。

```
acl example_sites dstdomain '/etc/squid/example_sites.txt'
```

`/etc/squid/example_sites.txt` の内容は以下のようになります。

```
# Write one value (domain name) per line
example.net
example.org # Temporarily remove example.org from example_sites acl
example.com
```



重要

さまざまなリソースへのアクセスを許可または拒否するには、ACL はアクセス制御のディレクティブと組み合わせる必要があります。**http_access** は、**Squid** 経由で HTTP トランザクションを実行するためのアクセスを許可する際に使用するディレクティブです。

ACL を使用した HTTP アクセスの制御

クライアントへのアクセスを許可または拒否するには、**http_access** ディレクティブと ACL を組み合わせる必要があります。

`/etc/squid/squid.conf` ファイルで **http_access** ディレクティブを編集します。`ACL_NAME` は、アクセスを許可または呼び出す必要のある要求にお聞かてください。

```
http_access allow|deny [!]ACL_NAME
```

例12.5 クライアントへのアクセスの許可または拒否

以下の設定は、localhost へのアクセスを許可します。

```
http_access allow localhost
```

この設定は、localhost へのアクセスを拒否します。

```
http_access deny localhost
```

感嘆符で始まる ACL 名もありますが、そのような場合には感嘆符も含みます。

```
http_access deny !Safe_ports
```

12.4. SQUID の認証

認証については、**Squid** のソースコードは SMB (Windows NT や Samba などの SMB サーバー)、DB (SQL データベース)、LDAP (Lightweight Directory Access Protocol) などの **helpers** と呼ばれる認証バックエンドと連携します。**Squid** が **proxy_auth** ACL を使用するように設定されている場合に、ユーザーは認証されます。

/etc/squid/squid.conf の **auth_param** ディレクティブで、どの認証ヘルパープログラムを使用するのかを **Squid** に指示を出します。プログラムと必要に応じてコマンドラインのオプションを指定します。

```
auth_param scheme parameter [setting]
```

例12.6 proxy_auth ACL の追加

個別のユーザー名を指定して、**proxy_auth** ACL エントリーを **Squid** の設定に追加します。この例では、lisa、sarah、joe、frank という名前のユーザーが常にプロキシーを使用できるようにしています。その他のユーザーは、日中の時間のみ許可されます。

```
acl foo proxy_auth REQUIRED
acl bar proxy_auth lisa sarah frank joe
acl daytime time 08:00-17:00
http_access allow foo daytime
http_access allow bar
http_access deny all
```

12.4.1. LDAP での認証

この設定では、**Squid** は LDAP を使用してユーザーを認証してから、ユーザーのインターネットアクセスを許可します。**Squid** のソースコードは、認証のために認証のバックエンド (LDAP) に接続します。ユーザーは、ユーザー名とパスワードを入力してからでないと、Web ページに進めません。**Squid** は、**Squid** LDAP 認証ヘルパー (**squid_ldap_auth**) を使用して、**Squid** が LDAP ディレクトリーに接続して、基本的な HTTP 認証のユーザー名とパスワードを検証できるようにします。

以下のように **/etc/squid/squid.conf** を編集して **Squid** を ldap.example.com に接続します。

```
auth_param basic program /usr/lib64/squid/basic_ldap_auth -b
"dc=example,dc=com" -f "uid=%s" -c 2 -t 2 -h ldap.example.com
otherldap.example.com
```

SSL/TLS のセキュアなチャンネルを使用して、LDAP サーバー上で **Squid** ユーザーを認証する場合は、**squid_ldap_auth** プログラムに **-ZZ** の引数を渡します。

```
auth_param basic program /usr/lib64/squid/basic_ldap_auth -v 3 -ZZ -b
"dc=yourcompany,dc=com" -D uid=some-user,ou=People,dc=yourcompany,dc=com
-w password -f uid=%s ldap.yourcompany.com
```

TLS と SSL など、複数の OpenLDAP サーバーに対して認証を行う場合には **/etc/squid/squid.conf** ファイルで **auth_param** を指定する必要があります。

1. TLS 用に `/etc/squid/squid.conf` を編集します。

```
auth_param basic program /usr/lib64/squid/basic_ldap_auth -Z -b
"dc=example,dc=com" -f "uid=%s" -c 2 -t 2 -h ldap.example.com
```

SSL の場合は以下のように変更します。

```
auth_param basic program /usr/lib64/squid/basic_ldap_auth -b
"dc=example,dc=com" -f "uid=%s" -c 2 -t 2 -H
ldaps://ldap.example.com
```

テストの実施場所

```
-b - Specifies the base DN under which the users are located.
-f - Specifies LDAP search filter to locate the user DN.
-c - Specifies timeout used when connecting to LDAP servers.
-t - Specifies time limit on LDAP search operations.
-h - Specifies the LDAP server to connect to.
-H - Specifies the LDAP server to connect to by LDAP URI
```

2. **Squid** サービスを再起動します。

```
~]# systemctl restart squid
```

12.4.2. Kerberos での認証

以下の手順に従い、Red Hat Enterprise Linux 7 の **Squid** プロキシが **Kerberos** 認証を使用するように設定します。また、前提条件として、まず Red Hat Enterprise Linux 用の Samba、Common Internet File System (CIFS) ファイルサーバーをインストールします。Smba のインストールに関する詳しい情報は、『Red Hat Enterprise Linux 7 システム管理者ガイド』の「[Samba](#)」のセクションを参照してください。

手順12.4 Red Hat Enterprise Linux 7 上の Squid が Kerberos 認証を使用するように設定する

1. **Squid** が Active Directory (AD) ドメインに参加するように設定します。

1. `/etc/krb5.conf` ファイルを編集します。

```
[libdefaults]
    default_realm = EXAMPLE.COM
    dns_lookup_kdc = no
    dns_lookup_realm = no
    default_keytab_name = /etc/krb5.keytab
; for Windows 2003
    default_tgs_enctypes = rc4-hmac des-cbc-crc des-cbc-md5
    default_tkt_enctypes = rc4-hmac des-cbc-crc des-cbc-md5
    permitted_enctypes = rc4-hmac des-cbc-crc des-cbc-md5

; for Windows 2008 with AES
;     default_tgs_enctypes = aes256-cts-hmac-sha1-96 rc4-
hmac des-cbc-crc des-cbc-md5
;     default_tkt_enctypes = aes256-cts-hmac-sha1-96 rc4-
hmac des-cbc-crc des-cbc-md5
```

```

;      permitted_encypes = aes256-cts-hmac-sha1-96 rc4-hmac
des-cbc-crc des-cbc-md5

[realms]
EXAMPLE.COM = {
    kdc = 192.168.0.1
    admin_server = 192.168.0.1
}

[domain_realm]
example.com = EXAMPLE.COM
.example.com = EXAMPLE.COM

[logging]
kdc = FILE:/var/log/kdc.log
admin_server = FILE:/var/log/kadmin.log
default = FILE:/var/log/krb5lib.log

```

2. **kinit** コマンドで検証します。

```
~]# kinit testuser1
```

```
~]# kinit administrator
```

3. 以下のように **/etc/samba/smb.conf** ファイルを編集します。

```

[global]
workgroup = EXAMPLE
password server = 192.168.0.1
# Remember to put the realm all in CAPS:
realm = EXAMPLE.COM
security = ads
idmap uid = 16777216-33554431
idmap gid = 16777216-33554431
template shell = /bin/bash
winbind use default domain = true
winbind offline logon = false
winbind enum users = yes
winbind enum groups = yes
encrypt passwords = yes
log file = /var/log/samba/log.%m
max log size = 50
passdb backend = tdbsam
load printers = yes
cups options = raw
kerberos method = system keytab

```

4. AD ドメインに参加します。

```
~]# net ads join -U Administrator
```

2. **net ads keytab** コマンドで、HTTP/fqdn の keytab を作成します。

```
~]# kinit administrator
~]# export KRB5_KTNAME=FILE:/etc/squid/HTTP.keytab
~]# net ads keytab CREATE
~]# net ads keytab ADD HTTP
```

keytab ファイルを検証します。

```
~]# klist -k /etc/squid/HTTP.keytab
```



注記

ホスト名が **/etc/hosts** ファイルで正しく設定されていることを確認します。

3. このファイルが **Squid** に含まれていることを確認してください。

```
~]# rpm -q squid
squid-3.1.10-1.el6.x86_64
```

```
~]# rpm -ql squid | grep kerb
```

```
/usr/lib64/squid/negotiate_kerberos_auth
/usr/lib64/squid/negotiate_kerberos_auth_test
/usr/lib64/squid/squid_kerb_auth
/usr/lib64/squid/squid_kerb_auth_test
```

4. 以下のように **/etc/squid/squid.conf** を変更してください。

```
auth_param negotiate program /usr/lib64/squid/squid_kerb_auth -d -s
HTTP/squid.example.com@EXAMPLE.COM
auth_param negotiate children 10
auth_param negotiate keep_alive on
acl kerb_auth proxy_auth REQUIRED
(content truncated)

http_access allow kerb_auth
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localnet
http_access allow localhost
http_access deny all
(content truncated)
```

5. **Squid** プロセスオーナーで読み込み可能な **.keytab** ファイルを設定します。

```
~]# chgrp squid /etc/squid/HTTP.keytab
```

```
~]# chmod g+r /etc/squid/HTTP.keytab
```

6. 以下の行を **/etc/sysconfig/squid** ファイルに追加します。


```
KRB5_KTNAME="/etc/squid/HTTP.keytab "
export KRB5_KTNAME
```

7. **Squid** サービスを起動します。

```
~]# service squid start
```

8. Kerberos クライアントを設定して、Web ブラウザーが **Squid** プロキシを使用するように設定します。Key Distribution Center (KDC) から Kerberos チケットを取得します。

```
~]# kinit testuser1
```

任意の Web サイトにアクセスしてみてください。Web ブラウザーで、ユーザー名またはパスワードが求められないはずです。

12.5. アクセス制限のための **SQUID** の使用

多くの場合、**Squid** は特定の Web コンテンツへのアクセスをブロックするために使用します。通常、特定のポートまたは特定の Web サイトがブロックされます。

12.5.1. ポートのブロックによるアクセス制限

ポートフィルタリングと呼ばれるこの手法では、**Squid** プロキシサーバーで固有のポート番号をブロックすることができます。これにより、プロトコル、サービス、Web サイト、アプリケーションの使用を制限することができます。たとえば、FTP トラフィックをブロックするには、ポート 21/TCP をブロックするだけで十分です。同様に、ポート 443/TCP をブロックすることで HTTPS サイトすべてをブロックすることができます。

手順12.5 ポート番号をブロックする

1. root ユーザーとしてログインして、**Squid** 設定ファイルを開きます。

```
~]# vi /etc/squid/squid.conf
```

2. ACL を使用してポートをブロックします。

```
acl Bad_ports port 443                #(create acl for port 443/tcp)
```

3. 変更を保存します。
4. **Squid** を再起動して、新規設定を適用します。

```
~]# service squid reload
```

Squid 設定ファイルには、**acl Safe_ports port** の行が含まれます。デフォルトでは、これらのポート番号は「Safe_Ports」として追加され、参照用に開放されます。

```
acl Safe_ports port 80
acl Safe_ports port 21
acl Safe_ports port 443
acl Safe_ports port 70
```

```

acl Safe_ports port 210
acl Safe_ports port 1025-65535
acl Safe_ports port 280
acl Safe_ports port 488
acl Safe_ports port 591
acl Safe_ports port 777

```

`/etc/squid/squid.conf` の各行を無効にして、適切なポートをブロックすることができます

例12.7 ポート 777/tcp のブロック

ポート 777/tcp をブロックするには、以下のように該当の行の前にハッシュ記号を追加します。

```
#acl Safe_ports port 777          # multiling http
```

12.5.2. 特定のサイトまたはアドレスをブロックしてアクセスを制御する手順

お使いのネットワークの **Squid** を設定して、固有のサイトへのアクセスを無効にします。

手順12.6 特定の Web サイトをブロックする

1. ネットワーク上で **Squid** へのアクセスを有効化します。`/etc/squid/squid.conf` ファイルを開き、「Access Controls」を検索します。**INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS** まで下方向にスクロールします。ブラウザが許可されているはずの場所から、内部 IP ネットワークに一覧を適用するようにしてください。以下の例は、ACL でローカルネットワーク 192.168.1.0/24 と 192.168.2.0/24 からのアクセスを許可します。

```

# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
acl our_networks src 192.168.1.0/24 192.168.2.0/24
http_access allow our_networks

```

2. ブロックするサイト一覧が含まれるファイルを作成します。`/usr/local/etc/allowed-sites.squid` や `/usr/local/etc/restricted-sites.squid` など、ファイルに名前を指定します。

```

~]# cat /usr/local/etc/allowed-sites.squid
www.redhat.com
fedoraproject.org

```

```

~]# cat /usr/local/etc/restricted-sites.squid
www.badsites.com
illegal.com

```

これらは、制限をしたサイトをブロックするために使用することができます。

```
~]# vi /etc/squid/squid.conf
```

```

acl our_networks src 192.168.1.0/24 192.168.2.0/24
acl GoodSites dstdomain "/usr/local/etc/allowed-sites.squid"
acl BadSites dstdomain "/usr/local/etc/restricted-sites.squid"

```

```
http_access allow our_networks
http_access deny BadSites
http_access allow home_network business_hours GoodSites
```

ファイルを保存してから閉じます。

3. **Squid** プロキシサーバーを再起動します。

```
~]# systemctl restart squid
```

4. Web ブラウザーが **Squid** サーバーの DNS 名または IP アドレスを使用して、実行中のポートと一致するように設定します。

12.6. その他のリソース

- **squid(8)**
- **squidclient(1)**
- **basic_ldap_auth(8)**
- **ext_ldap_group_acl(8)**
- **ext_session_acl(8)**
- **ext_unix_group_acl(8)**
- **negotiate_kerberos_auth(8)**

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。

ブリッジの構成

[Bridge Configuration](#) は、Red Hat Enterprise Linux 5.4 およびそれ以降を使用する 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ースを設定するために作成されました。

NETWORK BONDING HELPER

[Network Bonding Helper](#) を用いることで、管理者は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ースを使用して、複数のネットワークインターフェースコントローラーを単一のチャンネルにまとめることができます。

Network Bonding Helper を使用することにより、複数のネットワークインターフェースを 1 つのボンディングインターフェースとして機能させることができます。

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 を使用すると、インターフェースを選択して情報をコンソールに出力する `tcpdump` コマンドを生成することができます。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 0.10-01.1 翻訳ファイルを XML ソースバージョン 0.10-01 と同期	Tue Feb 20 2018	Terry Chuang
改訂 0.10-01 7.4 GA 公開用バージョン	Tue 25 Jul 2017	Mirek Jahoda
改訂 0.9-30 7.3 GA リリースのバージョン	Tue 18 Oct 2016	Mirek Jahoda
改訂 0.9-25 7.2 GA リリース向けのバージョン	Wed 11 Nov 2015	Jana Heves
改訂 0.9-15 7.1 GA リリース向けのバージョン	Tue 17 Feb 2015	Christian Huffman
改訂 0.9-14 nmtui および NetworkManager GUI セクションを更新。	Fri Dec 05 2014	Christian Huffman
改訂 0.9-12 『IP ネットワーク』、『802.1Q VLAN tagging』、および『チーミング』を更新。	Wed Nov 05 2014	Stephen Wadeley
改訂 0.9-11 『ボンディング』、『ブリッジング』、および『チーミング』を更新。	Tues Oct 21 2014	Stephen Wadeley
改訂 0.9-9 『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。	Tue Sep 2 2014	Stephen Wadeley
改訂 0.9-8 Red Hat Enterprise Linux 7.0 GA リリースのネットワークガイド	Tue July 8 2014	Stephen Wadeley
改訂 0-0 Red Hat Enterprise Linux 7 ネットワークガイドの作成。	Wed Dec 12 2012	Stephen Wadeley

索引

シンボル

`/etc/named.conf` (参照 [BIND](#))

カーネルモジュール

`bonding` モジュール, [チャンネルボンディングの使用](#)

ボンディングされたインターフェースのパラメーター, [ボンディングモジュールのディレクティブ](#)

詳細, [チャンネルボンディングの使用](#)

モジュールパラメーター

`bonding` モジュールのパラメーター, [ボンディングモジュールのディレクティブ](#)

チャンネルボンディング

ボンディングされたインターフェースのパラメーター, [ボンディングモジュールのディレクティブ](#)
設定, [チャンネルボンディングの使用](#)

詳細, [チャンネルボンディングの使用](#)

チャンネルボンディングインターフェース (参照 [カーネルモジュール](#))

デフォルトゲートウェイ, [静的ルートおよびデフォルトゲートウェイ](#)

ボンディング (参照 [チャンネルボンディング](#))

マルチホーム DHCP

サーバーの設定, [マルチホーム DHCP サーバーの設定](#)

ホストの設定, [ホストの設定](#)

リソースレコード (参照 [BIND](#))

動的ホスト構成プロトコル (参照 [DHCP](#))

静的ルート, [静的ルートおよびデフォルトゲートウェイ](#)

A

`authoritative nameserver` (参照 [BIND](#))

B

Berkeley Internet Name Domain (参照 [BIND](#))

BIND

types

`authoritative nameserver`, [ネームサーバーのタイプ](#)

`primary (master) nameserver`, [ネームサーバーのタイプ](#)

`recursive nameserver`, [ネームサーバーのタイプ](#)

`secondary (slave) nameserver`, [ネームサーバーのタイプ](#)

その他のリソース, [オンラインリソース](#)

インストールされているドキュメント, [インストールされているドキュメント](#)

ゾーン

`$INCLUDE` ディレクティブ, [一般的なディレクティブ](#)

`$ORIGIN` ディレクティブ, [一般的なディレクティブ](#)

`$TTL` ディレクティブ, [一般的なディレクティブ](#)

`A (Address)` リソースレコード, [一般的なリソースレコード](#)

`CNAME (Canonical Name)` リソースレコード, [一般的なリソースレコード](#)

`MX (Mail Exchange)` リソースレコード, [一般的なリソースレコード](#)

`NS (Nameserver)` リソースレコード, [一般的なリソースレコード](#)

`PTR (Pointer)` リソースレコード, [一般的なリソースレコード](#)

`SOA (Start of Authority)` リソースレコード, [一般的なリソースレコード](#)

コメントタグ, [コメントタグ](#)

使用法の例, [単純なゾーンファイル](#), [逆引き名前解決ゾーンファイル](#)

詳細, [ネームサーバーのゾーン](#)

タイプ

セカンダリー (スレーブ) ネームサーバー, [ネームサーバーのゾーン](#)

プライマリー (マスター) ネームサーバー, [ネームサーバーのゾーン](#)

ディレクトリー

`/etc/named/`, [named サービスの設定](#)

`/var/named/`, [ゾーンファイルの編集](#)

`/var/named/data/`, [ゾーンファイルの編集](#)

`/var/named/dynamic/`, [ゾーンファイルの編集](#)

`/var/named/slaves/`, [ゾーンファイルの編集](#)

ファイル

`/etc/named.conf`, [named サービスの設定](#), [ユーティリティーの設定](#)

`/etc/rndc.conf`, [ユーティリティーの設定](#)

`/etc/rndc.key`, [ユーティリティーの設定](#)

ユーティリティ

`dig`, [ネームサーバーとしての BIND](#)

`named`, [ネームサーバーとしての BIND](#), [named サービスの設定](#)

`rndc`, [ネームサーバーとしての BIND](#)

ユーティリティー

`dig`, [dig ユーティリティーの使用](#), [DNSSEC \(DNS Security Extensions\)](#)

`rndc`, [rndc ユーティリティーの使用](#)

リソースレコード, ネームサーバーのゾーン

一般的な間違い, [回避すべき一般的な間違い](#)

機能

[Automatic Zone Transfer \(AXFR\)](#), [IXFR \(Incremental Zone Transfers 差分ゾーン転送\)](#)

[DNS Security Extensions \(DNSSEC\)](#), [DNSSEC \(DNS Security Extensions\)](#)

Incremental Zone Transfer (IXFR), [IXFR \(Incremental Zone Transfers 差分ゾーン転送\)](#)
Internet Protocol version 6 (IPv6), [インターネットプロトコルバージョン 6 \(IPv6\)](#)
Transaction SIGNature (TSIG), [Transaction SIGNatures トランザクション署名 \(TSIG\)](#)
[複数表示](#), [複数表示](#)

設定

[acl](#) ステートメント, [一般的なステートメントのタイプ](#)
[controls](#) ステートメント, [その他のステートメントタイプ](#)
[include](#) ステートメント, [一般的なステートメントのタイプ](#)
[key](#) ステートメント, [その他のステートメントタイプ](#)
[logging](#) ステートメント, [その他のステートメントタイプ](#)
[options](#) ステートメント, [一般的なステートメントのタイプ](#)
[server](#) ステートメント, [その他のステートメントタイプ](#)
[trusted-keys](#) ステートメント, [その他のステートメントタイプ](#)
[view](#) ステートメント, [その他のステートメントタイプ](#)
[zone](#) ステートメント, [一般的なステートメントのタイプ](#)
[コメントタグ](#), [コメントタグ](#)

D

DHCP, [DHCP サーバー](#)

[dhcpd.conf](#), [設定ファイル](#)
[dhcpd.leases](#), [サーバーの起動と停止](#)
[dhcpd6.conf](#), [IPv6 の DHCP \(DHCPv6\)](#)
[DHCPv6](#), [IPv6 の DHCP \(DHCPv6\)](#)
[dhcrelay](#), [DHCP リレーエージェント](#)
[shared-network](#), [設定ファイル](#)
[subnet](#), [設定ファイル](#)
[その他のリソース](#), [その他のリソース](#)
[オプション](#), [設定ファイル](#)
[グループ](#), [設定ファイル](#)
[グローバルパラメーター](#), [設定ファイル](#)
[コマンドラインオプション](#), [サーバーの起動と停止](#)
[サーバーの停止](#), [サーバーの起動と停止](#)
[サーバーの設定](#), [DHCP サーバーの設定](#)
[サーバーの起動](#), [サーバーの起動と停止](#)
[リレーエージェント](#), [DHCP リレーエージェント](#)
[使用する理由](#), [DHCP を使用する理由](#)

[dhcpd.conf](#), [設定ファイル](#)
[dhcpd.leases](#), [サーバーの起動と停止](#)
[dhcrelay](#), [DHCP リレーエージェント](#)

dig (参照 BIND)

DNS

定義, [DNS サーバー](#)

(参照 BIND)

N

named (参照 BIND)

nameserver (参照 DNS)

NIC

単一のチャンネルにバインド, [チャンネルボンディングの使用](#)

P

primary nameserver (参照 BIND)

R

recursive nameserver (参照 BIND)

rndc (参照 BIND)

root ネームサーバー (参照 BIND)

S

secondary nameserver (参照 BIND)

Squid, [Squid](#)