Red Hat Enterprise Linux 7

システム管理者のガイド

RHEL 7 の導入、設定、および管理

Last Updated: 2024-03-03

Red Hat Enterprise Linux 7 システム管理者のガイド

RHEL 7 の導入、設定、および管理

法律上の通知

概要

システム管理者のガイド では、Red Hat Enterprise Linux 7 の導入、設定、管理の関連情報を説明します。本書は、システムに関する基本的な理解があるシステム管理者を対象としています。 専門知識を深めるために、トレーニングコース Red Hat System Administration I (RH124)、Red Hat System Administration II (RH134)、Red Hat System Administration III (RH254)、または RHCSA Rapid Track (RH199) を受講することを推奨します。 Linux コンテナーの機能とともに Red Hat Enterprise Linux 7 を使用する場合は、Red Hat Enterprise Linux Atomic Host の製品ドキュメントを参照してください。Linux コンテナーの一般概念、および Red Hat Enterprise Linux 7 に実装され

ているコンテナーの最新機能の概要は、Overview of Containers in Red Hat Systems をご覧ください。コンテナーの管理に関連するトピックは Red Hat Enterprise Linux Atomic Host 7 Managing Containersガイドで詳しく説明しています。

目次

パート I. システムの基本設定

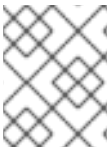
ここでは、キーボードの設定、日付と時刻の設定、ユーザーとグループの管理、権限の取得など、インストール後の基本的なタスクおよび基本的なシステム管理タスクを取り上げます。

第1章 はじめに

本章では、Red Hat Enterprise Linux 7 のインストール直後に実行する必要のある基本的なタスクを説明します。

これらのアイテムには、通常はインストールプロセス中に実行済みとなるタスクが含まれている可能性がありますが、システムの登録など、実行する必要がないものもあることに注意してください。本章の各セクションではこのようなタスクを扱い、インストール中に必要な方法と、別セクションにある関連ドキュメントへのリンクを紹介します。

Red Hat Enterprise Linux 7 のインストールに関する詳細は Red Hat Enterprise Linux 7 インストールガイド を参照してください。



注記

本章では、実行すべきコマンドをいくつか紹介します。**root** 権限が必要なコマンドには **#** がついており、一般ユーザーが実行できるコマンドには **$** がついています。

インストール後の一般的なタスクの詳細は、インストールガイドの Red Hat Enterprise Linux 7 インストール後 を参照してください。

インストール後のタスクはすべてコマンドラインから実行できますが、一部のコマンドは **Web コンソール** から実行することもできます。

Web コンソールの概要および使用できるタスク

Web コンソール はシステム管理ツールで、Web ブラウザーを通して監視サーバーおよび管理サーバーのユーザーインターフェイスを提供します。

Web コンソール は、以下のタスクを実行できます。

- ハードウェア、インターネット接続、パフォーマンスの特徴など、基本的なシステム機能の監視
- システムログファイルのコンテンツの分析
- インターフェイス、ネットワークログ、パケットサイズなど、基本的なネットワーキング機能の設定
- ユーザーアカウントの管理
- システムサービスの監視および設定
- 診断レポートの作成
- カーネルダンプ設定の設定
- SELinux の設定
- システムサブスクリプションの管理
- ターミナルへのアクセス

Web コンソール のインストールおよび使用の詳細は RHEL 7 の Web コンソールを使用したシステムの管理 を参照してください。

1.1. 環境の基本設定

環境の基本設定には以下が含まれます。

- 日付と時刻
- システムロケール
- キーボードのレイアウト

通常、これらのアイテムの設定は、インストールプロセスに含まれます。

詳細は、インストール方法に応じた適切な資料を参照してください。

- Anaconda インストーラーを使用してインストールする場合は、以下を参照してください。
  Red Hat Enterprise Linux 7 インストールガイドの 日付と時刻、言語サポート、および キーボードの設定。
- キックスタートファイルを使用してインストールする場合は、以下を参照してください。
  Red Hat Enterprise Linux 7 インストールガイドの キックスタートのコマンドとオプション

インストール後に、環境の基本的な特徴を再設定する必要がある場合は、このセクションの指示に従います。

1.1.1. 日付と時刻の設定について

正確な時間を維持することは、様々な理由で重要です。Red Hat Enterprise Linux 7 では、**NTP** プロトコルにより、時間の正確さが確保されます。ユーザー領域のデーモンは、カーネルで実行しているシステムクロックを更新します。システムクロックは、さまざまなクロックソースを使用して時間を維持します。

Red Hat Enterprise Linux 7 は以下のデーモンを使用して、**NTP** を実装します。

- **chronyd**
  デフォルトでは、**chronyd** デーモンを使用します。これは **chrony** パッケージから利用できます。**chronyd** で **NTP** を設定および使用する方法は、18章*chrony スイートを使用した NTP 設定* を参照してください。
- **ntpd**
  ntpd デーモンは、**ntp** パッケージから利用できます。**ntpd** を使用した **NTP** の設定および使用に関する詳細は 19章*ntpd を使用した NTP 設定* を参照してください。

デフォルトの **chronyd** ではなく **ntpd** を使用する場合は、**chronyd** を無効にし、19章*ntpd を使用した NTP 設定* に従って **ntpd** をインストールして有効にし、設定する必要があります。

システムの現在日時の表示
システムの現在日時を表示するには、以下のいずれかのコマンドを使用します。

```
~]$ date
```

```
~]$ timedatectl
```

timedatectl コマンドを使用すると、より詳細な出力が得られます。出力には、ユニバーサル時間、現在使用しているタイムゾーン、Network Time Protocol (NTP) 設定ステータスなどの情報が含まれます。

日付と時刻の設定に関する詳細は、*3章日付と時刻の設定* を参照してください。

1.1.2. システムロケールの設定について

システム全体にわたるロケール設定は **/etc/locale.conf** ファイルに保存され、システム起動の初期段階で **systemd** デーモンにより読み込まれます。**/etc/locale.conf** に設定したロケール設定は、個別のプログラムやユーザーが上書きしない限り、すべてのサービスやユーザーに継承されます。

システムロケールを処理する基本的なタスク

- 利用可能なシステムロケール設定のリスト表示

```
~]$ localectl list-locales
```

- システムロケール設定の現行ステータスの表示

```
~]$ localectl status
```

- デフォルトのシステムロケール設定または変更

```
~]# localectl set-locale LANG=locale
```

システムロケールの設定に関する詳細は、*2章システムロケールおよびキーボード設定* を参照してください。

1.1.3. キーボードレイアウトの設定

キーボードレイアウト設定では、テキストコンソールとグラフィカルユーザーインターフェイスで使用するレイアウトを管理します。

キーボードレイアウトを処理する基本的なタスクには、以下が含まれます。

- 利用可能なキーマップのリスト表示

```
~]$ localectl list-keymaps
```

- キーマップ設定の現行ステータスの表示

```
~]$ localectl status
```

- デフォルトのシステムキーマップの設定または変更

```
~]# localectl set-keymap
```

キーボードレイアウトの設定に関する詳細は、*2章システムロケールおよびキーボード設定* を参照してください。

1.2. ネットワークアクセスの設定および検査

通常、ネットワークアクセスはインストールプロセス中に設定されます。しかし、インストールプロセスでは、一部の共通インストールパスでネットワークインターフェイスの設定を求めるプロンプトは表示されません。その結果、インストール後にネットワークアクセスが設定されていない可能性があります。その場合は、インストール後にネットワークアクセスを設定します。

インストール中のネットワークアクセスの設定に関するクイックスタートは 「インストールプロセス時のネットワークアクセスの設定」 を参照してください。インストール後にネットワークアクセスを設定するには、Red Hat Enterprise Linux 7 ネットワークガイド で説明されている **nmcli** コマンドラインユーティリティーか、Red Hat Enterprise Linux 7 ネットワークガイド で説明されている テキスト形式のユーザーインターフェイスユーティリティー **nmtui** のいずれかを使用できます。

nmcli および **nmtui** ユーティリティーは、1 つ以上の新しいネットワーク接続を追加するだけでなく、既存接続の変更および調査も可能にします。**nmcli** を使用してネットワーク接続を作成し、管理する場合は「nmcli を使用したインストールプロセス後のネットワーク接続管理」を参照してください。**nmtui** を使用してネットワーク接続を作成し、管理する場合は 「nmtui を使用したインストールプロセス後のネットワーク接続管理」 を参照してください。

1.2.1. インストールプロセス時のネットワークアクセスの設定

インストールプロセス時のネットワークアクセスの設定方法

- Anaconda インストールプログラムにおけるグラフィカルユーザーインターフェイスのインストール概要画面に表示される **ネットワークとホスト名** メニュー
- Anaconda インストールプログラムのテキストモードの **ネットワーク設定** オプション
- キックスタートファイル

インストール完了後に初めてシステムを起動すると、インストール中に設定したネットワークインターフェイスが自動的にアクティブになります。

インストールプロセス中のネットワークアクセスの設定に関する詳細は Red Hat Enterprise Linux 7 インストールガイド を参照してください。

1.2.2. nmcli を使用したインストールプロセス後のネットワーク接続管理

nmcli ユーティリティーを使用してネットワーク接続を管理するには、**root** として以下のコマンドを実行します。

接続を新規作成するには、以下を実行します。

```
~]# nmcli con add type type of the connection "con-name" connection name ifname ifname interface-name the name of the interface ipv4 address ipv4 address gw4 address gateway address
```

既存の接続を修正するには、以下を実行します。

```
~]# nmcli con mod "con-name"
```

すべての接続を表示するには、以下を実行します。

```
~]# nmcli con show
```

アクティブな接続を表示するには、以下を実行します。

```
~]# nmcli con show --active
```

特定の接続の設定をすべて表示するには、以下を実行します。

```
~]# nmcli con show "con-name"
```

nmcli コマンドラインユーティリティーに関する詳細は Red Hat Enterprise Linux 7 ネットワークガイド の Red Hat Enterprise Linux 7 ネットワークガイド を参照してください。

1.2.3. nmtui を使用したインストールプロセス後のネットワーク接続管理

NetworkManager テキストユーザーインターフェイス (TUI) のユーティリティー (**nmtui**) は、**NetworkManager** を制御してネットワークを設定するテキストインターフェイスを提供します。

テキスト形式のインターフェイスツールである **nmtui** のインストールおよび使用に関する詳細は Red Hat Enterprise Linux 7 ネットワークガイド を参照してください。

1.2.4. Web コンソールでのネットワークの管理

Web コンソール の **Networking** メニューでは、以下が可能です。

- 最近送受信したパケットの表示
- 利用可能なネットワークインターフェイスの最も重要な特徴の表示
- ネットワーキングログのコンテンツの表示
- ネットワークインターフェイスの様々なタイプ (ボンディング、チーム、ブリッジ、VLAN) の追加

図1.1 Web コンソールでのネットワークの管理

1.3. システム登録およびサブスクリプション管理の基本

1.3.1. Red Hat サブスクリプションの概要、およびサブスクリプションを使用できるタスク

Red Hat Enterprise Linux 7 オペレーティングシステムと、そこにインストールされている製品は、サブスクリプションでカバーされています。

Red Hat コンテンツ配信ネットワーク (CDN) サブスクリプションを使用して、以下を追跡します。

- 登録したシステム
- 登録したシステムにインストールされている製品
- インストールされている製品に割り当てられているサブスクリプション

1.3.2. インストール時のシステム登録

本セクションでは、インストールプロセス中に行う Red Hat Enterprise Linux 7 の登録について簡単な概要を説明します。インストールしてもオペレーティングシステムが登録されていない場合は、本セクションを読むことで、インストール中に設定しなかった項目を確認できます。詳細は Red Hat Enterprise Linux 7 インストールガイド を参照してください。

基本的に、インストール中にシステムを登録する方法は 2 つあります。

- 通常、登録は、**初期設定の設定**プロセスで行います。詳細は Red Hat Enterprise Linux 7 インストールガイド を参照してください。
- または、**インストール後のスクリプトでサブスクリプションマネージャー** を実行して行います。この場合は、インストールの完了と同時 (システムが最初の再起動を実施する前) に、自動登録を実行します。これを行うには、キックスタートファイルの %post セクションを変更します。インストール後のスクリプトとしてサブスクリプションマネージャーを実行する場合は Red Hat Enterprise Linux 7 インストールガイド を参照してください。

1.3.3. インストール後のシステムの登録

インストールプロセス中にシステムの登録をしなかった場合は、以下の手順に従ってインストール後に登録できます。この手順で紹介するコマンドはすべて **root** で実行する必要があります。

システムの登録およびサブスクリプションの割り当て

1. システムを登録します。

   ```
   ~]# subscription-manager register
   ```

   コマンドを実行すると、Red Hat カスタマーポータルのユーザー名とパスワードの入力を求めるプロンプトが表示されます。

2. 必要なサブスクリプションのプール ID を確認します。

   ```
   ~]# subscription-manager list --available
   ```

   このコマンドは、使用している Red Hat アカウントで利用可能なサブスクリプションをすべて表示します。サブスクリプションごとに、プール ID を含むさまざまな情報が表示されます。

3. **pool_id** を、確認したプール ID に置き換えて、適切なサブスクリプションをシステムに割り当てます。

   ```
   ~]# subscription-manager attach --pool=pool_id
   ```

システムの登録および Red Hat コンテンツ配信ネットワークサブスクリプションの割り当て方法は、7 章 *システム登録およびサブスクリプション管理* を参照してください。

1.3.4. システムの EUS コンテンツへの登録

延長更新サポート (EUS) コンテンツにアクセスするには、以下のようにシステムを登録します。

1. EUS エンタイトルメントが利用可能であることを確認します。

```
~]# subscription-manager list --available --matches="*Extended Update Support"
```

```
   +-------------------------------------------+
      Available Subscriptions
   +-------------------------------------------+
   Subscription Name:   Extended Update Support
   Provides:            Red Hat Enterprise Linux High Availability for x86_64 - Extended Update
Support
                        Red Hat Enterprise Linux Resilient Storage for x86_64 - Extended Update
Support
                        Red Hat Enterprise Linux for x86_64 - Extended Update Support
                        Red Hat EUCJP Support (for RHEL Server) - Extended Update Support
                        RHEL for SAP - Extended Update Support
                        Red Hat Enterprise Linux Load Balancer (for RHEL Server) - Extended
Update Support
                        Red Hat Enterprise Linux Scalable File System (for RHEL Server) - Extended
Update Support
                        Red Hat CodeReady Linux Builder for x86_64 - Extended Update Support
                        RHEL for SAP HANA - Extended Update Support
                        Red Hat Enterprise Linux High Performance Networking (for RHEL Server) -
Extended Update Support
                        Oracle Java (for RHEL Server) - Extended Update Support
                        Red Hat S-JIS Support (for RHEL Server) - Extended Update Support
   SKU:                 RH00030
   Contract:            12069074
   Pool ID:             8a99f9ac7238188b01723d9c8a8a06a9
   Provides Management: No
   Available:           8
   Suggested:           0
   Service Level:       Layered
   Service Type:        L1-L3
   Subscription Type:   Instance Based
   Starts:              05/22/2020
   Ends:                05/21/2021
   System Type:         Physical
```

2. プール ID を使用して該当サブスクリプションを割り当てます。

```
~]# subscription-manager attach --pool 8a99f9ac7238188b01723d9c8a8a06a9
```

3. システムに有効なデフォルトのリポジトリーを EUS バリアントに置き換えます。

```
~]# subscription-manager repos --disable \*
```

4. 使用中の RHEL リビジョンの EUS コンテンツセットを表すリポジトリーを有効にします。

```
~]# subscription-manager repos --enable rhel-7-server-eus-rpms
```

5. エンドシステムで必要な、サポート対象のリリースを選択します。

```
~]# subscription-manager release --set 7.6
```

現在サポートされている EUS リリースについては、Extended Update Support Add-on を参照してください。

1.3.5. システムを E4S コンテンツに登録する

以下の手順では、システムを登録して E4S コンテンツを使用する方法を説明します。

1. 以下のコマンドを使用してシステムを登録します。

```
~]# subscription-manager register
```

2. E4S エンタイトルメントが利用可能であることを確認します。

```
~]# subscription-manager list --available --matches="*Update Services for SAP Solutions*"
```

```
+-------------------------------------------+
   Available Subscriptions
+-------------------------------------------+
Subscription Name:   Red Hat Enterprise Linux for SAP Solutions, Standard (Physical or
Virtual Nodes)
Provides:            dotNET on RHEL Beta (for RHEL Server)
                     Red Hat CodeReady Linux Builder for x86_64
                     Red Hat Enterprise Linux for SAP HANA for x86_64
                     Red Hat Ansible Engine
                     RHEL for SAP HANA - Update Services for SAP Solutions
                     Red Hat Enterprise Linux Scalable File System (for RHEL Server) - Extended
Update Support
                     RHEL for SAP HANA - Extended Update Support
                     Red Hat Enterprise Linux Atomic Host Beta
                     Red Hat Beta
                     Red Hat EUCJP Support (for RHEL Server) - Extended Update Support
                     Red Hat Enterprise Linux High Availability for x86_64
                     Red Hat Enterprise Linux Load Balancer (for RHEL Server) - Extended Update
Support
                     dotNET on RHEL (for RHEL Server)
                     Red Hat CodeReady Linux Builder for x86_64 - Extended Update Support
                     Red Hat Enterprise Linux High Availability - Update Services for SAP Solutions
                     Red Hat Enterprise Linux Resilient Storage for x86_64 - Extended Update
Support
                     Red Hat Enterprise Linux High Availability for x86_64 - Extended Update
Support
                     Oracle Java (for RHEL Server)
                     Red Hat Enterprise Linux Server - Update Services for SAP Solutions
                     Red Hat Software Collections (for RHEL Server)
                     Red Hat Enterprise Linux Scalable File System (for RHEL Server)
                     Red Hat Enterprise Linux High Performance Networking (for RHEL Server) -
Extended Update Support
                     RHEL for SAP - Update Services for SAP Solutions
```

```
                Oracle Java (for RHEL Server) - Extended Update Support
                Red Hat Enterprise Linux Atomic Host
                Red Hat Developer Tools (for RHEL Server)
                Red Hat Software Collections Beta (for RHEL Server)
                Red Hat Enterprise Linux Server
                Red Hat Enterprise Linux for SAP Applications for x86_64
                Red Hat Developer Tools Beta (for RHEL Server)
                Red Hat Enterprise Linux for x86_64
                Red Hat Enterprise Linux for x86_64 - Extended Update Support
                RHEL for SAP - Extended Update Support
                Red Hat Developer Toolset (for RHEL Server)
                Red Hat S-JIS Support (for RHEL Server) - Extended Update Support
SKU:            RH00764
Contract:       11977725
Pool ID:        8a85f99c6c4825eb016c4a30d3493064
Provides Management: Yes
Available:      18
Suggested:      0
Service Level:  Standard
Service Type:   L1-L3
Subscription Type:  Instance Based
Starts:         03/29/2020
Ends:           12/31/2021
System Type:    Physical
```

3. プール ID を使用して該当サブスクリプションを割り当てます。

```
~]# subscription-manager attach --pool=##################
```

4. システムに有効なデフォルトのリポジトリーを EUS バリアントに置き換えます。

```
~]# subscription-manager repos --disable="*"
```

5. 使用中の RHEL リビジョンの E4S コンテンツセットを表すリポジトリーを有効にします。

```
~]# subscription-manager --enable=rhel-7-server-e4s-rpms
```

6. リポジトリーキャッシュをクリアし、システムのロックを、有効なリリース (SAP アプリケーションをサポートする E4S) にリリースします。

```
~]# yum clean all && subscription-manager release --set=7.7
```

1.4. ソフトウェアのインストール

本セクションでは、Red Hat Enterprise Linux 7 システムにソフトウェアをインストールする際の基本的な内容を紹介します。「ソフトウェアインストールの前提条件」 では、ソフトウェアをインストールできるようにするために実行すべき前提条件を説明します。「ソフトウェアパッケージングとソフトウェアリポジトリーのシステム」 では、ソフトウェアパッケージングとソフトウェアリポジトリーに関する基本情報を説明します。また、「サブスクリプションマネージャーおよび yum を使用したソフトウェアインストールの基本タスクの管理」 では、ソフトウェアのインストールに関連する基本的なタスクの実行方法を説明します。

1.4.1. ソフトウェアインストールの前提条件

Red Hat コンテンツ配信ネットワークのサブスクリプションサービスは、Red Hat のソフトウェアインベントリーを処理するメカニズムを提供し、ソフトウェアを追加でインストールしたり、インストール済みのパッケージを更新したりできるようにします。「システム登録およびサブスクリプション管理の基本」 に従って、システムの登録とサブスクリプションの割り当てを完了したら、ソフトウェアのインストールを開始できます。

1.4.2. ソフトウェアパッケージングとソフトウェアリポジトリーのシステム

Red Hat Enterprise Linux システムにあるすべてのソフトウェアは、RPM パッケージに分類されます。RPM パッケージは、特定のリポジトリーに置かれています。Red Hat コンテンツ配信ネットワークにシステムをサブスクライブすると、**/etc/yum.repos.d/** ディレクトリーにリポジトリーファイルが作成されます。

パッケージ操作を管理するには、**yum** ユーティリティーを使用します。

- パッケージに関する情報の検索
- パッケージのインストール
- パッケージの更新
- パッケージの削除
- 現在利用可能なリポジトリーのリストの確認
- リポジトリーの追加または削除
- リポジトリーの有効化または無効化

ソフトウェアのインストールに関連する基本的なタスクの詳細は 「サブスクリプションマネージャーおよび yum を使用したソフトウェアインストールの基本タスクの管理」 を参照してください。ソフトウェアリポジトリーの管理に関する詳細は 「ソフトウェアリポジトリーの管理」 を参照してください。**yum** ユーティリティーの使用に関する詳細は、9章*Yum* を参照してください。

1.4.3. サブスクリプションマネージャーおよび yum を使用したソフトウェアインストールの基本タスクの管理

以下は、オペレーティングシステムのインストール後に必要になる可能性がある最も基本的なソフトウェアインストールタスクです。

- 利用可能なリポジトリーをすべて表示します。

```
~]# subscription-manager repos --list
```

- 現在有効になっているリポジトリーをすべて表示します。

```
~]$ yum repolist
```

- リポジトリーを有効または無効にします。

```
~]# subscription-manager repos --enable repository
```

```
~]# subscription-manager repos --disable repository
```

- 特定の文字列に一致するパッケージを検索します。

```
~]$ yum search string
```

- パッケージをインストールします。

```
~]# yum install package_name
```

- パッケージおよびその依存関係をすべて更新します。

```
~]# yum update
```

- パッケージを更新します。

```
~]# yum update package_name
```

- パッケージおよびそれに依存しているパッケージをすべてアンインストールします。

```
~]# yum remove package_name
```

- インストール済みで利用可能なパッケージの情報をすべて表示します。

```
~]$ yum list all
```

- インストール済みパッケージの情報をすべて表示します。

```
~]$ yum list installed
```

1.5. 起動時の SYSTEMD サービスの開始

systemd は、Linux オペレーティングシステム用のシステムおよびサービスのマネージャーで、systemd ユニットの概念が使用されています。systemd に関する詳細は 「systemd の概要」 を参照してください。

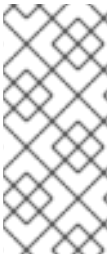
本セクションでは、システムの起動時にサービスを有効または無効にする方法を説明します。また、**Web コンソール** を使用してサービスを管理する方法も説明します。

1.5.1. サービスの有効化/無効化

インストールプロセス時に、システムの起動時に有効または無効にするサービスを設定できます。インストール済みのオペレーティングシステムでサービスを有効または無効にすることもできます。

インストールプロセスで、システムの起動時に有効または無効にするサービスのリストを作成する場合は、キックスタートファイルの **services** オプションを使用します。

```
services [--disabled=list] [--enabled=list]
```



注記

無効にするサービスのリストは、有効にするサービスのリストの前に処理されます。したがって、同じサービスが両方のリストに記載されていると、そのサービスは有効になります。サービスのリストはコンマ区切りのフォーマットで指定する必要があります。サービスのリストには空白文字を使用しないでください。詳細は Red Hat Enterprise Linux 7 インストールガイド を参照してください。

インストール後に、オペレーティングシステムのサービスを有効または無効にするには、以下を実行します。

```
~]# systemctl enableservice_name
```

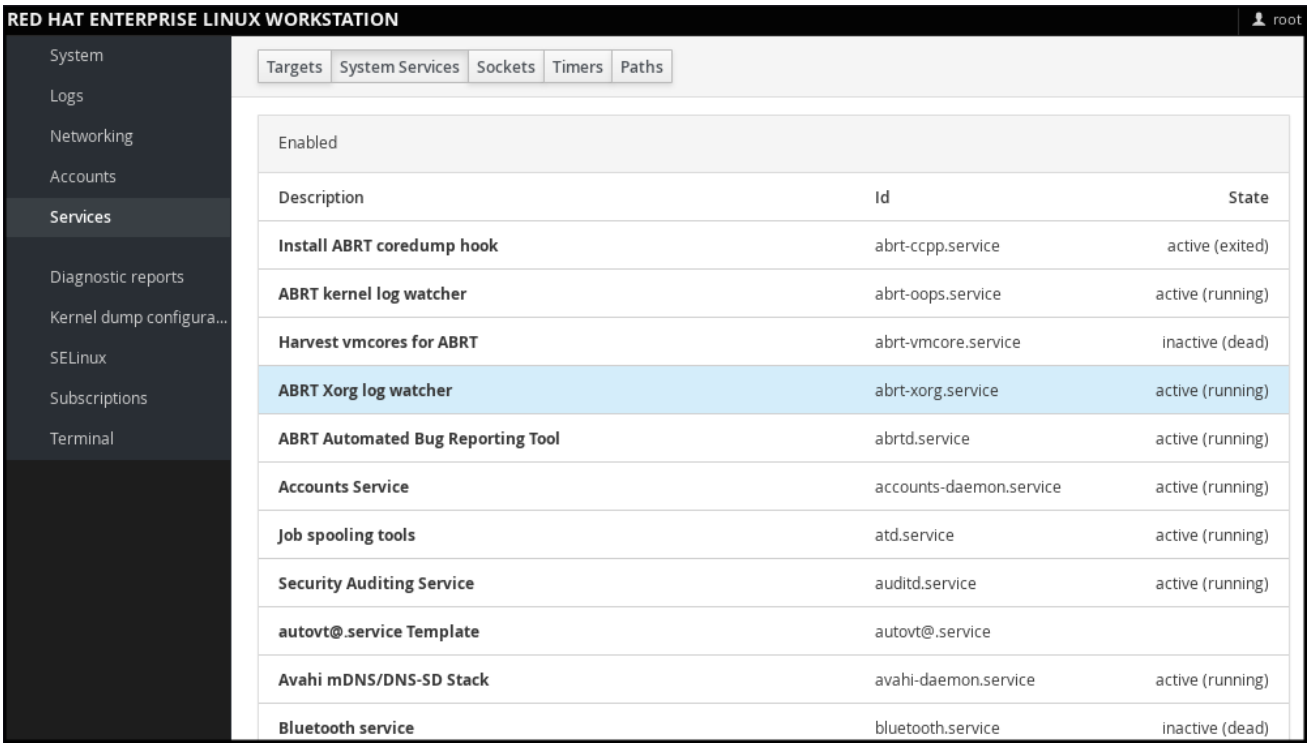
```
~]# systemctl disableservice_name
```

詳細は、「システムサービスの管理」 を参照してください。

1.5.2. Web コンソールでのサービスの管理

systemd のターゲット、サービス、ソケット、タイマー、およびパスを管理するには、**Web コンソール** で **Services** を選択します。ここでステータス確認、開始または停止、もしくは有効化または無効化を設定できます。

図1.2 Web コンソールでのサービスの管理



1.5.3. systemd サービス関連情報

systemd に関する詳細は 10章*systemd によるサービス管理* を参照してください。

1.6. ファイアーウォール、SELINUX、および SSH ログインを使用したシステムセキュリティーの強化

コンピューターセキュリティーとは、盗難やダメージからハードウェア、ソフトウェア、または情報を保護したり、提供するサービスの中断や誤りからコンピューターシステムを保護したりすることです。したがって、コンピューターセキュリティーの保護は、機密データやビジネストランザクションを扱う企業だけではなく、すべてのお客様に欠かせないタスクになります。

コンピューターのセキュリティーには、多種多様の機能およびツールがあります。本セクションでは、オペレーティングシステムのインストール後に設定が必要な基本的なセキュリティー機能のみを説明します。Red Hat Enterprise Linux 7 のセキュリティー保護に関する詳細は Red Hat Enterprise Linux 7 セキュリティーガイド を参照してください。

1.6.1. ファイアウォールが有効で実行しているのを確認

1.6.1.1. ファイアウォールの概要およびシステムセキュリティーの強化方法

ファイアウォールは、デフォルトのセキュリティールールに基づいてネットワークトラフィックの送受信の監視および制御を行うネットワークセキュリティーシステムです。ファイアウォールは、通常、信頼できる安全な内部ネットワークと、その他の外部ネットワークとの間に壁を作ります。

Red Hat Enterprise Linux 7 では、**firewalld** サービスがファイアーウォールを提供します。このサービスは、Red Hat Enterprise Linux のインストール時に自動的に有効になりますが、キックスタートの設定などでこのサービスを明示的に無効にした場合は、「ファイアウォールサービスの再有効化」 に従って、再度有効にすることができます。Kickstart ファイルにおけるファイアーウォールの設定オプションの概要は Red Hat Enterprise Linux 7 インストールガイド を参照してください。

1.6.1.2. ファイアウォールサービスの再有効化

インストール後に **firewalld** サービスが無効になっている場合は、再度有効にすることを Red Hat は推奨します。

一般ユーザー権限で、**firewalld** の現在のステータスを表示します。

```
~]$ systemctl status firewalld
```

firewalld が無効で未実行の場合は、**root** ユーザーに切り替えて、そのステータスを変更します。

```
~]# systemctl start firewalld
```

```
~]# systemctl enable firewalld
```

firewalld に関するインストール後の手順は Red Hat Enterprise Linux 7 セキュリティーガイド を参照してください。ファイアーウォールの設定および使用に関する詳細は Red Hat Enterprise Linux 7 セキュリティーガイド を参照してください。

1.6.2. SELinux の適切な状態の確認

1.6.2.1. SELinux の概要およびシステムセキュリティーの強化方法

Security Enhanced Linux (SELinux)は、どのプロセスがどのファイル、ディレクトリー、ポートにアクセスできるのかを指定するシステムセキュリティーの追加レイヤーです。

SELinux のステータス
SELinux のステータスには、以下の 2 つがあります。

- 有効
- 無効

SELinux が無効の場合は、Discretionary Access Control (DAC) ルールだけが使用されます。

SELinux モード
SELinux が有効な場合は、以下のいずれのモードで実行できます。

- Enforcing
- Permissive

Enforcing モードは、**SELinux** のポリシーが強制されることを意味します。**SELinux** は、**SELinux** ポリシールールに基づいてアクセスを拒否し、特別に許可された対話だけを有効にします。Enforcing モードは、インストール後のデフォルトモードで、最も安全な **SELinux** モードです。

Permissive モードは、**SELinux** のポリシーが強制されていないことを意味します。**SELinux** はアクセスを拒否しませんが、Enforcing モードでは拒否されたであろうアクションの拒否がログに記録されません。Permissive モードは、インストール時のデフォルトのモードです。Permissive モードは、問題のトラブルシューティング時に AVC (アクセスベクターキャッシュ) へのアクセスを拒否する必要がある場合など、特定のケースで役立ちます。

Red Hat Enterprise Linux 7 の **SELinux** に関する詳細は Red Hat Enterprise Linux 7 SELinux ユーザーおよび管理者のガイド を参照してください。

1.6.2.2. SELinux の状態の確認

デフォルトでは、**SELinux** は、インストール時には Permissive モードで動作し、インストールが完了すると Enforcing モードで動作します。

ただし、**SELinux** を明示的に Permissive モードに設定している場合や、インストール済みのオペレーティングシステムで無効になっている場合もあります。これは、たとえば、キックスタート設定で設定できます。キックスタートファイルにおける SELinux 設定オプションの概要は Red Hat Enterprise Linux 7 インストールガイド を参照してください。

重要

Red Hat は、Enforcing モードでシステムを使用することを推奨します。

現在の SELinux モードを表示し、必要に応じてモードを設定するには、以下を実行します。

SELinux の状態の確認

1. 現在有効な **SELinux** モードを表示します。

```
~]$ getenforce
```

2. 必要に応じて SELinux モードを切り替えます。
   切り替えは、一時的または永続的を選択できます。一時的な切り替えでは、システムを再起動すると設定が元に戻りますが、永続的に切り替えると、システムの再起動後もその設定が持続します。

   - 一時的に Enforcing モードまたは Permissive モードのいずれかに切り替えるには、以下を実行します。

```
~]# setenforce Enforcing
```

```
~]# setenforce Permissive
```

- **SELinux** モードを永続的に設定するには、**/etc/selinux/config** 設定ファイルで **SELINUX** 変数を変更します。
  たとえば、**SELinux** を Enforcing モードに切り替えるには、以下のように設定します。

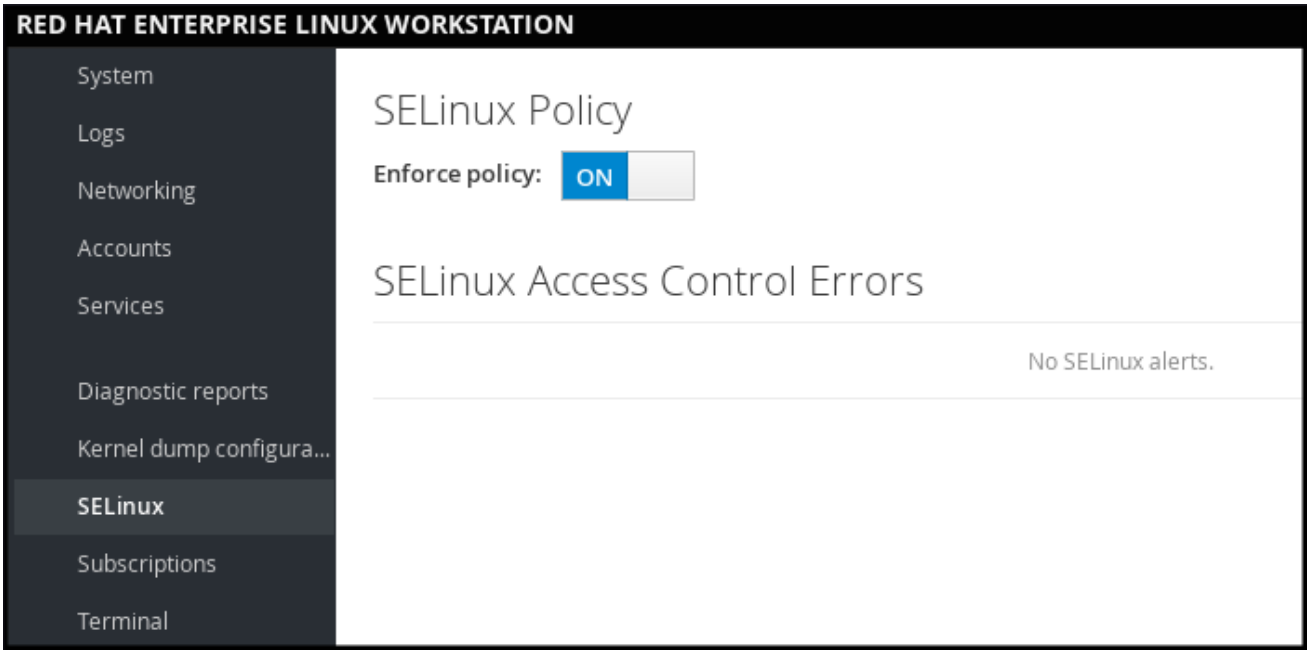
```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
SELINUX=enforcing
```

1.6.2.3. Web コンソールでの SELinux の管理

Web コンソール で、**SELinux** オプションを使用して **SELinux** の Enforcing ポリシーを有効または無効にします。

デフォルトでは、**Web コンソール** の **SELinux** の Enforcing ポリシーが有効になっており、**SELinux** が Enforcing モードで動作します。このモードを無効にして、**SELinux** を Permissive モードに切り替えることができます。このように、/etc/sysconfig/selinux ファイルのデフォルト設定から変更した内容は、次回システムを起動すると自動的に元に戻ります。

図1.3 Web コンソールでの SELinux の管理



1.6.3. SSH ベースの認証の使用

1.6.3.1. SSH ベースの認証の概要およびシステムセキュリティーの強化方法

別のコンピューターとの通信の安全性を確保したい場合は、SSH ベースの認証を使用できます。

SSH (Secure Shell) は、クライアントとサーバーとの間の通信を容易にし、SSH を実行するホストシステムにユーザーがリモートでログインできるようにするプロトコルです。SSH は接続を暗号化します。

クライアントは、暗号化した認証情報をサーバーへ送信します。セッション中に送受信したすべてのデータは暗号化されて転送されます。

SSH は、パスワードなしでユーザーが認証できるようにします。SSH で、公開鍵/秘密鍵のスキームを使用してこれを行います。

SSH の保護手段に関する詳細は 「主な特長」 を参照してください。

1.6.3.2. SSH 接続の確立

SSH 接続を使用できるようにするには、公開鍵と秘密鍵からなる鍵ペアを作成します。

鍵ファイルを作成してサーバーへコピー

1. 公開鍵と秘密鍵を生成するには、以下を実行します。

```
~]$ ssh-keygen
```

この鍵はともに ~/**.ssh**/ ディレクトリーに保存されます。

- ~/**.ssh/id_rsa.pub**: 公開鍵
- ~/**.ssh/id_rsa**: 秘密鍵
  公開鍵が秘密である必要はありません。秘密鍵の確認に使用されます。秘密鍵は秘密となります。秘密鍵を、鍵の生成プロセスで指定するパスフレーズで保護するように選択できます。パスフレーズにより認証はさらに安全となりますが、これを設定するとパスワードが毎回必要になります。パスワードを毎回入力するのを回避するには、**ssh-agent** コマンドを利用します。これにより、パスフレーズを入力するのはセッション開始時の 1 回のみとなります。**ssh-agent** 設定に関する詳細は 「鍵ベース認証の使用」 を参照してください。

2. 最近変更した公開鍵を、ログインするリモートマシンにコピーします。

```
~]# ssh-copy-id USER@hostname
```

その結果、パスワードを入力することなく、安全な方法でシステムにログインできるようになります。

1.6.3.3. SSH root ログインの無効化

デフォルトで有効になっている **root** ユーザーの SSH アクセスを無効にすることで、システムセキュリティーを高めることができます。

このトピックに関する詳細は Red Hat Enterprise Linux 7 セキュリティーガイド を参照してください。

SSH root ログインの無効化

1. **/etc/ssh/sshd_config** ファイルにアクセスします。

```
~]# vi /etc/ssh/sshd_config
```

2. **#PermitRootLogin yes** と書かれた行を以下のように変更します。

```
PermitRootLogin no
```

3. **sshd** サービスを再起動します。

```
~]# systemctl restart sshd
```

1.7. ユーザーアカウント管理の基礎

Red Hat Enterprise Linux 7 は、マルチユーザー向けのオペレーティングシステムです。つまり、1 台のマシンにインストールされた 1 つのシステムに、複数のユーザーが別々のコンピューターからアクセスできます。各ユーザーは自身のアカウントで操作します。このような方法でユーザーアカウントを管理することは、Red Hat Enterprise Linux のシステム管理の中心的要素になります。

通常のアカウントおよびシステムアカウント
通常のアカウントは特定システムのユーザー用に作成されます。このようなアカウントは、通常のシステム管理中に追加、削除、および修正できます。

システムアカウントは、システムで特定のアプリケーション識別子を表します。このようなアカウントは通常、ソフトウェアのインストール時にのみ追加または操作され、後で変更することはありません。

警告

システムアカウントは、システムでローカルに利用できると想定されています。アカウントがリモートで設定され、提供されている (LDAP の設定など) と、システムが破損したり、サービスが開始できない場合があります。

システムアカウント用に、1000 番未満のユーザー ID が予約されています。通常のアカウントには、1000 から始まる ID を使用できます。ただし、5000 以降の ID を割り当てることが推奨されます。詳細は、「ユーザーとグループの概要」 を参照してください。ID 割り当てのガイドラインは **/etc/login.defs** ファイルで参照できます。

```
# Min/max values for automatic uid selection in useradd
#
UID_MIN         1000
UID_MAX        60000
# System accounts
SYS_UID_MIN       201
SYS_UID_MAX       999
```

グループの概要およびその使用目的
グループとは、複数のユーザーアカウントを共通目的 (特定のファイルにアクセス権を与えるなど) で統合するエンティティーです。

1.7.1. ユーザーアカウントとグループを管理する最も基本的なコマンドラインツール

ユーザーアカウントとグループを管理する最も基本的なタスク、および適切なコマンドラインツールは、以下のとおりです。

- ユーザー ID およびグループ ID を表示します。

```
~]$ id
```

- ユーザーアカウントを新規作成します。

```
~]# useradd [options] user_name
```

- username に属するユーザーアカウントに、新しいパスワードを割り当てます。

```
~]# passwd user_name
```

- グループにユーザーを追加します。

```
~]# usermod -a -G group_name user_name
```

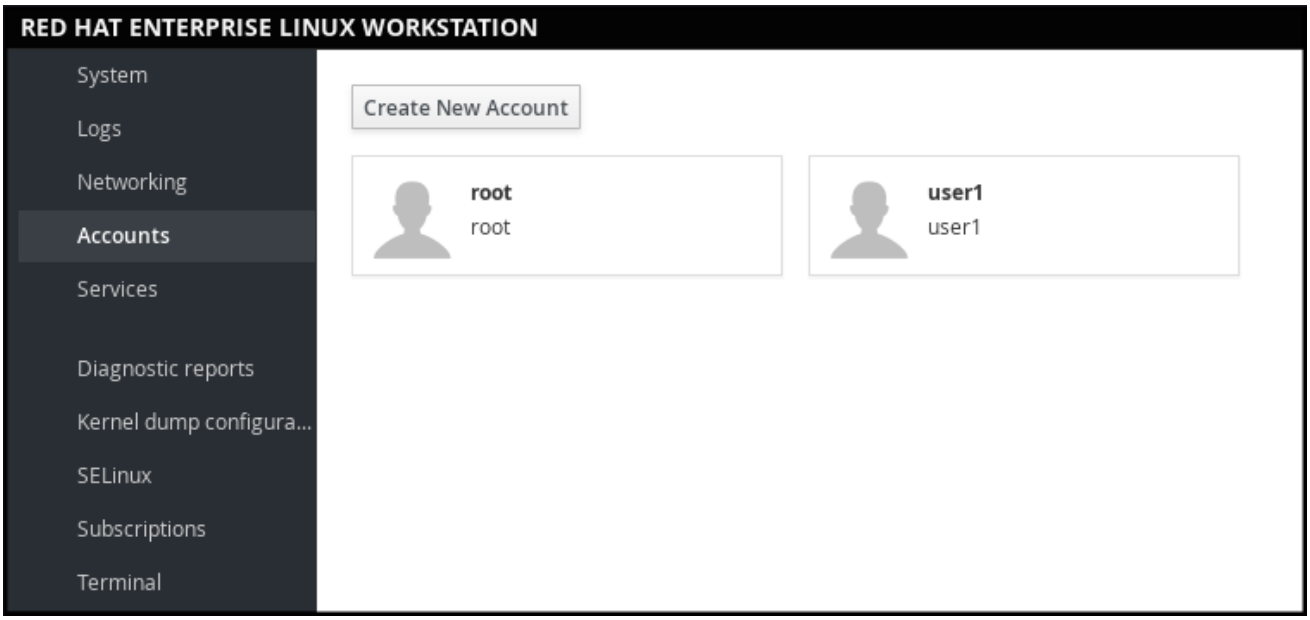
ユーザーおよびグループの管理方法は *4章 ユーザーとグループの管理* を参照してください。

ユーザーおよびグループの管理に GUI (グラフィカルユーザーインターフェイス) を使用する場合は「グラフィカル環境でのユーザーの管理」 を参照してください。

1.7.2. Web コンソールでのユーザーアカウントの管理

Web コンソール でアカウントを管理するには、**Accounts** メニューを選択します。

図1.4 Web コンソールでのユーザーアカウントの管理



1.8. KDUMP メカニズムを使用したクラッシュカーネルのダンプ

本セクションは、**kdump** と呼ばれるカーネルクラッシュダンプメカニズムの概要を説明します。「kdump の概要と使用できるタスク」 では、**kdump** で使用されるものを簡単に説明します。

kdump サービスの有効化はインストールプロセスで行われ、デフォルトではインストール時に **kdump** が有効になります。本セクションでは、「インストールプロセス中の kdump の有効化および実行」で、インストール時に **kdump** を有効にする方法を説明し、「インストールプロセス後に kdump のインストールと有効化」 で、インストール後に無効の **kdump** サービスを手動で有効にする方法を説明します。

Web コンソール を使用して **kdump** を設定できます。詳細は、「Web コンソールでの kdump の設定」を参照してください。

1.8.1. kdump の概要と使用できるタスク

システムがクラッシュした場合は、**kdump** と呼ばれるカーネルクラッシュダンプのメカニズムを利用できます。これにより、システムのメモリー内容を保存し、後で分析することができるようになります。**kdump** では、kexec システムコールにより、別のカーネルのコンテキストから Linux カーネルを起動し、BIOS を迂回して、通常は失われてしまう 1 番目のカーネルメモリーの内容を維持するメカニズムを採用しています。

カーネルクラッシュが発生すると、**kdump** は kexec を使用して 2 番目のカーネル (キャプチャーカーネル) で起動します。この 2 番目のカーネルはシステムメモリーの予約部分にあり、1 番目のカーネルからはアクセスできません。2 番目のカーネルが起動すると、クラッシュしたカーネルメモリーの内容 (クラッシュダンプ) をキャプチャーして保存します。

1.8.2. インストールプロセス中の kdump の有効化および実行

インストール中の **kdump** の有効化および実行は、Anaconda インストーラー、またはキックスタートファイルの **%addon com_redhat_kdump** コマンドのいずれかを使用して行います。

詳細は、インストール方法に応じた適切な資料を参照してください。

- Anaconda インストーラーを使用してインストールする場合は、以下を参照してください。
  Red Hat Enterprise Linux 7 インストールガイドの Anaconda を使用したインストール
- キックスタートファイルを使用してインストールする場合は、以下を参照してください。
  Red Hat Enterprise Linux 7 インストールガイドの キックスタートのコマンドとオプション

1.8.3. インストールプロセス後に kdump のインストールと有効化

kdump がインストールされているのを確認し、設定するには、以下を行います。

kdump がインストールされたかどうかの確認、および kdump の設定

1. システムに **kdump** がインストールされているかどうかを確認するには、以下のコマンドを実行します。

   ```
   ~]$ rpm -q kexec-tools
   ```

2. **kdump** がインストールされていない場合は、**root** で以下のコマンドを実行すればインストールできます。

   ```
   ~]# yum install kexec-tools
   ```

3. **kdump** を設定するには、以下を行います。
   コマンドラインまたはグラフィカルユーザーインターフェイスのいずれかを使用します。

   両方のオプションの詳細は Red Hat Enterprise Linux 7 カーネルクラッシュダンプガイド を参照してください。

   グラフィカル設定ツールをインストールする必要がある場合は、以下を実行します。

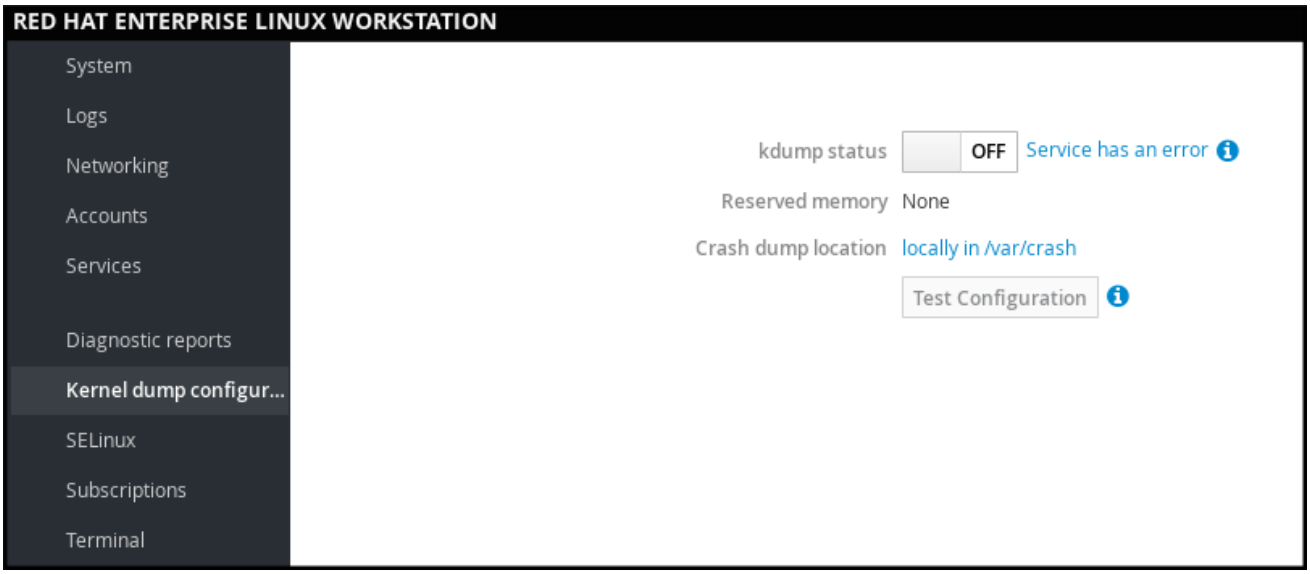
   ```
   ~]# yum install system-config-kdump
   ```

1.8.4. Web コンソールでの kdump の設定

Web コンソール で、**カーネルダンプ** を選択して以下を確認します。

- kdump ステータス
- kdump に予約されているメモリー量
- クラッシュダンプファイルの場所

図1.5 Web コンソールでの kdump の設定



1.8.5. kdump 関連情報

kdump に関する詳細は Red Hat Enterprise Linux 7 カーネルクラッシュダンプガイド を参照してください。

1.9. REAR を使用したシステムレスキューの実行およびシステムバックアップの作成

ソフトウェアやハードウェアの不具合でオペレーティングシステムが破損した場合は、システムを元に戻すためのメカニズムが必要です。システムの復旧にも、バックアップが役に立ちます。Red Hat は、この両方のニーズを満たすために、ReaR (Relax-and-Recover) ツールの使用を推奨します。

1.9.1. ReaR の概要および使用できるタスク

ReaR は、完全なレスキューシステムの作成を実現する障害復旧およびシステム移行ユーティリティーです。デフォルトでは、このレスキューシステムは、ストレージのレイアウトとブートローダーのみを復元し、実際のユーザーおよびシステムファイルは復元しません。

バックアップソフトウェアを使用すると、障害復旧向けに ReaR を統合できます。

ReaR を使用すると、以下のタスクを実行できます。

- 新規ハードウェア上でレスキューシステムを起動する
- オリジナルのストレージレイアウトを複製する
- ユーザーおよびシステムファイルを復元する

1.9.2. ReaR のインストールおよび設定のクイックスタート

ReaR をインストールするには、**root** ユーザーになり、以下のコマンドを実行します。

```
~]# yum install rear
```

/etc/rear/local.conf ファイルの設定を使用して、ReaR を設定します。

詳細は、「基本的な ReaR の使用方法」 を参照してください。

1.9.3. ReaR を使用したレスキューシステム作成のクイックスタート

レスキューシステムを作成するには、root ユーザーになり、以下のコマンドを実行します。

```
~]# rear mkrescue
```

ReaR を使用したレスキューシステムの作成方法は 「レスキューシステムの作成」 を参照してください。

1.9.4. バックアップソフトウェアを使用して ReaR を設定するクイックスタート

ReaR には、NETFS と呼ばれる、完全に統合されたビルトインまたは内部のバックアップメソッドが含まれます。

ReaR が内部のバックアップメソッドを使用するように設定するには、**/etc/rear/local.conf** ファイルに以下の行を追加します。

```
BACKUP=NETFS
BACKUP_URL=backup location
```

/etc/rear/local.conf に以下の行を追加すると、新規バックアップの作成時にこれまでのバックアップアーカイブを維持しておくように ReaR を設定できます。

```
NETFS_KEEP_OLD_BACKUP_COPY=y
```

増分バックアップ (実行するたびに変更されたファイルのみがバックアップされる) を設定する場合は、以下の行を **/etc/rear/local.conf** に追加します。

```
BACKUP_TYPE=incremental
```

ReaR NETFS の内部バックアップメソッドの使用方法は 「ビルトインバックアップの場合」 を参照してください。

サポート対象の外部バックアップメソッドおよびサポート対象外のバックアップメソッドの詳細は 「サポート対象のバックアップメソッド」 および 「サポート対象外のバックアップメソッド」 を参照してください。

1.10. 問題のトラブルシューティングにおけるログファイルの使用

問題をトラブルシューティングする際に、オペレーティングシステムに関するさまざまな情報とメッセージが含まれるログファイルを利用できます。Red Hat Enterprise Linux 7 におけるロギングシステムは、ビルトインの **syslog** プロトコルに基づいています。特定のプログラムがこのシステムを使用して

イベントを記録し、ログファイルに分類します。 これは、オペレーティングシステムの監査およびさまざまな問題のトラブルシューティングに役立ちます。

ログファイルの詳細は、*23章 ログファイルの表示と管理* を参照してください。

1.10.1. syslog メッセージを処理するサービス

syslog メッセージは、2 つのサービスで処理されます。

- **systemd-journald** デーモン: カーネル、システムの起動プロセスの初期段階、デーモンを開始して実行する際の標準出力およびエラー、syslog からのメッセージを収集し、それらのメッセージをさらに処理するために **rsyslog** サービスに転送します。
- **rsyslog** サービスは、タイプおよび優先順で syslog のメッセージを分類し、**/var/log** ディレクトリー内のファイルに書き込みます。 ここでは、ログが永続的に保存されます。

1.10.2. syslog メッセージを保存するサブディレクトリー

syslog メッセージは、そこに含まれるメッセージやログの種類に応じて、**/var/log** ディレクトリー配下のさまざまなサブディレクトリーに保存されます。

- **var/log/messages** - 以下に挙げるものを除いたすべての syslog メッセージ
- **var/log/secure** - セキュリティーおよび認証に関連するメッセージおよびエラー
- **var/log/maillog** - メールサーバーに関連するメッセージおよびエラー
- **var/log/cron** - 定期的に実行されるタスクに関連するログファイル
- **var/log/boot.log** - システムの起動に関連するログファイル

1.11. RED HAT サポートへのアクセス

Red Hat サポートを利用する場合は、Red Hat カスタマーポータル にアクセスしてください。カスタマーポータルでは、サブスクリプションで利用可能なものをすべて提供します。

このセクションでは、以下について説明します。

- Red Hat のサポートを利用する場合は 「Red Hat カスタマーポータルで利用できる Red Hat サポート」 を参照してください。
- **SOS レポート** を使用した問題のトラブルシューティングは 「SOS レポートを使用した問題のトラブルシューティング」 を参照してください。

1.11.1. Red Hat カスタマーポータルで利用できる Red Hat サポート

Red Hat カスタマーポータル を使用すると、以下のことができます。

- 新しいサポートケースの作成
- Red Hat 専門スタッフとのライブチャットを開始する
- 電話または電子メールで Red Hat 専門スタッフに問い合わせる

Red Hat カスタマーポータルには、https://access.redhat.com からアクセスしてください。

Red Hat カスタマーポータルサービスでは、以下の方法で Red Hat サポートをご利用いただけます。

- Web ブラウザー
- **Red Hat Support Tool**

1.11.1.1. Red Hat Support Tool の概要および利用できるタスク

Red Hat Support Toolは、サブスクリプションベースの Red Hat アクセスサービスにテキストコンソールインターフェイスを提供するコマンドラインベースのツールです。このツールは、**redhat-support-tool** パッケージに含まれています。

Red Hat Support Toolを利用すると、以下のようなサポート関連のタスクを実行できるようになります。

- サポートケースの作成または更新
- Red Hat ナレッジベースソリューションでの検索
- Python および Java のエラーの分析

インタラクティブモードでツールを起動するには、以下のコマンドを入力します。

```
~]$ redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help):
```

インタラクティブモードで **?** を入力すると、利用可能なコマンドが表示されます。

```
Command (? for help): ?
```

Red Hat Support Tool のインストールおよび利用に関する詳細は、*8章Red Hat Support Tool を使用したサポートへのアクセス* および Red Hat ナレッジベースの記事 Red Hat Access の Red Hat Support Tool を参照してください。

1.11.2. SOS レポートを使用した問題のトラブルシューティング

SOS レポート は設定の詳細、システム情報、および診断情報を Red Hat Enterprise Linux システムから収集します。サポートケースを作成する際にその SOS レポートを添付してください。

SOS レポート は、**sos** パッケージで提供されています。 これは、Red Hat Enterprise Linux 7 のデフォルトの最小インストールでは提供されません。

sos パッケージをインストールするには、以下のコマンドを実行します。

```
~]# yum install sos
```

SOS レポート を生成するには、以下のコマンドを実行します。

```
~]# sosreport
```

サポートケースに **SOS レポート** を添付する方法は、Red Hat ナレッジベースの記事 How can I attach a file to a Red Hat support case? を参照してください。**SOS レポート** を添付すると、サポートケース番号の入力を促すプロンプトが表示されます。

SOS レポート の詳細は、Red Hat ナレッジベースの記事 Red Hat Enterprise Linux 4.6 以降における sosreport のロールと取得方法 を参照してください。

第2章 システムロケールおよびキーボード設定

システムロケール では、システムサービスおよびユーザーインターフェイスの言語設定を指定します。**キーボードレイアウト** の設定では、テキストコンソールおよびグラフィカルユーザーインターフェイスで使用するレイアウトを管理します。

これらの設定は、**/etc/locale.conf** 設定ファイルを修正するか **localectl** ユーティリティーを使用して行います。グラフィカルユーザーインターフェイスを使用して設定することもできます。詳細は Red Hat Enterprise Linux 7 インストールガイド を参照してください。

2.1. システムロケールの設定

システム全体にわたるロケール設定は **/etc/locale.conf** ファイルに保存され、システム起動の初期段階で **systemd** デーモンにより読み込まれます。**/etc/locale.conf** に設定したロケール設定は、個別のプログラムやユーザーが上書きしない限り、すべてのサービスやユーザーに継承されます。

/etc/locale.conf の基本的なファイル形式は、改行で区切られた変数割り当てのリストです。たとえば、ロケールがドイツ語でメッセージが英語の場合、**/etc/locale.conf** は以下のようになります。

```
LANG=de_DE.UTF-8
LC_MESSAGES=C
```

ここでは LC_MESSAGES オプションを使用して、標準エラー出力に書き出される診断メッセージ用ロケールを決定します。**/etc/locale.conf** でさらにロケール設定を指定するには、いくつかのオプションが使用でき、関連性の高いものが 表2.1「/etc/locale.conf で設定可能なオプション」 にまとめられています。これらのオプションの詳細は、man ページの **locale(7)** を参照してください。**/etc/locale.conf** には、すべてのオプションを可能にする LC_ALL オプションは設定しないように注意してください。

表2.1 /etc/locale.conf で設定可能なオプション

オプション	詳細
LANG	システムロケールのデフォルト値になります。
LC_COLLATE	ローカルのアルファベット文字列を比較する機能の動作を変更します。
LC_CTYPE	文字処理、分類機能、マルチバイト文字機能の動作を変更します。
LC_NUMERIC	数値が通常出力される方法を設定します (小数点を表すコンマなど)。
LC_TIME	現在の時間表記を、24 時間表記または 12 時間表記に変更します。
LC_MESSAGES	標準エラー出力に書き出される診断メッセージに使用されるロケールを決定します。

2.1.1. 現行ステータスの表示

localectl コマンドを使用すると、システムロケールおよびキーボードレイアウト設定へのクエリーまたは変更が可能になります。現在の設定を表示するには、**status** オプションを使用します。

localectl status

例2.1 現行ステータスの表示

上記のコマンドを実行すると、ロケールの現行設定と、コンソールおよび X11 ウィンドウシステムに設定されているキーボードレイアウトが出力されます。

```
~]$ localectl status
  System Locale: LANG=en_US.UTF-8
    VC Keymap: us
   X11 Layout: n/a
```

2.1.2. 利用可能なロケールのリスト表示

ご使用のシステムで利用可能なロケールをリスト表示するには、以下を入力します。

localectl list-locales

例2.2 ロケールのリスト表示

特定の英語ロケールを選択するとき、そのロケールがシステムで利用可能であるかどうかわからないとします。以下のコマンドを使用すると、英語ロケールのリストを表示することで確認できます。

```
~]$ localectl list-locales | grep en_
en_AG
en_AG.utf8
en_AU
en_AU.iso88591
en_AU.utf8
en_BW
en_BW.iso88591
en_BW.utf8

output truncated
```

2.1.3. ロケールの設定

デフォルトのシステムロケールを設定するには、**root** で以下のコマンドを使用します。

localectl set-locale LANG=locale

locale を、**localectl list-locales** コマンドで見つかったロケール名に置き換えます。上記の構文は、表 2.1「/etc/locale.conf で設定可能なオプション」 のパラメーター設定にも使用できます。

例2.3 デフォルトロケールの変更

たとえば、イギリス英語をデフォルトのロケールに設定する場合は、最初に **list-locales** を使用して、このロケールの名前を見つけます。次に、**root** で以下の形式のコマンドを入力します。

```
~]# localectl set-locale LANG=en_GB.utf8
```

2.1.4. キックスタートを使用したインストール時にシステムロケールの設定を永続化

Red Hat キックスタートインストールを使用して Red Hat Enterprise Linux をインストールすると、オペレーティングシステムをアップグレードした後にシステムロケールの設定が永続化されないことがあります。

キックスタートファイルの **%packages** セクションに **--instLang** オプションが含まれる場合は、**_install_langs** RPM マクロはこのインストールの特定の値に設定され、インストールしたロケールのセットが調整されます。ただし、この設定は今回のインストールにのみ影響し、その後のアップグレードには影響しません。アップグレードの際に **glibc** パッケージを再インストールすると、インストール中にユーザーがリクエストしたロケールではなく、ロケール全体がアップグレードされます。

これを回避するには、永続化させるロケールを選択します。次のような方法があります。

- キックスタートインストールを開始する前であれば、キックスタートインストール中に RPM マクロの設定 の手順に従ってキックスタートファイルを修正し、RPM マクロをグローバルに設定する指示を追加します。
- システムをすでにインストールしている場合は、システム全体に RPM マクロの設定 の手順に従って、RPM マクロをシステム全体に設定します。

キックスタートインストール中に RPM マクロの設定

1. キックスタートファイルの **%post** セクションを変更します。

```
LANG=en_US
echo "%_install_langs $LANG" > /etc/rpm/macros.language-conf

yum-config-manager --setopt=override_install_langs=$LANG --save
```

2. キックスタートファイルの **%packages** セクションを変更します。

```
%packages
yum-utils*
%end
```

システム全体に RPM マクロの設定

1. 以下の設定を追加した RPM 設定ファイルを **/etc/rpm/macros.language-conf** に作成します。

```
%_install_langs LANG
```

LANG は、**instLang** オプションの値です。

2. 以下を使用して、**/etc/yum.conf** ファイルを更新します。

```
override_install_langs=LANG
```

2.2. キーボードレイアウトの変更

キーボードレイアウト設定では、ユーザーはテキストコンソールとグラフィカルユーザーインターフェイスで使用するレイアウトを管理できます。

2.2.1. 現行設定の表示

上記の説明にあるように、現行のキーボードレイアウト設定は、以下のコマンドで確認できます。

localectl status

例2.4 キーボード設定の表示

以下の出力では、仮想コンソールおよび X11 ウィンドウシステム用に設定されているキーボードレイアウトが表示されています。

```
~]$ localectl status
  System Locale: LANG=en_US.utf8
    VC Keymap: us
   X11 Layout: us
```

2.2.2. 利用可能なキーマップのリスト表示

システムに設定可能なキーボードレイアウトのリストを表示するには、以下を入力します。

localectl list-keymaps

例2.5 特定のキーマップの検索

grep を使用すると、上記のコマンド出力から特定のキーマップ名を探すことができます。現在設定されているロケールと互換性のあるキーマップが複数存在する場合も少なくありません。たとえば、利用可能な Czech キーボードレイアウトを見つけるには、以下のコマンドを実行します。

```
~]$ localectl list-keymaps | grep cz
cz
cz-cp1250
cz-lat2
cz-lat2-prog
cz-qwerty
cz-us-qwertz
sunt5-cz-us
sunt5-us-cz
```

2.2.3. キーマップの設定

システムでデフォルトのキーボードレイアウトを設定するには、**root** で以下のコマンドを使用します。

localectl set-keymap map

map を、**localectl list-keymaps** コマンドの出力から取得したキーマップの名前に置き換えます。**--no-convert** オプションが渡されない限り、選択した設定は、最も一致する X11 キーボードマッピングに変換してから X11 ウィンドウシステムのデフォルトのキーボードマッピングにも適用されます。これは逆の方法でも適用できます。**root** で以下のコマンドを実行すると、両方のキーマップを指定できます。

```
localectl set-x11-keymap map
```

このコンソールレイアウトを X11 レイアウトに設定しない場合は、**--no-convert** オプションを使用します。

```
localectl --no-convert set-x11-keymap map
```

このオプションを使用すると、X11 キーマップではこれまでのコンソールレイアウト設定が引き続き使用されます。

例2.6 X11 キーマップの個別設定

グラフィカルインターフェイスでは German キーボードレイアウトを使用し、コンソール操作では引き続き US キーマップを使用します。これを実行するには、**root** で次のコマンドを実行します。

```
~]# localectl --no-convert set-x11-keymap de
```

ステータスで、設定が正常に行われたかを確認できます。

```
~]$ localectl status
  System Locale: LANG=de_DE.UTF-8
    VC Keymap: us
   X11 Layout: de
```

キーボードレイアウト (**map**) の他に、以下の 3 つのオプションが指定できます。

```
localectl set-x11-keymap map model variant options
```

model はキーボードのモデル名、**variant** はキーボードのバリアント、そして **options** はオプションコンポーネントに置き換えます。このコマンドを使用すると、キーボードの動作を強化できます。これらのオプションは、デフォルトでは設定されていません。X11 モデル、X11 バリアント、および X11 オプションの詳細は、man ページの **kbd(4)** を参照してください。

2.3. 関連情報

Red Hat Enterprise Linux でキーボードレイアウトを設定する方法は、以下を参照してください。

インストールされているドキュメント

- **localectl**(1): **localectl** コマンドラインユーティリティーの man ページには、このツールを使用してシステムロケールとキーボードレイアウトを設定する方法が説明されています。
- **loadkeys**(1): **loadkeys** コマンドの man ページには、このツールを使用して仮想コンソールでキーボードレイアウトを変更する方法が説明されています。

関連項目

- 6章 *権限の取得* では、**su** および **sudo** コマンドを使用して管理者権限を取得する方法を説明し

ています。

- 10章*systemd によるサービス管理* では、**systemd** の詳細情報と、**systemctl** コマンドを使用してシステムサービスを管理する方法が説明されています。

第3章 日付と時刻の設定

最新のオペレーティングシステムは、以下の 2 つのタイプのクロックを区別します。

- **リアルタイムクロック** (RTC) は、一般に **ハードウェアクロック** と呼ばれ、通常、システムボード上の集積回路で、オペレーティングシステムの状態からは完全に独立しており、コンピューターがシャットダウンしても稼働します。
- **システムクロック** は **ソフトウェアクロック** とも呼ばれ、カーネルが維持し、その初期値はリアルタイムクロックに基づいています。システムが起動するとシステムクロックは初期化され、リアルタイムクロックからは完全に独立したものになります。

システム時間は常に **協定世界時** (**UTC**) で維持され、必要に応じてアプリケーション内でローカル時間に変換されます。**ローカルタイム** は、**夏時間** (**DST**) を考慮に入れた現行タイムゾーンの実際の時刻です。リアルタイムクロックは UTC またはローカルタイムのいずれかを使用できます。これは推奨オプションです。

Red Hat Enterprise Linux 7 は、システムの日付と時刻に関する情報を設定および表示するのに使用できる 3 つのコマンドラインツールを提供します。

- **timedatectl** ユーティリティー (Red Hat Enterprise Linux 7 の新機能で、 **systemd** に含まれる)。
- 従来の **date** コマンド
- ハードウェアクロックにアクセスするための **hwclock** ユーティリティー

3.1. TIMEDATECTL コマンドの使用

timedatectl ユーティリティーは、**systemd** システムおよびサービスマネージャーの一部として配布されており、システムクロック設定を確認および変更できます。このツールを使用すると、現在の日付および時間の変更、タイムゾーンの設定、リモートサーバーとシステムクロックとの自動同期の有効化が可能になります。

カスタマイズした形式で現在の日付と時間を表示する方法は 「date コマンドの使用」 も参照してください。

3.1.1. システムの現在日時の表示

現在の日時をシステムおよびハードウェアクロック設定の詳細情報と共に表示するには、**timedatectl** コマンドをオプションなしで実行します。

```
timedatectl
```

これで、ローカル時間、ユニバーサル時間、現在使用しているタイムゾーン、ネットワーク時刻プロトコル (**NTP**) 設定、DST に関する追加情報が表示されます。

例3.1 システムの現在日時の表示

以下の例は、システムクロックとリモートサーバーの同期に **NTP** を使用しないシステムで **timedatectl** コマンドを実行したときの出力です。

```
~]$ timedatectl
  Local time: Mon 2016-09-16 19:30:24 CEST
```

```
Universal time: Mon 2016-09-16 17:30:24 UTC
      Timezone: Europe/Prague (CEST, +0200)
    NTP enabled: no
NTP synchronized: no
 RTC in local TZ: no
      DST active: yes
 Last DST change: DST began at
         Sun 2016-03-31 01:59:59 CET
         Sun 2016-03-31 03:00:00 CEST
 Next DST change: DST ends (the clock jumps one hour backwards) at
         Sun 2016-10-27 02:59:59 CEST
         Sun 2016-10-27 02:00:00 CET
```

重要

timedatectl は、**chrony** または **ntpd** のステータスへの変更を即座に認識しません。これらのツールの設定またはステータスを変更した場合は、以下のコマンドを実行します。

```
~]# systemctl restart systemd-timedated.service
```

3.1.2. システムの現在時刻の変更

システムの現在時刻を変更するには、**root** でシェルプロンプトに以下を入力します。

```
timedatectl set-time HH:MM:SS
```

HH は時間、**MM** は分、**SS** は秒 (すべて 2 桁) の数字に置き換えます。

このコマンドは、システム時間とハードウェアクロックの両方を更新します。結果は、**date --set** および **hwclock --systohc** コマンドの両方を使用する場合と同様になります。

NTP サービスが有効になっていると、このコマンドは失敗します。このサービスを一時的に無効にする方法は 「システムクロックのリモートサーバーとの同期」 を参照してください。

例3.2 システムの現在時刻の変更

システムの現在時刻を午後 11 時 26 分に変更するには、**root** で以下のコマンドを実行します。

```
~]# timedatectl set-time 23:26:00
```

デフォルトでは、システムは UTC を使用するように設定されています。ローカルタイムでクロックを維持するようにシステムを設定するには、**root** 権限で、**set-local-rtc** オプションとともに **timedatectl** コマンドを実行します。

```
timedatectl set-local-rtc boolean
```

ローカルタイムでクロックを維持するようにシステムを設定するには、**boolean** を **yes** (または **y**、**true**、**t**、または **1**) に置き換えます。UTC を使用するようにシステムを設定するには、**boolean** を **no** (または **n**、**false**、**f**、または **0**) に置き換えます。デフォルトオプションは **no** です。

3.1.3. システムの現在日の変更

システムの現在の日付を変更するには、**root** でシェルプロンプトに以下を入力します。

```
timedatectl set-time YYYY-MM-DD
```

YYYY は 4 桁の年に、 MM と DD は 2 桁の月と日に置き換えます。

時間を指定せずに日付を変更すると、時間は 00:00:00 に設定されることに注意してください。

例3.3 システムの現在日の変更

システムの現在日を 2017 年 6 月 2 日に変更し、現在時刻は変更しない (午後 11:26) 場合は、**root** で以下のコマンドを実行します。

```
~]# timedatectl set-time "2017-06-02 23:26:00"
```

3.1.4. タイムゾーンの変更

利用可能なタイムゾーンのリストを表示するには、シェルプロンプトで以下を入力します。

```
timedatectl list-timezones
```

現在使用中のタイムゾーンを変更するには、**root** で以下を入力します。

```
timedatectl set-timezone time_zone
```

time_zone を、**timedatectl list-timezones** コマンドで表示される値に置き換えます。

例3.4 タイムゾーンの変更

現在の場所に最も適したタイムゾーンを特定するには、**timedatectl** コマンドに **list-timezones** オプションを付けて実行します。たとえば、ヨーロッパのタイムゾーンのリストを表示するには、以下のコマンドを実行します。

```
~]# timedatectl list-timezones | grep Europe
Europe/Amsterdam
Europe/Andorra
Europe/Athens
Europe/Belgrade
Europe/Berlin
Europe/Bratislava
...
```

タイムゾーンを **Europe/Prague** に変更するには、**root** で以下を入力します。

```
~]# timedatectl set-timezone Europe/Prague
```

3.1.5. システムクロックのリモートサーバーとの同期

上記の説明にある手動での調整の他に、**timedatectl** コマンドで **NTP** プロトコルを使用して、システムクロックを自動でリモートサーバーのグループと同期させる方法もあります。NTP を有効にすると、**chronyd** サービスまたは **ntpd** サービスのうち、インストールされている方が有効になります。

NTP サービスは、以下のコマンドを使用して有効または無効にできます。

```
timedatectl set-ntp boolean
```

システムでシステムクロックをリモートの **NTP** サーバーと同期させるには、**boolean** を **yes** に置き換えます (デフォルトのオプション)。この機能を無効にするには、**boolean** を **no** に置き換えます。

例3.5 システムクロックのリモートサーバーとの同期

システムクロックとリモートサーバーの自動同期を有効にするには、以下を入力します。

```
~]# timedatectl set-ntp yes
```

このコマンドは、**NTP** サービスがインストールされていないと失敗します。詳細は、「chrony のインストール」 を参照してください。

3.2. DATE コマンドの使用

date ユーティリティーはすべての Linux システムで利用可能で、システムの現在日時の表示および設定を可能にします。システムクロックに関する詳細情報をカスタマイズされた形式で表示するために、スクリプト内で使用されることがよくあります。

タイムゾーンの変更、またはシステムクロックとリモートサーバーの自動同期を有効にする方法は 「**timedatectl** コマンドの使用」 を参照してください。

3.2.1. システムの現在日時の表示

現在の日時を表示するには、**date** コマンドをオプションを付けずに実行します。

```
date
```

このコマンドを実行すると、曜日、日付、ローカルタイム、タイムゾーンの省略形、年が表示されます。

date コマンドは、デフォルトではローカルタイムを表示します。UTC の時間を表示するには、コマンドラインオプション **--utc** または **-u** を指定してコマンドを実行します。

```
date --utc
```

また、**+"format"** オプションをコマンドラインに指定することで、表示される情報のフォーマットをカスタマイズすることもできます。

```
date +"format"
```

例3.6「システムの現在日時の表示」 を参考にして、**フォーマット**を、サポートされる 1 つまたは複数のコントロールシーケンスに置き換えます。よく使用されるフォーマットオプションのリストは 表3.1「よく使われるコントロールシーケンス」 を参照してください。これらのオプションの完全なリストは、man ページの **date**(1) を参照してください。

表3.1 よく使われるコントロールシーケンス

コントロールシーケンス	詳細
%H	HH フォーマットでの時間 (例:**17**)。
%M	MM フォーマットでの分 (例:**30**)。
%S	SS フォーマットでの秒 (例:**24**)。
%d	DD フォーマットでの日 (例:**16**)。
%m	MM フォーマットでの月 (例:**09**)。
%Y	YYYY フォーマットでの年 (例:**2016**)。
%Z	タイムゾーンの省略形 (例: **CEST**)。
%F	非省略形の YYYY-MM-DD フォーマットでの日付 (例: **2016-09-16**)。このオプションは、**%Y-%m-%d** と同等です。
%T	非省略形の HH:MM:SS フォーマットでの時間 (例: 17:30:24)。このオプションは、**%H:%M:%S** と同等です。

例3.6 システムの現在日時の表示

現在日とローカル時間を表示するには、シェルプロンプトで以下を入力します。

```
~]$ date
Mon Sep 16 17:30:24 CEST 2016
```

UTC で現在日時を表示するには、シェルプロンプトで以下を入力します。

```
~]$ date --utc
Mon Sep 16 15:30:34 UTC 2016
```

date コマンドの出力をカスタマイズするには、以下を入力します。

```
~]$ date +"%Y-%m-%d %H:%M"
2016-09-16 17:30
```

3.2.2. システムの現在時刻の変更

システムの現在時刻を変更するには、**root** で **date** コマンドに **--set** オプションまたは **-s** オプションを指定して実行します。

```
date --set HH:MM:SS
```

HH は時間、MM は分、SS は秒 (すべて 2 桁) の数字に置き換えます。

date コマンドは、デフォルトでは、システムクロックをローカルタイムに設定します。システムクロックを UTC で設定するには、コマンドラインオプション **--utc** または **-u** を指定してコマンドを実行します。

```
date --set HH:MM:SS --utc
```

例3.7 システムの現在時刻の変更

システムの現在時刻を午後 11 時 26 分に変更するには、**root** で以下のコマンドを実行します。

```
~]# date --set 23:26:00
```

3.2.3. システムの現在日の変更

現在の日付を変更するには、**root** で **--set** または **-s** を指定して **date** コマンドを実行します。

```
date --set YYYY-MM-DD
```

YYYY は 4 桁の年に、MM と DD は 2 桁の月と日に置き換えます。

時間を指定せずに日付を変更すると、時間は 00:00:00 に設定されることに注意してください。

例3.8 システムの現在日の変更

システムの現在日を 2017 年 6 月 2 日に変更し、現在時刻は変更しない (午後 11:26) 場合は、**root** で以下のコマンドを実行します。

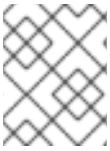
```
~]# date --set "2017-06-02 23:26:00"
```

3.3. HWCLOCK コマンドの使用

hwclock は、ハードウェアクロックにアクセスするためのユーティリティーです。これは、リアルタイムクロック (RTC) とも呼ばれています。ハードウェアクロックは使用中のオペレーティングシステムから独立しており、マシンがシャットダウンしても作動します。このユーティリティーは、ハードウェアクロックからの時間を表示するために使用されます。また、**hwclock** には、ハードウェアクロック内のシステム上のドリフトを補正する機能も含まれています。

ハードウェアクロックタイムは、年、月、日、時間、分、秒の値を保存します。ローカルタイムや協定世界時 (UTC) の時刻を保存したり、夏時間 (DST) を設定したりすることはできません。

hwclock ユーティリティーは、**/etc/adjtime** ファイルにその設定を保存します。このファイルは、時刻を手動で設定したり、ハードウェアクロックをシステム時間に同期したりするなどの初回の変更時に作成されます。



注記

Red Hat Enterprise Linux バージョン 6 から 7 の間の **hwclock** の動作の変更については、Red Hat Enterprise Linux 7 移行計画ガイド を参照してください。

3.3.1. システムの現在日時の表示

root ユーザーで、オプションを付けずに **hwclock** を実行すると、標準出力にローカルタイムの日時が返されます。

```
hwclock
```

hwclock コマンドで **--utc** オプションまたは **--localtime** オプションを指定しても、ハードウェアクロック時間が UTC またはローカルタイムで表示される訳ではないことに注意してください。これらのオプションは、ハードウェアクロックのいずれかの設定を変更するために使用されます。この時間は常にローカル時間で表示されます。したがって、**hwclock --utc** または **hwclock --local** コマンドを実行しても、**/etc/adjtime** 保存された設定が変更されるわけではありません。このコマンドは、**/etc/adjtime** に保存されている設定が正しくないことを把握していて設定を変更したくない場合に便利です。一方、このコマンドを誤った方法で使用すると、誤解を招く情報を受け取る可能性があります。詳細は、**hwclock**(8) man ページを参照してください。

例3.9 システムの現在日時の表示

ハードウェアクロックからシステムの現在の日付およびローカルタイムを表示するには、**root** で以下を実行します。

```
~]# hwclock
Tue 15 Apr 2017 04:23:46 PM CEST  -0.329272 seconds
```

CEST は 中央ヨーロッパ夏時間 (Central European Summer Time) の省略形です。

タイムゾーンの変更方法は 「タイムゾーンの変更」 を参照してください。

3.3.2. 日付と時刻の設定

ハードウェアクロックは、日付と時刻を表示するほかに、手動で特定の時刻に設定することができます。

ハードウェアクロックの日時を変更する場合は、コマンドに **--set** および **--date** オプションを追加します。

```
hwclock --set --date "dd mmm yyyy HH:MM"
```

dd は日 (2 桁)、**mmm** は月の省略形 (3 文字) に、**yyyy** は年 (4 桁)、**HH** は時間 (2 桁)、**MM** は分 (2 桁) に置き換えます。

同時に、**--utc** または **--localtime** オプションをそれぞれ追加して、UTC またはローカルタイムのいずれかで時刻を維持するようにハードウェアクロックを設定することもできます。この場合、**UTC** または **LOCAL** は **/etc/adjtime** ファイルに記録されます。

例3.10 ハードウェアクロックの特定の日時への設定

たとえば、日時を 2016 年 10 月 21 日 21:17 に設定し、UTC のハードウェアクロックを維持する場合は、**root** で以下のコマンドを実行します。

```
~]# hwclock --set --date "21 Oct 2016 21:17" --utc
```

3.3.3. 日付と時刻の同期

ハードウェアクロックとシステムの現在時刻の同期を両方向で実行できます。

- 以下のコマンドを使用して、ハードウェアクロックを、システムの現在時刻に設定できます。

  hwclock --systohc

  NTP を使用すると、ハードウェアクロックは 11 分ごとにシステムクロックに自動的に同期されるため、このコマンドは、システムのブート時に、妥当な初期のシステム時間を取得するためにのみ役立つことに注意してください。

- または、以下のコマンドを使用してハードウェアクロックからシステム時間を設定できます。

  hwclock --hctosys

ハードウェアクロックとシステム時間を同期する場合は、**--utc** または **--localtime** を追加してローカルタイムまたは UTC でハードウェアクロックを保持するかどうかを指定できます。**--set** の使用と同様に、**UTC** または **LOCAL** は **/etc/adjtime** ファイルに記録されます。

hwclock --systohc --utc コマンドは **timedatectl set-local-rtc false** に機能的に類似しており、**hwclock --systohc --local** コマンドは **timedatectl set-local-rtc true** の代替コマンドになります。

例3.11 システムタイムへのハードウェアクロックの同期

ハードウェアクロックをシステムの現在時刻に設定し、ハードウェアクロックをローカルタイムで維持するには、**root** で以下のコマンドを実行します。

```
~]# hwclock --systohc --localtime
```

タイムゾーンおよび DST の切り替えに関する問題を避けるには、ハードウェアクロックを UTC で維持することを推奨します。例3.11「システムタイムへのハードウェアクロックの同期」 は、Windows システムとのマルチブートの場合などに役立ちます。この場合、デフォルトでハードウェアクロックがローカルタイムで実行されることが想定され、その他のシステムもすべてローカルタイムを使用して対応する必要があります。同様に仮想マシンでも必要になる場合があります。ホストが提供する仮想ハードウェアクロックがローカルタイムで実行中の場合は、ゲストシステムもローカルタイムを使用するように設定する必要があります。

3.4. 関連情報

Red Hat Enterprise Linux 7 でキーボードレイアウトを設定する方法は、以下を参照してください。

インストールされているドキュメント

- **timedatectl**(1): **timedatectl** コマンドラインユーティリティーの man ページでは、このツールを使用して、システムクロックおよびその設定をクエリーして変更する方法が説明されています。
- **date**(1): **date** コマンドの man ページでは、サポートされるオプションの完全なリストが提供されます。
- **hwclock**(8): **hwclock** コマンドの man ページでは、サポートされるオプションの完全なリストが提供されます。

関連項目

- 2章 *システムロケールおよびキーボード設定* では、キーボードレイアウトの設定方法を説明しています。
- 6章 *権限の取得* では、**su** および **sudo** コマンドを使用して管理者権限を取得する方法を説明しています。
- 10章 *systemd によるサービス管理* では、systemd の詳細情報と、**systemctl** コマンドを使用してシステムサービスを管理する方法が説明されています。

第4章 ユーザーとグループの管理

ユーザーとグループの制御は、Red Hat Enterprise Linux システム管理の中核となる要素です。本章では、グラフィカルユーザーインターフェイスおよびコマンドラインを使用してユーザーとグループを追加、管理、削除する方法を説明し、グループディレクトリーの作成などの高度なトピックを扱います。

4.1. ユーザーとグループの概要

ユーザーは、人 (物理的なユーザーに結び付けられたアカウント)、または使用する特定のアプリケーションに対して存在するアカウントのいずれかを指し、グループは、共通の目的でユーザーをまとめた組織の論理的表現です。グループ内のユーザーは、そのグループが所有するファイルの読み取り、書き込み、実行を行う権限を共有します。

各ユーザーは、**ユーザー ID** (**UID**) と呼ばれる一意の数値 ID に関連付けられています。同様に、各グループは **グループ ID** (**GID**) に関連付けられています。ファイルを作成するユーザーは、そのファイルの所有者であり、グループ所有者でもあります。ファイルには、所有者、グループ、その他に対して読み取り、書き込み、実行のパーミッションが別々に割り当てられます。ファイル所有者の変更ができるのは **root** のみです。アクセスパーミッションは、**root** ユーザーとファイル所有者の両方が変更できます。

さらに、Red Hat Enterprise Linux はファイルとディレクトリーに対する **アクセス制御リスト** (**ACL**) をサポートします。これにより、所有者以外の特定のユーザーにパーミッションを設定できます。この機能の詳細は 5章 *アクセス制御リスト* を参照してください。

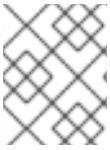
予備のユーザーとグループ ID

Red Hat Enterprise Linux は、1000 以下のユーザー ID とグループ ID を、システムユーザーとグループ用に予約しています。**ユーザー管理** には、デフォルトではシステムユーザーが表示されません。予約されているユーザー ID およびグループ ID の詳細は、**setup** パッケージに記載されています。このドキュメントを表示するには、以下のコマンドを実行します。

```
cat /usr/share/doc/setup*/uidgid
```

予約に使用される ID の範囲は今後広がる可能性があるため、ID には 5,000 以降の番号を割り当てることが推奨されます。新規ユーザーに割り当てる ID を 5,000 から始まるようにするには、**/etc/login.defs** ファイルのディレクティブ **UID_MIN** および **GID_MIN** を変更します。

```
[file contents truncated]
UID_MIN         5000
[file contents truncated]
GID_MIN         5000
[file contents truncated]
```



注記

UID_MIN ディレクティブおよび **GID_MIN** ディレクティブを変更する前に作成したユーザーの UID は、デフォルトの 1,000 から開始します。

新規ユーザーおよびグループの ID を 5,000 から始まるようにした場合でも、システムが予約する ID が 1000 より上にならないようにすることが推奨されます。こうすることで、1000 を上限とするシステムとの競合を避けることができます。

4.1.1. ユーザープライベートグループ

Red Hat Enterprise Linux では、**UPG (ユーザープライベートグループ)** スキームが使用されているため、UNIX グループを簡単に管理できます。ユーザープライベートグループは、新規ユーザーがシステムに追加されるたびに作成されます。ユーザープライベートグループは作成したユーザーと同じ名前となり、そのユーザーがそのユーザープライベートグループの唯一のメンバーになります。

ユーザープライベートグループを使用すると、新規に作成したファイルやディレクトリーに対して確実にデフォルトのパーミッションを設定できます。作成したユーザーと、**そのユーザーのグループ** の両方がファイルやディレクトリーを修正できるようになります。

新規に作成するファイルまたはディレクトリーに適用される権限を決める設定は **umask** と呼ばれ、**/etc/bashrc** ファイルで設定します。従来の UNIX ベースのシステムでは、**umask** は **022** に設定されており、ファイルまたはディレクトリーを作成したユーザーしか変更できませんでした。このスキームでは、**作成者のグループのメンバーなど**、他のユーザーは変更できませんでした。ただし、UPG スキームでは、すべてのユーザーがそれぞれプライベートグループを持つため、このグループ保護は必須ではなくなりました。詳細は、「**umask** を使用した、新規ファイルのデフォルト権限の設定」 を参照してください。

グループのリストは、**/etc/group** 設定ファイルに保存されます。

4.1.2. シャドウパスワード

マルチユーザー環境では、**shadow-utils** パッケージで提供される **シャドウパスワード** を使用することが非常に重要です。これを使用することで、システムの認証ファイルのセキュリティーを強化できます。このため、インストールプログラムでは、デフォルト設定でシャドウパスワードを有効にしています。

以下は、UNIX ベースシステムでパスワードを格納する従来の方法と比べた場合のシャドウパスワードの利点です。

- シャドウパスワードは、暗号化されたパスワードハッシュを、あらゆるユーザーが読み取り可能な **/etc/passwd** ファイルから、**root** ユーザーのみが読み取り可能な **/etc/shadow** に移動して、システムセキュリティーを向上させます。
- シャドウパスワードは、パスワードエージングに関する情報を保存します。
- シャドウパスワードを使用すると、**/etc/login.defs** ファイルで設定したセキュリティーポリシーの実施が可能になります。

shadow-utils パッケージが提供するほとんどのユーティリティーは、シャドウパスワードが有効かどうかに関わらず適切に動作します。ただし、パスワードエージングの情報は **/etc/shadow** ファイルにのみ格納されているため、シャドウパスワードを有効にしないと、以下のユーティリティーとコマンドは動作しません。

- パスワードエージングパラメーターを設定する **chage** ユーティリティー詳細は、**Red Hat Enterprise Linux 7 セキュリティーガイド**の Password Security セクションをご覧ください。
- **/etc/group** ファイルを管理する **gpasswd** ユーティリティー
- **-e, --expiredate** オプションまたは **-f (--inactive)** オプションを使用した **usermod** コマンド
- **-e, --expiredate** オプションまたは **-f, --inactive** オプションを使用した **useradd** コマンド

4.2. グラフィカル環境でのユーザーの管理

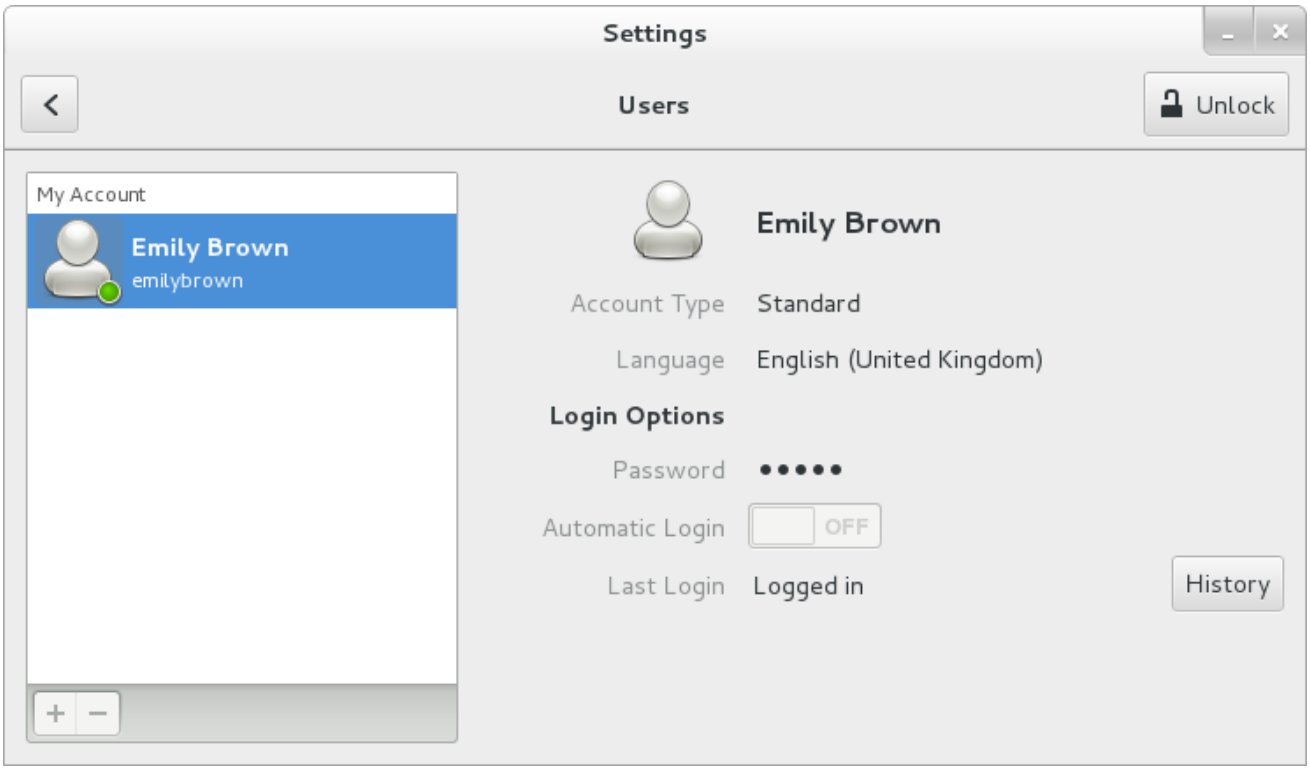
Users ユーティリティーは、グラフィカルユーザーインターフェイスで、ローカルユーザーを表示、編集、追加、削除できます。

4.2.1. ユーザー設定ツールの使用

Super キーを押してアクティビティーの概要に入り、**Users** と入力して **Enter** を押します。**ユーザー** 設定ツールが表示されます。**Super** キーはキーボードや他のハードウェアによって外見が異なりますが、通常はスペースバーの左側にある Windows キーまたは Command キーになります。また、画面の右上にある自分のユーザー名をクリックし、設定メニューから **ユーザー** ユーティリティーを開くこともできます。

ユーザーアカウントに変更を加えるには、最初に **Unlock (ロック解除)** ボタンを選択し、表示されたダイアログボックスに従って自身を認証します。スーパーユーザー特権がない場合は、**root** で認証するよう求められます。ユーザーを追加および削除するには、**+** ボタンと **-** ボタンをそれぞれクリックします。管理グループ **wheel** にユーザーを追加するには、アカウントの種類を **Standard** から **Administrator** に変更します。ユーザーの言語設定を編集するには、言語を選択します (ドロップダウンメニューが表示されます)。

図4.1 ユーザー設定ツール



新しいユーザーを作成しても、パスワードを設定するまでアカウントは無効になります。Password (パスワード) ドロップダウンメニュー (図4.2「パスワードメニュー」 を参照) には、管理者がすぐにパスワードを設定できるオプションが含まれます。最初のログイン時にユーザーがパスワードを選択するか、ログインパスワードなしでゲストアカウントを作成します。また、このメニューからアカウントを無効または有効にすることもできます。

図4.2 パスワードメニュー

4.3. コマンドラインツールの使用

「グラフィカル環境でのユーザーの管理」 で説明されている Users 設定ツール (ユーザーの基本的な管理用) のほかに、表4.1「ユーザーとグループを管理するためのコマンドラインユーティリティー」 に挙げられているユーザーとグループの管理コマンドラインツールを使用できます。

表4.1 ユーザーとグループを管理するためのコマンドラインユーティリティー

ユーティリティー	詳細
id	ユーザー ID およびグループ ID を表示します。
useradd、**usermod**、**userdel**	ユーザーアカウントを追加、修正、削除する標準ユーティリティーです。
groupadd、**groupmod**、**groupdel**	グループを追加、修正、削除する標準ユーティリティーです。
gpasswd	ユーティリティーは、主に、**newgrp** コマンドが使用する **/etc/gshadow** ファイルでグループパスワードの修正に使用されます。
pwck、**grpck**	パスワード、グループ、関連シャドウファイルを検証するユーティリティーです。

ユーティリティー	詳細
pwconv、**pwunconv**	通常のパスワードをシャドウパスワードに変換する、またはシャドウパスワードから通常のパスワードに変換するユーティリティーです。
grpconv、**grpunconv**	pwconv、pwunconv と同様、このユーティリティーは、グループアカウントのシャドウ化された情報を変換するのに使用できます。

4.3.1. 新規ユーザーの追加

システムにユーザーを追加するには、**root** で次のコマンドを実行します。

```
useradd options username
```

ここで、**options** は 表4.2「一般的な useradd コマンドラインオプション」 で説明されているコマンドラインオプションです。

デフォルトでは、**useradd** コマンドは、ロックされたユーザーアカウントを作成します。アカウントをアンロックするには、**root** で次のコマンドを実行して、パスワードを割り当てます。

```
passwd username
```

オプションで、パスワードエージングポリシーを設定できます。詳細は、**Red Hat Enterprise Linux 7 セキュリティーガイド** の Password Security セクションをご覧ください。

表4.2 一般的な useradd コマンドラインオプション

オプション	
-c 'comment'	comment にはどの文字列でも使用できます。このオプションは、通常、ユーザーの氏名を指定するのに使用されます。
-d home_directory	デフォルトの **/home/username/** の代わりに使用するホームディレクトリーです。
-e date	YYYY-MM-DD の形式でアカウントを無効にする日付です。
-f days	パスワードが失効してからアカウントが無効になるまでの日数です。**0** にすると、パスワードが失効した直後にアカウントが無効になります。**-1** にすると、パスワードが失効してもアカウントは無効になりません。

オプション	
-g group_name	ユーザーのデフォルト (プライマリー) グループ用のグループ名またはグループ番号です。グループは、ここで指定するよりも前に作成されている必要があります。
-G group_list	ユーザーがメンバーとなる追加 (補助、デフォルト以外のもの) のグループ名またはグループ番号のリストで、コンマで区切ります。グループは、ここで指定する前に作成しておく必要があります。
-m	ホームディレクトリーがない場合は、これを作成します。
-M	ホームディレクトリーを作成しません。
-N	ユーザー用のユーザープライベートグループを作成しません。
-p password	**crypt** を使用してパスワードを暗号化しました。
-r	UID が 1000 未満でホームディレクトリーがないシステムアカウントを作成します。
-s	ユーザーのログインシェルです。デフォルトは **/bin/bash** に設定されています。
-u uid	ユーザーのユーザー ID です。一意の番号で 999 より大きい数でなければなりません。

重要

Red Hat Enterprise Linux 7 では、システムユーザーおよび通常のユーザーのデフォルトの ID 範囲が変更になりました。以前はシステムユーザーに UID 1 ~ 499 が使用され、それよりも上の値が通常のユーザーに使用されていました。変更後は、システムユーザーのデフォルト範囲が 1 ~ 999 になりました。この変更により、既存のユーザーの UID と GID に 500 ~ 999 を使用している場合に Red Hat Enterprise Linux 7 に移行すると、問題が発生する場合があります。UID と GID のデフォルトの範囲は **/etc/login.defs** ファイルで変更できます。

プロセスの説明

以下の手順は、シャドウパスワードが有効なシステムで **useradd juan** コマンドを実行したときに発生する内容を解説したものです。

1. **juan** 用の新しい行が **/etc/passwd** に作成されます。

   ```
   juan:x:1001:1001::/home/juan:/bin/bash
   ```

   この行には以下の特徴があります。

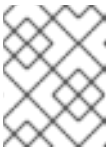
- ユーザー名 **juan** で始まります。
- パスワードフィールドには **x** が表示されます。これは、システムがシャドウパスワードを使用していることを示しています。
- 999 より大きい UID が作成されます。Red Hat Enterprise Linux 7 では、1000 未満の UID は、システムが使用するために予約されています。1000 未満の UID をユーザーに割り当てないでください。
- 999 より大きい GID が作成されます。Red Hat Enterprise Linux 7 では、1000 未満の GID は、システムが使用するために予約されています。1000 未満の GID はユーザーに割り当てないでください。
- オプションの **GECOS** 情報は空白のままになっています。GECOS フィールドは、氏名や電話番号などユーザーの追加情報を提供するために使用されます。
- **juan** のホームディレクトリーが、**/home/juan/** に設定されます。
- デフォルトシェルは **/bin/bash** に設定されます。

2. **juan** 用の新しい行が **/etc/shadow** に作成されます。

```
juan:!!:14798:0:99999:7:::
```

この行には以下の特徴があります。

- ユーザー名 **juan** で始まります。
- **/etc/shadow** ファイルのパスワードフィールドには 2 つの感嘆符 (**!!**) が表示され、アカウントがロックされていることを示しています。



注記

暗号化したパスワードを、**-p** フラグを使用して渡す場合は、そのユーザー用に、**/etc/shadow** ファイルに新しい行が追加されます。

- パスワードは有効期限なしで設定されています。

3. **juan** という名前のグループ用に、新しい行が **/etc/group** に作成されます。

```
juan:x:1001:
```

ユーザーと同じ名前のグループは、**ユーザープライベートグループ** と呼ばれます。ユーザープライベートグループの詳細は「ユーザープライベートグループ」を参照してください。

/etc/group に作成した行には、以下の特徴があります。

- グループ名 **juan** で始まります。
- パスワードフィールドには **x** が表示されます。これは、システムがシャドウグループパスワードを使用していることを示しています。
- GID は、**/etc/passwd** で設定されている **juan** のプライマリーグループに記載されているものと一致します。

4. **juan** という名前のグループ用の新しい行が **/etc/gshadow** に作成されました。

```
juan:!::
```

この行には以下の特徴があります。

- グループ名 **juan** で始まります。
- **/etc/gshadow** ファイルのパスワードフィールドには1つの感嘆符 (**!**) が表示され、グループがロックされていることを示しています。
- その他のフィールドはすべて空白です。

5. **/home** ディレクトリーに、ユーザー **juan** のディレクトリーが作成されます。

```
~]# ls -ld /home/juan
drwx------. 4 juan juan 4096 Mar 3 18:23 /home/juan
```

このディレクトリーは、ユーザー **juan** およびグループ **juan** が所有します。このディレクトリーの **読み取り**、**書き込み**、および **実行** の権限は、**juan** ユーザーに**のみ**割り当てられます。その他のパーミッションは拒否されます。

6. (デフォルトユーザー設定を含む) **/etc/skel/** ディレクトリーが、新しい **/home/juan/** ディレクトリーにコピーされます。

```
~]# ls -la /home/juan
total 28
drwx------. 4 juan juan 4096 Mar 3 18:23 .
drwxr-xr-x. 5 root root 4096 Mar 3 18:23 ..
-rw-r--r--. 1 juan juan  18 Jun 22 2010 .bash_logout
-rw-r--r--. 1 juan juan 176 Jun 22 2010 .bash_profile
-rw-r--r--. 1 juan juan 124 Jun 22 2010 .bashrc
drwxr-xr-x. 4 juan juan 4096 Nov 23 15:09 .mozilla
```

この時点では、**juan** という名前のロックされたアカウントがシステムに存在します。このアカウントをアクティブにするには、管理者が **passwd** コマンドを使用して、このアカウントにパスワードを割り当てる必要があります。オプションでパスワードエージングのガイドラインを設定することもできます (詳細は **Red Hat Enterprise Linux 7 セキュリティーガイド**の パスワードセキュリティー を参照)。

4.3.2. 新規グループの追加

システムに新しいグループを追加するには、**root** で次のコマンドを実行します。

```
groupadd options group_name
```

ここで、**options** は 表4.3「一般的な groupadd コマンドラインオプション」 で説明されているコマンドラインオプションです。

表4.3 一般的な groupadd コマンドラインオプション

オプション	詳細
-f, **--force**	**-g** gid を使用し、gid が存在していると、**groupadd** により、そのグループに別の一意の gid を選択します。

オプション	詳細
-g gid	グループのグループ ID です。一意の番号で 999 より大きい数でなければなりません。
-K, **--key** key=value	**/etc/login.defs** のデフォルトを上書きします。
-o、**--non-unique**	GID が重複するグループの作成を許可します。
-p, **--password** password	新規グループ用にこの暗号化されたパスワードを使用します。
-r	GID が 1000 未満のシステムグループを作成します。

4.3.3. 既存グループへの既存ユーザーの追加

usermod ユーティリティーを使用して、既存のユーザーを既存のグループに追加します。

usermod のさまざまなオプションは、ユーザーのプライマリーグループと補助グループにさまざまな影響を与えます。

ユーザーのプライマリーグループを上書きするには、**root** で以下のコマンドを実行します。

```
~]# usermod -g group_name user_name
```

ユーザーの補助グループを上書きするには、**root** で以下のコマンドを実行します。

```
~]# usermod -G group_name1,group_name2,... user_name
```

この場合、ユーザーの補助グループは、すべて新しいグループに置き換えられます。

ユーザーの補助グループにグループを追加するには、**root** で以下のコマンドのいずれかを実行します。

```
~]# usermod -aG group_name1,group_name2,... user_name
```

```
~]# usermod --append -G group_name1,group_name2,... user_name
```

この場合は、新しいグループが、ユーザーの現在の補助グループに追加されます。

4.3.4. グループディレクトリーの作成

システム管理者は、通常、主要なプロジェクトに対してそれぞれグループを作成し、そのプロジェクトのファイルにアクセスする必要がある場合に、そのユーザーをグループに割り当てる傾向があります。こうした従来型のスキームの場合は、誰かがファイルを作成すると、そのユーザーが属するプライマリーグループにそのファイルが関連付けられるため、ファイル管理は困難になります。このため、1 人のユーザーが複数のプロジェクトに関わっている場合に、正しいファイルを正しいグループに関連付けることは難しくなります。一方、UPG スキームを使用すると、グループは **setgid** ビットセットを持つディレクトリーに作成されたファイルに自動的に割り当てられます。**setgid** ビットにより、共通のディレクトリーを共有するグループプロジェクトを非常に簡単に管理できます。ユーザーがディレクトリー内で作成するすべてのファイルは、ディレクトリーを所有するグループが所有するためです。

たとえば、あるグループが **/opt/myproject/** ディレクトリーのファイルを作業する必要があるとします。グループの中には、このディレクトリーのコンテンツの修正を信頼して任せられる人もいますが、全員ではありません。

1. **root** で以下のコマンドを実行して、**/opt/myproject/** ディレクトリーを作成します。

```
mkdir /opt/myproject
```

2. **myproject** グループをシステムに追加します。

```
groupadd myproject
```

3. **/opt/myproject/** ディレクトリーのコンテンツを、**myproject** グループに関連付けます。

```
chown root:myproject /opt/myproject
```

4. グループのユーザーがそのディレクトリーにファイルを作成し、**setgid** ビットを設定できるようにします。

```
chmod 2775 /opt/myproject
```

この設定により、ユーザーがファイルを作成するたびに、管理者がファイルのパーミッションを変更しなくても、**myproject** グループの全メンバーが **/opt/myproject/** ディレクトリーにファイルを作成および編集できます。パーミッションが正しく設定されていることを確認するには、以下のコマンドを実行します。

```
~]# ls -ld /opt/myproject
drwxrwsr-x. 3 root myproject 4096 Mar 3 18:31 /opt/myproject
```

5. **myproject** グループにユーザーを追加します。

```
usermod -aG myproject username
```

4.3.5. umask を使用した、新規ファイルのデフォルト権限の設定

プロセスがファイルを作成すると、そのファイルにはデフォルト権限 (**-rw-rw-r--** など) が設定されます。こうした初期権限は、**ファイルモード作成マスク** (**ファイル権限マスク** または **umask** とも呼ばれる) で部分的に定義されます。たとえば、**bash** の **umask** は、デフォルトで **0022** となるなど、すべてのプロセスにそれぞれの umask が設定されています。プロセスの **umask** は変更できます。

umask を設定するもの
umask は、標準のファイル権限に対応するビットで設定されています。たとえば、**umask 0137** の場合、その数字は次のような意味になります。

- **0** = 意味なし。必ず **0** になります (umask は特別なビットに影響しません)。
- **1** = オーナーの権限。実行ビットが設定されます。
- **3** = グループの権限。実行ビットおよび書き込みビットが設定されます。
- **7** = その他のユーザーの権限。実行ビット、書き込みビット、読み取りビットが設定されます。

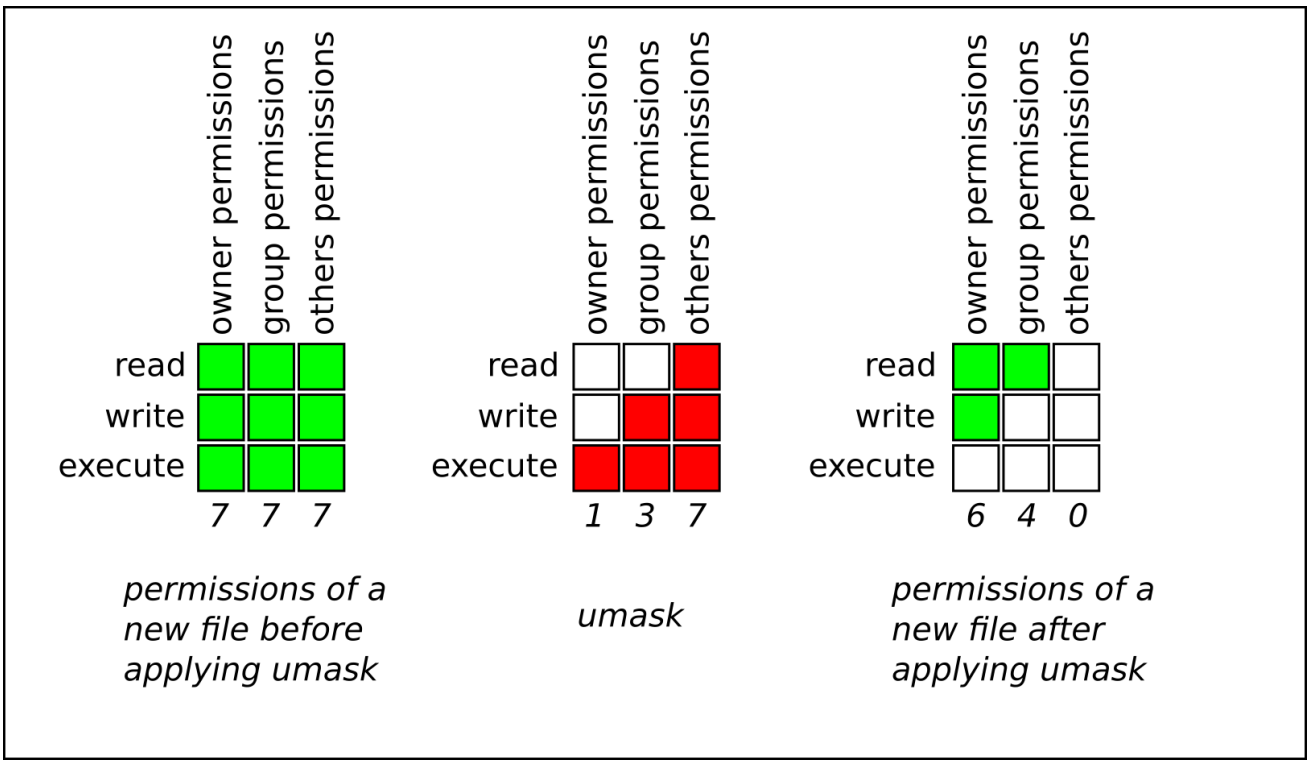
umask では 2 進法、8 進法、またはシンボリック表示が使用できます。たとえば、8 進法の **0137** はシンボリック表示では **u=rw-,g=r--,o=---** となります。シンボリック表示は 8 進法表示とは異なり、禁止権限ではなく、許可された権限を示します。

umask の仕組み
umask は、ファイルに **パーミッションを付与するのを禁止する** ようにします。

- umask に設定しているビットは、ファイルには設定されません。
- umask に設定していないビットは、他の要素にもよりますが、ファイルに設定されます。

以下の図は、**umask 0137** が新しいファイルの作成にどのように影響するかを示しています。

図4.3 ファイルの作成時に umask を適用



重要

セキュリティー上の理由から、レギュラーファイルにはデフォルトで実行権限が設定されていません。したがって、**umask** がいかなる権限も禁止しない **0000** であっても、新しいレギュラーファイルは実行権限を持ちません。ただし、ディレクトリーは実行権限を持つ状態で作成できます。

```
[john@server tmp]$ umask 0000
[john@server tmp]$ touch file
[john@server tmp]$ mkdir directory
[john@server tmp]$ ls -lh .
total 0
drwxrwxrwx. 2 john john 40 Nov 2 13:17 directory
-rw-rw-rw-. 1 john john 0 Nov 2 13:17 file
```

4.3.5.1. シェルで umask の管理

bash、**ksh**、**zsh**、**tcsh** などのよく使用されるシェルでは、**umask** は、**umask** シェルの **builtin** を使用して管理されます。シェルから起動したプロセスは、その umask を継承します。

現在のマスクの表示
現在の **umask** を 8 進法で表示するには、以下のコマンドを実行します。

```
~]$ umask
0022
```

現在の **umask** をシンボリック表示で表示するには、以下のコマンドを実行します。

```
~]$ umask -S
u=rwx,g=rx,o=rx
```

umask を使用したシェルにおけるマスクの設定
8 進法を使用して、現行シェルセッションに **umask** を設定するには、以下のコマンドを実行します。

```
~]$ umask octal_mask
```

octal_mask を、**0** から **7** の 4 桁以下の数値に置き換えます。3 桁以下の数値を指定すると、頭に 0 が付いた 4 桁の数値として権限が設定されます。たとえば、入力したコマンドが **umask 7** であれば、そのコマンドの数値は **0007** として解釈されます。

例4.1 8 進法を使用した umask の設定

新しいファイルで、オーナーとグループに書き込み権限と実行権限を持たせず、その他のユーザーにはいかなる権限も持たせないようにするには、以下を実行します。

```
~]$ umask 0337
```

もしくは、簡潔に次のコマンドを実行します。

```
~]$ umask 337
```

シンボリック表記法を使用して、現行シェルセッションの **umask** を設定するには、以下のコマンドを実行します。

```
~]$ umask -S symbolic_mask
```

例4.2 シンボリック表示を使用した umask の設定

シンボリック表記法を使用して **umask 0337** を設定するには、次のコマンドを実行します。

```
~]$ umask -S u=r,g=r,o=
```

デフォルトシェルの umask での作業
シェルには、通常、デフォルトの **umask** が設定されている設定ファイルがあります。**bash** の場合は **/etc/bashrc** です。デフォルトの **bash** の umask を表示するには、以下のコマンドを実行します。

```
~]$ grep -i -B 1 umask /etc/bashrc
```

出力では、**umask** の設定が、**umask** コマンドまたは **UMASK** 変数のいずれかを使用して行われていることが示されます。以下の例では、**umask** コマンドを使用して、**umask** が **022** に設定されています。

```
~]$ grep -i -B 1 umask /etc/bashrc
  # By default, we want umask to get set. This sets it for non-login shell.
--
  if [ $UID -gt 199 ] && [ “id -gn” = “id -un” ]; then
    umask 002
  else
    umask 022
```

bash のデフォルト umask を変更するには、**/etc/bashrc** で **umask** コマンドの呼び出し、または **UMASK** 変数の割り当てを変更します。この例では、デフォルトの umask を **0227** に変更します。

```
  if [ $UID -gt 199 ] && [ “id -gn” = “id -un” ]; then
    umask 002
  else
    umask 227
```

特定ユーザーのデフォルトシェルの umask での作業
デフォルトでは、新規ユーザーのデフォルトの **bash** の umask は、**/etc/bashrc** に定義したものに設定されます。

特定ユーザーの **bash** umask を変更するには、そのユーザーの **$HOME/.bashrc** ファイルで、**umask** コマンドの呼び出しを追加します。たとえば、ユーザー **john** の **bash** の umask を **0227** に変更するには、次のコマンドを実行します。

```
john@server ~]$ echo 'umask 227' >> /home/john/.bashrc
```

新しく作成されたホームディレクトリーのデフォルト権限設定
作成したユーザーホームディレクトリーの権限を変更するには、**/etc/login.defs** ファイルで **UMASK** 変数を変更します。

```
# The permission mask is initialized to this value. If not specified,
# the permission mask will be initialized to 022.
UMASK 077
```

4.4. 関連情報

Red Hat Enterprise Linux でユーザーとグループを管理する方法は、下記の資料を参照してください。

インストールされているドキュメント
ユーザーおよびグループの管理に使用する各種ユーティリティーの詳細情報は、以下の man ページを参照してください。

- **useradd**(8): **useradd** コマンドの man ページでは、新しいユーザーを作成する方法が説明されています。
- **userdel**(8): **userdel** コマンドの man ページでは、ユーザーを削除する方法が説明されています。
- **usermod**(8): **usermod** コマンドの man ページでは、ユーザーを変更する方法が説明されています。
- **groupadd**(8): **groupadd** コマンドの man ページでは、新しいグループを作成する方法が説明されています。

- **groupdel**(8): **groupdel** コマンドの man ページでは、グループを削除する方法が説明されています。
- **groupmod**(8): **groupmod** コマンドの man ページでは、グループのメンバーシップを修正する方法が説明されています。
- **gpasswd**(1): **gpasswd** コマンドの man ページでは、**/etc/group** ファイルを管理する方法が説明されています。
- **grpck**(8): **grpck** コマンドの man ページでは、**/etc/group** ファイルの統合を確認する方法が説明されています。
- **pwck**(8): **pwck** コマンドの man ページでは、**/etc/passwd** ファイルおよび **/etc/shadow** ファイルの統合を確認する方法が説明されています。
- **pwconv**(8): **pwconv**、**pwunconv**、**grpconv**、および **grpunconv** の各コマンドの man ページでは、パスワードおよびグループ用にシャドウ情報を変換する方法が説明されています。
- **id**(1): **id** コマンドの man ページでは、ユーザー ID およびグループ ID を表示する方法が説明されています。
- **umask**(2): **umask** コマンドの man ページでは、ファイルモード作成マスクの使用方法が説明されています。

関連する設定ファイルの詳細は、以下をご覧ください。

- **group**(5): **/etc/group** ファイルの man ページでは、システムグループを定義する方法が説明されています。
- **passwd**(5): **/etc/passwd** ファイルの man ページでは、このファイルを使用して、ユーザー情報を定義する方法が説明されています。
- **shadow**(5): **/etc/shadow** ファイルの man ページでは、システムでパスワードおよびアカウントの有効期限情報を設定する方法が説明されています。

オンラインドキュメント

- Red Hat Enterprise Linux 7 セキュリティーガイド : Red Hat Enterprise Linux 7 の **セキュリティーガイド** では、パスワードのエージングとユーザーアカウントのロックを有効にして、パスワードとワークステーションのセキュリティーを高める追加情報を提供しています。

関連項目

- *6章 権限の取得* では、**su** および **sudo** コマンドを使用して管理者権限を取得する方法を説明しています。

第5章 アクセス制御リスト

ファイルとディレクトリーには、ファイルの所有者、そのファイルに関連したグループ、およびシステムを使用する他のすべてのユーザーの権限セットが設定されます。しかし、これらの権限には制限があります。たとえば、ユーザーごとに異なる権限を設定することはできません。そのため **アクセス制御リスト** (ACL) が実装されています。

Red Hat Enterprise Linux カーネルは、ext3 ファイルシステムと NFS でエクスポートしたファイルシステムに対して ACL サポートを提供します。ACL は、Samba 経由でアクセスする ext3 ファイルシステムでも認識されます。

ACL の実装には、カーネルでのサポートと **acl** パッケージが必要になります。このパッケージには、ACL 情報の追加、修正、削除および、取得のためのユーティリティーが同梱されています。

cp コマンドと **mv** コマンドは、ファイルとディレクトリーに関連するすべての ACL のコピーまたは移動を実行します。

5.1. ファイルシステムのマウント

ファイルやディレクトリー用に ACL を使用する前に、そのファイルまたはディレクトリーのパーティションを ACL サポートでマウントする必要があります。ローカルの ext3 ファイルシステムの場合は、以下のコマンドでマウントできます。

mount -t ext3 -o acl device-name partition

以下に例を示します。

mount -t ext3 -o acl /dev/VolGroup00/LogVol02 /work

もしくは、パーティションが **/etc/fstab** ファイルにリストされている場合は、パーティションのエントリーに **acl** オプションを含むことができます。

```
LABEL=/work   /work    ext3  acl    1 2
```

Samba 経由で ext3 ファイルシステムにアクセスし、そのアクセスに対して ACL が有効になっている場合は、ACL が認識されます。これは、**--with-acl-support** オプションでコンパイルされているためです。Samba 共有のアクセス時またはマウント時に特別なフラグは必要ありません。

5.1.1. NFS

デフォルトでは、NFS サーバーでエクスポートされているファイルシステムが ACL をサポートし、NFS クライアントが ACL を読み込める場合は、クライアントシステムで ACL が使用されています。

サーバーを設定する際に NFS 共有上の ACL を無効にするには、**/etc/exports** ファイルに **no_acl** オプションを追加します。クライアントに NFS 共有をマウントする際に ACL を無効にするには、コマンドライン経由、または **/etc/fstab** ファイルに **no_acl** オプションを追加してマウントします。

5.2. アクセス ACL の設定

ACL には、**アクセス ACL** と **デフォルト ACL** と 2 つのタイプがあります。アクセス ACL は、特定のファイルまたはディレクトリーに対するアクセス制御リストです。デフォルト ACL は、ディレクトリーにのみ適用されます。ディレクトリー内のファイルにアクセス ACL が設定されていない場合は、そのディレクトリーにデフォルト ACL のルールが適用されます。デフォルト ACL は任意です。

ACL は以下のように設定できます。

1. 各ユーザー
2. 各グループ
3. 実効権マスクを使用して
4. ファイルのユーザーグループに属さないユーザーに対して

setfacl ユーティリティーは、ファイルとディレクトリー用の ACL を設定します。**-m** オプションを使用すると、ファイルまたはディレクトリーの ACL を追加または修正できます。

```
# setfacl -m rules files
```

ルール (**rules**) は、以下の形式で指定する必要があります。複数のルールをコンマで区切って同じコマンドに指定することもできます。

u:uid:perms
ユーザーにアクセス ACL を設定します。ユーザー名または UID を指定できます。システムで有効な任意のユーザーを指定できます。

g:gid:perms
グループにアクセス ACL を設定します。グループ名または GID を指定できます。システムで有効な任意のグループを指定できます。

m:perms
実効権マスクを設定します。このマスクは、所有グループの全権限と、ユーザーおよびグループの全エントリーを結合したものです。

o:perms
ファイルのグループに属さないユーザーにアクセス ACL を設定します。

権限 (**perms**) は、読み取り、書き込み、および実行を表す **r**、**w**、および **x** の文字の組み合わせで表示されます。

ファイルまたはディレクトリーにすでに ACL が設定されている状態で、**setfacl** コマンドを使用した場合は、設定するルールが既存の ACL に追加されるか、既存のルールが修正されます。

例5.1 読み取りと書き込みの権限付与

たとえば、ユーザー andrius に読み取りと書き込みの権限を付与するには以下を実行します。

```
# setfacl -m u:andrius:rw /project/somefile
```

ユーザー、グループ、またはその他のユーザーからすべての権限を削除するには、**-x** オプションにいずれの権限も指定せずにコマンドを実行します。

```
# setfacl -x rules files
```

例5.2 すべての権限の削除

たとえば、UID 500 のユーザーからすべての権限を削除するには以下を実行します。

```
# setfacl -x u:500 /project/somefile
```

5.3. デフォルト ACL の設定

デフォルト ACL を設定するには、**d:** をルールの前に追加してから、ファイル名ではなくディレクトリー名を指定します。

例5.3 デフォルト ACL の設定

たとえば、**/share/** ディレクトリーにデフォルト ACL を設定し、ユーザーグループに属さないユーザーの読み取りと実行を設定するには、以下のコマンドを実行します (これにより、個別ファイルのアクセス ACL が上書きされます)。

```
# setfacl -m d:o:rx /share
```

5.4. ACL の取り込み

ファイルまたはディレクトリーに設定されている ACL を確認するには、**getfacl** コマンドを使用します。以下の例では、**getfacl** でファイルの既存 ACL を確認します。

例5.4 ACL の取り込み

```
# getfacl home/john/picture.png
```

上記のコマンドは、次のような出力を返します。

```
# file: home/john/picture.png
# owner: john
# group: john
user::rw-
group::r--
other::r--
```

ディレクトリーにデフォルト ACL が指定されている場合は、以下のようにデフォルト ACL も表示されます。たとえば、**getfacl home/sales/** を実行すると以下のような出力になります。

```
# file: home/sales/
# owner: john
# group: john
user::rw-
user:barryg:r--
group::r--
mask::r--
other::r--
default:user::rwx
default:user:john:rwx
default:group::r-x
default:mask::rwx
default:other::r-x
```

■

5.5. ACL が設定されているファイルシステムのアーカイブ作成

デフォルトでは、**dump** コマンドによるバックアップ操作時に ACL が保存されます。**tar** コマンドで、ファイルまたはファイルシステムのアーカイブを作成する場合は、**--acls** オプションを付けて ACL を保存します。同様に、**cp** コマンドで、ACL が設定されているファイルをコピーする場合は、**--preserve=mode** オプションを付けて ACL もコピーされるようにします。さらに、**cp** の **-a** オプション (**-dR --preserve=all** と同等) も、バックアップ時にタイムスタンプ、SELinux コンテキストなどの情報と一緒に ACL を保存します。**dump**、**tar**、または **cp** の詳細は、それぞれの **man** ページを参照してください。

star ユーティリティーは、ファイルのアーカイブ生成に使用される点で **tar** ユーティリティーと似ています。しかし、一部のオプションは異なります。最も一般的に使用されるオプションのリストは 表5.1「**star** のコマンドラインオプション」 を参照してください。すべての利用可能なオプションは、**man star** を参照してください。このユーティリティーを使用するには **star** パッケージが必要になります。

表5.1 star のコマンドラインオプション

オプション	説明
-c	アーカイブファイルを作成します。
-n	ファイルを抽出しません。**-x** と併用すると、ファイルが行う抽出を表示します。
-r	アーカイブ内のファイルを入れ替えます。パスとファイル名が同じファイルが置き換えられ、アーカイブファイルの末尾に書き込まれます。
-t	アーカイブファイルのコンテンツを表示します。
-u	アーカイブファイルを更新します。アーカイブにファイルが存在しない場合や、アーカイブ内にある同名のファイルよりも新しい場合は、そのファイルがアーカイブの末尾に書き込まれます。このオプションは、アーカイブがファイルか、バックスペース可能な非ブロックテープの場合にのみ機能します。
-x	アーカイブからファイルを抽出します。**-U** と併用すると、アーカイブ内のファイルがファイルシステムにあるファイルよりも古い場合、そのファイルは抽出されません。
-help	最も重要なオプションを表示します。
-xhelp	最も重要ではないオプションを表示します。

オプション	説明
-/	アーカイブからファイルを抽出する際に、ファイル名から先頭のスラッシュを削除します。デフォルトでは、ファイルの抽出時に先頭のスラッシュが削除されます。
-acl	作成時または抽出時に、ファイルとディレクトリーに関連付けられているすべての ACL をアーカイブするか、復元します。

5.6. 旧システムとの互換性

指定したファイルシステムのいずかのファイルに ACL が設定されている場合、そのファイルシステムは **ext_attr** 属性を持ちます。この属性は、以下のコマンドを使用すると確認できます。

```
# tune2fs -l filesystem-device
```

ext_attr 属性を持つファイルシステムは古いカーネルでマウントできますが、それらのカーネルは設定されている ACL を強制しません。

バージョン 1.22 以降の **e2fsprogs** パッケージ (Red Hat Enterprise Linux 2.1 および 4 のバージョンも含む) に含まれている **e2fsck** ユーティリティーのバージョンは、**ext_attr** 属性を使用してファイルシステムを確認できます。古いバージョンではこの確認が拒否されます。

5.7. ACL 参照情報

詳細情報は以下の man ページを参照してください。

- **man acl**: ACL の説明
- **man getfacl**: ファイルアクセス制御リストの取得方法
- **man setfacl**: ファイルアクセス制御リストの設定方法
- **man star**: **star** ユーティリティーとそのオプションの詳細説明

第6章 権限の取得

システム管理者は (時にはユーザーも)、管理者アクセスでタスクを実行する必要があります。システムに **root** ユーザーでアクセスすることは危険を伴う可能性があり、システムおよびデータの著しい破損につながる場合もあります。本章では、**su** や **sudo** といった **setuid** プログラムを使用して管理者権限を取得する方法を説明します。これらのプログラムを使用すると、高レベルの制御およびシステムセキュリティーを維持しつつ、通常は **root** ユーザーしかできないタスクを特定のユーザーが実行できます。

管理者制御や潜在的な危険、特権アクセスの不適切な使用によるデータ破損を回避する方法は **Red Hat Enterprise Linux 7 セキュリティーガイド** を参照してください。

6.1. SU ユーティリティーを使用した管理アクセスの設定

ユーザーは、**su** を実行すると **root** パスワードを求められます。認証されると **root** シェルプロンプトが表示されます。

su コマンドでログインすると、そのユーザーは **root ユーザーとなり**、システムへの絶対管理アクセスを持つことになります。このアクセスが有効になっている場合は、SELinux によって課される制限が適用されることに注意してください。また、ユーザーが **root** になったら、パスワードを求められることなく、**su** コマンドを使用してシステム上の他のユーザーに変更を加えることができます。

このプログラムは非常に強力なので、組織内の管理者はこのコマンドにアクセスできるユーザーを制限してください。

簡単な制限方法は、**wheel** と呼ばれる特別な管理グループにユーザーを追加することです。これを実行するには、**root** で以下のコマンドを実行します。

```
~]# usermod -a -G wheel username
```

このコマンドで、username を、**wheel** グループに追加するユーザー名に置き換えます。

また、Users 設定ツールを使用して以下のようにグループのメンバーを修正することもできます。この手順を実行するには、管理者権限が必要なことに注意してください。

1. **Super** キーを押してアクティビティーの概要に入り、**Users** と入力して **Enter** を押します。**ユーザー** 設定ツールが表示されます。**Super** キーはキーボードまたはその他のハードウェアに応じて様々なキーで表示されますが、多くの場合、Windows または Command キーとして通常は **Spacebar** の左側に表示されます。

2. 変更を有効にするには、**ロック解除** ボタンをクリックし、有効な管理者パスワードを入力します。

3. 左側の列でユーザーアイコンをクリックし、右側のペインでユーザーのプロパティーを表示します。

4. アカウントの種類を **Standard** から **Administrator** に変更します。これにより、ユーザーが **wheel** グループに追加されます。

Users ツールの詳細は 「グラフィカル環境でのユーザーの管理」 を参照してください。

wheel グループにユーザーを追加したら、この追加した特定のユーザーにのみ **su** コマンドの使用を許可することが推奨されます。それには、su の PAM **(プラグ可能な認証モジュール)** 設定ファイル (**/etc/pam.d/su**) を編集する必要があります。このファイルをテキストエディターで開き、以下の行の **#** 文字を削除してコメント設定を解除します。

```
#auth      required    pam_wheel.so use_uid
```

この変更で、**wheel** の管理グループメンバーのみが、**su** コマンドを使用して別のユーザーに切り換えることができるようになります。

6.2. SUDO ユーティリティーを使用した管理アクセスを設定

ユーザーに管理アクセスを付与する別のアプローチとして **sudo** コマンドを利用できます。信頼されるユーザーが、管理コマンドの前に **sudo** を付けると、**このユーザー自身の** パスワードが要求されます。ユーザーが認証され、コマンドが許可されると、管理コマンドは **root** 権限で実行されます。

sudo コマンドの基本的なフォーマットは、以下のとおりです。

```
sudo command
```

上記の例の command の部分を、通常は **root** ユーザーのみが使用する **mount** といったコマンドに置き換えます。

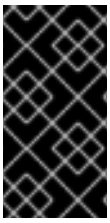
sudo コマンドでは、ハイレベルの柔軟性が可能になります。たとえば、**/etc/sudoers** 設定ファイルに記載されているユーザーのみが **sudo** コマンドを使うことができ、root シェルではなく、**そのユーザーの** シェルでコマンドが実行されます。これは、**Red Hat Enterprise Linux 7 セキュリティーガイド**に示されるように、**root** シェルを完全に無効にできることを意味します。

sudo コマンドを使用した正常な認証のログはすべて **/var/log/messages** ファイルに記録され、このコマンドを実行したユーザー名で実行されたコマンドは **/var/log/secure** ファイルに記録されます。新たなログが必要な場合は、以下の行を **/etc/pam.d/system-auth** ファイルに追加して、**pam_tty_audit** モジュールで特定ユーザーの TTY 監査を有効にします。

```
session required pam_tty_audit.so disable=pattern enable=pattern
```

pattern で表示されるのはコンマで区切ったユーザーのリストで、オプションでグロブを使用できます。たとえば、以下の設定は、**root** ユーザーの TTY 監査を有効にし、その他のユーザーについては無効にします。

```
session required pam_tty_audit.so disable=* enable=root
```



重要

TTY の監査システムの **pam_tty_audit** PAM モジュールを設定すると、TTY 入力のみが記録されます。つまり、監査されるユーザーがログインすると、**pam_tty_audit** には、**/var/log/audit/audit.log** ファイルに記録されるキーストロークと同じ内容が記録されます。詳細は、man ページの pam_tty_audit(8) を参照してください。

sudo コマンドのもう一つの利点は、各ユーザーのニーズに応じて特定のコマンドへのアクセスを管理者が許可できることです。

管理者が **sudo** 設定ファイルである **/etc/sudoers** を編集する場合は、**visudo** コマンドを使用することが推奨されます。

他のユーザーに完全な管理権限を付与する場合は、**visudo** と入力し、ユーザー権限の指定セクションに以下の行を追加します。

```
juan ALL=(ALL) ALL
```

この例では、ユーザーの **juan** は、**sudo** を使用すればどのホストからでもどのコマンドを実行できることを示しています。

以下の例では、**sudo** を設定する際に可能な粒度を示しています。

```
%users localhost=/usr/sbin/shutdown -h now
```

この例が示しているのは、コンソールからであれば、 **users** システムグループのどのユーザーでも、**/sbin/shutdown -h now** コマンドを実行できるということです。

sudoers の man ページには、このファイルのオプションの詳細なリストが記載されています。

/etc/sudoers ファイルで **NOPASSWD** オプションを指定して、パスワードを指定する必要がない sudo ユーザーを設定することもできます。

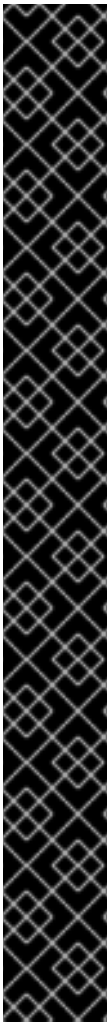
```
user_name ALL=(ALL) NOPASSWD: ALL
```

ただし、このようなユーザーであっても、**sudo** は PAM (Pluggable Authentication Module) アカウント管理モジュールを実行します。これにより、認証フェーズ外で PAM モジュールに課せられた制限を確認できるようになります。これにより、PAM モジュールが正しく動作するようになります。たとえば、**pam_time** モジュールの場合、時間ベースのアカウント制限は失敗しません。

警告

PAM ベースのすべてのアクセス制御ルールで、**sudo** を、許可されるサービスのリストに常に含めるようにしてください。そうしないと、ユーザーが **sudo** にアクセスしようとしたときに permission denied エラーメッセージが表示されますが、現行のアクセス制御ルールに基づいてアクセスが禁止されます。

詳細は、Red Hat ナレッジベースの記事 After patching to Red Hat Enterprise Linux 7.6 sudo gives a permission denied error. を参照してください。



重要

sudo コマンドの使用時には、潜在的なリスクがいくつか存在することを覚えておく必要があります。このリスクは、上記のように **visudo** を使用して **/etc/sudoers** 設定ファイルを編集することで回避できます。**/etc/sudoers** ファイルをデフォルトの状態にしておくと、**wheel** グループのユーザー全員に無制限の **root** アクセスを与えることになります。

- **sudo** は、デフォルトで 5 分間、パスワードを保存します。この間はコマンドを続けて使用しても、ユーザーはパスワードを要求されません。このため、ユーザーがログイン状態のままワークステーションを離れたりロックしない状態にしておくと、攻撃者に悪用されかねません。この動作は、以下の行を **/etc/sudoers** ファイルに追加することで変更できます。

  ```
  Defaults  timestamp_timeout=value
  ```

  value には、指定するタイムアウトの分数を入れます。**value** を 0 にすると **sudo** は毎回パスワードを要求します。

- sudo 使用者のアカウントが侵害されると、攻撃者は **sudo** を使用して管理権限のある新たなシェルを開くことができます。

  ```
  sudo /bin/bash
  ```

  この方法や同様の方法で **root** として新たなシェルを開くと、**/etc/sudoers** ファイルで指定されたタイムアウト時間を無視し、新たに開かれたセッションが閉じられるまで攻撃者に **sudo** パスワード入力を要求することがないため、理論上は時間の制限なく攻撃者に管理アクセスを与えることになります。

6.3. 関連情報

ユーザーに管理者権限を与えるプログラムは潜在的なセキュリティーリスクでありますが、セキュリティーの説明は本ガイドの対象外となります。セキュリティーや管理者アクセスに関する情報は、以下に挙げる資料を参照してください。

インストールされているドキュメント

- **su**(1): **su** の man ページには、このコマンドで利用可能なオプションの情報があります。
- **sudo**(8): **sudo** の man ページには、このコマンドの動作のカスタマイズで利用可能なオプションのリストがあります。
- **pam**(8): この man ページでは、Linux 向け Pluggable Authentication Modules (PAM) の使用方法が説明されています。

オンラインドキュメント

- Red Hat Enterprise Linux 7 セキュリティーガイド : Red Hat Enterprise Linux 7 の **セキュリティーガイド** では、**setuid** プログラムに関する潜在的なセキュリティー問題の詳細と、そのリスクを低減するテクニックを紹介します。

関連項目

- *4章 ユーザーとグループの管理* では、グラフィカルユーザーインターフェイスとコマンドラインを使用したシステムユーザーとグループの管理方法を説明します。

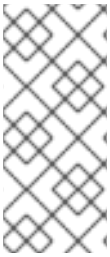
パート II. サブスクリプションおよびサポート

Red Hat Enterprise Linux システムのソフトウェアへの更新を受け取るには、**Red Hat Content Delivery Network** (**CDN**)、および有効で適切なリポジトリーをサブスクライブしている必要があります。ここでは、システムを Red Hat Content Delivery Network にサブスクライブする方法を説明します。

Red Hat は カスタマーポータル からサポートを提供していますが、このサポートには、**Red Hat Support Tool** を使用してコマンドラインから直接アクセスできます。ここでは、そのコマンドラインツールの使用方法を説明します。

第7章 システム登録およびサブスクリプション管理

サブスクリプションサービスは、Red Hat ソフトウェアインベントリーを処理するメカニズムを提供し、**yum** パッケージマネージャーを使用して追加のソフトウェアをインストールしたり、インストールされているプログラムを新規バージョンに更新したりすることを可能にします。Red Hat Enterprise Linux 7 で、システムを登録し、サブスクリプションを割り当てる方法としては **Red Hat サブスクリプション管理** の使用が推奨されます。



注記

さらに、初期設定プロセスでのインストール後にシステムを登録し、サブスクリプションを割り当てることもできます。初期設定の詳細情報は、Red Hat Enterprise Linux 7 の インストールガイド の 初期設定と初期起動 の章を参照してください。初期設定アプリケーションは、インストール時に X Window System でインストールしたシステムでのみ利用できることに注意してください。

7.1. システム登録およびサブスクリプションの割り当て

Red Hat Subscription Management を使用してシステムを登録し、1 つ以上のサブスクリプションを割り当てる手順を完了してください。**subscription-manager** コマンドはすべて **root** で実行することに注意してください。

1. 以下のコマンドを実行してシステムを登録します。ユーザー名とパスワードを入力するように求められます。ユーザー名とパスワードは、Red Hat カスタマーポータルのログイン認証情報と同じであることに注意してください。

   ```
   subscription-manager register
   ```

2. 必要なサブスクリプションのプール ID を確認します。これを行うには、シェルプロンプトで以下のコマンドを入力し、システムで利用できるサブスクリプションのリストを表示します。

   ```
   subscription-manager list --available
   ```

   このコマンドは、利用可能な各サブスクリプションの名前、固有 ID、有効期限、およびそのサブスクリプションに関連するその他の詳細情報を表示します。全アーキテクチャー向けのサブスクリプションをリスト表示するには、**--all** オプションを追加します。プール ID は、**Pool ID** で始まる行にリスト表示されます。

3. 以下のコマンドを実行して、該当するサブスクリプションをシステムに割り当てます。

   ```
   subscription-manager attach --pool=pool_id
   ```

   pool_id を、直前のステップで確認したプール ID に置き換えます。

   システムに割り当てているサブスクリプションのリストを随時確認するには、以下を実行します。

   ```
   subscription-manager list --consumed
   ```

Red Hat Subscription Management を使用してシステムを登録し、サブスクリプションに関連付ける方法は、Red Hat Subscription-Manager を使用して Red Hat カスタマーポータルにシステムを登録してサブスクライブする を参照してください。サブスクリプションに関する包括的な情報は Red Hat Subscription Management のガイドを参照してください。

7.2. ソフトウェアリポジトリーの管理

Red Hat コンテンツ配信ネットワークにシステムをサブスクライブすると、**/etc/yum.repos.d/** ディレクトリーにリポジトリーファイルが作成されます。これを確認するには、**yum** を使用して有効にしたリポジトリーのリストを表示します。

```
yum repolist
```

Red Hat Subscription Management を使用すると、Red Hat が提供するソフトウェアリポジトリーを手動で有効にしたり、無効にしたりすることもできます。利用可能なリポジトリーのリストを表示するには、以下のコマンドを実行します。

```
subscription-manager repos --list
```

リポジトリー名は、使用している Red Hat Enterprise Linux のバージョンによって異なり、以下のフォーマットに基づいています。

```
rhel-version-variant-rpms
rhel-version-variant-debug-rpms
rhel-version-variant-source-rpms
```

ここで、**version** は Red Hat Enterprise Linux システムのバージョン (**6** または **7**) を示し、**variant** は Red Hat Enterprise Linux システムのバリアント (**server** または **workstation**) を示します。以下は例になります。

```
rhel-7-server-rpms
rhel-7-server-debug-rpms
rhel-7-server-source-rpms
```

リポジトリーを有効にするには、以下のコマンドを入力します。

```
subscription-manager repos --enable repository
```

repository を、有効にするリポジトリーの名前に置き換えます。

同様に、リポジトリーを無効にするには以下のコマンドを使用します。

```
subscription-manager repos --disable repository
```

「Yum と Yum リポジトリーの設定」 では、**yum** を使用したソフトウェアリポジトリー管理の詳細情報を説明します。

リポジトリーの更新を自動的にダウンロードするには、**yum-cron** サービスを使用できます。詳細は、「yum-cron を使用したパッケージデータベースの自動更新および更新のダウンロード」 を参照してください。

7.3. サブスクリプションの削除

特定のサブスクリプションを削除するには、以下の手順を行います。

1. すでに割り当てられているサブスクリプションの情報をリスト表示し、削除する必要があるサブスクリプションのシリアル番号を確認します。

```
subscription-manager list --consumed
```

シリアル番号は、**serial** に記載されている番号です。たとえば、以下の例では **744993814251016831** になります。

```
SKU:        ES0113909
Contract:     01234567
Account:      1234567
Serial:      744993814251016831
Pool ID:      8a85f9894bba16dc014bccdd905a5e23
Active:      False
Quantity Used:   1
Service Level:   SELF-SUPPORT
Service Type:   L1-L3
Status Details:
Subscription Type: Standard
Starts:      02/27/2015
Ends:        02/27/2016
System Type:     Virtual
```

2. 以下のコマンドを実行して、選択したサブスクリプションを削除します。

```
subscription-manager remove --serial=serial_number
```

serial_number を、直前のステップで確認したシリアル番号に置き換えます。

システムに割り当てられているすべてのサブスクリプションを削除するには、以下のコマンドを実行します。

```
subscription-manager remove --all
```

7.4. 関連情報

Red Hat Subscription Management を使用してシステムを登録し、サブスクリプションに関連付ける方法は、以下の資料を参照してください。

インストールされているドキュメント

- **subscription-manager**(8): Red Hat Subscription Management の man ページは、サポートされているオプションおよびコマンドの完全リストを提供します。

関連書籍

- Red Hat Subscription Management の一連のガイド: これらのガイドには、Red Hat Subscription Management の使用方法に関する詳細情報が記載されています。
- インストールガイド: 初期設定プロセス中に登録する詳細な手順は、初期設定 の章を参照してください。

関連項目

- 6章 *権限の取得* では、**su** および **sudo** コマンドを使用して管理者権限を取得する方法を説明しています。

- 9章*Yum* では、**yum** パッケージマネージャーを使用してソフトウェアをインストールし、更新する方法を提供しています。

第8章 RED HAT SUPPORT TOOL を使用したサポートへのアクセス

redhat-support-tool パッケージの **Red Hat Support Tool**はインタラクティブシェルおよび単一実行プログラムとして機能します。**SSH** または任意のターミナルで実行できます。また、コマンドラインから Red Hat ナレッジベースを検索したり、コマンドラインでソリューションを直接コピーしたり、サポートケースを作成または更新したり、分析のために Red Hat にファイルを送信したりできます。

8.1. RED HAT SUPPORT TOOL のインストール

Red Hat Support Toolはデフォルトで Red Hat Enterprise Linux にインストールされます。必要な場合は、確実にインストールするために **root** で以下のコマンドを入力します。

```
~]# yum install redhat-support-tool
```

8.2. コマンドラインを使用した RED HAT SUPPORT TOOL の登録

コマンドラインを使用して Red Hat Support Tool をカスタマーポータルに登録するには、以下のコマンドを実行します。

```
~]# redhat-support-tool config user username
```

username は、Red Hat カスタマーポータルアカウントのユーザー名に置き換えます。

```
~]# redhat-support-tool config password
Please enter the password for username:
```

8.3. インタラクティブシェルモードでの RED HAT SUPPORT TOOL の使用

インタラクティブモードでツールを起動するには、以下のコマンドを入力します。

```
~]$ redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help):
```

ツールは、非特権ユーザーまたは **root** で実行できます。非特権ユーザーの場合は使用できるコマンドが少なくなります、

? 文字を入力するとコマンドのリストを表示できます。プログラムまたはメニューの選択は、**q** または **e** の文字を入力して終了できます。ナレッジベースまたはサポートケースを初めて検索する場合は、Red Hat カスタマーポータルのユーザー名とパスワードを入力するよう求められます。また、インタラクティブモードで Red Hat カスタマーポータルアカウントのユーザー名とパスワードを設定し、オプションで設定ファイルに保存することもできます。

8.4. RED HAT SUPPORT TOOL の設定

インタラクティブモードの場合は、コマンド **config --help** を入力して設定オプションのリストを表示できます。

```
~]# redhat-support-tool
Welcome to the Red Hat Support Tool.
```

```
Command (? for help): config --help

Usage: config [options] config.option <new option value>

Use the 'config' command to set or get configuration file values.
Options:
 -h, --help  show this help message and exit
 -g, --global Save configuration option in /etc/redhat-support-tool.conf.
 -u, --unset  Unset configuration option.

The configuration file options which can be set are:
 user   : The Red Hat Customer Portal user.
 password : The Red Hat Customer Portal password.
 debug   : CRITICAL, ERROR, WARNING, INFO, or DEBUG
 url    : The support services URL. Default=https://api.access.redhat.com
 proxy_url : A proxy server URL.
 proxy_user: A proxy server user.
 proxy_password: A password for the proxy server user.
 ssl_ca  : Path to certificate authorities to trust during communication.
 kern_debug_dir: Path to the directory where kernel debug symbols should be downloaded and
cached. Default=/var/lib/redhat-support-tool/debugkernels

Examples:
- config user
- config user my-rhn-username
- config --unset user
```

インタラクティブモードでの Red Hat Support Tool の登録

インタラクティブモードを使用して Red Hat Support Tool をカスタマーポータルに登録するには、以下のコマンドを実行します。

1. 以下のコマンドを入力してツールを起動します。

```
~]# redhat-support-tool
```

2. Red Hat カスタマーポータルのユーザー名を入力します。

```
Command (? for help): config user username
```

ユーザー名をグローバル設定ファイルに保存するには、**-g** オプションを追加します。

3. Red Hat カスタマーポータルのパスワードを入力します。

```
Command (? for help): config password
Please enter the password for username:
```

8.4.1. 設定ファイルへの設定の保存

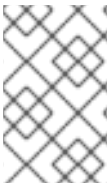
Red Hat Support Toolは、(他の方法が設定されていない場合に) ~/**.redhat-support-tool/redhat-support-tool.conf** 設定ファイルを使用して現在のユーザーのホームディレクトリーに値とオプションをローカルで保存します。必要に応じて、パスワードをこのファイルに保存することが推奨されます。この場合、パスワードは特定のユーザーのみが確認できます。ツールが起動すると、グローバル設定ファイル /**etc/redhat-support-tool.conf** とローカル設定ファイルから値が読み取られます。ローカルに保存された値とオプションは、グローバルに保存された設定よりも優先されます。

警告

グローバルな /etc/redhat-support-tool.conf 設定ファイルにパスワードを保存することは **推奨されません**。パスワードは **base64** でエンコードされているだけなので簡単にデコードできます。また、ファイルは誰でも読み取り可能です。

値とオプションをグローバル設定ファイルに保存するには、以下のように **-g, --global** オプションを追加します。

```
Command (? for help): config setting -g value
```



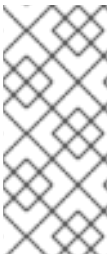
注記

-g, --global オプションを使用して設定をグローバルで保存できるようにするには、Red Hat Support Tool を **root** で実行する必要があります。これは、通常のユーザーには **/etc/redhat-support-tool.conf** への書き込みに必要なパーミッションがないためです。

値またはオプションをローカル設定ファイルから削除するには、以下のように **-u, --unset** オプションを追加します。

```
Command (? for help): config setting -u value
```

これにより、ツールからパラメーターが削除および設定解除され、(利用可能な場合は) グローバル設定ファイルにある同等の設定が使用されます。



注記

非特権ユーザーとして実行している場合は、グローバル設定ファイルに保存される値は **-u, --unset** オプションで削除できません。しかし、**-g, --global** オプションと **-u, --unset** オプションを同時に使用して、ツールの現在の実行中のインスタンスから、設定を解除することができます。**root** で実行している場合は、**-g, --global** と **-u, --unset** オプションを同時に使用してグローバル設定ファイルから値とオプションを削除できます。

8.5. インタラクティブモードでのサポートケースの作成および更新

インタラクティブモードでの新しいサポートケースの作成

インタラクティブモードで新しいサポートケースを作成するには、以下の手順を実行します。

1. 以下のコマンドを入力してツールを起動します。

   ```
   ~]# redhat-support-tool
   ```

2. **opencase** コマンドを入力します。

   ```
   Command (? for help): opencase
   ```

3. 画面に表示されたプロンプトに従って製品とバージョンを選択します。

4. ケースの要約を入力します。
5. ケースの説明を入力し、完了したら空の行で **Ctrl**+**D** を押します。
6. ケースの重大度を選択します。
7. オプションで、サポートケースを作成する前に、この問題のソリューションが存在するかどうかを確認することを選択します。
8. サポートケースを作成することを確定します。

```
Support case 0123456789 has successfully been opened
```

9. オプションで、SOS レポートを添付することを選択します。
10. オプションで、ファイルを添付することを選択します。

インタラクティブモードでの既存のサポートケースの表示および更新

インタラクティブモードで既存のサポートケースを表示および更新するには、以下の手順を実行します。

1. 以下のコマンドを入力してツールを起動します。

```
~]# redhat-support-tool
```

2. **getcase** コマンドを入力します。

```
Command (? for help): getcase case-number
```

case-number は、表示および更新するケースの番号です。

3. 画面に表示されたプロンプトに従ってケースを表示し、コメントを変更または追加して、添付ファイルを取得または追加します。

インタラクティブモードでの既存のサポートケースの変更

インタラクティブモードで既存のサポートケースの属性を変更するには、以下の手順を実行します。

1. 以下のコマンドを入力してツールを起動します。

```
~]# redhat-support-tool
```

2. **modifycase** コマンドを入力します。

```
Command (? for help): modifycase case-number
```

case-number は、表示および更新するケースの番号です。

3. 変更の選択リストが表示されます。

```
Type the number of the attribute to modify or 'e' to return to the previous menu.
 1 Modify Type
 2 Modify Severity
 3 Modify Status
 4 Modify Alternative-ID
```

```
 5 Modify Product
 6 Modify Version
End of options.
```

画面に表示されたプロンプトに従って 1 つまたは複数のオプションを変更します。

4. たとえば、ステータスを変更する場合は、**3** と入力します。

```
Selection: 3
 1  Waiting on Customer
 2  Waiting on Red Hat
 3  Closed
Please select a status (or 'q' to exit):
```

8.6. コマンドラインでのサポートケースの表示

コマンドラインでケースの内容を表示すると、コマンドラインからソリューションを素早く簡単に適用できます。

コマンドラインで既存のサポートケースを表示するには、以下のようにコマンドを入力します。

```
~]# redhat-support-tool getcase case-number
```

case-number は、ダウンロードするケースの番号です。

8.7. 関連情報

Red Hat ナレッジベースの記事 Red Hat Support Tool には、追加情報、例、および動画チュートリアルが含まれます。

パート III. ソフトウェアのインストールおよび管理

Red Hat Enterprise Linux システムのソフトウェアすべては RPM パッケージとして分割されており、インストール、アップグレード、削除が可能です。ここで、**Yum** を使用して Red Hat Enterprise Linux でパッケージを管理する方法を説明します。

第9章 YUM

yum は、Red Hat のパッケージマネージャーです。yum を使用すれば、利用可能なパッケージ情報に関するクエリー、リポジトリーからのパッケージのフェッチ、パッケージのインストールおよびアンインストール、さらには利用可能な最新バージョンへのシステム全体の更新が可能です。yum は、パッケージの更新、インストール、削除を行っている時に、依存関係の自動解決を行います。そのため、利用可能なすべての依存パッケージを自動的に決定、フェッチ、インストールできます。

yum は、新たに追加されたリポジトリー、または **パッケージソース** で設定でき、その機能を強化または拡張するプラグインを多数提供します。また、Yum は **RPM** が実行可能な同じタスクの多くを行うことができます。さらに、多数のコマンドラインオプションも似ています。Yum を使用することで、1つのマシンまたはマシンのグループ上でのパッケージ管理を簡単かつシンプルに行うことができます。

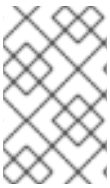
以下のセクションでは、ご使用のシステムが、 Red Hat Enterprise Linux 7 インストールガイド の従ってインストール中に Red Hat サブスクリプション管理で登録されたことを前提としています。システムがサブスクライブされていない場合は、*7章 システム登録およびサブスクリプション管理* を参照してください。

重要

Yum は、GPG (Gnu Privacy Guard (別名 GnuPG)) の署名付きパッケージの GPG 署名認証をすべてのパッケージリポジトリー (パッケージソース) または個々のリポジトリーで有効にすることで、セキュアなパッケージ管理を実現します。署名認証が有効になっていると、Yum は正しいキーで GPG 署名されていないパッケージのそのリポジトリーへのインストールを拒否します。つまり、使用中のシステムにダウンロードしてインストールする **RPM** パッケージが Red Hat などの信頼されたソースからのものであり、ダウンロード中に変更されていないことを保証します。Yum の署名認証を有効にする方法は 「Yum と Yum リポジトリーの設定」 を参照してください。

Yum を使用すると、他のマシンへダウンロードし、インストールするための **RPM** パッケージのリポジトリーを簡単に設定することもできます。可能な場合は、yum は複数パッケージとメタデータの **並行ダウンロード** を使用してダウンロードのスピードを高めます。

システム管理タスクの実行には Yum が最速の方法であることが多いため、これを使用することが推奨されます。また、Yum は、**PackageKit** グラフィカルパッケージ管理ツールが提供する以上の機能を提供します。



注記

yum を使用して、システムにパッケージをインストール、更新、削除するにはスーパーユーザー権限が必要です。本章のすべての例では、**su** または **sudo** コマンドを使用することでスーパーユーザー権限をすでに持っていると仮定しています。

9.1. パッケージの確認と更新

Yum を使用すると、使用中のシステムに適用される更新があるかどうかをチェックできます。更新が必要なパッケージをリスト表示して一度に更新したり、パッケージを個別に選択して更新したりできます。

9.1.1. 更新の確認

使用しているシステムに利用可能な更新があるインストール済みのパッケージを確認するには、以下のコマンドを実行します。

yum check-update

例9.1 yum check-update コマンドの出力例

yum check-update の出力は以下のようになります。

```
~]# yum check-update
Loaded plugins: product-id, search-disabled-repos, subscription-manager
dracut.x86_64              033-360.el7_2   rhel-7-server-rpms
dracut-config-rescue.x86_64      033-360.el7_2   rhel-7-server-rpms
kernel.x86_64              3.10.0-327.el7   rhel-7-server-rpms
rpm.x86_64              4.11.3-17.el7   rhel-7-server-rpms
rpm-libs.x86_64            4.11.3-17.el7   rhel-7-server-rpms
rpm-python.x86_64          4.11.3-17.el7   rhel-7-server-rpms
yum.noarch              3.4.3-132.el7   rhel-7-server-rpms
```

上記の出力に表示されているパッケージには利用可能な更新があります。リストの最初のパッケージは **dracut** です。出力例の各行は複数の項目で設定されます。**dracut** の場合は、以下の設定になっています。

- **dracut**: パッケージ名
- **x86_64**: パッケージがビルドされた CPU アーキテクチャー
- **033**: インストールする更新パッケージのバージョン
- **360.el7**: 更新パッケージのリリース
- **_2**: ビルドバージョン (z-stream 更新として追加)
- **rhel-7-server-rpms**: 更新済みのパッケージがあるリポジトリー

また上記の出力はすべて、**yum** コマンドを使用してカーネル (**yum** パッケージ)、Yum および RPM (**yum** および **rpm** パッケージ)、さらにはその依存関係 (**rpm-libs**、**rpm-python** パッケージ) をすべて更新できることも示しています。

9.1.2. パッケージの更新

一度に更新するパッケージ数を 1 つ、複数、または全てのパッケージから選択できます。更新するパッケージの依存関係、またはパッケージに利用可能な更新がある場合は、併せて更新されます。

単一パッケージの更新
1 つのパッケージを更新するには、**root** で以下のコマンドを実行します。

yum update **package_name**

例9.2 rpm パッケージの更新

rpm パッケージを更新するには、以下を入力します。

```
~]# yum update rpm
Loaded plugins: langpacks, product-id, subscription-manager
Updating Red Hat repositories.
```

```
INFO:rhsm-app.repolib:repos updated: 0
Setting up Update Process
Resolving Dependencies
--> Running transaction check
---> Package rpm.x86_64 0:4.11.1-3.el7 will be updated
--> Processing Dependency: rpm = 4.11.1-3.el7 for package: rpm-libs-4.11.1-3.el7.x86_64
--> Processing Dependency: rpm = 4.11.1-3.el7 for package: rpm-python-4.11.1-3.el7.x86_64
--> Processing Dependency: rpm = 4.11.1-3.el7 for package: rpm-build-4.11.1-3.el7.x86_64
---> Package rpm.x86_64 0:4.11.2-2.el7 will be an update
--> Running transaction check
...
--> Finished Dependency Resolution

Dependencies Resolved
================================================================================

 Package         Arch    Version     Repository    Size
================================================================================

Updating:
 rpm            x86_64   4.11.2-2.el7  rhel      1.1 M
Updating for dependencies:
 rpm-build      x86_64   4.11.2-2.el7  rhel      139 k
 rpm-build-libs      x86_64   4.11.2-2.el7  rhel       98 k
 rpm-libs       x86_64   4.11.2-2.el7  rhel      261 k
 rpm-python     x86_64   4.11.2-2.el7  rhel       74 k

Transaction Summary
================================================================================

Upgrade 1 Package (+4 Dependent packages)

Total size: 1.7 M
Is this ok [y/d/N]:
```

この出力で重要となる項目がいくつかあります。

1. **Loaded plugins: langpacks, product-id, subscription-manager**: yum は、どの Yum プラグインがインストールされ有効であるかを常に通知します。Yum プラグインに関する一般的情報は 「yum のプラグイン」 を参照してください。また、個別のプラグインに関する説明は 「yum プラグインの使用方法」 を参照してください。

2. **rpm.x86_64**: 新しい **rpm** パッケージとその依存関係をダウンロードしてインストールできます。これらの各パッケージに対してトランザクションチェックが行われます。

3. yum を使用すると、更新情報を表示し、更新を確認できます。yum は、デフォルトで対話的に動作します。**yum** コマンドが実行する予定のトランザクションがすでに分かっている場合は、**-y** オプションを使用して、yum が質問する質問 (この場合は非対話的に実行) に対して、自動的に **yes** と回答できます。オプションを使用して、yum が尋ねるすべての質問に自動的に yes と回答するように設定できます (この場合は非対話的に実行されます)。ただし、yum によりシステムに行われる変更を常に調べる必要があります。これを行うには、ダウンロードプロンプトで **d** オプションを選択します。これにより、選択されたパッケージのバックグラウンドでのダウンロードが開始します。
   トランザクションが正しく行われなかった場合は、「トランザクション履歴の活用」 にあるように **yum history** コマンドを使用して Yum のトランザクション履歴を表示できます。

重要

yum update コマンドまたは **yum install** コマンドを使用しているかどうかに関係なく、Yum は常に新しいのカーネルを **インストール** します。

一方、RPM を使用する場合は、**rpm -u kernel** コマンド (現在のカーネルを **置き換える**) ではなく、**rpm -i kernel** コマンド (新しいカーネルをインストール) を使用することが重要です。

同様に、パッケージグループを更新できます。**root** で次のコマンドを入力します。

```
yum group update group_name
```

group_name を、更新するパッケージグループの名前に置き換えます。パッケージグループの詳細は「パッケージグループでの作業」 を参照してください。

Yum は、**obsoletes** 設定オプションを有効にして **更新** する **upgrade** コマンドも提供します (「[main] オプションの設定」 を参照)。**obsoletes** は **/etc/yum.conf** で on になっており、これによりこの 2 つのコマンドが同等のものになっています。

すべてのパッケージとそれらの依存関係の更新

パッケージとその依存関係をすべて更新するには、引数なしで **yum update** コマンドを実行します。

```
yum update
```

セキュリティー関連パッケージの更新

パッケージでセキュリティー更新が利用可能な場合は、そのパッケージのみを最新のバージョンに更新できます。**root** で次のコマンドを入力します。

```
yum update --security
```

また、最新のセキュリティー更新を含むバージョンにのみパッケージを更新することもできます。**root** で次のコマンドを入力します。

```
yum update-minimal --security
```

たとえば、以下の例を考えてみます。

- kernel-3.10.0-1 パッケージがシステムにインストールされている。
- kernel-3.10.0-2 パッケージがセキュリティー更新としてリリースされている。
- kernel-3.10.0-3 パッケージがバグ修正の更新としてリリースされている。

この場合、**yum update-minimal --security** だとパッケージが kernel-3.10.0-2 に更新され、**yum update --security** だとパッケージが kernel-3.10.0-3 に更新されます。

パッケージの自動更新

パッケージのデータベースを更新し、更新を自動的にダウンロードするには、**yum-cron** サービスを使用できます。詳細は、「yum-cron を使用したパッケージデータベースの自動更新および更新のダウンロード」 を参照してください。

9.1.3. ISO と Yum を使用してシステムをオフラインでアップグレード

インターネットまたは Red Hat Network から切断されたシステムの場合は、**yum update** コマンドと Red Hat Enterprise Linux インストール ISO イメージを使用すると、システムを最新のマイナーバージョンに簡単かつ素早くアップグレードできます。以下の手順はアップグレードプロセスを示しています。

1. ISO イメージをマウントするターゲットディレクトリーを作成します。このディレクトリーは、マウント時に自動的に作成されません。**root** で以下のコマンドを実行します。

   ```
   mkdir mount_dir
   ```

   mount_dir は、マウントディレクトリーへのパスに置き換えます。通常は、ユーザーが **/media** ディレクトリー内のサブディレクトリーとして作成します。

2. 以前に作成されたターゲットディレクトリーに Red Hat Enterprise Linux 7 インストール ISO イメージをマウントします。**root** で以下のコマンドを実行します。

   ```
   mount -o loop iso_name mount_dir
   ```

   iso_name を ISO イメージへのパスと置き換え、 **mount_dir** をターゲットディレクトリーへのパスと置き換えます。ブロックデバイスとしてファイルをマウントするには、**-o loop** オプションが必要です。

3. **media.repo** ファイルをマウントディレクトリーから **/etc/yum.repos.d/** ディレクトリーにコピーします。正常に機能するために、このディレクトリーの設定ファイルの拡張子は **.repo** である必要があります。

   ```
   cp mount_dir/media.repo /etc/yum.repos.d/new.repo
   ```

   これにより、yum リポジトリーの設定ファイルが作成されます。**new.repo** をファイル名と置き換えます (例: **rhel7.repo**)。

4. Red Hat Enterprise Linux インストール ISO を参照するよう新しい設定ファイルを編集します。以下の行を **/etc/yum.repos.d/new.repo** ファイルに追加します。

   ```
   baseurl=file:///mount_dir
   ```

   mount_dir をマウントポイントへのパスと置き換えます。

5. 前の手順で作成された **/etc/yum.repos.d/new.repo** を含むすべての yum リポジトリーを更新します。**root** で以下のコマンドを実行します。

   ```
   yum update
   ```

   これにより、システムはマウントされた ISO イメージで提供されたバージョンにアップグレードされます。

6. アップグレードに成功したら、ISO イメージをアンマウントできます。**root** で以下のコマンドを実行します。

   ```
   umount mount_dir
   ```

   ここで、**mount_dir** はマウントディレクトリーへのパスです。また、最初の手順で作成されたマウントディレクトリーを削除することもできます。**root** で以下のコマンドを実行します。

```
rmdir mount_dir
```

7. 以前に作成された設定ファイルを別のインストールまたは更新に使用しない場合は、その設定ファイルを削除できます。**root** で以下のコマンドを実行します。

```
rm /etc/yum.repos.d/new.repo
```

例9.3 Red Hat Enterprise Linux 7.0 から 7.1 へのアップグレード

インターネットにアクセスせずに ISO イメージ (例: **rhel-server-7.1-x86_64-dvd.iso**) を使用してシステムを新しいバージョンにアップグレードする必要がある場合は、**/media/rhel7/** などのマウント用ターゲットディレクトリーを作成します。**root** で ISO イメージがあるディレクトリーに移動し、以下のコマンドを入力します。

```
~]# mount -o loop
  rhel-server-7.1-x86_64-dvd.iso /media/rhel7/
```

次に、マウントディレクトリーから **media.repo** ファイルをコピーして、イメージ用の yum リポジトリーをセットアップします。

```
~]# cp /media/rhel7/media.repo /etc/yum.repos.d/rhel7.repo
```

yum にマウントポイントをリポジトリーとして認識させるために、前の手順でコピーした **/etc/yum.repos.d/rhel7.repo** に以下の行を追加します。

```
baseurl=file:///media/rhel7/
```

この時点で、yum リポジトリーを更新すると、**rhel-server-7.1-x86_64-dvd.iso** により提供されたバージョンにシステムがアップグレードされます。**root** で以下のコマンドを実行します。

```
~]# yum update
```

システムが正常にアップグレードされたら、イメージをアンマウントし、ターゲットディレクトリーと設定ファイルを削除できます。

```
~]# umount /media/rhel7/
```

```
~]# rmdir /media/rhel7/
```

```
~]# rm
    /etc/yum.repos.d/rhel7.repo
```

9.2. パッケージでの作業

Yum では、パッケージの検索、パッケージについての情報の表示、パッケージのインストールおよび削除など、ソフトウェアパッケージの完全な操作が可能です。

9.2.1. パッケージの検索

以下のコマンドを使用することで、すべての **RPM** のパッケージ名、詳細、サマリーを検索できます。

yum search term…

term を、検索するパッケージ名に置き換えます。

例9.4 特定の文字列に一致するパッケージの検索

vim や gvim、または emacs に一致するパッケージをリスト表示するには、以下を入力します。

```
~]$ yum search vim gvim emacs
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
============================== N/S matched: vim ==============================
vim-X11.x86_64 : The VIM version of the vi editor for the X Window System
vim-common.x86_64 : The common files needed by any version of the VIM editor
[output truncated]

============================== N/S matched: emacs ==============================
emacs.x86_64 : GNU Emacs text editor
emacs-auctex.noarch : Enhanced TeX modes for Emacs
[output truncated]

 Name and summary matches mostly, use "search all" for everything.
Warning: No matches found for: gvim
```

yum search コマンドは、パッケージ名は分からないものの、関連用語を知っている場合にパッケージを検索する際に役立ちます。デフォルトでは、**yum search** はパッケージ名とサマリーが一致したものを返すため、検索には時間がかかりません。**yum search all** コマンドを使用して、より詳細な検索を行いますが、検索は遅くなります。

結果のフィルタリング

yum の list コマンドでは、1つ以上の **glob 表現** を引数として追加することで、結果をフィルタリングできます。glob 表現は、1つ以上のワイルドカード文字 ***** (任意の文字サブセットに拡張) と **?** (任意の1文字に拡張) を含む通常の文字列です。

yum コマンドに glob 表現を引数として渡す場合には、glob 表現をエスケープするように注意してください。これを行わないと、bash シェルはこの表現を **パス名のデプロイメント** と解釈してしまい、glob と適合する現在のディレクトリー内の全ファイルを **yum** に渡すおそれがあります。確実に glob 表現を **yum** に渡すには、以下のいずれかの方法で行います。

- ワイルドカード文字の前にバックスラッシュ記号を入力して、ワイルドカード文字をエスケープする
- glob 表現全体を二重引用符または単一引用符でくくる

以下のセクションの例では、上記の両方の使用例を説明します。

9.2.2. パッケージのリスト表示

すべてのインストール済み **および** 利用可能なパッケージに関する情報をリスト表示するには、シェルプロンプトで以下を入力します。

yum list all

glob 表現に一致するインストール済み **および** 利用可能なパッケージをリスト表示するには、以下のコマンドを使用します。

```
yum list glob_expression&hellip;
```

例9.5 ABRT 関連パッケージのリスト

各種の ABRT アドオンとプラグインを持つパッケージは abrt-addon- か abrt-plugin- で始まります。これらのパッケージをリスト表示するには、シェルプロンプトで以下を入力します。ワイルドカード文字の前にバックスラッシュ文字を置くことでエスケープしていることに注意してください。

```
~]$ yum list abrt-addon\* abrt-plugin\*
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
Installed Packages
abrt-addon-ccpp.x86_64          2.1.11-35.el7       @rhel-7-server-rpms
abrt-addon-kerneloops.x86_64    2.1.11-35.el7       @rhel-7-server-rpms
abrt-addon-pstoreoops.x86_64    2.1.11-35.el7       @rhel-7-server-rpms
abrt-addon-python.x86_64        2.1.11-35.el7       @rhel-7-server-rpms
abrt-addon-vmcore.x86_64        2.1.11-35.el7       @rhel-7-server-rpms
abrt-addon-xorg.x86_64          2.1.11-35.el7       @rhel-7-server-rpms
```

installed キーワードを使用して、システムにインストールされているパッケージをリスト表示するには、以下のコマンドを実行します。出力の右端の列には、パッケージを取得したリポジトリーが表示されます。

```
yum list installed glob_expression&hellip;
```

例9.6 インストール済み krb パッケージのリスト表示

以下の例では、krb で始まり、その後に正確に 1 文字とハイフンが続くインストール済みパッケージをリスト表示する方法を示しています。この方法では数字でバージョンが見分けられるので、特定コンポーネントの全バージョンをリスト表示したい場合に便利です。glob 表現全体を引用符で囲むことで適切な処理が確実になります。

```
~]$ yum list installed "krb?-*"
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
Installed Packages
krb5-libs.x86_64        1.13.2-10.el7         @rhel-7-server-rpms
```

有効なすべてのリポジトリーでインストール可能なパッケージをリスト表示するには、以下の形式のコマンドを使用します。

```
yum list available glob_expression&hellip;
```

例9.7 利用可能な gstreamer プラグインのリスト表示

たとえば、gstreamer とその後に plugin を含む名前の利用可能なパッケージをリスト表示するには、以下のコマンドを実行します。

```
~]$ yum list available gstreamer*plugin\*
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
Available Packages
gstreamer-plugins-bad-free.i686       0.10.23-20.el7     rhel-7-server-rpms
gstreamer-plugins-base.i686           0.10.36-10.el7     rhel-7-server-rpms
gstreamer-plugins-good.i686           0.10.31-11.el7     rhel-7-server-rpms
gstreamer1-plugins-bad-free.i686      1.4.5-3.el7        rhel-7-server-rpms
gstreamer1-plugins-base.i686          1.4.5-2.el7        rhel-7-server-rpms
gstreamer1-plugins-base-devel.i686    1.4.5-2.el7        rhel-7-server-rpms
gstreamer1-plugins-base-devel.x86_64  1.4.5-2.el7        rhel-7-server-rpms
gstreamer1-plugins-good.i686          1.4.5-2.el7        rhel-7-server-rpms
```

リポジトリーのリスト表示

リポジトリーの ID、名前、使用中のシステム上で **有効な** 各リポジトリーでのパッケージ数をリスト表示するには、以下のコマンドを実行します。

```
yum repolist
```

これらのリポジトリーの詳細情報をリスト表示するには、**-v** オプションを追加します。このオプションを有効にすると、各リポジトリーでファイル名や全体のサイズ、最終更新日、ベース URL といった情報が表示されます。別の方法としては、**repoinfo** コマンドを使用して同じ出力を作成することもできます。

```
yum repolist -v
```

```
yum repoinfo
```

有効および無効なリポジトリーの両方を表示するには、以下のコマンドを実行します。ステータスのコラムが出力リストに追加され、どのリポジトリーが有効になっているかが分かります。

```
yum repolist all
```

最初の引数を **disabled** にすることで、コマンドの出力を無効なリポジトリーに制限できます。また、リポジトリーの ID、名前、関連する glob 表現を引数として指定することもできます。リポジトリー ID または名前が引数と完全に一致する場合は、**enabled** フィルターまたは **disabled** フィルターを通過しないリポジトリーであっても表示されることに注意してください。

9.2.3. パッケージ情報の表示

1 つ以上のパッケージに関する情報を表示するには、以下のコマンドを実行します (ここでは glob 表現も有効)。

```
yum info package_name&hellip;
```

package_name を、パッケージ名に置き換えます。

例9.8 abrt パッケージ情報の表示

abrt パッケージに関する情報を表示するには、以下を入力します。

```
~]$ yum info abrt
```

```
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
Installed Packages
Name    : abrt
Arch    : x86_64
Version   : 2.1.11
Release   : 35.el7
Size    : 2.3 M
Repo    : installed
From repo  : rhel-7-server-rpms
Summary   : Automatic bug detection and reporting tool
URL     : https://fedorahosted.org/abrt/
License   : GPLv2+
Description : abrt is a tool to help users to detect defects in applications and
      : to create a bug report with all information needed by maintainer to fix
      : it. It uses plugin system to extend its functionality.
```

yum info package_name コマンドは **rpm -q --info package_name** コマンドに似ていますが、追加情報として RPM パッケージのインストール元である yum リポジトリーの名前を提供します (出力の **From repo:** の行を参照)。

yumdb の使用

以下のコマンドを使用して、パッケージに関する代替情報や有用な情報について Yum データベースにクエリーすることもできます。

```
yumdb info package_name
```

このコマンドは、パッケージのチェックサム (および SHA-256 などのチェックサムを算出するためのアルゴリズム)、パッケージのインストール開始に使用されたコマンドラインのコマンド (存在する場合)、パッケージがシステムにインストールされた理由 (**user** はユーザーがインストールしたことを、**dep** は依存関係として取り入れたことを意味します) などのパッケージに関する追加情報を提供します。

例9.9 yum パッケージに関する情報を yumdb でクエリー

yum パッケージに関する追加情報を表示するには、以下を入力します。

```
~]$ yumdb info yum
Loaded plugins: langpacks, product-id
yum-3.4.3-132.el7.noarch
  changed_by = 1000
  checksum_data = a9d0510e2ff0d04d04476c693c0313a11379053928efd29561f9a837b3d9eb02
  checksum_type = sha256
  command_line = upgrade
  from_repo = rhel-7-server-rpms
  from_repo_revision = 1449144806
  from_repo_timestamp = 1449144805
  installed_by = 4294967295
  origin_url = https://cdn.redhat.com/content/dist/rhel/server/7/7Server/x86_64/os/Packages/yum-
3.4.3-132.el7.noarch.rpm
  reason = user
  releasever = 7Server
  var_uuid = 147a7d49-b60a-429f-8d8f-3edb6ce6f4a1
```

yumdb コマンドの詳細は、man ページの **yumdb**(8) を参照してください。

9.2.4. パッケージのインストール

1 つのパッケージと、そのパッケージの依存関係でインストールされていないものをすべてインストールするには、**root** で以下の形式のコマンドを入力します。

```
yum install package_name
```

複数パッケージを同時にインストールするには、その名前を引数として追加します。これを実行するには、**root** で次のコマンドを実行します。

```
yum install package_name package_name&hellip;
```

AMD64 マシンや Intel 64 マシンなどの **multilib** システムにパッケージをインストールする場合は、パッケージ名に **.arch** を追加して、パッケージのアーキテクチャーを指定できます (ただし、有効なリポジトリーで利用可能な場合のみ)。

```
yum install package_name.arch
```

例9.10 multilib システムでのパッケージのインストール

i686 アーキテクチャー用の **sqlite** パッケージをインストールするには、以下を入力します。

```
~]# yum install sqlite.i686
```

glob 表現を使用すると、名前が似ている複数のパッケージを迅速にインストールできます。**root** で以下のコマンドを実行します。

```
yum install glob_expression&hellip;
```

例9.11 audacious の全プラグインのインストール

似た名前の複数のパッケージをインストールするには、glob 表現が便利です。audacious プラグインをすべてインストールするには、以下の形式でコマンドを使用します。

```
~]# yum install audacious-plugins-\*
```

パッケージ名と glob 表現に加えて、**yum install** にはファイル名も追加できます。インストールするバイナリー名が分かっていて、パッケージ名が分からない場合は、**yum install** にパス名を付けて実行します。**root** で以下のコマンドを実行します。

```
yum install /usr/sbin/named
```

yum はパッケージリストで検索を行い、**/usr/sbin/named** を提供するパッケージを探します。パッケージが存在すると、yum により、そのパッケージをインストールするかどうかを尋ねられます。

上記の例で分かるように、**yum install** コマンドで必要な変数は、厳密に定義されていません。様々な形式のパッケージ名や glob 表現を処理できるため、ユーザーによるインストールを容易にします。一方で、**yum** が入力を正確に分析するには時間がかかります。指定するパッケージの数が多くなれば、そ

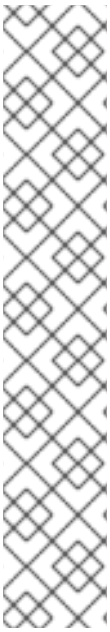
れだけ時間がかかります。したがって、パッケージ検索を最適化するために、以下のコマンドを実行して引数の分析方法を明示的に定義できます。

```
yum install-n name
```

```
yum install-na name.architecture
```

```
yum install-nevra name-epoch:version-release.architecture
```

yum は、**install-n** コマンドでは name をパッケージの正確な名前として解釈します。また、**install-na** コマンドでは、後続の引数で、ピリオドを使用してパッケージ名とアーキテクチャーを指定していると yum は解釈します。一方、**install-nevra** では、yum で、name-epoch:version-release.architecture の形で引数を指定していることが必要になります。同様に、削除するパッケージを検索する際は、**yum remove-n**、**yum remove-na**、および **yum remove-nevra** が必要です。



注記

named バイナリーを含むパッケージをインストールする前に、ファイルがインストールされているのが **bin/** ディレクトリーか **sbin/** ディレクトリーか分からない場合は、glob 表現を付けて **yum provides** コマンドを実行します。

```
~]# yum provides "*bin/named"
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-
      : manager
32:bind-9.9.4-14.el7.x86_64 : The Berkeley Internet Name Domain (BIND) DNS
            : (Domain Name System) server
Repo    : rhel-7-server-rpms
Matched from:
Filename  : /usr/sbin/named
```

yum provides "*/file_name" は、file_name が含まれるパッケージを検索するのに便利です。

例9.12 インストールプロセス

以下の例は、**yum** を使用したインストールの概要を示しています。最新バージョンの **httpd** パッケージをダウンロードして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 httpd
Loaded plugins: langpacks, product-id, subscription-manager
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.4.6-12.el7 will be updated
---> Package httpd.x86_64 0:2.4.6-13.el7 will be an update
--> Processing Dependency: 2.4.6-13.el7 for package: httpd-2.4.6-13.el7.x86_64
--> Running transaction check
---> Package httpd-tools.x86_64 0:2.4.6-12.el7 will be updated
---> Package httpd-tools.x86_64 0:2.4.6-13.el7 will be an update
--> Finished Dependency Resolution

Dependencies Resolved
```

上記のコマンドを実行した後、**yum** は必要なプラグインを読み込み、トランザクションチェックを

実行します。このケースでは、**httpd** がすでにインストールされています。インストール済みのパッケージが利用可能な最新バージョンよりも古いことから、これは更新されます。**httpd** が依存する **httpd-tools** にも同様のことが行われます。すると、トランザクションサマリーは以下のように表示されます。

```
================================================================================
 Package    Arch   Version         Repository        Size
================================================================================
Updating:
 httpd      x86_64  2.4.6-13.el7      rhel-x86_64-server-7  1.2 M
Updating for dependencies:
 httpd-tools  x86_64  2.4.6-13.el7      rhel-x86_64-server-7   77 k

Transaction Summary
================================================================================
Upgrade 1 Package (+1 Dependent package)

Total size: 1.2 M
Is this ok [y/d/N]:
```

このステップでは、**yum** がインストールを確認するプロンプトを表示します。**y** (yes) および **N** (no) オプションのほかに、**d** (ダウンロードのみ) を選択してパッケージをダウンロードしますが、直接インストールすることはできません。**y** を選択すると、以下のメッセージが出て、インストールが正常に完了するまで続行します。

```
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
 Updating  : httpd-tools-2.4.6-13.el7.x86_64              1/4
 Updating  : httpd-2.4.6-13.el7.x86_64                    2/4
 Cleanup   : httpd-2.4.6-12.el7.x86_64                    3/4
 Cleanup   : httpd-tools-2.4.6-12.el7.x86_64              4/4
 Verifying : httpd-2.4.6-13.el7.x86_64                    1/4
 Verifying : httpd-tools-2.4.6-13.el7.x86_64              2/4
 Verifying : httpd-tools-2.4.6-12.el7.x86_64              3/4
 Verifying : httpd-2.4.6-12.el7.x86_64                    4/4

Updated:
 httpd.x86_64 0:2.4.6-13.el7

Dependency Updated:
 httpd-tools.x86_64 0:2.4.6-13.el7

Complete!
```

ダウンロード済みのパッケージを、システム上のローカルディレクトリーからインストールするには、以下のコマンドを使用します。

```
yum localinstall path
```

-

path を、インストールするパッケージのパスに置き換えます。

9.2.5. パッケージのダウンロード

例9.12「インストールプロセス」 にあるように、インストールプロセスのある時点で、インストールを確認する以下のメッセージが表示されます。

```
...
Total size: 1.2 M
Is this ok [y/d/N]:
...
```

d オプションを指定すると、yum は、パッケージをインストールせずにダウンロードを行います。ダウンロードしたパッケージは、キャッシュディレクトリーのサブディレクトリー (デフォルトでは **/var/cache/yum/$basearch/$releasever/packages/**) の 1 つに保存されます。ダウンロードしたパッケージは、キャッシュディレクトリーのサブディレクトリー (デフォルトでは **/var/cache/yum/$basearch/$releasever/packages/**) のいずれかに保存されます。ダウンロードはバックグラウンドモードで続行されるため、**yum** を並行して他の操作に使用できます。

9.2.6. パッケージの削除

パッケージのインストールと同様に、yum を使用するとパッケージのアンインストールができます。特定のパッケージと、そのパッケージの依存関係パッケージをすべてアンインストールをするには、**root** で以下のコマンドを実行します。

```
yum remove package_name&hellip;
```

複数のパッケージをインストールする場合と同様、コマンドに複数のパッケージ名を追加すると、一度に複数のパッケージを削除できます。

例9.13 複数パッケージの削除

totem を削除するには、シェルプロンプトで以下を入力します。

```
~]# yum remove totem
```

install と同じように、**remove** では、以下の引数を使用できます。

- パッケージ名
- glob 表現
- ファイルリスト
- パッケージが提供する機能

警告

Yum では、パッケージを削除して、その依存パッケージを残すことはできません。こうした動作は **RPM** でのみ実行可能であり、推奨されません。システムが機能しなくなる、またはアプリケーションに誤作動やクラッシュが生じる恐れがあるためです。

9.3. パッケージグループでの作業

パッケージグループは、たとえば **システムツール** や **サウンドとビデオ** などの共通の目的でサービスを行うパッケージの集合です。パッケージグループをインストールすると、依存パッケージも取得するため、時間が大幅に短縮できます。**yum groups** コマンドは、yum のパッケージグループに作用するすべての操作をカバーするトップレベルのコマンドです。

9.3.1. パッケージグループのリスト表示

summary オプションを使用すると、インストール済みのグループ数、利用可能なグループ数、利用可能な環境グループ数、インストール済みの言語グループ数、利用可能な言語グループ数が表示されます。

> **yum groups summary**

> **例9.14 yum groups summary の出力例**
>
> ```
> ~]$ yum groups summary
> Loaded plugins: langpacks, product-id, subscription-manager
> Available Environment Groups: 12
> Installed Groups: 10
> Available Groups: 12
> ```

yum リポジトリーからすべてのパッケージグループをリスト表示するには、**list** オプションを追加します。コマンドの出力は、グループ名でフィルターを設定できます。

> yum group list **glob_expression**…

このコマンドで使用できる任意の引数がいくつかあります。たとえば、**hidden** は、ユーザーに表示可能とされていないグループもリスト表示し、**ids** はグループ ID を表示します。**language** オプション、**environment** オプション、**installed**、または **available** オプションを追加して、出力を特定のグループタイプに制限することもできます。

特定のグループに含まれている必須およびオプションパッケージをリスト表示するには、以下のコマンドを使用します。

> yum group info **glob_expression**…

> **例9.15 LibreOffice パッケージグループの情報表示**

```
 ~]$ yum group info LibreOffice
Loaded plugins: langpacks, product-id, subscription-manager

Group: LibreOffice
 Group-Id: libreoffice
 Description: LibreOffice Productivity Suite
 Mandatory Packages:
 =libreoffice-calc
  libreoffice-draw
 -libreoffice-emailmerge
  libreoffice-graphicfilter
 =libreoffice-impress
 =libreoffice-math
 =libreoffice-writer
 +libreoffice-xsltfilter
 Optional Packages:
  libreoffice-base
  libreoffice-pyuno
```

上記の例で分かるように、このパッケージグループに含まれているパッケージは、以下の記号でマークされている状態に分けられます。

- **-**: パッケージはインストールされておらず、このパッケージグループではインストールされません。
- **+**: パッケージはインストールされていませんが、次回の **yum upgrade** または **yum group upgrade** でインストールされます。
- **=**: パッケージはインストールされており、パッケージグループの一部としてインストールされました。
- **記号なし**: パッケージはインストールされていますが、パッケージグループとしてはインストールされませんでした。このため、**yum group remove** ではこのパッケージを削除できません。

この区別は、**group_command** 設定パラメーターがデフォルト設定の **objects** に設定されている場合にのみ行われます。グループの一部として、または個別にパッケージをインストールしたかを yum で追跡したくない場合は、このパラメーターを異なる値に設定します。すると、"**記号なし**" パッケージと "**=**" パッケージが同じ意味になります。

yum group mark コマンドを使用する上記のパッケージの状態を変更することもできます。たとえば、**yum group mark packages** は、特定のインストール済みパッケージを指定されたグループのメンバーとしてマークします。グループ更新で新たなパッケージのインストールをしないようにするには、**yum group mark blacklist** を使用します。**yum group mark** の詳細な機能は man ページの **yum**(8) を参照してください。

注記

@^ 接頭辞を使用すると環境グループが特定でき、パッケージグループには @ のマークが付きます。**yum group list**、**info**、**install**、または **remove** を使用する場合は、**@group_name** を渡すてパッケージグループを指定し、**@^group_name** で環境グループを指定します。または group_name の両方を含める場合は **group_name** を渡します。

9.3.2. パッケージグループのインストール

パッケージグループにはそれぞれ、名前とグループ ID (**groupid**) があります。パッケージグループの名前とグループ ID (括弧内に表示される) をリスト表示するには、以下のコマンドを入力します。

```
yum group list ids
```

例9.16 パッケージグループの名前と groupid の表示

パッケージグループ (例: KDE デスクトップ環境に関連するグループ) の名前または ID を検索するには、以下のコマンドを入力します。

```
~]$ yum group list ids kde\*
Available environment groups:
  KDE Plasma Workspaces (kde-desktop-environment)
Done
```

一部のグループは、設定されたリポジトリーの設定により非表示になっています。たとえば、サーバーで **hidden** コマンドオプションを使用すると、非表示グループも表示されます。

```
~]$ yum group list hidden ids kde\*
Loaded plugins: product-id, subscription-manager
Available Groups:
  KDE (kde-desktop)
Done
```

パッケージグループをインストールするには、**group install** コマンドに正式なグループ名 (groupid は含めない) を渡します。**root** で以下のコマンドを実行します。

```
yum group install "group name"
```

groupid を使用してインストールすることもできます。**root** で、以下のコマンドを実行します。

```
yum group install groupid
```

groupid、または引用付きグループ名の先頭に @ 記号を追加して **install** コマンドに渡すことで、**group install** と同じように **yum** を実行できます。**root** で以下のコマンドを実行します。

```
yum install @group
```

group を、groupid、または引用符で囲んだグループ名に置き換えます。同じ論理が環境グループにも適用できます。

```
yum install @^group
```

例9.17 KDE Desktop グループをインストールする 4 つの方法

上記で説明したように、パッケージグループをインストールする方法は 4 つあります。KDE Desktop の場合、コマンドは以下のようになります。

```
~]# yum group install "KDE Desktop"
~]# yum group install kde-desktop
~]# yum install @"KDE Desktop"
```

```
~]# yum install @kde-desktop
```

9.3.3. パッケージグループの削除

install 構文に類似した構文で、パッケージグループ名またはその ID を使用してパッケージグループを削除できます。**root** で以下のコマンドを実行します。

```
yum group remove group_name
```

```
yum group remove groupid
```

また、groupid または引用付き名前の先頭に @ 記号を追加して、**remove** コマンドに渡すことで、**group remove** と同じように yum を実行できます。 **root** で以下のコマンドを実行します。

```
yum remove @group
```

group を、groupid、または引用符で囲んだグループ名に置き換えます。同様に、環境グループに置き換えることもできます。

```
yum remove @^group
```

例9.18 KDE Desktop グループを削除する 4 つの方法

インストールする場合と同様に、パッケージグループを削除する方法は 4 つあります。KDE Desktop の場合、コマンドは以下のようになります。

```
~]# yum group remove "KDE Desktop"
~]# yum group remove kde-desktop
~]# yum remove @"KDE Desktop"
~]# yum remove @kde-desktop
```

9.4. トランザクション履歴の活用

yum history コマンドを使用すると、yum のトランザクションのタイムライン、トランザクションの発生日時、影響を受けたパッケージ数、トランザクション成功の有無、RPM データベースがトランザクション間で変更されたかどうかといった情報を確認できます。さらに、このコマンドを使用すると、特定のトランザクションを元に戻す、またはやり直すことが可能です。すべての履歴データは、**/var/lib/yum/history/** ディレクトリーの **history DB** に保存されます。

9.4.1. トランザクションのリスト表示

最近発生した 20 件のトランザクションをリスト表示するには、**root** で引数なしで **yum history** を実行するか、以下のコマンドを実行します。

```
yum history list
```

すべてのトランザクションを表示するには、**all** のキーワードを追加します。

```
yum history list all
```

特定の範囲内のトランザクションのみを表示したい場合は、以下の形式でコマンドを使用します。

```
yum history list start_id..end_id
```

特定のパッケージに関するトランザクションのみをリスト表示することもできます。そのためには、パッケージ名か glob 表現を付けてコマンドを実行します。

```
yum history list glob_expression&hellip;
```

例9.19 最も古いトランザクション 5 件を表示する

yum history list の出力では、最新のトランザクションがリストの上部に表示されます。履歴データベースにある最も古い 5 件のトランザクションに関する情報を表示するには、以下を入力します。

```
~]# yum history list 1..5
Loaded plugins: langpacks, product-id, subscription-manager
ID   | Login user        | Date and time  | Action(s)   | Altered
-------------------------------------------------------------------------------
   5 | User <user>       | 2013-07-29 15:33 | Install    |  1
   4 | User <user>       | 2013-07-21 15:10 | Install    |  1
   3 | User <user>       | 2013-07-16 15:27 | I, U       |  73
   2 | System <unset>    | 2013-07-16 15:19 | Update     |  1
   1 | System <unset>    | 2013-07-16 14:38 | Install    | 1106
history list
```

yum history list コマンドのすべての形式で、以下のコラムで設定される各行を含む表形式出力を生成します。

- **ID**: 特定のトランザクションを識別する整数値です。
- **Login user**: トランザクションが開始したログインセッションのユーザー名。この情報は、通常 **Full Name <username>** の形式で表示されます。ユーザーが実行しなかったトランザクションに関しては (システムの自動更新など)、代わりに **System <unset>** が使用されます。
- **Date and time**: トランザクションが発生した日時です。
- **Action(s)**: 表9.1「Action フィールドの値」 の説明通りに、トランザクション中に実行された動作のリストです。
- **Altered**: 表9.2「Altered フィールドの値」 の説明通りに、トランザクションにより影響を受けたパッケージ数、場合によっては追加情報も含まれます。

表9.1 Action フィールドの値

Action	省略形	詳細
Downgrade	**D**	1 つ以上のパッケージが旧バージョンにダウングレードされました。

Action	省略形	詳細
Erase	**E**	1つ以上のパッケージが削除されました。
Install	**I**	1つ以上の新しいパッケージがインストールされました。
Obsoleting	**O**	1つ以上のパッケージが廃止として記録されました。
Reinstall	**-R**	1つ以上のパッケージが再インストールされました。
Update	**U**	1つ以上のパッケージが新しいバージョンに更新されました。

表9.2 Altered フィールドの値

記号	詳細
<	トランザクションが終了する前に、**rpmdb** データベースが yum 以外で変更されました。
>	トランザクションが終了した後に、**rpmdb** データベースが yum 以外で変更されました。
*	トランザクションは失敗して終了しました。
#	トランザクションは正常に終了しましたが、yum はゼロ以外の終了コードを返しました。
E	トランザクションは正常に終了しましたが、エラーまたは警告が表示されました。
%P	トランザクションは正常に終了しましたが、**rpmdb** データベースに問題がすでに存在していました。
s	トランザクションは正常に終了しましたが、**--skip-broken** コマンドラインオプションが指定され、特定のパッケージが省略されました。

インストール済みパッケージの **rpmdb** または **yumdb** データベースのコンテンツを、現在使用されている **rpmdb** または **yumdb** データベースと同期するには、以下を入力します。

```
yum history sync
```

現在使用している履歴データベースに関する全体的な統計数字を表示するには、以下のコマンドを使用します。

yum history stats

例9.20 yum history stats の出力例

```
~]# yum history stats
Loaded plugins: langpacks, product-id, subscription-manager
File    : //var/lib/yum/history/history-2012-08-15.sqlite
Size    : 2,766,848
Transactions: 41
Begin time : Wed Aug 15 16:18:25 2012
End time  : Wed Feb 27 14:52:30 2013
Counts   :
 NEVRAC : 2,204
 NEVRA : 2,204
 NA   : 1,759
 NEVR  : 2,204
 rpm DB : 2,204
 yum DB : 2,204
history stats
```

Yum を使用すると、過去に発生したすべてのトランザクションのサマリーを表示することもできます。**root** で以下の形式のコマンドを実行します。

yum history summary

特定の範囲内でのトランザクションのみを表示するには、以下を入力します。

yum history summary **start_id**..**end_id**

yum history list コマンドと同様に、パッケージの名前または glob 表現を指定することで、特定のパッケージに関するトランザクションのサマリーを表示できます。

yum history summary **glob_expression**…

例9.21 最新のトランザクション 5 件のサマリー

```
~]# yum history summary 1..5
Loaded plugins: langpacks, product-id, subscription-manager
Login user         | Time         | Action(s)    | Altered
-------------------------------------------------------------------------------
Jaromir ... <jhradilek>  | Last day      | Install     |    1
Jaromir ... <jhradilek>  | Last week      | Install     |    1
Jaromir ... <jhradilek>  | Last 2 weeks    | I, U       |    73
System <unset>       | Last 2 weeks    | I, U       |   1107
history summary
```

yum history summary コマンドはすべての形式で、**yum history list** の出力に似た、簡略化された表形式出力を生成します。

上記のように、**yum history list** および **yum history summary** とも、トランザクション向けに設定さ

れています。特定のパッケージに関連するトランザクションのみを表示することができますが、パッケージバージョンのような重要な詳細は表示されません。パッケージに関連するトランザクションをリスト表示するには、**root** で以下のコマンドを実行します。

```
yum history package-list glob_expression&hellip;
```

例9.22 パッケージ履歴の追跡

たとえば、**subscription-manager** および関連パッケージの履歴を調べるには、シェルプロンプトで以下を入力します。

```
~]# yum history package-list subscription-manager\*
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
ID   | Action(s)   | Package
-------------------------------------------------------------------------------
   2 | Updated     | subscription-manager-1.13.22-1.el7.x86_64     EE
   2 | Update      |          1.15.9-15.el7.x86_64     EE
   2 | Obsoleted   | subscription-manager-firstboot-1.13.22-1.el7.x86_64 EE
   2 | Updated     | subscription-manager-gui-1.13.22-1.el7.x86_64   EE
   2 | Update      |            1.15.9-15.el7.x86_64   EE
   2 | Obsoleting  | subscription-manager-initial-setup-addon-1.15.9-15.el7.x86_64 EE
   1 | Install     | subscription-manager-1.13.22-1.el7.x86_64
   1 | Install     | subscription-manager-firstboot-1.13.22-1.el7.x86_64
   1 | Install     | subscription-manager-gui-1.13.22-1.el7.x86_64
history package-list
```

上記の例では、初期のシステムインストール時に **subscription-manager**、**subscription-manager-firstboot**、**subscription-manager-gui** の 3 パッケージがインストールされています。3 つ目のトランザクションでは、これらの全パッケージはバージョン 1.10.11 から 1.10.17 に更新されています。

9.4.2. トランザクションの検証

単一のトランザクションのサマリーを表示するには、**root** で以下の形式で **yum history summary** コマンドを使用します。

```
yum history summary id
```

ここでは、**id** はトランザクションの ID を表します。

特定のトランザクションを詳しく調べる場合は、**root** で以下のコマンドを実行します。

```
yum history info id&hellip;
```

id の引数はオプションです。これを省略する場合は、yum は自動的に最後のトランザクションを使用します。複数のトランザクションを指定する場合は、範囲を指定することもできます。

```
yum history info start_id..end_id
```

例9.23 yum history info の出力例

以下は、2 つのトランザクションに関する出力のサンプルです。それぞれ新しいパッケージを 1 つインストールしています。

```
~]# yum history info 4..5
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
Transaction ID : 4..5
Begin time   : Mon Dec 7 16:51:07 2015
Begin rpmdb  : 1252:d2b62b7b5768e855723954852fd7e55f641fbad9
End time     :     17:18:49 2015 (27 minutes)
End rpmdb    : 1253:cf8449dc4c53fc0cbc0a4c48e496a6c50f3d43c5
User         : Maxim Svistunov <msvistun>
Return-Code  : Success
Command Line  : install tigervnc-server.x86_64
Command Line  : reinstall tigervnc-server
Transaction performed with:
  Installed   rpm-4.11.3-17.el7.x86_64         @rhel-7-server-rpms
  Installed   subscription-manager-1.15.9-15.el7.x86_64 @rhel-7-server-rpms
  Installed   yum-3.4.3-132.el7.noarch         @rhel-7-server-rpms
Packages Altered:
  Reinstall tigervnc-server-1.3.1-3.el7.x86_64 @rhel-7-server-rpms
history info
```

また、トランザクション時に使用された設定オプション、特定のパッケージをインストールしたリポジトリー、その理由などの追加情報も閲覧できます。特定のトランザクションに関して入手可能な追加情報を表示する場合は、**root** としてシェルプロンプトで以下を入力します。

```
yum history addon-info id
```

yum history info と同様に、**id** が指定されていない場合、yum は自動的に最新のトランザクションを使用します。別の方法として、最新のトランザクションを参照するには、**last** キーワードを使用することもできます。

```
yum history addon-info last
```

例9.24 yum history addon-info の出力例

履歴の 4 番目のトランザクションを指定すると、**yum history addon-info** は以下のような出力を返します。

```
~]# yum history addon-info 4
Loaded plugins: langpacks, product-id, subscription-manager
Transaction ID: 4
Available additional history information:
 config-main
 config-repos
 saved_tx

history addon-info
```

yum history addon-info コマンドの出力では、以下の 3 種類の情報が表示されます。

- **config-main**: トランザクション時に使用された yum のグローバルオプション。グローバルオプションの変更方法は 「[main] オプションの設定」 を参照してください。

- **config-repos**: 個々の yum リポジトリー用のオプションです。個々のリポジトリー用のオプションを変更する方法は 「[repository] オプションの設定」 を参照してください。
- **saved_tx**: 別のマシンでトランザクションを繰り返すために **yum load-transaction** コマンドにより利用できるデータです (下記参照)。

選択した種類の追加情報を表示するには、**root** で以下のコマンドを実行してください。

```
yum history addon-info id information
```

9.4.3. トランザクションを元に戻す/繰り返す

トランザクション履歴の確認以外に、**yum history** コマンドは選択したトランザクションを元に戻す、または繰り返す方法を提供します。トランザクションを元に戻すには、**root** で次のコマンドを実行します。

```
yum history undo id
```

特定のトランザクションを繰り返すには、**root** で次のコマンドを実行します。

```
yum history redo id
```

どちらのコマンドでも **last** キーワードを使用して、最新のトランザクションを元に戻す、または繰り返すことができます。

yum history undo コマンドおよび **yum history redo** コマンドのどちらも、トランザクション中に実行したステップを元に戻す、または繰り返すだけである点に注意してください。このトランザクションで新しいパッケージがインストールされた場合に、**yum history undo** コマンドを実行すると、今回インストールしたパッケージがアンインストールされます。逆に、このトランザクションでパッケージがアンインストールされた場合は、このコマンドにより再度インストールされます。またこのコマンドは、(古いパッケージが引き続き利用可能な場合に) 更新済みパッケージをすべて以前のバージョンにダウングレードする試みも行います。

複数の同一システムを管理する場合、yum を使用すると、1 つのシステムでトランザクションを実行して、そのトランザクションの詳細をファイルに格納し、テスト期間の終了後に残りのシステムで同じトランザクションを繰り返すことができます。トランザクションの詳細をファイルに保存するには、**root** でシェルプロンプトに以下を入力します。

```
yum -q history addon-info id saved_tx > file_name
```

このファイルを目的のシステムにコピーしたら、**root** で以下のコマンドを使用してトランザクションを繰り返すことができます。

```
yum load-transaction file_name
```

欠けているパッケージまたは rpmdb バージョンを無視するように **load-transaction** を設定できます。これらの設定オプションの詳細は、**yum.conf**(5) man ページを参照してください。

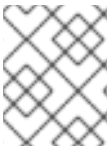
9.4.4. 新しいトランザクション履歴の開始

Yum は単一の SQLite データベースファイルにトランザクション履歴を保存します。新しいトランザクションの履歴を開始するには、**root** で以下のコマンドを実行します。

```
yum history new
```

これにより **/var/lib/yum/history/** ディレクトリーに新しい空のデータベースファイルが作成されます。古いトランザクション履歴は保存されますが、新しいデータベースファイルがディレクトリーにある限りアクセスすることはできません。

9.5. YUM と YUM リポジトリーの設定



注記

専門知識を深めるために、Red Hat System Administration III (RH254) トレーニングコースと RHCSA Rapid Track (RH199) トレーニングコースを受講することを推奨します。

yum および関連ユーティリティーの設定情報は **/etc/yum.conf** に存在します。このファイルには、必須の **[main]** セクションが 1 つあり、ここで全体に影響を与える yum オプションを設定できます。また、**[repository]** セクションを 1 つ以上追加して、リポジトリー固有のオプションを設定することもできます。ただし、**/etc/yum.repos.d/** ディレクトリーにある、新規または既存の **.repo** ファイルに個々のリポジトリーを定義することが推奨されます。**/etc/yum.conf** ファイルの各 **[repository]** セクションで定義した値は、**[main]** セクションに設定した値をオーバーライドします。

このセクションでは以下の方法を紹介します。

- **/etc/yum.conf** 設定ファイルの **[main]** セクションを編集して、yum のグローバルオプションを設定する方法
- **/etc/yum.conf** ファイル、および **/etc/yum.repos.d/** ディレクトリーの **.repo** ファイルの **[repository]** セクションを編集して、個々のレポジトリーにオプションを設定する方法
- 動的バージョンとアーキテクチャーの値が適切に処理されるように **/etc/yum.conf** の yum 変数と **/etc/yum.repos.d/** ディレクトリー内のファイルを使用する方法
- コマンドラインで yum リポジトリーを追加、有効、無効にする方法
- カスタムの yum リポジトリーを設定する方法

9.5.1. [main] オプションの設定

/etc/yum.conf 設定ファイルには、**[main]** セクションが 1 つだけ含まれます。本セクションにあるキー値ペアの中には、yum の動作に影響を与えるものもあれば、yum がリポジトリーを処理する方法に影響を与えるものもあります。

/etc/yum.conf の **[main]** セクションの下に、オプションを多数追加できます。

以下は、**/etc/yum.conf** 設定ファイルのサンプルです。

```
[main]
cachedir=/var/cache/yum/$basearch/$releasever
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
```

```
installonly_limit=3

[comments abridged]

# PUT YOUR REPOS HERE OR IN separate files named file.repo
# in /etc/yum.repos.d
```

以下は、**[main]** セクションで最もよく使用されるオプションです。

assumeyes=value

assumeyes オプションは、yum が重要なアクションに関する確認を行うかどうかを決定します。**value** を、以下のいずれかで置き換えます。
0: (**デフォルト**)。yum は、実行する重要な動作の確認を行います。

1: **yum** は、実行する重要な動作の確認を行いません。**assumeyes=1** を設定すると、yum はコマンドラインオプション **-y** および **--assumeyes** と同じように動作します。

cachedir=directory

このオプションを使用して、yum がキャッシュおよびデータベースファイルを保存するディレクトリーを設定します。**directory** をディレクトリーへの絶対パスで置き換えます。デフォルトでは、yum のキャッシュディレクトリーは **/var/cache/yum/$basearch/$releasever/** です。
yum 変数 **$basearch** および **$releasever** の詳細は、「yum 変数の使用」 を参照してください。

debuglevel=value

このオプションは、yum が生成するデバッグ出力の詳細を指定します。ここでは、**value** は **1** から **10** までの整数になります。**debuglevel** 値を高く設定すると、yum がより詳細なデバッグ出力を表示します。**debuglevel=2** はデフォルトで、**debuglevel=0** がデバッグ出力を無効にします。

exactarch=value

このオプションを使用すると、インストール済みのパッケージを更新する際に、yum が正確なアーキテクチャーを考慮するように設定できます。**value** を以下のいずれかで置き換えます。
0: パッケージの更新時には正しいアーキテクチャーを考慮に入れて実行しません。

1 (**デフォルト値**): パッケージの更新時に正しいアーキテクチャーを考慮します。このように設定すると、yum が、64 ビットアーキテクチャーのシステムにインストールされているパッケージの更新に、32 ビットアーキテクチャーのパッケージを使用しません。

exclude=package_name more_package_names

exclude オプションでは、インストールまたはシステム更新の際にキーワードでパッケージを除外できます。除外する複数のパッケージのリストを表示するには、スペースで区切ったパッケージのリストを引用符で囲みます。ワイルドカードを使用したシェル glob 表現 (***** や **?** など) を使用できます。

gpgcheck=value

gpgcheck オプションを指定して、GPG 著名チェックをパッケージで行う必要があるかどうかを指定します。**value** を以下のいずれかで置き換えます。
0: インストールされるローカルパッケージなど、全リポジトリー内のパッケージでの GPG 署名確認を無効にします。

1 (**デフォルト**): インストールされるローカルパッケージなど、全リポジトリーのすべてのパッケージで GPG 署名確認を有効にします。**gpgcheck** を有効にすると、すべてのパッケージ署名が確認されます。

このオプションが **/etc/yum.conf** ファイルの **[main]** セクションで設定されている場合は、全リポジ

トリーに対して GPG 確認ルールが設定されます。ただし、個々のリポジトリーに **gpgcheck=value** を設定することもできます。つまり、あるリポジトリーで GPG チェックを有効にしつつ、別のレポジトリーで無効にすることができます。対応の **.repo** ファイルの個別のレポジトリーに **gpgcheck=value** を設定すると、**/etc/yum.conf** にデフォルト値がある場合はそれを無効にします。

group_command=value

group_command オプションを指定して、**yum group install**、**yum group upgrade**、**yum group remove** がパッケージグループを処理する方法を指定します。**value** を、以下のいずれかで置き換えます。
simple: パッケージグループのすべてのメンバーをインストールします。以前にインストールされたパッケージのみを更新し、その間にグループに追加されたパッケージはインストールしません。

compat: **simple** に似ていますが、**yum upgrade** は前回の更新以降にグループに追加されたパッケージもインストールします。

オブジェクト: (**デフォルト**)。 このオプションでは、yum は以前にインストールされたグループを追跡し、グループの一部としてインストールされたパッケージと、個別にインストールされたパッケージを区別します。例9.15「LibreOffice パッケージグループの情報表示」 を参照してください。

group_package_types=package_type more_package_types

yum group install コマンドが呼び出されたときにインストールされるパッケージのタイプ (**オプション**、**デフォルト**、**必須**) を指定することができます。**default** および **mandatory** のパッケージタイプがデフォルトで選択されます。

history_record=value

このオプションを使用すると、yum はトランザクション履歴を記録します。**value** を、以下のいずれかで置き換えます。
0: yum はトランザクションの履歴エントリーを**記録しません**。

1 (**デフォルト値**): yum はトランザクションの履歴エントリーを記録します。この操作により、ある程度のディスク領域が使用され、トランザクションの時間が少し長くなりますが、過去の操作に関する多くの情報が提供されます。これは、**yum history** で表示できます。**history_record=1** がデフォルトです。

yum history コマンドの詳細は、「トランザクション履歴の活用」 を参照してください。

注記

yum は履歴記録を使用して、yum 以外で行われた **rpmdb** データベースへの変更を検出します。変更が検出されると、yum は警告を表示し、**rpmdb** の変更によって起こり得る問題を自動的に検索します。**history_record** がオフになっていると、yum はこのような変更を検出できず、自動チェックは実行されません。

installonlypkgs=space separated list of packages

ここでは、yum で**インストール**を行い、**更新**を行わないパッケージのリストをスペースで区切って提供できます。デフォルトでインストールのみに設定されているパッケージのリストは、**yum.conf**(5) man ページを参照してください。
installonlypkgs ディレクティブを **/etc/yum.conf** に追加する場合は、**yum.conf**(5) の **installonlypkgs** セクションにリスト表示されているものを含め、インストールのみのパッケージを**すべて**リストするようにしてください。特に (デフォルトで) カーネルパッケージが常に **installonlypkgs** でリスト表示され、**installonly_limit** の値が常に **2** よりも大きく設定され、デフォルトが起動に失敗した場合にバックアップカーネルを常に利用可能にするようにしてください。

installonly_limit=value

このオプションは、**installonlypkgs** ディレクティブにリストされている多くのパッケージを同時にインストールできる数を設定します。**installonlypkgs** にリスト表示されている単一のパッケージに同時にインストールできるようにバージョンの最大数を示す整数に**値**を置き換えます。
installonlypkgs ディレクティブのデフォルトには複数のカーネルパッケージが含まれています。そのため、**installonly_limit** の値を変更すると、インストール済みの単一のカーネルパッケージのバージョンの最大数にも影響することに注意してください。**/etc/yum.conf** にリスト表示されるデフォルト値は **installonly_limit=3** で、使用できる最小値は **installonly_limit=2** です。

installonly_limit=1 に設定すると yum によって実行中のカーネルが削除されるため、設定することはできません。**installonly_limit=1** を使用すると、yum は失敗します。

installonly_limit=2 を使用すると、1つのバックアップカーネルが利用可能になります。ただし、2つのバックアップカーネルを利用できるようにするためにも、デフォルト設定 **installonly_limit=3** を使用し続けることが推奨されます。

keepcache=value

keepcache オプションはインストールに成功した後に、yum がヘッダーのキャッシュを維持するかどうかを決めます。**値**は、以下のいずれかになります。
0: (**デフォルト**)。インストールの成功後は、ヘッダーとパッケージのキャッシュを保持しません。

1: インストールの成功後も、キャッシュを保持します。

logfile=file_name

ログ出力の場所を指定するため、**file_name** を、yum がログ出力を書き込むファイルへの絶対パスで置き換えます。デフォルトでは、yum は **/var/log/yum.log** にログを記録します。

max_connenctions=number

ここでの **value** は、同時接続の最大数を表します。デフォルトは 5 です。

multilib_policy=value

multilib_policy は、複数のアーキテクチャーバージョンがパッケージのインストールに利用できる場合に、インストール動作を設定します。ここで、**value** は以下を表します。
best: このシステムに最適なアーキテクチャーをインストールします。たとえば、AMD64 システムに **multilib_policy=best** を設定すると、yum は全パッケージの 64 ビットバージョンをインストールします。

all: 常に全パッケージ用の可能なあらゆるアーキテクチャーをインストールします。たとえば、AMD64 システムで **multilib_policy** を **all** に設定すると、yum は i686 および AMD64 のパッケージが利用可能であれば両方のバージョンをインストールします。

obsoletes=value

obsoletes オプションでは、更新中に廃止プロセス論理を有効化します。あるパッケージのスペックファイルで、別のパッケージを**廃止する**ことを宣言すると、廃止宣言したパッケージのインストール時に、廃止宣言されたパッケージが、廃止宣言したパッケージに置き換えられます。たとえば、パッケージ名が変更された場合などに廃止が宣言されます。**value** を、以下のいずれかで置き換えます。
0: 更新の実行時に yum の廃止処理ロジックを無効にします。

1: (**デフォルト**)。更新の実行時に yum の廃止処理ロジックを有効にします。

plugins=value

これは、yum プラグインを有効または無効にするグローバルスイッチです。**value** は以下のいずれかになります。

0: yum のプラグインをグローバルで無効にします。

重要

一部のプラグインは重要な yum サービスを提供するため、すべてのプラグインを無効にすることは推奨されません。特に、**product-id** および **subscription-manager** プラグインは、証明書ベースの **Content Delivery Network** (CDN) のサポートを提供します。プラグインをグローバルで無効にするオプションは便利なオプションとして提供されていますが、通常は yum に潜在的な問題があると判断された場合にのみ使用することが推奨されます。

1 (**デフォルト値**): すべての yum プラグインを全体的に有効にします。**plugins=1** では、そのプラグインの設定ファイルに **enabled=0** を設定して、特定の yum プラグインを無効にすることができます。

yum の各種プラグインの詳細は 「yum のプラグイン」 を参照してください。プラグインの制御に関する詳細は 「yum プラグインを有効、設定、および無効にする方法」 を参照してください。

reposdir=directory

ここでの directory は **.repo** ファイルがあるディレクトリーへの絶対パスです。すべての **.repo** ファイルには、リポジトリー情報 (**/etc/yum.conf** の **[repository]** セクションと類似) が含まれています。yum は **.repo** ファイルおよび **/etc/yum.conf** ファイルの **[repository]** セクションからすべてのリポジトリー情報を収集し、トランザクションに使用するリポジトリーのマスター一覧を作成します。**reposdir** が設定されていない場合は、yum はデフォルトのディレクトリーである **/etc/yum.repos.d/** を使用します。

retries=value

このオプションは、エラーを返す前に yum がファイルの取得を試行する回数を設定します。**値** は整数 **0** 以上です。値を **0** に設定すると、yum はその試行を何度も続けます。デフォルト値は **10** です。

利用可能な **[main]** オプションの詳細なリストは、man ページの **yum.conf**(5) の **[main] OPTIONS** セクションを参照してください。

9.5.2. [repository] オプションの設定

[repository] セクションでは、個別の yum リポジトリーを定義できます。**repository** は、**my_personal_repo** (スペースは使用不可) などの一意のリポジトリー ID になります。競合を回避するために、カスタムリポジトリーには、Red Hat リポジトリーで使用されている名前を使用しないでください。

[repository] セクションでは、少なくとも以下の例のような形式が必要になります。

```
[repository]
name=repository_name
baseurl=repository_url
```

すべての **[repository]** セクションには、以下のディレクティブを含める必要があります。

name=repository_name

repository_name は、人間が判読可能な、リポジトリーを説明する文字列になります。

baseurl=repository_url

repository_url を、リポジトリーの repodata ディレクトリーが置かれているディレクトリーへの URL に置き換えます。

- リポジトリーが HTTP にある場合は、**http://path/to/repo**を使用します。
- リポジトリーが FTP にある場合は、**ftp://path/to/repo**を使用します。
- リポジトリーがマシンのローカルにある場合は、**file:///path/to/local/repo**を使用します。
- 特定のオンラインリポジトリーでベーシック HTTP 認証が必要な場合は、**username:password@link** として URL にプリペンドして、ユーザー名とパスワードを指定できます。たとえば、**http://www.example.com/repo/** のリポジトリーで、ユーザー名 user とパスワード password が必要な場合、**baseurl** リンクは **http://user:/repo/** として指定できます。
  通常この URL は以下のような HTTP リンクになります。

  ```
  baseurl=http://path/to/repo/releases/$releasever/server/$basearch/os/
  ```

  yum は、常に URL の **$releasever**、**$arch**、**$basearch** 変数をデプロイメントする点に注意して下さい。yum 変数の詳細は 「yum 変数の使用」 を参照してください。

以下のような便利な **[repository]** ディレクティブもあります。

enabled=value

このオプションを使用すれば、yum が特定のリポジトリーを使用するか無視するかを簡単に設定できます。**value** は、以下のいずれかになります。
0: 更新およびインストールの実行時には、パッケージソースとしてこのリポジトリーを含めません。これはリポジトリーを迅速に有効または無効にする簡単な方法です。更新またはインストールには無効にしているリポジトリーから、単一パッケージが欲しい場合に便利です。

1: パッケージソースとしてこのリポジトリーを含めます。

リポジトリーのオンとオフは、**--enablerepo=repo_name** または **--disablerepo=repo_name** オプションを **yum** に渡すか、PackageKit ユーティリティーの **ソフトウェアの追加/削除** ウィンドウから実行できます。

async=value

リポジトリーパッケージの並行ダウンロードを制御します。**値**は、以下のいずれかになります。
auto (**デフォルト**): 可能な場合、並行ダウンロードを使用します。つまり、失敗を回避するために **yum** はプラグインが作成したリポジトリーについては自動的にこれを無効にします。

on: リポジトリーの並行ダウンロードが有効になります。

off: リポジトリーの並行ダウンロードが無効になります。

他にも多くの **[repository]** オプションがあります。このうちのいくつかは、**[main]** オプションと同一形式、同一機能になります。オプションのリストは、**yum.conf**(5) man ページの [repository] OPTIONS セクションを参照してください。

例9.25 /etc/yum.repos.d/redhat.repo ファイルのサンプル

以下は、**/etc/yum.repos.d/redhat.repo** ファイルのサンプルです。

```
#
# Red Hat Repositories
# Managed by (rhsm) subscription-manager
#

[red-hat-enterprise-linux-scalable-file-system-for-rhel-6-entitlement-rpms]
name = Red Hat Enterprise Linux Scalable File System (for RHEL 6 Entitlement) (RPMs)
baseurl = https://cdn.redhat.com/content/dist/rhel/entitlement-
6/releases/$releasever/$basearch/scalablefilesystem/os
enabled = 1
gpgcheck = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
sslverify = 1
sslcacert = /etc/rhsm/ca/redhat-uep.pem
sslclientkey = /etc/pki/entitlement/key.pem
sslclientcert = /etc/pki/entitlement/11300387955690106.pem

[red-hat-enterprise-linux-scalable-file-system-for-rhel-6-entitlement-source-rpms]
name = Red Hat Enterprise Linux Scalable File System (for RHEL 6 Entitlement) (Source RPMs)
baseurl = https://cdn.redhat.com/content/dist/rhel/entitlement-
6/releases/$releasever/$basearch/scalablefilesystem/source/SRPMS
enabled = 0
gpgcheck = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
sslverify = 1
sslcacert = /etc/rhsm/ca/redhat-uep.pem
sslclientkey = /etc/pki/entitlement/key.pem
sslclientcert = /etc/pki/entitlement/11300387955690106.pem

[red-hat-enterprise-linux-scalable-file-system-for-rhel-6-entitlement-debug-rpms]
name = Red Hat Enterprise Linux Scalable File System (for RHEL 6 Entitlement) (Debug RPMs)
baseurl = https://cdn.redhat.com/content/dist/rhel/entitlement-
6/releases/$releasever/$basearch/scalablefilesystem/debug
enabled = 0
gpgcheck = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
sslverify = 1
sslcacert = /etc/rhsm/ca/redhat-uep.pem
sslclientkey = /etc/pki/entitlement/key.pem
sslclientcert = /etc/pki/entitlement/11300387955690106.pem
```

9.5.3. yum 変数の使用

yum コマンドおよびすべての yum 設定ファイル (つまり **/etc/yum.conf** および **/etc/yum.repos.d/** ディレクトリー内のすべての **.repo** ファイル) 内で、以下の組み込み変数を使用および参照できます。

$releasever

この変数を使用すると、Red Hat Enterprise Linux のリリースバージョンを参照できます。yum は **/etc/yum.conf** 設定ファイルにある **distroverpkg=value** の行から **$releasever** の値を取得します。**/etc/yum.conf** にそのような行がない場合、yum は、**redhat-release** ファイルを提供する **redhat-releaseproduct** パッケージからバージョン番号を取得することにより、正しい値を推測します。

$arch

この変数を使用して、Python の **os.uname()** 関数を呼び出す時に返り値としてシステムの CPU アーキテクチャーを参照できます。**$arch** の有効な値は、**i586**、**i686**、**x86_64** です。

$basearch

$basearch を使用すると、システムのベースアーキテクチャーを参照できます。たとえば、i686 および i586 の両マシンには **i386** のベースアーキテクチャーがあり、AMD64 および Intel 64 の両マシンには **x86_64** のベースアーキテクチャーがあります。

$YUM0-9

これら 10 個の変数は、それぞれ同じ名前を持つシェル環境変数の値に置換されます。これら変数のいずれかが (たとえば **/etc/yum.conf** で) 参照され、同じ名前を持つシェル環境変数が存在しない場合、設定ファイルの変数は置換されません。

カスタム変数の定義、既存の変数値の上書きを行うには、**/etc/yum/vars/** ディレクトリー内に変数と同じ名前を持つファイルを作成して (**$** 記号はなし)、1 行目に希望する値を追加します。

たとえば多くの場合、リポジトリーの詳細にはオペレーティングシステムの名前が含まれます。**$osname** と呼ばれる新しい変数を定義するには、1 行目に Red Hat Enterprise Linux の名前を持つ新しいファイルを作成して、**/etc/yum/vars/osname** として保存します。

```
~]# echo "Red Hat Enterprise Linux 7" > /etc/yum/vars/osname
```

.repo ファイルでは、Red Hat Enterprise Linux 7 の代わりに以下を使用することができます。

```
name=$osname $releasever
```

9.5.4. 現在の設定の表示

yum グローバルオプションの現在の値 (つまり **/etc/yum.conf** ファイルの **[main]** セクションで指定されたオプション) を表示するには、コマンドラインのオプションなしで **yum-config-manager** コマンドを実行します。

```
yum-config-manager
```

別の設定セクションの内容をリスト表示するには、以下の形式でコマンドを実行します。

```
yum-config-manager section&hellip;
```

glob 表現を使用して、適合する全セクションの設定を表示することもできます。

```
yum-config-manager glob_expression&hellip;
```

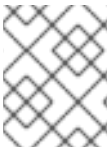
例9.26 main セクションの設定を表示する

すべての設定オプションとそれらに対応する値をリスト表示するには、シェルプロンプトで以下を入力します。

```
~]$ yum-config-manager main \*
Loaded plugins: langpacks, product-id, subscription-manager
================================== main ==================================
[main]
alwaysprompt = True
assumeyes = False
bandwith = 0
```

```
bugtracker_url = https://bugzilla.redhat.com/enter_bug.cgi?
product=Red%20Hat%20Enterprise%20Linux%206&component=yum
cache = 0
[output truncated]
```

9.5.5. yum リポジトリーの追加、有効化、および無効化



注記

専門知識を深めるには、Red Hat System Administration III (RH254) トレーニングコースの受講を推奨します。

「[repository] オプションの設定」 では、yum リポジトリーの定義に使用できるさまざまなオプションを説明しています。本セクションでは、**yum-config-manager** コマンドを使用してリポジトリーを追加、もしくは有効または無効にする方法を説明します。

重要

システムが Red Hat Subscription Management で証明書ベースの **Content Delivery Network** (CDN) に登録されている場合、**/etc/yum.repos.d/redhat.repo** ファイル内のリポジトリー管理には **Red Hat サブスクリプションマネージャー** ツールが使用されます。

yum リポジトリーの追加

新しいリポジトリーを定義するには、**[repository]** セクションを、**/etc/yum.conf** ファイルか、**/etc/yum.repos.d/** ディレクトリーの **.repo** ファイルに追加します。このディレクトリーにある、**.repo** ファイル拡張子が付いたすべてのファイルを、yum が読み取ります。リポジトリーは、**/etc/yum.conf** ではなく、ここに定義することが推奨されます。

警告

ソフトウェアパッケージを、Red Hat の認証ベース **Content Delivery Network** (CDN) 以外の未検証または信頼できないソフトウェアソースから取得してインストールする場合には、セキュリティー上のリスクが伴います。セキュリティー、安定性、互換性、保全性に関する問題につながる恐れがあります。

yum リポジトリーは、一般的に **.repo** ファイルを提供します。このようなリポジトリーをシステムに追加して有効にするには、**root** で以下のコマンドを実行します。

```
yum-config-manager --add-repo repository_url
```

repository_url は、**.repo** ファイルへのリンクになります。

例9.27 example.repo の追加

http://www.example.com/example.repo にあるリポジトリーを追加するには、シェルプロンプトで以下を入力します。

```
~]# yum-config-manager --add-repo http://www.example.com/example.repo
Loaded plugins: langpacks, product-id, subscription-manager
adding repo from: http://www.example.com/example.repo
grabbing file http://www.example.com/example.repo to /etc/yum.repos.d/example.repo
example.repo                                | 413 B   00:00
repo saved to /etc/yum.repos.d/example.repo
```

yum リポジトリーの有効化

特定のリポジトリーを有効にするには、**root** で以下のコマンドを入力します。

```
yum-config-manager --enable repository&hellip;
```

ここでの **repository** は一意のリポジトリー ID になります (利用可能なリポジトリー ID をリスト表示するには **yum repolist all** を使用)。別の方法では、glob 表現を使用すると、一致するすべてのリポジトリーを有効にできます。

```
yum-config-manager --enable glob_expression&hellip;
```

例9.28 /etc/yum.conf のカスタムセクションで定義されるリポジトリーを有効にする

[example]、**[example-debuginfo]**、**[example-source]** セクション内で定義されたリポジトリーを有効にするには、以下を入力します。

```
~]# yum-config-manager --enable example\*
Loaded plugins: langpacks, product-id, subscription-manager
============================== repo: example ==============================
[example]
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/7Server
baseurl = http://www.example.com/repo/7Server/x86_64/
cache = 0
cachedir = /var/cache/yum/x86_64/7Server/example
[output truncated]
```

例9.29 すべてのリポジトリーの有効化

/etc/yum.conf ファイルと **/etc/yum.repos.d/** ディレクトリーで定義されたすべてのリポジトリーを有効にするには、以下のコマンドを入力します。

```
~]# yum-config-manager --enable \*
Loaded plugins: langpacks, product-id, subscription-manager
============================== repo: example ==============================
[example]
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/7Server
baseurl = http://www.example.com/repo/7Server/x86_64/
cache = 0
cachedir = /var/cache/yum/x86_64/7Server/example
[output truncated]
```

成功すると、**yum-config-manager --enable** コマンドは現在のリポジトリー設定を表示します。

yum リポジトリーの無効化

yum リポジトリーを無効にするには、**root** で以下のコマンドを実行します。

```
yum-config-manager --disable repository&hellip;
```

ここでの **repository** は一意のリポジトリー ID になります (利用可能なリポジトリー ID をリスト表示するには **yum repolist all** を使用)。**yum-config-manager --enable** と同様に、glob 表現を使用して、一致するすべてのリポジトリーを同時に無効にできます。

```
yum-config-manager --disable glob_expression&hellip;
```

例9.30 すべてのリポジトリーの無効化

/etc/yum.conf ファイルと **/etc/yum.repos.d/** ディレクトリーで定義されたすべてのリポジトリーを無効にするには、以下のコマンドを入力します。

```
~]# yum-config-manager --disable \*
Loaded plugins: langpacks, product-id, subscription-manager
============================== repo: example ==============================
[example]
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/7Server
baseurl = http://www.example.com/repo/7Server/x86_64/
cache = 0
cachedir = /var/cache/yum/x86_64/7Server/example
[output truncated]
```

成功すると、**yum-config-manager --disable** コマンドは現在の設定を表示します。

9.5.6. yum リポジトリーの作成

yum リポジトリーを設定するには、以下を行います。

1. **createrepo** パッケージをインストールします。

   ```
   # yum install createrepo
   ```

2. 新しいリポジトリーのパッケージをすべて 1 つのディレクトリー (**/tmp/local_repo/** など) にコピーします。

   ```
   cp /your/packages/*.rpm /tmp/local_repo/
   ```

3. リポジトリーの実行を作成するには、以下を行います。

   ```
   createrepo /tmp/local_repo/
   ```

   これにより、yum リポジトリーから必要なメタデータが、新たに作成したサブディレクトリー **repodata** に作成されます。

このリポジトリーを yum で利用できるようになりました。このリポジトリーは HTTP または FTP プロトコルで共有でき、ローカルマシンから直接参照できます。yum リポジトリーを設定する方法は 「[repository] オプションの設定」 セクションを参照してください。

注記

リポジトリーの URL を構築するには、**/mnt/local_repo/repodata** ではなく **/mnt/local_repo** を参照してください。実際の yum パッケージは **/mnt/local_repo** にあります。

9.5.6.1. 作成した yum リポジトリーへのパッケージの追加

作成した yum リポジトリーにパッケージを追加するには、以下を行います。

1. リポジトリーディレクトリー (**/tmp/local_repo/** など) に新しいパッケージをコピーします。

```
cp /your/packages/*.rpm /tmp/local_repo/
```

2. メタデータで新たに追加されたパッケージを反映するには、以下を実行します。

```
createrepo --update /tmp/local_repo/
```

3. 任意: 新たに更新したリポジトリーで yum コマンドを使用している場合は、以下のコマンドを実行します。

```
yum clean expire-cache
```

9.5.7. Optional および Supplementary リポジトリーの追加

Optional および Supplementary のサブスクリプションチャンネルでは、オープンソースのライセンス付きソフトウェア (Optional チャンネル) と商用ライセンス付きソフトウェア (Supplementary チャンネル) をカバーする Red Hat Enterprise Linux 用の追加ソフトウェアパッケージが提供されます。

Optional および Supplementary チャンネルをサブスクライブする前に、対象範囲の詳細 を参照してください。これらのチャンネルからパッケージをインストールする場合は、Red Hat カスタマーポータルの記事 証明書ベースの管理を使用して、Optional および Supplementary チャンネル、-devel パッケージにアクセスする方法 で説明されている手順を行ってください。

9.6. YUM のプラグイン

yum は、その操作を拡張し、強化するプラグインを提供します。特定のプラグインが、デフォルトでインストールされています。yum コマンドを呼び出すたびに、読み込まれ、アクティブになっているプラグインがあれば、**yum** がそれを通知します。以下に例を示します。

```
~]# yum info yum
Loaded plugins: langpacks, product-id, subscription-manager
[output truncated]
```

Loaded plugins に続くプラグイン名は **--disableplugin=plugin_name** オプションに指定できる名前です。

9.6.1. yum プラグインを有効、設定、および無効にする方法

yum プラグインを有効にする場合は、**/etc/yum.conf** の **[main]** セクションに **plugins=** で始まる行を追加し、その値を **1** にします。

```
plugins=1
```

すべてのプラグインを無効にするには、この行を **plugins=0** に変更します。

重要

一部のプラグインは重要な yum サービスを提供するため、すべてのプラグインを無効にすることは推奨されません。その中でも **product-id** プラグインおよび **subscription-manager** プラグインは、証明書ベースの **Content Delivery Network** (CDN) への対応に必要です。プラグインをグローバルで無効にするオプションは便利なオプションとして提供されていますが、通常は yum に潜在的な問題があると判断された場合にのみ使用することが推奨されます。

すべてのインストール済みプラグインには、**/etc/yum/pluginconf.d/** ディレクトリーにそれぞれの設定ファイルがあります。このファイルに、プラグイン固有のオプションを設定できます。たとえば、以下のように **aliases** プラグインの **aliases.conf** 設定ファイルがあるとします。

```
[main]
enabled=1
```

/etc/yum.conf ファイルと同様、プラグイン設定ファイルには常に **[main]** セクションが含まれます。このセクションでは、**enabled=** オプションで、**yum** コマンドを実行する際にプラグインを有効にするかどうかを制御します。このオプションがファイルに含まれていない場合は手動で追加できます。

/etc/yum.conf に **enabled=0** を設定してすべてのプラグインを無効にすると、個々の設定ファイルで有効かどうかに関わらず、すべてのプラグインが無効になります。

1 つの **yum** コマンドで yum プラグインをすべて無効にする場合は、**--noplugins** オプションを使用します。

1 つの **yum** コマンドで、1 つ以上の yum プラグインを無効にする場合は、そのコマンドに **--disableplugin=plugin_name** オプションを追加します。たとえば、システムの更新中に **aliases** プラグインを無効にするには、以下を入力します。

```
~]# yum update --disableplugin=aliases
```

--disableplugin= オプションに指定したプラグイン名は、**yum** コマンドの出力の **Loaded プラグイン** 行の後にリスト表示される名前と同じです。名前をコンマで区切ることにより、複数のプラグインを無効にすることができます。さらに、glob 表現を使用して、複数のプラグイン名に一致したり、長いプラグイン名を短くすることができます。

```
~]# yum update --disableplugin=aliases,lang*
```

9.6.2. 追加の Yum プラグインのインストール

yum プラグインは通常、 **yum-plugin-plugin_name** 規則に従いますが、常に **kabi** プラグインを提供するパッケージは **kabi-yum-plugins** という名前です。yum プラグインは、他のパッケージをインストールするのと同じ方法でインストールできます。たとえば、**yum-aliases** プラグインをインストールするには、シェルプロンプトで次のコマンドを実行します。

```
~]# yum install yum-plugin-aliases
```

9.6.3. yum プラグインの使用方法

以下では、便利な yum プラグインの説明と使用方法を紹介しています。プラグインは名前で表示されており、括弧内はパッケージ名になります。

search-disabled-repos (subscription-manager)

search-disabled-repos プラグインを使用すると、依存関係を解決するために無効なリポジトリーを一時的または永久的に有効にできます。このプラグインが有効な場合は、依存関係の解決に失敗して Yum がパッケージのインストールに失敗したときに、無効なリポジトリーを一時的に有効し、再試行することが提示されます。インストールが成功した場合、Yum は使用されているリポジトリーを永久的に有効にすることも提示します。プラグインは **subscription-manager** により管理されているリポジトリーとのみ連携し、カスタムリポジトリーとは連携しません。

重要

yum が **--assumeyes** または **-y** オプションで実行されるか、**/etc/yum.conf** で **assumeyes** ディレクティブが有効になっている場合、プラグインは、確認を求めるプロンプトなしに、一時的に、そして永続的に無効なリポジトリーを有効にします。この結果、有効にしたくないリポジトリーが有効になるといった問題が発生することがあります。

search-disabled-repos プラグインを設定するには、**/etc/yum/pluginconf.d/search-disabled-repos.conf** にある設定ファイルを編集します。**[main]** セクションで使用できるディレクティブのリストについては、以下の表を参照してください。

表9.3 サポートされている search-disabled-repos.conf ディレクティブ

ディレクティブ	詳細
enabled=value	プラグインを有効または無効にできます。**value** は **1** (有効) または **0** (無効) にする必要があります。プラグインはデフォルトで有効です。
notify_only=value	プラグインの動作を通知のみに制限できます。**value** は **1** (Yum の動作の変更なしで通知のみ) または **0** (Yum の動作の変更) のいずれかにする必要があります。デフォルトでは、プラグインはユーザーへの通知のみを行います。
ignored_repos=repositories	プラグインで有効でないリポジトリーを指定できます。

kabi (kabi-yum-plugins)

kabi プラグインは、ドライバー更新パッケージが公式の Red Hat **kernel Application Binary Interface** (**kABI**) と適合するかどうかを確認します。このプラグインが有効な状態で、ユーザーがホワイトリストにないカーネルシンボルを使用するパッケージのインストールを試行する場合は、警告メッセージがシステムログに書き込まれます。さらには、プラグインを enforcing モードで実行するように設定すると、そうしたパッケージがインストールされないようにできます。

kabi プラグインを設定するには、**/etc/yum/pluginconf.d/kabi.conf** にある設定ファイルを編集します。**[main]** セクションに使用できるディレクティブのリストを以下の表に示します。

表9.4 サポートされている kabi.conf ディレクティブ

ディレクティブ	詳細
enabled=value	プラグインを有効または無効にできます。value は **1** (有効) または **0** (無効) にする必要があります。インストール時には、プラグインはデフォルトで有効です。
whitelists=directory	サポートされているカーネルシンボルを持つファイルがある directory を指定できます。デフォルトでは、kabi プラグインは kernel-abi-whitelists パッケージ (**/usr/lib/modules/kabi-rhel70/** ディレクトリー) が提供するファイルを使用します。
enforce=value	enforcing モードを有効または無効にできます。value は **1** (有効) または **0** (無効) にする必要があります。デフォルトでは、このオプションはコメントアウトされ kabi プラグインは警告メッセージのみを表示します。

product-id (subscription-manager)

product-id プラグインは、Content Delivery Network (コンテンツ配信ネットワーク) からインストールされた製品の製品識別証明書を管理します。**product-id** プラグインはデフォルトでインストールされています。

langpacks (yum-langpacks)

langpacks プラグインは、インストールされているすべてのパッケージ用に選択された言語のロケールパッケージを検索するために使用します。**langpacks** プラグインはデフォルトでインストールされます。

エイリアス (yum-plugin-aliases)

aliases プラグインは、**yum** のエイリアスの設定と使用を可能にする **alias** コマンドラインオプションを追加します。

yum-changelog (yum-plugin-changelog)

yum-changelog プラグインは、更新前後にパッケージ変更ログの表示を有効にする **--changelog** コマンドラインオプションを追加します。

yum-tmprepo (yum-plugin-tmprepo)

yum-tmprepo プラグインは、リポジトリーファイルの URL を取り扱う **--tmprepo** コマンドラインオプションを追加し、1 つのトランザクションに対してのみダウンロードおよび有効化します。このプラグインはリポジトリーの安全な一時的使用を確保します。デフォルトでは、gpg 確認を無効にしません。

yum-verify (yum-plugin-verify)

yum-verify プラグインは、システム上の検証データを表示するための **verify**、**verify-rpm**、**verify-all** コマンドラインオプションを追加します。

yum-versionlock (yum-plugin-versionlock)

yum-versionlock プラグインは選択されたパッケージの他のバージョンを除外し、パッケージが最新バージョンに更新されることを防ぎます。**versionlock** コマンドラインオプションを使用すると、ロックされたパッケージを表示して、編集できます。

9.7. YUM-CRON を使用したパッケージデータベースの自動更新および更新のダウンロード

yum-cron サービスがパッケージの更新を自動的に確認してダウンロードします。**yum-cron** が提供する cron ジョブは、yum-cron パッケージのインストール後すぐにアクティブになります。また、**yum-cron** サービスは、ダウンロードした更新を自動的にインストールできます。

yum-cron サービスのデフォルト設定では、以下を行います。

- 1時間ごとに yum cache のメタデータを更新。
- yum cache で保留になっているパッケージの更新を1日1回ダウンロード。リポジトリーで新しいパッケージが利用可能になると、メールが送信されます。詳細は 「任意のメール通知の設定」 の章を参照してください。

yum-cron サービスには設定ファイルが 2 つあります。

/etc/yum/yum-cron.conf
日次タスクの場合。

/etc/yum/yum-cron-hourly.conf
1時間ごとのタスクの場合。

9.7.1. 更新の自動インストールの有効化

ダウンロードした更新の自動インストールを有効にするには、日次設定ファイル (日次インストールの場合) または1時間単位の設定ファイル (1時間ごとのインストールの場合) で **apply_updates** を以下のように設定します。

```
apply_updates = yes
```

9.7.2. 任意のメール通知の設定

デフォルトでは、**yum-cron** サービスは **cron** を使用して、実行したコマンドの出力を含むメール送信を行います。このメールは **cron** 設定に従って送信されますが、通常はローカルのスーパーユーザー、および **/var/spool/mail/root** ファイルに設定されているユーザーに送信されます。

すべての **cron** ジョブに影響する設定とは異なり、特定のメール設定を使用できます。ただし、このメール設定では TLS および全体のメールの組み込みロジックはサポートされません。

yum-cron の組み込みメール通知を有効にするには、以下を実行します。

1. 選択した **yum-cron** 設定ファイルを開きます。

   /etc/yum/yum-cron.conf
   日次タスクの場合。

   /etc/yum/yum-cron-hourly.conf
   1時間ごとのタスクの場合。

2. **[emitters]** セクションで、以下のオプションを設定します。

```
emit_via = email
```

3. 必要に応じて、**email_from**、**email_to**、**email_host** オプションを指定します

9.7.3. 個別のリポジトリーの有効化または無効化

yum-cron は、リポジトリーの特定の設定をサポートしません。**yum** ではなく、**yum-cron** で個別のリポジトリーを有効または無効にするには、以下の手順を行ってください。

1. システムの任意の場所に空のリポジトリー設定ディレクトリーを作成します。
2. **/etc/yum.repos.d/** ディレクトリーから、新たに作成したこのディレクトリーに設定ファイルをすべてコピーします。
3. **/etc/yum.repos.d/** 内の各 **.repo** 設定ファイルで、以下のように **enabled** オプションを設定します。

   enabled = 1

   リポジトリーを有効にするには、以下を行います。

   enabled = 0

   リポジトリーを無効にするには、以下を行います。

4. 選択した **yum-cron** 設定ファイルの最後に、新たに作成したリポジトリーのディレクトリーを指定する以下のオプションを追加します。

```
reposdir=/path/to/new/reposdir
```

9.7.4. Yum-cron 設定のテスト

次に予定されている **yum-cron** タスクを待たずに **yum-cron** 設定をテストするには、以下を行います。

1. 選択した **yum-cron** 設定ファイルを開きます。

   /etc/yum/yum-cron.conf

   日次タスクの場合。

   /etc/yum/yum-cron-hourly.conf

   1時間ごとのタスクの場合。

2. 以下のように、選択した設定ファイルに **random_sleep** オプションを設定します。

```
random_sleep = 0
```

3. 設定を保存します。

```
# yum-cron /etc/yum/yum-cron.conf
# yum-cron /etc/yum/yum-cron-hourly.conf
```

9.7.5. yum-cron メッセージの無効

yum-cron メッセージを完全に無効にすることはできませんが、優先度が重大なメッセージだけに制限することはできます。メッセージを制限するには、以下を行います。

1. 選択した **yum-cron** 設定ファイルを開きます。

   /etc/yum/yum-cron.conf
   日次タスクの場合。

   /etc/yum/yum-cron-hourly.conf
   1時間ごとのタスクの場合。

2. 設定ファイルの **[base]** セクションに以下のオプションを設定します。

```
debuglevel = -4
```

9.7.6. パッケージの自動削除

yum-cron サービスは、**yum clean all** コマンドと同様に、パッケージを削除する設定オプションはサポートしていません。パッケージを自動的に削除するには、実行可能なシェルスクリプトとして cron ジョブを作成できます。

1. **/etc/cron.daily/** ディレクトリーにシェルスクリプトを作成し、以下を追加します。

```
#!/bin/sh
yum clean all
```

2. スクリプトを実行可能にします。

```
# chmod +x /etc/cron.daily/script-name.sh
```

9.8. 関連情報

Red Hat Enterprise Linux でソフトウェアパッケージを管理する方法についての詳細情報は、下記のリソースを参照してください。

インストールされているドキュメント

- **yum**(8): yum コマンドラインユーティリティーの man ページには、サポートされるオプションおよびコマンドの完全なリストを提供します。
- **yumdb**(8): **yumdb** コマンドラインユーティリティーの man ページでは、このツールを使用してクエリーを行い、必要であれば yum データベースを変更する方法が説明されています。
- **yum.conf**(5): **yum.conf** の man ページでは、利用できる yum 設定オプションが説明されています。
- **yum-utils**(1): **yum-utils** の man ページでは、yum 設定の管理、リポジトリーの操作、yum データベースの作業を行う追加ユーティリティーのリスト表示と簡単な説明が提供されます。

オンラインリソース

- Yum Guides: プロジェクトホームページ上の **Yum Guides** では、追加のドキュメンテーションのリンクがあります。
- Red Hat Customer Portal Labs: Red Hat Customer Portal Labs で Yum Repository Configuration Helper をご利用になれます。

関連項目

- 6章 *権限の取得* では、**su** および **sudo** コマンドを使用して管理者権限を取得する方法を説明しています。

パート IV. インフラストラクチャーサービス

ここでは、サービスおよびデーモンの設定方法と、Red Hat Enterprise Linux マシンへのリモートアクセスを可能にする方法を提供します。

第10章 SYSTEMD によるサービス管理

10.1. SYSTEMD の概要

systemd は、Linux オペレーティングシステム用のシステムおよびサービスのマネージャーです。SysV init スクリプトと後方互換するように設計されており、システム起動時のシステムサービスの並行スタートアップや、デーモンのオンデマンドのアクティベーション、依存関係ベースのサービス制御論理などの多くの機能を提供します。Red Hat Enterprise Linux 7 では、systemd は Upstart に代わるデフォルトの init システムです。

systemd は、**systemd unit** の概念を導入します。これらの unit は 表10.2「systemd のユニットファイルの場所」 にあるディレクトリーの 1 つに置かれる unit 設定ファイルで表示され、システムサービスやリスニングソケット、init システムに関連するその他のオブジェクトに関する情報を要約します。利用可能な systemd unit タイプの完全なリストは、表10.1「利用可能な systemd のユニットタイプ」 を参照してください。

表10.1 利用可能な systemd のユニットタイプ

ユニットのタイプ	ファイルの拡張子	詳細
サービスユニット	**.service**	システムサービス
ターゲットユニット	**.target**	systemd ユニットのグループ
自動マウントユニット	**.automount**	ファイルシステムの自動マウントポイント
デバイスユニット	**.device**	カーネルが認識するデバイスファイル
マウントユニット	**.mount**	ファイルシステムのマウントポイント
パスユニット	**.path**	ファイルシステム内のファイルまたはディレクトリー
スコープユニット	**.scope**	外部作成のプロセス
スライスユニット	**.slice**	システムプロセスを管理する、階層的に設定されたユニットのグループ
スナップショットユニット	**.snapshot**	systemd マネージャーの保存状態。
ソケットユニット	**.socket**	プロセス間の通信ソケット
スワップユニット	**.swap**	スワップデバイスまたはスワップファイル

ユニットのタイプ	ファイルの拡張子	詳細
タイマーユニット	**.timer**	systemd タイマー

表10.2 systemd のユニットファイルの場所

ディレクトリー	説明
/usr/lib/systemd/system/	インストール済みの RPM パッケージで配布された systemd のユニットファイル。
/run/systemd/system/	ランタイム時に作成された systemd ユニットファイル。このディレクトリーは、インストール済みのサービスのユニットファイルのディレクトリーよりも優先されます。
/etc/systemd/system/	**systemctl enable** で作成された systemd ユニットファイル、およびサービス拡張向けに追加されたユニットファイル。このディレクトリーは、runtime のユニットファイルのディレクトリーよりも優先されます。

system.conf を使用してデフォルトの systemd 設定の上書き

systemd のデフォルト設定はコンパイル中に定義され、**/etc/systemd/system.conf** にある systemd 設定ファイルで確認できます。ここに記載されるデフォルトではなく、systemd ユニットでグローバルに選択したデフォルト値を上書きする場合は、このファイルを使用します。

たとえば、タイムアウト制限のデフォルト値 (90 秒) を上書きする場合は、**DefaultTimeoutStartSec** パラメーターを使用して、上書きする値を秒単位で入力します。

```
DefaultTimeoutStartSec=required value
```

例10.21「タイムアウト制限の変更」 も参照してください。

10.1.1. 主な特長

Red Hat Enterprise Linux 7 では、systemd システムおよびサービスマネージャーは以下の主要な機能を提供します。

- **ソケットベースのアクティベーション**: 起動時に systemd は、このタイプのアクティベーションをサポートするすべてのシステムサービス用のリスニングソケットを作成し、サービスが開始するとすぐにこれらのソケットをサービスに渡します。これで systemd がサービスを並行で開始できるだけでなく、サービスが利用可能でない間に送信されたメッセージを失うことなくサービスの再起動が可能になります。これは、対応するソケットがアクセス可能なままで、すべてのメッセージがキューに登録されるためです。
  systemd は、ソケットベースの有効化に**ソケットユニット**を使用します。

- **バスベースのアクティベーション**: プロセス間の通信に D-Bus を使用するシステムサービスは、クライアントアプリケーションがシステムサービスとの通信を初めて試みる時にオンデマンドで開始します。systemd は、バスベースのアクティベーションに**D-Bus service files** を使用します。

- **デバイスベースのアクティベーション**: デバイスベースのアクティベーションをサポートするシステムサービスは、特定のタイプのハードウェアがプラグインするか利用可能になると、オンデマンドで開始できます。systemd は、デバイスベースのアクティベーションに、**デバイスユニット**を使用します。

- **パスベースのアクティベーション**: パスベースのアクティベーションをサポートするシステムサービスは、特定のファイルまたはディレクトリーのステータスが変更になると、オンデマンドで開始できます。systemd は、パスベースのアクティベーションに **パスユニット** を使用します。

- **マウントおよび自動マウントポイント管理**: systemd は、マウントポイントおよび自動マウントポイントを監視および管理します。systemd は、**マウントポイントに**マウントユニットを使用し、自動マウントポイントに**自動マウントユニット**を使用します。

- **アグレッシブな並列化**: ソケットベースのアクティベーションを使用するため、systemd はすべてのリスニングソケットが配置されると同時に並行してシステムサービスを開始できます。並列アクティベーションは、オンデマンドのアクティベーションをサポートするシステムサービスと組み合わせることで、システムの起動に必要な時間を大幅に短縮します。

- **トランザクション unit アクティベーション論理**: unit のアクティブ化または非アクティブ化の前に、systemd はその依存関係を計算して一時的なトランザクションを作成し、このトランザクションの一貫性を検証します。トランザクションに一貫性がない場合、systemd は自動的にこれを正そうとし、エラーをレポートする前に必須ではないジョブを削除しようと試みます。

- **SysV init との後方互換性**: Linux Standard Base Core Specificationにあるように、systemd は SysV init スクリプトに対応しています。これにより、systemd サービスのユニットへのアップグレードが容易になります。

10.1.2. 互換性の変更点

systemd システムおよびサービスマネージャーは、その大部分が SysV init および Upstart と互換性があるように設計されています。以下では、Red Hat Enterprise Linux システムの以前のメジャーリリースとの比較で最も顕著な互換性の変更点を挙げています。

- systemd のランレベルのサポートは限定的なものです。ランレベルに直接マッピング可能なターゲットユニットを数多く提供し、互換性のために以前の **runlevel** コマンドで配布されます。ただし、systemd ターゲットのすべてがランレベルに直接マッピングできるわけではないため、このコマンドが不明なランレベルを示す **N** を返す場合もあります。可能な場合は、**runlevel** コマンドの使用を避けることが推奨されます。
  systemd ターゲットの詳細と、ランレベルとの比較は、「systemd ターゲットでの作業」を参照してください。

- **systemctl** ユーティリティーは、カスタマイズされたコマンドをサポートしません。SysV init スクリプトでは、**start**、**stop**、**status** といった標準のコマンドのほかに、任意のコマンドを実装して多くの機能を提供できます。たとえば、Red Hat Enterprise Linux 6 の **iptables** の init スクリプトは、**panic** コマンドで実行できます。これにより、パニックモードが即座に有効になり、システムを再設定して受信パケットおよび送信パケットをすべて切断します。これは systemd ではサポートされておらず、**systemctl** は文書化されたコマンドのみ受け付けます。**systemctl** ユーティリティー、および以前の **service** ユーティリティーとの比較は、「システムサービスの管理」を参照してください。

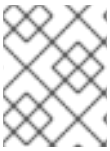
- **systemctl** ユーティリティーは、systemd が開始していないサービスとは通信しません。systemd がシステムサービスを開始すると、メインプロセスの ID を保存して、メインプロセスを追跡します。すると、**systemctl** ユーティリティーがこの PID を使用してクエリーを行い、

サービスを管理します。このため、ユーザーがコマンドラインで特定のデーモンを直接開始すると、**systemctl** がそのデーモンの最新の状態を判断したり、停止したりすることができません。

- systemd が停止するのは、実行中のサービスのみです。Red Hat Enterprise Linux 6 以前のリリースでは、シャットダウンシーケンスが開始すると、**/etc/rc0.d/** ディレクトリーにあるシンボリックリンクを使用して、利用可能なシステムサービスをそのステータスに関係なくすべて停止していました。systemd では、実行中のサービスだけを、シャットダウン時に停止します。
- システムサービスは、標準の入力ストリームからは読み取れません。systemd がサービスを開始すると、標準入力を **/dev/null** に接続し、ユーザーとの対話を行わないようにします。
- システムサービスは、呼び出したユーザーやそのセッションから、(環境変数 **HOME**、**PATH** などの) コンテキストを継承しません。各サービスは、クリーンな実行コンテキストで実行します。
- SysV init スクリプトを読み込む際に、systemd は Linux Standard Base (LSB) ヘッダーにエンコードされている依存関係情報を読み取り、ランタイム時に解釈します。
- サービスユニット上のすべての操作は、デフォルトで 5 分でタイムアウトになるように設定されており、サービスの故障でシステムがフリーズすることを防ぎます。この値は initscript から生成されるサービス用にハードコーディングされ、変更することができません。ただし、個別の設定ファイルを使用して、サービスごとにタイムアウト値を長くすることができます。例 10.21「タイムアウト制限の変更」 を参照してください。

systemd で導入された互換性の変更点に関する詳細なリストは、Red Hat Enterprise Linux 7 の 移行計画ガイド を参照してください。

10.2. システムサービスの管理



注記

専門知識のさらなる拡充を図るには、Red Hat System Administration II (RH134) トレーニングコースもあります。

以前のバージョンの Red Hat Enterprise Linux は、SysV init または Upstart で配布されており、**/etc/rc.d/init.d/** ディレクトリーにある **init スクリプト** を使用していました。この init スクリプトは通常の Bash で書かれており、システム管理者がシステム内で、サービスの状態とデーモンを管理できるようになっていました。Red Hat Enterprise Linux 7 では、この init スクリプトは、**サービスユニット** に代わっています。

サービスユニットは、ファイル拡張子 **.service** で終わり、init スクリプトと同様のロールを担います。システムサービスの表示、開始、停止、再開、有効化、無効化には、表10.3「service ユーティリティーと systemctl の比較」、表10.4「chkconfig ユーティリティーと systemctl の比較」、および本セクションで説明されているように、**systemctl** コマンドラインを使用します。**service** コマンドおよび **chkconfig** コマンドは、引き続きシステムで利用可能になっており、期待通りに機能しますが、これらは互換性のために含まれており、使用は推奨されていません。

表10.3 service ユーティリティーと systemctl の比較

サービス	systemctl	詳細
service name start	**systemctl start name.service**	サービスを起動します。

サービス	systemctl	詳細
service name stop	**systemctl stop name.service**	サービスを停止します。
service name restart	**systemctl restart name.service**	サービスを再起動します。
service name condrestart	**systemctl try-restart name.service**	サービスが実行中の場合のみ、再起動します。
service name reload	**systemctl reload name.service**	設定を再読み込みします。
service name status	**systemctl status name.service** **systemctl is-active name.service**	サービスが実行中かどうかをチェックします。
service --status-all	**systemctl list-units --type service --all**	すべてのサービスのステータスを表示します。

表10.4 chkconfig ユーティリティーと systemctl の比較

chkconfig	systemctl	詳細
chkconfig name on	**systemctl enable name.service**	サービスを有効にします。
chkconfig name off	**systemctl disable name.service**	サービスを無効にします。
chkconfig --list name	**systemctl status name.service** **systemctl is-enabled name.service**	サービスが有効かどうかを確認します。
chkconfig --list	**systemctl list-unit-files --type service**	サービスをリスト表示し、各サービスが有効かどうかを確認します。
chkconfig --list	**systemctl list-dependencies --after**	指定されたユニットの前に開始するように指定されているサービスをリスト表示します。
chkconfig --list	**systemctl list-dependencies --before**	指定されたユニットの後に開始するように指定されているサービスをリスト表示します。

サービスユニットの指定
分かりやすくするため、本セクションの残りの部分のコマンド例では、**.service** ファイル拡張子がついた完全なユニット名を使用します。以下に例を示します。

```
~]# systemctl stop nfs-server.service
```

ただし、ファイル拡張子は省略することができます。省略すると、**systemctl** は、引数がサービスユニットであることを想定します。以下のコマンドは、上記のコマンドと同等のものになります。

```
~]# systemctl stop nfs-server
```

また、ユニットによってはエイリアス名を持つものもあります。この名前は、ユニット名よりも短くすることができ、ユニット名の代わりとして使用できます。特定のユニットに使用できるエイリアスを見つけるには、以下のコマンドを実行します。

```
~]# systemctl show nfs-server.service -p Names
```

chroot 環境における systemctl の挙動
chroot コマンドを使用して root ディレクトリーを変更すると、ほとんどの **systemctl** コマンドは、アクションの実行をすべて拒否します。なぜなら、**systemd** プロセスと、**chroot** コマンドを使用しているユーザーでは、ファイルシステムの見え方が異なるからです。このような状況は、**systemctl** が **キックスタート** ファイルから呼び出されたときなどに発生します。

例外が、**systemctl enable** や **systemctl disable** などのユニットファイルコマンドです。このコマンドは、実行中のシステムを必要とせず実行中のプロセスに影響を与えませんが、ユニットファイルには影響を及ぼします。したがってこのようなコマンドは、**chroot** 環境であっても実行することが可能です。たとえば、**/srv/website1/** ディレクトリー配下で、システムの **httpd** サービスを有効にするときは、以下のコマンドを実行します。

```
~]# chroot /srv/website1
~]# systemctl enable httpd.service
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service, pointing to
/usr/lib/systemd/system/httpd.service.
```

10.2.1. サービスのリスト表示

読み込み済みのサービスユニットのリストを表示するには、シェルプロンプトで以下を実行します。

```
systemctl list-units --type service
```

各サービスのユニットファイルに対して、このコマンドは正式名 (**UNIT**) の後に、そのユニットファイルが読み込まれているかどうか (**LOAD**)、そのユニットファイルアクティベーションの状態の概要 (**ACTIVE**) および詳細 (**SUB**) な状態、そして簡単な説明 (**DESCRIPTION**) を示します。

デフォルトでは、**systemctl list-units** コマンドは、アクティブなユニットのみを表示します。状態に関係なく読み込み済みユニットをすべて表示する場合は、コマンドラインオプションの **--all** または **-a** を付けて、以下のコマンドを実行します。

```
systemctl list-units --type service --all
```

また、利用可能なサービスユニットをリスト表示して、各ユニットが有効かどうかを確認できます。これには、以下のコマンドを実行します。

systemctl list-unit-files --type service

このコマンドにより、各サービスユニットの完全な名前 (**UNIT FILE**) と、サービスユニットが有効かどうか (**STATE**) が表示されます。個別のサービスユニットの状態を判断する方法は 「サービスステータスの表示」 を参照してください。

例10.1 サービスのリスト表示

読み込み済みのサービスユニットのリストを表示するには、以下のコマンドを実行します。

```
~]$ systemctl list-units --type service
UNIT             LOAD  ACTIVE SUB    DESCRIPTION
abrt-ccpp.service       loaded active exited Install ABRT coredump hook
abrt-oops.service       loaded active running ABRT kernel log watcher
abrt-vmcore.service     loaded active exited Harvest vmcores for ABRT
abrt-xorg.service       loaded active running ABRT Xorg log watcher
abrtd.service           loaded active running ABRT Automated Bug Reporting Tool
...
systemd-vconsole-setup.service loaded active exited Setup Virtual Console
tog-pegasus.service     loaded active running OpenPegasus CIM Server

LOAD  = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB  = The low-level unit activation state, values depend on unit type.

46 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'
```

インストール済みのサービスユニットファイルをリスト表示して、そのユニットが有効かどうかを判断するには、以下コマンドを実行します。

```
~]$ systemctl list-unit-files --type service
UNIT FILE                 STATE
abrt-ccpp.service             enabled
abrt-oops.service             enabled
abrt-vmcore.service           enabled
abrt-xorg.service             enabled
abrtd.service                 enabled
...
wpa_supplicant.service            disabled
ypbind.service                disabled

208 unit files listed.
```

10.2.2. サービスステータスの表示

システムサービスに対応するサービスユニットに関する詳細情報を表示するには、シェルプロンプトで以下を実行します。

systemctl status name.service

name を、確認するサービスユニット名 (**gdm** など) に置き換えます。このコマンドでは、選択した

サービスユニット名の後に、簡単な説明と、表10.5「利用可能なサービスユニットの情報」 にある 1 つ以上のフィールド、さらに **root** ユーザーが実行している場合は最新のログエントリーが表示されます。

表10.5 利用可能なサービスユニットの情報

フィールド	説明
Loaded	サービスユニットが読み込まれているかどうか、ユニットファイルへの絶対パス、ユニットが有効かどうかについての説明
Active	サービスユニットが実行中かどうかの説明と、タイムスタンプ
Main PID	対応するシステムサービスの PID と、その名前
状態	対応するシステムサービスに関する追加情報
Process	関連プロセスに関する追加情報
CGroup	関連するコントロールグループ (cgroup) に関する追加情報

特定のサービスユニットが実行中かどうかだけを確認する場合は、以下のコマンドを実行します。

```
systemctl is-active name.service
```

同様に、特定のサービスユニットが有効かどうかを確認するには、以下のコマンドを実行します。

```
systemctl is-enabled name.service
```

systemctl is-active および **systemctl is-enabled** は両方とも、指定したサービスユニットが実行中または有効な場合に、終了ステータス **0** を返すことに注意してください。現在読み込み済みのサービスユニットのリストを表示する方法は、「サービスのリスト表示」 を参照してください。

例10.2 サービスステータスの表示

GNOME Display Manager のサービスユニット名は **gdm.service** になります。このサービスユニットの現在のステータスを確認するには、シェルプロンプトで次のコマンドを実行します。

```
~]# systemctl status gdm.service
gdm.service - GNOME Display Manager
  Loaded: loaded (/usr/lib/systemd/system/gdm.service; enabled)
  Active: active (running) since Thu 2013-10-17 17:31:23 CEST; 5min ago
 Main PID: 1029 (gdm)
  CGroup: /system.slice/gdm.service
      ├─1029 /usr/sbin/gdm
      ├─1037 /usr/libexec/gdm-simple-slave --display-id /org/gno...
      └─1047 /usr/bin/Xorg :0 -background none -verbose -auth /r...

Oct 17 17:31:23 localhost systemd[1]: Started GNOME Display Manager.
```

例10.3 サービスの前に開始するサービスの表示

特定のサービスの前に開始するサービスを確認するには、シェルプロンプトで次のコマンドを実行します。

```
~]# systemctl list-dependencies --after gdm.service
gdm.service
├─dbus.socket
├─getty@tty1.service
├─livesys.service
├─plymouth-quit.service
├─system.slice
├─systemd-journald.socket
├─systemd-user-sessions.service
└─basic.target
[output truncated]
```

例10.4 サービスの後に開始するサービスの表示

特定のサービスの後に開始するサービスを確認するには、シェルプロンプトで次のコマンドを実行します。

```
~]# systemctl list-dependencies --before gdm.service
gdm.service
├─dracut-shutdown.service
├─graphical.target
│ ├─systemd-readahead-done.service
│ ├─systemd-readahead-done.timer
│ └─systemd-update-utmp-runlevel.service
└─shutdown.target
  ├─systemd-reboot.service
  └─final.target
    └─systemd-reboot.service
```

10.2.3. サービスの起動

システムサービスに対応するサービスユニットを開始する場合は、**root** で次のコマンドを実行します。

```
systemctl start name.service
```

name を、開始するサービスユニット名 (たとえば、**gdm**) に置き換えます。このコマンドは、選択したサービスを現行セッションで開始します。起動時にサービスユニットを開始するようにする方法は、「サービスの有効化」 を参照してください。特定のサービスユニットのステータスを確認する方法は 「サービスステータスの表示」 を参照してください。

例10.5 サービスの起動

Apache HTTP サーバー用のサービスユニットは **httpd.service** です。サービスユニットをアクティブにし、現行セッションで **httpd** デーモンを起動するには、**root** で以下のコマンドを実行します。

```
~]# systemctl start httpd.service
```

10.2.4. サービスの停止

システムサービスに対応するサービスユニットを停止するには、**root** で次のコマンドを実行します。

systemctl stop name.service

name を、停止するサービスユニット名 (たとえば **bluetooth**) に置き換えます。このコマンドは、選択したサービスユニットを現行セッションで停止します。システムの起動時にサービスユニットを無効にし、開始しないようにする方法は、「サービスの無効化」 を参照してください。特定のサービスユニットのステータスを確認する方法は 「サービスステータスの表示」 を参照してください。

例10.6 サービスの停止

bluetoothd デーモンのサービスユニットは **bluetooth.service** です。サービスユニットを無効にし、現行セッションで **bluetoothd** デーモンを停止する場合は、**root** で以下のコマンドを実行します。

```
~]# systemctl stop bluetooth.service
```

10.2.5. サービスの再開

システムサービスに対応するサービスユニットを再開する場合は、**root** で次のコマンドを実行します。

systemctl restart name.service

name を、再開するサービスユニット名 (たとえば、**httpd**) に置き換えます。このコマンドは、現行セッションで選択したサービスユニットを停止し、即座に再起動します。重要なのは、選択したサービスユニットが実行中でないと、このコマンドがそのサービスユニットを起動するということです。対応するサービスがすでに実行中の場合にのみ、サービスユニットを再起動するように systemd に設定するには、**root** で以下のコマンドを実行します。

systemctl try-restart name.service

システムサービスによっては、サービスの実行を中断することなく設定の再読み込みが可能です。これを実行するには、**root** で次のコマンドを実行します。

systemctl reload name.service

システムサービスがこの機能をサポートしない場合は、このコマンドが無視されることに注意してください。代わりに、**systemctl** コマンドでは、この機能をサポートしないサービスを代わりに再起動する **reload-or-restart** コマンドおよび **reload-or-try-restart** コマンドもサポートします。特定のサービスユニットのステータスを確認する方法は 「サービスステータスの表示」 を参照してください。

例10.7 サービスの再開

ユーザーが不要なエラーメッセージや、部分的に表示される Web ページに遭遇しないようにするため、Apache HTTP Server では設定を再起動したり、処理されたリクエストをアクティブに妨害したりせずに、設定を編集したり再読み込みしたりできます。これを行うには、**root** で次のコマンドを

実行します。

```
~]# systemctl reload httpd.service
```

10.2.6. サービスの有効化

システムの起動時にシステムサービスに対応するサービスユニットを自動的に起動するように設定するには、**root** で次のコマンドを実行します。

```
systemctl enable name.service
```

name を、有効にするサービスユニット名 (**httpd** など) に置き換えます。このコマンドは、選択したサービスユニットの **[Install]** セクションを読み取り、**/etc/systemd/system/** ディレクトリーおよびそのサブディレクトリーにある **/usr/lib/systemd/system/name.service** ファイルへの適切なシンボリックリンクを作成します。ただし、このコマンドは既存のリンクを上書きしません。シンボリックリンクが確実に再作成されるようにするには、**root** で以下のコマンドを実行します。

```
systemctl reenable name.service
```

このコマンドは、選択したサービスユニットを無効にし、即座に再度有効にします。特定のサービスユニットが起動時に有効になるかどうかを確認する方法は 「サービスステータスの表示」 を参照してください。現行セッションでサービスを開始する方法は 「サービスの起動」 を参照してください。

例10.8 サービスの有効化

システムの起動時に Apache HTTP Server が自動的に開始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。

```
~]# systemctl enable httpd.service
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to
/usr/lib/systemd/system/httpd.service.
```

10.2.7. サービスの無効化

システムの起動時に、システムサービスに対応するサービスユニットを自動的に起動しないように設定するには、**root** で次のコマンドを実行します。

```
systemctl disable name.service
```

name を、無効にするサービスユニット名 (**bluetooth** など) に置き換えます。このコマンドは、選択したサービスユニットの **[Install]** セクションを読み取り、**/etc/systemd/system/** ディレクトリーおよびそのサブディレクトリーから、**/usr/lib/systemd/system/name.service** ファイルへの適切なシンボリックリンクを削除します。さらに、サービスユニットにマスクをして、手動で開始したり、別のサービスがこれを開始することを防いだりできます。これを行うには、**root** で以下のコマンドを実行します。

```
systemctl mask name.service
```

このコマンドにより、**/etc/systemd/system/name.service** ファイルを、**/dev/null** へのシンボリックリンクに置き換え、実際のユニットファイルが systemd ファイルにアクセスできないようにします。この動作を元に戻してサービスユニットのマスクを解除するには、**root** で以下のコマンドを実行します。

systemctl unmask name.service

特定のサービスユニットが起動時に有効になるかどうかを確認する方法は 「サービスステータスの表示」 を参照してください。現行セッションでサービスを停止する方法は 「サービスの停止」 を参照してください。

例10.9 サービスの無効化

例10.6「サービスの停止」 では、現行セッションで **bluetooth.service** ユニットを停止する方法を説明しました。システムの起動時にこのサービスユニットが開始しないようにするには、**root** で次のコマンドを実行します。

```
~]# systemctl disable bluetooth.service
Removed symlink /etc/systemd/system/bluetooth.target.wants/bluetooth.service.
Removed symlink /etc/systemd/system/dbus-org.bluez.service.
```

10.2.8. 競合するサービスの起動

systemd では、サービスとサービスとの間に正と負の依存関係が存在します。特定のサービスを起動するとき、別のサービスを 1 つまたは複数開始 (正の依存関係)、あるいはサービスを 1 つまたは複数停止 (負の依存関係) することが必要となる場合があります。

ユーザーが新しいサービスを起動しようとすると、**systemd** がすべての依存関係を自動的に解決します。これは、ユーザーに明示的に通知されることなく実行されるため注意が必要です。サービスが実行されているときに、負の依存関係を持つ別のサービスを起動しようとすると、実行しているサービスが自動的に停止します。

たとえば、**postfix** サービスを実行している時に **sendmail** サービスを起動すると、**systemd** は、自動的に **postfix** を停止します。 この 2 つのサービスは競合するため、同じポートでは実行できません。

10.3. SYSTEMD ターゲットでの作業

以前のバージョンの Red Hat Enterprise Linux は、SysV init または Upstart で配布されており、特定モードのオペレーションを表す事前定義の **ランレベル** を実装していました。このランレベルは 0 から 6 までの数字で表され、システム管理者が各ランレベルを有効にしたときに実行するシステムサービスが定義されていました。Red Hat Enterprise Linux 7 では、ランレベルの概念が **systemd ターゲット** に代わっています。

systemd ターゲットは **ターゲットユニット** で表現されます。ターゲットユニットは **.target** ファイル拡張子で終わり、その唯一の目的は依存関係の連鎖で他の systemd ユニットをグループ化することです。たとえば、グラフィカルセッションの開始に使用する **graphical.target** ユニットは、GNOME Display Manager (**gdm.service**) または Accounts Service (**accounts-daemon.service**) といったシステムサービスを開始するとともに、**multi-user.target** ユニットもアクティブにします。同様に、**multi-user.target** ユニットは、NetworkManager (**NetworkManager.service**)、D-Bus (**dbus.service**) といった、その他の必須システムサービスを開始し、**basic.target** という別のターゲットユニットをアクティブにします。

Red Hat Enterprise Linux 7 では、以前のシステムリリースの標準ランレベルと類似する定義済みターゲットが多数同梱されています。互換性の理由から、このようなターゲットのエイリアスも SysV ランレベルに直接マッピングします。表10.6「SysV ランレベルと systemd ターゲットの比較」 SysV ランレベルとそれに対応する systemd ターゲットの完全リストを提供します。

表10.6 SysV ランレベルと systemd ターゲットの比較

ランレベル	ターゲットユニット	詳細
0	**runlevel0.target**, **poweroff.target**	システムをシャットダウンし、電源を切ります。
1	**runlevel1.target**, **rescue.target**	レスキューシェルを設定します。
2	**runlevel2.target**, **multi-user.target**	非グラフィカルなマルチユーザーシステムを設定します。
3	**runlevel3.target**, **multi-user.target**	非グラフィカルなマルチユーザーシステムを設定します。
4	**runlevel4.target**, **multi-user.target**	非グラフィカルなマルチユーザーシステムを設定します。
5	**runlevel5.target**, **graphical.target**	グラフィカルなマルチユーザーシステムを設定します。
6	**runlevel6.target**, **reboot.target**	システムをシャットダウンして再起動します。

systemd ターゲットを表示、変更、または設定するには、表10.7「SysV init コマンドと systemctl の比較」 および以下のセクションの説明に従って **systemctl** ユーティリティーを使用します。**runlevel** コマンドおよび **telinit** コマンドは今も利用でき、期待どおりに機能しますが、このコマンドは互換性のために同梱されているため、なるべく使用しないでください。

表10.7 SysV init コマンドと systemctl の比較

古いコマンド	新しいコマンド	詳細
runlevel	**systemctl list-units --type target**	現在読み込まれているターゲットユニットをリスト表示します。
telinit runlevel	**systemctl isolate name.target**	現在のターゲットを変更します。

10.3.1. デフォルトターゲットの表示

デフォルトでどのターゲットユニットが使用されるかを決定するには、以下のコマンドを実行します。

```
systemctl get-default
```

このコマンドは、**/etc/systemd/system/default.target** にあるシンボリックリンクを解決して、結果を表示します。デフォルトのターゲット変更する方法は 「デフォルトターゲットの変更」 を参照してください。現在、読み込み済みのターゲットユニットをリスト表示する方法は 「現在のターゲットの表示」 を参照してください。

例10.10 デフォルトターゲットの表示

デフォルトのターゲットユニットを表示するには、以下のコマンドを実行します。

```
~]$ systemctl get-default
graphical.target
```

10.3.2. 現在のターゲットの表示

読み込み済みのターゲットユニットのリストを表示するには、以下のコマンドを実行します。

```
systemctl list-units --type target
```

このコマンドを実行すると、各ターゲットユニットの正式名 (**UNIT**) の後に、そのユニットが読み込まれているかどうか (**LOAD**)、そのユニットファイルアクティベーションの状態の概要 (**ACTIVE**) および詳細 (**SUB**) な状態、簡単な説明 (**DESCRIPTION**) が表示されます。

デフォルトでは、**systemctl list-units** コマンドは、アクティブなユニットのみを表示します。状態に関係なく読み込み済みユニットをすべて表示する場合は、コマンドラインオプションの **--all** または **-a** を付けて、以下のコマンドを実行します。

```
systemctl list-units --type target --all
```

デフォルトターゲットの表示方法は、「デフォルトターゲットの表示」 を参照してください。現在のターゲットを変更する方法は、「現在のターゲットの変更」 を参照してください。

例10.11 現在のターゲットの表示

現在読み込み済みのターゲットユニットをリスト表示するには、以下のコマンドを実行します。

```
~]$ systemctl list-units --type target
UNIT          LOAD  ACTIVE SUB  DESCRIPTION
basic.target      loaded active active Basic System
cryptsetup.target   loaded active active Encrypted Volumes
getty.target      loaded active active Login Prompts
graphical.target   loaded active active Graphical Interface
local-fs-pre.target  loaded active active Local File Systems (Pre)
local-fs.target     loaded active active Local File Systems
multi-user.target   loaded active active Multi-User System
network.target     loaded active active Network
paths.target      loaded active active Paths
remote-fs.target   loaded active active Remote File Systems
sockets.target     loaded active active Sockets
sound.target      loaded active active Sound Card
spice-vdagentd.target loaded active active Agent daemon for Spice guests
swap.target       loaded active active Swap
sysinit.target     loaded active active System Initialization
time-sync.target   loaded active active System Time Synchronized
timers.target      loaded active active Timers

LOAD  = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB  = The low-level unit activation state, values depend on unit type.
```

```
17 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

10.3.3. デフォルトターゲットの変更

システムがデフォルトで異なるターゲットユニットを使用するように設定するには、**root** で次のコマンドを実行します。

```
systemctl set-default name.target
```

name を、デフォルトで使用するターゲットユニットの名前 (**multi-user** など) に置き換えます。このコマンドにより、**/etc/systemd/system/default.target** ファイルが、**/usr/lib/systemd/system/name.target** へのシンボリックリンクに置き換わります。name は、使用するターゲットユニットの名前になります。現在のターゲットを変更する方法は、「現在のターゲットの変更」 を参照してください。現在、読み込み済みのターゲットユニットをリスト表示する方法は「現在のターゲットの表示」 を参照してください。

例10.12 デフォルトターゲットの変更

デフォルトで **multi-user.target** ユニットを使用するようにシステムを設定するには、**root** で以下のコマンドを実行します。

```
~]# systemctl set-default multi-user.target
rm '/etc/systemd/system/default.target'
ln -s '/usr/lib/systemd/system/multi-user.target' '/etc/systemd/system/default.target'
```

10.3.4. 現在のターゲットの変更

現行セッションで異なるターゲットユニットに変更するには、**root** で次のコマンドを実行します。

```
systemctl isolate name.target
```

name を、使用するターゲットユニットの名前 (**multi-user** など) に置き換えます。このコマンドは、name という名前のターゲットユニットと、その従属ユニットをすべて起動し、その他のユニットを直ちに停止します。デフォルトのターゲット変更する方法は 「デフォルトターゲットの変更」 を参照してください。現在、読み込み済みのターゲットユニットをリスト表示する方法は 「現在のターゲットの表示」 を参照してください。

例10.13 現在のターゲットの変更

グラフィカルユーザーインターフェイスを無効にし、現行セッションで **multi-user.target** ユニットに変更するには、**root** で以下のコマンドを実行します。

```
~]# systemctl isolate multi-user.target
```

10.3.5. レスキューモードへの変更

レスキューモード は、便利なシングルユーザー環境を提供し、通常の起動プロセスを完了できない状況

でのシステムの修復を可能にします。レスキューモードでは、システムはすべてのローカルファイルシステムのマウントと、いくつかの重要なシステムサービスの開始を試みますが、ネットワークインターフェイスをアクティブにしたり、他のユーザーによるシステムへの同時ログインを許可したりすることはしません。Red Hat Enterprise Linux 7 では、レスキューモードはシングルユーザーモードと同等であり、**single user mode** パスワードを必要とします。

現在のターゲットを変更し、現行セッションでレスキューモードに入るには、**root** で次のコマンドを実行します。

```
systemctl rescue
```

このコマンドは **systemctl isolate rescue.target** と似ていますが、システムに現在ログインしているすべてのユーザーに情報メッセージを送信します。systemd がこのメッセージを送信しないようにするには、コマンドラインオプション **--no-wall** を付けてこのコマンドを実行します。

```
systemctl --no-wall rescue
```

緊急モードに入る方法は、「緊急モードへの変更」 を参照してください。

例10.14 レスキューモードへの変更

現行セッションでレスキューモードに入るには、**root** で以下のコマンドを実行します。

```
~]# systemctl rescue

Broadcast message from root@localhost on pts/0 (Fri 2013-10-25 18:23:15 CEST):

The system is going down to rescue mode NOW!
```

10.3.6. 緊急モードへの変更

緊急モード は、可能な限り最小限の環境を提供し、レスキューモードに入れないシステム状態でのシステムの修復を可能にします。緊急モードでは、システムは root ファイルシステムを読み込み専用でマウントし、他のローカルファイルシステムのマウントは試みません。また、ネットワークインターフェイスのアクティブ化も行わず、限定的な必須サービスのみを起動します。Red Hat Enterprise Linux 7 では、緊急モードでは root パスワードが必要です。

現在のターゲットを変更し、緊急モードに入るには、**root** で次のコマンドを実行します。

```
systemctl emergency
```

このコマンドは **systemctl isolate emergency.target** と似ていますが、システムに現在ログインしているすべてのユーザーに情報メッセージを送信します。systemd がこのメッセージを送信しないようにするには、コマンドラインオプション **--no-wall** を付けてこのコマンドを実行します。

```
systemctl --no-wall emergency
```

レスキューモードに入る方法は、「レスキューモードへの変更」 を参照してください。

例10.15 緊急モードへの変更

現在システムにログインしている全ユーザーにメッセージを送信せずに緊急モードに入るには、**root** で以下のコマンドを実行します。

```
~]# systemctl --no-wall emergency
```

10.4. システムのシャットダウン、サスペンド、および休止状態

Red Hat Enterprise Linux 7 では、**systemctl** ユーティリティーが、これまでのバージョンの Red Hat Enterprise Linux システムで使用されていた多くの電源管理コマンドに置き換わっています。表 10.8「電源管理コマンドと systemctl の比較」 に表示されているコマンドは、互換性の理由から現在も利用することができますが、可能な場合は **systemctl** の使用が推奨されます。

表10.8 電源管理コマンドと systemctl の比較

古いコマンド	新しいコマンド	詳細
halt	**systemctl halt**	システムを停止します。
poweroff	**systemctl poweroff**	システムの電源を切ります。
reboot	**systemctl reboot**	システムを再起動します。
pm-suspend	**systemctl suspend**	システムをサスペンドします。
pm-hibernate	**systemctl hibernate**	システムを休止状態にします。
pm-suspend-hybrid	**systemctl hybrid-sleep**	システムを休止状態にしてサスペンドします。

10.4.1. システムのシャットダウン

systemctl ユーティリティーは、システムをシャットダウンするコマンドを提供します。ただし、従来の **shutdown** コマンドもサポートされます。**shutdown** コマンドは、**systemctl** ユーティリティーを呼び出してシャットダウンを実行しますが、time 引数もサポートするという利点があります。これは、計画メンテナンスにとりわけ役立ち、システムシャットダウンの予定に関する警告にユーザーが対応する時間をより長く確保できます。シャットダウンをキャンセルするオプションがあることも利点です。

systemctl コマンドの使用
システムをシャットダウンし、マシンの電源を切るには、**root** で次のコマンドを実行します。

```
systemctl poweroff
```

マシンの電源を切らずにシステムをシャットダウンして停止するには、**root** で以下のコマンドを実行します。

```
systemctl halt
```

デフォルトでは、このコマンドのいずれかを実行すると、systemd が、システムに現在ログインしたすべてのユーザーに情報メッセージを送信します。systemd がメッセージを送信しないようにするには、たとえば、コマンドラインオプション **--no-wall** を付けてコマンドを実行します。

```
systemctl --no-wall poweroff
```

shutdown コマンドの使用
指定した時間にシステムをシャットダウンしてマシンの電源を切るには、**root** で、以下の形式でコマンドを実行します。

```
shutdown --poweroff hh:mm
```

hh:mm は 24 時間形式の時刻となります。新たなログインを防ぐために、システムをシャットダウンする 5 分前に **/run/nologin** ファイルが作成されます。時間引数を使用する場合は、コマンドに任意のメッセージ **wall message** を付けることができます。

マシンの電源を切らずに、少し待ってシステムをシャットダウンして停止するには、**root** で以下の形式のコマンドを実行します。

```
shutdown --halt +m
```

+m は遅らせる時間 (分) です。キーワード **now** は、**+0** のエイリアスとなります。

保留中のシャットダウンは、**root** で以下のコマンドを実行するとキャンセルできます。

```
shutdown -c
```

詳細なコマンドオプションは、**shutdown(8)** man ページを参照してください。

10.4.2. システムの再起動

システムを再起動するには、**root** で以下のコマンドを実行します。

```
systemctl reboot
```

デフォルトでは、このコマンドにより、systemd が、システムに現在ログインしたすべてのユーザーに情報メッセージを送信します。systemd がこのメッセージを送信しないようにするには、コマンドラインオプション **--no-wall** を付けてこのコマンドを実行します。

```
systemctl --no-wall reboot
```

10.4.3. システムのサスペンド

システムをサスペンドするには、**root** で次のコマンドを実行します。

```
systemctl suspend
```

このコマンドは、システムの状態を RAM に保存し、マシンにある、RAM モジュール以外のほとんどのデバイスの電源を切ります。マシンの電源を戻すと、システムは再起動せずに RAM からその状態を復元します。システムの状態がハードディスクではなく RAM に保存されるので、システムのサスペンドモードから復元する場合は、休止状態からの復元と比べて著しく速くなります。ただし、システムをサスペンドした状態は、停電に対して脆弱となります。

システムを休止状態にする方法は、「システムの休止状態」 を参照してください。

10.4.4. システムの休止状態

システムを休止状態にするには、**root** で次のコマンドを実行します。

```
systemctl hibernate
```

このコマンドは、システムの状態をハードディスクドライブに保存し、マシンの電源を切ります。マシンの電源を戻すと、システムは再起動せずに、保存されたデータからその状態を復元します。システムの状態が RAM ではなくハードディスクに保存されるため、マシンが RAM モジュールに電力を維持する必要がありません。ただし、システムの休止状態から復元する場合は、サスペンドモードからの復元と比べて大幅に遅くなります。

システムを休止状態にしてサスペンドするには、**root** で次のコマンドを実行します。

```
systemctl hybrid-sleep
```

システムをサスペンドする方法は、「システムのサスペンド」 を参照してください。

10.5. リモートマシン上での SYSTEMD の制御

systemd システムおよびサービスマネージャーをローカルで制御することに加え、**systemctl** ユーティリティーでは、SSH プロトコルを使用してリモートマシン上で実行している systemd と対話できます。**sshd** サービスがリモートマシン上で実行中であれば、**systemctl** コマンドに **--host** または **-H** オプションを指定して実行することで、このマシンに接続できます。

```
systemctl --host user_name@host_name command
```

user_name を、リモートユーザーの名前、**host_name** をマシンのホスト名に置き換え、**command** は上記の **systemctl** コマンドのいずれかに置き換えます。指定したユーザーが SSH プロトコルを使用してリモートアクセスできるようにリモートマシンを設定する必要があることに注意してください。SSH サーバーの設定に関する詳細情報は、12章*OpenSSH* を参照してください。

例10.16 リモート管理

server-01.example.com という名前のリモートマシンに **root** ユーザーとしてログインし、**httpd.service** ユニットの現在の状態を判断するには、シェルプロンプトに以下を入力します。

```
~]$ systemctl -H root@server-01.example.com status httpd.service
>>>>>>> systemd unit files -- update
root@server-01.example.com's password:
httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled)
  Active: active (running) since Fri 2013-11-01 13:58:56 CET; 2h 48min ago
 Main PID: 649
  Status: "Total requests: 0; Current requests/sec: 0; Current traffic:  0 B/sec"
  CGroup: /system.slice/httpd.service
```

10.6. SYSTEMD のユニットファイルの作成および変更

ユニットファイルには、ユニットを説明し、その動作を定義する設定ディレクティブが含まれます。複数の **systemctl** コマンドがバックグラウンドでユニットファイルと連携します。詳細な調整を行うには、システム管理者がユニットファイルを手動で編集または作成する必要があります。表

10.2「systemd のユニットファイルの場所」 には、システムにユニットファイルが保存される 3 つのメインディレクトリーが記載されています。**/etc/systemd/system/** ディレクトリーは、システム管理者が作成またはカスタマイズするユニットファイル用に予約されます。

ユニットファイル名は、以下のフォーマットを使用します。

```
unit_name.type_extension
```

unit_name はユニットの名前を表し、**type_extension** はユニットタイプを識別します。ユニットタイプの完全なリストは、表10.1「利用可能な systemd のユニットタイプ」 を参照してください。たとえば、通常は、システムには **sshd.service** ユニットおよび **sshd.socket** ユニットがあります。

ユニットファイルには、追加の設定ファイルのディレクトリーを追加できます。たとえば、カスタム設定オプションを **sshd.service** に追加するには、**sshd.service.d/custom.conf** ファイルを作成し、追加のディレクティブを挿入します。設定ディレクティブの詳細情報は、「既存のユニットファイルの変更」 を参照してください。

さらに、**sshd.service.wants/** ディレクトリーおよび **sshd.service.requires/** ディレクトリーを作成することもできます。このディレクトリーには、**sshd** サービスの依存関係であるユニットファイルへのシンボリックリンクが含まれます。シンボリックリンクは、[Install] ユニットファイルオプションに従ってインストール時に自動的に作成されるか (表10.11「[Install] セクションの重要なオプション」 を参照)、または [Unit] オプションに基づいてランタイム時に自動的に作成されます (表10.9「[Unit] セクションの重要なオプション」 を参照)。このディレクトリーとシンボリックリンクを手動で作成することもできます。

多くのユニットファイルオプションは、いわゆる **ユニット指定子** を使用して設定できます。これは、ユニットファイルが読み込まれる際にユニットパラメーターに動的に置き換えられるワイルドカード文字列です。これにより、インスタンス化されたユニットを生成するテンプレートとしてのロールを担う汎用ユニットファイルを作成できます。詳しくは 「インスタンス化されたユニットの使用」 を参照してください。

10.6.1. ユニットファイル構造の概要

通常、ユニットファイルは 3 つのセクションで設定されています。

- [Unit]: ユニットのタイプに依存しない汎用的なオプションが含まれます。このセクションに含まれるオプションはユニットを説明し、ユニットの動作を指定し、他のユニットへの依存関係を設定します。最も頻繁に使用される [Unit] オプションのリストは、表10.9「[Unit] セクションの重要なオプション」 を参照してください。

- [**unit type**]: ユニットにタイプ固有のディレクティブがある場合、それらはユニットタイプに基づいて名前が付けられるセクションにグループ分けされます。たとえば、サービスユニットファイルには [Service] セクションが含まれます。最もよく使われる [Service] オプションは 表10.10「[Service] セクションの重要なオプション」 を参照してください。

- [Install]: **systemctl enable** コマンドおよび **disable** コマンドで使用されるユニットインストールに関する情報が含まれます。[Install] オプションのリストは、表10.11「[Install] セクションの重要なオプション」 を参照してください。

表10.9 [Unit] セクションの重要なオプション

オプション[a]	セクション、**systemd.unit(5)** man ページ]	詳細

オプション[a] セクション、systemd.unit(5) man ページ]	詳細
説明	ユニットの説明です。このテキストは、たとえば **systemctl status** コマンドの出力に表示されます。
Documentation	ユニットのドキュメントを参照する URI のリストを提供します。
After[b]	ユニットが開始する順序を定義します。このユニットは、**After** で指定されたユニットがアクティブになると開始します。**Requires** とは異なり、**After** は、指定したユニットを明示的にアクティブにしません。**Before** オプションには、**After** と機能が反対になります。
Requires	その他のユニットに依存関係を設定します。**Requires** にリスト表示されるユニットは、対応するユニットと共にアクティブになります。必要なユニットのいずれかが開始しないと、このユニットはアクティブになりません。
Wants	**Requires** よりも強度の弱い依存関係を設定します。リストに示されるユニットのいずれかが正常に開始しなくても、このユニットのアクティベーションには影響を与えません。これは、カスタムのユニット依存関係を設定する際に推奨される方法です。
Conflicts	**Requires** と反対の依存関係 (負の依存関係) を設定します。

[a] [Unit] で設定可能なオプションの完全リスト

[b] ほとんどの場合、ユニットファイルオプションの **After** および **Before** で依存関係の並び順を設定するだけで十分です。**Wants** (推奨) または **Requires** で要件の依存関係も設定する場合は、依存関係の並び順を指定する必要があります。これは、並び順と要件の依存関係が相互に依存していないためです。

表10.10 [Service] セクションの重要なオプション

オプション[a] セクション、systemd.service(5) のマニュアルページを参照してください。]	詳細

オプション[a] セクション、systemd.service(5) のマニュアルページを参照してください。]	詳細
Type	**ExecStart** および関連オプションの機能に影響を与えるユニットプロセスの起動タイプを設定します。以下のいずれかになります。 * **simple** - デフォルト値です。**ExecStart** で起動するプロセスは、サービスのメインプロセスです。 * **forking** - **ExecStart** で起動するプロセスは、サービスのメインプロセスになる子プロセスを起動します。親プロセスは、このプロセスが完了すると終了します。 * **oneshot** - このタイプは **simple** と似ていますが、結果として生じるユニットを起動する前に終了します。 * **dbus** - このタイプは **simple** と似ていますが、メインプロセスが D-Bus 名を取得する前に、結果として生じるユニットが起動します。 * **notify** - このタイプは **simple** と似ていますが、結果として生じるユニットは、通知メッセージが sd_notify() 関数で送信されないと起動しません。 * **idle** - **simple** と似ていますが、サービスバイナリーの実行は、すべてのジョブが終了するまで行いません (遅らせます)。 これにより、ステータスの出力とサービスのシェル出力を分けることができます。
ExecStart	ユニットの開始時に実行するコマンドまたはスクリプトを指定します。**ExecStartPre** および **ExecStartPost** は、**ExecStart** の前後に実行するカスタムコマンドを指定します。**Type=oneshot** を使用すれば、連続して実行する複数のカスタムコマンドを指定できます。
ExecStop	ユニットの停止時に実行するコマンドまたはスクリプトを指定します。
ExecReload	ユニットの再読み込み時に実行するコマンドまたはスクリプトを指定します。
Restart	このオプションを有効にすると、**systemctl** コマンドによる完全な停止の例外により、そのプロセスの終了後にサービスが再起動します。

オプション[a] セクション、systemd.service(5) のマニュアルページを参照してください。]	詳細
RemainAfterExit	True に設定すると、サービスは、そのプロセスがすべて終了していてもアクティブと見なされます。デフォルトの値は False です。このオプションは、特に **Type=oneshot** が設定されている場合に役に立ちます。

[a] [Service] で設定可能なオプションの完全リスト

表10.11 [Install] セクションの重要なオプション

オプション[a] セクション、systemd.unit(5) man ページ]	詳細
Alias	ユニット名の追加リストを、スペース区切りで提供します。**systemctl enable** を除くほとんどの **systemctl** コマンドでは、ユニット名ではなくエイリアスを使用できます。
RequiredBy	そのユニットに依存するユニットのリストです。このユニットが有効な場合に、**RequiredBy** にリスト表示されるユニットは、このユニットに関する **Require** 依存関係を取得します。
WantedBy	このユニットへの依存が弱いユニットのリストです。このユニットが有効になると、**WantedBy** にリスト表示されるユニットが、このユニットに関する **Want** 依存関係を取得します。
Also	対応するユニットと共にインストールまたはアンインストールされるユニットのリストを指定します。
DefaultInstance	インスタンス化されているユニットだけが対象となりますが、このオプションは、ユニットが有効になっているデフォルトのインスタンスを指定します。「インスタンス化されたユニットの使用」 を参照してください。

[a] [Install] で設定可能なオプションの完全リスト

ユニット設定の調整に使用できるさまざまなオプションは、例10.17「postfix.service ユニットファイル」 では、システムにインストールされているサービスユニットの例を示しています。さらに、ユニットファイルのオプションは、「インスタンス化されたユニットの使用」 に説明されているようにユニットの動的な作成が可能な方法で定義できます。

例10.17 postfix.service ユニットファイル

次に、**postfix** パッケージで現在提供されている **/usr/lib/systemd/system/postfix.service** ユニットファイルの内容が続きます。

```
[Unit]
Description=Postfix Mail Transport Agent
After=syslog.target network.target
Conflicts=sendmail.service exim.service

[Service]
Type=forking
PIDFile=/var/spool/postfix/pid/master.pid
EnvironmentFile=-/etc/sysconfig/network
ExecStartPre=-/usr/libexec/postfix/aliasesdb
ExecStartPre=-/usr/libexec/postfix/chroot-update
ExecStart=/usr/sbin/postfix start
ExecReload=/usr/sbin/postfix reload
ExecStop=/usr/sbin/postfix stop

[Install]
WantedBy=multi-user.target
```

[Unit] セクションは、サービスについて説明し、競合するユニットおよび並び順依存関係を指定します。[Service] には、ユニットのアクティブ化の実行時、停止時、および再ロード時に、一連のカスタムスクリプトを実行するように指定します。**EnvironmentFile** は、サービスの環境変数が定義する場所を指定し、**PIDFile** は、サービスのメインプロセスに安定した PID を指定します。最後に、[Install] セクションはサービスに依存するユニットをリスト表示します。

10.6.2. カスタムユニットファイルの作成

ユニットファイルをゼロから作成するための複数のユースケースがあります。カスタムデーモンの実行、一部の既存のサービスのセカンドインスタンスの作成 (例10.19「sshd サービスの 2 つ目のインスタンスの作成」)、または SysV init スクリプト (「SysV Init スクリプトのユニットファイルへの変換」) のインポートを行うことができます。一方、既存のユニットの動作を変更または拡張する場合は、「既存のユニットファイルの変更」 の手順を使用します。以下の手順では、カスタムサービスを作成する一般的なプロセスを説明します。

1. カスタムサービスで実行可能なファイルを用意します。カスタムで作成されたスクリプトや、ソフトウェアプロバイダーが提供する実行ファイルがこれにあたります。必要な場合は、カスタムサービスのメインプロセスの PID を保持するため、PID ファイルを用意します。また、サービスのシェル変数を保存するために環境ファイルを組み込むこともできます。(**chmod a+x** を実行して) ソーススクリプトが実行でき、インタラクティブではないことを確認してください。

2. **/etc/systemd/system/** ディレクトリーにユニットファイルを作成し、ファイルに適切なパーミッションがあることを確認します。**root** で以下のコマンドを実行します。

   ```
   touch /etc/systemd/system/name.service
   chmod 664 /etc/systemd/system/name.service
   ```

   name を、作成するサービスの名前に置き換えます。ファイルには実行権限が必要ありません。

3. 上の手順で作成した **name.service** ファイルを開き、サービス設定オプションを追加します。作成するサービスのタイプに応じて、さまざまなオプションを使用できます。「ユニットファ

イル構造の概要」 を参照してください。以下は、ネットワーク関連サービスのユニットの設定例になります。

```
[Unit]
Description=service_description
After=network.target

[Service]
ExecStart=path_to_executable
Type=forking
PIDFile=path_to_pidfile

[Install]
WantedBy=default.target
```

詳細は以下のようになります。

- service_description は、ジャーナルログファイルおよび **systemctl status** コマンドの出力に表示される役立つ説明です。
- **After** 設定により、このサービスは、ネットワークが実行してから起動します。関連するサービスまたはターゲットは、スペースで区切って追加します。
- path_to_executable は、サービス実行ファイルへのパスを表します。
- **Type=forking** は、fork システム呼び出しを行うデーモンに使用します。サービスのメインプロセスは、**path_to_pidfile** で指定した PID で作成されます。その他の起動タイプは 表 10.10「[Service] セクションの重要なオプション」 を参照してください。
- **WantedBy** では、サービスを起動するターゲットを指定します。ターゲットは、従来のランレベル概念の代替と見なすことができます。詳細は 「systemd ターゲットでの作業」 を参照してください。

4. **root** で以下のコマンドを実行すると、新しい **name.service** ファイルが存在することが、systemd に通知されます。

```
systemctl daemon-reload
systemctl start name.service
```

警告

新しいユニットファイルを作成したり、既存のユニットファイルを修正したら常に **systemctl daemon-reload** コマンドを実行します。このコマンドを実行しないと、systemd のステータスと、ディスクの実際のサービスユニットファイルが一致しなくなるため、**systemctl start** コマンドや **systemctl enable** コマンドが失敗します。

name.service ユニットは、 「システムサービスの管理」 に説明通りにコマンドでその他のシステムサービスとして管理できます。

例10.18 emacs.service ファイルの作成

Emacs テキストエディターを使用する場合は、ファイルの編集時にプログラムの新規インスタンスを起動するよりも、バックグラウンドで実行する方がスピードが速いため便利です。以下の手順では、Emacs のユニットファイルを作成し、これをサービスのように処理する方法を説明します。

1. **/etc/systemd/system/** ディレクトリーにユニットファイルを作成し、ファイルパーミッションが正しいことを確認してください。**root** で以下のコマンドを実行します。

```
~]# touch /etc/systemd/system/emacs.service
~]# chmod 664 /etc/systemd/system/emacs.service
```

2. 以下の内容をファイルに追加します。

```
[Unit]
Description=Emacs: the extensible, self-documenting text editor

[Service]
Type=forking
ExecStart=/usr/bin/emacs --daemon
ExecStop=/usr/bin/emacsclient --eval "(kill-emacs)"
Environment=SSH_AUTH_SOCK=%t/keyring/ssh
Restart=always

[Install]
WantedBy=default.target
```

   上記の設定では、サービスの起動時に、**/usr/bin/emacs** 実行ファイルがデーモンモードで開始します。SSH_AUTH_SOCK 環境変数は、ランタイムディレクトリーを表す %t ユニット指定子で設定されます。また、emacs プロセスが予期せずに終了した場合は、このサービスによりこれが再開します。

3. 以下のコマンドを実行して、設定の再読み込みを行い、カスタムサービスを開始します。

```
~]# systemctl daemon-reload
~]# systemctl start emacs.service
```

これでエディターは systemd サービスとして登録されるため、標準的な **systemctl** コマンドがすべて使用できます。たとえば、**systemctl status emacs** でエディターのステータスを表示したり、**systemctl enable emacs** でシステム起動時にエディターを自動的に起動したりできます。

例10.19 sshd サービスの 2 つ目のインスタンスの作成

システム管理者は、サービスのインスタンスを複数設定し、実行しなければならないことが多々あります。これは、サービスの主なインスタンスとの競合を避けるために、元のサービス設定ファイルのコピーを作成し、特定のパラメーターを変更することで実行します。以下の手順は、**sshd** サービスの 2 つ目のインスタンスを作成する方法を示しています。

1. 2 つ目のデーモンで使用する、**sshd_config** ファイルのコピーを作成します。

```
~]# cp /etc/ssh/sshd{,-second}_config
```

2. 作成した **sshd-second_config** ファイルを編集し、2 つ目のデーモンに別のポート番号と PID ファイルを割り当てます。

```
Port 22220
PidFile /var/run/sshd-second.pid
```

Port オプションおよび **PidFile** オプションの詳細は、man ページの **sshd_config**(5) を参照してください。他のサービスで使用されていないポートを選択してください。PID ファイルはサービスの実行時に存在していなければいけないものではありません。存在しない場合は、サービスの起動時に自動的に生成されます。

3. **sshd** サービスの systemd ユニットファイルのコピーを作成します。

```
~]# cp /usr/lib/systemd/system/sshd.service /etc/systemd/system/sshd-second.service
```

4. 作成した **sshd-second.service** を以下のように変更します。

   a. **Description** オプションを変更します。

```
Description=OpenSSH server second instance daemon
```

   b. **After** オプションを指定するサービスに sshd.service を追加し、最初のインスタンスが起動した場合に限り 2 つ目のインスタンスが起動するようにします。

```
After=syslog.target network.target auditd.service sshd.service
```

   c. sshd の最初のインスタンスには鍵の生成が含まれるため、**ExecStartPre=/usr/sbin/sshd-keygen** 行を削除します。

   d. **sshd** コマンドに **-f /etc/ssh/sshd-second_config** パラメーターを追加して、代替の設定ファイルが使用されるようにします。

```
ExecStart=/usr/sbin/sshd -D -f /etc/ssh/sshd-second_config $OPTIONS
```

   e. 上記のように変更すると、sshd-second.service は以下のようになります。

```
[Unit]
Description=OpenSSH server second instance daemon
After=syslog.target network.target auditd.service sshd.service

[Service]
EnvironmentFile=/etc/sysconfig/sshd
ExecStart=/usr/sbin/sshd -D -f /etc/ssh/sshd-second_config $OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s

[Install]
WantedBy=multi-user.target
```

5. SELinux を使用している場合は、sshd の 2 番目のインスタンスのポートを SSH ポートに追加します。追加しないと、sshd の 2 番目のインスタンスがポートにバインドされません。

```
~]# semanage port -a -t ssh_port_t -p tcp 22220
```

6. システムの起動時にこのサービスが自動的に起動するように、sshd-second.service を有効にします。

```
~]# systemctl enable sshd-second.service
```

systemctl status コマンドを使用して sshd-second.service が実行中かどうかを確認します。さらに、ポートをサービスに接続して、適切に有効になっているかどうかを確認します。

```
~]$ ssh -p 22220 user@server
```

ファイアウォールを使用している場合は、sshd の 2 番目のインスタンスへの接続を許可するように適切に設定されていることを確認してください。

カスタムユニットファイルの並び順および依存関係のターゲットを適切に選択する方法は、以下の Red Hat ナレッジベースの記事を参照してください。

- How to write a service unit file which enforces that particular services have to be started
- How to decide what dependencies a systemd service unit definition should have

ユニットファイルの並び順および依存関係にトリガーされたケースの実例の一部と追加情報は、Is there any useful information about writing unit files? を参照してください。

systemd が開始するサービスへの制限の設定は、Red Hat ナレッジベースの記事 RHEL 7 および systemd でサービスに制限を設定する を参照してください。この制限は、サービスのユニットファイルで設定する必要があります。**systemd** は、**/etc/security/limits.conf** 設定ファイルおよび **/etc/security/limits.d/*.conf** 設定ファイルに設定した制限を無視する点に注意してください。このファイルに定義した制限は、ログインセッションの開始時に PAM により設定されますが、**systemd** が開始するデーモンは、PAM ログインセッションを使用しません。

10.6.3. SysV Init スクリプトのユニットファイルへの変換

SysV init スクリプトをユニットファイルに変換する前に、すでに別の場所で変換が行われていないことを確認します。Red Hat Enterprise Linux 7 にインストールされるすべてのコアサービスにデフォルトのユニットファイルが同梱されていますが、数多くのサードパーティーソフトウェアパッケージにも同様のことが言えます。

init スクリプトをユニットファイルに変換するには、スクリプトを分析し、そこから必要な情報を抽出することが必要になります。このデータに基づいて、「カスタムユニットファイルの作成」 の説明通りにユニットファイルを作成できます。init スクリプトはサービスのタイプによって大きく異なるため、この章で概略されているよりも多くの設定オプションの変換を使用しなければならない場合もあります。init スクリプトで利用できるカスタマイズの一部のレベルは systemd ユニットでサポートされなくなっていることに注意してください。「互換性の変更点」 を参照してください。

変換に必要とされるほとんどの情報はスクリプトのヘッダーに提供されます。以下の例は、Red Hat Enterprise Linux 6 で **postfix** サービスを起動するために使用される init スクリプトの開始セクションになります。

```
#!/bin/bash
#
```

```
# postfix   Postfix Mail Transfer Agent
#
# chkconfig: 2345 80 30
# description: Postfix is a Mail Transport Agent, which is the program \
#       that moves mail from one machine to another.
# processname: master
# pidfile: /var/spool/postfix/pid/master.pid
# config: /etc/postfix/main.cf
# config: /etc/postfix/master.cf

### BEGIN INIT INFO
# Provides: postfix MTA
# Required-Start: $local_fs $network $remote_fs
# Required-Stop: $local_fs $network $remote_fs
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: start and stop postfix
# Description: Postfix is a Mail Transport Agent, which is the program that
#       moves mail from one machine to another.
### END INIT INFO
```

上記の例では、**# chkconfig** および **# description** で始まる行のみが必須になり、init ファイルによってはその他の記載はない可能性もあります。**# BEGIN INIT INFO** と **# END INIT INFO** *行に囲まれたテキストは* **Linux Standard Base (LSB) ヘッダー** *と呼ばれています。LSB ヘッダーを指定している場合は、サービスの説明、依存関係、およびデフォルトのランレベルを定義するディレクティブがこれに含まれます。次に、新規のユニットファイルに必要なデータを収集する分析タスクの概要が続きます。postfix init スクリプトはサンプルとして使用されます。作成される postfix ユニットは* 例 10.17「postfix.service ユニットファイル」 *を参照してください。*

サービス説明の検索
#description *で始まる行で、スクリプトに関する説明情報を確認します。この説明は、サービス名と共に、ユニットファイルの [Unit] セクションの* ***Description*** *オプションで使用します。LSB ヘッダーの* **#Short-Description** *行および* **#Description** *行に同様のデータが含まれる場合があります。*

サービス依存関係の検索
*LSB ヘッダーには、サービス間の依存関係を形成する複数のディレクティブが含まれる場合があります。そのほとんどは systemd ユニットオプションに変換できます。*表10.12「LSB ヘッダーの依存関係オプション」 *を参照してください。*

表10.12 LSB ヘッダーの依存関係オプション

LSB オプション	説明	同等のユニットファイル
Provides	サービスの起動ファシリティー名を指定します。この名前は他の init スクリプトで参照できます ("$" 接頭辞を使用)。ただし、ユニットファイルが他のユニットをファイル名で参照できるようになったため、これは不要になりました。	–

LSB オプション	説明	同等のユニットファイル
Required-Start	必要なサービスの起動ファシリティー名が含まれます。これは、並び順の依存関係として変換され、起動ファシリティー名は、対応するサービスまたはそのサービスが属するターゲットに置き換えられます。たとえば、**postfix** の場合、$network の Required-Start 依存関係は、network.target の After 依存関係に変換されました。	**After**、**Before**
Should-Start	Required-Start よりも弱い依存関係を設定します。Should-Start 依存関係が失敗しても、サービスの起動には影響を及ぼしません。	**After**、**Before**
Required-Stop、**Should-Stop**	負の依存関係を設定します。	**Conflicts**

サービスのデフォルトターゲットの検索
*#chkconfig で始まる行には 3 つの数値があります。最も重要な値は最初の数値で、サービスが起動するデフォルトのランレベルを示しています。これらのランレベルを同等の systemd ターゲットにマップするには、表10.6「SysV ランレベルと systemd ターゲットの比較」 を使用します。次に、そのターゲットを、ユニットファイルの [Install] セクションの **WantedBy** オプションに記述します。たとえば、**postfix** はこれまではランレベル 2、3、4、および 5 で起動しました。これは、Red Hat Enterprise Linux 7 では multi-user.target と graphical.target に変換されます。ただし、graphical.target は multiuser.target に依存するため、例10.17「postfix.service ユニットファイル」 の説明にあるように、両方を記述する必要はありません。また、LSB ヘッダーの #Default-Start および #Default-Stop 行に、デフォルト、および動作するべきでないランレベルの情報がある場合は、そちらも参照してください。*

#chkconfig 行で指定した他の 2 つの値は、init スクリプトの起動およびシャットダウンの優先順位を表します。この値は、init スクリプトが読み込まれる場合は systemd により解釈されますが、同等のユニットファイルはありません。

サービスで使用されるファイルの検索
*init スクリプトでは、専用ディレクトリーから関数ライブラリーを読み込み、設定ファイル、環境ファイル、および PID ファイルのインポートを許可します。環境変数は init スクリプトヘッダーの #config で始まる行で指定し、**EnvironmentFile** ユニットファイルオプションに変換されます。#pidfile init スクリプト行に指定した PID ファイルは、**PIDFile** オプションでユニットファイルにインポートされます。*

*init スクリプトヘッダーに含まれない主要な情報は、サービス実行ファイルへのパス、またはサービスで必要になる可能性のあるその他のファイルへのパスです。以前のバージョンの Red Hat Enterprise Linux では、init スクリプトは、カスタム定義のアクションと共に **起動**、**停止**、**再起動** などのデフォルトアクションのサービスの動作を定義する Bash ケーステートメントを使用しました。**postfix** init スクリプトからの以下の抜粋は、サービス起動時に実行するコードのブロックを示しています。*

```
conf_check() {
  [ -x /usr/sbin/postfix ] || exit 5
```

```
  [ -d /etc/postfix ] || exit 6
  [ -d /var/spool/postfix ] || exit 5
}

make_aliasesdb() {
 if [ "$(/usr/sbin/postconf -h alias_database)" == "hash:/etc/aliases" ]
 then
  # /etc/aliases.db might be used by other MTA, make sure nothing
  # has touched it since our last newaliases call
  [ /etc/aliases -nt /etc/aliases.db ] ||
   [ "$ALIASESDB_STAMP" -nt /etc/aliases.db ] ||
   [ "$ALIASESDB_STAMP" -ot /etc/aliases.db ] || return
  /usr/bin/newaliases
  touch -r /etc/aliases.db "$ALIASESDB_STAMP"
 else
  /usr/bin/newaliases
 fi
}

start() {
 [ "$EUID" != "0" ] && exit 4
 # Check that networking is up.
 [ ${NETWORKING} = "no" ] && exit 1
 conf_check
 # Start daemons.
 echo -n $"Starting postfix: "
 make_aliasesdb >/dev/null 2>&1
 [ -x $CHROOT_UPDATE ] && $CHROOT_UPDATE
 /usr/sbin/postfix start 2>/dev/null 1>&2 && success || failure $"$prog start"
 RETVAL=$?
 [ $RETVAL -eq 0 ] && touch $lockfile
    echo
 return $RETVAL
}
```

*init スクリプトの拡張性により、**start()** 関数ブロックから呼び出される **conf_check()** および **make_aliasesdb()** の 2 つのカスタム関数を指定することができました。さらに詳しくみると、上記コードでは外部のファイルおよびディレクトリーが複数記述されています (主なサービス実行ファイル **/usr/sbin/postfix**、設定ディレクトリー **/etc/postfix/**、**/var/spool/postfix/**、および **/usr/sbin/postconf/** ディレクトリー) 。*

*systemd は、事前に定義されたアクションのみをサポートしますが、オプションの **ExecStart**、**ExecStartPre**、**ExecStartPost**、**ExecStop**、**ExecReload** でカスタムの実行ファイルを有効にできます。Red Hat Enterprise Linux 7 の **postfix** の場合、**/usr/sbin/postfix** はサポートスクリプトと共に、サービスの起動時に実行されます。**postfix** ユニットファイルの詳細は、例 10.17「postfix.service ユニットファイル」 を参照してください。*

複雑な init スクリプトを変換する際には、スクリプトのすべてのステートメントの目的を理解している必要があります。一部のステートメントはオペレーティングシステムのバージョンに固有のものであるため、そのステートメントを変換する必要はありません。一方、新規の環境では、サービス実行ファイルおよびサポートファイルやユニットファイルで調整が一部必要となる場合があります。

10.6.4. 既存のユニットファイルの変更

*システムにインストールされるサービスは、**/usr/lib/systemd/system/** ディレクトリーに保存されるデフォルトのユニットファイルと共に提供されます。システム管理者はこのファイルを直接変更できない*

*ため、カスタマイズは **/etc/systemd/system/** ディレクトリーの設定ファイルに制限される必要があります。必要とされる変更の程度に応じて、以下の方法のいずれかを実施してください。*

- *補助設定ファイルのディレクトリーを **/etc/systemd/system/unit.d/** に作成します。この方法は、ほとんどのユースケースで推奨されます。これにより、元のユニットファイルを参照しつつも、デフォルト設定を追加の機能で拡張できます。この場合、パッケージのアップグレードで導入されるデフォルトユニットへの変更は自動的に適用されます。詳細は、「デフォルトのユニット設定の拡張」 を参照してください。*

- *元のユニットファイル **/usr/lib/systemd/system/** のコピーを **/etc/systemd/system/** に作成し、そこで変更を行います。コピーは元のファイルを上書きするため、パッケージの更新で導入される変更は適用されません。この方法は、パッケージの更新とは無関係に永続する重要なユニット変更を行う際に役に立ちます。詳しくは 「デフォルトのユニット設定の上書き」 を参照してください。*

*ユニットのデフォルト設定に戻るには、**/etc/systemd/system/** でカスタム作成した設定ファイルを削除します。システムを再起動せずにユニットファイルへの変更を適用するには、以下を実行します。*

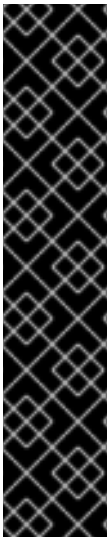
```
systemctl daemon-reload
```

***daemon-reload** オプションは、すべてのユニットファイルを再読み込みし、ユニットファイルへの変更をすぐに適用するのに必要な依存関係ツリー全体を再作成します。別の方法として、次のコマンドで同じ結果を得ることができます。*

```
init q
```

変更したユニットファイルが実行中のサービスに属する場合は、このサービスを再起動して新たな設定を反映させる必要があります。

```
systemctl restart name.service
```



重要

*SysV init スクリプトが処理しているサービスのプロパティー (依存関係やタイムアウトなど) を変更するときは、init スクリプト自体は変更しないでください。代わりに、「デフォルトのユニット設定の拡張」 と 「デフォルトのユニット設定の上書き」 にあるように、サービスの **systemd** ドロップイン設定ファイルを作成します。その後、通常の **systemd** サービスと同じ方法でサービスを管理します。*

*たとえば、**network** サービスの設定を拡張するときは、init スクリプトファイル **/etc/rc.d/init.d/network** を変更しないでください。代わりに、新しいディレクトリー **/etc/systemd/system/network.service.d/** と、**systemd** ドロップインファイル **/etc/systemd/system/network.service.d/my_config.conf** を作成します。そして、ドロップインファイルの値を変更します。**systemd** は、**network** サービスを **network.service** として認識することに注意してください。作成したディレクトリーが **network.service.d** と命名されるのはそのためです。*

デフォルトのユニット設定の拡張
*追加の設定オプションでデフォルトのユニットファイルを拡張するには、最初に **/etc/systemd/system/** に設定ディレクトリーを作成します。サービスユニットを拡張する場合は、**root** で以下のコマンドを実行します。*

```
mkdir /etc/systemd/system/name.service.d/
```

name を、拡張する必要のあるサービスの名前に置き換えます。上記の構文はすべてのユニットタイプに適用されます。

作成したディレクトリーに設定ファイルを作成します。ファイル名の接尾辞は **.conf** にする必要があることに注意してください。以下のコマンドを実行します。

```
touch /etc/systemd/system/name.service.d/config_name.conf
```

config_name を、設定ファイルの名前に置き換えます。このファイルは、通常のユニットファイル構造に基づくため、すべてのディレクティブは該当するセクションで指定する必要があります。「ユニットファイル構造の概要」 を参照してください。

たとえば、カスタムの依存性を追加するには、以下の内容で設定ファイルを作成します。

```
[Unit]
Requires=new_dependency
After=new_dependency
```

new_dependency は、依存性としてマークが付けられるユニットを表します。次の例は、30 秒の遅延後のメインプロセス終了後にサービスを再起動する設定ファイルです。

```
[Service]
Restart=always
RestartSec=30
```

1 つのタスクだけを扱う簡単な設定ファイルを作成することが推奨されます。これにより、他のサービスの設定ディレクトリーに簡単に移動したり、リンクできます。

そのユニットに行った変更を適用するには、**root** で以下のコマンドを実行します。

```
systemctl daemon-reload
systemctl restart name.service
```

例10.20 httpd.service 設定の拡張

Apache サービスの起動時にカスタムシェルスクリプトが自動的に実行されるように httpd.service ユニットを変更するには、以下の手順で行います。まず、ディレクトリーおよびカスタム設定ファイルを作成します。

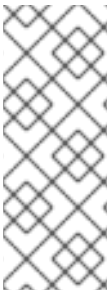
```
~]# mkdir /etc/systemd/system/httpd.service.d/
~]# touch /etc/systemd/system/httpd.service.d/custom_script.conf
```

Apache で自動的に起動するスクリプトが **/usr/local/bin/custom.sh** にある場合は、以下のテキストを **custom_script.conf** ファイルに追加します。

```
[Service]
ExecStartPost=/usr/local/bin/custom.sh
```

ユニットの変更を適用するには、以下を実行します。

```
~]# systemctl daemon-reload
~]# systemctl restart httpd.service
```



注記

/etc/systemd/system/ の設定ディレクトリーの設定ファイル は、/usr/lib/systemd/system/ のユニットファイルに優先します。そのため、設定ファイルに、一度だけ指定できるオプション (Description、ExecStart など) が含まれる場合は、このオプションのデフォルト値が上書きされます。「上書きされたユニットの監視」 で説明されているように、systemd-delta コマンドの出力では、一部のオプションは実際に上書きされますが、該当のユニットは常に [EXTENDED] とマークされます。

デフォルトのユニット設定の上書き
ユニットファイルを提供するパッケージの更新後も変更を持続させるには、最初にファイルを /etc/systemd/system/ ディレクトリーにファイルをコピーします。それを行うには、root で以下のコマンドを実行します。

```
cp /usr/lib/systemd/system/name.service /etc/systemd/system/name.service
```

name *は、変更するサービスユニットの名前を表します。上記の構文はすべてのユニットタイプに適用されます。*

コピーされたファイルをテキストエディターで開き、必要な変更を行います。ユニットの変更を適用するには、root で以下のコマンドを実行します。

```
systemctl daemon-reload
systemctl restart name.service
```

例10.21 タイムアウト制限の変更

サービスごとにタイムアウト値を指定すると、正常に動作していないサービスによってシステムがフリーズすることを防ぐことができます。タイムアウト値を指定しないサービスには、通常のサービスの場合は 90 秒、そして SysV と互換性のあるサービスの場合は 300 秒と、それぞれデフォルトのタイムアウトが設定されています。

たとえば、httpd サービスのタイムアウト制限を延長するときは、以下を行います。

1. *httpd ユニットファイルを、/etc/systemd/system/ ディレクトリーにコピーします。*

```
cp /usr/lib/systemd/system/httpd.service /etc/systemd/system/httpd.service
```

2. */etc/systemd/system/httpd.service ファイルを開き、[Service] セクションに TimeoutStartSec 値を指定します。*

```
...
[Service]
...
PrivateTmp=true
TimeoutStartSec=10

[Install]
WantedBy=multi-user.target
...
```

3. *systemd デーモンを再ロードします。*

```
systemctl daemon-reload
```

4. **任意です。** *新しいタイムアウト値を確認します。*

```
systemctl show httpd -p TimeoutStartUSec
```

注記

*グローバルでタイムアウト制限を変更するには、**/etc/systemd/system.conf** ファイルの **DefaultTimeoutStartSec** を変更します。「systemd の概要」を参照してください。*

上書きされたユニットの監視
上書きされたユニット、または変更したユニットファイルの概要を表示するには、以下のコマンドを実行します。

```
systemd-delta
```

上記のコマンドを実行すると、以下のような出力になります。

```
[EQUIVALENT] /etc/systemd/system/default.target → /usr/lib/systemd/system/default.target
[OVERRIDDEN] /etc/systemd/system/autofs.service → /usr/lib/systemd/system/autofs.service

--- /usr/lib/systemd/system/autofs.service   2014-10-16 21:30:39.000000000 -0400
+++ /etc/systemd/system/autofs.service 2014-11-21 10:00:58.513568275 -0500
@@ -8,7 +8,8 @@
 EnvironmentFile=-/etc/sysconfig/autofs
 ExecStart=/usr/sbin/automount $OPTIONS --pid-file /run/autofs.pid
 ExecReload=/usr/bin/kill -HUP $MAINPID
-TimeoutSec=180
+TimeoutSec=240
+Restart=Always

 [Install]
 WantedBy=multi-user.target

[MASKED]   /etc/systemd/system/cups.service → /usr/lib/systemd/system/cups.service
[EXTENDED]  /usr/lib/systemd/system/sssd.service →
/etc/systemd/system/sssd.service.d/journal.conf

4 overridden configuration files found.
```

*表10.13「systemd-delta の相違タイプ」は、**systemd-delta** の出力で表示される上書きタイプをリスト表示します。ファイルが上書きされると、**systemd-delta** が、**diff** コマンドの出力に似た変更の要約をデフォルトで表示します。*

表10.13 systemd-delta の相違タイプ

Type	詳細
[MASKED]	マスクされたユニットファイルです。ユニットマスクの説明は、「サービスの無効化」を参照してください。

Type	詳細
[EQUIVALENT]	このコピーは、元のファイルを上書きしますが、コンテンツは変更されません。通常はシンボリックリンクです。
[REDIRECTED]	別のファイルにリダイレクトするファイルです。
[OVERRIDEN]	上書きされ、変更されたファイルです。
[EXTENDED]	**/etc/systemd/system/unit.d/** ディレクトリーの .conf ファイルで拡張されているファイルです。
[UNCHANGED]	**--type=unchanged** オプションが使用される場合にのみ表示される、変更されていないファイルです。

*システムの更新後に、**systemd-delta** を実行して、デフォルトユニットに対してカスタム設定で上書きした更新があるかどうかを確認できます。さらに、出力する更新を、特定のタイプに制限することもできます。たとえば、上書きされたユニットのみを表示するには、以下のコマンドを実行します。*

```
systemd-delta --type=overridden
```

10.6.5. インスタンス化されたユニットの使用

*ランタイム時に、1 つのテンプレート設定ファイルから複数のユニットをインスタンス化できます。@文字は、テンプレートにマークを付け、ユニットをこれに関連付けるために使用されます。インスタンス化されたユニットは、(**Requires** オプションまたは **Wants** オプションを使用して) 別のユニットから開始することも、**systemctl start** コマンドで開始することもできます。インスタンス化されたサービスユニットの名前は以下のような形式となります。*

```
template_name@instance_name.service
```

*ここで、**template_name** は、テンプレート設定ファイルの名前になります。**instance_name** を、ユニットインスタンスの名前に置き換えます。複数のインスタンスが同じテンプレートファイルを参照し、このテンプレートには、ユニットの全インスタンスに共通する設定オプションが含まれます。テンプレートユニットの名前には以下の形式が使用されます。*

```
unit_name@.service
```

*たとえば、ユニットファイルに次の **Wants** 設定を指定すると、*

```
Wants=getty@ttyA.service,getty@ttyB.service
```

*この設定により、systemd が、最初に指定したサービスユニットを検索します。該当するユニットが見つからないと、@とタイプ接尾辞の間にある部分は無視され、systemd が **getty@.service** ファイルを検索し、そこから設定を読み取り、サービスを起動します。*

*ワイルドカード文字 (**ユニット指定子** とも呼ばれる) を、すべてのユニット設定ファイルで使用できます。ユニット指定子は、特定のユニットパラメーターを置き換え、ランタイム時に解釈されます。表 10.14「重要なユニット指定子」 は、特にテンプレートユニットで便利なユニット指定子をリスト表示*

します。

表10.14 重要なユニット指定子

ユニット指定子	意味	説明
%n	完全ユニット名	タイプ接尾辞を含む完全ユニット名を表します。**%N** には同じ意味がありますが、禁止文字を ASCII コードに置き換えます。
%p	接頭辞名	タイプ接尾辞が削除されたユニット名を表します。インスタンス化されたユニットの %p は、ユニット名の@文字の前の部分を表します。
%i	インスタンス名	インスタンス化されたユニット名の@文字およびタイプ接尾辞間の部分です。**%I** には同じ意味がありますが、禁止文字を ASCII コードにも置き換えます。
%H	ホスト名	ユニット設定を読み込んだ時に稼働しているシステムのホスト名を表します。
%t	ランタイムディレクトリー	ランタイムディレクトリーを表します。これは、**root** ユーザーの **/run**、または非特権ユーザーの XDG_RUNTIME_DIR 変数の値になります。

*ユニット指定子の詳細なリストは、man ページの **systemd.unit(5)** を参照してください。*

*たとえば、**getty@.service** テンプレートには以下のディレクティブが含まれます。*

```
[Unit]
Description=Getty on %I
...
[Service]
ExecStart=-/sbin/agetty --noclear %I $TERM
...
```

*上記のテンプレートから getty@ttyA.service および getty@ttyB.service をインスタンス化する場合、**Description**= は* **Getty on ttyA** *および* **Getty on ttyB** *として解決されます。*

10.7. サービスの管理中に考慮すべき事項

通常の操作時に、systemd は、ユニットの抽象化と、システムでアクティブな基本プロセスの関連付けを維持します。

From: ***man systemd***

> *Processes systemd spawns are placed in individual Linux control groups named after the unit which they belong to in the private systemd hierarchy. (see cgroups.txt[1] for more information about control groups, or short "cgroups"). systemd uses this to effectively keep track of processes. Control group information is maintained in the kernel, and is accessible via the file system hierarchy (beneath /sys/fs/cgroup/systemd/), or in tools such as ps(1) (ps xawf -eo pid,user,cgroup,args is particularly useful to list all processes and the systemd units they belong to).*

cgroup 階層は、systemd のプロセスおよびサービスの健全性の表示に重要です。プロセスがそれ自体を分岐すると、プロセスは作成プロセスの cgroup を継承します。この場合は、特定のユニットに関連付けられているすべてのプロセスは、次のような適切な cgroup.procs ファイルの内容を読み取ることで検証できます。

```
~]# cat /sys/fs/cgroup/systemd/system.slice/httpd.service/cgroup.procs
11854
11855
11856
11857
11858
11859
```

*この出力は、**systemctl status** unit 操作時に返される CGroup 情報と一致します。*

```
~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
  Active: active (running) since Wed 2019-05-29 12:08:16 EDT; 45s ago
    Docs: man:httpd(8)
          man:apachectl(8)
 Main PID: 11854 (httpd)
  Status: "Total requests: 0; Current requests/sec: 0; Current traffic:  0 B/sec"
  CGroup: /system.slice/httpd.service
          ├─11854 /usr/sbin/httpd -DFOREGROUND
          ├─11855 /usr/sbin/httpd -DFOREGROUND
          ├─11856 /usr/sbin/httpd -DFOREGROUND
          ├─11857 /usr/sbin/httpd -DFOREGROUND
          ├─11858 /usr/sbin/httpd -DFOREGROUND
          └─11859 /usr/sbin/httpd -DFOREGROUND

May 29 12:08:16 localhost systemd[1]: Starting The Apache HTTP Server...
May 29 12:08:16 localhost systemd[1]: Started The Apache HTTP Server.
```

*システム全体で、このようなプロセスの分類を直接表示するには、**systemd-cgls** ユーティリティーを使用できます。*

```
~]# systemd-cgls | head -17
├─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 22
├─user.slice
│ └─user-0.slice
│   └─session-168.scope
│     ├─ 3049 login -- root
│     ├─11884 -bash
│     ├─11943 systemd-cgls
│     └─11944 head -17
```

```
└─system.slice
  ├─httpd.service
  │ ├─11854 /usr/sbin/httpd -DFOREGROUND
  │ ├─11855 /usr/sbin/httpd -DFOREGROUND
  │ ├─11856 /usr/sbin/httpd -DFOREGROUND
  │ ├─11857 /usr/sbin/httpd -DFOREGROUND
  │ ├─11858 /usr/sbin/httpd -DFOREGROUND
  │ └─11859 /usr/sbin/httpd -DFOREGROUND
  ├─rhnsd.service
```

systemd が適切に機能するには、サービスを systemd システムから起動または停止して、ユニットの分類に対して適切なプロセスを維持する必要があります。外部アクションを実行する操作を行うと、必要な cgroup 構造が作成されません。これは、systemd が、開始されるプロセスの特別な性質を認識しないために発生します。

*上記の制約で、たとえば **httpd** サービスを停止し、**/usr/sbin/httpd** を直接実行すると、以下のようになります。*

```
~]# systemctl stop httpd
~]# /usr/sbin/httpd
# systemd-cgls | head -17
├─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 22
├─user.slice
│ └─user-0.slice
│   └─session-168.scope
│     ├─ 3049 login -- root
│     ├─11884 -bash
│     ├─11957 /usr/sbin/httpd
│     ├─11958 /usr/sbin/httpd
│     ├─11959 /usr/sbin/httpd
│     ├─11960 /usr/sbin/httpd
│     ├─11961 /usr/sbin/httpd
│     ├─11962 /usr/sbin/httpd
│     ├─11963 systemd-cgls
│     └─11964 head -17
└─system.slice
  ├─rhnsd.service
  │ └─3261 rhnsd
```

***httpd** プロセスは、user-0.slice および a session-168.scope の下に表示されることに注意してください。このサービスは systemd が直接監視および管理する必要があるシステムサービスとは対照的に、ユーザーが開始したプロセスとして扱われます。この不整合により発生する可能性のある障害には次のものが含まれますが、これに限定されません。*

- *システムのシャットダウンまたは再起動のイベント時に、サービスが適切にシャットダウンされません。*
- *ユーザーのログアウト中に、SIGHUP や SIGTERM などの予期しない信号が配信されます。*
- ***Restart=** ディレクティブがあるにも関わらず、失敗したプロセスが自動的に再起動しません。*

注記

アプリケーションシャットダウンイベントが正常に行われないと、クライアント側の障害やデータ損失、ディスク上の破損などの多くのアプリケーション障害が発生することがあります。

10.8. 関連情報

systemd および Red Hat Enterprise Linux 7 でのその使用については、以下にリスト表示されているリソースを参照してください。

インストールされているドキュメント

- **systemctl**(1) - **systemctl** コマンドラインユーティリティーの man ページには、サポートされるオプションとコマンドの完全なリストが提供されます。
- **systemd**(1) - **systemd** システムおよびサービスマネージャーの man ページでは、その概念に関する詳細情報が提供され、利用可能なコマンドラインオプションと環境変数、サポートされる設定ファイルとディレクトリー、認識されるシグナル、および利用可能なカーネルオプションが説明されています。
- **systemd-delta**(1): 拡張され、上書きされた設定ファイルの検索を可能にする **systemd-delta** ユーティリティーの man ページです。
- **systemd.unit**(5): **systemd.unit** の man ページでは、systemd ユニットファイルの詳細情報と、利用可能なすべての設定オプションが説明されてます。
- **systemd.service**(5) - **systemd.service** の man ページでは、サービスユニットファイルの形式が紹介されています。
- **systemd.target**(5) - **systemd.target** の man ページには、ターゲットユニットファイルの形式が説明されています。
- **systemd.kill**(5) - **systemd.kill** の man ページでは、プロセスの強制終了手順の設定が説明されています。

オンラインドキュメント

- *Red Hat Enterprise Linux 7 ネットワークガイド*: Red Hat Enterprise Linux 7 の **ネットワークガイド** では、このシステムにおけるネットワークインターフェイス、ネットワーク、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。**hostnamectl** ユーティリティーの概要のほか、これを使用してコマンドラインでホスト名を表示して設定する方法が説明されています。また、選択されたホスト名およびドメイン名に関する重要な情報も提供されています。
- *Red Hat Enterprise Linux 7 デスクトップの移行および管理ガイド*: Red Hat Enterprise Linux 7 の **デスクトップの移行および管理ガイド** は、本システム上での GNOME 3 デスクトップの移行計画、導入、設定、および管理について説明しています。**loginctl** サービスが導入され、最も重要な機能を列挙し、**logind** ユーティリティーを使用してアクティブなセッションをリスト表示し、マルチシートサポートを有効にする方法を説明します。
- *Red Hat Enterprise Linux 7 SELinux ユーザーおよび管理者のガイド*: Red Hat Enterprise Linux 7 の **SELinux ユーザーおよび管理者のガイド** では、SELinux の原則と、SELinux を Apache HTTP Server や Postfix、PostgreSQL、OpenShift などの様々なサービスに設定して使用する方法が詳細に説明されています。また、SELinux アクセスパーミッションを systemd が管理するシステムサービス用に設定する方法も説明しています。
- *Red Hat Enterprise Linux 7 インストールガイド* - Red Hat Enterprise Linux 7 の **インストールガイド** では、AMD64 システムおよび Intel 64 システム、64 ビット IBM Power Systems サーバー、および IBM Z にシステムをインストールする方法を説明します。また、キックスタートインストール、PXE インストール、VNC プロトコルへの高度なインストール方法も取り上げて

います。さらに、インストール後の一般的なタスクと、レスキューモードへの起動方法や root パスワードの回復方法などの詳細な手順を含むインストールの問題に関するトラブルシューティングについても説明されています。

- *Red Hat Enterprise Linux 7 セキュリティーガイド* - Red Hat Enterprise Linux 7 の **セキュリティーガイド** は、ユーザーおよび管理者が、ローカルおよびリモートからの侵入、悪用、悪意のある行為に対してワークステーションおよびサーバーを保護するプロセスとプラクティスを学習する際に役に立ちます。また、重大なシステムサービスを保護する方法についても説明しています。

- *systemd ホームページ* - このプロジェクトのホームページでは、systemd に関する詳細情報が提供されています。

関連項目

- *2章システムロケールおよびキーボード設定* では、システムロケールとキーボードのレイアウトを管理する方法を説明しています。コマンドラインで **localectl** ユーティリティーを使用して現在のロケールを表示、利用可能なロケールをリスト表示、コマンドラインでシステムロケールを設定、現在のキーボードレイアウトを表示、利用可能なキーマップをリスト表示、そして特定のキーボードレイアウトを有効にする方法が説明されています。

- *3章日付と時刻の設定* では、システムの日時を管理する方法を説明しています。リアルタイムクロックとシステムクロックの違いや、**timedatectl** ユーティリティーを使用して現在のシステムクロックの設定を表示、日時を設定、タイムゾーンを変更、およびシステムクロックをリモートサーバーと同期させる方法が説明されています。

- *6章権限の取得* では、**su** および **sudo** コマンドを使用して管理者権限を取得する方法を説明しています。

- *12章OpenSSH* は、SSH サーバーの設定方法や、クライアントユーティリティーの **ssh**、**scp**、および **sftp** を使用してこのサーバーにアクセスする方法を説明しています。

- *23章ログファイルの表示と管理* では、**journald** の概要が提供されています。ジャーナルの説明と **journald** サービスの概要のほか、**journalctl** ユーティリティーを使用してログエントリーを表示する方法、ライブ表示モードへの切り替え、ログエントリーのフィルター方法を説明しています。さらに、この章では、root 以外のユーザーにシステムログへのアクセスを許可し、一貫性のあるログファイルの保存を可能にする方法も説明されています。

第11章 アクセシビリティーのためのシステム設定

Red Hat Enterprise Linux 7 のアクセシビリティーは、オペレーティングシステムのデフォルトインストールに含まれる Orca スクリーンリーダーによって確保されています。本章では、システム管理者が、視覚障害を持つユーザーに有用なシステムを設定する方法を説明します。

Orca は画面から情報を読み取り、以下を使用してユーザーに伝えます。

- 音声合成装置 (音声出力を提供)
- 点字ディスプレイ (触知可能な出力を提供)

Orca の設定に関する詳細は、ヘルプページ をご覧ください。

Orca のコミュニケーション出力を正常に機能させるために、システム管理者は以下を行う必要があります。

- 「**brltty** サービスの設定」 の記載どおりに **brltty** サービスを設定します。
- 「**Always Show Universal Access Menu** をオンにします。」 の記載どおりに、ユニバーサルアクセスメニューを常に表示 **(Always Show Universal Access Menu)** のスイッチを入れます。
- 「Festival Speech Synthesis System の有効化」 の記載どおりに、Festival 音声合成装置を有効にします。

11.1. BRLTTY サービスの設定

点字ディスプレイは、**brltty** サービスを使用して視覚障害のあるユーザーに触知可能な出力を提供します。

brltty サービスを有効する

点字ディスプレイは、**brltty** が実行していないと作動しません。デフォルトでは、**brltty** は無効になっています。**brltty** を有効にし、ブート時に起動するようにします。

```
~]# systemctl enable brltty.service
```

ユーザーへの点字ディスプレイの使用許可

点字ディスプレイの使用を許可されているユーザーを設定するには、以下のいずれか 1 つの手順を選択してください。**/etc/brltty.conf** ファイルを使用した手順は、ユーザーまたはグループをファイルに割り当てることができないファイルシステムにも適しています。**/etc/brlapi.key** ファイルを使用する方法は、ユーザーまたはグループをファイルに割り当てられるファイルシステムのみに適しています。

/etc/brltty.conf を使用した点字ディスプレイへのアクセスの設定

1. **/etc/brltty.conf** ファイルを開き、Application Programming Interface Parameters というセクションを見つけます。
2. ユーザーを指定します。
   a. 1 人以上の個々のユーザーを指定するには、次の行にユーザーを追加します。

   ```
   api-parameters Auth=user:user_1, user_2, ...  # Allow some local user
   ```

   b. ユーザーグループを指定するには、次の行に名前を入力します。

```
api-parameters Auth=group:group  # Allow some local group
```

/etc/brlapi.key を使用して点字ディスプレイへのアクセス設定

1. **/etc/brlapi.key** ファイルを作成します。

```
~]# mcookie > /etc/brlapi.key
```

2. **/etc/brlapi.key** の所有権を特定のユーザーまたはグループに変更します。

   a. 個々のユーザーを指定するには、次のコマンドを実行します。

```
~]# chown user_1 /etc/brlapi.key
```

   b. グループを指定するには、次のコマンドを実行します。

```
~]# chown group_1 /etc/brlapi.key
```

3. **/etc/brltty.conf** の内容を調整し、以下を追加します。

```
api-parameters Auth=keyfile:/etc/brlapi.key
```

点字ドライバーの設定

/etc/brltty.conf ファイルの **braille-driver** ディレクティブは、点字ディスプレイ用のドライバーの 2 文字のドライバー識別コードを指定します。

点字ドライバーの設定

1. 適切な点字ドライバーを見つけるときに、自動検知機能を使用するかどうか決定します。

   a. 自動検出を使用する場合は、点字ドライバー をデフォルトのオプションである **auto** に指定しておきます。

```
braille-driver auto  # autodetect
```

> **警告**
>
> 自動検出はすべてのドライバーを試行します。そのため、時間がかかるか、失敗することさえあります。そのため、特定の点字ドライバーに設定しておくことが推奨されます。

   b. 自動検出を使用しない場合は、**braille-driver** ディレクティブで必要な点字ドライバーの識別コードを指定します。
   /etc/brltty.conf に記載されているリストから、必要な点字ドライバーの識別コードを選択します。たとえば、以下のようになります。

```
braille-driver xw  # XWindow
```

複数のドライバーをコンマで区切って設定することもでき、それらの間で自動検出が実行されます。

点字装置の設定

/etc/brltty.conf *の* ***braille-device*** *ディレクティブは、点字ディスプレイに接続する装置を指定します。以下の装置の種類に対応しています (表11.1「点字装置の種類と対応する構文」 を参照)。*

表11.1 点字装置の種類と対応する構文

点字デバイスの種類	タイプの構文
シリアルデバイス	serial:path [a]
USB デバイス	[serial-number] [b]
Bluetooth デバイス	bluetooth:address

[a] 相対パスは **/dev** にあります。

[b] 括弧はオプションを示します。

以下は、特定の装置の設定例です。

```
braille-device serial:ttyS0        # First serial device
braille-device usb:            # First USB device matching braille driver
braille-device usb:nnnnn       # Specific USB device by serial number
braille-device bluetooth:xx:xx:xx:xx:xx:xx # Specific Bluetooth device by address
```

また、複数のデバイスをコンマで区切って設定することもでき、各デバイスは順番にプローブされます。

警告

デバイスが serial-to-USB アダプターで接続されている場合は、 ***braille-device*** *を* ***usb:*** *に設定しても機能しません。この場合は、カーネルがアダプター用に作成した仮想シリアルデバイスを識別します。仮想シリアルデバイスは次のようになります。*

```
serial:ttyUSB0
```

You can find the actual device name in the kernel messages on the device plug with the following command:

```
~]# dmesg | fgrep ttyUSB0
```

点字ディスプレイへ特定のパラメーターの設定

特定の点字ディスプレイデバイスに特定のパラメーターを設定するには、**/etc/brltty.conf** ファイルで **braille-parameters** ディレクティブを使用します。**braille-parameters** ディレクティブは、非汎用パラメーターを点字ドライバーに渡します。**/etc/brltty.conf** のリストから必要なパラメーターを選択します。

テキストテーブルの設定

/etc/brltty.conf の **text-table** ディレクティブは、シンボルのエンコードに使用されるテキストテーブルを指定します。テキストテーブルの相対パスは **/etc/brltty/Text/** ディレクトリーにあります。

テキストテーブルの設定

1. 適切なテキストテーブルを見つけるために自動選択を使用するかどうかを決定します。
2. a. 自動選択を使用する場合は、**text-table** をデフォルトオプションの **auto** に指定したままにします。

   ```
   text-table auto  # locale-based autoselection
   ```

   これにより、**en-nabcc** へのフォールバックを使用したローカルベースの自動選択が実行されます。

   b. 自動選択を使用しない場合は、**/etc/brltty.conf** のリストから必要な **text-table** を選択します。
   たとえば、アメリカ英語のテキストテーブルを使用するには、次を指定します。

   ```
   text-table en_US  # English (United States)
   ```

Contraction テーブルの設定

/etc/brltty.conf の **contraction-table** ディレクティブは、略語のエンコードに使用されるテーブルを指定します。特定の contraction テーブルの相対パスは **/etc/brltty/Contraction/** ディレクトリーにあります。

/etc/brltty.conf のリストから必要な **contraction-table** を選択します。

たとえば、グレード 2 のアメリカ英語の収縮表を使用するには、次を指定します。

```
contraction-table en-us-g2  # English (US, grade 2)
```

警告

指定しない場合、収縮表は使用されません。

11.2. ALWAYS SHOW UNIVERSAL ACCESS MENU をオンにします。

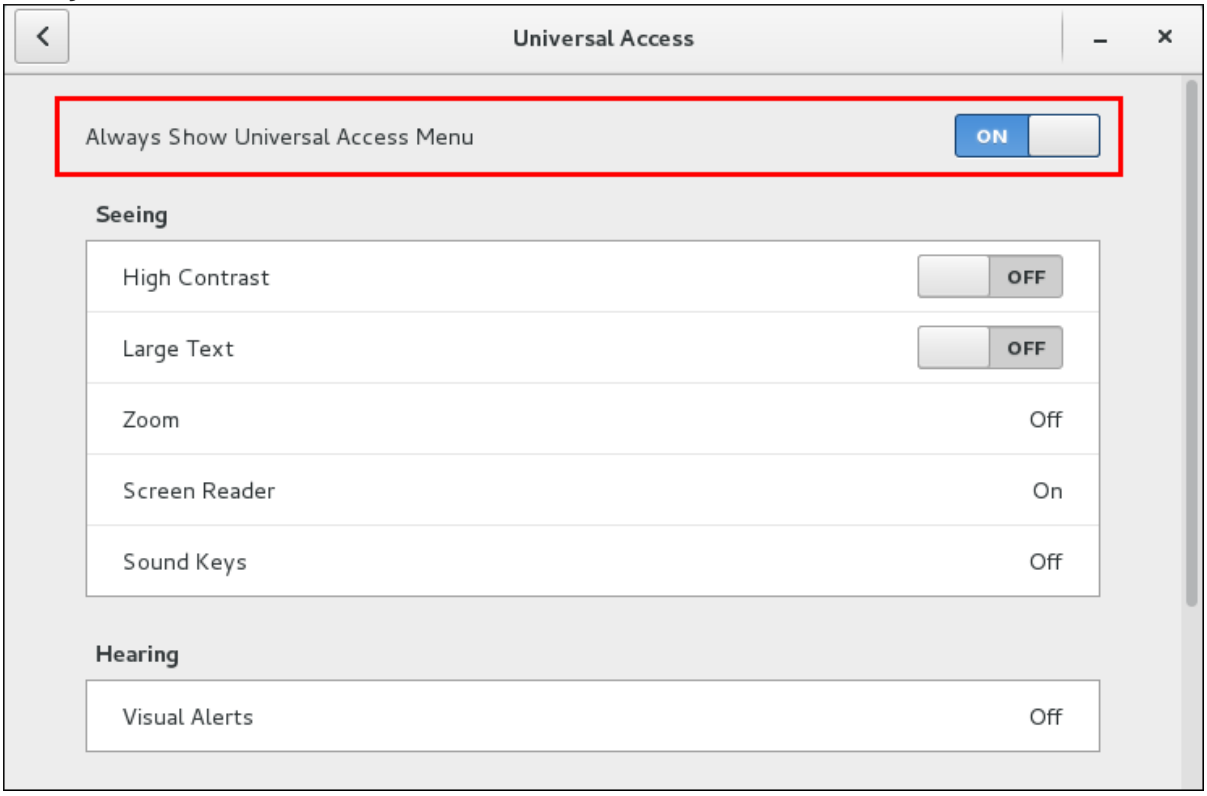
Orca スクリーンリーダーのスイッチを入れるには、**Super+Alt+S** キーを同時に押します。これにより、Universal Access Menu アイコンがトップバーに表示されます。

警告

*ユーザーがユニバーサルアクセスメニュー (Universal Access Menu) にあるすべてのオプションのスイッチをオフにしていると、このアイコンは表示されません。アイコンがないと、視覚障害のあるユーザーのご利用が難しくなります。システム管理者は **Always Show Universal Access Menu** のスイッチをオンにすれば、アイコンが利用できない状態を防げます。**Always Show Universal Access Menu** を有効にすると、このメニューのすべてのオプションが無効になっている場合でも、アイコンがトップバーに表示されます。*

*ユニバーサルアクセスメニューを常に表示 (**Always Show Universal Access Menu**) のスイッチをオンにする*

1. *Gnome Settings メニューを開き、**Universal Access** をクリックします。*
2. ***Always Show Universal Access Menu** をオンにします。*



3. ***オプション:** このメニューのすべてのオプションのスイッチがオフになっていても、ユニバーサルアクセスメニュー (Universal Access Menu) アイコンがトップバーに表示されていることを確認します。*

11.3. FESTIVAL SPEECH SYNTHESIS SYSTEM の有効化

*デフォルトでは **Orca** は **eSpeak** 音声合成装置を使用していますが、**Festival Speech Synthesis System** にも対応しています。**eSpeak** および **Festival Speech Synthesis System** (Festival) は、異なる方法で音声を合成します。一部のユーザーは、デフォルトの **eSpeak**シンセサイザーより、Festival を好む場合があります。Festival を有効化する手順は以下のとおりです。*

Festival のインストールとブート時の実行

1. *Festival をインストールします。*

```
~]# yum install festival festival-freebsoft-utils
```

2. *Festival を起動時に実行します。*

   a. *新規 **systemd** ユニットファイルを作成します。*
   *ファイルを **/etc/systemd/system/** ディレクトリーに作成し、実行可能な状態にします。*

```
~]# touch /etc/systemd/system/festival.service
~]# chmod 664 /etc/systemd/system/festival.service
```

   b. ***/usr/bin/festival_server** ファイルにあるスクリプトが Festival の実行に使用されるよう設定します。以下の内容を **/etc/systemd/system/festival.service** ファイルに追加します。*

```
[Unit]
Description=Festival speech synthesis server
[Service]
ExecStart=/usr/bin/festival_server
Type=simple
```

   c. ***systemd** に新しい **festival.service** ファイルが存在することを通知します。*

```
~]# systemctl daemon-reload
~]# systemctl start festival.service
```

   d. ***festival.service** を有効化します。*

```
~]# systemctl enable festival.service
```

Festival の音声の選択

Festival には複数の音声が用意されています。

音声を利用するには、以下のリストから関連するパッケージをインストールしてください。

- *festvox-awb-arctic-hts*
- *festvox-bdl-arctic-hts*
- *festvox-clb-arctic-hts*
- *festvox-kal-diphone*
- *festvox-ked-diphone*
- *festvox-rms-arctic-hts*
- *festvox-slt-arctic-hts*
- *hispavoces-pal-diphone*
- *hispavoces-sfl-diphone*

特定の音声に関する詳細は以下でご覧ください。

```
~]# yum info package_name
```

必要な音声を利用するには、その音声を含むパッケージをインストールして、再起動します。

```
~]# yum install package_name
~]# reboot
```

第12章 OPENSSH

SSH (Secure Shell) は、クライアント/サーバーアーキテクチャーを使用する 2 つのシステム間でのセキュアな通信を容易にし、ユーザーがリモートでサーバーホストシステムにログインできるようにするプロトコルです。**FTP**、**Telnet** などの他のリモート通信プロトコルとは異なり、SSH はログインセッションを暗号化するため、侵入者が接続して暗号化されていないパスワードを入手するのが困難になります。

ssh プログラムは、**telnet** や **rsh** などのリモートホストへのログインに使用される、旧式で、セキュリティー保護が十分でない端末アプリケーションに代わるものとして設計されています。また、**scp** と呼ばれる関連プログラムが、ホスト間でファイルをコピーするために設計された **rcp** などの旧式プログラムの代わりとなります。このような旧式アプリケーションは、クライアントとサーバーとの間で送信するパスワードを暗号化しないため、可能な限り使用しないようにしてください。リモートシステムへのログインにセキュアな方法を使用することで、クライアントシステムとリモートホストの両方に対するリスクが低減されます。

Red Hat Enterprise Linux には、一般的な **openssh** パッケージと、OpenSSH のサーバー (**openssh-server**) およびクライアント (**openssh-clients**) のパッケージが含まれます。OpenSSH パッケージには、OpenSSL パッケージ (**openssl-libs**) が必要です。このパッケージは、重要な暗号化ライブラリーをインストールし、暗号化通信を提供する OpenSSH を有効にします。

12.1. SSH プロトコル

12.1.1. SSH を使用する理由

潜在的な侵入者は、ネットワークトラフィックの中断、傍受、経路変更を可能にする様々なツールを自由に駆使して、システムに侵入します。一般的には、これらの脅威は以下のとおり分類できます。

2 つのシステム間の通信の傍受

攻撃者は、ネットワーク上で通信を行う二者の間のどこかに潜み、両者間で渡される情報をコピーしている可能性があります。攻撃者は情報を傍受して保持する、または情報を改ざんして元の受信者に送信する場合があります。
このような攻撃は、通常 **パケットスニファー** を使用して行われます。パケットスニファーは、ネットワークを通過する各パケットをキャプチャーしてその内容を分析する、ごく一般的なネットワークユーティリティーです。

特定のホストの偽装

攻撃者のシステムは、送信の対象となる受信者を装うように設定されます。これに成功すると、ユーザーのシステムは不正なホストと通信していることに気がつかないままとなります。
この攻撃は、**DNS ポイズニング** として知られる手法か、**IP スプーフィング** と呼ばれる手法を用いて実行されます。前者の場合、侵入者はクラックされた DNS サーバーを使用して、クライアントシステムを不当に複製されたホストへ指定します。後者の場合、侵入者は、信頼されたホストから送信されたように見せかけた偽装ネットワークパケットを送信します。

いずれの手法でも、潜在的な機密情報を傍受することが可能です。その傍受が悪意のある理由で行われる場合には、多大な損害をもたらしかねません。リモートシェルログインとファイルコピー用に SSH を使用すると、こうしたセキュリティーの脅威を大幅に軽減できます。これは、SSH クライアントとサーバーがデジタル署名を使用してそれぞれの ID を確認するためです。さらに、クライアントシステムとサーバーシステムとの間の通信はすべて暗号化されます。各パケットはローカルシステムとリモートシステムのみに知られている鍵を使用して暗号化されるため、通信のいずれか一方の ID をスプーフィングする試みは成功しません。

12.1.2. 主な特長

SSH プロトコルは、以下のような保護手段を提供します。

対象のサーバーになりすますことができない

クライアントは、初回接続後に、以前接続したサーバーと同じサーバーに接続していることを確認できます。

認証情報の取得ができない

クライアントは、強力な128 ビット暗号化を使用して、サーバーへ認証情報を送信します。

通信の傍受ができない

セッション中に送受信された全データは、128 ビット暗号化を使用して転送されるため、傍受された送信データの暗号解読と読み取りは非常に困難になります。

さらに、以下のようなオプションも提供されます。

ネットワーク上でグラフィカルアプリケーションを使用するセキュアな手段を提供する

クライアントは、**X11 転送** と呼ばれる手法を使用して、サーバーから **X11** (**X Window System**) アプリケーションを転送できます。

セキュアでないプロトコルをセキュアにする手段を提供する

SSH プロトコルは、送受信するものをすべて暗号化します。SSH サーバーは、**ポート転送** と呼ばれる技術を使用して、**POP** のようなセキュアではないプロトコルをセキュアにし、システムとデータ全体のセキュリティーを強化できます。

セキュアなチャンネルを作成する

OpenSSH サーバーとクライアントは、サーバーマシンとクライアントマシンとの間のトラフィックに対して、仮想プライベートネットワークに似たトンネルを作成するように設定できます。

Kerberos 認証をサポートする

OpenSSH サーバーとクライアントは、Kerberos ネットワーク認証プロトコルの **GSSAPI** (Generic Security Services Application Program Interface: 汎用セキュリティーサービス API) 実装を使用して認証を行うように設定できます。

12.1.3. プロトコルのバージョン

現在、SSH にはバージョン1 とバージョン2 があります。Red Hat Enterprise Linux 7 の OpenSSH スイートは、SSH バージョン2 を使用します。バージョン1 の既知の不正使用の影響を受けない、強化された鍵交換アルゴリズムを備えています。Red Hat Enterprise Linux 7 では、OpenSSH スイートはバージョン1 の接続に対応していません。

12.1.4. SSH 接続のイベントシーケンス

以下に挙げる一連のイベントは、2 つのホスト間で行われる SSH 通信の整合性を保護するのに役立ちます。

1. 暗号化ハンドシェイクが行われ、クライアントが正しいサーバーと通信していることを確認できます。
2. クライアントとリモートホストとの間の接続のトランスポート層が、対称暗号方式を使用して暗号化されます。
3. クライアントが、サーバーに対して自己認証します。
4. クライアントは、暗号化された接続でリモートホストと対話します。

12.1.4.1. トランスポート層

トランスポート層の主なロールは、認証時とその後の通信中に、2 つのホスト間の通信を簡単に安全でセキュアなものにすることです。トランスポート層は、データの暗号化と復号を処理し、データパケットの送受信時にその整合性を保護することでそのロールを果たします。また、トランスポート層は、情報を圧縮して転送を高速にします。

SSH クライアントがサーバーに接続すると鍵情報が交換されるため、両システムでトランスポート層が適正に構築できます。以下は、こうした鍵情報の交換中に発生する手順です。

- 鍵を交換する
- 公開鍵暗号化アルゴリズムが決定する
- 対称暗号化アルゴリズムが決定する
- メッセージ認証アルゴリズムが決定する
- ハッシュアルゴリズムが決定する

鍵交換の間、サーバーは一意の **ホスト鍵** を用いて、クライアントに対して自己識別を行います。クライアントがこの特定のサーバーと過去に通信したことがなければ、クライアントはサーバーのホスト鍵を知らないため、接続が成立しません。OpenSSH は、この問題に対処するためにサーバーのホスト鍵を承認します。これは、ユーザーが通知を受けて新規のホスト鍵を受け取り、検証した後に行われます。それ以降の接続では、サーバーのホスト鍵が、クライアント上に保存されているバージョンと照合され、クライアントが実際に目的のサーバーと通信していることを確信できます。この後、ホスト鍵が一致しなくなった場合は、接続前にクライアントに保存してあるバージョンをユーザーが削除する必要があります。

警告

ローカルシステムは、対象サーバーと攻撃者が設定した偽サーバーとの違いを認識しないため、攻撃者は初回コンタクト中に SSH サーバーをマスカレードすることが可能です。この問題を防ぐために、初回接続の前かホスト鍵の不一致が発生した場合には、サーバー管理者へ連絡して新しい SSH サーバーの整合性を確認してください。

SSH は、ほとんどすべての公開鍵アルゴリズムまたはエンコード形式に対応するように設計されています。初回の鍵交換で、交換に使用されるハッシュ値と共有秘密値が作成されると、2 つのシステムは新しい鍵とアルゴリズムの計算を直ちに開始して、認証と、今後の接続で送信されるデータを保護します。

所定の鍵とアルゴリズムを使用して一定量のデータ (正確な量は SSH 実装により異なる) が送信された後に、もう 1 回鍵交換が行われてハッシュ値と新しい共有秘密値の別のセットが生成されます。攻撃者がハッシュ値と共有秘密値を判別できたとしても、その情報が役に立つのは限られた時間のみです。

12.1.4.2. 認証

トランスポート層が、2 つのシステム間で情報を渡すためのセキュアなトンネルを構築すると、サーバーは、秘密鍵でエンコードされた署名の使用やパスワードの入力など、サポートされている別の認証方法をクライアントに伝えます。次に、クライアントが、対応しているいずれかの方法を使用して、

サーバーに対して自己認証を試みます。

SSH サーバーとクライアントは、異なるタイプの認証を採用するように設定できるため、双方の制御が最適化されます。サーバーは、そのセキュリティーモデルに基づいて、対応する暗号化方法を決定できます。クライアントは、利用可能なオプションの中から、試行する認証方法の順番を選択できます。

12.1.4.3. チャンネル

SSH トランスポート層での認証に成功すると、**多重化** と呼ばれる手法により複数のチャンネルが開きます。[1] これらの各チャンネルは、異なるターミナルセッションと、転送された X11 セッションの通信を処理します。

クライアントとサーバーの両方で、新しいチャンネルを作成できます。その後、各接続の両端に、別々の番号が割り当てられます。クライアントが新しいチャンネルを開こうとする際、要求と共にチャンネル番号を送信します。この情報はサーバーにより保存され、そのチャンネルに通信を移動するのに使用されます。これは、異なるタイプのセッションが相互に影響しないように、あるセッションの終了時にそのチャンネルが SSH による一次接続を停止せずに閉じることができるようにするためです。

また、チャンネルは **フロー制御** にも対応しているため、規則的な方法でデータを送受信できます。この方法では、チャンネルが開いているというメッセージをクライアントが受信するまで、チャンネルでデータが送信されません。

クライアントが要求するサービスのタイプと、ユーザーがネットワークに接続される方法に応じて、クライアントとサーバーは、各チャンネルの特性を自動的にネゴシエートします。これにより、プロトコルの基本インフラストラクチャーを変更しなくても、異なるタイプのリモート接続を非常に柔軟に処理できます。

12.2. OPENSSH の設定

12.2.1. 設定ファイル

設定ファイルには、クライアントプログラム用 (**ssh**、**scp** および **sftp**) とサーバー用 (**sshd** デーモン) の異なる 2 つのセットがあります。

システム全体の SSH 設定情報は、表12.1「システム全体の設定ファイル」 にあるように、**/etc/ssh/** ディレクトリー内に格納されています。ユーザー固有の SSH 設定情報は、ユーザーのホームディレクトリー内の **~/.ssh/** に格納されています。詳細は、表12.2「ユーザー固有の設定ファイル」 に記載しています。

表12.1 システム全体の設定ファイル

ファイル	詳細
/etc/ssh/moduli	セキュアなトランスポート層を構築するのに非常に重要となる、Diffie-Hellman 鍵交換に使用される Diffie-Hellman グループが置かれています。SSH セッションの初めに鍵が交換される時、共有秘密値が作成されますが、どちらか一方の当事者だけでは決定できません。この値は、ホスト認証を行うのに使用されます。

ファイル	詳細
/etc/ssh/ssh_config	デフォルトの SSH クライアント設定ファイルです。**~/.ssh/config** が存在する場合は、これにより上書きされる点に注意して下さい。
/etc/ssh/sshd_config	**sshd** デーモンの設定ファイルです。
/etc/ssh/ssh_host_ecdsa_key	**sshd** デーモンが使用する ECDSA 秘密鍵です。
/etc/ssh/ssh_host_ecdsa_key.pub	**sshd** デーモンが使用する ECDSA 公開鍵です。
/etc/ssh/ssh_host_rsa_key	**sshd** デーモンが使用する SSH プロトコルのバージョン 2 用の RSA 秘密鍵です。
/etc/ssh/ssh_host_rsa_key.pub	**sshd** デーモンが使用する SSH プロトコルのバージョン 2 用の RSA 公開鍵です。
/etc/pam.d/sshd	**sshd** デーモンの PAM 設定ファイルです。
/etc/sysconfig/sshd	**sshd** サービスの設定ファイルです。

表12.2 ユーザー固有の設定ファイル

ファイル	詳細
~/.ssh/authorized_keys	サーバー用の認証済み公開鍵のリストがあります。クライアントがサーバーに接続すると、サーバーが、このファイル内に格納されている署名済み公開鍵を確認してクライアントを認証します。
~/.ssh/id_ecdsa	ユーザーの ECDSA 秘密鍵を格納します。
~/.ssh/id_ecdsa.pub	ユーザーの ECDSA 公開鍵です。
~/.ssh/id_rsa	**ssh** が使用する SSH プロトコルのバージョン 2 用の RSA 秘密鍵です。
~/.ssh/id_rsa.pub	**ssh** が使用する SSH プロトコルのバージョン 2 用の RSA 公開鍵です。
~/.ssh/known_hosts	ユーザーがアクセスする SSH サーバーのホスト鍵を格納します。このファイルは、SSH クライアントが正しい SSH サーバーに接続していることを確認するのに使用するため、非常に重要です。

警告

SSH サーバーを設定する場合は、**/etc/ssh/sshd_config** ファイルの UsePrivilegeSeparation no ディレクティブで **Privilege Separation** 機能をオフにしないでください。**Privilege Separation** をオフにすると、多くのセキュリティー機能が無効になるため、サーバーが潜在的な脆弱性にさらされ、攻撃対象となります。UsePrivilegeSeparation の詳細は、**sshd_config**(5) man ページ、または Red Hat ナレッジベースの記事 What is the significance of UsePrivilegeSeparation directive in /etc/ssh/sshd_config file and how to test it? を参照してください。Red Hat ナレッジベースの記事。

SSH 設定ファイルに使用可能な各種ディレクティブの情報は、**ssh_config**(5) および **sshd_config**(5) の man ページを参照してください。

12.2.2. OpenSSH サーバーの起動

OpenSSH サーバーを実行するには、**openssh-server** パッケージがインストールされている必要があります。新規パッケージのインストール方法は 「パッケージのインストール」 を参照してください。

現行のセッションで **sshd** デーモンを起動するには、**root** 権限でシェルプロンプトに以下を入力します。

```
~]# systemctl start sshd.service
```

現行のセッションで **sshd** デーモンを停止するには、**root** として以下のコマンドを使用します。

```
~]# systemctl stop sshd.service
```

ブート時にデーモンが自動的に起動するようにするには、**root** で以下を入力します。

```
~]# systemctl enable sshd.service
Created symlink from /etc/systemd/system/multi-user.target.wants/sshd.service to
/usr/lib/systemd/system/sshd.service.
```

sshd デーモンは **network.target** ターゲットユニットに依存しますが、静的設定のネットワークインターフェイスやデフォルトの **ListenAddress 0.0.0.0** オプションの場合はこれで十分です。**ListenAddress** ディレクティブで別のアドレスを指定し、より遅い動的ネットワーク設定を使用するには、**network-online.target** ターゲットユニットの依存関係を **sshd.service** ユニットファイルに追加します。これを行うには、**/etc/systemd/system/sshd.service.d/local.conf** ファイルを以下のオプションで作成します。

```
[Unit]
Wants=network-online.target
After=network-online.target
```

この後、次のコマンドを実行して、**systemd** マネージャー設定を再ロードします。

```
~]# systemctl daemon-reload
```

Red Hat Enterprise Linux でシステムサービスを管理する方法は、10章systemd によるサービス管理 を参照してください。

システムを再インストールすると、新しい識別鍵のセットが作成される点に注意してください。したがって、再インストールの前にいずれかの OpenSSH ツールを使用してシステムに接続したことがあるクライアントには、以下のようなメッセージが表示されます。

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@: REMOTE HOST IDENTIFICATION HAS CHANGED!   @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
```

これを防ぐには、**/etc/ssh/** ディレクトリーから関連ファイルをバックアップしておきます。ファイルのリストは 表12.1「システム全体の設定ファイル」 を参照してください。これで、システムの再インストール時にファイルを復元できます。

12.2.3. リモート接続に必要な SSH

SSH を本当の意味で有効なものにするためには、セキュリティー保護されていない接続プロトコルは使用しないことを推奨します。このような接続プロトコルを使用すると、ユーザーのパスワード自体は SSH を使用した 1 回のセッションで保護されても、その後に Telnet を使用してログインした時に傍受されてしまうためです。無効にするサービスには、**telnet**、**rsh**、**rlogin**、**vsftpd** などがあります。

vsftpd サービスの設定方法は、「FTP」 を参照してください。Red Hat Enterprise Linux 7 でシステムサービスを管理する方法については、10章systemd によるサービス管理 をお読みください。

12.2.4. 鍵ベース認証の使用

システムのセキュリティーをさらに強化するには、SSH 鍵のペアを生成し、パスワード認証を無効にすることで鍵ベース認証を強制します。これを行うには、**vi**、**nano** などのテキストエディターで **/etc/ssh/sshd_config** 設定ファイル開き、**PasswordAuthentication** オプションを以下のように変更します。

```
PasswordAuthentication no
```

新規のデフォルトインストール以外のシステムで作業をしている場合は、**PubkeyAuthentication no** が設定されて **いない** ことを確認してください。リモートで接続している場合は、コンソールもしくは帯域外アクセスを使用せず、パスワード認証を無効にする前にプロセス内で鍵ベースのログをテストすることが推奨されます。

ssh、**scp**、または **sftp** を使用してクライアントマシンからサーバーに接続できるようにするには、以下の手順に従って認証鍵ペアを生成します。鍵はユーザーごとに別々に生成する必要がある点に注意してください。

NFS がマウントされたホームディレクトリーで鍵ベースの認証を使用するには、最初に SELinux ブール値 **use_nfs_home_dirs** を有効にします。

```
~]# setsebool -P use_nfs_home_dirs 1
```

Red Hat Enterprise Linux 7 は、デフォルトで SSH プロトコル 2 および RSA 鍵を使用します (詳細は 「プロトコルのバージョン」 を参照してください)。

重要

これらのステップを **root** で完了すると、鍵を使用できるのは **root** のみになります。

注記

システムを再インストールする際に、生成しておいた鍵ペアを引き続き使用する場合は、~/**.ssh**/ ディレクトリーのバックアップを作成します。再インストール後に、このディレクトリーをホームディレクトリーにコピーします。この手順は、システムの全ユーザー (**root** を含む) が実行できます。

12.2.4.1. 鍵ペアの生成

以下の手順に従って、SSH プロトコルのバージョン 2 用の RSA 鍵ペアを生成します。

1. RSA 鍵ペアを生成するには、シェルプロンプトで次のコマンドを実行します。

```
~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/USER/.ssh/id_rsa):
```

2. **Enter** キーを押して、新規作成される鍵のデフォルトの場所 (~/**.ssh**/**id_rsa**) を確認します。

3. パスフレーズを入力します。プロンプトが表示されたら再入力して確認します。セキュリティー上の理由により、アカウントのログイン時に使用するパスワードは使用しないでください。
この後、以下のようなメッセージが表示されます。

```
Your identification has been saved in /home/USER/.ssh/id_rsa.
Your public key has been saved in /home/USER/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:UNIgIT4wfhdQH/K7yqmjsbZnnyGDKiDviv492U5z78Y
USER@penguin.example.com
The key's randomart image is:
+---[RSA 2048]----+
|o ..==o+.    |
|.+ . .=oo    |
| .o. ..o     |
| ... ..    |
|    .S    |
|o .   .    |
|o+ o .o+ ..   |
|+.++=o*.o .E   |
|BBBo+Bo. oo   |
+----[SHA256]-----+
```

注記

以前のバージョンでデフォルトのフィンガープリントである MD5 鍵フィンガープリントを取得する場合は、**ssh-keygen** コマンドで **-E md5** オプションを使用します。

4. デフォルトで、**~/.ssh/** ディレクトリーのパーミッションは、**rwx------** または 8 進数表記の **700** に設定されます。これは、USER のみがコンテンツを表示できるようにする設定です。必要に応じて、次のコマンドで確認できます。

```
~]$ ls -ld ~/.ssh
drwx------. 2 USER USER 54 Nov 25 16:56 /home/USER/.ssh/
```

5. 公開鍵をリモートマシンにコピーするには、次の形式でコマンドを実行します。

```
ssh-copy-id user@hostname
```

これにより、最近変更した **~/.ssh/id*.pub** 公開鍵がインストールされていない場合は、その公開鍵をコピーします。または、以下のように、公開鍵のファイルを指定します。

```
ssh-copy-id -i ~/.ssh/id_rsa.pub user@hostname
```

これにより、**~/.ssh/id_rsa.pub** の内容が、接続するマシンの **~/.ssh/authorized_keys** ファイルにコピーされます。ファイルが存在する場合は、鍵がその最後に追加されます。

SSH プロトコルのバージョン 2 用の ECDSA 鍵ペアを生成するには、以下の手順に従います。

1. ECDSA 鍵ペアを生成するには、シェルプロンプトで次のコマンドを実行します。

```
~]$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/USER/.ssh/id_ecdsa):
```

2. **Enter** キーを押して、新規作成された鍵用のデフォルトの場所 (**~/.ssh/id_ecdsa**) を確認します。

3. パスフレーズを入力します。プロンプトが表示されたら再入力して確認します。セキュリティー上の理由により、アカウントのログイン時に使用するパスワードは使用しないでください。
この後、以下のようなメッセージが表示されます。

```
Your identification has been saved in /home/USER/.ssh/id_ecdsa.
Your public key has been saved in /home/USER/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:8BhZageKrLXM99z5f/AM9aPo/KAUd8ZZFPcPFWqK6+M
USER@penguin.example.com
The key's randomart image is:
+---[ECDSA 256]---+
|   . .   +=|
| . . . =   o.o|
| + . * .  o...|
| = . . * . + +..|
|. + . . So o * ..|
|  . o . .+ = ..|
|   o oo ..=. .|
|    ooo...+ |
|    .E++oo  |
+----[SHA256]-----+
```

4. デフォルトで、**~/.ssh/** ディレクトリーのパーミッションは、**rwx------** または 8 進数表記の **700** に設定されます。これは、USER のみがコンテンツを表示できるようにする設定です。必要に応じて、次のコマンドで確認できます。

```
~]$ ls -ld ~/.ssh
      ~]$ ls -ld ~/.ssh/
drwx------. 2 USER USER 54 Nov 25 16:56 /home/USER/.ssh/
```

5. 公開鍵をリモートマシンにコピーするには、次の形式でコマンドを実行します。

```
ssh-copy-id USER@hostname
```

これにより、最近変更した **~/.ssh/id*.pub** 公開鍵がインストールされていない場合は、その公開鍵をコピーします。または、以下のように、公開鍵のファイルを指定します。

```
ssh-copy-id -i ~/.ssh/id_ecdsa.pub USER@hostname
```

これにより、**~/.ssh/id_ecdsa.pub** の内容が、接続するマシンの **~/.ssh/authorized_keys** にコピーされます。ファイルが存在する場合は、鍵がその最後に追加されます。

システムにパスフレーズを記憶させる設定方法については 「ssh-agent の設定」 を参照してください。

重要

秘密鍵は、個人使用を目的としているため、他人には決して教えないでください。

12.2.4.2. ssh-agent の設定

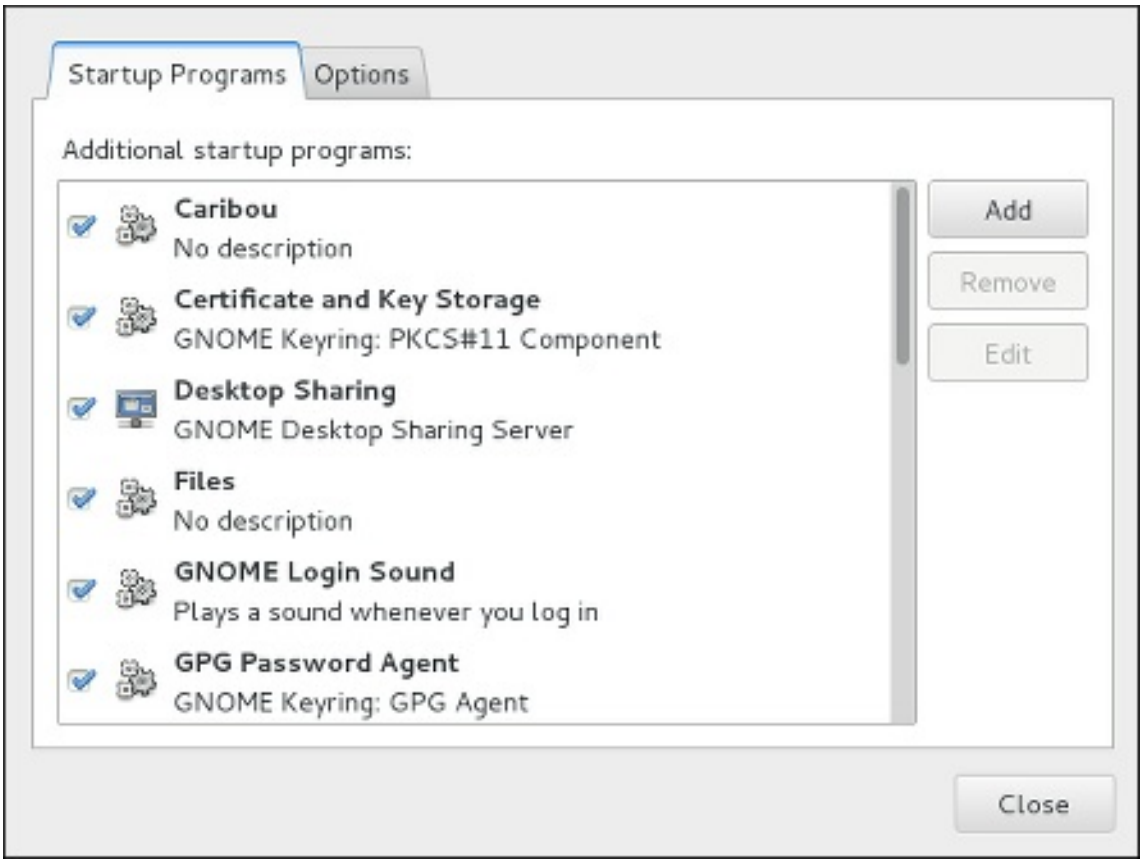
ssh-agent 認証エージェントを使用するとパスフレーズを保存することができるため、リモートマシンとの接続を開始する度にパスフレーズを入力する必要がなくなります。GNOME を実行している場合は、ログイン時には常にパスフレーズを求めるプロンプトを表示して、セッションを通してそのパスフレーズを記憶させておくように設定できます。それ以外の方法として、特定のシェルプロンプト用にパスフレーズを保存しておくことも可能です。

以下のステップに従って、GNOME セッション中にパスフレーズを保存します。

1. **openssh-askpass** パッケージがインストールされていることを確認します。Red Hat Enterprise Linux に新しいパッケージをインストールする方法の詳細は、「パッケージのインストール」 を参照してください。

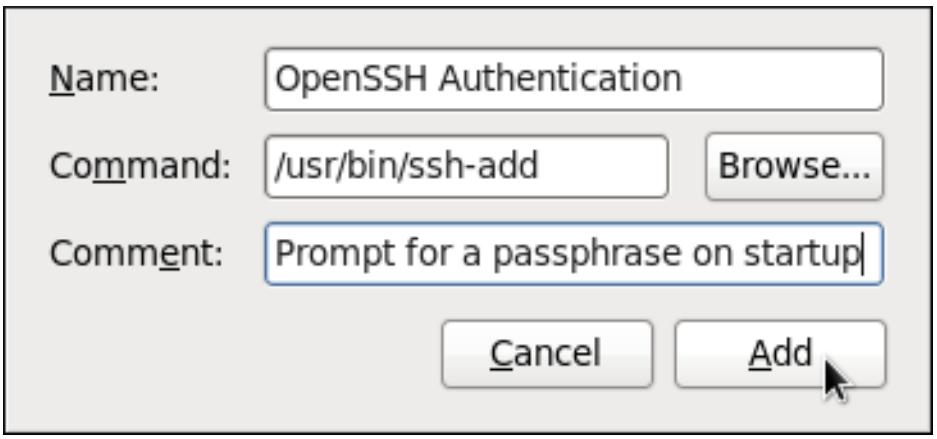
2. **Super** キーを押してアクティビティーの概要に入り、 **Startup Applications** と入力して **Enter** を押します。**Startup Applications Preferences** ツールが表示されます。デフォルトでは、利用可能なスタートアッププログラムのリストを含むタブが表示されます。**Super** キーはキーボードや他のハードウェアによって外見が異なりますが、通常はスペースバーの左側にある Windows キーまたは Command キーになります。

図12.1 自動起動するアプリの設定



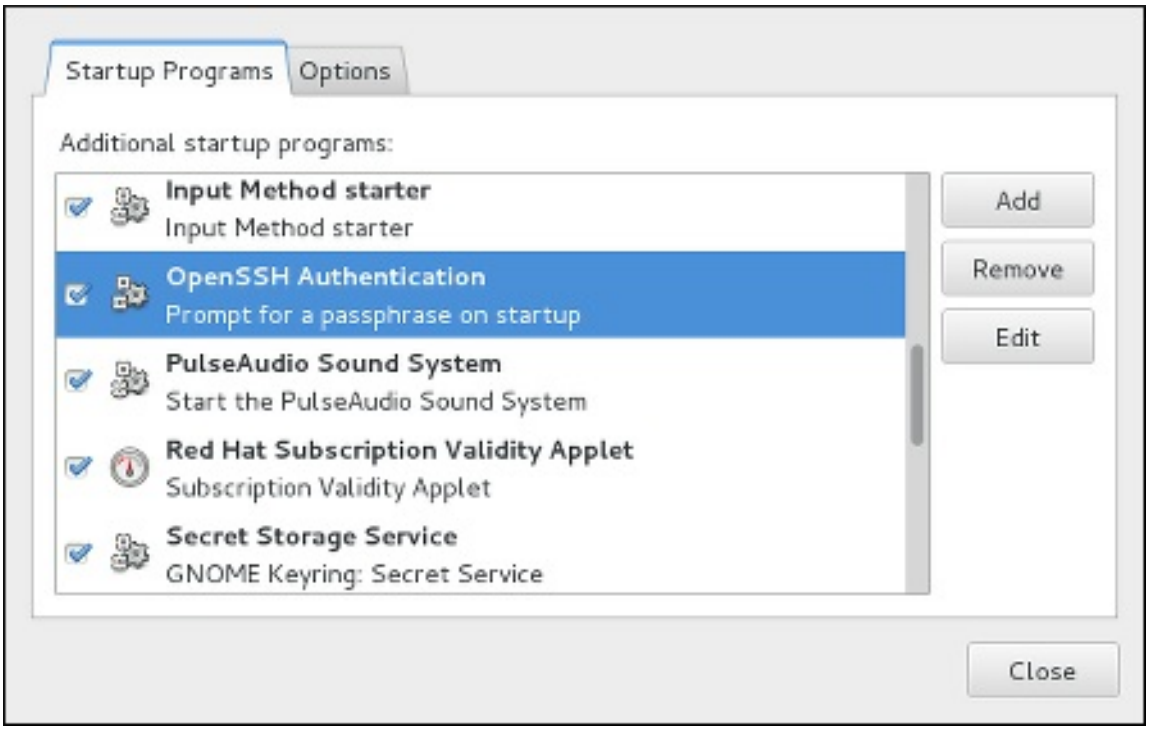
3. *右側の 追加 ボタンをクリックして、コマンド フィールドに **/usr/bin/ssh-add** と入力します。*

   図12.2 新規アプリケーションの追加



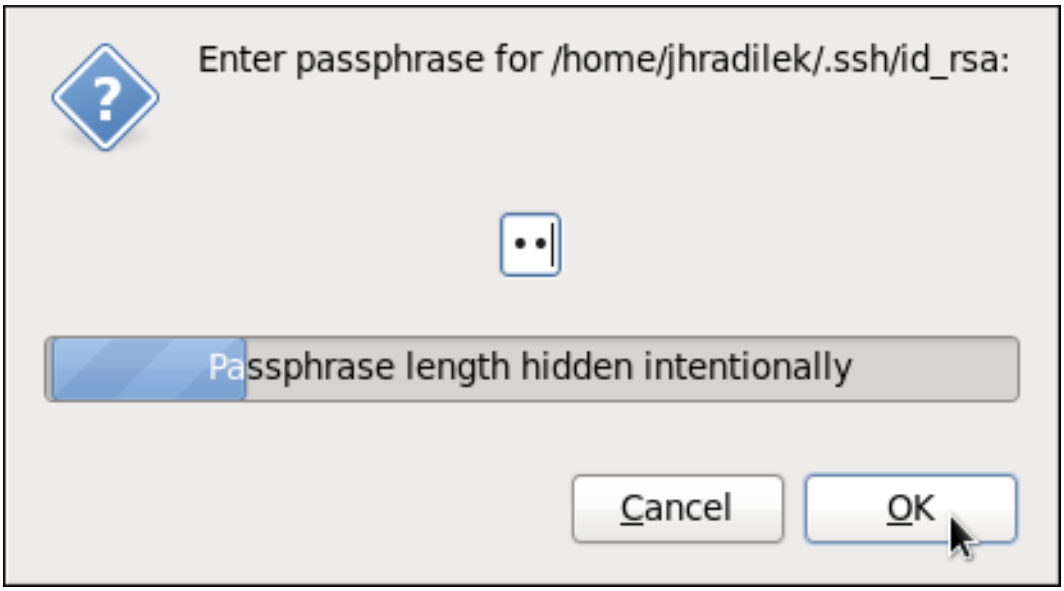
4. *追加 をクリックした後に、新しく追加した項目の横のチェックボックスにチェックマークが付いていることを確認してください。*

図12.3 アプリケーションの有効化



5. *一度ログアウトしてから再度ログインします。パスフレーズの入力を求めるダイアログボックスが表示されます。これ以降は、**ssh**、**scp**、または **sftp** によるパスワードの入力を要求されることはありません。*

図12.4 パスフレーズの入力



特定のシェルプロンプト用のパスフレーズを保存するには、以下のコマンドを使用します。

```
~]$ ssh-add
Enter passphrase for /home/USER/.ssh/id_rsa:
```

ログアウト時には、パスフレーズは記憶されない点に注意してください。仮想コンソールまたはターミナルウィンドウにログインする度にコマンドを実行する必要があります。

12.3. OPENSSH クライアント

クライアントマシンから OpenSSH サーバーに接続するには、**openssh-clients** パッケージがインストールされている必要があります (&MAJOROS; に新規パッケージをインストールする方法については 「パッケージのインストール」 を参照)。

12.3.1. ssh ユーティリティーの使用

ssh ユーティリティーを使用すると、リモートマシンにログインしてそのマシン上でコマンドを実行することできます。これは、**rlogin**、**rsh** および **telnet** プログラムに代わるセキュアな手段です。

telnet コマンドと同様に、以下のコマンドを使用してリモートマシンにログインします。

```
ssh hostname
```

たとえば、**penguin.example.com** という名前のリモートマシンにログインするには、シェルプロンプトで以下を入力します。

```
~]$ ssh penguin.example.com
```

これで、ローカルマシンで使用しているユーザー名でログインします。別のユーザー名を指定する場合には、以下の形式のコマンドを使用してください。

```
ssh username@hostname
```

たとえば、**USER** として **penguin.example.com** にログインするには、以下のように入力します。

```
~]$ ssh USER@penguin.example.com
```

初回接続時には、以下のようなメッセージが表示されます。

```
The authenticity of host 'penguin.example.com' can't be established.
ECDSA key fingerprint is SHA256:vuGKK9dsW34zrZzwjl5g+vOE6EZQvHRQ8zObKYO2mW4.
ECDSA key fingerprint is MD5:7e:15:c3:03:4d:e1:dd:ee:99:dc:3e:f4:b9:67:6b:62.
Are you sure you want to continue connecting (yes/no)?
```

このダイアログの質問に答える前に、常にフィンガープリントが正しいか確認してください。ユーザーは、鍵が正しいかサーバー管理者に尋ねることができます。これは、安全で事前に合意した方法で行う必要があります。サーバーのホスト鍵にユーザーがアクセスできる場合、フィンガープリントは以下のように **ssh-keygen** コマンドを使用することで確認できます。

```
~]# ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key.pub
SHA256:vuGKK9dsW34zrZzwjl5g+vOE6EZQvHRQ8zObKYO2mW4
```

注記

以前のバージョンでデフォルトのフィンガープリントである MD5 鍵フィンガープリントを取得する場合は、**ssh-keygen** コマンドで **-E md5** オプションを使用します。例:

```
~]# ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key.pub -EM md5
MD5:7e:15:c3:03:4d:e1:dd:ee:99:dc:3e:f4:b9:67:6b:62
```

yes と入力して鍵を受け入れ、接続を確定します。サーバーが既知ホストのリストに追加されたことを知らせるメッセージと、パスワードの入力を求めるプロンプトが以下のように表示されます。

```
Warning: Permanently added 'penguin.example.com' (ECDSA) to the list of known hosts.
USER@penguin.example.com's password:
```

重要

SSH サーバーのホスト鍵が変更された場合、クライアントはサーバーのホスト鍵が **~/.ssh/known_hosts** ファイルから削除されるまで接続を開始できないことをユーザーに知らせます。ただし、これを実行する前に、SSH サーバーのシステム管理者に連絡して、サーバーが被害を受けていないことを確認してください。

~/.ssh/known_hosts ファイルから鍵を削除するには、以下のようにコマンドを発行します。

```
~]# ssh-keygen -R penguin.example.com
# Host penguin.example.com found: line 15 type ECDSA
/home/USER/.ssh/known_hosts updated.
Original contents retained as /home/USER/.ssh/known_hosts.old
```

パスワードを入力すると、リモートマシン用のシェルプロンプトが表示されます。

別の方法として、シェルプロンプトにログインせずに、**ssh** プログラムを使用してリモートマシン上でコマンドを実行することができます。

```
ssh username@hostname command
```

たとえば、**/etc/redhat-release** ファイルは、Red Hat Enterprise Linux のバージョンに関する情報を提供します。**penguin.example.com** でこのファイルの内容を表示するには、以下を入力します。

```
~]$ ssh USER@penguin.example.com cat /etc/redhat-release
USER@penguin.example.com's password:
Red Hat Enterprise Linux Server release 7.0 (Maipo)
```

正しいパスワードを入力すると、ユーザー名が表示され、ローカルのシェルプロンプトに戻ります。

12.3.2. scp ユーティリティーの使用

scp を使用すると、暗号化されたセキュアな接続でマシン間のファイル転送を行うことができます。設計に関しては、**rcp** と非常に似ています。

ローカルファイルをリモートシステムへ転送するには、以下の形式でコマンドを使用します。

```
scp localfile username@hostname:remotefile
```

たとえば、**taglist.vim** を **penguin.example.com** という名前のリモートマシンに転送したい場合は、シェルプロンプトで以下のように入力します。

```
~]$ scp taglist.vim USER@penguin.example.com:.vim/plugin/taglist.vim
USER@penguin.example.com's password:
taglist.vim                   100% 144KB 144.5KB/s   00:00
```

一度に複数のファイルを指定することも可能です。**.vim/plugin/** の内容を **penguin.example.com** のリモートマシン上の同じディレクトリーに転送するには、以下のコマンドを入力します。

```
~]$ scp .vim/plugin/* USER@penguin.example.com:.vim/plugin/
USER@penguin.example.com's password:
closetag.vim                  100%  13KB 12.6KB/s  00:00
snippetsEmu.vim               100%  33KB 33.1KB/s  00:00
taglist.vim                   100% 144KB 144.5KB/s  00:00
```

リモートファイルをローカルシステムへ転送するには、以下の構文を使用します。

```
scp username@hostname:remotefile localfile
```

たとえば **.vimrc** 設定ファイルをリモートマシンからダウンロードするには、以下のように入力します。

```
~]$ scp USER@penguin.example.com:.vimrc .vimrc
USER@penguin.example.com's password:
.vimrc                        100% 2233   2.2KB/s  00:00
```

12.3.3. sftp ユーティリティーの使用

sftp ユーティリティーを使用すると、セキュアでインタラクティブなFTP セッションを開始することができます。その設計では、**ftp** と似ていますが、暗号化された接続を使用します。

リモートシステムに接続するには、以下の形式でコマンドを使用します。

```
sftp username@hostname
```

たとえば **penguin.example.com** という名前のリモートマシンに **USER** というユーザー名でログインするには、以下のように入力します。

```
~]$ sftp USER@penguin.example.com
USER@penguin.example.com's password:
Connected to penguin.example.com.
sftp>
```

正しいパスワードを入力すると、プロンプトが表示されます。**sftp** ユーティリティーは、**ftp** で使用されるコマンドセットと同様のものを使用します (表12.3「利用可能な sftp コマンドの抜粋」 を参照)。

表12.3 利用可能な **sftp** コマンドの抜粋

コマンド	詳細
ls [directory]	リモート directory の内容をリスト表示します。指定がない場合は、デフォルトで現在の作業ディレクトリーが使用されます。
cd directory	リモートの作業ディレクトリーを directory に変更します。
mkdir ディレクトリー	リモートの directory を作成します。
rmdir path	リモートの directory を削除します。

コマンド	詳細
put localfile [remotefile]	localfile をリモートマシンに転送します。
get remotefile [localfile]	remotefile をリモートマシンから転送します。

利用可能なコマンドの詳細リストは、 ***sftp****(1) の man ページを参照してください。*

12.4. セキュアシェルの追加

セキュアなコマンドラインインターフェイスは、数多くある SSH の用途の中でも初歩的なものに過ぎません。十分な帯域幅があれば、X11 セッションは SSH チャンネル上で送信できます。あるいは、TCP/IP 転送を使用することで、以前はセキュリティー保護されていなかったシステム間のポート接続を特定の SSH チャンネルにマッピングすることができます。

12.4.1. X11 転送

SSH 接続上で X11 セッションを開始するには、以下の形式でコマンドを使用します。

```
ssh -Y username@hostname
```

たとえば ***penguin.example.com*** *という名前のリモートマシンに* ***USER*** *というユーザー名でログインするには、以下のように入力します。*

```
~]$ ssh -Y USER@penguin.example.com
USER@penguin.example.com's password:
```

セキュアシェルプロンプトから X プログラムを実行すると、SSH クライアントとサーバーは新しいセキュアなチャンネルを作成し、X プログラムデータはそのチャンネル上で透過的にクライアントマシンに送信されます。

X11 転送の前に X Window System がインストールされている必要があることに注意してください。 ***root*** *で以下のコマンドを入力し、X11 パッケージグループをインストールします。*

```
~]# yum group install "X Window System"
```

パッケージグループの詳細は「パッケージグループでの作業」 を参照してください。

X11 転送は非常に便利なものです。たとえば、X11 転送を使用すると、 ***Print Settings*** *ユーティリティーのセキュアかつインタラクティブなセッションを作成できます。これを行うには、* ***ssh*** *を使用してサーバーに接続し、以下のコマンドを入力します。*

```
~]$ system-config-printer &
```

印刷設定 ツールが表示され、リモートユーザーがリモートシステムで安全に印刷を設定できます。

12.4.2. ポート転送

*SSH は、ポート転送によりセキュリティー保護されていない **TCP/IP** プロトコルをセキュアにすることができます。この手法を使用する場合、SSH サーバーは SSH クライアントをつなぐ暗号化された経路となります。*

ポート転送は、クライアント上のローカルポートをサーバー上のリモートポートにマッピングすることで機能します。SSH ではサーバーの任意のポートをクライアント上の任意のポートにマッピングすることが可能です。このテクニックが機能するためにポート番号が一致する必要はありません。

注記

*1024 未満のポートで待機するようにポート転送を設定するには、**root** レベルのアクセスが必要です。*

***localhost** 上で接続を待機する TCP/IP ポート転送チャンネルを作成するには、以下の形式でコマンドを使用します。*

```
ssh -L local-port:remote-hostname:remote-port username@hostname
```

*たとえば、暗号化された接続で **POP3** を使用して、**mail.example.com** と呼ばれるサーバーでメールを確認するには、以下のコマンドを使用します。*

```
~]$ ssh -L 1100:mail.example.com:110 mail.example.com
```

*ポート転送チャンネルがクライアントマシンとメールサーバー間に配置されたら、POP3 メールクライアントに対し **localhost** 上のポート **1100** を使用して、新規のメールを確認するように指示します。クライアントシステム上のポート **1100** に送信される要求は、安全に **mail.example.com** サーバーに向けられます。*

*SSH サーバーを実行しているのが **mail.example.com** ではなく、同一のネットワーク上にある別のマシンの場合でも、SSH を使用して接続の一部をセキュアにすることができます。ただし、若干異なるコマンドが必要になります。*

```
~]$ ssh -L 1100:mail.example.com:110 other.example.com
```

*この例では、クライアントマシン上のポート **1100** からの POP3 要求がポート **22** の SSH 接続を介して SSH サーバー **other.example.com** に転送されます。次に、**other.example.com** は **mail.example.com** 上のポート **110** に接続して、新規のメールを確認します。この手法を使用する場合、クライアントシステムと **other.example.com** SSH サーバー間の接続のみがセキュアである点に注意してください。*

*OpenSSH スイートは、UNIX ドメインソケットのローカルおよびリモートのポート転送も提供します。UNIX ドメインソケットをネットワークを介して別のマシンへ転送するには、**ssh -L local-socket:remote-socket username@hostname** コマンドを使用します。以下に例を示します。*

```
~]$ ssh -L /var/mysql/mysql.sock:/var/mysql/mysql.sock username@hostname
```

ポート転送は、ネットワークのファイアウォール経由でセキュアに情報を取得する場合にも使用できます。ファイアウォールが標準ポート (ポート 22) 経由の SSH トラフィックを許可するよう設定されているものの、他のポートへのアクセスはブロックする場合でも、確立された SSH 接続にそのような通信をリダイレクトすることにより、ブロックされたポートを使用した 2 つのホスト間の接続は可能になります。

重要

この方法でポート転送を使用して接続を転送すると、クライアントシステム上のどのユーザーもそのサービスに接続できます。クライアントシステムが侵害された場合、攻撃者は転送されたサービスにアクセスすることもできます。

*ポート転送に関するシステム管理者は、**/etc/ssh/sshd_config** の **AllowTcpForwarding** 行に **No** パラメーターを指定して **sshd** サービスを再起動することで、この機能をサーバー上で無効にできます。*

12.5. 関連情報

Red Hat Enterprise Linux で OpenSSH サーバーを設定し、接続する方法は、以下の資料を参照してください。

インストールされているドキュメント

- ***sshd***(8): **sshd** デーモンの man ページでは、利用可能なコマンドラインオプションと、対応している設定ファイルおよびディレクトリーがすべて説明されています。
- ***ssh***(1): **ssh** クライアントアプリケーションの man ページでは、利用可能なコマンドラインオプションと、対応している設定ファイルおよびディレクトリーがすべて説明されています。
- ***scp***(1): **scp** ユーティリティーの man ページでは、このユーティリティーの詳細な説明とその使用方法が説明されています。
- ***sftp***(1) - **sftp** ユーティリティーの man ページ
- ***ssh-keygen***(1) - **ssh-keygen** ユーティリティーの man ページは、このユーティリティーを使用して **ssh** が使用する認証鍵を生成、管理、変換する詳細な方法が説明されています。
- ***ssh_config***(5) - **ssh_config** の man ページでは、利用可能な SSH クライアント設定オプションが説明されています。
- ***sshd_config***(5) - **sshd_config** の man ページでは、利用可能な SSH デーモン設定オプションが詳細に説明されています。

オンラインドキュメント

- OpenSSH Home Page - その他のドキュメントや FAQ、メーリングリストへのリンク、バグレポートなどの役立つリソースを掲載した OpenSSH のホームページです。
- OpenSSL Home Page - その他のドキュメントや FAQ、メーリングリストへのリンクなどの役立つリソースを掲載した OpenSSL のホームページです。

関連項目

- *6章権限の取得* では、**su** および **sudo** コマンドを使用して管理者権限を取得する方法を説明しています。
- *10章systemd によるサービス管理* では、systemd の詳細情報と、**systemctl** コマンドを使用してシステムサービスを管理する方法が説明されています。

---

[1] 多重接続は、共有されている共通のメディアで送信されるいくつかのシグナルで設定されます。SSH により、異なるチャンネルが共通のセキュアな接続で送信されます。

第13章 TIGERVNC

TigerVNC (Tiger Virtual Network Computing) は、グラフィカルデスクトップシェアリングのシステムで、他のコンピューターのリモート制御を可能にします。

TigerVNC は、クライアントサーバープリンシパルで機能します。**サーバー**はその出力 (**vncserver**) を共有し、**クライアント** (**vncviewer**) はサーバーに接続します。

注記

以前の Red Hat Enterprise Linux ディストリビューションとは異なり、Red Hat Enterprise Linux 7 の **TigerVNC** は、その設定に **systemd** システム管理デーモンを使用します。**/etc/sysconfig/vncserver** 設定ファイルに代わり、**/etc/systemd/system/vncserver@.service** が使用されるようになりました。

13.1. VNC SERVER

vncserver は、VNC (Virtual Network Computing) デスクトップを起動するユーティリティーです。適切なオプションで **Xvnc** を実行し、VNC デスクトップでウィンドウマネージャーを起動します。**vncserver** ユーザーは、どこからでも任意の数のクライアントがアクセスできるマシン上で、別々のセッションを並行して実行できます。

13.1.1. VNC サーバーのインストール

TigerVNC サーバーをインストールするには、**root** 権限で以下のコマンドを実行します。

```
~]# yum install tigervnc-server
```

13.1.2. VNC サーバーの設定

VNC サーバーでは、表示、ネットワークアドレス、ポート、セキュリティーの設定などに対するオプションのパラメーターを使用して、1 人または複数のユーザー用の画面を起動するよう設定できます (ユーザーのシステムがシステムに存在する場合)。

1 人のユーザー用に VNC サーバーの設定

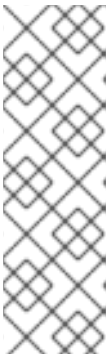
1. **/etc/systemd/system/vncserver@.service** という名前の設定ファイルが必要になります。このファイルを作成するには、**/usr/lib/systemd/system/vncserver@.service** ファイルを **root** でコピーします。

   ```
   ~]# cp /usr/lib/systemd/system/vncserver@.service
   /etc/systemd/system/vncserver@.service
   ```

   systemd は、オンデマンドでメモリー内に適切な名前が付けられたインスタンスを自動的に作成し、サービスファイル内の **'%i'** はディスプレイ番号に置き換えられるため、ファイル名に表示番号を含める必要はありません。単一ユーザーの場合は、ファイルの名前を変更する必要がありません。複数ユーザーの場合は、たとえば、ユーザー名をファイル名に追加して、各ユーザーに対して一意な名前が付けられたサービスファイルが必要です。詳しくは 「2 人のユーザー用に VNC サーバーの設定」 を参照してください。

2. **/etc/systemd/system/vncserver@.service** を修正して、USER を実際のユーザー名で置き換えます。その他の行は、そのままにしておきます。

```
ExecStart=/usr/bin/vncserver_wrapper <USER> %i
```



注記

VNC デスクトップのデフォルトサイズは 1024x768 です。

ユーザーの VNC セッションは、**~/.vnc/config** ファイルを使用してさらに設定できます。

たとえば、VNC ウィンドウサイズを変更するには、次の行を追加します。

geometry= <WIDTH> x <HEIGHT>

3. 変更を保存します。

4. 変更を直ちに反映させるには、以下のコマンドを実行します。

```
~]# systemctl daemon-reload
```

5. 設定ファイルで定義されたユーザー用のパスワードを設定します。最初に **root** から USER に切り替える必要があることに注意してください。

```
~]# su - USER
~]$ vncpasswd
Password:
Verify:
```

重要

保存されたパスワードは暗号化されていません。パスワードファイルへのアクセスが可能であれば、誰でもプレーンテキストのパスワードを見ることができます。

「VNC サーバーの起動」に進みます。

13.1.2.1. 2 人のユーザー用に VNC サーバーの設定

同一マシンで複数のユーザーを設定する場合は、ユーザーごとに異なるテンプレートタイプのサービスファイルを作成します。

1. たとえば、**vncserver-USER_1@.service** と **vncserver-USER_2@.service** の 2 つのサービスファイルを作成します。これら 2 つのファイルで、USER を正しいユーザー名に置き換えます。

2. 両方のユーザーでパスワードを設定します。

```
~]$ su - USER_1
~]$ vncpasswd
Password:
Verify:
~]$ su - USER_2
```

```
~]$ vncpasswd
Password:
Verify:
```

13.1.3. VNC サーバーの起動

サービスを起動もしくは有効にするには、コマンドで直接ディスプレイ番号を指定します。1 人のユーザー用に VNC サーバーの設定 の上記で設定したファイルはテンプレートとして機能し、そこでは **%i** が **systemd** によりディスプレイ番号に置き換えます。有効な番号を用いて、以下のコマンドを実行します。

```
~]# systemctl start vncserver@:display_number.service
```

また、システム起動時に自動的にサービスが開始するようにすることもできます。そうすると、ログイン時に **vncserver** が自動的に開始されます。**root** で以下のコマンドを発行します。

```
~]# systemctl enable vncserver@:display_number.service
```

この時点で、他のユーザーは VNC ビューアープログラムを使用して、定義されたディスプレイ番号とパスワードで VNC サーバーに接続できます。グラフィカルデスクトップがインストールされている場合は、そのデスクトップのインスタンスが表示されます。これは、ターゲットマシンで現在表示されているものと同じインスタンスではありません。

13.1.3.1. ユーザー 2 人および 2 つの別個のディスプレイ用に VNC サーバーの設定

vncserver-USER_1@.service および vncserver-USER_2@.service という 2 つの設定済み VNC サーバーで、異なるディスプレイ番号を有効にできます。たとえば、以下のコマンドでは、USER_1 の VNC サーバーがディスプレイ 3 で起動し、USER_2 の VNC サーバーがディスプレイ 5 で起動することになります。

```
~]# systemctl start vncserver-USER_1@:3.service
~]# systemctl start vncserver-USER_2@:5.service
```

13.1.4. GDM 用の XDMCP を使用した xinetd ベースの VNC セットアップ

GDM 用の X Display Manager Control Protocol (XDMCP) を使用した xinetd ベースの VNC セットアップは、主にシンクライアントで設定されているクライアントシステムに有益なセットアップです。セットアップ後、クライアントは GDM ログインウィンドウにアクセスでき、システムアカウントにログインできます。設定の前提条件は、**gdm**、**vnc -server** &、および **xinetd** パッケージがインストールされていることです。

```
~]# yum install gdm tigervnc tigervnc-server xinetd
```

サービス xinetd を有効化する必要があります。

```
~]# systemctl enable xinetd.service
```

システムのデフォルトターゲットユニットは **graphical.target** になっているはずです。現在設定されているデフォルトターゲットユニットを取得するには、以下を使用します。

```
~]# systemctl get-default
```

以下を使用して、デフォルトターゲットユニットを変更できます。

```
~]# systemctl set-default target_name
```

GDM ログインウィンドウへのアクセスとログイン

1. **/etc/gdm/custom.conf** 設定ファイルを編集して、GDM を設定してXDMCP を有効にします。

```
[xdmcp]
Enable=true
```

2. 以下のコンテンツで **/etc/xinetd.d/xvncserver** というファイルを作成します。

```
service service_name
{
disable = no
protocol = tcp
socket_type = stream
wait = no
user = nobody
server = /usr/bin/Xvnc
server_args = -inetd -query localhost -once -geometry selected_geometry -depth
selected_depth securitytypes=none
}
```

server_args セクションでは、**-query localhost** オプションにより、xdmcp セッションに対して各 Xvnc インスタンスのクエリー localhost が作成されます。**-depth** オプションは、作成する VNC デスクトップの–深さ (ビット) を指定します。使用できる値は、8、15、16、および 24 です。その他の値は、アプリケーションの予測できない動作を引き起こします。

3. **/etc/services** ファイルを編集して、サービスを定義します。これを行うには、以下の内容を **/etc/services** ファイルに追加します。

```
# VNC xinetd GDM base
service_name 5950/tcp
```

4. 設定の変更が反映されるようにするには、マシンを再起動します。
あるいは、以下を実行します。init レベルを 3 に変更し、5 に戻して、gdm をリロードします。

```
# init 3
# init 5
```

gdm が UDP ポート 177 をリッスンしていることを確認します。

```
# netstat -anu|grep 177
udp    0   0 0.0.0.0:177         0.0.0.0:*
```

xinetd サービスを再起動します。

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

xinetd サービスが新しいサービスを読み込んだことを確認します。

```
# netstat -anpt|grep 595
tcp    0   0 :::5950            :::*            LISTEN   3119/xinetd
```

5. *vncviewer コマンドを使用して設定をテストします。*

```
# vncviewer localhost:5950
```

コマンドは、パスワードが求められない localhost に対して VNC セッションを起動します。GDM ログイン画面が表示され、有効なユーザー名とパスワードでシステムのユーザーアカウントにログインできます。それから、リモート接続で同じテストを実行できます。

その設定に対してファイアウォールを設定します。ファイアウォール設定ツールを実行し、TCP ポート 5950 を追加してシステムへの接続を許可します。

```
~]# firewall-cmd --permanent --zone=public --add-port=5950/tcp
~]# firewall-cmd --reload
```

13.1.5. VNC セッションの終了

***vncserver** サービスの有効化と同様に、システム開始時に自動的にサービスの起動を無効にできます。*

```
~]# systemctl disable vncserver@:display_number.service
```

*または、システムの実行中に、以下のコマンドを **root** で発行すると、サービスを停止できます。*

```
~]# systemctl stop vncserver@:display_number.service
```

13.2. 既存のデスクトップの起動

*デフォルトでは、ログインしているユーザーはディスプレイ **0** の X サーバーにより提供されたデスクトップを使用します。ユーザーは **TigerVNC** サーバー **x0vncserver** を使用してデスクトップを共有できます。*

X デスクトップの共有

*ログインしているユーザーのデスクトップを **x0vncserver** を使用して共有するには、以下の手順を実行します。*

1. ***root** で次のコマンドを実行します。*

```
~]# yum install tigervnc-server
```

2. *ユーザーの VNC パスワードを設定します。*

```
~]$ vncpasswd
Password:
Verify:
```

3. *そのユーザーで以下のコマンドを入力します。*

```
~]$ x0vncserver -PasswordFile=.vnc/passwd -AlwaysShared=1
```

ファイアウォールがポート **5900** への接続を許可するよう設定されている場合、リモートビューアーはディスプレイ **0** に接続し、ログインしているユーザーのデスクトップを表示できます。ファイアウォールの設定方法は 「VNC のためのファイアウォールの設定」 を参照してください。

13.3. VNC ビューアー

vncviewer は、グラフィカルユーザーインターフェイスを表示し、**vncserver** をリモートで制御するプログラムです。

vncviewer の操作では、エントリーを含むポップアップメニューがあり、これでフルスクリーンモードの切り替え、ビューアーの終了などの様々なアクションを実行します。また、ターミナルから **vncviewer** を操作することもできます。**vncviewer** のパラメーターのリストを表示するには、コマンドラインで **vncviewer -h** を入力します。

13.3.1. VNC ビューアーのインストール

TigerVNC (**vncviewer**) クライアントをインストールするには **root** で以下のコマンドを発行します。

```
~]# yum install tigervnc
```

13.3.2. VNC サーバーへの接続

VNC サーバーを設定すると、VNC サーバーを任意の VNC ビューアーに接続できます。

SSH を使用した VNC サーバーへの接続

1. 引数なしで **vncviewer** コマンドを入力すると、**VNC Viewer: Connection Details** ユーティリティーが表示されます。接続する **VNC** サーバーを指定するよう要求されます。
2. 必要な場合は、同じ画面への既存の VNC 接続の切断を回避するために、以下のようにデスクトップの共有を許可するオプションを選択します。
   a. **Options (オプション)** ボタンを選択します。
   b. **Misc. (その他)** タブを選択します。
   c. **Shared (共有)** ボタンを選択します。
   d. **OK** を選択してメインメニューに戻ります。
3. 接続するアドレスとディスプレイ番号を入力してください。

   ```
   address:display_number
   ```

4. **Connect (接続)** を押して VNC サーバー画面に接続します。
5. VNC パスワードを入力するよう求められます。これは、グローバルなデフォルトの VNC パスワードが設定されていない限り、ディスプレイ番号に対応するユーザーの VNC パスワードです。
   VNC サーバーデスクトップを示すウィンドウが表示されます。これは通常のユーザーに表示されるデスクトップではなく、Xvnc デスクトップであることに注意してください。

CLI を使用した VNC サーバーへの接続

1. 引数としてアドレスとディスプレイ番号を指定して、**viewer** コマンドを入力します。

```
vncviewer address:display_number
```

ここで、address は **IP** アドレスまたはホスト名です。

2. VNC パスワードを入力して自分自身を認証します。これは、グローバルなデフォルトの VNC パスワードが設定されていない限り、ディスプレイ番号に対応するユーザーの VNC パスワードです。

3. VNC サーバーデスクトップを示すウィンドウが表示されます。これは通常のユーザーに表示されるデスクトップではなく、Xvnc デスクトップであることに注意してください。

13.3.2.1. VNC のためのファイアウォールの設定

暗号化されていない接続を使用する場合は、**firewalld** が接続を拒否する可能性があります。**firewalld** が VNC パケットを通過させることを許可するには、**TCP** トラフィックに特定のポートを開きます。**-via** オプションを使用する場合、トラフィックは **firewalld** においてデフォルトで有効な **SSH** を介してリダイレクトされます。

注記

VNC サーバーのデフォルトのポートは 5900 です。リモートデスクトップにアクセス可能なポートに到達するには、このデフォルトのポートとユーザーに割り当てられたディスプレイ番号の合計を計算します。たとえば、2 つ目のディスプレイは 2 + 5900 = 5902 のようになります。

ディスプレイ **0** から **3** については、以下で説明しているように **service** オプションによって、VNC サービスの **firewalld** のサポートを利用します。ディスプレイ番号が **3** よりも大きい場合は、*firewalld でポートを開く* で説明されているように、対応するポートを特別に開く必要があります。

firewalld での VNC サービスの有効化

1. **firewalld** 設定の情報を確認するには、以下のコマンドを実行します。

```
~]$ firewall-cmd --list-all
```

2. 特定のアドレスからのすべての VNC 接続を許可するには、以下のコマンドを実行します。

```
~]# firewall-cmd --add-rich-rule='rule family="ipv4" source address="192.168.122.116"
service name=vnc-server accept'
success
```

ここで変更する内容は、システムの再起動後は維持されないことに注意してください。ファイアウォールを永続的に変更するには、コマンドに **--permanent** オプションを繰り返し追加してください。ファイアウォールのリッチ言語コマンドの使用方法は *Red Hat Enterprise Linux 7 Security Guide (Red Hat Enterprise Linux 7 セキュリティーガイド)* を参照してください。

3. 上記の設定を確認するには、以下のコマンドを使用します。

```
~]# firewall-cmd --list-all
public (default, active)
 interfaces: bond0 bond0.192
 sources:
```

```
services: dhcpv6-client ssh
ports:
masquerade: no
forward-ports:
icmp-blocks:
rich rules:
rule family="ipv4" source address="192.168.122.116" service name="vnc-server" accept
```

特定のポートまたはポートの範囲を開くには、**--add-port** オプションを使用して **firewall-cmd** コマンドラインツールに使用します。たとえば、VNC 画面 **4** では、**TCP** トラフィックに対してポート **5904** を開く必要があります。

firewalld でポートを開く

1. パブリックゾーンで **TCP** トラフィックのポートを開くには、**root** で以下のようにコマンドを発行します。

```
~]# firewall-cmd --zone=public --add-port=5904/tcp
success
```

2. パブリックゾーンに対して現在開かれているポートを表示するには、以下のコマンドを発行します。

```
~]# firewall-cmd --zone=public --list-ports
5904/tcp
```

ポートは、**firewall-cmd --zone=**zone **--remove-port=**number/protocol コマンドを使用して削除できます。

ここで変更する内容は、システムの再起動後は維持されないことに注意してください。ファイアウォールを永続的に変更するには、コマンドに **--permanent** オプションを繰り返し追加してください。**firewalld** でのポートのオープンおよびクローズの詳細については、Red Hat Enterprise Linux 7 Security Guide を参照してください。

13.3.3. SSH を使用した VNC サーバーへの接続

VNC は、通信への攻撃に対するセキュリティーがないクリアテキストネットワークプロトコルです。通信をセキュアにするには、**-via** オプションを指定してサーバークライアント接続を暗号化します。これにより、VNC サーバーとクライアントとの間に **SSH** トンネルが作成されます。

VNC サーバークライアント接続を暗号化するコマンドの形式は以下のとおりです。

```
vncviewer -via user@host:display_number
```

例13.1 **-via** オプションの使用

1. **SSH** を使用して VNC サーバーに接続するには、以下のようなコマンドを入力します。

```
~]$ vncviewer -via USER_2@192.168.2.101:3
```

2. プロンプトが表示されたら、パスワードを入力し、**Enter** を押して確認します。
3. リモートデスクトップのウィンドウが画面に表示されます。

–

VNC アクセスの制限
*暗号化した接続のみを選択する場合は、**systemd.service** ファイルの ExecStart 行で **-localhost** オプションを使用すると、暗号化されていない接続を完全に回避できます。*

```
ExecStart=/usr/sbin/runuser -l user -c "/usr/bin/vncserver -localhost %i"
```

*これにより、**vncserver** は、**-via** オプションの結果として **SSH** を使用して送信されたローカルホストおよびポート転送された接続以外の接続を許可しません。*

***SSH** の使用の詳細は、12章OpenSSH を参照してください。*

13.4. 関連情報

TigerVNC に関する詳細は、以下の資料を参照してください。

インストールされているドキュメント

- ***vncserver(1)**: VNC サーバーユーティリティーの man ページです。*
- ***vncviewer(1)**: VNC ビューアーの man ページです。*
- ***vncpasswd(1)**: VNC パスワードコマンドの man ページです。*
- ***Xvnc(1)**: Xvnc サーバー設定オプションの man ページです。*
- ***x0vncserver(1)**: 既存の X サーバーを共有する **TigerVNC** サーバーの man ページです。*

パート V. サーバー

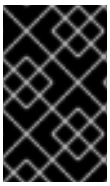
ここでは、Web サーバーの設定方法やネットワーク上でのファイルとディレクトリーの共有方法など、サーバーに関連する様々なトピックを説明します。

第14章 WEB サーバー

Web サーバー は、Web 経由でクライアントにコンテンツを提供するネットワークサービスです。これは通常 Web ページを指しますが、他のドキュメントも当てはまります。Web サーバーは、**ハイパーテキスト転送プロトコル (HTTP)** を使用するため、HTTP サーバーとも呼ばれます。

Red Hat Enterprise Linux 7 で利用可能な Web サーバーは以下のとおりです。

- Apache HTTP サーバー
- nginx



重要

nginx web サーバーは、Red Hat Enterprise Linux 7 の Software Collection としてのみ利用できます。nginx へのアクセス方法や、Software Collections などの使用方法は Red Hat Software Collections のリリースノートを参照してください。

14.1. APACHE HTTP サーバー

本セクションでは、**Apache HTTP Server 2.4** の **httpd** を説明します。これは、Apache Software Foundation により開発されたオープンソースの Web サーバーです。

Red Hat Enterprise Linux の以前のリリースからアップグレードする場合は、適切に **httpd** サービス設定を更新する必要があります。本セクションでは、新たに追加された機能のいくつか、Apache HTTP Server 2.4 とバージョン 2.2 の重要な違い、古い設定ファイルの更新方法について説明します。

14.1.1. 主な変更点

Red Hat Enterprise Linux 6 と比較して Red Hat Enterprise Linux 7 の Apache HTTP Server には以下の変更があります。

httpd サービスの制御

SysV init スクリプトからの移行に伴い、サーバー管理者は **service** コマンドの代わりに **apachectl** コマンドと **systemctl** コマンドを使用してサービスを制御することが推奨されます。以下の例は、**httpd** サービスに固有です。
コマンド:

```
service httpd graceful
```

は以下に置き換えられます。

```
apachectl graceful
```

httpd 用の **systemd** ユニットファイルは、以下のように init スクリプトと異なる動作を持ちます。

- サービスがリロードすると、デフォルトで正常な再起動が使用されます。
- サービスが停止すると、デフォルトで正常な停止が使用されます。
  コマンド:

```
service httpd configtest
```

は以下に置き換えられます。

```
apachectl configtest
```

プライベート /tmp

システムのセキュリティーを高めるため、**systemd** ユニットファイルは、プライベートの **/tmp** ディレクトリーを使用して **httpd** デーモンを実行します。これは、システムの **/tmp** ディレクトリーとは別のものです。

設定レイアウト

モジュールをロードする設定ファイルは、**/etc/httpd/conf.modules.d/** ディレクトリーにあります。このディレクトリーには、**php** など、追加でロードできる **httpd** 用モジュールを提供するパッケージによりファイルが配置されます。**/etc/httpd/conf/httpd.conf** ファイルのメインセクションの前に置かれた **Include** ディレクティブは、**/etc/httpd/conf.modules.d/** ディレクトリー内にファイルを追加するために使用されます。つまり、**conf.modules.d/** にある設定ファイルはすべて、**httpd.conf** のメインセクションが処理される前に処理されます。**/etc/httpd/conf.d/** ディレクトリーのファイルに対する **IncludeOptional** ディレクティブは、**httpd.conf** ファイルの最後に配置されます。したがって、**/etc/httpd/conf.d/** 内のファイルは、**httpd.conf** のセクションが処理されてから処理されます。
そのほかにも、**httpd** パッケージが設定ファイルをいくつか提供しています。

- **/etc/httpd/conf.d/autoindex.conf**: これは mod_autoindex ディレクトリーのインデックス作成を設定します。
- **/etc/httpd/conf.d/userdir.conf**: これにより、http://example.com/~username/ などのユーザーディレクトリーへのアクセスが設定されます。このようなアクセスは、セキュリティー上の理由からデフォルトでは無効になっています。
- **/etc/httpd/conf.d/welcome.conf**: 以前のリリースと同様に、コンテンツがない場合に http://localhost/ に表示されるウェルカムページが設定されます。

デフォルト設定

最小限の **httpd.conf** ファイルがデフォルトで提供されるようになりました。**Timeout** や **KeepAlive** といった一般的な設定の多くは、デフォルトで明示的に設定されることはなくなり、ハードコーディングされるようになります。このハードコーディングされた全設定ディレクティブ用のデフォルト設定は、マニュアルに記載されています。詳細は、「インストール可能なドキュメント」 を参照してください。

互換性がない構文の変更

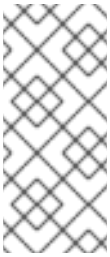
既存の設定を **httpd 2.2** から **httpd 2.4** に移行する場合は、**httpd** 設定の構文に後方互換性がない変更が含まれるため、変更が必要になります。http://httpd.apache.org/docs/2.4/upgrading.html のアップグレードに関する詳細は、以下の Apache ドキュメントを参照してください。

処理モデル

Red Hat Enterprise Linux の以前のリリースでは、さまざまな**マルチプロセスモデル** (**MPM**) が、さまざまな **httpd** バイナリーとして利用できていました。つまり、分岐モデルの prefork を **/usr/sbin/httpd** として、またスレッドベースのモデルである worker を **/usr/sbin/httpd.worker** としてしました。
Red Hat Enterprise Linux 7 では、単独の **httpd** バイナリーのみが使われ、3 つの MPM はロード可能なモジュール (worker、prefork (デフォルト)、および event) として利用可能です。必要に応じて、コメント文字 **#** を追加および削除して 3 つの MPM モジュールの 1 つだけがロードされるよう設定ファイル **/etc/httpd/conf.modules.d/00-mpm.conf** を編集します。

パッケージ変更

*LDAP 認証および承認の各モジュールは、個別のサブパッケージ **mod_ldap** で提供されています。新たなモジュール **mod_session** と関連のヘルパーモジュールは、新しいサブパッケージ **mod_session** で提供されています。新しいモジュールの **mod_proxy_html** と **mod_xml2enc** は、新しいサブパッケージ **mod_proxy_html** で提供されています。これらのパッケージはすべて Optional チャンネルにあります。*



注記

Optional および Supplementary チャンネルをサブスクライブする前に、対象範囲の詳細 を参照してください。これらのチャンネルからパッケージをインストールする場合は、Red Hat カスタマーポータルの記事 証明書ベースの管理を使用して、Optional および Supplementary チャンネル、-devel パッケージにアクセスする方法 で説明されている手順を行ってください。

ファイルシステムのレイアウトのパッケージ

***/var/cache/mod_proxy/** ディレクトリーは提供されなくなりました。代わりに、**/var/cache/httpd/** ディレクトリーが **proxy** サブディレクトリーおよび **ssl** サブディレクトリーとパッケージ化されています。*
***httpd** で提供されていたパッケージ化されたコンテンツは、**/var/www/** から **/usr/share/httpd/** に移動しています。*

- ***/usr/share/httpd/icons/**: ディレクトリーインデックスで使用されるアイコンセットを含むディレクトリーは、(以前の **/var/www/icons/** ディレクトリーから) **/usr/share/httpd/icons/** に変更になりました。デフォルト設定では* http://localhost/icons/ *で利用可能です。アイコンの場所と利用可能性は、**/etc/httpd/conf.d/autoindex.conf** ファイルで設定可能です。*

- ***/usr/share/httpd/manual/**: **/var/www/manual/** は **/usr/share/httpd/manual/** に変更になりました。**httpd**-manual パッケージに含まれるこのディレクトリーには **httpd** の HTML バージョンのマニュアルが含まれます。このパッケージがインストールされている場合は* http://localhost/manual/ *で利用できます。マニュアルの場所と利用可能性は **/etc/httpd/conf.d/manual.conf** ファイルで設定可能です。*

- ***usr/share/httpd/error/**: **/var/www/error/** は **/usr/share/httpd/error/** に移動しました。カスタムの複数言語の HTTP エラーページです。デフォルトでは設定されておらず、設定ファイルの例は **/usr/share/doc/httpd-**VERSION**/httpd-multilang-errordoc.conf** で提供されています。*

認証、認可、およびアクセス制御

*認証、認可、およびアクセス制御に使用される設定ディレクティブは、大幅に変更されました。**Order**、**Deny**、および **Allow** の各ディレクティブを使用している既存の設定ファイルは、新たな **Require** 構文を使うようにしてください。詳細は、以下の Apache ドキュメントを参照してください。http://httpd.apache.org/docs/2.4/howto/auth.html*

suexec

*システムのセキュリティーを改善するために、**suexec** バイナリーは、**root** ユーザーと同じようにインストールされなくなりました。代わりに、より限定的なパーミッションセットを可能にするファイルシステムのケイパビリティーセットがあります。この変更に加えて、**suexec** バイナリーは **/var/log/httpd/suexec.log** ログファイルを使用しなくなりました。代わりに、ログメッセージは **syslog** に送信されます。デフォルトでは、これらのメッセージは **/var/log/secure** ログファイルに表示されます。*

モジュールインターフェイス

***httpd** モジュールインターフェイスが変更したため、**httpd 2.2** に対して構築されたサードパーティーのバイナリーモジュールは、**httpd 2.4** と互換性がありません。これらのモジュールは、必要に応じて **httpd 2.4** モジュールインターフェイス用に調整し、再構築する必要があります。バージョ*

ン **2.4** の API の変更点の詳細リスト
は、http://httpd.apache.org/docs/2.4/developer/new_api_2_4.html を参照してください。
ソースからのモジュール構築に使用される **apxs** バイナリーは、**/usr/sbin/apxs** および **/usr/bin/apxs** に移動しました。

削除されたモジュール

Red Hat Enterprise Linux 7 で削除された **httpd** モジュールのリスト:

mod_auth_mysql, mod_auth_pgsql

httpd 2.4 は、**mod_authn_dbd** モジュールで SQL データベース認証サポートを内部で提供します。

mod_perl

mod_perl では、アップストリームにより **httpd 2.4** が公式にサポートされていません。

mod_authz_ldap

httpd 2.4 では、**mod_authnz_ldap** を使用してサブパッケージ **mod_ldap** の LDAP サポートが提供されます。

14.1.2. 設定の更新

Apache HTTP サーバーバージョン 2.2 の設定ファイルを更新するには、以下の手順を行います。

1. モジュールは変更されている可能性があるため、すべてのモジュール名が正しいことを確認してください。名前が変更された各モジュールの **LoadModule** ディレクティブを調整します。
2. サードパーティーのモジュールは、読み込みを試みる前にすべて再コンパイルをします。これは一般的に認証と権限付与のモジュールに該当します。
3. **mod_userdir** モジュールを使用する場合は、ディレクトリー名 (通常は **public_html**) を示す **UserDir** ディレクティブが指定されていることを確認してください。
4. Apache HTTP Secure Server を使用する場合は、Secure Sockets Layer (SSL) プロトコルの有効化に関する重要な情報について 「mod_ssl モジュールの有効化」 を参照してください。

以下のコマンドを使用すると、設定エラーを確認できます。

```
~]# apachectl configtest
Syntax OK
```

Apache HTTP サーバーの設定をバージョン 2.2 から 2.4 に更新する方法は
http://httpd.apache.org/docs/2.4/upgrading.html を参照してください。

14.1.3. httpd サービスの実行

本セクションでは、Apache HTTP Server の起動、停止、再起動、および現在のステータスの確認を行う方法を説明します。**httpd** サービスを使用するには、**httpd** パッケージがインストールされていることを確認します。これを行うには、次のコマンドを使用します。

```
~]# yum install httpd
```

ターゲットの概念と Red Hat Enterprise Linux 内のシステムサービスの管理方法全般の詳細は、10 章systemd によるサービス管理 を参照してください。

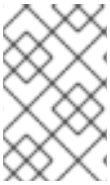
14.1.3.1. サービスの起動

httpd サービスを実行する場合は、**root** で次のコマンドを実行します。

```
~]# systemctl start httpd.service
```

システムの起動時にサービスを自動的に起動するようにするには、以下のコマンドを使用します。

```
~]# systemctl enable httpd.service
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to
/usr/lib/systemd/system/httpd.service.
```



注記

Apache HTTP サーバーをセキュアサーバーとして実行している場合は、暗号化したプライベートの SSL キーを使用すると、マシンが起動した後にパスワードが要求される可能性があります。

14.1.3.2. サービスの停止

実行中の **httpd** サービスを停止するには、**root** で次のコマンドを実行します。

```
~]# systemctl stop httpd.service
```

システムの起動時にサービスが自動的に起動しないようにするには、以下を入力します。

```
~]# systemctl disable httpd.service
Removed symlink /etc/systemd/system/multi-user.target.wants/httpd.service.
```

14.1.3.3. サービスの再起動

実行中の **httpd** サービスを再起動する方法は、3 通りあります。

1. サービスを完全に再起動するには、**root** で次のコマンドを入力します。

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

   これにより、稼働中の **httpd** サービスが停止し、すぐに再起動します。このコマンドは、PHP など、動的に読み込んだモジュールをインストールまたは削除した後に使用します。

2. 設定のリロードだけを行うには、**root** で以下のコマンドを入力します。

   ```
   ~]# systemctl reload httpd.service
   ```

   これにより、実行中の **httpd** サービスが、設定ファイルを再読み込みします。現在処理中の要求はいずれも割り込みされるので、クライアントのブラウザーがエラーメッセージを表示したり、ページの一部のみが表示される可能性があります。

3. アクティブな要求に影響を与えずに設定を再読み込みするには、**root** で次のコマンドを入力します。

   ```
   ~]# apachectl graceful
   ```

これにより、実行中の **httpd** サービスが、設定ファイルを再読み込みします。現在処理中のリクエストは、引き続き古い設定を使用します。

Red Hat Enterprise Linux 7 でシステムサービスを管理する方法は、10章systemd によるサービス管理 を参照してください。

14.1.3.4. サービスステータスの確認

httpd サービスが実行していることを確認するには、シェルプロンプトで以下を入力します。

```
~]# systemctl is-active httpd.service
active
```

14.1.4. 設定ファイルの編集

httpd サービスが起動すると、デフォルトでは、表14.1「httpd サービスの設定ファイル」 にリスト表示されている場所から設定が読み込まれます。

表14.1 httpd サービスの設定ファイル

パス	詳細
/etc/httpd/conf/httpd.conf	主要設定ファイル。
/etc/httpd/conf.d/	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。

デフォルト設定はほとんどの状況に適していますが、重要な設定オプションをいくつか知っておくと役に立ちます。変更を有効にするには、web サーバーを再起動する必要があることに注意してください。**httpd** サービスを再起動する方法は、「サービスの再起動」 を参照してください。

設定に誤りがないことを確認するには、シェルプロンプトで以下のコマンドを実行します。

```
~]# apachectl configtest
Syntax OK
```

ファイルの変更に失敗してもすぐにやり直せるように、ファイルを編集する前にコピーを作成しておくことを推奨します。

14.1.5. モジュールの使用

httpd サービスは、モジュールアプリケーションであるため、多数の Dynamic Shared Objects (DSO) と一緒に配布されます。これは、必要に応じてランタイム時に、動的に読み込まれたり、読み込みが破棄されたりします。Red Hat Enterprise Linux 7 では、これらのモジュールは **/usr/lib64/httpd/modules/** 内にあります。

14.1.5.1. モジュールの読み込み

特定の DSO モジュールを読み込むには、**LoadModule** ディレクティブを使用します。個別のパッケージで提供されるモジュールでは多くの場合、設定ファイルが **/etc/httpd/conf.d/** ディレクトリーにあることに注意してください。

例14.1 mod_ssl DSO の読み込み

```
LoadModule ssl_module modules/mod_ssl.so
```

操作が終了したら、Web サーバーを再起動して設定を再読み込みします。**httpd** サービスを再起動する方法は、「サービスの再起動」 を参照してください。

14.1.5.2. モジュールの作成

新たに DSO モジュールを作成する場合は、**httpd-devel** パッケージがインストールされていることを確認してください。これを行うには、**root** で以下のコマンドを実行します。

```
~]# yum install httpd-devel
```

このパッケージには、モジュールをコンパイルするために必要なインクルードファイル、ヘッダーファイル、および **APache eXtenSion** (**apxs**) ユーティリティーが含まれます。

作成したら、以下のコマンドでモジュールを構築できます。

```
~]# apxs -i -a -c module_name.c
```

ビルドが成功すると、Apache HTTP Server で配布されるその他のモジュールと同じ方法でモジュールを読み込むことができます。

14.1.6. 仮想ホストの設定

Apache HTTP Server に組み込まれている仮想ホストを使用すると、どの IP アドレス、ホスト名、またはポートが要求されているかに基づいて、サーバーが異なる情報を提供できます。

名前ベースの仮想ホストを作成するには、まず設定ファイルの例である **/usr/share/doc/httpd-**VERSION**/httpd-vhosts.conf** を **/etc/httpd/conf.d/** ディレクトリーにコピーし、**@@Port@@** と **@@ServerRoot@@** のプレースホルダーの値を置き換えます。例14.2「仮想ホストの設定例」 に従って、要件に応じてオプションをカスタマイズします。

例14.2 仮想ホストの設定例

```
<VirtualHost *:80>
  ServerAdmin webmaster@penguin.example.com
  DocumentRoot "/www/docs/penguin.example.com"
  ServerName penguin.example.com
  ServerAlias www.penguin.example.com
  ErrorLog "/var/log/httpd/dummy-host.example.com-error_log"
  CustomLog "/var/log/httpd/dummy-host.example.com-access_log" common
</VirtualHost>
```

ServerName は、マシンに割り当てられた有効な DNS 名を指定する必要があることに注意してください。**<VirtualHost>** コンテナーは高度なカスタマイズが可能で、メインサーバーの設定で利用可能なディレクティブのほとんどを使用できます。このコンテナーで対応して**いない** ディレクティブには、**SuexecUserGroup** に置き換えられた **User** および **Group** が含まれます。

注記

*仮想ホストがデフォルト以外のポートをリッスンするように設定している場合は、それに応じて **/etc/httpd/conf/httpd.conf** ファイルのグローバル設定セクションの **Listen** ディレクティブを必ず更新してください。*

*新規に作成された仮想ホストをアクティブ化するには、Web サーバーを再起動する必要があります。**httpd** サービスを再起動する方法は、「サービスの再起動」 を参照してください。*

14.1.7. SSL サーバーの設定

*Secure Sockets Layer (**SSL**) は、サーバーとクライアントが安全に接続できるようにする暗号化プロトコルです。Transport Layer Security (**TLS**) と呼ばれる拡張され改善されたバージョンとともに、プライバシーとデータの整合性の両方が確保されます。Apache HTTP Server を **mod_ssl** と組み合わせて、OpenSSL ツールキットを使用して SSL/TLS サポートを提供するモジュールは、**SSL サーバー** と呼ばれます。Red Hat Enterprise Linux は、TLS 実装としての Mozilla NSS の使用もサポートします。Mozilla NSS のサポートは **mod_nss** モジュールにより提供されます。*

傍受できる人であればだれでも読み取りや修正が可能な HTTP 接続とは異なり、HTTPS と呼ばれる SSL/TLS over HTTP の使用により、送信したコンテンツの検査や修正が阻止されます。本セクションでは、Apache HTTP Server 設定内でこのモジュールを有効にする基本的な方法と、秘密鍵と自己署名の証明書の生成プロセスを説明します。

14.1.7.1. 証明書およびセキュリティーの概要

*通信の安全性は、鍵の使用により確保されます。従来の暗号または **対称暗号** では、トランザクションの両端に、相互に送信をデコードするのに使用する鍵と同じ鍵があります。一方、公開暗号または **非対称暗号** では、**秘密鍵** (秘密を保持する) と **公開鍵** (通常は公共と共有) が共存します。公開鍵でエンコードされたデータは秘密鍵でしかデコードできませんが、秘密鍵でエンコードされたデータは公開鍵でしかデコードできません。*

*SSL を使用した安全な通信を提供するには、SSL サーバーが **認証機関** (**CA**) が署名したデジタル証明書を使用する必要があります。証明書は、サーバーの各種属性 (サーバーのホスト名、会社名、会社の場所など) と、CA の秘密鍵を使用して生成した署名をリスト表示します。この署名は、特定の認証局が証明書を署名し、証明書がいかなる方法でも変更されていないことを確認するものです。*

Web ブラウザーが新しい SSL 接続を確立すると、Web サーバーが提供する証明書が確認されます。証明書に、信頼できる CA からの署名がない場合や、証明書にリスト表示されているホスト名が、接続の確立に使用されたホスト名と一致しない場合は、サーバーとの通信が拒否され、通常は、ユーザーに適切なエラーメッセージが表示されます。

デフォルトでは、ほとんどの Web ブラウザーは、幅広く使用されている一連の証明書局を信頼するように設定されています。このため、安全なサーバーを設定する際には、適切な CA を選択して、ターゲットユーザーが接続を信頼できる状態にする必要があります。適切な CA を選択しないとエラーメッセージが表示され、証明書を手動で指定することが必要になります。ユーザーが証明書エラーを上書きするようにすると、攻撃者が接続を傍受できるため、可能な場合は信頼できる CA を使用する必要があります。詳細は、表14.2「一般的な Web ブラウザーが使用する CA リストに関する情報」 を参照してください。

*表14.2 一般的な **Web** ブラウザーが使用する **CA** リストに関する情報*

Web ブラウザー	リンク
Mozilla Firefox	Mozilla root CA のリスト

Web ブラウザー	リンク
Opera	Opera で使用されるルート証明書に関する情報
Internet Explorer	Microsoft Windows で使用されるルート証明書に関する情報
Chromium	Chromium プロジェクトが使用するルート証明書に関する情報。

SSL サーバーをセットアップする際には、証明書要求と秘密鍵を生成してから、証明書要求と企業の識別証明と支払いを認証局に送ります。CA が証明書の要求および ID を確認すると、サーバーで使用できる署名付きの証明書が送信されます。また、CA 署名を含まない自己署名証明書を作成することもできるため、テスト目的でのみ使用してください。

14.1.8. mod_ssl モジュールの有効化

mod_ssl *を使用して SSL または HTTPS サーバーをセットアップする場合は、別のアプリケーションまたはモジュール (***mod_nss*** など) で同じポートを使用するよう 設定できません。ポート* ***443*** *は、HTTPS のデフォルトポートです。*

mod_ssl *モジュールおよび OpenSSL ツールキットを使用して SSL サーバーを設定するには、* ***mod_ssl*** *および* ***openssl*** *パッケージをインストールします。* ***root*** *で次のコマンドを実行します。*

```
~]# yum install mod_ssl openssl
```

これにより、 ***/etc/httpd/conf.d/ssl.conf*** *に* ***mod_ssl*** *設定ファイルが作成されます。このファイルは、デフォルトでメインとなる Apache HTTP Server 設定ファイルに含まれています。モジュールを読み込むには、「サービスの再起動」 の説明どおりに* ***httpd*** *サービスを再起動します。*

重要

POODLE: SSLv3 脆弱性 (CVE-2014-3566) *で説明されている脆弱性のため、Red Hat は、* ***SSL*** *を無効にし、* ***TLSv1.1*** *または* ***TLSv1.2*** *のみを使用することを推奨します。後方互換性は、* ***TLSv1.0*** *を使用して実現できます。Red Hat がサポートする多くの製品は* ***SSLv2*** *または* ***SSLv3*** *プロトコルを使用するか、デフォルトでそれらのプロトコルを有効にできます。ただし、* ***SSLv2*** *または* ***SSLv3*** *を使用することが強く推奨されます。*

14.1.8.1. mod_ssl での SSL および TLS の有効化および無効化

SSL および TLS プロトコルの特定のバージョンを有効および無効にするには、設定ファイルの ## SSL Global Context セクション に ***SSLProtocol*** *ディレクティブを追加し、他のすべての部分でそのディレクティブを削除するか、すべての VirtualHost セクションの # SSL Protocol support の下にあるデフォルトエントリーを編集することによりグローバルで設定します。ドメインごとの VirtualHost セクションで指定しない場合、設定はグローバルセクションから継承されます。プロトコルバージョンが確実に無効になるように、管理者は* ***SSLProtocol*** *を、SSL Global Context セクションに* ***のみ*** *指定するか、ドメインごとの* ***すべて*** *の VirtualHost セクションで指定する必要があります。*

SSLv2 および SSLv3 の無効化

すべての VirtualHost セクションで SSL バージョン 2 および SSL バージョン 3 を無効にするには (SSL バージョン 2 および SSL バージョン 3 以外をすべて有効にすることを意味します)、以下の手順を実行します。

1. ***root*** *で* ***/etc/httpd/conf.d/ssl.conf*** *ファイルを開き、*SSLProtocol *ディレクティブの すべてのインスタンスを検索します。デフォルトでは、設定ファイルに以下のような1 つのセクションが含まれます。*

```
~]# vi /etc/httpd/conf.d/ssl.conf
#  SSL Protocol support:
# List the enable protocol levels with which clients will be able to
# connect. Disable SSLv2 access by default:
SSLProtocol all -SSLv2
```

このセクションは、VirtualHost セクション内にあります。

2. ***SSLProtocol*** *行を以下のように編集します。*

```
#  SSL Protocol support:
# List the enable protocol levels with which clients will be able to
# connect. Disable SSLv2 access by default:
SSLProtocol all -SSLv2 -SSLv3
```

すべての VirtualHost セクションに対してこのアクションを繰り返します。ファイルを保存してから閉じます。

3. *すべての* ***SSLProtocol*** *ディレクティブが以下のように変更されたことを確認します。*

```
~]# grep SSLProtocol /etc/httpd/conf.d/ssl.conf
SSLProtocol all -SSLv2 -SSLv3
```

この手順は、 デフォルトの VirtualHost セクションが複数の場合に特に重要になります。

4. *以下のように Apache デーモンを再起動します。*

```
~]# systemctl restart httpd
```

セッションが中断されることに注意してください。

TLS 1 以上を除くすべての SSL および TLS プロトコルの無効化

TLS バージョン1 以上を除く SSL および TLS プロトコルバージョンをすべて無効にするには、以下の手順を実行します。

1. ***root*** *で* ***/etc/httpd/conf.d/ssl.conf*** *ファイルを開き、* ***SSLProtocol*** *ディレクティブの* **すべて** *のインスタンスを検索します。デフォルトでは、このファイルには以下のようなセクションが含まれます。*

```
~]# vi /etc/httpd/conf.d/ssl.conf
#  SSL Protocol support:
# List the enable protocol levels with which clients will be able to
# connect. Disable SSLv2 access by default:
SSLProtocol all -SSLv2
```

2. ***SSLProtocol*** *行を以下のように編集します。*

```
#  SSL Protocol support:
# List the enable protocol levels with which clients will be able to
# connect. Disable SSLv2 access by default:
```

```
SSLProtocol -all +TLSv1 +TLSv1.1 +TLSv1.2
```

ファイルを保存してから閉じます。

3. 以下のように変更を確認します。

```
~]# grep SSLProtocol /etc/httpd/conf.d/ssl.conf
SSLProtocol -all +TLSv1 +TLSv1.1 +TLSv1.2
```

4. 以下のように Apache デーモンを再起動します。

```
~]# systemctl restart httpd
```

セッションが中断されることに注意してください。

SSL および TLS プロトコルのステータスのテスト

有効または無効な SSL および TLS のバージョンを確認するには、**openssl s_client -connect** コマンドを使用します。このコマンドの形式は以下のとおりです。

```
openssl s_client -connect hostname:port -protocol
```

port は、テストするポートで、protocol は、テストするプロトコルバージョンです。ローカルで稼働している SSL サーバーをテストする場合は、**localhost** をホスト名として使用します。たとえば、SSLv3 が有効であるかどうかを確認するためにセキュアな HTTPS 接続のデフォルトポートであるポート **443** をテストするには、以下のようにコマンドを発行します。

```
~]# openssl s_client -connect localhost:443 -ssl3
CONNECTED(00000003)
139809943877536:error:14094410:SSL routines:SSL3_READ_BYTES:sslv3 alert handshake
failure:s3_pkt.c:1257:SSL alert number 40
139809943877536:error:1409E0E5:SSL routines:SSL3_WRITE_BYTES:ssl handshake
failure:s3_pkt.c:596:
output omitted
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
SSL-Session:
  Protocol : SSLv3
output truncated
```

上記の出力は、ハンドシェイクが失敗し、暗号化がネゴシエートされなかったことを示しています。

```
~]$ openssl s_client -connect localhost:443 -tls1_2
CONNECTED(00000003)
depth=0 C = --, ST = SomeState, L = SomeCity, O = SomeOrganization, OU =
SomeOrganizationalUnit, CN = localhost.localdomain, emailAddress = root@localhost.localdomain
output omitted
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
```

```
SSL-Session:
  Protocol : TLSv1.2
output truncated
```

上記の出力は、ハンドシェイクが失敗せず、暗号化セットがネゴシエートされたことを示しています。

openssl s_client コマンドのオプションは、**s_client(1)** man ページに記載されています。

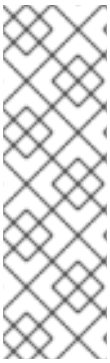
SSLv3 の脆弱性とそのテスト方法の詳細は、Red Hat ナレッジベースの記事 POODLE: SSLv3 脆弱性 (CVE-2014-3566) を参照してください。

14.1.9. mod_nss Module の有効化

mod_nss を使用して HTTPS サーバーをセットアップする場合は、**mod_ssl** がデフォルトで **443** (デフォルトの HTTPS ポート) を使用するため、**mod_ssl** パッケージをデフォルト設定でインストール**できません**。可能な限り、このパッケージを削除します。

mod_ssl を削除するには、**root** で以下のコマンドを入力します。

```
~]# yum remove mod_ssl
```



注記

他の目的のために **mod_ssl** が必要な場合は、**443** 以外のポートを使用するよう **/etc/httpd/conf.d/ssl.conf** を変更して、リッスンするポートが **443** に変更されたときに **mod_ssl** と **mod_nss** が競合しないようにします。

1 つのモジュールのみがポートを所有できるため、**mod_nss** と **mod_ssl** は一意のポートを使用している場合のみ同時に共存できます。このため、**mod_nss** はデフォルトで **8443** を使用しますが、HTTPS のデフォルトポートはポート **443** です。ポートは **Listen** ディレクティブと VirtualHost 名またはアドレスで指定されます。

NSS のすべてはトークンに関連付けられます。ソフトウェアトークンは NSS データベースに存在しますが、証明書を含む物理トークンを使用することもできます。OpenSSL では、個別証明書と秘密鍵は PEM ファイルに保持されます。NSS では、これらはデータベースに格納されます。各証明書およびキーはトークンと関連付けられ、各トークンにはパスワードを設定して保護することもできます。このパスワードはオプションですが、パスワードが使用される場合、Apache HTTP サーバーはシステムの起動時にユーザーの介入なしでデータベースを開くためにパスワードのコピーを必要とします。

mod_nss の設定

1. **root** で mod_nss をインストールします。

   ```
   ~]# yum install mod_nss
   ```

   これにより、**/etc/httpd/conf.d/nss.conf** に **mod_nss** 設定ファイルが作成されます。**/etc/httpd/conf.d/** ディレクトリーは、デフォルトでメインの Apache HTTP Server 設定ファイルに含まれます。モジュールを読み込むには、「サービスの再起動」 の説明どおりに **httpd** サービスを再起動します。

2. **root** で **/etc/httpd/conf.d/nss.conf** ファイルを開き、Listen ディレクティブの すべての インスタンスを検索します。
   Listen 8443 行を以下のように編集します。

```
Listen 443
```

ポート **443** は、**HTTPS** のデフォルトポートです。

3. デフォルトの **VirtualHost default:8443** 行を以下のように編集します。

```
VirtualHost default:443
```

デフォルト以外の他のすべての仮想ホストセクション (存在する場合) を編集します。ファイルを保存してから閉じます。

4. Mozilla NSS では、証明書は **/etc/httpd/conf.d/nss.conf** ファイルの **NSSCertificateDatabase** ディレクティブで指定された サーバー証明書データベース に保存されます。デフォルトでは、パスはインストール時に作成された NSS データベースである **/etc/httpd/alias** に設定されます。
デフォルトの NSS データベースを表示するには、以下のコマンドを実行します。

```
~]# certutil -L -d /etc/httpd/alias

Certificate Nickname                    Trust Attributes
                                        SSL,S/MIME,JAR/XPI

cacert                                  CTu,Cu,Cu
Server-Cert                             u,u,u
alpha                                   u,pu,u
```

上記のコマンド出力で、**Server-Cert** はデフォルトの **NSSNickname** です。**-L** オプションは、証明書データベースにすべての証明書をリスト表示したり、名前付き証明書に関する情報を表示します。**-d** オプションは、証明書およびキーデータベースファイルを含むデータベースディレクトリーを指定します。その他のコマンドラインオプションは、**certutil(1)** man ページを参照してください。

5. 別のデータベースを使用するよう **mod_nss** を設定するには、**/etc/httpd/conf.d/nss.conf** ファイルの **NSSCertificateDatabase** 行を編集します。デフォルトファイルの VirtualHost セクション内には以下の行が含まれます。

```
#  Server Certificate Database:
#  The NSS security database directory that holds the certificates and
#  keys. The database consists of 3 files: cert8.db, key3.db and secmod.db.
#  Provide the directory that these files exist.
NSSCertificateDatabase /etc/httpd/alias
```

上記のコマンド出力で、**alias** はデフォルトの NSS データベースディレクトリーである **/etc/httpd/alias/** です。

6. デフォルトの NSS 証明書データベースにパスワードを適用するには、**root** で以下のコマンドを使用します。

```
~]# certutil -W -d /etc/httpd/alias
Enter Password or Pin for "NSS Certificate DB":
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.
```

```
Enter new password:
Re-enter password:
Password changed successfully.
```

7. *HTTPS サーバーをデプロイする前に、認証局 (CA) により署名された証明書を使用して新しい証明書データベースを作成します。*

   例14.3 Mozilla NSS データベースへの証明書の追加

   ***certutil** コマンドは、CA 証明書を NSS データベースファイルに追加するために使用されます。*

   ```
   certutil -d /etc/httpd/nss-db-directory/ -A -n "CA_certificate" -t CT,, -a -i
   certificate.pem
   ```

   *上記のコマンドは、**certificate.pem** という名前の PEM 形式のファイル に保存されている CA 証明書を追加します。**-d** オプションは、証明書およびキーデータベースファイルを含む NSS データベースディレクトリーを指定します。**-n** オプションは、証明書の名前 (**-t CT,,**) を設定します。これは、証明書が TLS クライアントおよびサーバーで使用されるように信頼されることを意味します。**-A** オプションでは、既存の証明書を証明書データベースに追加します。データベースが存在しない場合は、作成されます。**-a** オプションでは、入力または出力に ASCII 形式を使用でき、**-i** オプションは **certificate.pem** 入力ファイルをコマンドに渡します。*

   *その他のコマンドラインオプションは、**certutil(1)** man ページを参照してください。*

8. *秘密鍵を保護するために、NSS データベースはパスワードで保護する必要があります。*

   例14.4 Mozilla NSS データベースのパスワード設定

   *NSS データベースのパスワードを設定するには、以下のように **certutil** ツールを使用できます。*

   ```
   certutil -W -d /etc/httpd/nss-db-directory/
   ```

   *たとえば、デフォルトのデータベースの場合は、**root** で以下のコマンドを実行します。*

   ```
   ~]# certutil -W -d /etc/httpd/alias
   Enter Password or Pin for "NSS Certificate DB":
   Enter a password which will be used to encrypt your keys.
   The password should be at least 8 characters long,
   and should contain at least one non-alphabetic character.

   Enter new password:
   Re-enter password:
   Password changed successfully.
   ```

9. *以下のように NSSPassPhraseDialog ディレクティブ で行を変更して、NSS 内部ソフトウェアトークンを使用するように **mod_nss** を設定します。*

   ```
   ~]# vi /etc/httpd/conf.d/nss.conf
   NSSPassPhraseDialog file:/etc/httpd/password.conf
   ```

これにより、システムの起動時にパスワードを手動で入力する必要がなくなります。ソフトウェアトークンは NSS データベースに存在しますが、証明書を含む物理トークンを使用することもできます。

10. NSS データベースに含まれる SSL サーバー証明書が RSA 証明書である場合は、**NSSNickname** パラメーターのコメントを解除し、パラメーターが上記の手順 4 で示されたニックネームに一致するようにしてください。

```
~]# vi /etc/httpd/conf.d/nss.conf
NSSNickname Server-Cert
```

NSS データベースに含まれる SSL サーバー証明書が ECC 証明書である場合は、**NSSECCNickname** パラメーターのコメントが解除され、上記の手順 4 で示されたニックネームに一致するようにします。

```
~]# vi /etc/httpd/conf.d/nss.conf
NSSECCNickname Server-Cert
```

NSSCertificateDatabase パラメーターがコメント解除され、上記の手順 4 で示された、または手順 5 で設定された NSS データベースディレクトリーを参照するようにします。

```
~]# vi /etc/httpd/conf.d/nss.conf
NSSCertificateDatabase /etc/httpd/alias
```

/etc/httpd/alias を、使用する証明書データベースへのパスに置き換えます。

11. **root** で **/etc/httpd/password.conf** ファイルを作成します。

```
~]# vi /etc/httpd/password.conf
```

次の形式の行を追加します。

```
internal:password
```

上記の手順 6 で、**パスワード** を NSS セキュリティーデータベースに適用したパスワードに置き換えます。

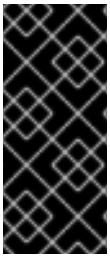
12. 適切な所有権とパーミッションを **/etc/httpd/password.conf** ファイルに適用します。

```
~]# chgrp apache /etc/httpd/password.conf
~]# chmod 640 /etc/httpd/password.conf
~]# ls -l /etc/httpd/password.conf
-rw-r-----. 1 root apache 10 Dec 4 17:13 /etc/httpd/password.conf
```

13. NSS ソフトウェアトークンを使用するよう **/etc/httpd/password.conf** で **mod_nss** を設定するには、**/etc/httpd/conf.d/nss.conf** を以下のように編集します。

```
~]# vi /etc/httpd/conf.d/nss.conf
```

14. 変更を反映するために、「サービスの再起動」 の説明通りに Apache サーバーを再起動します。



重要

POODLE: SSLv3 脆弱性 (CVE-2014-3566) で説明されている脆弱性のため、Red Hat は、**SSL** を無効にし、**TLSv1.1** または **TLSv1.2** のみを使用することを推奨します。後方互換性は、**TLSv1.0** を使用して実現できます。Red Hat がサポートする多くの製品は **SSLv2** または **SSLv3** プロトコルを使用するか、デフォルトでそれらのプロトコルを有効にできます。ただし、**SSLv2** または **SSLv3** を使用することが強く推奨されます。

14.1.9.1. mod_nss での SSL および TLS の有効化および無効化

SSL および TLS プロトコルの特定のバージョンを有効および無効にするには、設定ファイルの## SSL Global Context セクションに **NSSProtocol** ディレクティブを追加し、他のすべての部分でそのディレクティブを削除するか、すべての VirtualHost セクションの# SSL Protocol の下にあるデフォルトエントリーを編集することによりグローバルで設定します。ドメインごとの VirtualHost セクションで指定しない場合、設定はグローバルセクションから継承されます。プロトコルバージョンが確実に無効になるように、管理者は **NSSProtocol** を、SSL Global Context セクションに**のみ**指定するか、ドメインごとの**すべて**の VirtualHost セクションで指定する必要があります。

mod_nss での TLS 1 以上を除くすべての SSL および TLS プロトコルの無効化

TLS バージョン 1 以上を除く SSL および TLS プロトコルバージョンをすべて無効にするには、以下の手順を実行します。

1. **root** で **/etc/httpd/conf.d/nss.conf** ファイルを開き、NSSProtocol ディレクティブの すべてのインスタンスを検索します。デフォルトでは、設定ファイルに以下のような 1 つのセクションが含まれます。

```
~]# vi /etc/httpd/conf.d/nss.conf
#  SSL Protocol:
output omitted
#  Since all protocol ranges are completely inclusive, and no protocol in the
#  middle of a range may be excluded, the entry "NSSProtocol SSLv3,TLSv1.1"
#  is identical to the entry "NSSProtocol SSLv3,TLSv1.0,TLSv1.1".
NSSProtocol SSLv3,TLSv1.0,TLSv1.1
```

   このセクションは、VirtualHost セクション内にあります。

2. **NSSProtocol** 行を以下のように編集します。

```
#  SSL Protocol:
NSSProtocol TLSv1.0,TLSv1.1
```

   すべての VirtualHost セクションに対してこのアクションを繰り返します。

3. **Listen 8443** 行を以下のように編集します。

```
Listen 443
```

4. デフォルトの **VirtualHost default:8443** 行を以下のように編集します。

```
VirtualHost default:443
```

   デフォルト以外の他のすべての仮想ホストセクション (存在する場合) を編集します。ファイルを保存してから閉じます。

5. すべての **NSSProtocol** ディレクティブが以下のように変更したことを確認します。

```
~]# grep NSSProtocol /etc/httpd/conf.d/nss.conf
#  middle of a range may be excluded, the entry "NSSProtocol SSLv3,TLSv1.1"
#  is identical to the entry "NSSProtocol SSLv3,TLSv1.0,TLSv1.1".
NSSProtocol TLSv1.0,TLSv1.1
```

この手順は、VirtualHost セクションが複数の場合に特に重要になります。

6. 以下のように Apache デーモンを再起動します。

```
~]# service httpd restart
```

セッションが中断されることに注意してください。

mod_nss での SSL および TLS プロトコルのステータスのテスト

mod_nss で有効または無効な SSL および TLS のバージョンを確認するには、**openssl s_client -connect** コマンドを使用します。**root** として **openssl** パッケージをインストールします。

```
~]# yum install openssl
```

openssl s_client -connect コマンドの形式は次のとおりです。

```
openssl s_client -connect hostname:port -protocol
```

port は、テストするポートで、**protocol** は、テストするプロトコルバージョンです。ローカルで稼働している SSL サーバーをテストする場合は、**localhost** をホスト名として使用します。たとえば、SSLv3 が有効であるかどうかを確認するためにセキュアな HTTPS 接続のデフォルトポートであるポート **443** をテストするには、以下のようにコマンドを発行します。

```
~]# openssl s_client -connect localhost:443 -ssl3
CONNECTED(00000003)
3077773036:error:1408F10B:SSL routines:SSL3_GET_RECORD:wrong version
number:s3_pkt.c:337:
output omitted
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
SSL-Session:
  Protocol : SSLv3
output truncated
```

上記の出力は、ハンドシェイクが失敗し、暗号化がネゴシエートされなかったことを示しています。

```
~]$ openssl s_client -connect localhost:443 -tls1
CONNECTED(00000003)
depth=1 C = US, O = example.com, CN = Certificate Shack
output omitted
New, TLSv1/SSLv3, Cipher is AES128-SHA
Server public key is 1024 bit
Secure Renegotiation IS supported
Compression: NONE
```

```
Expansion: NONE
SSL-Session:
  Protocol : TLSv1
output truncated
```

上記の出力は、ハンドシェイクが失敗せず、暗号化セットがネゴシエートされたことを示しています。

openssl s_client コマンドのオプションは、**s_client(1)** man ページに記載されています。

SSLv3 の脆弱性とそのテスト方法の詳細は、Red Hat ナレッジベースの記事 POODLE: SSLv3 脆弱性 (CVE-2014-3566) を参照してください。

14.1.10. 既存の鍵および証明書の使用

以前に作成した鍵と証明書がある場合は、新しいファイルを生成する代わりに、SSL サーバーを設定してこのファイルを使用できます。これが可能ではない状況は 2 つしかありません。

1. **IP アドレスまたはドメイン名を変更している。**
   証明書が、特定の IP アドレスとドメイン名のペアに対して発行されます。この値のいずれかが変更すると、証明書が無効になります。

2. **VeriSign からの証明書があり、サーバーソフトウェアを変更している。**
   Verivsftpd は、幅広く使用されている認証局で、特定のソフトウェア製品、IP アドレス、およびドメイン名の証明書を発行します。ソフトウェア製品を変更すると、証明書が無効になります。

上記のいずれの場合も、新しい証明書を入手する必要があります。このトピックの詳細については、「新しい鍵と証明書の生成」 を参照してください。

既存の鍵と証明書を使用する場合は、その関連ファイルを **/etc/pki/tls/private/** ディレクトリーと **/etc/pki/tls/certs/** ディレクトリーにそれぞれ移動します。これを行うには **root** で以下のコマンドを入力します。

```
~]# mv key_file.key /etc/pki/tls/private/hostname.key
~]# mv certificate.crt /etc/pki/tls/certs/hostname.crt
```

次に、以下の行を **/etc/httpd/conf.d/ssl.conf** 設定ファイルに追加します。

```
SSLCertificateFile /etc/pki/tls/certs/hostname.crt
SSLCertificateKeyFile /etc/pki/tls/private/hostname.key
```

更新した設定を読み込むには、in 「サービスの再起動」 の説明に従って、**httpd** サービスを再起動します。

例14.5 Red Hat Secure Web Server からの鍵と証明書の使用

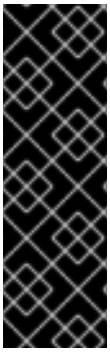
```
~]# mv /etc/httpd/conf/httpsd.key /etc/pki/tls/private/penguin.example.com.key
~]# mv /etc/httpd/conf/httpsd.crt /etc/pki/tls/certs/penguin.example.com.crt
```

14.1.11. 新しい鍵と証明書の生成

新しい鍵と証明書のペアを生成するには、システムに **crypto-utils** パッケージをインストールしておく必要があります。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入力します。

```
~]# yum install crypto-utils
```

このパッケージには、SSL 証明書と秘密鍵を生成して管理するツールセットを提供し、鍵を生成できる Red Hat Keypair Generation ユーティリティーである ***genkey*** *が含まれます。*



重要

有効な証明書を所有していて、それを新規のものに置き換える予定の場合は、異なるシリアル番号を指定します。これにより、クライアントのブラウザーがこの変更の通知を受けて予定どおりに新規の証明書に更新して、このページへのアクセスに失敗しないようにします。 ***root*** *でカスタムのシリアル番号で新規の証明書を作成するには、* ***genkey*** *の代わりに、以下のコマンドを使用します。*

```
~]# openssl req -x509 -new -set_serial number -key hostname.key -out hostname.crt
```

注記

使用中のシステムに特定のホスト名用の鍵ファイルがすでに存在すると、 ***genkey*** *は起動しません。その場合は、* ***root*** *で以下のコマンドを使用して既存のファイルを削除します。*

```
~]# rm /etc/pki/tls/private/hostname.key
```

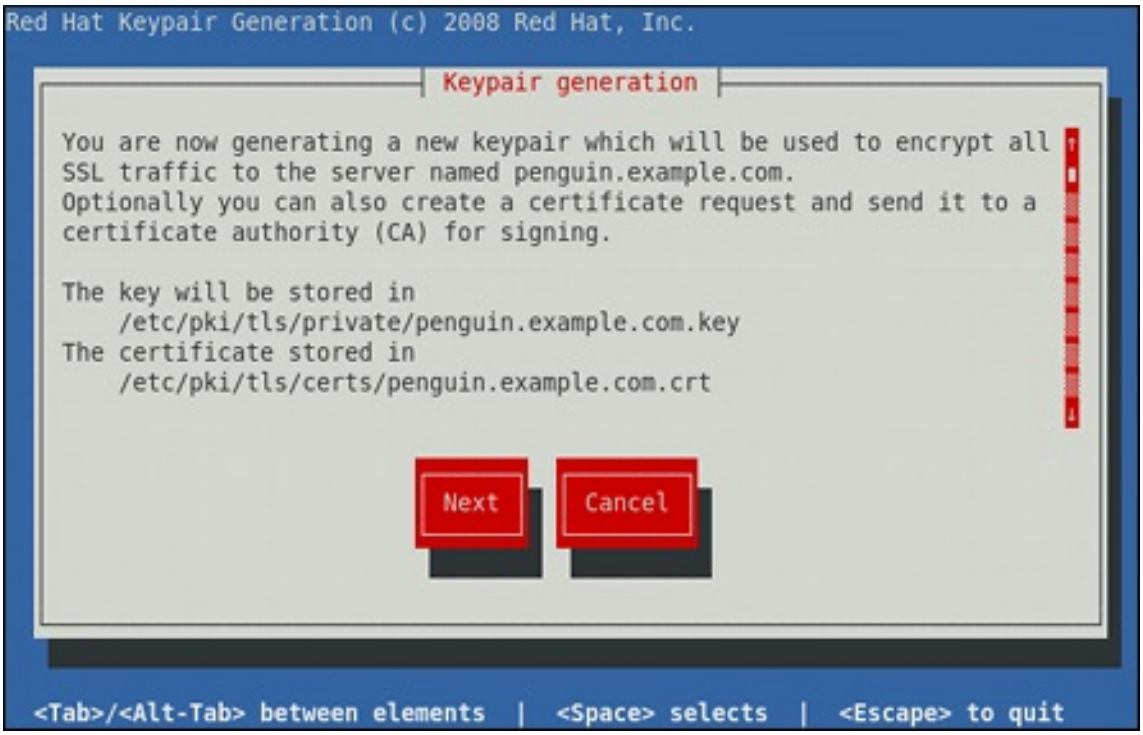
ユーティリティーを実行するには、 ***root*** *で* ***genkey*** *コマンドの後に該当するホスト名 (たとえば、* ***penguin.example.com*** *など) を付けて使用します。*

```
~]# genkey hostname
```

鍵と証明書の生成を完了するには、以下の手順を行います。

1. *鍵と証明書を保存する場所を確認します。*

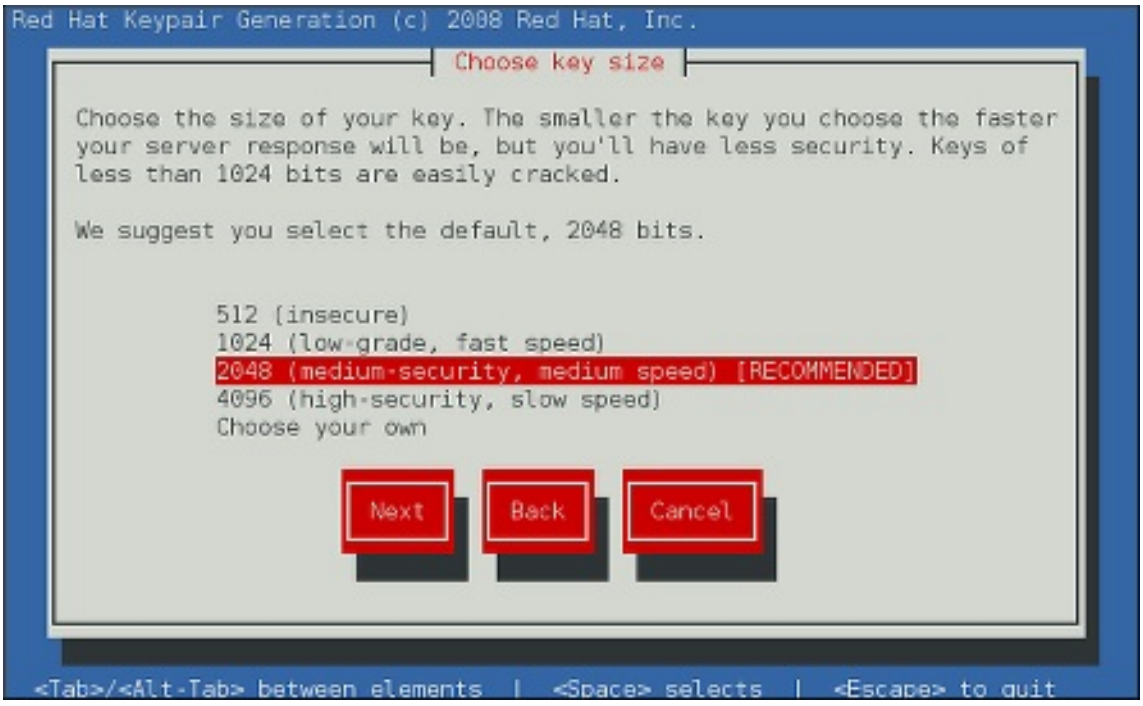
図14.1 genkey ユーティリティーの実行



__Tab__ キーを使用して __Next (次へ)__ ボタンを選択してから、__Enter__ キーを押すと次の画面を進みます。

2. *__up__ と __down__ の矢印キーを使用して、適切な鍵のサイズを選択します。鍵が大きくなればセキュリティーは向上しますが、サーバーの応答時間も長くなることに注意してください。NIST では、__2048 bits__ の使用が推奨されています。NIST Special Publication 800-131A を参照してください。*

図14.2 鍵のサイズ選択



終了したら、タブ キーを使用して 次へ ボタンを選択し、__Enter__ キーを押すと、ランダムなビットの生成プロセスが開始します。選択した鍵のサイズによっては、時間がかかることもあります。

3. *証明書要求を認証局に送信するかどうかを決定します。*

図14.3 証明書要求の生成

*タブ キーを使用して **Yes** (はい) を選択し、証明書要求を組み立てるか、**No** (いいえ) を選択して、自己署名の証明書を生成します。その後、**Enter** キーを押して、選択を確認します。*

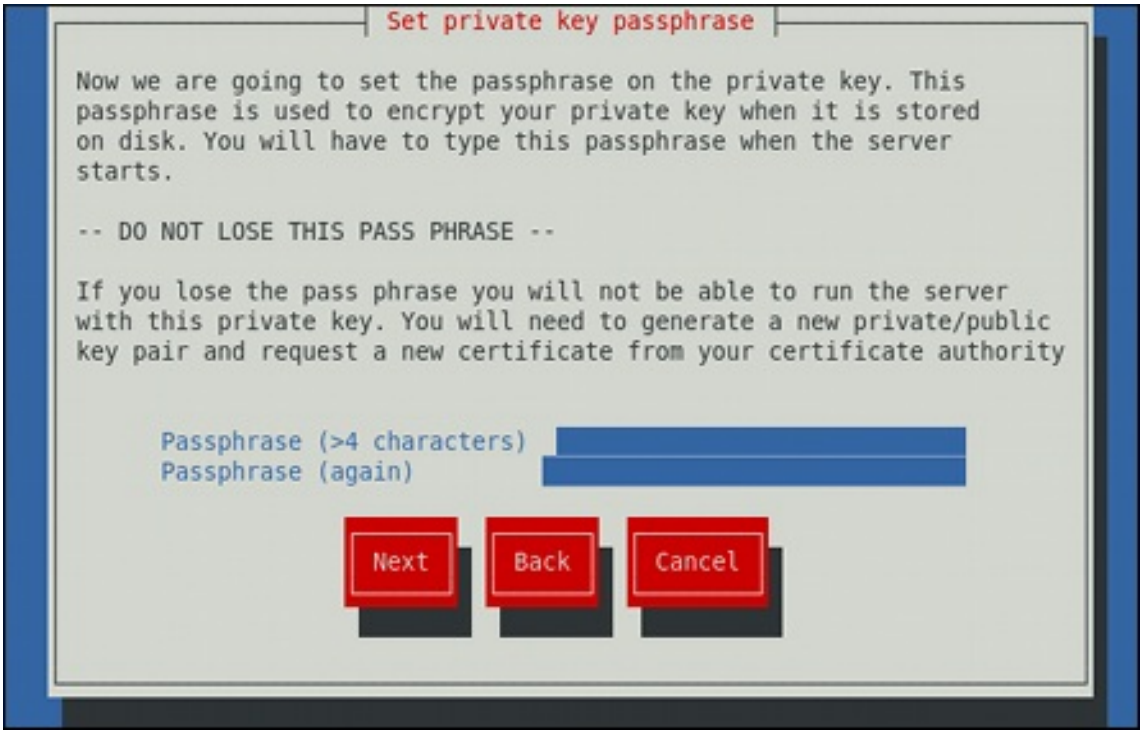
4. ***Spacebar** (スペースバー) キーを使用すると、秘密鍵の暗号化を有効にする (**[*]**) か、無効にする (**[ ]**) 選択ができます。*

図14.4 秘密鍵の暗号化

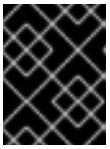
***Tab** キーを使用して **Next** (次へ) ボタンを選択してから、**Enter** キーを押すと次の画面を進みます。*

5. *秘密鍵の暗号化を有効にしている場合は、適切なパスフレーズを入力します。セキュリティーの理由により入力時には文字が表示されませんが、最低でも 5 文字の長さが必要です。*

   図14.5 パスフレーズの入力



   ***Tab** キーを使用して **Next** (次へ) ボタンを選択してから、**Enter** キーを押すと次の画面を進みます。*

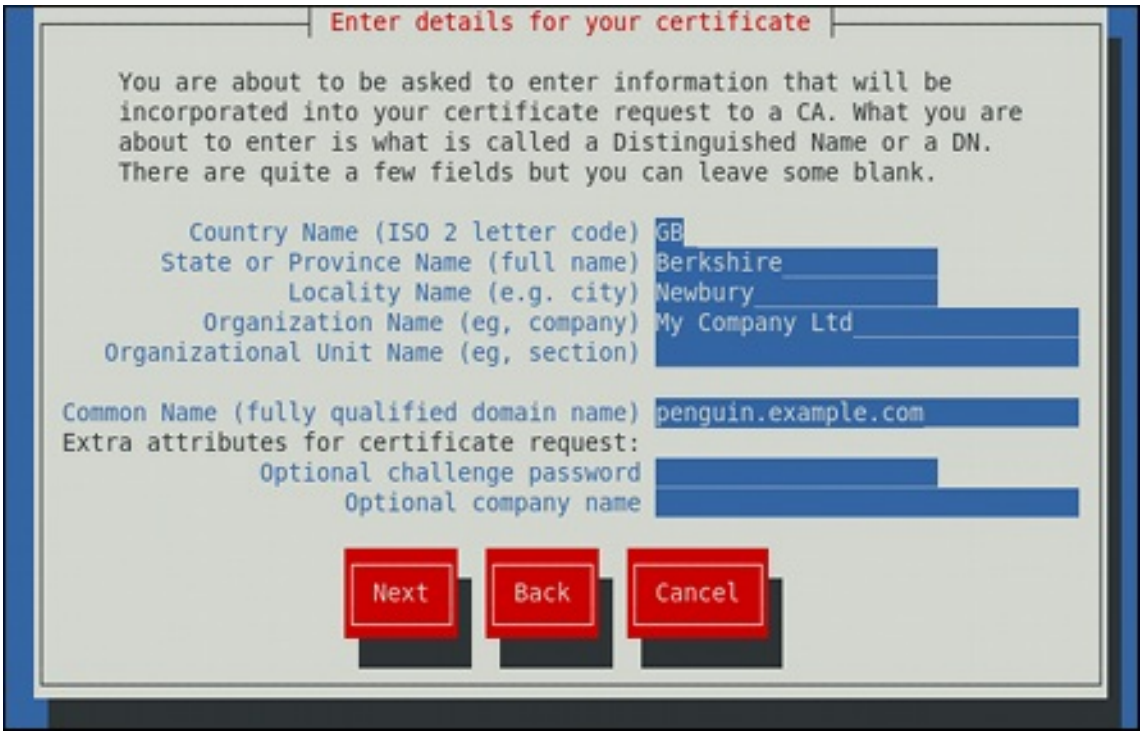


   重要

   サーバーを起動するには正しいパスフレーズの入力が必要です。それを紛失したり忘れたりした場合は、新しい鍵と証明書を生成する必要があります。

6. *証明書詳細のカスタマイズ*

図14.6 証明書情報の指定



*タブ キーを使用して **Next** ボタンを選択し、**Enter** を押すと鍵の生成が完了します。*

7. *証明書要求の生成を有効にしていた場合は、それを認証局に送信するように求められます。*

図14.7 証明書要求を送信する方法の指示

```
You now need to submit your CSR and documentation to your certificate
authority. Submitting your CSR may involve pasting it into an online
web form, or mailing it to a specific address. In either case, you
should include the BEGIN and END lines.

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBqjCCARMCAQAwajELMAkGA1UEBhMCR0IxEjAQBgNVBAgTCUJlcmtzaGlyZTEQ
MA4GA1UEBxMHTmV3YnVyeTEXMBUGA1UEChMOTXkgQ29tcGFueSBMdGQxHDAaBgNV
BAMTE3Blbmd1aW4uZXhhbXBsZS5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJ
AoGBAJjw8bXq7WKGGXNZsNZltEe9849wUMc4uAh+X8251b8x+ptJQCanGeNhLlXU
xiL5srY2TjoTSQ5DvyFgPQmFFe3cn7v//bKNgNqd4h0EbRFGaj/hDUG3fXnjujkX
hP+9iY/eIAQZlHQSkABh/2egtIllpfDeRvsTUX376TnkIWLhAgMBAAGgADANBgkq
hkiG9w0BAQQFAAOBgQBUTjgjcnts1hZKO70c5j+b4IfsBCwm4lnvGx3jOwpLdRq/
rHpx5cbHV99vcKnF3CwDrze9DgpTdjdbAccSCVgSG5GE8JZXWYD8EK8p2naJNQl1
YVX1KPi5MPLZuZ9cTb+k4K0cbug0IQiYaKNLNI/0zLE1VEWZXYFXOUBFM2gXYw==
-----END NEW CERTIFICATE REQUEST-----


A copy of this CSR has been saved in the file
/etc/pki/tls/certs/penguin.example.com.1.csr

Press return when ready to continue
```

***Enter** を押してシェルプロンプトに戻ります。*

*生成が終了したら、鍵と証明書の場所を **/etc/httpd/conf.d/ssl.conf** 設定ファイルに追加します。*

```
SSLCertificateFile /etc/pki/tls/certs/hostname.crt
SSLCertificateKeyFile /etc/pki/tls/private/hostname.key
```

最後に、「サービスの再起動」 に記載通りに **httpd** サービスを再起動すると、更新された設定が読み込まれます。

14.1.12. コマンドラインを使用して HTTP 用および HTTPS 用にファイアウォールを設定

Red Hat Enterprise Linux は、デフォルトで **HTTP** および **HTTPS** トラフィックを許可しません。システムが Web サーバーとして機能するようにするには、**firewalld** がサポートするサービスを使用して、必要に応じて **HTTP** および **HTTPS** のトラフィックがファイアウォールを通過するようにします。

コマンドラインで **HTTP** を有効にするには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --add-service http
 success
```

コマンドラインで **HTTPS** を有効にするには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --add-service https
 success
```

ここで変更する内容は、システムの再起動後は維持されないことに注意してください。ファイアウォールを永続的に変更するには、コマンドに **--permanent** オプションを繰り返し追加してください。

14.1.12.1. コマンドラインで着信 HTTPS および HTTPS のネットワークアクセスの確認

ファイアウォールが許可するサービスを確認するには、**root** で以下のコマンドを実行します。

```
~]# firewall-cmd --list-all
public (default, active)
 interfaces: em1
 sources:
 services: dhcpv6-client ssh
output truncated
```

この例では、デフォルトのインストールでファイアウォールは有効になっていますが、**HTTP** と **HTTPS** は通過できません。

HTTP および **HTTP** のファイアウォールサービスが有効になると、**services** 行は以下のようになります。

```
services: dhcpv6-client http https ssh
```

ファイアウォールサービスの有効化、または **firewalld** でのポートの開閉方法は Red Hat Enterprise Linux 7 セキュリティーガイド を参照してください。

14.1.13. 関連情報

Apache HTTP Server の詳細は、以下のリソースを参照してください。

インストールされているドキュメント

- **httpd(8)**: **httpd** サービスの man ページで、コマンドラインオプションのリストが記載されます。

- ***genkey(1)***: *crypto-utils パッケージにより提供される* ***genkey*** *ユーティリティーの man ページです。*

- ***apachectl (8)***: *Apache HTTP Server の制御インターフェイスの man ページです。*

インストール可能なドキュメント

- **http://localhost/manual/**: *Apache HTTP Server の公式ドキュメントで、そのディレクティブおよび利用可能なモジュールの詳細を説明します。本書にアクセスするには、* ***httpd-manual*** *パッケージがインストールされ、Web サーバーが稼働している必要があることに注意してください。*
  ドキュメントにアクセスする前に、 ***root*** *で次のコマンドを実行します。*

  ```
  ~] yum install httpd-manual ~] apachectl graceful
  ```

オンラインドキュメント

- *http://httpd.apache.org/: Apache HTTP Server の公式 Web サイトです。すべてのディレクティブおよびデフォルトモジュールの説明が記載されています。*

- *OpenSSL Home Page - その他のドキュメント、FAQ、メーリングリストへのリンクなどの役に立つリソースを掲載した OpenSSL のホームページです。*

第15章 メールサーバー

Red Hat Enterprise Linux は、メールを提供し、アクセスするための高度なアプリケーションを多数提供します。本章では、現在使用されている最新の電子メールプロトコルと電子メールを送受信するプログラムについて説明します。

15.1. メールプロトコル

今日、電子メールはクライアント/サーバーのアーキテクチャーを使用して配信されています。電子メールのメッセージは、メールクライアントプログラムを使用して作成されます。次に、このプログラムがメッセージをサーバーに送信します。メッセージは、サーバーが受信者の電子メールサーバーに転送し、そこで受信者の電子メールクライアントに渡されます。

このプロセスを有効にするために、各種の標準のネットワークプロトコルが異なるマシンによる (多くの場合、異なるオペレーティングシステムで、異なる電子メールプログラムを使用) 電子メールの送受信を可能にしています。

以下は、電子メールの転送に最も一般的に使用されているプロトコルです。

15.1.1. メール転送プロトコル

クライアントアプリケーションからサーバーへのメール配信、および送信元サーバーから転送先サーバーへのメール配信は、**SMTP (簡易メール転送プロトコル)** により処理されます。

15.1.1.1. SMTP

SMTP の第一の目的は、メールサーバー間における電子メールの転送です。ただし、これは、メールクライアントにも重要です。メールを送信するには、クライアントが送信メールサーバーにメッセージを送信し、配信先メールサーバーに接続します。ただし、このチェーンにはさらに多くの中間 SMT P サーバーが含まれる場合があります。この概念はメールリレーと呼ばれます。このため、メールクライアントの設定時に SMTP サーバーを指定する必要があります。

Red Hat Enterprise Linux では、ユーザーはローカルマシンで SMTP サーバーを設定してメール配信を処理できます。ただし、送信メール用にリモート SMTP サーバーを設定することも可能です。

SMTP プロトコルに関して重要なのは認証が不要である点です。これにより、インターネット上の誰でも、個人や大規模なグループに対してでも電子メールを送信できます。迷惑メールや **スパム** が可能になるのは SMTP のこうした特性が原因です。リレー制限を課すと、インターネット上の任意のユーザーが、ご使用の SMTP サーバーを介してインターネット上の別のサーバーへ電子メールを送信することが制限されます。リレー制限を課さないサーバーは、**オープンリレー** サーバーと呼ばれます。

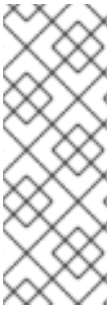
Red Hat Enterprise Linux 7 は、Postfix および Sendmail SMTP プログラムを提供します。

15.1.2. メールアクセスプロトコル

メールサーバーから電子メールを取得するために、電子メールクライアントアプリケーションが使用する主要なプロトコルには、**POP (ポストオフィスプロトコル)** と **IMAP (インターネットメッセージアクセスプロトコル)** の 2 つがあります。

15.1.2.1. POP

Red Hat Enterprise Linux のデフォルトの POP サーバーは **Dovecot** で、**dovecot** パッケージで提供されます。



注記

Dovecot をインストールするには、以下のコマンドを実行します。

```
~]# yum install dovecot
```

yum を使用したパッケージのインストールは 「パッケージのインストール」 を参照してください。

POP サーバーを使用する場合、電子メールメッセージは電子メールクライアントのアプリケーションがダウンロードします。デフォルトでは、ほとんどの POP 電子メールクライアントでは、電子メールサーバーのメッセージが正しく転送されるとそのメッセージは削除されるように自動的に設定されています。ただし、この設定は通常は変更できます。

POP は、電子メールのファイル添付を可能にする MIME (多目的インターネットメール拡張) *などの重要なインターネットメッセージング標準と完全な互換性があります。*

POP は、電子メールを読むためのシステムが 1 つであるユーザーの場合に最適に機能します。また、インターネットやメールサーバーを持つネットワークに常時接続していないユーザーにもうまく機能します。ネットワーク速度が遅いユーザーの場合は、POP はクライアントプログラムに対して、認証を行った上で各メッセージのコンテンツ全体をダウンロードするよう要求します。このプロセスは、メッセージに大きなファイルが添付されている場合に長時間かかる場合があります。

標準 POP プロトコルの最新版は POP3 です。

ただし、あまり使用されていない POP プロトコルのバリアントにも様々な種類があります。

- APOP: *MD5 認証を使用した POP3 です。暗号化されていないパスワードを送信するのではなく、エンコードされたユーザーパスワードのハッシュが電子メールクライアントからサーバーへ送信されます。*
- KPOP: *Kerberos 認証を使用した POP3 です。*
- RPOP: *RPOP 認証を使用した POP3 です。これは、パスワードに似たユーザーごとの ID を使用し、POP 要求を認証します。ただしこの ID は暗号化されていないため、RPOP のセキュリティーレベルは標準 POP と同程度です。*

セキュリティーを改善するために、クライアント認証およびデータ転送セッションに Secure Socket Layer (SSL) *暗号化を使用できます。SSL 暗号化を有効にするには、以下を使用します。*

- *pop3s サービス*
- *stunnel アプリケーション*
- *starttls コマンド*

メール通信のセキュリティー保護に関する詳細は、 「通信のセキュリティー保護」 を参照してください。

15.1.2.2. IMAP

Red Hat Enterprise Linux のデフォルトの IMAP サーバーは Dovecot で、dovecot パッケージで提供されます。Dovecot のインストール方法は 「POP」 を参照してください。

IMAP メールサーバーを使用する場合は電子メールメッセージはサーバーに残るため、ユーザーはメッセージの読み取り、または削除を行うことができます。また、IMAP により、クライアントアプリケー

ションがサーバー上でメールディレクトリーの作成、名前変更、削除を行い電子メールを整理、保存することもできます。

IMAP は複数のマシンを使用して電子メールにアクセスするユーザーに特に役立ちます。このプロトコルでは、メッセージが開封されるまでは、電子メールのヘッダー情報しかダウンロードされず帯域幅を節減できるため、低速な接続でメールサーバーに接続するユーザーにも便利です。ユーザーは、メッセージを表示またはダウンロードすることなく削除することも可能です。

便宜上、**IMAP** クライアントアプリケーションは、メッセージのコピーをローカルでキャッシュすることが可能です。そのため、ユーザーは **IMAP** サーバーに直接接続していない時でも、既読メッセージを閲覧することができます。

IMAP は **POP** と同様に、電子メールのファイル添付を可能にする MIME などの重要なインターネットメッセージング標準と完全に互換性があります。

セキュリティーを強化するには、**SSL** 暗号化をクライアント認証とデータ転送セッションに使用することができます。これは、**imaps** サービスまたは **stunnel** プログラムを使用して有効にできます。

- **pop3s** サービス
- **stunnel** アプリケーション
- **starttls** コマンド

メール通信のセキュリティー保護に関する詳細は、「通信のセキュリティー保護」 を参照してください。

無償や商用の IMAP クライアントおよびサーバーは他にも提供されています。これらの多くは、IMAP プロトコルを拡張し、追加機能を提供します。

15.1.2.3. Dovecot

IMAP および **POP3** プロトコルを実装する **imap-login** プロセスと **pop3-login** プロセスは、dovecot パッケージに含まれているマスターの **dovecot** デーモンが生成します。**IMAP** および **POP** の使用は、**/etc/dovecot/dovecot.conf** 設定ファイルで設定されます。デフォルトでは、**dovecot** は、**SSL** を使用するセキュアなバージョンとともに **IMAP** および **POP3** を実行します。**POP** を使用するように **dovecot** を設定するには、以下の手順を実行します。

1. **protocols** 変数がコメント解除されていて (行頭のハッシュ記号 (**#**) を削除)、**pop3** 引数を含むよう **/etc/dovecot/dovecot.conf** 設定ファイルを編集します。以下に例を示します。

   ```
   protocols = imap pop3 lmtp
   ```

   protocols 変数がコメントアウトされている場合、**dovecot** は上記のようにデフォルト値を使用します。

2. **root** で以下のコマンドを実行して、現行セッションで変更を可能にします。

   ```
   ~]# systemctl restart dovecot
   ```

3. この変更を次回の再起動後に有効にするには、以下のコマンドを実行します。

   ```
   ~]# systemctl enable dovecot
   Created symlink from /etc/systemd/system/multi-user.target.wants/dovecot.service to
   /usr/lib/systemd/system/dovecot.service.
   ```

注記

dovecot が報告するのは **IMAP** サーバーを起動したことだけですが、**POP3** サーバーも起動する点に注意してください。

SMTP とは異なり、**IMAP** と **POP3** は、接続するクライアントを、ユーザー名とパスワードを使用して認証する必要があります。デフォルトでは、両方のプロトコルのパスワードは、暗号化されていないネットワーク上で渡されます。

dovecot で **SSL** を設定するには、以下を実行します。

- **/etc/dovecot/conf.d/10-ssl.conf** 設定を編集して、**ssl_protocols** 変数がコメント解除されていて、**!SSLv2 !SSLv3** 変数を含めるようにします。

```
ssl_protocols = !SSLv2 !SSLv3
```

これらの値により、**dovecot** は、安全でないことがわかっている SSL バージョン 2 および 3 を回避するようになります。これは POODLE: SSLv3 脆弱性 (CVE-2014-3566) で説明されている脆弱性が原因です。詳細は、Postfix および Dovecot における POODLE SSL 3.0 脆弱性問題 (CVE-2014-3566) の解決方法 を参照してください。

/etc/dovecot/conf.d/10-ssl.conf には、以下のオプションが含まれます。

```
ssl=required
```

- **/etc/pki/dovecot/dovecot-openssl.cnf** 設定ファイルを必要に応じて編集します。ただし、標準的なインストールではこのファイルへの変更は必要ありません。
- **/etc/pki/dovecot/certs/dovecot.pem** ファイルおよび **/etc/pki/dovecot/private/dovecot.pem** ファイルの名前変更、移動、削除を行います。
- **/usr/libexec/dovecot/mkcert.sh** のスクリプトを実行して、**dovecot** の自己署名証明書を作成します。証明書は **/etc/pki/dovecot/certs** および **/etc/pki/dovecot/private** ディレクトリーにコピーされます。変更を実装にするには、**root** で以下のコマンドを実行して **dovecot** を再起動します。

```
~]# systemctl restart dovecot
```

dovecot の詳細は http://www.dovecot.org でオンラインで参照できます。

15.2. 電子メールプログラムの分類

一般的に、すべての電子メールアプリケーションは 3 つのタイプのうち 1 つ以上に分類されます。それぞれの分類は、電子メールメッセージの移動および管理のプロセスにおいてそれぞれ特定のロールを果たします。大半のユーザーはメッセージの送受信に使用する特定の電子メールプログラムだけを認識しますが、電子メールを正しい送信先に届けるにはすべての電子メールプログラムが重要になります。

15.2.1. メール転送エージェント (Mail Transport Agent)

MTA (**メール転送エージェント**) は、**SMTP** を使用してホスト間で電子メールメッセージを転送します。メッセージは目的の送信先に移動する時、様々な MTA に関わることがあります。

マシン間のメッセージ配信は簡単に見えるかもしれませんが、配信のためにある MTA がメッセージを受け入れることが可能か、受け入れるべきかを判断する過程全体は非常に複雑です。その上、スパムの

問題により、特定の MTA の使用は通常 MTA の設定または MTA が常駐するネットワークのアクセス設定により制限されます。

メールクライアントプログラムの中には、メール送信時に MTA として機能するものもあります。しかし、このようなメールクライアントプログラムは、アウトバウンドメッセージを MTA のみに送信することができるため、このようなメールクライアントプログラムは、メッセージを目的の受信者のメールサーバーに直接送信することはできません。この機能は、アプリケーションを実行しているホストに独自の MTA がない場合に便利です。

Red Hat Enterprise Linux では **Postfix** と **Sendmail** の 2 つの MTA を提供しています。これらメールクライアントプログラムは MTA として動作する必要はありません。Red Hat Enterprise Linux には、**Fetchmail** と呼ばれる特別な目的の MTA も含まれています。

Postfix、Sendmail、Fetchmail の詳細は 「メール転送エージェント (MTA)」 を参照してください。

15.2.2. メール配信エージェント (MDA)

MDA (メール配信エージェント) は MTA により呼び出され、適切なユーザーのメールボックスに受信メールをファイル保存します。多くの場合、MDA は実際には **mail** や Procmail などの **LDA (ローカル配信エージェント)** です。

電子メールクライアントアプリケーションが読み取り可能なポイントに配信されるメッセージを実際に処理するプログラムは、いずれも MDA と見なすことができます。このため、一部の MTA (Sendmail、Postfix など) は、ローカルユーザーのメールスプールファイルに新規の電子メールメッセージを追加する時に、MDA のロールを果たすことができます。通常、MDA はシステム間でのメッセージの転送やユーザーインターフェイスの提供は行いません。MDA は、ローカルマシン上でメッセージの配信と並べ替えを行い、電子メールクライアントアプリケーションがアクセスできるようにします。

15.2.3. メールユーザーエージェント

Mail User Agent (MUA) は、電子メールクライアントアプリケーションと同義語です。MUA は、最低限、電子メールメッセージを読んで作成するプログラムです。MUA は以下のタスクを処理できます。

- ***POP*** または ***IMAP*** プロトコルを使用したメッセージの取得
- メッセージを保存するメールボックスを設定します。
- MTA に出力メッセージを送信します。

MUA は、***Thunderbird***、***Evolution*** のようなグラフィカルな場合と、***mail*** または ***Mutt*** などの単純なテキストベースインターフェイスになります。

15.3. メール転送エージェント (MTA)

Red Hat Enterprise Linux 7 には、Postfix と Sendmail の 2 つの主要 MTA が装備されています。Postfix がデフォルトの MTA として設定されており、Sendmail は非推奨となっています。デフォルトの MTA を Sendmail に変更する必要がある場合は、Postfix をアンインストールするか、***root*** で次のコマンドを使用して Sendmail に切り替えます。

```
~]# alternatives --config mta
```

以下のコマンドを使用して、希望のサービスを有効にすることもできます。

```
~]# systemctl enable service
```

同様に、サービスを無効にするには、シェルプロンプトで以下を入力します。

```
~]# systemctl disable service
```

Red Hat Enterprise Linux 7 でシステムサービスを管理する方法は、10章systemd によるサービス管理 を参照してください。

15.3.1. postfix

当初、IBM のセキュリティーエキスパートであるプログラマーの Wietse Venema 氏によって開発された Postfix は、Sendmail 互換の MTA で、セキュア、高速、かつ容易に設定できるように設計されています。

セキュリティー向上のために、Postfix ではモジュラー型設計を採用しており、権限が限定された小さなプロセスは、**マスター** デーモンが起動します。より小さく、権限の低いプロセスは、メール配信の様々な段階に関連する非常に特殊なタスクを実行してルートディレクトリーが変更された環境で稼働し、攻撃の影響を制限します。

Postfix がローカルコンピューター以外のホストからのネットワーク接続を受け入れるよう設定するには、設定ファイルを多少変更するだけでできます。さらに、より複雑なニーズのために、Postfix は様々な設定オプションだけでなくサードパーティーのアドオンも提供するため、多用途でフル機能の MTA となっています。

Postfix の設定ファイルは人間に解読可能で、250 以上のディレクティブに対応しています。Sendmail とは異なり、変更を反映するためにマクロ処理は必要なく、また最も一般的に使用されるオプションの大部分は、多数のコメントが付いたファイルで説明されています。

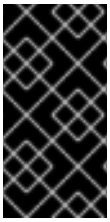
15.3.1.1. Postfix のデフォルトインストール

Postfix 実行可能ファイルは **postfix** です。このデーモンは、メール配信の処理に必要なすべての関連プロセスを起動します。

Postfix は設定ファイルを **/etc/postfix/** ディレクトリーに格納します。以下は、一般的に使用されるその他のファイルのリストです。

- ***access***: アクセス制御に使用します。このファイルは、Postfix に接続可能なホストを指定します。
- ***main.cf***: グローバル Postfix 設定ファイル設定オプションの大部分がこのファイルで指定されています。
- ***master.cf***: メール配信を完了するために Postfix が様々なプロセスとやりとりを行う方法を指定します。
- ***transport***: 電子メールアドレスをリレーホストにマッピングします。

aliases ファイルは **/etc** ディレクトリーにあります。このファイルは Postfix と Sendmail 間で共有されます。ユーザー ID エイリアスを記述するメールプロトコルが必要な設定可能なリストです。



重要

デフォルトの **/etc/postfix/main.cf** ファイルでは、Postfix はローカルコンピューター以外のホストからのネットワーク接続を受け付けられないように設定されています。Postfix を他のクライアント用のサーバーとして設定する方法は 「Postfix の基本設定」 を参照してください。

/etc/postfix/ ディレクトリーにある設定ファイルのオプションに変更を加えた後には、変更を反映させるために **postfix** サービスを再起動してください。これを行うには、**root** で以下のコマンドを実行します。

```
~]# systemctl restart postfix
```

15.3.1.2. 以前のリリースからのアップグレード

Red Hat Enterprise Linux 7 の設定は、以前のリリースとは異なります。

- **disable_vrfy_command = no**: Sendmail のデフォルトとは異なり、これはデフォルトで無効になります。**yes** に変更した場合は、電子メールアドレスを収集する一部の方法を回避できます。
- **allow_percent_hack = yes**: これはデフォルトで有効になります。これにより、電子メールの **%** 文字を削除できるようになります。パーセント記号を使用したハックは、電子メールを送信者が制御してルーティングできるようにする古い回避策です。**DNS** と電子メールのルーティングの信頼性は非常に高まりましたが、Postfix はこのハックを引き続きサポートします。パーセント記号の書き換えを無効にするには、**allow_percent_hack** を **no** に設定します。
- **smtpd_helo_required = no**: 一部のアプリケーションが電子メールを送信できなくなることがあるため、Sendmail の場合と同様に、これはデフォルトで無効になります。MAIL、FROM、または ETRN コマンドを送信する前にクライアントが HELO または EHLO コマンドを送信するようにするには、**yes** に変更します。

15.3.1.3. Postfix の基本設定

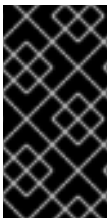
デフォルトでは、Postfix はローカルホスト以外のホストからのネットワーク接続を受け付けません。ネットワーク上の他のホストを対象としたメール配信を有効にするには、**root** で以下のステップを実行します。

- **vi** などのテキストエディターで **/etc/postfix/main.cf** ファイルを編集します。
- **mydomain** 行のハッシュ記号 (**#**) を削除してコメント解除してから、domain.tld の箇所を **example.com** などのメールサーバーがサービスを提供しているドメインに置き換えます。
- **myorigin = $mydomain** 行のコメントを解除します。
- **myhostname** 行のコメントを解除し、host.domain.tld をマシンのホスト名に置き換えます。
- **mydestination = $myhostname, localhost.$mydomain** 行のコメントを解除します。
- **mynetworks** 行のコメントを解除して、168.100.189.0/28 の箇所を、サーバーに接続可能なホスト用の有効なネットワーク設定に置き換えます。
- **inet_interfaces = all** 行のコメントを解除します。
- **inet_interfaces = localhost** をコメント化します。
- **postfix** サービスを再起動します。

これらの手順が完了したら、ホストは配信のため外部の電子メールを受け入れるようになります。

Postfix には様々な設定オプションがあります。Postfix の設定方法を学習する最適な方法の 1 つは、**/etc/postfix/main.cf** 設定ファイルのコメントを読むことです。Postfix 設定、SpamAssassin 統合、**/etc/postfix/main.cf** パラメーターの詳細などの補足情報は http://www.postfix.org/ で参照できま

す。



重要

POODLE: SSLv3 脆弱性 (CVE-2014-3566) で説明されている脆弱性のため、Red Hat は、**SSL** を無効にし、**TLSv1.1** または **TLSv1.2** のみを使用することを推奨します。詳細は、Postfix および Dovecot における POODLE SSL 3.0 脆弱性問題 (CVE-2014-3566) の解決方法 を参照してください。

15.3.1.4. LDAP での Postfix の使用

Postfix は **LDAP** ディレクトリーを様々なルックアップテーブルのソースとして利用できます (たとえば、**aliases**、**virtual**、**canonical** など)。これにより **LDAP** は階層的なユーザー情報を保存でき、Postfix は **LDAP** クエリーの結果を必要な場合にのみ知らされます。この情報をローカルに保存しないことで、管理者は容易に管理することができます。

15.3.1.4.1. /etc/aliases ルックアップのサンプル

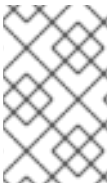
以下は **/etc/aliases** ファイルをルックアップする **LDAP** を使用する基本的な例です。**/etc/postfix/main.cf** ファイルに以下の内容が含まれていることを確認してください。

```
alias_maps = hash:/etc/aliases, ldap:/etc/postfix/ldap-aliases.cf
```

/etc/postfix/ldap-aliases.cf ファイルがない場合は作成し、以下の内容を追加します。

```
server_host = ldap.example.com
search_base = dc=example, dc=com
```

ldap.example.com、**example**、**com** パラメーターは、既存の利用可能な **LDAP** サーバーの仕様と置き換える必要があります。



注記

/etc/postfix/ldap-aliases.cf ファイルは、**LDAP SSL** と **STARTTLS** を有効にするパラメーターなどの様々なパラメーターを指定できます。詳細は、**ldap_table(5)** man ページを参照してください。

LDAP の詳細は、System-Level Authentication Guide の OpenLDAP を参照してください。

15.3.2. Sendmail

Sendmail の主な目的は、他の MTA と同様に、通常 **SMTP** プロトコルを使用して、ホスト間で電子メールを安全に転送することです。Sendmail は非推奨とみなされており、可能な場合は Postfix の使用が推奨されます。詳細は、「postfix」 を参照してください。

15.3.2.1. 用途と制約

認識すべき重要な点は、Sendmail ができないことではなく、Sendmail が何であるか、何ができるのかということです。複数のロールを果たすモノリシックなアプリケーションの時代には、Sendmail は組織内で電子メールサーバーを稼働するために必要な唯一のアプリケーションと思われるかもしれません。Sendmail は各ユーザーのディレクトリーにメールを送信し、ユーザー用にアウトバウンドメールを送信できるため、技術的にはこれが当てはまります。Sendmail はメールを各ユーザーのディレクトリーにスプールして、ユーザーに送信メールを配信できるからです。ユーザーは通常、**POP** または

IMAP を使用してメッセージをローカルマシンにダウンロードする MUA を使用してメールと対話したい場合があります。ユーザーは通常、POP または IMAP を使用する MUA で電子メールとやりとりを行い、ローカルマシンにメッセージをダウンロードする方法を望みます。こうした他のアプリケーションを Sendmail と連動させることは可能ですが、実際、それらが存在する理由は異なり、独立して機能することができます。

Sendmail で設定すべき、また設定できるすべての用途の説明は、本セクションの対象外となります。Sendmail には文字どおり数百におよぶ様々なオプションやルールセットがあるため、Sendmail のあらゆる機能や問題修正方法に関する専門的な資料が多くあります。Sendmail リソースのリスト 「関連情報」 についてはを参照してください。

本セクションでは、Sendmail と共にデフォルトでインストール済みのファイルを概説し、迷惑メール (スパム) の停止方法や **LDAP** での Sendmail の拡張方法など基本設定の変更について説明します。

15.3.2.2. Sendmail のデフォルトのインストール

Sendmail を使用するには、**root** で以下を実行し、まず使用中のシステムに **sendmail** パッケージがインストールされていることを確認します。

```
~]# yum install sendmail
```

Sendmail を設定するには、**root** で実行して、お使いのシステムに **sendmail-cf** パッケージがインストールされていることを確認します。

```
~]# yum install sendmail-cf
```

yum を使用したパッケージのインストールは 「パッケージのインストール」 を参照してください。

Sendmail を使用する前に、デフォルトの MTA が Postfix から切り替わっている必要があります。デフォルトの MTA の切り替え方法は、 「メール転送エージェント (MTA)」 を参照してください。

Sendmail 実行可能ファイルは **/sendmail** です。

Sendmail 設定ファイルは **/etc/mail/sendmail.cf** に保存されます。**sendmail.cf** ファイルを直接編集しないでください。sendmail で設定を変更するには、**/etc/mail/sendmail.mc** ファイルを編集し、元々の **/etc/mail/sendmail.cf** ファイルのバックアップを取得し、**sendmail** サービスを再起動します。再起動時に、sendmail.cf ファイル、およびデータベースの全バイナリー表現がすべて再構築されます。

```
# systemctl restart sendmail
```

Sendmail の設定に関する詳細は 「Sendmail の一般的な設定変更」 を参照してください。

以下のような様々な Sendmail 設定ファイルが、**/etc/mail/** ディレクトリーにインストールされています。

- **access**: 電子メールの送信に Sendmail を使用できるシステムを指定します。
- **domaintable**: ドメイン名のマッピングを指定します。
- **local-host-names**: ホストのエイリアスを指定します。
- **mailertable**: 特定のドメインのルーティングを上書きする方法を指定します。
- **virtusertable**: ドメイン固有のエイリアス形式を設定し、1 台のマシンに複数の仮想ドメインのホスティングを可能にします。

access、**domaintable**、**mailertable**、**virtusertable** など、**/etc/mail/** ディレクトリーにあるいくつかの設定ファイルには、Sendmail が設定変更を使用できるようになる前に、データベースファイルに情報を保存します。

データベースファイルの設定に変更を追加する場合は、以下のコマンドを実行します。

```
# systemctl restart sendmail
```

15.3.2.3. Sendmail の一般的な設定変更

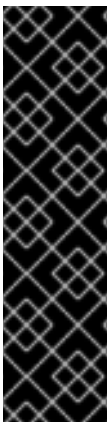
Sendmail 設定ファイルを変更する場合は、既存ファイルを編集せずに、新たに **/etc/mail/sendmail.cf** ファイルを生成するのが最適な方法です。

警告

sendmail.cf ファイルを置き換えたり変更したりする前に、バックアップコピーを作成してください。

希望する機能を Sendmail に追加する場合は、**root** で **/etc/mail/sendmail.mc** ファイルを編集します。編集が終了したら、**sendmail** サービスを再起動します。**m4** パッケージがインストールされている場合は、**m4** マクロプロセッサーが新しい **sendmail.cf** 設定ファイルを自動的に生成します。

```
~]# systemctl restart sendmail
```



重要

デフォルトの **sendmail.cf** ファイルでは、Sendmail はローカルコンピューター以外のホストからのネットワーク接続を受け入れないように設定されています。Sendmail を他のクライアント用のサーバーとして設定するには、**/etc/mail/sendmail.mc** ファイルを編集して、**DAEMON_OPTIONS** ディレクティブの **Addr=** オプションで指定されているアドレスを、**127.0.0.1** から、アクティブなネットワークデバイスの IP アドレスに変更するか、行頭に **dnl** を付けて、 **DAEMON_OPTIONS** ディレクティブをすべてコメントアウトします。終了したら、サービスを再起動して **/etc/mail/sendmail.cf** を再生成します。

```
~]# systemctl restart sendmail
```

Red Hat Enterprise Linux のデフォルト設定は、ほとんどの **SMTP** 専用サイトで機能します。

/usr/share/sendmail-cf/ ディレクトリー下のディレクトリーにあるファイルを編集する前に、**/usr/share/sendmail-cf/README** ファイルを確認してください。**/etc/mail/sendmail.cf** ファイルの今後の設定に影響を及ぼす場合があるためです。

15.3.2.4. マスカレーディング

一般的な Sendmail の設定の 1 つとして、1 台のマシンがネットワーク上の全マシンのメールのゲートウェイとして機能するように設定する方法があります。たとえば、ある企業が **mail.example.com** という名前のマシンですべての電子メールを処理して、すべての送信メールに対して一貫した返信アドレスを割り当てるとします。

このような状況では、Sendmail サーバーは、返信アドレスが **user@host.example.com** ではなく **user@example.com** となるように、その企業のネットワーク上のマシン名をマスカレードする必要があります。

これを行うには、**/etc/mail/sendmail.mc** に以下の行を追加します。

```
FEATURE(always_add_domain)dnl
FEATURE(masquerade_entire_domain)dnl
FEATURE(masquerade_envelope)dnl
FEATURE(allmasquerade)dnl
MASQUERADE_DOMAIN(`example.com.')dnl
MASQUERADE_AS(`example.com')dnl
```

sendmail.mc で変更した設定から新しい **sendmail.cf** ファイルを生成したら、以下のコマンドを実行して sendmail サービスを再起動します。

```
# systemctl restart sendmail
```

メールサーバー、**DNS** サーバーおよび **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フォーマットに合意するべきであることに注意してください。推奨される命名プラクティスの詳細は Red Hat Enterprise Linux 7 ネットワークガイド を参照してください。

15.3.2.5. Spam の停止

電子メールのスパムは、通信を要求したことがないユーザーから受信した、不要な迷惑メールとして定義することができます。これは、破壊的でコストがかかる、広く蔓延したインターネット通信標準の悪用です。

Sendmail を使用すると、迷惑メールの送信に使用されている新たなスパム技術を比較的に簡単にブロックすることができます。さらに、数多くの一般的なスパム手法もデフォルトでブロックします。Sendmail で利用できる主要なアンチスパム機能は **ヘッダーのチェック**、**リレーの否認** (バージョン 8.9 からデフォルト)、**アクセスのデータベース、送信者情報の確認** です。

たとえば、リレーとも呼ばれる **SMTP** メッセージの転送は、Sendmail バージョン 8.9 以降デフォルトでは無効になっています。この変更前には、Sendmail はメールホスト (**x.edu**) に対して、ある当事者 (**y.com**) からのメッセージを受け入れるよう指示し、そのメッセージを別の当事者 (**z.net**) に送信していました。しかし、現在は任意のドメインがサーバーを介してメールをリレーするよう Sendmail を設定する必要があります。リレードメインを設定するには、**/etc/mail/relay-domains** ファイルを編集して Sendmail を再起動してください。

```
~]# systemctl restart sendmail
```

ただし、インターネット上のサーバーからスパムメッセージを送信することもあります。その場合は、**/etc/mail/access** ファイルで利用可能な Sendmail のアクセス制御機能を使用して、迷惑なホストからの接続を阻止することができます。以下の例は、このファイルを使用したブロックの方法と Sendmail サーバーへのアクセスを具体的に許可する方法を示しています。

```
badspammer.com ERROR:550 "Go away and do not spam us anymore" tux.badspammer.com
OK 10.0 RELAY
```

この例では、**badspammer.com** から送信された電子メールはいずれも 550 RFC-821 準拠のエラーコードでブロックされ、メッセージは送り返されます。**tux.badspammer.com** のサブドメインから送信される電子メールは受け入れられます。最後の行は、10.0.. ネットワークから送信された電子メールはい

ずれもメールサーバーを通してリレー可能であることを示しています。

/etc/mail/access.db ファイルがデーターベースであるため、以下のコマンドを使用して変更を更新します。

```
# systemctl restart sendmail
```

上記の例は、アクセスの許可や阻止に関する Sendmail が持つ機能のほんの一部です。詳細情報とその他の例は、**/usr/share/sendmail-cf/README** ファイルを参照してください。

Sendmail は、メールの配信時に Procmail MDA を呼び出すため、SpamAssassin のようなスパムフィルタリングプログラムを使用して、ユーザーに対してスパムを識別してファイルに保存することも可能です。SpamAssassin の詳細な使用方法は、「spam フィルター」 を参照してください。

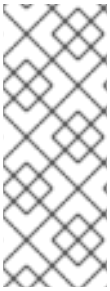
15.3.2.6. LDAP での Sendmail の使用

LDAP の使用は、大規模なグループからある特定のユーザーに関する特定の情報を検索する、非常に迅速かつ強力な方法です。たとえば、**LDAP** サーバーを使用すると、一般的な企業ディレクトリーから特定の電子メールアドレスをユーザーの苗字で検索できます。この種の実装では、**LDAP** はほどんど Sendmail から分離されており、**LDAP** が階層別のユーザー情報を保存し、Sendmail は事前にアドレスが入力された電子メールメッセージの形式で **LDAP** のクエリー結果を知らされるだけです。

ただし、Sendmail は **LDAP** とのより優れた統合をサポートします。この場合、**LDAP** を使用して、中規模レベルの組織をサポートするさまざまなメールサーバーで、**/etc/aliases** や **/etc/mail/virtusertables** などのメンテナンスされたファイルを置き換えます。つまり、**LDAP** はメールルーティングレベルを Sendmail と、その別個の設定ファイルから、様々なアプリケーションで活用できる強力な **LDAP** クラスターに抽象化します。

Sendmail の現行版は **LDAP** に対応しています。**LDAP** を使用して Sendmail を拡張するには、最初に **OpenLDAP** などの **LDAP** サーバーを稼働して、適切な設定を行います。次に、**/etc/mail/sendmail.mc** を編集して以下を追加します。

```
LDAPROUTE_DOMAIN('yourdomain.com')dnl
FEATURE('ldap_routing')dnl
```



注記

これは **LDAP** を使用した非常に基本的な Sendmail の設定にすぎません。実際の設定は **LDAP** の実装に従い、これとは大幅に異なる可能性があります。特に、共通の **LDAP** サーバーを使用するために数種の Sendmail マシンを設定する場合がそうです。

詳しい **LDAP** のルーティング設定に関する説明と例は、**/usr/share/sendmail-cf/README** を参照してください。

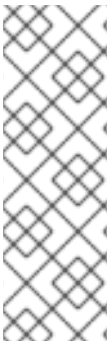
次に、**m4** マクロプロセッサーを実行し、再び Sendmail を再起動して、**/etc/mail/sendmail.cf** ファイルを再作成します。手順については 「Sendmail の一般的な設定変更」 を参照してください。

LDAP の詳細は、System-Level Authentication Guide の OpenLDAP を参照してください。

15.3.3. Fetchmail

Fetchmail は、リモートサーバーから電子メールを取得してローカルの MTA に配信する MTA です。多くのユーザーは、リモートサーバー上にあるメッセージをダウンロードするプロセスと、MUA で電子メールを読み取り、整理するプロセスを別々にする機能性を評価してます。ダイアルアップユーザーの

ニーズを踏まえて設計されている Fetchmail は、**POP3** や **IMAP** などのプロトコルを使用して、メールスプールファイルに接続し、すべての電子メールメッセージを迅速にダウンロードします。また、必要に応じて、電子メールメッセージを **SMTP** サーバーに転送することもできます。



注記

Fetchmail を使用するには、最初に **root** で以下を実行し、使用中のシステムに **fetchmail** パッケージがインストールされていることを確認します。

```
~]# yum install fetchmail
```

yum を使用したパッケージのインストールは 「パッケージのインストール」 を参照してください。

Fetchmail は、各ユーザーのホームディレクトリー内の **.fetchmailrc** ファイルを使用して、それぞれのユーザーに設定されています。これがない場合は、ホームディレクトリーに **.fetchmailrc** ファイルを作成してください。

Fetchmail は **.fetchmailrc** ファイル内の詳細設定を使用して、リモートサーバー上にある電子メールを確認し、ダウンロードします。次に、電子メールをローカルマシン上のポート 25 に配信し、ローカルの MTA を使用して電子メールを適正なユーザーのスプールファイルに配置します。Procmail が利用できる場合は起動して電子メールをフィルターし、MUA が読み込むことができるようにメールボックスに配置します。

15.3.3.1. Fetchmail の設定オプション

Fetchmai を実行する時にすべての必要なオプションをコマンドラインで渡し、リモートサーバー上の電子メールを確認することは可能ですが、**.fetchmailrc** ファイルを使用した方がはるかに簡単です。希望の設定オプションを **.fetchmailrc** ファイル内に設定し、それらのオプションが、**fetchmail** コマンドを実行するときに毎回使用されるようにします。Fetchmail の実行時にオプションを上書きしたい場合は、コマンドラインでそのオプションを指定します。

ユーザーの **.fetchmailrc** ファイルには、3 つのクラスの設定オプションが格納されています。

- **グローバルオプション**: プログラムの動作を制御する、または電子メールを確認する全接続の設定を提供する指示を Fetchmail に指定します。
- server options: ポーリングされるサーバーに必要な情報を指定します。ホスト名をはじめ、確認するポートやタイムアウトになるまでの秒数など、特定の電子メールサーバーの設定などです。こうしたオプションは、該当するサーバーを使用する全ユーザーに影響を及ぼします。
- **ユーザーオプション**: 指定された電子メールサーバーを使用して、電子メールの認証や確認を行うにあたって必要なユーザー名、パスワードなどの情報を格納します。

グローバルオプションは **.fetchmailrc** ファイルの上部に表示され、その後に 1 つ以上のサーバーオプションが表示されます。各オプションは Fetchmail がチェックする異なるメールサーバーを指定します。ユーザーオプションは、そのメールサーバーをチェックする各ユーザーアカウントのサーバーオプションに従います。サーバーオプションと同様に、複数のユーザーオプションを指定することで特定のサーバーでの使用、同一サーバー上の複数の電子メールアカウントの確認を行うことができます。

サーバーオプションを **.fetchmailrc** ファイルで利用するには、サーバーの情報の先頭に **poll** または **skip** などの特別なオプションの動詞を使用します。**poll** アクションは、Fetchmail の実行時にこのサーバーオプションを使用して、指定されたユーザーオプションで電子メールを確認するよう Fetchmail に指示します。ただし、**skip** アクションの後にあるサーバーオプションは、Fetchmail が呼び出された時

にサーバーのホスト名が指定されていない限り確認されません。**skip** オプションは、特定して呼び出された時にスキップされたサーバーのみを確認し、現在稼働中の設定には影響を及ぼさないため **.fetchmailrc** ファイルの設定をテストする時に役立ちます。

以下は、**.fetchmailrc** ファイルの例です。

```
set postmaster "user1"
set bouncemail

poll pop.domain.com proto pop3
  user 'user1' there with password 'secret' is user1 here

poll mail.domain2.com
  user 'user5' there with password 'secret2' is user1 here
  user 'user7' there with password 'secret3' is user1 here
```

この例では、グローバルオプションにより、最終手段としてユーザーに電子メールが送信されるように指定されており (**postmaster** オプション)、すべての電子メールエラーは送信者ではなく、ポストマスターに送信されます (**bouncemail** オプション)。**set** アクションは、この行にグローバルオプションが含まれていることを Fetchmail に伝えます。その後、2 つのメールサーバーが指定されており、もう 1 つは **POP3** を使用してチェックするように設定されます。1 つは **user1** を使用して確認するように設定され、もう 1 つは様々なプロトコルを試みて機能するものを見つけるように設定されます。これにより、1 つの MUA 受信トレイに表示され、複数のサーバーで複数のクラスを確認できます。各ユーザーの固有の情報は、**user** アクションで開始します。

注記

ユーザーはパスワードを **.fetchmailrc** ファイルに配置する必要はありません。**with password 'password'** のセクションを省略すると、Fetchmail は起動時にパスワードを要求するようになります。

Fetchmail には、グローバルオプション、サーバーオプション、ローカルオプションが多数あります。これらの多くのオプションは、ほとんど使用されないか、非常に特殊な状況にのみ適用されます。**fetchmail** の man ページに、各オプションの詳細が記載されていますが、最も一般的なものを以下の 3 セクションで説明します。

15.3.3.2. グローバルオプション

グローバルオプションは、**set** アクションの後に、それぞれ 1 行ずつ配置する必要があります。

- ***daemon* seconds**: Fetchmail がバックグラウンドに残るデーモンモードを指定します。seconds を、Fetchmail がサーバーをポーリングするまでの待機時間の秒数に置き換えます。
- ***postmaster***: 配信に問題が生じた場合にローカルユーザーがメールを送信するよう指定します。
- ***syslog***: エラーとステータスメッセージのログファイルを指定します。デフォルトは **/var/log/maillog** です。

15.3.3.3. サーバーオプション

サーバーオプションは、**.fetchmailrc** 内の **poll** アクションまたは **skip** アクションの後の行にそれぞれ追加する必要があります。

- ***auth* auth-type**: auth-type を、使用する認証のタイプに置き換えます。デフォルトでは、***password*** 認証が使用されますが、一部のプロトコルは、他のタイプの認証 (***kerberos_v5***、***kerberos_v4***、および ***ssh*** など) をサポートしています。***any*** の認証タイプを使用した場合、Fetchmail は、パスワードを必要としない方法を最初に試みます。次に、パスワードをマスクする方法を試みた後、最後にサーバーに暗号化されていないパスワードを送信して認証を試みます。

- ***interval* number**: 指定されたサーバーを、**number** で指定した時間間隔でポーリングし、設定された全サーバーの電子メールを確認します。このオプションは、通常のユーザーがほとんどメッセージを受信しない電子メールサーバーに使用されます。

- ***port* port-number** : port-number をポート番号に置き換えます。この値は、指定されたプロトコルのデフォルトのポート番号を上書きします。

- ***proto* protocol**: protocol を、サーバーのメッセージを確認する時に使用する ***pop3*** や ***imap*** などのプロトコルに置き換えます。

- ***timeout* seconds**: seconds を、Fetchmail が接続の試行をやめてからサーバーが非アクティブとなる秒数に置き換えます。この値を設定しないと、デフォルトの ***300*** 秒が使用されます。

15.3.3.4. ユーザーオプション

ユーザーオプションは、サーバーオプションの下の各行に置かれる場合と、サーバーオプションと同じ行に置かれる場合があります。いずれの場合も、定義されるオプションは ***user*** オプション (以下で説明) に従う必要があります。

- ***fetchall***: 既読メッセージを含め Fetchmail がキューにあるすべてのメッセージをダウンロードするように命令します。デフォルトでは、Fetchmail は新規メッセージのみをダウンロードするようになっています。

- ***fetchlimit* number**: number を、停止する前に取得するメッセージ数に置き換えます。

- ***flush***: 新規メッセージを取得する前に、キューにあるすべての既読メッセージを削除します。

- ***limit* max-number-bytes**: max-number-bytes を、Fetchmail で取得する時に許容されているメッセージの最大バイトサイズに置き換えます。このオプションでは低速のネットワークリンクが提供されるため、サイズが大きいメッセージのダウンロードに時間がかかりすぎる場合に有用です。

- ***password* 'password'**: password を、ユーザーのパスワードに置き換えます。

- ***preconnect* "command"**: command を、ユーザー宛のメッセージを取得する前に実行するコマンドに置き換えます。

- ***postconnect* "command"**: command を、ユーザー宛のメッセージを取得した後に実行するコマンドに置き換えます。

- ***ssl***: SSL 暗号化を有効にします。この英語版が公開された時点で、デフォルトのアクションでは ***SSL2***、***SSL3***、***SSL23***、***TLS1***、***TLS1.1***、および ***TLS1.2*** から最良のものを使用します。***SSL2*** は廃止されたものと見なされ、POODLE: SSLv3 脆弱性 (CVE-2014-3566) のため、***SSLv3*** を使用しないようにする必要があります。ただし、TLS1 以降の使用を強制できないため、接続するメールサーバーが ***SSLv2*** と ***SSLv3*** を**使用しない**よう設定する必要があります。サーバーが ***SSLv2*** および ***SSLv3*** を使用**しない**ように設定できな場合は、***stunnel*** を使用します。

- ***sslproto***: 許可された SSL プロトコルまたは TLS プロトコルを定義します。可能な値は ***SSL2***、***SSL3***、***SSL23***、および ***TLS1*** です。***sslproto*** が省略された場合、未設定の場合、また

*は無効な値に設定された場合のデフォルトの値は **SSL23** です。デフォルトのアクションは **SSLv2**、**SSLv3**、**TLSv1**、**TLS1.1**、および **TLS1.2** から最適なものを使用します。SSL または TLS の他の値を設定すると、他のすべてのプロトコルが無効になることに注意してください。POODLE: SSLv3 脆弱性 (CVE-2014-3566) のため、このオプションを省略するか、**SSLv23** に設定し、対応するメールサーバーが SSLv2 と **SSLv3** を 使用しない よう設定することが推奨されます。サーバーが **SSLv2** および **SSLv3** を使用**しない** ように設定できな場合は、**stunnel** を使用します。*

- ***user "username"**: username を、Fetchmail がメッセージの取得に使用するユーザー名に置き換えます。**このオプションは、他のすべてのユーザーオプションの前に付ける必要があります。***

15.3.3.5. Fetchmail のコマンドオプション

***fetchmail** コマンドの実行時にコマンドライン上で使用される Fetchmail オプションの大半は、**.fetchmailrc** 設定オプションを反映します。この方法では、Fetchmail は設定ファイルの有無を問わず使用できます。これらの設定オプションは、**.fetchmailrc** ファイルに残しておいた方が簡単なため、多くの場合はコマンドライン上では使用しません。*

***fetchmail** コマンドは、特定の用途のオプションと併せて実行した方が望ましい場合もあります。コマンドラインで指定されるオプションはいずれも設定ファイルオプションを上書きするため、エラーが発生した場合は、コマンドオプションを使用して、エラーの原因になっている **.fetchmailrc** 設定を一時的に上書きすることが可能です。*

15.3.3.6. 情報提供またはデバッグのオプション

***fetchmail** コマンドの後に使用されるオプションの一部は、重要な情報を提供する場合があります。*

- ***--configdump**: **.fetchmailrc** および Fetchmail のデフォルト値からの情報に基づいた可能なオプションをすべて表示します。このオプションを使用すると、どのユーザーの電子メールも取得されません。*
- ***-s**: Fetchmail をサイレントモードで実行し、**fetchmail** コマンドの後にエラー以外のメッセージが表示されないようにします。*
- ***-v**: Fetchmail を verbose モードで実行し、Fetchmail とリモート電子メールサーバー間のすべての通信を表示します。*
- ***-V**: 詳細なバージョン情報の表示、グローバルオプションのリスト表示、電子メールプロトコルや認証メソッドなどの各ユーザーと使用する設定の表示を行います。このオプションを使用すると、どのユーザーの電子メールも取得されません。*

15.3.3.7. 特殊なオプション

*これらのオプションは **.fetchmailrc** ファイルによく見られるデフォルト値を上書きする時に役立つ場合があります。*

- ***-a**: Fetchmail は、新規または既読を問わず、すべてのメッセージをリモートの電子メールサーバーからダウンロードします。デフォルトでは、Fetchmail は新規メッセージのみをダウンロードします。*
- ***-k**: Fetchmail はメッセージをダウンロードした後、リモートの電子メールサーバー上にメッセージを残します。このオプションを使用すると、メッセージをダウンロード後に削除するデフォルトの動作は上書きされます。*

- **-l max-number-bytes** : Fetchmail は一定のサイズを超えるメッセージはダウンロードせず、リモートの電子メールサーバー上に残します。

- **--quit** Fetchmail デーモンのプロセスを終了します。

その他のコマンドと **.fetchmailrc** オプションは、**fetchmail** の man ページを参照してください。

15.3.4. メール転送エージェント (MTA) の設定

Mail Transport Agent (MTA) は電子メールの送信に不可欠です。**Evolution** や **Mutt** などの Mail User Agent (MUA) を使用してメールの読み取り、作成を行うことができます。ユーザーが MUA から電子メールを送信すると、メッセージは MTA に渡されます。MTA は一連の MTA を通じて、メッセージが送信先に届くまで送信します。

ユーザーがシステムから電子メールを送信する予定でなくても、一部の自動化されたタスクまたはシステムプログラムは、**mail** コマンドを使用して、ログメッセージを含む電子メールをローカルシステムの **root** ユーザーに送信する場合があります。

Red Hat Enterprise Linux 7 は Postfix と Sendmail の 2 つの MTA を提供します。両方がインストールされている場合は、Postfix がデフォルトの MTA になります。

15.4. メール配信エージェント (MDA)

Red Hat Enterprise Linux には、プライマリー MDA として **Procmail** が含まれています。両方のアプリケーションは LDA とみなされ、MTA のスプールファイルからユーザーのメールボックスにメールを移動します。ただし、Procmail の方が堅牢なフィルタリングシステムを提供します。

このセクションでは、Procmail についてのみ詳しく説明します。**mail** コマンドの詳細は、man ページの (**man mail**) を参照してください。

ローカルホストのメールスプールファイルに電子メールが置かれると、Procmail が配信とフィルタリングを行います。Procmail は強力な上、システムリソースの使用が低いため、幅広く利用されています。Procmail は、電子メールクライアントアプリケーションが読み取る電子メールを配信するという重要なロールを果たします。

Procmail は、様々な方法で呼び出すことができます。MTA が電子メールをメールスプールファイルの中に置くと常に Procmail が起動します。次に、Procmail は電子メールを MUA のためにフィルタリング、ファイル保存して、終了します。別の方法としては、メッセージを受信すると常に Procmail を実行するように MUA を設定し、メッセージが正しいメールボックスに移動するようにできます。デフォルトでは、**/etc/procmailrc** または **~/.procmailrc** ファイル (別名 rc ファイル) がユーザーのホームディレクトリーにあると、MTA が新規メッセージを受信するたびに Procmail が呼び出されます。

デフォルトでは、**/etc** ディレクトリーにはシステム全体の **rc** ファイルが存在せず、ユーザーのホームディレクトリーに **.procmailrc** ファイルが存在しません。このため、Procmail を使用するには、各ユーザーが特定の環境変数とルールを用いて **.procmailrc** ファイルを構築する必要があります。

Procmail が電子メールメッセージに対応するかどうかは、そのメッセージが rc ファイルの特定の条件または レシピ と適合するかどうかによって決まります。あるメッセージが任意のレシピと適合する場合、電子メールは特定のファイルに置かれるか削除され、それ以外は処理されます。

Procmail が起動すると、電子メールメッセージを読み取り、ヘッダー情報から本文を切り離します。次に、Procmail は **/etc/procmailrcs/** ディレクトリー内の **/etc/procmailrc** ファイルと **rc** ファイルで、デフォルトのシステム全体の Pcocmail 環境用変数とレシピを探します。その後 Procmail は、ユーザーのホームディレクトリー内で **.procmailrc** ファイルを探します。多くのユーザーは、Procmail 用に追加の **rc** ファイルも作成します。これは、ホームディレクトリーの **.procmailrc** ファイル内で参照されます。

15.4.1. Procmail の設定

Procmail の設定ファイルには、重要な環境変数が含まれています。これらの変数は、並べ替えするメッセージ、およびどのレシピとも適合しないメッセージの処理を指定します。

これらの環境変数は通常 **~/.procmailrc** ファイルの冒頭に、以下のような形式で表示されます。

```
env-variable="value"
```

この例では、**env-variable** が変数の名前で、**value** が変数を定義します。

ほとんどの Procmail ユーザーが使用していない環境変数が多くあります。また、重要な環境変数の多くがデフォルト値で定義されています。重要な環境変数の多くは、すでにデフォルト値で定義されています。大抵の場合は、以下のような変数が使用されます。

- **DEFAULT**: どのレシピにも適合しないメッセージが配置された場合のデフォルトのメールボックスを設定します。
  デフォルトの **DEFAULT** 値は、**$ORGMAIL** と同じです。

- **INCLUDERC**: 照合するメッセージに対する多くのレシピを格納する追加の **rc** ファイルを指定します。これにより、Procmail レシピのリストは、スパムのブロック、電子メールリストの管理など異なるロールを果たす個別のファイルに分割されます。その結果、そうしたファイルは、ユーザーの **~/.procmailrc** ファイル内のコメント文字を使用して、オンやオフにすることができます。
  たとえば、ユーザーの **~/.procmailrc** ファイル内の行は以下のようになります。

  ```
  MAILDIR=$HOME/Msgs
  INCLUDERC=$MAILDIR/lists.rc
  INCLUDERC=$MAILDIR/spam.rc
  ```

  電子メールのリストの Procmail フィルターをオフにしつつスパム制御を維持する場合は、最初の **INCLUDERC** 行をハッシュ記号 (**#**) でコメントアウトします。現在のディレクトリーに相対的なパスが使用されることに注意してください。

- **LOCKSLEEP**: Procmail が特定のロックファイルの使用を試みる時間間隔を秒単位で設定します。デフォルトは 8 秒です。

- **LOCKTIMEOUT**: ロックファイルが最後に修正された後、Procmail がそれは古くて削除可能であるとみなすまでに経過する必要のある時間を秒単位で設定します。デフォルトは 1024 秒です。

- **LOGFILE**: Procmail の情報やエラーメッセージが書き込まれるファイルです。

- **MAILDIR**: Procmail 用の現在作業中のディレクトリーを設定します。設定されると、他の Procmail のパスはすべてこのディレクトリーに対する相対パスになります。

- **ORGMAIL** - 元のメールボックス、またはデフォルトやレシピで必要な場所にメッセージを配置できなかった場合に、メッセージを配置する別の場所を指定します。
  デフォルトでは、**/var/spool/mail/$LOGNAME** の値が使用されます。

- **SUSPEND**: スワップ領域など必要なリソースが利用できない場合に、Procmail が一時停止する時間を秒単位で設定します。

- **SWITCHRC**: 追加の Procmail レシピが格納されている外部ファイルをユーザーが指定できるようにします。これは、**INCLUDERC** オプションとよく似ていますが、レシピのチェックが参照先の設定ファイル上で実際に停止され、**SWITCHRC** が指定するファイル上のレシピのみが使用

される点が異なります。

- ***VERBOSE***: Procmail が詳細な情報をログ記録するようにします。このオプションはデバッグに役立ちます。

その他の重要な環境変数は、シェルから引き出されます。例えば、ログイン名の ***LOGNAME***、ホームディレクトリーの場所である ***HOME***、デフォルトのシェルである ***SHELL*** などです。

すべての環境変数に関する包括的な説明やデフォルト値は、man ページ ***procmailrc*** を参照してください。

15.4.2. Procmail レシピ

多くの場合、新規ユーザーが Procmail の使用法を学習するにあたって最も難しいと感じるのは、レシピの構築です。これは、レシピが適合する文字列の条件を指定するために **正規表現** を使用してメッセージ照合を行うためです。ただ、正規表現の構築はそれほど難しくなく、読んで理解することも簡単です。その上、Procmail のレシピを書く方法は、正規表現にかかわらず一貫性があるため、例を使用して学習すると簡単です。Procmail のレシピの例は、「レシピの例」 を参照してください。

Procmail レシピは以下の形式を使用します:

```
:0 flags : lockfile-name
* condition_1_special-condition-character condition_1_regular_expression
* condition_2_special-condition-character condition-2_regular_expression
* condition_N_special-condition-character condition-N_regular_expression
  special-action-character
  action-to-perform
```

Procmail レシピの最初の 2 文字は、コロンとゼロです。ゼロの後に様々なフラグを追加して、Procmail がレシピを処理する方法を制御します。**flags** セクションの後ろにコロンを付けると、このメッセージ用にロックファイルが作成されることを示しています。ロックファイルが作成されると、その名前は **lockfile-name** を置き換えて指定することが可能です。

レシピには、メッセージと適合させる様々な条件を追加できます。条件がない場合は、すべてのメッセージがレシピと適合することになります。正規表現は、メッセージ照合を容易にするために、一部の条件で使用されます。複数の条件を使用する場合は、アクションが実行されるためにはすべてが適合しなければなりません。条件は、レシピの 1 行目に設定されているフラグに基づいてチェックされます。アスタリスク文字 (*) の後にオプションの特殊文字を追加すると、さらに条件を制御できます。

action-to-perform 引数は、メッセージが条件の 1 つに適合する場合に実行するアクションを指定します。1 つのレシピに指定できるアクションは 1 つのみとなります。多くの場合、メールボックスの名前がここで使用され、適合するメッセージをファイルに誘導し、電子メールを効果的に並べ替えます。特別なアクションの文字は、アクションが指定される前に使用することもできます。詳細は、「特別な条件とアクション」 を参照してください。

15.4.2.1. 配信と非配信レシピ

レシピがある特定のメッセージと適合した場合に使用されるアクションにより、それが **配信** レシピ、または **非配信** レシピとみなされるかが判断されます。配信レシピには、ファイルへのメッセージの書き込み、別のプログラムへのメッセージ送信、別の電子メールアドレスへのメッセージ転送などのアクションが含まれています。非配信レシピは、**ネストされたブロック** などその他のアクションをカバーします。ネストされたブロックは、中括弧 **{ }** で囲まれたアクションセットで、レシピの条件に適合するメッセージで実行されます。ネストされたブロックは、互いにネストさせることができるため、メッセージに対するアクションを特定して実行するにあたっての制御力が強化されます。

メッセージが配信レシピと適合すると、Procmail は指定されたアクションを実行し、その他のレシピとメッセージとの比較を停止します。非配信レシピと適合するメッセージの場合は、他のレシピに対する照合は継続されます。

15.4.2.2. フラグ

フラグは、レシピの条件をメッセージに照合する方法、またはそれを行うかどうかを決定するにあたって不可欠です。**egrep** ユーティリティーは、条件の照合のために内部で使用されます。一般的に使用されるフラグは以下のとおりです。

- ***A***: ***A*** や ***a*** のフラグが付いていない以前のレシピもこのメッセージに適合する場合にのみ、このレシピが使用されることを指定します。
- ***a***: ***A*** や ***a*** のフラグが付いた以前のレシピもこのメッセージに適合し、**かつ** 正常に完了した場合にのみこのレシピが使用されることを指定します。
- ***B***: メッセージの本文を解析し、適合する条件を検索します。
- ***b***: ファイルへのメッセージの書き込みや転送など、結果として生じるアクションにその本文を使用します。これはデフォルトの動作です。
- ***c*** – 電子メールのカーボンコピーを生成します。必要なアクションをメッセージで実行し、メッセージのコピーは ***rc*** のファイル内で引き続き処理することができるため、レシピの配信に役立ちます。
- ***D***: ***egrep*** の照合で大文字と小文字を区別します。デフォルトでは、照合プロセスでは大文字と小文字を区別していません。
- ***E***: ***A*** フラグと類似していますが、レシピ内の条件は、直前にある ***E*** フラグなしのレシピが適合しない場合のみに、メッセージと照合されます。これは ***else*** アクションと類似しています。
- ***e***: 直前のレシピで指定されたアクションが失敗した場合のみ、レシピがメッセージに照合されます。
- ***f***: フィルターとしてパイプを使用します。
- ***H***: メッセージのヘッダーを解析し、適合する条件を検索します。これはデフォルトの動作です。
- ***h***: 結果として生じるアクションでヘッダーを使用します。これはデフォルトの動作です。
- ***w***: Procmail に対して、指定されたフィルターまたはプログラムが終了するのを待ち、メッセージがフィルターされたとみなす前に正常に終了したかどうかを報告するよう指示します。
- ***W***: プログラム障害のメッセージが抑制されている点を除いては ***w*** と同じです。

その他のフラグの詳細なリストは、***procmailrc*** man ページを参照してください。

15.4.2.3. ローカルロックファイルの指定

ロックファイルは、Procmail で複数のプロセスが 1 つのメッセージを同時に変更しないようにするために非常に役立ちます。ローカルロックファイルを指定するには、レシピの 1 行目の任意のフラグの後にコロン (**:**) を追加します。これにより、送信先のファイル名に基づいたローカルロックファイルと、***LOCKEXT*** のグローバル環境変数で設定されたものすべてが作成されます。

別の方法としては、このレシピで使用するローカルロックファイルの名前をコロンの後に指定します。

15.4.2.4. 特別な条件とアクション

Procmail レシピの条件とアクションの前に使用される特殊文字により、解釈の仕方が変わります。

以下の文字は、レシピの条件の行頭でアスタリスク文字 (*) の後に使用できます。

- **!**: 条件の行では、この文字により条件が反転し、条件がメッセージに一致しない場合にのみ、適合が発生するようになります。
- **<** メッセージが、指定されているバイト数に収まっているかどうかを確認します。
- **>**: メッセージが、指定されているバイト数を超えているかどうかを確認します。

以下の文字は、特別なアクションを実行するために使用されます。

- **!**: アクションの行では、この文字は、指定された電子メールアドレスにメッセージを転送するように Procmail に指示します。
- **$**: **rc** ファイルで以前に設定された変数を参照します。多くの場合は、さまざまなレシピによって参照される共通のメールボックスを設定するために使用されます。
- **|**: 指定したプログラムを開始し、メッセージを処理します。
- **{** および **}**: 適合するメッセージに適用する追加のレシピを格納するために使用される、ネストされたブロックを構築します。

アクションの行頭に特殊文字を使用しない場合、Procmail はアクションの行がメッセージを書き込むためのメールボックスを指定していると仮定します。

15.4.2.5. レシピの例

Procmail は極めて柔軟性の高いプログラムですが、この柔軟性が原因で、新規ユーザーが Procmail のレシピを一から作成するのが難しい場合があります。

Procmail レシピの条件を構築するスキルを向上させる最適な方法は、正規表現をしっかり理解し、他の人が構築した多くの例を参照することから始まります。正規表現に関する詳細な説明は、本セクションでは扱いません。Procmail のレシピの構造と役立つ Procmail のサンプルレシピは、インターネット上の様々なところに掲載されています。正規表現の適切な使用と調整方法は、これらのレシピ例を参照してください。また、基本的な正規表現ルールの概要は、man ページの **grep(1)** を参照してください。

以下にあげる簡単な例は、Procmail のレシピの基本構造を記載しており、構造をさらに複雑にするための基盤を示しています。

以下の例に示すように、基本的なレシピには条件さえも含まれていません。

```
:0:
new-mail.spool
```

最初の行は、ローカルのロックファイルを作成することを指定しますが、名前を指定していません。そのため、Procmail は宛先ファイル名を使用して、**LOCKEXT** 環境変数に指定された値を追加します。条件が指定されていないため、すべてのメッセージがこのレシピと一致し、**MAILDIR** 環境変数で指定されたディレクトリー内にある、**new-mail.spool** という単一の spool ファイルに配置されます。その後、MUA はこのファイルでメッセージを表示できます。

このような基本レシピは、**rc** ファイルの末尾に置かれ、メッセージをデフォルトの場所に送ります。

以下の例では、特定の電子メールアドレスからのメッセージを照合して、削除します。

```
:0
* ^From: spammer@domain.com
/dev/null
```

この例では、**spammer@domain.com** から送信されたメッセージはすべて **/dev/null** デバイスに送信され、削除されます。

警告

メッセージを **/dev/null** に送信して永久に削除してしまう前に、ルールが目的どおりに機能していることを確認してください。レシピが間違えて目的以外のメッセージを対象にすると、それらのメッセージは消えてしまい、ルールのトラブルシューティングが困難になります。

この問題に対処する優れた方法としては、レシピのアクションを特別なメールボックスに移動させることです。これは、後検出を見つけるのに、時折使用できます。メッセージが間違って適合されることがなく満足できる状態になったら、そのメールボックスは削除して、メッセージを **/dev/null** に送信するよう指示します。

以下のレシピでは、特定のメーリングリストから送信された電子メールを取得して、特定のフォルダーに配置します。

```
:0:
* ^(From|Cc|To).*tux-lug
tuxlug
```

tux-lug@domain.com のメーリングリストから送信されたメッセージはすべて、MUA 用に自動的に **tuxlug** メールボックスに置かれます。**From**、**Cc**、**To** の行にメーリングリストの電子メールアドレスが入っている場合は、この例の条件がメッセージに適合する点に注意してください。

さらに詳しい強力なレシピについては、「関連情報」 の Procmail に関する多くのオンライン資料を参照してください。

15.4.2.6. spam フィルター

Procmail は、新規の電子メールを受信すると Sendmail、Postfix、Fetchmail によって呼び出されるため、スパム対策の強力なツールとして使用できます。

これは、Procmail が SpamAssassin と併用された場合に特に有効です。これらの 2 つのアプリケーションを併用すると、スパムメールを迅速に特定して、並び替えまたは破棄できます。

SpamAssassin は、ヘッダー分析、テキスト分析、ブラックリスト、スパム追跡データベース、自己学習型 Bayesian スパム分析を使用して、迅速かつ正確にスパムの特定とタグ付けを行います。

注記

*SpamAssassin を使用するには、**root** で以下を実行して、最初にご使用のシステムに **spamassassin** パッケージがインストールされていることを確認します。*

```
~]# yum install spamassassin
```

yum を使用したパッケージのインストールは 「パッケージのインストール」 を参照してください。

*ローカルユーザーが SpamAssassin を使用する最も簡単な方法は、**~/.procmailrc** ファイルの最上部付近に以下の行を追加することです。*

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-default.rc
```

***/etc/mail/spamassassin/spamassassin-default.rc** には、シンプルな Procmail ルールが記載されており、受信するすべての電子メールに対して SpamAssassin を有効にします。電子メールがスパムであると判断された場合には、ヘッダー内でタグ付けされ、タイトルの先頭には以下のようなパターンが追加されます。*

```
SPAM
```

電子メールのメッセージ本文にも、スパム診断の理由となった要素の継続的な記録が先頭に追加されます。

スパムとしてタグ付けされた電子メールをファイル保存するには、以下と同様のルールを使用することができます。

```
:0 Hw
* ^X-Spam-Status: Yes
spam
```

*このルールにより、スパムとしてヘッダーにタグ付けされた電子メールはすべて、**spam** と呼ばれるメールボックスにファイルとして保存されます。*

*SpamAssassin は Perl スクリプトであるため、ビジー状態のサーバーではバイナリー SpamAssassin デーモン (**spamd**) とクライアントアプリケーション (spamc) を使用する必要がある場合があります。ただし、SpamAssassin をこのように設定するには、ホストへの **root** アクセスが必要です。*

***spamd** デーモンを起動するには、以下のコマンドを入力します。*

```
~]# systemctl start spamassassin
```

システムの起動時に SpamAssassin デーモンを起動するには、以下のコマンドを実行します。

```
systemctl enable spamassassin.service
```

サービスの起動および停止に関する詳細は、10章systemd によるサービス管理 を参照してください。

*Procmail が Perl スクリプトではなく、SpamAssassin クライアントアプリケーションを使用するように設定するには、**~/.procmailrc** ファイルの最上部付近に以下の行を追加します。システム全体の設定には、**/etc/procmailrc** に配置します。*

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-spamc.rc
```

15.5. メールユーザーエージェント

Red Hat Enterprise Linux には、利用可能な電子メールプログラムが多数あります。たとえば、Evolution のようなグラフィカル電子メールクライアントプログラムと、**mutt** のようなテキストベースの電子メールプログラムがあります。

本セクションでは、クライアントとサーバー間の通信のセキュリティー保護について重点的に説明していきます。

15.5.1. 通信のセキュリティー保護

Thunderbird、Evolution、Mutt など、Red Hat Enterprise Linux に装備されている MUA は、SSL 暗号化電子メールセッションを提供します。

暗号化されていないネットワークを行き来する他のサービスと同様に、ユーザー名、パスワード、メッセージ全体などの電子メールに関する重要な情報は、ネットワーク上のユーザーによって傍受、閲覧される可能性があります。また、標準の **POP** および **IMAP** プロトコルは、認証情報を暗号化せずに渡すため、ユーザー名とパスワードはネームサーバー上で渡される時に攻撃者がそれらを収集して、ユーザーのアカウントに侵入することが可能性があります。

15.5.1.1. セキュアな電子メールクライアント

リモートサーバー上の電子メールを確認するように設計されている Linux MUA のほとんどは、SSL 暗号化に対応しています。電子メールを取得する時に SSL を使用するためには、SSL は電子メールクライアントとサーバーの両方で有効である必要があります。

SSL はクライアント側で簡単に有効にできます。多くの場合、MUA の設定ウィンドウでボタンをクリックするか、MUA 設定ファイルのオプションを使用して実行できます。セキュアな **IMAP** および **POP** には、MUA がメッセージの認証およびダウンロードに使用する既知のポート番号 (993 と 995) があります。

15.5.1.2. 電子メールクライアントの通信のセキュリティー保護

電子メールサーバー上の **IMAP** および **POP** のユーザーに SSL 暗号化を行うことは簡単です。

最初に SSL 証明書を作成します。これは、**認証局** (CA) に SSL 証明書を申請するか、自己署名証明書を作成するかのいずれかの方法が選択できます。

警告

自己署名証明書は、テスト目的のみで使用することを推奨します。実稼働環境で使用するサーバーは、CA が署名した SSL 証明書を使用してください。

IMAP や **POP** に対し自己署名 SSL 証明書を作成するには、**/etc/pki/dovecot/** ディレクトリーに移動し、**/etc/pki/dovecot/dovecot-openssl.cnf** 設定ファイルの証明書パラメーターを編集し、**root** で次のコマンドを入力します。

```
dovecot]# rm -f certs/dovecot.pem private/dovecot.pem
dovecot]# /usr/libexec/dovecot/mkcert.sh
```

終了したら、**/etc/dovecot/conf.d/10-ssl.conf** ファイルに次の設定ファイルがあることを確認してください。

```
ssl_cert = </etc/pki/dovecot/certs/dovecot.pem
ssl_key = </etc/pki/dovecot/private/dovecot.pem
```

以下のコマンドを実行して、**dovecot** デーモンを再起動します。

```
~]# systemctl restart dovecot
```

別の方法として、**stunnel** コマンドを **IMAP** サービスまたは **POP** サービスへの標準的なセキュアでない接続で暗号化ラッパーとして使用することも可能です。

stunnel ユーティリティーは、Red Hat Enterprise Linux に装備されている外部の OpenSSL ライブラリーを使用して、強力な暗号化を実現し、ネットワーク接続を保護します。SSL 証明書を取得するためには、CA に申請することが推奨されますが、自己署名証明書を作成することも可能です。

stunnel のインストール方法とその基本設定の作成方法は、Red Hat Enterprise Linux 7 セキュリティーガイドの **stunnel** の使用 を参照してください。**stunnel** を **IMAPS** と **POP3S** のラッパーとして設定するには、以下の行を **/etc/stunnel/stunnel.conf** 設定ファイルに追加します。

```
[pop3s]
accept = 995
connect = 110

[imaps]
accept = 993
connect = 143
```

セキュリティーガイドでは、**stunnel** の起動および停止の方法を説明します。起動後は、**IMAP** または **POP** の電子メールクライアントを使用し、SSL 暗号化を使用して電子メールサーバーに接続できます。

15.6. メールサーバーのスパム対策およびウイルス対策設定

メール配信が始まると、受信メールにはスパムで知られる迷惑メールが入っている可能性があります。これらのメールには、有害なウイルスおよびマルウェアも含まれている可能性があり、システムにとってセキュリティー上のリスクがあるほか、潜在的な生産性の損失につながります。

これらのリスクを回避するために、受信メールをフィルタリングし、スパム対策およびウイルス対策ソリューションを使用して、ウイルスがあるかどうかを確認することができます。

15.6.1. メール転送エージェント (MTA) またはメール配信エージェント (MDA) のスパムフィルタリング設定

メール転送エージェント (MTA)、メール配信エージェント (MDA)、またはメールユーザーエージェント (MUA) にスパムフィルターを設定できます。本章では MTA および MDA のスパムフィルタリングについて説明します。

15.6.1.1. メール転送エージェント (MTA) のスパムフィルタリング設定

Red Hat Enterprise Linux 7 には、Postfix と Sendmail の 2 つの主要 MTA が装備されています。

MTA のインストールおよび設定に関する詳細は、「メール転送エージェント (MTA)」 を参照してください。

Sendmail を使用すると、MTA 側でスパムを停止できます。これには、**ヘッダーチェック**、**リレー拒否**、**アクセスデータベース**、および**送信者情報チェック** など、いくつかのスパム対策機能があります。詳細は、「Spam の停止」 を参照してください。

さらに、Postfix および Sendmail の両方は、サードパーティーのメールフィルター (milter) に対応するため、メール処理チェーンでスパムおよびウイルスをフィルタリングできます。Postfix の場合、milter へのサポートは postfix パッケージに直接含まれています。Sendmail の場合、milter を使用するには sendmail-milter パッケージをインストールする必要があります。

15.6.1.2. メール配信エージェント (MDA) にスパムフィルタリングの設定

Red Hat Enterprise Linux には、Procmail および **mail** ユーティリティーの 2 つの主要 MDA が装備されています。詳細は、「メール配信エージェント (MDA)」 を参照してください。

MDA のスパムに対処するために、Procmail ユーザーは spamassassin パッケージで利用可能な SpamAssassin という名前のサードパーティーのソフトウェアをインストールできます。SpamAssassin は、受信メールのスパムを識別するために様々な方法を使用するスパム検出システムです。Spamassassin のインストール、設定、およびデプロイメントに関する詳細は、「spam フィルター」または Red Hat ナレッジベースの記事 Spamassassin で、サーバー上のすべての受信メールにフィルターを設定する方法 を参照してください。Red Hat ナレッジベースの記事。SpamAssassin の詳細については、SpamAssassin プロジェクトの Web サイトを参照してください。

警告

SpamAssasin はサードパーティーのソフトウェアのため、Red Hat ではサポートしていない点に注意してください。

spamassassin パッケージは、Extra Packages for Enterprise Linux (EPEL) リポジトリーでのみ利用可能です。EPEL リポジトリーの使用方法の詳細は、「EPEL リポジトリーを使用したスパム対策およびウイルス対策ソフトウェアのインストール」 を参照してください。

Red Hat によるサードパーティーのソフトウェアの取り扱い方法、および Red Hat がそのようなソフトウェアに提供するサポートレベルに関する詳細は、Red Hat グローバルサポートサービスは、サードパーティーのソフトウェア、ドライバー、そして認定されていないハードウェアおよびハイパーバイザー、もしくはゲストのオペレーティングシステムについてどのようなサポートを提供していますか? を参照してください。Red Hat ナレッジベースの記事。

15.6.2. ウイルス対策保護の設定

システムのウイルス対策として ClamAV をインストールできます。ClamAV の追加情報は、ClamAV project website を参照してください。

警告

ClamAV はサードパーティーのソフトウェアであるため、Red Hat ではサポートしていない点に注意してください。

clamav パッケージ、clamav-data パッケージ、clamav-server パッケージ、および clamav-update パッケージは、Enterprise Linux (EPEL) リポジトリーの追加パッケージでのみ利用できます。EPEL リポジトリーの使用方法の詳細は、「EPEL リポジトリーを使用したスパム対策およびウイルス対策ソフトウェアのインストール」 を参照してください。

Red Hat によるサードパーティーのソフトウェアの取り扱い方法、および Red Hat がそのようなソフトウェアに提供するサポートレベルに関する詳細は、Red Hat グローバルサポートサービスは、サードパーティーのソフトウェア、ドライバー、そして認定されていないハードウェアおよびハイパーバイザー、もしくはゲストのオペレーティングシステムについてどのようなサポートを提供していますか？ を参照してください。Red Hat ナレッジベースの記事。

EPEL リポジトリーを有効にしたら、以下のコマンドを **root** ユーザーで実行し、ClamAV をインストールします。

```
~]# yum install clamav clamav-data clamav-server clamav-update
```

15.6.3. EPEL リポジトリーを使用したスパム対策およびウイルス対策ソフトウェアのインストール

EPEL は、Fedora Special Interest Group で、Red Hat Enterprise Linux 用に高品質の追加パッケージのセットを作成、維持、そして管理します。詳細は Fedora EPEL website を参照してください。

EPEL リポジトリーを使用するには、Red Hat Enterprise Linux 7 の最新番の epel-release パッケージをダウンロードします。以下のコマンドを **root** ユーザーで実行することもできます。

```
~]# yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpmzu
```

EPEL リポジトリーを初めて使用する際は、公開 GPG 鍵で認証する必要があります。詳細は、Fedora パッケージ署名鍵 を参照してください。

15.7. 関連情報

以下は、電子メールアプリケーションに関する補足のドキュメントのリストです。

15.7.1. インストールされているドキュメント

- Sendmail の設定に関する情報は、sendmail パッケージおよびsendmail-cf パッケージに含まれています。
  - **/usr/share/sendmail-cf/README: m4** マクロプロセッサーの情報、Sendmail のファイルの場所、対応するメーラー、強化機能へのアクセス方法などの情報が記載されています。

さらに、**sendmail** および **aliases** の man ページには、Sendmail の様々なオプションと、Sendmail の **/etc/mail/aliases** ファイルの適切な設定に関する役立つ情報が記載されています。

- **/usr/share/doc/postfix-**version-number : Postfix の設定方法に関する多くの情報が含まれています。version-number を Postfix のバージョン番号に置き換えてください。

- **/usr/share/doc/fetchmail-**version-number/ : Fetchmail の機能の全リストを記載した **FEATURES** ファイルと初歩的な **FAQ** ドキュメントが含まれています。version-number を、Fetchmail のバージョン番号に置き換えてください。

- **/usr/share/doc/procmail-**version-number: Procmail の概要を記載した **README** ファイル、各プログラム機能を詳細に記した **FEATURES** ファイル、設定に関する多数のよくある質問に対する回答をまとめた **FAQ** が含まれています。version-number を、Procmail のバージョン番号に置き換えてください。
  Procmail の仕組みや新しいレシピの作成方法を学習する場合は、以下にあげる Procmail の man ページが非常に役立ちます。

  - **procmail**: Procmail の仕組みと電子メールのフィルタリングに必要な手順を概説します。
  - **procmailrc**: レシピの構築に使用される **rc** のファイル形式を説明します。
  - **procmailex**: 実環境向けの役立つ Procmail のサンプルレシピを多数紹介します。
  - **procmailsc** - 特定のレシピとメッセージを適合するために Procmail で使用される加重スコアリング手法を説明します。
  - **/usr/share/doc/spamassassin-**version-number/: SpamAssassin に関する多くの情報が含まれています。version-number を、spamassassin パッケージのバージョン番号に置き換えてください。

15.7.2. オンラインドキュメント

- How to configure postfix with TLS?: postfix が TLS を使用するように設定することに関する Red Hat ナレッジベースの記事です。
- How to configure a Sendmail Smart Host: sendmail スマートホストの設定について説明している Red Hat ナレッジベースのソリューション記事です。
- http://www.sendmail.org/: Sendmail の機能に関する完全な技術詳細、ドキュメント、設定例が記載されています。
- http://www.sendmail.com/: Sendmail に関連するニュース、インタビュー、記事が掲載されており、利用可能な数多くのオプションの幅広い詳細が含まれています。
- http://www.postfix.org/: Postfix プロジェクトのホームページで、Postfix に関する豊富な情報が掲載されています。メーリングリストは、特に情報検索に役立ちます。
- http://www.fetchmail.info/fetchmail-FAQ.html: 特に Fetchmail に関する詳細な FAQ です。
- http://www.spamassassin.org/: SpamAssassin プロジェクトの公式サイトです。

15.7.3. 関連書籍

- Sendmail Milters: A Guide for Fighting Spam*(Bryan Costales、Marcia Flynt 共著、Addison-Wesley 社出版): Sendmail の優れたガイドで、メールフィルターのカスタマイズに役立ちます。*

- Sendmail *(Bryan Costales、Eric Allman 他共著、O'Reilly & Associates 社出版) - Delivermail と Sendmail の考案者のサポートを受け執筆された優れた Sendmail の参考書籍です。*

- Removing the Spam: Email Processing and Filtering*(Geoff Mulligan 著、Addison-Wesley 社出版) - Sendmail や Procmail などの確立されたツールを用いてスパム問題に対処する電子メール管理者が使用する様々な手法を考察した書籍です。*

- Internet Email Protocols: A Developer's Guide*(Kevin Johnson 著、Addison-Wesley 社出版): 主要な電子メールプロトコルと、そのプロトコルが提供するセキュリティーに関する非常に詳細なレビューをまとめています。*

- Managing IMAP *(Dianna Mullet、Kevin Mullet 共著、O'Reilly & Associates 社出版): IMAP サーバーの設定に必要な手順が詳述されています。*

第16章 ファイルとプリントサーバー

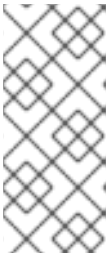
本章では、Server Message Block (SMB) およびcommon Internet file system (CIFS) プロトコルのオープンソース実装であるSamba と、Red Hat Enterprise Linux に同梱されているプライマリーFTP サーバーであるvsftpd のインストールおよび設定の方法を紹介します。また、プリンターを設定するPrint Settings ツールの使用方法についても説明します。

16.1. SAMBA

Samba は、Red Hat Enterprise Linux にサーバーメッセージブロック (SMB) プロトコルを実装します。SMB プロトコルは、ファイル共有、共有プリンターなど、サーバーのリソースにアクセスするのに使用されます。また、Samba は、Microsoft Windows が使用する分散コンピューティング環境のリモートプロシージャコール (DCE RPC) のプロトコルを実装します。

Samba は以下のように実行できます。

- Active Directory (AD) またはNT4 ドメインメンバー
- スタンドアロンサーバー
- NT4 プライマリードメインコントローラー (PDC) またはバックアップドメインコントローラー (BDC)



注記

Red Hat は、NT4 ドメインをサポートするWindows バージョンでの既存のインストールでのみ、これらのモードをサポートします。Red Hat は、新規のSamba NT4 ドメインのセットアップを推奨しません。なぜなら、Microsoft のオペレーティングシステム (Windows 7 以降) およびWindows Server 2008 R2 は、NT4 ドメインをサポートしないからです。

インストールモードとは関係なく、必要に応じてディレクトリーやプリンターを共有できます。これにより、Samba がファイルサーバーおよびプリントサーバーとして機能できるようになります。

注記

Red Hat は、Samba をAD ドメインコントローラー (DC) として実行することはサポートしていません。

16.1.1. Samba サービス

Samba は以下のサービスを提供します。

smbd

このサービスは、SMB プロトコルを使用してファイル共有およびプリントサービスを提供します。また、サービスは、リソースのロックと、接続ユーザーの認証を担当します。**smb systemd** サービスが起動し、**smbd** デーモンが停止します。
smbd サービスを使用するには、samba パッケージをインストールします。

nmbd

このサービスは、NetBIOS over IPv4 プロトコルを使用してホスト名およびIP 解決を提供します。名前解決に加え、**nmbd** サービスでSMB ネットワークを参照して、ドメイン、作業グループ、ホスト、ファイル共有、およびプリンターを探すことができます。このため、サービスはこの情報をブ

ロードキャストクライアントに直接報告するか、ローカルまたはマスターのブラウザーに転送します。**nmb systemd** サービスは、**nmbd** デーモンを起動し、停止します。
最近の SMB ネットワークは、クライアントおよび IP アドレスの解決に DNS を使用することに注意してください。

nmbd サービスを使用するには、samba パッケージをインストールします。

winbindd

winbindd サービスは、Name Service Switch (NSS) のインターフェイスを提供し、ローカルシステムで AD または NT4 ドメインユーザーおよびグループを使用します。これにより、たとえばドメインユーザーを、Samba サーバーにホストされるサービスや他のローカルサービスに認証できます。**winbind systemd** サービスは、**winbindd** デーモンを開始および停止します。
Samba をドメインメンバーとして設定する場合は、**smbd** サービスの前に **winbindd** を起動する必要があります。そうしないと、ドメインユーザーおよびグループはローカルシステムで使用できなくなります。

winbindd サービスを使用するには、samba-winbind パッケージをインストールします。

重要

Red Hat は、ドメインユーザーおよびグループをローカルシステムに提供するために、Samba を、**winbindd** サービスを使用するサーバーとして実行することのみをサポートします。Windows アクセス制御リスト (ACL) サポートの欠落や NT LAN マネージャー (NTLM) フォールバックなどの特定の制限のため、Samba でのシステムセキュリティーサービスデーモン (SSSD) の使用は現在これらのユースケースではサポートされていません。詳細は、Red Hat ナレッジベースの記事 What is the support status for Samba file server running on IdM clients or directly enrolled AD clients where SSSD is used as the client daemon を参照してください。

16.1.2. testparm ユーティリティーを使用した smb.conf ファイルの検証

testparm ユーティリティーは、**/etc/samba/smb.conf** ファイルの Samba 設定が正しいことを確認します。このユーティリティーは、無効なパラメーターおよび値を検出しますが、ID マッピングなどの間違った設定も検出します。**testparm** が問題を報告しないと、Samba サービスは **/etc/samba/smb.conf** ファイルを正常に読み込みます。**testparm** は、設定されたサービスが利用可能であること、または期待通りに機能するかを確認できないことに注意してください。

重要

Red Hat では、このファイルの変更後に毎回 **testparm** を使用して、**/etc/samba/smb.conf** ファイルを検証することが推奨されます。

/etc/samba/smb.conf ファイルを確認するには、root ユーザーとして **testparm** ユーティリティーを実行してください。**testparm** が設定内の間違ったパラメーター、値、またはその他のエラーを報告する場合は、問題を修正してから再度ユーティリティーを実行してください。

例16.1 **testparm** の使用

以下の出力は、存在しないパラメーターおよび間違った ID マッピング設定を報告しています。

```
~]# testparm
Load smb config files from /etc/samba/smb.conf
```

```
rlimit_max: increasing rlimit_max (1024) to minimum Windows limit (16384)
Unknown parameter encountered: "log levell"
Processing section "[example_share]"
Loaded services file OK.
ERROR: The idmap range for the domain * (tdb) overlaps with the range of DOMAIN (ad)!

Server role: ROLE_DOMAIN_MEMBER

Press enter to see a dump of your service definitions

# Global parameters
[global]
 ...

[example_share]
 ...
```

16.1.3. Samba のセキュリティーモードについて

/etc/samba/smb.conf ファイルの **[global]** セクションの **security** パラメーターは、Samba がサービスに接続しているユーザーを認証する方法を管理します。Samba をインストールするモードに応じて、パラメーターは異なる値に設定する必要があります。

- AD ドメインメンバーに、**security = ads** を設定します。
  このモードでは、Samba は Kerberos を使用して AD ユーザーを認証します。

  Samba をドメインメンバーとして設定する方法は、「Samba をドメインメンバーとしてセットアップ」 を参照してください。

- スタンドアロンサーバーで、**security = user** を設定する。
  このモードでは、Samba がローカルデータベースを使用して接続ユーザーを認証します。

  Samba をスタンドアロンサーバーとして設定する方法は、「Samba をスタンドアロンサーバーとして設定」 を参照してください。

- NT4 PDC または BDC に **security = user** を設定する。
  Samba は、このモードでは、ユーザーをローカルまたは LDAP データベースに認証します。

- NT4 ドメインメンバーで、**security = domain** を設定する。
  Samba は、このモードでは、NT4 PDC または BDC にユーザーを接続する認証を行います。このモードは、AD ドメインメンバーには使用できません。

  Samba をドメインメンバーとして設定する方法は、「Samba をドメインメンバーとしてセットアップ」 を参照してください。

詳細は、smb.conf(5) man ページの **security** パラメーターに関する説明を参照してください。

16.1.4. Samba をスタンドアロンサーバーとして設定

特定の状況では、管理者はドメインメンバーではない Samba サーバーのセットアップを必要とします。このインストールモードでは、Samba はユーザーを中央 DC ではなくローカルデータベースに認証します。また、ゲストアクセスを有効にして、ユーザーが、認証なしで 1 つまたは複数のサービスに接続できるようにすることもできます。

16.1.4.1. スタンドアロンサーバーのサーバー設定の設定

Samba をスタンドアロンサーバーとしてセットアップ

Samba をスタンドアロンサーバーとして設定

1. samba パッケージをインストールします。

```
~]# yum install samba
```

2. /etc/samba/smb.conf ファイルを編集して、以下のパラメーターを設定します。

```
[global]
 workgroup = Example-WG
 netbios name = Server
 security = user

 log file = /var/log/samba/%m.log
 log level = 1
```

この設定では、Example-WG ワークグループに、スタンドアロンサーバー (Server) を定義します。また、この設定により最小レベル (1) でのログ記録が可能になり、ログファイルは /var/log/samba/ ディレクトリーに保存されます。Samba は、log file パラメーターの %m マクロを、接続しているクライアントの NetBIOS 名までデプロイメントします。これにより、クライアントごとに個別のログファイルが有効になります。

詳細は、smb.conf (5) man ページのパラメーターの説明を参照してください。

3. ファイルまたはプリンターの共有を設定します。参照:

- 「Samba サーバーでのファイル共有の設定」
- 「Samba プリントサーバーの設定」

4. /etc/samba/smb.conf ファイルを検証します。

```
~]# testparm
```

詳細は、「testparm ユーティリティーを使用した smb.conf ファイルの検証」 を参照してください。

5. 認証が必要な共有を設定する場合は、ユーザーアカウントを作成します。詳細は、「ローカルユーザーアカウントの作成および有効化」 を参照してください。

6. firewall-cmd ユーティリティーを使用して必要なポートを開き、ファイアウォール設定を再読み込みします。

```
~]# firewall-cmd --permanent --add-port={139/tcp,445/tcp}
~]# firewall-cmd --reload
```

7. smb サービスを起動します。

```
~]# systemctl start smb
```

8. 必要に応じて、smb サービスがシステムの起動時に自動的に起動するようにします。

```
~]# systemctl enable smb
```

16.1.4.2. ローカルユーザーアカウントの作成および有効化

共有への接続時にユーザーが認証を行えるようにするには、オペレーティングシステムと Samba データベースの両方で Samba ホストにアカウントを作成する必要があります。Samba では、ファイルシステムオブジェクトでアクセス制御リスト (ACL) を検証するオペレーティングシステムアカウントと、接続ユーザーの認証を行う Samba アカウントが必要です。

passdb backend = tdbsam のデフォルト設定を使用すると、Samba はユーザーアカウントを **/var/lib/samba/private/passdb.tdb** データベースに保存します。

たとえば、**example** Samba ユーザーを作成するには、以下を実行します。

Samba ユーザーの作成

1. オペレーティングシステムアカウントを作成します。

   ```
   ~]# useradd -M -s /sbin/nologin example
   ```

   前述のコマンドは、ホームディレクトリーを作成することなく **example** アカウントを追加します。アカウントが Samba への認証のみに使用される場合は、**/sbin/nologin** コマンドをシェルとして割り当て、アカウントがローカルでログインしないようにします。

2. オペレーティングシステムのアカウントにパスワードを設定して、これを有効にします。

   ```
   ~]# passwd example
   Enter new UNIX password: password
   Retype new UNIX password: password
   passwd: password updated successfully
   ```

   Samba は、オペレーティングシステムのアカウントに設定されたパスワードを使用して認証を行いません。ただし、アカウントを有効にするには、パスワードを設定する必要があります。アカウントが無効になると、そのユーザーが接続した時に Samba がアクセスを拒否します。

3. Samba データベースにユーザーを追加し、そのアカウントにパスワードを設定します。

   ```
   ~]# smbpasswd -a example
   New SMB password: password
   Retype new SMB password: password
   Added user example.
   ```

   このアカウントを使用して Samba 共有に接続する場合に、このパスワードを使用して認証を行います。

4. Samba アカウントを有効にします。

   ```
   ~]# smbpasswd -e example
   Enabled user example.
   ```

16.1.5. Samba をドメインメンバーとしてセットアップ

AD または NT4 ドメインを実行する管理者は多くの場合、ドメインのメンバーとして Red Hat Enterprise Linux サーバーに参加するために Samba を使用したいと考えています。これを使用すると、以下が可能になります。

- その他のドメインメンバーのドメインリソースにアクセスする
- **sshd** などのローカルサービスに対してドメインユーザーを認証する
- サーバーにホストされているディレクトリーおよびプリンターを共有して、ファイルサーバーおよびプリントサーバーとして動作する

16.1.5.1. ドメインの参加

Red Hat Enterprise Linux システムをドメインに参加させるには、以下を実行します。

Red Hat Enterprise Linux システムの、ドメインへの参加

1. 以下のパッケージをインストールします。

```
~]# yum install realmd oddjob-mkhomedir oddjob samba-winbind-clients \
    samba-winbind samba-common-tools
```

2. ドメインメンバーでディレクトリーまたはプリンターを共有するには、samba パッケージをインストールします。

```
~]# yum install samba
```

3. AD に参加する場合は、追加で samba-winbind-krb5-locator パッケージをインストールします。

```
~]# yum install samba-winbind-krb5-locator
```

このプラグインを使用すると、Kerberos は DNS サービスレコードを使用して、AD サイトに基づいて鍵配布センター (KDC) を探すことができます。

4. あるいは、既存の **/etc/samba/smb.conf** Samba 設定ファイルの名前を変更します。

```
~]# mv /etc/samba/smb.conf /etc/samba/smb.conf.old
```

5. ドメインに参加します。たとえば、ドメイン **ad.example.com** に参加するには、以下のコマンドを実行します。

```
~]# realm join --membership-software=samba --client-software=winbind
ad.example.com
```

上記のコマンドを使用すると、**realm** ユーティリティーが自動的に以下を実行します。

- **ad.example.com** ドメインのメンバーシップに **/etc/samba/smb.conf** ファイルを作成します。
- ユーザーおよびグループの検索用の **winbind** モジュールを、**/etc/nsswitch.conf** ファイルに追加します。

- */etc/pam.d/* ディレクトリーの PAM (プラグ可能な認証モジュール) 設定ファイルを更新します。
- **winbind** サービスを起動し、システムの起動時にサービスを起動できるようにします。**realm** ユーティリティーに関する詳細は、*realm(8)* の man ページと、Red Hat Windows Integration Guide の関連のセクションを参照してください。

6. 必要に応じて、**/etc/samba/smb.conf** ファイルの別の ID マッピングバックエンド、またはカスタマイズした ID マッピングを設定します。詳細は、「ID マッピングについて」 を参照してください。
7. 必要に応じて設定を検証します。「Samba がドメインメンバーとして正しく参加されたことを確認」を参照してください。
8. **winbindd** が実行していることを確認します。

```
~]# systemctl status winbind
```

重要

Samba を有効にして、ドメインユーザーおよびグループ情報をクエリーするには、**smbd** を起動する前に **winbindd** サービスを起動する必要があります。

9. samba パッケージをインストールしてディレクトリーおよびプリンターを共有する場合は、**smbd** サービスを起動します。

```
~]# systemctl start smb
```

16.1.5.2. Samba がドメインメンバーとして正しく参加されたことを確認

ドメインメンバーとして Red Hat Enterprise Linux に参加した後、様々なテストを実行して正常に参加できたことを確認します。参照:

- 「オペレーティングシステムが、ドメインのユーザーアカウントおよびグループを取得できるかどうかの検証」
- 「AD ドメインユーザーが Kerberos チケットを取得できるかどうかの確認」
- 「利用可能なドメインのリスト表示」

オペレーティングシステムが、ドメインのユーザーアカウントおよびグループを取得できるかどうかの検証
getent ユーティリティーを使用して、オペレーティングシステムがドメインユーザーおよびグループを取得できることを確認します。以下に例を示します。

- AD ドメインの **administrator** アカウントをクエリーするには、以下を実行します。

```
~]# getent passwd AD\\administrator
AD\administrator:*:10000:10000::/home/administrator@AD:/bin/bash
```

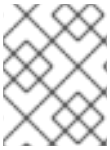
- AD ドメイン内の **Domain Users** グループのメンバーにクエリーするには、以下を実行します。

```
~]# getent group "AD\\Domain Users"
AD\domain users:x:10000:user
```

コマンドが正常に機能する場合は、ファイルおよびディレクトリーに権限を設定する際にドメインユーザーとグループを使用できるかを確認してください。たとえば、**/srv/samba/example.txt** ファイルの所有者を **AD\administrator** に設定し、グループを **AD\Domain Users** に設定するには、以下のコマンドを実行します。

```
~]# chown "AD\administrator":"AD\Domain Users" /srv/samba/example.txt
```

AD ドメインユーザーが Kerberos チケットを取得できるかどうかの確認
AD 環境では、DC から Kerberos チケットを取得できます。たとえば、**administrator** ユーザーが Kerberos チケットを取得できるかどうかを確認するには、以下を実行してください。



注記

kinit ユーティリティーおよび**klist** ユーティリティーを使用するには、Samba ドメインメンバー上の**krb5-workstation** パッケージをインストールします。

Kerberos チケットの取得

1. **administrator@AD.EXAMPLE.COM** プリンシパルのチケットを取得します。

```
~]# kinit administrator@AD.EXAMPLE.COM
```

2. キャッシュされた Kerberos チケットを表示します。

```
~]# klist
Ticket cache: KEYRING:persistent:0:0
Default principal: administrator@AD.EXAMPLE.COM

Valid starting     Expires        Service principal
11.09.2017 14:46:21 12.09.2017 00:46:21
krbtgt/AD.EXAMPLE.COM@AD.EXAMPLE.COM
 renew until 18.09.2017 14:46:19
```

利用可能なドメインのリスト表示
winbindd サービスで利用可能なすべてのドメインをリスト表示するには、以下を入力します。

```
~]# wbinfo --all-domains
```

Samba がドメインメンバーとして適切に参加すると、このコマンドは組み込みおよびローカルのホスト名を表示し、ドメインの Samba は信頼されるドメインを含むメンバーとなります。

例16.2 利用可能なドメインの表示

```
~]# wbinfo --all-domains
BUILTIN
SAMBA-SERVER
AD
```

16.1.5.3. ID マッピングについて

Windows ドメインは、ユーザーおよびグループを一意のセキュリティー識別子 (SID) で区別します。ただし、Linux では、ユーザーおよびグループごとに一意の UID と GID が必要です。Samba をドメインメンバーとして実行する場合は、**winbindd** サービスが、ドメインユーザーおよびグループに関する情報をオペレーティングシステムに提供します。

winbindd サービスが、ユーザーおよびグループの一意の ID を Linux に提供するようにするには、**/etc/samba/smb.conf** ファイルで ID マッピングを設定する必要があります。

- ローカルデータベース (デフォルトドメイン)
- Samba サーバーがメンバーになっている AD または NT4 のドメイン
- ユーザーがこの Samba サーバーのリソースにアクセスする必要のある信頼ドメイン

16.1.5.3.1. ID の範囲の計画

Linux の UID および GID を AD に保存するか、Samba がそれを生成するように設定するかに関係なく、各ドメイン設定には、他のドメインと重複**しない**一意の ID 範囲が必要です。

警告

重複する ID 範囲を設定すると、Samba が正常に機能しなくなります。

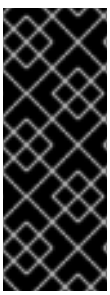
例16.3 一意の ID 範囲

以下は、デフォルト (*)、**AD-DOM**、および **TRUST-DOM** のドメインの非オーバーランディングの ID マッピング範囲を示しています。

```
[global]
...
idmap config * : backend = tdb
idmap config * : range = 10000-999999

idmap config AD-DOM:backend = rid
idmap config AD-DOM:range = 2000000-2999999

idmap config TRUST-DOM:backend = rid
idmap config TRUST-DOM:range = 4000000-4999999
```



重要

1 つのドメインに割り当てられるのは 1 つの範囲だけです。したがって、ドメイン範囲間で十分な容量を残しておきます。これにより、ドメインが拡大した場合に、後で範囲を拡張できます。

後で別の範囲をドメインに割り当てると、このユーザーおよびグループが作成したファイルおよびディレクトリーの所有権が失われます。

16.1.5.3.2. * デフォルトドメイン

ドメイン環境では、以下の各 ID マッピング設定を追加します。

- Samba サーバーがメンバーとなっているドメイン
- Samba サーバーにアクセスできる信頼された各ドメイン

ただし、Samba が、その他のすべてのオブジェクトに、デフォルトドメインから ID を割り当てます。これには以下が含まれます。

- ローカルの Samba ユーザーおよびグループ
- Samba の組み込みアカウントおよびグループ (**BUILTIN\Administrators** など)

重要

Samba が正常に機能できるようにするには、このセクションで説明されているデフォルトのドメインを設定する必要があります。

割り当てられた ID を永続的に格納するには、デフォルトのドメインバックエンドを書き込み可能にする必要があります。

デフォルトドメインには、以下のいずれかのバックエンドを使用できます。

tdb

デフォルトのドメインを、**tdb** バックエンドを使用するように設定する場合は、ID 範囲を設定します。この ID 範囲には、将来作成されるオブジェクトや、定義されたドメイン ID マッピング設定には含まれないオブジェクトを追加できます。
たとえば、**/etc/samba/smb.conf** ファイルの **[global]** セクションで以下を設定します。

```
idmap config * : backend = tdb
idmap config * : range = 10000-999999
```

詳細は、「**tdb** ID マッピングバックエンドの使用」 を参照してください。

autorid

autorid バックエンドを使用するように、デフォルトのドメインを設定する場合、ドメイン用の ID マッピング設定を追加するかどうかは任意になります。
たとえば、**/etc/samba/smb.conf** ファイルの **[global]** セクションで以下を設定します。

```
idmap config * : backend = autorid
idmap config * : range = 10000-999999
```

詳細は、**autorid** バックエンドの設定 を参照してください。

16.1.5.4. 様々な ID マッピングバックエンド

Samba は、特定の設定に対して異なる ID マッピングバックエンドを提供します。最も頻繁に使用されるバックエンドは、以下の通りです。

表16.1 よく使われる ID マッピングバックエンド

バックエンド	ユースケース
tdb	* デフォルトドメインのみ
ad	AD ドメインのみ
rid	AD ドメインおよび NT4 ドメイン
autorid	AD、NT4、および * デフォルトのドメイン

以下のセクションでは、利点、バックエンドを使用する際の推奨シナリオ、および設定方法を説明します。

16.1.5.4.1. tdb ID マッピングバックエンドの使用

***winbindd** サービスは、デフォルトで書き込み可能な **tdb ID** マッピングバックエンドを使用して、セキュリティー識別子 (SID)、UID、および GID のマッピングテーブルを格納します。これには、ローカルユーザー、グループ、組み込みプリンシパルが含まれます。*

このバックエンドは、 デフォルトドメインにのみ使用してください。以下に例を示します。*

```
idmap config * : backend = tdb
idmap config * : range = 10000-999999
```

* デフォルトドメインに関する詳細は、「* デフォルトドメイン」 を参照してください。

16.1.5.4.2. ad ID マッピングバックエンドの使用

***ad** ID マッピングバックエンドは、読み取り専用 API を実装し、AD からアカウントおよびグループの情報を読み取ります。これには、以下の利点があります。*

- *ユーザーとグループの全設定は、AD に集中的に保存されます。*
- *ユーザーおよびグループの ID は、このバックエンドを使用するすべての Samba サーバーで一貫しています。*
- *ID は、破損する可能性のあるローカルデータベースには保存されないため、ファイルの所有権は失われません。*

***ad** バックエンドは、AD から以下の属性を読み込みます。*

*表16.2 **ad** バックエンドが、ユーザーおよびグループオブジェクトから読み取る属性*

AD 属性名	オブジェクトタイプ	マッピング先
sAMAccountName	ユーザーおよびグループ	オブジェクトのユーザー名またはグループ名
uidNumber	ユーザー	ユーザー ID (UID)
gidNumber	グループ	グループ ID (GID)

AD 属性名	オブジェクトタイプ	マッピング先
loginShell [a]	ユーザー	ユーザーのシェルのパス
unixHomeDirectory	ユーザー	ユーザーのホームディレクトリーのパス
primaryGroupID [b]	ユーザー	プライマリーグループ ID

[a] **idmap config DOMAIN:unix_nss_info = yes** を設定している場合に限り、Samba がこの属性を読み込みます。

[b] **idmap config DOMAIN:unix_primary_group = yes** を設定している場合に限り、Samba がこの属性を読み込みます。

ad バックエンドの前提条件

ad ID マッピングバックエンドを使用する場合:

- *ユーザーおよびグループはいずれも、AD で一意の ID が設定され、ID が、/etc/samba/smb.conf ファイルで設定されている範囲内にある。ID が範囲外にあるオブジェクトは、Samba サーバーでは利用できません。*
- *ユーザーおよびグループには、AD ですべての必須属性が設定されている。必要な属性がないと、ユーザーまたはグループは Samba サーバーで使用できなくなります。必要な属性は、設定によって異なります。表16.2「ad バックエンドが、ユーザーおよびグループオブジェクトから読み取る属性」を参照してください。*

ad バックエンドの設定

ad ID マッピングバックエンドを使用するために Samba AD メンバーを設定するには、以下に該当する必要があります。

ドメインメンバー上での ad バックエンドの設定

1. */etc/samba/smb.conf ファイルの [global] セクションを編集します。*
   a. *デフォルトドメイン (*) に ID マッピング設定が存在しない場合は追加します。以下に例を示します。*

   ```
   idmap config * : backend = tdb
   idmap config * : range = 10000-999999
   ```

   デフォルトドメインに関する詳細は 「 デフォルトドメイン」 を参照してください。*

   b. *AD ドメインの ad ID マッピングバックエンドを有効にします。*

   ```
   idmap config DOMAIN : backend = ad
   ```

   c. *AD ドメインのユーザーおよびグループに割り当てられている ID の範囲を設定します。以下に例を示します。*

   ```
   idmap config DOMAIN : range = 2000000-2999999
   ```

重要

この範囲は、このサーバーの他のドメイン設定と重複させることはできません。また、この範囲には、今後割り当てられる ID がすべて収まる大きさを設定する必要があります。詳細は、「ID の範囲の計画」を参照してください。

d. *Samba が AD から属性を読み取る際に RFC 2307 スキーマを使用するように設定します。*

```
idmap config DOMAIN : schema_mode = rfc2307
```

e. *Samba が、対応する AD 属性からログインシェルおよびユーザーホームディレクトリーのパスを読み取るようにする場合は、以下を設定します。*

```
idmap config DOMAIN : unix_nss_info = yes
```

または、すべてのユーザーに適用される、ドメイン全体のホームディレクトリーのパスおよびログインシェルを統一して設定できます。以下に例を示します。

```
template shell = /bin/bash
template homedir = /home/%U
```

*変数の置換の詳細は、smb.conf(5) man ページの*VARIABLE SUBSTITUTIONS *セクションを参照してください。*

f. *デフォルトでは、Samba は、ユーザーオブジェクトの **primaryGroupID** 属性を、Linux のユーザーのプライマリーグループとして使用します。または、代わりに **gidNumber** 属性に設定されている値を使用するように Samba を設定できます。*

```
idmap config DOMAIN : unix_primary_group = yes
```

2. */etc/samba/smb.conf ファイルを検証します。*

```
~]# testparm
```

*詳細は、「**testparm** ユーティリティーを使用した **smb.conf** ファイルの検証」を参照してください。*

3. *Samba 設定を再読み込みします。*

```
~]# smbcontrol all reload-config
```

4. *設定が期待どおりに機能することを確認します。「オペレーティングシステムが、ドメインのユーザーアカウントおよびグループを取得できるかどうかの検証」を参照してください。*

詳細は、smb.conf(5) と idmap_ad(8) の man ページを参照してください。

*16.1.5.4.3. **rid** ID マッピングバックエンドの使用*

Samba は、Windows SID の相対識別子 (RID) を使用して、Red Hat Enterprise Linux で ID を生成できます。

注記

*RID は、SID の最後の部分です。たとえば、ユーザーの SID が **S-1-5-21-5421822485-1151247151-421485315-30014** の場合、対応する RID は **30014** になります。Samba のローカル ID の算出方法は、idmap_rid(8) man ページを参照してください。*

***rid** ID マッピングバックエンドは、AD ドメインおよび NT4 ドメインのアルゴリズムマッピングスキームに基づいてアカウントおよびグループの情報を計算する読み取り専用 API を実装します。バックエンドを設定する場合は、**idmap config** DOMAIN**: range** パラメーターで、RID の最小値および最大値を設定する必要があります。Samba は、このパラメーターで設定される RID の最小値および最大値を超えるユーザーまたはグループをマッピングしません。*

重要

*読み取り専用のバックエンドとして、**rid** は、**BUILTIN** グループなど、新しい ID を割り当てることができません。したがって、* デフォルトドメインにはこのバックエンドを使用しないでください。*

利点

- *設定された範囲内の RID があるドメインユーザーとグループはすべて、自動的にドメインメンバーで利用可能になります。*
- *ID、ホームディレクトリー、およびログインシェルを手動で割り当てる必要はありません。*

短所

- *すべてのドメインユーザーは、割り当てられた同じログインシェルとホームディレクトリーを取得します。ただし、変数を使用できます。*
- *同じ ID 範囲設定で **rid** バックエンドを使用している Samba ドメインメンバーでは、ユーザー ID とグループ ID が同じになります。*
- *ドメインメンバーで個々のユーザーまたはグループを除外して、利用できないようにすることはできません。設定されている範囲外にあるユーザーとグループのみが除外されます。*
- *異なるドメインのオブジェクトの RID が同じ場合は、**winbindd** サービスが ID の計算に使用する式に基づき、複数ドメインの環境で重複する ID が発生する場合があります。*

rid バックエンドの設定

*Samba ドメインメンバーが **rid** ID マッピングバックエンドを使用するように設定するには、以下が必要です。*

*ドメインメンバー上での **rid** バックエンドの設定*

1. */etc/samba/smb.conf* ファイルの **[global]** セクションを編集します。
   a. *デフォルトドメイン (*) に ID マッピング設定が存在しない場合は追加します。以下に例を示します。*

```
idmap config * : backend = tdb
idmap config * : range = 10000-999999
```

   デフォルトドメインに関する詳細は 「 デフォルトドメイン」 を参照してください。*

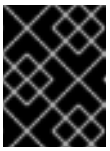
b. ドメインの **rid** ID マッピングバックエンドを有効にします。

```
idmap config DOMAIN : backend = rid
```

c. 今後割り当てられるすべての RID が収まる大きさの範囲を設定します。以下に例を示します。

```
idmap config DOMAIN : range = 2000000-2999999
```

Samba は、そのドメインの RID がその範囲内にないユーザーおよびグループを無視します。



重要

この範囲は、このサーバーの他のドメイン設定と重複させることはできません。詳細は、「ID の範囲の計画」 を参照してください。

d. すべてのマッピングユーザーに割り当てられるシェルおよびホームディレクトリーのパスを設定します。以下に例を示します。

```
template shell = /bin/bash
template homedir = /home/%U
```

変数の置換の詳細は、smb.conf(5) man ページの VARIABLE SUBSTITUTIONS セクションを参照してください。

2. **/etc/samba/smb.conf** ファイルを検証します。

```
~]# testparm
```

詳細は、「**testparm** ユーティリティーを使用した **smb.conf** ファイルの検証」 を参照してください。

3. Samba 設定を再読み込みします。

```
~]# smbcontrol all reload-config
```

4. 設定が期待どおりに機能することを確認します。「オペレーティングシステムが、ドメインのユーザーアカウントおよびグループを取得できるかどうかの検証」を参照してください。

16.1.5.4.4. autorid ID マッピングバックエンドの使用

autorid バックエンドは、**rid** ID マッピングバックエンドと同様の動作をしますが、異なるドメインに対して自動的に ID を割り当てることができます。これにより、以下の状況で **autorid** バックエンドを使用できます。

- * デフォルトドメインのみ
- * デフォルトドメインと追加のドメインでは、追加のドメインごとに ID マッピング設定を作成する必要はありません。
- 特定のドメインのみ

利点

- *設定された範囲内に計算した UID と GID があるすべてのドメインユーザーおよびグループは、ドメインメンバーで自動的に利用可能になります。*
- *ID、ホームディレクトリー、およびログインシェルを手動で割り当てる必要はありません。*
- *複数ドメイン環境内の複数のオブジェクトが同じ RID を持つ場合でも、重複する ID はありません。*

短所

- *Samba ドメインメンバー間では、ユーザー ID とグループ ID は同じではありません。*
- *すべてのドメインユーザーは、割り当てられた同じログインシェルとホームディレクトリーを取得します。ただし、変数を使用できます。*
- *ドメインメンバーで個々のユーザーまたはグループを除外して、利用できないようにすることはできません。計算された UID または GID が、設定された範囲外にあるユーザーとグループのみが除外されます。*

autorid バックエンドの設定

** デフォルトドメイン向けに autorid ID マッピングバックエンドを使用するために Samba ドメインメンバーを設定するには、以下を実行します。*

注記

デフォルトドメインに autorid を使用する場合は、ドメイン用の ID マッピング設定を追加するかどうかは任意です。

ドメインメンバー上での autorid バックエンドの設定

1. */etc/samba/smb.conf ファイルの [global] セクションを編集します。*
   a. ** デフォルトドメインの autorid ID マッピングバックエンドを有効にします。*

      ```
      idmap config * : backend = autorid
      ```

   b. *既存および将来の全オブジェクトに ID を割り当てられる大きさの範囲を設定します。以下に例を示します。*

      ```
      idmap config * : range = 10000-999999
      ```

      *Samba は、このドメインで計算した ID が範囲内にないユーザーおよびグループを無視します。バックエンドの計算された ID の詳細は、idmap_autorid(8) man ページの*THE MAPPING FORMULAS *セクションを参照してください。*

警告

範囲を設定し、Samba がそれを使用して開始してからは、範囲の上限を小さくすることはできません。これ以外の範囲への変更は、新しい ID の割り当てとなり、その結果ファイルの所有権を失うことになります。

c. 必要に応じて、範囲サイズを設定します。以下に例を示します。

```
idmap config * : rangesize = 200000
```

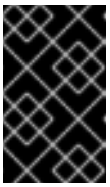
Samba は、**idmap config * : range** パラメーターに設定されている範囲からすべての ID を取得するまで、各ドメインのオブジェクトにこの数の連続 ID を割り当てます。詳細は、idmap_autorid(8) man ページの **rangesize** パラメーターの説明を参照してください。

d. すべてのマッピングユーザーに割り当てられるシェルおよびホームディレクトリーのパスを設定します。以下に例を示します。

```
template shell = /bin/bash
template homedir = /home/%U
```

変数の置換の詳細は、smb.conf(5) man ページの VARIABLE SUBSTITUTIONS セクションを参照してください。

e. 必要に応じて、ドメイン用の ID マッピング設定を追加します。個別のドメインの設定が利用できない場合、Samba は以前に設定した * デフォルトドメインの **autorid** バックエンド設定を使用して ID を計算します。



重要

各ドメインに追加のバックエンドを設定する場合は、すべての ID マッピング設定の範囲がオーバーラップしないようにしてください。詳細は、「ID の範囲の計画」を参照してください。

2. **/etc/samba/smb.conf** ファイルを検証します。

```
~]# testparm
```

詳細は、「**testparm** ユーティリティーを使用した **smb.conf** ファイルの検証」を参照してください。

3. Samba 設定を再読み込みします。

```
~]# smbcontrol all reload-config
```

4. 設定が期待どおりに機能することを確認します。「オペレーティングシステムが、ドメインのユーザーアカウントおよびグループを取得できるかどうかの検証」を参照してください。

16.1.6. Samba サーバーでのファイル共有の設定

Samba をファイルサーバーとして使用する場合は、スタンドアロンまたはドメインメンバー設定の **/etc/samba/smb.conf** ファイルに共有を追加します。

以下のいずれかを使用する共有を追加できます。

- POSIX ACL。「POSIX ACL で共有の設定」を参照してください。
- きめ細かな Windows ACL。「Windows ACL で共有の設定」を参照してください。

16.1.6.1. POSIX ACL で共有の設定

Samba は、Linux サービスとして、POSIX ACL との共有に対応します。**chmod** などのユーティリティーを使用して、Samba サーバーの権限をローカルに管理できます。拡張属性に対応するファイルシステムに共有が保存されている場合は、複数のユーザーおよびグループで ACL を定義できます。

注記

代わりにきめ細かな Windows ACL を使用する必要がある場合は、「Windows ACL で共有の設定」を参照してください。

共有を追加する前に、Samba をセットアップします。参照:

- 「Samba をスタンドアロンサーバーとして設定」
- 「Samba をドメインメンバーとしてセットアップ」

16.1.6.1.1. POSIX ACL を使用する共有の追加

/srv/samba/example/ ディレクトリーのコンテンツを提供し、POSIX ACL を使用する**example** という名前の共有を作成するには、以下を実行します。

POSIX ACL を使用する共有の追加

1. または、フォルダーが存在しない場合は作成します。以下に例を示します。

```
~]# mkdir -p /srv/samba/example/
```

2. SELinux を、**enforcing** モードで実行する場合は、そのディレクトリーに**samba_share_t** コンテキストを設定します。

```
~]# semanage fcontext -a -t samba_share_t "/srv/samba/example(/.*)?"
~]# restorecon -Rv /srv/samba/example/
```

3. ディレクトリーにファイルシステムの ACL を設定します。詳細は、「ACL の設定」を参照してください。

4. **/etc/samba/smb.conf** ファイルに**example** 共有を追加します。たとえば、共有の **write-enabled** を追加するには、次のコマンドを実行します。

```
[example]
 path = /srv/samba/example/
 read only = no
```

注記

*ファイルシステムの ACL に関係なく、**read only = no** を設定しないと、Samba がディレクトリーを読み取り専用モードで共有します。*

5. */etc/samba/smb.conf ファイルを検証します。*

```
~]# testparm
```

*詳細は、「**testparm** ユーティリティーを使用した **smb.conf** ファイルの検証」を参照してください。*

6. ***firewall-cmd** ユーティリティーを使用して必要なポートを開き、ファイアウォール設定を再読み込みします。*

```
~]# firewall-cmd --permanent --add-service=samba
~]# firewall-cmd --reload
```

7. ***smb** サービスを再起動します。*

```
~]# systemctl restart smb
```

8. *必要に応じて、システムの起動時に、**smb** サービスが自動的に起動するようにします。*

```
~]# systemctl enable smb
```

16.1.6.1.2. ACL の設定

POSIX ACL を使用する共有は、以下に対応します。

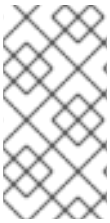
- *標準的な Linux ACL。詳細は、標準の Linux ACL の設定を参照してください。*
- *拡張 ACL。詳細は、拡張 ACL の設定を参照してください。*

標準の Linux ACL の設定

*Linux の標準 ACL は、所有者、グループ、その他の未定義ユーザーの権限の設定に対応します。ユーティリティーの **chown**、**chgrp**、および **chmod** を使用して ACL を更新できます。正確な制御が必要な場合は、より複雑な POSIX ACL を使用します。拡張 ACL の設定を参照してください。*

*たとえば、**/srv/samba/example/** ディレクトリーのオーナーを **root** ユーザーに設定するには、**Domain Users** グループに読み取りと書き込みのパーミッションを与え、その他のすべてのユーザーへのアクセスを拒否します。*

```
~]# chown root:"Domain Users" /srv/samba/example/
~]# chmod 2770 /srv/samba/example/
```



注記

ディレクトリーで set-group-ID (SGID) ビットを有効にすると、新しいディレクトリーエントリーを作成したユーザーのプライマリーグループに設定する通常の動作の代わりに、すべての新しいファイルとサブディレクトリーのデフォルトグループが、そのディレクトリーグループのデフォルトグループに自動的に設定されます。

権限の詳細は、chown (1) およびchmod (1) man ページを参照してください。

拡張 ACL の設定

共有ディレクトリーが保存されているファイルシステムが拡張ACL に対応している場合は、それを使用して複雑な権限を設定できます。拡張ACL には、複数のユーザーおよびグループの権限を指定できます。

拡張POSIX ACL を使用すると、複数のユーザーおよびグループで複雑なACL を設定できます。ただし、設定できるのは以下の権限のみです。

- アクセスなし
- 読み取りアクセス
- 書き込みアクセス
- 完全な制御

Create folder / append data など、詳細なWindows 権限が必要な場合は、Windows ACL を使用するように共有を設定します。「Windows ACL で共有の設定」を参照してください。

共有で拡張POSIX ACL を使用するには、以下を実行します。

共有における拡張POSIX ACL の有効化

1. **/etc/samba/smb.conf** ファイルの共有セクションで以下のパラメーターを有効にして、拡張ACL のACL 継承を有効にします。

   ```
   inherit acls = yes
   ```

   詳細は、smb.conf(5) man ページのパラメーターの説明を参照してください。

2. **smb** サービスを再起動します。

   ```
   ~]# systemctl restart smb
   ```

3. 必要に応じて、システムの起動時に、**smb** サービスが自動的に起動するようにします。

   ```
   ~]# systemctl enable smb
   ```

4. ディレクトリーのACL を設定します。拡張ACL の使用に関する詳細は5章アクセス制御リストを参照してください。

   例16.4 拡張ACL の設定

   以下の手順は、**/srv/samba/example/** ディレクトリーに対して、**Domain Admins** グループに読み取り、書き込み、および実行の権限、**Domain Users** グループに対する読み取りおよび実行の権限を設定し、その他の全員のアクセスを拒否します。

   拡張ACL の設定

   a. ユーザーアカウントのプライマリーグループへの自動許可権限を無効にします。

      ```
      ~]# setfacl -m group::--- /srv/samba/example/
      ~]# setfacl -m default:group::--- /srv/samba/example/
      ```

*ディレクトリーのプライマリーグループは、さらに動的な **CREATOR GROUP** プリンシパルにマッピングされます。Samba 共有で拡張 POSIX ACL を使用すると、このプリンシパルは自動的に追加され、削除できません。*

b. *ディレクトリーに権限を設定します。*

   i. ***Domain Admins** グループに読み取り、書き込み、および実行の権限を付与します。*

   ```
   ~]# setfacl -m group:"DOMAIN\Domain Admins":rwx /srv/samba/example/
   ```

   ii. ***Domain Users** グループに読み取りおよび実行の権限を付与します。*

   ```
   ~]# setfacl -m group:"DOMAIN\Domain Users":r-x /srv/samba/example/
   ```

   iii. *その他 の ACL エントリーに権限を設定し、その他の ACL エントリーに一致しないユーザーへのアクセスを拒否します。*

   ```
   ~]# setfacl -R -m other::--- /srv/samba/example/
   ```

   この設定は、このディレクトリーにのみ適用されます。Windows では、これらの ACL は このフォルダーのみ のモードにマッピングされます。

c. *前の手順で設定した権限を、このディレクトリーに作成した新規ファイルシステムのオブジェクトから継承できるようにするには、以下のコマンドを実行します。*

```
~]# setfacl -m default:group:"DOMAIN\Domain Admins":rwx
/srv/samba/example/
~]# setfacl -m default:group:"DOMAIN\Domain Users":r-x /srv/samba/example/
~]# setfacl -m default:other::--- /srv/samba/example/
```

この設定では、プリンシパルの このフォルダーのみ モードが、このフォルダー、サブフォルダー、およびファイル に設定されます。

Samba は、以前設定したパーミッションを以下の Windows ACL にマッピングします。

プリンシパル	アクセス	適用先
DOMAIN\Domain Admins	完全な制御	このフォルダー、サブフォルダー、およびファイル
DOMAIN\Domain Users	読み取りおよび実行	このフォルダー、サブフォルダー、およびファイル
Everyone [a]	なし	このフォルダー、サブフォルダー、およびファイル
所有者 (Unix Userpass:attributes[]**owner**) [b]	完全な制御	このフォルダーのみ

プリンシパル	アクセス	適用先
primary_group (Unix Userpass:attributes[]**primary _group**) [c]	なし	このフォルダーのみ
CREATOR OWNER [d] [e]	完全な制御	サブフォルダーおよびファイルのみ
CREATOR GROUP [f]	なし	サブフォルダーおよびファイルのみ

[a] Samba は、このプリンシパルの権限を **その他** の ACL エントリーからマッピングします。

[b] Samba は、ディレクトリーの所有者をこのエントリーにマッピングします。

[c] Samba は、ディレクトリーのプライマリーグループをこのエントリーにマッピングします。

[d] 新規ファイルシステムオブジェクトでは、作成者はこのプリンシパルの権限を自動的に継承します。

[e] POSIX ACL を使用する共有では、このプリンシパルの設定または削除には対応していません。

[f] 新規ファイルシステムオブジェクトの場合、作成者のプライマリーグループは、自動的にこのプリンシパルの権限を継承します。

16.1.6.1.3. 共有でパーミッションを設定

*必要に応じて、Samba 共有へのアクセスを制限または許可するには、**/etc/samba/smb.conf** ファイルの共有のセクションに特定のパラメーターを設定します。*

注記

*共有ベースの権限は、ユーザー、グループ、またはホストが共有にアクセスできるかどうかを管理します。この設定は、ファイルシステムの **ACL** には影響しません。*

共有へのアクセスを制限するには、共有ベースの設定を使用します。たとえば、特定のホストからのアクセスを拒否する場合などです。

ユーザーおよびグループに基づいた共有アクセスの設定

*ユーザーおよびグループに基づいたアクセス制御により、特定のユーザーおよびグループの共有へのアクセスを許可または拒否できます。たとえば、**Domain Users** グループの全メンバーが、ユーザー アカウントのアクセスが拒否されている時に共有にアクセスできるようにするには、共有の設定に以下のパラメーターを追加します。*

```
valid users = +DOMAIN\"Domain Users"
invalid users = DOMAIN\user
```

invalid users パラメーターは、valid users パラメーターよりも優先度が高いです。たとえば、ユ

invalid users パラメーターは、**valid users** パラメーターよりも優先度が高いです。たとえば、ユーザー アカウントが **Domain Users** グループのメンバーである場合に上述の例を使用すると、このアカウントへのアクセスは拒否されます。

詳細は、smb.conf (5) man ページのパラメーターの説明を参照してください。

ホストベースの共有アクセスの設定

ホストベースのアクセス制御により、クライアントのホスト名、IP アドレス、または IP 範囲に基づいて、共有へのアクセスを許可または拒否できます。

127.0.0.1 IP アドレス、**192.0.2.0/24** IP 範囲、および **client1.example.com** ホストを有効にし、共有にアクセスして **client2.example.com** ホストへのアクセスを追加的に拒否するには、以下を実行します。

ホストベースの共有アクセスの設定

1. **/etc/samba/smb.conf** における共有の設定に以下のパラメーターを追加します。

```
hosts allow = 127.0.0.1 192.0.2.0/24 client1.example.com
hosts deny = client2.example.com
```

2. Samba 設定を再読み込みします。

```
~]# smbcontrol all reload-config
```

hosts deny パラメーターは、**hosts allow** よりも優先順位が高くなります。たとえば、**client1.example.com** が **hosts allow** パラメーターにリスト表示されている IP アドレスに解決すると、このホストへのアクセスは拒否されます。

詳細は、smb.conf(5) man ページのパラメーターに関する説明を参照してください。

16.1.6.2. Windows ACL で共有の設定

Samba は、共有およびファイルシステムオブジェクトへの Windows ACL の設定に対応します。これを使用すると、以下が可能になります。

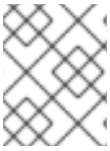
- きめ細かな Windows ACL を使用する
- Windows を使用して共有権限およびファイルシステムの ACL を管理する

または、POSIX ACL を使用するように共有を設定することもできます。詳細は、「POSIX ACL で共有の設定」 を参照してください。

16.1.6.2.1. SeDiskOperatorPrivilege 特権の付与

Windows ACL を使用する共有に対する権限を設定できるのは、**SeDiskOperatorPrivilege** 特権が付与されているユーザーおよびグループのみです。たとえば、**DOMAIN\Domain Admins** グループに特権を付与する場合は、以下を実行します。

```
~]# net rpc rights grant "DOMAIN\Domain Admins" SeDiskOperatorPrivilege \
    -U "DOMAIN\administrator"
Enter DOMAIN\administrator's password:
Successfully granted rights.
```



注記

*ドメイン環境では、**SeDiskOperatorPrivilege** をドメイングループに付与します。これにより、ユーザーのグループメンバーシップを更新し、権限を集中的に管理できます。*

***SeDiskOperatorPrivilege** が付与されているすべてのユーザーおよびグループをリスト表示するには、以下のコマンドを実行します。*

```
~]# net rpc rights list privileges SeDiskOperatorPrivilege \
   -U "DOMAIN\administrator"
Enter administrator's password:
SeDiskOperatorPrivilege:
 BUILTIN\Administrators
 DOMAIN\Domain Admins
```

16.1.6.2.2. Windows ACL サポートの有効化

*Windows ACL に対応する共有を設定するには、Samba でこの機能を有効にする必要があります。すべての共有に対してグローバルに有効にするには、次の設定を **/etc/samba/smb.conf** ファイルの **[global]** セクションに追加します。*

```
vfs objects = acl_xattr
map acl inherit = yes
store dos attributes = yes
```

または、共有のセクションに同じパラメーターを追加して、個別の共有に対して Windows ACL サポートを有効にできます。

16.1.6.2.3. Windows ACL を使用する共有の追加

***/srv/samba/example/** ディレクトリーのコンテンツを共有し、Windows ACL を使用する **example** という名前の共有を作成するには、以下を実行します。*

Windows ACL を使用する共有の追加

1. *または、フォルダーが存在しない場合は作成します。以下に例を示します。*

   ```
   ~]# mkdir -p /srv/samba/example/
   ```

2. *SELinux を、**enforcing** モードで実行する場合は、そのディレクトリーに **samba_share_t** コンテキストを設定します。*

   ```
   ~]# semanage fcontext -a -t samba_share_t "/srv/samba/example(/.*)?"
   ~]# restorecon -Rv /srv/samba/example/
   ```

3. ***/etc/samba/smb.conf** ファイルに **example** 共有を追加します。たとえば、共有の write-enabled を追加するには、次のコマンドを実行します。*

   ```
   [example]
    path = /srv/samba/example/
    read only = no
   ```

注記

*ファイルシステムの ACL に関係なく、**read only = no** を設定しないと、Samba がディレクトリーを読み取り専用モードで共有します。*

4. *すべての共有の **[global]** セクションで Windows ACL サポートを有効にしていない場合は、以下のパラメーターを **[example]** セクションに追加して、この共有に対してこの機能を有効にします。*

```
vfs objects = acl_xattr
map acl inherit = yes
store dos attributes = yes
```

5. */etc/samba/smb.conf ファイルを検証します。*

```
~]# testparm
```

*詳細は、「**testparm** ユーティリティーを使用した **smb.conf** ファイルの検証」を参照してください。*

6. ***firewall-cmd** ユーティリティーを使用して必要なポートを開き、ファイアウォール設定を再読み込みします。*

```
~]# firewall-cmd --permanent --add-service=samba
~]# firewall-cmd --reload
```

7. ***smb** サービスを再起動します。*

```
~]# systemctl restart smb
```

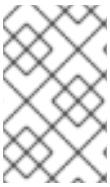
8. *必要に応じて、システムの起動時に、**smb** サービスが自動的に起動するようにします。*

```
~]# systemctl enable smb
```

16.1.6.2.4. Windows ACL を使用する共有の共有権限とファイルシステム ACL の管理

Windows ACL を使用する Samba 共有上の共有およびファイルシステム ACL を管理するには、Computer Management などの Windows アプリケーションを使用します。詳細は Windows ドキュメントを参照してください。

*または、**smbcacls** ユーティリティーを使用して ACL を管理します。詳細は、「**smbcacls** で SMB 共有上の ACL の管理」を参照してください。*



注記

*Windows からファイルシステムの権限を変更するには、**SeDiskOperatorPrivilege** 権限が付与されたアカウントを使用する必要があります。「**SeDiskOperatorPrivilege** 特権の付与」を参照してください。*

16.1.6.3. smbcacls で SMB 共有上の ACL の管理

smbcacls ユーティリティーは、SMB 共有に保存されたファイルおよびディレクトリーの ACL をリスト表示、設定、および削除できます。*smbcacls* を使用して、ファイルシステムの ACL を管理できます。

- 高度な Windows ACL または POSIX ACL を使用するローカルまたはリモートの Samba サーバー
- Red Hat Enterprise Linux で、Windows でホストされる共有の ACL をリモートで管理

16.1.6.3.1. アクセス制御エントリーについて

ファイルシステムオブジェクトの各 ACL エントリーには、以下の形式のアクセス制御エントリー (ACE) が含まれます。

```
security_principal:access_right/inheritance_information/permissions
```

例16.5 アクセス制御エントリー

AD\Domain Users グループに、Windows 上のこのフォルダー、サブフォルダー、およびファイルに適用される 変更 権限がある場合、ACL には以下の ACE が含まれます。

```
AD\Domain Users:ALLOWED/OI|CI/CHANGE
```

以下は、個々の ACE を説明しています。

セキュリティープリンシパル
: セキュリティープリンシパルは、ACL の権限が適用されるユーザー、グループ、または SID です。

アクセス権
: オブジェクトへのアクセスが許可または拒否されるかどうかを定義します。値は **ALLOWED** または **DENIED** です。

継承情報
: 次の値を取ります。

表16.3 継承の設定

値	詳細	マップ先
OI	オブジェクトの継承	このフォルダーおよびファイル
CI	コンテナーの継承	このフォルダーおよびサブフォルダー
IO	継承のみ	ACE は、現在のファイルまたはディレクトリーには適用されません。
ID	継承済	親ディレクトリーから ACE が継承されました。

また、値は以下のように組み合わせることができます。

表16.4 継承設定の組み合わせ

値の組み合わせ	Windows の 適用先 設定にマップします。
OI/CI	このフォルダー、サブフォルダー、およびファイル
OI/CI/IO	サブフォルダーおよびファイルのみ
CI/IO	サブディレクトリーのみ
OI/IO	ファイルのみ

権限

*この値は、Windows の権限または **smbcacls** エイリアスを表す 16 進値になります。*

- *1 つ以上の Windows の権限を表す 16 進値。*
  次の表に、Windows の高度な権限とそれに対応する値を 16 進法で表示します。

  *表16.5 Windows の権限とそれに対応する **smbcacls** 値を 16 進法で設定*

Windows の権限	16 進値
完全な制御	**0x001F01FF**
フォルダーのスキャンおよびファイルの実行	**0x00100020**
フォルダーのリスト表示 / データの読み取り	**0x00100001**
属性の読み取り	**0x00100080**
拡張属性の読み取り	**0x00100008**
ファイルの作成/データの書き込み	**0x00100002**
フォルダーの作成/データの追加	**0x00100004**
属性の書き込み	**0x00100100**
拡張属性の書き込み	**0x00100010**
サブフォルダーおよびファイルの削除	**0x00100040**
削除	**0x00110000**
権限の読み取り	**0x00120000**
権限の変更	**0x00140000**

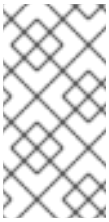
Windows の権限	16 進値
所有権の取得	**0x00180000**

*ビット単位の **OR** 演算を使用すると、複数の権限を 1 つの 16 進値として組み合わせることができます。詳細は、「ACE マスクの計算」を参照してください。*

- ***smbcacls** エイリアス。以下の表には、利用可能なエイリアスが表示されます。*

  *表16.6 既存の **smbcacls** エイリアスとそれに対応する Windows の権限*

smbcacls エイリアス	Windows の権限へのマッピング
-R	読み取り
READ	読み取りおよび実行
W	Special ○ ファイルの作成/データの書き込み ○ フォルダーの作成/データの追加 ○ 属性の書き込み ○ 拡張属性の書き込み ○ 権限の読み取り
D	削除
%P	権限の変更
O	所有権の取得
X	スキャン / 実行
CHANGE	修正
FULL	完全な制御



注記

*権限を設定する際に、1 文字のエイリアスを組み合わせることができます。たとえば、Windows の権限の **Read** および **Delete** を適用するように **RD** を設定できます。ただし、1 文字以外のエイリアスを複数組み合わせたり、エイリアスと 16 進値を組み合わせることはできません。*

*16.1.6.3.2. **smbcacls** を使用した ACL の表示*

--add などの操作パラメーターを付けずに smbcacls を実行すると、ユーティリティーは、ファイルシステムオブジェクトの ACL を表示します。

たとえば、//server/example 共有のルートディレクトリーの ACL をリスト表示するには、以下のコマンドを実行します。

```
~]# smbcacls //server/example / -U 'DOMAIN\administrator'
Enter DOMAIN\administrator's password:
REVISION:1
CONTROL:SR|PD|DI|DP
OWNER:AD\Administrators
GROUP:AD\Domain Users
ACL:AD\Administrator:ALLOWED/OI|CI/FULL
ACL:AD\Domain Users:ALLOWED/OI|CI/CHANGE
ACL:AD\Domain Guests:ALLOWED/OI|CI/0x00100021
```

コマンドの出力は以下のようになります。

- ***REVISION** - セキュリティー記述子の内部 Windows NT ACL リビジョン*
- ***CONTROL** - セキュリティー記述子の制御*
- ***OWNER** - セキュリティー記述子の所有者の名前または SID*
- ***GROUP** - セキュリティー記述子のグループの名前または SID*
- ***ACL** エントリー。詳細は、「アクセス制御エントリーについて」 を参照してください。*

16.1.6.3.3. ACE マスクの計算

*ほとんどの場合、ACE を追加または更新する場合は、表16.6「既存の **smbcacls** エイリアスとそれに対応する Windows の権限」 に記載されている **smbcacls** エイリアスを使用します。*

*ただし、表16.5「Windows の権限とそれに対応する **smbcacls** 値を 16 進法で設定」 にあるように高度な Windows の権限を設定する場合は、ビット単位の **OR** 演算を使用して、正しい値を計算する必要があります。以下のシェルコマンドを使用して値を計算できます。*

```
~]# echo $(printf '0x%X' $
hex_value_1 | hex_value_2 | ...)
```

例16.6 ACE マスクの計算

以下のパーミッションを設定したい場合。

- *フォルダーのスキャン / ファイルの実行 (**0x00100020**)*
- *フォルダーのリスト表示 / データの読み取り (**0x00100001**)*
- *属性の読み取り (**0x00100080**)*

以前の権限の 16 進値を計算するには、以下を入力します。

```
~]# echo $(printf '0x%X' $(( 0x00100020 | 0x00100001 | 0x00100080 )))
0x1000A1
```

ACE を設定または更新する場合は、戻り値を使用します。

16.1.6.3.4. smbcacls を使用した ACL の追加、更新、および削除

smbcacls ユーティリティーに渡すパラメーターに応じて、ファイルまたはディレクトリーから ACL を追加、更新、および削除できます。

ACL の追加

このフォルダー、サブフォルダー、およびファイルの **CHANGE** 権限を **AD\Domain Users** グループに付与する **//server/example** 共有のルートに ACL を追加するには、以下のコマンドを実行します。

```
~]# smbcacls //server/example / -U 'DOMAIN\administrator \
   --add ACL:"AD\Domain Users":ALLOWED/OI|CI/CHANGE
```

ACL の更新

ACL の更新は、新しい ACL の追加に似ています。ACL を更新する場合は、**--modify** パラメーターと既存のセキュリティープリンシパルを使用して ACL を上書きします。**smbcacls** が ACL リスト内でセキュリティープリンシパルを検出すると、ユーティリティーは権限を更新します。これを行わないと、以下のエラーでコマンドが失敗します。

```
ACL for SID principal_name not found
```

たとえば、**AD\Domain Users** グループの権限を更新し、このフォルダー、サブフォルダー、およびファイルの **READ** に設定するには、以下のコマンドを実行します。

```
~]# smbcacls //server/example / -U 'DOMAIN\administrator \
   --modify ACL:"AD\Domain Users":ALLOWED/OI|CI/READ
```

ACL の削除

ACL を削除するには、正確な ACL と共に **--delete** を **smbcacls** ユーティリティーにパスします。以下に例を示します。

```
~]# smbcacls //server/example / -U 'DOMAIN\administrator \
   --delete ACL:"AD\Domain Users":ALLOWED/OI|CI/READ
```

16.1.6.4. ユーザーが Samba サーバーのディレクトリーを共有できるようにする

Samba サーバーでは、**root** 権限なしでユーザーがディレクトリーを共有できるように設定できます。

16.1.6.4.1. ユーザーの共有機能の有効化

ユーザーがディレクトリーを共有できるようにするには、管理者が Samba でユーザー共有を有効にする必要があります。たとえば、ローカルの **example** グループのメンバーのみがユーザー共有を作成できるようにするには、以下を実行します。

ユーザー共有の有効化

1. ローカルの **example** グループが存在しない場合は作成します。

```
~]# groupadd example
```

2. ユーザー共有の定義を保存し、その権限を正しく設定するために、Samba 用のディレクトリーを準備します。以下に例を示します。

   a. ディレクトリーを作成します。

```
~]# mkdir -p /var/lib/samba/usershares/
```

   b. **example** グループの書き込み権限を設定します。

```
~]# chgrp example /var/lib/samba/usershares/
~]# chmod 1770 /var/lib/samba/usershares/
```

   このディレクトリーの他のユーザーが保存したファイルの名前変更や削除を禁止するように **sticky** ビットを設定します。

3. **/etc/samba/smb.conf** ファイルを編集し、以下を **[global]** セクションに追加します。

   a. ユーザー共有の定義を保存するように設定したディレクトリーのパスを設定します。以下に例を示します。

```
usershare path = /var/lib/samba/usershares/
```

   b. このサーバーで Samba を作成できるユーザー共有の数を設定します。以下に例を示します。

```
usershare max shares = 100
```

   usershare max shares パラメーターにデフォルトの **0** を使用すると、ユーザー共有が無効になります。

   c. 必要に応じて、ディレクトリーの絶対パスのリストを設定します。たとえば、Samba が **/data** および **/srv** ディレクトリーのサブディレクトリーの共有のみを許可するように設定するには、以下を設定します。

```
usershare prefix allow list = /data /srv
```

   設定可能なユーザー共有関連のパラメーターのリストは、man ページの smb.conf(5) の USERSHARES セクションを参照してください。

4. **/etc/samba/smb.conf** ファイルを検証します。

```
~]# testparm
```

   詳細は、「**testparm** ユーティリティーを使用した **smb.conf** ファイルの検証」 を参照してください。

5. Samba 設定を再読み込みします。

```
~]# smbcontrol all reload-config
```

これで、ユーザーが、ユーザー共有を作成できるようになりました。詳細は、「ユーザー共有の追加」を参照してください。

16.1.6.4.2. ユーザー共有の追加

「ユーザーの共有機能の有効化」に従ってSambaを設定したら、ユーザーは**root**権限がなくても、**net usershare add**コマンドを実行してSambaサーバーのディレクトリーを共有できます。

net usershare addコマンドの構文: **net usershare add**share_namepathcommentACLsguest_ok=y|n

重要

ユーザー共有の作成時にACLを設定する場合は、ACLの前にcommentパラメーターを指定する必要があります。空のコメントを設定するには、空の文字列を二重引用符で囲みます。

管理者が、**/etc/samba/smb.conf**ファイルの**[global]**セクションで**usershare allow guests = yes**に設定すると、ユーザーはユーザー共有上でのみゲストアクセスを有効にできることに注意してください。

> 例16.7 ユーザー共有の追加
>
> ユーザーが、Sambaサーバーで**/srv/samba/**ディレクトリーを共有する場合があります。共有には、**example**という名前を付け、コメントを設定しないようにし、ゲストユーザーがアクセスできるようにします。また、共有権限は、**AD\Domain Users**グループへのフルアクセスと、その他のユーザーへの読み取り権限を設定する必要があります。この共有を追加するには、そのユーザーで以下を実行します。
>
> ```
> ~]$ net usershare add example /srv/samba/ "" \
>   "AD\Domain Users":F,Everyone:R guest_ok=yes
> ```

16.1.6.4.3. ユーザー共有の設定の更新

ユーザー共有の設定を更新したい場合は、同じ共有名と新規の設定で**net usershare add**コマンドを使用し、共有をオーバーライドします。「ユーザー共有の追加」を参照してください。

16.1.6.4.4. 既存のユーザー共有に関する情報の表示

ユーザーは、Sambaサーバーで**net usershare info**コマンドを実行して、ユーザーの共有および設定を表示できます。

任意のユーザーが作成したすべてのユーザー共有を表示するには、以下のコマンドを実行します。

```
~]$ net usershare info -l
[share_1]
path=/srv/samba/
comment=
usershare_acl=Everyone:R,host_name\user:F,
guest_ok=y
...
```

コマンドを実行するユーザーが作成した共有のみをリスト表示するには、**-l**パラメーターを省略します。

特定の共有に関する情報のみを表示するには、共有名またはワイルドカードをコマンドに渡します。たとえば、名前が **share_** で始まる共有の情報を表示する場合は、以下のコマンドを実行します。

```
~]$ net usershare info -l share*_
```

16.1.6.4.5. ユーザー共有のリスト表示

Samba サーバーで設定を行わずに利用可能なユーザー共有のみをリスト表示するには、**net usershare list** コマンドを使用します。

任意のユーザーが作成した共有をリスト表示するには、以下のコマンドを実行します。

```
~]$ net usershare list -l
share_1
share_2
...
```

コマンドを実行するユーザーが作成した共有のみをリスト表示するには、**-l** パラメーターを省略します。

特定の共有のみをリスト表示するには、共有名またはワイルドカードをコマンドに渡します。たとえば、名前が **share_** で始まる共有のみをリスト表示するには、以下のコマンドを実行します。

```
~]$ net usershare list -l share_*
```

16.1.6.4.6. ユーザー共有の削除

ユーザー共有を削除するには、共有を作成したユーザーとして、または **root** ユーザーとして以下を入力します。

```
~]$ net usershare delete share_name
```

16.1.6.5. 共有へのゲストアクセスの有効化

特定の状況では、認証なしでユーザーが接続できるディレクトリーを共有します。これを設定するには、共有でゲストアクセスを有効にします。

警告

共有に認証を使用しないと、セキュリティーリスクとなる場合があります。

共有でゲストアクセスが有効になっている場合、Samba はゲスト接続を、**guest account** パラメーターで設定したオペレーティングシステムアカウントにマッピングします。ゲストユーザーは、以下の条件のうち少なくとも 1 つを満たしている場合にこれらのファイルへアクセスできます。

- アカウントがファイルシステムの ACL にリスト表示されます。
- その他 のユーザーの POSIX 権限はこれを許可します。

例16.8 ゲスト共有の権限

*ゲストアカウントを **nobody** (デフォルト) にマッピングするように Samba を設定している場合は、下記の例の ACL が、以下を行うようになります。*

- *ゲストユーザーが **file1.txt** の読み込みを許可する*
- *ゲストユーザーによる **file2.txt** の読み込みおよび修正を許可する*
- *ゲストユーザーが **file3.txt** を読み込んだり修正しないようにする*

```
-rw-r--r--. 1 root    root   1024 1. Sep 10:00 file1.txt
-rw-r-----. 1 nobody   root   1024 1. Sep 10:00 file2.txt
-rw-r-----. 1 root    root   1024 1. Sep 10:00 file3.txt
```

*たとえば、既存の **[example]** 共有のゲストアクセスを有効にするには、以下を実行します。*

ゲスト共有のセットアップ

1. */etc/samba/smb.conf* ファイルを編集します。
   a. *これが、このサーバーで設定した最初のゲスト共有である場合は、以下を行います。*
      i. ***[global]** セクションに **map to guest = Bad User** を設定し ます。*

      ```
      [global]
      ...
      map to guest = Bad User
      ```

      この設定により、ユーザー名が存在しない限り、Samba は間違ったパスワードを使用したログイン試行を拒否します。指定したユーザー名がなく、ゲストアクセスが共有で有効になっている場合、Samba は接続をゲストのログインとして処理します。

      ii. *デフォルトでは、Samba は、Red Hat Enterprise Linux の **nobody** アカウントにゲストアカウントをマッピングします。オプションで、別のアカウントを設定できます。以下に例を示します。*

      ```
      [global]
      ...
      guest account = user_name
      ```

      このパラメーターに設定するアカウントは、Samba サーバーにローカルに存在する必要があります。セキュリティー上の理由から、Red Hat は有効なシェルを割り当てていないアカウントを使用することを推奨しています。

   b. ***[example]** セクションに **guest ok = yes** 設定を追加します。*

   ```
   [example]
   ...
   guest ok = yes
   ```

2. */etc/samba/smb.conf* ファイルを検証します。

   ```
   ~]# testparm
   ```

詳細は、「**testparm** ユーティリティーを使用した **smb.conf** ファイルの検証」を参照してください。

3. Samba 設定を再読み込みします。

```
~]# smbcontrol all reload-config
```

16.1.7. Samba プリントサーバーの設定

Samba をプリントサーバーとして設定すると、ネットワーク上のクライアントが Samba を使用して印刷できます。さらに、Windows クライアントは、(Samba サーバーが設定されている場合は) Samba サーバーからドライバーをダウンロードすることもできます。

プリンターを共有する前に、Samba をセットアップします。

- 「Samba をスタンドアロンサーバーとして設定」
- 「Samba をドメインメンバーとしてセットアップ」

16.1.7.1. Samba の spoolssd サービス

Samba の **spoolssd** は、**smbd** サービスに統合されるサービスです。Samba 設定の **spoolssd** を有効にすると、大量のジョブまたはプリンターがあるプリントサーバーのパフォーマンスが大幅に向上します。

spoolssd がないと、Samba は **smbd** プロセスをフォークし、各プリントジョブの **printcap** キャッシュを初期化します。プリンターが多数あると、キャッシュの初期化中に **smbd** サービスが数秒間応答しなくなることがあります。**spoolssd** サービスを使用すると、遅延なしでプリントジョブを処理している、プレフォークされた **smbd** プロセスを開始することができます。主な **spoolssd** **smbd** プロセスは、少ないメモリーを使用し、子プロセスをフォークして終了します。

spoolssd サービスを有効にするには、以下を実行します。

spoolssd サービスの有効化

1. **/etc/samba/smb.conf** ファイルの **[global]** セクションを編集します。
   a. 以下のパラメーターを追加します。

```
rpc_server:spoolss = external
rpc_daemon:spoolssd = fork
```

   b. 必要に応じて、以下のパラメーターを設定できます。

パラメーター	デフォルト	詳細
spoolssd:prefork_min_children	5	子プロセスの最小数
spoolssd:prefork_max_children	25	子プロセスの最大数

パラメーター	デフォルト	詳細
spoolssd:prefork_spawn_rate	5	新しい接続が確立されると、Samba は、このパラメーターに設定した新しい子プロセスの数を、**spoolssd:prefork_max_children** に設定された値までフォークします。
spoolssd:prefork_max_allowed_clients	100	子プロセスが処理するクライアントの数
spoolssd:prefork_child_min_life	60	子プロセスの最小有効期間 (秒単位)。最小は 60 秒です。

2. */etc/samba/smb.conf* ファイルを検証します。

```
~]# testparm
```

*詳細は、「**testparm** ユーティリティーを使用した **smb.conf** ファイルの検証」 を参照してください。*

3. ***smb*** *サービスを再起動します。*

```
~]# systemctl restart smb
```

*サービスを再起動すると、Samba が自動的に **smbd** 子プロセスを開始します。*

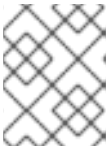
```
~]# ps axf
...
30903 smbd
30912 \_ smbd
30913   \_ smbd
30914   \_ smbd
30915   \_ smbd
...
```

16.1.7.2. Samba でのプリントサーバーのサポートの有効化

プリントサーバーサポートを有効化するには、以下を実行します。

Samba でのプリントサーバーのサポートの有効化

1. *Samba サーバーで CUPS を設定し、そのプリンターを CUPS バックエンドに追加します。詳細は、「印刷設定」 を参照してください。*



注記

Samba は、CUPS が Samba プリントサーバーにローカルにインストールされている場合に限り、CUPS に印刷ジョブを転送できます。

2. **/etc/samba/smb.conf** ファイルを編集します。

   a. **spoolssd** サービスを有効にする場合は、以下のパラメーターを **[global]** セクションに追加します。

   ```
   rpc_server:spoolss = external
   rpc_daemon:spoolssd = fork
   ```

   詳細は、「Samba の **spoolssd** サービス」 を参照してください。

   b. 印刷バックエンドを設定するには、**[printers]** セクションを追加します。

   ```
   [printers]
    comment = All Printers
    path = /var/tmp/
    printable = yes
    create mask = 0600
   ```

重要

printers 共有名はハードコーディングされており、変更はできません。

3. **/etc/samba/smb.conf** ファイルを検証します。

   ```
   ~]# testparm
   ```

   詳細は、「**testparm** ユーティリティーを使用した **smb.conf** ファイルの検証」 を参照してください。

4. **firewall-cmd** ユーティリティーを使用して必要なポートを開き、ファイアウォール設定を再読み込みします。

   ```
   ~]# firewall-cmd --permanent --add-service=samba
   ~]# firewall-cmd --reload
   ```

5. **smb** サービスを再起動します。

   ```
   ~]# systemctl restart smb
   ```

サービスを再起動すると、Samba は CUPS バックエンドに設定したすべてのプリンターを自動的に共有します。特定のプリンターのみを手動で共有する場合は、「特定のプリンターの手動共有」 を参照してください。

16.1.7.3. 特定のプリンターの手動共有

Samba をプリントサーバーとして設定している場合、Samba は、デフォルトで CUPS バックエンドで設定されたプリンターをすべて共有します。特定のプリンターのみを共有する場合は、以下を実行します。

特定のプリンターを手動で共有

1. **/etc/samba/smb.conf** ファイルを編集します。

   a. **[global]** セクションで、以下の設定で自動プリンター共有を無効にします。

   ```
   load printers = no
   ```

   b. 共有するプリンターごとにセクションを追加します。たとえば、Samba で CUPS バックエンドで **example** という名前のプリンターを **Example-Printer** として共有するには、以下のセクションを追加します。

   ```
   [Example-Printer]
    path = /var/tmp/
    printable = yes
    printer name = example
   ```

   各プリンターに個別のスプールディレクトリーは必要ありません。**[printers]** セクションに設定したのと同じ spool ディレクトリーを、プリンターの **path** パラメーターに設定できます。

2. **/etc/samba/smb.conf** ファイルを検証します。

   ```
   ~]# testparm
   ```

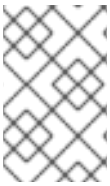
   詳細は、「**testparm** ユーティリティーを使用した **smb.conf** ファイルの検証」を参照してください。

3. Samba 設定を再読み込みします。

   ```
   ~]# smbcontrol all reload-config
   ```

16.1.7.4. Windows クライアント用の自動プリンタードライバーダウンロードの設定

Windows クライアント用に Samba プリントサーバーを実行している場合は、ドライバーをアップロードし、プリンターを事前設定できます。ユーザーがプリンターに接続すると、Windows により、ドライバーが自動的にクライアントにダウンロードされ、インストールされます。ユーザーがインストールするのに、ローカル管理者の権限を必要としません。また、Windows は、トレイの数などの事前設定済みのドライバー設定を適用します。



注記

自動プリンタードライバーダウンロードをセットアップする前に、Samba をプリントサーバーとして設定し、プリンターを共有する必要があります。詳細は、「Samba プリントサーバーの設定」を参照してください。

16.1.7.4.1. プリンタードライバーに関する基本情報

本セクションでは、プリンタードライバーに関する一般的な情報を説明します。

対応しているドライバーモデルのバージョン

Samba は、Windows 2000 以降および Windows Server 2000 以降でサポートされているプリンタードライバーのモデルバージョン 3 のみに対応します。Samba は、Windows 8 および Windows Server 2012 で導入されたドライバーモデルのバージョン 4 には対応していません。ただし、これ以降の

Windows バージョンは、バージョン3 のドライバーにも対応しています。

パッケージ対応ドライバー

Samba は、パッケージ対応ドライバーに対応していません。

アップロードするプリンタードライバーの準備

Samba プリントサーバーにドライバーをアップロードする場合は、以下を行います。

- *ドライバーが圧縮形式で提供されている場合は、ドライバーをデプロイメントします。*
- *一部のドライバーでは、Windows ホストにドライバーをローカルにインストールするセットアップアプリケーションを起動する必要があります。特定の状況では、インストーラーはセットアップの実行中にオペレーティングシステムの一時フォルダーに個別のファイルを抽出します。アップロードにドライバーファイルを使用するには、以下のコマンドを実行します。*
  a. *インストーラーを起動します。*
  b. *一時フォルダーから新しい場所にファイルをコピーします。*
  c. *インストールをキャンセルします。*

プリントサーバーへのアップロードをサポートするドライバーは、プリンターの製造元にお問い合わせください。

クライアントに 32 ビットおよび 64 ビットのプリンター用ドライバーを提供

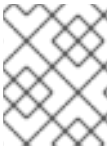
*32 ビットと 64 ビットの両方の Windows クライアントのプリンターにドライバーを提供するには、両方のアーキテクチャーに対して、同じ名前のドライバーをアップロードする必要があります。たとえば、32 ビットのドライバー **Example PostScript** および 64 ビットのドライバー**Example PostScript (v1.0)** をアップロードする場合は、その名前が一致しません。その結果、ドライバーのいずれかをプリンターに割り当てることしかできなくなり、両方のアーキテクチャーでそのドライバーが使用できなくなります。*

16.1.7.4.2. ユーザーがドライバーをアップロードおよび事前設定できるようにする

*プリンタードライバーをアップロードおよび事前設定できるようにするには、ユーザーまたはグループに **SePrintOperatorPrivilege** 特権が付与されている必要があります。**printadmin** グループにユーザーを追加する必要があります。Red Hat Enterprise Linux では、samba パッケージをインストールする際に、このグループが自動的に作成されます。**printadmin** グループには、1000 未満で利用可能な一番小さい動的システムの GID が割り当てられます。*

***SePrintOperatorPrivilege** 権限を **printadmin** グループに付与するには、次のコマンドを実行します。*

```
~]# net rpc rights grant "printadmin" SePrintOperatorPrivilege \
   -U "DOMAIN\administrator"
Enter DOMAIN\administrator's password:
Successfully granted rights.
```



注記

*ドメイン環境では、**SePrintOperatorPrivilege** をドメイングループに付与します。これにより、ユーザーのグループメンバーシップを更新し、権限を集中的に管理できます。*

SePrintOperatorPrivilege が付与されているユーザーとグループのリストを表示するには、以下を実行します。

```
~]# net rpc rights list privileges SePrintOperatorPrivilege \
    -U "DOMAIN\administrator"
Enter administrator's password:
SePrintOperatorPrivilege:
 BUILTIN\Administrators
 DOMAIN\printadmin
```

16.1.7.4.3. print$ 共有の設定

Windows のオペレーティングシステムは、プリントサーバーの共有 **print$** から、プリンタードライバーをダウンロードします。この共有名は Windows でハードコーディングされており、変更はできません。

/var/lib/samba/drivers/ ディレクトリーを **print$** として共有し、ローカルの **printadmin** グループのメンバーがプリンタードライバーをアップロードできるようにするには、以下を実行します。

print$ 共有の設定

1. **[print$]** セクションを **/etc/samba/smb.conf** ファイルに追加します。

   ```
   [print$]
    path = /var/lib/samba/drivers/
    read only = no
    write list = @printadmin
    force group = @printadmin
    create mask = 0664
    directory mask = 2775
   ```

   以下の設定を使用します。

   - **printadmin** グループのメンバーだけがプリンタードライバーを共有にアップロードできます。
   - 新規に作成されたファイルおよびディレクトリーのグループは **printadmin** に設定されます。
   - 新規ファイルの権限は **664** に設定されます。
   - 新しいディレクトリーの権限は、**2775** に設定されます。

2. プリンターの 64 ビットドライバーのみをアップロードするには、**/etc/samba/smb.conf** ファイルの **[global]** セクションにこの設定を含めます。

   ```
   spoolss: architecture = Windows x64
   ```

   この設定がないと、少なくとも 32 ビットバージョンでアップロードしたドライバーのみが表示されます。

3. **/etc/samba/smb.conf** ファイルを検証します。

   ```
   ~]# testparm
   ```

詳細は、「testparm ユーティリティーを使用した smb.conf ファイルの検証」 を参照してください。

4. Samba 設定を再読み込みします。

```
~]# smbcontrol all reload-config
```

5. printadmin グループが存在しない場合は作成します。

```
~]# groupadd printadmin
```

6. SePrintOperatorPrivilege 権限を、printadmin グループに付与します。

```
~]# net rpc rights grant "printadmin" SePrintOperatorPrivilege \
   -U "DOMAIN\administrator"
Enter DOMAIN\administrator's password:
Successfully granted rights.
```

詳細は、「ユーザーがドライバーをアップロードおよび事前設定できるようにする」. を参照してください。

7. SELinux を、enforcing モードで実行する場合は、そのディレクトリーに samba_share_t コンテキストを設定します。

```
~]# semanage fcontext -a -t samba_share_t "/var/lib/samba/drivers(/.*)?"
~]# restorecon -Rv /var/lib/samba/drivers/
```

8. /var/lib/samba/drivers/ ディレクトリーに権限を設定します。

- POSIX ACL を使用する場合は、以下を設定します。

```
~]# chgrp -R "printadmin" /var/lib/samba/drivers/
~]# chmod -R 2775 /var/lib/samba/drivers/
```

- Windows ACL を使用する場合は、以下を設定します。

プリンシパル	アクセス	適用先
CREATOR OWNER	完全な制御	サブフォルダーおよびファイルのみ
認証されたユーザー	読み取りおよび実行、フォルダーのコンテンツのリスト表示、読み取り	このフォルダー、サブフォルダー、およびファイル
printadmin	完全な制御	このフォルダー、サブフォルダー、およびファイル

Windows での ACL 設定に関する詳細は、Windows ドキュメントを参照してください。

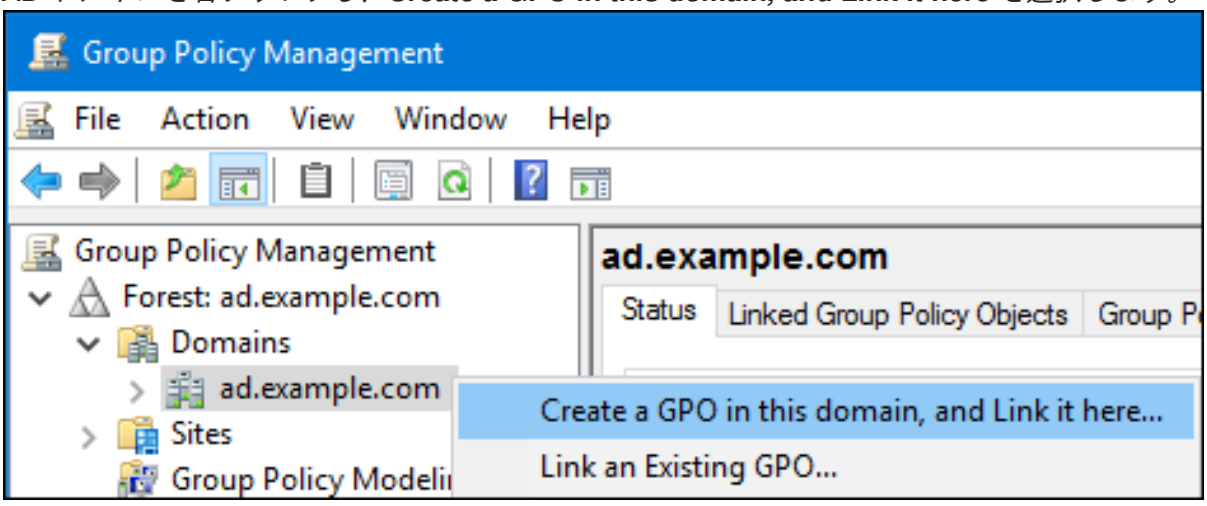
16.1.7.4.4. クライアントが Samba プリントサーバーを信頼できるようにする GPO の作成

セキュリティー上の理由から、最新の Windows オペレーティングシステムでは、クライアントが、信頼できないサーバーから、パッケージ対応ではないプリンタードライバーをダウンロードできないようにします。プリントサーバーが AD のメンバーである場合は、Samba サーバーを信頼するために、ドメインに Group Policy Object (GPO) を作成できます。

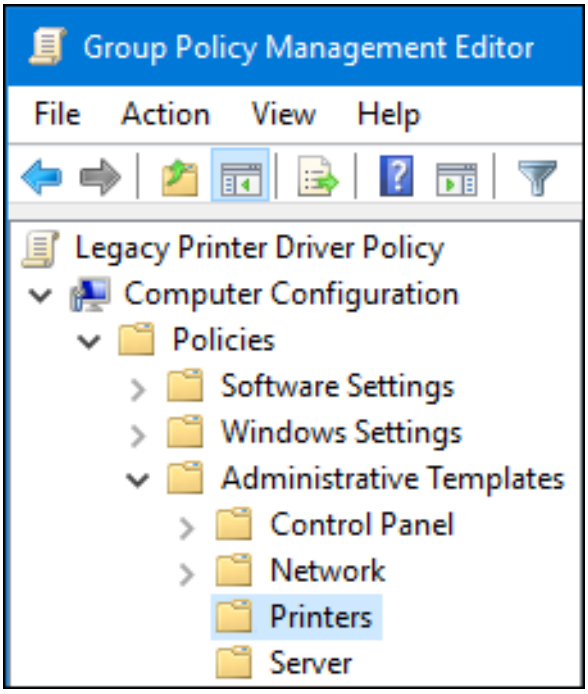
GPO を作成するには、お使いの Windows コンピューターに Windows Remote Server Administration Tools (RSAT) がインストールされていなければなりません。詳細は Windows ドキュメントを参照してください。

クライアントが Samba プリントサーバーを信頼できるようにする GPO の作成

1. AD ドメインの 管理者 ユーザーなど、グループポリシーの編集が可能なアカウントを使用して、Windows コンピューターにログインします。
2. Group Policy Management を開きます。
3. AD ドメインを右クリックし、**Create a GPO in this domain, and Link it here** を選択します。



4. **Legacy Printer Driver Policy** などの GPO の名前を入力して、**OK** をクリックします。新しい GPO がドメインエントリーの下に表示されます。
5. 新たに作成した GPO を右クリックして **Edit** を選択し、Group Policy Management Editor を開きます。
6. Computer Configuration → Policies → Administrative Templates → Printers の順にクリックします。

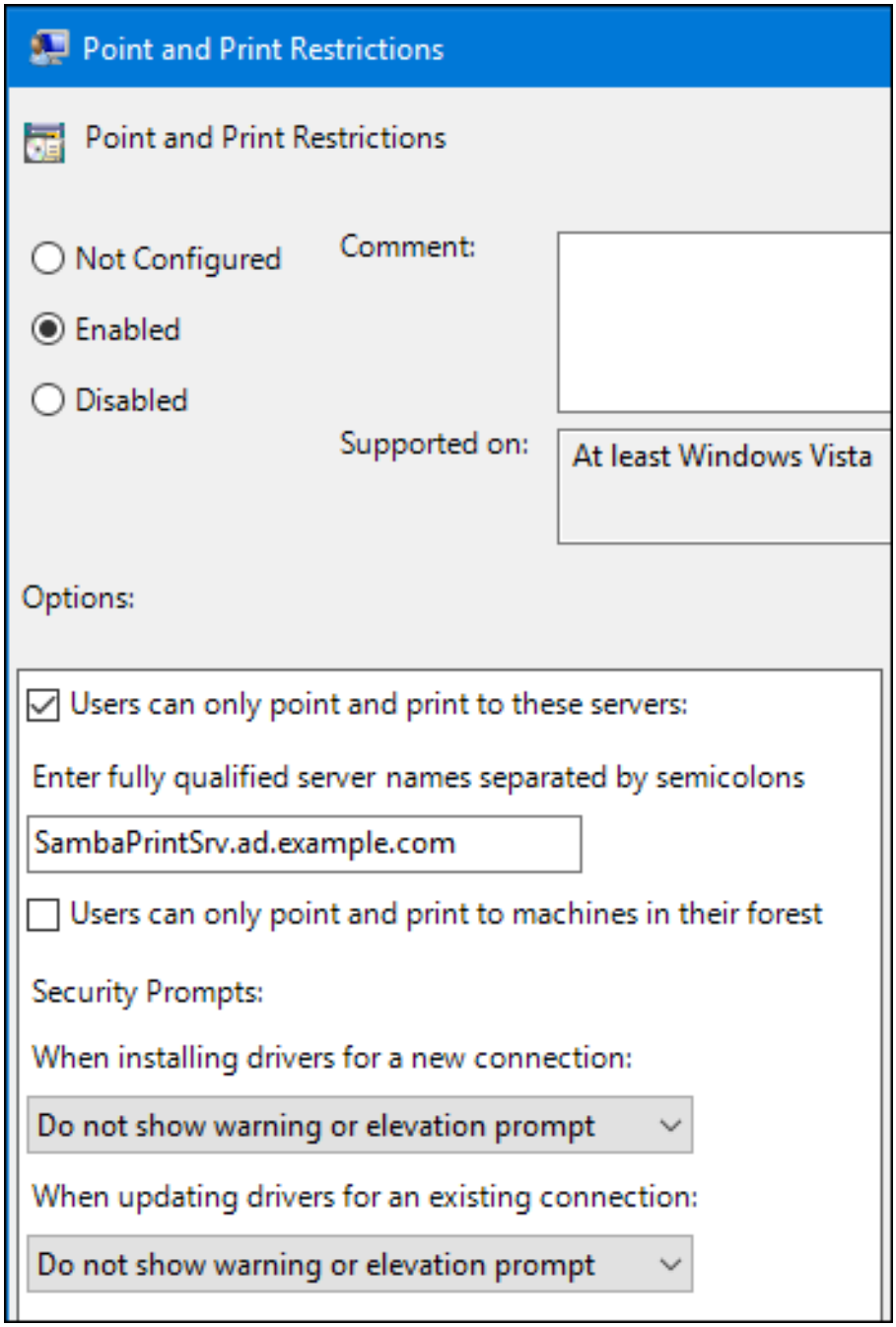


7. *ウィンドウの右側で、**Point and Print Restriction** をダブルクリックして、ポリシーを編集します。*

    a. *ポリシーを有効にし、以下のオプションを設定します。*

        i. ***Users can only point and print to these servers** を選択し、このオプションの横にあるフィールドに、Samba プリントサーバーの完全修飾ドメイン名 (FQDN) を入力します。*

        ii. ***Security Prompts** の両チェックボックスで、**Do not show warning or elevation prompt** を選択します。*

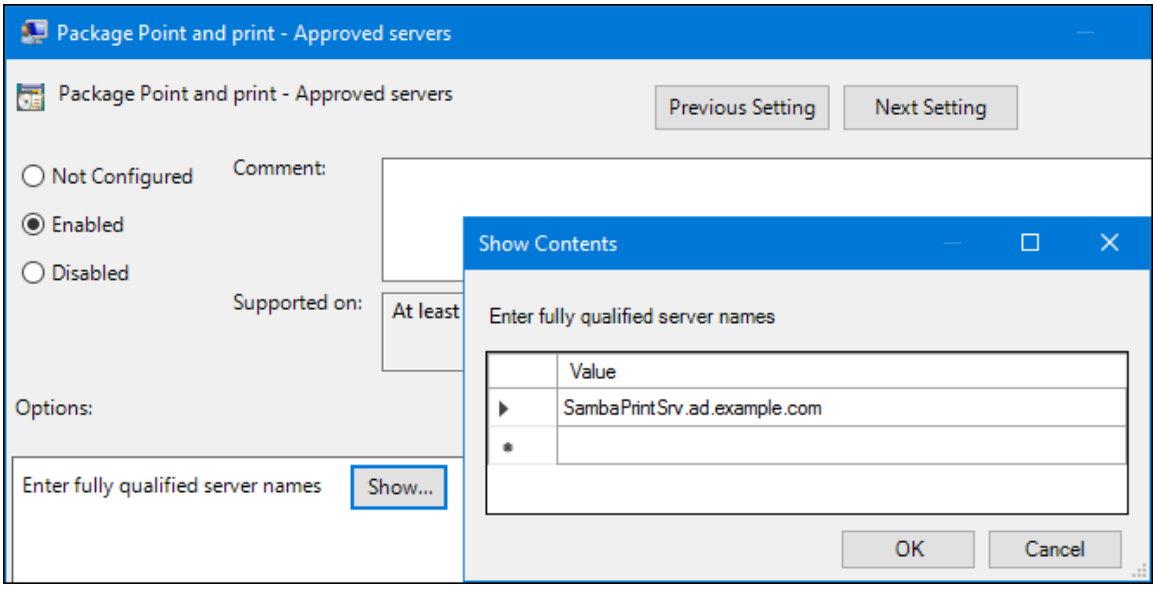


b. ***OK*** *をクリックします。*

8. ***Package Point and Print - Approved servers*** *をダブルクリックして、ポリシーを編集します。*

   a. *ポリシーを有効にして、****Show*** *ボタンをクリックします。*

   b. ***Samba*** *プリントサーバーの* ***FQDN*** *を入力します。*



c. ***OK*** *をクリックして、* ***Show Contents*** *とポリシープロパティーウィンドウの両方を閉じます。*

9. *Group Policy Management Editor を閉じます。*

10. *Group Policy Management を閉じます。*

Windows ドメインメンバーがこのグループポリシーを適用すると、ユーザーがプリンターに接続する際に、プリンタードライバーが Samba サーバーから自動的にダウンロードされます。

グループポリシーの使用に関する詳細は、Windows ドキュメントを参照してください。

16.1.7.4.5. ドライバーのアップロードおよびプリンターの事前設定

Windows クライアントで Print Management アプリケーションを使用してドライバーをアップロードし、Samba プリントサーバーでホストされるプリンターを事前設定します。詳細は Windows ドキュメントを参照してください。

16.1.8. Samba サーバーのパフォーマンスチューニング

本セクションでは、特定の状況における Samba のパフォーマンスを向上させる設定、そしてパフォーマンスを低下させる設定を説明します。

16.1.8.1. SMB プロトコルバージョンの設定

新しい SMB バージョンごとに機能が追加され、プロトコルのパフォーマンスが向上します。最新の Windows および Windows Server オペレーティングシステムは、常に最新のプロトコルバージョンに対応しています。Samba がプロトコルの最新バージョンも使用している場合は、Samba に接続する Windows クライアントで、このパフォーマンス改善を活用できます。Samba では、 ***server max protocol*** *のデフォルト値が、対応している安定した SMB プロトコルの最新バージョンに設定されます。*

常に最新の安定した SMB プロトコルバージョンを有効にするには、 ***server max protocol*** *パラメーターを設定しないでください。このパラメーターを手動で設定する場合は、最新のプロトコルバージョンを有効にするために、それぞれ新しいバージョンの SMB プロトコルで設定を変更する必要があります。*

/etc/samba/smb.conf ファイルの ***[global]*** *セクションから* ***server max protocol*** *を設定解除して削除するには、以下を実行します。*

16.1.8.2. 大量のファイルを含むディレクトリーとの共有の調整

100,000 以上のファイルがあるディレクトリーを含む共有のパフォーマンスを向上させるには、以下を実行します。

大量のファイルを含むディレクトリーとの共有の調整

1. 共有の全ファイルの名前を小文字に変更します。

注記

この手順の設定を使用すると、小文字以外の名前が付けられたファイルは表示されなくなります。

2. 共有のセクションに、以下のパラメーターを設定します。

```
case sensitive = true
default case = lower
preserve case = no
short preserve case = no
```

パラメーターの詳細は、**smb.conf(5)** man ページを参照してください。

3. Samba 設定を再読み込みします。

```
~]# smbcontrol all reload-config
```

この設定が適用されと、この共有に新たに作成されるすべてのファイルの名前が小文字になります。この設定により、Samba はディレクトリーを大文字と小文字で分けたスキャンが不要になり、パフォーマンスが向上します。

16.1.8.3. パフォーマンスが低下する可能性のある設定

デフォルトでは、Red Hat Enterprise Linux のカーネルは、ネットワークパフォーマンスが高くなるように調整されています。たとえば、カーネルはバッファーサイズに自動チューニングメカニズムを使用しています。**/etc/samba/smb.conf** ファイルに **socket options** パラメーターを設定すると、このカーネル設定が上書きされます。その結果、このパラメーターの設定により、ほとんどの場合は、Samba ネットワークのパフォーマンスが低下します。

カーネルの最適化された設定を使用するには、**/etc/samba/smb.conf** の **[global]** セクションから **socket options** パラメーターを削除します。

16.1.9. 頻繁に使用される Samba コマンドラインユーティリティー

本セクションでは、Samba サーバーを使用する際によく使うコマンドを説明します。

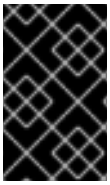
16.1.9.1. net ユーティリティーの使用

net ユーティリティーを使用すると、Samba サーバーで複数の管理タスクを実行できます。本セクションでは、**net** ユーティリティーで最も頻繁に使用されるサブコマンドを説明します。

詳細は、**net(8)** man ページを参照してください。

16.1.9.1.1. net ads join コマンドおよび net rpc join コマンドの使用

net ユーティリティーの join サブコマンドを使用すると、Samba を AD ドメインまたは NT4 ドメインに参加させることができます。ドメインに参加するには、/etc/samba/smb.conf ファイルを手動で作成し、必要に応じて PAM などの追加設定を更新する必要があります。



重要

Red Hat は、realm ユーティリティーを使用してドメインに参加させることを推奨します。realm ユーティリティーは、関連するすべての設定ファイルを自動的に更新します。詳細は、「ドメインの参加」 を参照してください。

net コマンドを使用してドメインに参加するには、以下を実行します。

net コマンドを使用したドメインへの参加

1. 以下の設定で /etc/samba/smb.conf ファイルを手動で作成します。
   - AD ドメインメンバーの場合:

     ```
     [global]
     workgroup = domain_name
     security = ads
     passdb backend = tdbsam
     realm = AD_REALM
     ```

   - NT4 ドメインメンバーの場合:

     ```
     [global]
     workgroup = domain_name
     security = user
     passdb backend = tdbsam
     ```

2. デフォルトドメインの *、および参加するドメインの ID マッピング設定を /etc/samba/smb.conf の [global] セクションに追加します。詳細は、「ID マッピングについて」 を参照してください。

3. /etc/samba/smb.conf ファイルを検証します。

   ```
   ~]# testparm
   ```

   詳細は、「testparm ユーティリティーを使用した smb.conf ファイルの検証」 を参照してください。

4. ドメイン管理者としてドメインに参加します。
   - AD ドメインに参加するには、以下のコマンドを実行します。

     ```
     ~]# net ads join -U "DOMAIN\administrator"
     ```

   - NT4 ドメインに参加するには、以下のコマンドを実行します。

     ```
     ~]# net rpc join -U "DOMAIN\administrator"
     ```

5. /etc/nsswitch.conf ファイルのデータベースエントリーpasswd および group に winbind ソースを追加します。

```
passwd:   files winbind
group:   files winbind
```

6. winbind サービスを有効にして起動します。

```
~]# systemctl enable winbind
~]# systemctl start winbind
```

7. オプションで、authconf ユーティリティーを使用して PAM を設定します。
詳細は、Red Hat System-Level Authentication Guide のPAM (プラグ可能な認証モジュール) の使用 セクションを参照してください。

8. AD 環境では、必要に応じて Kerberos クライアントを設定します。
詳細は Red Hat システムレベルの認証ガイド のConfiguring a Kerberos Client セクションを参照してください。

16.1.9.1.2. net rpc rights コマンドの使用

Windows では、アカウントおよびグループに特権を割り当て、共有での ACL の設定やプリンタードライバーのアップロードなどの特別な操作を実行できます。Samba サーバーでは、net rpc rights コマンドを使用して権限を管理できます。

権限のリスト表示

利用可能な特権とその所有者をすべて表示するには、net rpc rights list コマンドを使用します。以下に例を示します。

```
net rpc rights list -U "DOMAIN\administrator"
Enter DOMAIN\administrator's password:
  SeMachineAccountPrivilege Add machines to domain
  SeTakeOwnershipPrivilege Take ownership of files or other objects
     SeBackupPrivilege Back up files and directories
    SeRestorePrivilege Restore files and directories
  SeRemoteShutdownPrivilege Force shutdown from a remote system
  SePrintOperatorPrivilege Manage printers
    SeAddUsersPrivilege Add users and groups to the domain
 SeDiskOperatorPrivilege Manage disk shares
    SeSecurityPrivilege System security
```

特権の付与

アカウントまたはグループへの特権を付与するには、net rpc rights grant コマンドを使用します。

たとえば、SePrintOperatorPrivilege 権限を、DOMAIN\printadmin グループに付与します。

```
~]# net rpc rights grant "DOMAIN\printadmin" SePrintOperatorPrivilege \
    -U "DOMAIN\administrator"
Enter DOMAIN\administrator's password:
Successfully granted rights.
```

特権の取り消し

アカウントまたはグループから権限を取り消すには、**net rpc rights revoke** を使用します。

たとえば、**DOMAIN\printadmin** グループから **SePrintOperatorPrivilege** 権限を取り消すには、以下のコマンドを実行します。

```
~]# net rpc rights remoke "DOMAIN\printadmin" SePrintOperatorPrivilege \
    -U "DOMAIN\administrator"
Enter DOMAIN\administrator's password:
Successfully revoked rights.
```

16.1.9.1.3. net rpc share コマンドの使用

net rpc share コマンドは、ローカルまたはリモートの Samba または Windows サーバーの共有のリスト表示、追加、および削除を行う機能を提供します。

共有のリスト表示

SMB サーバーの共有をリスト表示するには、**net rpc share list command** コマンドを使用します。必要に応じて、**-S server_name** パラメーターをコマンドに渡して、リモートサーバーの共有をリスト表示します。以下に例を示します。

```
~]# net rpc share list -U "DOMAIN\administrator" -S example
Enter DOMAIN\administrator's password:
IPC$
share_1
share_2
...
```

注記

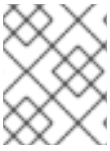
/etc/samba/smb.conf ファイルのセクション内に **browseable = no** が設定されている Samba サーバーでホストされている共有は、出力には表示されません。

共有の追加

net rpc share add コマンドを使用すると、SMB サーバーに共有を追加できます。

たとえば、**C:\example** ディレクトリーを共有するリモートの Windows サーバーに、共有**example** を追加するには、以下のコマンドを実行します。

```
~]# net rpc share add example="C:\example" -U "DOMAIN\administrator" -S server
```



注記

Windows のディレクトリー名を指定する際は、パスの末尾のバックスラッシュを省略する必要があります。

このコマンドを使用して Samba サーバーに共有を追加するには、以下を行います。

- **-U** パラメーターで指定したユーザーは、**SeDiskOperatorPrivilege** 特権が付与されている必要があります。
- 共有セクションを、**/etc/samba/smb.conf** ファイルに追加し、Samba を再読み込みするスクリ

プトを記述する必要があります。スクリプトは、**/etc/samba/smb.conf** の **[global]** セクションの **add share command** パラメーターで設定する必要があります。詳細は、smb.conf(5) man ページの **add share command** の説明を参照してください。

共有の削除

net rpc share delete コマンドを使用すると、SMB サーバーから共有を削除できます。

たとえば、**example** という名前の共有を、リモートの Windows サーバーから削除するには、以下のコマンドを実行します。

```
~]# net rpc share delete example -U "DOMAIN\administrator" -S server
```

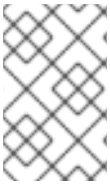
このコマンドを使用して Samba サーバーから共有を削除するには、以下のコマンドを実行します。

- **-U** パラメーターで指定したユーザーは、**SeDiskOperatorPrivilege** 特権が付与されている必要があります。
- **/etc/samba/smb.conf** ファイルから共有のセクションを削除し、Samba を再読み込みするスクリプトを記述する必要があります。スクリプトは、**/etc/samba/smb.conf** の **[global]** セクションの **delete share command** パラメーターで設定する必要があります。詳細は、smb.conf(5) man ページの **delete share** コマンドの説明を参照してください。

16.1.9.1.4. net user コマンドの使用

net user コマンドを使用すると、AD DC または NT4 PDC で以下の操作を実行できます。

- すべてのユーザーアカウントのリストを表示
- ユーザーの追加
- ユーザーの削除



注記

AD ドメイン用の **ads**、NT4 ドメイン用の **rpc** などの接続方法の指定は、ドメインユーザーアカウントをリスト表示する場合にのみ必要です。その他のユーザー関連のサブコマンドは、接続メソッドを自動検出できます。

-U user_name パラメーターをコマンドに渡して、要求されたアクションを実行できるユーザーを指定します。

ドメインユーザーアカウントのリスト表示

AD ドメイン内のユーザーをリスト表示するには、以下を実行します。

```
~]# net ads user -U "DOMAIN\administrator"
```

NT4 ドメインのユーザーをリスト表示するには、以下を実行します。

```
~]# net rpc user -U "DOMAIN\administrator"
```

ユーザーアカウントのドメインへの追加

*Samba ドメインメンバーの場合は、**net user add** コマンドを使用して、ユーザーアカウントをドメインに追加できます。*

*たとえば、**user** アカウントをドメインに追加します。*

ユーザーアカウントのドメインへの追加

1. *以下のアカウントを追加します。*

```
~]# net user add user password -U "DOMAIN\administrator"
User user added
```

2. *必要に応じて、リモートプロシージャコール (RPC) シェルを使用して、AD DC または NT4 PDC でアカウントを有効にします。以下に例を示します。*

```
~]# net rpc shell -U DOMAIN\administrator -S DC_or_PDC_name
Talking to domain DOMAIN (S-1-5-21-1424831554-512457234-5642315751)

net rpc> user edit disabled user no
Set user's disabled flag from [yes] to [no]

net rpc> exit
```

ドメインからのユーザーアカウントの削除

*Samba ドメインメンバーの場合は、**net user delete** コマンドを使用して、ドメインからユーザーアカウントを削除できます。*

*たとえば、ドメインから **user** アカウントを削除するには、以下のコマンドを実行します。*

```
~]# net user delete user -U "DOMAIN\administrator"
User user deleted
```

16.1.9.1.5. net usershare コマンドの使用

「ユーザーが Samba サーバーのディレクトリーを共有できるようにする」を参照してください。

16.1.9.2. rpcclient ユーティリティーの使用

***rpcclient** ユーティリティーを使用すると、ローカルまたはリモートの SMB サーバーでクライアント側の Microsoft Remote Procedure Call (MS-RPC) 機能を手動で実行できます。ただし、ほとんどの機能は、Samba が提供する個別のユーティリティーに統合されています。**rpcclient** は、MS-PRC 関数のテストにのみ使用します。*

たとえば、ユーティリティーを使用すると以下が可能となります。

- *プリンターのスプールサブシステム (SPOOLSS) を管理します。*

  例16.9 プリンターへのドライバーの割り当て

```
~]# rpcclient server_name -U "DOMAINpass:quotes[administrator]" \
    -c 'setdriver "printer_name" "driver_name"'
Enter DOMAINpass:quotes[administrator]s password:
Successfully set printer_name to driver driver_name.
```

- SMB サーバーに関する情報を取得します。

  例16.10 すべてのファイル共有および共有プリンターのリスト表示

  ```
  ~]# rpcclient server_name -U "DOMAIN\administrator" -c
  'netshareenum'
  Enter DOMAIN\administrator's password:
  netname: Example_Share
   remark:
   path:  C:\srv\samba\example_share\
   password:
  netname: Example_Printer
   remark:
   path:  C:\var\spool\samba\
   password:
  ```

- Security Account Manager Remote (SAMR) プロトコルを使用して操作を実行します。

  例16.11 SMB サーバー上のユーザーのリスト表示

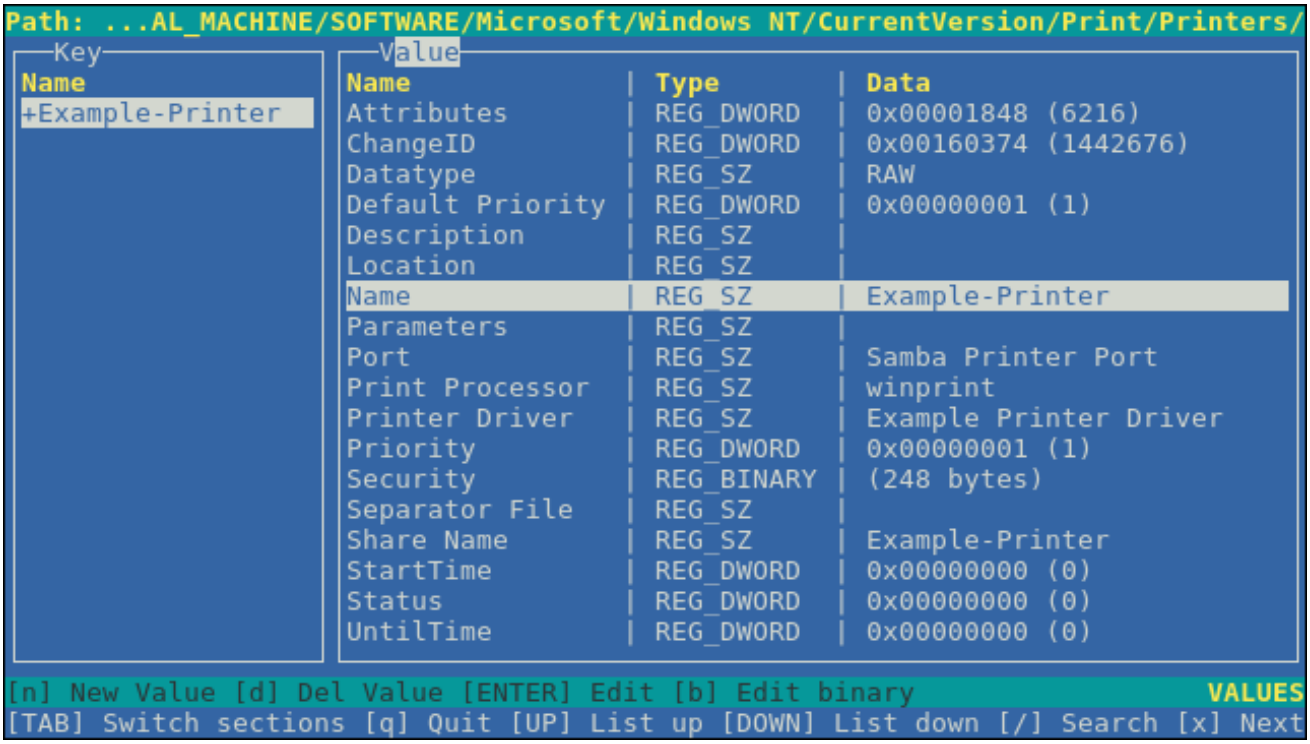
  ```
  ~]# rpcclient server_name -U "DOMAIN\administrator" -c
  'enumdomusers'
  Enter DOMAIN\administrator's password:
  user:[user1] rid:[0x3e8]
  user:[user2] rid:[0x3e9]
  ```

  スタンドアロンサーバーまたはドメインメンバーに対してコマンドを実行すると、ローカルデータベースのユーザーのリストが表示されます。ADDC または NT4 PDC に対してコマンドを実行すると、ドメインユーザーのリストが表示されます。

サポートされるサブコマンドのリストは、man ページの rpcclient(1) のCOMMANDS セクションを参照してください。

16.1.9.3. samba-regedit アプリケーションの使用

プリンター設定などの特定の設定は、Samba サーバーのレジストリーに保存されます。ncurses ベースの **samba-regedit** アプリケーションを使用して、Samba サーバーのレジストリーを編集できます。



アプリケーションを起動するには、次のコマンドを入力します。

```
~]# samba-regedit
```

次のキーを使用します。

- *カーソルを上下に動かして、レジストリーツリーと値の間を移動します。*
- ***Enter** - キーを開くか、値を編集します。*
- ***Tab** - **Key** ペインと **Value** ペインを切り替えます。*
- ***Ctrl+C**: アプリケーションを閉じます。*

16.1.9.4. smbcacls ユーティリティーの使用

*「**smbcacls** で SMB 共有上の ACL の管理」を参照してください。*

16.1.9.5. smbclient ユーティリティーの使用

***smbclient** ユーティリティーを使用すると、コマンドラインの FTP クライアントと同様に、SMB サーバーのファイル共有にアクセスできます。たとえば、ファイルを共有にアップロードしたり、共有からダウンロードしたりできます。*

*たとえば、**DOMAIN\user** アカウントを使用して **server** でホストされる **example** 共有に認証するには、以下のコマンドを実行します。*

```
~]# smbclient -U 'DOMAIN\user' //server/example
Enter domain\user's password:
Domain=[SERVER] OS=[Windows 6.1] Server=[Samba 4.6.2]
smb: \>
```

smbclient が共有に正常に接続すると、ユーティリティーはインタラクティブモードになり、以下のプロンプトが表示されます。

```
smb: \>
```

対話式シェルで利用可能なすべてのコマンドを表示するには、以下のコマンドを実行します。

```
smb: \> help
```

特定のコマンドのヘルプを表示するには、以下のコマンドを実行します。

```
smb: \> help command_name
```

インタラクティブシェルで利用可能なコマンドの詳細と説明は、man ページの **smbclient(1)** を参照してください。

16.1.9.5.1. 対話モードでの smbclient の使用

-c パラメーターを指定せずに **smbclient** を使用すると、ユーティリティーは対話モードを開始します。

以下の手順では、SMB 共有に接続し、サブディレクトリーからファイルをダウンロードする方法を説明します。

smbclient を使用して SMB 共有からファイルをダウンロード

1. 共有に接続します。

```
~]# smbclient -U 'DOMAIN\user_name' //server_name/share_name
```

2. **/example/** ディレクトリーに移動します。

```
smb: \> cd /example/
```

3. ディレクトリー内のファイルをリスト表示します。

```
smb: \example\> ls
.          D     0 Mon Sep 1 10:00:00 2017
..         D     0 Mon Sep 1 10:00:00 2017
example.txt     N  1048576 Mon Sep 1 10:00:00 2017

    9950208 blocks of size 1024. 8247144 blocks available
```

4. **example.txt** ファイルをダウンロードします。

```
smb: \example\> get example.txt
getting file \directory\subdirectory\example.txt of size 1048576 as example.txt
(511975,0 KiloBytes/sec) (average 170666,7 KiloBytes/sec)
```

5. 共有から切断します。

```
smb: \example\> exit
```

16.1.9.5.2. スクリプトモードでの smbclient の使用

-c commands パラメーターを **smbclient** へ渡した場合、リモートの SMB 共有上のコマンドを自動的に実行できます。これにより、スクリプトで **smbclient** を使用できます。

以下のコマンドは、SMB 共有への接続方法およびサブディレクトリーからのファイルのダウンロード方法を表示します。

```
~]# smbclient -U DOMAIN[user_name] //server_name/share_name \
   -c "cd /example/ ; get example.txt ; exit"
```

16.1.9.6. smbcontrol ユーティリティーの使用

smbcontrol ユーティリティーを使用すると、**smbd**、**nmbd**、**winbindd**、またはこのすべてのサービスにコマンドメッセージを送信できます。この制御メッセージは、設定の再読み込みなどのサービスを指示します。

例16.12 **smbd**、**nmbd**、および **winbindd** サービスの設定を再読み込み

たとえば、**smbd**、**nmbd**、および **winbindd** の設定を再読み込みするには、**reload-config** メッセージタイプを **all** 送信先に送信します。

```
~]# smbcontrol all reload-config
```

詳細情報および利用可能なコマンドメッセージタイプのリストは、smbcontrol(1) man ページを参照してください。

16.1.9.7. smbpasswd ユーティリティーの使用

smbpasswd ユーティリティーは、ローカルの Samba データベースでユーザーアカウントおよびパスワードを管理します。

ユーザーとしてコマンドを実行すると、**smbpasswd** はユーザーの Samba パスワードを変更します。以下に例を示します。

```
[user@server ~]$ smbpasswd
New SMB password:
Retype new SMB password:
```

root で **smbpasswd** を実行すると、たとえば以下のようにユーティリティーを使用できます。

- 新しいユーザーを作成します。

```
[root@server ~]# smbpasswd -a user_name
New SMB password:
Retype new SMB password:
Added user user_name.
```

注記

Samba データベースにユーザーを追加する前に、ローカルのオペレーティングシステムにアカウントを作成する必要があります。「新規ユーザーの追加」 を参照してください。

- Samba ユーザーを有効にします。

```
[root@server ~]# smbpasswd -e user_name
Enabled user user_name.
```

- Samba ユーザーを無効にします。

```
[root@server ~]# smbpasswd -x user_name
Disabled user user_name.
```

- ユーザーを削除します。

```
[root@server ~]# smbpasswd -x user_name
Deleted user user_name.
```

詳細は、smbpasswd(8) man ページを参照してください。

16.1.9.8. smbstatus ユーティリティーの使用

smbstatus ユーティリティーは以下について報告します。

- 各 **smbd** デーモンの PID ごとの接続を Samba サーバーに接続します。このレポートには、ユーザー名、プライマリーグループ、SMB プロトコルのバージョン、暗号、および署名の情報が含まれます。
- Samba 共有ごとの接続このレポートには、**smbd** デーモンの PID、接続しているマシンの IP、接続が確立された時点のタイムスタンプ、暗号、および署名情報が含まれます。
- ロックされたファイルのリスト。レポートエントリーには、日和見ロック (oplock) タイプなどの詳細が含まれます。

例16.13 **smbstatus** ユーティリティーの出力

```
~]# smbstatus

Samba version 4.6.2
PID Username      Group       Machine            Protocol Version Encryption Signing
----------------------------------------------------------------------------------------------------------------
------
963 DOMAIN\administrator DOMAIN\domain users client-pc (ipv4:192.0.2.1:57786) SMB3_02
-     AES-128-CMAC

Service pid Machine  Connected at        Encryption Signing:
---------------------------------------------------------------------------
example 969 192.0.2.1 Mo Sep 1 10:00:00 2017 CEST -     AES-128-CMAC

Locked files:
Pid Uid  DenyMode  Access  R/W   Oplock   SharePath     Name   Time
```

```
------------------------------------------------------------------------------------------------
969 10000 DENY_WRITE 0x120089 RDONLY LEASE(RWH) /srv/samba/example file.txt Mon
Sep 1 10:00:00 2017
```

詳細は、smbstatus(1) man ページを参照してください。

16.1.9.9. smbtar ユーティリティーの使用

smbtar ユーティリティーは、SMB 共有またはそのサブディレクトリーのコンテンツのバックアップを作成し、そのコンテンツを **tar** アーカイブに保存します。または、コンテンツをテープデバイスに書き込むこともできます。

たとえば、**//server/example/** 共有上の **demo** ディレクトリーのコンテンツをバックアップして、**/root/example.tar** アーカイブにコンテンツを保存するには、以下を実行します。

```
~]# smbtar -s server -x example -u user_name -p password -t /root/example.tar
```

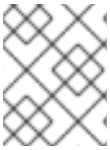
詳細は、smbtar(1) man ページを参照してください。

16.1.9.10. testparm ユーティリティーの使用

「**testparm** ユーティリティーを使用した **smb.conf** ファイルの検証」を参照してください。

16.1.9.11. wbinfo ユーティリティーの使用

wbinfo ユーティリティーは、**winbindd** サービスが作成および使用する情報をクエリーし、返します。



注記

winbindd サービスは、**wbinfo** を使用できるように設定および実行される必要があります。

たとえば、以下のように、**wbinfo** を使用できます。

- ドメインユーザーのリストを表示します。

```
~]# wbinfo -u
AD\administrator
AD\guest
...
```

- ドメイングループのリストを表示します。

```
~]# wbinfo -g
AD\domain computers
AD\domain admins
AD\domain users
...
```

- ユーザーの SID を表示します。

```
~]# wbinfo --name-to-sid="AD\administrator"
S-1-5-21-1762709870-351891212-3141221786-500 SID_USER (1)
```

- ドメインおよび信頼に関する情報を表示します。

```
~]# wbinfo --trusted-domains --verbose
Domain Name  DNS Domain      Trust Type Transitive In  Out
BUILTIN                      None    Yes     Yes Yes
server                       None    Yes     Yes Yes
DOMAIN1    domain1.example.com  None    Yes     Yes Yes
DOMAIN2    domain2.example.com  External  No     Yes Yes
```

詳細は、wbinfo(1) man ページを参照してください。

16.1.10. 関連情報

- Red Hat Samba パッケージには、パッケージがインストールするすべてのSamba コマンドおよび設定ファイルのman ページが含まれています。たとえば、**/etc/samba/smb.conf** ファイルのman ページを表示して、このファイルに設定できる設定パラメーターをすべて説明します。

```
~]# man 5 smb.conf
```

- **/usr/share/docs/samba-version/**: Samba プロジェクトが提供する一般的なドキュメント、スクリプトの例、およびLDAP スキーマファイルが含まれます。

- Red Hat Cluster Storage Administration Guide: Samba と Clustered Trivial Database (CDTB) の設定に関する情報を提供し、GlusterFS ボリュームに保存されているディレクトリーを共有します。

- Red Hat Enterprise Linux High Availability Add-on AdministrationのAn active/active Samba Server in a Red Hat High Availability Cluster の章では、Samba の高可用性インストールの設定方法を説明します。

- Red Hat Enterprise Linux 上へのSMB 共有のマウントに関する詳細は、*Red Hat ストレージ管理ガイド* の該当するセクションを参照してください。

16.2. FTP

ファイル転送プロトコル (FTP) は、今日インターネット上で見られる、最も古く、一般的に使用されているプロトコルです。この目的は、ユーザーがリモートホストに直接ログインしなくても、もしくはリモートシステムの使用法についての知識がなくとも、ネットワーク上のコンピューターホスト間で確実にファイルを転送することです。これにより、ユーザーは、標準の簡単なコマンドセットを使用してリモートシステム上のファイルにアクセスすることができます。

本セクションでは、**FTP** プロトコルの基本およびRed Hat Enterprise Linux で推奨される**FTP** サーバーの**vsftpd** について概説します。

16.2.1. ファイル転送プロトコル (FTP)

FTP は、クライアント/サーバーアーキテクチャーを使用し、**TCP** ネットワークプロトコルを使用してファイルを転送します。**FTP** は古いプロトコルであることから、暗号化されていないユーザー名とパスワード認証を使用します。このため、安全でないプロトコルとみなされており、絶対的に必要でない限

り、使用するべきではありません。しかし、**FTP** はインターネット上で非常に普及しているので、共有ファイルの公開で必要となる場合がよくあります。このため、システム管理者は、**FTP** プロトコルの特性を認識しておくべきです。

本セクションでは、vsftpd を設定して**TLS** による安全を確保する接続の確立方法と、SELinux を用いて**FTP** サーバーを安全にする方法を説明しています。**FTP** の代用となるのは、OpenSSH スイートからの sftp です。OpenSSH の設定方法および**SSH** プロトコル全般に関する情報は、12章OpenSSH を参照してください。

インターネット上で使用されているほとんどのプロトコルとは異なり、**FTP** が正しく機能するためには複数のネットワークポートを必要とします。**FTP** クライアントアプリケーションが**FTP** サーバーへの接続を開始する際に、**コマンドポート**として知られるポート 21 をサーバー上で開きます。このポートは、すべてのコマンドをサーバーに発行するために使用されます。サーバーから要求されたデータはいずれも**データポート** を介してクライアントに返されます。データ接続が開始されるデータ接続用ポートの番号は、クライアントがactive またはpassive のモードでデータを要求するかによって異なります。

これらのモードの定義は以下のとおりです。

アクティブモード

アクティブモードは、**FTP** プロトコルでクライアントへのデータ転送に使用される独自の方法です。**FTP** クライアントがアクティブモードのデータ転送を開始すると、サーバーは、サーバー上のポート 20 からクライアントの指定する **IP** アドレスと、ランダムで権限のないポート (1024 以上) への接続を開きます。この方法では、クライアントマシンがポート 1024 以上での接続を受け入れるように許可されている必要があります。インターネットのようなセキュリティー保護されていないネットワークが増加するにともない、ファイアウォールを使用したクライアントマシンの保護が普及しています。このようなクライアント側のファイアウォールは、アクティブモードの**FTP** サーバーから着信する接続を拒否する場合が多いため、パッシブモードが考案されました。

パッシブモード

パッシブモードはアクティブモードと同様に、**FTP** クライアントアプリケーションによって開始されます。サーバーからのデータを要求する際に、**FTP** クライアントはパッシブモードでデータにアクセスしたいことを知らせると、サーバーはサーバー上の **IP** アドレスとランダムな非特権ポート (1024 以上) を提供します。クライアントは、サーバー上のそのポートに接続して要求した情報をダウンロードします。
パッシブモードは、クライアント側のファイアウォールによるデータ接続障害の問題を解決しますが、サーバー側のファイアウォール管理を複雑化させてしまう場合があります。**FTP** サーバー上の特権のないポートの範囲を制限することにより、サーバー上で開いておりポート数を減らすことができます。またこの方法により、サーバーを対象としたファイアウォールのルール設定の手順が簡略化されます。

16.2.2. vsftpd サーバー

Very Secure FTP Daemon(**vsftpd**) は、高速で安定性があり、また重要な点として安全性を確保するため、土台から設計されています。**vsftpd** は、多数の接続を効率的かつ安全に処理できるため、Red Hat Enterprise Linux とともに配布されるのはスタンドアロン **FTP** サーバーのみです。

vsftpd で使用されるセキュリティーモデルには、以下に挙げる 3 つの主要な側面があります。

- **特権プロセスと非特権プロセスの確固たる分離**: 別個のプロセスが異なるタスクを処理します。各プロセスは、そのタスクに必要な最低限の権限で稼働します。
- **高い権限を必要とするタスクを、必要最小限の権限を伴うプロセスで処理**: **libcap** ライブラリー内にある互換性を利用して、通常は完全な root 権限を必要とするタスクを、権限が低いプロセスでより安全に実行できます。

- **ほとんどのプロセスを chroot jail で実行:** *可能な場合は常に、プロセスは、ルートディレクトリーを共有ディレクトリーに変更します。すると、このディレクトリーは、**chroot jail** と見なされます。たとえば、**/var/ftp/** ディレクトリーが主要な共有ディレクトリーの場合、**vsftpd** は **/var/ftp/** を新しいルートディレクトリー **(/)** に再割り当てします。これにより、新たな **root** ディレクトリー下に格納されていないディレクトリーに対する、ハッカーの潜在的な悪質行為を行うことができなくなります。*

*これらのセキュリティープラクティスを使用すると、**vsftpd** によるリクエスト対応方法に以下のような影響があります。*

- **親プロセスは、必要最小限の権限で稼働:** *親プロセスは、リスクレベルを最低限に抑えるために必要とされる権限のレベルを動的に算出します。子プロセスは、**FTP** クライアントとの直接的なインタラクションを処理し、可能な限り権限なしに近い形で稼働します。*

- **高い権限を必要とするオペレーションはすべて、小さな親プロセスによって処理:** *Apache **HTTP** Server の場合とほぼ同様に、**vsftpd** は権限のない子プロセスを起動し、着信接続を処理します。これにより、権限のある親プロセスを最小限に抑えられ、比較的少ないタスクを処理することになります。*

- **親プロセスは、権限のない子プロセスからのリクエストはどれも信頼しない:** *子プロセスとの通信はソケット上で受信し、子プロセスからの情報の有効性は動作を実施する前に確認されます。*

- **FTP クライアントとのインタラクションの大半は、chroot jail 内の権限のない子プロセスによって処理:** *これらの子プロセスには権限がなく、共有ディレクトリーへのアクセスしかないため、プロセスがクラッシュした際に攻撃者がアクセスできるのは共有ファイルのみです。*

16.2.2.1. vsftpd の起動と停止

*現行のセッションで **vsftpd** サービスを起動するには、シェルプロンプトで **root** として以下を入力します。*

```
~]# systemctl start vsftpd.service
```

*現在のセッションでサービスを停止するには、**root** で以下を入力します。*

```
~]# systemctl stop vsftpd.service
```

***vsftpd** サービスを再起動するには、**root** で以下のコマンドを実行します。*

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

*このコマンドは停止し、即座に **vsftpd** サービスを起動します。これは、この **FTP** サーバーの設定ファイルを編集した後に設定変更を行う最も効率的な方法です。では、以下のコマンドを使用して、すでに実行している場合にのみ、**vsftpd** サービスを再起動することができます。*

```
~]# systemctl try-restart vsftpd.service
```

*デフォルトでは、**vsftpd** サービスがブート時に自動的に起動することは**ありません**。ブート時に **vsftpd** サービスが起動するようにするには、**root** でシェルプロンプトに以下を入力します。*

```
~]# systemctl enable vsftpd.service
Created symlink from /etc/systemd/system/multi-user.target.wants/vsftpd.service to
/usr/lib/systemd/system/vsftpd.service.
```

Red Hat Enterprise Linux 7 でシステムサービスを管理する方法は、10章systemd によるサービス管理を参照してください。

16.2.2.2. vsftpd の複数コピーの起動

*1 台のコンピューターを複数の **FTP** ドメインに使用する場合があります。これは、**マルチホーミング** と呼ばれるテクニックです。**vsftpd** を使用してマルチホーミングを行う方法の1 つに、デーモンの複数コピーを実行し、各コピーに設定ファイルを与える方法があります。*

*これを行うには、最初に、関連するすべての **IP** アドレスをシステム上のネットワークデバイスまたはエイリアスネットワークデバイスに割り当てます。ネットワークデバイス、デバイスのエイリアス、およびネットワーク設定スクリプトの追加情報は、Red Hat Enterprise Linux 7 Networking Guideを参照してください。*

*次に、**FTP** ドメインのDNS サーバーが正しいマシンを参照するように設定する必要があります。BIND、Red Hat Enterprise Linux で使用されている**DNS** プロトコル実装、設定ファイルの詳細は、Red Hat Enterprise Linux 7 Networking Guideを参照してください。*

***vsftpd** が異なる **IP** アドレスにおけるリクエストに応答するには、デーモンの複数コピーが実行中である必要があります。**vsftpd** デーモンの複数インスタンスの起動を促進するために、特別なsystemd サービスユニット (**vsftpd@.service**) が**vsftpd** 起動用にインスタンス化されたサービスとしてvsftpd パッケージ内で提供されています。*

*このサービスユニットを活用するには、**FTP** サーバーで必要な各インスタンスの個別の**vsftpd** 設定ファイルを作成し、それを **/etc/vsftpd/** ディレクトリーに格納する必要があります。これらの設定ファイルは、(**/etc/vsftpd/vsftpd-site-2.conf** などの) 一意の名前を持ち、**root** ユーザーのみが読み取り、書き込み可能とする必要があることに注意してください。*

***IPv4** ネットワーク上で待機している各 **FTP** サーバーの設定ファイル内で、以下のディレクティブは一意のものである必要があります。*

```
listen_address=N.N.N.N
```

N.N.N.N を、使用中のFTP サイト用の一意の **IP** アドレスに置き換えます。サイトが**IPv6** を使用している場合は、代わりに **listen_address6** ディレクティブを使用します。

*複数の設定ファイルを **/etc/vsftpd/** ディレクトリーに格納しておけば、**vsftpd** デーモンの個別インスタンスは、**root** で以下のコマンドを実行すると開始できます。*

```
~]# systemctl start vsftpd@configuration-file-name.service
```

*上記のコマンドで、configuration-file-name を、**vsftpd-site-2** などの、要求しているサーバーの設定ファイルの一意の名前に置き換えます。設定ファイルの **.conf** 拡張子は、コマンドに含めないことに注意してください。*

***vsftpd** デーモンの複数インスタンスを同時に開始する場合は、vsftpd パッケージに含まれる、systemd ターゲットユニットファイル (**vsftpd.target**) を活用することができます。このsystemd ターゲットでは、**/etc/vsftpd/** ディレクトリーで利用できる各 **vstpd** 設定ファイルに対して、独立した **vsftpd** デーモンが起動します。**root** として次のコマンドを実行し、ターゲットを有効にします。*

```
~]# systemctl enable vsftpd.target
Created symlink from /etc/systemd/system/multi-user.target.wants/vsftpd.target to
/usr/lib/systemd/system/vsftpd.target.
```

上記のコマンドは、システムの起動時に、(設定された **vsftpd** サーバーインスタンスとともに) **vsftpd** サービスを起動するように systemd サービスマネージャーを設定します。システムを再起動することなく、サービスをすぐに開始するには、**root** で以下のコマンドを実行します。

```
~]# systemctl start vsftpd.target
```

systemd ターゲットを使用してサービスを管理する方法は、「systemd ターゲットでの作業」 を参照してください。

サーバーごとに変更するディレクティブには、以下のものがあります。

- **anon_root**
- **local_root**
- **vsftpd_log_file**
- **xferlog_file**

16.2.2.3. TLS を使用した vsftpd 接続の暗号化

ユーザー名、パスワード、およびデータを暗号化せずに送信する **FTP** の従来の不安定な性質に対抗するために、**vsftpd** デーモンは **TLS** プロトコルを使用して接続を認証し、すべての送信を暗号化するように設定できます。**TLS** をサポートする **FTP** クライアントは、**TLS** が有効になっている **vsftpd** と通信する必要があることに注意してください。

注記

SSL (Secure Sockets Layer) は、セキュリティープロトコルの古い実装の名前です。新規のバージョンは **TLS** (Transport Layer Security) と呼ばれています。**SSL** にはセキュリティーに関する深刻な脆弱性があるため、新規のバージョン (**TLS**) のみを使用してください。vsftpd サーバーに付随するドキュメントや **vsftpd.conf** ファイルで使用される設定ディレクティブでは、セキュリティー関連の項目を参照する際に **SSL** の名前を使用しますが、**TLS** はサポートされており、**ssl_enable** ディレクティブが **YES** に設定されているときにデフォルトで使用されています。

vsftpd.conf ファイルの **ssl_enable** 設定ディレクティブを **YES** に設定して、**TLS** サポートを有効にします。**ssl_enable** オプションが有効になると自動的にアクティブになる、その他の **TLS** 関連のディレクティブのデフォルト設定により、合理的に適切に設定された **TLS** のセットアップが提供されます。たとえば、全接続に **TLS** v1 プロトコルの使用を必須とする (安全でない **SSL** プロトコルバージョンはデフォルトで無効になります) ことや、非匿名の全ログインでパスワードおよびデータ送信での **TLS** の使用を強制することなどです。

例16.14 TLS を使用するように vsftpd の設定

以下の例では、設定ディレクティブは **vsftpd.conf** ファイルでセキュリティープロトコルの古い **SSL** バージョンを明示的に無効にします。

```
ssl_enable=YES
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
```

設定を変更したら、**vsftpd** サービスを再起動します。

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

vsftpd で **TLS** の使用を調整するためのその他の **TLS** 関連の設定ディレクティブについては、*vsftpd.conf(5)* マニュアルページを参照してください。

16.2.2.4. vsftpd 用の SELinux ポリシー

(他の **ftpd** プロセスとともに) **vsftpd** デーモンを管理する SELinux ポリシーは、強制アクセス制御を定義します。これはデフォルトでは、最低限必要なアクセスに基づいています。**FTP** デーモンが特定のファイルやディレクトリーにアクセスできるようにするには、それらに適切なラベルを割り当てる必要があります。

たとえば、ファイルを匿名で共有できるようにするには、**public_content_t** ラベルを共有するファイルおよびディレクトリーに割り当てる必要があります。**chcon** コマンドを **root** で使用すると、これが可能になります。

```
~]# chcon -R -t public_content_t /path/to/directory
```

上記のコマンドでは、/path/to/directory を、ラベルを割り当てるディレクトリーへのパスに置き換えます。同様に、ファイルのアップロード用にディレクトリーを設定する場合は、その特定のディレクトリーに **public_content_rw_t** ラベルを割り当てる必要があります。さらに、SELinux のブール値オプション **allow_ftpd_anon_write** を **1** に設定する必要があります。以下のように、**setsebool** コマンドを **root** で実行します。

```
~]# setsebool -P allow_ftpd_anon_write=1
```

ローカルユーザーが **FTP** (Red Hat Enterprise Linux 7 のデフォルト設定) を介してホームディレクトリーにアクセスできるようにするには、**ftp_home_dir** ブール値オプションを **1** に設定する必要があります。**vsftpd** をスタンドアロンモードで実行できるようにするには (Red Hat Enterprise Linux 7 ではデフォルトで有効)、**ftpd_is_daemon** オプションも **1** に設定する必要があります。

他の有用なラベルやブール値オプションの例や **FTP** に関する SELinux ポリシーの設定方法の詳細情報は、ftpd_selinux(8) man ページを参照してください。SELinux 全般に関する詳細情報は、*Red Hat Enterprise Linux 7 SELinux User's and Administrator's Guide* も併せて参照してください。

16.2.3. 関連情報

vsftpd の詳細情報は、以下の参考資料をご覧ください。

16.2.3.1. インストールされているドキュメント

- **/usr/share/doc/vsftpd-**version-number**/** ディレクトリー: version-number を、インストールした vsftpd パッケージのバージョンに置き換えます。このディレクトリーには、ソフトウェアの基本情報を記載した **README** ファイルが格納されています。**TUNING** ファイルには基本的なパフォーマンス調整のヒント、そして **SECURITY/** ディレクトリーには **vsftpd** で使用されているセキュリティーモデルに関する情報が含まれています。
- **vsftpd** 関連の man ページ: デーモンおよび設定ファイルに関する man ページは多数あります。以下は、その中でも重要な man ページです。

  サーバーアプリケーション

  ```
  {blank}
  ```

–

- *vsftpd(8): **vsftpd** で利用可能なコマンドラインオプションを説明しています。*

設定ファイル

> *{blank}*

- *vsftpd.conf(5): **vsftpd** の設定ファイル内で利用可能なオプションの詳細なリストを格納しています。*
- *hosts_access(5): **hosts.allow** および **hosts.deny** の **TCP** ラッパーの設定ファイルで利用可能なフォーマットとオプションを説明します。*

SELinux とのインタラクション

> *{blank}*

- *ftpd_selinux(8): **ftpd** プロセスを管理するSELinux ポリシーと、SELinux ラベルの割り当て方法、およびブール値セットが説明されています。*

16.2.3.2. オンラインドキュメント

vsftpd およびFTP 全般について

> *{blank}*

- *http://vsftpd.beasts.org/: **vsftpd** プロジェクトページは、最新のドキュメントやソフトウェアの作成者の連絡先を入手することができる便利なサイトです。*
- *http://slacksite.com/other/ftp.html: この Web サイトでは、アクティブモードと passive-mode **FTP** の相違点を簡単に説明します。*

Red Hat Enterprise Linux のドキュメンテーション

> *{blank}*

- *Red Hat Enterprise Linux 7 ネットワークガイド: Red Hat Enterprise Linux 7 の **ネットワークガイド** では、このシステムにおけるネットワークインターフェイス、ネットワーク、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。**hostnamectl** ユーティリティーの概要のほか、これを使用してコマンドラインでホスト名を表示したり、設定する方法を説明しています。*
- *Red Hat Enterprise Linux 7 SELinux ユーザーおよび管理者のガイド: Red Hat Enterprise Linux 7 の SELinux **ユーザーおよび管理者のガイド** では、SELinux の原則と、SELinux を Apache HTTP Server やPostfix、PostgreSQL、OpenShift などの様々なサービスに設定して使用する方法が詳細に説明されています。また、SELinux アクセスパーミッションを **systemd** が管理するシステムサービス用に設定する方法も説明しています。*
- *Red Hat Enterprise Linux 7 セキュリティーガイド - Red Hat Enterprise Linux 7 の **セキュリティーガイド** は、ユーザーおよび管理者が、ローカルおよびリモートからの侵入、悪用、悪意のある行為に対してワークステーションおよびサーバーを保護するプロセスとプラクティスを学習する際に役に立ちます。また、重大なシステムサービスを保護する方法についても説明しています。*

関連RFC ドキュメント

> *{blank}*

- RFC 0959*: IETF からの**FTP** プロトコルのオリジナルの*Request for Comments*(RFC)*。
- RFC 1123*: 短い**FTP** 関連のセクションで、RFC 0959 を拡張、明確化します。*
- RFC 2228*: **FTP** セキュリティー拡張機能。vsftpd は、TLS 接続および*SSL 接続のサポートに必要な小規模のサブセットを実装します。
- RFC 2389*: **FEAT** および**OPTS** コマンドを指定します。*
- RFC 2428*: **IPv6** サポート。*

16.3. 印刷設定

Print Settings ツールを使用すると、プリンター設定、プリンター設定ファイルの管理、印刷スプールディレクトリー、印刷フィルター、プリンタークラスの管理ができます。

このツールは CUPS (Common Unix Printing System) を基本にしています。CUPS を使用した以前の Red Hat Enterprise Linux バージョンからシステムをアップグレードする場合、アップグレードプロセスは設定を行ったプリンターの設定を保持します。

重要

***cupsd.conf** の man ページには、CUPS サーバーの設定が記載されています。これには、**SSL** サポートを有効にするためのディレクティブが含まれています。ただし、CUPS では使用されるプロトコルバージョンのコントロールが許可されません。Resolution for POODLE SSLv3.0 vulnerability (CVE-2014-3566) for components that do not allow SSLv3 to be disabled via configuration settings (設定から SSLv3 を無効にできないコンポーネントで POODLE SSLv3.0 脆弱性 (CVE-2014-3566) を解決する方法) に説明されている脆弱性により、Red Hat はセキュリティー保護のためにこれに依存しないことを推奨しています。stunnel を使用してセキュアなトンネルを提供し、**SSLv3** を無効にすることが推奨されます。stunnel の使用方法についての詳細は、Red Hat Enterprise Linux 7 セキュリティーガイド を参照してください。*

*リモートシステムの印刷設定 ツールへのアドホックのセキュアな接続は、「X11 転送」の記載通りに、**SSH** で X11 転送を使用します。*

注記

*CUPS Web アプリケーションまたはコマンドラインから直接、同一および追加の動作をプリンターで実行できます。Web ブラウザーでアプリケーションにアクセスするには、http://localhost:631/ にアクセスします。CUPS マニュアルについては、この Web サイトの **Home** タブのリンクからご覧ください。*

16.3.1. 印刷設定の設定ツールの起動

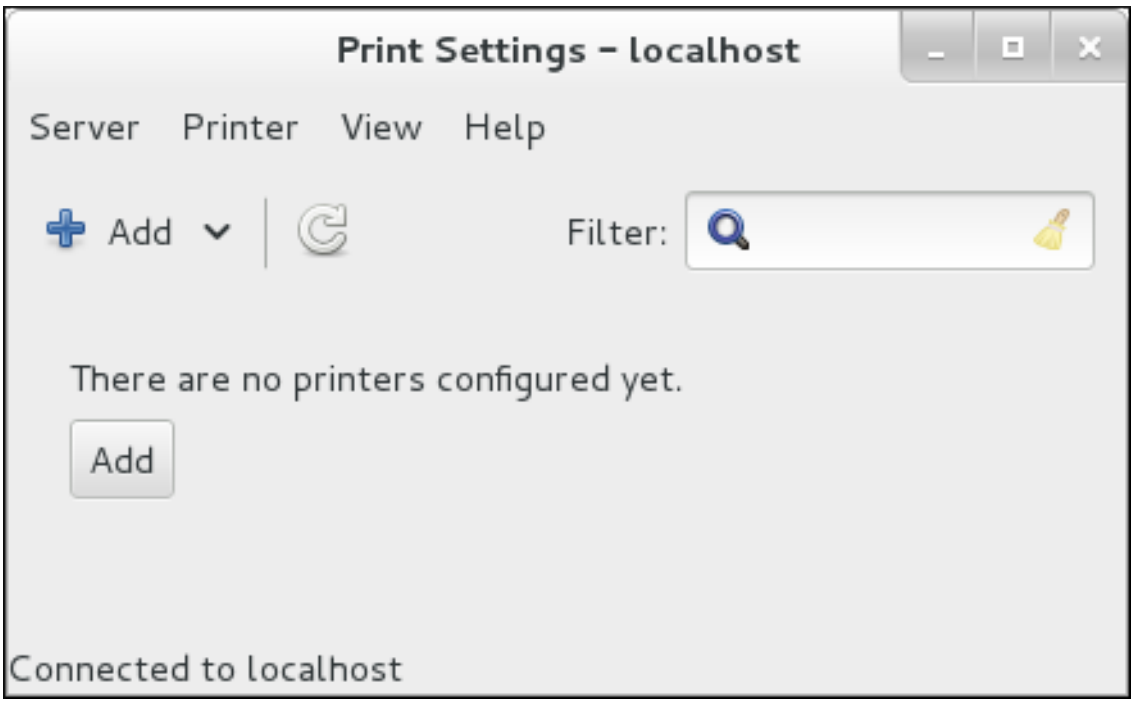
印刷設定 設定ツールを使用すると、既存のプリンターで様々な操作を実行したり、新規プリンターを設定できます。CUPS を直接使用することもできます(CUPS Web アプリケーションにアクセスするには http://localhost:631/ *を参照)。*

*コマンドラインから Print Settings ツールを起動するには、シェルプロンプトで **system-config-printer***

と入力します。この結果、印刷設定 ツールが表示されます。もしくは、GNOME デスクトップを使用している場合は、**Super** キーを押してアクティビティーの概要に入り、**Print Settings** と入力し、**Enter** を押します。この結果、印刷設定 ツールが表示されます。**Super** キーはキーボードまたはその他のハードウェアに応じて様々なキーで表示されますが、多くの場合、**Windows** または **Command** キーとして通常は **Spacebar** の左側に表示されます。

図16.1「印刷設定ウィンドウ」 にあるように 印刷設定 ウィンドウが表示されます。

図16.1 印刷設定ウィンドウ



16.3.2. プリンター設定の開始

プリンターの設定プロセスは、プリンターキューのタイプにより異なります。

USB に接続されているローカルプリンターを設定すると、プリンターが検出され、自動的に追加されます。インストールするパッケージの確認、管理者の指定、または **root** ユーザーパスワードの入力が求められます。他のポートタイプに接続したローカルプリンターやネットワークプリンターの場合は、手動で設定する必要があります。

プリンターを手動で設定するには、以下の手順に従います。

1. 印刷設定ツールを起動します (「印刷設定の設定ツールの起動」 を参照)。
2. **Server** → **New** → **Printer** に移動します。
3. **Authenticate** ダイアログボックスで、管理者または **root** ユーザーのパスワードを入力します。リモートプリンターの初回設定時の場合は、ファイアウォールの調整の承認を求めるプロンプトが出されます。
4. プリンターの接続タイプを選択し、右側エリアでその詳細を記入します。

16.3.3. ローカルプリンターの追加

以下の手順に従って、シリアルポート以外に接続されたローカルプリンターを追加します。

1. 追加 プリンターダイアログを開きます (「プリンター設定の開始」 を参照)。

2. デバイスが自動的に表示されない場合は、左側のリストでプリンターを接続するポートを選択してください (**Serial Port #1**、**LPT #1** など)。

3. 右側で、接続プロパティーを入力します。

   Other の場合

   URI (例: file:/dev/lp0)

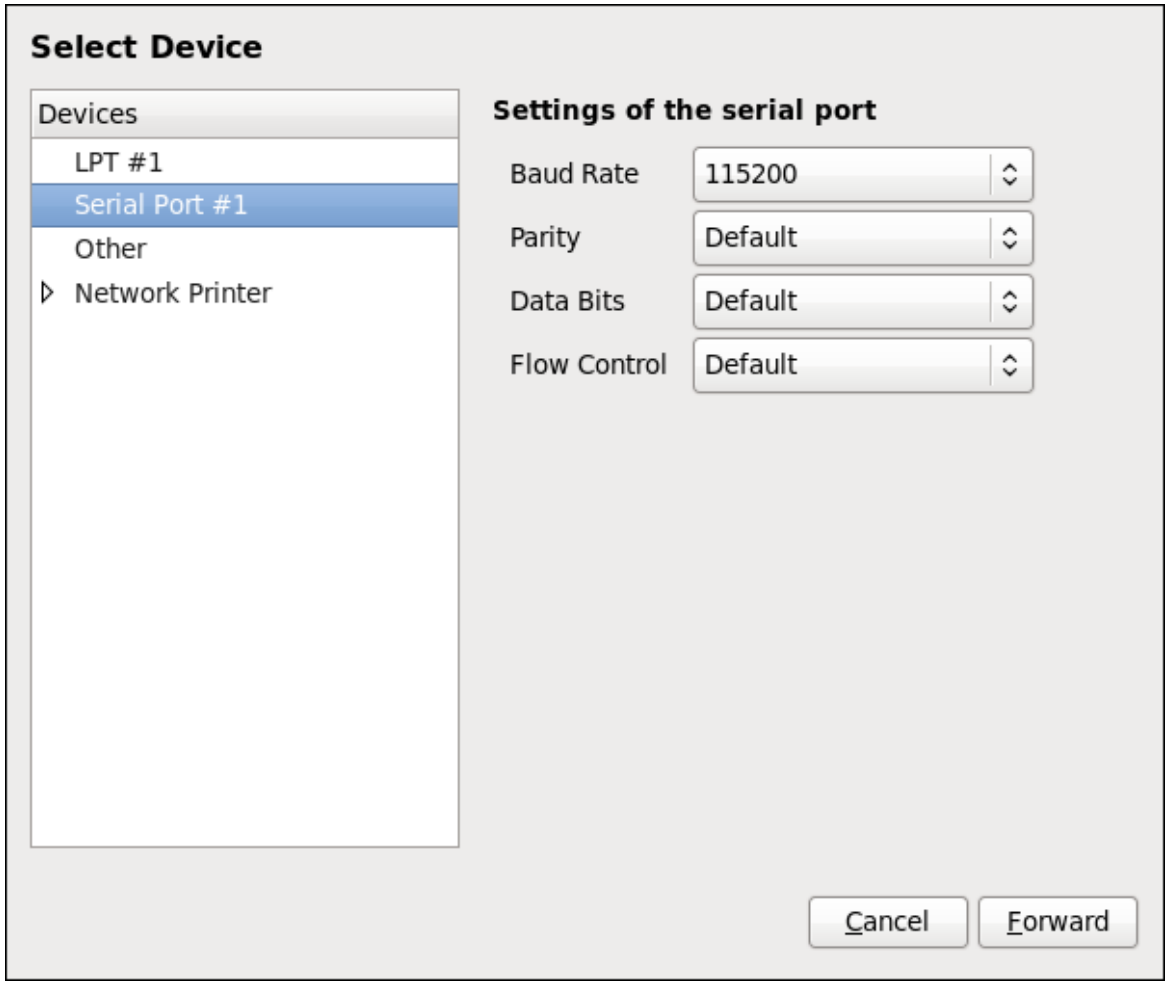
   Serial Port の場合

   通信速度
   パリティー

   データビット

   フロー制御

   図16.2 ローカルプリンターの追加



4. **Forward** をクリックします。

5. プリンターのモデルを選択します。詳しくは「プリンターモデルの選択と完了」を参照してください。

16.3.4. AppSocket/HP JetDirect プリンターの追加

以下の手順に従って AppSocket/HP JetDirect プリンターを追加します。

1. 新規プリンター ダイアログを開きます(「印刷設定の設定ツールの起動」 を参照)。
2. 左側のリストで**Network Printer**→**AppSocket/HP JetDirect** の順に選択します。
3. 右側で、接続設定を入力します。

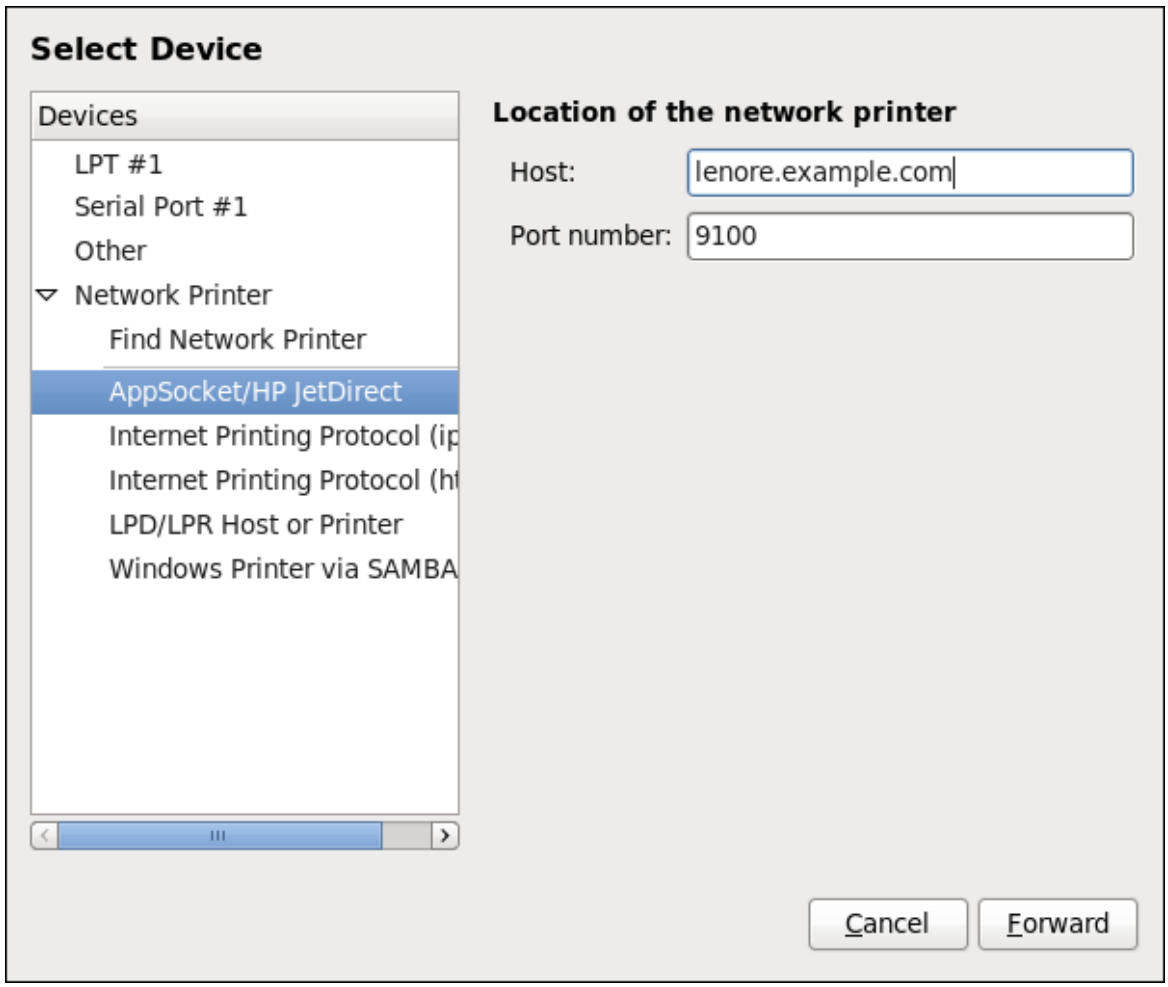
   Hostname

   プリンターホスト名または**IP** アドレス。

   Port Number

   印刷ジョブをリッスンするプリンターポート(デフォルトは**9100**)

   図16.3 JetDirect プリンターの追加



4. **Forward** をクリックします。
5. プリンターのモデルを選択します。詳しくは 「プリンターモデルの選択と完了」 を参照してください。

16.3.5. IPP プリンターの追加

IPP プリンターは同じ**TCP/IP** ネットワークにある異なるシステムに接続されているプリンターです。このプリンターが取り付けられているシステムは**CUPS** を実行しているか、単に**IPP** を使用するように設定されているだけです。

プリンターサーバーでファイアウォールが有効な場合は、ポート**631** で受信**TCP** 接続が可能になるようにファイアウォールを設定する必要があります。プロトコルを参照する**CUPS** により、クライアントマシンは共有**CUPS** キューを自動的に検出することが可能です。これを有効にするには、クライアント

マシンのファイアウォールをポート **631** で受信 **UDP** パッケージを許可するよう設定する必要があります。

以下の手順にしたがって **IPP** プリンターを追加します。

1. 新規プリンター ダイアログを開きます (「プリンター設定の開始」 を参照)。
2. 左側のデバイスのリストで、ネットワークプリンターおよび、**Internet Printing Protocol (ipp)** または **Internet Printing Protocol (https)**. の順に選択します。
3. 右側で、接続設定を入力します。

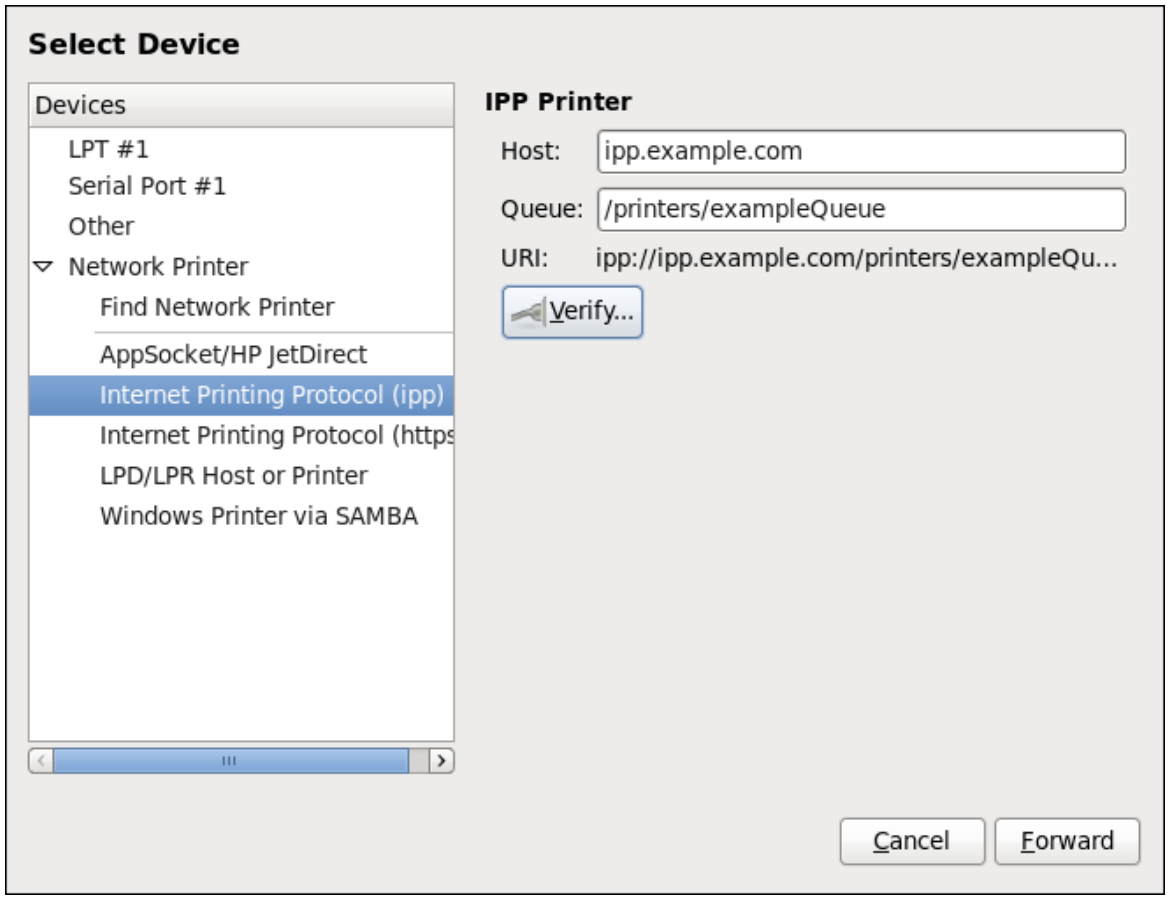
   Host

   IPP プリンターのホスト名。

   Queue

   新規のキューに与えるキューの名前です (このボックスを空白のままにすると、デバイスノードに基づいた名前が使用されます)。

   図16.4 IPP プリンターの追加



4. 進む をクリックして続けます。
5. プリンターのモデルを選択します。詳しくは 「プリンターモデルの選択と完了」 を参照してください。

16.3.6. LPD/LPR Host or Printer の追加

以下の手順に従って **LPD/LPR** ホストまたはプリンターを追加します。

1. 新規プリンター ダイアログを開きます(「プリンター設定の開始」 を参照)。
2. 左のデバイスリストから、**Network Printer**→**LPD/LPR Host or Printer** の順に選択します。
3. 右側で、接続設定を入力します。

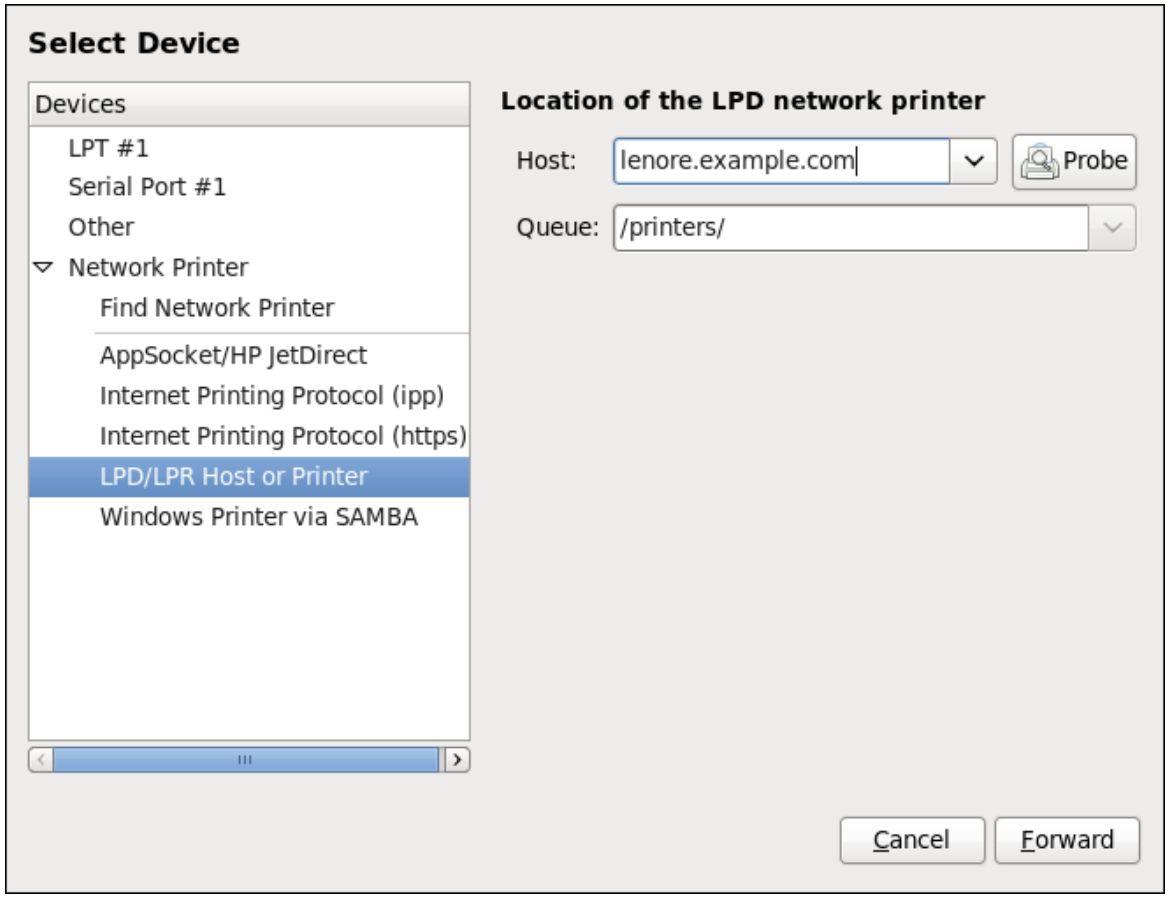
   Host

   LPD/LPR プリンターまたはホストのホスト名
   必要に応じて、**Probe** をクリックし、LSD ホスト上のキューを検索します。

   Queue

   新規のキューに与えるキューの名前です(このボックスを空白のままにすると、デバイスノードに基づいた名前が使用されます)。

   図16.5 LPD/LPR プリンターの追加



4. 進む をクリックして続けます。
5. プリンターのモデルを選択します。詳しくは 「プリンターモデルの選択と完了」 を参照してください。

16.3.7. Samba (SMB) プリンターの追加

Samba プリンターを追加するには、以下の手順を実行します。

注記

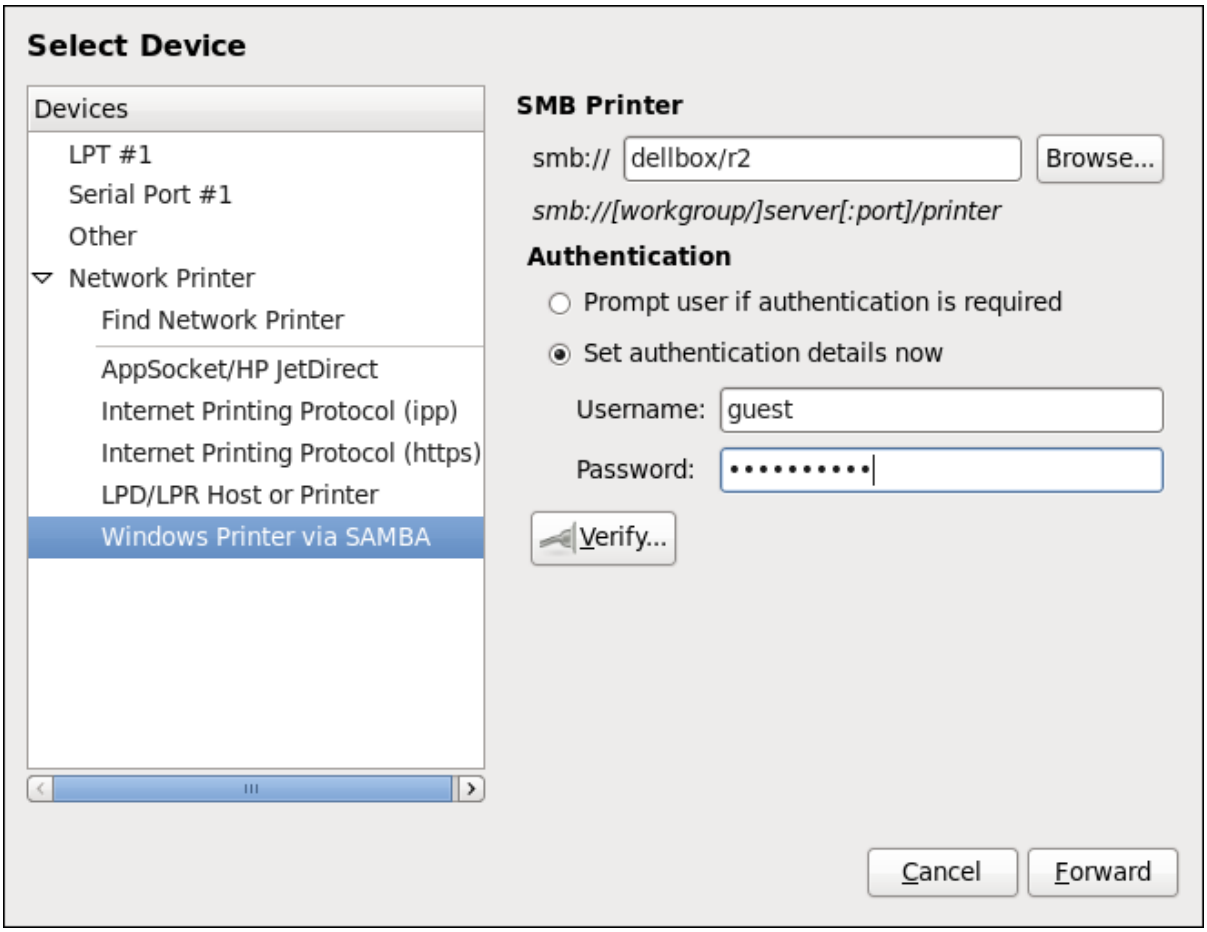
*Samba プリンターを追加するには、samba-client パッケージをインストールしておく必要があります。**root** で以下のコマンドを実行してこれを実行できます。*

```
yum install samba-client
```

Yum を使用したパッケージのインストールの詳細は、「パッケージのインストール」 を参照して下さい。

1. *新規プリンター ダイアログを開きます (「プリンター設定の開始」 を参照)。*
2. *左側のリストで、**Network Printer → Windows Printer via SAMBA** の順に選択します。*
3. ***smb://** フィールドに SMB アドレスを入力します。入力形式は*computer name/printer share *にします。図16.6「SMB プリンターの追加」 では、*computer name *は **dellbox**、*printer share *は **r2** です。*

   図16.6 SMB プリンターの追加



4. *利用できるワークグループやドメインを確認するには、閲覧する **(Browse)** をクリックします。特定のホストのキューだけを表示させるには、ホスト名 (NetBios 名) を入力して、閲覧する をクリックします。*
5. *以下のオプションのいずれかを選択します。*
   a. *認証が必要な場合はプロンプトを表示する **(Prompt user if authentication is required)** を選択すると、文書を印刷する際に、ユーザー名とパスワードを入力することが必要になります。*

b. *ここで認証の詳細を設定する* ***(Set authentication details now)*** *を選択すると、前もって認証情報を提供するため後で入力する必要がなくなります。* ***Username*** *フィールドには、プリンターにアクセスするユーザー名を入力します。このユーザーは、SMB システムで存在している必要があり、ユーザーはプリンターへのアクセス権限を持っている必要があります。デフォルトのユーザー名は通常、Windows のサーバーでは* ***guest***、*あるいは Samba サーバーでは* ***nobody*** *です。*

6. *(必要な場合は) ユーザー名 フィールドに指定したユーザーのパスワード を入力します。*

警告

Samba プリンターのユーザー名とパスワードは、 ***root*** *および* ***lpd*** *(Linux Printing Daemon) が読み取り可能な非暗号化ファイルとしてプリンターサーバーに保存されています。そのため、プリンターサーバーに* ***root*** *アクセスを持つ他のユーザーは、Samba プリンターへのアクセスに使用するユーザー名とパスワードを閲覧することができます。*

Samba プリンターへアクセスするためのユーザー名とパスワードを選択する場合は、ローカルの Red Hat Enterprise Linux システムにアクセスする時に使用するパスワードとは異なるパスワードを選ぶことを推奨します。

Samba プリンターサーバーで共有するファイルがある場合も、印刷キューで使用されるパスワードとは異なるパスワードを使用することが推奨されます。

7. *確認* ***(Verify)*** *をクリックし、接続をテストします。確認が成功すると、ダイアログボックスが表示され、プリンター共有のアクセスを確認します。*

8. ***Forward*** *をクリックします。*

9. *プリンターのモデルを選択します。詳しくは 「プリンターモデルの選択と完了」 を参照してください。*

16.3.8. プリンターモデルの選択と完了

適切なプリンターの接続タイプを選択すると、システムはドライバーを取得するよう試行します。プロセスが失敗した場合は、ドライバーリソースを手動で検索できます。

以下の手順に従い、プリンタードライバーを設定してインストールを完了します。

1. *ドライバーの自動検知が失敗すると、ウィンドウが表示されます。以下のオプションのいずれかを選択します。*

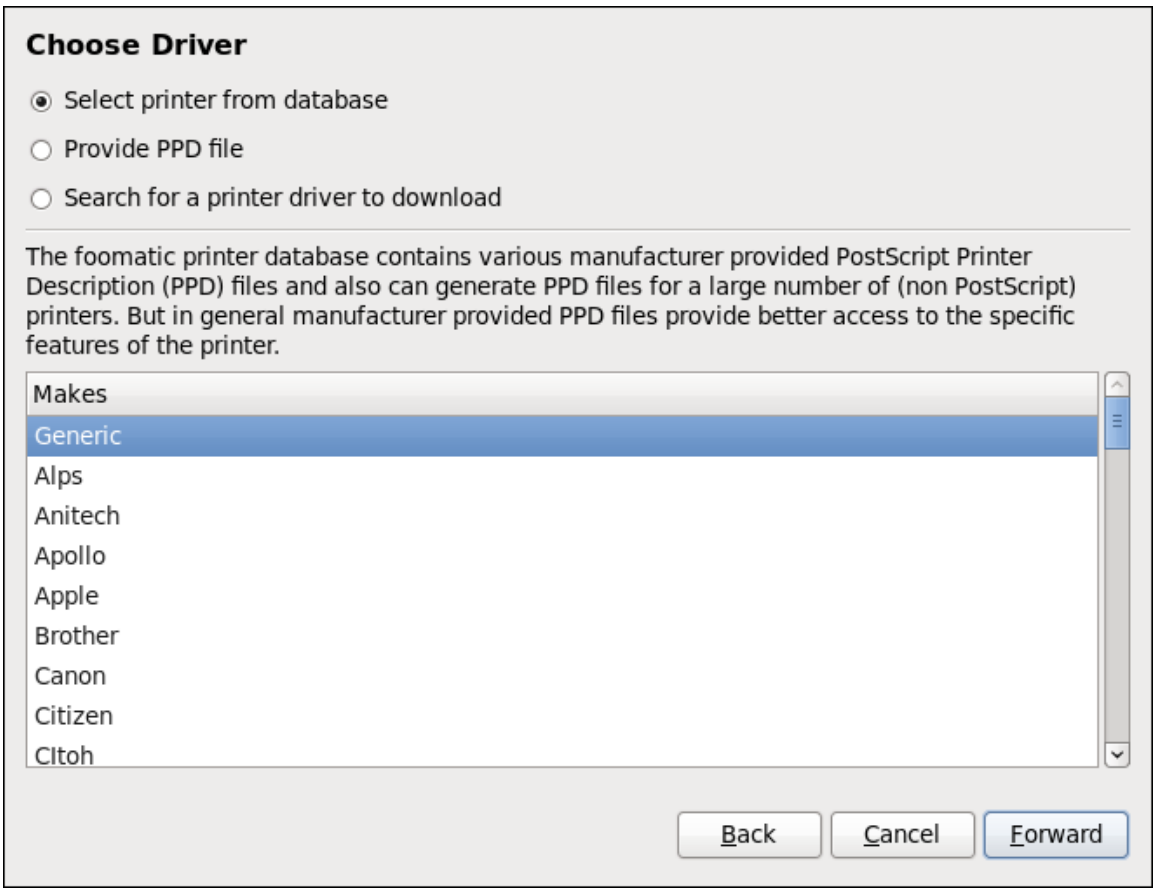
   a. *データベースからプリンターを選択* ***(Select a Printer from database)****: システムは、製造元のリストから、使用するプリンターの製造元に基づきドライバーを選択します。ご使用のプリンターのモデルがリストにない場合は、* ***Generic*** *を選択してください。*

   b. ***PPD*** *ファイルを提供* ***(Provide PPD file)****: システムは、備わっている* ***ポストスクリプトプリンターデスクリプション (PPD)*** *を使用します。PPD ファイルは製造元が通常提供するプリンターに同梱されています。PPD ファイルが利用可能な場合は、このオプションを選択*

し、オプションの詳細の下にあるブラウザーバーを使用して、PPD ファイルを選択できます。

c. ダウンロードするプリンタードライバーを検出 **(Search for a printer driver to download)**: 製造元とプリンターモデルを 製造元とモデル フィールドに入力し、OpenPrinting.org で適切なパッケージを検索します。

図16.7 プリンターブランドの選択

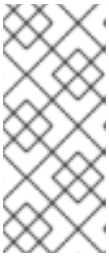


2. 上で選択した内容により、以下に表示される詳細は異なります。

   - データベースからプリンターを選択 オプションの場合は、プリンターブランドを表示
   - **PPD** ファイルを提供 オプションの場合は、PPD ファイルの場所を表示。
   - ダウンロードするプリンタードライバーを検出 オプションの場合は、プリンターの製造元とモデルを表示

3. 進む をクリックして続けます。

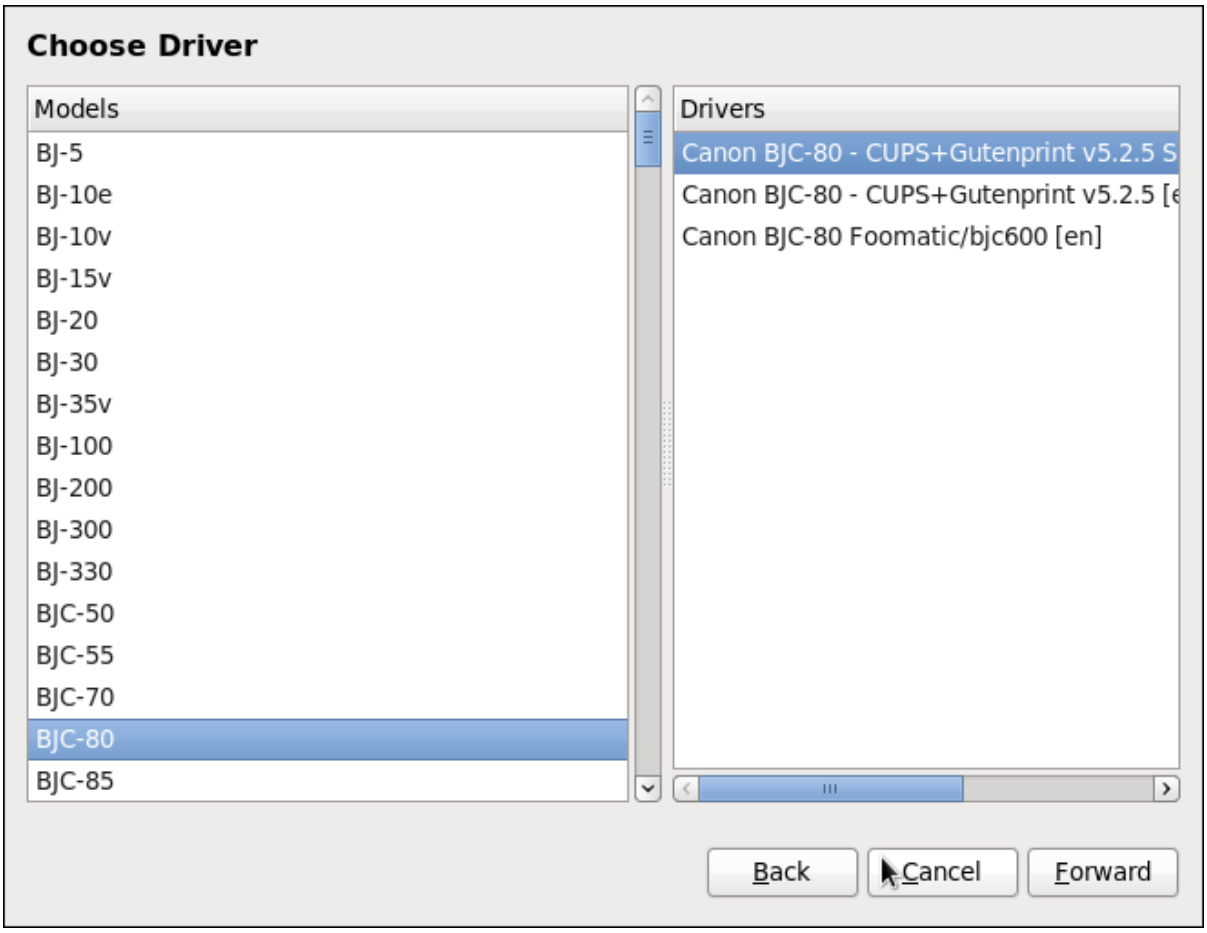
4. 選択したオプションが該当する場合は、図16.8「プリンターモデルの選択」 のようなウィンドウが表示されます。左側の モデル の列で該当するモデルを選択します。



注記

右側で、推奨される印刷ドライバーが自動的に選択されています。ただし、別の利用可能なドライバーを選択することもできます。ただし、別の利用可能なドライバーを選ぶことも可能です。ローカルプリンターはコンピューターに直接接続されているため、プリンターに送信されるデータを処理するにはプリンタードライバーが必要です。

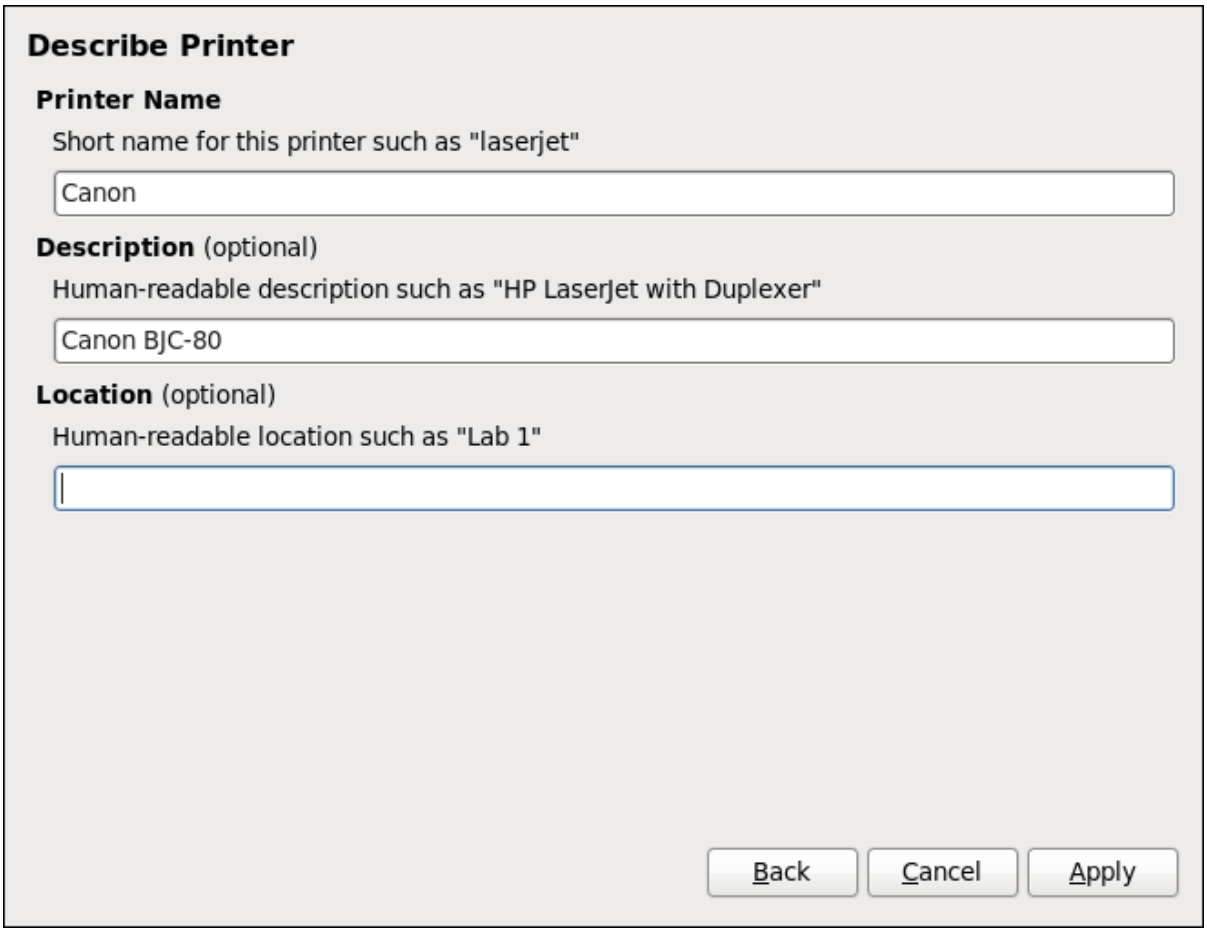
図16.8 プリンターモデルの選択



5. ***Forward*** *をクリックします。*

6. *プリンターの説明* ***(Describe Printer)*** *の下の、プリンター名* ***(Printer Name)*** *フィールドに一意のプリンター名を入力します。名前には文字、数字、ハイフン* ***(-)***、*アンダースコア* ***(_)*** *を使用できますが、* **スペースは使用できません**。*また、説明* ***(Description)*** *フィールドと場所* ***(Location)*** *フィールドに詳細情報を追加することもできます。どちらもオプションで、スペースを入れることは可能です。*

図16.9 プリンターの設定



7. 設定が正しければ、適用 **(Apply)** をクリックして、ご使用のプリンター設定を確認し、印刷キューを追加できます。戻る **(Back)** をクリックすると、プリンター設定を変更できます。

8. 変更が適用されると、テストページの印刷を行うダイアログボックスが表示されます。**Yes** をクリックするとテストページが印刷されます。「テストページの印刷」 の記載通りに、テストページを後で印刷することもできます。

16.3.9. テストページの印刷

プリンターを設定、またはプリンターの設定を変更した後は、テストページを印刷して、プリンターが適切に機能していることを確認します。

1. 印刷 **(Printing)** ウィンドウでプリンターを右クリックし、プロパティー **(Properties)** をクリックします。

2. プロパティーウィンドウで、左側の 設定 **(Settings)** をクリックします。

3. 表示されている 設定 タブで、テストページの印刷 **(Print Test Page)** ボタンをクリックします。

16.3.10. 既存プリンターの修正

既存のプリンターを削除するには、印刷設定 ウィンドウで該当するプリンターを選択し、**Printer** → **Delete** の順に選択します。プリンターの削除を確認します。別の方法として、**Delete** キーを押しても削除できます。

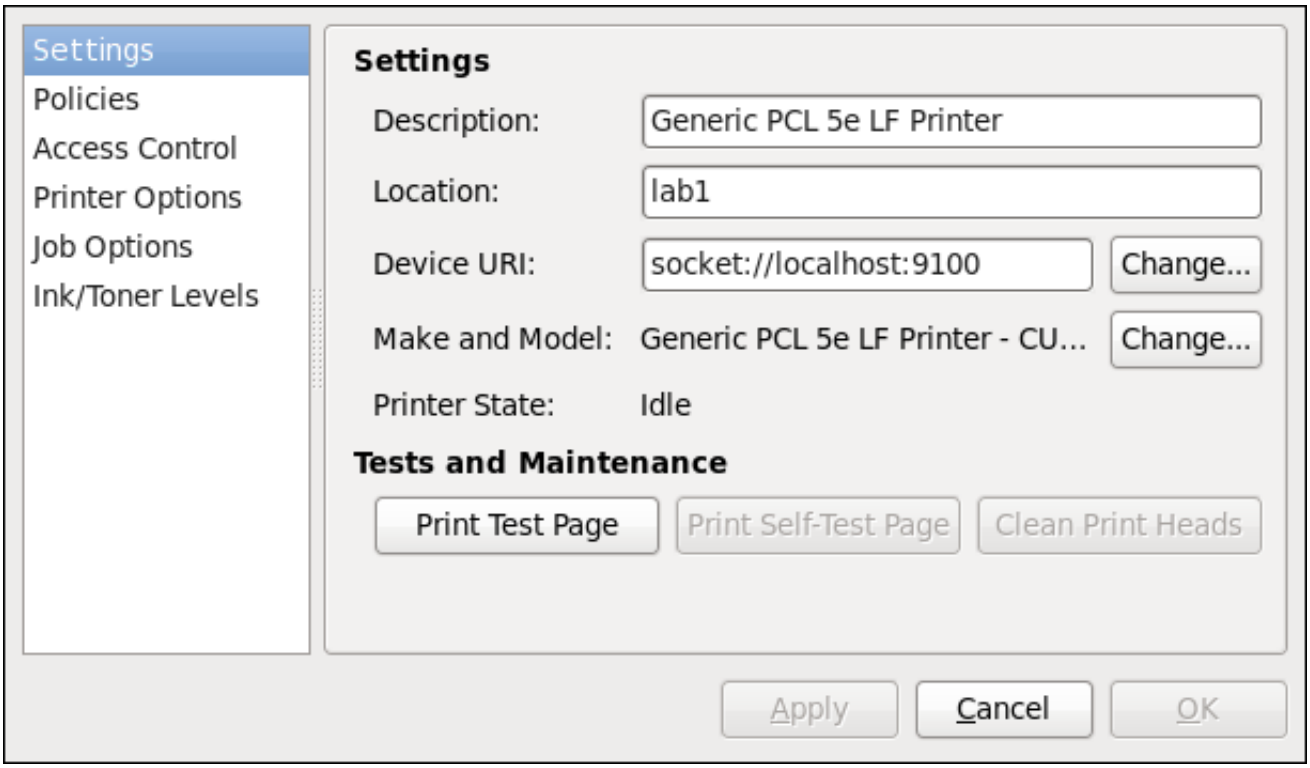
デフォルトのプリンターを設定するには、プリンターのリストで該当するプリンターを右クリックし、コンテキストメニューの デフォルトに設定 ボタンをクリックします。

16.3.10.1. 設定ページ

プリンターのドライバー設定を変更するには、プリンター リストで該当する名前をダブルクリックします。そして、左側の 設定 ラベルをクリックし、設定 ページを表示します。

製造元やモデルなどのプリンター設定の変更、テストページの印刷、デバイスの場所 (URI) の変更など行うことができます。

図16.10 設定ページ



16.3.10.2. ポリシーページ

プリンターの状態や出力を変更するには、左側の ポリシー (Policies) ボタンをクリックします。

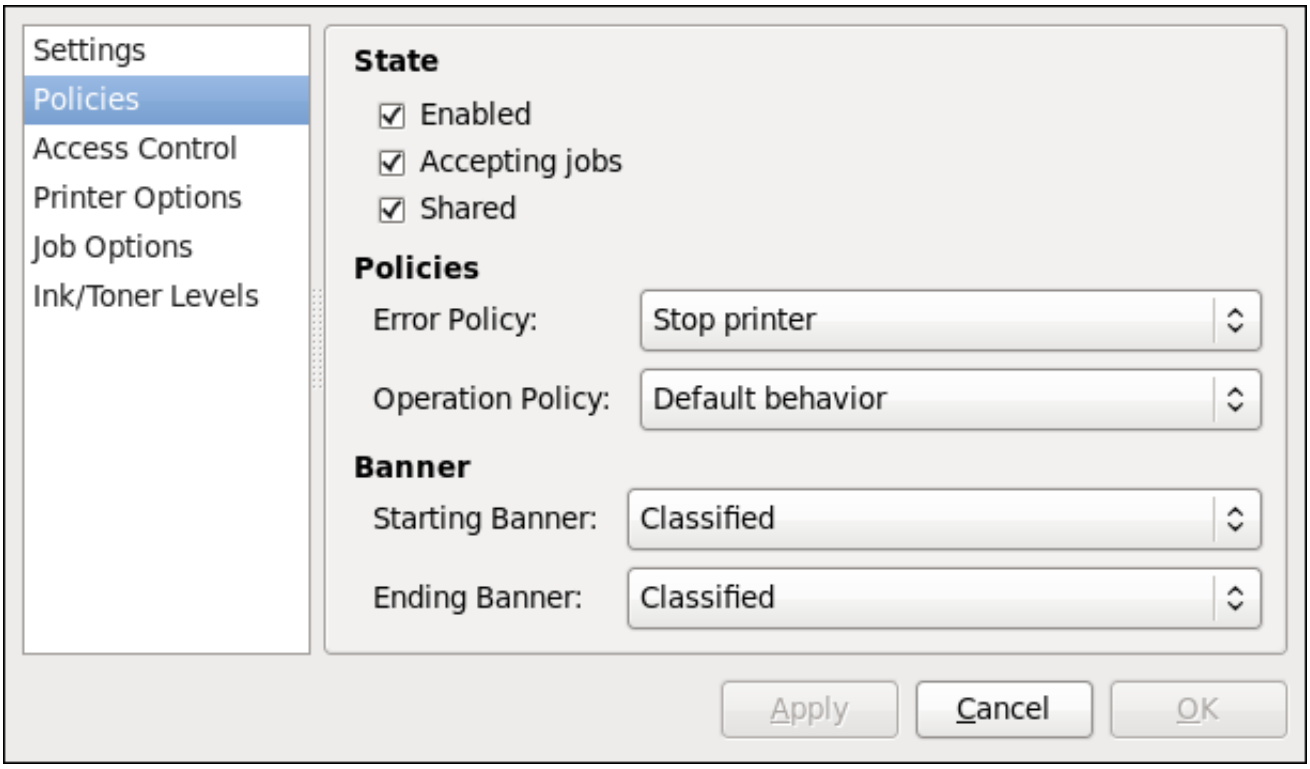
プリンターの状態を選択したり、プリンターの エラーポリシー (Error Policy) を設定できます (エラーが発生した場合は、印刷ジョブを中止、再試行、停止できます)。

バナーページ (banner page) (送信元プリンター、ジョブを開始したユーザー名、印刷中の文書のセキュリティー状態など、印刷ジョブの特徴を説明するページ) の作成も可能です。開始バナー (**Starting Banner**) または 終了バナー (**Ending Banner**) のドロップメニューをクリックし、印刷ジョブの性質に最適なオプションを選択します (秘密 (**confidential**) など)。

16.3.10.2.1. プリンターの共有

ポリシー ページでは、プリンターを共有としてマークできます。プリンターが共有されている場合は、ネットワーク上で公開されているユーザーはそれを使用できます。プリンターの共有機能を有効にするには、Server → **Settings** の順にクリックし、このシステムに接続されている共有プリンターを公開 (**Publish shared printers connected to this system**) を選択します。

図16.11 ポリシーページ



*ファイアウォールがポート **631** への受信 **TCP** 接続、Network Printing Server (**IPP**) のポートを許可していることを確認してください。Red Hat Enterprise Linux 7 のファイアウォールを通過する **IPP** トラフィックを許可するには、**firewalld** の **IPP** サービスを使用します。これを実行するには、以下を行います。*

firewalld での IPP サービスの有効化

1. *グラフィカルの firewall-config ツールを起動するには、**Super** キーを押してアクティビティーの概要に入り、**firewall** と入力して **Enter** を押します。**Firewall Configuration** ウィンドウが開きます。次に管理者または **root** のパスワード入力が求められます。*
   *または、コマンドラインを使用してグラフィカルなファイアウォール設定ツールを起動するには、**root** で以下のコマンドを入力します。*

   ```
   ~]# firewall-config
   ```

   ***Firewall Configuration** ウィンドウが開きます。*

   *ウィンドウ左下の接続しましたの表示を確認してください。これは、firewall-config ツールがユーザースペースデーモン **firewalld** に接続されていることを示します。*

   現行のファイアウォール設定をただちに変更するには、設定 というラベルのドロップダウン選択メニューが 実行時 に設定されていることを確認します。または、システムの次回の起動時またはファイアウォールの再読み込み時に設定が適用されるように編集するには、ドロップダウンリストから 永続 を選択します。

2. ***Zones** タブを選択してから、使用されるネットワークインターフェイスに対応するファイアウォールゾーンを選択します。デフォルトは **public** ゾーンです。**Interfaces** (インターフェイス) タブには、ゾーンに割り当てられたインターフェイスが表示されます。*

3. ***Services** (サービス) タブを選択してから **ipp** サービスを選択して共有を有効にします。**ipp-client** サービスがネットワークプリンターへのアクセスに必要です。*

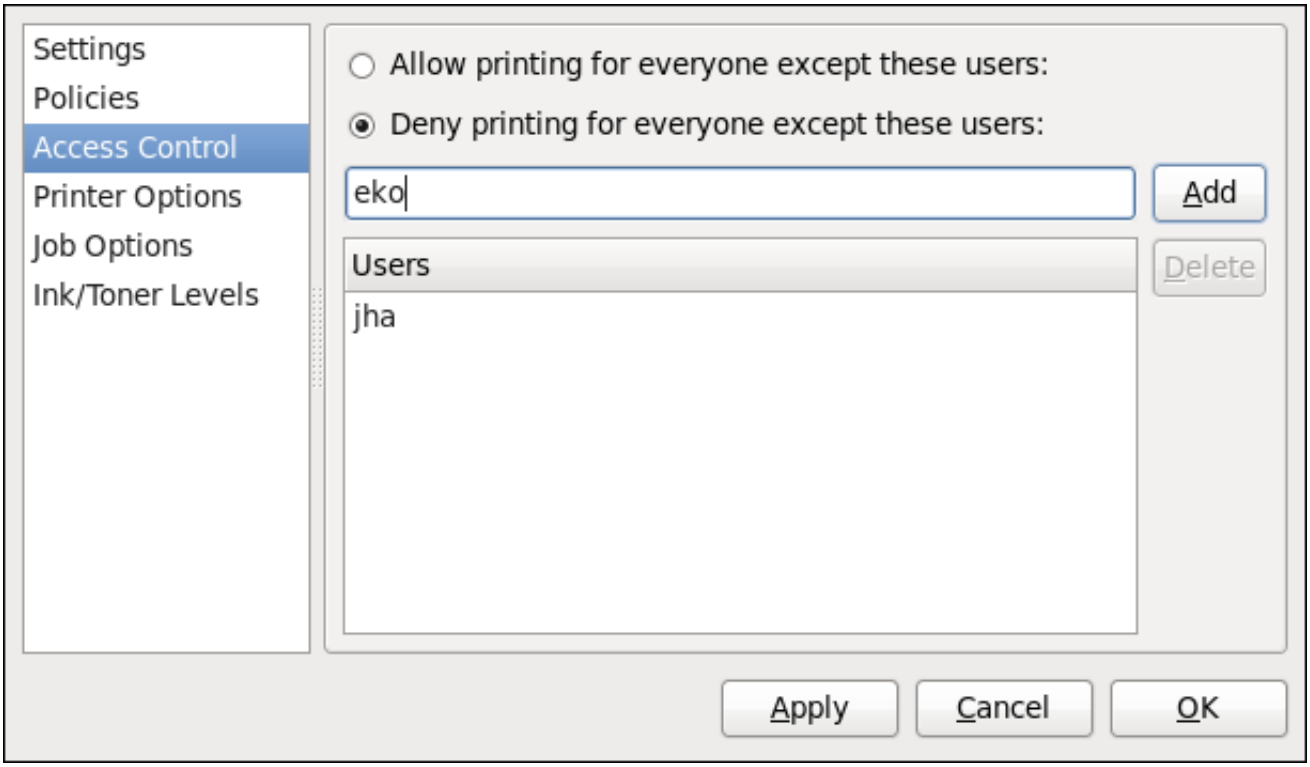
4. *firewall-config* ツールを起動します。

firewalld でのポートのオープンおよびクローズの詳細については、*Red Hat Enterprise Linux 7 Security Guide* を参照してください。

16.3.10.2.2. アクセス制御のページ

アクセス制御 ページで設定したプリンターへのユーザーレベルのアクセスを変更できます。左側の *アクセス制御* **(Access Control)** のラベルをクリックするとページが表示されます。次のユーザーを除いて全員に印刷を許可する **(Allow printing for everyone except these users)** または次のユーザー以外の印刷を拒否する **(Deny printing for everyone except these users)** のどちらかを選択し、以下のようにユーザーセットを定義します。テキストボックスにユーザー名を入力し、*追加* **(Add)** ボタンをクリックしてユーザーセットにユーザーを追加します。

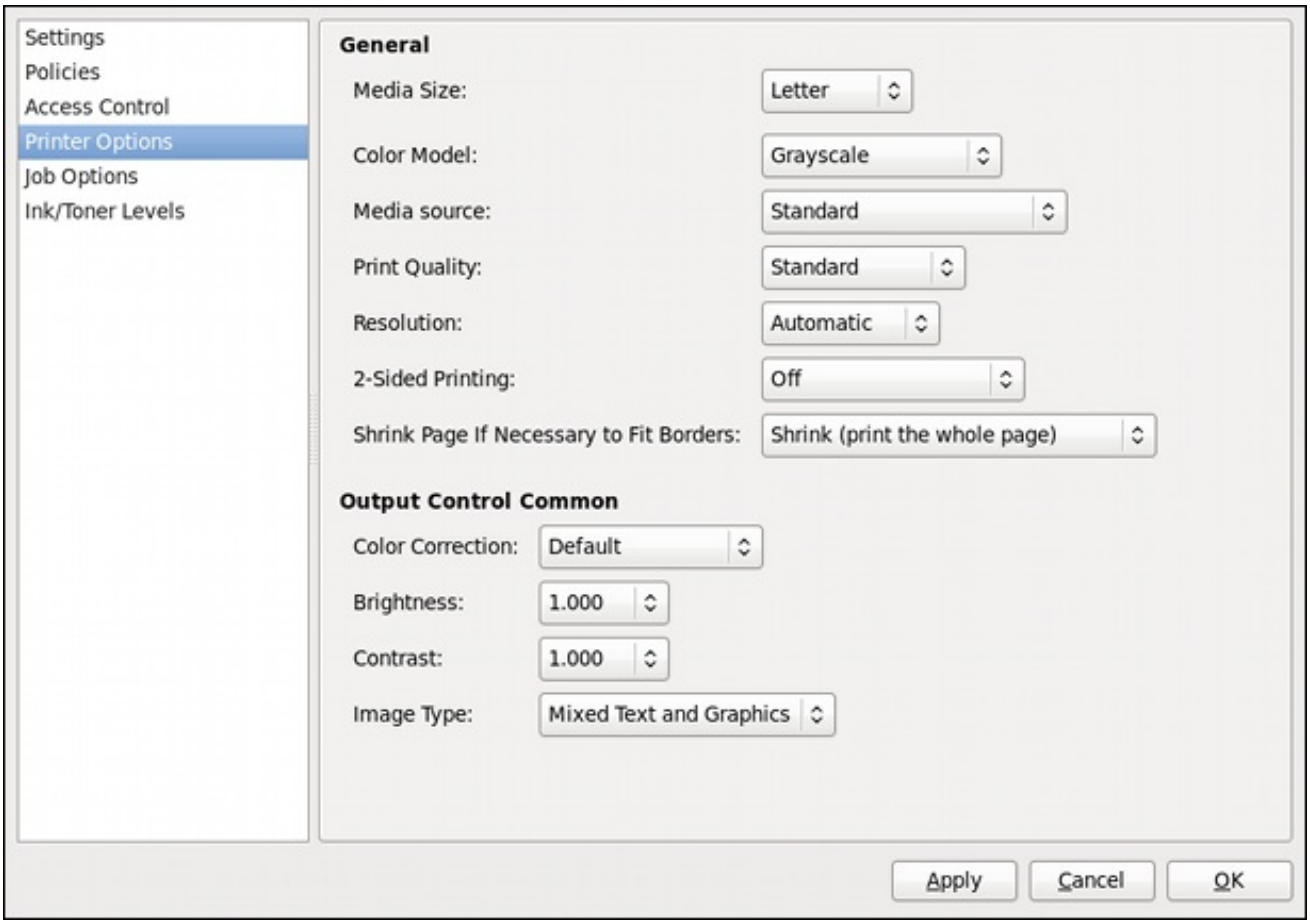
図16.12 アクセス制御のページ



16.3.10.2.3. プリンターオプションのページ

プリンターオプション **(Printer Options)** ページにはプリンターのメディアや出力の様々な設定オプションがあります。内容はプリンターごとに異なる場合があります。一般的な印刷の用紙、品質、サイズ設定が含まれます。

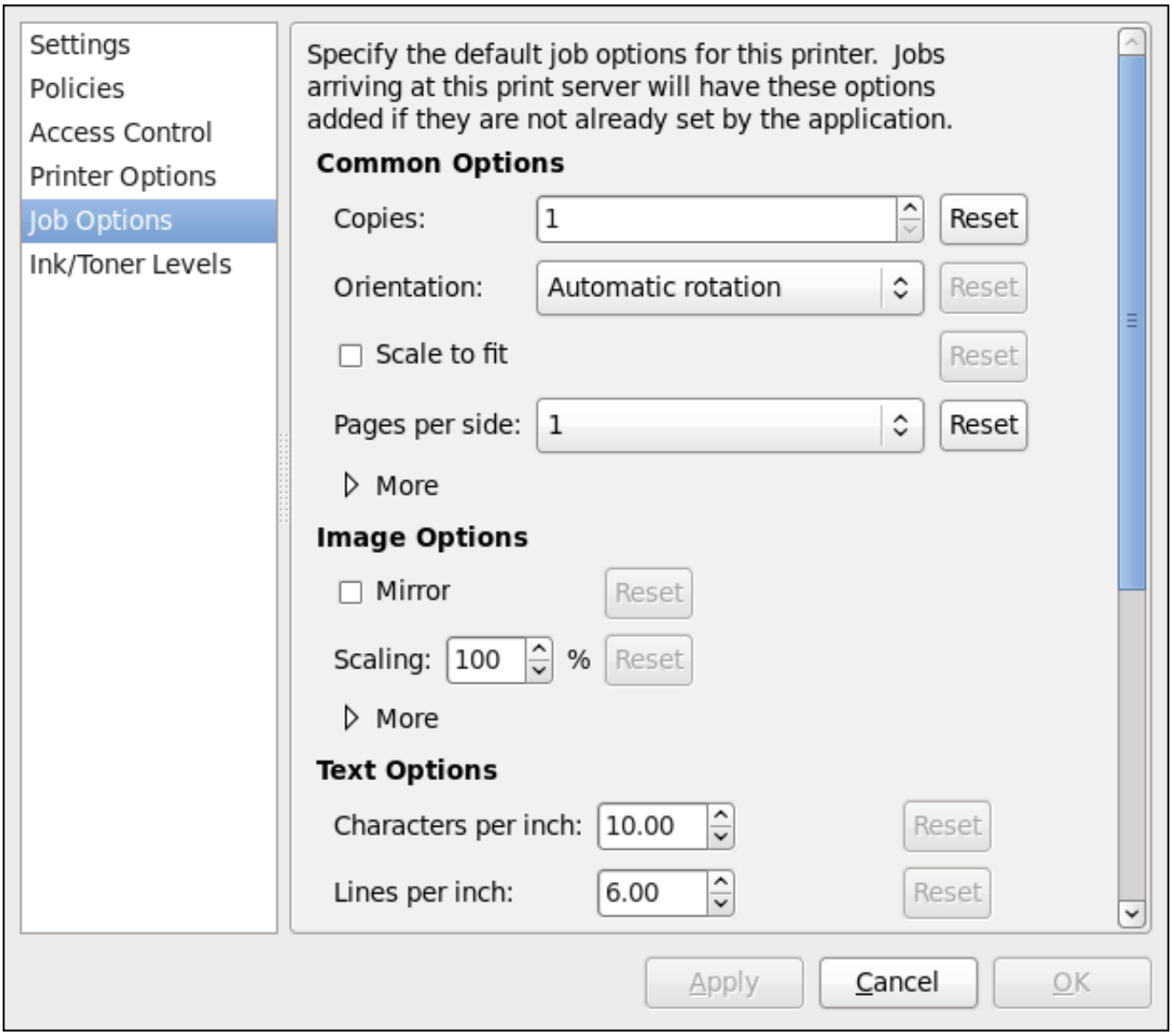
図16.13 プリンターオプションのページ



16.3.10.2.4. ジョブオプションページ

*ジョブオプション **(Job Options)** ページでは、プリンタージョブのオプションの詳細を設定できます。左側の ジョブオプション をクリックすると、ページが表示されます。デフォルト設定を編集し、部数、印刷の向き、スライドごとのページ、拡大縮小 (印刷可能領域のサイズを拡大または縮小して、サイズが印刷領域を超えるものを印刷媒体である用紙に合うようにします)、テキストオプションなど、カスタムのジョブオプションを適用します。*

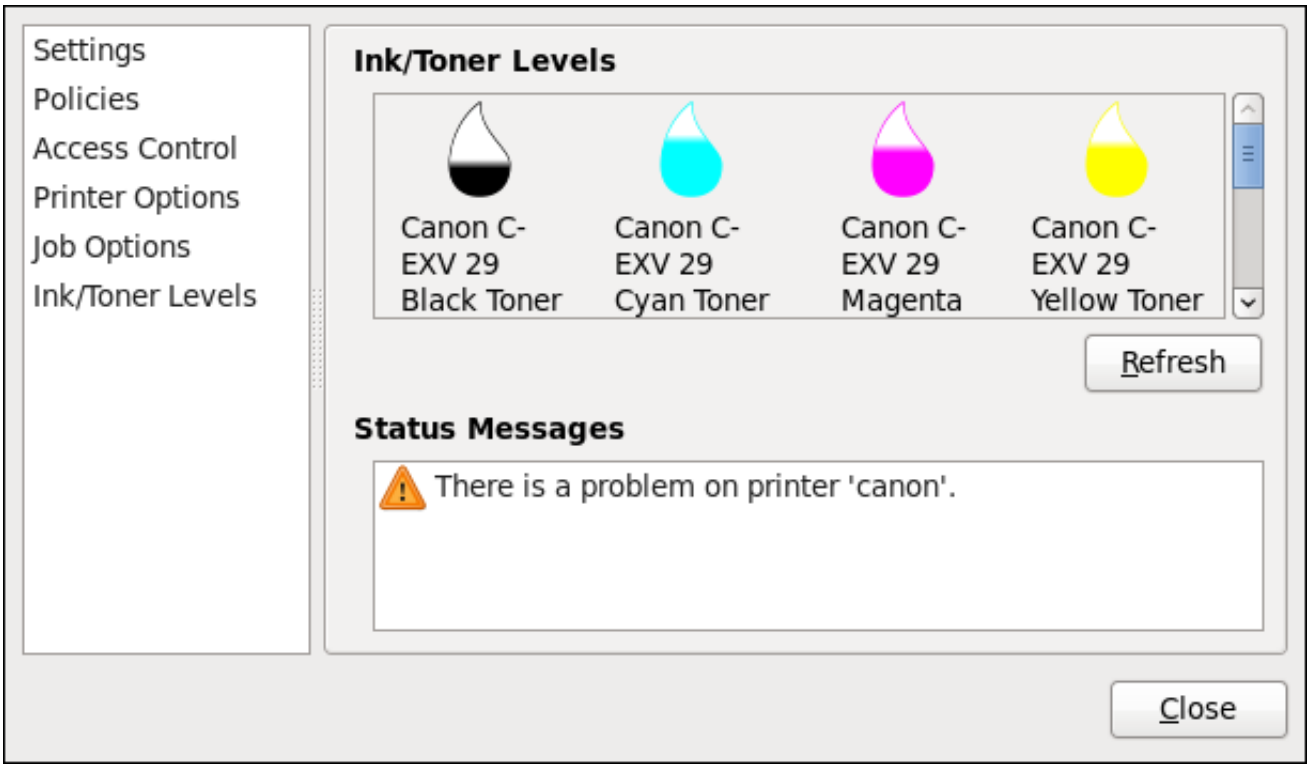
図16.14 ジョブオプションページ



16.3.10.2.5. Ink/Toner Levels ページ

*インク/トナーのレベル **(Ink/Toner Levels)** ページには、トナーの状態の詳細 (該当する場合)、およびプリンターの状態のメッセージが表示されます。左側の **Ink/Toner Levels** のラベルをクリックすると、ページが表示されます。*

図16.15 Ink/Toner Levels ページ

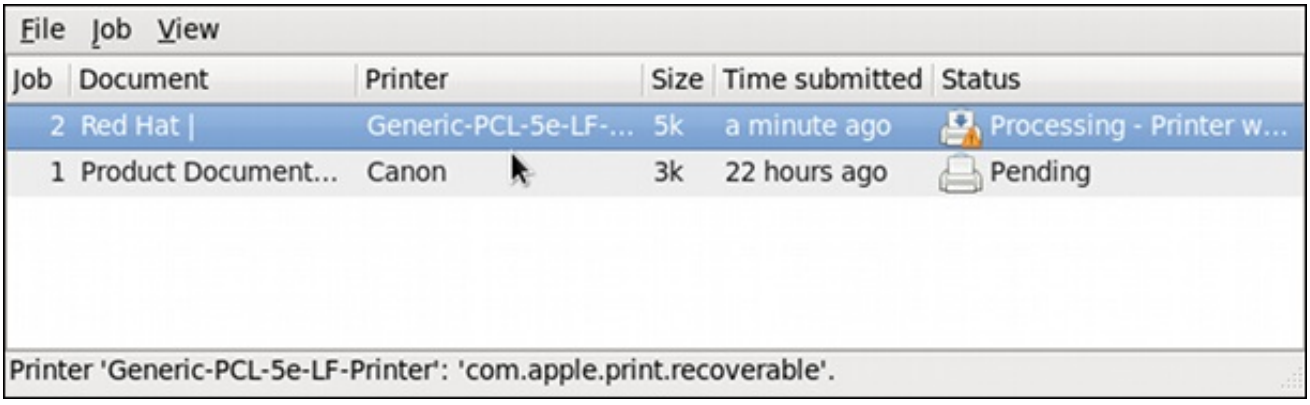


16.3.10.3. 印刷ジョブの管理

Emacs からのテキストファイルの印刷、GIMP からのイメージの印刷など、プリンターデーモンに印刷ジョブを送ると、印刷ジョブは印刷のスプールキューに追加されます。印刷のスプールキューはプリンターに送られた印刷ジョブのリストで、各印刷要求に関する情報 (印刷要求の状態、ジョブ番号など) を表示します。

*印刷が処理されている間、**Printer Status** のアイコンがパネルの Notification Area に表示されます。その **Printer Status** をクリックすると、図16.16「GNOME 印刷の状態」 のようなウィンドウが表示され、印刷ジョブの状態を確認できます。*

図16.16 GNOME 印刷の状態



印刷ジョブをキャンセル、保留、解除、再印刷、認証するためには、GNOME Print Status でジョブを選択し、ジョブメニューでそれぞれコマンドをクリックします。

*シェルプロンプトから印刷スプールの印刷ジョブのリストを表示させるには、**lpstat -o** コマンドを入力します。最後の数行は以下のようになります。*

> *例16.15 **lpstat -o** 出力の例*
>
> _

```
$ lpstat -o
Charlie-60       twaugh      1024  Tue 08 Feb 2011 16:42:11 GMT
Aaron-61         twaugh      1024  Tue 08 Feb 2011 16:42:44 GMT
Ben-62           root        1024  Tue 08 Feb 2011 16:45:42 GMT
```

印刷ジョブをキャンセルするには、コマンド **lpstat -o** コマンドで要求のジョブ番号を見つけ、**cancel job number** を使用します。たとえば、**cancel 60** は、例16.15「**lpstat -o** 出力の例」 で印刷ジョブをキャンセルします。**cancel** コマンドでは、その他のユーザーによって開始された印刷ジョブはキャンセルできません。ただし、**cancel -U root** job_number コマンドを使用して、強制的にジョブを削除することはできます。このようなキャンセルは、プリンターの操作ポリシーを 認証済 に変更し、**root** 認証を強制することで、禁止することができます。

シェルプロンプトから直接ファイルを印刷することもできます。たとえば **lp sample.txt** コマンドはテキストファイル **sample.txt** を印刷します。印刷フィルターはファイルのタイプを決定し、プリンターが理解できる形式に変換します。

16.3.11. 関連情報

Red Hat Enterprise Linux での印刷の詳細については、以下のリソースを参照してください。

インストールされているドキュメント

- **lp(1)**: コマンドラインからのファイルの印刷を可能にする **lp** コマンドの man ページです。
- **lpr(1)**: コマンドラインからのファイルの出力を可能にする **lpr** コマンドの man ページです。
- **cancel(1)**: 印刷キューから印刷ジョブを削除するためのコマンドユーティリティーの man ページです。
- **mpage(1)**: 1 枚の用紙に複数ページを印刷するためのコマンドラインユーティリティーの man ページです。
- **cupsd(8)**: CUPS プリンターデーモンの man ページです。
- **cupsd.conf(5)**: CUPS プリンターデーモンの設定ファイルの man ページです。
- **classes.conf(5)**: CUPS のクラス設定ファイルの man ページです。
- **lpstat(1)**: クラス、ジョブ、プリンターなどの状態情報を表示する **lpstat** コマンドの man ページです。

オンラインドキュメント

- http://www.linuxprinting.org/: Linux Foundation web サイトの OpenPrinting グループには、Linux の印刷に関する豊富な情報が記載されています。
- http://www.cups.org/: CUPS web サイトでは、CUPS のドキュメンテーション、FAQ、およびニュースグループについて記載しています。

第17章 データベースサーバー

本章では、MySQL テクノロジーに基づくオープンソースの高速で強力なデータベースサーバーである MariaDB サーバーのインストールと設定について説明します。本章では、MariaDB データのバックアップも説明します。

17.1. MARIADB

MariaDB は、データを構造化情報に変換して、データにアクセスする SQL インターフェイスを提供するリレーショナルデータベースです。これには、複数のストレージエンジンやプラグイン、地理情報システム (GIS) が含まれます。

Red Hat Enterprise Linux 7 には、MySQL データベースファミリーからのサーバーのデフォルトの実装として、MariaDB 5.5 が同梱されています。新しいバージョンのMariaDB データベースサーバーは、Red Hat Enterprise Linux 6 および Red Hat Enterprise Linux 7 の Software Collections として利用できます。最新バージョンの詳細は、Release Notes for Red Hat Software Collectionsを参照してください。

17.1.1. MariaDB サーバーのインストール

MariaDB サーバーをインストールするには、以下の手順に従います。

MariaDB のインストール

1. mariadb および mariadb-server パッケージが、必要なサーバーにインストールされていることを確認します。

```
~]# yum install mariadb mariadb-server
```

2. **mariadb** サービスを起動します。

```
~]# systemctl start mariadb.service
```

3. **mariadb** サービスが、システムの起動時に起動するようにします。

```
~]# systemctl enable mariadb.service
```

17.1.1.1. MariaDB インストールセキュリティーの改善

mysql_secure_installation コマンドを実行すると、MariaDB サーバーをインストールする際にセキュリティーを強化できます。

```
~]# mysql_secure_installation
```

このコマンドは、プロセスの各ステップを要求する完全にインタラクティブなスクリプトを起動します。このスクリプトを使用すると、次の方法でセキュリティーを改善できます。

- root アカウントのパスワードの設定
- 匿名ユーザーの削除
- リモート (ローカルホスト外) の root ログインの拒否

- テストデータベースの削除

17.1.2. MariaDB サーバーのネットワーク設定

MariaDB サーバーをネットワーク用に設定するには、**/etc/my.cnf.d/server.cnf** ファイルの **[mysqld]** セクションを使用します。ここで以下の設定ディレクティブを設定できます。

- bind-address
  bind-address は、サーバーがリッスンするアドレスです。

  可能なオプションは、ホスト名、IPv4 アドレス、または IPv6 アドレスです。

- skip-networking
  以下の値が使用できます。

  0 - すべてのクライアントをリッスンする

  1 - ローカルクライアントのみをリッスンする

- ポート
  MariaDB が TCP/IP 接続をリッスンするポート。

17.1.3. MariaDB データのバックアップ

MariaDB データベースからデータのバックアップを取得するには、以下の 2 つの方法があります。

- 論理バックアップ
- 物理バックアップ

17.1.3.1. 論理バックアップ

論理バックアップは、データの復元に必要な SQL ステートメントで設定されます。この種類のバックアップは、情報およびレコードをプレーンテキストファイルにエクスポートします。

物理バックアップに対する論理バックアップの主な利点は、移植性と柔軟性です。データは、物理バックアップではできない他のハードウェア設定である MariaDB バージョンまたはデータベース管理システム (DBMS) で復元できます。

警告

論理バックアップは、**mariadb.service** が実行中の場合にのみ実行できます。論理バックアップには、ログと設定ファイルが含まれません。

17.1.3.2. 物理バックアップ

物理バックアップは、コンテンツを格納するファイルおよびディレクトリーのコピーで設定されます。

物理バックアップは、論理バックアップと比較して、以下の利点があります。

- 出力が少なくなる。

- *バックアップのサイズが小さくなる。*
- *バックアップおよび復元が速くなる。*
- *バックアップには、ログファイルと設定ファイルが含まれる。*

警告

mariadb.service が実行していない場合、またはバックアップ中に変更されないようにデータベースのテーブルがすべてロックされている場合は、物理バックアップを実行する必要があります。

第18章 CHRONY スイートを使用した NTP 設定

IT では、多くの理由で正確な時間の維持が重要です。たとえばネットワーキングでは、パケットとログのタイムスタンプが正確であることが必要になります。Linux システムでは、**NTP** プロトコルがユーザー領域で実行しているデーモンにより実装されます。

ユーザー領域のデーモンは、カーネルで実行しているシステムクロックを更新します。システムクロックは、さまざまなクロックソースを使用して時間を維持します。一般的に使用されるのは Time Stamp Counter (TSC) です。TSC は、最後にリセットされた時点からのサイクル数を計測する CPU レジスターです。非常に高速でハイレゾリューションであり、中断も発生しません。

デーモンは、ntp と chrony の各パッケージにあるリポジトリーで利用できる **ntpd** デーモンと **chronyd** デーモンから選びます。

本章では、chrony スイートの使い方について説明します。

18.1. CHRONY スイートの概要

Chrony は Network Time Protocol (NTP) の実装です。Chrony を使用すると、以下のことができます。

- システムクロックを、**NTP** サーバーと同期する
- システムクロックを、GPS レシーバーなどの基準クロックと同期する
- システムクロックを、手動で入力した時間と同期する
- ネットワーク内の他のコンピューターにタイムサービスを提供する **NTPv4(RFC 5905)** サーバーまたはピアとして

chrony は、ネットワーク接続が頻繁に切断される、ネットワークの混雑が長時間続く、温度が変わる (一般的なコンピューターのクロックは温度に敏感) といったさまざまな状況や、継続して実行されない、または仮想マシンで実行されているといったシステムにおいても、良好に動作します。

インターネット上で同期している 2 つのマシン間の一般的精度は数ミリ秒以内、LAN 上のマシン間では数十マイクロ秒以内です。ハードウェアのタイムスタンプまたはハードウェア基準クロックは、同期している 2 つのマシン間の精度をサブマイクロ秒レベルにまで高めることができます。

chrony は、ユーザー領域で実行する **chronyd** と、**chronyd** のパフォーマンスを監視し、実行時にさまざまなオペレーティングパラメーターを変更するのに使用できるコマンドラインプログラムである chronyc で設定されます。

18.1.1. ntpd と chronyd の相違点

chronyd が **ntpd** よりも優れている点として、以下が挙げられます。

- **chronyd** は、時間参照へのアクセスが途切れやすい環境でも良好に機能します。一方 **ntpd** は、良好に機能するためには時間参照の定期的なポーリングを必要とします。
- **chronyd** はネットワークの混雑が長時間にわたる場合でも機能します。
- **chronyd** は通常、クロックをより高速に、より高い精度で同期できます。
- **chronyd** は、水晶振動子の温度変化などによってクロックのレートが突然変更しても素早く適応します。一方、**ntpd** の場合は、落ち着くまでに時間がかかる場合があります。
- デフォルト設定では、**chronyd** は実行中の他のプログラムが発生しないように、システムの起

動時にクロックを同期した後には時間をステップしません。**ntpd** は時間をステップできないように設定することも可能ですが、クロックを調整する別の方法を使用する必要があります。これは、クロックの精度にマイナスの影響を含む短所があります。

- **chronyd** は Linux 上のクロックのレートを幅広い範囲で調整できるため、たとえば仮想マシン上など、壊れたクロックもしくは不安定なクロックのあるマシン上でも操作が可能になります。たとえば、一部の仮想マシン。

- **chronyd** の方が小型でメモリー使用量が少なく、必要なときだけ CPU を起動させるので、省電力性に優れています。

chronyd では可能で、**ntpd** ではできない点は、以下のとおりです。

- **chronyd** は、時間補正の方法がクロックを監視している管理者などによる手動入力のみである隔離されたネットワークにサポートを提供します。たとえば、管理者がクロックを探します。**chronyd** は、異なる更新時に修正されたエラーを検証し、コンピューターの時刻が進んだり遅れたりする時点のレートを予測して、この予測を使用してその後のコンピュータークロックの調整を行います。

- **chronyd** は、たとえばコンピューターの電源を切った後も作動し続けるクロックのような、リアルタイムクロックの時間の増減率の計算をサポートしています。このデータがあれば、システムの起動時に、リアルタイムクロックから取得した適合された時間の値を使用してシステム時間を設定できます。これらのリアルタイムクロック機能が利用できるのは、現在 Linux のシステムのみです。

- **chronyd** は Linux のハードウェアタイムスタンプに対応しており、ローカルネットワークにおける非常に精度の高い同期を可能にしています。

ntpd では可能で **chronyd** ではできない点は、以下のとおりです。

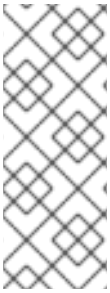
- **ntpd** は、ブロードキャスト、マルチキャスト、メニーキャストのクライアントやサーバーなど、**NTP** バージョン 4 (RFC 5905) のすべてのオペレーティングモードに対応しています。ただし、ブロードキャストとマルチキャストの各モードは、認証はされても、本来、一般的なサーバーやクライアントのモードと比べて精度と安全性が低いため、通常は回避することが推奨されます。

- **ntpd** は、公開鍵暗号でサーバーを認証する Autokey プロトコル (RFC 5906) に対応します。ただし、このプロトコルは安全性が低いことが実証されており、将来 Network Time Security (NTS) 仕様の実装に置き換えられる可能性があるため注意してください。

- **ntpd** には多くの基準クロックのドライバーが含まれています。一方の **chronyd** は、共有メモリー (SHM) または Unix ドメインソケット (SOCK) を使用して基準クロックのデータにアクセスするために、gpsd のような他のプログラムに依存しています。

18.1.2. NTP デーモンの選択

Chrony は、chrony に対応していないツールで管理または監視されているシステムや、chrony と互換性のないハードウェア基準クロックを搭載しているシステムを除いた、すべてのシステムに推奨されます。



注記

chronyd は Autokey プロトコルをサポートしていないため、このプロトコルを使用してパケット認証を行う必要があるシステムには ntpd しか使用できません。Autokey プロトコルには重大なセキュリティー上の問題があるため、使用を回避することが推奨されます。Autokey の代わりに、chronyd と ntpd の両方に対応している、対称鍵を使用してください。Chrony は SHA256 や SHA512 といったより強力なハッシュ関数に対応しています。一方、ntpd が使用できるのは MD5 と SHA1 のみです。

18.2. CHRONY の概要および設定

18.2.1. chronyd および chronyc について

chrony デーモンである chronyd は、コマンドラインユーティリティーの chronyc を使用して監視と管理を行います。このユーティリティーは、chronyd の現在の状態に対してクエリーを実行し、その設定を変更する多数のコマンド入力を可能にするコマンドプロンプトを提供します。デフォルトでは、chronyd は chronyc のローカルインスタンスのコマンドのみを受け付けますが、リモートホストから監視コマンドを受け付けるように設定することも可能です。リモートアクセスは制限する必要があります。

18.2.2. chrony 設定コマンドの概要

chronyd のデフォルト設定ファイルは /etc/chrony.conf です。-f オプションは、代替の設定ファイルパスを指定するのに使用できます。その他のオプションについては、 chronyd の man ページを参照してください。

以下は、chronyd 設定オプションの一部です。

コメント

コメントの前には、#、%、;、または ! を付けます。

allow

必要に応じて、NTP サーバーとして動作しているマシンへの NTP 接続が許可されるホスト、サブネット、またはネットワークを指定します。デフォルトでは、接続は許可されません。

例18.1 allow オプションを使用してアクセスを付与します。

- *このコマンドを使用して、IPv4 へのアクセスを付与します。*

  ```
  allow 192.0.2.0/24
  ```

- *このコマンドを使用して、IPv6 へのアクセスを付与します。*

  ```
  allow 2001:0db8:85a3::8a2e:0370:7334
  ```

注記

*クライアントアクセスを可能にするため、ファイアウォールで **UDP** ポート番号 **123** を開いておく必要があります。*

```
~]# firewall-cmd --zone=public --add-port=123/udp
```

*ポート 123 を永続的に開くには、**--permanent** オプションを使用します。*

```
~]# firewall-cmd --permanent --zone=public --add-port=123/udp
```

cmdallow

***allow** ディレクティブ (**allow** セクションを参照) と似ていますが、特定のサブセットまたはホストへの制御アクセスを許可します (これは **NTP** クライアントアクセスとは異なります)。制御アクセスとは、**chronyc** がこのようなホストで実行可能であり、このコンピューターで **chronyd** に正常に接続できることを意味します。 構文は同じになります。また、**cmdallow all** ディレクティブと動作が似ている **cmddeny all** もあります。*

dumpdir

***chronyd** を再起動しても測定履歴を保存するディレクトリーへのパスです (これが実行していない時にシステムクロックの動作が変更されていないことを想定しています)。(設定ファイルの **dumponexit** コマンド、chronyc の **dump** コマンドから) この機能を使用する場合は、**dumpdir** コマンドを使用して測定履歴を保存するディレクトリーを定義してください。*

dumponexit

*このコマンドが存在する場合は、プログラムの終了時に、**chronyd** が、常に記録された各時間ソースの測定履歴を保存することを示しています。(上記の **dumpdir** コマンドを参照してください)。*

hwtimestamp

***hwtimestamp** ディレクティブは、ハードウェアのタイムスタンプで非常に精度の高い同期を可能にします。詳細は **chrony.conf(5)** man ページを参照してください。*

local

***local** キーワードは、現在の同期ソースがない場合でも、(それをポーリングしているクライアントから見ると) **chronyd** がリアルタイムに同期しているように見えるようにします。このオプションは、通常、孤立したネットワークで master となるコンピューターで使用されます。ここでは、いくつかのコンピューターが相互に同期するようになり、master は、手動入力でリアルタイムと一致させます。*
以下はコマンド例を示します。

```
local stratum 10
```

*10 という大きな値が示しているのは基準クロックからのホップ数が非常に多いため、時間の信頼性が低いということです。基準クロックに最終的に同期している別のコンピューターにアクセスするコンピューターは、確実に 10 よりも下の階層に存在することになります。このため、**local** コマンドで 10 のように高い値を設定すると、リアルサーバーの視認性があるクライアントにリークしたとしても、マシン自体の時間がリアルタイムと混同することを防ぎます。*

log

***log** コマンドは、特定情報がログに記録されることを意味します。以下のオプションを取ります。*

measurements

*このオプションは、生の **NTP** 測定値とその関連情報を、**measurements.log** という名前のログファイルに記録します。*

statistics

*このオプションは、回帰処理の情報を **statistics.log** ファイルにログ記録します。*

tracking

*このオプションは、システムの時間が進んだり遅れたりする予測に対する変更や、発生した slew 調整を、**tracking.log** と呼ばれるファイルにログ記録します。*

rtc

このオプションは、システムのリアルタイムクロックに関する情報をログ記録します。

refclocks

*このオプションは、生およびフィルター処理された基準クロックの測定を、**refclocks.log** ファイルにログ記録します。*

tempcomp

*このオプションは、温度測定とシステムレートの補正を、**tempcomp.log** と呼ばれるファイルにログ記録します。*
*ログファイルは、**logdir** コマンドが指定するディレクトリーに書き込まれます。*

以下はコマンド例を示します。

```
log measurements statistics tracking
```

logdir

このディレクティブは、ログファイルが書き込まれるディレクトリーを指定します。
このディレクティブの使用例は、以下のようになります。

```
logdir /var/log/chrony
```

makestep

*通常、**chronyd** は、必要に応じてクロックの速度を下げるかまたは上げることで、システムに対して時間オフセットの段階的修正を実施します。特定の状況では、このスルーイング (slewing) プロセスでシステムクロックを修正するのに非常に時間がかかり、システムクロックが不安定な状態になることがあります。このディレクティブは、調整がしきい値を上回ったときに、**chronyd** でシステムクロックを一度に修正 (step) します。ただし、**chronyd** が開始してからのクロックの更新が、指定した制限を上回らなかった場合に限ります (負の値は、制限を無効にします)。**initstepslew** ディレクティブは **NTP** のソースのみに対応しているため、この方法は基準クロックを使用しているときに特に有用です。*
このディレクティブの使用例は、以下のようになります。

```
makestep 1000 10
```

この場合は、調整幅が 1000 秒よりも大きければシステムクロックが更新されますが、最初の 10 回の更新しか行われません。

maxchange

*このディレクティブは、クロック更新で修正される許容オフセットの最大値を設定します。更新が指定された回数行われた後にのみチェックが実行されるため、システムクロックの初回の調整が大きくなります。指定された最大値よりも大きなオフセットが発生すると、そのオフセットは指定した回数の間無視され、その後に **chronyd** が終了します (負の値を使用すると、終了しないようになります)。いずれの場合も、メッセージは syslog に送信されます。*
このディレクティブの使用例は、以下のようになります。

```
maxchange 1000 1 2
```

最初のクロック更新後に、**chronyd** がすべてのクロック更新でオフセットをチェックし、1000 秒よりも大きい調整を 2 回無視して、3 回目に終了します。

maxupdateskew

chronyd のタスクの 1 つは、コンピューターのクロックがその基準源に対してどれくらい速くまたは遅く動くかを調べることです。また、推定値の誤差範囲の概算を計算します。
エラーの範囲が広すぎる場合は、測定が落ち着いていないこと、そして推定増加または損失率に信頼性があまりないことを示しています。

maxupdateskew パラメーターは、推定値を使用できるほど信頼できるかどうかを判定するためのしきい値です。デフォルトでは、しきい値は 1000 ppm です。

構文は以下のようになります。

```
maxupdateskew skew-in-ppm
```

skew-in-ppm の一般的な値は、電話回線を介してサーバーにダイアルアップ接続を行う場合は 100、LAN 上のコンピューターの場合は 5 または 10 になります。

信頼性のない推定値の使用に対する保護の方法は、これだけではないことに注意してください。**chronyd** は常に、予想される進みまたは遅れのレートと、予測のエラー範囲を追跡しています。ソースの 1 つで測定が作成された後に別の予測が新たに生成されると、加重組み合わせのアルゴリズムを使用してマスター予測が更新されます。このため、**chronyd** に信頼性の高い既存のマスター予測があり、エラー範囲の広い新たな予測が生成されると、既存のマスター予測が新規のマスター予測を特徴づけることになります。

minsources

minsources ディレクティブは、ローカルクロックを更新する前にソース選択アルゴリズムで選択可能なものとして考慮されるべきソースの最小数を設定します。
構文は以下のようになります。

```
minsources number-of-sources
```

number-of-sources は、デフォルトでは 1 です。minsource を 1 よりも大きくすると、信頼性が向上します。複数のソースが互いに対応することが必要となるためです。

noclientlog

引数を取らないこのディレクティブは、クライアントアクセスをログに記録しないことを指定します。通常はログに記録されるため、chronyc のクライアントコマンドを使用して統計が報告されます。

reselectdist

chronyd が、利用可能なソースから同期ソースを選択する際に、同期距離が最短のものを優先します。ただし、同様の距離のソースが複数存在して、再選択が繰り返されるのを回避するため、現在選択されていないソースからの距離に、固定距離が追加されます。これは、**reselectdist** オプションで設定できます。デフォルトでは、この距離は 100 ミリ秒となります。
構文は以下のようになります。

```
reselectdist dist-in-seconds
```

stratumweight

stratumweight *ディレクティブは、****chronyd*** *が利用可能なソースから同期ソースを選択する際に、階層ごとに追加される同期距離を設定します。*
構文は以下のようになります。

```
stratumweight dist-in-seconds
```

*デフォルトでは、*dist-in-seconds *は1ミリ秒です。つまり、通常は、距離の値が非常に悪い場合でも、stratum が高いソースよりも、stratum が低いソースが優先されることを意味します。****stratumweight*** *を0に設定すると、****chronyd*** *は、ソースを選択するときに* stratum *を無視します。*

rtcfile

rtcfile *ディレクティブは、システムのリアルタイムクロック(RTC)の正確性の追跡に関連するパラメーターを****chronyd*** *が保存できるファイル名を定義します。*
構文は以下のようになります。

```
rtcfile /var/lib/chrony/rtc
```

chronyd *は、このファイルが存在し、*chronyc *で****writertc*** *コマンドを実行すると、情報をこのファイルに保存します。保存される情報は、あるエポックでの****RTC*** *のエラー、そのエポック(1970年1月1日以降の秒数)、および****RTC*** *の時間が早まるもしくは遅れる変化量です。リアルタイムクロックのコードはシステム固有のものなので、すべてのリアルタイムクロックがサポートされるわけではありません。このディレクティブを使用する場合は、リアルタイムクロックを手動で調整しないでください。リアルタイムクロックがランダムな間隔で調整されると、その変化量を測定する* chrony *の必要性が干渉されるためです。*

rtcsync

rtcsync *ディレクティブは、デフォルトで****/etc/chrony.conf*** *ファイルにあります。これにより、システムクロックが同期されていることがカーネルに知らされ、カーネルによりリアルタイムクロックが11分ごとに更新されます。*

18.2.3. chronyc のセキュリティー

*Chronyc は、以下の2つの方法で****chronyd*** *にアクセスします。*

- *インターネットプロトコル(****IPv4*** *または* IPv6*)*
- *Unix ドメインソケット(****root*** *または* chrony *ユーザーがローカルにアクセス可能)*

*デフォルトでは、*chronyc *は、Unix ドメインソケットに接続します。デフォルトのパスは****/var/run/chrony/chronyd.sock*** *です。この接続に失敗すると(たとえば非特権ユーザーで*chronyc *を実行していると失敗する可能性があります)、*chronyc *は127.0.0.1への接続を試み、その後 ::1 への接続を試みます。*

chronyd *の動作に影響しない次の監視コマンドのみが、ネットワークに許可されています。*

- *activity*
- *manual list*
- *rtcdata*
- *smoothing*

- *sources*
- *sourcestats*
- *tracking*
- *waitsync*

***chronyd** がこのコマンドを受け取るホスト郡は、**chronyd** の設定ファイルにある **cmdallow** ディレクティブ、または chronyc の **cmdallow** コマンドで設定できます。デフォルトでは、このコマンドが許可されるのは、ローカルホスト (127.0.0.1 または ::1) のものだけになります。*

*その他のコマンドはすべて、Unix ドメインソケットのみを介して許可されます。ネットワーク上で送信されると、たとえローカルホストであっても、**chronyd** は **Not authorised** エラーを返します。*

chronyc を使用して chronyd にリモートでアクセス

1. *以下を **/etc/chrony.conf** ファイルに追加すると、IPv4 と IPv6 の両方のアドレスからアクセスが可能になります。*

   ```
   bindcmdaddress 0.0.0.0
   ```

   または

   ```
   bindcmdaddress :
   ```

2. ***cmdallow** ディレクティブを使用すると、リモート IP アドレス、ネットワーク、またはサブネットからのコマンドが許可されます。*
   ***/etc/chrony.conf** ファイルに以下の内容を追加します。*

   ```
   cmdallow 192.168.1.0/24
   ```

3. *ファイアウォールでポート 323 を開き、リモートシステムから接続します。*

   ```
   ~]# firewall-cmd --zone=public --add-port=323/udp
   ```

   *ポート 323 を永続的に開く場合は、**--permanent** を使用します。*

   ```
   ~]# firewall-cmd --permanent --zone=public --add-port=323/udp
   ```

***allow** ディレクティブは **NTP** のアクセス用で、**cmdallow** ディレクティブは、リモートコマンドの受け取りを可能にします。ローカルで実行している chronyc を使用すると、この変更を一時的なものにできます。永続的に変更するときは設定ファイルを変更します。*

18.3. CHRONY の使用

18.3.1. chrony のインストール

*chrony スイートは一部のバージョンの Red Hat Enterprise Linux 7 でデフォルトでインストールされます。必要な場合には、インストールされていることを確認するために **root** で以下のコマンドを実行します。*

```
~]# yum install chrony
```

chrony デーモンのデフォルトの場所は、**/usr/sbin/chronyd** です。このコマンドラインユーティリティーは **/usr/bin/chronyc** にインストールされます。

18.3.2. chronyd ステータスの確認

chronyd 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。

```
~]$ systemctl status chronyd
chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled)
  Active: active (running) since Wed 2013-06-12 22:23:16 CEST; 11h ago
```

18.3.3. chronyd の起動

chronyd を開始するには、**root** で以下のコマンドを実行します。

```
~]# systemctl start chronyd
```

システムの起動時に **chronyd** を自動的に起動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。

```
~]# systemctl enable chronyd
```

18.3.4. chronyd の停止

chronyd を停止するには、**root** で以下のコマンドを実行します。

```
~]# systemctl stop chronyd
```

システムの起動時に **chronyd** を自動的に起動しないように設定するには、**root** で以下のコマンドを実行します。

```
~]# systemctl disable chronyd
```

18.3.5. chrony の同期確認

chrony が同期されているかどうかを確認するには、**tracking** コマンド、**sources** コマンド、および **sourcestats** コマンドを使用します。

18.3.5.1. chrony 追跡の確認

chrony の追跡を確認するには、以下のコマンドを実行します。

```
~]$ chronyc tracking
Reference ID  : CB00710F (foo.example.net)
Stratum     : 3
Ref time (UTC) : Fri Jan 27 09:49:17 2017
System time   : 0.000006523 seconds slow of NTP time
Last offset   : -0.000006747 seconds
```

```
RMS offset   : 0.000035822 seconds
Frequency    : 3.225 ppm slow
Residual freq  : 0.000 ppm
Skew      : 0.129 ppm
Root delay   : 0.013639022 seconds
Root dispersion : 0.001100737 seconds
Update interval : 64.2 seconds
Leap status   : Normal
```

各フィールドは、以下のとおりです。

Reference ID

(利用可能な場合) コンピューターが現在同期しているサーバーの参照 ID および参照名 (または IP アドレス) です。参照 ID は IPv4 アドレスとの混同を避けるため 16 進数の数値になっています。

Stratum

基準クロックのあるコンピューターから何ホップ離れているかを示します。上記の例のコンピューターは stratum-1 コンピューターであるため、2 ホップ離れていることになります (つまり、a.b.c が stratum-2 で、stratum-1 から同期しています)。

Ref time

参照ソースからの最後の測定が処理された時間 (UTC) です。

System time

chronyd は、通常の操作ではシステムクロックを更新しません。タイムスケールにおけるジャンプは、いかなるものでも特定のアプリケーションプログラムに有害な結果をもたらすためです。代わりに、システムクロックのエラーをわずかに早めたり遅くしたりして、エラーがなくなるまで修正し、修正が完了したら、システムクロックを通常のスピードに戻します。その結果、(**gettimeofday()** システムコールを使用した他のプログラム、またはシェルの **date** コマンドが読み取る) システムクロックが、**chronyd** が予測する現在の実際の時間 (サーバーモードで稼働している場合はこれを **NTP** クライアントに報告) と異なる期間が発生します。この行で報告される値は、これによる差異です。

Last offset

最後のクロック更新におけるローカルオフセットの予測です。

RMS offset

長期的な、オフセット値の平均です。

Frequency

chronyd が修正しない場合にシステムクロックが間違う変化量です。これは、ppm (100 万分の 1) で表されます。たとえば、1 ppm という値は、システムクロックにおける 1 秒が、実際の時間と比較すると 1.000001 秒進んでいることを意味します。

Residual freq

現在選択されている基準源の残留周波数を示しています。基準源からの測定値が、示すべき周波数と、実際に使用されている周波数との違いを反映しています。
これが常にゼロにならない理由は、補正する手順が周波数に適用されているためです。基準源から測定を取得し、新たな剰余周波数が計算されるたびに、この剰余の推定精度が、既存の周波数の推定精度 (**skew** を参照) と比較されます。新たな周波数の加重平均は、その精度によって異なる加重で計算されます。基準源からの測定に一貫した傾向がある場合、剰余は時間をかけてゼロになります。

Skew

周波数の予測されるエラー範囲です。

Root delay

コンピューターが最終的に同期する stratum-1 コンピューターの、ネットワークパスの遅延の合計数です。Root delay の値はナノ秒の分解能で出力されます。値は、極端な状況では負数になります。(コンピューター同士が互いの周波数を追跡せず、各コンピューターのターンアラウンド時間に比較してネットワークの遅延が非常に短い、対称的なピア配置で、これが発生する場合があります。)

Root dispersion

コンピューターが最後に同期する stratum-1 コンピューターに戻るすべてのコンピューターを介して累積された合計分散です。分散は、システムクロックの分解能や統計的測定の変動等に起因します。Root の分散値は、ナノ秒の分解能で出力されます。

Leap status

Leap のステータスで、Normal、Insert second、Delete second、または Not synchronized のいずれかになります。

18.3.5.2. chrony ソースの確認

sources コマンドは、**chronyd** がアクセスしている現在の時間ソースの情報を表示します。

任意の引数 -v (verbose (詳細) の意) を指定できます。この例では、余分なキャプション行は、コラムの意味を説明するものとして表示されます。

```
~]$ chronyc sources
 210 Number of sources = 3
MS Name/IP address     Stratum Poll Reach LastRx Last sample
===============================================================================

#* GPS0           0  4  377  11  -479ns[ -621ns] +/- 134ns
^? a.b.c         2  6  377  23  -923us[ -924us] +/-  43ms
^+ d.e.f         1  6  377  21 -2629us[-2619us] +/-  86ms
```

各コラムの表示内容は、以下のとおりです。

M

ソースのモードを示します。^ はサーバーを意味し、= はピアを意味し、# はローカルに接続された基準クロックを意味します。

S

この列は、ソースの状態を示します。* は、**chronyd** が現在同期しているソースを表します。+ は、選択したソースと結合する、受け入れ可能なソースを表します。- は、受け入れ可能なソースで、結合アルゴリズムにより除外されたものを表します。? は、接続が切断されたソース、またはパケットがすべてのテストをパスしないソースを表します。x は、**chronyd** が falseticker と考える (つまり、その時間が他の大半のソースと一致しない) クロックを表します。~ は、時間の変動性が大きすぎるように見えるソースを表します。? 条件は、少なくとも 3 つのサンプルが収集されるまで開始時にも表示されます。

Name/IP address

ソースの名前または IP アドレス、もしくは基準クロックの参照 ID を表示します。

Stratum

直近で受け取ったサンプルでレポートされているソースの stratum を表示します。Stratum 1 は、ローカルで基準クロックに接続しているコンピューターを示します。Stratum 1 コンピューターに同期しているコンピューターは、stratum 2 に存在することになります。同じく、Stratum 2 コンピューターに同期しているコンピューターは stratum 3 に存在することになり、以後も同様に続きます。

Poll

ソースがポーリングされるレートで、間隔のベース-2 対数を秒数で示します。つまり、値が 6 の場合は、64 秒ごとに測定が行われます。
chronyd は、一般的な条件に応じて、ポーリングレートを自動的に変更します。

Reach

ソースの到達可能性のレジスターで、8 進法で表示されます。レジスターは 8 ビットで、ソースからパケットを受信するたびに、またはミスするたびに更新されます。値が 377 の場合は、最近の 8 回の通信全体で、有効な返信を受け取ったことを表します。

LastRx

このコラムは、ソースから最後のサンプルがいつ受信されたかを表示します。通常は、秒数で表示されます。**m**、**h**、**d**、および **y** の各文字は、それぞれ分、時間、日、年を表します。値が 10 年の場合は、このソースからまだサンプルを受信していないことを示します。

Last sample

この列は、ローカルクロックと、最後に測定されたソースの間のオフセットを表示します。角括弧内の数字は、実際に測定されたオフセットを表示します。これには **ns** (ナノ秒)、**us** (マイクロ秒)、**ms** (ミリ秒)、または **s** (秒) の各接尾辞が付く場合があります。角括弧の左側は元の測定を示し、slew がそれ以降にローカルクロックに適用可能になるように調整されています。**+/-** に続く数字は、測定におけるエラーのマージンを示します。オフセットの値がプラスの場合は、ローカルクロックがソースよりも進んでいることを意味します。

18.3.5.3. chrony ソースの統計情報の確認

sourcestats コマンドは、**chronyd** が現在調べている各ソースに関するドリフト量とオフセット推定プロセスの情報を表示します。

任意の引数 **-v** (verbose (詳細) の意) を指定できます。この例では、余分なキャプション行は、コラムの意味を説明するものとして表示されます。

```
~]$ chronyc sourcestats
210 Number of sources = 1
Name/IP Address      NP NR Span Frequency Freq Skew Offset Std Dev
==============================================================================

abc.def.ghi        11  5  46m   -0.001   0.045   1us  25us
```

各コラムの表示内容は、以下のとおりです。

Name/IP address

これは、**NTP** サーバー (またはピア) の名前または **IP** アドレス、またはその行の残りの部分が関連する基準クロックの参照 **ID** です。

NP

現在サーバーで保持されているサンプルポイントの数です。誤差レートと現在のオフセットは、このポイントを使用して線形回帰を実行することで予測されます。

NR

最新の回帰を追跡している同一サインを持つ剰余の実行数です。この数字がサンプル数に対して少なくなりすぎる場合は、直線がデータに適合しなくなったことを意味します。実行数が少なすぎる場合、**chronyd** は古いサンプルを破棄し、実行数が受け入れ可能な値になるまで回帰を再実行します。

Span

一番古いサンプルと最新のサンプルの間隔です。単位が表示されない場合は、秒数を表しています。この例の間隔は 46 分です。

Frequency

これは予測されるサーバーの剰余周波数で、ppm (100 万分の 1) で表されます。この場合、このコンピューターのクロックは、このサーバーに対比して 10^9 分の 1 遅く稼働していると見積もられています。

Freq Skew

Freq の予測されるエラー範囲です (ppm (100 万分の 1) で表されます) 。

Offset

ソースの予測されるオフセットです。

Std Dev

サンプルの予測される標準偏差です。

18.3.6. システムクロックの手動調整

*システムクロックを徐々に調整していく (slew) のを止め、一度に修正 (step) するには、**root** で以下のコマンドを実行します。*

```
~]# chronyc makestep
```

***rtcfile** ディレクティブを使用している場合は、リアルタイムクロックを手動で調整しないでください。ランダムな調整を行うと、リアルタイムクロックがずれる変化量を測定する必要がある **chrony** に影響を与えます。*

18.4. 異なる環境での CHRONY の設定

18.4.1. 孤立したネットワークでのシステムにおける chrony の設定

*インターネットに接続していないネットワークに関しては、1 台のコンピューターをマスタータイムサーバーとします。もう 1 台のコンピューターをマスターの直接クライアント、またはクライアントのクライアントとします。マスターでは、ドリフトファイルは、システムクロックのドリフトの平均率を使用して手動で設定します。マスターを再起動すると、周囲のシステムから時間を取得し、システムクロックを設定するために平均値を計算します。その後、**drift** ファイルに基づいて調整の適用を再開します。**drift** ファイルは、**settime** コマンドが使用されたときに自動的に更新されます。*

*マスターに選ばれたシステムで、テキストエディターを使用して、**root** で **/etc/chrony.conf** を以下のように編集します。*

```
driftfile /var/lib/chrony/drift
commandkey 1
keyfile /etc/chrony.keys
initstepslew 10 client1 client3 client6
local stratum 8
manual
allow 192.0.2.0
```

***192.0.2.0** は、クライアントが接続できるネットワークアドレスまたはサブネットアドレスです。*

*マスターのダイレクトクライアントにしたシステムで、**root** でテキストエディターを実行し、以下のように **/etc/chrony.conf** を実行します。*

```
server master
driftfile /var/lib/chrony/drift
```

```
logdir /var/log/chrony
log measurements statistics tracking
keyfile /etc/chrony.keys
commandkey 24
local stratum 10
initstepslew 20 master
allow 192.0.2.123
```

ここでの **192.0.2.123** はマスターのアドレスで、**master** はマスターのホスト名になります。この設定になっているクライアントは、システムが再起動するとマスターと再同期を行います。

マスターの直接のクライアントにはならないクライアントシステムの **/etc/chrony.conf** ファイルでは、**local** ディレクティブおよび **allow** ディレクティブが省略される以外は、同じになるべきです。

孤立したネットワークでは、ローカルの参照モードを有効にする **local** ディレクティブも使用できます。これにより、NTP サーバーとして動作している **chronyd** は、サーバーが一度も同期されていなかったり、クロックの最終更新から長い時間が経過している場合でも、リアルタイムに同期してるように見えます。

複数のサーバーをポーリングしているクライアントを混同することなく、ネットワーク上の複数のサーバーに同じローカル設定を使用し、互いを同期させるには、Orphan モードを有効にする **local** ディレクティブの **orphan** オプションを使用します。各サーバーは、他のすべてのサーバーを **local** でポーリングするように設定する必要があります。これにより、最小の参照 ID を持つサーバーでのみローカル参照が有効になり、他のサーバーはそれに同期します。サーバーが失敗すると別のサーバーが引き継ぎます。

18.5. CHRONYC の使用

18.5.1. chronyc を使用した chronyd の制御

対話モードでコマンドラインユーティリティー **chronyc** を使用して、**chronyd** のローカルインスタンスを変更するには、**root** で以下のコマンドを実行します。

```
~]# chronyc
```

制限されているコマンドを使用する場合は、**root** で **chronyc** を実行する必要があります。

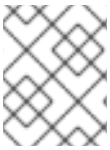
以下のように、**chronyc** コマンドプロンプトが表示されます。

```
chronyc>
```

このコマンドのリストを表示するには、**help** と入力します。

このユーティリティーは、以下のようなコマンドで呼び出すと、非対話型コマンドモードで呼び出すこともできます。

```
chronyc command
```



注記

chronyc を使用して変更した内容は永続的ではなく、**chronyd** を再起動すると元に戻ります。永続的に変更する場合は、**/etc/chrony.conf** を変更してください。

18.6. ハードウェアのタイムスタンプを使用した CHRONY

18.6.1. ハードウェアのタイムスタンプの概要

ハードウェアのタイムスタンプは、一部の Network Interface Controller (NIC) でサポートされている機能です。着信パケットおよび送信パケットのタイムスタンプを正確に提供します。**NTP** タイムスタンプは通常、カーネルにより作成され、システムクロックを使用して chronyd が作成されます。ただし、ハードウェアのタイムスタンプが有効な場合、NIC は独自のクロックを使用して、パケットがリンク層または物理層に出入りするときにタイムスタンプを生成します。ハードウェアスタンプで **NTP** を使用すると、同期の精度を大幅に向上できます。最高精度を実現するには、**NTP** サーバーおよび**NTP** クライアントの両方が、ハードウェアのタイムスタンプを使用している必要があります。理想的な条件下では、サブマイクロ秒単位の精度を実現できるかもしれません。

ハードウェアのタイムスタンプを使用する時間同期の別のプロトコルには、**PTP** があります。**PTP** に関する詳細は、20章ptp4l を使用した PTP の設定を参照してください。**NTP** とは異なり、**PTP** は、ネットワークスイッチおよびルーターの補助に依存しています。同期の精度を最高の状態にしたい場合は、**PTP** をサポートしているスイッチやルーターがあるネットワークで**PTP** を使用し、そのようなスイッチおよびルーターがないネットワークでは、**NTP** を使用することが推奨されます。

18.6.2. ハードウェアタイムスタンプのサポートの確認

NTP を使用したハードウェアのタイムスタンプがインターフェイスでサポートされていることを確認するには、**ethtool -T** コマンドを実行します。**ethtool** が、**SOF_TIMESTAMPING_TX_HARDWARE** 機能、**SOF_TIMESTAMPING_TX_SOFTWARE** 機能、および**HWTSTAMP_FILTER_ALL** フィルターモードをリスト表示する場合は、**NTP** を使用して、ハードウェアのタイムスタンプにインターフェイスを使用できます。

例18.2 特定のインターフェイスにおけるハードウェアのタイムスタンプのサポートの確認

```
~]# ethtool -T eth0
```

出力:

```
Timestamping parameters for eth0:
Capabilities:
    hardware-transmit   (SOF_TIMESTAMPING_TX_HARDWARE)
    software-transmit   (SOF_TIMESTAMPING_TX_SOFTWARE)
    hardware-receive   (SOF_TIMESTAMPING_RX_HARDWARE)
    software-receive   (SOF_TIMESTAMPING_RX_SOFTWARE)
    software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
    hardware-raw-clock  (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 0
Hardware Transmit Timestamp Modes:
    off         (HWTSTAMP_TX_OFF)
    on          (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
    none        (HWTSTAMP_FILTER_NONE)
    all         (HWTSTAMP_FILTER_ALL)
    ptpv1-l4-sync     (HWTSTAMP_FILTER_PTP_V1_L4_SYNC)
    ptpv1-l4-delay-req  (HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ)
    ptpv2-l4-sync     (HWTSTAMP_FILTER_PTP_V2_L4_SYNC)
    ptpv2-l4-delay-req  (HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ)
    ptpv2-l2-sync     (HWTSTAMP_FILTER_PTP_V2_L2_SYNC)
    ptpv2-l2-delay-req  (HWTSTAMP_FILTER_PTP_V2_L2_DELAY_REQ)
```

```
ptpv2-event     (HWTSTAMP_FILTER_PTP_V2_EVENT)
ptpv2-sync      (HWTSTAMP_FILTER_PTP_V2_SYNC)
ptpv2-delay-req    (HWTSTAMP_FILTER_PTP_V2_DELAY_REQ)
```

18.6.3. ハードウェアのタイムスタンプの有効化

ハードウェアのタイムスタンプを有効にするには、**/etc/chrony.conf** ファイルの **hwtimestamp** ディレクティブを使用します。ディレクティブは、個別のインターフェイスを指定できますが、ワイルドカード文字 **()** を使用して、ハードウェアのタイムスタンプをサポートするすべてのインターフェイスでハードウェアのタイムスタンプを有効にすることもできます。**linuxptp** パッケージの **ptp4l** などのアプリケーションが、ハードウェアのタイムスタンプを使用していない場合は、ワイルドカード仕様を使用してください。**chrony** 設定ファイルで、複数の **hwtimestamp** ディレクティブが使用されます。

例18.3 hwtimestamp のディレクティブを使用したハードウェアのタイムスタンプの有効化

```
hwtimestamp eth0
hwtimestamp eth1
hwtimestamp *
```

18.6.4. クライアントポーリング間隔の設定

インターネット上のサーバーのポーリング間隔は、デフォルトの範囲である 64 秒から 1024 秒が推奨されています。ローカルサーバーおよびハードウェアのタイムスタンプでは、システムクロックのオフセットを最小限にとどめるため、ポーリング間隔は短く設定する必要があります。

/etc/chrony.conf における以下のディレクティブは、1 秒のポーリング間隔を使用してローカルの **NTP** サーバーを指定します。

```
server ntp.local minpoll 0 maxpoll 0
```

18.6.5. インターリーブモードの有効化

ハードウェアの **NTP** アプライアンスではなく、**chrony** など、ソフトウェアの **NTP** 実装を実行する汎用コンピューターの **NTP** サーバーは、パケット送信後にのみハードウェア送信タイムスタンプを取得します。この動作により、サーバーは、対応するパケットのタイムスタンプを保存できません。**NTP** クライアントが、送信後に生成された送信タイムスタンプを受け取るようにするには、**/etc/chrony.conf** のサーバーディレクティブに **xleave** オプションを追加し、クライアントが **NTP** インターリーブモードを使用するように設定します。

```
server ntp.local minpoll 0 maxpoll 0 xleave
```

18.6.6. 多数のクライアント向けのサーバーの設定

デフォルトのサーバー設定では、最多で数千のクライアントが同時にインターリーブモードを使用できます。さらに多くのクライアント向けにサーバーを設定するには、**/etc/chrony.conf** の **clientloglimit** ディレクティブを増やします。このディレクティブは、サーバーでクライアントのアクセスログに割り当てられるメモリーの最大サイズを指定します。

```
clientloglimit 100000000
```

18.6.7. ハードウェアのタイムスタンプの確認

インターフェイスがハードウェアのタイムスタンプを有効にできたことを確認するには、システムログを確認してください。ログには、**chronyd** からの各インターフェイス向けメッセージに、有効にしたハードウェアのタイムスタンプが追記されているはずです。

例18.4 ハードウェアのタイムスタンプが有効になったインターフェイスのログメッセージ

```
chronyd[4081]: Enabled HW timestamping on eth0
chronyd[4081]: Enabled HW timestamping on eth1
```

chronyd が、**NTP** クライアントまたはピアとして設定されている場合は、**chronyc ntpdata** コマンドにより、送信先と受信先のタイムスタンプおよびインターリーブモードを、各 **NTP** ソースに報告できます。

例18.5 各 NTP ソースの送信先および受信先のタイムスタンプおよびインターリーブモードの報告

```
~]# chronyc ntpdata
```

出力:

```
Remote address : 203.0.113.15 (CB00710F)
Remote port   : 123
Local address  : 203.0.113.74 (CB00714A)
Leap status   : Normal
Version     : 4
Mode      : Server
Stratum     : 1
Poll interval  : 0 (1 seconds)
Precision    : -24 (0.000000060 seconds)
Root delay   : 0.000015 seconds
Root dispersion : 0.000015 seconds
Reference ID  : 47505300 (GPS)
Reference time : Wed May 03 13:47:45 2017
Offset     : -0.000000134 seconds
Peer delay   : 0.000005396 seconds
Peer dispersion : 0.000002329 seconds
Response time  : 0.000152073 seconds
Jitter asymmetry: +0.00
NTP tests    : 111 111 1111
Interleaved   : Yes
Authenticated  : No
TX timestamping : Hardware
RX timestamping : Hardware
Total TX    : 27
Total RX    : 27
Total valid RX : 27
```

例18.6 NTP 測定の安定性の報告

```
# chronyc sourcestats
```

> ハードウェアのタイムスタンプを有効にすると、**NTP** 測定の安定性は、通常のロードにおいて数十ナノ秒または数百ナノ秒となります。この安定性は、**chronyc sourcestats** コマンドの出力の **Std Dev** 列に報告されます。
>
> 出力:
>
> ```
> 210 Number of sources = 1
> Name/IP Address      NP NR Span Frequency Freq Skew Offset Std Dev
> ntp.local        12  7  11   +0.000   0.019   +0ns  49ns
> ```

18.6.8. PTP-NTP ブリッジの設定

非常に精度が高い **PTP** (Precision Time Protocol) のグランドマスターが、**PTP** サポートのあるスイッチまたはルーターを持たないネットワークで利用可能な場合、コンピューターは、**PTP** スレーブおよび stratum-1 の **NTP** サーバーとしての操作に専念する可能性があります。このようなコンピューターには、2 つ以上のネットワークインターフェイスが必要なほか、グランドマスターに近いか、グランドマスターに直接接続されている必要があります。これにより、ネットワークで非常に精度の高い同期が確実に実行されます。

1 つのインターフェイスを使用して、**PTP** でシステムクロックを同期するように、linuxptp パッケージの ptp4l および phc2sys プログラムを設定します。この設定は、20章ptp4l を使用した PTP の設定 で説明しています。chronyd を設定して、その他のインターフェイスを使用してシステム時間を提供するには、以下を行います。

> 例18.7 その他のインターフェイスを使用してシステム時間を提供するように chronyd を設定
>
> ```
> bindaddress 203.0.113.74
> hwtimestamp eth1
> local stratum 1
> ```

18.7. 関連情報

以下の資料は、chrony に関するその他の情報を提供します。

18.7.1. インストールされているドキュメント

- **chronyc(1)** の man ページ -chronyc コマンドラインインターフェイスと、そのコマンドおよびコマンドオプションを説明します。
- **chronyd(8)** の man ページ:chronyd デーモンと、そのコマンドおよびコマンドオプションが説明されています。
- **chrony.conf(5)** の man ページ:chrony 設定ファイルが説明されています。

18.7.2. オンラインドキュメント

- http://chrony.tuxfamily.org/doc/3.1/chronyc.html
- http://chrony.tuxfamily.org/doc/3.1/chronyd.html
- http://chrony.tuxfamily.org/doc/3.1/chrony.conf.html

FAQ への回答は http://chrony.tuxfamily.org/faq.html を参照してください。

第19章 NTPD を使用した NTP 設定

19.1. NTP の概要

Network Time Protocol (NTP) は正確な日時情報を広く行き渡らせ、ネットワークまたはインターネットで共通の参照先に同期しているネットワーク化されたコンピューターシステム上のタイムクロックを維持します。世界中の多くの標準機関には原子時計があり、これが参照先として利用可能になっている場合があります。Global Position システムを設定するサテライトには、複数の atomic クロックが含まれるため、時間のシグナルが非常に正確になる可能性があります。この信号は、軍事的な理由で意図的に弱められる場合があります。理想的な状況では、各サイトに独自の参照時計があるサーバーがあり、これがサイト全体のタイムサーバーとして機能します。低周波数のラジオ伝送またはグローバル位置システム (GPS) を介して、日時と日付を取得するデバイスが多数存在します。ただし多くの場合、インターネットに接続され、各地に分散する公開されたアクセス可能なタイムサーバーを使うことができます。これらの **NTP** サーバーは、**協定世界時** (UTC) を提供します。これらのタイムサーバーに関する情報は、www.pool.ntp.org で確認できます。

IT では、多くの理由で正確な時間の維持が重要です。たとえばネットワーキングでは、パケットとログのタイムスタンプが正確であることが必要になります。ログはサービスとセキュリティーの問題を調査するために使われるので、異なるシステム上のタイムスタンプは、同期されたクロックで記録される必要があります。システムおよびネットワークがますます高速化するにつれ、これに対応してクロックの正確性と精度の必要性も高まっています。国によっては、正確な同期クロックを保持することが法律で定められているところもあります。詳細は、www.ntp.org を参照してください。Linux システムでは、**NTP** はユーザースペースで実行しているデーモンにより実装されます。Red Hat Enterprise Linux 7 のデフォルトの **NTP** ユーザー空間デーモンは **chronyd** です。**ntpd** デーモンを使用する場合は、無効にする必要があります。chrony の情報については、*18章chrony スイートを使用した NTP 設定* を参照してください。

ユーザースペースのデーモンは、カーネルで実行しているソフトウェアクロックであるシステムクロックを更新します。Linux は、Real Time Clock(RTC) と呼ばれる一般的な埋め込みハードウェアクロックよりも、より優れた解像度を実現するために、ソフトウェアクロックをシステムクロックとして使用します。ハードウェアクロックの情報は、**rtc(4)** および **hwclock(8)** man ページを参照してください。システムクロックは、さまざまなクロックソースを使用して時間を維持します。一般的に使用されるのは Time Stamp Counter (TSC) です。TSC は、最後にリセットされた時点からのサイクル数を計測する CPU レジスターです。TSC は、それが最後にリセットされてからのサイクル数をカウントする CPU レジスターです。これは非常に高速で精度が高く、割り込みがありません。RTC が保持する時間は、気温の変化により、1 カ月あたり最大 5 分ごとに実際の時間から離れます。RTC が維持している時間は実際の時間と比べて、温度変化によりひと月で最大 5 分間の誤差を生じます。**ntpd** がシステムクロックを同期している場合には、カーネルは自動的に RTC を 11 分ごとに更新します。

19.2. NTP STRATA (階層)

NTP サーバーは、時間信号のソースとなる原子時計からの同期距離によって分類されます。サーバーは、上は 1 から下は 15 までの stratum (階層) に分類されていると考えられます。このため、特定の層に言及する際は、stratum という言葉が使用されます。アトミッククロックはソースである Stratum 0 と呼ばれます。ただし、インターネットでは Stratum 0 パケットが送信されず、すべての stratum 0 アトミッククロックは stratum 1 と呼ばれるサーバーにアタッチされます。これらのサーバーは、Stratum 1 としてマークされたパケットを送信します。stratum **n** とマークされたパケットで同期されるサーバーは、次の下位の stratum に属し、そのパケットを stratum **n+1** とマークします。stratum n のマークがついているパケットで同期されるサーバーは、その次に下位の stratum に所属し、パケットを stratum n+1 とマークします。Stratum 16 という名称は、サーバーが現在信頼できるタイムソースと同期していないことを意味します。

デフォルトでは、**NTP** クライアントはそれよりも下位の stratum にあるシステムのサーバーとして機能することに注意してください。

以下は NTP Strata (階層) のまとめになります。

Stratum 0
: 原子時計と無線および GPS による信号送信

  - GPS (全地球測位システム)
  - 携帯電話システム
  - 低周波無線送信 WWVB (米国コロラド州)、JJY-40 および JJY-60 (日本)、DCF77 (ドイツ)、および MSF (英国)
    これらの信号は専用デバイスで受信可能で、通常は RS-232 で組織全体またはサイト全体のタイムサーバーとして使用されるシステムに接続されます。

Stratum 1
: 電波時計、GPS 時計、または原子時計に接続しているコンピューター

Stratum 2
: stratum 1 から読み取り、下位の strata に提供

Stratum 3
: stratum 2 から読み取り、下位の strata に提供

Stratum n+1
: stratum n から読み取り、下位の strata に提供

Stratum 15
: stratum 14 から読み取り、これが最下位の stratum になります。

このプロセスは有効な最下位の Stratum 15 まで続きます。Stratum 16 というラベルは、非同期状態を示します。

19.3. NTP の概要

Red Hat Enterprise Linux で使用する NTP のバージョンは、RFC 1305 Network Time Protocol (Version 3) Specification, Implementation and Analysis および RFC 5905 Network Time Protocol Version 4: Protocol and Algorithms Specification で説明されています。

NTP を実装すると、10 秒以下の正確性が達成できます。インターネット上では、数十ミリ秒の正確性の維持は普通のことです。ローカルエリアネットワーク (LAN) 上では、1 ミリ秒の正確性は理想的な条件下では可能です。これは、クロックのドリフトを考慮して修正され、以前の、シンプルな時間プロトコルシステムで行われていないためです。64 ビットのタイムスタンプを使用することにより、233 ピコ秒の解像度が得られます。ここではタイムスタンプの最初の 32 ビットが秒に使われ、次の 32 ビットが 1 秒未満に使われます。

NTP が示すのは、1900 年 1 月 1 日の GMT 午前 0 時 00 分からの積算秒数です。秒数のカウントには 32 ビットが使われているので、時間は 2036 年にロールオーバーしてしまいます。しかし、NTP はタイムスタンプ間の差異で機能するため、タイムプロトコルの他の実装ほどの問題はもたらされません。誤差が 68 年以内のハードウェアクロックが起動時に利用可能であれば、NTP は正確に現在の日時を解釈します。NTP4 仕様は、Era Number と Era Offset を提供します。これを使用することで、68 年間の時間の長さを扱う場合にソフトウェアをより堅牢にすることができます。この問題は Unix の 2038 年問題と混同しないようにしてください。

NTP プロトコルは、正確性を高めるために追加情報を提供します。4 つのタイムスタンプを使うことで、往復時間とサーバー応答時間の計算が可能になります。NTP クライアントとしてのロールでシステムが参照時間サーバーと同期するために、送信元タイムスタンプの付いたパケットが送信されます。パ

ケットが届くと、タイムサーバーが受信先タイムスタンプを追加します。日時情報の要求を処理した後、パケットの返信前に転送先タイムスタンプが追加されます。返信パケットが **NTP** クライアントに届くと、受信先タイムスタンプが生成されます。クライアントはこれで往復時間が計算でき、処理時間を差し引くことで実際の移動時間が導き出されます。送信時間と受信時間が同じだと仮定すると、**NTP** データを受信する際の 1 回の移動での遅延が計算されます。ただし、正式な **NTP** アルゴリズムは、ここで示されているものよりもはるかに複雑です。

時間情報を含むパケットは、受信後にただちに処理されるのではなく、最初に検証され、その後にいくつかの他の時間サンプルと一緒に処理されて、時間を予想します。その後、システムクロックを比較して時間オフセットを決定し、システムクロックの時間と **ntpd** が時間と決定する時間の違いを決定します。システムクロックは、使用されているカウンターの頻度を変更することで、このオフセットを低減するために、最大で 0.5 / ミリ秒の割合で調整されます。この方法を使用してクロックを 1 秒で調整するには、最低 2000 秒かかります。この方法でクロックを 1 秒調整するには、少なくとも 2000 秒かかります。クロックの時間オフセットが 128 ミリ秒を超える場合 (デフォルト設定)、**ntpd** はクロックの転送または後方をステップできます。システムののタイムオフセットが 1000 ミリ秒を超える場合は、ユーザーまたはインストールスクリプトで手動の調整を行う必要があります。3章日付と時刻の設定を参照してください。**-g** オプションを **ntpd** コマンドで使用 (デフォルトで使用) すると、システム起動時のオフセットは修正されますが、通常の操作中に修正されるオフセットは最大 1000 秒までです。

時間を遅らせると、失敗したりエラーになるソフトウェアもいくつかあります。時間のステップ変更の影響を受けるシステムでは、128 ms ではなく、しきい値を **-x** オプション (**-g** オプションとは関連しない) を使用して 600 s に変更できます。**-x** オプションを使用してステップの制限を 0.128 秒から 600 秒に増やすと、クロック制御に異なる方法が使用されるので、マイナス面もあります。カーネルのクロック規範が無効になり、クロックの正確性にマイナスの影響が出る可能性があります。**-x** オプションは、**/etc/sysconfig/ntpd** 設定ファイルに追加することができます。

19.4. 誤差ファイルの概要

誤差ファイルは、通常の周波数で稼働しているシステムクロックと、UTC と同期し続けるために必要な周波数との間の周波数オフセットを保存するために使われます。誤差ファイルに値がある場合は、システム起動時に読み取られ、クロックソースの修正に使われます。誤差ファイルを使用すると、安定的かつ正確な時間の達成に必要な時間が短縮されます。この値は、1 時間ごとに **ntpd** が計算して、誤差ファイルはそのたび置換されます。誤差ファイルは更新されるのではなく置換されるので、**ntpd** が書き込みパーミッションのあるディレクトリーに格納される必要があります。

19.5. UTC、タイムゾーン、および DST

NTP は完全に UTC (協定世界時) を使用しているため、タイムゾーンと夏時間 (DST) はシステムがローカルで適用します。**/etc/localtime** ファイルは、**/usr/share/zoneinfo** のコピーであるか、このファイルへのシンボリックリンクです。RTC は、**/etc/adjtime** の 3 行目に指定してあるとおり、ローカルタイムまたは UTC になります。これは、RTC クロックの設定方法を示す LOCAL または UTC のいずれかです。Date and Time グラフィカル設定ツールの **System Clock Uses UTC** で、この設定を簡単に変更できます。そのツールの使用方法は、3章日付と時刻の設定 を参照してください。夏時間を変更する際には、各種の問題を避けるために RTC を UTC で実行することが推奨されます。

ntpd の詳細な操作方法は、**ntpd(8)** の man ページで説明されています。リソースセクションでは、役に立つ情報ソースを紹介しています。「関連情報」を参照してください。

19.6. NTP の認証オプション

NTPv4 NTPv4 は、公開非対称暗号をベースとしながら対称鍵暗号にも対応している Autokey Security Architecture 向けのサポートを開始しました。Autokey プロトコルについては RFC 5906 Network Time Protocol Version 4: Autokey Specification で説明しています。ただし、このプロトコルには深刻なセキュリティー問題があるため、Red Hat は代わりに対称鍵の使用を強く推奨しています。**ntpd** の認証オプションとコマンドについては、**ntp_auth(5)** man ページで説明しています。

ネットワーク上の攻撃者は、不正確な時間情報のある **NTP** パケットを送信することで、サービス妨害を試みることがあります。**NTP** サーバーのパブリックプールを使用しているシステムでは、**/etc/ntp.conf** のパブリック **NTP** リスト内に 4 つ以上の **NTP** サーバーを記載することでこのリスクが軽減されます。1 つのタイムソースのみが危険にさらされるか、なりすましを受けた場合、**ntpd** はそのソースを無視します。リスク評価を実行し、不正確な時間がアプリケーションおよび組織に及ぼす影響を検討してください。内部のタイムソースがある場合は、**NTP** パケットが配布されるネットワークを保護する手段を検討してください。リスク評価を実行して、リスクを許容でき、アプリケーションへの影響が最小限であると判断した場合は、認証を使わないことを選択することもできます。

ブロードキャストおよびマルチキャストの両モードは、デフォルトで認証を必要とします。ネットワークが信頼できると判断した場合は、**ntp.conf** ファイル内の **disable auth** ディレクティブを使用して認証を無効にできます。別の方法では、SHA1 または MD5 シンメトリックキーを使用して認証を設定するか、Autokey スキームを使用して公開 (非対称) キー暗号法で認証を設定する必要があります。非対称暗号法の Autokey スキームは、**ntp_auth(8)** man ページで、キーの生成については **ntp-keygen(8)** で説明されています。シンメトリックキー暗号法の実装方法は、**key** オプションの説明の「鍵を使用した対称認証の設定」を参照してください。

19.7. 仮想マシン上での時間管理

仮想マシンは実際のハードウェアクロックにアクセスできず、仮想クロックの安定性はホストシステムの作業量に依存することから、十分な安定性がありません。このため、使用する仮想化アプリケーションが準仮想化クロックを提供する必要があります。KVM 搭載した Red Hat Enterprise Linux では、デフォルトのクロックソースは **kvm-clock** です。Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide の KVM guest timing management の章を参照してください。

19.8. うるう秒の概要

グリニッジ標準時 (GMT) は太陽日の測定から導き出されており、これは地球の自転に依存しています。原子時計が最初に作成された際に、より正確な時間の定義が可能になりました。1958 年に国際原子時 (TAI) がより正確で安定的な原子時計を基に導入されました。さらに正確な天文時である世界時 1 (UT1) も導入され、GMT に代わるものとなりました。実際、原子時計は地球の自転よりもはるかに安定しているので、この 2 つの時間の差異が広がり始めました。これが理由で、現実的な方法として UTC が導入されました。これは UT1 の 1 秒以内に維持されますが、わずかな調整を何度も行うことを避けるため、**うるう秒** の概念を導入し、管理可能な方法でこの差異を調整することにしました。UT1 と UTC の差異は、0.5 秒以上になるまで監視されます。1 秒進めるまたは遅らす調整が必要とみなされた場合にのみ、これが実行されます。地球の自転速度は不規則なため、長期の将来にわたって調整の必要性を予測することはできません。調整をいつ行うかについては、**国際地球回転・基準系事業 (IERS)** が判断します。しかし、**NTP** は保留中のうるう秒についての情報を転送し、これらを自動的に適用するので、この発表が重要となるのは、Stratum 1 サーバーの管理者のみになります。

19.9. NTPD 設定ファイルについて

ntpd デーモンは、システム起動時またはサービスの再起動時に設定ファイルを読み取ります。このファイルのデフォルトの位置は **/etc/ntp.conf** で、以下のコマンド入力して確認することができます。

```
~]$ less /etc/ntp.conf
```

設定コマンドについては、本章の後半「NTP の設定」で簡単に説明されており、詳しくは **ntp.conf(5)** man ページで説明されています。

ここでは、以下でデフォルトの設定ファイルを簡単に説明します。

driftfile エントリー

drift ファイルへのパスが指定されていると、*Red Hat Enterprise Linux* のデフォルトエントリーは以下のようになります。

```
driftfile /var/lib/ntp/drift
```

これを変更する場合は、ディレクトリーが**ntpd** で書き込み可能となっていることを確認してください。ファイルには、システムもしくはサービス起動時に毎回システムクロックの周波数を調整する値が含まれます。詳細情報は、*誤差ファイルの概要* を参照してください。

アクセス制御エントリー

以下の行は、デフォルトのアクセス制御制限を設定します。

```
restrict default nomodify notrap nopeer noquery
```

- **nomodify** オプションは、設定に変更が加えられないようにします。
- **notrap** オプションは、**ntpdc** 制御メッセージプロトコルトラップを防ぎます。
- **nopeer** オプションは、ピア関連付けが形成されないようにします。
- **noquery** オプションは、**ntpq** および**ntpdc** クエリーへの応答を防ぎますが、タイムクエリーは除外されます。

重要

ntpq および**ntpdc** クエリーは増幅攻撃に使用できます。そのため、アクセスを公開しているシステムでは、**restrict default** コマンドから**noquery** オプションの指定を外さないでください。

詳細は、CVE-2013-5211 を参照してください。

127.0.0.0/8 範囲内のアドレスは、種々のプロセスやアプリケーションが必要とすることがあります。上記の "restrict default" 行は明示的に許可されていないすべてのものへのアクセスを妨害するので、**IPv4** および**IPv6** の標準ループバックアドレスへのアクセスは以下の行で許可されます。

```
# the administrative functions.
restrict 127.0.0.1
restrict ::1
```

別のアプリケーションで特に必要とされる場合は、アドレスはすぐ下に追加できます。

ローカルネットワーク上のホストは、上記の "restrict default" 行のために許可されません。これを変更して、たとえば**192.0.2.0/24** ネットワークからのホストが時間および統計情報のみをクエリーできるようにするには、以下の形式の行が必要になります。

```
restrict 192.0.2.0 mask 255.255.255.0 nomodify notrap nopeer
```

特定のホスト、たとえば**192.0.2.250/32** からの無制限のアクセスを許可するには、以下の形式の行が必要になります。

```
restrict 192.0.2.250
```

*指定がない場合は、**255.255.255.255** のマスクが適用されます。*

*制限コマンドは、**ntp_acc(5)** man ページで説明されています。*

公開サーバーエントリー

*デフォルトでは、以下の 4 つの公開サーバーエントリーが **ntp.conf** ファイルに格納されています。*

```
server 0.rhel.pool.ntp.org iburst
server 1.rhel.pool.ntp.org iburst
server 2.rhel.pool.ntp.org iburst
server 3.rhel.pool.ntp.org iburst
```

ブロードキャストマルチキャストサーバーエントリー

*デフォルトでは、**ntp.conf** ファイルにはコメントアウトされた例がいくつか含まれています。これらは見ればすぐにわかります。特定のコマンドの説明は、「NTP の設定」を参照してください。必要に応じて、例のすぐ下にコマンドを追加します。*

注記

***DHCP** クライアントプログラムである dhclient は、**DHCP** サーバーから **NTP** サーバーのリストを受信すると、これに **ntp.conf** を追加してサービスを再起動します。この機能を無効にするには、**PEERNTP=no** を **/etc/sysconfig/network** に追加します。*

19.10. NTPD SYSCONFIG ファイルの概要

*このファイルは、サービス起動時に **ntpd** init スクリプトが読み取ります。デフォルトのコンテンツは以下のとおりです。*

```
# Command line options for ntpd
OPTIONS="-g"
```

***-g** オプションを指定すると、**ntpd** は 1000 秒のオフセット制限を無視し、オフセットが 1000 秒よりも大きい場合でも、システムの起動時にのみ同期を試みます。このオプションを使用しないと、タイムオフセットが 1000 秒を超える場合、ntpd は終了します。また、**-g** オプションを使用しても、サービスが再起動し、オフセットが 1000 秒を超えると、システムの起動後に終了します。*

19.11. CHRONY の無効化

***ntpd** を使用するには、デフォルトのユーザースペースデーモンである **chronyd** を停止して、無効にする必要があります。これには、**root** で以下のコマンドを発行します。*

```
~]# systemctl stop chronyd
```

*システム起動時に自動的に起動しないようにするには、**root** で以下のコマンドを発行します。*

```
~]# systemctl disable chronyd
```

***chronyd** 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。*

```
~]$ systemctl status chronyd
```

19.12. NTP デーモンのインストールを確認する

ntpd がインストールされていることを確認するには、**root** で以下のコマンドを発行します。

```
~]# yum install ntp
```

NTP デーモンまたはサービス **ntpd** で実装されます。これは、**ntp** パッケージに含まれています。

19.13. NTP デーモン (NTPD) のインストール

ntpd をインストールするには、**root** で以下のコマンドを発行します。

```
~]# yum install ntp
```

システム起動時に **ntpd** を有効にするには、**root** で以下のコマンドを発行します。

```
~]# systemctl enable ntpd
```

19.14. NTP ステータスの確認

ntpd が実行中で、システム起動時に実行される設定になっていることを確認するには、以下のコマンドを発行します。

```
~]$ systemctl status ntpd
```

ntpd から簡単なステータスレポートを取得するには、以下のコマンドを発行します。

```
~]$ ntpstat
unsynchronised
 time server re-starting
  polling server every 64 s
```

```
~]$ ntpstat
synchronised to NTP server (10.5.26.10) at stratum 2
  time correct to within 52 ms
  polling server every 1024 s
```

19.15. 着信 NTP パケットを許可するファイアウォールの設定

NTP トラフィックはポート **123** 上の **UDP** パケットで設定されており、**NTP** が機能するにはネットワークおよびホストベースのファイアウォール通過が許可されている必要があります。

グラフィカルの Firewall Configuration ツールを使用して、ファイアウォールの設定でクライアントが着信 **NTP** トラフィックを許可しているかどうかを確認します。

グラフィカルの **firewall-config** ツールを起動するには、**Super** キーを押してアクティビティーの概要に入り、**firewall** と入力して **Enter** を押します。**Firewall Configuration** ウィンドウが開きます。次にパスワード入力が求められます。

コマンドラインを使用してグラフィカルなファイアウォール設定ツールを起動するには、**root** で以下のコマンドを入力します。

```
~]# firewall-config
```

Firewall Configuration ウィンドウが開きます。このコマンドは一般ユーザーとしても実行できますが、その場合は時折 **root** パスワードの入力を求められることに注意してください。

ウィンドウ左下の接続しましたの表示を確認してください。これは、**firewall-config** ツールがユーザースペースデーモン **firewalld** に接続されていることを示します。

19.15.1. ファイアウォールの設定変更

現行のファイアウォール設定をただちに変更するには、設定 というラベルのドロップダウン選択メニューが 実行時 に設定されていることを確認します。または、システムの次回の起動時またはファイアウォールの再読み込み時に設定が適用されるように編集するには、ドロップダウンリストから 永続 を選択します。

注記

Runtime モードでファイアウォール設定を変更する際には、サービスに関連するチェックボックスにチェックを入れたり解除したりすると、その選択がただちに反映されます。他のユーザーが使用している可能性のあるシステムでこの作業を行う場合は、この点を考慮してください。

Parmanent モードでファイアウォール設定を変更すると、ファイアウォールを再読み込みした場合か、システムの再起動時にのみ変更が反映されます。ファイアウォールを再読み込みするには、オプションメニューを選択して **Reload Firewall** を選択します。

19.15.2. NTP パケット用にファイアウォールでポートを開く

特定のポートへのトラフィックがファイアウォールを通過できるようにするには、**firewall-config** ツールを起動して、設定を変更するネットワークゾーンを選択します。ポート タブを選んで 追加 ボタンをクリックします。**Port and Protocol** ウィンドウが開きます。

ポート番号 **123** を入力し、ドロップダウンリストから **udp** を選択します。

19.16. NTPDATE サーバーの設定

ntpdate サービスの目的は、システム起動時にクロックを設定することです。このサービスはこれまで、**ntpdate** が正確な時間を確保して、クロックでジャンプが生じないようにしてからサービスが開始するために使われていました。**ntpdate** の使用と step-tickers のリストは非推奨とみなされるため、Red Hat Enterprise Linux 7 では **-g** オプションを **ntpd** コマンドに指定しますが、デフォルトでは **ntpdate** は使用しません。

Red Hat Enterprise Linux 7 の **ntpdate** サービスは、**ntpd service** なしで使用するか、**ntpd** コマンドに **-x** オプションが指定されている場合は利点があります。**ntpd** が **-x** とともに使用されていも、**ntpdate** サービスを有効にしない場合は、時間の差が 600 秒より大きい場合に限りステップにより修正されます。600 秒に満たない場合は徐々に修正されるため、1 秒修正するために約 2000 秒かかります。

ntpdate がシステム起動時に実行されるようになっていることを確認するには、以下のコマンドを発行します。

```
~]$ systemctl status ntpdate
```

システム起動時にサービスが実行するようにするには、**root** で以下のコマンドを発行します。

```
~]# systemctl enable ntpdate
```

Red Hat Enterprise Linux 7 では、デフォルトの **/etc/ntp/step-tickers** ファイルには **0.rhel.pool.ntp.org** が含まれます。追加の **ntpdate** サーバーを設定するには、テキストエディターを **root** で実行し、**/etc/ntp/step-tickers** を編集します。**ntpdate** は、システム起動時に日付情報を取得するためにこのファイルを 1 回使用するだけなので、記載されているサーバー数は重要ではありません。内部のタイムサーバーがある場合は、そのホスト名を 1 行目に使います。2 行目に追加のホストをバックアップとしておくのがよいでしょう。バックアップサーバーにどれを選ぶか、また 2 番目のホストを内部または外部とするかは、リスク評価によります。たとえば、1 番目のサーバーに影響する問題が 2 番目のサーバーにも影響する可能性はどの程度か。1 番目のサーバーにアクセスできなくなるネットワーク障害時に、より接続性が高いのは外部サーバーかそれとも内部サーバーか、といった点を考慮します。

19.17. NTP の設定

NTP サービスのデフォルト設定を変更するには、**root** でテキストエディターを使用して **/etc/ntp.conf** ファイルを編集します。このファイルは、**ntpd** とともにインストールされ、Red Hat プールからのタイムサーバーを使用するデフォルト設定になっています。**ntp.conf(5)** man ページでは、アクセスおよびレート制限コマンドを除く、設定ファイルで使用可能なコマンドオプションが説明されています。アクセスおよびレート制限コマンドは、**ntp_acc(5)** man ページで説明されています。

19.17.1. NTP サービスへのアクセス制御の設定

システム上で実行中の **NTP** サービスへのアクセスを制限または制御するには、**ntp.conf** ファイル内の **restrict** コマンドを利用します。コメントアウトされた例は以下のとおりです。

```
# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

restrict コマンドは以下の形式になります。

```
restrict address [mask mask] option
```

address および mask は、制限を適用する IP アドレスを指定します。option は 1 つ以上の制限になります。

- **ignore: ntpq** および **ntpdc** クエリーを含むすべてのパケットが無視されます。
- **kod:** Kiss-o'-death パケットが送信され、不要なクエリーが少なくなります。
- **limited:** パケットがレート制限のデフォルト値または **discard** コマンドで指定された値に違反する場合は、タイムサービス要求に応答しません。**ntpq** および **ntpdc** クエリーは影響を受けません。**discard** コマンドおよびデフォルト値に関する詳細情報は、「NTP サービスへのレート制限アクセスの設定」 を参照してください。
- **lowpriotrap:** 一致するホストがトラップを低い優先度に設定します。
- **nomodify:** 設定に変更を加えられないようにします。
- **noquery: ntpq** および **ntpdc** クエリーに応答しないようにしますが、タイムクエリーは除外されます。
- **nopeer:** ピア関連付けの形成をできないようにします。
- **noserve: ntpq** および **ntpdc** クエリーを除くすべてのパケットを拒否します。

- **notrap**: **ntpdc** 制御メッセージプロトコルトラップをできないようにします。
- **notrust**: 暗号法で認証されないパケットを拒否します。
- **ntpport**: 発信元ポートが標準の **NTP UDP** ポート **123** の場合、一致アルゴリズムが制限のみを適用するように修正します。
- **version**: 現在の **NTP** バージョンに一致しないパケットを拒否します。

レート制限アクセスがクエリーに対してまったく応答しないように設定するには、各 **restrict** コマンドに **limited** オプションを指定する必要があります。**ntpd** が **KoD** パケットで応答するには、**restrict** コマンドにオプションが **limited** および **kod** を指定する必要があります。

ntpq および **ntpdc** クエリーは増幅攻撃に使用できます (詳細は CVE-2013-5211 を参照)。そのため、アクセスを公開しているシステムでは、**restrict default** コマンドから **noquery** オプションの指定を外さないでください。

19.17.2. NTP サービスへのレート制限アクセスの設定

システムで実行している **NTP** サービスへのレート制限アクセスを有効にするには、「NTP サービスへのアクセス制御の設定」 で説明しているように、**limited** オプションを **restrict** コマンドに追加します。デフォルトの discard パラメーターを使用したくない場合は、以下の説明にあるように **discard** コマンドも使用できます。

discard コマンドは以下の形式を取ります。

> ***discard average*** value ***minimum*** value ***monitor*** value

- **average**: 許可される最小平均パケット範囲を指定します。log2 秒で引数を受け入れます。デフォルト値は 3 (2^3 は 8 秒) です。
- **minimum**: 許可される最小限のパケット間隔を指定し、log2 秒の引数を取ります。デフォルト値は 1 (2^1 は 2 に相当) です。
- **monitor**: 許可されるレート制限を超えた場合のパケットの discard の確率を指定します。デフォルト値は、3000 秒です。このオプションは、1 秒あたり 1000 以上のリクエストを受信するサーバー用に用意されています。

discard コマンドの例は次のとおりです。

> ***discard average 4***

> ***discard average 4 minimum 2***

19.17.3. ピアアドレスの追加

ピアのアドレス、つまり同一 stratum の **NTP** サービスを実行しているサーバーのアドレスを追加するには、**ntp.conf** ファイルの **peer** コマンドを利用します。

peer コマンドは以下の形式になります。

> ***peer*** address

ここでの address は、**IP** ユニキャストアドレスまたは **DNS** の解決可能な名前になります。アドレス

は、同一stratum のメンバーに既知のシステムのものである必要があります。ピアは、相互に異なる時間ソースを少なくとも 1 つ持っている必要があります。ピアは通常、同一管理制御下にあるシステム群です。

19.17.4. サーバーアドレスの追加

サーバーのアドレス、つまり一段階上のstratum の **NTP** サービスを実行しているサーバーのアドレスを追加するには、**ntp.conf** ファイルの **server** コマンドを利用します。

server コマンドは以下の形式になります。

```
server address
```

ここでのaddress は、**IP** ユニキャストアドレスまたは **DNS** の解決可能な名前になります。パケットの送信元となるリモートリファレンスサーバーまたはローカルリファレンスクロックのアドレスになります。

19.17.5. ブロードキャストまたはマルチキャストサーバーアドレスの追加

送信用のブロードキャストまたはマルチキャストアドレス、つまり **NTP** パケットをブロードキャストまたはマルチキャストする宛先のアドレスを追加 するには、**ntp.conf** ファイルの **broadcast** コマンドを利用します。

ブロードキャストおよびマルチキャストの両モードは、デフォルトで認証を必要とします。「NTP の認証オプション」を参照してください。

broadcast コマンドは以下の形式になります。

```
broadcast address
```

ここでのaddress は、パケットの送信先となる **IP** ブロードキャストまたはマルチキャストアドレスになります。

このコマンドは、システムが **NTP** ブロードキャストサーバーとして作動するように設定します。使用するアドレスは、ブロードキャストかマルチキャストアドレスである必要があります。ブロードキャストアドレスの場合は、**IPv4** アドレス **255.255.255.255** を意味します。デフォルトでは、ルーターはブロードキャストメッセージを渡しません。マルチキャストアドレスの場合は、**IPv4** Class D アドレスか **IPv6** アドレスになります。IANA は **IPv4** マルチキャストアドレス **224.0.1.1** と **IPv6** アドレス **FF05::101** (サイトローカル) を **NTP** に割り当てています。RFC 2365 Administratively Scoped IP Multicastの説明にあるように、管理者が範囲指定した **IPv4** マルチキャストアドレスも使用可能です。

19.17.6. Manycast クライアントアドレスの追加

manycast クライアントアドレスを追加する、つまり **NTP** サーバーの発見に使用するマルチキャストアドレスを設定するには、**ntp.conf** ファイルの **manycastclient** コマンドを利用します。

manycastclient コマンドは以下の形式になります。

```
manycastclient address
```

ここでのaddress は **IP** マルチキャストアドレスで、ここからパケットが受信されます。クライアントはこのアドレスにリクエストを送信し、応答から最善のサーバーを選んで他を無視します。その後は、**NTP** 通信は発見された **NTP** サーバーが **ntp.conf** リストされているかのようにユニキャスト関連付けを使用します。

*このコマンドは、システムが**NTP** クライアントのように動作するように設定します。システムは、同時にクライアントとサーバーの両方になることができます。*

19.17.7. ブロードキャストクライアントアドレスの追加

*ブロードキャストクライアントのアドレスを追加するには、つまりブロードキャスト **NTP** パケット用にブロードキャストアドレスを監視するように設定するには、**ntp.conf** ファイルの**broadcastclient** コマンドを利用します。*

***broadcastclient** コマンドは以下の形式になります。*

> ***broadcastclient***

ブロードキャストメッセージを受信可能にします。デフォルトで認証を必要とします。「NTP の認証オプション」を参照してください。

*このコマンドは、システムが**NTP** クライアントのように動作するように設定します。システムは、同時にクライアントとサーバーの両方になることができます。*

19.17.8. Manycast サーバーアドレスの追加

*manycast サーバーのアドレスを追加する、つまり **NTP** パケットをマルチキャストすることでクライアントがサーバーを発見できるようにするアドレスを設定するには、**ntp.conf** ファイルの **manycastserver** コマンドを利用します。*

***manycastserver** コマンドは以下の形式になります。*

> ***manycastserver*** **address**

*マルチキャストメッセージの送信ができるようになります。ここでの*address *は、マルチキャスト送信先のアドレスになります。これは認証と合わせて使用して、サービス中断を防ぎます。*

*このコマンドは、システムが**NTP** サーバーのように動作するように設定します。システムは、同時にクライアントとサーバーの両方になることができます。*

19.17.9. マルチキャストクライアントアドレスの追加

*マルチキャストクライアントのアドレスを追加するには、つまりマルチキャスト **NTP** パケット用にマルチキャストアドレスを監視するように設定するには、**ntp.conf** ファイルの**multicastclient** コマンドを利用します。*

***multicastclient** コマンドは以下の形式になります。*

> ***multicastclient*** **address**

*マルチキャストメッセージの受信ができるようになります。ここでの*address *は、サブスクライブするアドレスになります。これは認証と合わせて使用して、サービス中断を防ぎます。*

*このコマンドは、システムが**NTP** クライアントのように動作するように設定します。システムは、同時にクライアントとサーバーの両方になることができます。*

19.17.10. Burst オプションの設定

公開サーバーに対して **burst** オプションを使用することは、濫用とみなされます。公開 **NTP** サーバーでは、このオプションを使用しないでください。このオプションは、組織内のアプリケーションにのみ使用するようにしてください。

時間オフセットの統計情報の平均的な品質を向上させるには、サーバーコマンドの最後に以下のオプションを追加します。

> ***burst***

サーバーが反応している場合は、ポーリングの間隔ごとにシステムが通常の 1 パケットではなく、最大 8 パケットのバーストを送信します。**server** コマンドを使うと、時間オフセット計算の平均的な質が向上します。

19.17.11. iburst オプションの設定

初回同期にかかる時間を改善するには、サーバーコマンドの最後に以下のオプションを追加します。

> ***iburst***

サーバーに到達できない場合は、1 パケットではなく、8 パケットのバーストが送信されます。パケットの間隔は通常は 2 秒間隔ですが、1 番目と 2 番目のパケット間の間隔は、モデムまたは ISDN 呼び出しの完了に必要な追加の時間を許可できるように **calldelay** コマンドを使用して変更することができます。**server** コマンドと使用すると、初回の同期にかかる時間が短縮されます。これが設定ファイルでのデフォルトオプションになります。

19.17.12. 鍵を使用した対称認証の設定

鍵を使用した対称認証を設定するには、サーバーまたはピアコマンドの最後に以下のオプションを追加します。

> ***key*** number

ここでの number は、**1** から **65534** までの数字になります。このオプションを使うと、パケット内で **メッセージ認証コード** (MAC) が使えるようになります。このオプションは、**peer**、**server**、**broadcast**、および **manycastclient** の各コマンドで使用します。

このオプションは、**/etc/ntp.conf** 内で以下のように使用します。

```
server 192.168.1.1 key 10
broadcast 192.168.1.255 key 20
manycastclient 239.255.254.254 key 30
```

「NTP の認証オプション」 も参照してください。

19.17.13. ポーリング間隔の設定

デフォルトのポーリング間隔を変更するには、サーバーまたはピアコマンドの最後に以下のオプションを追加します。

> ***minpoll*** value ***and maxpoll*** value

デフォルトのポーリング間隔を変更するオプション。ここでは、秒単位の間隔を 2 の **値** 乗で計算します。つまり、間隔は log2 秒で表示されます。デフォルトの **minpoll** 値は 6 で、2^6 は 64 に相当しま

*す。**maxpoll** のデフォルト値は 10 で、1024 秒に相当します。使用できる値は、3 から 17 の範囲で、それぞれ 8 s から 36.4 h に相当する値になります。これらのオプションは、**peer** または **server** で使用します。**maxpoll** を短く設定すると、クロックの精度が向上する場合があります。*

19.17.14. サーバー優先順位の設定

特定のサーバーが他の同様の統計情報のサーバーよりも優先されるよう指定するには、サーバーまたはピアコマンドの最後に以下のオプションを追加します。

```
prefer
```

*他の同様の統計情報のサーバーに優先して、このサーバーが同期に使用されます。このオプションは、**peer** または **server** コマンドで使用します。*

19.17.15. NTP パケットの Time-to-Live (有効期限) の設定

デフォルトで使用される特定の time-to-live (TTL: 有効期限) の値を指定するには、サーバーまたはピアコマンドの最後に以下のオプションを追加します。

```
ttl value
```

*ブロードキャストサーバーおよびマルチキャスト **NTP** サーバーが送信するパケットで使用される **TTL** の値を指定します。**manycast** クライアントが **expanding ring search** を使用するには、**TTL** の最大値を指定します。デフォルト値は **127** です。*

19.17.16. 使用する NTP バージョンの設定

*デフォルトで使用される特定バージョンの **NTP** を指定するには、サーバーまたはピアコマンドの最後に以下のオプションを追加します。*

```
version value
```

*作成された **NTP** パケット内の **NTP** セットのバージョンを指定します。値は **1** から **4** になります。デフォルト値は **4** です。*

19.18. ハードウェアクロック更新の設定

リアルタイムクロック (RTC) とも呼ばれるハードウェアクロックの更新には、システムクロックが使用できます。本セクションでは、これを実行する 3 つのアプローチを説明します。

即時ワンタイム更新

*ハードウェアクロックの即時ワンタイム更新を行うには、**root** で以下のコマンドを実行します。*

```
~]# hwclock --systohc
```

ブートごとの更新

ntpdate 同期ユーティリティー実行後にブートごとに毎回ハードウェアクロックが更新するようにするには、以下を実行します。

a. *以下の行を **/etc/sysconfig/ntpdate** ファイルに追加します。*

```
SYNC_HWCLOCK=yes
```

b. *ntpdate サービスを root で有効にします。*

```
~]# systemctl enable ntpdate.service
```

ntpdate サービスは /etc/ntp/step-tickers ファイルで定義されている NTP サーバーを使用することに留意してください。

注記

仮想マシンの場合は、仮想マシン自体の次回起動時ではなく、ホストマシンの次回起動時にハードウェアクロックが更新されます。

NTP から更新

ntpd または chronyd サービスを使うと、システムクロックが更新されるたびにハードウェアクロックを更新することができます。
root として ntpd サービスを起動します。

```
~]# systemctl start ntpd.service
```

この動作を次回ブート後にも維持するには、サービスがブート時に自動的に起動するようにします。

```
~]# systemctl enable ntpd.service
```

または

root で chronyd サービスを起動します。

```
~]# systemctl start chronyd.service
```

この動作を次回ブート後にも維持するには、サービスがブート時に自動的に起動するようにします。

```
~]# systemctl enable chronyd.service
```

その結果、ntpd または chronyd がシステムクロックを同期するたびに、カーネルがハードウェアクロックを 11 分ごとに自動更新します。

警告

上記の 11 分モードは常に有効になっているわけではないので、このアプローチが常に機能するとは限りません。このため、ハードウェアクロックはシステムクロックの更新時に更新されるとは限りません。

ソフトウェアクロックがハードウェアクロックに同期しているかを確認するには、root として ntpdc -c kerninfo または ntptime コマンドを使用します。

```
~]# ntpdc -c kerninfo
```

以下のような結果になります。

```
pll offset:      0 s
pll frequency:    0.000 ppm
maximum error:    8.0185 s
estimated error:   0 s
status: 2001 pll nano
pll time constant:  6
precision:      1e-09 s
frequency tolerance: 500 ppm
```

または

```
~]# ntptime
```

以下のような結果になります。

```
ntp_gettime() returns code 0 (OK)
 time dcba5798.c3dfe2e0 Mon, May 8 2017 11:34:00.765, (.765135199),
 maximum error 8010000 us, estimated error 0 us, TAI offset 0
ntp_adjtime() returns code 0 (OK)
 modes 0x0 (),
 offset 0.000 us, frequency 0.000 ppm, interval 1 s,
 maximum error 8010000 us, estimated error 0 us,
status 0x2001 (PLL,NANO),
 time constant 6, precision 0.001 us, tolerance 500 ppm,
```

ハードウェアクロックがシステムクロックに同期しているかどうかを確認するには、出力の status 行を確認します。その行に **unsync** または **UNSYNC** という単語が含まれている場合は、ハードウェアクロックがシステムクロックと同期していません。

ハードウェアクロックがシステムクロックに同期しています。

```
status 0x2001 (PLL,NANO)
```

ハードウェアクロックがシステムクロックに同期していません。

```
status 0x41 (PLL,UNSYNC)
```

19.19. クロックソースの設定

システムで使用可能なクロックソースをリスト表示するには、以下のコマンドを発行します。

```
~]$ cd /sys/devices/system/clocksource/clocksource0/
clocksource0]$ cat available_clocksource
kvm-clock tsc hpet acpi_pm
clocksource0]$ cat current_clocksource
kvm-clock
```

上記の例では、カーネルは **kvm-clock** を使用しています。これは仮想マシンなので、起動時にこのクロックソースが選択されています。利用可能なクロックソースはアーキテクチャーに依存することに注意してください。

デフォルトのクロックソースを上書きするには、**clocksource** ディレクティブをカーネルの **GRUB 2** メニューエントリーの末尾に追加します。**grubby** ツールを使用して変更します。たとえば、システムのデフォルトのカーネルが **tsc** クロックソースを使用するように強制するには、以下のコマンドを入力します。

```
~]# grubby --args=clocksource=tsc --update-kernel=DEFAULT
```

--update-kernel パラメーターはキーワード **ALL**、またはカーネルインデックス番号のコンマ区切りのリストも受け入れます。

GRUB 2 メニューの変更方法は、*26章GRUB 2 での作業* を参照してください。

19.20. 関連情報

以下の情報ソースで **NTP** および **ntpd** に関する追加リソースが提供されています。

19.20.1. インストールされているドキュメント

- **ntpd(8)** man ページ: **ntpd** の詳細な説明があり、コマンドラインオプションも含まれています。
- **ntp.conf(5)** man ページ: サーバーおよびピアとの関連付けの設定方法に関する情報が含まれています。
- **ntpq(8)** man ページ: **NTP** サーバーの監視およびクエリー用の **NTP** クエリーユーティリティーが説明されています。
- **ntpdc(8)** man ページ: **ntpd** の状態をクエリー、変更するための **ntpd** ユーティリティーを説明しています。
- **ntp_auth(5)** man ページ: **ntpd** の認証オプション、コマンド、および鍵管理を説明しています。
- **ntp_keygen(8)** man ページ: **ntpd** の公開および秘密鍵生成を説明しています。
- **ntp_acc(5)** man ページ: **restrict** コマンドを使用したアクセス制御オプションを説明しています。
- **ntp_mon(5)** man ページ: 統計情報収集に関する監視オプションを説明しています。
- **ntp_clock(5)** man ページ: 基準クロックを設定するコマンドを説明しています。
- **ntp_misc(5)** man ページ: その他のオプションが説明されています。
- **ntp_decode(5)** man ページ: **ntpd** レポーティングおよび監視に使用されるステータス文字、イベントメッセージ、およびエラーコードをリスト表示します。
- **ntpstat(8)** man ページ: ローカルマシン上で実行している **NTP** デーモンの同期状態をレポートするユーティリティーを説明しています。
- **ntptime(8)** man ページ: カーネル時間変数を読み取り、設定するユーティリティーを説明しています。

- ***tickadj(8)*** *man ページ: ティックの長さを読み取り、オプションで設定するユーティリティーを説明しています。*

19.20.2. 便利な Web サイト

http://doc.ntp.org/

NTP 資料のアーカイブ

http://www.eecis.udel.edu/~mills/ntp.html

Network Time Synchronization Research Project

http://www.eecis.udel.edu/~mills/ntp/html/manyopt.html

NTPv4 *での Automatic Server Discovery に関する情報*

第20章 PTP4L を使用した PTP の設定

20.1. PTP の概要

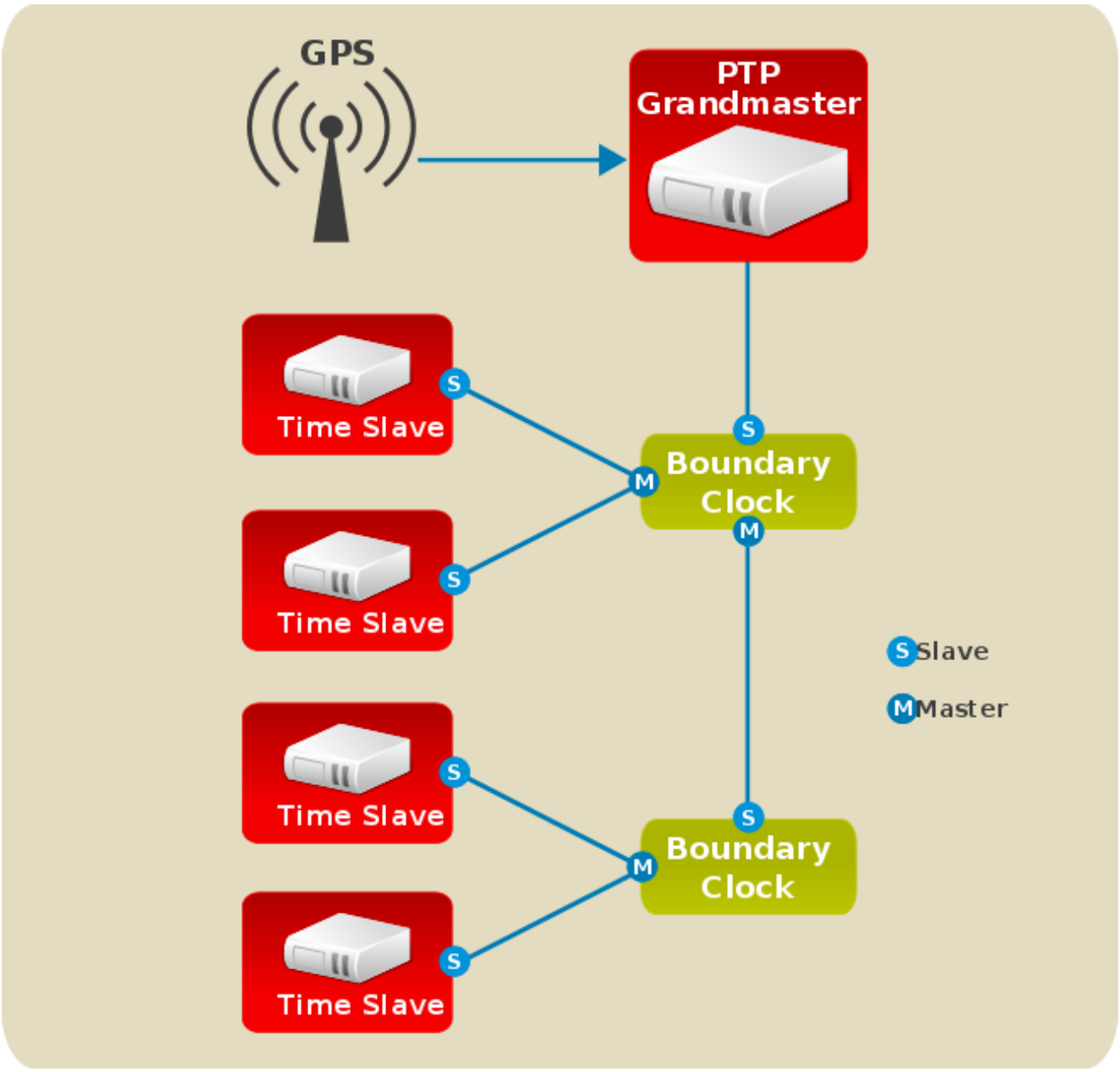
Precision Time Protocol *(PTP)* は、ネットワーク内でのクロック同期に使用されるプロトコルです。ハードウェアサポートと合わせて使用すると **PTP** はマイクロ秒以下の正確性があり、これは **NTP** で得られる正確性よるもはるかに優れています。**PTP** サポートは、カーネルとユーザースペースに分けられます。Red Hat Enterprise Linux のカーネルには **PTP** クロックのサポートが含まれており、これはネットワークドライバーが提供しています。実際のプロトコル実装は linuxptp と呼ばれ、これは Linux の IEEE 標準 1588 に準拠した **PTPv2** 実装です。

linuxptp パッケージには、クロック同期用の ptp4l および phc2sys プログラムが含まれています。ptp4l プログラムは、**PTP** 境界クロックと通常のクロックを実装します。ハードウェアタイムスタンプでは **PTP** ハードウェアクロックのマスタークロックとの同期に使用され、ソフトウェアタイムスタンプではシステムクロックのマスタークロックとの同期に使用されます。phc2sys プログラムは、システムクロックを network interface card *(NIC)* 上の **PTP** ハードウェアクロックと同期するハードウェアタイムスタンプでのみ必要となります。

20.1.1. PTP を理解する

PTP で同期するクロックは、マスター/スレーブ階層で組織されています。スレーブはマスターと同期し、このマスターは別のマスターのスレーブとなっています。この階層は、best master clock *(BMC)* アルゴリズムで作成され、自動的に更新されます。クロックにポートが 1 つしかない場合は、**マスター** または **スレーブ** にすることができます。このようなクロックは **通常のクロック** *(OC)* と呼ばれます。クロックにポートが 1 つしかない場合、これはマスターにもスレーブにもなることができ、boundary クロック *(BC)* と呼ばれます。トップレベルのマスターは、Global Positioning System *(GPS)* のタイムソースを使用して同期できる **グランドマスタークロック** と呼ばれます。GPS ベースの時間ソースを使うことで、高度の正確性を保って異なるネットワークが同期可能になります。

図20.1 PTP グランドマスター、境界、スレーブの各クロック



20.1.2. PTP の利点

PTP がNetwork Time Protocol (NTP) よりも優れている点の1 つは、様々なnetwork interface controllers (NIC) およびネットワークスイッチにあるハードウェアサポートです。この特化されたハードウェアにより、**PTP** はメッセージ送信の遅れを説明でき、時間同期の精度を大幅に高められます。ネットワーク内でPTP 以外に対応するハードウェアコンポーネントを使用することは可能ですが、その場合、変動が増えたり、遅れが非対称となったりして、同期が不正確になります。これにより、通信パスで使用される複数の非PTP 対応コンポーネントが追加されます。可能な限りの精度を実現するには、**PTP** クロック間の全ネットワークコンポーネントが**PTP** ハードウェアを有効にすることが推奨されます。ネットワークハードウェアが**PTP** に対応していない大規模なネットワークでの時間同期は、**NTP** に適している場合があります。

ハードウェア**PTP** サポートでは、NIC には独自のオンボードクロックがあります。これは受信および送信の**PTP** メッセージのタイムスタンプで使用されます。**PTP** マスターに同期しているオンボードクロックで、コンピューターのシステムクロックはNIC 上の**PTP** ハードウェアクロックに同期されます。ソフトウェア**PTP** サポートでは、システムクロックは**PTP** メッセージのタイムスタンプに使用され、**PTP** マスターに直接同期します。ソフトウェア**PTP** サポートではオペレーティングシステムによる追加の**PTP** パケット処理を必要としますが、ハードウェア**PTP** サポートでは、NIC が**PTP** パケットの送受信時の瞬間にタイムスタンプができるので、より優れた正確性が得られます。

20.2. PTP の使用

PTP を使用するには、対象とするインターフェイス用のカーネルネットワークドライバーがソフトウェアまたはハードウェアのタイムスタンプ機能をサポートしている必要があります。

20.2.1. ドライバーおよびハードウェアサポートの確認

ドライバーがハードウェアタイムスタンプをサポートしていることに加え、NIC も物理的なハードウェアでこの機能をサポートできる必要があります。特定のドライバーおよび NIC のタイムスタンプ機能を検証する最善の方法は、ethtool ユーティリティーを使ってインターフェイスにクエリーすることです。この例では、eth3 がチェックする必要があるインターフェイスです。

```
~]# ethtool -T eth3
Time stamping parameters for eth3:
Capabilities:
   hardware-transmit   (SOF_TIMESTAMPING_TX_HARDWARE)
   software-transmit   (SOF_TIMESTAMPING_TX_SOFTWARE)
   hardware-receive   (SOF_TIMESTAMPING_RX_HARDWARE)
   software-receive   (SOF_TIMESTAMPING_RX_SOFTWARE)
   software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
   hardware-raw-clock  (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 0
Hardware Transmit Timestamp Modes:
   off        (HWTSTAMP_TX_OFF)
   on         (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
   none       (HWTSTAMP_FILTER_NONE)
   all        (HWTSTAMP_FILTER_ALL)
```

注記

ethtool によって出力される **PTP Hardware Clock** 値は、PTP ハードウェアクロックのインデックスです。この値は、**/dev/ptp*** デバイスの名前に対応します。最初の PHC のインデックスは 0 です。

ソフトウェアタイムスタンプのサポートについては、パラメーターリストに以下を含めます。

- **SOF_TIMESTAMPING_SOFTWARE**
- **SOF_TIMESTAMPING_TX_SOFTWARE**
- **SOF_TIMESTAMPING_RX_SOFTWARE**

ハードウェアタイムスタンプのサポートについては、パラメーターリストに以下を含めます。

- **SOF_TIMESTAMPING_RAW_HARDWARE**
- **SOF_TIMESTAMPING_TX_HARDWARE**
- **SOF_TIMESTAMPING_RX_HARDWARE**

20.2.2. PTP のインストール

Red Hat Enterprise Linux のカーネルには、**PTP** のサポートが含まれています。ユーザー空間のサポートは、linuxptp パッケージのツールで提供されます。linuxptp をインストールするには、以下のコマンドを **root** で発行します。

```
~]# yum install linuxptp
```

これでptp4l およびphc2sys がインストールされます。

システムクロックの時間を設定するサービスを同時に 2 つ以上実行しないでください。**NTP** を使用して **PTP** 時間を実行したい場合は、「NTP を使用した PTP 時間の実行」 を参照してください。

20.2.3. ptp4l の起動

ptp4l プログラムはコマンドラインから起動することも、サービスとして起動することもできます。サービスとして実行する場合、オプションは **/etc/sysconfig/ptp4l** ファイルに指定されます。サービスおよびコマンドラインの両方で使用する必要のあるオプションは **/etc/ptp4l.conf** ファイルに指定されます。**/etc/sysconfig/ptp4l** ファイルには **-f /etc/ptp4l.conf** コマンドラインオプションが含まれ、これにより **ptp4l** プログラムが **/etc/ptp4l.conf** ファイルの読み取りと、これに含まれるオプションの処理を実行します。**/etc/ptp4l.conf** の使用については、「設定ファイルの指定」 で説明されています。各種の異なる ptp4l オプションおよび設定ファイルの設定についての詳細は、**ptp4l(8)** man ページを参照してください。

ptp4l をサービスとして起動
ptp4l をサービスとして起動するには、**root** で以下のコマンドを発行します。

```
~]# systemctl start ptp4l
```

Red Hat Enterprise Linux 7 でのシステムサービスの管理の詳細については、10 章systemd によるサービス管理 を参照してください。

コマンドラインからのptp4l の使用
ptp4l プログラムは、デフォルトでハードウェアタイムスタンプを使用しようとします。ハードウェアタイムスタンプ対応のドライバーおよびNIC でptp4l を使用するには、使用するネットワークインターフェイスに **-i** オプションを指定します。**root** で次のコマンドを実行します。

```
~]# ptp4l -i eth3 -m
```

eth3 は、設定するインターフェイスに置き換えます。以下は、NIC 上の**PTP** クロックがマスターに同期された際のptp4l からの出力例です。

```
~]# ptp4l -i eth3 -m
selected eth3 as PTP clock
port 1: INITIALIZING to LISTENING on INITIALIZE
port 0: INITIALIZING to LISTENING on INITIALIZE
port 1: new foreign master 00a069.fffe.0b552d-1
selected best master clock 00a069.fffe.0b552d
port 1: LISTENING to UNCALIBRATED on RS_SLAVE
master offset -23947 s0 freq +0 path delay    11350
master offset -28867 s0 freq +0 path delay    11236
master offset -32801 s0 freq +0 path delay    10841
master offset -37203 s1 freq +0 path delay    10583
master offset -7275 s2 freq -30575 path delay  10583
port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
master offset -4552 s2 freq -30035 path delay  10385
```

マスターオフセットの値は、マスターからナノ秒で測定されたオフセットです。**s0**、**s1**、**s2** の各ストリングは、異なるクロックのサーボ状態を示しています。**s0** はアンロックされています。**s1** はクロックステップで、**s2** です。servo がロック状態 (**s2**) の場合は、**pi_offset_const** オプションが設定ファイルの正の値 (**ptp4l(8)** で説明) に設定されていない限り、クロックはステップになりません (徐々に調整されるのみ)。**adj** 値は、10 億分の 1 (ppb) 単位のクロックの周波数調整です。パス遅延値は、マスターから送信される同期メッセージの遅延 (ナノ秒単位) です。Port 0 は、ローカル **PTP** 管理に使用される Unix ドメインソケットです。Port 1 は、**eth3** インターフェイスです (上の例に基づく)。
INITIALIZING、LISTENING、UNCALIBRATED、および SLAVE は、INITIALIZE、RS_SLAVE、MASTER_CLOCK_SELECTED イベントで変化する可能性のあるポート状態です。最後の状態変更メッセージでは、ポート状態が UNCALIBRATED から SLAVE に変更し、**PTP** マスタークロックとの同期が成功したことを示しています。

ptp4l からのメッセージのロギング
デフォルトでは、メッセージは **/var/log/messages** に送信されます。ただし、**-m** オプションを指定すると標準出力へのロギングが可能になり、これはデバッグで役に立ちます。

ソフトウェアタイムスタンプを有効にするには、以下のように **-S** オプションを使用する必要があります。

```
~]# ptp4l -i eth3 -m -S
```

20.2.3.1. 遅延測定メカニズムの選択

遅延測定メカニズムには 2 種類あり、以下のように **ptp4l** コマンドにオプションを追加することで選択できます。

-P

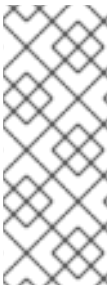
-P は peer-to-peer (P2P) の遅延測定メカニズムを選択します。
P2P メカニズムはネットワークトポロジーの変更に他のメカニズムよりも速く反応し、遅延の測定がより正確であることから、より好まれます。P2P メカニズムが使用できるのは、各ポートが最大で他の 1 つの P2P ポートで PTP メッセージを交換するトポロジーだけです。これは、透過クロックを含む通信パス上のすべてのハードウェアでサポートされ、使用される必要があります。

-E

-E は、エンドツーエンド (E2E) の遅延測定メカニズムを選択します。これはデフォルトです。
E2E メカニズムは、遅延リクエスト-レスポンスメカニズムとも呼ばれます。

-A

-A では、遅延測定メカニズムの自動選択を有効にします。
自動オプションは、E2E で ptp4l を起動します。ピアの遅延リクエストを受け取ると、P2P モードに変更します。



注記

1 つの **PTP** 通信パス上のクロックはすべて、遅延を測定するために同じメカニズムを使用する必要があります。以下の場合に警告が表示されます。

- E2E メカニズムを使用しているポートでピアの遅延リクエストを受け取る場合
- P2P メカニズムを使用しているポートで E2E の遅延リクエストを受け取る場合

20.3. 複数のインターフェイスでの PTP の使用

PTP を異なるネットワークの複数のインターフェイスで使用する場合、reverse path forwarding (逆方向パス転送) モードを loose mode (緩やかなモード) に変更する必要があります。Red Hat Enterprise Linux 7 はデフォルトで Strict Reverse Path Forwarding を使用します。これは、RFC 3704, Ingress Filtering for Multihomed Networks の厳密な逆方向パスに関する推奨事項に準拠します。詳細は、Red Hat Enterprise Linux 7 セキュリティーガイド の 逆方向パス転送 セクションを参照してください。

sysctl ユーティリティーは、チューニング可能なカーネルの値を読み取り、値を書き込むために使用されます。実行中のシステムへの変更は、**sysctl** コマンドを使用してコマンドラインで直接実行できます。永続的な変更は、行を **/etc/sysctl.conf** ファイルに追加することで実行できます。

- loose mode (緩やかなモード) のフィルタリングにグローバルに変更するには、**root** で以下のコマンドを入力します。

```
~]# sysctl -w net.ipv4.conf.default.rp_filter=2
~]# sysctl -w net.ipv4.conf.all.rp_filter=2
```

- ネットワークインターフェイスごとに reverse path filtering mode (逆パスフィルタリングモード) を変更するには、すべての PTP インターフェイスで **net.ipv4.**interface**.rp_filter** コマンドを使用します。たとえば、デバイス名が **em1** のインターフェイスの場合は、以下のようになります。

```
~]# sysctl -w net.ipv4.conf.em1.rp_filter=2
```

これらの設定を再起動しても保持するには、**/etc/sysctl.conf** ファイルを変更します。すべてのインターフェイスのモード、または特定のインターフェイスのモードを変更できます。

すべてのインターフェイスのモードを変更するには、**root** ユーザーとして実行しているエディターで **/etc/sysctl.conf** ファイルを開き、以下の行を追加します。

```
net.ipv4.conf.all.rp_filter=2
```

一部のインターフェイスのみを変更するには、以下の形式で複数の行を追加します。

```
net.ipv4.conf.interface.rp_filter=2
```

注記

すべてのインターフェイスと特定のインターフェイスの設定を使用する場合、各インターフェイスのソースを検証する際に **conf/{all,interface}/rp_filter** の最大値が使用されます。

default 設定を使用してモードを変更することも可能です。これは、新規に作成されたインターフェイスのみに適用されることを意味します。

sysctl パラメーターにおける all、**default**、または特定のデバイス設定の使用に関する詳細は、Red Hat ナレッジベースの記事 What is the difference between "all", "default" and a specific device in a sysctl parameter? を参照してください。

起動プロセス中に実行される **sysctl** サービスのタイミングにより、2 つのタイプの問題が発生する可能性がある点に注意してください。

1. ドライバーは、**sysctl** サービスの実行前に読み込まれます。

この場合、影響のあるネットワークインターフェイスは、カーネルで事前に設定されたモードを使用し、**sysctl** デフォルトは無視されます。

この問題の解決方法については、Red Hat ナレッジベースの記事 What is the difference between "all", "default" and a specific device in a sysctl parameter? を参照してください。

2. ドライバーは、**sysctl** サービスの実行後に読み込まれるか、再度読み込まれます。
この場合、再起動後に一部の **sysctl.conf** パラメーターは使用されていない可能性があります。これらの設定は利用できないか、デフォルトに戻る可能性があります。

この問題の解決方法については、Red Hat ナレッジベースの記事 Some sysctl.conf parameters are not used after reboot, manually adjusting the settings works as expected を参照してください。

20.4. 設定ファイルの指定

コマンドラインオプションやコマンドラインで設定できないその他のオプションは、オプションの設定ファイルで設定することができます。

デフォルトでは、設定ファイルは読み取られないため、**-f** を指定してランタイム時に指定する必要があります。以下に例を示します。

```
~]# ptp4l -f /etc/ptp4l.conf
```

上記の **-i eth3 -m -S** オプションと同等の設定ファイルは、以下のようになります。

```
~]# cat /etc/ptp4l.conf
[global]
verbose         1
time_stamping     software
[eth3]
```

20.5. PTP 管理クライアントの使用

PTP 管理クライアントである pmc を使うと、以下のように ptp4l から追加情報を取得することができます。

```
~]# pmc -u -b 0 'GET CURRENT_DATA_SET'
sending: GET CURRENT_DATA_SET
   90e2ba.fffe.20c7f8-0 seq 0 RESPONSE MANAGMENT CURRENT_DATA_SET
      stepsRemoved    1
      offsetFromMaster -142.0
      meanPathDelay   9310.0
```

```
~]# pmc -u -b 0 'GET TIME_STATUS_NP'
sending: GET TIME_STATUS_NP
   90e2ba.fffe.20c7f8-0 seq 0 RESPONSE MANAGMENT TIME_STATUS_NP
      master_offset       310
      ingress_time        1361545089345029441
      cumulativeScaledRateOffset  +1.000000000
      scaledLastGmPhaseChange  0
      gmTimeBaseIndicator    0
```

```
lastGmPhaseChange     0x0000'0000000000000000.0000
gmPresent             true
gmIdentity            00a069.fffe.0b552d
```

-b オプションを **zero** に設定すると、ローカルで実行している **ptp4l** インスタンスへの境界を制限されます。大きな境界値が大きいと、ローカルクロックからさらに **PTP** ノードから情報を取得します。取得できる情報は次のとおりです。

- **stepsRemoved** は、グランドマスタークロックへの通信パスの数です。
- **offsetFromMaster** および **master_offset** は、マスターから最後に測定されたオフセットをナノ秒で表します。
- **meanPathDelay** は、マスターから送信された同期メッセージの遅延予測をナノ秒で表します。
- **gmPresent** が **true** の場合、**PTP** クロックがマスターに同期され、ローカルクロックはグランドマスタークロックに同期されません。
- **gmIdentity** は、グランドマスターの **ID** です。

pmc コマンドの完全なリストを表示するには、**root** で以下を入力します。

```
~]# pmc help
```

pmc(8) man ページでは詳細情報が見られます。

20.6. クロックの同期

phc2sys プログラムは、システムクロックを **NIC** 上の **PHC** ハードウェアクロック **(PTP)** と同期するために使用されます。**phc2sys** サービスは **/etc/sysconfig/phc2sys** 設定ファイルで設定されます。**/etc/sysconfig/phc2sys** ファイルのデフォルト設定は以下のようになります。

```
OPTIONS="-a -r"
```

-a オプションを使用すると、**phc2sys** が、**ptp4l** アプリケーションから同期されるクロックを読み取るようになります。これは、**PTP** ポートの状態の変更に従い、**NIC** ハードウェアクロック間の同期を適宜調整します。**-r** オプションも指定されていない限り、システムクロックは同期されません。システムクロックをタイムソースにするには、**-r** オプションを 2 回指定します。

/etc/sysconfig/phc2sys への変更後に、**root** でコマンドを実行し、コマンドラインから **phc2sys** サービスを再起動します。

```
~]# systemctl restart phc2sys
```

通常の状態では、**systemctl** コマンドを使用して、**phc2sys** サービスを起動し、停止し、再起動します。

phc2sys をサービスとして起動するには、コマンドラインからこれを起動できます。たとえば、**root** で以下のコマンドを入力します。

```
~]# phc2sys -a -r
```

-a オプションを使用すると、**phc2sys** が、**ptp4l** アプリケーションから同期されるクロックを読み取るようになります。システムクロックをタイムソースにするには、**-r** オプションを 2 回指定します。

*または、**-s** オプションを指定して、システムクロックを特定のインターフェイスの**PTP** ハードウェアクロックに同期します。以下に例を示します。*

```
~]# phc2sys -s eth3 -w
```

***-w** オプションは、実行中のptp4l アプリケーションが**PTP** クロックを同期するまで待機し、ptp4l から UTC オフセットへのTAI を取得します。*

*通常、**PTP** は**国際原子時** (TAI) のタイムスケールで作動し、システムクロックは**協定世界時** (UTC) で維持されます。TAI と UTC のタイムスケール間の現在のオフセットは、36 秒です。このオフセットは、うるう秒が追加もしくは取り除かれると変化します。次のように、**-w** オプションを使用しない場合は、**-O** オプションを使用してこのオフセットを手動で設定する必要があります。*

```
~]# phc2sys -s eth3 -O -36
```

*phc2sys servo がロックされた状態になると、**-S** オプションを指定しない限り、クロックはステップされません。つまり、phc2sys プログラムは、ptp4l プログラムが**PTP** ハードウェアクロックを同期した後に起動すべきということになります。ただし、**-w** を使用すると、ptp4l の後にphc2sys を起動する必要はありません。これは、クロックの同期を待つためです。*

phc2sys プログラムは、以下のコマンドを実行してサービスとしても起動できます。

```
~]# systemctl start phc2sys
```

*サービスとして実行する場合は、オプションは **/etc/sysconfig/phc2sys** ファイルで指定されます。phc2sys の他のオプションについての詳細情報は、**phc2sys(8)** man ページを参照してください。*

本セクションの例では、コマンドがスレーブシステムまたはスレーブポートで実行されている想定であることに注意してください。

20.7. 時間同期の検証

***PTP** の時間同期が正常に機能していると、オフセットと頻度調整を含む新しいメッセージが、ptp4l と、ハードウェアタイムスタンプを使用している場合はphc2sys の、各出力に定期的に印刷されます。出力値はすぐに収束します。これらのメッセージは **/var/log/messages** ファイルで見ることができます。*

以下のptp4l およびphc2sys の出力例には次の内容が含まれています。

- *オフセット (ナノ秒)*
- *頻度のオフセット (10 億分の 1 (ppb))*
- *経路遅延 (ナノ秒)*

ptp4l の出力例:

```
ptp4l[352.359]: selected /dev/ptp0 as PTP clock
ptp4l[352.361]: port 1: INITIALIZING to LISTENING on INITIALIZE
ptp4l[352.361]: port 0: INITIALIZING to LISTENING on INITIALIZE
ptp4l[353.210]: port 1: new foreign master 00a069.fffe.0b552d-1
ptp4l[357.214]: selected best master clock 00a069.fffe.0b552d
ptp4l[357.214]: port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[359.224]: master offset    3304 s0 freq   +0 path delay   9202
ptp4l[360.224]: master offset    3708 s1 freq -29492 path delay   9202
```

```
ptp4l[361.224]: master offset   -3145 s2 freq -32637 path delay   9202
ptp4l[361.224]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[362.223]: master offset    -145 s2 freq -30580 path delay   9202
ptp4l[363.223]: master offset    1043 s2 freq -29436 path delay   8972
ptp4l[364.223]: master offset    266 s2 freq -29900 path delay   9153
ptp4l[365.223]: master offset    430 s2 freq -29656 path delay   9153
ptp4l[366.223]: master offset    615 s2 freq -29342 path delay   9169
ptp4l[367.222]: master offset    -191 s2 freq -29964 path delay   9169
ptp4l[368.223]: master offset    466 s2 freq -29364 path delay   9170
ptp4l[369.235]: master offset     24 s2 freq -29666 path delay   9196
ptp4l[370.235]: master offset    -375 s2 freq -30058 path delay   9238
ptp4l[371.235]: master offset    285 s2 freq -29511 path delay   9199
ptp4l[372.235]: master offset    -78 s2 freq -29788 path delay   9204
```

phc2sys の出力例:

```
phc2sys[526.527]: Waiting for ptp4l...
phc2sys[527.528]: Waiting for ptp4l...
phc2sys[528.528]: phc offset   55341 s0 freq   +0 delay  2729
phc2sys[529.528]: phc offset   54658 s1 freq -37690 delay  2725
phc2sys[530.528]: phc offset    888 s2 freq -36802 delay  2756
phc2sys[531.528]: phc offset   1156 s2 freq -36268 delay  2766
phc2sys[532.528]: phc offset    411 s2 freq -36666 delay  2738
phc2sys[533.528]: phc offset    -73 s2 freq -37026 delay  2764
phc2sys[534.528]: phc offset    39 s2 freq -36936 delay  2746
phc2sys[535.529]: phc offset    95 s2 freq -36869 delay  2733
phc2sys[536.529]: phc offset   -359 s2 freq -37294 delay  2738
phc2sys[537.529]: phc offset   -257 s2 freq -37300 delay  2753
phc2sys[538.529]: phc offset    119 s2 freq -37001 delay  2745
phc2sys[539.529]: phc offset    288 s2 freq -36796 delay  2766
phc2sys[540.529]: phc offset   -149 s2 freq -37147 delay  2760
phc2sys[541.529]: phc offset   -352 s2 freq -37395 delay  2771
phc2sys[542.529]: phc offset    166 s2 freq -36982 delay  2748
phc2sys[543.529]: phc offset    50 s2 freq -37048 delay  2756
phc2sys[544.530]: phc offset    -31 s2 freq -37114 delay  2748
phc2sys[545.530]: phc offset   -333 s2 freq -37426 delay  2747
phc2sys[546.530]: phc offset    194 s2 freq -36999 delay  2749
```

*ptp4l の出力を減らして値のみを出力するには、**summary_interval** ディレクティブを使用します。**summary_interval** ディレクティブは、秒において2 のn 乗として指定されています。たとえば、出力を **1024** 秒ごとに減らすには、以下の行を **/etc/ptp4l.conf** ファイルに追加します。*

```
summary_interval 10
```

***summary_interval** を 6 に設定したptp4l 出力の例:*

```
ptp4l: [615.253] selected /dev/ptp0 as PTP clock
ptp4l: [615.255] port 1: INITIALIZING to LISTENING on INITIALIZE
ptp4l: [615.255] port 0: INITIALIZING to LISTENING on INITIALIZE
ptp4l: [615.564] port 1: new foreign master 00a069.fffe.0b552d-1
ptp4l: [619.574] selected best master clock 00a069.fffe.0b552d
ptp4l: [619.574] port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l: [623.573] port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l: [684.649] rms 669 max 3691 freq -29383 ± 3735 delay 9232 ± 122
ptp4l: [748.724] rms 253 max 588 freq -29787 ± 221 delay 9219 ± 158
```

```
ptp4l: [812.793] rms 287 max 673 freq -29802 ± 248 delay 9211 ± 183
ptp4l: [876.853] rms 226 max 534 freq -29795 ± 197 delay 9221 ± 138
ptp4l: [940.925] rms 250 max 562 freq -29801 ± 218 delay 9199 ± 148
ptp4l: [1004.988] rms 226 max 525 freq -29802 ± 196 delay 9228 ± 143
ptp4l: [1069.065] rms 300 max 646 freq -29802 ± 259 delay 9214 ± 176
ptp4l: [1133.125] rms 226 max 505 freq -29792 ± 197 delay 9225 ± 159
ptp4l: [1197.185] rms 244 max 688 freq -29790 ± 211 delay 9201 ± 162
```

デフォルトでは **summary_interval** は 0 に設定されるため、メッセージは最大頻度である 1 秒ごとに出力されます。したがってメッセージは 1 秒ごとに印刷されます。メッセージを無効にするには、**-l** オプションを使用して最大ログレベルを 5 以下に設定します。

```
~]# phc2sys -l 5
```

-u オプションを指定すると、phc2sys 出力を減らすことができます。

```
~]# phc2sys -u summary-updates
```

ここでの summary-updates は、統計情報に含めるクロック更新数です。例を示します。

```
~]# phc2sys -s eth3 -w -m -u 60
phc2sys[700.948]: rms 1837 max 10123 freq -36474 ± 4752 delay 2752 ± 16
phc2sys[760.954]: rms 194 max 457 freq -37084 ± 174 delay 2753 ± 12
phc2sys[820.963]: rms 211 max 487 freq -37085 ± 185 delay 2750 ± 19
phc2sys[880.968]: rms 183 max 440 freq -37102 ± 164 delay 2734 ± 91
phc2sys[940.973]: rms 244 max 584 freq -37095 ± 216 delay 2748 ± 16
phc2sys[1000.979]: rms 220 max 573 freq -36666 ± 182 delay 2747 ± 43
phc2sys[1060.984]: rms 266 max 675 freq -36759 ± 234 delay 2753 ± 17
```

このオプションで使用すると、統計の更新の間隔は 60 秒 (**-u**) に設定され、**phc2sys** は、ptp4l が同期状態 (**-w**) になるまで待機します。また、メッセージは標準出力 (**-m**) に出力されます。**phc2sys** オプションの詳細は、**phc2sys(5)** man ページを参照してください。

出力には以下が含まれます。

- オフセット二乗平均平方根 (rms)
- 最大絶対オフセット (max)
- 頻度オフセット (freq): 平均および標準偏差
- 経過遅延 (delay): 平均および標準偏差

20.8. NTP を使用した PTP 時間の実行

ntpd デーモンは、ptp4l または phc2sys で同期されたシステムクロック、または LOCAL 参照クロックドライバーを使用して同期されたシステムクロックの時間を配布するように設定可能です。**ntpd** がシステムクロックを調整することを防ぐには、**ntp.conf** ファイルで **NTP** サーバーを指定しないようにします。以下に、**ntp.conf** の最小限の例を示します。

```
~]# cat /etc/ntp.conf
server  127.127.1.0
fudge  127.127.1.0 stratum 0
```

注記

DHCP クライアントプログラムである dhclient は、DHCP サーバーから NTP サーバーのリストを受信すると、これに ntp.conf を追加してサービスを再起動します。この機能を無効にするには、PEERNTP=no を /etc/sysconfig/network に追加します。

20.9. PTP を使用した NTP 時間の実行

逆方向の NTP から PTP への同期も可能です。ntpd を使用してシステムクロックを同期する場合、ptp4l は priority1 オプション (または、最適なマスタークロックアルゴリズムに含まれるその他のクロックオプション) でグランドマスタークロックとして設定し、システムクロックから PTP を介して時間を分配できます。

```
~]# cat /etc/ptp4l.conf
[global]
priority1 127
[eth3]
# ptp4l -f /etc/ptp4l.conf
```

ハードウェアタイムスタンプでは、phc2sys を使用して PTP ハードウェアクロックをシステムクロックに同期する必要があります。phc2sys をサービスとして実行している場合は、/etc/sysconfig/phc2sys 設定ファイルを編集します。/etc/sysconfig/phc2sys ファイルのデフォルト設定は以下のようになります。

```
OPTIONS="-a -r"
```

root として、その行を次のように編集します。

```
~]# vi /etc/sysconfig/phc2sys
OPTIONS="-a -r -r"
```

-r オプションは 2 回使用され、NIC 上の PTP ハードウェアクロックのシステムクロックとの同期を許可しています。変更を有効にするには、phc2sys サービスを再起動します。

```
~]# systemctl restart phc2sys
```

PTP クロックの周波数の迅速な変更を防ぐため、システムクロックへの同期は、PI servo に小規模な P (比例) および I (積分) 定数を使用して緩めることができます。

```
~]# phc2sys -a -r -r -P 0.01 -I 0.0001
```

20.10. TIMEMASTER を使用した PTP または NTP 時間への同期

複数の PTP ドメインがネットワーク上で利用できるか、NTP へのフォールバックが必要な場合、timemaster プログラムを使用して、利用可能なすべてのタイムソースにシステムクロックを同期できます。PTP 時間は、共有メモリードライバー (chronyd または ntpd への SHM のリファレンスクロック) から phc2sys および ptp4l で提供されます (システム上で設定されている NTP デーモンにより異なります)。NTP デーモンは次に、すべてのタイムソース、PTP および NTP の両方を比較でき、システムクロックを同期させるのに最善のソースを使用できます。

開始時に、timemaster は NTP および PTP タイムソースを指定する設定ファイルを読み取り、独自のクロックを持つか、PTP ハードウェアクロック (PHC) を共有するネットワークインターフェイスを確認し、ptp4l および chronyd または ntpd の設定ファイルを生成し、必要に応じて ptp4l、phc2sys、およ

び **chronyd** または **ntpd** プロセスを起動します。終了時には生成された設定ファイルは削除されます。**chronyd**、**ntpd**、ptp4l の設定ファイルは **/var/run/timemaster/** に書き込まれます。

20.10.1. timemaster をサービスとして起動

timemaster をサービスとして起動するには、**root** で以下のコマンドを発行します。

```
~]# systemctl start timemaster
```

これにより、 **/etc/timemaster.conf** のオプションが読み取られます。Red Hat Enterprise Linux 7 でのシステムサービスの管理の詳細については、10章systemd によるサービス管理 を参照してください。

20.10.2. timemaster 設定ファイルの概要

Red Hat Enterprise Linux のデフォルトの **/etc/timemaster.conf** ファイルには、デフォルトオプションを含む数多くのセクションがあります。このセクションの見出しは括弧で囲まれます。

デフォルト設定を表示するには、以下のコマンドを実行します。

```
~]$ less /etc/timemaster.conf
# Configuration file for timemaster

#[ntp_server ntp-server.local]
#minpoll 4
#maxpoll 4

#[ptp_domain 0]
#interfaces eth0

[timemaster]
ntp_program chronyd

[chrony.conf]
include /etc/chrony.conf

[ntp.conf]
includefile /etc/ntp.conf

[ptp4l.conf]

[chronyd]
path /usr/sbin/chronyd
options -u chrony

[ntpd]
path /usr/sbin/ntpd
options -u ntp:ntp -g

[phc2sys]
path /usr/sbin/phc2sys

[ptp4l]
path /usr/sbin/ptp4l
```

次の名前のセクションに注意してください。

[ntp_server address]

これは、NTP サーバーセクションの例 ntp-server.local は、ローカル LAN 上のNTP サーバーのホスト名の例です。セクション名の一部としてホスト名または IP アドレスを使用し、必要に応じてセクションを追加してください。この例の短いポーリング値はパブリックサーバーに適していません。適切な minpoll 値および maxpoll 値の説明は、19章ntpd を使用した NTP 設定を参照してください。

次の名前のセクションに注意してください。

[ptp_domain number]

PTP ドメインは、互いに同期する 1 つ以上の PTP クロックのグループです。それらは別のドメインでクロックに同期される場合もありますが、同期されない場合もあります。同じドメイン番号で設定されているクロックがドメインを設定します。これには、PTP グランドマスタークロックが含まれます。各 PTP domain セクションのドメイン番号は、ネットワーク上に設定される PTP ドメインのいずれかに対応している必要があります。

ptp4l のインスタンスは、独自の PTP クロックを持つすべてのインスタンスで起動し、ハードウェアのタイムスタンプは自動的に有効にされます。ハードウェアのタイムスタンプをサポートするインターフェイスには PTP クロック (PHC) が割り当てられます。同じ PHC を共有するインターフェイスグループごと、ソフトウェアのタイムスタンプのみに対応した各インターフェイスに対して、別のptp4l インスタンスが開始されます。すべてのptp4l インスタンスは、スレーブとして実行するように設定されています。ハードウェアのタイムスタンプが設定されたインターフェイスが 1 つ以上の PTP ドメインに指定される場合、作成される最初のptp4l インスタンスのみで、ハードウェアのタイムスタンプが有効にされます。

次の名前のセクションに注意してください。

[timemaster]

デフォルトの timemaster 設定には、アクセス制限および認証キーの設定を組み込むためにシステムの ntpd および chrony 設定 (/etc/ntp.conf または /etc/chronyd.conf) が含まれます。つまり、そこに指定されるすべての NTP サーバーが timemaster で使用されることを意味しています。

セクションの見出しは以下のようになります。

- ***[ntp_server ntp-server.local]**: このサーバーのポーリング間隔を指定します。必要に応じて追加のセクションを作成します。セクション見出しのホスト名または IP アドレスを含めます。*
- ***[ptp_domain 0]**: このドメインに PTP クロックを設定するインターフェイスを指定します。必要に応じて、適切なドメイン番号で追加のセクションを作成します。*
- ***[timemaster]**: NTP デーモンが使用されるように指定します。使用できる値は chronyd および ntpd です。*
- ***[chrony.conf]**: chronyd に生成される設定ファイルにコピーされる追加設定を指定します。*
- ***[ntp.conf]**: ntpd に生成される設定ファイルにコピーされる追加設定を指定します。*
- ***[ptp4l.conf]**: ptp4l 用に生成された設定ファイルにコピーされるオプションを指定します。*
- ***[chronyd]**: chronyd に対してコマンドラインで渡される追加設定を指定します。*
- ***[ntpd]**: ntpd に対してコマンドラインで渡される追加設定を指定します。*
- ***[phc2sys]**: phc2sys に対してコマンドラインで渡される追加設定を指定します。*

- **[ptp4l]**: ptp4l のすべてのインスタンスに対してコマンドラインで渡される追加設定を指定します。

セクション見出しおよびその内容の詳細は、**timemaster(8)** man ページで説明されています。

20.10.3. timemaster オプションの設定

timemaster 設定ファイルの編集

1. デフォルト設定を変更するには、**root** で編集するために **/etc/timemaster.conf** ファイルを開きます。

```
~]# vi /etc/timemaster.conf
```

2. timemaster を使用して制御する必要のあるそれぞれの **NTP** サーバーについて、**[ntp_server address]** セクションを作成します。この例の短いポーリング値はパブリックサーバーに適していません。適切な **minpoll** と **maxpoll** 値の説明は、19章ntpd を使用した NTP 設定を参照してください。

3. ドメインで使用するインターフェイスを追加するには、**#[ptp_domain 0]** セクションを編集してインターフェイスを追加します。必要に応じて追加のドメインを作成します。以下に例を示します。

```
[ptp_domain 0]
   interfaces eth0

   [ptp_domain 1]
   interfaces eth1
```

4. このシステムの **NTP** デーモンとして **ntpd** を使用することが必要な場合、**[timemaster]** セクションのデフォルトエントリーを、**chronyd** から **ntpd** に変更します。ntpd と chronyd の違いについての情報は、18章chrony スイートを使用した NTP 設定を参照してください。

5. このシステムで **chronyd** を **NTP** サーバーとして使用する場合は、**[chrony.conf]** のデフォルトの **include /etc/chrony.conf** エントリーの下に追加オプションを含めます。**/etc/chrony.conf** へのパスが変更された場合は、デフォルトの **include** エントリーを編集します。

6. このシステムで **ntpd** を **NTP** サーバーとして使用する場合は、**[ntp.conf]** のデフォルトの **include /etc/ntp.conf** エントリーの下に追加オプションを含めます。**/etc/ntp.conf** へのパスが変更された場合は、デフォルトの **include** エントリーを編集します。

7. **[ptp4l.conf]** セクションに、ptp4l に生成される設定ファイルにコピーされるオプションを追加します。この章では、共通オプションについて説明します。詳細は、**ptp4l(8)** man ページを参照してください。

8. **[chronyd]** セクションに、timemaster で呼び出される場合に **chronyd** に渡されるコマンドラインのオプションを追加します。**chronyd** の使用についての情報は、18章chrony スイートを使用した NTP 設定 を参照してください。

9. **[ntpd]** セクションに、timemaster で呼び出される場合に **ntpd** に渡されるコマンドラインのオプションを追加します。**ntpd** の使用についての情報は、19章ntpd を使用した NTP 設定を参照してください。

10. **[phc2sys]** セクションで、**timemaster** により呼び出される場合に**phc2sys** に渡すコマンドラインオプションを追加します。この章では、共通オプションについて説明します。詳細は、**phy2sys(8)** man ページを参照してください。

11. **[ptp4l]** セクションに、**timemaster** で呼び出される場合に**ptp4l** に渡されるコマンドラインオプションを追加します。この章では、共通オプションについて説明します。詳細は、**ptp4l(8)** man ページを参照してください。

12. 設定ファイルを保存し、**timemaster** を再起動するには、**root** で以下のコマンドを実行します。

```
~]# systemctl restart timemaster
```

20.11. 精度の向上

これまでテストの結果では、ティックレスカーネル機能を無効にするとシステムクロックの安定性が大幅に改善され、**PTP** 同期の精度が高まることが示されています (電力消費量は高まります)。カーネル起動オプションパラメーターに **nohz=off** を追加することで、カーネルティックモードを無効にすることができます。ただし、**kernel-3.10.0-197.el7** に適用される最近の改善により、システムクロックの安定性が大幅に改善され、**nohz=off** のあるクロックの安定性の相違点が、ほとんどのユーザーに対して大幅に小さくなるはずです。

ptp4l および**phc2sys** アプリケーションは、新しい適応サービスを使用するように設定できます。**PI** サーボに対するこれの利点は、適正に機能させるために **PI** 定数を設定する必要がない点にあります。これを**ptp4l** に使用するには、以下の行を **/etc/ptp4l.conf** ファイルに追加します。

```
clock_servo linreg
```

/etc/ptp4l.conf に変更を加えたら、**root** で以下のコマンドを実行して、コマンドラインから**ptp4l** サービスを再起動します。

```
~]# systemctl restart ptp4l
```

これを**phc2sys** に利用するには、次の行を **/ etc / sysconfig / phc2sys** ファイルに追加します。

```
-E linreg
```

/etc/sysconfig/phc2sys への変更後に、**root** で以下のコマンドを実行し、コマンドラインから**phc2sys** サービスを再起動します。

```
~]# systemctl restart phc2sys
```

20.12. 関連情報

以下の情報ソースで **PTP** および**ptp4l** ツールに関する追加リソースが提供されています。

20.12.1. インストールされているドキュメント

- **ptp4l(8)** の man ページ: 設定ファイルの形式を含む**ptp4l** オプションを説明しています。
- **pmc(8)** の man ページ: **PTP** 管理クライアントおよびそのコマンドオプションを説明しています。

- ***phc2sys(8)*** *man ページ: システムクロックを* ***PTP*** *ハードウェアクロック (PHC) に同期させるツールを説明しています。*
- ***timemaster(8)*** *の man ページ:* ***chronyd*** *または* ***ntpd*** *を使用してシステムクロックを同期するために* ***ptp4l*** *および* ***phc2sys*** *を使用するプログラムについて説明しています。*

20.12.2. 便利な Web サイト

http://www.nist.gov/el/isd/ieee/ieee1588.cfm

IEEE 1588 規格

パート VI. 監視と自動化

ここでは、システム管理者がシステムパフォーマンスのモニター、システムタスクの自動化、およびバグの報告を行うための各種ツールを説明します。

第21章 システムモニタリングツール

システムを設定するには、多くの場合システム管理者は空きメモリーの容量、空きディスク領域、ハードディスクのパーティション設定状況、実行中のプロセスを決定する必要があります。

21.1. システムプロセスの表示

21.1.1. ps コマンドの使用

ps コマンドは、実行中のプロセスについての情報を表示します。静的なリスト、すなわちコマンドを実行するときに実行しているプロセスのスナップショットです。実行中のプロセスを定期的に更新したリストを表示させるには、**top** コマンドまたはSystem Monitor アプリケーションを代わりに使用します。

他のユーザーが所有しているプロセスを含め、現在システム上で実行中の全プロセスをリスト表示するには、シェルプロンプトで以下を入力します。

```
ps ax
```

リストのプロセスごとに、**ps ax** コマンドが、プロセス ID **(PID)**、関連しているターミナル **(TTY)**、現在のステータス **(STAT)**、累積された CPU 時間 **(TIME)** および実行可能ファイルの名前 **(COMMAND)** を示します。以下に例を示します。

```
~]$ ps ax
PID TTY   STAT  TIME COMMAND
 1 ?    Ss   0:01 /usr/lib/systemd/systemd --switched-root --system --deserialize 23
 2 ?    S   0:00 [kthreadd]
 3 ?    S   0:00 [ksoftirqd/0]
 5 ?    S>   0:00 [kworker/0:0H]
[output truncated]
```

各プロセスと同時に所有者も表示するには、以下のコマンドを使用します。

```
ps aux
```

ps ax コマンドで提供される情報以外に、**ps aux** はプロセス所有者の有効なユーザー名 **(USER)**、CPU のパーセンテージ **(%CPU)** およびメモリー使用率 **(%MEM)**、キロバイト単位での仮想メモリーサイズ **(VSZ)**、キロバイト単位での非スワップの物理メモリーサイズ **(RSS)**、プロセスの開始日時を表示します。以下に例を示します。

```
~]$ ps aux
USER PID %CPU %MEM  VSZ  RSS TTY  STAT START  TIME COMMAND
root  1 0.3 0.3 134776 6840 ?   Ss  09:28  0:01 /usr/lib/systemd/systemd --switched-root --
system --d
root  2 0.0 0.0   0   0 ?   S  09:28  0:00 [kthreadd]
root  3 0.0 0.0   0   0 ?   S  09:28  0:00 [ksoftirqd/0]
root  5 0.0 0.0   0   0 ?   S>  09:28  0:00 [kworker/0:0H]
[output truncated]
```

ps コマンドを **grep** と組み合わせて使用して、特定のプロセスが実行中かどうかを確認することもできます。たとえば、Emacs が実行中かどうかを知るには、以下を入力します。

```
~]$ ps ax | grep emacs
12056 pts/3  S+   0:00 emacs
12060 pts/2  S+   0:00 grep --color=auto emacs
```

利用可能なコマンドラインオプションのリストは、ps(1) の man ページを参照してください。

21.1.2. top コマンドの使用

top コマンドは、システムで実行しているプロセスのリアルタイムのリストを表示します。また、システムのアップタイム、現在の CPU およびメモリー使用率、実行中のプロセスの合計数についての追加情報も表示します。さらには、リストの並び替えやプロセスの kill などの操作も実行できます。

top コマンドを実行するには、シェルプロンプトで以下を入力します。

```
top
```

リスト表示された各プロセスについて **top** コマンドはプロセス ID (**PID**)、プロセス所有者の実効ユーザー名、(**USER**)、優先度 (**PR**)、nice 値 (**NI**)、プロセスが使用する仮想メモリー容量 (**VIRT**)、プロセスが使用する非スワップ物理メモリー容量 (**RES**)、プロセスが使用する共有メモリー容量 (**SHR**)、プロセスステータスフィールド (**S**)、CPU 使用率 (**%CPU**) およびメモリー使用率 (**%MEM**)、累積 CPU 時間 (**TIME+**)、実行ファイル名 (**COMMAND**) を表示します。以下に例を示します。

```
~]$ top
top - 16:42:12 up 13 min, 2 users, load average: 0.67, 0.31, 0.19
Tasks: 165 total,  2 running, 163 sleeping,  0 stopped,  0 zombie
%Cpu(s): 37.5 us, 3.0 sy, 0.0 ni, 59.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 1016800 total,  77368 free,  728936 used,  210496 buff/cache
KiB Swap:  839676 total,  776796 free,  62880 used.  122628 avail Mem

 PID USER   PR NI  VIRT  RES  SHR S %CPU %MEM   TIME+ COMMAND
3168 sjw    20  0 1454628 143240 15016 S 20.3 14.1  0:22.53 gnome-shell
4006 sjw    20  0 1367832 298876 27856 S 13.0 29.4  0:15.58 firefox
1683 root   20  0 242204 50464  4268 S 6.0 5.0  0:07.76 Xorg
4125 sjw    20  0 555148 19820 12644 S 1.3 1.9  0:00.48 gnome-terminal-
 10 root   20  0    0   0   0 S 0.3 0.0  0:00.39 rcu_sched
3091 sjw    20  0  37000  1468  904 S 0.3 0.1  0:00.31 dbus-daemon
3096 sjw    20  0 129688  2164  1492 S 0.3 0.2  0:00.14 at-spi2-registr
3925 root   20  0    0   0   0 S 0.3 0.0  0:00.05 kworker/0:0
 1 root   20  0 126568  3884  1052 S 0.0 0.4  0:01.61 systemd
 2 root   20  0    0   0   0 S 0.0 0.0  0:00.00 kthreadd
 3 root   20  0    0   0   0 S 0.0 0.0  0:00.00 ksoftirqd/0
 6 root   20  0    0   0   0 S 0.0 0.0  0:00.07 kworker/u2:0
[output truncated]
```

表21.1「インタラクティブな top コマンド」には、**top** で使用できる便利な対話型のコマンドが含まれています。詳細は、top(1) の man ページを参照してください。

表21.1 インタラクティブな top コマンド

コマンド	詳細
Enter、**Space**	表示を最新の情報に直ちに更新します。

コマンド	詳細
h	インタラクティブコマンドのヘルプ画面を表示します。
h, **?**	ウィンドウおよびフィールドグループのヘルプ画面を表示します。
k	プロセスを強制終了します。プロセス ID およびプロセスに送信するシグナルがプロンプトされます。
-n	表示されるプロセス番号を変更します。番号を入力するようプロンプトされます。
u	リストをユーザー別に並べ替えます。
M	リストをメモリー使用率で並べ替えます。
%P	リストを CPU 使用率で並べ替えます。
q	ユーティリティーを終了して、シェルプロンプトに戻ります。

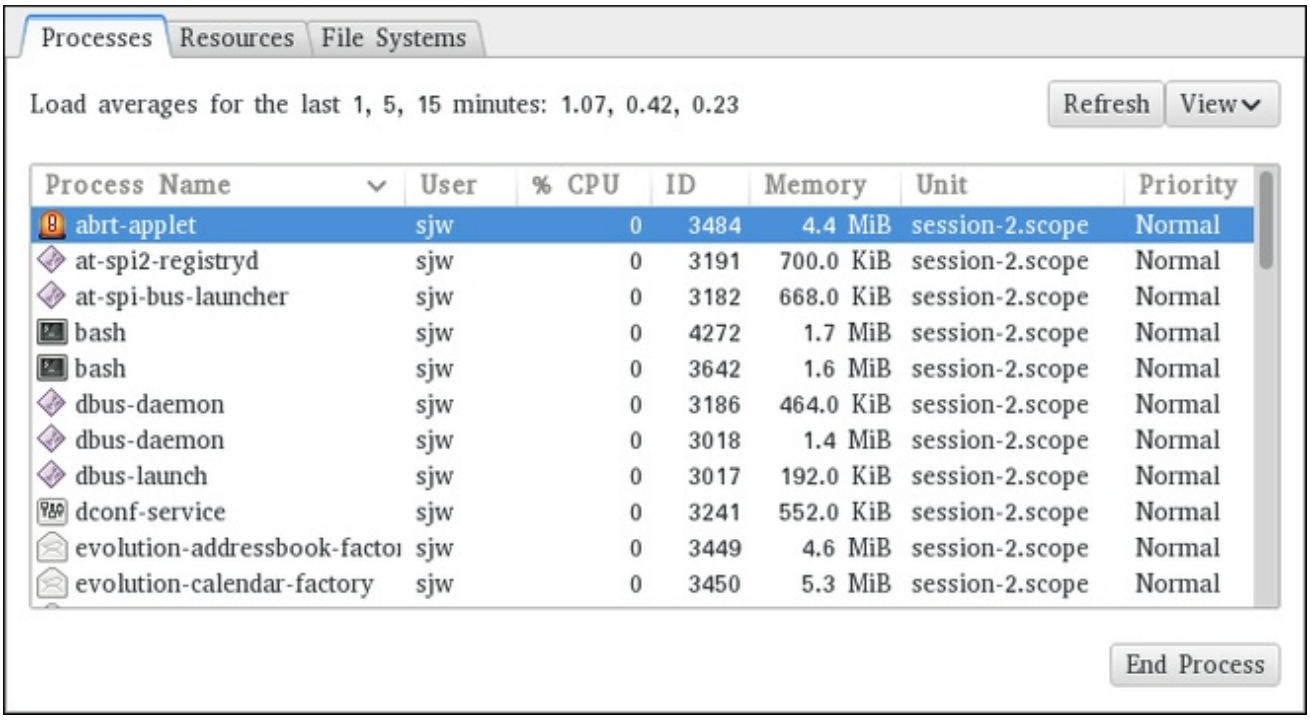
21.1.3. システムモニターツールの使用

*System Monitor ツールの **Processes** タブを使用することで、グラフィカルユーザーインターフェイスからプロセスの表示、検索、優先度の変更、**kill** を行うことができます。*

*コマンドラインから System Monitor ツールを起動するには、シェルプロンプトで **gnome-system-monitor** と入力します。この結果、System Monitor ツールが表示されます。また、GNOME デスクトップで **Super** キーを押してアクティビティーの概要を入力する場合は、**System Monitor** と入力し、**Enter** を押します。この結果、System Monitor ツールが表示されます。**Super** キーはキーボードまたはその他のハードウェアに応じて様々なキーで表示されますが、多くの場合、Windows または Command キーとして通常は **Spacebar** の左側に表示されます。*

***Processes** (プロセス) タブをクリックして実行中プロセスのリストを表示します。*

図21.1 システムモニター – プロセス



リストのプロセスごとに、System Monitor ツールは 名前 (**Process Name**)、現在のステータス (**Status**)、CPU 使用率のパーセンテージ (**% CPU**)、nice 値 (**Nice**)、プロセス ID (**ID**)、メモリー使用量 (**Memory**)、プロセスが待機中のチャネル (**Waiting Channel**)、およびセッション (**Session**) に関する追加情報を示します。特定の列で昇順で情報を並べ替えるには、その列の名前をクリックします。特定のコラム別に情報を昇順で並び替えるには、コラム名をクリックします。

デフォルトでは、System Monitor ツールは現在のユーザーが所有しているプロセスのリストを表示します。表示メニューから各種オプションを選択すると、以下を実行できます。

- 実行中のプロセスのみの表示
- すべてのプロセスの表示
- ユーザーのプロセスの表示
- プロセスの依存関係の表示

また、2 つのボタンを使用して以下のことを行えます。

- プロセスのリストを更新する
- リストからプロセスを選択し、**End Process** (プロセスの終了) ボタンをクリックすることによりプロセスを終了する

21.2. メモリー使用量の表示

21.2.1. free コマンドの使用

free コマンドは、システムのメモリーの空き容量と使用量を表示します。シェルプロンプトで以下を入力してください。

```
free
```

free コマンドは、物理メモリー (**Mem**) およびスワップ領域 (**Swap**) に関する情報を提供します。メモリーの合計量 (**total**) だけでなく、使用メモリー容量 (**used**)、空きメモリー容量 (**free**)、共有メモリー容量 (**shared**)、バッファーおよびキャッシュメモリー容量の合計 (**buff/cache**)、利用可能なメモリー容量 (**available**) を表示します。以下に例を示します。

```
~]$ free
      total    used    free   shared buff/cache  available
Mem:   1016800   727300    84684    3500   204816   124068
Swap:   839676    66920   772756
```

デフォルトでは、**free** は値をキロバイトで表示します。メガバイトで値を表示するには、**-m** コマンドラインオプションを指定します。

```
free -m
```

たとえば、以下のようになります。

```
~]$ free -m
      total    used    free   shared buff/cache  available
Mem:     992     711     81      3     200     120
Swap:     819     65     754
```

利用可能なコマンドラインオプションのリストは、free(1) の man ページを参照してください。

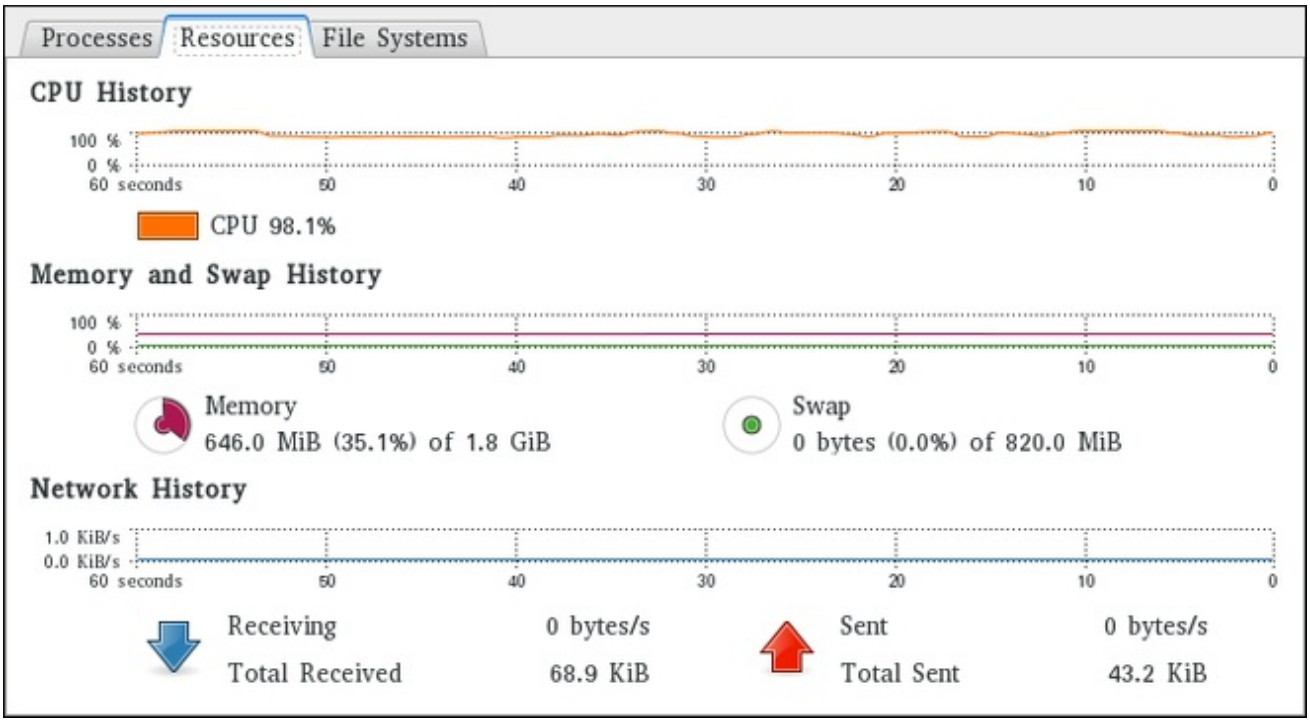
21.2.2. システムモニターツールの使用

System Monitor ツールの **Resources** タブは、システムのメモリーの空き容量と使用量を表示します。

コマンドラインから System Monitor ツールを起動するには、シェルプロンプトで **gnome-system-monitor** と入力します。この結果、System Monitor ツールが表示されます。また、GNOME デスクトップで **Super** キーを押してアクティビティーの概要を入力する場合は、**System Monitor** と入力し、**Enter** を押します。この結果、System Monitor ツールが表示されます。**Super** キーはキーボードまたはその他のハードウェアに応じて様々なキーで表示されますが、多くの場合、Windows または Command キーとして通常は **Spacebar** の左側に表示されます。

Resources (リソース) タブをクリックしてシステムのメモリー使用率を表示します。

図21.2 システムモニター – リソース



Memory and Swap History セクションでは、System Monitor ツールがメモリーとスワップの使用量履歴をグラフィック表示します。また、物理メモリー (**Memory**) およびスワップ領域 (**Swap**) の合計量および使用中の量が表示されます。

21.3. CPU 使用率の表示

21.3.1. システムモニターツールの使用

System Monitor ツールの **Resources** タブは、システムの現在の CPU 使用率を表示します。

コマンドラインから System Monitor ツールを起動するには、シェルプロンプトで **gnome-system-monitor** と入力します。この結果、System Monitor ツールが表示されます。また、GNOME デスクトップで **Super** キーを押してアクティビティーの概要を入力する場合は、**System Monitor** と入力し、**Enter** を押します。この結果、System Monitor ツールが表示されます。**Super** キーはキーボードまたはその他のハードウェアに応じて様々なキーで表示されますが、多くの場合、Windows または Command キーとして通常は **Spacebar** の左側に表示されます。

Resources タブをクリックしてシステムの CPU 使用率を表示します。

CPU History セクションで、System Monitor ツールは、CPU 使用率履歴をグラフィックで表し、現在使用中の CPU 容量のパーセンテージを表示します。

21.4. ブロックデバイスとファイルシステムの表示

21.4.1. lsblk コマンドの使用

lsblk コマンドを使用すると、利用可能なブロックデバイスのリストを表示できます。**blkid** コマンドよりも詳細な情報が提供され、出力フォーマットを細かく制御できるようになります。**udev** から情報を読み取るため、**root** 以外のユーザーも使用できます。ブロックデバイスのリストを表示するには、シェルプロンプトで以下のコマンドを入力します。

```
lsblk
```

リスト表示された各ブロックデバイスについて **lsblk** コマンドが表示するのは次のとおりです。デバイス名 (**NAME**)、メジャーデバイス番号 (**MAJ:MIN**) およびマイナーデバイス番号 (**MAJ:MIN**) を表示します。デバイスが読み取り専用 (**RO**) の場合は、そのサイズ (**SIZE**)、そのタイプ (**TYPE**)、デバイスがマウントされている場所 (**MOUNTPOINT**) が表示されます。以下に例を示します。

```
~]$ lsblk
NAME          MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0            11:0  1 1024M 0 rom
vda           252:0  0  20G 0 rom
|-vda1         252:1  0  500M 0 part /boot
`-vda2         252:2  0 19.5G 0 part
 |-vg_kvm-lv_root (dm-0) 253:0  0  18G 0 lvm /
 `-vg_kvm-lv_swap (dm-1) 253:1  0  1.5G 0 lvm [SWAP]
```

デフォルトでは、**lsblk** はツリーのような形式でブロックデバイスをリスト表示します。情報を通常のリストとして表示するには、**-l** コマンドラインオプションを追加します。

```
lsblk -l
```

たとえば、以下のようになります。

```
~]$ lsblk -l
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0          11:0  1 1024M 0 rom
vda         252:0  0  20G 0 rom
vda1         252:1  0  500M 0 part /boot
vda2         252:2  0 19.5G 0 part
vg_kvm-lv_root (dm-0) 253:0  0  18G 0 lvm /
vg_kvm-lv_swap (dm-1) 253:1  0  1.5G 0 lvm [SWAP]
```

利用可能なコマンドラインオプションのリストは、lsblk(8) の man ページを参照してください。

21.4.2. blkid コマンドの使用

blkid コマンドを使用すると、利用可能なブロックデバイスに関する低レベルの情報を表示できます。**root** 権限が必要であるため、**root** 以外のユーザーは **lsblk** コマンドを使用する必要があります。これを行うには、**root** で次のコマンドを実行します。

```
blkid
```

リスト表示された各ブロックデバイスについて **blkid** コマンドは、**汎用一意識別子 (UUID)**、ファイルシステムのタイプ (**TYPE**)、ボリュームラベル (**LABEL**) などの属性を表示します。以下に例を示します。

```
~]# blkid
/dev/vda1: UUID="7fa9c421-0054-4555-b0ca-b470a97a3d84" TYPE="ext4"
/dev/vda2: UUID="7IvYzk-TnnK-oPjf-ipdD-cofz-DXaJ-gPdgBW" TYPE="LVM2_member"
/dev/mapper/vg_kvm-lv_root: UUID="a07b967c-71a0-4925-ab02-aebcad2ae824" TYPE="ext4"
/dev/mapper/vg_kvm-lv_swap: UUID="d7ef54ca-9c41-4de4-ac1b-4193b0c1ddb6" TYPE="swap"
```

デフォルトでは、**blkid** コマンドはすべての利用可能なブロックデバイスをリスト表示します。特定のデバイスの情報のみを表示するには、コマンドラインでデバイス名を指定します。

```
blkid device_name
```

たとえば、**/dev/vda1** に関する情報を表示するには、**root** で以下のコマンドを入力します。

```
~]# blkid /dev/vda1
/dev/vda1: UUID="7fa9c421-0054-4555-b0ca-b470a97a3d84" TYPE="ext4"
```

また、上記のコマンドを **-p** および **-o udev** のコマンドラインオプションと共に使用して、詳細情報を取得することもできます。このコマンドを実行するには、**root** 権限が必要な点に注意してください。

```
blkid -po udev device_name
```

以下に例を示します。

```
~]# blkid -po udev /dev/vda1
ID_FS_UUID=7fa9c421-0054-4555-b0ca-b470a97a3d84
ID_FS_UUID_ENC=7fa9c421-0054-4555-b0ca-b470a97a3d84
ID_FS_VERSION=1.0
ID_FS_TYPE=ext4
ID_FS_USAGE=filesystem
```

利用可能なコマンドラインオプションのリストは、blkid(8) の man ページを参照してください。

21.4.3. findmnt コマンドの使用

findmnt コマンドは、現在マウントされているファイルシステムのリストを表示します。シェルプロンプトで以下を入力してください。

```
findmnt
```

リスト表示された各ファイルシステムについて **findmnt** コマンドが表示するのは、次のとおりです。ターゲットマウントポイント **(TARGET)**、ソースデバイス **(SOURCE)**、ファイルシステムタイプ **(FSTYPE)**、および関連するマウントオプション **(OPTIONS)**。以下に例を示します。

```
~]$ findmnt
TARGET                SOURCE      FSTYPE     OPTIONS
/                     /dev/mapper/rhel-root
                                  xfs        rw,relatime,seclabel,attr2,inode64,noquota
├─/proc               proc        proc       rw,nosuid,nodev,noexec,relatime
│ ├─/proc/sys/fs/binfmt_misc   systemd-1    autofs
rw,relatime,fd=32,pgrp=1,timeout=300,minproto=5,maxproto=5,direct
│ └─/proc/fs/nfsd     sunrpc      nfsd       rw,relatime
├─/sys                sysfs       sysfs      rw,nosuid,nodev,noexec,relatime,seclabel
│ ├─/sys/kernel/security     securityfs  securityfs  rw,nosuid,nodev,noexec,relatime
│ ├─/sys/fs/cgroup    tmpfs       tmpfs      rw,nosuid,nodev,noexec,seclabel,mode=755
[output truncated]
```

デフォルトでは、**findmnt** はツリーのような形式でファイルシステムをリスト表示します。情報を通常のリストとして表示するには、**-l** コマンドラインオプションを追加します。

```
findmnt -l
```

たとえば、以下のようになります。

```
~]$ findmnt -l
TARGET          SOURCE       FSTYPE     OPTIONS
/proc           proc         proc       rw,nosuid,nodev,noexec,relatime
/sys            sysfs        sysfs      rw,nosuid,nodev,noexec,relatime,seclabel
/dev            devtmpfs     devtmpfs
rw,nosuid,seclabel,size=933372k,nr_inodes=233343,mode=755
/sys/kernel/security  securityfs  securityfs  rw,nosuid,nodev,noexec,relatime
/dev/shm        tmpfs        tmpfs      rw,nosuid,nodev,seclabel
/dev/pts        devpts       devpts
rw,nosuid,noexec,relatime,seclabel,gid=5,mode=620,ptmxmode=000
/run            tmpfs        tmpfs      rw,nosuid,nodev,seclabel,mode=755
/sys/fs/cgroup  tmpfs        tmpfs      rw,nosuid,nodev,noexec,seclabel,mode=755
[output truncated]
```

特定のタイプのファイルシステムのみをリスト表示するよう選択することも可能です。これを行うには、**-t** コマンドラインオプションに続けてファイルシステムタイプを追加します。

```
findmnt -t type
```

たとえば、すべての **xfs** ファイルシステムをリスト表示するには、以下のコマンドを入力します。

```
~]$ findmnt -t xfs
TARGET SOURCE      FSTYPE OPTIONS
/    /dev/mapper/rhel-root xfs  rw,relatime,seclabel,attr2,inode64,noquota
└─/boot /dev/vda1      xfs  rw,relatime,seclabel,attr2,inode64,noquota
```

利用可能なコマンドラインオプションのリストは、findmnt(8) の man ページを参照してください。

21.4.4. df コマンドの使用

df コマンドは、システムのディスク領域の使用量についての詳しいレポートを表示します。シェルプロンプトで以下を入力してください。

```
df
```

リスト表示された各ファイルシステムについて **df** コマンドが表示するのは次のとおりです。名前 **(Filesystem)**、サイズ **(1K-blocks** または **Size)**、使用容量 **(Used)**、利用可能な領域のサイズ **(**利用可能**)**、領域使用量のパーセンテージ **(%)**、およびファイルシステムがマウントされている場所 **(Mounted on)**。以下に例を示します。

```
~]$ df
Filesystem          1K-blocks   Used Available Use% Mounted on
/dev/mapper/vg_kvm-lv_root 18618236  4357360 13315112 25% /
tmpfs                380376    288  380088  1% /dev/shm
/dev/vda1            495844   77029  393215 17% /boot
```

デフォルトでは、**df** コマンドは 1 キロバイトブロック単位でパーティションのサイズを、キロバイト単位で使用中および利用可能なディスク領域の容量を表示します。メガバイトおよびギガバイトで情報を表示するには、コマンドラインオプション **-h** を指定して、**df** が人間が判読可能な形式で値を表示します。

```
df -h
```

たとえば、以下のようになります。

```
~]$ df -h
Filesystem        Size Used Avail Use% Mounted on
/dev/mapper/vg_kvm-lv_root  18G 4.2G  13G 25% /
tmpfs            372M 288K 372M  1% /dev/shm
/dev/vda1        485M  76M 384M 17% /boot
```

*利用可能なコマンドラインオプションのリストは、*df*(1) の* man *ページを参照してください。*

21.4.5. du コマンドの使用

***du** コマンドはディレクトリー内のファイルが使用している領域を表示します。現在の作業ディレクトリー内のサブディレクトリー用のディスク使用量を表示するには、オプションなしで以下のコマンドを実行します。*

```
du
```

以下に例を示します。

```
~]$ du
14972  ./Downloads
4    ./.mozilla/extensions
4    ./.mozilla/plugins
12   ./.mozilla
15004  .
```

*デフォルトでは、**du** コマンドはキロバイト単位でディスク使用量を表示します。メガバイトおよびギガバイトで情報を表示するには、コマンドラインオプション **-h** を指定して、**df** が人間が判読可能な形式で値を表示します。*

```
du -h
```

たとえば、以下のようになります。

```
~]$ du -h
15M   ./Downloads
4.0K  ./.mozilla/extensions
4.0K  ./.mozilla/plugins
12K   ./.mozilla
15M   .
```

***du** コマンドは、リストの最後で常に現在のディレクトリーの合計量を表示します。この情報のみを表示するには、**-s** コマンドラインオプションを指定します。*

```
du -sh
```

以下に例を示します。

```
~]$ du -sh
15M   .
```

*利用可能なコマンドラインオプションのリストは、*du*(1) の* man *ページを参照してください。*

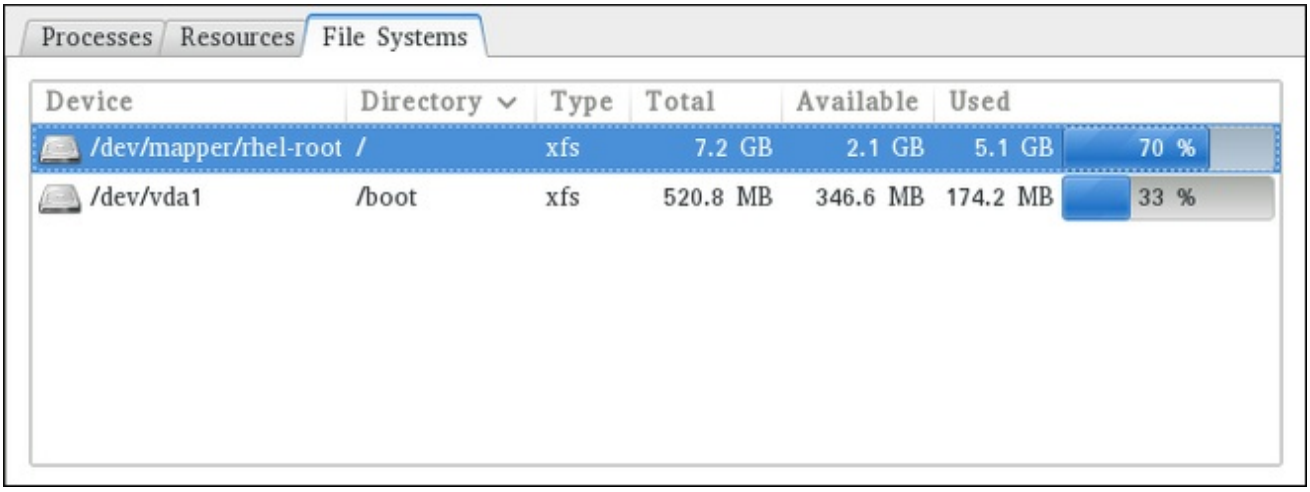
21.4.6. システムモニターツールの使用

System Monitor ツールの **File System** タブは、グラフィカルユーザーインターフェイスでファイルシステムおよびディスク領域の使用量を表示します。

コマンドラインから System Monitor ツールを起動するには、シェルプロンプトで **gnome-system-monitor** と入力します。この結果、System Monitor ツールが表示されます。また、GNOME デスクトップで **Super** キーを押してアクティビティーの概要を入力する場合は、**System Monitor** と入力し、**Enter** を押します。この結果、System Monitor ツールが表示されます。**Super** キーはキーボードまたはその他のハードウェアに応じて様々なキーで表示されますが、多くの場合、Windows または Command キーとして通常は **Spacebar** の左側に表示されます。

File Systems (ファイルシステム) タブをクリックしてファイルシステムのリストを表示します。

図21.3 システムモニター – ファイルシステム



リスト表示されている各ファイルシステムについて、システム System Monitor ツールはソースデバイス (**Device**)、ターゲットマウントポイント (**Directory**)、およびファイルシステムタイプ (**Type**) 、そのサイズ (**Total**)、利用可能な容量 (**Available**)、および使用中 (**Used**) が表示されます。

21.5. ハードウェア情報の表示

21.5.1. lspci コマンドの使用

lspci コマンドは、PCI バスおよびそれらに接続されているデバイスの情報を表示します。システム内にあるすべての PCI デバイスをリスト表示するには、シェルプロンプトで以下を入力してください。

```
lspci
```

これは、以下のようにデバイスのシンプルなリストを表示します。

```
~]$ lspci
00:00.0 Host bridge: Intel Corporation 82X38/X48 Express DRAM Controller
00:01.0 PCI bridge: Intel Corporation 82X38/X48 Express Host-Primary PCI Express Bridge
00:1a.0 USB Controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller #4 (rev
02)
00:1a.1 USB Controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller #5 (rev
02)
```

```
00:1a.2 USB Controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller #6 (rev
02)
[output truncated]
```

-v コマンドラインオプションを使用して詳細の出力を表示することもできます。詳細の出力には**-vv** を指定することもできます。

```
lspci -v|-vv
```

たとえば、システムのビデオカードの製造元やモデル、メモリーサイズを確認するには、以下を入力します。

```
~]$ lspci -v
[output truncated]

01:00.0 VGA compatible controller: nVidia Corporation G84 [Quadro FX 370] (rev a1) (prog-if
00 [VGA controller])
    Subsystem: nVidia Corporation Device 0491
    Physical Slot: 2
    Flags: bus master, fast devsel, latency 0, IRQ 16
    Memory at f2000000 (32-bit, non-prefetchable) [size=16M]
    Memory at e0000000 (64-bit, prefetchable) [size=256M]
    Memory at f0000000 (64-bit, non-prefetchable) [size=32M]
    I/O ports at 1100 [size=128]
    Expansion ROM at <unassigned> [disabled]
    Capabilities: <access denied>
    Kernel driver in use: nouveau
    Kernel modules: nouveau, nvidiafb

[output truncated]
```

利用可能なコマンドラインオプションのリストは、lspci(8) の man ページを参照してください。

21.5.2. lsusb コマンドの使用

lsusb コマンドは、USB バスおよびそれらに接続されているデバイスの情報を表示します。システム内にあるすべての USB デバイスをリスト表示するには、シェルプロンプトで以下を入力してください。

```
lsusb
```

これは、以下のようにデバイスのシンプルなリストを表示します。

```
~]$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
[output truncated]
Bus 001 Device 002: ID 0bda:0151 Realtek Semiconductor Corp. Mass Storage Device
(Multicard Reader)
Bus 008 Device 002: ID 03f0:2c24 Hewlett-Packard Logitech M-UAL-96 Mouse
Bus 008 Device 003: ID 04b3:3025 IBM Corp.
```

-v コマンドラインオプションを使用すると、さらに詳細な出力を表示することもできます。

```
lsusb -v
```

-

たとえば、以下のようになります。

```
~]$ lsusb -v
[output truncated]

Bus 008 Device 002: ID 03f0:2c24 Hewlett-Packard Logitech M-UAL-96 Mouse
Device Descriptor:
 bLength         18
 bDescriptorType     1
 bcdUSB         2.00
 bDeviceClass      0 (Defined at Interface level)
 bDeviceSubClass     0
 bDeviceProtocol     0
 bMaxPacketSize0     8
 idVendor       0x03f0 Hewlett-Packard
 idProduct      0x2c24 Logitech M-UAL-96 Mouse
 bcdDevice       31.00
 iManufacturer       1
 iProduct        2
 iSerial         0
 bNumConfigurations   1
 Configuration Descriptor:
  bLength         9
  bDescriptorType     2
[output truncated]
```

*利用可能なコマンドラインオプションのリストは、*lsusb*(8) の man ページを参照してください。*

21.5.3. lscpu コマンドの使用

***lscpu** コマンドを使用すると、CPU の数、アーキテクチャー、ベンダー、ファミリー、モデル、CPU キャッシュなど、システムに存在する CPU に関する情報をリスト表示できます。シェルプロンプトで以下を入力してください。*

```
lscpu
```

以下に例を示します。

```
~]$ lscpu
Architecture:     x86_64
CPU op-mode(s):    32-bit, 64-bit
Byte Order:       Little Endian
CPU(s):         4
On-line CPU(s) list:  0-3
Thread(s) per core:  1
Core(s) per socket:  4
Socket(s):        1
NUMA node(s):     1
Vendor ID:        GenuineIntel
CPU family:       6
Model:          23
Stepping:         7
CPU MHz:          1998.000
BogoMIPS:         4999.98
```

```
Virtualization:    VT-x
L1d cache:         32K
L1i cache:         32K
L2 cache:          3072K
NUMA node0 CPU(s):   0-3
```

利用可能なコマンドラインオプションのリストは、lscpu(1) の man ページを参照してください。

21.6. ハードウェアエラーの確認

Red Hat Enterprise Linux 7 では、新しい**ハードウェアイベントレポートメカニズム** (HERM) が導入されました。 このメカニズムは、システムにより報告されたエラーと DIMM (Dual In-line Memory: デュアルインラインメモリーモジュール) 向けの EDAC (Error Detection And Correction:エラー検出および修正) メカニズムにより報告されたエラーを収集し、ユーザースペースに報告します。ユーザースペースデーモンである **rasdaemon** は、カーネルの追跡メカニズムから送信されるすべての RAS (Reliability, Availability, and Serviceability: 信頼性、利用可能性、およびサービス性) エラーイベントを取得および処理し、ログに記録します。以前に **edac-utils** により提供されていた機能は、**rasdaemon** により置き換えられました。

rasdaemon をインストールするには、**root** で以下のコマンドを発行します。

```
~]# yum install rasdaemon
```

サービスを以下のように起動します。

```
~]# systemctl start rasdaemon
```

システム起動時にサービスを実行するには、以下のコマンドを入力します。

```
~]# systemctl enable rasdaemon
```

ras-mc-ctl ユーティリティーは、EDAC ドライバーと連携する手段を提供します。以下のコマンドを入力してコマンドオプションのリストを表示します。

```
~]$ ras-mc-ctl --help
Usage: ras-mc-ctl [OPTIONS...]
 --quiet      Quiet operation.
 --mainboard    Print mainboard vendor and model for this hardware.
 --status       Print status of EDAC drivers.
output truncated
```

メモリーコントローラーイベントのサマリーを表示するには、**root** として実行します。

```
~]# ras-mc-ctl --summary
Memory controller events summary:
    Corrected on DIMM Label(s): 'CPU_SrcID#0_Ha#0_Chan#0_DIMM#0' location: 0:0:0:-1
errors: 1

No PCIe AER errors.

No Extlog errors.
MCE records summary:
    1 MEMORY CONTROLLER RD_CHANNEL0_ERR Transaction: Memory read error errors
    2 No Error errors
```

- メモリーコントローラーが報告するエラーのリストを表示するには、**root** として実行します。

```
~]# ras-mc-ctl --errors
Memory controller events:
1 3172-02-17 00:47:01 -0500 1 Corrected error(s): memory read error at
CPU_SrcID#0_Ha#0_Chan#0_DIMM#0 location: 0:0:0:-1, addr 65928, grain 7, syndrome 0
area:DRAM err_code:0001:0090 socket:0 ha:0 channel_mask:1 rank:0

No PCIe AER errors.

No Extlog errors.

MCE events:
1 3171-11-09 06:20:21 -0500 error: MEMORY CONTROLLER RD_CHANNEL0_ERR Transaction:
Memory read error, mcg mcgstatus=0, mci Corrected_error, n_errors=1, mcgcap=0x01000c16,
status=0x8c00004000010090, addr=0x1018893000, misc=0x15020a086, walltime=0x57e96780,
cpuid=0x00050663, bank=0x00000007
2 3205-06-22 00:13:41 -0400 error: No Error, mcg mcgstatus=0, mci Corrected_error
Error_enabled, mcgcap=0x01000c16, status=0x9400000000000000, addr=0x0000abcd,
walltime=0x57e967ea, cpuid=0x00050663, bank=0x00000001
3 3205-06-22 00:13:41 -0400 error: No Error, mcg mcgstatus=0, mci Corrected_error
Error_enabled, mcgcap=0x01000c16, status=0x9400000000000000, addr=0x00001234,
walltime=0x57e967ea, cpu=0x00000001, cpuid=0x00050663, apicid=0x00000002,
bank=0x00000002
```

これらのコマンドは、**ras-mc-ctl(8)** man ページでも説明されています。

21.7. NET-SNMP を使用したパフォーマンスのモニタリング

Red Hat Enterprise Linux 7 には、柔軟かつ拡張可能な SNMP (simple network management protocol シンプルネットワークマネージメントプロトコル) エージェントを含む Net-SNMP ソフトウェアスイートが含まれています。このエージェントと関連ユーティリティーを使用すると、多くのシステムからのパフォーマンスデータを **SNMP** プロトコルによるポーリングに対応する各種ツールに提供することができます。

このセクションでは、ネットワーク上でパフォーマンスデータを安全に提供するための Net-SNMP エージェントの設定方法、SNMP プロトコルを使用したデータの取得方法、カスタムのパフォーマンスメトリックを提供するための SNMP エージェントの拡張方法について説明します。

21.7.1. Net-SNMP のインストール

Net-SNMP ソフトウェアスイートは、Red Hat Enterprise Linux ソフトウェアディストリビューションの RPM パッケージのセットとして利用可能です。表21.2「利用可能な Net-SNMP パッケージ」 は、各パッケージと内容を要約したものです。

表21.2 利用可能な Net-SNMP パッケージ

パッケージ	提供する項目
net-snmp	SNMP Agent Daemon とドキュメント。このパッケージは、パフォーマンスデータをエクスポートするために必要です。

パッケージ	提供する項目
net-snmp-libs	**netsnmp** ライブラリーと、同梱の **MIB** (Management Information Base: **管理情報ベース**)。このパッケージは、パフォーマンスデータをエクスポートするために必要です。
net-snmp-utils	**snmpget** や **snmpwalk** などの SNMP クライアント。このパッケージは、SNMP によりシステムのパフォーマンスデータをクエリーするために必要です。
net-snmp-perl	**mib2c** ユーティリティーおよび **NetSNMP** Perl モジュールこのパッケージは Optional チャンネルにより提供されることに注意してください。Red Hat 追加チャンネルの詳細については、「Optional および Supplementary リポジトリーの追加」 を参照してください。
net-snmp-python	Python 向け SNMP クライアントライブラリー。このパッケージは Optional チャンネルにより提供されることに注意してください。Red Hat 追加チャンネルの詳細については、「Optional および Supplementary リポジトリーの追加」 を参照してください。

*これらのパッケージをインストールするには、以下の形式で **yum** コマンドを使用します。*

```
yum install package&hellip;
```

*たとえば、本セクションで使用される SNMP Agent Daemon および SNMP クライアントをインストールするには、**root** としてシェルプロンプトで以下を入力します。*

```
~]# yum install net-snmp net-snmp-libs net-snmp-utils
```

Red Hat Enterprise Linux に新しいパッケージをインストールする方法の詳細は、「パッケージのインストール」 を参照してください。

21.7.2. Net-SNMP Daemon の実行

*net-snmp パッケージには、SNMP Agent Daemon である **snmpd** が含まれています。本セクションでは、**snmpd** サービスを起動、停止、再起動する方法を説明します。Red Hat Enterprise Linux 7 でのシステムサービスの管理の詳細については、10章systemd によるサービス管理 を参照してください。*

21.7.2.1. サービスの起動

*現行のセッションで **snmpd** サービスを実行するには、シェルプロンプトで **root** として以下を入力します。*

```
systemctl start snmpd.service
```

起動時にサービスが自動的に起動するよう設定するには、以下のコマンドを使用します。

```
systemctl enable snmpd.service
```

21.7.2.2. サービスの停止

実行中の **snmpd** サービスを停止するには、シェルプロンプトで **root** として以下を入力します。

```
systemctl stop snmpd.service
```

ブート時のサービスの起動を無効にするには、以下のコマンドを使用します。

```
systemctl disable snmpd.service
```

21.7.2.3. サービスの再起動

実行中の **snmpd** サービスを再起動するには、シェルプロンプトで以下を入力します。

```
systemctl restart snmpd.service
```

このコマンドで、サービスの停止と再起動が連続して行われます。サービスを停止せずに設定の再読み込みだけを行いたい場合は、代わりに以下のコマンドを実行します。

```
systemctl reload snmpd.service
```

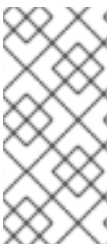
これにより、実行中の **snmpd** サービスが設定を再読み込みします。

21.7.3. Net-SNMP の設定

Net-SNMP エージェントデーモン設定を変更するには、**/etc/snmp/snmpd.conf** 設定ファイルを編集します。Red Hat Enterprise Linux 7 に備わっているデフォルトの **snmpd.conf** ファイルには、多くのコメントが含まれているため、エージェント設定の際の適切なスタート地点となります。

本セクションでは、システム情報と認証の設定という 2 つの一般的なタスクにフォーカスしています。利用可能な設定ディレクティブの詳細については、**snmpd.conf**(5) の man ページを参照してください。また、net-snmp パッケージには **snmpconf** と呼ばれるユーティリティーがあり、これを使用して有効なエージェント設定を対話形式で作成できます。

本セクションで説明されている **snmpwalk** ユーティリティーを使用するには、net-snmp-utils パッケージがインストールされている必要がある点に注意してください。



注記

設定ファイルの変更を反映させるには、**root** として以下のコマンドを実行し、**snmpd** サービスに設定の再読み取りを強制します。

```
systemctl reload snmpd.service
```

21.7.3.1. システム情報の設定

Net-SNMP は、**system** ツリー経由で基本的なシステム情報を提供します。たとえば、次の**snmpwalk** コマンドはデフォルトのエージェント設定を持つ**system** ツリーを示しています。

```
~]# snmpwalk -v2c -c public localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 3.10.0-123.el7.x86_64 #1
SMP Mon May 5 11:16:57 EDT 2014 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (464) 0:00:04.64
SNMPv2-MIB::sysContact.0 = STRING: Root <root@localhost> (configure
/etc/snmp/snmp.local.conf)
[output truncated]
```

デフォルトでは、**sysName** オブジェクトはホスト名に設定されています。**sysLocation** および **sysContact** オブジェクトは、**syslocation** ディレクティブおよび**syscontact** ディレクティブの値を変更することで、**/etc/snmp/snmpd.conf** ファイルで設定できます。以下に例を示します。

```
syslocation Datacenter, Row 4, Rack 3
syscontact UNIX Admin <admin@example.com>
```

設定ファイルに変更を加えた後は、再度 **snmpwalk** コマンドを実行し、設定を再読み込みしてテストします。

```
~]# systemctl reload snmp.service
~]# snmpwalk -v2c -c public localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 3.10.0-123.el7.x86_64 #1
SMP Mon May 5 11:16:57 EDT 2014 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (35424) 0:05:54.24
SNMPv2-MIB::sysContact.0 = STRING: UNIX Admin <admin@example.com>
SNMPv2-MIB::sysName.0 = STRING: localhost.localdomain
SNMPv2-MIB::sysLocation.0 = STRING: Datacenter, Row 4, Rack 3
[output truncated]
```

21.7.3.2. 認証の設定

Net-SNMP Agent Daemon は SNMP プロトコルの 3 つの全バージョンに対応します。最初の 2 つのバージョン (1 と 2c) は、**コミュニティー文字列** を使用した簡易認証を提供します。この文字列は、エージェントとクライアントのユーティリティー間で共有される秘密です。この文字列は、ネットワーク上でクリアテキストで渡されますが、安全であるとはみなされません。SNMP プロトコルのバージョン 3 は、各種プロトコルを使用したユーザー認証とメッセージの暗号化に対応しています。また Net-SNMP エージェントは、SSH によるトンネリング、および X.509 証明書を用いた TLS 認証にも対応しています。

SNMP Version 2c Community の設定

SNMP version 2c community を設定するには、**/etc/snmp/snmpd.conf** 設定ファイルの**rocommunity** または **rwcommunity** ディレクティブを使用します。ディレクティブの形式は、以下のとおりです。

```
directive community source OID
```

ここでの community は使用するコミュニティー文字列、source は IP アドレスまたはサブネット、OID はアクセスを提供する SNMP ツリーです。たとえば、次のディレクティブは、ローカルマシン上でコミュニティー文字列 redhat を使用するクライアントに **system** ツリーへの読み取り専用のアクセスを与えます。

```
rocommunity redhat 127.0.0.1 .1.3.6.1.2.1.1
```

設定をテストするには、**-v** と **-c** オプションを指定して、**snmpwalk** コマンドを使用します。

```
~]# snmpwalk -v2c -c redhat localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 3.10.0-123.el7.x86_64 #1
SMP Mon May 5 11:16:57 EDT 2014 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (101376) 0:16:53.76
SNMPv2-MIB::sysContact.0 = STRING: UNIX Admin <admin@example.com>
SNMPv2-MIB::sysName.0 = STRING: localhost.localdomain
SNMPv2-MIB::sysLocation.0 = STRING: Datacenter, Row 4, Rack 3
[output truncated]
```

SNMP Version 3 User の設定

SNMP version 3 user を設定するには、**net-snmp-create-v3-user** コマンドを使用します。このコマンドは、**/var/lib/net-snmp/snmpd.conf** および **/etc/snmp/snmpd.conf** ファイルへエントリーを追加し、ユーザーを作成してそのユーザーにアクセスを付与します。**net-snmp-create-v3-user** コマンドは、エージェントが実行中でない時にのみ実行可能な点に注意してください。以下の例では redhatsnmp というパスワードを持つ admin ユーザーを作成します。

```
~]# systemctl stop snmpd.service
~]# net-snmp-create-v3-user
Enter a SNMPv3 user name to create:
admin
Enter authentication pass-phrase:
redhatsnmp
Enter encryption pass-phrase:
 [press return to reuse the authentication pass-phrase]

adding the following line to /var/lib/net-snmp/snmpd.conf:
  createUser admin MD5 "redhatsnmp" DES
adding the following line to /etc/snmp/snmpd.conf:
  rwuser admin
~]# systemctl start snmpd.service
```

net-snmp-create-v3-user が **/etc/snmp/snmpd.conf** に追加される **rwuser** ディレクティブ (**-ro** コマンドオプションを指定する場合は **rouser**) には、**rwcommunity** および **rocommunity** ディレクティブに類似した形式があります。

```
directive user noauth|auth|priv OID
```

ここでの user はユーザー名で、OID はアクセスを提供する SNMP ツリーです。デフォルトでは、Net-SNMP Agent Daemon は認証済み要求のみを許可します (**auth** オプション)。**noauth** オプションを使用すると、認証されていないリクエストを許可でき、**priv** オプションは暗号化の使用を強制します。**authpriv** オプションを使用すると、要求の認証、応答の暗号化が必要であると指定します。

たとえば、以下の行では、ユーザー admin にツリー全体への読み取りと書き込みのアクセスを付与します。

```
rwuser admin authpriv .1
```

設定をテストするには、使用しているユーザーのホームディレクトリー内に **.snmp/** ディレクトリーを作成して、そのディレクトリーに次の行を含む **snmp.conf** と呼ばれる設定ファイルを作成します (**~/.snmp/snmp.conf**):

```
defVersion 3
defSecurityLevel authPriv
defSecurityName admin
defPassphrase redhatsnmp
```

これで、エージェントをクエリーする場合、**snmpwalk** コマンドはこれらの認証設定を使用するようになります。

```
~]$ snmpwalk -v3 localhost system
SNMPv2-MIB::sysDescr.0 = STRING: Linux localhost.localdomain 3.10.0-123.el7.x86_64 #1
SMP Mon May 5 11:16:57 EDT 2014 x86_64
[output truncated]
```

21.7.4. SNMP によるパフォーマンスデータの取得

Red Hat Enterprise Linux の Net-SNMP Agent は、SNMP プロトコルによりパフォーマンスの各種情報を提供します。さらにエージェントは、システム上のインストールされた RPM パッケージのリスト、システム上で現在実行中のプロセスリスト、またはシステムのネットワーク設定をクエリーすることもできます。

本セクションでは、SNMP による利用可能なパフォーマンスチューニングに関連する OID の概要について説明します。ここでは、**net-snmp-utils** パッケージがインストールされ、「認証の設定」 の記載通りに、ユーザーは SNMP ツリーへのアクセスが許可されていることを前提としています。

21.7.4.1. ハードウェアの設定

Net-SNMP に含まれる **Host Resources MIB** は、ホストの現在のハードウェアおよびソフトウェア設定に関する情報をクライアントユーティリティーに表示します。表21.3「利用可能な OID」 は、その MIB で利用可能なさまざまな OID の概要を示します。

表21.3 利用可能な OID

OID	詳細
HOST-RESOURCES-MIB::hrSystem	アップタイム、ユーザー数、実行中のプロセス数などのシステム情報全般が含まれています。
HOST-RESOURCES-MIB::hrStorage	メモリーおよびファイルシステムの使用に関するデータが含まれています。
HOST-RESOURCES-MIB::hrDevices	すべてのプロセッサー、ネットワークデバイス、ファイルシステムのリストが含まれています。
HOST-RESOURCES-MIB::hrSWRun	実行中の全プロセスリストが含まれています。
HOST-RESOURCES-MIB::hrSWRunPerf	HOST-RESOURCES-MIB::hrSWRun からのプロセステーブル上のメモリーと CPU 統計が含まれています。

OID	詳細
HOST-RESOURCES-MIB::hrSWInstalled	RPM データベースのリストが含まれています。

*入手可能な情報の概要を取得するために使用できる Host Resources MIB には、多くの SNMP テーブルがあります。次の例は、**HOST-RESOURCES-MIB::hrFSTable** を表示しています。*

```
~]$ snmptable -Cb localhost HOST-RESOURCES-MIB::hrFSTable
SNMP table: HOST-RESOURCES-MIB::hrFSTable

 Index MountPoint RemoteMountPoint             Type
  Access Bootable StorageIndex LastFullBackupDate LastPartialBackupDate
   1    "/"         "" HOST-RESOURCES-TYPES::hrFSLinuxExt2
 readWrite   true      31   0-1-1,0:0:0.0     0-1-1,0:0:0.0
   5 "/dev/shm"         ""   HOST-RESOURCES-TYPES::hrFSOther
 readWrite  false      35   0-1-1,0:0:0.0     0-1-1,0:0:0.0
   6  "/boot"         "" HOST-RESOURCES-TYPES::hrFSLinuxExt2
 readWrite  false      36   0-1-1,0:0:0.0     0-1-1,0:0:0.0
```

***HOST-RESOURCES-MIB** の詳細については、**/usr/share/snmp/mibs/HOST-RESOURCES-MIB.txt** ファイルを参照してください。*

21.7.4.2. CPU およびメモリー情報

*大半のシステムのパフォーマンスデータは **UCD SNMP MIB** にあります。**systemStats** OID は、プロセッサー使用率に関する多くのカウンターを提供します。*

```
~]$ snmpwalk localhost UCD-SNMP-MIB::systemStats
UCD-SNMP-MIB::ssIndex.0 = INTEGER: 1
UCD-SNMP-MIB::ssErrorName.0 = STRING: systemStats
UCD-SNMP-MIB::ssSwapIn.0 = INTEGER: 0 kB
UCD-SNMP-MIB::ssSwapOut.0 = INTEGER: 0 kB
UCD-SNMP-MIB::ssIOSent.0 = INTEGER: 0 blocks/s
UCD-SNMP-MIB::ssIOReceive.0 = INTEGER: 0 blocks/s
UCD-SNMP-MIB::ssSysInterrupts.0 = INTEGER: 29 interrupts/s
UCD-SNMP-MIB::ssSysContext.0 = INTEGER: 18 switches/s
UCD-SNMP-MIB::ssCpuUser.0 = INTEGER: 0
UCD-SNMP-MIB::ssCpuSystem.0 = INTEGER: 0
UCD-SNMP-MIB::ssCpuIdle.0 = INTEGER: 99
UCD-SNMP-MIB::ssCpuRawUser.0 = Counter32: 2278
UCD-SNMP-MIB::ssCpuRawNice.0 = Counter32: 1395
UCD-SNMP-MIB::ssCpuRawSystem.0 = Counter32: 6826
UCD-SNMP-MIB::ssCpuRawIdle.0 = Counter32: 3383736
UCD-SNMP-MIB::ssCpuRawWait.0 = Counter32: 7629
UCD-SNMP-MIB::ssCpuRawKernel.0 = Counter32: 0
UCD-SNMP-MIB::ssCpuRawInterrupt.0 = Counter32: 434
UCD-SNMP-MIB::ssIORawSent.0 = Counter32: 266770
UCD-SNMP-MIB::ssIORawReceived.0 = Counter32: 427302
UCD-SNMP-MIB::ssRawInterrupts.0 = Counter32: 743442
UCD-SNMP-MIB::ssRawContexts.0 = Counter32: 718557
```

```
UCD-SNMP-MIB::ssCpuRawSoftIRQ.0 = Counter32: 128
UCD-SNMP-MIB::ssRawSwapIn.0 = Counter32: 0
UCD-SNMP-MIB::ssRawSwapOut.0 = Counter32: 0
```

特に、**ssCpuRawUser**、**ssCpuRawSystem**、**ssCpuRawWait**、**ssCpuRawIdle** OID は、システムがカーネルスペース、ユーザー空間、または I/O でプロセッサー時間の大半を要しているかどうかを判断するのに役に立つカウンターを提供します。**ssRawSwapIn** および **ssRawSwapOut** は、システムがメモリー消費から不足しているかどうかを判断する際に役立ちます。

さらなるメモリー情報は、**free** コマンドと類似するデータを提供する **UCD-SNMP-MIB::memory** OID で入手できます。

```
~]$ snmpwalk localhost UCD-SNMP-MIB::memory
UCD-SNMP-MIB::memIndex.0 = INTEGER: 0
UCD-SNMP-MIB::memErrorName.0 = STRING: swap
UCD-SNMP-MIB::memTotalSwap.0 = INTEGER: 1023992 kB
UCD-SNMP-MIB::memAvailSwap.0 = INTEGER: 1023992 kB
UCD-SNMP-MIB::memTotalReal.0 = INTEGER: 1021588 kB
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 634260 kB
UCD-SNMP-MIB::memTotalFree.0 = INTEGER: 1658252 kB
UCD-SNMP-MIB::memMinimumSwap.0 = INTEGER: 16000 kB
UCD-SNMP-MIB::memBuffer.0 = INTEGER: 30760 kB
UCD-SNMP-MIB::memCached.0 = INTEGER: 216200 kB
UCD-SNMP-MIB::memSwapError.0 = INTEGER: noError(0)
UCD-SNMP-MIB::memSwapErrorMsg.0 = STRING:
```

負荷平均は、**UCD SNMP MIB** でも入手可能です。SNMP テーブル **UCD-SNMP-MIB::laTable** には、1 分、5 分、15 分の負荷平均を示したリストがあります。

```
~]$ snmptable localhost UCD-SNMP-MIB::laTable
SNMP table: UCD-SNMP-MIB::laTable

 laIndex laNames laLoad laConfig laLoadInt laLoadFloat laErrorFlag laErrMessage
    1 Load-1  0.00  12.00     0  0.000000   noError
    2 Load-5  0.00  12.00     0  0.000000   noError
    3 Load-15  0.00  12.00     0  0.000000   noError
```

21.7.4.3. ファイルシステムとディスク情報

Host Resources MIB は、ファイルシステムのサイズと使用量についての情報を表示します。**HOST-RESOURCES-MIB::hrStorageTable** テーブルには、各ファイルシステム (および各メモリープール) のエントリーがあります。

```
~]$ snmptable -Cb localhost HOST-RESOURCES-MIB::hrStorageTable
SNMP table: HOST-RESOURCES-MIB::hrStorageTable

 Index                   Type      Descr
AllocationUnits  Size  Used AllocationFailures
   1      HOST-RESOURCES-TYPES::hrStorageRam Physical memory
1024 Bytes 1021588 388064         ?
   3 HOST-RESOURCES-TYPES::hrStorageVirtualMemory Virtual memory
1024 Bytes 2045580 388064         ?
   6     HOST-RESOURCES-TYPES::hrStorageOther Memory buffers
1024 Bytes 1021588 31048         ?
```

```
   7    HOST-RESOURCES-TYPES::hrStorageOther  Cached memory
1024 Bytes 216604 216604         ?
  10 HOST-RESOURCES-TYPES::hrStorageVirtualMemory   Swap space
1024 Bytes 1023992   0         ?
  31   HOST-RESOURCES-TYPES::hrStorageFixedDisk        /
4096 Bytes 2277614 250391         ?
  35   HOST-RESOURCES-TYPES::hrStorageFixedDisk    /dev/shm
4096 Bytes 127698   0         ?
  36   HOST-RESOURCES-TYPES::hrStorageFixedDisk      /boot
1024 Bytes 198337 26694         ?
```

HOST-RESOURCES-MIB::hrStorageSize および*HOST-RESOURCES-MIB::hrStorageUsed* の OID を使用すると、それぞれマウントされたファイルシステムの残存容量を算出することができます。

I/O データは *UCD-SNMP-MIB::systemStats (ssIORawSent.0* と *ssIORawRecieved.0)* および*UCD-DISKIO-MIB::diskIOTable* の両方で利用可能です。後者は、前者と比べてより粒度の細かいデータを提供します。このテーブルには、*diskIONReadX* および*diskIONWrittenX* の OID があり、システムブートから問題のブロックデバイスに対し読み取りおよび書き込みを実行したバイト数のカウンターを提供します。

```
~]$ snmptable -Cb localhost UCD-DISKIO-MIB::diskIOTable
SNMP table: UCD-DISKIO-MIB::diskIOTable

 Index Device   NRead NWritten Reads Writes LA1 LA5 LA15  NReadX NWrittenX
...
  25  sda 216886272 139109376 16409  4894  ?  ?  ? 216886272 139109376
  26  sda1  2455552   5120  613   2  ?  ?  ?  2455552   5120
  27  sda2  1486848     0  332   0  ?  ?  ?  1486848     0
  28  sda3 212321280 139104256 15312  4871  ?  ?  ? 212321280 139104256
```

21.7.4.4. ネットワーク情報

Interfaces MIB はネットワークデバイスの情報を提供します。*IF-MIB::ifTable* は、SNMP テーブルにシステム上の各インターフェイスのエントリー、インターフェイスの設定、インターフェイス用の各種パケットカウンターを提供します。以下の例は、2 つの物理ネットワークインターフェイスを持つシステム上の *ifTable* の最初の数コラムを示しています。

```
~]$ snmptable -Cb localhost IF-MIB::ifTable
SNMP table: IF-MIB::ifTable

 Index Descr       Type  Mtu  Speed   PhysAddress AdminStatus
   1  lo softwareLoopback 16436 10000000              up
   2 eth0  ethernetCsmacd 1500    0 52:54:0:c7:69:58     up
   3 eth1  ethernetCsmacd 1500    0 52:54:0:a7:a3:24     down
```

ネットワークトラフィックは、*IF-MIB::ifOutOctets* および*IF-MIB::ifInOctets* の OID にあります。以下の SNMP クエリーは、このシステム上の各インターフェイスに対するネットワークトラフィックを取得します。

```
~]$ snmpwalk localhost IF-MIB::ifDescr
IF-MIB::ifDescr.1 = STRING: lo
IF-MIB::ifDescr.2 = STRING: eth0
IF-MIB::ifDescr.3 = STRING: eth1
~]$ snmpwalk localhost IF-MIB::ifOutOctets
```

```
IF-MIB::ifOutOctets.1 = Counter32: 10060699
IF-MIB::ifOutOctets.2 = Counter32: 650
IF-MIB::ifOutOctets.3 = Counter32: 0
~]$ snmpwalk localhost IF-MIB::ifInOctets
IF-MIB::ifInOctets.1 = Counter32: 10060699
IF-MIB::ifInOctets.2 = Counter32: 78650
IF-MIB::ifInOctets.3 = Counter32: 0
```

21.7.5. Net-SNMP の拡張

Net-SNMP Agent は、raw システムメトリックに加えてアプリケーションメトリックを提供するために拡張することができます。これにより、パフォーマンス問題のトラブルシューティングだけでなく容量計画も行うことができます。たとえば、試験中に電子メールシステムの 5 分の負荷平均が 15 であったことを把握しておくことは役に立つかもしれませんが、毎秒 80,000 メッセージの処理中に電子メールシステムの負荷平均が 15 であることを知っておく方がはるかに役立ちます。アプリケーションメトリックがシステムメトリックと同じインターフェイスで使用可能な場合、システムパフォーマンスの様々な負荷状況の影響も視覚化することができます (たとえば 10,000 メッセージが追加されると、負荷平均は 100,000 まで直線的に増加します)。

Red Hat Enterprise Linux に同梱されているアプリケーションの多くは、Net-SNMP Agent を拡張して、SNMP によるアプリケーションメトリックを提供します。カスタムアプリケーション用にエージェントを拡張する方法もいくつかあります。本セクションでは、Optional チャンネルのシェルスクリプトと Perl プラグインを使用したエージェントの拡張方法について説明します。本セクションでは、net-snmp-utils および net-snmp-perl パッケージがインストールされ、「認証の設定」 で説明されているようにユーザーに SNMP ツリーへのアクセスが許可されていることを前提としています。

21.7.5.1. シェルスクリプトによる Net-SNMP の拡張

Net-SNMP Agent は、任意のシェルスクリプトをクエリーするために使用可能な拡張 MIB (**NET-SNMP-EXTEND-MIB**) を提供します。実行するシェルスクリプトを指定するには、**/etc/snmp/snmpd.conf** ファイルの **extend** ディレクティブを使用します。定義されると、Agent は SNMP により終了コードとコマンドの出力を提供します。以下の例では、プロセステーブルの **httpd** プロセスの数を決定するスクリプトを使用してこの仕組みを説明しています。

注記

Net-SNMP エージェントでは、**proc** ディレクティブでプロセステーブルをチェックする組み込みメカニズムも利用できます。詳細は snmpd.conf(5) の man ページを参照してください。

以下のシェルスクリプトの終了コードは、任意の時点におけるシステム上での実行中の **httpd** プロセス数です。

```
#!/bin/sh

NUMPIDS=pgrep httpd | wc -l

exit $NUMPIDS
```

SNMP 経由でこのスクリプトを利用できるようにするには、スクリプトをシステムパス上の場所にコピーし、実行可能なビットを設定して、**/etc/snmp/snmpd.conf** ファイルに **extend** ディレクティブを追加します。**extend** ディレクティブの形式は次のとおりです。

```
extend name prog args
```

-

*ここでのname は拡張するための識別文字列、prog は実行するプログラム、args はプログラムに渡す引数です。たとえば、上記のシェルスクリプトが **/usr/local/bin/check_apache.sh** にコピーされた場合、以下のディレクティブは SNMP ツリーにスクリプトを追加します。*

```
extend httpd_pids /bin/sh /usr/local/bin/check_apache.sh
```

*スクリプトは **NET-SNMP-EXTEND-MIB::nsExtendObjects** でクエリーできます。*

```
~]$ snmpwalk localhost NET-SNMP-EXTEND-MIB::nsExtendObjects
NET-SNMP-EXTEND-MIB::nsExtendNumEntries.0 = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendCommand."httpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendArgs."httpd_pids" = STRING:
/usr/local/bin/check_apache.sh
NET-SNMP-EXTEND-MIB::nsExtendInput."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendCacheTime."httpd_pids" = INTEGER: 5
NET-SNMP-EXTEND-MIB::nsExtendExecType."httpd_pids" = INTEGER: exec(1)
NET-SNMP-EXTEND-MIB::nsExtendRunType."httpd_pids" = INTEGER: run-on-read(1)
NET-SNMP-EXTEND-MIB::nsExtendStorage."httpd_pids" = INTEGER: permanent(4)
NET-SNMP-EXTEND-MIB::nsExtendStatus."httpd_pids" = INTEGER: active(1)
NET-SNMP-EXTEND-MIB::nsExtendOutput1Line."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendOutNumLines."httpd_pids" = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
NET-SNMP-EXTEND-MIB::nsExtendOutLine."httpd_pids".1 = STRING:
```

*注意する点は、終了コード (この例では 8) は INTEGER (整数) タイプとして、出力は STRING (文字列) タイプとして、示されていることです。複数のメトリックを整数として公開するには、**extend** ディレクティブを使用してスクリプトに異なる引数を指定します。例えば、以下のシェルスクリプトを使用すると、任意の文字列に一致するプロセスの数を見つけ出すことができ、プロセスの数を示すテキスト文字列も出力します。*

```
#!/bin/sh

PATTERN=$1
NUMPIDS=pgrep $PATTERN | wc -l

echo "There are $NUMPIDS $PATTERN processes."
exit $NUMPIDS
```

*次の **/etc/snmp/snmpd.conf** ディレクティブは、上記のスクリプトが **/usr/local/bin/check_proc.sh** にコピーされる時に **httpd** PID の数と併せて **snmpd** PID の数を与えます。*

```
extend httpd_pids /bin/sh /usr/local/bin/check_proc.sh httpd
extend snmpd_pids /bin/sh /usr/local/bin/check_proc.sh snmpd
```

*以下の例は、**nsExtendObjects** OID の **snmpwalk** の出力を示しています。*

```
~]$ snmpwalk localhost NET-SNMP-EXTEND-MIB::nsExtendObjects
NET-SNMP-EXTEND-MIB::nsExtendNumEntries.0 = INTEGER: 2
NET-SNMP-EXTEND-MIB::nsExtendCommand."httpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendCommand."snmpd_pids" = STRING: /bin/sh
NET-SNMP-EXTEND-MIB::nsExtendArgs."httpd_pids" = STRING: /usr/local/bin/check_proc.sh
httpd
```

```
NET-SNMP-EXTEND-MIB::nsExtendArgs."snmpd_pids" = STRING:
/usr/local/bin/check_proc.sh snmpd
NET-SNMP-EXTEND-MIB::nsExtendInput."httpd_pids" = STRING:
NET-SNMP-EXTEND-MIB::nsExtendInput."snmpd_pids" = STRING:
...
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
NET-SNMP-EXTEND-MIB::nsExtendResult."snmpd_pids" = INTEGER: 1
NET-SNMP-EXTEND-MIB::nsExtendOutLine."httpd_pids".1 = STRING: There are 8 httpd
processes.
NET-SNMP-EXTEND-MIB::nsExtendOutLine."snmpd_pids".1 = STRING: There are 1 snmpd
processes.
```

警告

整数の終了コードは 0 から 255 の範囲に制限されています。256 を超える可能性がある値については、スクリプトの標準出力 (文字列として入力されるもの) を使用するか、エージェントを拡張するという別の方法を実行してください。

この最後の例では、システムの空きメモリーと **httpd** プロセスの数のクエリーを示しています。このクエリーは、パフォーマンステスト中にメモリー負担に与えるプロセス数の影響を知るために使用することができます。

```
~]$ snmpget localhost \
  'NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids"' \
  UCD-SNMP-MIB::memAvailReal.0
NET-SNMP-EXTEND-MIB::nsExtendResult."httpd_pids" = INTEGER: 8
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 799664 kB
```

21.7.5.2. Perl による Net-SNMP の拡張

extend ディレクティブを使用したシェルスクリプトの実行は、ＳＮＭＰ でカスタムアプリケーションメトリックを公開する非常に限定的な方法です。Net-SNMP エージェントは、カスタムオブジェクトを公開するための埋め込み Perl インターフェイスも提供します。Optional チャンネルの net-snmp-perl パッケージは、Red Hat Enterprise Linux で組み込み Perl プラグインの作成に使用される **NetSNMP::agent** Perl モジュールを提供します。

注記

Optional および Supplementary チャンネルをサブスクライブする前に、対象範囲の詳細を参照してください。これらのチャンネルからパッケージをインストールする場合は、Red Hat カスタマーポータルの記事 証明書ベースの管理を使用して、Optional および Supplementary チャンネル、-devel パッケージにアクセスする方法 で説明されている手順を行ってください。

NetSNMP::agent Perl モジュールは、エージェントの OID ツリーの一部に対する要求を処理するために使用される **agent** オブジェクトを提供します。**agent** オブジェクトのコンストラクターには、エージェントを **snmpd** のサブエージェントまたはスタンドアロンエージェントとして実行するためのオプションがあります。埋め込みエージェントを作成するために必要な引数はありません。

```
use NetSNMP::agent (':all');

my $agent = new NetSNMP::agent();
```

*agent オブジェクトには、コールバック関数を特定の OID に登録するために使用される **register** メソッドがあります。**register** 関数は、名前、OID、コールバック関数へのポインターを取ります。以下の例は、**hello_handler** と呼ばれるコールバック関数を OID **.1.3.6.1.4.1.8072.9999.9999** で要求を処理する SNMP Agent に登録しています:*

```
$agent->register("hello_world", ".1.3.6.1.4.1.8072.9999.9999",
        \&hello_handler);
```

注記

*通常、OID **.1.3.6.1.4.1.8072.9999.9999** (**NET-SNMP-MIB::netSnmpPlaypen**) は表示目的のみに使用されます。お客様の組織に root OID がない場合は、ISO Name Registration Authority (米国では ANSI) にご連絡いただくと取得できます。*

*ハンドラー関数は、**HANDLER**、**REGISTRATION_INFO**、**REQUEST_INFO**、**REQUESTS** の 4 つのパラメーターで呼び出されます。**REQUESTS** パラメーターには、現在の呼び出しの要求リストが含まれており、反復されデータが追加されるはずです。リストの **request** オブジェクトには get 及び set メソッドがあるため、要求の **OID** と **value** を操作することができます。例として、以下の呼び出しは hello world という文字列に要求オブジェクトの値を設定します。*

```
$request->setValue(ASN_OCTET_STR, "hello world");
```

*ハンドラー関数は、GET 要求と GETNEXT 要求という 2 種類の SNMP 要求に応答できます。要求のタイプは、ハンドラー関数に第 3 のパラメーターとして渡される **request_info** オブジェクトの **getMode** メソッドを呼び出すことによって、決定されます。要求が GET 要求である場合、呼び出し元は、ハンドラーに要求の OID に応じて **request** オブジェクトの **value** を設定するよう求めます。要求が GETNEXT 要求である場合、呼び出し元は、ハンドラーに要求の OID をツリー内で次に利用可能な OID に設定するよう求めます。以下のコードは、この例を示しています。*

```
my $request;
my $string_value = "hello world";
my $integer_value = "8675309";

for($request = $requests; $request; $request = $request->next()) {
 my $oid = $request->getOID();
 if ($request_info->getMode() == MODE_GET) {
  if ($oid == new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.0")) {
   $request->setValue(ASN_OCTET_STR, $string_value);
  }
  elsif ($oid == new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.1")) {
   $request->setValue(ASN_INTEGER, $integer_value);
  }
 } elsif ($request_info->getMode() == MODE_GETNEXT) {
  if ($oid == new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.0")) {
   $request->setOID(".1.3.6.1.4.1.8072.9999.9999.1.1");
   $request->setValue(ASN_INTEGER, $integer_value);
  }
  elsif ($oid < new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.0")) {
   $request->setOID(".1.3.6.1.4.1.8072.9999.9999.1.0");
   $request->setValue(ASN_OCTET_STR, $string_value);
```

```
  }
 }
}
```

getMode が *MODE_GET* を返す場合、ハンドラーは *request* オブジェクトの *getOID* 呼び出しの値を分析します。*request* の *value* は、OID が *.1.0* で終わる場合は *string_value* に、OID が *.1.1* で終わる場合は *integer_value* に設定されます。*getMode* が *MODE_GETNEXT* を返す場合、ハンドラーは要求の OID が *.1.0* かどうかを見つけ出し、その後 *.1.1* に対する OID と値を設定します。ツリーで要求が *.1.0* より高い場合は、*.1.0* に対する OID と値が設定されます。実際、これはツリーの次の値を返すため、*snmpwalk* のようなプログラムは構造に関する事前知識なくツリーをトラバースできます。

変数のタイプは *NetSNMP::ASN* からの定数を使用して設定されます。利用可能な定数の全リストについては、*NetSNMP::ASN* の *perldoc* を参照して下さい。

この例の Perl プラグインのコード全リストは、以下のとおりです:

```
#!/usr/bin/perl

use NetSNMP::agent (':all');
use NetSNMP::ASN qw(ASN_OCTET_STR ASN_INTEGER);

sub hello_handler {
 my ($handler, $registration_info, $request_info, $requests) = @_;
 my $request;
 my $string_value = "hello world";
 my $integer_value = "8675309";

 for($request = $requests; $request; $request = $request->next()) {
  my $oid = $request->getOID();
  if ($request_info->getMode() == MODE_GET) {
   if ($oid == new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.0")) {
    $request->setValue(ASN_OCTET_STR, $string_value);
   }
   elsif ($oid == new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.1")) {
    $request->setValue(ASN_INTEGER, $integer_value);
   }
  } elsif ($request_info->getMode() == MODE_GETNEXT) {
   if ($oid == new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.0")) {
    $request->setOID(".1.3.6.1.4.1.8072.9999.9999.1.1");
    $request->setValue(ASN_INTEGER, $integer_value);
   }
   elsif ($oid < new NetSNMP::OID(".1.3.6.1.4.1.8072.9999.9999.1.0")) {
    $request->setOID(".1.3.6.1.4.1.8072.9999.9999.1.0");
    $request->setValue(ASN_OCTET_STR, $string_value);
   }
  }
 }
}

my $agent = new NetSNMP::agent();
$agent->register("hello_world", ".1.3.6.1.4.1.8072.9999.9999",
        \&hello_handler);
```

プラグインをテストするには、上記のプログラムを */usr/share/snmp/hello_world.pl* にコピーして、以下の行を */etc/snmp/snmpd.conf* の設定ファイルに追加して下さい:

```
perl do "/usr/share/snmp/hello_world.pl"
```

新しい Perl プラグインをロードするためには、SNMP Agent Daemon を再起動する必要があります。再起動すると、**snmpwalk** は新しいデータを返すはずです。

```
~]$ snmpwalk localhost NET-SNMP-MIB::netSnmpPlaypen
NET-SNMP-MIB::netSnmpPlaypen.1.0 = STRING: "hello world"
NET-SNMP-MIB::netSnmpPlaypen.1.1 = INTEGER: 8675309
```

snmpget を使用すると、ハンドラーの他のモードを使用することも可能です:

```
~]$ snmpget localhost \
  NET-SNMP-MIB::netSnmpPlaypen.1.0 \
  NET-SNMP-MIB::netSnmpPlaypen.1.1
NET-SNMP-MIB::netSnmpPlaypen.1.0 = STRING: "hello world"
NET-SNMP-MIB::netSnmpPlaypen.1.1 = INTEGER: 8675309
```

21.8. 関連情報

システム情報の収集方法についてさらに知るには、以下のリソースを参照してください。

21.8.1. インストールされているドキュメント

- lscpu*(1)*: **lscpu** コマンドの man ページです。
- lsusb*(8)*: **lsusb** コマンドの man ページです。
- findmnt*(8)*: **findmnt** コマンドの man ページです。
- blkid*(8)*: **blkid** コマンドの man ページです。
- lsblk*(8)*: **lsblk** コマンドの man ページです。
- ps*(1)*: **ps** コマンドの man ページです。
- top*(1)*: **top** コマンドの man ページです。
- free*(1)*: **free** コマンドの man ページです。
- df*(1)*: **df** コマンドの man ページ。
- du*(1)*: **du** コマンドの man ページです。
- lspci*(8)*: **lspci** コマンドの man ページです。
- snmpd*(8)*: **snmpd** サービスの man ページです。
- snmpd.conf*(5)*: 利用可能な設定ディレクティブについての全ドキュメントを含む **/etc/snmp/snmpd.conf** ファイルの man ページです。

第22章 OPENLMI

Open Linux Management Infrastructure は、通常OpenLMI と短縮形で呼ばれる、Linux システムを管理する一般的なインフラストラクチャーです。これは既存のツール上に構築され、システム管理者から基礎となるシステムの複雑性を隠すために抽出化レイヤーとして機能します。OpenLMI には、ローカルおよびリモートでのアクセスが可能なサービス一式が含まれており、ハードウェア、オペレーティングシステム、システムサービスの管理および監視に使用できる複数言語のバインディング、標準 API、および標準スクリプトインターフェイスを提供します。

22.1. OPENLMI の概要

OpenLMI は、物理マシンおよび仮想マシンの両方で Red Hat Enterprise Linux システムを実行している実稼働サーバーに共通の管理インターフェイスを提供するように設計されています。以下の 3 つのコンポーネントで設定されています。

1. **システム管理エージェント**: *このエージェントは管理されるシステムにインストールされ、標準オブジェクトブローカーに提示されるオブジェクトモデルを実装します。OpenLMI に実装される最初のエージェントにはストレージおよびネットワークの設定が含まれますが、その後の作業がシステム管理の追加要素を処理します。システム管理エージェントは、*Common Information Model **プロバイダー** *または*CIM **プロバイダー** *と呼ばれます。*

2. **標準オブジェクトブローカー**: *オブジェクトブローカーはシステム管理エージェントを管理し、インターフェイスを提供します。標準オブジェクトブローカーは、*CIM **オブジェクトモニター** *または*CIMOM *とも呼ばれます。*

3. **クライアントアプリケーションおよびスクリプト**: *クライアントアプリケーションおよびスクリプトは、標準オブジェクトブローカーでシステム管理エージェントを呼び出します。*

OpenLMI プロジェクトは、スクリプトまたはシステム管理コンソールで使用可能な低レベルのインターフェイスを提供することで、既存の管理イニシアチブを補完します。OpenLMI とともに配布されるインターフェイスには、C、C++、Python、Java、およびイニシアチブコマンドラインクライアントが含まれており、これらすべてが各エージェントで実装されている機能に同一の完全なアクセスを提供します。これにより、どのプログラミングインターフェイスを使用していても、まったく同一の機能に常にアクセスできることが保証されています。

22.1.1. 主な特長

以下は、OpenLMI をシステムにインストールして使用する主な利点です。

- *OpenLMI は、ローカルおよびリモートのシステムの設定、管理、モニタリングのための標準インターフェイスを提供します。*
- *物理および仮想の両方のマシン上の実稼働サーバーの設定、管理、監視ができるようになります。*
- *CIM プロバイダーのコレクションがともに配布され、ストレージデバイスおよび複雑なネットワークの設定、管理、監視が可能になります。*
- *C、C++、Python、および Java プログラムからシステム管理機能を呼び出すことが可能で、コマンドラインインターフェイスを提供する LMIShell も含まれます。*
- *オープンな業界標準に基づく無料ソフトウェアです。*

22.1.2. 管理機能

OpenLMI の主な機能には、ストレージデバイス、ネットワーク、システムサービス、ユーザーアカウントの管理、ハードウェアおよびソフトウェアの設定、電源管理、Active Directory との相互作用などがあります。Red Hat Enterprise Linux 7 で配布される CIM プロバイダーの完全リストは、表22.1「利用可能な CIM プロバイダー」 を参照してください。

表22.1 利用可能な CIM プロバイダー

パッケージ名	詳細
openlmi-account	ユーザーアカウント管理用の CIM プロバイダー。
openlmi-logicalfile	ファイルおよびディレクトリー読み取り用の CIM プロバイダー。
openlmi-networking	ネットワーク管理用の CIM プロバイダー。
openlmi-powermanagement	電源管理用の CIM プロバイダー。
openlmi-service	システムサービス管理用の CIM プロバイダー。
openlmi-storage	ストレージ管理用の CIM プロバイダー。
openlmi-fan	コンピューターファン制御用の CIM プロバイダー。
openlmi-hardware	ハードウェア情報取得用の CIM プロバイダー。
openlmi-realmd	realmd 設定用の CIM プロバイダー。
openlmi-software[a]	ソフトウェア管理用の CIM プロバイダー。

[a] Red Hat Enterprise Linux 7 では、OpenLMI Software プロバイダーは テクノロジープレビュー と提供されています。このプロバイダーは完全に機能するものですが、多くのソフトウェアパッケージをリスト化する際にメモリーと時間が過剰に消費されるという既知のパフォーマンススケーリング問題があります。この問題を回避するには、パッケージ検索ができるだけ少ないパッケージを返すように調整します。

22.2. OPENLMI のインストール

OpenLMI は RPM パッケージのコレクションとして配布され、これには CIMOM、個別の CIM プロバイダー、およびクライアントアプリケーションが含まれます。これにより、管理システムとクライアントシステムの区別ができ、必要なコンポーネントのみをインストールできるようになります。

22.2.1. 管理システムへの OpenLMI のインストール

管理システム *は、OpenLMI クライアントツールを使用して監視および管理が可能なシステムです。管理システムに OpenLMI をインストールするには、以下のステップに従います。*

1. *シェルプロンプトで、**root** で以下のコマンドを入力し、**tog-pegasus** パッケージをインストールします。*

```
yum install tog-pegasus
```

これで OpenPegasus CIMOM とすべての依存関係がシステムにインストールされ、**pegasus** ユーザーのユーザーアカウントが作成されます。

2. 以下のコマンドを **root** で実行して、必要な CIM プロバイダーをインストールします。

```
yum install openlmi-{storage,networking,service,account,powermanagement}
```

これでストレージ、ネットワーク、アカウント、および電源管理用の CIM プロバイダーがインストールされます。Red Hat Enterprise Linux 7 で配布される CIM プロバイダーの完全リストは、表22.1「利用可能な CIM プロバイダー」 を参照してください。

3. **/etc/Pegasus/access.conf** 設定ファイルを編集して、OpenPegasus CIMOM への接続が許可されるユーザーのリストをカスタマイズします。デフォルトでは、**pegasus** ユーザーのみがリモートとローカルの両方で CIMOM にアクセスできます。このユーザーアカウントを有効にするには、**root** で以下のコマンドを実行して、ユーザーのパスワードを設定します。

```
passwd pegasus
```

4. **tog-pegasus.service** ユニットをアクティブにして OpenPegasus CIMOM を起動します。現行セッションで **tog-pegasus.service** ユニットをアクティブにするには、**root** でシェルプロンプトに以下を入力します。

```
systemctl start tog-pegasus.service
```

tog-pegasus.service が起動時に自動的に開始するように設定するには、**root** で以下を入力します。

```
systemctl enable tog-pegasus.service
```

5. リモートマシンから管理システムに対話するする場合は、TCP 通信をポート **5989** で有効にします (**wbem-https**)。現行セッションでこのポートを開くには、**root** で以下のコマンドを実行します。

```
firewall-cmd --add-port 5989/tcp
```

ポート **5989** を TCP 通信用に永続的に開いておくには、**root** で以下を入力します。

```
firewall-cmd --permanent --add-port 5989/tcp
```

これで、「LMIShell の使用」 の説明にあるように、管理システムに接続して OpenLMI クライアントツールを使用して対話できます。管理システムで直接 OpenLMI 操作を実行する場合は、「クライアントシステムへの OpenLMI のインストール」 で説明されている手順も完了してください。

22.2.2. クライアントシステムへの OpenLMI のインストール

管理システムと対話する基となるのが、**クライアントシステム** です。通常のシナリオでは、クライアントシステムと管理システムは別の 2 つのマシンにインストールされますが、管理システムにクライアントツールをインストールして、直接対話することも可能です。

クライアントシステムに OpenLMI をインストールするには、以下の手順を実行します。

1. シェルプロンプトで **root** として以下を入力し、openlmi-tools パッケージをインストールします。

■

```
yum install openlmi-tools
```

これにより、CIM オブジェクトにアクセスするためのインタラクティブクライアントおよびインタプリターである LMIShell と、そのすべての依存関係がシステムにインストールされます。LMIShell は OpenPegasus が提供します。

2. 「OpenPegasus 用に SSL 証明書を設定する」の説明に従って、OpenPegasus 用の SSL 証明書を設定します。

これで、「LMIShell の使用」で説明されているように、LMIShell クライアントを使用して管理システムと対話できるようになりました。

22.3. OPENPEGASUS 用に SSL 証明書を設定する

OpenLMI は、HTTP トランスポート層で機能する WBEM (Web-Based Enterprise Management) プロトコルを使用します。標準の HTTP ベーシック認証がこのプロトコルで実行されます。つまり、ユーザー名とパスワードがリクエストと共に送信されることになります。

認証を保護するには、OpenPegasus CIMOM を設定して通信に HTTPS を使用することが必要になります。管理システム上で暗号化チャンネルを確立するには、SSL (Secure Sockets Layer) 証明書または TLS (Transport Layer Security) 証明書が必要になります。

システムで SSL/TLS 証明書を管理するには、2 つの方法があります。

- 自己署名証明書では必要なインフラストラクチャーは少なくなりますが、クライアントへの導入および安全な管理はより難しくなります。
- 認証局が署名する証明書は、設定さえ行えばクライアントへの導入が容易になりですが、必要な初期投資が大きくなる可能性があります。

認証局が署名する証明書を使用する場合は、クライアントシステムで信頼できる認証局を設定する必要があります。その後に、その権限を使用して、管理システムのすべての CIMOM 証明書を署名できるようになります。証明書は証明書チェーンの一部となることもあるため、管理システムの証明書の署名に使用された証明書が別の、より高い水準の認証局 (Verisign、CAcert、RSA など多数の認証局) により署名される場合もあります。

ファイルシステムにおける、デフォルトの証明書とトラストストアの保存場所は、表22.2「証明書およびトラストストアの保存場所」となります。

表22.2 証明書およびトラストストアの保存場所

設定オプション	場所	詳細
sslCertificateFilePath	**/etc/Pegasus/server.pem**	CIMOM の公開証明書
sslKeyFilePath	**/etc/Pegasus/file.pem**	CIMOM にのみ知られている秘密鍵
sslTrustStore	**/etc/Pegasus/client.pem**	信頼できる証明書機関のリストを提供するファイルまたはディレクトリー

重要

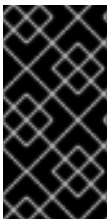
表22.2「証明書およびトラストストアの保存場所」にあるファイルを修正した場合は、**tog-pegasus** サービスを再起動して新たな証明書が確実に認識されるようにします。サービスを再起動するには、**root** でシェルプロンプトに以下のコマンドを入力します。

```
systemctl restart tog-pegasus.service
```

Red Hat Enterprise Linux 7 でシステムサービスを管理する方法は、10章systemd によるサービス管理 を参照してください。

22.3.1. 自己署名証明書の管理

自己署名証明書は、自身の秘密鍵を使用して署名し、その他の信頼チェーンには接続されません。管理システムでは、**tog-pegasus** サービスを初めて起動する前に管理者が証明書を提供していない場合は、システムのプライマリーホスト名を証明書の件名にした自己署名証明書のセットが自動的に生成されます。



重要

自動生成された自己署名証明書は、デフォルトで 10 年間有効ですが、自動的に更新する機能はありません。このような証明書を修正するには、OpenSSL または Mozilla NSS のドキュメンテーションが提供するガイドラインに従って、新たな証明書を手動で作成する必要があります。

クライアントシステムが自己署名証明書を信頼するように設定するには、以下の手順に従います。

1. 管理システムから、クライアントシステムの **/etc/pki/ca-trust/source/anchors/** ディレクトリーに、**/etc/Pegasus/server.pem** 証明書をコピーします。これを行うには、**root** で次のコマンドを実行します。

   ```
   scp root@hostname:/etc/Pegasus/server.pem /etc/pki/ca-trust/source/anchors/pegasus-hostname.pem
   ```

   hostname を管理システムのホスト名に置き換えます。このコマンドは、**sshd** サービスが管理システム上で実行中に、**root** ユーザーが SSH プロトコルを使用してシステムにログインできるような設定でのみ機能することに注意してください。**sshd** サービスをインストールして設定する方法、そして **scp** コマンドを使用して SSH プロトコルでファイルを送信する方法は、12章OpenSSH を参照してください。

2. チェックサムを元のファイルのものと比較して、クライアントシステムの証明書の整合性を検証します。管理システムの **/etc/Pegasus/server.pem** ファイルのチェックサムを計算するには、そのシステムで **root** として以下のコマンドを実行します。

   ```
   sha1sum /etc/Pegasus/server.pem
   ```

   クライアントシステムの **/etc/pki/ca-trust/source/anchors/pegasus-hostname.pem** ファイルのチェックサムを計算するには、このシステムで以下のコマンドを実行します。

   ```
   sha1sum /etc/pki/ca-trust/source/anchors/pegasus-hostname.pem
   ```

   hostname を管理システムのホスト名に置き換えます。

3. クライアントシステムでトラストストアを更新するには、**root** で以下のコマンドを実行します。

```
update-ca-trust extract
```

22.3.2. Identity Management を使用した、認証局が署名した証明書の管理 (推奨)

Red Hat Enterprise Linux の Identity Management 機能は、ドメインに参加したシステムの SSL 証明書管理を簡素化するドメインコントローラーを提供します。Identity Management サーバーは、埋め込み認証局を提供します。クライアントシステムおよび管理システムをドメインに参加させる方法は、*Red Hat Enterprise Linux 7 Linux ドメイン ID、認証、およびポリシーガイド* または FreeIPA のドキュメントを参照してください。

管理システムでは、Identity Management への登録が必須になります。クライアントシステムでは、登録はオプションになります。

管理システムでは、以下の手順が必要です。

1. *Red Hat Enterprise Linux 7 Linux ドメイン ID、認証、およびポリシーガイド* で説明されているように、**ipa-client** パッケージをインストールして、システムを Identity Management に登録します。

2. **root** で以下のコマンドを入力して、Identity Management の署名した証明書をトラストストアにコピーします。

```
cp /etc/ipa/ca.crt /etc/pki/ca-trust/source/anchors/ipa.crt
```

3. トラストストアを更新するには、**root** で以下のコマンドを実行します。

```
update-ca-trust extract
```

4. 権限のあるドメインユーザーとして以下のコマンドを実行して、Pegasus をサービスとして Identity Management ドメインに登録します。

```
ipa service-add CIMOM/hostname
```

hostname を管理システムのホスト名に置き換えます。

コマンドは、Identity Management ドメイン内のシステムで **ipa-admintools** パッケージがインストールされているものから実行できます。Identity Management にサービスエントリーが作成され、これを署名済み SSL 証明書の生成に使用できるようになります。

5. **/etc/Pegasus/** ディレクトリーにある PEM ファイルのバックアップを作成します (推奨)。

6. **root** で以下のコマンドを実行して、署名済み証明書を取得します。

```
ipa-getcert request -f /etc/Pegasus/server.pem -k /etc/Pegasus/file.pem -N
CN=hostname -K CIMOM/hostname
```

hostname を管理システムのホスト名に置き換えます。

これで証明書と鍵ファイルが正常な場所に保存されます。**ipa-client-install** スクリプトが管理システムにインストールする **certmonger** デーモンにより、証明書が必要に応じて最新に保たれ、更新されます。

詳細情報は、Red Hat Enterprise Linux 7 Linux ドメイン ID、認証、およびポリシーガイドを参照してください。

クライアントシステムを登録して、トラストストアを更新するには、以下のステップに従います。

1. Red Hat Enterprise Linux 7 Linux ドメイン ID、認証、およびポリシーガイドで説明されているように、**ipa-client** パッケージをインストールして、システムを Identity Management に登録します。
2. **root** で以下のコマンドを入力して、Identity Management の署名した証明書をトラストストアにコピーします。

```
cp /etc/ipa/ca.crt /etc/pki/ca-trust/source/anchors/ipa.crt
```

3. トラストストアを更新するには、**root** で以下のコマンドを実行します。

```
update-ca-trust extract
```

クライアントシステムを Identity Management に登録しない場合は、以下の手順に従ってトラストストアを更新します。

1. **root** として、同一の Identity Management ドメインに参加しているその他のシステムから、安全に **/etc/ipa/ca.crt** ファイルをトラストストア **/etc/pki/ca-trust/source/anchors/** ディレクトリーにコピーします。
2. トラストストアを更新するには、**root** で以下のコマンドを実行します。

```
update-ca-trust extract
```

22.3.3. 認証局が署名する証明書を手動で管理

認証局が署名する証明書を Identity Management 以外のメカニズムで管理するには、手動での設定が必要になります。

管理システムの証明書に署名する認証局の証明書を、すべてのクライアントが信頼するようにする必要があります。

- 認証局がデフォルトで信頼されている場合は、信頼させるのに必要な手順はありません。
- 認証局がデフォルトで信頼されていない場合は、クライアントシステムと管理システムで証明書をインポートする必要があります。
  a. **root** で以下のコマンドを入力して、証明書をトラストストアにコピーします。

```
cp /path/to/ca.crt /etc/pki/ca-trust/source/anchors/ca.crt
```

  b. トラストストアを更新するには、**root** で以下のコマンドを実行します。

```
update-ca-trust extract
```

管理システムで以下のステップを実行します。

1. 新たな SSL 設定ファイル **/etc/Pegasus/ssl.cnf** を作成し、証明書に関する情報を保管します。このファイルのコンテンツは、以下の例のようにする必要があります。

```
[ req ]
distinguished_name   = req_distinguished_name
prompt        = no
[ req_distinguished_name ]
C         = US
ST          = Massachusetts
L         = Westford
O         = Fedora
OU          = Fedora OpenLMI
CN          = hostname
```

hostname を、管理システムの完全修飾ドメイン名に置き換えます。

2. **root** で以下のコマンドを実行して、管理システムで秘密鍵を生成します。

```
openssl genrsa -out /etc/Pegasus/file.pem 1024
```

3. **root** で以下のコマンドを実行し、証明書の署名要求 (CSR) を生成します。

```
openssl req -config /etc/Pegasus/ssl.cnf -new -key /etc/Pegasus/file.pem -out
/etc/Pegasus/server.csr
```

4. **/etc/Pegasus/server.csr** を認証局に送信して署名します。ファイル提出に関する詳細な手順は、個々の認証局によって異なります。

5. 認証局から署名済み証明書を受け取ったら、**/etc/Pegasus/server.pem** として保存します。

6. **root** で以下のコマンドを実行し、信頼できる認証局の証明書を Pegasus トラストストアにコピーして、Pegasus が自らの証明書を信頼できるようにします。

```
cp /path/to/ca.crt /etc/Pegasus/client.pem
```

上記の手順をすべて完了したら、署名の認証局を信頼するクライアントが、管理サーバーの CIMOM と正常に通信できるようなります。

重要

Identity Management ソリューションとは異なり、証明書の有効期限が切れて更新が必要となった場合には、上記の手動の手順を再度実行する必要があります。証明書は有効期限が切れる前に更新することが推奨されます。

22.4. LMISHELL の使用

LMIShell はインタラクティブクライアントかつ非インタラクティブインタープリターで、OpenPegasus CIMOM が提供する CIM オブジェクトにアクセスするために使用できます。Python インタープリターに基づいていますが、CIM オブジェクトとの対話のための追加の関数およびクラスも実装します。

22.4.1. LMIShell の開始、使用、終了

Python インタープリターと同様に、LMIShell は LMIShell スクリプトでインタラクティブクライアントまたは非インタラクティブインタープリターとして使用できます。

インタラクティブモードでの LMIShell の開始

インタラクティブモードでLMIShell インタープリターを開始するには、**lmishell** コマンドを引数なしで実行します。

```
lmishell
```

デフォルトでは、LMIShell はCIMOM との接続確立を試みる際に、サーバー側の証明書を認証局のトラストストアに対して確認します。この検証を無効にするには、**--noverify** または**-n** コマンドラインオプションを指定して**lmishell** コマンドを実行します。

```
lmishell --noverify
```

Tab Completion (タブ入力) の使用
インタラクティブモードでの実行時には、LMIShell インタープリターでは **Tab** キーを使用した基本的なプログラミング構造体やCIM オブジェクトの入力が可能になります。これにはネームスペース、クラス、メソッド、およびオブジェクト属性が含まれます。

履歴の閲覧
デフォルトでは、LMIShell はユーザーがインタラクティブプロンプトで入力したすべてのコマンドを**~/.lmishell_history** ファイルに保存します。これにより、コマンド履歴が閲覧でき、インタラクティブモードで入力した行をプロンプトで再度入力することなく再使用できます。コマンド履歴で後ろに戻るには 上向き矢印 キーか**Ctrl+p** のキーの組み込み合わせを押します。コマンド履歴で前に進むには、下向き矢印 キーか**Ctrl+n** のキーの組み合わせを押します。

LMIShell は、増分逆検索もサポートします。コマンド履歴で特定の行を探すには、**Ctrl+r** を押してからコマンドの一部を入力します。以下に例を示します。

```
> (reverse-i-search)`connect': c = connect("server.example.com", "pegasus")
```

コマンド履歴を削除するには、以下のように**clear_history()** 機能を使用します。

```
clear_history()
```

~/.lmishellrc 設定ファイルの**history_length** オプションの値を変更することで、コマンド履歴に保存されている行数を設定できます。さらに、この設定ファイルの**history_file** オプションの値を変更することにより、履歴ファイルの場所を変更できます。たとえば、履歴ファイルの場所を**~/.lmishell_history** に設定し、最大**1000** 行を保存するようにLMIShell を設定するには、以下の行を**~/.lmishellrc** ファイルに追加します。

```
history_file = "~/.lmishell_history"
history_length = 1000
```

例外処理
デフォルトでは、LMIShell インタープリターはすべての例外を処理し、戻り値を使用します。すべての例外をコードで処理するためにこの動作を無効にするには、以下のように**use_exceptions()** 関数を使用します。

```
use_exceptions()
```

自動例外処理を再度有効にするには、以下を使用します。

```
use_exception(False)
```

*~/.lmishellrc 設定ファイルの **use_exceptions** オプションの値を **True** に変更すると、例外処理を永続的に無効にできます。*

```
use_exceptions = True
```

一時的なキャッシュの設定
*デフォルト設定では、LMIShell 接続オブジェクトは一時的なキャッシュを使用して CIM クラス名および CIM クラスを保存し、ネットワーク通信を減らします。この一時的なキャッシュを削除するには、以下のように **clear_cache()** メソッドを使用します。*

```
object_name.clear_cache()
```

object_name *を接続オブジェクトの名前に置き換えます。*

*特定の接続オブジェクトの一時的なキャッシュを無効にするには、以下のように **use_cache()** メソッドを使用します。*

```
object_name.use_cache(False)
```

再度これを有効にするには、以下を使用します。

```
object_name.use_cache(True)
```

*~/.lmishellrc 設定ファイルの **use_cache** オプションの値を **False** に変更すると、接続オブジェクトの一時キャッシュを永続的に無効にできます。*

```
use_cache = False
```

LMIShell の終了
*LMIShell インタープリターを終了してシェルプロンプトに戻るには、**Ctrl+d** のキーの組み合わせを押すか、以下のように **quit()** 関数を発行します。*

```
> quit()
~]$
```

LMIShell スクリプトの実行
*LMIShell スクリプトを実行するには、以下のように **lmishell** コマンドを実行します。*

```
lmishell file_name
```

file_name *をスクリプト名に置き換えます。実行後に └Shell スクリプトを検証するには、**--interact** または **-i** コマンドラインオプションも指定します。*

```
lmishell --interact file_name
```

*LMIShell スクリプトのファイル拡張子は、**.lmi** が適切です。*

22.4.2. CIMOM への接続

LMIShell では、同一システムにローカルで実行している CIMOM、またはネットワーク経由でアクセス可能なリモートマシンの CIMOM に接続できます。

リモートの CIMOM への接続

*リモートの CIMOM が提供する CIM オブジェクトにアクセスするには、以下のように **connect()** 関数を使用して接続オブジェクトを作成します。*

> ***connect(*host_name, user_name, password*)***

host_name *を管理システムのホスト名に、*user_name *をそのシステム上で実行中の OpenPegasus CIMOM に接続許可されているユーザー名に、*password *をユーザーのパスワードに置き換えます。パスワードを省略すると、LMIShell によりパスワードの入力が求められます。この関数は **LMIConnection** オブジェクトを返します。*

> *例22.1 リモートの CIMOM への接続*
>
> ***server.example.com** 上で実行中の OpenPegasus CIMOM にユーザー**pegasus** として接続するには、インタラクティブプロンプトに以下のコマンドを入力します。*
>
> ```
> > c = connect("server.example.com", "pegasus")
> password:
> >
> ```

ローカルの CIMOM への接続
*LMIShell では、Unix ソケットを使用することでローカルの CIMOM に接続できます。この種の接続では、LMIShell インタープリターを **root** ユーザーとして実行し、**/var/run/tog-pegasus/cimxml.socket** ソケットが存在する必要があります。*

*ローカルの CIMOM が提供する CIM オブジェクトにアクセスするには、以下のように **connect()** 関数を使用して接続オブジェクトを作成します。*

> ***connect(*host_name*)***

host_name *を **localhost**、**127.0.0.1**、または **::1** のいずれかで置き換えます。関数は **LMIConnection** オブジェクトか **None** を返します。*

> *例22.2 ローカルの CIMOM への接続*
>
> ***localhost** 上で実行中の OpenPegasus CIMOM に**root** ユーザーとして接続するには、インタラクティブプロンプトに以下のコマンドを入力します。*
>
> ```
> > c = connect("localhost")
> >
> ```

CIMOM への接続の確認
***connect()** 関数は、**LMIConnection** オブジェクト、または接続が確立されないと **None** を返します。さらに、**connect()** 関数が接続確立に失敗すると、標準エラー出力にエラーメッセージを出力します。*

*CIMOM への接続が正常に確立されたことを確認するには、以下のように **isinstance()** 関数を使用します。*

> ***isinstance(*object_name, LMIConnection*)***

object_name *を接続オブジェクト名に置き換えます。この関数は、**object_name** が*LMIConnection *オブジェクトの場合は **True** を、それ以外の場合は **False** を返します。*

例22.3 CIMOM への接続の確認

例22.1「リモートの CIMOM への接続」 で作成した ***c*** *変数に* ***LMIConnection*** *オブジェクトが含まれていることを確認するには、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> isinstance(c, LMIConnection)
True
>
```

または、 ***c*** *が* ***None*** *でないことを確認することもできます。*

```
> c is None
False
>
```

22.4.3. ネームスペースの使用

LMIShell ネームスペースは、利用可能なクラスを組織化する自然な方法を提供し、他のネームスペースおよびクラスへの階層的なアクセスポイントとして機能します。 ***root*** *ネームスペースは、接続オブジェクトの最初のエントリーポイントです。*

利用可能なネームスペースのリスト表示
利用可能なネームスペースのリストを表示するには、以下のように ***print_namespaces()*** *メソッドを使用します。*

```
object_name.print_namespaces()
```

object_name *を、検査するオブジェクトの名前に置き換えます。このメソッドは、利用可能なネームスペースを標準出力に出力します。*

利用可能なネームスペースリストを取得するには、オブジェクト属性 ***namespaces*** *にアクセスします。*

```
object_name.namespaces
```

これは文字列のリストを返します。

例22.4 利用可能なネームスペースのリスト表示

例22.1「リモートの CIMOM への接続」 で作成した ***c*** *接続オブジェクトの* ***root*** *ネームスペースオブジェクトを検査するには、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> c.root.print_namespaces()
cimv2
interop
PG_InterOp
PG_Internal
>
```

このネームスペースのリストを変数 ***root_namespaces*** *に割り当てるには、以下を入力します。*

```
> root_namespaces = c.root.namespaces
>
```

ネームスペースオブジェクトへのアクセス
特定のネームスペースオブジェクトにアクセスするには、以下の構文を使用します。

```
object_name.namespace_name
```

object_name *を検査するオブジェクト名に置き換え、*namespace_name *をアクセスするネームスペース名に置き換えます。これは* ***LMINamespace*** *オブジェクトを返します。*

例22.5 ネームスペースオブジェクトへのアクセス

例22.1「リモートの CIMOM への接続」で作成した ***c*** *接続オブジェクトの* ***cimv2*** *ネームスペースにアクセスするには、* ***ns*** *という名前の変数に割り当て、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> ns = c.root.cimv2
>
```

22.4.4. クラスの使用

LMIShell クラスは、CIMOM が提供するクラスを表します。これらのプロパティー、メソッド、インスタンス、インスタンス名、および ValueMap プロパティーにアクセスしたりこれらをリスト表示したりでき、ドキュメンテーションストリングを出力したり、新たなインスタンスやインスタンス名を作成できます。

利用可能なクラスのリスト表示
特定のネームスペースで利用可能なクラスをリスト表示するには、以下のように ***print_classes()*** *メソッドを使用します。*

```
namespace_object.print_classes()
```

namespace_object *を検査するネームスペースオブジェクトに置き換えます。このメソッドは、利用可能なクラスを標準出力に出力します。*

利用可能なクラスリストを取得するには、 ***classes()*** *メソッドを使用します。*

```
namespace_object.classes()
```

このメソッドは文字列のリストを返します。

例22.6 利用可能なクラスのリスト表示

例22.5「ネームスペースオブジェクトへのアクセス」で作成した ***ns*** *ネームスペースオブジェクトを検査して利用可能なクラスをリスト表示するには、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> ns.print_classes()
CIM_CollectionInSystem
```

```
CIM_ConcreteIdentity
CIM_ControlledBy
CIM_DeviceSAPImplementation
CIM_MemberOfStatusCollection
...
>
```

このクラスのリストを変数 ***cimv2_classes*** *に割り当てるには、以下を入力します。*

```
> cimv2_classes = ns.classes()
>
```

クラスオブジェクトへのアクセス
CIMOM が提供する特定のクラスオブジェクトにアクセスするには、以下の構文を使用します。

```
namespace_object.class_name
```

namespace_object を、*検査するネームスペースオブジェクト名にに置き換え、*class_name を、*アクセスするクラス名に置き換えます。*

例22.7 クラスオブジェクトへのアクセス

例22.5「ネームスペースオブジェクトへのアクセス」で作成した ***ns*** *ネームスペースオブジェクトの* ***LMI_IPNetworkConnection*** *クラスにアクセスし、これを変数* ***cls*** *に割り当てるには、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> cls = ns.LMI_IPNetworkConnection
>
```

クラスオブジェクトの検査
すべてのクラスオブジェクトには、その名前と所属するネームスペースに関する情報、および詳細なクラスのドキュメンテーションが保存されています。特定のクラスオブジェクトの名前を取得するには、以下の構文を使用します。

```
class_object.classname
```

class_object を、*検査するクラスオブジェクト名に置き換えます。これは、オブジェクト名を表す文字列を返します。*

クラスオブジェクトが所属するネームスペースに関する情報を取得するには、以下を使用します。

```
class_object.namespace
```

これは、名前空間を表す文字列を返します。

詳細なクラスのドキュメンテーションを表示するには、以下のように ***doc()*** *メソッドを使用します。*

```
class_object.doc()
```

例22.8 クラスオブジェクトの検査

例22.7「クラスオブジェクトへのアクセス」で作成した**cls**クラスオブジェクトを検査して、その名前と対応するネームスペースを表示するには、インタラクティブプロンプトに以下のコマンドを入力します。

```
> cls.classname
'LMI_IPNetworkConnection'
> cls.namespace
'root/cimv2'
>
```

クラスのドキュメンテーションにアクセスするには、以下を入力します。

```
> cls.doc()
Class: LMI_IPNetworkConnection
  SuperClass: CIM_IPNetworkConnection
  [qualifier] string UMLPackagePath: 'CIM::Network::IP'

  [qualifier] string Version: '0.1.0'
...
```

利用可能なメソッドのリスト表示
特定のクラスオブジェクトで利用可能なメソッドのリストを表示するには、以下のように**print_methods()**メソッドを使用します。

```
class_object.print_methods()
```

class_objectを、検査するクラスオブジェクト名に置き換えます。このメソッドは、利用可能なメソッドを標準出力に出力します。

利用可能なメソッドリストを取得するには、**methods()**メソッドを使用します。

```
class_object.methods()
```

このメソッドは文字列のリストを返します。

例22.9 利用可能なメソッドのリスト表示

例22.7「クラスオブジェクトへのアクセス」で作成した**cls**クラスオブジェクトを検査して利用可能なメソッドのリストを表示するには、インタラクティブプロンプトに以下のコマンドを入力します。

```
> cls.print_methods()
RequestStateChange
>
```

このメソッドのリストを変数**service_methods**に割り当てるには、以下を入力します。

```
> service_methods = cls.methods()
>
```

利用可能なプロパティーのリスト表示

特定のクラスオブジェクトで利用可能なプロパティーのリストを表示するには、以下のように ***print_properties()*** *メソッドを使用します。*

```
class_object.print_properties()
```

class_object *を、検査するクラスオブジェクト名に置き換えます。このメソッドは、利用可能なプロパティーを標準出力に出力します。*

利用可能なプロパティーリストを取得するには、 ***properties()*** *メソッドを使用します。*

```
class_object.properties()
```

このメソッドは文字列のリストを返します。

例22.10 利用可能なプロパティーのリスト表示

例22.7「クラスオブジェクトへのアクセス」 で作成した ***cls*** *クラスオブジェクトを検査して利用可能なすべてのプロパティーのリストを表示するには、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> cls.print_properties()
RequestedState
HealthState
StatusDescriptions
TransitioningToState
Generation
...
>
```

このクラスリストを変数 ***service_properties*** *に割り当てるには、以下を入力します。*

```
> service_properties = cls.properties()
>
```

ValueMap プロパティーのリスト表示と閲覧
*CIM クラスには、Managed Object Format (*MOF*) 定義でValueMap プロパティー が含まれる場合があります。ValueMap プロパティーには定数値が含まれ、これはメソッドを呼び出す場合や戻り値をチェックする場合に便利なものです。*

特定のクラスオブジェクトの利用可能なValueMap プロパティーのリストを表示するには、以下のように ***print_valuemap_properties()*** *メソッドを使用します。*

```
class_object.print_valuemap_properties()
```

class_object *を、検査するクラスオブジェクト名に置き換えます。このメソッドは、利用可能なValueMap プロパティーを標準出力に出力します。*

利用可能なValueMap プロパティーのリストを取得するには、 ***valuemap_properties()*** *メソッドを使用します。*

```
class_object.valuemap_properties()
```

このメソッドは文字列のリストを返します。

> *例22.11 ValueMap プロパティーのリスト表示*
>
> *例22.7「クラスオブジェクトへのアクセス」で作成した* ***cls*** *クラスオブジェクトを検査して、利用可能な ValueMap プロパティーのリストを表示するには、インタラクティブプロンプトに以下のコマンドを入力します。*
>
> ```
> > cls.print_valuemap_properties()
> RequestedState
> HealthState
> TransitioningToState
> DetailedStatus
> OperationalStatus
> ...
> >
> ```
>
> *この ValueMap プロパティーリストを変数* ***service_valuemap_properties*** *に割り当てるには、以下を入力します。*
>
> ```
> > service_valuemap_properties = cls.valuemap_properties()
> >
> ```

特定の ValueMap プロパティーにアクセスするには、以下の構文を使用します。

```
class_object.valuemap_propertyValues
```

valuemap_property を、*アクセスする ValueMap プロパティー名に置き換えます。*

利用可能な定数値のリストを表示するには、以下のように ***print_values()*** *メソッドを使用します。*

```
class_object.valuemap_propertyValues.print_values()
```

このメソッドは、名前の付いた利用可能な定数値を標準出力に出力します。また、 ***values()*** *メソッドを使用して利用可能な定数値のリストを取得することもできます。*

```
class_object.valuemap_propertyValues.values()
```

このメソッドは文字列のリストを返します。

> *例22.12 ValueMap プロパティーへのアクセス*
>
> *例22.11「ValueMap プロパティーのリスト表示」では、ValueMap プロパティー* ***RequestedState*** *について説明しました。このプロパティーを検査して、利用可能な定数値のリストを表示するには、インタラクティブプロンプトに以下のコマンドを入力します。*
>
> ```
> > cls.RequestedStateValues.print_values()
> Reset
> NoChange
> NotApplicable
> Quiesce
> ```

```
Unknown
...
>
```

この定数値のリストを変数 **requested_state_values** に割り当てるには、以下を入力します。

```
> requested_state_values = cls.RequestedStateValues.values()
>
```

特定の定数値にアクセスするには、以下の構文を使用します。

```
class_object.valuemap_propertyValues.constant_value_name
```

constant_value_name を、定数値の名前を置き換えます。また、以下のように **value()** メソッドを使用することもできます。

```
class_object.valuemap_propertyValues.value("constant_value_name")
```

特定の定数値の名前を確認するには、**value_name()** メソッドを使用します。

```
class_object.valuemap_propertyValues.value_name("constant_value")
```

このメソッドは文字列を返します。

例22.13 定数値へのアクセス

例22.12「ValueMap プロパティーへのアクセス」 では、**RequestedState** プロパティーが定数値 **Reset** を提供していました。この名前の定数値にアクセスするには、インタラクティブプロンプトに以下のコマンドを入力します。

```
> cls.RequestedStateValues.Reset
11
> cls.RequestedStateValues.value("Reset")
11
>
```

この定数値の名前を確認するには、以下を入力します。

```
> cls.RequestedStateValues.value_name(11)
u'Reset'
>
```

CIMClass オブジェクトの取り込み
クラスメソッドの多くは **CIMClass** オブジェクトへのアクセスを必要としません。これは、呼び出されたメソッドが実際に必要とするときにのみ、LMIShell が CIMOM からオブジェクトを取得するためです。**CIMClass** オブジェクトを手動で取り込むには、以下のように **fetch()** メソッドを使用します。

```
class_object.fetch()
```

class_object を、クラスオブジェクト名に置き換えます。**CIMClass** オブジェクトへのアクセスを必要とするメソッドは、これを自動的に取り込むことに注意してください。

22.4.5. インスタンスの使用

LMIShell インスタンスは、*CIMOM* が提供するインスタンスを表します。プロパティーの取得や設定、メソッドのリスト表示や呼び出し、ドキュメンテーション文字列の表示、関連オブジェクトリストの取得、修正されたオブジェクトの *CIMOM* へのプッシュ、*CIMOM* から個別インスタンスの削除が行えます。

インスタンスへのアクセス
特定のクラスオブジェクトの利用可能なインスタンスをすべて取得するには、以下のように **instances()** メソッドを使用します。

```
class_object.instances()
```

class_object を、検査するクラスオブジェクト名に置き換えます。このメソッドは、**LMIInstance** オブジェクトのリストを返します。

クラスオブジェクトの最初のインスタンスにアクセスするには、**first_instance()** メソッドを使用します。

```
class_object.first_instance()
```

このメソッドは、**LMIInstance** オブジェクトを返します。

instances() および **first_instance()** は、インスタンスのリストを表示したり、最初のインスタンスを返したりするほかにも、オプションの引数をサポートして結果をフィルターすることもできます。

```
class_object.instances(criteria)
```

```
class_object.first_instance(criteria)
```

criteria を、鍵と値のペアで設定される辞書に置き換えます。この鍵はインスタンスプロパティーを表し、値はこのプロパティーの必要な値を表します。

例22.14 インスタンスへのアクセス

例22.7「クラスオブジェクトへのアクセス」 で作成した **cls** クラスオブジェクトの最初のインスタンスで、**eth0** と同等の **ElementName** プロパティーがあるものを見つけ、これを変数 **device** に割り当てるには、インタラクティブプロンプトに以下のコマンドを入力します。

```
> device = cls.first_instance({"ElementName": "eth0"})
>
```

インスタンスの検査
すべてのインスタンスオブジェクトには、そのクラス名と所属するネームスペースに関する情報、およびプロパティーと値に関する詳細なドキュメンテーションが保存されています。さらに、インスタンスオブジェクトを使用すると、一意の *ID* オブジェクトを検索できます。

特定のインスタンスオブジェクトのクラス名を取得するには、以下の構文を使用します。

instance_object.*classname*

instance_object を、検査するインスタンスオブジェクト名に置き換えます。これは、クラス名を表す文字列を返します。

インスタンスオブジェクトが所属するネームスペースに関する情報を取得するには、以下を使用します。

instance_object.*namespace*

これは、名前空間を表す文字列を返します。

インスタンスオブジェクト用に一意の ID オブジェクトを検索するには、以下を使用します。

instance_object.*path*

これは、**LMIInstanceName** オブジェクトを返します。

最後に、詳細なドキュメンテーションを表示するには、以下のように **doc()** メソッドを使用します。

instance_object.*doc()*

例22.15 インスタンスの検査

例22.14「インスタンスへのアクセス」 で作成した **device** インスタンスオブジェクトを検査して、そのクラス名と対応するネームスペースを表示するには、インタラクティブプロンプトに以下のコマンドを入力します。

```
> device.classname
u'LMI_IPNetworkConnection'
> device.namespace
'root/cimv2'
>
```

インスタンスオブジェクトのドキュメンテーションにアクセスするには、以下を入力します。

```
> device.doc()
Instance of LMI_IPNetworkConnection
  [property] uint16 RequestedState = '12'

  [property] uint16 HealthState

  [property array] string [] StatusDescriptions
...
```

新規インスタンスの作成
いくつかの CIM プロバイダーでは、特定のクラスオブジェクトの新規インスタンスを作成できます。クラスオブジェクトの新規インスタンスを作成するには、以下のように **create_instance()** メソッドを使用します。

class_object.*create_instance*(properties*)*

class_object を、クラスオブジェクトとproperties の名前を、キーと値のペアで設定されるディクショナリーに置き換えます。キーはインスタンスのプロパティーと値はプロパティー値を表します。このメソッドは、**LMIInstance** オブジェクトを返します。

> *例22.16 新規インスタンスの作成*
>
> ***LMI_Group*** *クラスはシステムグループを表し、* ***LMI_Account*** *クラスは管理システム上のユーザーアカウントを表します。例22.5「ネームスペースオブジェクトへのアクセス」 で作成した* ***ns*** *ネームスペースオブジェクトを使用して、システムグループ* ***pegasus*** *と、ユーザー* ***lmishell-user*** *用にこれら2 つのクラスのインスタンスを作成し、そのインスタンスを* ***group*** *変数および* ***user*** *変数に割り当てるには、インタラクティブプロンプトに以下のコマンドを入力します。*
>
> ```
> > group = ns.LMI_Group.first_instance({"Name" : "pegasus"})
> > user = ns.LMI_Account.first_instance({"Name" : "lmishell-user"})
> >
> ```
>
> ***lmishell-user*** *ユーザー用に* ***LMI_Identity*** *クラスのインスタンスを取得するには、以下を入力します。*
>
> ```
> > identity = user.first_associator(ResultClass="LMI_Identity")
> >
> ```
>
> ***LMI_MemberOfGroup*** *クラスは、システムグループのメンバーシップを表します。* ***LMI_MemberOfGroup*** *クラスを使用して* ***lmishell-user*** *を* ***pegasus*** *グループに追加するには、以下のようにこのクラスの新規インスタンスを作成します。*
>
> ```
> > ns.LMI_MemberOfGroup.create_instance({
> ...   "Member" : identity.path,
> ...   "Collection" : group.path})
> LMIInstance(classname="LMI_MemberOfGroup", ...)
> >
> ```

個別インスタンスの削除
CIMOM から特定のインスタンスを削除するには、以下のように ***delete()*** *メソッドを使用します。*

```
instance_object.delete()
```

instance_object を、*削除するインスタンスオブジェクト名に置き換えます。このメソッドはブール値を返します。インスタンスを削除すると、そのプロパティーとメソッドにはアクセスできなくなることに注意してください。*

> *例22.17 個別インスタンスの削除*
>
> ***LMI_Account*** *クラスは、管理システム上のユーザーアカウントを表します。例22.5「ネームスペースオブジェクトへのアクセス」 で作成した* ***ns*** *ネームスペースオブジェクトを使用して、ユーザー* ***lmishell-user*** *用に* ***LMI_Account*** *クラスのインスタンスを作成し、それを変数* ***user*** *に割り当てるには、インタラクティブプロンプトに以下のコマンドを入力します。*
>
> ```
> > user = ns.LMI_Account.first_instance({"Name" : "lmishell-user"})
> >
> ```
>
> *このインスタンスを削除して、システムから* ***lmishell-user*** *を取り除くには、以下を入力します。*
>
> ```
> _
> ```

```
> user.delete()
True
>
```

利用可能なプロパティーのリスト表示とアクセス
特定のインスタンスオブジェクトの利用可能なプロパティーのリストを表示するには、以下のように ***print_properties()*** *メソッドを使用します。*

```
instance_object.print_properties()
```

instance_object *を、検査するインスタンスオブジェクト名に置き換えます。このメソッドは、利用可能なプロパティーを標準出力に出力します。*

利用可能なプロパティーリストを取得するには、 ***properties()*** *メソッドを使用します。*

```
instance_object.properties()
```

このメソッドは文字列のリストを返します。

例22.18 利用可能なプロパティーのリスト表示

例22.14「インスタンスへのアクセス」 で作成した ***device*** *インスタンスオブジェクトを検査して、利用可能なプロパティーのリストを表示するには、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> device.print_properties()
RequestedState
HealthState
StatusDescriptions
TransitioningToState
Generation
...
>
```

このプロパティーのリストを変数 ***device_properties*** *に割り当てるには、以下を入力します。*

```
> device_properties = device.properties()
>
```

特定のプロパティーの現在の値を取得するには、以下の構文を使用します。

```
instance_object.property_name
```

property_name *を、アクセスするプロパティー名に置き換えます。*

特定のプロパティーの値を修正するには、以下のように値を割り当てます。

```
instance_object.property_name = value
```

value を、*プロパティーの新たな値に置き換えます。CIMOM への変更を伝達するには、**push()** メソッドも実行する必要があることに注意してください。*

```
instance_object.push()
```

このメソッドは、戻り値、戻り値のパラメーター、およびエラー文字列で設定される 3 項目タプルを返します。

例22.19 個別プロパティーへのアクセス

*例22.14「インスタンスへのアクセス」 で作成した **device** インスタンスオブジェクトを検査して、プロパティー **SystemName** の値を表示するには、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> device.SystemName
u'server.example.com'
>
```

利用可能なメソッドのリスト表示と使用
*特定のインスタンスオブジェクトの利用可能なメソッドリストを表示するには、以下のように **print_methods()** メソッドを使用します。*

```
instance_object.print_methods()
```

instance_object を、*検査するインスタンスオブジェクト名に置き換えます。このメソッドは、利用可能なメソッドを標準出力に出力します。*

*利用可能なメソッドリストを取得するには、**methods()** メソッドを使用します。*

```
instance_object.methods()
```

このメソッドは文字列のリストを返します。

例22.20 利用可能なメソッドのリスト表示

*例22.14「インスタンスへのアクセス」 で作成した **device** インスタンスオブジェクトを検査して利用可能なメソッドのリストを表示するには、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> device.print_methods()
RequestStateChange
>
```

*これらメソッドのリストを変数 **network_device_methods** に割り当てるには、以下を入力します。*

```
> network_device_methods = device.methods()
>
```

特定のメソッドを呼び出すには、以下の構文を使用します。

```
instance_object.method_name(
  parameter=value,
  ...)
```

instance_object を、使用するインスタンスオブジェクト名に、method_name を、呼び出すメソッド名に、parameter を、設定するパラメーター名に、value を、このパラメーターの値に置き換えます。メソッドは、戻り値、戻り値のパラメーター、およびエラー文字列で設定される 3 項目タプルを返します。

重要

LMIInstance オブジェクトは、自動的にそのコンテンツ (プロパティー、メソッド、修飾子など) を更新しません。自動的に更新するようにするには、以下のように refresh() メソッドを使用します。

例22.21 メソッドの使用

PG_ComputerSystem クラスはシステムを表します。例22.5「ネームスペースオブジェクトへのアクセス」 で作成した ns ネームスペースオブジェクトを使用してこのクラスのインスタンスを作成し、これを変数 sys に割り当てるには、インタラクティブプロンプトに以下のコマンドを入力します。

```
> sys = ns.PG_ComputerSystem.first_instance()
>
```

LMI_AccountManagementService クラスは、システム内でユーザーおよびグループの管理を可能にするメソッドを実装します。このクラスのインスタンスを作成して変数 acc に割り当てるには、以下を使用します。

```
> acc = ns.LMI_AccountManagementService.first_instance()
>
```

システムに lmishell-user という名前の新規ユーザーを作成するには、以下のように CreateAccount() メソッドを使用します。

```
> acc.CreateAccount(Name="lmishell-user", System=sys)
LMIReturnValue(rval=0, rparams=NocaseDict({u'Account':
LMIInstanceName(classname="LMI_Account"...), u'Identities':
[LMIInstanceName(classname="LMI_Identity"...),
LMIInstanceName(classname="LMI_Identity"...)]}), errorstr='')
```

LMIShell は、同期メソッド呼び出しをサポートします。同期メソッドを使用するとき、LMIShell は関連のジョブオブジェクトを待ち、finished に状態を変更します。そして、このジョブのリターンパラメーターを返します。LMIShell は、指定のメソッドが以下のクラスの 1 つのオブジェクトを返すと、同期メソッド呼び出しを実行できます。

- *LMI_StorageJob*
- *LMI_SoftwareInstallationJob*
- *LMI_NetworkJob*

LMIShell は、最初に、待機メソッドに indication を使用します。それが失敗すると、代わりにポーリングメソッドを使います。

同期メソッド呼び出しを実行するには、以下の構文を使用します。

```
instance_object.Syncmethod_name(
  parameter=value,
  ...)
```

instance_object を、*使用するインスタンスオブジェクト名に*、method_name を、*呼び出すメソッド名に*、parameter を、*設定するパラメーター名に*、value を、*このパラメーターの値に置き換えます。すべての同期メソッドには、その名前に* ***Sync*** *接頭辞があり、ジョブの戻り値、ジョブの戻り値のパラメーター、およびジョブのエラー文字列で設定される 3 項目タプルを返します。*

また、LMIShell がポーリングメソッドのみを使用するように強制することもできます。これを行うには、以下のように ***PreferPolling*** *パラメーターを指定します。*

```
instance_object.Syncmethod_name(
  PreferPolling=True
  parameter=value,
  ...)
```

ValueMap パラメーターのリスト表示と閲覧
*CIM メソッドには、Managed Object Format (*MOF*) 定義でValueMap パラメーター が含まれる場合があります。ValueMap パラメーターには定数値が含まれます。*

特定メソッドで利用可能なValueMap パラメーターのリストを表示するには、以下のように ***print_valuemap_parameters()*** *メソッドを使用します。*

```
instance_object.method_name.print_valuemap_parameters()
```

instance_object を、*検査するインスタンスオブジェクト名に置き換え*、method_name を、*検査するメソッド名に置き換えます。このメソッドは、利用可能な* ***ValueMap*** *パラメーターを標準出力に出力します。*

利用可能なValueMap パラメーターのリストを取得するには、 ***valuemap_parameters()*** *メソッドを使用します。*

```
instance_object.method_name.valuemap_parameters()
```

このメソッドは文字列のリストを返します。

例22.22 ValueMap パラメーターのリスト表示

例22.21「メソッドの使用」 で作成した ***acc*** *インスタンスオブジェクトを検査して、* ***CreateAccount()*** *メソッドで利用可能なValueMap パラメーターのリストを表示するには、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> acc.CreateAccount.print_valuemap_parameters()
CreateAccount
>
```

このValueMap パラメーターのリストを変数 ***create_account_parameters*** *に割り当てるには、以下を入力します。*

```
> create_account_parameters = acc.CreateAccount.valuemap_parameters()
>
```

特定の ValueMap パラメーターにアクセスするには、以下の構文を使用します。

```
instance_object.method_name.valuemap_parameterValues
```

valuemap_parameter を、アクセスする ValueMap パラメーター名に置き換えます。

利用可能な定数値のリストを表示するには、以下のように **print_values()** メソッドを使用します。

```
instance_object.method_name.valuemap_parameterValues.print_values()
```

このメソッドは、名前の付いた利用可能な定数値を標準出力に出力します。また、**values()** メソッドを使用して利用可能な定数値のリストを取得することもできます。

```
instance_object.method_name.valuemap_parameterValues.values()
```

このメソッドは文字列のリストを返します。

例22.23 ValueMap パラメーターへのアクセス

例22.22「ValueMap パラメーターのリスト表示」では、ValueMap パラメーター **CreateAccount** について説明しました。このパラメーターを検査して利用可能な定数値のリストを表示するには、インタラクティブプロンプトに以下のコマンドを入力します。

```
> acc.CreateAccount.CreateAccountValues.print_values()
Operationunsupported
Failed
Unabletosetpasswordusercreated
Unabletocreatehomedirectoryusercreatedandpasswordset
Operationcompletedsuccessfully
>
```

この定数値のリストを変数 **create_account_values** に割り当てるには、以下を入力します。

```
> create_account_values = acc.CreateAccount.CreateAccountValues.values()
>
```

特定の定数値にアクセスするには、以下の構文を使用します。

```
instance_object.method_name.valuemap_parameterValues.constant_value_name
```

constant_value_name を、定数値の名前を置き換えます。また、以下のように **value()** メソッドを使用することもできます。

```
instance_object.method_name.valuemap_parameterValues.value("constant_value_name")
```

特定の定数値の名前を確認するには、**value_name()** メソッドを使用します。

```
instance_object.method_name.valuemap_parameterValues.value_name("constant_value")
```

このメソッドは文字列を返します。

例22.24 定数値へのアクセス

例22.23「ValueMap パラメーターへのアクセス」 では、**CreateAccount** ValueMap パラメーターが定数値 **Failed** を提供していました。この名前の定数値にアクセスするには、インタラクティブプロンプトに以下のコマンドを入力します。

```
> acc.CreateAccount.CreateAccountValues.Failed
2
> acc.CreateAccount.CreateAccountValues.value("Failed")
2
>
```

この定数値の名前を確認するには、以下を入力します。

```
> acc.CreateAccount.CreateAccountValues.value_name(2)
u'Failed'
>
```

インスタンスオブジェクトの更新
LMIShell が使用するローカルオブジェクトは、CIMOM 側での CIM オブジェクトを表し、このオブジェクトが LMIShell のオブジェクトと連携中に変更されると、古くなる場合があります。特定のインスタンスオブジェクトのプロパティーとメソッドを更新するには、以下のように **refresh()** *メソッドを使用します。*

```
instance_object.refresh()
```

instance_object *を、更新するオブジェクト名に置き換えます。このメソッドは、戻り値、戻り値のパラメーター、およびエラー文字列で設定される 3 項目タプルを返します。*

例22.25 インスタンスオブジェクトの更新

例22.14「インスタンスへのアクセス」 で作成した **device** インスタンスオブジェクトのプロパティーとメソッドを更新するには、インタラクティブプロンプトに以下のコマンドを入力します。

```
> device.refresh()
LMIReturnValue(rval=True, rparams=NocaseDict({}), errorstr='')
>
```

MOF 表現の表示
インスタンスオブジェクトの MOF (Managed Object Format) 表現を表示するには、以下のように **tomof()** *メソッドを使用します。*

```
instance_object.tomof()
```

instance_object *を、検査するインスタンスオブジェクト名に置き換えます。このメソッドは、オブジェクトの MOF 表現を標準出力に出力します。*

-

例22.26 MOF 表現の表示

例22.14「インスタンスへのアクセス」 で作成した ***device*** *インスタンスオブジェクトの MOF 表現を表示するには、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> device.tomof()
instance of LMI_IPNetworkConnection {
    RequestedState = 12;
    HealthState = NULL;
    StatusDescriptions = NULL;
    TransitioningToState = 12;
...
```

22.4.6. インスタンス名の使用

LMIShell インスタンス名は、プライマリーキーとその値のセットを持つオブジェクトです。この種類のオブジェクトは、インスタンスを正確に識別します。

インスタンス名へのアクセス
CIMInstance *オブジェクトは* ***CIMInstanceName*** *オブジェクトにより識別されます。利用可能なインスタンス名オブジェクトのリストを取得するには、以下のように* ***instance_names()*** *メソッドを使用します。*

```
class_object.instance_names()
```

class_object *を、検査するクラスオブジェクト名に置き換えます。このメソッドは、* ***LMIInstanceName*** *オブジェクトのリストを返します。*

クラスオブジェクトの最初のインスタンス名オブジェクトにアクセスするには、 ***first_instance_name()*** *メソッドを使用します。*

```
class_object.first_instance_name()
```

このメソッドは、 ***LMIInstanceName*** *オブジェクトを返します。*

instance_names() *と* ***first_instance_name()*** *は、インスタンス名オブジェクトのリストを表示したり、最初のインスタンス名オブジェクトを返したりするほかにも、オプションの引数をサポートして結果をフィルターすることもできます。*

```
class_object.instance_names(criteria)
```

```
class_object.first_instance_name(criteria)
```

criteria *を、鍵と値のペアで設定される辞書で置き換えます。この鍵は鍵プロパティーを表し、値はこの鍵プロパティーの必要な値を表します。*

例22.27 インスタンス名へのアクセス

例22.7「クラスオブジェクトへのアクセス」 で作成した ***cls*** *クラスオブジェクトの最初のインスタンス名で、* ***eth0*** *と同等の* ***Name*** *鍵プロパティーがあるものを見つけ、これを変数* ***device_name*** *に割り当てるには、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> device_name = cls.first_instance_name({"Name": "eth0"})
>
```

インスタンス名の検査
すべてのインスタンス名オブジェクトには、そのクラス名と、所属するネームスペースに関する情報が保存されています。

特定のインスタンス名オブジェクトのクラス名を取得するには、以下の構文を使用します。

```
instance_name_object.classname
```

instance_name_object *を、検査するインスタンス名オブジェクトの名前に置き換えます。これは、クラス名を表す文字列を返します。*

インスタンス名オブジェクトが所属するネームスペースに関する情報を取得するには、以下を使用します。

```
instance_name_object.namespace
```

これは、名前空間を表す文字列を返します。

例22.28 インスタンス名の検査

例22.27「インスタンス名へのアクセス」 で作成した **device_name** *インスタンス名オブジェクトを検査して、そのクラス名と対応するネームスペースを表示するには、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> device_name.classname
u'LMI_IPNetworkConnection'
> device_name.namespace
'root/cimv2'
>
```

新規インスタンス名の作成
LMIShell *を使用すると、リモートオブジェクトのすべてのプライマリーキーを知っている場合に、新たにラップされた* **CIMInstanceName** *オブジェクトを作成できます。このインスタンス名オブジェクトを使用して、インスタンスオブジェクト全体を取得できます。*

クラスオブジェクトの新規インスタンス名を作成するには、以下のように **new_instance_name()** *メソッドを使用します。*

```
class_object.new_instance_name(key_properties)
```

class_object *をクラスオブジェクト名に置き換え、*key_properties *を、鍵と値のペアで設定される辞書に置き換えます。このメソッドは、* **LMIInstanceName** *オブジェクトを返します。*

例22.29 新規インスタンス名の作成

LMI_Account *クラスは、管理システム上のユーザーアカウントを表します。例22.5「ネームスペースオブジェクトへのアクセス」 で作成した* **ns** *ネームスペースオブジェクトを使用して、管理システムの* **lmishell-user** *ユーザーを表す* **LMI_Account** *クラスの新規インスタンス名を作成するには、イ*

ンタラクティブプロンプトに以下のコマンドを入力します。

```
> instance_name = ns.LMI_Account.new_instance_name({
...   "CreationClassName" : "LMI_Account",
...   "Name" : "lmishell-user",
...   "SystemCreationClassName" : "PG_ComputerSystem",
...   "SystemName" : "server"})
>
```

鍵プロパティーのリスト表示とアクセス
特定のインスタンス名オブジェクトの利用可能な鍵プロパティーのリストを表示するには、以下のように **print_key_properties()** メソッドを使用します。

```
instance_name_object.print_key_properties()
```

instance_name_object を、検査するインスタンス名オブジェクトの名前に置き換えます。このメソッドは、利用可能な鍵プロパティーを標準出力に出力します。

利用可能な鍵プロパティーのリストを取得するには、**key_properties()** メソッドを使用します。

```
instance_name_object.key_properties()
```

このメソッドは文字列のリストを返します。

例22.30 利用可能な鍵プロパティーのリスト表示

例22.27「インスタンス名へのアクセス」 で作成した **device_name** インスタンス名オブジェクトを検査して、利用可能な鍵プロパティーのリストを表示するには、インタラクティブプロンプトに以下のコマンドを入力します。

```
> device_name.print_key_properties()
CreationClassName
SystemName
Name
SystemCreationClassName
>
```

この鍵プロパティーのリストを変数 **device_name_properties** に割り当てるには、以下を入力します。

```
> device_name_properties = device_name.key_properties()
>
```

特定の鍵プロパティーの現在の値を取得するには、以下の構文を使用します。

```
instance_name_object.key_property_name
```

key_property_name をアクセスする鍵プロパティー名に置き換えます。

例22.31 個別の鍵プロパティーへのアクセス

例22.27「インスタンス名へのアクセス」で作成した**device_name**インスタンス名オブジェクトを検査して、鍵プロパティー**SystemName**の値を表示するには、インタラクティブプロンプトに以下のコマンドを入力します。

```
> device_name.SystemName
u'server.example.com'
>
```

インスタンス名のインスタンスへの変換
インスタンス名はインスタンスに変換できます。これを行うには、以下のように**to_instance()**メソッドを使用します。

```
instance_name_object.to_instance()
```

instance_name_object を、変換するインスタンス名オブジェクトの名前に置き換えます。このメソッドは、**LMIInstance**オブジェクトを返します。

例22.32 インスタンス名のインスタンスへの変換

例22.27「インスタンス名へのアクセス」で作成した**device_name**インスタンス名オブジェクトをインスタンスオブジェクト変換して、変数**device**に割り当てるには、インタラクティブプロンプトに以下のコマンドを入力します。

```
> device = device_name.to_instance()
>
```

22.4.7. 関連するオブジェクトの使用

Common Information Model は、管理オブジェクト間の関係を定義します。

関連するインスタンスへのアクセス
特定のインスタンスオブジェクトに関連するオブジェクトのリストを取得するには、以下のように**associators()**メソッドを使用します。

```
instance_object.associators(
  AssocClass=class_name,
  ResultClass=class_name,
  ResultRole=role,
  IncludeQualifiers=include_qualifiers,
  IncludeClassOrigin=include_class_origin,
  PropertyList=property_list)
```

特定のインスタンスオブジェクトに関連する最初のオブジェクトにアクセスするには、**first_associator()**メソッドを使用します。

```
instance_object.first_associator(
  AssocClass=class_name,
  ResultClass=class_name,
  ResultRole=role,
```

> ***IncludeQualifiers***=include_qualifiers,
> ***IncludeClassOrigin***=include_class_origin,
> ***PropertyList***=property_list*)*

instance_object を、*検査するインスタンスオブジェクト名に置き換えます。以下のパラメーターを指定すると結果をフィルターにかけられます。*

- ***AssocClass**: 返される各オブジェクトは、このクラスまたはそのサブクラスの1 つのインスタンスを通してソースオブジェクトに関連付けられている必要があります。デフォルト値は **None** です。*
- ***ResultClass**: 返される各オブジェクトは、このクラス、またはそのいずれかのサブクラスのインスタンスである必要があります。もしくは、このクラスまたはいずれかのサブクラスである必要があります。デフォルト値は **None** です。*
- ***Role**: 返される各オブジェクトは、ソースオブジェクトが特定のロールを果たす関連付けでソースオブジェクトと関連付けられている必要があります。このオブジェクトは、ソースオブジェクトが特定のロールを果たす関連付けでソースオブジェクトと関連付けられている必要があります。デフォルト値は **None** です。*
- ***ResultRole**: 返される各オブジェクトは、返されるオブジェクトが特定のロールを果たす関連付けでソースオブジェクトと関連付けられている必要があります。返されるオブジェクトを参照する関連付けクラス内のプロパティー名は、このパラメーターの値と合致する必要があります。デフォルト値は **None** です。*

他のパラメーターは以下のとおりです。

- ***IncludeQualifiers**: 応答に各オブジェクトのすべての修飾子 (オブジェクトおよび返されるプロパティー上の修飾子を含む) を QUALIFIER 要素として組み込むかどうかを示すブール値。デフォルト値は **False** です。*
- ***IncludeClassOrigin**: 返される各オブジェクトで適切な全要素に CLASSORIGIN 属性が存在すべきかどうかを示すブール値。デフォルト値は **False** です。*
- ***PropertyList**: このリストのメンバーは1 つ以上のプロパティー名を定義します。返されたオブジェクトは、このリストにないプロパティーには要素を含めません。**PropertyList** が空のリストである場合は、返されるオブジェクトにプロパティーは含まれません。**None** の場合、追加のフィルタリングは定義されません。デフォルト値は **None** です。*

> *例22.33 関連するインスタンスへのアクセス*
>
> ***LMI_StorageExtent** クラスは、システム内で利用可能なブロックデバイスを表します。例 22.5「ネームスペースオブジェクトへのアクセス」 で作成した **ns** ネームスペースオブジェクトを使用して、ブロックデバイス **/dev/vda** 用に **LMI_StorageExtent** クラスのインスタンスを作成し、これを変数 **vda** に割り当てるには、インタラクティブプロンプトに以下のコマンドを入力します。*
>
> ```
> > vda = ns.LMI_StorageExtent.first_instance({
> ...   "DeviceID" : "/dev/vda"})
> >
> ```
>
> *このブロックデバイスにおけるディスクパーティションのリストを取得して、これを変数 **vda_partitions** に割り当てるには、以下のように **associators()** メソッドを使用します。*
>
> ```
> > vda_partitions = vda.associators(ResultClass="LMI_DiskPartition")
> >
> ```

■

関連するインスタンス名へのアクセス
特定のインスタンスオブジェクトで関連するインスタンス名のリストを取得するには、以下のように ***associator_names()*** *メソッドを使用します。*

```
instance_object.associator_names(
  AssocClass=class_name,
  ResultClass=class_name,
  Role=role,
  ResultRole=role)
```

特定のインスタンスオブジェクトで最初に関連するインスタンス名にアクセスするには、 ***first_associator_name()*** *メソッドを使用します。*

```
instance_object.first_associator_name(
  AssocClass=class_object,
  ResultClass=class_object,
  Role=role,
  ResultRole=role)
```

instance_object *を、検査するインスタンスオブジェクト名に置き換えます。以下のパラメーターを指定すると結果をフィルターにかけられます。*

- ***AssocClass****: 返される名前により、このクラスのインスタンス、またはサブクラスの1 つから、ソースオブジェクトに関連づけられる必要のあるオブジェクトが識別されます。デフォルト値は* **None** *です。*
- ***ResultClass****: 返される各名前はオブジェクトを識別します。オブジェクトは、このクラスのインスタンスまたはそのサブクラスの1 つである必要があります。または、このクラスまたはそのサブクラスのいずれかである必要があります。デフォルト値は* **None** *です。*
- ***Role****: 返された各名前は、ソースオブジェクトが指定されたロールを実行する関連付けでソースオブジェクトと関連付ける必要があるオブジェクトを識別します。このオブジェクトは、ソースオブジェクトが特定のロールを果たす関連付けでソースオブジェクトと関連付けられている必要があります。デフォルト値は* **None** *です。*
- ***ResultRole****: 返された各名前はオブジェクトを特定し、返された名前付きオブジェクトが指定されたロールを実行する関連付けでソースオブジェクトと関連付ける必要があるオブジェクトを識別します。返されるオブジェクトを参照する関連付けクラス内のプロパティー名は、このパラメーターの値と合致する必要があります。デフォルト値は* **None** *です。*

例22.34 関連するインスタンス名へのアクセス

例22.33「関連するインスタンスへのアクセス」 で作成した **vda** *インスタンスオブジェクトを使用して、その関連するインスタンス名のリストを取得し、変数* **vda_partitions** *に割り当てるには、以下を入力します。*

```
> vda_partitions = vda.associator_names(ResultClass="LMI_DiskPartition")
>
```

22.4.8. 関連付けオブジェクトの使用

Common Information Model は、管理オブジェクト間の関係を定義します。関連付けオブジェクトは、他の 2 つのオブジェクト間の関係を定義します。

関連付けインスタンスへのアクセス
*特定のターゲットオブジェクトを参照する関連付けオブジェクトのリストを取得するには、以下のように **references()** メソッドを使用します。*

```
instance_object.references(
  ResultClass=class_name,
  Role=role,
  IncludeQualifiers=include_qualifiers,
  IncludeClassOrigin=include_class_origin,
  PropertyList=property_list)
```

*特定のターゲットオブジェクトを参照する最初の関連付けオブジェクトにアクセスするには、**first_reference()** メソッドを使用します。*

```
instance_object.first_reference(
...  ResultClass=class_name,
...  Role=role,
...  IncludeQualifiers=include_qualifiers,
...  IncludeClassOrigin=include_class_origin,
...  PropertyList=property_list)
>
```

instance_object *を、検査するインスタンスオブジェクト名に置き換えます。以下のパラメーターを指定すると結果をフィルターにかけられます。*

- ***ResultClass**: 返される各オブジェクトは、このクラス、またはそのいずれかのサブクラスのインスタンスである必要があります。もしくは、このクラスまたはいずれかのサブクラスである必要があります。デフォルト値は **None** です。*

- ***Role**: 返される各オブジェクトは、このパラメーターの値に合致する名前を持つプロパティーでターゲットオブジェクトを参照する必要があります。デフォルト値は **None** です。*

他のパラメーターは以下のとおりです。

- ***IncludeQualifiers**: 応答に、QUALIFIER 要素として各オブジェクト (オブジェクト、および返されるすべてのプロパティー上の修飾子を含む) を追加するどうかを示すブール値。デフォルト値は **False** です。*

- ***IncludeClassOrigin**: 返される各オブジェクトで適切な全要素に CLASSORIGIN 属性が存在すべきかどうかを示すブール値。デフォルト値は **False** です。*

- ***PropertyList**: このリストのメンバーは 1 つ以上のプロパティー名を定義します。返されたオブジェクトは、このリストにないプロパティーには要素を含めません。**PropertyList** が空のリストである場合は、返されるオブジェクトにプロパティーは含まれません。**None** の場合、追加のフィルタリングは定義されません。デフォルト値は **None** です。*

例22.35 関連付けインスタンスへのアクセス

***LMI_LANEndpoint** クラスは、特定のネットワークインターフェイスデバイスに関連付けられる通信エンドポイントを表します。例22.5「ネームスペースオブジェクトへのアクセス」 で作成した **ns** ネームスペースオブジェクトを使用して、ネットワークインターフェイスデバイス **eth0** 用に **LMI_LANEndpoint** クラスのインスタンスを作成し、これを変数 **lan_endpoint** に割り当てるには、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> lan_endpoint = ns.LMI_LANEndpoint.first_instance({
...   "Name" : "eth0"})
>
```

LMI_BindsToLANEndpoint オブジェクトを参照する最初の関連付けオブジェクトにアクセスし、これを変数 bind に割り当てるには、以下を入力します。

```
> bind = lan_endpoint.first_reference(
...   ResultClass="LMI_BindsToLANEndpoint")
>
```

これで Dependent プロパティーを使用して、対応するネットワークインターフェイスデバイスの IP アドレスを表す依存 LMI_IPProtocolEndpoint クラスにアクセスすることができます。

```
> ip = bind.Dependent.to_instance()
> print ip.IPv4Address
192.168.122.1
>
```

関連付けインスタンス名へのアクセス
特定のインスタンスオブジェクトの関連付けインスタンス名のリストを取得するには、以下のように reference_names() メソッドを使用します。

```
instance_object.reference_names(
  ResultClass=class_name,
  Role=role)
```

特定のインスタンスオブジェクトの最初の関連付けインスタンスにアクセスするには、first_reference_name() メソッドを使用します。

```
instance_object.first_reference_name(
  ResultClass=class_name,
  Role=role)
```

instance_object を、検査するインスタンスオブジェクト名に置き換えます。以下のパラメーターを指定すると結果をフィルターにかけられます。

- **ResultClass**: 返される各オブジェクトは、このクラスのインスタンスかそのサブクラスの 1 つのインスタンス、または、このクラスまたはそのサブクラスの 1 つである必要があります。デフォルト値は **None** です。
- **Role**: 返される各オブジェクトは、このパラメーターの値に合致する名前を持つプロパティーでターゲットインスタンスを参照するオブジェクトを識別します。デフォルト値は **None** です。

例22.36 関連付けインスタンス名へのアクセス

例22.35「関連付けインスタンスへのアクセス」 で作成した **lan_endpoint** インスタンスオブジェクトを使用して、**LMI_BindsToLANEndpoint** オブジェクトを参照する最初の関連付けインスタンス名にアクセスし、これを変数 **bind** に割り当てるには、以下を入力します。

```
> bind = lan_endpoint.first_reference_name(
...   ResultClass="LMI_BindsToLANEndpoint")
```

これで **Dependent** プロパティーを使用して、対応するネットワークインターフェイスデバイスの IP アドレスを表す依存 **LMI_IPProtocolEndpoint** クラスにアクセスすることができます。

```
> ip = bind.Dependent.to_instance()
> print ip.IPv4Address
192.168.122.1
>
```

22.4.9. Indication の使用

Indication は、データにおける特定の変化に反応して発生する特有のイベントへのリアクションです。LMIShell は、indication をサブスクライブしてこのようなイベント対応を受信できます。

Indication のサブスクライブ
indication をサブスクライブするには、以下のように **subscribe_indication()** メソッドを使用します。

```
connection_object.subscribe_indication(
  QueryLanguage="WQL",
  Query='SELECT * FROM CIM_InstModification',
  Name="cpu",
  CreationNamespace="root/interop",
  SubscriptionCreationClassName="CIM_IndicationSubscription",
  FilterCreationClassName="CIM_IndicationFilter",
  FilterSystemCreationClassName="CIM_ComputerSystem",
  FilterSourceNamespace="root/cimv2",
  HandlerCreationClassName="CIM_IndicationHandlerCIMXML",
  HandlerSystemCreationClassName="CIM_ComputerSystem",
  Destination="http://host_name:5988")
```

別の方法では、以下のように、このメソッド呼び出しの短いバージョンを使用することもできます。

```
connection_object.subscribe_indication(
  Query='SELECT * FROM CIM_InstModification',
  Name="cpu",
  Destination="http://host_name:5988")
```

connection_object を接続オブジェクトに置き換え、host_name を、indication を配信するシステムのホスト名に置き換えます。

デフォルトでは、LMIShell インタープリターが作成したすべてのサブスクリプションは、インタープリターの終了時に自動的に削除されます。この動作を変更するには、**Permanent=True** キーワードパラメーターを **subscribe_indication()** メソッド呼び出しに渡します。これで、LMIShell がサブスクリプションを削除しなくなります。

例22.37 Indication のサブスクライブ

例22.1「リモートの CIMOM への接続」 で作成した **c** 接続オブジェクトを使用して、**cpu** という名前の indication をサブスクライブするには、インタラクティブプロンプトに以下のコマンドを入力します。

```
> c.subscribe_indication(
...   QueryLanguage="WQL",
...   Query='SELECT * FROM CIM_InstModification',
```

```
... Name="cpu",
... CreationNamespace="root/interop",
... SubscriptionCreationClassName="CIM_IndicationSubscription",
... FilterCreationClassName="CIM_IndicationFilter",
... FilterSystemCreationClassName="CIM_ComputerSystem",
... FilterSourceNamespace="root/cimv2",
... HandlerCreationClassName="CIM_IndicationHandlerCIMXML",
... HandlerSystemCreationClassName="CIM_ComputerSystem",
... Destination="http://server.example.com:5988")
LMIReturnValue(rval=True, rparams=NocaseDict({}), errorstr='')
>
```

サブスクライブしている indication のリスト表示
サブスクライブしている indication のリストを表示するには、以下のように ***print_subscribed_indications()*** *メソッドを使用します。*

```
connection_object.print_subscribed_indications()
```

connection_object を、*検査する接続オブジェクト名に置き換えます。このメソッドは、サブスクライブしている indication を標準出力に出力します。*

サブスクライブしている indication のリストを取得するには、 ***subscribed_indications()*** *メソッドを使用します。*

```
connection_object.subscribed_indications()
```

このメソッドは文字列のリストを返します。

例22.38 サブスクライブしている indication のリスト表示

例22.1「リモートの CIMOM への接続」 *で作成した* ***c*** *接続オブジェクトを検査し、サブスクライブしている indication のリストを表示するには、インタラクティブプロンプトに以下のコマンドを入力します。*

```
> c.print_subscribed_indications()
>
```

この indication のリストを変数 ***indications*** *に割り当てるには、以下を入力します。*

```
> indications = c.subscribed_indications()
>
```

Indication のサブスクライブ解除
デフォルトでは、LMIShell インタープリターが作成したすべてのサブスクリプションは、インタープリターの終了時に自動的に削除されます。これより前に個別のサブスクリプションを削除するには、以下のように ***unsubscribe_indication()*** *メソッドを使用します。*

```
connection_object.unsubscribe_indication(indication_name)
```

connection_object を、*検査する接続オブジェクト名に置き換え、*indication_name を、*削除する indication の名前に置き換えます。*

すべてのサブスクリプションを削除するには、**unsubscribe_all_indications()** メソッドを使用します。

```
connection_object.unsubscribe_all_indications()
```

例22.39 Indication のサブスクライブ解除

例22.1「リモートの CIMOM への接続」 で作成した **c** 接続オブジェクトを使用して、例22.37「Indication のサブスクライブ」 で作成した indication のサブスクライブを解除するには、インタラクティブプロンプトに以下のコマンドを入力します。

```
> c.unsubscribe_indication('cpu')
LMIReturnValue(rval=True, rparams=NocaseDict({}), errorstr='')
>
```

Indication ハンドラーの実装

subscribe_indication() メソッドを使用すると、indication を配信するシステムのホスト名を指定できます。以下の例では、indication ハンドラーの実装方法を示します。

```
> def handler(ind, arg1, arg2, kwargs): ... exported_objects = ind.exported_objects() ...
do_something_with(exported_objects) > listener = LmiIndicationListener("0.0.0.0",
listening_port) > listener.add_handler("indication-name-XXXXXXXX", handler, arg1, arg2,
kwargs)
> listener.start()
>
```

ハンドラーの最初の引数は **LmiIndication** オブジェクトで、これには indication がエクスポートしたメソッドとオブジェクトのリストが含まれます。他のパラメーターはユーザー固有なので、ハンドラーをリスナーに追加する際にこれらの引数を指定する必要があります。

上記の例では、**add_handler()** メソッド呼び出しは、8 つの **X** 文字の特別な文字列を使用しています。この文字は、ハンドラー名が競合しないように、リスナーが生成するランダムな文字列に置き換えられます。ランダムな文字列を使用するには、最初にインデックスリスナーを起動してからインデックスにサブスクライブし、ハンドラーオブジェクトの **Destination** プロパティーに **schema://host_name/random_string** の値が含まれるようにします。

例22.40 Indication ハンドラーの実装

以下のスクリプトでは、**192.168.122.1** にある管理システムを監視し、新規ユーザーアカウントの作成時に常に **indication_callback()** 関数を呼び出すハンドラーの書き込み方法を示しています。

```
#!/usr/bin/lmishell

import sys
from time import sleep
from lmi.shell.LMIUtil import LMIPassByRef
from lmi.shell.LMIIndicationListener import LMIIndicationListener

# These are passed by reference to indication_callback
var1 = LMIPassByRef("some_value")
var2 = LMIPassByRef("some_other_value")

def indication_callback(ind, var1, var2):
  # Do something with ind, var1 and var2
```

```
  print ind.exported_objects()
  print var1.value
  print var2.value

c = connect("hostname", "username", "password")

listener = LMIIndicationListener("0.0.0.0", 65500)
unique_name = listener.add_handler(
  "demo-XXXXXXXX",   # Creates a unique name for me
  indication_callback, # Callback to be called
  var1,        # Variable passed by ref
  var2         # Variable passed by ref
)

listener.start()

print c.subscribe_indication(
  Name=unique_name,
  Query="SELECT * FROM LMI_AccountInstanceCreationIndication WHERE
SOURCEINSTANCE ISA LMI_Account",
  Destination="192.168.122.1:65500"
)

try:
  while True:
    sleep(60)
except KeyboardInterrupt:
  sys.exit(0)
```

22.4.10. 使用例

本セクションでは、OpenLMI パッケージで配布される様々な CIM プロバイダーの例を示します。本セクションのすべての例で、以下の 2 つの変数定義を使用します。

```
c = connect("host_name", "user_name", "password")
ns = c.root.cimv2
```

host_name を、管理システムのホスト名に置き換え、user_name を、そのシステムで実行している OpenPegasus CIMOM への接続が許可されているユーザー名に置き換え、password を、ユーザーのパスワードに置き換えます。

OpenLMI サービスプロバイダーの使用
openlmi-service パッケージは、システムサービスの管理用に CIM プロバイダーをインストールします。以下の例では、この CIM プロバイダーを使用して利用可能なシステムサービスのリストを表示する方法と、そのサービスの開始、停止、有効化、および無効化を行う方法を示しています。

例22.41 利用可能なサービスのリスト表示

管理システムで利用可能なすべてのサービスと、そのサービスの開始 (**TRUE**) または (**FALSE**) に関する情報、およびステータス文字列をリスト表示するには、以下のコードスニペットを使用します。

```
for service in ns.LMI_Service.instances():
  print "%s:\t%s" % (service.Name, service.Status)
```

デフォルトで有効になっているサービスのみをリスト表示するには、以下のコードスニペットを使用します。

```
cls = ns.LMI_Service
for service in cls.instances():
  if service.EnabledDefault == cls.EnabledDefaultValues.Enabled:
    print service.Name
```

有効なサービスの場合は、**EnabledDefault** プロパティーの値が**2** と等しくなり、無効なサービスの場合は**3** と等しくなることに注意してください。

cups サービスに関する情報を表示するには、以下を使用します。

```
cups = ns.LMI_Service.first_instance({"Name": "cups.service"})
cups.doc()
```

例22.42 サービスの起動と停止

cups サービスを起動して現行ステータスを確認するには、以下のコードスニペットを使用します。

```
cups = ns.LMI_Service.first_instance({"Name": "cups.service"})
cups.StartService()
print cups.Status
cups.StopService()
print cups.Status
```

例22.43 サービスの有効化および無効化

cups サービスを有効または無効にして、**EnabledDefault** プロパティーを表示するには、以下のコードスニペットを使用します。

```
cups = ns.LMI_Service.first_instance({"Name": "cups.service"})
cups.TurnServiceOff()
print cups.EnabledDefault
cups.TurnServiceOn()
print cups.EnabledDefault
```

OpenLMI ネットワーキングプロバイダーの使用
openlmi-networking パッケージは、ネットワーキング用の CIM プロバイダーをインストールします。以下の例では、この CIM プロバイダーを使用して特定のポート番号に関連付けられた IP アドレスをリスト表示する方法、新規接続を作成する方法、静的 IP アドレスを設定する方法、および接続をアクティブ化する方法を説明しています。

例22.44 特定のポート番号に関連付けられている IP アドレスのリスト表示

eth0 ネットワークインターフェイスに関連付けられている IP アドレスのリストを表示するには、以下のコードスニペットを使用します。

```
device = ns.LMI_IPNetworkConnection.first_instance({'ElementName': 'eth0'})
for endpoint in device.associators(AssocClass="LMI_NetworkSAPSAPDependency",
ResultClass="LMI_IPProtocolEndpoint"):
  if endpoint.ProtocolIFType == ns.LMI_IPProtocolEndpoint.ProtocolIFTypeValues.IPv4:
    print "IPv4: %s/%s" % (endpoint.IPv4Address, endpoint.SubnetMask)
  elif endpoint.ProtocolIFType == ns.LMI_IPProtocolEndpoint.ProtocolIFTypeValues.IPv6:
    print "IPv6: %s/%d" % (endpoint.IPv6Address, endpoint.IPv6SubnetPrefixLength)
```

*このコードスニペットは、指定した **LMI_IPNetworkConnection** クラスに関連付けられた **LMI_IPProtocolEndpoint** クラスを使用します。*

デフォルトのゲートウェイを表示するには、以下のコードスニペットを使用します。

```
for rsap in
device.associators(AssocClass="LMI_NetworkRemoteAccessAvailableToElement",
ResultClass="LMI_NetworkRemoteServiceAccessPoint"):
  if rsap.AccessContext ==
ns.LMI_NetworkRemoteServiceAccessPoint.AccessContextValues.DefaultGateway:
    print "Default Gateway: %s" % rsap.AccessInfo
```

*デフォルトのゲートウェイは、**AccessContext** プロパティーが**DefaultGateway** と等しい **LMI_NetworkRemoteServiceAccessPoint** インスタンスで表されます。*

DNS サーバーのリストを取得するには、以下のようにオブジェクトモデルをトラバースする必要があります。

1. ***LMI_NetworkSAPSAPDependency** を使用して、指定した **LMI_IPNetworkConnection** に関連付けられた **LMI_IPProtocolEndpoint** インスタンスを取得する。*

2. ***LMI_DNSProtocolEndpoint** インスタンスに同一の関連付けを使用する。*

***LMI_NetworkRemoteAccessAvailableToElement** で関連付けられている DNS サーバーと同等の **AccessContext** プロパティーを持つ **LMI_NetworkRemoteServiceAccessPoint** インスタンスには、 **AccessInfo** プロパティーに DNS サーバーのアドレスがあります。*

***RemoteServiceAccessPath** の取得には他のパスもあり、エントリーは重複可能です。以下のコードスニペットは、**set()** 関数を使用して DNS サーバーから重複エントリーを削除します。*

```
dnsservers = set()
for ipendpoint in device.associators(AssocClass="LMI_NetworkSAPSAPDependency",
ResultClass="LMI_IPProtocolEndpoint"):
  for dnsedpoint in
ipendpoint.associators(AssocClass="LMI_NetworkSAPSAPDependency",
ResultClass="LMI_DNSProtocolEndpoint"):
    for rsap in
dnsedpoint.associators(AssocClass="LMI_NetworkRemoteAccessAvailableToElement",
ResultClass="LMI_NetworkRemoteServiceAccessPoint"):
      if rsap.AccessContext ==
ns.LMI_NetworkRemoteServiceAccessPoint.AccessContextValues.DNSServer:
        dnsservers.add(rsap.AccessInfo)
print "DNS:", ", ".join(dnsservers)
```

例22.45 新規接続の作成および静的IP アドレスの設定

eth0 ネットワークインターフェイスで静的IPv4 とステートレスのIPv6 設定を作成するには、以下のコードスニペットを使用します。

```
capability = ns.LMI_IPNetworkConnectionCapabilities.first_instance({ 'ElementName':
'eth0' })
result = capability.LMI_CreateIPSetting(Caption='eth0 Static',
    IPv4Type=capability.LMI_CreateIPSetting.IPv4TypeValues.Static,
    IPv6Type=capability.LMI_CreateIPSetting.IPv6TypeValues.Stateless)
setting = result.rparams["SettingData"].to_instance()
for settingData in
setting.associators(AssocClass="LMI_OrderedIPAssignmentComponent"):
  if setting.ProtocolIFType == ns.LMI_IPAssignmentSettingData.ProtocolIFTypeValues.IPv4:
    # Set static IPv4 address
    settingData.IPAddresses = ["192.168.1.100"]
    settingData.SubnetMasks = ["255.255.0.0"]
    settingData.GatewayAddresses = ["192.168.1.1"]
    settingData.push()
```

*このコードスニペットは、**LMI_IPNetworkConnectionCapabilities** のインスタンスで **LMI_CreateIPSetting()** メソッドを呼び出して新しい設定を作成します。これは **LMI_IPNetworkConnectionElementCapabilities** を介して **LMI_IPNetworkConnection** に関連付けられます。また、**push()** メソッドを使用して設定の修正も行います。*

例22.46 接続のアクティブ化

*設定をネットワークインターフェイスに適用するには、**LMI_IPConfigurationService** クラスの **ApplySettingToIPNetworkConnection()** メソッドを呼び出します。このメソッドは非同期で、ジョブを返します。以下のコードスニペットでは、このメソッドを同期で呼び出す方法を示しています。*

```
setting = ns.LMI_IPAssignmentSettingData.first_instance({ "Caption": "eth0 Static" })
port = ns.LMI_IPNetworkConnection.first_instance({ 'ElementName': 'ens8' })
service = ns.LMI_IPConfigurationService.first_instance()
service.SyncApplySettingToIPNetworkConnection(SettingData=setting,
IPNetworkConnection=port, Mode=32768)
```

***Mode** パラメーターは、設定の適用方法に影響を与えます。このパラメーターで最もよく使われる値は、以下のとおりです。*

- ***1**: 設定を直ちに適用し、自動アクティブ化します。*
- ***2**: 設定を自動アクティブ化しますが、すぐには適用しません。*
- ***4**: 切断して自動アクティブ化を無効にします。*
- ***5**: 設定状態を変更せず、自動アクティブ化のみを無効にします。*
- ***32768**: 設定を適用します。*
- ***32769**: 切断します。*

OpenLMI ストレージプロバイダーの使用
openlmi-storage パッケージは、ストレージ管理用の CIM プロバイダーをインストールします。以下の例では、この CIM プロバイダーを使用したボリュームグループおよび論理ボリュームの作成、ファイルシステムの構築およびマウント、システムが認知するブロックデバイスのリスト表示の方法を説明します。

***c** および **ns** の変数に加え、以下の例では下記の変数定義を使用します。*

```
MEGABYTE = 1024*1024
storage_service = ns.LMI_StorageConfigurationService.first_instance()
filesystem_service = ns.LMI_FileSystemConfigurationService.first_instance()
```

例22.47 ボリュームグループの作成

*/**dev**/**myGroup**/ にあり、3 つのメンバーがありデフォルトの拡張サイズが 4 MB の新規ボリュームグループを作成するには、以下のコードスニペットを使用します。*

```
# Find the devices to add to the volume group
# (filtering the CIM_StorageExtent.instances()
# call would be faster, but this is easier to read):
sda1 = ns.CIM_StorageExtent.first_instance({"Name": "/dev/sda1"})
sdb1 = ns.CIM_StorageExtent.first_instance({"Name": "/dev/sdb1"})
sdc1 = ns.CIM_StorageExtent.first_instance({"Name": "/dev/sdc1"})

# Create a new volume group:
(ret, outparams, err) = storage_service.SyncCreateOrModifyVG(
    ElementName="myGroup",
    InExtents=[sda1, sdb1, sdc1])
vg = outparams['Pool'].to_instance()
print "VG", vg.PoolID, \
    "with extent size", vg.ExtentSize, \
    "and", vg.RemainingExtents, "free extents created."
```

例22.48 論理ボリュームの作成

サイズが 100 MB の論理ボリューム 2 つを作成するには、以下のコードスニペットを使用します。

```
# Find the volume group:
vg = ns.LMI_VGStoragePool.first_instance({"Name": "/dev/mapper/myGroup"})

# Create the first logical volume:
(ret, outparams, err) = storage_service.SyncCreateOrModifyLV(
    ElementName="Vol1",
    InPool=vg,
    Size=100 * MEGABYTE)
lv = outparams['TheElement'].to_instance()
print "LV", lv.DeviceID, \
    "with", lv.BlockSize * lv.NumberOfBlocks,\
    "bytes created."

# Create the second logical volume:
(ret, outparams, err) = storage_service.SyncCreateOrModifyLV(
    ElementName="Vol2",
```

```
    InPool=vg,
    Size=100 * MEGABYTE)
lv = outparams['TheElement'].to_instance()
print "LV", lv.DeviceID, \
    "with", lv.BlockSize * lv.NumberOfBlocks, \
    "bytes created."
```

例22.49 ファイルシステムの作成

例22.48「論理ボリュームの作成」 の論理ボリューム **lv** に **ext3** ファイルシステムを作成するには、以下のコードスニペットを使用します。

```
(ret, outparams, err) = filesystem_service.SyncLMI_CreateFileSystem(
    FileSystemType=filesystem_service.LMI_CreateFileSystem.FileSystemTypeValues.EXT3,
    InExtents=[lv])
```

例22.50 ファイルシステムのマウント

例22.49「ファイルシステムの作成」 で作成したファイルシステムをマウントするには、以下のコードスニペットを使用します。

```
# Find the file system on the logical volume:
fs = lv.first_associator(ResultClass="LMI_LocalFileSystem")

mount_service = ns.LMI_MountConfigurationService.first_instance()
(rc, out, err) = mount_service.SyncCreateMount(
  FileSystemType='ext3',
  Mode=32768, # just mount
  FileSystem=fs,
  MountPoint='/mnt/test',
  FileSystemSpec=lv.Name)
```

例22.51 ブロックデバイスのリスト表示

システムが認識するブロックデバイスのリストを表示するには、以下のコードスニペットを使用します。

```
devices = ns.CIM_StorageExtent.instances()
for device in devices:
  if lmi_isinstance(device, ns.CIM_Memory):
    # Memory and CPU caches are StorageExtents too, do not print them
    continue
  print device.classname,
  print device.DeviceID,
  print device.Name,
  print device.BlockSize*device.NumberOfBlocks
```

OpenLMI ハードウェアプロバイダーの使用

openlmi-hardware パッケージは、ハードウェアを監視するための CIM プロバイダーをインストールします。以下の例では、この CIM プロバイダーを使用して CPU、メモリーモジュール、PCI デバイス、およびマシンの製造元およびモデルに関する情報を取得する方法を説明します。

例22.52 CPU 情報の表示

CPU 名、プロセッサーコア数、ハードウェアスレッド数などの基本的な CPU 情報を表示するには、以下のコードスニペットを使用します。

```
cpu = ns.LMI_Processor.first_instance()
cpu_cap = cpu.associators(ResultClass="LMI_ProcessorCapabilities")[0]
print cpu.Name
print cpu_cap.NumberOfProcessorCores
print cpu_cap.NumberOfHardwareThreads
```

例22.53 メモリー情報の表示

メモリーモジュールの個別サイズなどの基本情報を表示するには、以下のコードスニペットを使用します。

```
mem = ns.LMI_Memory.first_instance()
for i in mem.associators(ResultClass="LMI_PhysicalMemory"):
  print i.Name
```

例22.54 シャーシ情報の表示

製造元やモデルなどのマシンの基本情報を表示するには、以下のコードスニペットを使用します。

```
chassis = ns.LMI_Chassis.first_instance()
print chassis.Manufacturer
print chassis.Model
```

例22.55 PCI デバイスのリスト表示

システムが認識する PCI デバイスのリストを表示するには、以下のコードスニペットを使用します。

```
for pci in ns.LMI_PCIDevice.instances():
  print pci.Name
```

22.5. OPENLMI スクリプトの使用

LMIShell インタープリターは、カスタム管理ツールの開発に使用可能な Python モジュールに構築されます。OpenLMI スクリプトのプロジェクトは、OpenLMI プロバイダーへのインターフェイス用に Python ライブラリーを多数提供します。さらに、コマンドラインからこれらのライブラリーに対話するために使用可能な拡張ユーティリティーである lmi とともに配布されます。

OpenLMI スクリプトをシステムにインストールするには、シェルプロンプトに以下のコマンドを入力します。

```
easy_install --user openlmi-scripts
```

このコマンドで、Python モジュールと **lmi** ユーティリティーが **~/.local/** ディレクトリーにインストールされます。**lmi** ユーティリティーの機能を拡張するには、以下のコマンドを使用して追加の OpenLMI モジュールをインストールします。

```
easy_install --user package_name
```

利用可能なモジュールのリストは、Python の web サイト を参照してください。OpenLMI スクリプトの詳細は、OpenLMI スクリプトのドキュメント を参照してください。

22.6. 関連情報

OpenLMI およびシステム管理全般に関する詳細情報は、以下に挙げる資料を参照してください。

インストールされているドキュメント

- *lmishell(1)*: **lmishell** クライアントおよびインタープリターの man ページで、実行方法および使用方法の詳細情報が提供されています。

オンラインドキュメント

- *Red Hat Enterprise Linux 7 ネットワークガイド*: Red Hat Enterprise Linux 7 の Networking Guide では、システムのネットワークインターフェイスおよびネットワークサービスの設定および管理を説明しています。
- *Red Hat Enterprise Linux 7 ストレージ管理ガイド*: Red Hat Enterprise Linux 7 の Storage Administration Guide は、システムでのストレージデバイスおよびファイルシステムの管理方法を説明しています。
- *Red Hat Enterprise Linux 7 電力管理ガイド*: Red Hat Enterprise Linux 7 の Power Management Guide は、システムの電力消費量を効果的に管理する方法を説明しています。サーバーとノートパソコンの両方で電力消費量を抑えるテクニックと、各テクニックがシステムのパフォーマンス全体に与える影響を説明しています。
- *Red Hat Enterprise Linux 7 Linux ドメイン ID、認証、およびポリシーガイド*: Red Hat Enterprise Linux 7 の Linux Domain Identity, Authentication, and Policy Guide では、サーバーおよびクライアントを含む IPA ドメインのインストール、設定、管理について説明しています。このガイドは、IT およびシステム管理者用です。
- *FreeIPA Documentation*: FreeIPA Documentation は、FreeIPA Identity Management プロジェクトを使用する際のメインのユーザー資料となります。
- *OpenSSL Home Page*: OpenSSL のホームページでは、OpenSSL プロジェクトの概要が説明されています。
- *Mozilla NSS Documentation*: Mozilla NSS Documentation は、Mozilla NSS プロジェクトを使用する際にユーザーが主に使用するドキュメントとなります。

関連項目

- *4章ユーザーとグループの管理* では、グラフィカルユーザーインターフェイスとコマンドラインを使用したシステムユーザーとグループの管理方法を説明します。

- *9章Yum では、Yum パッケージマネージャーを使用して、コマンドラインでのパッケージの検索、インストール、更新、アンインストール方法について説明しています。*

- *10章systemd によるサービス管理 では、**systemd** の概説と、**systemctl** コマンドを使用したシステムサービスの管理方法、systemd ターゲットの設定方法、および電源管理コマンドの実行方法について説明しています。*

- *12章OpenSSH は、SSH サーバーの設定方法や、クライアントユーティリティーの**ssh**、**scp**、および**sftp** を使用してこのサーバーにアクセスする方法を説明しています。*

第23章 ログファイルの表示と管理

ログファイルは、システム (カーネル、サービス、および実行中のアプリケーションなど) に関するメッセージが含まれるファイルです。各情報にはそれぞれ異なるログファイルがあります。例えば、デフォルトのシステムログファイル、セキュリティーメッセージ専用のログファイル、cron タスク用のログファイルなどです。

ログファイルは、カーネルドライバーのロードを試行するなどシステムの問題を解決する場合や、システムへの無許可のログイン試行を探す場合に役立ちます。本章では、ログファイルの場所、ログファイルの閲覧方法、ログファイルの注意すべき項目を説明します。

一部のログファイルは、**rsyslogd** という名前のデーモンによって制御されます。**rsyslogd** デーモンは、sysklogd の拡張版であり、拡張されたフィルタリング、暗号化で保護されたメッセージリレー、様々な設定オプション、入出力モジュール、**TCP** または **UDP** プロコトルを介した伝送のサポートを提供します。rsyslog と sysklogd は互換性があることに注意してください。

ログファイルは、**systemd** のコンポーネントである **journald** デーモンで制御することもできます。**journald** デーモンは全サービスの標準出力および標準エラー出力に書き込まれたメッセージに加えて、Syslog メッセージ、カーネルログメッセージ、初期 RAM ディスクおよび初期起動メッセージを取り込みます。構造化およびインデックス化されたバイナリーファイルであるネイティブジャーナルファイル形式は、検索を改善します。また、より高速に操作を提供するため、タイムスタンプやユーザー ID などのメタデータ情報も格納します。**journald** が生成するログファイルはデフォルトで永続的ではなく、ログファイルは **/run/log/journal/** ディレクトリー内のメモリーまたはリングバッファーにのみ保存されます。ログに記録されるデータの量は、容量制限に達すると、最も古いエントリーが削除されます。ただし、この設定は変更できます。「永続的ストレージの有効化」 を参照してください。ジャーナルの詳細情報は、「Journal の使用」 を参照してください。

デフォルトでは、これら 2 つのロギングツールはシステム上で共存しています。**journald** デーモンは、トラブルシューティング用の主要ツールです。また、構造化ログメッセージの作成に必要な追加データも提供します。**journald** が取得したデータは、**/run/systemd/journal/syslog** ソケットに転送され、さらにデータを処理するために **rsyslogd** が使用する場合があります。ただし、rsyslog は **imjournal** 入力モジュールを介して実際の統合を行うため、上記のソケットを回避します。また、**omjournal** モジュールを使用して **rsyslogd** から **journald** に逆方向でデータを移動することもできます。詳細は、「Rsyslog と Journal の相互作用」 を参照してください。統合によりテキストベースのログを一貫性のある形式で保持できることになり、**rsyslogd** に依存するアプリケーションや設定との互換性を確保できます。また、構造化形式で rsyslog メッセージを保持することもできます (「Rsyslog での構造化ロギング」 を参照)。

23.1. ログファイルの場所の特定

rsyslogd により保持されるログファイルのリストは **/etc/rsyslog.conf** 設定ファイルにあります。ほとんどのログファイルは **/var/log/** ディレクトリーにあります。**httpd** および **samba** などの一部のアプリケーションでは、ログファイル用のディレクトリーが **/var/log/** 内にあります。

/var/log/ ディレクトリー内には末尾に番号が付いた複数のファイル (例: **cron-20100906**) があることに気付くかもしれません。これらの番号はローテーションを行ったログファイルに追加されたタイムスタンプを表します。ログファイルは、ファイルサイズが大きくなり過ぎないようにローテーションが行われます。**logrotate** パッケージには cron タスクが含まれており、これが **/etc/logrotate.conf** 設定ファイルと **/etc/logrotate.d/** ディレクトリー内の設定ファイルに従って自動的にログファイルのローテーションを行います。

23.2. RSYSLOG の基本設定

rsyslog の主な設定ファイルは **/etc/rsyslog.conf** です。このファイルでは、**グローバルディレクティブ**、**モジュール**、および**フィルター** と**アクション** の部分で設定される**ルール** を指定できます。また、ハッシュ記号 **(#)** の後にテキスト形式でコメントを追加することもできます。

23.2.1. フィルター

ルールは、syslog メッセージのサブセットを選択する **フィルター** の部分と、選択したメッセージで何をするかを指定する **アクション** の部分で指定されます。**/etc/rsyslog.conf** 設定ファイル内でルールを定義するには、フィルターとアクションの両方を同じ行に、1 つ以上の空白かタブで区切って定義します。

rsyslog は、選択されたプロパティーに従って syslog メッセージをフィルターする様々な方法を提供します。利用可能なフィルタリングの方法は、Facility/Priority **ベース**、Property **ベース**、Expression **ベース** の 3 種類のフィルターに分けられます。

Facility (ファシリティー) /Priority (優先度) ベースのフィルター

syslog メッセージをフィルターする最もよく知られた方法は、syslog メッセージをフィルターするファシリティー/priority ベースのフィルターを使用して、2 つの条件 (**ファシリティー** および **優先度** をドットで区切る) を使用することです。セレクターを作成するには以下の構文を使用します。

> **FACILITY.PRIORITY**

詳細は以下のようになります。

- FACILITY は、特定の syslog メッセージを作成するサブシステムを指定します。たとえば、**mail** サブシステムはメール関連のすべての syslog メッセージを処理します。FACILITY は、以下のキーワード (または数字コード) のいずれかで表すことができます。**kern** (0), **user** (1)、**mail** (2)、**daemon** (3)、**auth** (4), **syslog** (5)、**lpr** (6)、**news** (7)、**cron** (8)、**authpriv** (9)、**ftp** (10)、および **local0** ~ **local7** (16 ~ 23)。

- PRIORITY は、syslog メッセージの優先度を指定します。PRIORITY は以下のキーワード (または数字) のいずれかで表示できます。**debug** (7)、**info** (6)、**notice** (5)、**warning** (4)、**err** (3)、**crit** (2)、**alert** (1)、および **emerg** (0)。
  上記の構文は、定義された優先度もしくは **より高い** 優先度で syslog メッセージを選択します。いずれかの優先度のキーワード前に等号 **(=)** を付けると、指定された優先度の syslog メッセージのみ選択されることが指定されます。他のすべての優先度は無視されます。反対に、感嘆符 **(!)** を優先度のキーワードの前に付けると、この優先度以外のすべての syslog メッセージが選択されます。

  上記で指定されているキーワード以外に、アスタリスク **(*)** を使用してすべてのファシリティーもしくは優先度を定義することもできます (アスタリスクをコンマの前か後に配置するかによる)。優先度キーワード **none** を指定すると、優先度のないファシリティーを指定することになります。ファシリティーおよび優先度の条件は、どちらも大文字と小文字を区別しません。

  定義するファシリティーや優先度が複数になる場合は、コンマ **(,)** を使用して区切ります。同じ行に複数のセレクターを定義するには、セミコロン **(;)** を使用して区切ります。セレクターフィールド内の各セレクターは、以前のものを上書きすることに注意してください。これにより、パターンから優先度が除外される可能性があります。

  > *例23.1 ファシリティー/優先度ベースのフィルター*
  >
  > 以下は、**/etc/rsyslog.conf** で指定できる単純な facility/priority ベースのフィルターの例です。優先度ですべてのカーネル syslog メッセージを選択するには、設定ファイルに以下のテキストを追加します。

> ***kern.****
>
> *優先度が **crit** 以上になるメール **syslog** メッセージすべてを選択するには、以下の形式を使用します。*
>
> ***mail.crit***
>
> ***info** または **debug** 優先度以外のすべての **cron syslog** メッセージを選択するには、設定ファイルで以下の形式を設定します。*
>
> ***cron.!info,!debug***

プロパティーベースのフィルター

*プロパティーベースのフィルターでは、**timegenerated** や **syslogtag** などのプロパティーで **syslog** メッセージのフィルター処理ができます。プロパティーに関する詳細情報は、「プロパティー」 を参照してください。指定された各プロパティーは、表23.1「プロパティーベースの比較処理」 でリスト表示されている比較処理のいずれかを使用して特定の値に対して比較できます。プロパティー名と比較処理はどちらも大文字と小文字を区別します。*
*プロパティーベースのフィルターは、コロン (**:**) で開始する必要があります。フィルターの定義には、以下の構文を使用します。*

> **:PROPERTY, *[!]*COMPARE_OPERATION, "STRING"**

詳細は以下のようになります。

- PROPERTY *属性は希望するプロパティーを指定します。*
- *オプションの感嘆符 (**!**) は比較処理の出力を無効にします。他のブール値演算子は現在、プロパティーベースのフィルターではサポートされていません。*
- COMPARE_OPERATION *属性は、表23.1「プロパティーベースの比較処理」 にリスト表示してある比較処理のいずれかを指定します。*
- STRING *属性は、プロパティーが提供するテキストの比較対象となる値を指定します。この値は、引用符で囲む必要があります。この文字列内の特定の文字をエスケープさせるには (たとえば、引用符 (**"**))、バックスラッシュ (****) を使用します。*

  表23.1 プロパティーベースの比較処理

比較処理	詳細
contains	提供された文字列が、プロパティーで提供されたテキストのいずれかの部分に適合するかどうかをチェックします。
isequal	用意された文字列をプロパティーで提供されたすべてのテキストと比較します。これら 2 つの値が適合するには、完全に等しいものである必要があります。

比較処理	詳細
startswith	提供された文字列が、プロパティーで提供されたテキストのちょうど最初にあるかどうかをチェックします。
regex	指定された POSIX BRE (Basic Regular Expression) をプロパティーが提供したテキストと比較します。
ereregex	指定された POSIX ERE (Extended Regular Expression) 正規表現をプロパティーが提供したテキストと比較します。
isempty	プロパティーが空かどうかをチェックします。値は破棄されます。これは、いくつかのフィールドが正規化の結果に基づいて設定される正規化データでの作業時に特に有用です。

例23.2 プロパティーベースのフィルター

*以下は、**/etc/rsyslog.conf** で指定できるプロパティーベースのフィルターの例です。**syslog** メッセージのテキストに文字列 **error** が含まれているものを選択するには、以下を使用します。*

```
:msg, contains, "error"
```

*以下のフィルターは、ホスト名 **host1** から受信した **syslog** メッセージを選択します。*

```
:hostname, isequal, "host1"
```

*(**fatal** **lib** **error** など) **fatal** と **error** の間にテキストがあるかどうかに関わらず、これらを含まない **syslog** メッセージを選択するには、以下を入力します。*

```
:msg, !regex, "fatal .* error"
```

式ベースのフィルター

*式ベースのフィルターは、定義されている算術演算、ブール演算、または文字列演算に従って **syslog** メッセージを選択します。複雑なフィルターを構築するために、式ベースのフィルターは、RainerScript と呼ばれる **rsyslog** の独自のスクリプト言語を使用します。*
式ベースのフィルターの基本的な構文は、以下のようになります。

```
if EXPRESSION then ACTION else ACTION
```

詳細は以下のようになります。

- EXPRESSION 属性は、**$msg startswith 'DEVNAME'** や**$syslogfacility-text == 'local0'** などの評価される式を表します。**and** および**or** 演算子を使用することで、単一フィルター内に複数の式を指定できます。
  rsyslog は、式ベースのフィルターでは、大文字と小文字を区別しない比較をサポートすることに注意してください。EXPRESSION 属性内の**contains_i** または**startswith_i** compare-operations を使用できます。以下に例を示します。

  if $hostname startswith_i "<HOST_NAME>" then ACTION.

- ACTION 属性は、式が**true** の値を返す場合に実行される動作を表します。これは単一のアクションの場合と、波括弧で囲まれた任意の複雑なスクリプトになる場合があります。

- 式ベースのフィルターは、行の最初のif キーワードで示されます。then キーワードは、EXPRESSION をACTION から離します。オプションで、else キーワードを使用して条件が満たされない場合に実行されるアクションを指定することもできます。
  式ベースのフィルターでは、例23.3「式ベースのフィルター」 にあるように、波括弧に囲まれた式を使うことで条件をネスト化することができます。このスクリプトでは、式内でfacility/priority-based フィルターを使用することができます。ただし、ここでproperty-based フィルターを使用することは推奨されません。RainerScript は、特別関数**re_match()** および**re_extract()** を伴う正規表現をサポートします。

  例23.3 式ベースのフィルター

  以下の式には、ネスト化された条件が2 つ含まれています。prog1 と呼ばれるプログラムが生成したログファイルが、メッセージ内の文字列 "test" の有無に基づいて2 つのファイルに分割されます。

```
if $programname == 'prog1' then {
  action(type="omfile" file="/var/log/prog1.log")
  if $msg contains 'test' then
   action(type="omfile" file="/var/log/prog1test.log")
  else
   action(type="omfile" file="/var/log/prog1notest.log")
}
```

様々な式ベースのフィルターの他の例は、「オンラインドキュメント」 を参照してください。
RainerScript は rsyslog の新しい設定形式の基礎となります。「新規設定フォーマットの使用」 を参照してください。

23.2.2. アクション

アクションは、定義済みのセレクターでフィルターされたメッセージで実行すべきことを指定します。以下にルール内で定義できるアクションをいくつか示します。

ログファイルへのsyslog メッセージの保存

アクションの大半は、どのログファイルにsyslog メッセージを保存するかを指定します。これは定義済みセレクターの後にファイルパスを指定することで行います。

FILTER PATH

ここでのFILTER はユーザーが指定したセレクターを表し、PATH はターゲットファイルのパスを表します。

たとえば、以下のルールは、すべてのcron syslog メッセージを選択するセレクターと、それらのメッセージを/var/log/cron.log ログファイルに保存するアクションで設定されています。

```
cron.* /var/log/cron.log
```

デフォルトでは、syslog メッセージの生成時に毎回ログファイルは同期されます。同期を省略する場合は、ダッシュ記号(-) を該当するファイルパスの接頭辞として使います。

```
FILTER -PATH
```

書き込みの直後にシステムが終了すると、情報が失われる場合があることに注意してください。ただし、この設定では、特に非常に詳細なログメッセージを生成するプログラムを実行する場合には、パフォーマンスも改善されます。

指定したファイルパスは、static またはdynamic のいずれかになります。静的ファイルは、上記の例で示されているように固定ファイルパスで示されます。動的ファイルパスは、受け取ったメッセージによって異なります。動的ファイルパスは、テンプレートと疑問符(?) の接頭辞で示されます。

```
FILTER ?DynamicFile
```

ここでは、DynamicFile は出力パスを修正する定義済みテンプレート名になります。ダッシュ記号(-) の接頭辞を使うと同期を無効にでき、またコロン(;) 区切りで複数のテンプレートを使用できます。テンプレートに関する詳細は 「動的なファイル名の生成」 を参照してください。

指定したファイルが既存のterminal または/dev/console デバイスである場合、syslog メッセージは標準出力(特別なterminal 処理を使用) へ送信されるか、X Window システムの使用時には使用中のコンソール(特別な/dev/console 処理を使用) へそれぞれ送信されます。

ネットワークを使用したsyslog メッセージの送信

rsyslog を使用すると、ネットワークを使用してsyslog メッセージを送受信できます。この機能により、1 台のマシンで複数ホストのsyslog メッセージを管理できます。syslog メッセージをリモートマシンに転送するには、以下の構文を使用します。

```
@(zNUMBER)HOST:PORT
```

詳細は以下のようになります。

- アットマーク(@) は、syslog メッセージがUDP プロトコルを使用してホストへ転送されることを示します。TCP プロトコルを使用するには、2 つのアットマークを空白なしで(@@) 使用します。
- オプションのzNUMBER 設定を使用すると、syslog メッセージのzlib 圧縮が可能になります。NUMBER 属性は、圧縮レベルを指定します(最低の1 から最高の9 まで)。圧縮が得られたことはrsyslogd が自動的にチェックします。メッセージが圧縮されるのは圧縮が可能になった場合のみで、60 バイト未満のメッセージは圧縮されません。
- HOST 属性は、選択したsyslog メッセージを受信するホストを指定します。
- PORT 属性は、ホストマシンのポートを指定します。
  IPv6 アドレスをホストとして指定する場合は、アドレスを角括弧([,]) で囲みます。

例23.4 ネットワークを使用したsyslog メッセージの送信

> *以下の例は、ネットワーク上で syslog メッセージを転送するアクションです (注記: すべてのアクションの前には、いずれかの優先度を持つすべてのメッセージを選択するセレクターが付いています)。メッセージを **UDP** 経由で **192.168.0.1** に転送するには、以下を入力します。*
>
> ***. @192.168.0.1***
>
> *ポート 6514 と **TCP** プロトコルを使用してメッセージを "example.com" に転送するには、以下を使用します。*
>
> ***. @@example.com:6514***
>
> *以下ではメッセージを zlib (レベル 9 圧縮) で圧縮し、**UDP** プロトコルを使用して **2001:db8::1** に転送します。*
>
> ***. @(z9)[2001:db8::1]***

出力チャンネル

*出力チャンネルは主にログファイルの最大サイズを指定するために使用されます。これは、ログファイルのローテーションに非常に便利です (詳細は 「ログローテーション」 を参照してください)。出力チャンネルは基本的に出力アクションに関する情報を集めたものです。出力チャンネルは、**$outchannel** ディレクティブで定義されます。**/etc/rsyslog.conf** で出力チャンネルを定義するには、以下の構文を使用します。*

> ***$outchannel** NAME, FILE_NAME, MAX_SIZE, ACTION*

詳細は以下のようになります。

- NAME *属性は、出力チャンネル名を指定します。*
- FILE_NAME *属性は、出力ファイル名を指定します。出力チャンネルはファイルにのみ書き込み可能で、パイプやターミナル、その他の出力には書き込みできません。*
- MAX_SIZE *属性は、(*FILE_NAME *内にある) 指定されたファイルが拡張可能な最大サイズを表します。この値は* **バイト** *単位で指定します。*
- ACTION *属性は、*MAX_SIZE *で定義された最大サイズに到達した際に取るべきアクションを指定します。*
  定義済みの出力チャンネルをルール内のアクションとして使用するには、以下を入力します。

  > **FILTER *:omfile:*$NAME**

  > *例23.5 出力チャンネルのログローテーション*
  >
  > *以下の出力は、出力チャンネルを使用した簡単なログローテーションを示しています。まず、出力チャンネルは **$outchannel** ディレクティブにより定義されます。*
  >
  > ***$outchannel log_rotation, /var/log/test_log.log, 104857600, /home/joe/log_rotation_script***

その後に、優先度を持つすべての syslog メッセージを選択し、取得した syslog メッセージ上の事前定義された出力チャンネルを実行するルール内で使用されます。

```
. :omfile:$log_rotation
```

制限 (例では 100 MB) に達すると、**/home/joe/log_rotation_script** が実行されます。このスクリプトには、ファイルを異なるフォルダーに移動することやその中の特別なコンテンツを編集すること、単にそれを削除することなど、様々なタスクを含めることができます。

特定ユーザーへの syslog メッセージの送信

メッセージの送信先となるユーザー名を指定することで、rsyslog は syslog メッセージを送信することができます (例23.7「複数アクションの指定」 で表示)。複数のユーザーを指定するには、各ユーザー名をコンマ (,) で区切ります。現在ログオンしている全ユーザーにメッセージを送るには、アスタリスク (*) を使用します。

プログラムの実行

rsyslog を使用すると、選択した syslog メッセージに対してプログラムを実行でき、**system()** 呼び出しを使用してシェルでプログラムを実行できます。実行するプログラムを指定するには、そのプログラムの前に キャレット文字 (^) を付けます。その後に、受信したメッセージをフォーマットしてそれを 1 行のパラメーターとして指定した実行ファイルに渡すテンプレートを指定します (テンプレートに関する詳細は 「テンプレート」 を参照)。

```
FILTER ^EXECUTABLE; TEMPLATE
```

ここでは、FILTER 条件の出力は EXECUTABLE で表されるプログラムで処理されます。このプログラムは、有効な実行ファイルであればどれでも構いません。TEMPLATE をフォーマットするテンプレートに置き換えます。

例23.6 プログラムの実行

以下の例では、すべての優先度の syslog メッセージが選択され、**template** テンプレートでフォーマットされた後にパラメーターとして test-program プログラムに渡されます。その後は、提供されているパラメーターで実行されます。

```
. ^test-program;template
```

警告

ホストからメッセージを受信して、シェル実行アクションを使用する際には、コマンドインジェクションに対する脆弱性があります。ユーザーが自身のアクションで実行されるように指定しているプログラム内で、攻撃者が別のコマンドの挿入と実行を試みる可能性があります。セキュリティー脅威の可能性を回避するには、シェル実行アクションの使用をよく検討してください。

syslog メッセージのデータベースでの保存

*選択された syslog メッセージは、**データベースライター** の動作を使用して、直接データベーステーブルに書き込むことができます。データベースライターは、以下の構文を使用します。*

```
:PLUGIN:DB_HOST,DB_NAME,DB_USER,DB_PASSWORD;TEMPLATE
```

詳細は以下のようになります。

- PLUGIN *はデータベースの書き込みを処理する指定プラグインを呼び出します (例えば、**ommysql** プラグイン)。*
- DB_HOST *属性は、データベースのホスト名を指定します。*
- DB_NAME *属性はデータベースの名前を指定します。*
- DB_USER *属性はデータベースのユーザーを指定します。*
- DB_PASSWORD *属性は上述のデータベースユーザーが使用するパスワードを指定します。*
- TEMPLATE *属性は syslog メッセージを修正するテンプレートのオプション使用を指定します。テンプレートに関する詳細は 「テンプレート」 を参照してください。*

重要

*現在 rsyslog は、**MySQL** と **PostgreSQL** データベースにのみ対応しています。**MySQL** および **PostgreSQL** のデータベースライター機能を使用するには、rsyslog-mysql および rsyslog-pgsql パッケージをそれぞれインストールします。また、**/etc/rsyslog.conf** 設定ファイルに適切なモジュールを読み込んでください。*

```
module(load="ommysql")  # Output module for MySQL support
module(load="ompgsql")  # Output module for PostgreSQL support
```

rsyslog モジュールの詳細は 「Rsyslog モジュールの使用」 を参照してください。

*もしくは、**omlibdb** モジュールが提供する汎用のデータベースインターフェイスを使用することもできます (サポート対象: Firebird/Interbase、MS SQL、Sybase、SQLLite、Ingres、Oracle、mSQL)。*

syslog メッセージの破棄

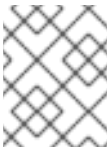
*選択したメッセージを破棄する場合は、**stop** を使用します。*
破棄するアクションは、ほとんどの場合、さらに処理をする前にメッセージをフィルタリングするために使用されます。繰り返されるメッセージを省略しなければログファイルがいっぱいになってしまう場合に、これは効果的です。破棄アクションの結果は、指定された設定ファイル内の場所によって異なります。最善の結果を得るためにも、これらのアクションは、アクションリストの上に配置します。一旦破棄したメッセージを、設定ファイルの後の行で回復することはできないことに注意してください。

*たとえば、以下のルールを使用すると、**local5.*** フィルターに一致するメッセージをすべて破棄します。*

```
local5.* stop
```

以下の例では、cron syslog メッセージがすべて破棄されます。

```
cron.* stop
```



注記

*rsyslog 7 より前のバージョンでは、syslog メッセージを破棄する際に、**stop** の代わりにチルダ文字 (~) が使用されていました。*

複数アクションの指定
各セレクターで、複数のアクションを指定できます。1 つのセレクターに複数アクションを指定するには、各アクションを別々の行に書き込んでそれらの先頭にアンパサンド文字 (&) を付けます。

```
FILTER ACTION
& ACTION
& ACTION
```

指定されたセレクターが評価されるのは 1 回のみであるため、複数の動作を指定すると、希望する結果の全体的なパフォーマンスが向上します。

例23.7 複数アクションの指定

*下記の例では、重大の優先度 (**crit**) を持つすべてのカーネル syslog メッセージはユーザー**user1** に送信され、テンプレート **temp** によって処理されてから、**test-program** 実行ファイルに渡され、その後に **UDP** プロトコルを介して **192.168.0.1** に転送されます。*

```
kern.=crit user1
& ^test-program;temp
& @192.168.0.1
```

すべてのアクションで、メッセージをフォーマットするテンプレートが後に続きます。テンプレートを指定するには、アクションの末尾にセミコロン (;) を付け、続けてテンプレートの名前を指定します。テンプレートに関する詳細は 「テンプレート」 を参照してください。

警告

*テンプレートはアクションで使用される前に定義する必要があり、アクションの後に定義しても無視されます。つまり、**/etc/rsyslog.conf** では、テンプレート定義が常にルール定義の前にくるようにする必要があります。*

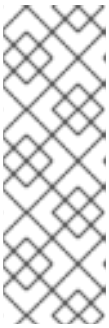
23.2.3. テンプレート

*rsyslog で生成された出力はすべて、**テンプレート** を使用して、ユーザーのニーズに応じて修正とフォーマット設定ができます。テンプレートを作成するには、**/etc/rsyslog.conf** に以下の構文を使用します。*

```
template(name="TEMPLATE_NAME" type="string" string="text %PROPERTY% more text"
[option.OPTION="on"])
```

詳細は以下のようになります。

- **template()** はテンプレートを定義するディレクティブ導入ブロックです。
- **TEMPLATE_NAME** の必須引数はテンプレートの参照に使用します。**TEMPLATE_NAME** は固有にする必要があります。
- **type** の必須引数は list、subtree、string、plugin のいずれかの値を取得します。
- **string** 引数は実際のテンプレートテキストです。このテキスト内では、改行文字の **\n**、キャリッジリターンの **\r** などの特殊文字を使用できます。% または " などのその他の文字を使用する場合は、それらの文字を文字どおりエスケープする必要があります。このテキスト内では、改行文字の **\n**、キャリッジリターンの **\r** などの特殊文字を使用できます。**%** または **"** などのその他の文字を使用する場合は、それらの文字を文字どおりエスケープする必要があります。
- 2 つのパーセントマーク (**%**) の間にあるテキストは、syslog メッセージの特定のコンテンツにアクセスできるようにする **プロパティー** を指定します。プロパティーに関する詳細情報は、「プロパティー」 を参照してください。
- **OPTION** 属性は、テンプレート機能を修正するオプションを指定します。現在サポートされているテンプレートオプションは、テキストを SQL クエリーとしてフォーマット作成する **sql** と **stdsql**、または JSON 処理に適した形にテキストをフォーマットする JSON です。**casesensitive** はプロパティー名の大文字小文字の区別を設定します。



注記

データベースライターは、**sql** オプションまたは **stdsql** オプションがテンプレート内で指定されているかどうかをチェックします。指定されていないと、データベースライターはアクションを実行しません。これは SQL インジェクションなどのセキュリティーの脅威を回避するためです。

詳細は、「アクション」 のsyslog **メッセージのデータベースでの保存** を参照してください。

動的なファイル名の生成
テンプレートを使用して、動的なファイル名を生成できます。プロパティーをファイルパスの一部として指定すると、それぞれ一意のプロパティーに対して新しいファイルが作成されます。これは、syslog メッセージを分類する便利な方法です。

たとえば、メッセージからタイムスタンプを抽出する **timegenerated** プロパティーを使用すると、各 syslog メッセージに一意のファイル名を生成できます。

```
template(name="DynamicFile" type="list") {
constant(value="/var/log/test_logs/")
property(name="timegenerated")
constant(value"-test.log")
}
```

$template ディレクティブは単にテンプレートを指定するだけです。効果を反映するためには、それをルール内で使用しなければなりません。**/etc/rsyslog.conf** 内のアクション定義でクエスチョンマーク (**?**) を使用して、動的ファイル名テンプレートをマークします。

```
. ?DynamicFile
```

プロパティー

テンプレート内 (2 つのパーセントマーク (%) の内側) で定義されたプロパティーにより、**プロパティー置換関数** を使用して syslog メッセージの各種コンテンツにアクセスできるようになります。テンプレート内 (2 つの引用符 (“...") の間) でプロパティーを定義するには、以下の構文を使用します。

```
%PROPERTY_NAME:FROM_CHAR:TO_CHAR:OPTION%
```

詳細は以下のようになります。

- PROPERTY_NAME 属性は、プロパティー名を指定します。利用可能なすべてのプロパティーとその詳細な説明は、man ページ **rsyslog.conf(5)** の Available Properties セクションでご覧になれます。
- FROM_CHAR 属性と TO_CHAR 属性は、指定したプロパティーが動作する文字の範囲を表します。他の方法では、正規表現を使用して文字の範囲を指定することもできます。これを行うには、文字 **R** を FROM_CHAR 属性として指定し、希望する正規表現を TO_CHAR 属性として指定します。
- OPTION 属性は、入力を小文字に変換する **lowercase** オプションなど、プロパティーオプションを指定します。利用可能なすべてのプロパティーオプションとその詳細な説明は、**rsyslog.conf(5)** の man ページの Property Options セクションでご覧になれます。

簡単なプロパティーの例を以下に示します。

- 以下のプロパティーは、syslog メッセージのメッセージテキスト全体を取得します。

  ```
  %msg%
  ```

- 以下のプロパティーは、syslog メッセージにあるメッセージテキストの最初の 2 文字を取得します。

  ```
  %msg:1:2%
  ```

- 以下のプロパティーは、syslog メッセージの全メッセージテキストを取得して、最後のラインフィード文字を省きます。

  ```
  %msg:::drop-last-lf%
  ```

- 以下のプロパティーは、syslog メッセージを受信し、RFC 3999 日付基準に従ってそれをフォーマットした時に生成されるタイムスタンプの最初の 10 文字を取得します。

  ```
  %timegenerated:1:10:date-rfc3339%
  ```

テンプレートの例
このセクションでは、rsyslog テンプレートの例をいくつか示しています。

例23.8「詳細な syslog メッセージのテンプレート」 は、syslog メッセージのフォーマットを作成するテンプレートを示しています。これにより、メッセージの重要性、機能、メッセージ受信時のタイムスタンプ、ホスト名、メッセージのタグ、メッセージテキストが出力され、改行後に終了します。

例23.8 詳細な syslog メッセージのテンプレート

```
template(name=”verbose” type=”list”) {
property(name="syslogseverity”)
property(name="syslogfacility”)
```

```
  property(name="timegenerated")
  property(name="HOSTNAME")
  property(name="syslogtag")
  property(name="msg")
  constant(value="\n")
}
```

例23.9「ウォールメッセージのテンプレート」は、従来のウォールメッセージ (ログインしていてその **mesg(1)** パーミッションが **yes** に設定されている全ユーザーに送信されるメッセージ) に似ているテンプレートを示しています。このテンプレートは改行後 (**\r** と **\n** を使用) にメッセージテキストと共にホスト名、メッセージタグ、およびタイムスタンプを出力してベル (**\7** を使用) を鳴らします。

例23.9 ウォールメッセージのテンプレート

```
  template(name="wallmsg" type="list") {
  constant(value="\r\n\7Message from syslogd@")
  property(name="HOSTNAME")
  constant(value=" at ")
  property(name="timegenerated")
  constant(value=" ...\r\n ")
  property(name="syslogtag")
  constant(value=" ")
  property(name="msg")
  constant(value="\r\n")
}
```

例23.10「データベースフォーマットが設定されたメッセージのテンプレート」は、syslog メッセージのフォーマットを作成してデータベースクエリーとして使用できるようにするテンプレートを示しています。テンプレートの末尾でテンプレートオプションとして指定されている **sql** オプションに注目してください。これにより、データベースライターが、メッセージを MySQL **SQL** クエリーのフォーマットに指定します。

例23.10 データベースフォーマットが設定されたメッセージのテンプレート

```
template(name="dbFormat" type="list" option.sql="on") {
constant(value="insert into SystemEvents (Message, Facility, FromHost, Priority,
DeviceReportedTime, ReceivedAt, InfoUnitID, SysLogTag)")
constant(value=" values ('")
property(name="msg")
constant(value="', ")
property(name="syslogfacility")
constant(value=", '")
property(name="hostname")
constant(value="', ")
property(name="syslogpriority")
constant(value=", '")
property(name="timereported" dateFormat="mysql")
constant(value="', '")
property(name="timegenerated" dateFormat="mysql")
constant(value="', ")
property(name="iut")
constant(value=", '")
```

```
property(name="syslogtag")
constant(value="")")
}
```

*rsyslog には、**RSYSLOG_** 接頭辞で識別される事前定義のテンプレートのセットも含まれています。これらは syslog の使用に確保されており、競合を防止するためにこの接頭辞を使用したテンプレートを作成しないことが推奨されます。以下のリストでは、これらの事前定義のテンプレートとその定義を示しています。*

RSYSLOG_DebugFormat

プロパティー問題のトラブルシューティングに使われる特別なフォーマット。

```
template(name="RSYSLOG_DebugFormat" type="string" string="Debug line with all
properties:\nFROMHOST: '%FROMHOST%', fromhost-ip: '%fromhost-ip%', HOSTNAME:
'%HOSTNAME%', PRI: %PRI%,\nsyslogtag '%syslogtag%', programname:
'%programname%', APP-NAME: '%APP-NAME%', PROCID: '%PROCID%', MSGID:
'%MSGID%',\nTIMESTAMP: '%TIMESTAMP%', STRUCTURED-DATA: '%STRUCTURED-
DATA%',\nmsg: '%msg%'\nescaped msg: '%msg:::drop-cc%'\nrawmsg: '%rawmsg%'\n\n")
```

RSYSLOG_SyslogProtocol23Format

IETF のインターネットドラフト ietf-syslog-protocol-23 で指定されるフォーマット。これは新たな syslog 標準 RFC になると想定されています。

```
template(name="RSYSLOG_SyslogProtocol23Format" type="string" string="%PRI%1
%TIMESTAMP:::date-rfc3339% %HOSTNAME% %APP-NAME% %PROCID% %MSGID%
%STRUCTURED-DATA% %msg%\n ")
```

RSYSLOG_FileFormat

TraditionalFileFormat に類似した新しいスタイルのログファイルフォーマットですが、タイムスタンプとタイムゾーン情報の精度がより高くなります。

```
template(name="RSYSLOG_FileFormat" type="list") {
property(name="timestamp" dateFormat="rfc3339")
constant(value=" ")
property(name="hostname")
constant(value=" ")
property(name="syslogtag")
property(name="msg" spifno1stsp="on" )
property(name="msg" droplastlf="on" )
constant(value="\n")
}
```

RSYSLOG_TraditionalFileFormat

タイムスタンプの精度の低い古いスタイルのデフォルトのログファイルフォーマット。

```
template(name="RSYSLOG_TraditionalFileFormat" type="list") {
property(name="timestamp")
constant(value=" ")
property(name="hostname")
constant(value=" ")
property(name="syslogtag")
```

```
property(name="msg" spifno1stsp="on" )
property(name="msg" droplastlf="on" )
constant(value="\n")
}
```

RSYSLOG_ForwardFormat

高精度のタイムスタンプとタイムゾーン情報が含まれる転送フォーマット。

```
template(name="ForwardFormat" type="list") {
constant(value="<")
property(name="pri")
constant(value=">")
property(name="timestamp" dateFormat="rfc3339")
constant(value=" ")
property(name="hostname")
constant(value=" ")
property(name="syslogtag" position.from="1" position.to="32")
property(name="msg" spifno1stsp="on" )
property(name="msg")
}
```

RSYSLOG_TraditionalForwardFormat

精度の低いタイムスタンプを使用する従来の転送フォーマット。

```
template(name="TraditionalForwardFormat" type="list") {
constant(value="<")
property(name="pri")
constant(value=">")
property(name="timestamp")
constant(value=" ")
property(name="hostname")
constant(value=" ")
property(name="syslogtag" position.from="1" position.to="32")
property(name="msg" spifno1stsp="on" )
property(name="msg")
}
```

23.2.4. グローバルディレクティブ

グローバルディレクティブは、**rsyslogd** デーモンに適用される設定オプションです。通常、これらは **rsyslogd** デーモンの動作に影響を与える事前定義された特定の変数の値、または生じるルールを指定します。グローバルディレクティブはすべて、**global** 設定ブロックに含まれています。以下は、上書きするログメッセージのローカルホスト名を指定するグローバルディレクティブのサンプルです。

```
global(localHostname="machineXY")
```

/etc/rsyslog.conf 設定ファイルで複数のディレクティブを定義することもできます。1 つのディレクティブは、同じディレクティブの発生が再度検出されるまですべての設定オプションの動作に影響します。グローバルディレクティブは、アクション、キュー、デバッグの設定に使用できます。利用可能なすべての設定ディレクティブのリストは、「オンラインドキュメント」 でご覧になれます。現在、$

ベースの構文 (「新規設定フォーマットの使用」 を参照) に代わる新たな設定フォーマットが開発されています。ただし、従来のグローバルディレクティブはレガシーフォーマットとして引き続きサポートされます。

23.2.5. ログローテーション

以下に、**/etc/logrotate.conf** 設定ファイルのサンプルを示します。

```
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# uncomment this if you want your log files compressed
compress
```

上記の設定ファイル内のすべての行は、すべてのログファイルに適用されるグローバルオプションを定義しています。この例では、ログファイルが週ごとにローテーションされ、ローテーションされたログファイルは 4 週間保持され、ローテーションされるログファイルはすべて **gzip** により **.gz** 形式に圧縮されます。ハッシュマーク (**#**) で始まる行はすべてコメントで、これは処理されません。

特定のログファイル用に設定オプションを定義して、それをグローバルオプションの下に置くこともできます。ただし、**/etc/logrotate.d/** ディレクトリーに、特定ログファイル用の個別の設定ファイルを作成し、そこに設定オプションを定義することが推奨されます。

/etc/logrotate.d/ ディレクトリーに、設定ファイルが保存されている例を以下に示します。

```
/var/log/messages {
  rotate 5
  weekly
  postrotate
  /usr/bin/killall -HUP syslogd
  endscript
}
```

このファイル内の設定オプションは、**/var/log/messages** ログファイル専用の特有なものです。ここで指定された設定は、可能な場合はグローバルオプションを上書きします。そのため、ローテートされた **/var/log/messages** ログファイルは、グローバルオプションで定義された 4 週間ではなく、5 週間保管されます。

以下は、logrotate 設定ファイル内で指定できるディレクティブのリストです。

- **weekly**: ログファイルの週ごとのローテーションを指定します。同様なディレクティブには以下のものがあります。
  - **daily**
  - **monthly**
  - **yearly**
- **compress**: ローテートしたログファイルの圧縮を有効にします。同様なディレクティブには以下のものがあります。
  - **nocompress**
  - **compresscmd**: 圧縮に使用するコマンドを指定します。

- ***uncompresscmd***
- ***compressext:*** *圧縮に使用する拡張子を指定します。*
- ***compressoptions:*** *使用される圧縮プログラムに渡すオプションを指定します。*
- ***delaycompress:*** *ログファイルの圧縮を次回のログファイルのローテーションまで延期します。*

- ***rotate* INTEGER:** *ログファイルが削除される、または特定のアドレスに送信されるまでにログファイルがローテーションされる回数を指定します。値 0 が指定されると、古いログファイルはローテーションではなく削除されます。*
- ***mail* ADDRESS:** *このオプションは、**rotate** ディレクティブで定義された数だけローテーションされたログファイルを特定のアドレスへメール送信できるようにします。同様なディレクティブには以下のものがあります。*
  - ***nomail***
  - ***mailfirst:*** *間もなく期限切れになるログファイルではなく、ローテートされたばかりのログファイルがメール送信されるよう指定します。*
  - ***maillast:*** *交代されたばかりのログファイルではなく、間もなく期限切れになるログファイルがメール送信されるよう指定します。**mail** が有効の場合は、これがデフォルトのオプションです。*

*ディレクティブおよび設定オプションのリストは、man ページ **logrotate(5)** を参照してください。*

23.2.6. オープンファイルの制限の増加

*特定の状況では、**rsyslog** がオープンファイルの最大数の制限を超えています。したがって、**rsyslog** は新しいファイルを開くことができません。*

***rsyslog** でオープンファイルの制限を増やすには、以下を行います。*

***/etc/systemd/system/rsylog.service.d/increase_nofile_limit.conf** ファイルを作成し、以下の内容を追加します。*

```
[Service]
LimitNOFILE=16384
```

23.3. 新規設定フォーマットの使用

*rsyslog バージョン 7 では、rsyslog パッケージの Red Hat Enterprise Linux 7 のデフォルトでインストールされる新しい設定構文が導入されています。この新しい設定形式の目的は、より強力かつ直感的なものにすることと、特定の無効なコンストラクトを許可しないことによりよくあるミスを防ぐことです。この構文の拡張は、RainerScript に依存する新たな設定プロセッサーによって可能になります。以前の形式は引き続き完全にサポートされ、**/etc/rsyslog.conf** 設定ファイルでデフォルトで使用されます。*

*RainerScript は、ネットワークイベントの処理と rsyslog などのイベントプロセッサーの設定用に設計されたスクリプト言語です。RainerScript は最初に、式ベースのフィルターを定義するために使用されました (例23.3「式ベースのフィルター」 を参照)。rsyslog バージョン 7 の RainerScript のバージョンは、**input()** ステートメントおよび**ruleset()** ステートメントを実装し、**/etc/rsyslog.conf** 設定ファイルは新しい構文で記述できます。新しい構文は主に設定される点で異なります。パラメーターは input、action、template、module load などのステートメントへの引数として渡されます。オプションの範囲*

はブロックによって制限されます。オプションのスコープはブロックにより制限されます。これにより、可読性が増し、設定の間違えによって発生するバグの数が減ります。一部の機能は両方の構文で公開され、他の機能は新しい構文でのみ公開されます。

以前のスタイルのパラメーターで記述された設定を比較します。

```
$InputFileName /tmp/inputfile
$InputFileTag tag1:
$InputFileStateFile inputfile-state
$InputRunFileMonitor
```

同じ設定を新たなフォーマットステートメントを使用すると、以下のようになります。

```
input(type="imfile" file="/tmp/inputfile" tag="tag1:" statefile="inputfile-state")
```

これにより、設定で使用されるパラメーター数が大幅に削減され、読みやすさが向上するとともに、実行速度も速まります。RainerScript ステートメントおよびパラメーターに関する詳細な情報は、「オンラインドキュメント」 を参照してください。

23.3.1. Rulesets

特定なディレクティブを除いて、rsyslog は、フィルター条件と、条件が当てはまる場合に実行されるアクションで設定される rules で定義したようにメッセージを処理します。従来の **/etc/rsyslog.conf** ファイルでは、すべてのルールは、どの入力メッセージにおいても表示順に評価されます。このプロセスは最初のルールで開始し、すべてのルールが処理されるか、ルールのいずれかがメッセージを破棄するまで続きます。

ただし、ルールは **ルールセット** と呼ばれるシーケンスにグループ化できます。このルールセットでは、特定のルールの効果を選択された入力のみに制限したり、特定の入力にバインドした明確なアクションセットを定義することで rsyslog のパフォーマンスを強化したりできます。つまり、特定の種類のメッセージでは必然的に false と評価されるフィルター条件をスキップできます。**/etc/rsyslog.conf** 内の以前のルールセット定義は以下のようになります。

```
$RuleSet rulesetname
rule
rule2
```

ルールは、別のルールが定義されたとき、または以下のようにデフォルトのルールセットが呼び出されたときに終了します。

```
$RuleSet RSYSLOG_DefaultRuleset
```

rsyslog 7 の新しい設定形式では、この操作のために **input()** ステートメントおよび **ruleset()** ステートメントが予約されます。**/etc/rsyslog.conf** の新しい形式のルールセット定義は以下のようになります。

```
ruleset(name="rulesetname") {
  rule
  rule2
  call rulesetname2
  &hellip;
}
```

rulesetname を、使用する ruleset の識別子で置き換えます。この名前空間はrsyslog によって使用されるように予約されているため、ルールセット名は **RSYSLOG_** で始めることはできませ

ん。**RSzySLOG_DefaultRuleset** は、メッセージが他のルールセットを割り当てていない場合に実行するデフォルトのルールセットを定義します。rule および rule2 では、上記の **filter-action** 形式でルールを定義できます。**call** パラメーターでは、他の **ruleset** ブロック内から **ruleset** を呼び出すことでこれらをネスト化できます。

ruleset の作成後は、これが適用される入力を指定する必要があります。

```
input(type="input_type" port="port_num" ruleset="rulesetname");
```

ここでは、メッセージを収集する入力モジュールである input_type か、ポート番号である port_num で入力メッセージを特定できます。file や tag といった他のパラメーターは、**input()** 用に指定できます。rulesetname を、メッセージに対して評価されるルールセットの名前に置き換えます。入力メッセージが明示的に **ruleset** にバインドされていない場合は、デフォルトの **ruleset** が適用されます。

レガシーフォーマットを使用して **ruleset** を定義することもできます。詳細は 「オンラインドキュメント」 を参照してください。

例23.11 ルールセットの使用

以下の **ruleset** は、異なるポートからのリモートメッセージが確実に異なる方法で処理されるようにします。**/etc/rsyslog.conf** に以下を追加します。

```
ruleset(name="remote-6514") {
  action(type="omfile" file="/var/log/remote-6514")
}

ruleset(name="remote-601") {
  cron.* action(type="omfile" file="/var/log/remote-601-cron")
  mail.* action(type="omfile" file="/var/log/remote-601-mail")
}

input(type="imtcp" port="6514" ruleset="remote-6514");
input(type="imtcp" port="601" ruleset="remote-601");
```

上記の例に示されるルールセットは、2 つのポートからのログ宛先を定義します。ポート **601** の場合、メッセージはファシリティーに従ってソートされます。次に、TCP 入力が有効になり、ルールセットにバインドされます。この設定が機能するには、必須モジュール (imtcp) の読み込みが必要なことに注意してください。

23.3.2. sysklogd との互換性

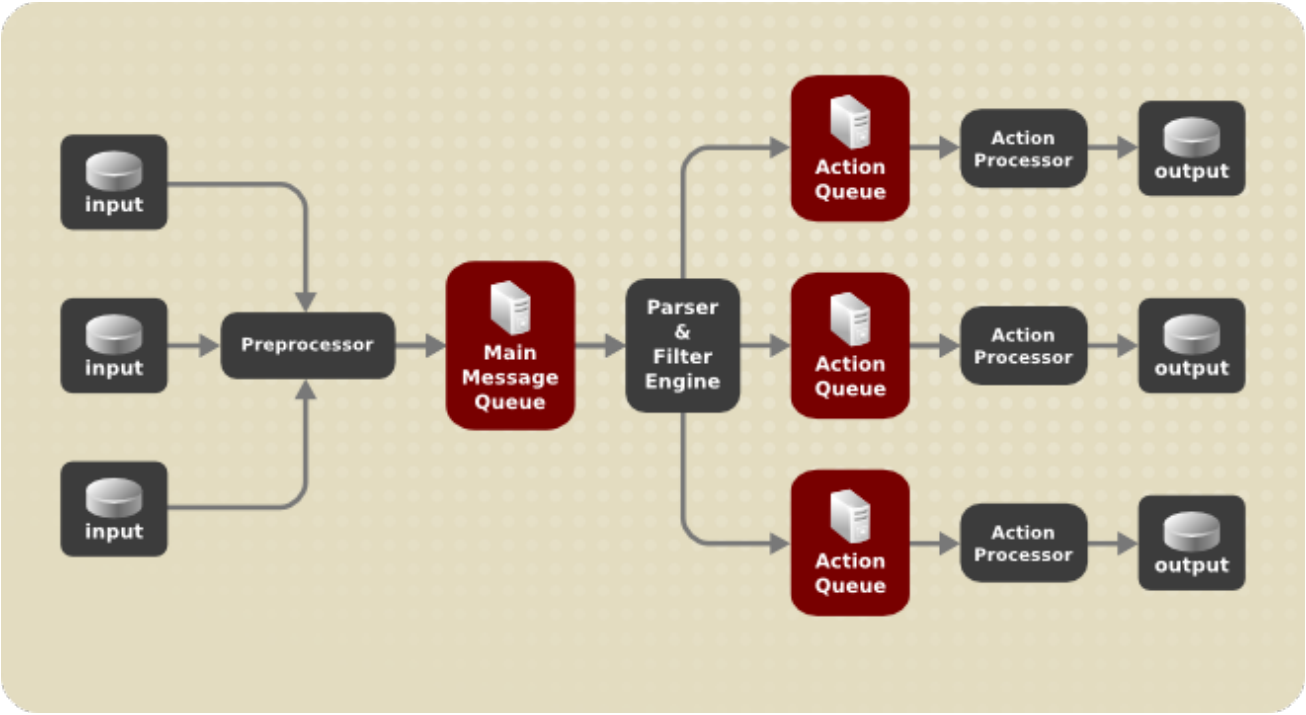
-c オプションで指定される互換性モードは rsyslog バージョン 5 にありますが、バージョン 7 には含まれません。また、sysklogd-style コマンドラインオプションが非推奨となり、これらのコマンドラインオプションを指定した rsyslog の設定は回避する必要があります。ただし、複数のテンプレートおよびディレクティブを使用して、**rsyslogd** が sysklogd のような動作をエミュレートするように設定できます。

rsyslogd オプションの詳細情報は、man ページ **rsyslogd(8)** を参照してください。

23.4. RSYSLOG でのキュー (QUEUE) を使用した操作

キューは、rsyslog のコンポーネント間でコンテンツ (主に syslog メッセージ) を渡すために使用されます。キューを使うと、rsyslog は複数のメッセージを同時に処理することができ、複数のアクションを単一メッセージに一度に適用できます。rsyslog 内のデータフローは、以下のようになります。

図23.1 Rsyslog 内のメッセージフロー



rsyslog がメッセージを受信するたびに、このメッセージをプリプロセッサーに渡して、**メインのメッセージキュー** に配置します。メッセージはそこでキューから取り出され、rule processor に渡されるのを待ちます。

rule processor は、分析およびフィルタリングのエンジンです。ここでは、**/etc/rsyslog.conf** で定義されたルールが適用されます。これらのルールに基づいて、rule processor はどのアクションが実行されるかを評価します。アクションにはそれぞれ、独自のアクションキューがあります。メッセージはこのキューにより各アクションプロセッサーに渡され、これが最終的な出力を作成します。この時点では、1 つのメッセージに関して複数のアクションが同時に実行可能であることに注意してください。このためにメッセージは複製され、複数のアクションプロセッサーに渡されます。

1 アクションにつき、1 つのキューのみが可能です。設定により、メッセージはアクションキューなしですぐにアクションプロセッサーに送信されることもあります。これはダイレクトキュー (direct queues) (下記参照) と呼ばれる動作です。出力アクションが失敗すると、アクションプロセッサーがアクションキューに通知を行い、それにより未処理要素が戻され、しばらくのインターバルの後、アクションが再度試されます。

まとめると、rsyslog では 2 か所のキュー (ルールプロセッサーの前にある単一の**メインメッセージキュー** と、さまざまなタイプの出力アクションの前にある**アクションキュー**) に並びます。キューは、メッセージ処理のパフォーマンス向上につながる以下の 2 つの利点を提供します。

- rsyslog 構造内での decouple のプロデューサーとコンシューマーのバッファーとして機能します。
- メッセージで実行されるアクションの**並列化** を可能にします。

それ以外では、キューを複数のディレクティブで設定して、システムに最適なパフォーマンスを提供することができます。これらの設定オプションは、以下の項で説明されています。

警告

*出力プラグインがメッセージを提供できない場合、メッセージは、先行のメッセージキューに保存されます。キューがいっぱいになると、空きができるまで入力がブロックされます。これにより、ブロックされたキューを使用して新しいメッセーがログに記録されることが回避されます。個別のアクションキューが存在しないため、**SSH** ロギングが阻止され、**SSH** アクセスが阻止されるなどの重大な問題が発生することがあります。したがって、ネットワークを介して転送される、またはデータベースに転送される出力専用アクションキューを使用することが推奨されます。*

23.4.1. キューの定義

メッセージが格納されている場所に基づいて、最も一般的に使用される **ダイレクト**、**インメモリー**、**ディスク**、**ディスクアシストメモリー** *キューの複数のタイプのキューがあります。最もよく使用されるのは、**direct**、**in-memory**、**disk**、および **disk-assisted in-memory** のキューです。**/etc/rsyslog.conf** に以下を追加します。*

```
object(queue.type="queue_type")
```

これを追加することで、設定を適用できます。

- *メインメッセージキュー:* object *を* ***main_queue*** *に置き換える*
- *アクションキュー:* object *を* ***action*** *に置き換える*
- *ルールセット:* object *を* ***ruleset*** *に置き換える*

queue_type *を* ***direct***、***linkedlist*** *または* ***fixedarray*** *(インメモリーキュー) または* ***disk*** *のいずれかと置き換えます。*

メインメッセージキューのデフォルト設定は、FixedArray queue で 10,000 メッセージの制限があります。アクションキューはデフォルトで、Direct queue に設定されます。

ダイレクトキュー (Direct Queue)
出力をローカルファイルに書き出すなど、多くの単純な操作では、アクションの前にキューを構築することは不要です。キューの使用を回避するには、以下を使用します。

```
object(queue.type="Direct")
```

object *を* ***main_queue***、***action***、*または* ***ruleset*** *に置き換え、このオプションをメインメッセージキュー、アクションキュー、またはルールセットにそれぞれ使用します。ダイレクトキューを使用すると、メッセージはプロデューサーからコンシューマーに直ちに直接渡されます。*

ディスクキュー
*ディスクキューはメッセージを厳密にハードドライブに保存します。これにより、信頼性が高くなりますが、すべてのキューモードが最も遅くなります。こうすることで信頼性は非常に高くなりますが、queuing モードのなかでは一番遅くなります。ただし、ほとんどのユースケースでは、ディスクキューは推奨されません。ディスクキューを設定するには、**/etc/rsyslog.conf** に以下を入力します。*

```
object(queue.type="Disk")
```

-

object を **main_queue**、**action**、または **ruleset** に置き換え、このオプションをメインメッセージキュー、アクションキュー、またはルールセットにそれぞれ使用します。キューのデフォルトサイズは、以下の設定ディレクティブで変更できます。

```
object(queue.size="size")
```

ここでは、size はディスクキューの一部の指定サイズを表します。定義されたサイズ制限は限定的なものではなく、rsyslog はキューエントリーがサイズ制限を超えていても、常に 1 つの完全なキューエントリーを書き込みます。ディスクキューの各部分は、個別ファイルに適合します。これらファイルの命名ディレクティブは、以下のとおりです。

```
object(queue.filename="name")
```

これでファイルに name 接頭辞が設定され、1 で始まる 7 桁の数字が続き、ファイルごとにこれが増加します。

```
object(queue.maxfilesize="size")
```

ディスクキューは、デフォルトサイズの 1 MB でそれぞれに書き込まれています。別の値を使用する場合は size を使用します。

メモリー内キュー
インメモリーキューでは、キューに格納されたメッセージをメモリーに保持し、プロセスを高速化します。コンピューターの電源を切ってすぐに入れ直すか、シャットダウンが行われると、キューのデータは失われます。ただし、**action (queue.saveonshutdown="on")** 設定を使用して、シャットダウン前にデータを保存することができます。インメモリーキューには 2 種類あります。

- FixedArray キュー: メインメッセージキューのデフォルトモードで、10,000 要素の制限があります。このタイプのキューは、キュー要素へのポインターを保有する固定かつ事前割り当てのアレイを使用します。これらのポインターのために、キューが空であってもある程度のメモリーが消費されます。しかし、**FixedArray** は、最善のランタイムパフォーマンスを提供し、比較的少ないキューに登録済みのメッセージと高パフォーマンスを期待する場合に最適なものです。

- LinkedList キュー: ここではすべての構造物はリンクされたリストに動的に割り当てられるので、メモリーは必要な場合にのみ割り当てられます。**LinkedList** キューは時折発生するメッセージバーストにもうまく対応します。

一般的に、疑いが残る場合は **LinkedList** キューを使用してみてください。**FixedArray** と比べてメモリー消費量が少なく、プロセスオーバヘッドが低くて済みます。

以下の構文を使用して、インメモリーキューを設定します。

```
object(queue.type="LinkedList")
```

```
object(queue.type="FixedArray")
```

object を **main_queue**、**action**、または **ruleset** に置き換え、このオプションをメインメッセージキュー、アクションキュー、またはルールセットにそれぞれ使用します。

ディスク補助のインメモリーキュー (Disk-Assisted In-memory Queues)
ディスクとインメモリーキューにはそれぞれ利点があり、rsyslog によって**ディスク補助のインメモリーキュー** でそれらを統合できます。これを行うには、通常のインメモリーキューを設定し、**queue.filename="file_name"** ディレクティブをブロックに追加して、ディスクのファイル名を定

*義します。このキューは*disk-assisted *となり、インメモリーキューとディスクキューが連携します。*

インメモリーキューがいっぱい、もしくはシャットダウン後にも継続する必要がある場合に、ディスクキューがアクティブ化されます。ディスク補助のキューでは、ディスク固有およびインメモリー固有の設定パラメーターの両方を設定できます。このタイプのキューはおそらく最も広く使われているもので、潜在的に長期間実行され、信頼性が低いアクションで特に便利です。

ディスク補助インメモリーキューの機能を指定するには、いわゆるウォーターマークを使用します。

> **object*(queue.highwatermark=*”number”*)***

> **object*(queue.lowwatermark=*”number”*)***

object *を* ***main_queue***、***action***、*または* ***ruleset*** *に置き換え、このオプションをメインメッセージキュー、アクションキュー、またはルールセットにそれぞれ使用します。*number *をキューに格納されたメッセージの数に置き換えます。インメモリーキューがハイウォーターマークで定義された数字に到達すると、ディスクへのメッセージの書き込みが開始し、インメモリーキューのサイズがローウォーターマークで定義された数字になるまで続きます。****watermarks*** *を正しく設定すると、ディスクファイルへの書き込みが長くなるため、不要なディスク書き込みが最小限に抑えられます。そのため、高い基準は*queue.size *で設定されたキュー容量全体よりも低い値である必要があります。そのために、ハイウォーターマークは* ***queue.size*** *で設定されているキューのキャパシティ全体より低い必要があります。一方で、ハイウォーターマークを低く設定しすぎると、不必要なディスク補助が頻繁に発生してしまいます。*

> *例23.12 サーバーへのログメッセージの確実な転送*
>
> *多くの場合、****Rsyslog*** *は、ログメッセージがネットワークを介してサーバーに転送される中央ロギングシステムを保持するために使用されます。サーバーが利用できない場合にメッセージが失われないように、転送アクションに対するアクションキューを設定することが推奨されます。これにより、送信に失敗したメッセージは、サーバーが再度到達可能になるまでローカルに保存されます。このようなキューは****UDP*** *プロトコルを使用している接続には設定できないことに注意してください。*
>
> ***単一サーバーへの転送***
>
> *ログメッセージをシステムからホスト名が*example.com *のサーバーに転送し、サーバー停止時にメッセージをバッファーするようアクションキューを設定するタスクを考えてみます。このタスクを実行するには、以下の手順を実行します。*
>
> 1. ***/etc/rsyslog.conf*** *の以下の設定を使用するか、****/etc/rsyslog.d/*** *ディレクトリーに以下の内容のファイルを作成します。*
>
>    ```
>    . action(type=”omfwd”
>    queue.type=”LinkedList”
>    queue.filename=”example_fwd”
>    action.resumeRetryCount="-1"
>    queue.saveonshutdown="on"
>    Target="example.com" Port="6514" Protocol="tcp")
>    ```
>
>   *詳細は以下のようになります。*
>
>   - ***queue.type*** *は LinkedList インメモリーキューを有効にします。*
>   - ***queue.filename*** *はディスクストレージを定義します。この場合、バックアップファイルが*example_fwd *接頭辞を持つ****/var/lib/rsyslog/*** *ディレクトリーで作成されます。*

- *action.resumeRetryCount -1 設定は、サーバーが応答しない場合に接続を再試行するときに rsyslog がメッセージを破棄しないようにします。*
- *rsyslog がシャットダウンすると、有効になっている queue.saveonshutdown はインメモリーデータを保存します。*
- *最後の行は信頼できる TCP デリバリーを使用して受信したすべてのメッセージをロギングサーバーに送ります。ポート指定はオプションです。*
  上記の設定では、rsyslog は、リモートサーバーに到達できない場合にメッセージをメモリーに保持します。ディスク上にあるファイルは、設定されたメモリーキュー領域が rsyslog で不足するか、シャットダウンする必要がある場合にのみ作成されます。これにより、システムパフォーマンスが向上します。

複数のサーバーへの転送

複数のサーバーにログメッセージを転送するプロセスは前の手順に似ています。

1. *各送信先サーバーでは、個別の転送ルール、アクションキュー指定、ディスク上のバックアップファイルが必要です。たとえば、/etc/rsyslog.conf の以下の設定を使用するか、/etc/rsyslog.d/ ディレクトリーに以下の内容のファイルを作成します。*

```
. action(type=”omfwd”
   queue.type=”LinkedList”
   queue.filename=”example_fwd1”
   action.resumeRetryCount="-1"
   queue.saveonshutdown="on"
   Target="example1.com" Protocol="tcp")
. action(type=”omfwd”
   queue.type=”LinkedList”
   queue.filename=”example_fwd2”
   action.resumeRetryCount="-1"
   queue.saveonshutdown="on"
   Target="example2.com" Protocol="tcp")
```

23.4.2. rsyslog ログファイルの新しいディレクトリーの作成

rsyslog は、syslogd デーモンとして実行され、SELinux により管理されます。したがって、rsyslog が書き込む必要があるすべてのファイルでは適切な SELinux ファイルコンテキストが設定されている必要があります。

新規作業用ディレクトリーの作成

1. *作業用ファイルを格納する別のディレクトリーを使用する必要がある場合は、以下のようにディレクトリーを作成します。*

```
~]# mkdir /rsyslog
```

2. *SELinux ポリシーを管理するためにユーティリティーをインストールします。*

```
~]# yum install policycoreutils-python
```

3. *SELinux ディレクトリーコンテキストタイプを /var/lib/rsyslog/ ディレクトリーと同じものに設定します。*

```
~]# semanage fcontext -a -t syslogd_var_lib_t /rsyslog
```

4. SELinux コンテキストを適用します。

```
~]# restorecon -R -v /rsyslog
restorecon reset /rsyslog context unconfined_u:object_r:default_t:s0-
>unconfined_u:object_r:syslogd_var_lib_t:s0
```

5. 必要な場合は、以下のように SELinux コンテキストを確認します。

```
~]# ls -Zd /rsyslog
drwxr-xr-x. root root system_u:object_r:syslogd_var_lib_t:s0  /rsyslog
```

6. 必要に応じてサブディレクトリーを作成します。以下に例を示します。

```
~]# mkdir /rsyslog/work/
```

サブディレクトリーが親ディレクトリーと同じ SELinux コンテキストで作成されます。

7. /etc/rsyslog.conf を有効にする直前にそのファイルに次の行を追加します。

```
global(workDirectory="/rsyslog/work")
```

この設定は、設定ファイルを解析するときに次の **WorkDirectory** ディレクティブが検出されるまで有効になります。

23.4.3. キューの管理

キューはすべてのタイプで、使用中の要件に合致するようにさらなる設定が可能です。複数のディレクティブを使用してアクションキューとメインメッセージキューの両方を修正できます。現時点では、20 以上のキューパラメーターが利用可能です。「オンラインドキュメント」 を参照してください。これらの設定の一部は一般的に使用されます。ワーカースレッド管理などでは、キューの動作をより詳細に制御でき、上級ユーザー用に予約されています。高度な設定では、rsyslog のパフォーマンスやスケジュールキューイングを最適化したり、システムシャットダウン時のキュー動作を修正したりできます。

キューのサイズ制限
キューが保有できるメッセージ数は、以下の設定で制限できます。

```
object(queue.highwatermark="number")
```

object を **main_queue**、**action**、または **ruleset** に置き換え、このオプションをメインメッセージキュー、アクションキュー、またはルールセットにそれぞれ使用します。number をキューに格納されたメッセージの数に置き換えます。キューサイズを実際のメモリーサイズではなくメッセージの数として設定します。デフォルトのキューサイズは、メインメッセージキューとルールセットキューの場合は 10,000 メッセージ、アクションキューの場合は 1,000 となります。

ディスク補助キューはデフォルトでは無制限で、このディレクティブでは制限できませんが、以下の設定で物理的なディスク領域をバイト単位で確保することは可能です。

```
object(queue.maxdiskspace="number")
```

object を、**main_queue**、**action**、または **ruleset** に置き換えます。数によって指定されるサイズ上限に達している場合、キューから取り出されたメッセージによって十分なスペースが解放されるまでメッセージは破棄されます。

メッセージの破棄
キューのメッセージがある数に達すると、重要でないメッセージを破棄して、より優先度が高いエントリーのためにキューのスペースを節約できます。破棄プロセスを開始するしきい値は、discard mark と呼ばれるもので設定できます。

```
object(queue.discardmark="number")
```

object を **MainMsg** または **Action** に置き換えて、このオプションをメインメッセージキューまたはアクションキューにそれぞれ使用します。ここでは、number は、破棄プロセスを開始するためにキューにある必要がある多くのメッセージを表します。破棄するメッセージを定義するには、以下を使用します。

```
object(queue.discardseverity="number")
```

number を、プライオリティーに対応するいずれかの数字に置き換えます (**7** (debug)、**6** (info)、**5** (notice)、**4** (warning)、**3** (err)、**2** (crit)、**1** (alert)、**0** (emerg))。この設定により、定義されたプライオリティーを下回る、新しく受信したメッセージおよびすでにキューに格納されたメッセージは、破棄マークに到達すると直ちにキューから消去されます。

タイムフレームの使用
特定の時間帯にキューを処理するように rsyslog を設定できます。このオプションを使用すると、たとえば処理をオフピーク時に移すことができます。時間帯を定義するには、以下の構文を使用します。

```
object(queue.dequeuetimebegin="hour")
```

```
object(queue.dequeuetimeend="hour")
```

hour では、時間帯を区切る時間を指定します。分は指定せず、24 時間形式を用います。

ワーカースレッドの設定
ワーカースレッド は キューに入っているメッセージに指定されたアクションを実行します。たとえば、メインメッセージキューでは、ワーカーのタスクは、入ってくるメッセージにフィルター論理を適用し、関連のアクションキューに入れることです。メッセージが届くと、ワーカースレッドは自動的に開始します。メッセージ数がある数に達すると、別のワーカースレッドがオンになります。この数字を指定するには、以下を使用します。

```
object(queue.workerthreadminimummessages="number")
```

number を、補助のワーカースレッドを起動するメッセージ数に置き換えます。たとえばnumber を 100 に設定すると、100 を超えるメッセージが届いた際に新たなワーカースレッドが起動します。200 を超えるメッセージが到着すると、3 番目のワーカースレッドが開始されます。ただし、並行して実行している作業スレッド数が多すぎるため、以下を使用してそれらの最大スレッド数を制限できます。

```
object(queue.workerthreads="number")
```

ここでのnumber は、並列で実行可能なワーカースレッドの最大数になります。メインメッセージキューのデフォルトは、1 スレッドです。ワーカースレッドが一旦起動すると、非アクティブタイムアウトが現れるまで、実行し続けます。タイムアウトを設定するには、以下を入力します。

```
object(queue.timeoutworkerthreadshutdown="time")
```

-

time をミリ秒単位の期間設定に置き換えます。新しいメッセージなしの時間を指定すると、ワーカースレッドが閉じます。デフォルトの設定は 1 分です。

バッチのデキュー
複数のメッセージを同時にデキューするように rsyslog を設定して、パフォーマンスを高めることができます。このデキューの最大値を設定するには、以下を使用します。

```
$object(queue.DequeueBatchSize="number")
```

number を同時にデキュー可能な最大数に置き換えます。この数字を高く設定して、許可されるワーカースレッドの結果を大きくすると、メモリー消費量が大きくなることに注意してください。

キューの終了
メッセージを含んでいるキューを終了する際には、ワーカースレッドがキューの処理を完了する間隔を指定することで、データ損失を最小限に抑えることができます。

```
object(queue.timeoutshutdown="time")
```

time をミリ秒で指定します。この期間の後にまだキューに入っているメッセージがある場合、ワーカーは現在のデータ要素を完了してから終了します。このため、未処理のメッセージは失われます。ワーカーが最終要素を完了する間隔も設定できます。

```
object(queue.timeoutactioncompletion="time")
```

このタイムアウトが切れると、残りのワーカーはシャットダウンします。シャットダウン時にデータを保存するには、以下を使用します。

```
object(queue.saveonshutdown="on")
```

これが設定されていると、rsyslog の終了前にすべてのキュー要素がディスクに保存されます。

23.4.4. rsyslog キューの新規構文の使用

rsyslog 7 で利用可能な新規構文では、キューは **/etc/rsyslog.conf** で個別に使用、またはルールセット内部で使用できる **action()** オブジェクト内で定義されます。アクションキューの形式は以下のようになります。

```
action(type="action_type" "queue.size="queue_size" queue.type="queue_type"
queue.filename="file_name"
```

action_type を、アクションを実行するモジュールの名前に置き換え、queue_size を、キューに含めることができるメッセージの最大数に置き換えます。queue_type には、**disk** を選択するか、インメモリーキュー (**direct**、**linkedlist**、または **fixedarray**) のいずれかを選択します。file_name には、パスではなくファイル名のみを指定します。ログファイルを保持する新規ディレクトリーを作成する場合は、SELinux コンテキストを設定する必要があることに注意してください。例は、「rsyslog ログファイルの新しいディレクトリーの作成」 を参照してください。

例23.13 アクションキューの定義

出力アクションで、最大 10,000 件のメッセージを保持できる非同期リンクリストベースのアクションキューを設定するには、以下のようにコマンドを入力します。

```
action(type="omfile" queue.size="10000" queue.type="linkedlist" queue.filename="logfile")
```

直接アクションキューの rsyslog 7 構文は以下のとおりです。

```
. action(type="omfile" file="/var/lib/rsyslog/log_file
  )
```

複数のパラメーターがアクションキュー用 rsyslog 7 構文は以下のように記述できます。

```
. action(type="omfile"
      queue.filename="log_file"
      queue.type="linkedlist"
      queue.size="10000"
  )
```

デフォルトの作業ディレクトリー、または最後に設定した作業ディレクトリーが使用されます。別の作業ディレクトリーを使用する必要がある場合は、アクションキューの前に次の行を追加します。

```
global(workDirectory="/directory")
```

例23.14 新規構文を使用した単一サーバーへの転送

以下の例は、単一サーバーへの転送 の手順に基づき、従来の構文と rsyslog 7 の構文の違いを示しています。**omfwd** プラグインは、**UDP** または **TCP** を介した転送を提供するために使用されます。デフォルト値は **UDP** です。プラグインは組み込まれているため、ロードする必要がありません。

/etc/rsyslog.conf の以下の設定を使用するか、**/etc/rsyslog.d/** ディレクトリーに以下の内容のファイルを作成します。

```
. action(type="omfwd"
  queue.type="linkedlist"
  queue.filename="example_fwd"
  action.resumeRetryCount="-1"
  queue.saveOnShutdown="on"
  target="example.com" port="6514" protocol="tcp"
  )
```

詳細は以下のようになります。

- **queue.type="linkedlist"** は、LinkedList インメモリーキューを有効にします。
- **queue.filename** はディスクストレージを定義します。バックアップファイルは、前のグローバルの **workDirectory** ディレクティブで指定された作業ディレクトリーに example_fwd 接頭辞を付けて作成されます。
- **action.resumeRetryCount -1** 設定は、サーバーが応答しない場合に接続を再試行するときに rsyslog がメッセージを破棄しないようにします。
- rsyslog がシャットダウンすると、有効になっている **queue.saveOnShutdown="on"** はインメモリーデータを保存します。
- 最後の行は受信メッセージをすべてロギングサーバーに転送します。ポートの指定は任意です。

23.5. ロギングサーバーでの RSYSLOG の設定

rsyslog サービスは、ロギングサーバーを実行する機能と、個別のシステムがログファイルをロギングサーバーに送信するように設定する機能の両方を提供します。クライアントの **rsyslog** 設定の詳細は、例23.12「サーバーへのログメッセージの確実な転送」 を参照してください。

rsyslog サービスは、ロギングサーバーとして使用するシステムと、そのシステムにログを送信するように設定する全システムにインストールする必要があります。Rsyslog は Red Hat Enterprise Linux 7 にデフォルトでインストールされます。必要な場合は、確実にインストールするために **root** で以下のコマンドを入力します。

```
~]# yum install rsyslog
```

/etc/services ファイルに設定されているように、syslog トラフィックのデフォルトのプロトコルとポートは **UDP** および **514** です。ただし、**rsyslog** はデフォルトで、ポート **514** で **TCP** を使用するように設定されています。設定ファイル **/etc/rsyslog.conf** において、**TCP** は **@@** で示されます。

例では他のポートが使用されることがありますが、SELinux には、デフォルトで以下のポートでの送受信のみを許可するように設定されています。

```
~]# semanage port -l | grep syslog
syslog_tls_port_t      tcp   6514, 10514
syslog_tls_port_t      udp   6514, 10514
syslogd_port_t         tcp   601, 20514
syslogd_port_t         udp   514, 601, 20514
```

semanage ユーティリティーは、policycoreutils-python パッケージの一部として提供されます。必要な場合は、以下のようにパッケージをインストールします。

```
~]# yum install policycoreutils-python
```

また、デフォルトで **rsyslog** の SELinux タイプである **rsyslogd_t** は、SELinux タイプが **rsh_port_t** のリモートシェル **(rsh)** ポートでの送受信を許可するよう設定されます **(**デフォルトでポート **514** の **TCP** に設定されます**)**。したがって、**semanage** を使用してポート **514** で **TCP** を明示的に許可する必要はありません。たとえば、SELinux がポート **514** でその TCP を許可するよう設定されているかを確認するには、以下のコマンドを入力します。

```
~]# semanage port -l | grep 514
output omitted
rsh_port_t          tcp   514
syslogd_port_t      tcp   6514, 601
syslogd_port_t      udp   514, 6514, 601
```

SELinux の詳細は、Red Hat Enterprise Linux 7 SELinux ユーザーおよび管理者のガイドを参照してください。

ロギングサーバーとして使用するシステムで以下の手順を実行します。これらの手順はすべて **root** ユーザーで実行する必要があります。

ポートで rsyslog トラフィックを許可する SELinux の設定

rsyslog トラフィックに新しいポートを使用する必要がある場合は、ロギングサーバーとクライアントでこの手順を実行します。たとえば、ポート **10514** で **TCP** トラフィックを送受信するには、以下のコマンドシーケンスに進みます。

1. 以下のパラメーターを指定して **semanage port** コマンドを実行します。

```
~]# semanage port -a -t syslogd_port_t -p tcp 10514
```

2. 以下のコマンドを入力して SELinux ポートを確認します。

```
~]# semanage port -l | grep syslog
```

3. 新しいポートがすでに **/etc/rsyslog.conf** に設定されている場合は、**rsyslog** を再起動して変更を反映します。

```
~]# service rsyslog restart
```

4. **rsyslog** が現在リッスンしているポートを確認します。

```
~]# netstat -tnlp | grep rsyslog
tcp    0   0 0.0.0.0:10514      0.0.0.0:*  LISTEN   2528/rsyslogd
tcp    0   0 :::10514           :::*       LISTEN   2528/rsyslogd
```

semanage port コマンドの詳細は、man ページの **semanage-port(8)** を参照してください。

firewalld の設定

firewalld を設定して、受信 **rsyslog** トラフィックを許可します。たとえば、ポート **10514** で **TCP** トラフィックを許可するには、以下の手順を実行します。

```
~]# firewall-cmd --zone=zone --add-port=10514/tcp
success
```

ここで、zone は使用するインターフェイスのゾーンです。ここで変更する内容は、システムの再起動後は維持されないことに注意してください。ファイアウォールを永続的に変更するには、コマンドに **--permanent** オプションを繰り返し追加してください。**firewalld** でのポートのオープンおよびクローズの詳細については、Red Hat Enterprise Linux 7 Security Guideを参照してください。

1. 上記の設定を確認するには、以下のコマンドを使用します。

```
~]# firewall-cmd --list-all
public (default, active)
 interfaces: eth0
 sources:
 services: dhcpv6-client ssh
 ports: 10514/tcp
 masquerade: no
 forward-ports:
 icmp-blocks:
 rich rules:
```

rsyslog で、リモートログメッセージを受信してソートするように設定

1. テキストエディターで **/etc/rsyslog.conf** ファイルを開き、以下の手順を行います。

   a. モジュールセクションと **Provides UDP syslog reception** セクションの間に以下の行を追加します。

```
# Define templates before the rules that use them

# Per-Host Templates for Remote Systems #
$template TmplAuthpriv,
"/var/log/remote/auth/%HOSTNAME%/%PROGRAMNAME:::secpath-replace%.log"
$template TmplMsg,
"/var/log/remote/msg/%HOSTNAME%/%PROGRAMNAME:::secpath-replace%.log"
```

b. デフォルトの **Provides TCP syslog reception** セクションを、以下の内容に置き換えます。

```
# Provides TCP syslog reception
$ModLoad imtcp
# Adding this ruleset to process remote messages
$RuleSet remote1
authpriv.*  ?TmplAuthpriv
*.info;mail.none;authpriv.none;cron.none  ?TmplMsg
$RuleSet RSYSLOG_DefaultRuleset  #End the rule set by switching back to the
default rule set
$InputTCPServerBindRuleset remote1 #Define a new input and bind it to the
"remote1" rule set
$InputTCPServerRun 10514
```

/etc/rsyslog.conf ファイルへの変更を保存します。

2. **rsyslog** サービスは、ログサーバーと、そのサーバーにログ記録を試みるシステムの両方で実行する必要があります。

a. **systemctl** コマンドで **rsyslog** サービスを起動します。

```
~]# systemctl start rsyslog
```

b. **rsyslog** サービスを今後自動的に起動するために、**root** で以下のコマンドを入力します。

```
~]# systemctl enable rsyslog
```

環境内の他のシステムからログファイルを受け取り、保存するように、ログサーバーが設定されています。

23.5.1. ロギングサーバーでの新規テンプレート構文の使用

rsyslog 7 にはテンプレートスタイルが多数あります。文字列テンプレートは従来の形式に最もよく似ています。文字列形式を使用して上記の例からテンプレートを生成する場合は、以下のようになります。

```
template(name="TmplAuthpriv" type="string"
    string="/var/log/remote/auth/%HOSTNAME%/%PROGRAMNAME:::secpath-replace%.log"
    )

template(name="TmplMsg" type="string"
    string="/var/log/remote/msg/%HOSTNAME%/%PROGRAMNAME:::secpath-replace%.log"
    )
```

また、これらのテンプレートは、以下のようにリスト形式で記述することもできます。

```
template(name="TmplAuthpriv" type="list") {
  constant(value="/var/log/remote/auth/")
  property(name="hostname")
  constant(value="/")
  property(name="programname" SecurePath="replace")
  constant(value=".log")
  }

template(name="TmplMsg" type="list") {
  constant(value="/var/log/remote/msg/")
  property(name="hostname")
  constant(value="/")
  property(name="programname" SecurePath="replace")
  constant(value=".log")
  }
```

このテンプレートテキスト形式は、rsyslog の初心者にとって理解しやすいかもしれません。したがって、要件が変更したら簡単に変更できます。

新規構文への変更を完了するには、モジュールロードコマンドを再作成し、ルールセットを追加して、プロトコル、ポート、およびルールセットにルールセットをバインドする必要があります。

```
module(load="imtcp")

ruleset(name="remote1"){
   authpriv.*  action(type="omfile" DynaFile="TmplAuthpriv")
   *.info;mail.none;authpriv.none;cron.none action(type="omfile" DynaFile="TmplMsg")
}

input(type="imtcp" port="10514" ruleset="remote1")
```

23.6. RSYSLOG モジュールの使用

モジューラーの設計上、rsyslog は、追加の機能が含まれる各種 **モジュール** *を提供します。モジュールはサードパーティーによる書き込みが可能であることに注意してください。ほとんどのモジュールは、追加入力 (以下の* Input Modules *参照) または出力 (以下の* Output Modules *参照) を提供します。その他のモジュールは、各モジュールに固有の機能を提供します。モジュールは、モジュールの読み込み後に利用可能になる追加の設定ディレクティブを提供する場合があります。モジュールを読み込むには、以下の構文を使用します。*

```
module(load="MODULE")
```

MODULE *は希望のモジュールを表します。たとえば、rsyslog を有効にして標準テキストファイルを syslog メッセージに変換する Text File Input Module (***imfile***) を読み込む場合は、* **/etc/rsyslog.conf** *設定ファイルの以下の行を指定します。*

```
module(load="imfile")
```

rsyslog は多くのモジュールを提供し、これらは以下の主なカテゴリーに分けられます。

- *入力モジュール: 入力モジュールは、さまざまなソースからメッセージを収集します。入力モジュールの名前は、常に* **imfile** *および* **imjournal** *のように接頭辞* **im** *で始まります。*
- *出力モジュール: 出力モジュールはメッセージをネットワーク上に送信したり、データベース内*

に保存したり、暗号化するなど、様々なターゲットにメッセージを発行するための機能を提供します。出力モジュールの名前は常に **omsnmp** や **omrelp** などのように接頭辞 **om** で始まります。

- パーサーモジュール: これらのモジュールは、カスタムの解析ルールの作成や不正な形式のメッセージ解析に使用されます。C プログラミング言語についてある程度の知識があれば、独自のメッセージパーサーが作成できます。パーサーモジュールの名前は常に **pmrfc5424** や **pmrfc3164** のように接頭辞 **pm** で始まります。

- メッセージ修正モジュール: メッセージ修正モジュールは、syslog メッセージの内容を変更します。このモジュールの名前は、**mm** 接頭辞で始まります。**mmanon**、**mmnormalize**、**mmjsonparse** といったメッセージ修正モジュールは、メッセージの匿名化や正常化に使用されます。

- 文字列生成モジュール - 文字列生成モジュールは、メッセージ内容を基にして文字列を生成し、rsyslog で用意されているテンプレート機能と密接に機能します。テンプレートに関する詳細は 「テンプレート」 を参照してください。文字列生成モジュールの名前は、**smfile** または **smtradfile** のように常に接頭辞 **sm** で始まります。

- ライブラリーモジュール - ライブラリーモジュールは、他の読み込み可能なモジュール用の機能を提供します。これらのモジュールは、必要であるのにユーザーが設定できない場合に rsyslog が自動的に読み込みます。

利用可能なモジュールのリストと、そのモジュールの詳しい説明は、http://www.rsyslog.com/doc/rsyslog_conf_modules.html を参照してください。

警告

rsyslog はモジュールを読み込む際に、モジュールに対して一部の機能とデータへのアクセスを提供します。これはセキュリティー上の脅威となる可能性があります。セキュリティーリスクを最小限にするために、信頼できるモジュールのみを使用するようにしてください。

23.6.1. テキストファイルのインポート

テキストファイル入力モジュールは **imfile** と省略され、rsyslog がテキストファイルを syslog メッセージのストリームに変換できるようにします。**imfile** を使用して、独自のテキストファイルログを作成するアプリケーションからログメッセージをインポートできます。**imfile** を読み込むには、**/etc/rsyslog.conf** に以下を追加します。

```
module(load="imfile"
  PollingInterval="int")
```

複数のファイルをインポートする場合でも、**imfile** を一度読み込むだけで十分です。PollingInterval モジュール引数は、接続済みテキストファイルの変更に対して rsyslog チェックをどのくらいの頻度で行うか指定します。デフォルトの間隔は 10 秒で、変更するには int を秒で指定した時間間隔に置き換えます。

インポートするテキストファイルを特定するには、**/etc/rsyslog.conf** で以下の構文を使用します。

```
# File 1
```

```
input(type="imfile"
  File="path_to_file"
  Tag="tag:"
  Severity="severity"
  Facility="facility")

# File 2
input(type="imfile"
  File="path_to_file2")
...
```

入力テキストファイルを指定するために必要な設定:

- path_to_file をテキストファイルへのパスに置き換えます。
- tag: をこのメッセージのタグ名に置き換えます。

必要なディレクティブ以外に、テキスト入力に適用可能な設定がいくつかあります。severity を適切なキーワードで置き換えて、インポートされたメッセージの重要度を設定します。facility を、メッセージを作成したサブシステムを定義するキーワードに置き換えます。重要度と機能のキーワードは、機能または優先度ベースのフィルターで使用されたものと同じものです(「フィルター」 を参照)。

例23.15 テキストファイルのインポート

Apache HTTP サーバーはログファイルをテキスト形式で作成します。rsyslog の処理機能を apache エラーメッセージに適用するには、まず **imfile** モジュールを使用してメッセージをインポートします。**/etc/rsyslog.conf** に以下を追加します。

```
module(load="imfile")
input(type="imfile"
  File="/var/log/httpd/error_log"
  Tag="apache-error:")
```

23.6.2. データベースへのメッセージのエクスポート

ログデータの処理は、テキストファイルではなくデータベース内で実行するとより速く、より便利なものになります。使用される DBMS のタイプに基づいて、**ommysql**、**ompgsql**、**omoracle**、**ommongodb** などの出力モジュールを選択します。別の方法としては、**libdbi** ライブラリーに依存する一般的な**omlibdbi** 出力モジュールを使用します。**omlibdbi** モジュールは、Firebird/Interbase、MS SQL、Sybase、SQLite、Ingres、Oracle、mSQL、MySQL、および PostgreSQL のデータベースシステムをサポートします。

例23.16 データベースへの rsyslog メッセージのエクスポート

rsyslog メッセージを MySQL データベースに保存するには、**/etc/rsyslog.conf** に以下を追加します。

```
module(load="ommysql")

. action(type"ommysql"
server="database-server"
db="database-name"
```

```
uid="database-userid"
pwd="database-password"
serverport="1234")
```

最初に出力モジュールが読み込まれ、その後に通信ポートが指定されます。上記の例では、サーバー名、データベース名、認証データなどの追加情報は、最後の行で指定されています。

23.6.3. 暗号化トランスポートの有効化

ネットワーク転送の機密性と統合性は、TLS または GSSAPI 暗号化プロトコルのいずれかによって提供されます。

Transport Layer Security (TLS) は、ネットワーク上の通信セキュリティーを提供するために設計された暗号プロトコルです。TLS を使用すると、rsyslog メッセージは送信前に暗号化され、送信者と受信者間で相互認証が行われます。TLS の設定は 「TLS を使用した暗号化メッセージ転送の設定」 を参照してください。

Generic Security Service API (GSSAPI) は、プログラムがセキュリティーサービスにアクセスするためのアプリケーションプログラミングインターフェイスです。rsyslog と接続して使用するには、機能している Kerberos 環境が必要です。GSSAPI の設定は、「GSSAPI を使用した暗号化メッセージ転送の設定」 を参照してください。

TLS を使用した暗号化メッセージ転送の設定
TLS を経由して暗号化トランスポートを使用するには、サーバーとクライアント両方を設定する必要があります。

1. 公開鍵、秘密鍵、証明書ファイルを作成し、「新しい鍵と証明書の生成」 を参照します。
2. **サーバー** 側で、**/etc/rsyslog.conf** 設定ファイルで以下を設定します。
   a. **gtls netstream** ドライバーをデフォルトドライバーに設定します。

   ```
   global(defaultnetstreamdriver="gtls")
   ```

   b. 証明書ファイルへのパスを指定します。

   ```
   global(defaultnetstreamdrivercafile="path_ca.pem"
   defaultnetstreamdrivercertfile="path_cert.pem"
   defaultnetstreamdriverkeyfile="path_key.pem")
   ```

   簡潔な設定ファイルを好む場合は、グローバルディレクティブをすべて 1 つのブロックに統合できます。

   以下を置き換えます。

   - path_ca.pem を、公開鍵へのパスに置き換え
   - path_cert.pem を、証明書ファイルへのパスに置き換え
   - path_key.pem を、秘密鍵へのパスに置き換え

   c. **imtcp** モジュールを読み込み、ドライバーオプションを設定します。

```
module(load=”imtcp”
StreamDriver.Mode=“number”
StreamDriver.AuthMode=”anon”)
```

d. *サーバーを起動します。*

```
input(type="imtcp" port="port")
```

以下を置き換えます。

- number *はドライバーモードを指定します。TCP オンリーモードを有効にするには、**1** を使用します。*
- port *を、リスナーを起動するポート番号に置き換えます。たとえば **10514** にします。**anon** 設定は、クライアントが認証されていないことを意味します。*

3. ***クライアント** 側の **/etc/rsyslog.conf** 設定ファイルで以下を設定します。*

a. *公開鍵を読み込みます。*

```
global(defaultnetstreamdrivercafile="path_ca.pem")
```

path_ca.pem *を公開鍵へのパスで置き換えます。*

b. ***gtls netstream** ドライバーをデフォルトドライバーに設定します。*

```
global(defaultnetstreamdriver="gtls")
```

c. *ドライバーを設定し、実行するアクションを指定します。*

```
module(load=”imtcp”
  streamdrivermode=”number”
  streamdriverauthmode=”anon”)
input(type=”imtcp”
  address=”server.net”
  port=”port”)
```

number、anon、port *を、サーバーと同じ値に置き換えます。*

*上記リストの最後の行で、例のアクションがサーバーから指定の **TCP** ポートにメッセージを転送します。*

***GSSAPI** を使用した暗号化メッセージ転送の設定*
***rsyslog** では、**GSSAPI** とのやり取りは*imgssapi *モジュールによって提供されます。**GSSAPI** 転送モードをオンにするには、以下を実行します。*

1. ***/etc/rsyslog.conf** に以下の設定をします。*

```
$ModLoad imgssapi
```

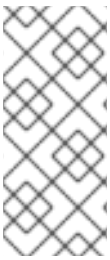
*このディレクティブは、**imgssapi** モジュールを読み込みます。*

2. *以下のように入力を指定します。*

```
$InputGSSServerServiceName name
```

```
$InputGSSServerPermitPlainTCP on
$InputGSSServerMaxSessions number
$InputGSSServerRun port
```

- name を、GSS サーバーの名前に置き換えます。
- number を置き換え、サポートされる最大セッション数を設定します。デフォルトで、この数字は制限されていません。
- port を、GSS サーバーを起動したい選択ポートに置き換えます。**$InputGSSServerPermitPlainTCP on** 設定により、サーバーは同じポートでプレーン TCP メッセージを受信できます。デフォルトでオフになっています。



注記

設定ファイルリーダーが **/etc/rsyslog.conf** 設定ファイルで **$InputGSSServerRun** ディレクティブに遭遇すると、**imgssapi** モジュールはすぐに初期化されます。このため、**$InputGSSServerRun** より後に設定されている補助オプションは無視されます。設定を有効にするには、すべての **imgssapi** 設定オプションを、**$InputGSSServerRun** の前に配置する必要があります。

例23.17 GSSAPI の使用

以下の設定では、ポート 1514 上の GSS サーバーが有効になります。この設定では、同一ポート上でプレーンの tcp syslog メッセージの受信も許可されます。

```
$ModLoad imgssapi
$InputGSSServerPermitPlainTCP on
$InputGSSServerRun 1514
```

23.6.4. RELP の使用

Reliable Event Logging Protocol *(RELP)* は、コンピューターネットワークにおけるデータロギング用のネットワーキングプロトコルです。信頼性のあるイベントメッセージの配信を提供するように設計されています。これは、メッセージ損失が許されない環境で便利なものです。

RELP の設定
RELP を設定するには、**/etc/rsyslog.conf** ファイルを使用してサーバーとクライアントの両方を設定します。

1. クライアントを設定するには、以下を実行します。
   a. 必須モジュールを読み込みます。

   ```
   module(load="imuxsock")
   module(load="omrelp")
   module(load="imtcp")
   ```

   b. 以下のように TCP 入力を設定します。

   ```
   input(type="imtcp" port="port")
   ```

   port を、必要なポートに置き換えてリスナーを開始します。

c. トランスポート設定を設定します。

```
action(type="omrelp" target="target_IP" port="target_port")
```

target_IP とtarget_port をターゲットサーバーを識別する IP アドレスとポートに置き換えます。

2. サーバーを設定するには、以下を実行します。

a. モジュールの読み込みを設定します。

```
module(load="imuxsock")
module(load="imrelp" ruleset="relp")
```

b. クライアント設定と同様の TCP 入力を設定します。

```
input(type="imrelp" port="target_port")
```

target_port をクライアントと同じ値に置き換えます。

c. ルールを設定し、実行するアクションを選択します。次の例では、log_path はメッセージを保存するためのパスを指定しています。

```
ruleset (name="relp") {
action(type="omfile" file="log_path")
}
```

TLS を使用した RELP の設定
TLS を使用して RELP を設定するには、認証を設定する必要があります。続いて、**/etc/rsyslog.conf** ファイルを使用してサーバーとクライアントの両方を設定する必要があります。

1. 公開鍵、秘密鍵、証明書ファイルを作成します。手順は、「新しい鍵と証明書の生成」 を参照してください。

2. クライアントを設定するには、以下を実行します。

a. 必須モジュールを読み込みます。

```
module(load="imuxsock")
module(load="omrelp")
module(load="imtcp")
```

b. 以下のように TCP 入力を設定します。

```
input(type="imtcp" port="port")
```

port を、必要なポートに置き換えてリスナーを開始します。

c. トランスポート設定を設定します。

```
action(type="omrelp" target="target_IP" port="target_port" tls="on"
tls.caCert="path_ca.pem"
tls.myCert="path_cert.pem"
tls.myPrivKey="path_key.pem"
```

```
tls.authmode="mode"
tls.permittedpeer=["peer_name"]
)
```

以下を置き換えます。

- target_IP と target_port を、ターゲットサーバーを識別する IP アドレスとポートにそれぞれ置き換えます。
- path_ca.pem、path_cert.pem、および path_key.pem を、証明書ファイルのパスに置き換えます。
- mode を、トランザクションの認証モードに置き換えます。"name" または "fingerprint" のいずれかを使用します。
- peer_name を、許可されたピアの証明書フィンガープリントに置き換えます。これを指定した場合、**tls.permittedpeer** は、選択したピアグループへの接続を制限します。tls="on" の設定は、TLS プロトコルを有効にします。

3. サーバーを設定するには、以下を実行します。

a. モジュールの読み込みを設定します。

```
module(load="imuxsock")
module(load="imrelp" ruleset="relp")
```

b. クライアント設定と同様の TCP 入力を設定します。

```
input(type="imrelp" port="target_port" tls="on"
tls.caCert="path_ca.pem"
tls.myCert="path_cert.pem"
tls.myPrivKey="path_key.pem"
tls.authmode="name"
tls.permittedpeer=["peer_name","peer_name1","peer_name2"]
)
```

強調した値をクライアントと同じものに置き換えます。

c. ルールを設定し、実行するアクションを選択します。次の例では、log_path はメッセージを保存するためのパスを指定しています。

```
ruleset (name="relp") {
action(type="omfile" file="log_path")
}
```

23.7. RSYSLOG と JOURNAL の相互作用

ここまで説明してきたように、使用中のシステムに存在する Rsyslog と Journal の 2 つのロギングアプリケーションには、特別のユースケースに適したいくつかの特徴的な機能があります。多くの状況では、これらの機能を組み合わせると便利になります。たとえば、構造化メッセージを作成して、ファイルデータベースに格納する場合が考えられます (「Rsyslog での構造化ロギング」 を参照)。ここで必要となる通信インターフェイスは、Rsyslog 側の出入力モジュールと Journal の通信ソケットが提供します。

デフォルトで **rsyslogd** は、ジャーナルファイルのデフォルト入力モードとして **imjournal** モジュール

を使用します。このモジュールを使用すると、メッセージだけでなく、**journald** が提供する構造化データもインポートできます。また、古いデータは **journald** からインポートできます (**IgnorePreviousMessages** オプションで禁止されない限り)。**imjournal** の基本設定は「Journal からのデータのインポート」 を参照してください。

別の方法では、syslog ベースのアプリケーション用の出力に、**journal** が提供するソケットから読み取るように **rsyslogd** を設定することもできます。ソケットへのパスは、**/run/systemd/journal/syslog** です。プレーンな rsyslog メッセージを維持したい場合は、このオプションを使用します。**imjournal** と比較すると、ソケット入力は現在、ruleset バインディングやフィルタリングなど、より多くの機能を提供しています。ソケットを使用して Journal データをインポートするには、**/etc/rsyslog.conf** で以下の設定を使用します。

```
module(load="imuxsock"
   SysSock.Use="on"
   SysSock.Name="/run/systemd/journal/syslog")
```

また、**omjournal** モジュールで Rsyslog から Journal にメッセージを出力することもできます。**/etc/rsyslog.conf** で出力を以下のように設定します。

```
module(load="omjournal")
action(type="omjournal")
```

たとえば、以下の設定では、tcp ポート 10514 で受信したメッセージをすべて Journal に転送します。

```
module(load="imtcp")
module(load="omjournal")

ruleset(name="remote") {
 action(type="omjournal")
}

input(type="imtcp" port="10514" ruleset="remote")
```

23.8. RSYSLOG での構造化ロギング

大量のログデータを生成するシステムでは、ログメッセージを**構造化されたフォーマット** で維持すると便利です。構造化メッセージは、特定情報の検索や統計情報の作成、メッセージ構造の変更およびその不整合への対応が容易になります。Rsyslog は JSON (JavaScript Object Notation) フォーマットを使用してログメッセージに構造を提供します。

以下の非構造化ログメッセージを

```
Oct 25 10:20:37 localhost anacron[1395]: Jobs will be executed sequentially
```

次の構造化メッセージと比較してください。

```
{"timestamp":"2013-10-25T10:20:37", "host":"localhost", "program":"anacron", "pid":"1395",
"msg":"Jobs will be executed sequentially"}
```

鍵と値のペアを使用した構造化データの検索は、正規表現によるテキストファイルの検索よりも速く、より正確です。構造化データでは、異なるアプリケーションで作成されたメッセージで同一エントリーを検索することもできます。また、JSON ファイルは、追加のパフォーマンスおよび分析機能を提供す

る MongoDB のようなドキュメントデータベースで保存することもできます。一方で、構造化メッセージは、非構造化メッセージよりも多くのディスクスペースを必要とします。

rsyslog では、メタデータを伴うログメッセージは、**imjournal** モジュールを使用して Journal からプルされます。**mmjsonparse** モジュールでは、Journal やその他のソースからインポートしたデータを解析し、たとえばデータベース出力としてさらに処理できます。解析が成功するには、**mmjsonparse** では、Lumberjack プロジェクトで定義された方法で入力メッセージが構築されている必要があります。

Lumberjack プロジェクトは、後方互換性がある方法で構造化ロギングを rsyslog に追加することを目指しています。構造化メッセージを特定するために、Lumberjack は実際の JSON 構造の前に付く @cee: 文字列を指定します。また、Lumberjack は、JSON 文字列内のエントリーに使用する標準フィールド名のリストも定義します。Lumberjack の詳細情報は「オンラインドキュメント」 を参照してください。

以下は、lumberjack 形式のメッセージの例です。

```
@cee: {"pid":17055, "uid":1000, "gid":1000, "appname":"logger", "msg":"Message text."}
```

この構造を Rsyslog 内に構築するには、テンプレートを使用します。「構造化メッセージのフィルタリング」 を参照してください。アプリケーションおよびサーバーは、**libumberlog** ライブラリーを使用して、lumberjack 準拠の形式でメッセージを生成することができます。**libumberlog** の詳細情報は、「オンラインドキュメント」 を参照してください。

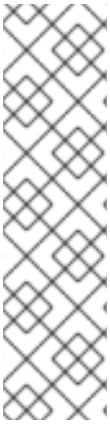
23.8.1. Journal からのデータのインポート

imjournal モジュールは Rsyslog の入力モジュールで、ネイティブに journal ファイルを読み取ります (「Rsyslog と Journal の相互作用」 を参照)。その後、Journal メッセージは、他の rsyslog メッセージのようにテキスト形式でログ記録されます。しかし、さらに処理することで、Journal が提供するメタデータを構造化メッセージに変換できます。

Journal から Rsyslog にデータをインポートするには、**/etc/rsyslog.conf** で以下の設定を使用します。

```
module(load="imjournal"
  PersistStateInterval="number_of_messages"
  StateFile="path"
  ratelimit.interval="seconds"
  ratelimit.burst="burst_number"
  IgnorePreviousMessages="off/on")
```

- number_of_messages では、journal データの保存頻度を指定できます。メッセージの数が指定された数値に達するとデータが保存されます。
- path は、state ファイルのパスに置き換えます。このファイルは、最後に処理された journal エントリーを追跡します。
- seconds では、レート制限の間隔を設定します。この間隔内に処理されるメッセージ数は、burst_number で指定した値を超えることはできません。デフォルト設定は、600 秒あたり 20,000 メッセージです。Rsyslog は、この指定された時間枠内で最大バースト後に届いたメッセージを破棄します。
- **IgnorePreviousMessages** を使用すると、現在 Journal にあるメッセージを無視し、新しいメッセージのみをインポートできます。このメッセージは、状態ファイルが指定されていない場合に使用されます。デフォルト設定は **off** です。この設定がオフで state ファイルが存在しない場合、前回の rsyslog セッションで処理されたものであっても、ジャーナルのすべてのメッセージが処理されます。



注記

imjournal を従来のシステムログ入力である imuxsock モジュールと同時に使用できます。ただし、メッセージの重複を避けるために、imuxsock がジャーナルのシステムソケットを読まないようにする必要があります。これを行うには、SysSock.Use ディレクティブを使用します。

```
module(load”imjournal”)
module(load”imuxsock”
  SysSock.Use=”off”
  Socket="/run/systemd/journal/syslog")
```

Journal が保存したデータおよびメタデータはすべて、構造化メッセージに変換することができます。これらのメタデータエントリーの一部は、例23.19「詳細な journalctl 出力」にリスト表示されています。ジャーナルフィールドの完全なリストは、systemd.journal-fields(7) の man ページを参照してください。たとえば、kernel を元とするメッセージが使用する kernel journal fields *にフォーカスできます。これは、カーネルのメッセージで使用されています。*

23.8.2. 構造化メッセージのフィルタリング

rsyslog の解析モジュールで必要となる lumberjack 形式のメッセージを作成するには、以下のテンプレートを使用します。

```
template(name="CEETemplate" type="string" string="%TIMESTAMP% %HOSTNAME%
%syslogtag% @cee: %$!all-json%\n")
```

このテンプレートは、JSON 文字列の前に @cee: 文字列を付加し、たとえば、omfile モジュールで出力ファイルを作成する際に適用できます。JSON フィールド名にアクセスするには、$! 接頭辞を使用します。たとえば、以下のフィルター条件では、特定の hostname *と* UID *のメッセージが検索されます。*

```
($!hostname == "hostname" && $!UID== "UID")
```

23.8.3. JSON の解析

構造化メッセージの解析には、mmjsonparse モジュールが使用されます。

*これらのメッセージは Journal から来る場合もあれば、他の入力ソースから来る場合もあり、*Lumberjack *プロジェクトで定義された方法でフォーマットされている必要があります。これらのメッセージは* @cee: *文字列の存在により識別されます。そして、JSON 構造が有効かどうかを mmjsonparse がチェックした後、メッセージが解析されます。*

lumberjack 形式の JSON メッセージを mmjsonparse で解析するには、/etc/rsyslog.conf で以下の設定を使用します。

```
module(load”mmjsonparse”)

. :mmjsonparse:
```

この例では、mmjsonparse モジュールが最初の行で読み込まれ、その後にすべてのメッセージがそこに転送されます。現在は、mmjsonparse で使用可能な設定パラメーターはありません。

23.8.4. MongoDB でのメッセージの保存

Rsyslog は、ommongodb 出力モジュールで、MongoDB ドキュメントデータベースでの JSON ログの保存をサポートします。

MongoDB にログメッセージを転送するには、/etc/rsyslog.conf で以下の構文を使用します (ommongodb 用の設定パラメーターは、新たな設定フォーマットでのみ利用可能です。「新規設定フォーマットの使用」 を参照してください)。

```
module(load”ommongodb”)

. action(type="ommongodb" server="DB_server" serverport="port" db="DB_name"
collection="collection_name" uid="UID" pwd="password")
```

- DB_server は MongoDB サーバーの名前またはアドレスに置き換えます。port を指定して、MongoDB サーバーから非標準ポートを選択します。port のデフォルト値は **0** で、通常はこのパラメーターを変更する必要はありません。
- DB_name では、出力先となる MongoDB サーバー上のデータベースを特定します。collection_name を、このデータベース内のコレクション名に置き換えます。MongoDB では、コレクションはドキュメントのグループで、RDBMS テーブルと同等のものです。
- UID と password を、ログイン情報に置き換えて設定します。

テンプレートを使用すると、最終的なデータベースの出力形式を作成できます。デフォルトでは、rsyslog は標準 lumberjack フィールド名をベースにしたテンプレートを使用します。

23.9. RSYSLOG のデバッグ

***rsyslogd** をデバッグモードで実行するには、以下のコマンドを使用します。*

```
rsyslogd -dn
```

*このコマンドでは、**rsyslogd** がデバッグ情報を作成し、標準出力に印刷します。**-n** は no fork を意味します。たとえば、デバッグ出力をログファイルに保存して、環境変数を使用してデバッグを変更できます。**rsyslogd** を起動する前に、コマンドラインで次のコマンドを実行します。*

```
export RSYSLOG_DEBUGLOG="path"
export RSYSLOG_DEBUG="Debug"
```

path を、デバッグ情報をログ記録するファイルの場所に置き換えます。RSYSLOG_DEBUG 変数に利用可能なオプションのリストは、man ページ **rsyslogd(8)** の関連セクションを参照してください。

/etc/rsyslog.conf ファイルで使用している構文が有効かどうかをチェックするには、以下を使用します。

```
rsyslogd -N 1
```

***1** は、出力メッセージの長さのレベルを表します。現在提供されているのは 1 つのレベルのみなので、これは前方互換性オプションになります。ただし、検証を実行するには、この引数を追加する必要があります。*

23.10. JOURNAL の使用

*Journal は systemd のコンポーネントで、ログファイルの表示および管理を担います。**rsyslogd** などの従来の syslog デーモンとの並列もしくはその代わりとして使用できます。Journal は、従来のロギン*

グに関連する問題を処理するために開発されました。システムの他の部分と緊密に統合されており、さまざまなロギング技術およびログファイルのアクセス管理をサポートします。

ロギングデータは、Journal の **journald** サービスが収集、保存、処理を行います。カーネルやユーザープロセス、標準出力、システムサービスの標準エラー出力、またはネイティブ **API** 経由から受信したロギング情報に基づいて、journals と呼ばれるバイナリーファイルを作成、維持します。これらの journals は構造化およびインデックス化され、これによりシーク時間が比較的速くなります。Journal エントリーは、一意の識別子を持つことが可能です。**journald** サービスは、各ログメッセージについて多数のメタデータを収集します。実際の journal ファイルはセキュリティー保護され、手動での編集はできません。

23.10.1. ログファイルの表示

journal ログにアクセスするには、journalctl ツールを使用します。ログタイプの基本的なビューを見るには、**root** で以下を入力します。

```
journalctl
```

このコマンドの出力は、システム上で生成されたすべてのログファイルリストで、これにはシステムコンポーネントやユーザーが生成したメッセージも含まれます。この出力の構造は、**/var/log/messages/** で使用されているものに似ていますが、以下の改善点があります。

- エントリーの優先度が視覚的にマークされています。エラー優先度およびそれ以上の行は赤色のハイライト表示がされており、注意および警告の優先度の行には太字フォントが使われています。
- タイムスタンプが使用中のシステムのローカルタイムゾーンに変換されます。
- ローテーションされたログを含めて、すべてのログ記録済みデータが表示されます。
- ブートの最初が特別行にタグ付けされます。

例23.18 journalctl の出力例

以下は、journalctl ツールが提供する出力例です。パラメーターなしで呼び出されると、エントリーリストがタイムスタンプで始まり、その後にホスト名、操作を実行したアプリケーションが示され、実際のメッセージが続きます。この例では、journal ログの初めの 3 つのエントリーを表示しています。

```
# journalctl
-- Logs begin at Thu 2013-08-01 15:42:12 CEST, end at Thu 2013-08-01 15:48:48 CEST. --
Aug 01 15:42:12 localhost systemd-journal[54]: Allowing runtime journal files to grow to
49.7M.
Aug 01 15:42:12 localhost kernel: Initializing cgroup subsys cpuset
Aug 01 15:42:12 localhost kernel: Initializing cgroup subsys cpu

[...]
```

多くのケースでは、journal ログの最新のエントリーにのみ関連します。**journalctl** 出力を減らす最も簡単な方法は、**-n** オプションを指定して、指定された数の最新のログエントリーのみをリスト表示することです。

```
journalctl -n Number
```

Number を、表示する行数に置き換えます。数字を指定しないと、**journalctl** では最新の 10 エントリーが表示されます。

journalctl コマンドを使用すると、以下の構文で出力形式をできます。

```
journalctl -o form
```

form を、出力形式を指定するキーワードに置き換えます。複数のオプションがあります。すべてのフィールドを含む完全な構造化エントリーアイテムを返す **verbose**。バックアップおよびネットワーク転送に適したバイナリーストリームを作成する **export**。JSON データ構造としてエントリーをフォーマットする **json**。キーワードの完全なリストについては、**journalctl(1)** man ページを参照してください。

例23.19 詳細な journalctl 出力

すべてのエントリーに関する完全なメタデータを表示するには、以下を入力します。

```
# journalctl -o verbose
[...]

Fri 2013-08-02 14:41:22 CEST
[s=e1021ca1b81e4fc688fad6a3ea21d35b;i=55c;b=78c81449c920439da57da7bd5c56a770;m=
27cc
    _BOOT_ID=78c81449c920439da57da7bd5c56a770
    PRIORITY=5
    SYSLOG_FACILITY=3
    _TRANSPORT=syslog
    _MACHINE_ID=69d27b356a94476da859461d3a3bc6fd
    _HOSTNAME=localhost.localdomain
    _PID=562
    _COMM=dbus-daemon
    _EXE=/usr/bin/dbus-daemon
    _CMDLINE=/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --
systemd-activation
    _SYSTEMD_CGROUP=/system/dbus.service
    _SYSTEMD_UNIT=dbus.service
    SYSLOG_IDENTIFIER=dbus
    SYSLOG_PID=562
    _UID=81
    _GID=81
    _SELINUX_CONTEXT=system_u:system_r:system_dbusd_t:s0-s0:c0.c1023
    MESSAGE=[system] Successfully activated service 'net.reactivated.Fprint'
    _SOURCE_REALTIME_TIMESTAMP=1375447282839181

[...]
```

この例では、単一ログエントリーを識別するフィールドをリスト表示しています。「高度なフィルタリング」 の説明にあるように、これらのメタデータはメッセージのフィルタリングに使用できます。全フィールドの説明は、**systemd.journal-fields(7)** man ページを参照してください。

23.10.2. アクセス制御

デフォルトでは、**root** 権限のない Journal ユーザーは、自身の生成したログファイルしか見れません。システム管理者が特定のユーザーを adm グループに追加すると、これらユーザーはすべてのログファイルにアクセスできるようになります。これを実行するには、**root** で次のコマンドを実行します。

```
usermod -a -G adm username
```

ここで、adm グループに追加されるように、username をユーザーの名前に置き換えます。次に、このユーザーは、root ユーザーと同じ **journalctl** コマンドの出力を受け取ります。アクセス制御は、永続ストレージが Journal に対して有効な場合にのみ機能することに注意してください。

23.10.3. ライブビューの使用

パラメーターを付けずに **journalctl** を呼び出すと、収集されたエントリーのうち最も古いものからリスト表示します。ライブビューでは、新たなエントリーが現れると継続的に印刷されるので、リアルタイムでログメッセージを監視できます。journalctl をライブビューモードで開始するには、以下を入力します。

```
journalctl -f
```

このコマンドは、最新のログ 10 行を返します。その後は journalctl ユーティリティーの実行が継続され、新たな変化があるとそれを直ちに表示します。

23.10.4. メッセージのフィルタリング

パラメーターを付けずに実行した **journalctl** コマンドの出力は大規模なものになることが多いため、様々なフィルタリング方法を使うとユーザーのニーズに合った情報を抽出できます。

優先度によるフィルタリング
ログメッセージは、システム上の間違った動作を追跡するために使用されることがよくあります。特定のエントリーまたはより高い優先度のエントリーのみを表示するには、以下の構文を使用します。

```
journalctl -p priority
```

priority を、キーワード **debug** (7)、**info** (6)、**notice** (5)、**warning** (4)、**err** (3)、**crit** (2)、**alert** (1)、または **emerg** (0) のいずれか (または数字) に置き換えます。

例23.20 優先度によるフィルタリング

error もしくはそれ以上の優先度のエントリーのみを表示するには、以下を使用します。

```
journalctl -p err
```

時間によるフィルタリング
現在のブートのログエントリーのみを表示するには、以下を入力します。

```
journalctl -b
```

時折システムを再起動した場合は、**-b** を指定しても **journalctl** の出力が大幅に減りません。この場合は、時間ベースのフィルタリングの方が役に立ちます。

```
journalctl --since=value --until=value
```

--since および--until を使用すると、指定した時間範囲内に作成されたログメッセージのみを表示できます。以下の例のように、日付や時刻の形式で、これらのオプションに値 を渡すことができます。

> *例23.21 時間および優先度によるフィルタリング*
>
> *フィルタリングオプションは組み合わせることで、特定のリクエストに沿って結果を絞り込むことができます。たとえば、warning またはそれ以上の優先度で、特定の時刻以降のメッセージのみを表示するには、以下を使用します。*
>
> ```
> journalctl -p warning --since="2013-3-16 23:59:59"
> ```

高度なフィルタリング
例23.19「詳細な journalctl 出力」 では、ログエントリーを特定し、フィルタリングに使用できるフィールドをリスト表示しています。systemd が保存できるメタデータの完全な説明は、systemd.journal-fields(7) man ページを参照してください。各ログメッセーに対するこのメタデータは、ユーザーが介入することなく収集されます。値は通常テキストベースですが、バイナリーの大きな値になることもあります。フィールドには複数の値がある場合もありますが、一般的ではありません。

指定されたフィールドで発生する一意の値をリスト表示するには、以下の構文を使用します。

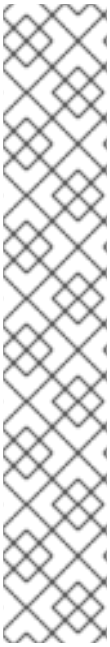
```
journalctl -F fieldname
```

fieldname *を関心のあるフィールド名に置き換えます。*

特定条件のみに合致するログエントリーのみを表示するには、以下の構文を使用します。

```
journalctl fieldname=value
```

fieldname *をフィールド名に、*value *をそのフィールドに含まれる特定の値に置き換えます。そうすると、この条件に合致する行のみが返されます。*



> *注記*
>
> *systemd に保存されているメタデータフィールドはかなりの数になるので、関心のあるフィールド名そのものを忘れることがよくあります。名前が不確かな場合は、以下を入力します。*
>
> ```
> journalctl
> ```
>
> *そして、Tab キーを 2 回押します。これで、利用可能なフィールド名がリスト表示されます。コンテキストベースの Tab 補完入力はフィールド名で機能するので、フィールド名の明確な文字を入力して Tab を押すと、名前が自動的に完了します。同様に、フィールドから一意の値をリスト表示することもできます。タイプ:*
>
> ```
> journalctl fieldname=
> ```
>
> *そして Tab を 2 回押します。これは、journalctl -F* fieldname *と同等の動作になります。*

1 つのフィールドに複数の値を指定できます。

```
journalctl fieldname=value1 fieldname=value2 ...
```

■

同一フィールドで2 つの値を指定すると、2 つの値の論理 **OR** 演算が行われます。value1 またはvalue2 に一致するエントリーが表示されます。

複数のフィールドと値の組み合わせを指定して、出力をさらにしぼり込むこともできます。

> ***journalctl* fieldname1=value fieldname2=value ...**

異なるフィールド名に対して2 つの値を指定すると、値は論理 **AND** で合算されます。エントリーが表示されるには、両方の条件を満たす必要があります。

+ 記号を使うと、複数のフィールドに対して複数の値の論理**OR** 演算を行えます。

> ***journalctl* fieldname1=value + fieldname2=value ...**

このコマンドは、両方の条件に合致するエントリーだけでなく、少なくとも条件の1 つに合致するエントリーを返します。

> 例23.22 高度なフィルタリング
>
> UID 70 のユーザーが **avahi-daemon.service** または **crond.service** で作成したエントリーを表示するには、以下のコマンドを使用します。
>
> > ***journalctl _UID=70 _SYSTEMD_UNIT=avahi-daemon.service _SYSTEMD_UNIT=crond.service***
>
> **_SYSTEMD_UNIT** フィールドに2 つの値があるため、両方の結果が表示されますが、これは **_UID=70** の条件に合致する場合のみです。これは単に (UID=70 and (avahi or cron)) と表すこともできます。

上記のフィルタリングをライブビューモードで適用して、特定のログエントリーグループにおける最新の変更を追跡することもできます。

> ***journalctl -f* fieldname=value ...**

23.10.5. 永続的ストレージの有効化

Journal はデフォルトでは、ログファイルをメモリーか **/run/log/journal/** ディレクトリー内の小さいリングバッファーにのみ保存します。これは、**journalctl** の最近のログ履歴を表示するには十分なものです。このディレクトリーは揮発性なので、ログデータは永続的には保存されません。デフォルト設定では、syslog は journal ログを読み取り、**/var/log/** ディレクトリーに保存します。永続的なロギングが有効になると、journal ファイル **/var/log/journal** に保存され、再起動後も維持されます。これにより、一部のユーザーにとっては、Journal が rsyslog の代わりとなることも可能です (ただし、本章の序論を参照のこと)。

永続的ストレージを有効にすると、以下の利点があります。

- 長期的なトラブルシュートに使用できる、より豊富なデータが記録されます。
- 直ちにトラブルシュートを行う場合には、再起動後により多くのデータが利用可能になります。
- サーバーコンソールがログファイルからではなく、journal からデータを読み取ります。

永続的なストレージには、不利な点もあります。

- 永続的なストレージを使用しても、保存されるデータ量はメモリーの空き容量に依存するため、全期間を保存できる保証はありません。
- ログにより、多くのディスクスペースが必要になります。

Journal 用に永続的なストレージを有効にするには、以下の例のように手動で journal ディレクトリーを作成します。**root** で以下のコマンドを実行します。

```
mkdir -p /var/log/journal/
```

その後に **journald** を再起動して、変更を適用します。

```
systemctl restart systemd-journald
```

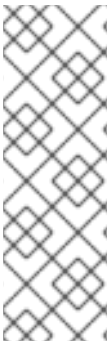
23.11. グラフィカル環境でのログファイルの管理

上記のコマンドラインユーティリティー以外に、Red Hat Enterprise Linux 7 はログメッセージ管理用の使いやすい GUI を提供します。

23.11.1. ログファイルの表示

ほとんどのログファイルはプレーンなテキスト形式で保存されるため、Vi や Emacs などのテキストエディターで表示できます。**Vi** や Emacs などのテキストエディターで表示できます。一部のログファイルは、システム上のすべてのユーザーが読み取り可能ですが、ほとんどのログファイルを読み取るには root 権限が必要になります。

インタラクティブなリアルタイムアプリケーションでシステムのログファイルを表示するには、System Log を使用します。



注記

System Log を使用するには、まず **root** で以下のコマンドを実行して gnome-system-log パッケージがインストールされていることを確認します。

```
~]# yum install gnome-system-log
```

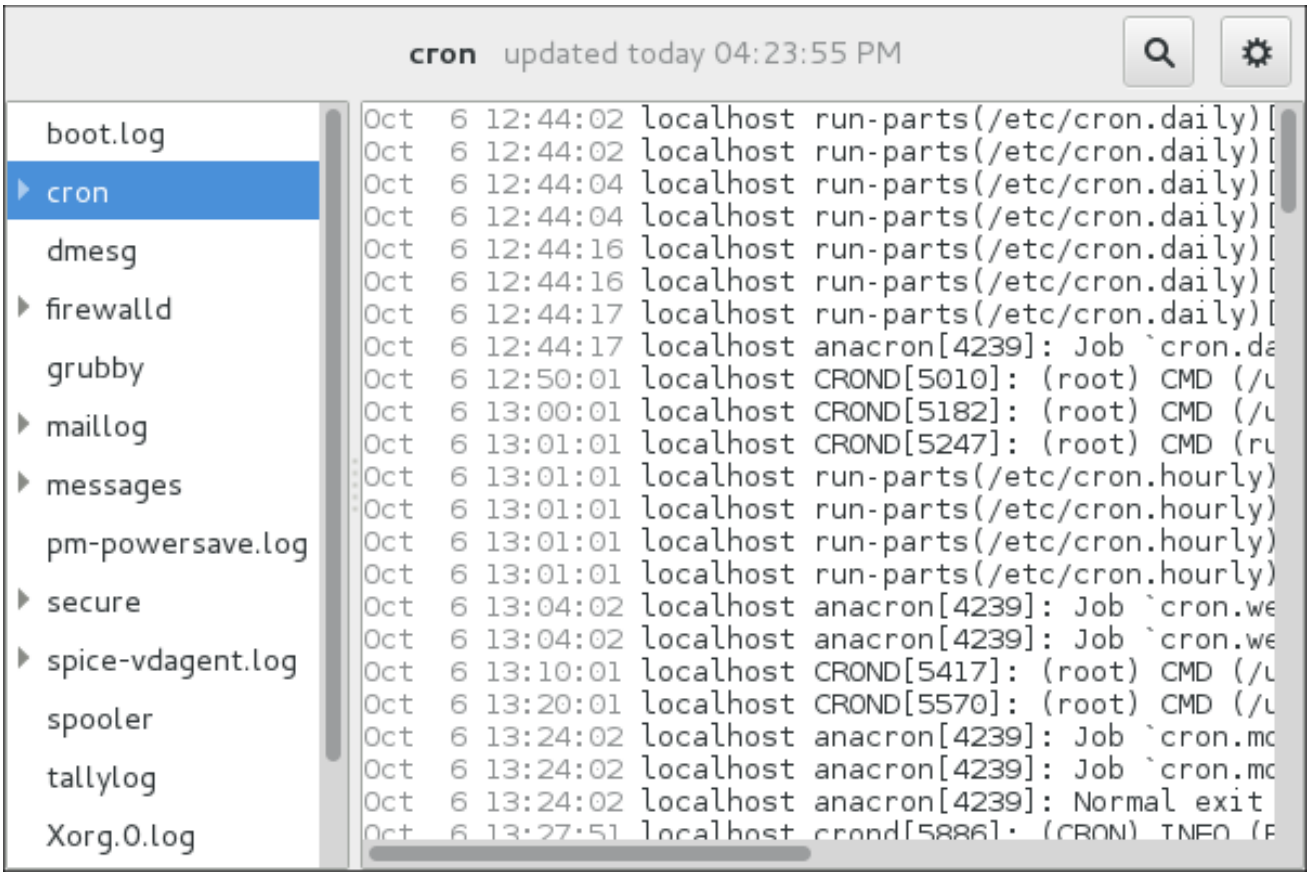
yum を使用したパッケージのインストールは 「パッケージのインストール」 を参照してください。

gnome-system-log パッケージをインストールしたら、Applications → System Tools → System Log の順にクリックするか、シェルプロンプトで以下のコマンドを入力して、System Log を開きます。

```
~]$ gnome-system-log
```

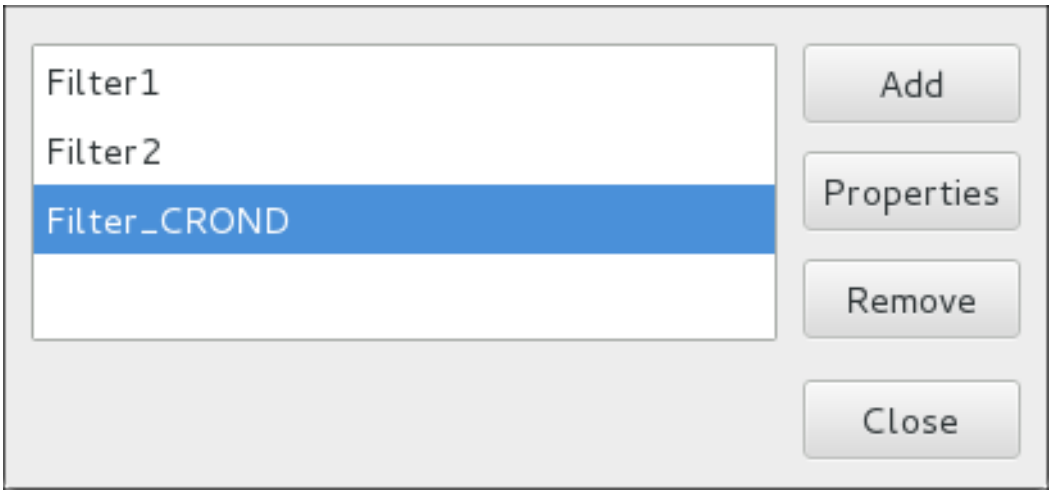
このアプリケーションは、存在するログファイルのみを表示します。そのため、図23.2「システムログ」 で表示されているリストとは異なる場合があります。

図23.2 システムログ



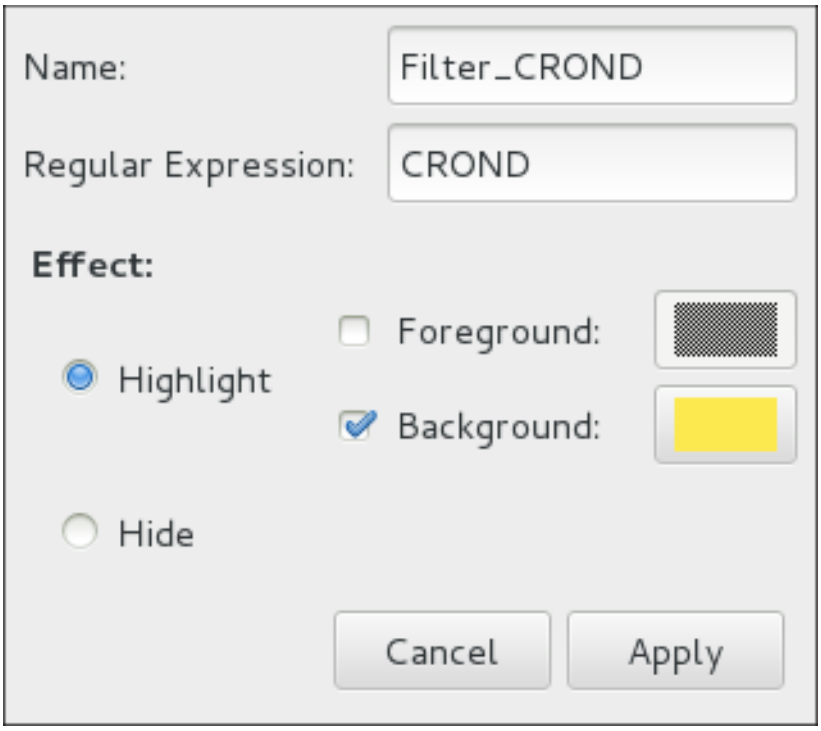
*システムログ アプリケーションを使用すると、既存のログファイルをフィルタリングできます。ギアの記号で示されたボタンをクリックしてメニューを表示し、**Filters** > >**Manage Filters** を選択して必要なフィルターを定義または編集します。*

図23.3 システムログ- フィルター



フィルターを追加または編集することで、図23.4「システムログ- フィルターの定義」のようにパラメーターを定義できます。

図23.4 システムログ- フィルターの定義

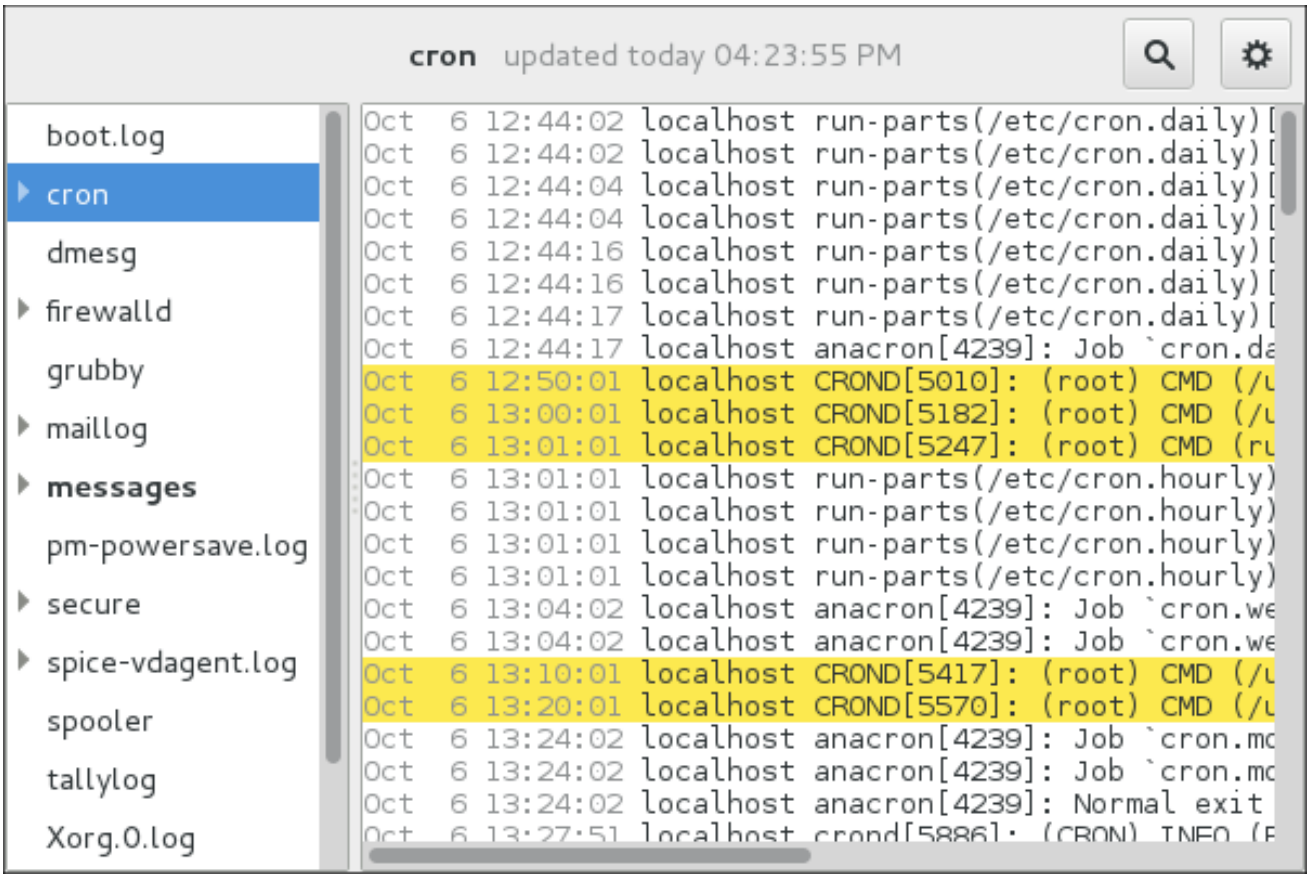


フィルターを定義する際には、以下のパラメーターを編集できます。

- *名前 - フィルターの名前を指定します。*
- *正規表現 - ログファイルに適用され、その中の実行可能なテキストの文字列に一致するよう試行する正規表現を指定します。*
- ***Effect***
  - ***Hilight**: これが有効な場合、検索結果は選択した色で強調されます。テキストのバックグラウンドまたはフォアグラウンドを強調するかどうかを選択できます。*
  - ***Hide**: これが有効な場合は、検索結果は閲覧中のログファイルからは非表示になります。*

少なくとも1つのフィルターが定義されていれば、フィルター メニューからそれを選択することができ、そのフィルターが定義済みの文字列を自動的に検索して、現在表示しているログファイル内でのマッチを強調表示または非表示にします。

図23.5 システムログ- フィルターの有効化

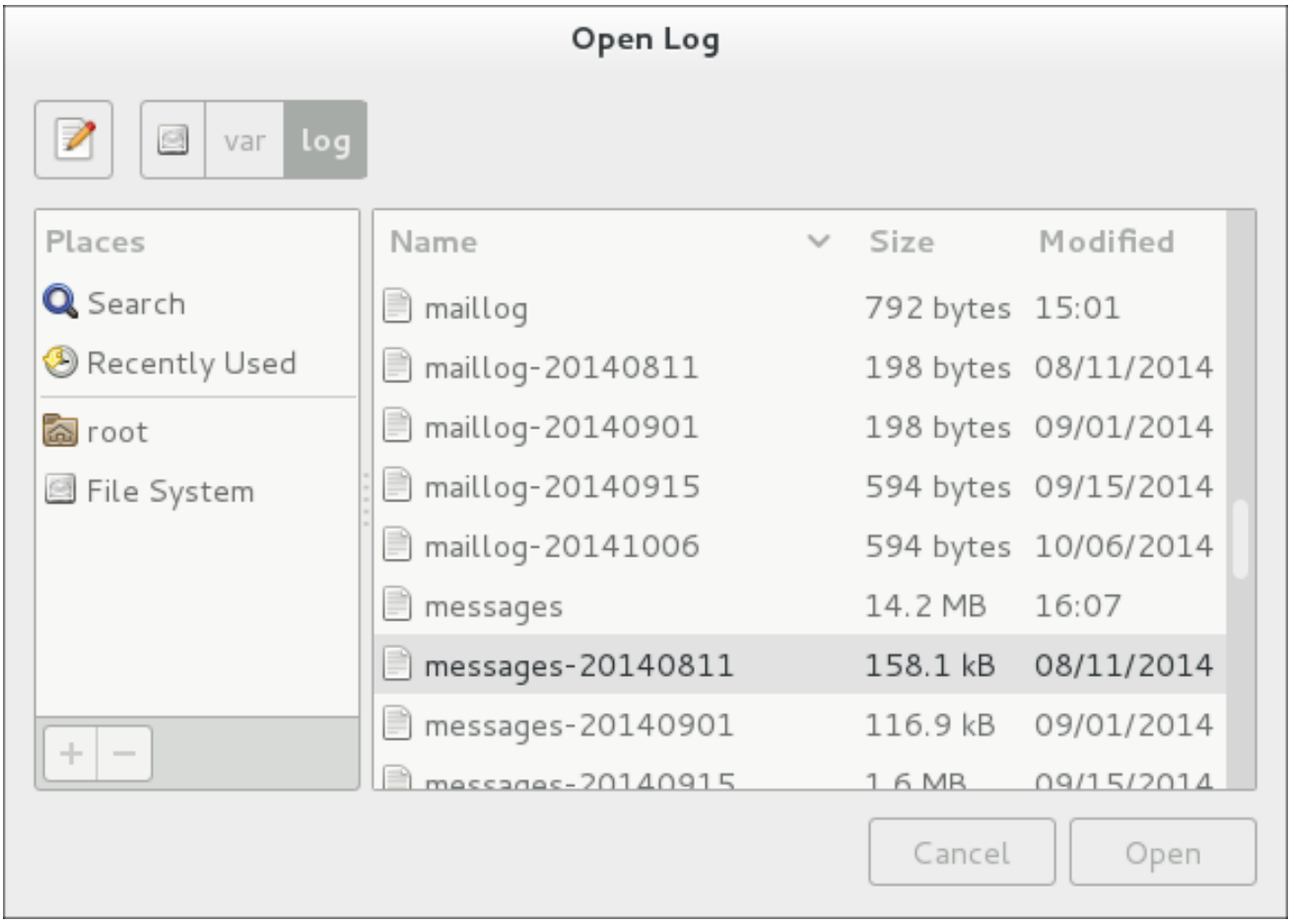


マッチしたもののみ表示 オプションを選択すると、現在表示されているログファイル内でマッチした文字列のみが表示されます。

23.11.2. ログファイルの追加

*リストに表示するログファイルを追加するには、**File → Open →** の順に選択します。これにより、表示するログファイルのディレクトリーおよびファイル名を選択できる **Open Log** ウィンドウが表示されます。図23.6「システムログ- ログファイルの追加」 では、**Open Log** ウィンドウを示しています。*

図23.6 システムログ- ログファイルの追加



ファイルを開くには、開く ボタンをクリックします。ファイルは表示中のリストに直ちに追加されるため、そのファイルを選択してコンテンツを表示できます。

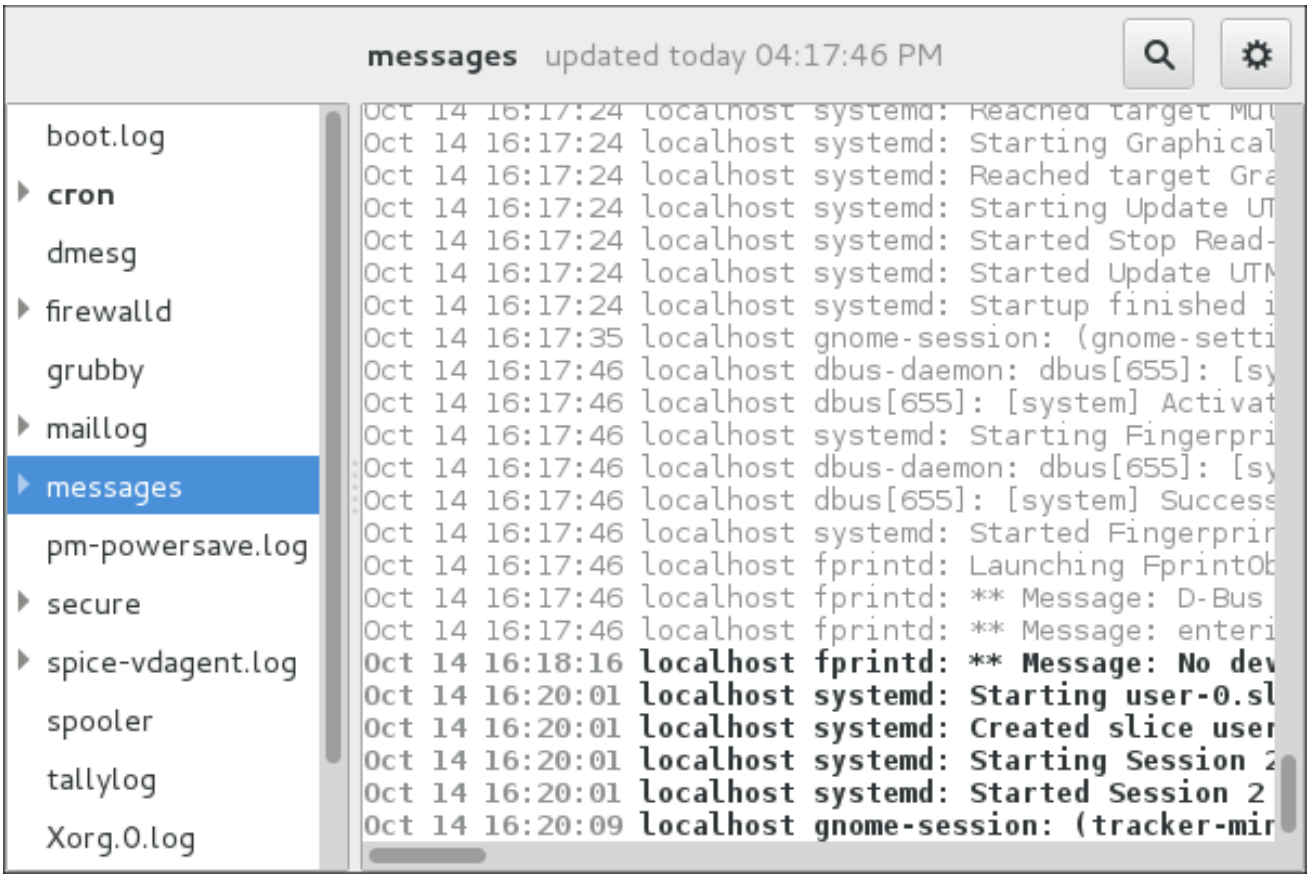
注記

*System Log を使用すると、**.gz** 形式のログファイルを開くことができます。*

23.11.3. ログファイルのモニタリング

*システムログはデフォルトで、開いているすべてのログを監視します。監視されているログファイルに新しい行が追加されると、そのログ名はログのリストに太字で表示されます。ログファイルが選択または表示されると、ログファイルの下部に新しい行が太字で表示されます。図23.7「システムログ- 新しいログ通知」 は、**cron** ログファイルと **messages** ログファイル内の新しいアラートを示しています。**messages** ログファイルをクリックすると、ファイル内のログが表示されます (新しい行は太字で表示されます)。*

図23.7 システムログ - 新しいログ通知



23.12. 関連情報

rsyslog デーモンの設定方法、およびログファイルの場所の特定、表示、モニタリング方法に関する詳細情報は、以下のリソースを参照してください。

インストールされているドキュメント

- **rsyslogd**(8): **rsyslogd** デーモンの man ページは、その使用方法を説明しています。
- **rsyslog.conf**(5): **rsyslog.conf** の man ページは、利用可能な設定オプションを説明しています。
- **logrotate**(8): logrotate ユーティリティーの man ページは、その設定方法と使用方法を詳細に説明しています。
- **journalctl**(1): **journalctl** デーモンの man ページは、その使用方法を説明しています。
- **journald.conf**(5): この man ページは、利用可能な設定オプションを説明しています。
- **systemd.journal-fields**(7): この man ページでは、特別な Journal フィールドをリスト表示しています。

インストール可能なドキュメント

/usr/share/doc/rsyslogversion**/html/index.html**: このファイルはオプションチャンネルの rsyslog-doc パッケージで提供され、rsyslog に関する情報を含みます。Red Hat 追加チャンネルの詳細については、「Optional および Supplementary リポジトリーの追加」を参照してください。ドキュメンテーションにアクセスする前に、**root** で以下のコマンドを実行する必要があります。

```
~]# yum install rsyslog-doc
```

オンラインドキュメント

rsyslog ホームページは、追加のドキュメンテーション、設定例、および動画チュートリアルを提供します。使用しているバージョンに関連するドキュメントを参照してください。

- RainerScript *documentation on the rsyslog Home Page*: RainerScript *で利用可能なデータタイプ、式、関数についての概要です。*
- *rsyslog ホームページ上の rsyslog バージョン 7 ドキュメンテーション*: *Red Hat Enterprise Linux 7 では、rsyslog のバージョン 7 はrsyslog パッケージで利用可能です。*
- *Description of* queues *on the rsyslog Home Page*- *様々なタイプのメッセージキューおよびその使用方法に関する全般情報です。*

関連項目

- *6 章権限の取得 では、**su** および**sudo** コマンドを使用して管理者権限を取得する方法を説明しています。*
- *10 章systemd によるサービス管理 では、systemd の詳細情報と、**systemctl** コマンドを使用してシステムサービスを管理する方法が説明されています。*

第24章 システムタスクの自動化

Red Hat Enterprise Linux は、タスク (**ジョブ** とも呼ばれます) を自動的に実行するように設定できます。

- 指定した時間に定期的に実行するには **cron** を使用します。「cron を使用した繰り返しジョブのスケジュール設定」 を参照してください。
- 特定の日に非同期的に実行するには、**anacron** を使用します。「Anacron を使用した繰り返しの非同期ジョブのスケジュール設定」 を参照してください。
- 特定の時間に 1 回実行するには **at** を使用します。「at を使用した特定の時間にジョブを実行するスケジュールの設定」 を参照してください。
- システムの負荷平均が指定した値を下回ったときに 1 回実行するには **batch** を使用します。「batch を使用した System Load Drop で実行するジョブのスケジュール設定」を参照してください。
- 次回のブート時に 1 回実行するときは、「systemd ユニットファイルを使用した次回ブート時のジョブの実行スケジュール」 を参照してください。

本章では、これらのタスクの実行方法を説明します。

24.1. CRON を使用した繰り返しジョブのスケジュール設定

cron は、タスク (別名ジョブ) を定期的に実行するためにスケジュールを設定するサービスです。**cron** のジョブは、設定した時間に**システムが稼働している** 場合のみ実行されます。システムの起動時まで実行を延期して、システムが稼動していない場合にジョブが消失しないようにするスケジューの設定の方法は、「at を使用した特定の時間にジョブを実行するスケジュールの設定」 を参照してください。

ユーザーは、cron テーブルファイル (**crontab** ファイルとも呼ばれます) で**cron** ジョブを指定します。その後、これらのファイルは **crond** サービスが読み取り、ジョブを実行します。

24.1.1. cron ジョブの前提条件

cron ジョブのスケジュール設定を行う前に、以下を行います。

1. **cronie** パッケージをインストールします。

```
~]# yum install cronie
```

2. **crond** サービスはインストール時に有効になっており、ブート時に自動的に開始するよう設定されています。サービスを無効にしている場合は有効にしてください。

```
~]# systemctl enable crond.service
```

3. 現在のセッションで **crond** サービスを開始します。

```
~]# systemctl start crond.service
```

4. **(オプション)** **cron** を設定します。たとえば、以下を変更できます。
   - ジョブの実行時に使用するシェル

- **PATH** 環境変数
- ジョブが **E** メールを送信する場合はメールアドレス
  cron の設定に関する詳細は、crontab(5) man ページを参照してください。

24.1.2. cron ジョブのスケジュール設定

root ユーザーとしてジョブをスケジュール設定する
root ユーザーは **/etc/crontab** にある cron テーブルを使用しますが、**/etc/cron.d/** に cron テーブルファイルを作成することを推奨します。**root** としてジョブをスケジュール設定するときは、この方法を使用します。

1. 以下を選択します。
   - ジョブを実行する時刻 (分)。たとえば、10 分間隔で指定する場合は **0,10,20,30,40,50** または **0/10** を使用します。
   - ジョブを実行する時刻 (時)。たとえば、17:00 から 20:59 までと指定する場合は **17-20** を使用します。
   - ジョブを実行する日。たとえば、15 日と指定する場合は **15** を使用します。
   - ジョブを実行する月。たとえば、夏季の月を指定する場合は **Jun,Jul,Aug** または **6,7,8** を使用します。
   - ジョブを実行する曜日。たとえば、曜日と無関係にジョブを実行する場合は * を使用します。
     選択した値を時間指定と組み合せます。上記の例をこの時間指定に適用すると、以下のようになります。

     0,10,20,30,40,50 17-20 15 Jun,Jul,Aug *
2. ユーザーを指定します。ジョブは、このユーザーが実行したように実行されます。たとえば、**root** を使用します。
3. 実行するコマンドを指定します。たとえば、**/usr/local/bin/my-script.sh** を使用します。
4. 上記の指定を 1 行にまとめると、以下のようになります。

   ```
   0,10,20,30,40,50 17-20 15 Jun,Jul,Aug * root /usr/local/bin/my-script.sh
   ```
5. 完成した行を **/etc/crontab** に追加するか、代わりに **/etc/cron.d/** に cron テーブルファイルを作成し、この行を追加します。後者が推奨されます。

これでジョブはスケジュールされたとおりに実行されます。

ジョブの指定方法に関する詳細は、crontab(5) man ページを参照してください。基本的情報は **/etc/crontab** ファイルの冒頭部分を参照してください。

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
```

```
# .---------------- minute (0 - 59)
# | .------------- hour (0 - 23)
# | | .---------- day of month (1 - 31)
# | | | .------- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
```

root 以外のユーザーとしてジョブをスケジュール設定する
非 root ユーザーは、crontab ユーティリティーを使用して cron ジョブを設定します。このジョブは、このユーザーが実行したように実行されます。

cron ジョブを特定のユーザーとして作成するには、以下を使用します。

1. ユーザーのシェルから以下を実行します。

   ```
   [bob@localhost ~]$ crontab -e
   ```

   これにより、**VISUAL** または **EDITOR** 環境変数により指定したエディターを使用して、ユーザーが自分で **crontab** ファイルを編集できます。

2. ???TITLE??? にある方法と同じ方法でジョブを指定します。たとえば、以下を追加する代わりに、

   ```
   0,10,20,30,40,50 17-20 15 Jun,Jul,Aug * bob /home/bob/bin/script.sh
   ```

   以下を追加します。

   ```
   0,10,20,30,40,50 17-20 15 Jun,Jul,Aug * /home/bob/bin/script.sh
   ```

3. ファイルを保存して、エディターを終了します。

4. **(オプション)** 新しいジョブを確認するときは、以下を実行し、現在のユーザーの crontab ファイルの内容を表示します。

   ```
   [bob@localhost ~]$ crontab -l
   @daily /home/bob/bin/script.sh
   ```

ジョブの時間、日、週、月ごとのスケジュール設定
ジョブを時間、日、週、または月ごとにスケジュールするには、以下を実行します。

1. ジョブに実行させたいアクションをシェルスクリプトに入力します。

2. シェルスクリプトを以下のディレクトリーのうちの1 つに入力します。

   - **/etc/cron.hourly/**
   - **/etc/cron.daily/**
   - **/etc/cron.weekly/**
   - **/etc/cron.monthly/**

以後、ユーザーのスクリプトが実行されます。**crond** サービスが **/etc/cron.hourly**、**/etc/cron.daily**、**/etc/cron.weekly**、および **/etc/cron.monthly** ディレクトリーにあるスクリプトを、対応する時間に自動的に実行します。

24.2. ANACRON を使用した繰り返しの非同期ジョブのスケジュール設定

Anacron は、**cron** と同様に、タスク (別名ジョブ) の定期的な実行をスケジュールするためのサービスです。ただし、**anacron** は、2 つの点で**cron** と異なります。

- 予定した時間にシステムが稼働していなかった場合、**anacron** のジョブはシステムが稼働するまで延期されます。
- **anacron** のジョブは、最大で1 日1 回実行することができます。

ユーザーは、**anacrontab** ファイルとも呼ばれる anacron テーブルファイルで**anacron** ジョブを指定します。その後、これらのファイルは **crond** サービスが読み取り、ジョブを実行します。

24.2.1. Anacrob ジョブの前提条件

anacron ジョブのスケジュール設定を行う前に、以下を行います。

1. cronie-anacron パッケージがすでにインストールされていることを確認します。

   ```
   ~]# rpm -q cronie-anacron
   ```

   cronie-anacron はすでにインストールされる可能性があります。これは、cronie パッケージのサブパッケージであるためです。インストールされていない場合は、以下のコマンドを使用します。

   ```
   ~]# yum install cronie-anacron
   ```

2. **crond** サービスはインストール時に有効になっており、ブート時に自動的に開始するよう設定されています。サービスを無効にしている場合は有効にしてください。

   ```
   ~]# systemctl enable crond.service
   ```

3. 現在のセッションで **crond** サービスを開始します。

   ```
   ~]# systemctl start crond.service
   ```

4. **(オプション)** anacron を設定します。たとえば、以下を変更できます。
   - ジョブの実行時に使用するシェル
   - **PATH** 環境変数
   - ジョブが**E** メールを送信する場合はメールアドレス
     anacron の設定は、anacrontab(5) man ページを参照してください。

重要

デフォルトでは、anacron 設定には、コンピューターが接続していない場合は実行できないようにする条件が含まれています。この設定により、anacron ジョブの実行によりバッテリーが消耗することはありません。

コンピューターがバッテリーで動作している場合でも anacron の実行を許可する場合は、**/etc/cron.hourly/0anacron** ファイルを開いて、次の部分をコメントアウトします。

```
# Do not run jobs when on battery power
online=1
for psupply in AC ADP0 ; do
  sysfile="/sys/class/power_supply/$psupply/online"

  if [ -f $sysfile ] ; then
    if [ `cat $sysfile 2>/dev/null`x = 1x ]; then
      online=1
      break
    else
      online=0
    fi
  fi
done
```

24.2.2. Anacron ジョブのスケジュール設定

root ユーザーとして anacron ジョブのスケジュール設定
root ユーザーは、**/etc/anacrontab** にある anacron テーブルを使用します。**root** としてジョブをスケジュール設定するときは、以下の手順を使用します。

root ユーザーとして anacron ジョブのスケジュール設定

1. 以下を選択します。
   - ジョブを実行する頻度。たとえば、毎日を指定する場合は **1**、3 日に 1 回を指定する場合は **3** を使用します。
   - ジョブ実行の遅延。たとえば、遅延なしを指定する場合は **0**、1 時間の遅延を指定する場合は **60** を使用します。
   - ロギングに使用されるジョブ識別子。ロギングに使用されます。たとえば、**my.anacron.job** 行にジョブをロギングするには、**my.anacron.job** を使用します。
   - 実行するコマンド。たとえば、**/usr/local/bin/my-script.sh** を使用します。
     選択した値をジョブ指定に組み合せます。以下は指定の例です。

     ```
     3 60 cron.daily /usr/local/bin/my-script.sh
     ```

2. 作成された行を **/etc/anacrontab** に追加します。

これでジョブはスケジュールされたとおりに実行されます。

簡単なジョブの例は、**/etc/anacrontab** ファイルを参照してください。ジョブの指定方法に関する詳細は、anacrontab(5) man ページを参照してください。

ジョブの時間、日、週、月ごとのスケジュール設定
ジョブは、anacron を使用して日、週、月ごとにスケジュールを設定できます。「ジョブの時間、日、週、月ごとのスケジュール設定」を参照してください。

24.3. AT を使用した特定の時間にジョブを実行するスケジュールの設定

1 回限りのタスク (別名ジョブ) を指定した時間に 1 回実行するようスケジュール設定するときは、**at** ユーティリティーを使用します。

ユーザーは、**at** ユーティリティーを使用して**at** ジョブを指定します。このジョブはその後**atd** サービスにより実行されます。

24.3.1. at ジョブの前提条件

at ジョブのスケジュール設定を行う前に、以下を行います。

1. **at** パッケージをインストールします。

```
~]# yum install at
```

2. **atd** サービスはインストール時に有効になっており、ブート時に自動的に開始するように設定されています。サービスを無効にしている場合は有効にしてください。

```
~]# systemctl enable atd.service
```

3. 現在のセッションで**atd** サービスを開始します。

```
~]# systemctl start atd.service
```

24.3.2. at ジョブのスケジュール設定

1. ジョブは常に複数のユーザーにより実行されます。希望するユーザーとしてログインし、以下を実行します。

```
~]# at time
```

time を時間指定に置き換えます。

時間の指定に関する詳細は、at(1) man ページと **/usr/share/doc/at/timespec** ファイルを参照してください。

例24.1 at の時間指定

ジョブを 15:00 に実行するには、以下を実行します。

```
~]# at 15:00
```

指定した時間を過ぎると、そのジョブは翌日の同じ時間に実行されます。

ジョブを 2017 年 8 月 20 日に実行するには、以下を実行します。

```
~]# at August 20 2017
```

または

```
~]# at 082017
```

ジョブを 5 日後に実行するには、以下を実行します。

```
~]# now + 5 days
```

2. ***at>** プロンプトが表示されたら、以下のコマンドを入力して実行し、**Enter** を押します。*

```
~]# at 15:00
at> sh /usr/local/bin/my-script.sh
at>
```

実行したいすべてのコマンドにこの手順を繰り返します。

注記

***at>** プロンプトに、使用されるシェルが表示されます。*

```
warning: commands will be executed using /bin/sh
```

***at** ユーティリティーは、ユーザーの **SHELL** 環境変数にあるシェルのセット、ユーザーのログインシェル、または **/bin/sh** の、いずれか最初に発見されたものを使用します。*

3. *空の行で **Ctrl+D** キーを押し、ジョブの指定を完了します。*

注記

コマンドセットやスクリプトが標準出力に情報を表示しようとする場合、その出力はユーザーにメールで送信されます。

保留中のジョブの表示
*保留中のジョブリストを表示するには、**atq** コマンドを使用します。*

```
~]# atq
26   Thu Feb 23 15:00:00 2017 a root
28   Thu Feb 24 17:30:00 2017 a root
```

各ジョブは、個別の行に以下のフォーマットで表示されます。

```
job_number scheduled_date scheduled_hour job_class user_name
```

***job_queue** カラムは、ジョブが **at** または **batch** のいずれのジョブであるかを指定します。**a** は **at** を表します。**b** は **batch** を表します。*

非 root ユーザーが閲覧できるのは、自分のジョブのみです。root ユーザーは、すべてのユーザーのジョブを閲覧できます。

スケジュール設定したジョブの削除

スケジュール設定したジョブを削除するには、以下を行います。

1. **atq** コマンドを使用して、保留中のジョブをリスト表示します。

```
~]# atq
26   Thu Feb 23 15:00:00 2017 a root
28   Thu Feb 24 17:30:00 2017 a root
```

2. スケジュールを設定した時間とユーザーを使用しえ、削除するジョブを検索します。

3. ジョブを番号で指定し、**atrm** コマンドを実行します。

```
~]# atrm 26
```

24.3.2.1. at と batch へのアクセスの制御

特定ユーザーによる **at** と **batch** コマンドへのアクセスを制限できます。次のルールに従って、ユーザー名を **/etc/at.allow** または **/etc/at.deny** に入力してください。

- 両方のアクセス制御ファイルは、同じフォーマットを使用します。ユーザー名は、各行に 1 人ずつです。
- いずれのファイルでも、空白は許可されません。
- **at.allow** ファイルが存在する場合は、ファイルに記載されているユーザーのみが **at** または **batch** を使用でき、**at.deny** ファイルは無視されます。
- **at.allow** がない場合は、**at.deny** に記載されているユーザーは **at** または **batch** を使用できません。
- **root** ユーザーはアクセス制御ファイルの影響を受けず、常に **at** コマンドおよび **batch** コマンドを実行できます。

アクセス制御ファイルを変更した場合でも、**at** デーモン (**atd**) を再起動する必要はありません。アクセス制御ファイルは、ユーザーが **at** または **batch** のコマンドの実行を試みるたびに読み込まれます。

24.4. BATCH を使用した SYSTEM LOAD DROP で実行するジョブのスケジュール設定

1 回限りのタスク (別名ジョブ) を、システムの平均負荷が指定値を下回ったときに実行するようスケジュール設定するときは、**batch** ユーティリティーを使用します。リソース負荷の高いタスクを実行したり、システムがアイドル状態になるのを防いだりするときに有用です。

ユーザーは、**batch** ユーティリティーを使用して batch ジョブを指定します。このジョブはその後 **atd** サービスにより実行されます。

24.4.1. Batch ジョブの前提条件

batch ユーティリティーは **at** パッケージで提供され、**batch** ジョブは **atd** サービスが管理します。したがって **batch** ジョブの前提条件は、**at** ジョブの前提条件と同じです。「at ジョブの前提条件」を参照してください。

24.4.2. Batch ジョブのスケジュール設定

1. ジョブは常に複数のユーザーにより実行されます。希望するユーザーとしてログインし、以下を実行します。

```
~]# batch
```

2. **at>** プロンプトが表示されたら、以下のコマンドを入力して実行し、**Enter** を押します。

```
~]# batch
at> sh /usr/local/bin/my-script.sh
```

実行したいすべてのコマンドにこの手順を繰り返します。

注記

at> プロンプトに、使用されるシェルが表示されます。

```
warning: commands will be executed using /bin/sh
```

batch ユーティリティーは、ユーザーの **SHELL** 環境変数にあるシェルのセット、ユーザーのログインシェル、または **/bin/sh** のうち最初に見つかったものを使用します。

3. 空の行で **Ctrl+D** キーを押し、ジョブの指定を完了します。

注記

コマンドセットやスクリプトが標準出力に情報を表示しようとする場合、その出力はユーザーにメールで送信されます。

デフォルトのシステム負荷平均の制限の変更
デフォルトでは、**batch** ジョブはシステムの負荷平均が **0.8** を下回ったときに開始されます。この設定は、**atq** サービスでも維持されます。システム負荷の限界を変更するには、以下を行います。

1. **/etc/sysconfig/atd** ファイルに以下の行を追加します。

```
OPTS='-l x'
```

x を新しい負荷平均と置き換えます。以下に例を示します。

```
OPTS='-l 0.5'
```

2. **atq** サービスを再起動します。

```
# systemctl restart atq
```

保留中のジョブの表示
保留中のジョブリストを表示するには、**atq** コマンドを使用します。「保留中のジョブの表示」を参照してください。

スケジュール設定したジョブの削除
スケジュールを設定したジョブを削除するには、**atrm** コマンドを使用します。「スケジュール設定したジョブの削除」を参照してください。

Batch へのアクセス制御
batch ユーティリティーの使用を制限することもできます。これは、batch と at ユーティリティーの両方に同時に実行されます。「at と batch へのアクセスの制御」を参照してください。

24.5. SYSTEMD ユニットファイルを使用した次回ブート時のジョブの実行スケジュール

cron、anacron、at、batch ユーティリティーを使うと、特定の時間に、またはシステム負荷が特定のレベルに達したときに、ジョブを実行するように設定できます。また、次回のシステムブート時に実行するジョブを作成することもできます。これは、実行するスクリプトとその依存関係を指定する systemd ユニットファイルを作成することで行います。

次回の起動時に実行するスクリプトを設定するには、以下を実行します。

1. *起動プロセスのどの段階でスクリプトを実行するかを指定する systemd ユニットファイルを作成します。この例で挙げているのは、Wants= と After= の合理的な一連の依存関係を備えたユニットファイルです。*

```
~]# cat /etc/systemd/system/one-time.service
[Unit]
# The script needs to execute after:
# network interfaces are configured
Wants=network-online.target
After=network-online.target
# all remote filesystems (NFS/_netdev) are mounted
After=remote-fs.target
# name (DNS) and user resolution from remote databases (AD/LDAP) are available
After=nss-user-lookup.target nss-lookup.target
# the system clock has synchronized
After=time-sync.target

[Service]
Type=oneshot
ExecStart=/usr/local/bin/foobar.sh

[Install]
WantedBy=multi-user.target
```

   この例を使用すると、以下が可能となります。

   - */usr/local/bin/foobar.sh を、自分のスクリプトの名前に置き換えます。*
   - *必要に応じて After= エントリーのセットを変更します。*
     起動プロセスの段階を指定する方法は「systemd のユニットファイルの作成および変更」を参照してください。

2. *スクリプトの実行後も systemd サービスをアクティブに維持したいときは、RemainAfterExit=yes 行を [Service] セクションに追加します。*

```
[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/local/bin/foobar.sh
```

3. *systemd デーモンをリロードします。*

```
~]# systemctl daemon-reload
```

4. ***systemd*** *サービスを有効にします。*

```
~]# systemctl enable one-time.service
```

5. *以下を実行するスクリプトを作成します。*

```
~]# cat /usr/local/bin/foobar.sh
#!/bin/bash

touch /root/test_file
```

6. *スクリプトを次回のブート時のみに実行したいときは、****systemd*** *ユニットを無効にする行を追加します。*

```
#!/bin/bash

touch /root/test_file
systemctl disable one-time.service
```

7. *スクリプトを実行可能にします。*

```
~]# chmod +x /usr/local/bin/foobar.sh
```

24.6. 関連情報

Red Hat Enterprise Linux でシステムタスクを自動化する方法には、下記の参考文献をご覧ください。

インストールされているドキュメント

- *cron: crond デーモンの man ページでは、****crond*** *の仕組みと、その動作の変更方法を記載しています。*
- *crontab: crontab ユーティリティーの man ページで、対応しているオプションの一覧を記載しています。*
- *crontab(5): crontab ユーティリティーのマニュアルページのセクションに、****crontab*** *ファイルのフォーマットを記載しています。*

第25章 自動バグ報告ツール (ABRT)

25.1. ABRT の概要

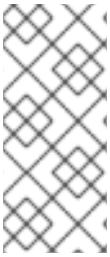
Automatic Bug Reporting Tool は通常ABRT と呼ばれるツールセットで、ユーザーがアプリケーションクラッシュを検出し、報告する手助けとなるように設計されています。主な目的は、問題の報告と解決方法の発見のプロセスを容易にすることです。この場合の解決方法には、Bugzilla チケット、ナレッジベースの記事、修正を含むバージョンへの更新の提案などが含まれます。

ABRT は、**abrtd** デーモンと、検出された問題を処理、分析、および報告する多数のシステムサービスおよびユーティリティーで設定されます。デーモンは、アプリケーションがクラッシュしたりカーネルの Oop が検出されると、ほとんどの時間と Springs のバックグラウンドでサイレントに実行されます。次に、デーモンは、コアファイルがある場合にコアファイル、クラッシュアプリケーションのコマンドラインパラメーター、およびフォレンジックユーティリティーのその他のデータなど、関連する問題データを収集します。

ABRT は現在、C、C++、Java、Python、および Ruby プログラミング言語で書かれたアプリケーションにおけるクラッシュと、X.Org クラッシュ、カーネルの oops、およびカーネルパニックの検出をサポートしています。対応している障害やクラッシュのタイプの詳細や、さまざまなタイプのクラッシュが検出される仕組みは、「ソフトウェア問題の検出」 を参照してください。

特定された問題はリモートの問題トラッカーに報告され、このレポーティングは、問題が検出された際に自動的に報告されるように設定することが可能です。問題データは、ローカルまたは専用のシステムに保存して、ユーザーが手動でレビュー、報告、削除することも可能です。報告ツールは、Bugzilla データベースまたは Red Hat テクニカルサポート (RHTSupport) の Web サイトに問題データを送信できます。ツールは、 **FTP** または **SCP** を使用してアップロードしたり、電子メールとして送信したり、ファイルに書き込んだりすることもできます。

既存の問題データを処理する ABRT コンポーネントは (新規の問題データの作成などとは対照的に)、libreport という別のプロジェクトの一部です。libreport ライブラリーは問題の分析および報告のための包括的メカニズムを提供し、ABRT 以外のアプリケーションでも使用されます。ただし、ABRT および libreport 操作と設定は密接に統合されています。したがって、それらはこのドキュメントの 1 つとして説明されています。



注記

ABRT レポートが生成されるのは、コアダンプが生成された時のみという点に注意してください。コアダンプは、一部のシグナル向けのみに生成されます。たとえば、SIGKILL (-9) はコアダンプを生成しないため、ABRT はこの失敗をキャッチできません。シグナルおよびコアダンプの生成に関する詳細は、man 7 シグナルを参照してください。

25.2. ABRT のインストールとサービスの起動

ABRT を使用するには、abrt-desktop または abrt-cli パッケージがシステムにインストールされていることを確認します。abrt-desktop パッケージは ABRT 用のグラフィカルユーザーインターフェイスを提供し、abrt-cli パッケージにはコマンドラインで ABRT を使用するためのツールが含まれています。この両方をインストールすることもできます。ABRT GUI とコマンドラインツールのいずれを使用しても、全般的な手順は大きく変わりません。

警告

ABRT パッケージをインストールすると、コアダンプファイルの命名に使用するテンプレートを含む /proc/sys/kernel/core_pattern ファイルが上書きされることに注意してください。このファイルのコンテンツは、以下のように上書きされます。

```
|/usr/libexec/abrt-hook-ccpp %s %c %p %u %g %t e
```

Yum パッケージマネージャーでパッケージをインストールする方法は「パッケージのインストール」を参照してください。

25.2.1. ABRT GUI のインストール

ABRT グラフィカルユーザーインターフェイス は、デスクトップ環境での作業用の操作が容易なフロントエンドを提供します。root ユーザーで以下のコマンドを実行すると、必要なパッケージがインストールできます。

```
~]# yum install abrt-desktop
```

インストール時、グラフィカルデスクトップセッションの開始時に ABRT 通知アプレットが自動的に開始するように設定されます。ターミナルで以下のコマンドを発行すると、ABRT アプレットが実行中であることを確認できます。

```
~]$ ps -el | grep abrt-applet
0 S  500 2036 1824 0 80  0 - 61604 poll_s ?    00:00:00 abrt-applet
```

アプレットが実行していない場合には、以下のように abrt-applet プログラムを実行すると、現行のデスクトップセッションでアプレットを手動で起動できます。

```
~]$ abrt-applet &
[1] 2261
```

25.2.2. コマンドラインによる ABRT のインストール

コマンドラインインターフェイス は、ヘッドレスマシンやネットワーク接続されたリモートシステム、またはスクリプト内で役に立ちます。root ユーザーで以下のコマンドを実行すると、必要なパッケージがインストールできます。

```
~]# yum install abrt-cli
```

25.2.3. 補助 ABRT ツールのインストール

ABRT が検出するクラッシュに関するメール通知を受け取るには、libreport-plugin-mailx パッケージがインストール済みである必要があります。root で以下のコマンドを実行すると、このパッケージをインストールできます。

```
~]# yum install libreport-plugin-mailx
```

デフォルトでは、これで通知はローカルマシンの **root** ユーザーに送信されます。メールの宛先は、**/etc/libreport/plugins/mailx.conf** ファイルで設定できます。

ログイン時にコンソールに通知を表示するには、abrt-console-notification パッケージもインストールします。

ABRT では様々なタイプのソフトウェア障害を検出し、分析し、報告できます。ABRT はデフォルトで、C および C++ のアプリケーションのクラッシュなど、最も一般的な障害のサポートと共にインストールされます。他のタイプの障害サポートは各パッケージで提供されます。たとえば、Java 言語で作成されたアプリケーションの例外を検出するサポート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 abrt-java-connector
```

ABRT がサポートする言語およびソフトウェアプロジェクトのリストは、「ソフトウェア問題の検出」を参照してください。このセクションには、様々なタイプの障害の検出を可能にする対応パッケージのリストも含まれています。

25.2.4. ABRT サービスの起動

abrtd デーモンは、**abrt** ディレクトリーでファイルシステムを管理するために、**/var/spool/abrt** ユーザーを必要とします。abrt パッケージがインストールされたときに、UID と GID が 173 の **abrt** ユーザーが存在していない場合は自動的に作成されます。それ以外の場合は、**abrt** ユーザーを手動で作成できます。その場合、**abrtd** は特定の UID と GID を要求しないため、任意の UID と GID を選択できます。

abrtd デーモンは、ブート時に起動するように設定されています。以下のコマンドを使用すると、現在のステータスを確認できます。

```
~]$ systemctl is-active abrtd.service
active
```

systemctl が **inactive** または **unknown** を返した場合は、デーモンが実行していません。以下のコマンドを **root** として入力すると、現在のセッションでデーモンを開始できます。

```
~]# systemctl start abrtd.service
```

同じコマンドを使用して、関連するエラー検出サービスの開始またはステータスチェックを行うことができます。たとえば、ABRT で C または C++ のクラッシュを検出したいときは、**abrt-ccpp** サービスが実行中であることを確認します。利用可能な ABRT 検出サービスと各パッケージのリストは「ソフトウェア問題の検出」を参照してください。

カーネルパニックまたはカーネル oops が発生したときのみ開始する **abrt-vmcore** サービスと **abrt-pstoreoops** サービス以外の全 ABRT サービスは、各パッケージがインストールされていれば、システムの起動時に自動的に有効になり開始します。ABRT サービスは、10 章systemd によるサービス管理 に説明されているとおり、**systemctl** ユーティリティーを使うことで無効または有効にできます。

25.2.5. ABRT のクラッシュ検出テスト

ABRT が正常に機能することをテストするには、**kill** コマンドを使用して SEGV 信号を送信し、プロセスを終了します。たとえば、以下の方法で **sleep** プロセスを開始して、**kill** コマンドでそれを終了します。

```
~]$ sleep 100 &
[1] 2823
~]$ kill -s SIGSEGV 2823
```

ABRT は **kill** コマンドの実行直後にクラッシュを検出し、グラフィカルセッションが実行されている場合に、GUI 通知アプレットにより検出された問題がユーザーに通知されます。コマンドラインで、**abrt-cli list** コマンドを実行するか、**/var/spool/abrt/** ディレクトリーに作成されたクラッシュダンプを調べることで、クラッシュが検出されていることを確認できます。検出されたクラッシュを処理する方法は、「検出された問題の処理」 を参照してください。

25.3. ABRT の設定

ABRT では、問題のライフサイクルは events によって決定されます。以下に例を示します。

- イベント 1 - 問題データのディレクトリー作成
- イベント 2: 問題データの分析
- イベント 3 - 問題の Bugzilla 報告

問題が検出されると、ABRT はその問題を既存の問題データと比較し、同じ問題が記録されているかどうかを確認します。記録されている場合には、その既存の問題データが更新され、最新の (重複している) 問題は再度記録されません。問題が ABRT で認識されない場合は、problem-data directory (**問題データディレクトリー**) が作成されます。通常、問題データディレクトリーは、**analyzer**、**architecture**、**coredump**, **cmdline**、**executable**、**kernel**、**os_release**、**reason**, **time**、**uid** などのファイルから成ります。

backtrace などのその他のファイルは、問題の分析中に作成される場合があります。これは、使用する分析方法と設定によって異なります。これは、使用するアナライザーメソッドやその設定によって異なります。たとえば **kernel** ファイルには、クラッシュしたカーネルのバージョンが記録されます。

問題データディレクトリーが作成され、問題データが収集された後は、ABRT GUI またはコマンドラインの abrt-cli ユーティリティーを使用してその問題を処理できます。記録された問題の処理に関する ABRT ツールの詳細は、「検出された問題の処理」 を参照してください。

25.3.1. イベントの設定

ABRT のイベントは、plugins を使用して実際のレポーティング操作を行います。プラグインは、問題データディレクトリーの内容を処理するイベント呼び出しを行うコンパクトなユーティリティーです。プラグインを使用することで、ABRT は問題を様々な宛先に報告できますが、多くの場合は、報告先について何らかの設定が必要になります。たとえば、Bugzilla では、ユーザー名とパスワード、および Bugzilla サービスのインスタンスを指す URL が必要になります。

設定な詳細の中にはデフォルト値を設定できるもの (Bugzilla URL など) もありますが、設定できないもの (ユーザー名など) もあります。ABRT は、このような設定を、システムワイドの設定の場合は **/etc/libreport/events/** ディレクトリー、ユーザー固有の場合は **$HOME/.cache/abrt/events/** ディレクトリーの **report_Bugzilla.conf** でこの設定を探します。設定ファイルには、ディレクティブと値のペアが含まれています。

このファイルは、イベントの実行と問題データディレクトリーの処理のために最小限必要なものです。**gnome-abrt** ツールと **abrt-cli** ツールはこれらのファイルから設定データを読み取り、実行するイベントにこれを渡します。

イベントに関する追加情報 (環境変数としてイベントに渡すことができるパラメーターの説明、名前、タイプ、その他の属性など) は、**/usr/share/libreport/events/** ディレクトリーの event_name**.xml** ファ

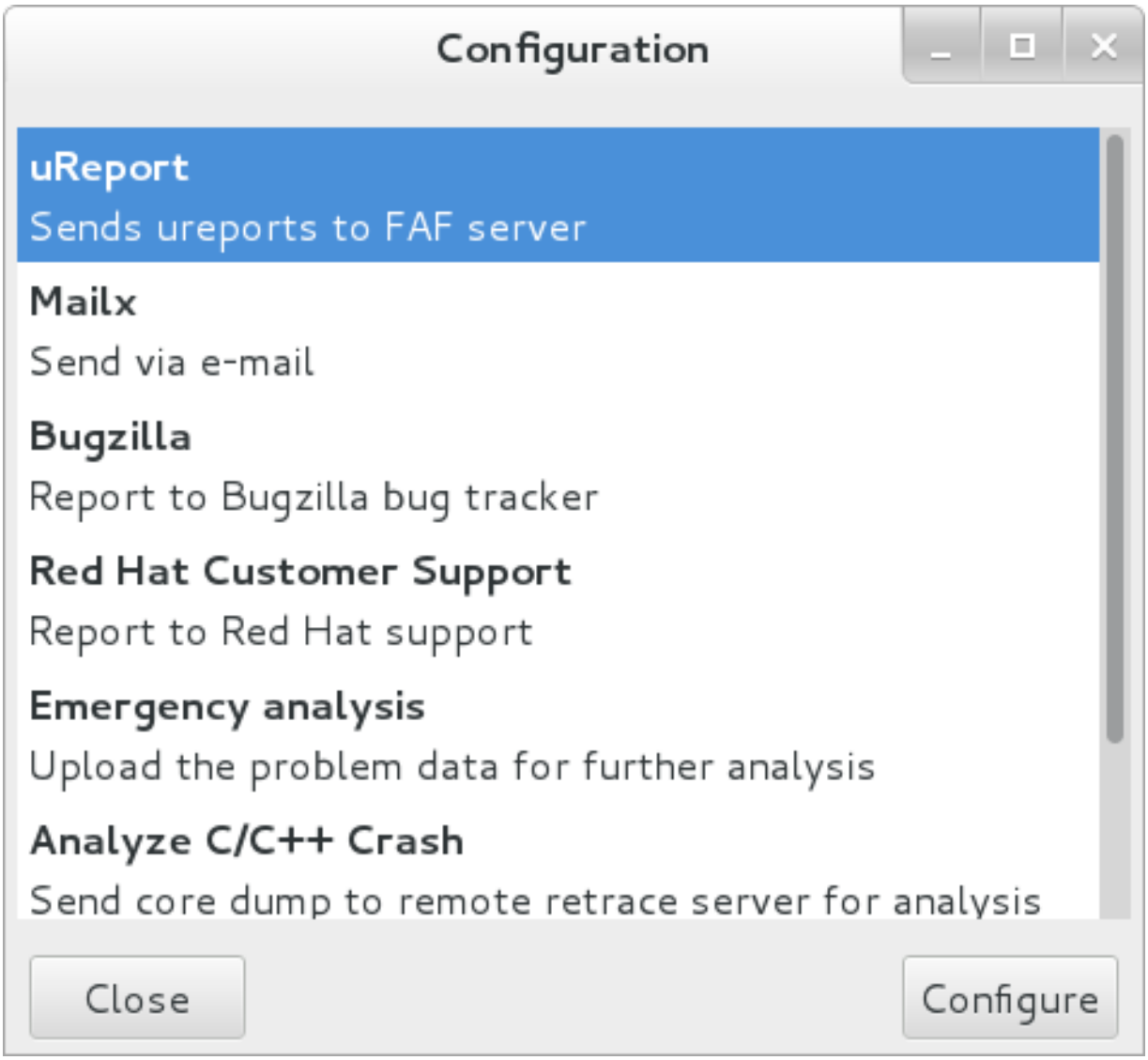
イルに保存されています。このファイルは、ユーザーインターフェイスをより使いやすくするために ***gnome-abrt*** *および* ***abrt-cli*** *で使用されます。標準インストールを変更する場合以外に、このファイルは編集しないでください。編集する場合は、対象ファイルを* ***/etc/libreport/events/*** *ディレクトリーにファイルをコピーして、コピーしたファイルを修正してください。これらのファイルには、以下の情報が含まれています。*

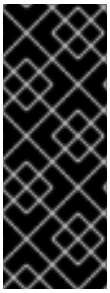
- *ユーザーフレンドリーのイベント名および説明 (****Bugzilla****、****Bugzilla*** *バグトラッカーへのレポート)*
- *イベントが正常に実行されるために必要な問題データディレクトリー内のアイテムリスト*
- *送信するおよび送信しないデフォルトおよび必須の選択アイテム*
- ***GUI*** *がデータ見直しを要求するかどうか*
- *存在する設定オプション、そのタイプ (文字列、ブール値など)、デフォルト値、プロンプト文字列など。これらにより、****GUI*** *が適切な設定ダイアログを構築できます。*

*たとえば、****report_Logger*** *イベントは、出力ファイル名をパラメーターとして受け入れます。それぞれの* **event_name*****.xml*** *ファイルを使用することで、****ABRT GUI*** *は、選択されたイベントに対してどのパラメーターを指定できるかを判断し、ユーザーはそれらのパラメーターの値を設定できるようになります。設定された値は、****ABRT GUI*** *で保存され、イベントが後で呼び出される際に再利用されます。****ABRT GUI*** *は****GNOME Keyring*** *ツールを使用して設定オプションを保存し、これらをイベントに渡すことでテキスト設定ファイルからデータを上書きすることに注意してください。*

グラフィカルの ***Configuration*** *ウィンドウを開くには、実行中の****gnome-abrt*** *アプリケーションのインスタンスから* ***Automatic Bug Reporting Tool*** *→ → → ****Preferences*** *→ を選択します。このウィンドウでは、****GUI*** *使用したレポーティングのプロセス時に選択可能な全イベントが表示されます。設定可能なイベントを選択すると、****Configure*** *ボタンがクリックできる状態となり、そのイベントの設定値を修正できます。*

図25.1 ABRT イベントの設定





重要

*/**etc/libreport**/ ディレクトリー階層内のファイルはすべて、全ユーザーが読み取り可能となっており、グローバル設定として使用するためにあります。このため、ユーザー名、パスワード、その他の機密データは、これらのファイル内に保存しないことが推奨されます。ユーザー別の設定 (GUI アプリケーションで設定され、**$HOME** の所有者のみが読み取り可能) は、GNOME Keyring に安全に保管されます。または、**abrt-cli** で使用するには、**$HOME/.abrt**/ 内のテキスト設定ファイルに保管することもできます。*

*以下の表では、ABRT の標準インストールで提供される、デフォルトの分析、収集、レポートに関するイベントの一部を表示しています。ここでは、**/etc/libreport/events.d**/ ディレクトリーからの各イベントの名前、識別子、設定ファイルと簡単な説明をリスト表示しています。設定ファイルはイベント識別子を使用しますが、ABRT GUI では個別のイベント名が望ましいことに注意してください。また、GUI ではイベントの一部の設定は可能ではないことにも注意してください。カスタムイベントの定義方法に関する情報は、「カスタムイベントの作成」 を参照してください。*

表25.1 標準 ABRT イベント

名前	識別子および設定ファイル	詳細
uReport	report_uReport	µReport を **FAF** サーバーにアップロードします。
mailx	report_Mailx **mailx_event.conf**	問題レポートを **Mailx** ユーティリティーを介して指定のメールアドレスに送信します。
Bugzilla	report_Bugzilla **bugzilla_event.conf**	問題を **Bugzilla** バグトラッカーの指定インストールにレポートします。
Red Hat Customer Support	report_RHTSupport **rhtsupport_event.conf**	問題を Red Hat テクニカルサポートシステムに報告します。
Analyze C/C++ Crash	analyze_CCpp **ccpp_event.conf**	コアダンプをリモートの retrace サーバーに送信して分析します。または、リモート分析が失敗した場合は、ローカルの分析を実行します。
Report uploader	report_Uploader **uploader_event.conf**	**FTP** または **SCP** プロトコルを使用して、問題データの tarball (**.tar.gz**) アーカイブを、選択した宛先にアップロードします。
VM コアの分析	analyze_VMcore **vmcore_event.conf**	カーネル oops の問題データに対して **GDB** (GNU Debugger) を実行し、カーネルの **backtrace** を生成します。
Local GNU Debugger	analyze_LocalGDB **ccpp_event.conf**	アプリケーションの問題データに対して **GDB** (GNU Debugger) を実行し、問題の **backtrace** を生成します。
Collect .xsession-errors	analyze_xsession_errors **ccpp_event.conf**	**~/.xsession-errors** ファイルから関連性のある行を問題レポートに保存します。
ロガー	report_Logger **print_event.conf**	問題レポートを作成して、指定のローカルファイルに保存します。
Kerneloops.org	report_Kerneloops **koops_event.conf**	カーネルの問題を kerneloops.org の oops トラッカーに送信します。

25.3.2. カスタムイベントの作成

各イベントは、それぞれの設定ファイル内の単一のルール構造により定義されます。これらの設定ファイルは、通常 **/etc/libreport/events.d/** ディレクトリーに格納されます。これらの設定ファイルは、メインの設定ファイル **/etc/libreport/report_event.conf** によりロードされます。**abrt** は **/etc/libreport/events.d/** に含まれているスクリプトを実行するため、デフォルトの設定ファイルは編集する必要がありません。このファイルはシェルのメタ文字 (*****、**$**、**?** など) を受け入れ、その位置に対する相対パスを解釈します。

各 **ルール** は、空白でない先頭文字の行で始まり、その後のすべての行は 空白 文字または タブ 文字で始まるすべての後続の行は、このルールの一部と見なされます。各**ルール**は、**条件** 部分と**プログラム** 部分の 2 つの部分で設定されています。条件パートには、以下のいずれかの形式で条件が記載されます。

- VAR=VAL
- VAR!=VAL
- VAL~=REGEX

詳細は以下のようになります。

- VAR は、*EVENT* のキーワードまたは問題データディレクトリーの要素の名前です (***executable***、***package***、***hostname*** など)。
- VAL は、イベントまたは問題データの要素名です。
- REGEX は正規表現です。

プログラムパートは、プログラム名とシェルが解釈可能なコードで設定されます。条件パートにある全条件が有効な場合、プログラムパートがシェルで実行されます。以下は**イベント** の一例です。

```
EVENT=post-create date > /tmp/dt
    echo $HOSTNAME uname -r
```

このイベントは、現在の日付と時刻で **/tmp/dt** ファイルの内容を上書きし、マシンのホスト名とカーネルバージョンを標準出力に表示します。

より複雑なイベント (実際の事前設定済みイベントの一例) の例は以下のようになります。このイベントは、**abrt-ccpp** サービスを使用して処理された問題の問題レポートに **~/.xsession-errors** ファイルの関連する行を保存します。クラッシュ発生時点で、クラッシュしたアプリケーションに **X11** ライブラリーが読み込まれていることが条件になります。

```
EVENT=analyze_xsession_errors analyzer=CCpp dso_list~=./libX11.
    test -f ~/.xsession-errors || { echo "No ~/.xsession-errors"; exit 1; }
    test -r ~/.xsession-errors || { echo "Can't read ~/.xsession-errors"; exit 1; }
    executable=cat executable &&
    base_executable=${executable##*/} &&
    grep -F -e "$base_executable" ~/.xsession-errors | tail -999 >xsession_errors &&
    echo "Element 'xsession_errors' saved"
```

使用される可能性のあるイベントのセットは限定されていません。システム管理者は、必要に応じてイベントを **/etc/libreport/events.d/** ディレクトリーに追加できます。

現在、*ABRT* と *libreport* の標準インストールでは、以下のイベント名が提供されています。

post-create

*このイベントは、**abrtd** により、新規に作成された問題データディレクトリーを処理するために実行されます。**post-create** イベントが実行すると、**abrtd** は、新規の問題データが既存の問題ディレクトリーと一致するかどうかを確認します。一致する問題ディレクトリーがある場合には、それが更新され、新規の問題データは破棄されます。**post-create** の定義内にスクリプトがゼロ以外の値で存在する場合、**abrtd** はプロセスを終了し、問題データが破棄されることに注意してください。*

notify**、**notify-dup

***notify** イベントは、**post-create** の完了後に実行されます。このイベントが実行すると、問題に注意する必要があることをユーザーが確認できます。**notify-dup** も同様ですが、これは同一問題が重複して発生した場合に使用されます。*

***analyze**_name_suffix*

*ここでの name_suffix は、イベント名の置き換え可能な部分です。このイベントは、収集データの処理に使用されます。たとえば、**analyze_LocalGDB** は GNU Debugger (GDB) ユーティリティー使用してアプリケーションのコアダンプを処理し、クラッシュのバックトレースを生成します。*

***collect**_name_suffix*

> ***{blank}***

ここでの name_suffix は、イベント名の調整可能な部分です。このイベントは、問題に関する追加情報を収集するために使用されます。

***report**_name_suffix*

> ***{blank}***

ここでの name_suffix は、イベント名の調整可能な部分です。このイベントは、問題を報告するために使用されます。

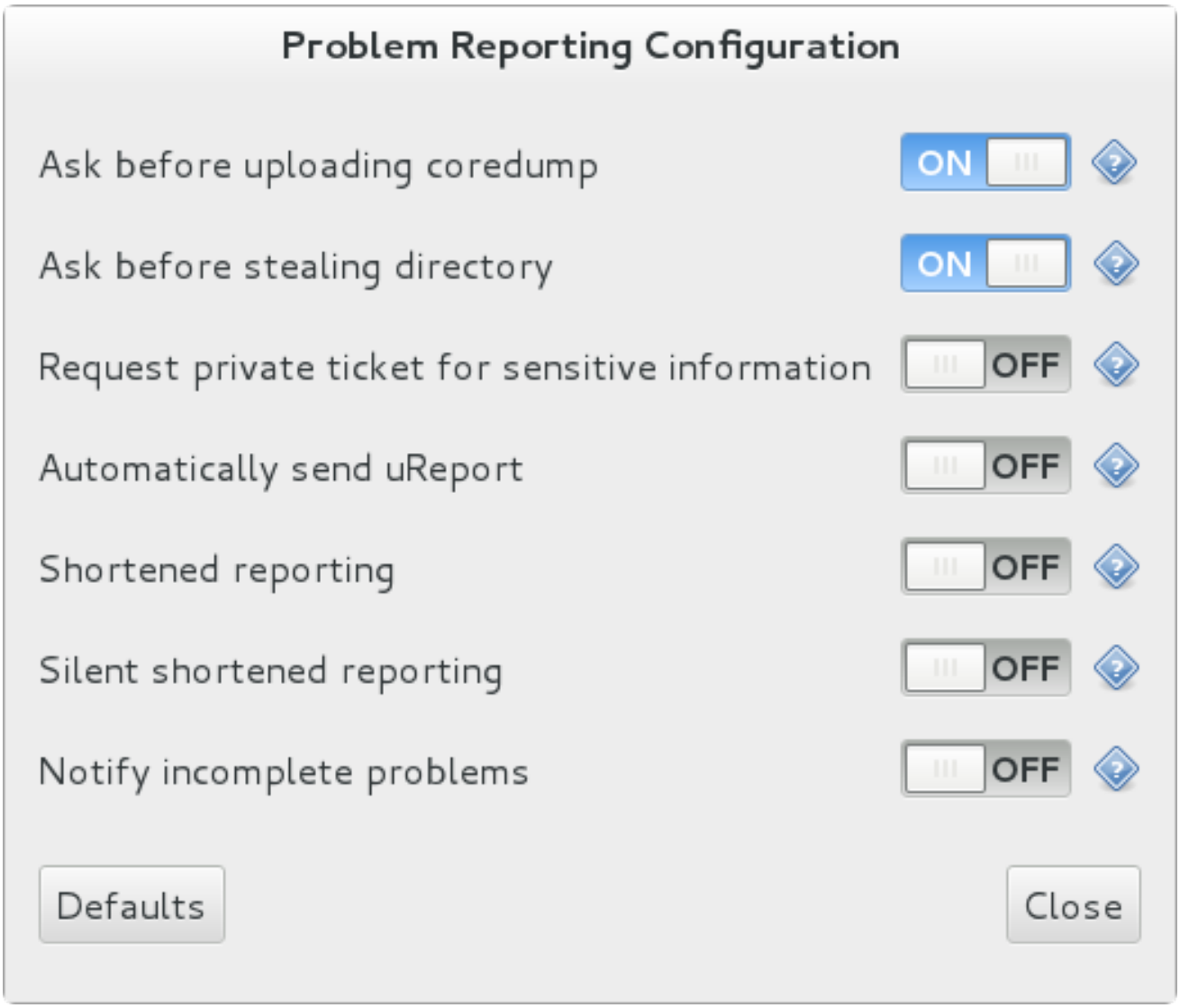
25.3.3. 自動レポーティングの設定

*ABRT は、検出されたすべての問題またはクラッシュの最初の匿名レポート (µReports とも呼ぶ) をユーザーの対話なしに自動送信するように設定できます。自動レポーティングがオンになっていると、クラッシュ検出の直後に、通常クラッシュレポーティングプロセスの最初に送信される **µReport** が送信されます。これにより、同一クラッシュについてのサポートケースの重複を効果的に防ぐことができます。自動レポーティング機能を有効にするには、**root** で以下のコマンドを発行します。*

```
~]# abrt-auto-reporting enabled
```

*上記のコマンドは、**/etc/abrt/abrt.conf** 設定ファイルの **AutoreportingEnabled** ディレクティブを **yes** に設定します。このシステムワイドの設定は、システムの全ユーザーに適用されます。このオプションを有効にすると、グラフィカルデスクトップ環境でも自動レポーティングが有効になることに注意してください。ABRT GUI の自動レポーティングのみを有効にするには、**Problem Reporting Configuration** ウィンドウ内で **Automatically send uReport** オプションを **YES** に切り替えてください。このウィンドウを開くには、gnome-abrt アプリケーションの実行中のインスタンス内で Automatic Bug Reporting Tool →→ → ABRT Configuration → を選択します。アプリケーションを起動するには、Applications →→ → Sundry →→ **Automatic Bug Reporting Tool** → に移動します。*

図25.2 ABRT 問題報告ツールの設定



デフォルトでは、ABRT はクラッシュを検出すると、問題の基本情報を含む **μReport** を Red Hat の ABRT サーバーに提出します。サーバーは問題が既知のものかどうかを判断し、既知の場合はそのケースの URL と問題の簡単な説明を提供します。既知でない場合は、ユーザーに報告するように促します。

注記

μReport (マイクロレポート) は、バイナリークラッシュやカーネル oops などの問題を表す JSON オブジェクトです。これらのレポートは、簡潔で、マシンが読み取ることができ、完全に匿名になるように設計されています。これが自動レポートに使用できる理由です。**μReports** を使用すると、バグの発生を追跡できます。ただし、通常はエンジニアがバグを修正するのに十分な情報を提供しません。サポートケースを作成するには、詳細なバグレポートが必要になります。

自動レポーティング機能の動作について、**μReport** の送信から別の動作に変更するには、**/etc/abrt/abrt.conf** 設定ファイルの **AutoreportingEvent** の値を別の ABRT 胃ベンドに変更します。標準イベントの概要は 表25.1「標準 ABRT イベント」 を参照してください。

25.4. ソフトウェア問題の検出

ABRT は、多くの異なるプログラミング言語で書かれたアプリケーションにおけるクラッシュを検出、

分析、プロセスできます。様々なタイプのクラッシュ検出のサポートを含むパッケージの多くは、ABRT パッケージ (abrt-desktop、abrt-cli) がインストールされる際に自動的にインストールされます。ABRT のインストール方法は「ABRT のインストールとサービスの起動」を参照してください。以下の表は、サポート対象のクラッシュタイプと対応パッケージをリスト表示したものです。

表25.2 サポート対象のプログラミング言語およびソフトウェアプロジェクト

言語/プロジェクト	パッケージ
C または C++	abrt-addon-ccpp
Python	abrt-addon-python
ruby	rubygem-abrt
Java	abrt-java-connector
X.Org	abrt-addon-xorg
Linux (カーネル oops)	abrt-addon-kerneloops
Linux (カーネルパニック)	abrt-addon-vmcore
Linux (永続的なストレージ)	abrt-addon-pstoreoops

25.4.1. C および C++ クラッシュの検出

abrt-ccpp サービスは独自のコアダンプハンドラーをインストールします。これにより、起動時にカーネルの **core_pattern** 変数のデフォルト値が上書きされ、C および C++ のクラッシュが **abrtd** によって処理されます。**abrt-ccpp** サービスを停止すると、指定さてある **core_pattern** の値が回復します。

デフォルトでは、**/proc/sys/kernel/core_pattern** ファイルは **core** 文字列を格納しています。これは、カーネルが、クラッシュしたプロセスの現在のディレクトリーにおける **core.** 接頭辞のあるファイルを生成することを意味しています。**abrt-ccpp** サービスは **core_pattern** ファイルを上書きして、以下のコマンドを格納します。

```
|/usr/libexec/abrt-hook-ccpp %s %c %p %u %g %t e
```

このコマンドは、カーネルがコアダンプを **abrt-hook-ccpp** プログラムにパイプ処理するように指示します。これにより、ABRT のダンプの場所に格納され、新しいクラッシュについて **abrtd** デーモンに通知します。また、**/proc/PID/** ディレクトリー (PID はクラッシュしたプロセスの ID の両方) から以下のファイルをデバッグ目的で格納します。**maps**、**limits**、**cgroup**、**status**。これらのファイルのフォーマットおよび意味は、proc(5) を参照してください。

25.4.2. Python 例外の検出

abrt-addon-python パッケージは、Python アプリケーション用のカスタム例外ハンドラーをインストールします。次に Python インタープリターは、**/usr/lib64/python2.7/site-packages/** にインストール済みの **abrt.pth** ファイルを自動的にインポートします。これにより、**abrt_exception_handler.py** が

インポートされます。これにより、Python のデフォルトの **sys.excepthook** とカスタムハンドラーによる sys.excepthook が上書きされ、処理されていない例外がそのソケット API 経由で **abrtd** に転送されます。

サイト固有のモジュールの自動インポートを無効にして、Python アプリケーション実行時に ABRT カスタム例外ハンドラーが使用されないようにするには、Python インタープリターに **-S** オプションを渡します。

```
~]$ python -S file.py
```

このコマンドでは、file.py を、サイト固有のモジュールを使用せずに実行する Python スクリプトの名前で置き換えます。

25.4.3. Ruby 例外の検出

rubygem-abrt パッケージは、プログラムが終了したときに熟考される、**at_exit** 機能を使用するカスタムハンドラーを登録します。これは、プログラムの終了時に実行します。処理されていない例外が捕捉されるたびに、ABRT ハンドラーはバグレポートを作成します。これは標準の ABRT ツールを使用して Red Hat Bugzilla に提出することができます。

25.4.4. Java 例外の検出

ABRT Java コネクターは、捕捉されなかった Java 例外を **abrtd** にレポートする JVM エージェントです。エージェントは複数の JVMTI イベントコールバックを登録し、**-agentlib** コマンドラインパラメーターを指定して JVM にロードする必要があります。コマンドラインパラメーターを使用して JVM に読み込む必要があります。ABRT が Java クラスから例外を取得できるようにするには、以下のコマンドを使用します。

```
~]$ java -agentlib:abrt-java-connector=abrt=on $MyClass -platform.jvmtiSupported true
```

ここでは、$MyClass を、テストする Java クラス名に置き換えます。**abrt=on** オプションをコネクターに渡すと、例外が **abrtd** によって処理されるようになります。コネクターが標準出力の例外を出力する場合は、このオプションを省略します。

25.4.5. X.Org クラッシュの検出

abrt-xorg は、X.Org server のクラッシュ情報を **/var/log/Xorg.0.log** ファイルから収集して処理します。ブラックリスト化された X.org モジュールが読み込まれている場合は、レポートが生成されないことに注意してください。その代わりに該当する説明の付いた **not-reportable** ファイルが問題データディレクトリーに作成されます。問題となっているモジュールリストは、**/etc/abrt/plugins/xorg.conf** ファイルで確認できます。デフォルトでは、商用のグラフィックドライバーのモジュールのみがブラックリスト化されています。

25.4.6. カーネル Oops およびカーネルパニックの検出

カーネルログの出力をチェックすることで、ABRT は致命的ではない Linux カーネルの正常な動作からの逸脱であるカーネル oops を見つけ、これを処理できます。この機能は、**abrt-oops** サービスが提供します。

ABRT は、**abrt-vmcore** サービスを使用して、致命的で再起動を必要とする回復不可能なエラーであるカーネルパニックも検出して処理できます。このサービスは、**vmcore** ファイル (カーネルコアダンプ) が **/var/crash/** ディレクトリーに表示される場合にのみ起動します。コアダンプファイルが見つかる

と、**abrt-vmcore** が新たな**problem-data directory** を**/var/spool/abrt/** ディレクトリー内に作成し、作成されたディレクトリーにコアダンプファイルを移動します。**/var/crash/** ディレクトリーを検索すると、このサービスは停止します。

ABRT がカーネルパニックを検出できるようにするには、システムで**kdump** サービスが有効になっている必要があります。kdump カーネル用に確保されたメモリー容量が正しく設定されている必要があります。これは、system-config-kdump グラフィカルツールを使用して設定することも、GRUB 2 メニューにあるカーネルオプションのリストに**crashkernel** パラメーターを指定して設定できます。**kdump** の有効化と設定の詳細は、*Red Hat Enterprise Linux 7 カーネルクラッシュダンプガイド* を参照してください。GRUB 2 メニューの変更方法は *26章GRUB 2 での作業* を参照してください。

ABRT は、**abrt-pstoreoops** サービスを使用して、pstore に対応するシステム上で、自動的にマウントされた**/sys/fs/pstore/** ディレクトリーに保存されているカーネルパニックの情報を収集および処理できます。これは、pstore 対応のシステムでは、自動マウントされた**/sys/fs/pstore/** ディレクトリーに保存されます。そのため、カーネルパニック情報を保持することができます。このサービスは、**/sys/fs/pstore/** ディレクトリーにカーネルのクラッシュダンプファイルが現れると、自動的に起動します。

25.5. 検出された問題の処理

abrtd で保存された問題データは、コマンドラインツール**abrt-cli** またはグラフィカルツール**gnome-abrt** を使用して表示し、報告し、削除することができます。

注記

ABRT は、新たな問題をローカルに保存されている全問題と比較することにより、問題の重複を特定することに注意してください。繰り返し発生しているクラッシュの場合、ABRT が対応を要求するのは1回のみです。ただし、この問題のクラッシュダンプを削除すると、次にこの問題が発生すると、ABRT はこれを新しいクラッシュとして扱います。ABRT は、ユーザーに警告を受け、説明を入力し、報告します。ABRT が繰り返し発生する問題の通知をしないようにするには、その問題データを削除しないようにしてください。

25.5.1. コマンドラインツールの使用

コマンドライン環境では、abrt-console-notification パッケージがインストール済みであれば、ログイン時に新たなクラッシュがユーザーに通知されます。コンソールでの通知は、以下のようになります。

```
ABRT has detected 1 problem(s). For more info run: abrt-cli list --since 1398783164
```

検出された問題を表示するには、**abrt-cli list** コマンドを入力します。

```
~]$ abrt-cli list
id 6734c6f1a1ed169500a7bfc8bd62aabaf039f9aa
Directory:   /var/tmp/abrt/ccpp-2014-04-21-09:47:51-3430
count:     1
executable:   /usr/bin/sleep
package:    coreutils-8.22-11.el7
time:      Mon 21 Apr 2014 09:47:51 AM EDT
uid:      1000
Run 'abrt-cli report /var/tmp/abrt/ccpp-2014-04-21-09:47:51-3430' for creating a case in Red Hat Customer Portal
```

abrt-cli list コマンドの出力でリスト表示されるクラッシュにはそれぞれ、一意の識別子と **abrt-cli** を使用した追加の操作に使用できるディレクトリーがあります。

特定の問題に関する情報のみを表示するには、**abrt-cli info** コマンドを使用します。

```
abrt-cli info -d directory_or_id
```

list および **info** の各サブコマンドの使用時に表示する情報量を増やすには、**-d (--detailed)** オプションを渡します。これにより、すでに生成されていれば、関連の **backtrace** ファイルを含むリストの問題についての格納されているすべての情報が表示されます。

特定の問題を分析して報告するには、**abrt-cli report** コマンドを使用します。

```
abrt-cli report directory_or_id
```

上記のコマンドを実行すると、Red Hat カスタマーサポートでサポートケースを作成するために必要なログイン情報の提供を求められます。次に、**abrt-cli** がレポートの内容を含むテキストエディターを開きます。これでレポート内容を確認でき、クラッシュを再現させるための指示やコメントを記入できます。さらに、バックトレースも確認するようにしてください。バックトレースは、問題報告イベントの設定によっては公開サーバーに送信され、誰でも見ることができる場合があります。

注記

レポートの確認に使用するテキストエディターを選択できます。**abrt-cli** は、**ABRT_EDITOR** 環境変数で定義されたエディターを使用します。変数が定義されていない場合は、**VISUAL** と **EDITOR** 変数が確認されます。いずれの変数も設定されていない場合は、**vi** エディターが使用されます。**.bashrc** 設定ファイルで優先エディターを設定することもできます。たとえば、GNU Emacs を使用する場合は、このファイルに以下の行を追加します。

```
export VISUAL=emacs
```

レポートでの作業が終了したら、変更を保存してエディターを終了します。問題を Red Hat カスタマーサポートのデータベースに報告した場合には、問題のケースがデータベースに作成されます。この時点から、報告プロセスで指定したメールに問題の解決の進捗状況が通知されるようになります。また、問題のケース作成時に提供された URL や Red Hat サポートからのメールを使用して問題ケースをモニタリングすることもできます。

特定の問題を報告したくないことが分かっている場合は、その問題を削除できます。問題を削除して、その情報が ABRT に保存されないようにするには、以下のコマンドを使用します。

```
abrt-cli rm directory_or_id
```

特定の **abrt-cli** コマンドに関するヘルプを表示するには、**-help** オプションを指定します。

```
abrt-cli command --help
```

25.5.2. GUI の使用

ABRT デーモンは、問題レポートが作成されると常に **D-Bus** メッセージをブロードキャストします。グラフィカルデスクトップ環境で ABRT アプレットが実行中の場合は、アプレットがそのメッセージを取得し、デスクトップに通知ダイアログを表示します。このダイアログの **Report** ボタンをクリックすれ

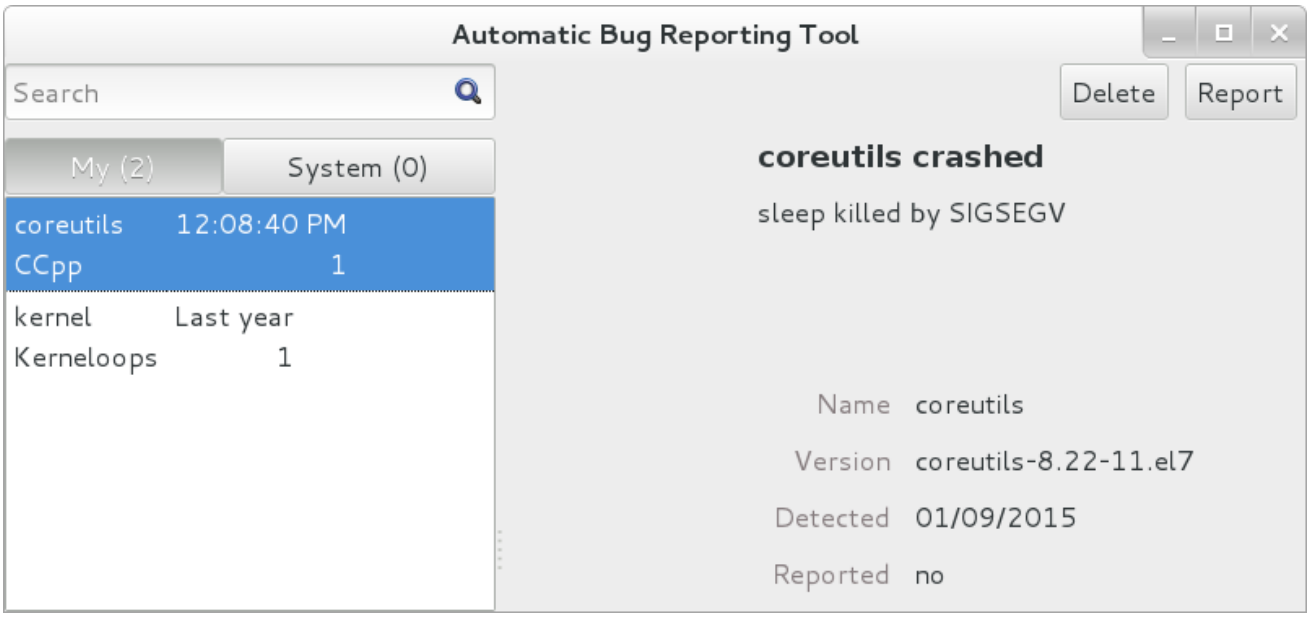
ば、ABRT GUI を開くことができます。また、Applications → → → Sundry → → Automatic Bug Reporting Tool → のメニューアイテムを選択して ABRT GUI を開くこともできます。

別の方法では、以下のようにコマンドラインから ABRT GUI を実行することもできます。

```
~]$ gnome-abrt &
```

ABRT GUI ウィンドウは、検出された問題のリストを表示します。各問題エントリーは、障害が発生したアプリケーション名、アプリケーションがクラッシュした理由、その問題が最後に発生した日付で設定されます。

図25.3 ABRT GUI



*問題の詳細な説明にアクセスするには、問題レポートの行をダブルクリックするか、各問題の行を選択し、**Report** ボタンをクリックします。その後、指示に従い、問題の記述手順、分析の方法、およびその報告先を判断します。問題を破棄するには、**Delete** ボタンをクリックします。*

25.6. 関連情報

ABRT および関連トピックについての詳細情報は、以下に挙げるリソースを参照してください。

インストールされているドキュメント

- *abrtd(8): **abrtd** デーモンの man ページは、このデーモンと使用可能なオプションについての情報を提供します。*
- *abrt_event.conf(5): **abrt_event.conf** 設定ファイルの man ページは、ディレクティブとルールのフォーマットを説明し、**XML** ファイル内のイベントメタデータ設定についての参照情報を提供します。*

オンラインドキュメント

- *Red Hat Enterprise Linux 7 ネットワークガイド: Red Hat Enterprise Linux 7 の **ネットワークガイド** では、システム上のネットワークインターフェイスおよびネットワークサービスの設定および管理に関する情報が説明されています。*
- *Red Hat Enterprise Linux 7 カーネルクラッシュダンプガイド: Red Hat Enterprise Linux 7 の **K カーネルクラッシュガイド** は、**kdump** クラッシュ回復サービスの設定し、テストし、使用する*

方法を説明します。また、それによって生じるコアダンプを crash デバッグユーティリティーで分析する方法の概要を説明します。

関連項目

- *23章ログファイルの表示と管理 では、**rsyslog** デーモンと systemd ジャーナルの設定、およびシステムログの検索、表示、監視方法について説明しています。*
- *9章Yum では、Yum パッケージマネージャーを使用して、コマンドラインでのパッケージの検索、インストール、更新、アンインストール方法について説明しています。*
- *10章systemd によるサービス管理 では、**systemd** の概説と、**systemctl** コマンドを使用したシステムサービスの管理方法、systemd ターゲットの設定方法、および電源管理コマンドの実行方法について説明しています。*

パート VII. ブートローダーを使用したカーネルのカスタマイズ

ここでは、管理者によるカーネルのカスタマイズを支援する GRUB 2 ブートローダーの使用方法について説明します。

第26章 GRUB 2 での作業

Red Hat Enterprise Linux 7 は、GNU GRand Unified Bootloader (GRUB 2) のバージョン 2 とともに配布され、ユーザーはシステム起動時にオペレーティングシステムまたはカーネルを選択できます。また GRUB 2 により、ユーザーはカーネルに引数を渡すことができます。

26.1. GRUB 2 について

GRUB 2 は、従来の BIOS ベースのマシンの場合は **/boot/grub2/grub.cfg** ファイルから、UEFI マシンの場合は **/boot/efi/EFI/redhat/grub.cfg** ファイルから設定を読み取ります。このファイルにはメニュー情報が含まれています。

GRUB 2 の設定ファイルである **grub.cfg** はインストール中に生成されるか、/usr/sbin/grub2-mkconfig ユーティリティーを呼び出すことによって生成され、新しいカーネルがインストールされるたびに **grubby** によって自動的に更新されます。grub2-mkconfig を使用して手動で再生成させると、ファイルは **/etc/grub.d/** にあるテンプレートファイルと **/etc/default/grub** ファイル内のカスタム設定に従って生成されます。**grub.cfg** の編集内容は、grub2-mkconfig を使用してファイルを再生成するときに失われるため、手動による変更は **/etc/default/grub** にも反映するようにする必要があります。

grub.cfg での通常の操作 (カーネルの削除や追加など)、**grubby** ツールと **new-kernel-pkg** ツール (スクリプト用) を使用して行う必要があります。**grubby** を使用してデフォルトのカーネルを変更する場合、変更内容は新しいカーネルがインストールされたときに継承されます。**grubby** の詳細は、「grubby ツールを使用した GRUB 2 メニューの永続的な変更」 を参照してください。

/etc/default/grub ファイルは、インストールプロセスで **grub.cfg** を作成するときに **anaconda** によって使用される **grub2-mkconfig** ツールにより使用され、システム障害の発生時に使用できます (たとえば、ブートローダーの設定を再作成する必要がある場合)。一般的に、他の手段がない場合を除き、**grub2-mkconfig** を手動で実行して **grub.cfg** ファイルを置き換えることは推奨されません。**/etc/default/grub** への手動による変更を反映するには、**grub.cfg** ファイルを再構築する必要があります。

grub.cfg のメニューエントリー

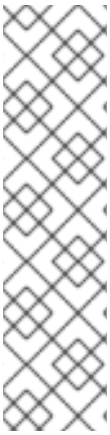
grub.cfg 設定ファイルには、多くのコードスニペットやディレクティブに加えて、1 つまたは複数の **menuentry** ブロックが含まれています。それぞれは、単一の GRUB 2 ブートメニューエントリーを示しています。これらのブロックは、常に **menuentry** キーワードで始まり、タイトル、オプションのリスト、および開きの波括弧で始まり、最後は閉じの波括弧で終わります。これらのブロックは、常に menuentry キーワードで始まり、それにタイトル、オプションリスト、および中括弧が続きます。以下は、Linux カーネル 3.8.0-0.40.el7.x86_64 を使用する Red Hat Enterprise Linux 7 のサンプル **menuentry** ブロックの例です。

```
menuentry 'Red Hat Enterprise Linux Server' --class red --class gnu-linux --class gnu --class
os $menuentry_id_option 'gnulinux-simple-c60731dc-9046-4000-9182-64bdcce08616' {
    load_video
    set gfxpayload=keep
    insmod gzio
    insmod part_msdos
    insmod xfs
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
      search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --
hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1' 19d9e294-65f8-4e37-8e73-d41d6daa6e58
    else
      search --no-floppy --fs-uuid --set=root 19d9e294-65f8-4e37-8e73-d41d6daa6e58
    fi
    echo  'Loading Linux 3.8.0-0.40.el7.x86_64 ...'
```

```
    linux16  /vmlinuz-3.8.0-0.40.el7.x86_64 root=/dev/mapper/rhel-root ro rd.md=0 rd.dm=0
rd.lvm.lv=rhel/swap crashkernel=auto rd.luks=0 vconsole.keymap=us rd.lvm.lv=rhel/root rhgb
quiet
    echo  'Loading initial ramdisk ...'
    initrd /initramfs-3.8.0-0.40.el7.x86_64.img
}
```

*インストール済み Linux カーネルを表す各 **menuentry** ブロックには、**linux** (64 ビット IBM POWER シリーズ用)、 **linux16** (x86_64 BIOS ベースのシステム用)、および**linuxefi** (UEFI ベースのシステム用) が含まれます。**initrd** ディレクティブの後には、それぞれカーネルへのパスと **initramfs** イメージへのパスが指定されます。個別の **/boot** パーティションを作成している場合は、カーネルと **initramfs** イメージへのパスは、**/boot** に対して相対的なものになります。上記の例では、**initrd /initramfs-3.8.0-0.40.el7.x86_64.img** の行は、**root** ファイルシステムのマウント時に **initramfs** イメージが実際には **/boot/initramfs-3.8.0-0.40.el7.x86_64.img** にあり、カーネルパスも同様であることを意味します。*

*各 **menuentry** ブロックの **linux16 /vmlinuz-kernel_version** 行にあるカーネルのバージョン番号は、**initrd /initramfs-kernel_version.img** 行の **initramfs** イメージのバージョン番号に適合する必要があります。初期 RAM ディスクイメージの検証方法は、Red Hat Enterprise 7 カーネル管理ガイド の初期 RAM ディスクイメージの検証を参照してください。*



注記

***menuentry** では、**initrd** 指示文は同じカーネルバージョンに対応する **initramfs** ファイルの位置 (別のパーティションの場合、**/boot/** ディレクトリーに対して相対的) を指している必要があります。初期 RAM ディスクイメージを作成した以前のツール、**mkinitrd** が **initrd** と呼ばれるファイルを作成したために、この指示文は **initrd** と呼ばれます。**grub.conf** 指示文は、他のツールとの互換性を維持するために **initrd** のまま残されています。初期 RAM ディスクイメージを作成するための **dracut** ユーティリティーを使用したシステムのファイル命名の規則名は、**initramfs-**kernel_version**.img** となります。*

Dracut の使用に関する詳細は、Red Hat Enterprise 7 カーネル管理ガイドを参照してください。

26.2. GRUB 2 の設定

GRUB 2 メニューへの変更は、起動時に一時的に行ったり、システムの実行中に単一システムに対して永続的に行ったり、新しい GRUB 2 設定ファイルの作成の一部として行ったりできます。

- *GRUB 2 メニューに永続的でない変更を加える場合は、「GRUB 2 メニューの一時的な変更」を参照してください。*
- *実行中のシステムに永続的な変更を行うには、「grubby ツールを使用した GRUB 2 メニューの永続的な変更」 を参照してください。*
- *GRUB 2 設定ファイルの作成およびカスタマイズは 「GRUB 2 設定ファイルのカスタマイズ」を参照してください。*

26.3. GRUB 2 メニューの一時的な変更

カーネルメニューエントリーの一時的な変更

これから実行する起動プロセスでのみカーネルパラメーターを変更するには、以下の手順を実行します。

1. システムを起動し、GRUB 2 ブート画面で、編集するメニューエントリーにカーソルを移動し、編集のために **e** キーを押します。

2. カーソルを下に移動してカーネルコマンドラインを見つけます。カーネルコマンドラインは、64 ビット IBM Power シリーズの場合は **linux**、x86-64 BIOS ベースのシステムの場合は **linux16**、UEFI システムの場合は **linuxefi** で始まります。

3. カーソルを行の最後に移動します。
Ctrl+a か **Ctrl+e** を押して行の最初または最後に移動します。システムによっては、**Home** と **End** が機能するものもあります。

4. 必要に応じてカーネルパラメーターを編集します。たとえば、緊急モードでシステムを実行するには、**linux16** 行の最後に emergency パラメーターを追加します。

```
linux16   /vmlinuz-3.10.0-0.rc4.59.el7.x86_64 root=/dev/mapper/rhel-root ro rd.md=0
rd.dm=0 rd.lvm.lv=rhel/swap crashkernel=auto rd.luks=0 vconsole.keymap=us
rd.lvm.lv=rhel/root rhgb quiet emergency
```

rhgb および **quiet** パラメーターは、システムメッセージを有効化するために削除できます。

この設定は永続的なものではなく、これから実行するブートにのみ適用されます。システムでメニューエントリーの変更を永続的に行うには、**grubby** ツールを使用します。**grubby** の詳細は 「GRUB 2 メニューエントリーに対する引数の追加および削除」を参照してください。

26.4. GRUBBY ツールを使用した GRUB 2 メニューの永続的な変更

grubby ツールは、**grub.cfg** ファイルから情報を読み取ったり、そのファイルに永続的な変更を行ったりするために使用できます。たとえば、GRUB 2 メニューエントリーを変更してシステム起動時にカーネルに渡す引数を指定したり、デフォルトのカーネルを変更したりできます。

Red Hat Enterprise Linux 7 では、GRUB 2 設定ファイルを指定せずに **grubby** を手動で呼び出すと、**grub.cfg** ファイル (場所はアーキテクチャーに依存します) へのシンボリックリンクである **/etc/grub2.cfg** がデフォルトで検索されます。このファイルが見つからない場合は、アーキテクチャー依存のデフォルトのファイルが検索されます。

デフォルトのカーネルのリスト表示

デフォルトのカーネルのファイル名を見つけるには、以下のようにコマンドを入力します。

```
~]# grubby --default-kernel
/boot/vmlinuz-3.10.0-229.4.2.el7.x86_64
```

デフォルトのカーネルのインデックス番号を見つけるには、以下のようにコマンドを入力します。

```
~]# grubby --default-index
0
```

デフォルトのブートエントリーの変更

デフォルトのカーネルとして指定されるカーネルの変更を永続的に行うには、以下のように **grubby** コマンドを使用します。

```
~]# grubby --set-default /boot/vmlinuz-3.10.0-229.4.2.el7.x86_64
```

カーネルの GRUB 2 メニューエントリーの表示

すべてのカーネルメニューエントリーをリスト表示するには、以下のようにコマンドを入力します。

```
~]$ grubby --info=ALL
```

UEFI システムでは、**grubby** コマンドはすべて**root** で入力する必要があります。

特定のカーネルの GRUB 2 メニューエントリーを表示するには、以下のようにコマンドを入力します。

```
~]$ grubby --info /boot/vmlinuz-3.10.0-229.4.2.el7.x86_64
index=0
kernel=/boot/vmlinuz-3.10.0-229.4.2.el7.x86_64
args="ro rd.lvm.lv=rhel/root crashkernel=auto rd.lvm.lv=rhel/swap vconsole.font=latarcyrheb-
sun16 vconsole.keymap=us rhgb quiet LANG=en_US.UTF-8"
root=/dev/mapper/rhel-root
initrd=/boot/initramfs-3.10.0-229.4.2.el7.x86_64.img
title=Red Hat Enterprise Linux Server (3.10.0-229.4.2.el7.x86_64) 7.0 (Maipo)
```

タブ補完を試行して**/boot/** ディレクトリー内の利用可能なカーネルを確認します。

GRUB 2 メニューエントリーに対する引数の追加および削除

--update-kernel オプションを使用して、**--args** と組み合わせて新しい引数を追加し、既存の引数を削除する **--remove-arguments** を追加する場合にメニューエントリーを更新することができます。これらのオプションは、引用符で囲まれたスペースを受け入れます。GRUB 2 メニューエントリーから引数を同時に追加および削除するコマンドは、以下の形式になります。

```
grubby --remove-args="argX argY" --args="argA argB" --update-kernel /boot/kernel
```

カーネルの GRUB 2 メニューエントリーに対して引数を追加および削除するには、以下のようにコマンドを使用します。

```
~]# grubby --remove-args="rhgb quiet" --args=console=ttyS0,115200 --update-kernel
/boot/vmlinuz-3.10.0-229.4.2.el7.x86_64
```

このコマンドにより、Red Hat グラフィカルブート引数が削除され、ブートメッセージの表示が可能になり、シリアルコンソールが追加されます。コンソール引数は行の最後に追加されるため、新しいコンソールは設定された他のすべてのコンソールよりも優先されます。

変更を確認するには、次のように **--info** コマンドオプションを使用します。

```
~]# grubby --info /boot/vmlinuz-3.10.0-229.4.2.el7.x86_64
index=0
kernel=/boot/vmlinuz-3.10.0-229.4.2.el7.x86_64
args="ro rd.lvm.lv=rhel/root crashkernel=auto rd.lvm.lv=rhel/swap vconsole.font=latarcyrheb-
sun16 vconsole.keymap=us LANG=en_US.UTF-8 ttyS0,115200"
root=/dev/mapper/rhel-root
initrd=/boot/initramfs-3.10.0-229.4.2.el7.x86_64.img
title=Red Hat Enterprise Linux Server (3.10.0-229.4.2.el7.x86_64) 7.0 (Maipo)
```

同じ引数を使用したすべてのカーネルメニューの更新

すべてのカーネルメニューエントリーに同じカーネルブート引数を追加するには、以下のようにコマンドを入力します。

```
~]# grubby --update-kernel=ALL --args=console=ttyS0,115200
```

--update-kernel パラメーターには、DEFAULT またはカーネルインデックス番号のコンマ区切りリストも指定できます。

カーネル引数の変更
既存のカーネル引数の値を変更するには、引数を、必要に応じて値を変更して再び指定します。たとえば、仮想コンソールのフォントサイズを変更するには、以下のようにコマンドを使用します。

```
~]# grubby --args=vconsole.font=latarcyrheb-sun32 --update-kernel /boot/vmlinuz-3.10.0-
229.4.2.el7.x86_64
index=0
kernel=/boot/vmlinuz-3.10.0-229.4.2.el7.x86_64
args="ro rd.lvm.lv=rhel/root crashkernel=auto rd.lvm.lv=rhel/swap vconsole.font=latarcyrheb-
sun32 vconsole.keymap=us LANG=en_US.UTF-8"
root=/dev/mapper/rhel-root
initrd=/boot/initramfs-3.10.0-229.4.2.el7.x86_64.img
title=Red Hat Enterprise Linux Server (3.10.0-229.4.2.el7.x86_64) 7.0 (Maipo)
```

*その他のコマンドオプションは、**grubby(8)** man ページを参照してください。*

26.5. GRUB 2 設定ファイルのカスタマイズ

GRUB 2 スクリプトはユーザーのコンピューターを検索して、スクリプトが見つけたオペレーティングシステムに基づくブートメニューを構築します。最新のシステムブートオプションを反映させるために、カーネルが更新されるか新規カーネルが追加されると、ブートメニューは自動的に再構築されます。

しかし、特定のエントリーを含むメニューの構築や、エントリーの特定の順番をユーザーが希望する場合もあります。GRUB 2 では基本的なブートメニューのカスタマイズが可能で、画面に表示されるものをユーザーが制御できます。

*GRUB 2 は、一連のスクリプトを使用してメニューを構築します。これらは **/etc/grub.d/** ディレクトリーにあります。このスクリプトは /etc/grub.d/ ディレクトリーに格納されており、以下のファイルが含まれます。*

- ***00_header**: **/etc/default/grub** ファイルから GRUB 2 設定を読み込みます。*
- ***01_users**: **user.cfg** ファイルからスーパーユーザーのパスワードを読み取ります。Red Hat Enterprise Linux 7.0 および 7.1 では、インストール中にキックスタートファイルでブートパスワードが定義された場合にのみこのファイルが作成され、含まれるパスワードはプレーンテキストになります。*
- ***10_linux**: Red Hat Enterprise Linux のデフォルトのパーティションでカーネルを見つけます。*
- ***30_os-prober**。別のパーティションで見つかったオペレーティングシステム用にエントリーを構築します。*
- ***40_custom**: 追加のメニューエントリー作成に使用可能なテンプレートです。*

***/etc/grub.d/** ディレクトリーからのスクリプトはアルファベット順に読み取られるので、名前を変更して特定のメニューエントリーの起動順を変更できます。*

重要

*ブート可能なカーネルのリストを非表示にするには、**/etc/default/grub** で **GRUB_TIMEOUT** を 0 に設定しないでください。このように設定すると、システムが常にデフォルトのメニューエントリーで直ちに起動し、デフォルトのカーネルが起動に失敗した場合に、古いカーネルを起動することができなくなります。*

*代わりに、システムの起動時に GRUB 2 が起動可能なカーネルのリストを表示しないようにするには、以下のように **/etc/default/grub** ファイルの **GRUB_TIMEOUT_STYLE** オプションを設定します。*

```
GRUB_TIMEOUT_STYLE=hidden
```

システムの起動時にリストを表示する場合は、キーボードまたは他のシリアルコンソールを使用して BIOS 情報が表示されている間に英数字キーを押し続けると、GRUB 2 で GRUB 2 メニューが表示されます。

26.5.1. デフォルトのブートエントリーの変更

*デフォルトでは、**/etc/default/grub** ファイルの **GRUB_DEFAULT** ディレクティブのキーは **saved** という単語です。これにより、**/boot/grub2/grubenv** にある GRUB 2 環境ファイルの **saved_entry** ディレクティブで指定されたカーネルを読み込むように GRUB 2 が指示されます。**grub2-set-default** コマンドを使用して別の GRUB 2 レコードをデフォルトにすることもできます (この結果、GRUB 2 環境ファイルが更新されます)。*

*デフォルトでは、**saved_entry** 値は、パッケージタイプ kernel の最新のインストール済みカーネルの名前に設定されます。これは **UPDATEDEFAULT** ディレクティブと **DEFAULTKERNEL** ディレクティブにより **/etc/sysconfig/kernel** で定義されます。このファイルは以下のように **root** ユーザーが表示できます。*

```
~]# cat /etc/sysconfig/kernel
# UPDATEDEFAULT specifies if new-kernel-pkg should make
# new kernels the default
UPDATEDEFAULT=yes

# DEFAULTKERNEL specifies the default kernel package type
DEFAULTKERNEL=kernel
```

***DEFAULTKERNEL** ディレクティブはデフォルトで使用するパッケージタイプを指定します。**DEFAULTKERNEL** がパッケージタイプ kernel に設定されている場合は、kernel-debug タイプのパッケージをインストールしても、デフォルトのカーネルは変更されません。*

*GRUB 2 は、 **saveed_entry** ディレクティブのキーとして数値を使用して、オペレーティングシステムがロードされるデフォルトの順序を変更することをサポートしています。どのオペレーティングシステムを最初に読み込むかを指定するには、その数字を **grub2-set-default** コマンドに渡します。以下に例を示します。*

```
~]# grub2-set-default 2
```

リスト内でのメニューエントリーの場所は、ゼロで始まる数字で示されることに注意してください。したがって、上記の例では、3 番目のエントリーがロードされます。この値は、次回にインストールされるカーネルの名前で上書きされます。

システムが常に特定のメニューエントリーを使用するようにするには、**/etc/default/grub** ファイルの **GRUB_DEFAULT** ディレクティブにメニューエントリー名をキーとして使用します。利用可能なメニューエントリーをリスト表示するには、**root** で以下のコマンドを実行します。

```
~]# awk -F\' '$1=="menuentry " {print $2}' /etc/grub2.cfg
```

ファイル名 **/etc/grub2.cfg** は **grub.cfg** ファイル (場所はアーキテクチャーに依存します) へのシンボリックリンクです。信頼性を確保するために、シンボリックリンクは本章の他の例では使用されません。ファイルに書き込む場合 (特にシステムを修復する場合) は、絶対パスを使用することが推奨されます。

/etc/default/grub への変更は、以下のように **grub.cfg** ファイルの再構築を必要とします。

- BIOS ベースのマシンでは、**root** で以下のコマンド実行します。

  ```
  ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
  ```

- UEFI ベースのマシンでは、**root** で以下のコマンド実行します。

  ```
  ~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
  ```

26.5.2. メニューエントリーの編集

新しい GRUB 2 ファイルを異なるパラメーターで準備する必要がある場合は、**/etc/default/grub** ファイルの **GRUB_CMDLINE_LINUX** キーの値を編集します。GRUB 2 ブートメニューに複数のパラメーターを追加する際と同様に、**GRUB_CMDLINE_LINUX** キーには複数のパラメーターを指定できることに注意してください。以下に例を示します。

```
GRUB_CMDLINE_LINUX="console=tty0 console=ttyS0,9600n8"
```

console=tty0 は最初の仮想ターミナルで、**console=ttyS0** は使用するシリアルターミナルです。

/etc/default/grub への変更は、以下のように **grub.cfg** ファイルの再構築を必要とします。

- BIOS ベースのマシンでは、**root** で以下のコマンド実行します。

  ```
  ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
  ```

- UEFI ベースのマシンでは、**root** で以下のコマンド実行します。

  ```
  ~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
  ```

26.5.3. 新規エントリーの追加

grub2-mkconfig コマンドを実行すると、GRUB 2 は **/etc/grub.d/** ディレクトリーにあるファイルに基づいて Linux カーネルと他のオペレーティングシステムを探します。**/etc/grub.d/10_linux** スクリプトは、インストールされている Linux カーネルを同じパーティションで検索します。**/etc/grub.d/30_os-prober** スクリプトは、他のオペレーティングシステムを検索します。また、カーネル更新時には、メニューエントリーがブートメニューに自動的に追加されます。

/etc/grub.d/ ディレクトリー内にある **40_custom** ファイルはカスタムエントリー用のテンプレートで、以下のようになっています。

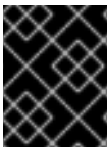
```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries. Simply type the
# menu entries you want to add after this comment. Be careful not to change
# the 'exec tail' line above.
```

このファイルは、編集またはコピーが可能です。有効なメニューエントリーには、少なくとも以下のものを含める必要があることに注意してください。

```
menuentry "<Title>"{
<Data>
}
```

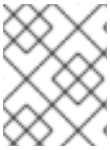
26.5.4. カスタムメニューの作成

メニューエントリーの自動更新を希望しない場合は、カスタムメニューを作成できます。



重要

*次に進む前に、後で変更を戻す必要に迫られた場合に備えて、**/etc/grub.d/** ディレクトリーのコンテンツのバックアップを作成してください。*



注記

***/etc/default/grub** ファイルを修正しても、カスタムメニューの作成には影響がないことに注意してください。*

1. *BIOS ベースのマシンでは **/boot/grub2/grub.cfg** のコンテンツを、UEFI マシンでは **/boot/efi/EFI/redhat/grub.cfg** のコンテンツをコピーします。**grub.cfg** のコンテンツを既存のヘッダー行の下で **/etc/grub.d/40_custom** ファイルに置きます。**40_custom** スクリプトの実行可能な部分は、維持される必要があります。*

2. ***/etc/grub.d/40_custom** ファイルに置かれたコンテンツからカスタムメニューの作成に必要となるのは、**menuentry** ブロックのみです。**/boot/grub2/grub.cfg** ファイルと **/boot/efi/EFI/redhat/grub.cfg** ファイルには、関数の仕様と他のコンテンツが **menuentry** ブロックの上下に含まれる可能性があります。ここまでの手順でこれらの不要な行を **40_custom** ファイルに置いた場合は、削除します。*
   *以下は、カスタムの **40_custom** スクリプトの例です。*

   ```
   #!/bin/sh
   exec tail -n +3 $0
   # This file provides an easy way to add custom menu entries. Simply type the
   # menu entries you want to add after this comment. Be careful not to change
   # the 'exec tail' line above.

   menuentry 'First custom entry' --class red --class gnu-linux --class gnu --class os
   $menuentry_id_option 'gnulinux-3.10.0-67.el7.x86_64-advanced-32782dd0-4b47-4d56-
   a740-2076ab5e5976' {
       load_video
       set gfxpayload=keep
       insmod gzio
       insmod part_msdos
       insmod xfs
   ```

```
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
     search --no-floppy --fs-uuid --set=root --hint='hd0,msdos1' 7885bba1-8aa7-4e5d-
a7ad-821f4f52170a
    else
     search --no-floppy --fs-uuid --set=root 7885bba1-8aa7-4e5d-a7ad-821f4f52170a
    fi
    linux16 /vmlinuz-3.10.0-67.el7.x86_64 root=/dev/mapper/rhel-root ro
rd.lvm.lv=rhel/root vconsole.font=latarcyrheb-sun16 rd.lvm.lv=rhel/swap
vconsole.keymap=us crashkernel=auto rhgb quiet LANG=en_US.UTF-8
    initrd16 /initramfs-3.10.0-67.el7.x86_64.img
}
menuentry 'Second custom entry' --class red --class gnu-linux --class gnu --class os
$menuentry_id_option 'gnulinux-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c-
advanced-32782dd0-4b47-4d56-a740-2076ab5e5976' {
    load_video
    insmod gzio
    insmod part_msdos
    insmod xfs
    set root='hd0,msdos1'
    if [ x$feature_platform_search_hint = xy ]; then
     search --no-floppy --fs-uuid --set=root --hint='hd0,msdos1' 7885bba1-8aa7-4e5d-
a7ad-821f4f52170a
    else
     search --no-floppy --fs-uuid --set=root 7885bba1-8aa7-4e5d-a7ad-821f4f52170a
    fi
    linux16 /vmlinuz-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c
root=/dev/mapper/rhel-root ro rd.lvm.lv=rhel/root vconsole.font=latarcyrheb-sun16
rd.lvm.lv=rhel/swap vconsole.keymap=us crashkernel=auto rhgb quiet
    initrd16 /initramfs-0-rescue-07f43f20a54c4ce8ada8b70d33fd001c.img
}
```

3. 以下を除いて、すべてのファイルを **/etc/grub.d/** ディレクトリーから削除します。

   - **00_header,**
   - **40_custom,**
   - **01_users** (存在する場合)
   - および **README**。
     別の方法では、**/etc/grub2.d/** ディレクトリーのファイルを維持したい場合は、**chmod a-x <file_name>** コマンドを実行して、実行不可にします。

4. **40_custom** ファイル内のメニューエントリーを希望に合わせて編集、追加、削除します。

5. 次のように **grub2-mkconfig -o** コマンドを実行して、**grub.cfg** ファイルを再構築します。

   - BIOS ベースのマシンでは、**root** で以下のコマンド実行します。

     ```
     ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
     ```

   - UEFI ベースのマシンでは、**root** で以下のコマンド実行します。

     ```
     ~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
     ```

26.6. パスワードを使用した GRUB 2 の保護

GRUB 2 では、以下の 2 種類のパスワード保護が選択できます。

- メニューエントリーの修正にはパスワードが必要ですが、既存のメニューエントリーの起動には**必要ありません**。
- メニューエントリーの修正にはパスワードが必要で、**かつ** メニューエントリーのいずれかまたは複数、もしくはすべての起動にもパスワードが必要。

エントリー修正のみにパスワードを必要とする GRUB 2 の設定

GRUB 2 エントリーの修正にパスワード認証を必要とするには、以下の手順に従います。

1. root で **grub2-setpassword** コマンドを実行します。

```
~]# grub2-setpassword
```

2. パスワードを入力し、確認します。

```
Enter password:
Confirm password:
```

この手順により、パスワードのハッシュを含む **/boot/grub2/user.cfg** ファイルが作成されます。このパスワードのユーザーである **root** は、**/boot/grub2/grub.cfg** ファイルで定義されます。この変更により、ブート中にブートエントリーを修正する場合は、**root** ユーザー名とパスワードの確認が必要になります。

エントリーの修正と起動にパスワードを必要とする GRUB 2 の設定

grub2-setpassword を使用してパスワードを設定すると、権限のない修正からメニューエントリーは保護されますが、権限のない起動からは保護されません。エントリーの起動にもパスワードを必要とするには、**grub2-setpassword** でパスワードを設定した後に、以下の手順を実行します。

警告

GRUB 2 パスワードを忘れると、以下の手順で再設定したエントリーは起動できなくなります。

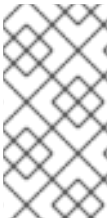
1. **/boot/grub2/grub.cfg** ファイルを開きます。
2. **menuentry** で始まる行を検索し、パスワード保護するブートエントリーを見つけます。
3. メニューエントリーブロックから **--unrestricted** パラメーターを削除します。次に例を示します。

```
[file contents truncated]
menuentry 'Red Hat Enterprise Linux Server (3.10.0-327.18.2.rt56.223.el7_2.x86_64) 7.2
(Maipo)' --class red --class gnu-linux --class gnu --class os --unrestricted
$menuentry_id_option 'gnulinux-3.10.0-327.el7.x86_64-advanced-c109825c-de2f-4340-
a0ef-4f47d19fe4bf' {
```

```
    load_video
    set gfxpayload=keep
[file contents truncated]
```

4. ファイルを保存してから閉じます。

これでエントリーの起動にも **root** ユーザー名とパスワードの入力が必要になります。



注記

/boot/grub2/grub.cfg への手動での変更は、新規カーネルのバージョンがインストールされても維持されますが、**grub2-mkconfig** コマンドを使用して **grub.cfg** を再生成すると失われます。このため、パスワード保護を維持するには、**grub2-mkconfig** を使用した後、毎回上記の手順を実行する必要があります。

注記

/boot/grub2/grub.cfg ファイルのすべてのメニューエントリーから **--unrestricted** パラメーターを削除すると、新しくインストールされたすべてのカーネルで作成されるメニューエントリーで **--unrestricted** がなくなり、自動的にパスワード保護を継承することになります。

Red Hat Enterprise Linux 7.2 への更新前でのパスワード設定
grub2-setpassword ツールは Red Hat Enterprise Linux 7.2 で追加され、GRUB 2 パスワード設定の標準メソッドになっています。Red Hat Enterprise Linux の以前のバージョンでは、**/etc/grub.d/40_custom** ファイルでブートエントリーを手動で指定し、**/etc/grub.d/01_users** ファイルでスーパーユーザーを指定する必要がありました。

GRUB 2 へのユーザーの追加
--unrestricted パラメーターなしでエントリーを起動するには、root パスワードが必要です。ただし、GRUB 2 では、パスワードを提供せずにこのエントリーを起動できる root 以外のユーザーも作成できるようになります。エントリーの変更にも root パスワードが必要です。このようなユーザーの作成については、GRUB 2 マニュアルを参照してください。

26.7. GRUB 2 の再インストール

GRUB 2 の誤ったインストールやファイルの欠如、システムの破損などで引き起こされる特定の問題を解決するには、GRUB 2 の再インストールが便利な方法となる場合があります。GRUB 2 を再インストールする理由には、これらの他に以下のものがあります。

- GRUB の以前のバージョンからのアップグレード
- インストール済みのオペレーティングシステムを制御するために、ユーザーが GRUB 2 ブートローダーを必要としている。ただし、オペレーティングシステムのなかには独自のブートローダーとインストールされるものもあります。GRUB 2 を再インストールすることで、希望するオペレーティングシステムに制御が戻されます。
- 別のドライブにブート情報を追加する。

26.7.1. BIOS ベースマシンへの GRUB 2 の再インストール

grub2-install コマンドを使用すると、ブート情報が更新され、不明なファイルが復元されます。ファイルは、破損していない場合のみ復元されます。

システムが正常に稼働している場合は、**grub2-install** device コマンドを使用して GRUB 2 を再インストールします。たとえば、**sda** がdevice の場合は、以下のようになります。

```
~]# grub2-install /dev/sda
```

26.7.2. UEFI ベースマシンへの GRUB 2 の再インストール

yum reinstall grub2-efi shim コマンドを使用すると、ブート情報が更新され、不明なファイルが復元されます。ファイルは、破損していない場合のみ復元されます。

システムが正常に稼働している場合は、**yum reinstall grub2-efi shim** コマンドを使用して GRUB 2 を再インストールします。以下に例を示します。

```
~]# yum reinstall grub2-efi shim
```

26.7.3. GRUB 2 の再設定と再インストール

この方法ではすべての GRUB 2 設定ファイルとシステム設定が完全に削除されます。この方法は全設定をデフォルト値にリセットする場合に使用します。設定ファイルを削除し、GRUB 2 を再インストールすると、破損されたファイルと間違った設定によって引き起こされた障害が修復されます。これを実行するには、**root** で以下の手順を実行します。

1. **rm /etc/grub.d/*** コマンドを実行します。
2. **rm /etc/sysconfig/grub** コマンドを実行します。
3. EFI システム**のみ**の場合は、以下のコマンドを実行します。

   ```
   ~]# yum reinstall grub2-efi shim grub2-tools
   ```

4. BIOS および EFI システムについては、以下のコマンドを実行します。

   ```
   ~]# yum reinstall grub2-tools
   ```

5. 次のように **grub2-mkconfig -o** コマンドを実行して、**grub.cfg** ファイルを再構築します。
   - BIOS ベースのマシンでは、**root** で以下のコマンド実行します。

     ```
     ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
     ```

   - UEFI ベースのマシンでは、**root** で以下のコマンド実行します。

     ```
     ~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
     ```

6. 「GRUB 2 の再インストール」にある手順に従い、**/boot/** パーティション上の GRUB 2 を復元します。

26.8. GRUB LEGACY から GRUB 2 へのアップグレード

Red Hat Enterprise Linux (RHEL) バージョン 6 を 7 へインプレースアップグレードを行う場合は、**GRUB Legacy** から **GRUB 2** へのアップグレードは自動的に行われませんが、手動で実行できます。以下の理由から GRUB アップグレードを実行します。

- *RHEL 7 以降では、**GRUB Legacy** は維持されず、更新を受信しません。*
- ***GRUB Legacy** は **/boot/** ディレクトリーがないと、システム上で起動できません。*
- ***GRUB 2** はより機能が豊富で、信頼性に優れています。*
- ***GRUB 2** は、より多くのハードウェア設定、ファイルシステム、ドライブレイアウトをサポートしています。*

オペレーティングシステムのインプレースアップグレード後に GRUB Legacy から GRUB 2 へアップグレード。

GRUB Legacy から GRUB 2 へのアップグレード

1. *GRUB Legacy パッケージが Red Hat Upgrade Tool によりアンインストールされていることを確認します。*

```
~]# yum remove grub
```

注記

*grub2 パッケージをアンインストールしても、インストールされている **GRUB Legacy** ブートローダーには影響しません。*

2. *grub2 パッケージがインストールされていることを確認します。**RHEL 7** へのアップグレード後に grub2 がシステムにない場合は、以下を実行して手動でインストールできます。*

```
~]# yum install grub2
```

3. *EFI を使用してシステムを起動した場合は、以下のパッケージをインストールします。*

```
~]# yum install grub2-efi-x64 shim-x64
```

GRUB 2 設定ファイルの生成

*本セクションでは、**GRUB Legacy** 設定を削除せずに **GRUB 2** 設定を追加する方法を説明します。**GRUB 2** が正常に作動しない場合に備えて、**GRUB Legacy** 設定を残します。*

1. *以下のオプションのいずれかを使用して、 **/etc/default/grub** ファイルを手動で作成します。*
   - */etc/default/grub ファイルを作成します。詳細は、How to re-create the missing /etc/default/grub file in Red Hat Enterprise Linux 7? の記事を参照してください。*
   - *同様の Red Hat Enterprise Linux 7 システムから **/etc/default/grub** ファイルをコピーし、そのファイルを調整します。*
2. *ブートローダーにより異なります。*
   a. *システムがレガシー BIOS を使用して起動する場合は、インストールデバイスを指定して **GRUB2** をインストールします。*

```
~]# grub2-install /dev/<DEVICE_NAME> --grub-setup=/bin/true
```

*grub2-install コマンドは、GRUB イメージを **/boot/grub** ターゲットディレクトリーにインストールします。*

--grub-setup=/bin/true オプションで、古い **GRUB Legacy** 設定が削除されないようにします。

b. *EFI* を使用してシステムを起動している場合は、*shim* ブートローダーのブートエントリーを作成し、**BootOrder** 変数を変更して、**shim** によってファームウェアブート **GRUB 2** を設定します。

```
~]# efibootmgr -c -L 'Red Hat Enterprise Linux 7' -d /dev/device_name -p 1 -l
'\EFI\redhat\shimx64.efi'
```

/dev/device_name を、起動可能なデバイスファイルに置き換えます。

警告

設定ファイル拡張子の違いに気を付けてください。

- **.conf** は、**GRUB** 向けです。
- **.cfg** は、**GRUB 2** 向けです。

次の手順で、誤って古い **GRUB** 設定ファイルを上書きしないよう気を付けてください。

3. **GRUB 2** 設定ファイルを生成します。

a. システムがレガシー **BIOS** を使用する場合は、以下を行います。

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

b. システムが **EFI** を使用する場合は、以下を行います。

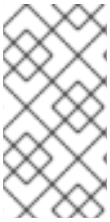
```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

注記

生成した **GRUB 2** 設定ファイルをカスタマイズするには、「GRUB 2 設定ファイルのカスタマイズ」 を参照してください。変更は、**/boot/grub2/grub.cfg** に直接行うのではなく、**/etc/default/grub** に行う必要があります。**/boot/grub2/grub.cfg** を直接変更しても、ファイルが再生成されるたびに変更が失われることになります。

GRUB Legacy ブートローダーをインストールしたまま GRUB 2 のテスト

本セクションでは、**GRUB Legacy** 設定を削除せずに **GRUB 2** をテストする方法を説明します。**GRUB Legacy** 設定は、**GRUB 2** 設定の検証が終わるまでそのままにします。**GRUB 2** 設定を安全にテストするために、**GRUB 2** を **GRUB Legacy** から起動します。



注記

このセクションは、レガシー BIOS ブートにのみ適用されます。EFI の場合、古いブートローダーと新しいブートローダーの両方にブートエントリーが存在し、EFI ファームウェア設定で任意のブートエントリーを選択することで、古いレガシーの GRUB を起動できます。

1. *新しいセクションを /boot/grub/grub.conf に追加します。*
   システムに別の /boot パーティションがある場合は、以下を使用します。

```
title GRUB 2 Test
 root (hd0,0)
 kernel /grub2/i386-pc/core.img
 boot
```

   (hd0,0) *を GRUB Legacy の起動可能なデバイスの表示記号と置き換えます。*

   システムに個別の /boot パーティションがない場合は、以下を使用します。

```
title GRUB 2 Test
 root (hd0,0)
 kernel /boot/grub2/i386-pc/core.img
 boot
```

   (hd0,0) *を GRUB Legacy の起動可能なデバイスの表示記号と置き換えます。*

2. *システムを再起動します。*
3. *GRUB Legacy メニューが表示されたら、GRUB 2 Test エントリーを選択します。*
4. *GRUB 2 メニューが表示されたら、起動するカーネルを選択します。*
5. *上記がうまく機能しない場合は再起動し、次の起動時に GRUB 2 Test エントリーを選択しないでください。*

BIOS を使用するシステムで GRUB Legacy ブートローダーの置き換え

GRUB 2 が正常に機能している場合は、以下を行います。

1. *GRUB Legacy ブートローダーを GRUB 2 ブートローダーに置き換えます。*

```
~]# grub2-install /dev/sdX
```

2. *古い GRUB Legacy 設定ファイルを削除します。*

```
~]# rm /boot/grub/grub.conf
```

3. *システムを再起動します。*

```
~]# reboot
```

EFI を使用するシステムの GRUB Legacy の削除

GRUB 2 が正常に機能している場合は、以下を行います。

1. */boot/efi/EFI/redhat/* ディレクトリーのコンテンツを確認して、Legacy GRUB のみに関連する古くなったファイルを削除します。

```
~]# rm /boot/efi/EFI/redhat/grub.efi
~]# rm /boot/efi/EFI/redhat/grub.conf
```

2. Preupgrade Assistant ユーティリティーおよびRed Hat Upgrade Tool ユーティリティーを使用して RHEL 6 から RHEL 7 へのインプレースアップグレードを実行した場合は、**.preupg** 接尾辞が付いたファイルのバックアップコピーも削除します。

```
~]# rm /boot/efi/EFI/redhat/*.preupg
```

3. **efibootmgr** コマンドを使用して **\EFI\redhat\grub.efi** ファイルを参照する古いブートエントリーを見つけます。

```
~]# efibootmgr -v | grep '\\EFI\\redhat\\grub.efi'
```

出力例:

```
Boot0001* Linux    HD(1,GPT,542e410f-cbf2-4cce-9f5d-
61c4764a5d54,0x800,0x64000)/File(\EFI\redhat\grub.efi)
```

この場合のエントリー番号は **0001** です。

4. 指定したブートエントリーを削除します。次のコマンドは、上記の例からブートエントリーを削除します。

```
~]# efibootmgr -Bb 0001
```

複数のブートエントリーがある場合は、特定された古いブートエントリーをすべて削除します。

警告

RHEL 6 などの古いリリースから RHEL 7 にアップグレードしたら、GRUB Legacy ブートローダーの GRUB 2 への手動アップグレードが正常に完了するまで、オペレーティングシステムはサポートされません。これは、メジャーリリース間でのパッケージのインストールがサポートされていないためです。RHEL 6 の GRUB Legacy とは異なり、RHEL 7 の場合は、GRUB 2 のみがサポート、開発、およびテストされます。

26.9. シリアルコンソールでの GRUB 2

本セクションでは、ディスプレイやキーボードのないマシンでのシリアル通信に **GRUB 2** を設定する方法を説明します。

シリアル接続で **GRUB 2**> 端末にアクセスするには、特定のカーネルがシリアル接続を監視するように、カーネル定義に追加オプションを追加する必要があります。

以下に例を示します。

```
console=ttyS0,9600n8
```

console=ttyS0 は使用するシリアルターミナルです。**9600** はボーレート、**n** は非パリティー用、および **8** はビット単位の単語長さを示します。たとえば、**115200** のように非常に高いボーレートは、ログファイルの後のタスクに適しています。

シリアルコンソール設定の詳細は、「インストールできる外部のドキュメント」 を参照してください。

26.9.1. シングルブート用に GRUB 2 の設定

単一ブートプロセス中にのみシステムがシリアルターミナルを使用するよう設定するには、GRUB 2 ブートメニューが表示されたときに、起動したいカーネルにカーソルを移動し、**e** キーを押してカーネルパラメーターを編集します。**rhgb** パラメーターと **quiet** パラメーターを削除し、以下のように **linux16** 行の最後にコンソールパラメーターを追加します。

```
linux16   /vmlinuz-3.10.0-0.rc4.59.el7.x86_64 root=/dev/mapper/rhel-root ro rd.md=0 rd.dm=0
rd.lvm.lv=rhel/swap crashkernel=auto rd.luks=0 vconsole.keymap=us rd.lvm.lv=rhel/root
console=ttyS0,9600
```

この設定は永続的なものではなく、これから実行するブートにのみ適用されます。

26.9.2. 永続的な変更のための GRUB 2 の設定

システムでメニューエントリーの変更を永続的に行うには、**grubby** ツールを使用します。たとえば、デフォルトのカーネルのエントリーを更新するには、以下のようにコマンドを入力します。

```
~]# grubby --remove-args="rhgb quiet" --args=console=ttyS0,9600 --update-kernel=DEFAULT
```

--update-kernel パラメーターはキーワード **ALL**、またはカーネルインデックス番号のコンマ区切りのリストも受け入れます。**grubby** の詳細は 「GRUB 2 メニューエントリーに対する引数の追加および削除」 を参照してください。

26.9.3. 新しい GRUB 2 ファイルの設定

新しい GRUB 2 設定ファイルを構築する必要がある場合は、**/etc/default/grub** ファイルに次の 2 行を追加します。

```
GRUB_TERMINAL="serial"
GRUB_SERIAL_COMMAND="serial --speed=9600 --unit=0 --word=8 --parity=no --stop=1"
```

最初の行は、グラフィカルターミナルを無効にします。**GRUB_TERMINAL** キーを指定すると、**GRUB_TERMINAL_INPUT** および **GRUB_TERMINAL_OUTPUT** の値を上書きすることに注意してください。2 行目では、ボーレートやパリティーなどの値を使用中の環境とハードウェアに適合するように調整します。たとえば、**115200** のように非常に高いボーレートは、ログファイルの後のタスクに適しています。**/etc/default/grub** ファイルを変更した場合は、GRUB 2 設定ファイルの更新が必要になります。

次のように **grub2-mkconfig -o** コマンドを実行して、**grub.cfg** ファイルを再構築します。

- BIOS ベースのマシンでは、**root** で以下のコマンド実行します。

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- *UEFI* ベースのマシンでは、**root** で以下のコマンド実行します。

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

26.9.4. screen を使用したシリアルコンソールへの接続

screen ツールは、シリアルターミナルとして機能します。このツールをインストールするには、**root** で以下を実行します。

```
~]# yum install screen
```

シリアルコンソールを使用してマシンに接続するには、コマンドを以下のように使用します。

```
screen /dev/console_port baud_rate
```

デフォルトでは、オプションが指定されない場合、*screen* は標準の 9600 ボーレートを使用します。大きいボーレートを設定するには、以下のコマンドを入力します。

```
~]$ screen /dev/console_port 115200
```

ここで、console_port は **ttyS0** や **ttyUSB0** などです。

screen でセッションを終了するには、**Ctrl+a** を押して、**:quit** と入力し、**Enter** を押します。

追加オプションおよび詳細情報は、**screen(1)** man ページを参照してください。

26.10. ブート中のターミナルメニューの編集

ブート時にメニューエントリーを修正し、カーネルに引数を渡すことができます。これは、メニューエントリーエディターインターフェイスを使用して行い、このインターフェイスはブートローダーメニュー内で選択したメニューエントリーで **e** キーを押すと開始します。**Esc** キーはすべての変更を破棄し、標準メニューインターフェイスを再度読み込みます。**c** キーは、コマンドラインインターフェイスを読み込みます。

コマンドラインインターフェイスは最も基本的な *GRUB 2* インターフェイスですが、制御が最も課せられます。コマンドラインでは、関連する *GRUB 2* コマンドの入力が可能で、それに続けて **Enter** キーを押すとそのコマンドが実行します。このインターフェイスには、コンテキストをベースにした **Tab** キー自動入力や行頭に移動する **Ctrl+a**、行末に移動する **Ctrl+e** など、*shell* と同様の高度な機能が搭載されています。さらに、矢印、**Home**、**End**、**Delete** のキー動作は、*bash* シェルでの機能と同じになります。

26.10.1. レスキューモードでの起動

レスキューモードは、便利なシングルユーザー環境を提供し、通常の起動プロセスを完了できない状況においてシステムの修復を可能にします。レスキューモードでは、システムはすべてのローカルファイルシステムのマウントと、いくつかの重要なシステムサービスの開始を試みますが、ネットワークインターフェイスをアクティブにしたり、他のユーザーによるシステムへの同時ログインを許可したりすることはしません。*Red Hat Enterprise Linux 7* では、レスキューモードはシングルユーザーモードと同等であり、**root** パスワードを必要とします。

1. システムの起動時にレスキューモードに入るには、GRUB 2 ブート画面で **e** キーを押して編集します。

2. 64 ビット IBM Power シリーズの場合は **linux** 行、x86-64 BIOS ベースシステムの場合は **linux16** 行、または UEFI システムの場合は **linuxefi** 行の最後に以下のパラメーターを追加します。

```
systemd.unit=rescue.target
```

Ctrl+a か **Ctrl+e** を押して行の最初または最後に移動します。システムによっては、**Home** と **End** が機能するものもあります。

1、**s**、および **single** という同等のパラメーターをカーネルに渡すことも可能であることに注意してください。

3. **Ctrl+x** を押して、パラメーターでシステムを起動します。

26.10.2. 緊急モードでのブート

緊急モードは、可能な限り最小限の環境を提供し、システムがレスキューモードに入れない状態でもシステムの修復を可能にします。緊急モードでは、システムは **root** ファイルシステムを読み込み専用でマウントし、他のローカルファイルシステムのマウントは試みません。また、ネットワークインターフェイスのアクティブ化も行わず、限定的な必須サービスのみを起動します。Red Hat Enterprise Linux 7 では、緊急モードでは **root** パスワードが必要です。

1. 緊急モードに入るには、GRUB 2 ブート画面で **e** キーを押して編集を行います。

2. 64 ビット IBM Power シリーズの場合は **linux** 行、x86-64 BIOS ベースシステムの場合は **linux16** 行、または UEFI システムの場合は **linuxefi** 行の最後に以下のパラメーターを追加します。

```
systemd.unit=emergency.target
```

Ctrl+a か **Ctrl+e** を押して行の最初または最後に移動します。システムによっては、**Home** と **End** が機能するものもあります。

emergency および **-b** という同等のパラメーターをカーネルに渡すことも可能であることに注意してください。

3. **Ctrl+x** を押して、パラメーターでシステムを起動します。

26.10.3. デバッグシェルのブート

systemd デバッグシェルは、起動プロセスの初期段階で **systemd** 関連のブート問題を診断するために使用できるシェルを提供します。デバッグシェルでは、**systemctl list-jobs** や **systemctl list-units** などの **systemctl** コマンドを使用してブート問題の原因を調べることができます。さらに、デバッグオプションをカーネルコマンドラインに追加して、ログメッセージの数を増やすこともできます。**systemd** では、カーネルコマンドラインオプションの **debug** が、**systemd.log_level=debug** のショートカットになりました。

デバッグシェルコマンドの追加

このセッションでのみデバッグシェルを有効にするには、以下の手順を実行します。

1. GRUB 2 ブート画面で、編集するメニューエントリーにカーソルを移動し、**e** キーを押して編集できるようにします。

2. *64 ビット IBM Power シリーズの場合は **linux** 行、x86-64 BIOS ベースシステムの場合は **linux16** 行、または UEFI システムの場合は **linuxefi** 行の最後に以下のパラメーターを追加します。*

   ```
   systemd.debug-shell
   ```

   必要に応じて、デバッグオプションを追加します。

   ***Ctrl+a** か **Ctrl+e** を押して行の最初または最後に移動します。システムによっては、**Home** と **End** が機能するものもあります。*

3. ***Ctrl+x** を押して、パラメーターでシステムを起動します。*

*必要な場合は、**systemctl enable debug-shell** コマンドで有効にして、デバッグシェルがブート時に常に起動されるよう設定できます。また、**grubby** ツールを使用して GRUB 2 メニューのカーネルコマンドラインへの変更を永続的に行うこともできます。**grubby** の詳細は 「grubby ツールを使用した GRUB 2 メニューの永続的な変更」 を参照してください。*

警告

デバッグシェルを使用するには認証が必要ないため、デバッグシェルを永続的に有効にすることにはセキュリティー上のリスクがあります。デバッグシェルは、デバッグセッションが終了したときに無効にしてください。

デバッグシェルへの接続

*ブートプロセス中に、**systemd-debug-generator** により TTY9 にデバッグシェルが設定されます。*

1. ***Ctrl+Alt+F9** を押してデバッグシェルに接続します。仮想マシンを使用している場合は、このキーの組み合わせを送信するには、仮想化アプリケーションからのサポートが必要です。たとえば、Virtual Machine Manager を使用している場合は、メニューから **Send Key → Ctrl+Alt+F9** を選択します。*

2. *デバッグシェルには認証が必要ないため、**[root@localhost /]#** のようなプロンプトが **TTY9** に表示されます。*

3. *必要な場合は、以下のようなコマンドを実行して、デバッグシェルにいることを確認します。*

   ```
   /]# systemctl status $$
   ● debug-shell.service - Early root shell on /dev/tty9 FOR DEBUGGING ONLY
     Loaded: loaded (/usr/lib/systemd/system/debug-shell.service; disabled; vendor
   preset: disabled)
     Active: active (running) since Wed 2015-08-05 11:01:48 EDT; 2min ago
      Docs: man:sushell(8)
   Main PID: 450 (bash)
     CGroup: /system.slice/debug-shell.service
             ├─ 450 /bin/bash
             └─1791 systemctl status 450
   ```

4. *デフォルトのシェルに戻るには、ブートが成功した場合に **Ctrl+Alt+F1** を押します。*

起動に関する問題を診断するために、特定の **systemd** ユニットは、カーネルコマンドラインで **systemd.mask=**unit_name を 1 つ以上追加することにより、マスクできます。ブートプロセス中に追加のプロセスを起動するには、**systemd.wants=**unit_name をカーネルコマンドラインに追加します。これらのオプションは、**systemd-debug-generator(8)** man ページを参照してください。

26.10.4. root パスワードの変更およびリセット

Red Hat Enterprise Linux 7 インストールでは、**root** パスワードを設定する必要があります。**root** パスワードが分からなくなった場合は、パスワードをリセットできます。ユーザーが **wheel** グループのメンバーである場合は、以下のように **root** パスワードを変更できます。

```
~]$ sudo passwd root
```

GRUB 2 では、Red Hat Enterprise Linux 6 での GRUB のようなシングルユーザーモードでパスワードの再設定を行いません。**single-user** モードおよび **emergency** モードでの操作には、**root** パスワードが必要となっています。

root パスワードをリセットする 2 つの手順を以下に示します。

- インストールディスクを使用した root パスワードのリセット では、GRUB 2 メニューを編集せずにシェルプロンプトにアクセスする方法について説明しています。この方法は 2 番目の手順よりも短く、推奨される方法でもあります。ブートディスクまたは通常の Red Hat Enterprise Linux 7 インストールディスクを使用できます。
- rd.break を使用した root パスワードのリセット では、制御が **initramfs** から **systemd** に渡される前に **rd.break** を使用してブートプロセスを中断する方法について説明しています。この方法の欠点は、手順が多いことです。たとえば、GRUB 2 メニューを編集し、SELinux ファイルの再ラベル行いますが、時間がかかる可能性があります。または、SELinux 強制モードを変更し、ブート完了時に **/etc/shadow**/ の SELinux セキュリティーコンテキストを復元する必要があります。

インストールディスクを使用した root パスワードのリセット

1. システムを起動し、BIOS 情報が表示されたときに、ブートメニューのオプションを選択して、インストールディスクからのブートを選択します。
2. **Troubleshooting** (トラブルシューティング) を選択します。
3. **Rescue a Red Hat Enterprise Linux System** (**Red Hat Enterprise Linux** システムのレスキュー) を選択します。
4. デフォルトオプションである **Continue** (選択) を選択します。暗号化されたファイルシステムが見つかった場合は、この時点で、パスフレーズを入力するよう求められます。
5. シェルプロンプトが表示されるまで **OK** を押して、表示された情報を承認します。
6. 以下のようにファイルシステム **root** を変更します。

   ```
   sh-4.2# chroot /mnt/sysimage
   ```

7. **passwd** コマンドを入力し、コマンドラインに表示される指示にしたがって **root** パスワードを変更します。
8. 時間がかかるディスクの SELinux の再ラベルを防ぐために、**autorelable** ファイルを削除します。

```
sh-4.2# rm -f /.autorelabel
```

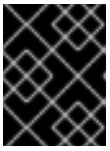
9. **exit** コマンドを入力して、**chroot** 環境を終了します。

10. **exit** コマンドを再び実行して初期化を再開し、システム起動を完了します。

rd.break を使用した root パスワードのリセット

1. システムを起動し、GRUB 2 ブート画面で **e** キーを押して編集を行います。

2. **rhgb** パラメーターおよび **quiet** パラメーターを、最後または末前 (**linux16** 行、UEFI システムの場合は **linuxefi**) から削除します。
Ctrl+a か **Ctrl+e** を押して行の最初または最後に移動します。システムによっては、**Home** と **End** が機能するものもあります。



重要

rhgb および **quiet** パラメーターは、システムメッセージを有効化するために削除する必要があります。

3. 64 ビット IBM Power シリーズの場合は **linux** 行、x86-64 BIOS ベースシステムの場合は **linux16** 行、または UEFI システムの場合は **linuxefi** 行の最後に以下のパラメーターを追加します。

```
rd.break enforcing=0
```

enforcing=0 オプションを追加すると、時間がかかる SELinux の再ラベルプロセスを省略できます。

initramfs は、Linux kernel に制御が渡される前に停止するため、**root** ファイルシステムで作業を行えます。

initramfs プロンプトは、Linux 行で指定された最後のコンソールに表示されます。

4. **Ctrl+x** を押して変更したパラメーターでシステムを起動します。
暗号化されたファイルシステムでは、この時点ではパスワードが必要になります。ただし、パスワードプロンプトはロギングメッセージに隠れて表示されないことがあります。**Backspace** キーを押してプロンプトを表示させることができます。ログメッセージを無視して、キーを放し、暗号化されたファイルシステムのパスワードを入力します。

initramfs の **initramfs switch_root** プロンプトが表示されます。

5. ファイルシステムが **/sysroot/** で読み取り専用でマウントされます。ファイルシステムが書き込み可能になっていないと、パスワードの変更はできません。
ファイルシステムを書き込み可能で再マウントします。

```
switch_root:/# mount -o remount,rw /sysroot
```

6. 書き込みが有効な状態でファイルシステムが再マウントされます。
以下のようにファイルシステムの **root** を変更します。

```
switch_root:/# chroot /sysroot
```

プロンプトが **sh-4.2#** に変わります。

7. **passwd** コマンドを入力し、コマンドラインに表示される指示にしたがって**root** パスワードを変更します。
システムが書き込み可能でないと、passwd ツールは失敗し、以下のエラーメッセージが表示されます。

```
Authentication token manipulation error
```

8. パスワードファイルを更新すると、SELinux セキュリティーコンテキストが正しくないファイルが作成されます。次回のシステムのブート時にすべてのファイルを再ラベルするには、以下のコマンドを入力します。

```
sh-4.2# touch /.autorelabel
```

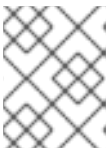
また、手順3 で **enforcing=0** オプションを指定した場合は、大きいディスクを再ラベルするのにかかる時間を節約するために、この手順を省略できます。

9. ファイルシステムを読み取り専用で再マウントします。

```
sh-4.2# mount -o remount,ro /
```

10. **exit** コマンドを入力して、**chroot** 環境を終了します。

11. **exit** コマンドを再び実行して初期化を再開し、システム起動を完了します。
暗号化されているファイルシステムの場合は、この時点でパスワードまたはパスフレーズが必要です。ただし、パスワードプロンプトはロギングメッセージに隠れて表示されないことがあります。**Backspace** キーを押して保持すると、プロンプトが表示されます。ログメッセージを無視して、キーを放し、暗号化されたファイルシステムのパスワードを入力します。



注記

SELinux の再ラベルプロセスには長時間がかかることがあることに注意してください。このプロセスが完了すると、システムが自動的に再起動されます。

12. 手順3 で **enforcing=0** オプションを追加し、手順8 で**touch /.autorelabel** コマンドを省略した場合は、以下のコマンドを入力して**/etc/shadow** ファイルの SELinux セキュリティーコンテキストを復元します。

```
~]# restorecon /etc/shadow
```

以下のコマンドを実行して、SELinux ポリシーの強制を再び有効にし、ポリシーの強制が有効になっていることを確認します。

```
~]# setenforce 1
~]# getenforce
Enforcing
```

26.11. UNIFIED EXTENSIBLE FIRMWARE INTERFACE (UEFI) セキュアブート

Unified Extensible Firmware Interface(UEFI) セキュアブートテクノロジーにより、ファームウェアに含まれる公開鍵のデータベースで承認された暗号化キーでシステムブートローダーが署名されているかどうかをシステムファームウェアがチェックするようになります。また、次の段階のブートローダーお

よびカーネルでの署名確認により、信頼できるキーで署名されていないカーネルスペースコードの実行を阻止できます。

信頼チェーンは、次のようにファームウェアから署名済みドライバーおよびカーネルモジュールの方向で確立されます。第一段階のブートローダー **shim.efi** は UEFI 秘密鍵により署名されてから、認証局 (CA) により署名され、ファームウェアデータベースに保存された公開鍵により認証されます。**shim.efi** には、GRUB 2 ブートローダー **grubx64.efi** と Red Hat カーネルの両方を認証するのに使用する Red Hat 公開鍵 Red Hat Secure Boot (CA key 1) が含まれます。カーネルには、ドライバーおよびモジュールを認証する公開鍵が含まれます。

セキュアブートは、Unified Extensible Firmware Interface (UEFI) 仕様のブートパス検証コンポーネントです。この仕様は、以下を定義します。

- 揮発性ではないストレージでの暗号で保護された UEFI 変数用のプログラミングインターフェイス
- 信頼できる X.509 root 証明書の UEFI 変数での保存方法
- ブートローダーやドライバーなどの UEFI アプリケーションの検証
- 既知の問題のある証明書およびアプリケーションハッシュを無効にする手順

UEFI Secure Boot は、第 2 ステージのブートローダーのインストールや削除を妨げたりせず、このような変更についてユーザーによる明示的な確認を必要としません。また、このような変更の明示的なユーザー確認を必要としません。したがって、UEFI セキュアブートはブートパスの操作を停止せず、承認されていない変更を検出するのに役立ちます。。新しいブートローダーまたはカーネルは、システムで信頼されている鍵で署名されている限り動作します。

26.11.1. Red Hat Enterprise Linux 7 での UEFI セキュアブートサポート

Red Hat Enterprise Linux 7 には UEFI セキュアブート機能が含まれているので、Red Hat Enterprise Linux 7 は UEFI セキュアブートが有効になっているシステム上でインストールし、実行できます。セキュアブートテクノロジーが有効になっている UEFI ベースのシステムでは、ロードされたすべてのドライバーが信頼できる鍵で署名されている必要があります。Red Hat が提供するすべてのドライバーは、Red Hat の秘密鍵のいずれかで署名され、カーネル内の対応する Red Hat 公開鍵によって認証されます。

Red Hat Enterprise Linux DVD では提供されていない外部構築のドライバーを読み込みたい場合は、これらのドライバーも署名されていることを確認してください。

カスタムドライバーの署名に関する情報は、Red Hat Enterprise Linux 7 カーネル管理ガイドの Signing Kernel Modules for Secure Boot を参照してください。

UEFI セキュアブートによる制限
Red Hat Enterprise Linux 7 の UEFI セキュアブートサポートは、カーネルモードコードをその署名が適切に認証された後にのみ実行するよう設計されています (制限が存在します)。

GRUB 2 モジュールの署名および検証を行うインフラストラクチャーがないため、GRUB 2 モジュールのロードは無効になります。これは、これらのロードを許可することで、セキュアブートが定義するセキュリティー境界内で信頼できないコードが実行されることがあることを意味します。代わりに、Red Hat は、すでに含まれている Red Hat Enterprise Linux 7 でサポートされるすべてのモジュールを含む署名済み GRUB 2 バイナリーを提供します。

詳細は、Red Hat ナレッジベースの記事 UEFI セキュアブートによる制約 を参照してください。

26.12. 関連情報

GRUB 2 ブートローダーに関する詳細情報は、以下のリソースを参照してください。

インストールされているドキュメント

- **/usr/share/doc/grub2-tools-**version-number**/**: このディレクトリーには、GRUB 2 の使用および設定に関する情報が記載されています。version-number は、インストールされている GRUB 2 パッケージのバージョンに対応します。
- **info grub2**: GRUB 2 情報ページにはチュートリアル、ユーザーリファレンスガイド、プログラマーリファレンスガイド、GRUB 2 とその使用方法に関する FAQ などが含まれています。
- **grubby(8)**: GRUB および GRUB 2 を設定するコマンドラインツールの man ページです。
- **new-kernel-pkg(8)**: カーネルインストールのスクリプトを記述するツールの man ページです。

インストールできる外部のドキュメント

- **/usr/share/doc/kernel-doc-**kernel_version**/Documentation/serial-console.txt**: kernel-doc パッケージが提供するこのファイルには、シリアルコンソールに関する情報が含まれます。カーネルのドキュメントにアクセスする前に、**root** で以下のコマンドを実行する必要があります。

```
~]# yum install kernel-doc
```

- Red Hat インストールガイド: このインストールガイドでは、インストール、用語、インターフェイス、コマンドなど、GRUB 2 に関する基本的な情報を紹介しています。

パート VIII. システムバックアップおよびリカバリー

*このパートでは、**Relax-and-Recover (ReaR)** 障害復旧およびシステム移行ユーティリティーを説明します。*

第27章 REAR (RELAX-AND-RECOVER)

ソフトウェアやハードウェア障害でシステムが破損した場合、システム管理者は新たなハードウェア環境上で完全に機能する状態にシステムを復元するために以下の3 つのタスクを実行する必要があります。

1. 新規ハードウェア上でレスキューシステムを起動する
2. オリジナルのストレージレイアウトを複製する
3. ユーザーおよびシステムファイルの復元

ほとんどのバックアップソフトウェアは、3 番目の問題しか解決しません。最初の2 つの問題を解決できるのが、障害復旧およびシステム移行ユーティリティーである Relax-and-Recover (ReaR) です。

バックアップソフトウェアはバックアップを作成します。ReaR は **レスキューシステム** を作成することでこれを補完します。新しいハードウェアでレスキューシステムを起動すると、復元プロセスが開始する **rear recover** コマンドを実行できます。このプロセス中に ReaR はパーティションのレイアウトとファイルシステムを複製し、バックアップソフトウェアが作成したバックアップからのユーザーおよびシステムファイルの復元を促進し、最後にブートローダーをインストールします。デフォルトでは、ReaR が作成したレスキューシステムはストレージレイアウトとブートローダーのみを復元し、実際のユーザーおよびシステムファイルは復元しません。

本章では、ReaR の使用方法を説明します。

27.1. 基本的な REAR の使用方法

27.1.1. ReaR のインストール

root で以下のコマンドを実行して、rear パッケージをインストールします。

```
~]# yum install rear
```

27.1.2. ReaR の設定

ReaR は **/etc/rear/local.conf** ファイルで設定します。以下の行を追加してレスキューシステムの設定を指定します。

```
OUTPUT=output format
OUTPUT_URL=output location
```

output format を、レスキューシステムの形式に置き換えます。たとえば、ISO ディスクイメージであれば **ISO**、起動可能な USB であれば **USB** などにします。

output location を、配置場所に置き換えます。たとえば、ローカルのファイルシステムディレクトリーであれば **file:///mnt/rescue_system/**、SFTP ディレクトリーであれば **sftp://backup:password@192.168.0.0/** などにします。

例27.1 レスキューシステムの形式および場所の設定

ReaR がレスキューシステムを ISO イメージで **/mnt/rescue_system/** ディレクトリーに出力するようにするには、以下の行を **/etc/rear/local.conf** ファイルに追加します。

```
OUTPUT=ISO
OUTPUT_URL=file:///mnt/rescue_system/
```

オプションリストは、man ページ rear(8) の Rescue Image Configuration のセクションを参照してください。

ISO 固有の設定
例27.1「レスキューシステムの形式および場所の設定」 の設定を使用すると、2 つの場所に 2 つの同等のファイルが出力されます。

- /var/lib/rear/output/: rear のデフォルトの出力ロケーション
- /mnt/rescue_system/HOSTNAME/rear-localhost.iso: OUTPUT_URL で指定された出力ロケーション

しかし、通常必要になるのは ISO イメージだけです。ReaR にユーザーが指定したディレクトリーに ISO イメージのみを作成させるには、以下の行を /etc/rear/local.conf に追加します。

```
OUTPUT=ISO
BACKUP=NETFS
OUTPUT_URL=null
BACKUP_URL="iso:///backup"
ISO_DIR="output location"
```

output location を出力先に置き換えます。

27.1.3. レスキューシステムの作成

以下の例では、出力結果が詳細モードとなるレスキューシステムを作成する方法を示しています。

```
~]# rear -v mkrescue
Relax-and-Recover 1.17.2 / Git
Using log file: /var/log/rear/rear-rhel7.log
mkdir: created directory '/var/lib/rear/output'
Creating disk layout
Creating root filesystem layout
TIP: To login as root via ssh you need to set up /root/.ssh/authorized_keys or
SSH_ROOT_PASSWORD in your configuration file
Copying files and directories
Copying binaries and libraries
Copying kernel modules
Creating initramfs
Making ISO image
Wrote ISO image: /var/lib/rear/output/rear-rhel7.iso (124M)
Copying resulting files to file location
```

例27.1「レスキューシステムの形式および場所の設定」 の設定で、ReaR は上記を出力します。最後の 2 行は、レスキューシステムが正常に作成され、設定されたバックアップの場所である /mnt/rescue_system/ にコピーされたことを示しています。システムのホスト名が rhel7 であることから、バックアップの場所には rhel7/ ディレクトリーが含まれ、レスキューシステムと補助ファイルが格納されます。

```
~]# ls -lh /mnt/rescue_system/rhel7/
```

```
total 124M
-rw-------. 1 root root 202 Jun 10 15:27 README
-rw-------. 1 root root 166K Jun 10 15:27 rear.log
-rw-------. 1 root root 124M Jun 10 15:27 rear-rhel7.iso
-rw-------. 1 root root 274 Jun 10 15:27 VERSION
```

レスキューシステムを外部メディアに移動して、障害の際になくならないようにします。

27.1.4. ReaR のスケジューリング

ReaR が cron ジョブスケジューラーを使用して定期的にレスキューシステムを作成するようにするには、以下の行を **/etc/crontab** ファイルに追加します。

```
minute hour day_of_month month day_of_week root /usr/sbin/rear mkrescue
```

上記のコマンドを cron 時間指定 (「cron ジョブのスケジュール設定」 を参照) に置き換えます。

例27.2 ReaR のスケジューリング

ReaR が平日の 22:00 時に毎日レスキューシステムを作成するようにするには、以下の行を **/etc/crontab** に追加します。

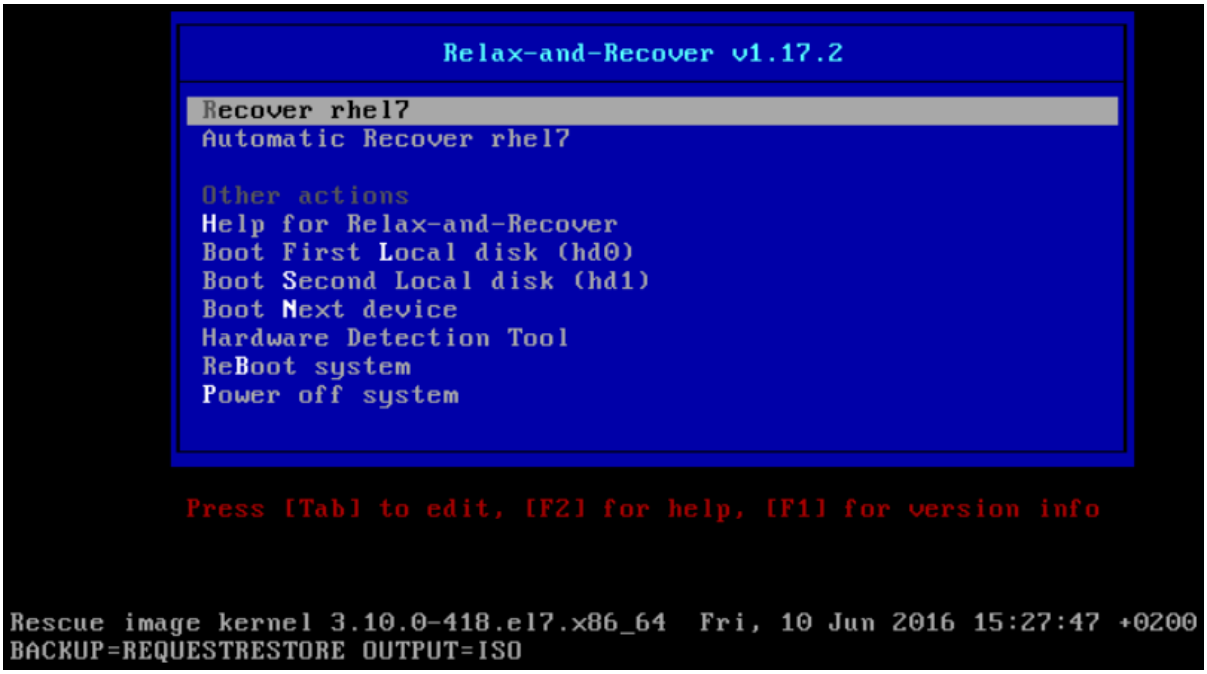
```
0 22 * * 1-5 root /usr/sbin/rear mkrescue
```

27.1.5. システムレスキューの実行

復旧または移行を実行するには、以下の手順を行います。

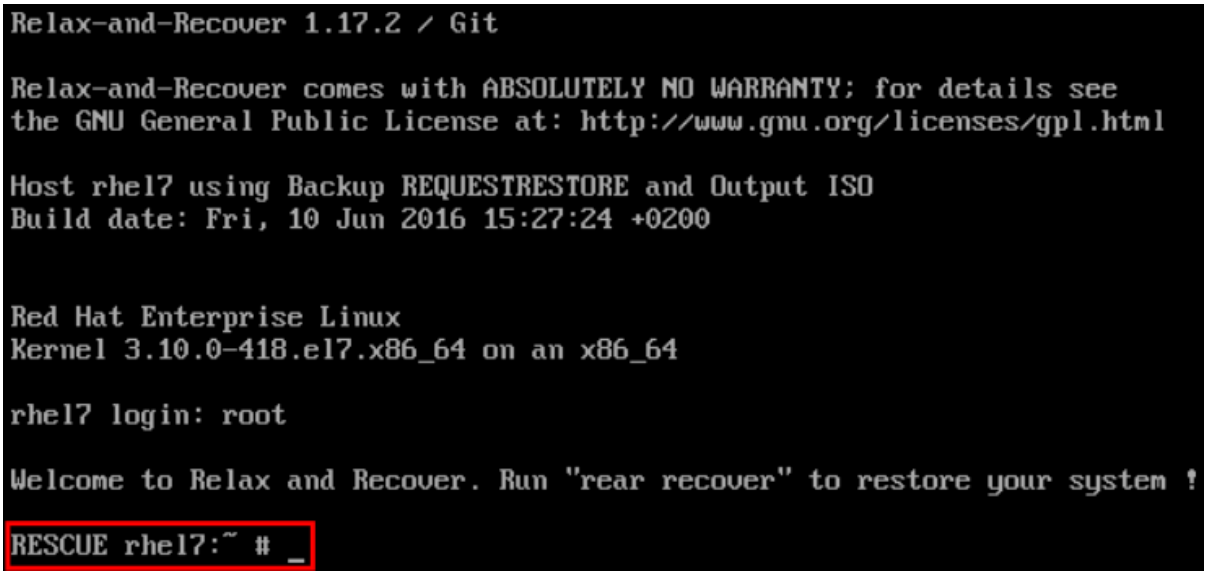
1. 新しいハードウェア上でレスキューシステムを起動します。たとえば、ISO イメージを DVD に書き込み、その DVD から起動します。
2. コンソールのインターフェイスで "Recover" オプションを選択します。

   図27.1 レスキューシステムのメニュー



3. *以下のプロンプトが表示されます。*

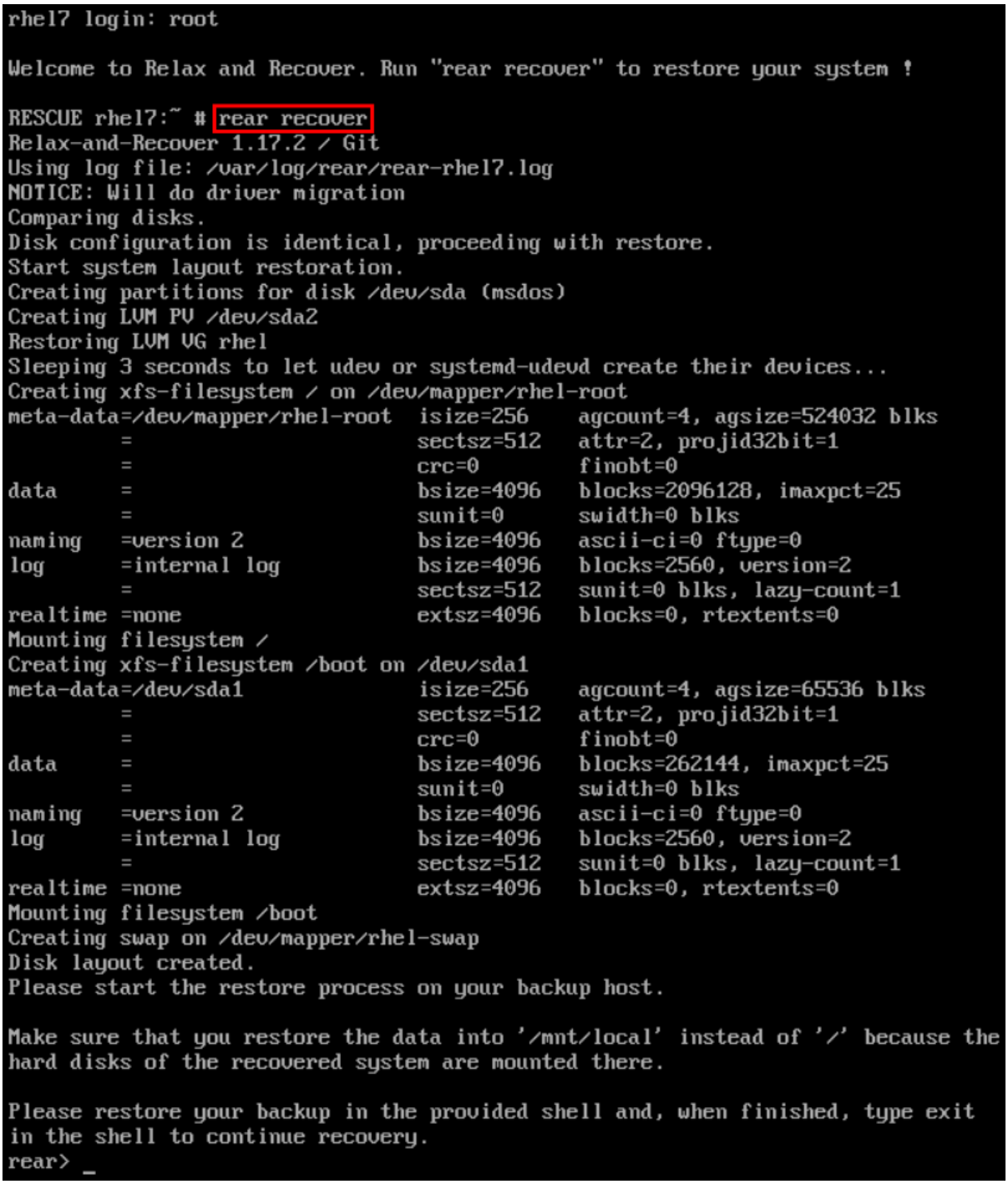
   図27.2 レスキューシステムのプロンプト



> ***警告***
>
> *次のステップでリカバリーを開始すると、元に戻すことができなくなり、システムの物理ディスクに保存されていたものが失われます。*

4. ***rear recover** コマンドを実行して復旧または移行を行います。するとレスキューシステムがパーティションレイアウトとファイルシステムを再作成します。*

図27.3 レスキューシステム: "rear recover" の実行



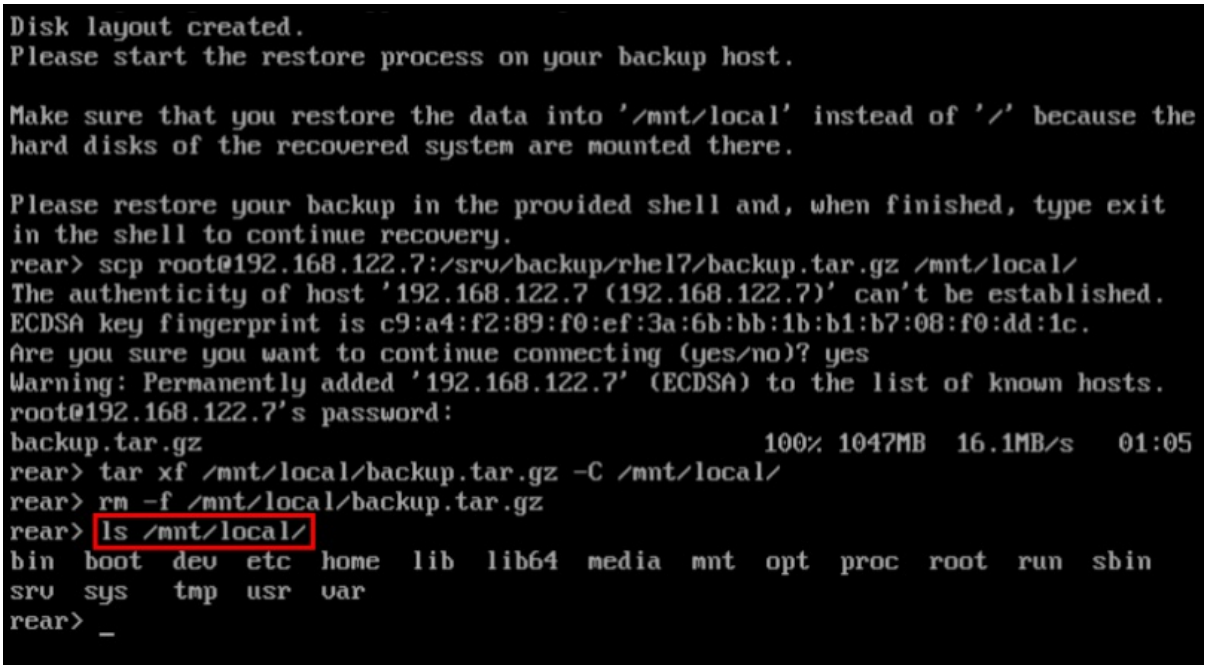
5. *バックアップから **/mnt/local/** ディレクトリーにユーザーおよびシステムファイルを復元します。*

> *例27.3 ユーザーおよびシステムファイルの復元*
>
> *この例では、バックアップファイルは、「内部バックアップメソッドの設定」の指示どおりに作成した tar アーカイブになります。まず、アーカイブをストレージからコピーして、ファイルを **/mnt/local/** にデプロイメントし、アーカイブを削除します。*
>
> ```
> ~]# scp root@192.168.122.7:/srv/backup/rhel7/backup.tar.gz /mnt/local/
> ~]# tar xf /mnt/local/backup.tar.gz -C /mnt/local/
> ~]# rm -f /mnt/local/backup.tar.gz
> ```
>
> *新規ストレージは、アーカイブとデプロイメントファイルの両方を格納できるサイズである必要があります。*

6. ファイルが復元されたことを確認します。

```
~]# ls /mnt/local/
```

図27.4 レスキューシステム: バックアップからのユーザーおよびシステムファイルの復元



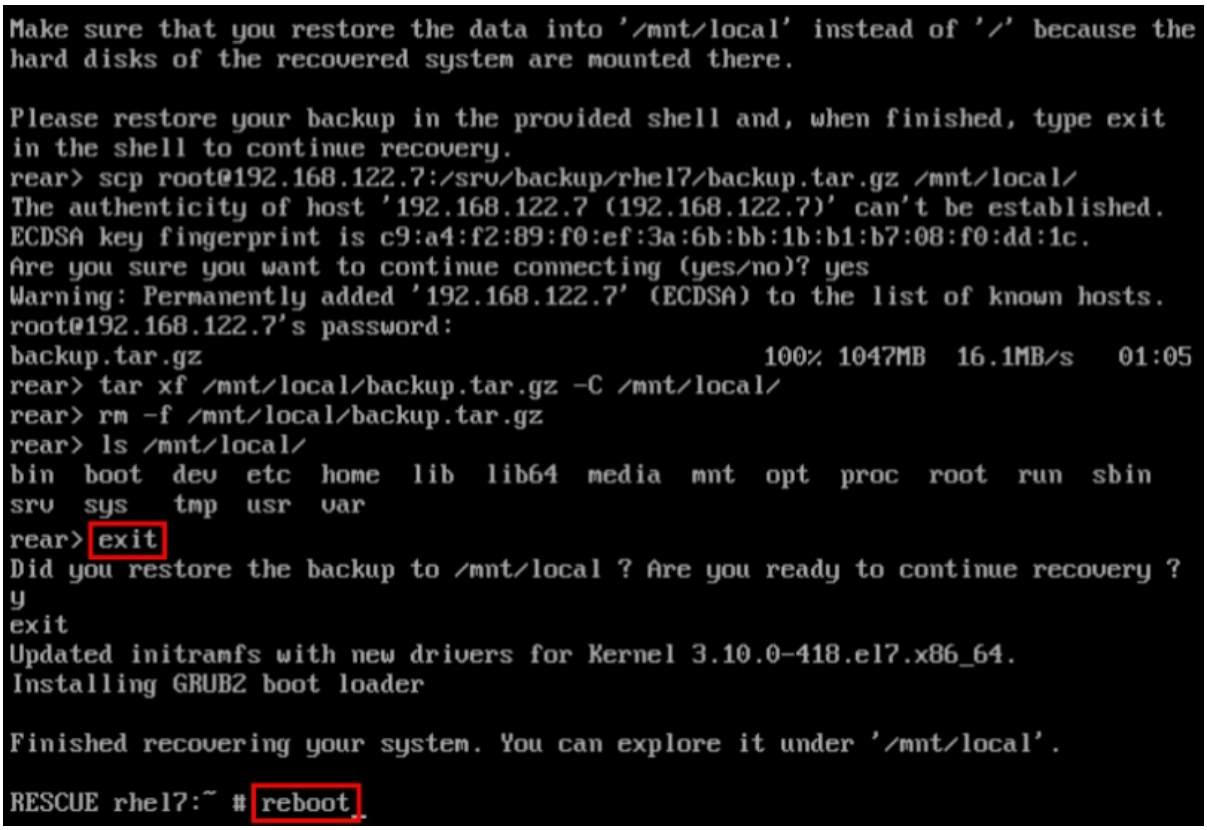
7. 次回の起動時に SELinux がファイルに再度ラベル付するようにします。

```
~]# touch /mnt/local/.autorelabel
```

これを実行しなかった場合、**/etc/passwd** ファイルの SELinux コンテキストが間違ったものとなり、システムにログインできなくなる可能性があります。

8. **exit** を実行してリカバリーを終了すると、ReaR がブートローダーを再インストールします。その後、ReaR はブートローダーを再インストールします。その後、システムを再起動します。

図27.5 レスキューシステム: リカバリーの終了



再起動すると、SELinux がファイルシステム全体に再ラベル付けされます。これでリカバリーしたシステムにログインできるようになります。

27.2. REAR をバックアップソフトウェアの統合

ReaR の主な目的はレスキューシステムを作成することですが、バックアップソフトウェアと統合することも可能です。統合は、ビルトイン、サポート対象、サポート対象外の各バックアップ方法で異なります。

27.2.1. ビルトインバックアップの場合

ReaR には、ビルトインもしくは内部のバックアップメソッドが含まれます。このメソッドは ReaR と完全に統合されており、以下の利点があります。

- ***rear mkbackup** コマンドを 1 つ使用して、レスキューシステムと完全システムバックアップを作成できます。*
- *レスキューシステムが自動でバックアップからファイルを復元します。*

このため、ReaR はレスキューシステムと完全システムバックアップの両方の作成プロセスを処理できます。

27.2.1.1. 内部バックアップメソッドの設定

*ReaR が内部バックアップメソッドを使用するようにするには、以下の行を **/etc/rear/local.conf** に追加します。*

```
BACKUP=NETFS
BACKUP_URL=backup location
```

これらの行によって、ReaR が **tar** コマンドを使用して完全システムバックアップのあるアーカイブを作成するようになります。backup location を、*man* ページの *rear(8)* の *Backup Software Integration* セクションにあるいずれかのオプションで置き換えます。バックアップの場所に十分な空き領域があるようにしてください。

> 例27.4 tar バックアップの追加
>
> 「基本的な ReaR の使用方法」 にある例を拡大して、ReaR が tar 完全システムバックアップを **/srv/backup/** ディレクトリーに出力するようにします。
>
> ```
> OUTPUT=ISO
> OUTPUT_URL=file:///mnt/rescue_system/
> BACKUP=NETFS
> BACKUP_URL=file:///srv/backup/
> ```

内部バックアップメソッドでは、さらなる設定が可能です。

- 新規バックアップの作成時にこれまでのバックアップアーカイブを維持しておくようにするには、以下の行を追加します。

  ```
  NETFS_KEEP_OLD_BACKUP_COPY=y
  ```

- デフォルトでは、ReaR は実行時に毎回、完全バックアップを作成します。変更分のみをバックアップする増分にするには、以下の行を追加します。

  ```
  BACKUP_TYPE=incremental
  ```

  これで **NETFS_KEEP_OLD_BACKUP_COPY** が自動的に **y** に設定されます。

- 増分バックアップに加えて、完全バックアップを定期的に実行するには、以下の行を追加します。

  ```
  FULLBACKUPDAY="Day"
  ```

  "Day" を "Mon"、"Tue"、"Wed"、"Thu"、"Fri"、"Sat"、"Sun" のいずれかに置き換えます。金、土、日。

- ReaR は、レスキューシステムとバックアップの両方を ISO イメージに含めることもできます。これを行うには、**BACKUP_URL** ディレクティブを **iso:///backup/** に設定します。

  ```
  BACKUP_URL=iso:///backup/
  ```

  これはレスキューシステムがリカバリー中にバックアップをフェッチする必要がないことから、完全システムバックアップの一番簡単なメソッドになります。ただし、ストレージに十分なスペースが必要になります。また、単発の ISO バックアップは増分とすることができません。

  > 例27.5 単一 ISO のレスキューシステムおよびバックアップの設定
  >
  > 以下の設定では、単一の ISO イメージとしてレスキューシステムとバックアップファイルが **/srv/backup/** ディレクトリーに作成されます。
  >
  > ```
  > OUTPUT=ISO
  > ```

```
OUTPUT_URL=file:///srv/backup/
BACKUP=NETFS
BACKUP_URL=iso:///backup/
```

注記

このシナリオでは、ISO イメージが大きくなる可能性があります。そのため、Red Hat は ISO イメージを 1 つだけ作成することを推奨しています。詳細は、「ISO 固有の設定」 を参照してください。

- **tar** ではなく **rsync** を使用する場合は、以下の行を追加します。

```
BACKUP_PROG=rsync
```

増分バックアップは **tar** 使用時にのみサポートされることに注意してください。

27.2.1.2. 内部バックアップメソッドを使用したバックアップの作成

BACKUP=NETFS を設定すると、ReaR は、レスキューシステムまたはバックアップのいずれか、もしくはその両方を作成できます。

- **レスキューシステムのみ** を作成するには、以下のコマンドを実行します。

```
rear mkrescue
```

- **バックアップのみ** を作成するには、以下のコマンドを実行します。

```
rear mkbackuponly
```

- **レスキューシステムとバックアップ** を作成するには、以下のコマンドを実行します。

```
rear mkbackup
```

ReaR によるバックアップの作成は、NETFS メソッドの使用時のみ可能となります。ReaR は他のバックアップメソッドを開始することはできません。

注記

復元時には、**BACKUP=NETFS** 設定で作成したレスキューシステムは、**rear recover** の実行前にバックアップが存在することを前提としています。このため、レスキューシステムが起動したら、**BACKUP_URL** で指定したディレクトリーにバックアップファイルをコピーします (単一 ISO イメージ使用時を除く)。この作業を終えてから、**rear recover** を実行してください。

不要なレスキューシステムを再作成しないためには、最後にレスキューシステムが作成されてからストレージレイアウトが変更されたかどうかを確認します。以下のコマンドを実行します。

```
~]# rear checklayout
~]# echo $?
```

セロ以外のステータスは、ディスクレイアウトに変更があったことを示します。また、ReaR 設定が変更した場合でも、ゼロ以外のステータスが返されます。

重要

rear checklayout コマンドはレスキューシステムがその時点で出力の場所にあるかどうかを確認せず、存在しない場合でも 0 を返す可能性があります。このため、レスキューシステムが利用可能であることを保証するのではなく、最後にレスキューシステムが作成されてからレイアウトに変更がないことのみが保証されます。

例27.6 rear checklayout の使用

レイアウトに変更があった場合にのみレスキューシステムを作成するようにするには、以下のコマンドを使用します。

```
~]# rear checklayout || rear mkrescue
```

27.2.2. サポート対象のバックアップメソッド

NETFS 内部バックアップメソッドのほかに、ReaR はいくつかの外部バックアップメソッドもサポートしています。この場合、レスキューシステムはバックアップから自動的にファイルを復元しますが、ReaR を使用してバックアップの作成を開始することはできません。

サポート対象の外部バックアップメソッドのリストおよび設定オプションは、rear(8) の man ページの Backup Software Integration セクションを参照してください。

27.2.3. サポート対象外のバックアップメソッド

サポート対象外のバックアップメソッドでは、以下の 2 つのオプションが可能です。

1. レスキューシステムでは、ユーザーに手動でファイルを復元するようプロンプトが出ます。このシナリオは基本的な ReaR の使用方法にあるものと同じですが、バックアップファイルの形式が tar アーカイブ以外のものである可能性があります。

2. ユーザーが提供するカスタムコマンドを ReaR が実行します。これを設定するには、**BACKUP** ディレクティブを **EXTERNAL** に設定します。それから、**EXTERNAL_BACKUP** と **EXTERNAL_RESTORE** のディレクティブを使用してバックアップおよび復元中に実行するコマンドを指定します。またオプションで、**EXTERNAL_IGNORE_ERRORS** と **EXTERNAL_CHECK** のディレクティブも指定します。設定例は、**/usr/share/rear/conf/default.conf** を参照してください。

27.2.4. 複数のバックアップの作成

バージョン 2.00 では、ReaR は複数のバックアップの作成をサポートしています。この機能をサポートするバックアップ手法は次のとおりです。

- **BACKUP=NETFS** (internal method)
- **BACKUP=BORG** (external method)

個々のバックアップは、**rear** コマンドの **-C** オプションで指定できます。引数は、**/etc/rear/** ディレクトリーにある追加のバックアップ設定ファイルのベースネームです。特定のバックアップごとのメソッド、バックアップ先、オプションはメインの設定ファイルではなく、特定の設定ファイルで定義されて

います。

システムの基本リカバリーを実行するには、以下を行います。

システムの基本リカバリー

1. ReaR リカバリーシステムの ISO イメージと基本システムのファイルのバックアップを一緒に作成します。

```
~]# rear -C basic_system mkbackup
```

2. /home ディレクトリーのファイルのバックアップを作成します。

```
~]# rear -C home_backup mkbackuponly
```

指定の設定ファイルに /boot、/root、/usr など、システムの基本リカバリーに必要なディレクトリーが含まれている必要があります。

rear リカバリーシェルでのシステムのリカバリー

rear リカバリーシェルでシステムをリカバリーするには、以下のコマンドのシーケンスを使用します。

```
~]# rear -C basic_system recover
```

```
~]# rear -C home_backup restoreonly
```

第28章 最適な RED HAT 製品の選択

Red Hat Cloud Infrastructure または Red Hat Cloud Suite のサブスクリプションは、様々な Red Hat 製品へのアクセスを補完機能一式とともに提供しています。

お客様の組織や使用事例に最適な製品を見つけるには、Cloud Deployment Planner (CDP) をご利用ください。CDP は、さまざまな製品リリースで、特定の相互運用性と機能互換性についての考慮事項をまとめたインタラクティブなツールです。

Red Hat Enterprise Linux のバージョンに応じて、特定製品の特定の機能および互換性のサポート性を比較するには、包括的な互換性マトリックス を参照してください。

第29章 システム管理に関連する RED HAT CUSTOMER PORTAL LAB

Red Hat Customer Portal Labs のツールは、パフォーマンスの向上、問題のトラブルシューティング、セキュリティー問題の特定、および設定の最適化を実行するユーザーをサポートするために設計されたツールです。この付録では、システム管理に関連する Red Hat カスタマーポータルの概要を解説します。すべての Red Hat Customer Portal Labs ツールは Customer Portal Labs から入手できます。

iSCSI Helper

iSCSI Helper は、インターネットプロトコル (IP) ネットワーク越しにブロックレベルのストレージを提供し、サーバーの仮想環境でストレージプールを使用できるようにします。

iSCSI Helper では、入力した設定に基づいて、iSCSI ターゲット (サーバー) または iSCSI イニシエーター (クライアント) のロールに合わせてシステムを設定するスクリプトを生成します。

NTP 設定

NTP (ネットワークタイムプロトコル) 設定を使用して、以下の設定を行います。

- NTP サービスを実行しているサーバー
- NTP サーバーと同期しているクライアント

Samba Configuration Helper

Samba Configuration Helper は、基本的なファイルと、Samba を通してプリンター共有を提供する設定を作成します。

- **Server** をクリックして、基本的なサーバー設定を指定します。
- **Shares** をクリックして、共有するディレクトリーを追加します。
- **Server** をクリックして、割り当てるプリンターを個別に追加します。

VNC Configurator

VNC Configurator は、VNC (仮想ネットワークコンピューティング) サーバーを Red Hat Enterprise Linux サーバーでインストールおよび設定するために設計されています。

VNC Configurator を使用して、VNC サービスを Red Hat Enterprise Linux サーバーでインストールおよび設定するように最適化された、オールインワンのスクリプトを生成します。

Bridge Configuration

Bridge Configuration は、Red Hat Enterprise Linux 5.4 以降を使用する KVM 等のアプリケーションに、ブリッジ接続のネットワークインターフェイスを設定するために作成されました。

Network Bonding Helper

管理者は、Network Bonding Helper を使用して、ボンディングカーネルモジュールおよびボンディングネットワークインターフェイスを使用して、複数のネットワークインターフェイスコントローラーを 1 つのチャンネルにまとめることができます。

Network Bonding Helper を使用すると、複数のネットワークインターフェイスを 1 つのボンディングインターフェイスとして機能させることができます。

LVM RAID Calculator

LVM RAID Calculator は、ストレージオプションを指定した後に、所定の RAID ストレージの論理ボリューム (LVM) を作成するのに最適なパラメーターを決定します。

LVM RAID Calculator を使用して、所定の RAID ストレージに LVM を作成する一連のコマンドを生成します。

NFS Helper

NFS Helper を使用すれば、新しい NFS サーバーやクライアントの設定が簡単に行えます。手順に従って、エクスポートとマウントのオプションを指定します。次に、ダウンロード可能な NFS 設定スクリプトを生成します。

Load Balancer Configuration Tool

Load Balancer Configuration Tool は、Apache をベースとするロードバランサーと JBoss/Tomcat アプリケーションサーバーとの間の最適なバランスを生成します。

Load Balancer Configuration Tool を使用して設定ファイルを作成し、お使いの環境のパフォーマンスを向上させる方法を提案します。

Yum Repository Configuration Helper

Yum Repository Configuration Helper は、単純な Yum リポジトリーを設定するために設計されています。

Yum Repository Configuration Helper を使用して以下の設定を行います。

- *ローカルの Yum リポジトリー*
- *HTTP/FTP ベースの Yum リポジトリー*

File System Layout Calculator

File System Layout Calculator は、ユーザーが、現在のストレージまたは予定されるストレージを記述したストレージオプションを提供した後に、ext3、ext4、xfs のファイルシステムの作成に最適なパラメーターを決定します。

File System Layout Calculator を使用して、指定された RAID ストレージに所定のパラメーターを持つファイルシステムを作成するコマンドを生成します。

RHEL Backup and Restore Assistant

RHEL Backup and Restore Assistant は、バックアップと復元のツールと、Linux における一般的なシナリオに関する情報を提供します。

対象ツール:

- ***dump** および **restore**: ext2、ext3、または ext4 のファイルシステムのバックアップ*
- ***tar** および **cpio**: たとえばテープドライブをバックアップする際に使用される、ファイルおよびフォルダーのアーカイブまたは復元*
- ***rsync**: バックアップ操作の実行と、ファイルとディレクトリーの同期*
- ***dd**: 関連するファイルシステムやオペレーティングシステムとは独立してブロックごとにファイルをコピーする場合。*

対象シナリオ:

- *障害回復*
- *ハードウェアの移行*
- *パーティションテーブルのバックアップ*

- 重要なフォルダーのバックアップ
- 増分バックアップ
- 差分バックアップ

DNS Helper
DNS Helper は、種類の異なる DNS サーバーの設定に、サポートを提供します。

DNS Helper は、DNS サーバーを自動的に作成および設定するための bash スクリプトを生成するために使用します。

AD Integration Helper (Samba FS - winbind)
AD Integration Helper は、Samba File System サーバーを Active Directory (AD) サーバーに接続するときに使用します。

AD Integration Helper を使用して、ユーザーが提供する基本的な AD サーバー情報を基にしたスクリプトを生成します。生成されたスクリプトは、Samba、Name Service Switch (NSS)、Pluggable Authentication Module (PAM) を設定します。

Red Hat Enterprise Linux Upgrade Helper
Red Hat Enterprise Linux Upgrade Helper は、Red Hat Enterprise Linux バージョン 6.5、6.6、6.7、または 6.8 をバージョン 7.x へアップグレードする際のサポートを提供するために作られています。

Registration Assistant
Registration Assistant は、お使いの Red Hat Enterprise Linux 環境に最適な登録オプションの選択をサポートします。

Rescue Mode Assistant
Rescue Mode Assistant は、Red Hat Enterprise Linux のレスキューモードで、以下の問題の解決をサポートします。

- root パスワードのリセット
- SOS レポートの生成
- ファイルシステムチェック (fsck) の実行
- GRUB の再インストール
- 初期 RAM ディスクイメージの再構築
- Root ファイルシステムのサイズ縮小

Kernel Oops Analyzer
Kernel Oops Analyzer は、カーネルクラッシュの解決をサポートするように作られています。

Kernel Oops Analyzer を使用して、1 つ以上のカーネル oops メッセージを含むテキストまたはファイルを入力し、ユーザーの事例に最適なソリューションを特定します。

Kdump ヘルパー
Kdump Helper は、Kdump メカニズムを設定するために作られています。

Kdump Helper を使用してスクリプトを生成し、メモリー内のデータを vmcore と呼ばれるダンプファイルへダンプするよう Kdump をセットアップします。

SCSI デコーダー

SCSI decoder は、SCSI エラーメッセージの理解を容易にするため、SCSI エラーメッセージを /log/* ファイルまたはログファイルのスニペットにデコードするように作られています。

SCSI decoder は、各 SCSI エラーメッセージを個別に診断し、問題を効果的に解決するためのソリューションを提示します。

Red Hat Memory Analyzer

Red Hat Memory Analyzer は、SAR ユーティリティーが取得した情報に基づき、ユーザーのシステムのメモリー使用量を視覚化します。

Multipath Helper

マルチパスヘルパー は、Red Hat Enterprise Linux 5、6、7 でマルチパスデバイスに最適な設定を作成します。

Multipath Helper を使用して、カスタムエイリアスやデバイスブラックリストといった高度なマルチパス設定を作成します。

また、Multipath Helper は、確認に使用する **multipath.conf** ファイルも提供します。必要な設定が終了したら、サーバーにインストールスクリプトをダウンロードして実行します。

Multipath Configuration Visualizer

Multipath Configuration Visualizer は SOS レポートにあるファイルを分析し、マルチパス設定を視覚化した図面を提供します。Multipath Configuration Visualizer を使用すると以下を表示できます。

- サーバーの HBA (Host Bus Adapters)、ローカルデバイス、および iSCSI デバイスを含むホストコンポーネント
- ストレージのストレージコンポーネント
- サーバーとストレージとの間のファブリック、またはイーサネットのコンポーネント
- 上記の全コンポーネントのパス

.xz、.gz、.bz2 のフォーマットに圧縮された SOS レポートをアップロードするか、クライアント側の分析のためにソースとして選択したディレクトリーに SOS レポートを抽出することができます。

Red Hat I/O Usage Visualizer

Red Hat I/O Usage Visualizer は、SAR ユーティリティーが取得した I/O デバイス使用統計を視覚化して表示します。

Storage/LVM Configuration Viewer

Storage / LVM configuration viewer は、SOS レポートに含まれるファイルを分析し、Storage/LVM 設定を視覚化した図面を作成します。

第30章 改訂履歴

0.14-26

2019 年 8 月 5 日 (月)、Marie Doleželová (mdolezel@redhat.com)

- 7.7 GA 公開用ドキュメントの準備

0.14-23

2018 年 8 月 13 日 (月)、Marie Doleželová (mdolezel@redhat.com)

- 7.6 ベータ版公開用ドキュメントの準備

0.14-19

2018 年 3 月 20 日 (火)、Marie Doleželová (mdolezel@redhat.com)

- 7.5 GA 公開用ドキュメントの準備

0.14-17

2017 年 12 月 5 日 (火)、Marie Doleželová (mdolezel@redhat.com)

- Samba のセクションを更新。TLS を使用した RELP の設定に関するセクションを追加。GRUB Legacy から GRUB2 へのアップグレードに関するセクションを更新。

0.14-16

2017 年 8 月 8 日 (月)、Marie Doleželová (mdolezel@redhat.com)

- 本ガイド全体に対する若干の修正。カスタムユニットファイルの作成の章で記事へのリンクを追加 (カスタムユニットファイルの並び順および依存関係のターゲット選択に関する記事へのリンク)。

0.14-14

2017 年 7 月 27 日 (木)、Marie Doleželová (mdolezel@redhat.com)

- 7.4 GA 公開用ドキュメントバージョン

0.14-8

2016 年 11 月 3 日 (月)、Maxim Svistunov (msvistun@redhat.com)

- 7.3 GA リリースのバージョン

0.14-7

2016 年 11 月 20 日 (月)、Maxim Svistunov (msvistun@redhat.com)

- Relax-and-Recover (ReaR) を追加。微修正。

0.14-6

2016 年 3 月 10 日 (木)、Maxim Svistunov (msvistun@redhat.com)

- 若干の更新

0.14-5

2016 年 1 月 21 日 (木) Lenka Špačková (lspackova@redhat.com)

- *若干の更新*

0.14-3

2015 年 11 月 11 日 (水) Jana Heves(jheves@redhat.com)

- *7.2 GA リリース向けのバージョン*

0.14-1

2015 年 11 月 9 日 (月) Jana Heves(jheves@redhat.com)

- *若干の修正が行われ、RH トレーニングコースへのリンクが追加されました。*

0.14-0.3

2015 年 4 月 3 日 (金) Stephen Wadeley (swadeley@redhat.com)

- **システム登録およびサブスクリプション管理** とRed Hat Support Tool を使用したサポートへのアクセス *が追加され、***ログファイルの表示と管理** *が更新されました。*

0.13-2

2015 年 2 月 24 日 (火) Stephen Wadeley (swadeley@redhat.com)

- *7.1 GA リリース向けバージョン*

0.12-0.6

2014 年 11 月 18 日 (火) Stephen Wadeley (swadeley@redhat.com)

- TigerVNC *の内容が改善されました。*

0.12-0.4

2014 年 11 月 10 日 (月) Stephen Wadeley (swadeley@redhat.com)

- Yum、systemd によるサービス管理、OpenLDAP、ログファイルの表示と管理、OProfile、 *および*GRUB 2 ブートローダーの操作 *の内容が改善されました。*

0.12-0

2014 年 8 月 19 日 (火) Stephen Wadeley (swadeley@redhat.com)

- *Red Hat Enterprise Linux 7.0 GA リリースのシステム管理者ガイド*

30.1. 承認

「Net-SNMP を使用したパフォーマンスのモニタリング」 は、Michael Solberg 氏が執筆した記事を基にしています。