Red Hat Enterprise Linux 7

システム管理者のガイド

Red Hat Enterprise Linux 7 の導入、設定、管理

Last Updated: 2018-10-04

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

Red Hat Enterprise Linux 7 の導入、設定、管理

Marie Doleželová
Red Hat Customer Content Services
mdolezel@redhat.com

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

Maxim Svistunov
Red Hat Customer Content Services

Stephen Wadeley
Red Hat Customer Content Services

Tomáš Čapek
Red Hat Customer Content Services

Jaromír Hradílek
Red Hat Customer Content Services

Douglas Silas
Red Hat Customer Content Services

Jana Heves
Red Hat Customer Content Services

Petr Kovář
Red Hat Customer Content Services

Peter Ondrejka
Red Hat Customer Content Services

Petr Bokoč
Red Hat Customer Content Services

Martin Prpič
Red Hat Product Security

Eliška Slobodová
Red Hat Customer Content Services

Eva Kopalová
Red Hat Customer Content Services

Miroslav Svoboda
Red Hat Customer Content Services

David O'Brien

法律上の通知

Copyright © 2018 Red Hat, Inc.
This document is licensed by Red Hat under the Creative Commons Attribution-ShareAlike 3.0 Unported License. If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

システム管理者のガイド は、Red Hat Enterprise Linux 7 の導入、設定、管理の関連情報について説明ます。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) を受講することをお勧めします。 Red Hat Enterprise Linux 7 を Linux Containers の機能と一緒に使用したいときは、Red Hat Enterprise Linux Atomic Host の製品ドキュメント をご覧ください。Linux Containers の一般概念、および Red Hat Enterprise Linux 7 に実装されているそれら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したトピックは、 『Red Hat Enterprise Linux Atomic Host 7 Managing Containers』 ガイドに詳しく説明されています。

目次

パート I. システムの基本設定 **7**

第1章 使用開始 **8**
COCKPIT と使用可能なタスクについて 8
1.1. 環境の基本設定 9
1.2. ネットワークアクセスの設定および検査 11
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 13
1.4. ソフトウェアのインストール 14
1.5. 起動時の SYSTEMD サービスの開始 16
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 17
1.7. ユーザーアカウント管理の基礎 22
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 23
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 25
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 26
1.11. RED HAT サポートへのアクセス 27

第2章 システムロケールおよびキーボード設定 **29**
2.1. システムロケールの設定 29
2.2. キーボードレイアウトの変更 32
2.3. 関連資料 33

第3章 日付と時刻の設定 **35**
3.1. TIMEDATECTL コマンドの使用 35
3.2. DATE コマンドの使用 38
3.3. HWCLOCK コマンドの使用 40
3.4. 関連資料 43

第4章 ユーザーとグループの管理 **44**
4.1. ユーザーとグループの概要 44
4.2. グラフィカル環境でのユーザーの管理 45
4.3. コマンドラインツールの使用 47
4.4. 関連資料 56

第5章 アクセス制御リスト **58**
5.1. ファイルシステムのマウント 58
5.2. アクセス ACL の設定 58
5.3. デフォルト ACL の設定 60
5.4. ACL の取り込み 60
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 61
5.6. 旧システムとの互換性 62
5.7. ACL 参照情報 62

第6章 権限の取得 **63**
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 63
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 64
6.3. 関連資料 65

パート II. サブスクリプションおよびサポート **67**

第7章 システム登録およびサブスクリプション管理 **68**
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 68
7.2. ソフトウェアリポジトリーの管理 69
7.3. サブスクリプションの削除 69
7.4. 関連資料 70

第8章 RED HAT SUPPORT TOOL を使用したサポートへのアクセス 71
8.1. RED HAT SUPPORT TOOL のインストール 71
8.2. コマンドラインを使用した RED HAT SUPPORT TOOL の登録 71
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 71
8.4. RED HAT SUPPORT TOOL の設定 71
8.5. インタラクティブモードでのサポートケースのオープンおよび更新 73
8.6. コマンドラインでのサポートケースの表示 75
8.7. 関連資料 75

パート III. ソフトウェアのインストールおよび管理 76

第9章 YUM 77
9.1. パッケージの確認と更新 77
9.2. パッケージでの作業 83
9.3. パッケージグループでの作業 92
9.4. トランザクション履歴の活用 95
9.5. YUM と YUM リポジトリーの設定 102
9.6. YUM のプラグイン 114
9.7. 関連資料 118

パート IV. インフラストラクチャーサービス 119

第10章 SYSTEMD によるサービス管理 120
10.1. SYSTEMD の概要 120
10.2. システムサービスの管理 123
10.3. SYSTEMD ターゲットでの作業 131
10.4. システムのシャットダウン、サスペンド、休止状態 136
10.5. リモートマシン上での SYSTEMD の制御 139
10.6. システムのユニットファイルの作成および変更 139
10.7. 関連資料 156

第11章 アクセシビリティーのためのシステム設定 159
11.1. BRLTTY サービスの設定 159
11.2. 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 163
11.3. FESTIVAL SPEECH SYNTHESIS SYSTEM の有効化 164

第12章 OPENSSH 166
12.1. SSH プロトコル 166
12.2. OPENSSH の設定 169
12.3. OPENSSH クライアント 178
12.4. セキュアシェルの高機能 181
12.5. 関連資料 183

第13章 TIGERVNC 185
13.1. VNC サーバー 185
13.2. 既存のデスクトップの起動 189
13.3. VNC ビューアー 190
13.4. 関連資料 193

パート V. サーバー関係 194

第14章 WEB サーバー 195
14.1. APACHE HTTP サーバー 195

第15章 メールサーバー 220

15.1. 電子メールプロトコル 220
15.2. 電子メールプログラムの分類 223
15.3. メール転送エージェント (MTA) 224
15.4. メール配信エージェント(MDA) 237
15.5. メールユーザーエージェント (MAIL USER AGENTS) 244
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 246
15.7. 関連資料 248

第16章 ファイルとプリントサーバー 250
16.1. SAMBA 250
16.2. FTP 308
16.3. 印刷設定 315

第17章 CHRONY スイートを使用した NTP 設定 335
17.1. CHRONY スイートの概要 335
17.2. CHRONY の概要および設定 337
17.3. CHRONY の使用 343
17.4. 異なる環境での CHRONY の設定 348
17.5. CHRONYC の使用 349
17.6. HW タイムスタンプを使った CHRONY 349
17.7. 関連資料 353

第18章 NTPD を使用した NTP 設定 354
18.1. NTP の概要 354
18.2. NTP STRATA (階層) 354
18.3. NTP の概要 355
18.4. 誤差ファイルの概要 356
18.5. UTC、タイムゾーン、および DST 356
18.6. NTP の認証オプション 357
18.7. 仮想マシン上での時間管理 357
18.8. うるう秒の概要 357
18.9. NTPD 設定ファイルの概要 358
18.10. NTPD SYSCONFIG ファイルの概要 359
18.11. CHRONY の無効化 360
18.12. NTP デーモンのインストールを確認する 360
18.13. NTP デーモン (NTPD) のインストール 360
18.14. NTP ステータスの確認 360
18.15. 着信 NTP パケットを許可するファイアウォールの設定 361
18.16. NTPDATE サーバーの設定 361
18.17. NTP の設定 362
18.18. ハードウェアクロック更新の設定 368
18.19. クロックソースの設定 370
18.20. 関連資料 370

第19章 PTP4L を使用した PTP の設定 372
19.1. PTP の概要 372
19.2. PTP の使用 374
19.3. 複数のインターフェースでの PTP の使用 377
19.4. 設定ファイルの指定 378
19.5. PTP 管理クライアントの使用 378
19.6. クロックの同期 379
19.7. 時間同期の検証 380
19.8. NTP を使った PTP 時間の実行 383
19.9. PTP を使った NTP 時間の実行 383

19.10. TIMEMASTER を使用した PTP または NTP 時間への同期 384
19.11. 精度の向上 388
19.12. 関連資料 388

パート VI. 監視と自動化 389

第20章 システムモニタリングツール 390
20.1. システムプロセスの表示 390
20.2. メモリ使用量の表示 393
20.3. CPU 使用率の表示 395
20.4. ブロックデバイスとファイルシステムの表示 395
20.5. ハードウェア情報の表示 400
20.6. ハードウェアエラーの確認 403
20.7. NET-SNMP を使用したパフォーマンスのモニタリング 404
20.8. 関連資料 418

第21章 OPENLMI 420
21.1. OPENLMI の概要 420
21.2. OPENLMI のインストール 421
21.3. OPENPEGASUS 用に SSL 証明書を設定する 423
21.4. LMISHELL の使用 427
21.5. OPENLMI スクリプトの使用 465
21.6. 関連資料 466

第22章 ログファイルの表示と管理 468
22.1. ログファイルの場所の特定 468
22.2. RSYSLOG の基本設定 468
22.3. 新規設定フォーマットの使用 484
22.4. RSYSLOG でのキュー (QUEUE) を使った操作 486
22.5. ロギングサーバーでの RSYSLOG の設定 496
22.6. RSYSLOG モジュールの使用 500
22.7. RSYSLOG と JOURNAL の相互作用 507
22.8. RSYSLOG での構造化ロギング 508
22.9. RSYSLOG のデバッグ 511
22.10. JOURNAL の使用 511
22.11. グラフィカル環境でのログファイルの管理 517
22.12. 関連資料 522

第23章 システムタスクの自動化 524
23.1. CRON を使用した繰り返しジョブのスケジュール設定 524
23.2. 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 527
23.3. 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設定 528
23.4. 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 531
23.5. 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 532
23.6. 関連資料 534

第24章 自動バグ報告ツール (ABRT) 535
24.1. ABRT の概要 535
24.2. ABRT のインストールとそのサービスの起動 535
24.3. ABRT の設定 538
24.4. ソフトウェア問題の検出 545
24.5. 検出された問題の処理 547
24.6. 関連資料 549

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 551

第25章 GRUB 2 での作業 552
25.1. GRUB 2 について 552
25.2. GRUB 2 の設定 553
25.3. GRUB 2 メニューの一時的な変更 553
25.4. GRUBBY ツールを使用した GRUB 2 メニューの永続的な変更 554
25.5. GRUB 2 設定ファイルのカスタマイズ 556
25.6. パスワードによる GRUB 2 の保護 561
25.7. GRUB 2 の再インストール 562
25.8. GRUB LEGACY から GRUB 2 へのアップグレード 564
25.9. シリアルコンソールでの GRUB 2 568
25.10. ブート中のターミナルメニューの編集 570
25.11. UNIFIED EXTENSIBLE FIRMWARE INTERFACE (UEFI) セキュアブート 575
25.12. 関連資料 576

パート VIII. システムバックアップおよびリカバリー 578

第26章 RELAX-AND-RECOVER (REAR) 579
26.1. 基本的な REAR の使用方法 579
26.2. REAR をバックアップソフトウェアの統合 585

付録A 最適な RED HAT 製品の選択 590

付録B システム管理に関連する RED HAT カスタマーポータルラボ 591
ISCSI HELPER 591
NTP 設定 591
SAMBA CONFIGURATION HELPER 591
VNC CONFIGURATOR 591
BRIDGE CONFIGURATION 591
NETWORK BONDING HELPER 591
LVM RAID CALCULATOR 591
NFS HELPER 592
LOAD BALANCER CONFIGURATION TOOL 592
YUM REPOSITORY CONFIGURATION HELPER 592
FILE SYSTEM LAYOUT CALCULATOR 592
RHEL BACKUP AND RESTORE ASSISTANT 592
DNS HELPER 593
AD INTEGRATION HELPER (SAMBA FS - WINBIND) 593
RED HAT ENTERPRISE LINUX UPGRADE HELPER 593
REGISTRATION ASSISTANT 593
RESCUE MODE ASSISTANT 593
KERNEL OOPS ANALYZER 593
KDUMP HELPER 593
SCSI DECODER 594
RED HAT MEMORY ANALYZER 594
MULTIPATH HELPER 594
MULTIPATH CONFIGURATION VISUALIZER 594
RED HAT I/O USAGE VISUALIZER 594
STORAGE/LVM CONFIGURATION VIEWER 594

付録C 改訂履歴 595

索引 596

パート I. システムの基本設定

ここでは、キーボード設定、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、基本的なインストール後の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 使用開始

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。このようなタスクを扱う本章の各セクションでは、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ています。

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ています。`root` ユーザーとして入力が必要なコマンドプロンプトには `#` がついており、一般ユーザーが実行可能なコマンドプロンプトには `$` 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。

インストール後のすべてのタスクはコマンドラインから実行できますが、一部のコマンドは **Cockpit** ツールから実行することもできます。

COCKPIT と使用可能なタスクについて

Cockpit はシステム管理ツールで、Web ブラウザーを通じ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ースを提供します。

Cockpit は、これらのタスクの実行を可能にします。

- ハードウェア、インターネット接続、または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を分析
- インターフェース、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モニタリングおよび設定
- 診断レポートの作成
- カーネルダンプ設定のセットアップ
- 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

Cockpit のインストールおよび使用に関する詳細は『Red Hat Enterprise Linux 7 Getting Started with Cockpit Guide』 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、数々の理由で重要です。Red Hat Enterprise Linux 7 では、**NTP** プロトコルにより、時間の正確さが確認に維持されます。このプロトコルは、ユーザースペースで実行されるデーモンにより実装されます。ユーザースペースのデーモンが、カーネルで実行中のシステムクロックを更新します。システムクロックは様々なクロックソースを使用することで、時間を維持することが可能となります。

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** を使用した **NTP** の設定および使用に関する詳細は 17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 18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** を無効化し、18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化し、設定する必要があります。

現在の日時の表示
現在の日時を表示するには、以下のいずれかのコマンドのを使用します。

- ```
  ~]$ date
  ```
  ```

- ```
  ~]$ timedatectl
  ```

  timedatectl コマンドは、より詳細な出力を提供します。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) 設定の状態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、**systemd** デーモンが初期ブート時に読み取ります。**/etc/locale.conf** で設定されたロケール設定は、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の設定を一覧表示

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

- システムロケールの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の設定または変更

  ```
  ~]# 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について

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一覧表示

  ```
  ~]$ 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ー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 **nmtui** テキスト形式のユーザーインターフェースユーティリティーのいずれかを使用できます。

nmcli ユーティリティー と **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、既存の接続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のグラフィカルユーザーインターフェースの Installation Summary 画面での **`Network & Hostname`** メニュー
- **Anaconda** インストールプログラムのテキストモードにおける **`Network settings`** 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ー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。

新規の接続を作成するには、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
```

特定の接続のすべての設定を表示するには、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「NetworkManager コマンドラインツール (nmcli) の使用」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 のテキスト形式のユーザーインターフェース (TUI) ユーティリティーである **nmtui** は、**NetworkManager** を制御するテキストインターフェースを提供してネットワークを設定します。

テキスト形式のインターフェースツールである **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. Cockpit でのネットワーキング管理

Cockpit の **Networking** メニューを使用すると、以下を実行できます。

- 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ー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ースの様々なタイプ (ボンディング、チーム、ブリッジ、VLAN) の追加

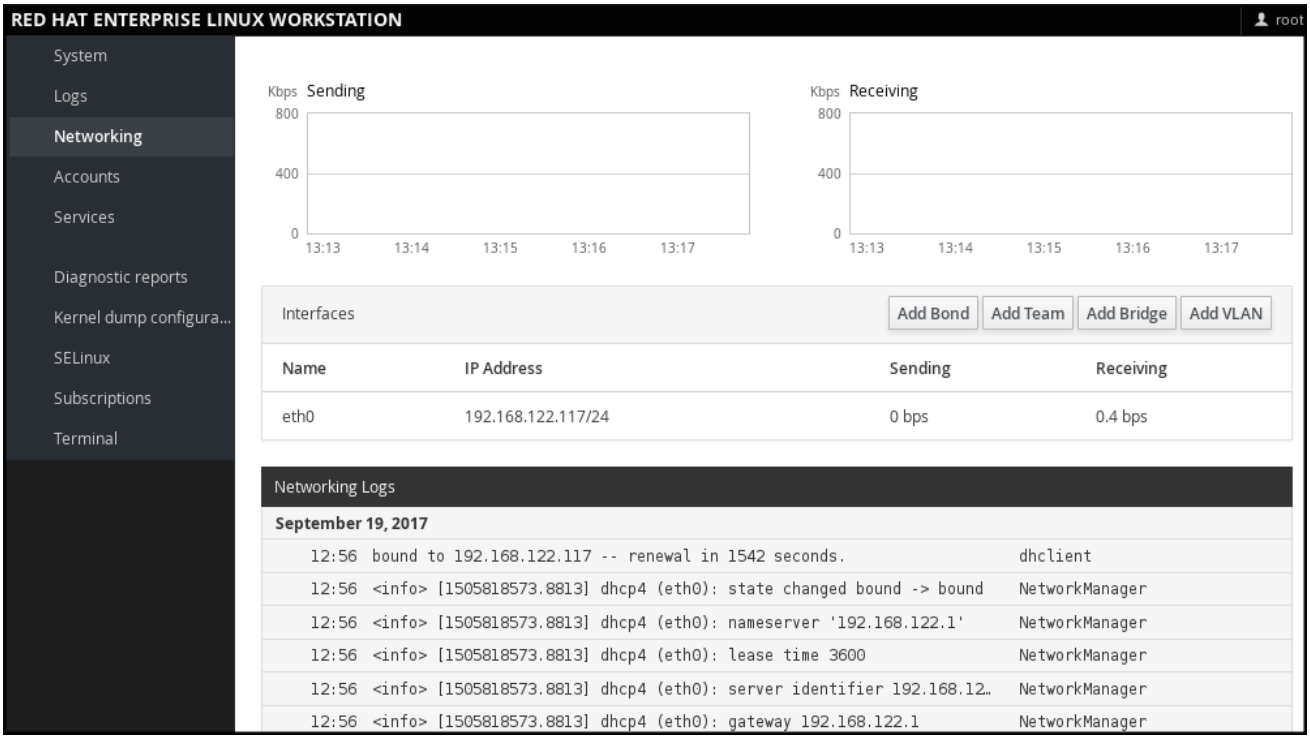


図1.1 Cockpit でのネットワーキング管理

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と使用可能なタスクについて

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) の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. インストール中のシステムの登録

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、インストール中に何が抜けていたのかを見つけることができます。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。

インストール中にシステムを登録する方法は基本的に 2 つあります。

- 登録は通常、**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。
- もしくは、**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** 実行することです。この場合は、インストールの完了と同時に、そして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。これを確実にするには、Kickstart 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の詳細は、『Red Hat Enterprise Linux 7 Installation Guide』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。この手順におけるコマンドはすべて **root** 権限で実行する必要がある点に注意してください。

手順1.1 システムの登録およびサブスクリプション

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 を決定します。

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

   このコマンドは、ご利用中の Red Hat アカウントで利用可能なすべてのサブスクリプションを表示します。サブスクリプションごとに、プール ID を含む様々な情報が表示されます。

3. *pool_id* を、前の手順で決定されたプール ID に置き換えることで、システムに適切なサブスクリプションを割り当てます。

```
~]# 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に関する詳細は、*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。

1.4. 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可能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について」 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と Yum を使用した基本的なソフトウェアインストールタスクの管理」 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、お使いの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について

Red Hat Enterprise Linux システム上のすべてのソフトウェアは、RPM パッケージに分類されます。これらの RPM パッケージは、特定のリポジトリーに保存されます。 システムが Red Hat のコンテンツ配信ネットワークにサブスクライブされると、**/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一覧の確認
- リポジトリーの追加または削除
- リポジトリーの有効化/無効化

ソフトウェアのインストールに関する基本タスクの詳細は、「サブスクリプションマネージャーと Yum を使用した基本的なソフトウェアインストールタスクの管理」 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と Yum を使用した基本的なソフトウェアインストールタスクの管理

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場合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の一覧を表示します。

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

- 現在有効になっているリポジトリーの一覧を表示します。

```
~]$ 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すべてのパッケージの情報を一覧表示します。

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

- インストール済みパッケージの情報すべて一覧表示します。

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

1.5. 起動時の SYSTEMD サービスの開始

Systemd は、systemd units という概念を導入する Linux オペレーティングシステムのシステムおよびサービスマネージャーです。systemd に関する詳細は 「systemd の概要」 を参照してください。

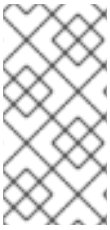
本セクションでは、システムの起動時にサービスを有効または無効にする方法を説明します。**Cockpit** を使用したサービスの管理方法についても説明します。

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決定できます。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の一覧を作成するには、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

無効なサービスの一覧は、有効なサービスの一覧より先に処理されます。したがって、両方の一覧ににあるサービスは有効となります。サービスの一覧は、コンマ区切りの形式で指定する必要があります。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。

インストール済みの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. Cockpit でのサービス管理

systemd 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、**Cockpit** で **Services** を選択します。ここで状態を確認するほか、開始/停止、有効化/無効化を設定できます。

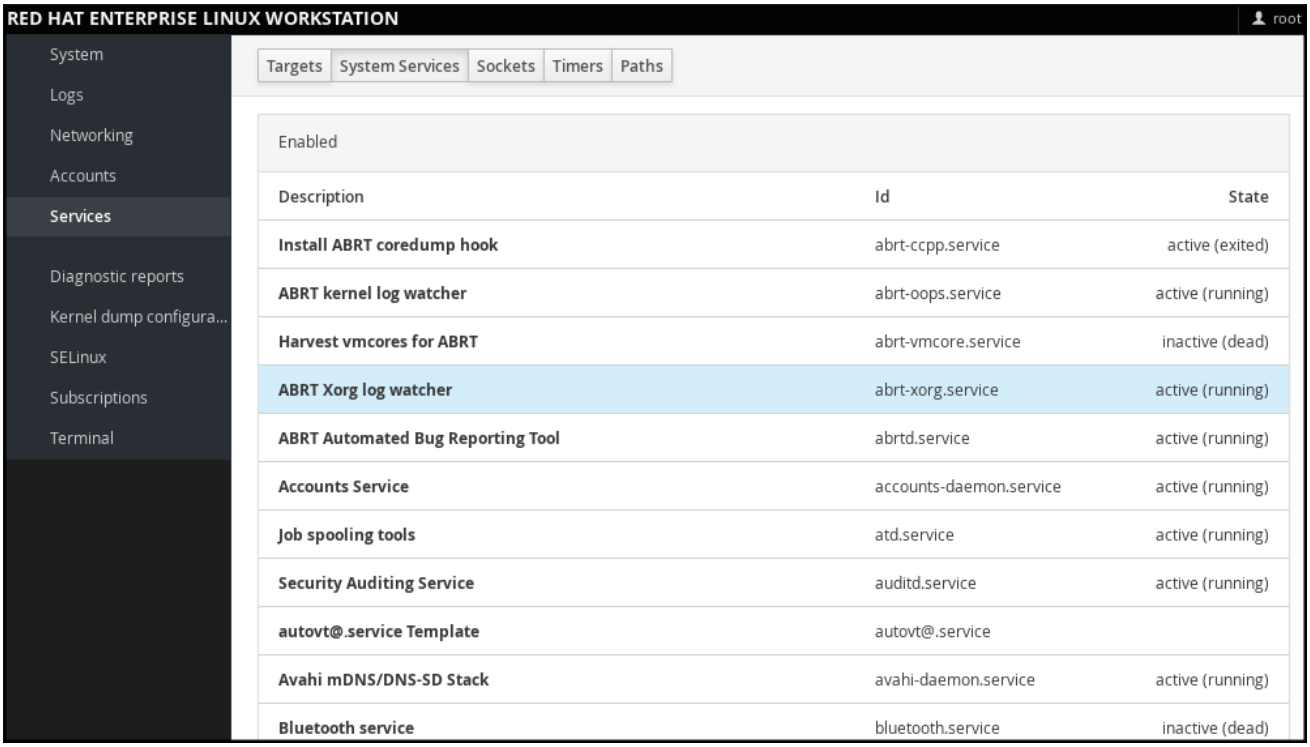


図1.2 Cockpit でのサービス管理

1.5.3. systemd サービスのその他のリソース

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は、ハードウェア、ソフトウェア、または情報を盗難やダメージから、そして提供するサービスの混乱や誤りからコンピューターシステムを保護します。したがって、コンピューターセキュリティーの確保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セキュリティー機能の基本のみを説明します。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. ファイアウォールの有効化と実行を確認

1.6.1.1. ファイアウォールとシステムセキュリティーの強化方法について

ファイアウォールは、既定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、信頼済みの安全な内部ネットワークと、別の外部ネットワークとの間にバリヤーを築きます。

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。キックスタートファイにおけるの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** サービスを無効にした場合、Red Hat では再有効化を検討するよう推奨しています。

firewalld の現在のステータスは、通常ユーザーでも確認できます。

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

firewalld が有効になっておらず実行していない場合は、**root** ユーザーに切り替えて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 とシステムセキュリティーの強化方法について

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決定するシステムセキュリティーの追加レイヤーです。

SELinux の状態
SELinux は、2 つの状態が可能です。

- 有効
- 無効

SELinux を無効して場合は、 任意アクセス制御 (DAC) のルールのみが使用されます。

SELinux モード
SELinux を有効にした場合は、以下のモードのいずれかで実行可能です。

- 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くアクセスを拒否し、特別に許可された対話のみを有効にします。enforcing モードは、インストール後のデフォルトモードであり、最も安全な **SELinux** モードでもあります。

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す。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 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。

しかし、**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可能性もあります。これは、たとえばキックスタート設定でこれを指定するできます。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、お使いのシステムを enforcing モードに保つことを推奨します。

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。

手順1.2 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、システムを再起動時すると設定が元に戻りますが、永続的に切り替えると再起動後も持続します。

   - enforcing モードまたは permissive モードのいずれかに一時的に切り替えるには、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** 設定ファイルの *SELINUX* の変数を変更します。

     たとえば、**SELinux** を enforcing モードに切り替えるには、以下を実行します。

```
# 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. Cockpit での SELinux 管理

Cockpit では、**SELinux** オプションを使用して **SELinux** の enforcing ポリシーをオンまたはオフにします。

デフォルトでは、**Cockpit** における **SELinux** の enforcing ポリシーはオンのため、**SELinux** は enforcing モードで動作します。これをオフにすると、**SELinux** を permissive モードに切り替えることができます。ただし、このように /etc/sysconfig/selinux ファイルのデフォルト設定を変更しても、次回のシステム起動時に設定が自動的に元に戻る点に注意してください。

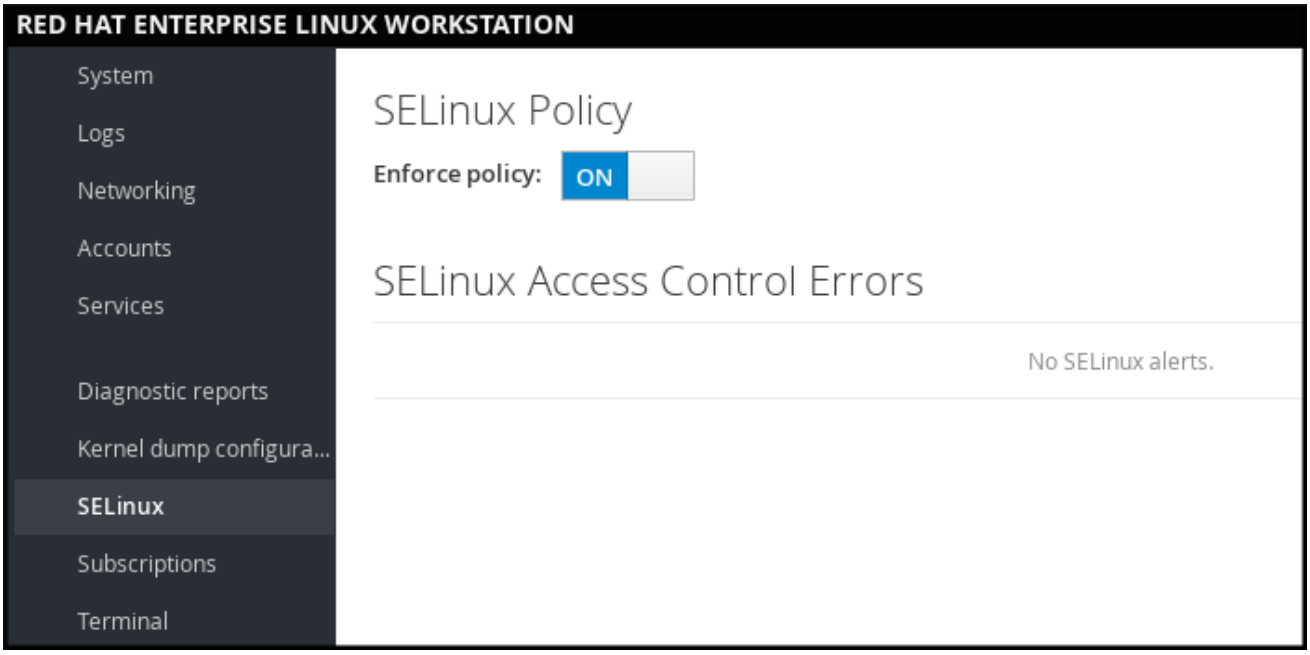


図1.3 Cockpit での SELinux 管理

1.6.3. SSH ベース認証の使用

1.6.3.1. SSH ベースの認証とシステムセキュリティーの強化方法について

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することが可能です。

SSH (Secure Shell: セキュアシェル) は、クライアントとサーバーとの間の通信を容易にし、ユーザーが SSH を実行するホストシステムに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。そのためには、SSH は公開/秘密鍵スキームを使用します。

SSH の保護手段に関する詳細は、「主な機能」 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成る 2 組みの鍵を作成します。

手順1.3 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
```

鍵は両方とも **~/.ssh/** ディレクトリーに保存されます。

- **~/.ssh/id_rsa.pub** - public key
- **~/.ssh/id_rsa** - private key

公開鍵は秘密である必要はありません。これは、秘密鍵の確認に使用されます。秘密鍵は秘密となります。秘密鍵は、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、今後はパスワードが毎回必要になります。これを回避するには、**ssh-agent** コマンドを利用します。この場合、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. 最近変更された公開鍵をログインしたい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

デフォルトで有効となっている **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。

手順1.4 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へ変更します。

```
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 つのマシンにインストールされた 1 つのシステムに対し、複数のユーザーが別々のコンピューターからアクセスすることが可能となります。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を表しています。

グループと使用可能な目的について

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。

- ユーザーおよびグループ ID の表示

```
~]$ id
```

- 新規のユーザーアカウントの作成

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

- *username* に属するユーザーアカウントへの新規パスワードの割り当て

```
~]# passwd user_name
```

- ユーザーのグループへの追加

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

ユーザーおよびグループの管理についての詳細は 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ース) 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. Cockpit におけるユーザーアカウントの管理

Cockpit のアカウントを管理するには、**Accounts** メニューを選択します。

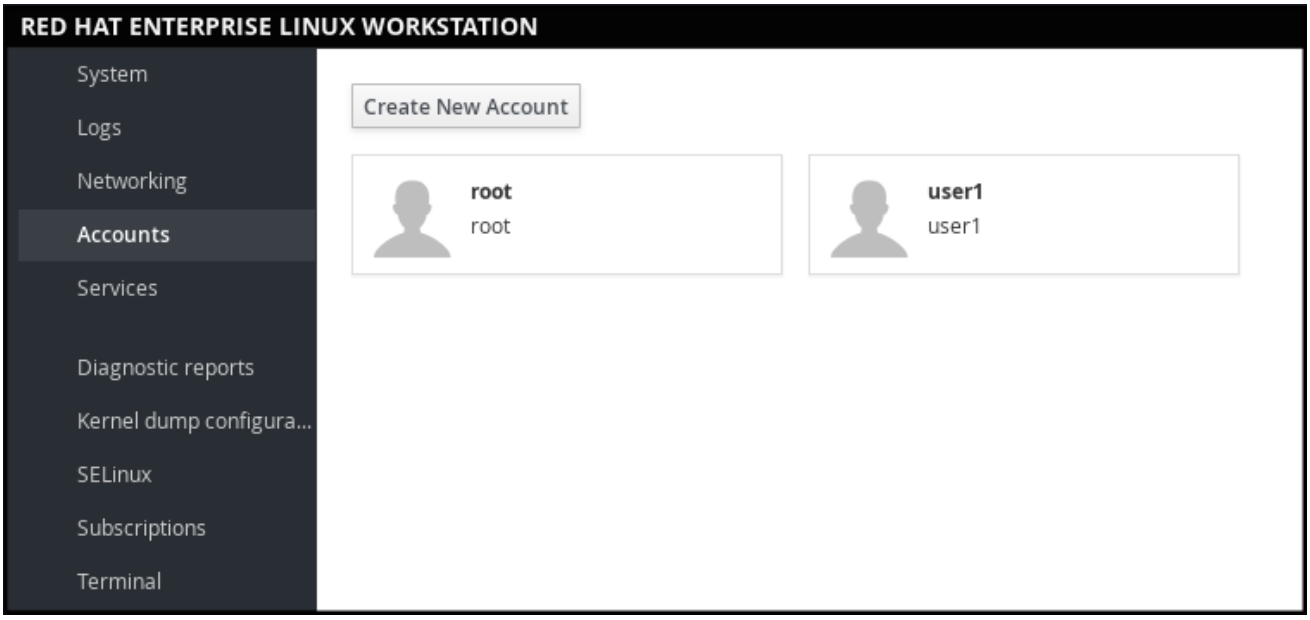


図1.4 Cockpit におけるユーザーアカウントの管理

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

本セクションでは、カーネルクラッシュダンプ (別名 **kdump**) のメカニズムを紹介します。また、**kdump** の使用方法は 「kdump と使用可能なタスクについて」 で簡単に説明します。

kdump サービスの有効化はインストールプロセスで行い、デフォルトではインストール中に **kdump** が有効化になります。本セクションでは、インストール時に **kdump** を有効にする方法を 「インストールプロセス中の kdump の有効化および実行」 で説明し、また、無効の **kdump** サービスをインストール後に手動で有効にする方法を 「インストールプロセス後の kdump のインストールと有効化の確認」 にまとめています。

Cockpit を使用して **kdump** を設定することも可能です。詳細は 「Cockpit での kdump の設定」 を参照してください。

1.8.1. kdump と使用可能なタスクについて

システムがクラッシュした場合、**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することができます。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可能となります。**kdump** のメカニズムは、 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 kexec システムコールに依存しています。

カーネルクラッシュが発生すると、**kdump** は kexec を使用して 2 番目のカーネルで起動します (キャプチャーカーネル)。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 番目のカーネルは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の kdump のインストールと有効化の確認

kdump のインストールを確認し、設定するには以下を実行します。

手順1.5 kdump がインストールされたかどうかを確認し、kdump を設定

1. お使いのシステムに **kdump** がインストールされたかどうかを確認するには、以下を実行します。

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

2. インストールされていない場合に **kdump** をインストールするには、**root** ユーザーとして以下のコマンドを実行します。

   ```
   ~]# 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ー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. Cockpit での kdump の設定

Cockpit で **カーネルダンプ設定** を選択し、以下を確認します。

- **kdump** のステータス
- **kdump** 用に確保しているメモリー容量
- クラッシュダンプファイルの場所

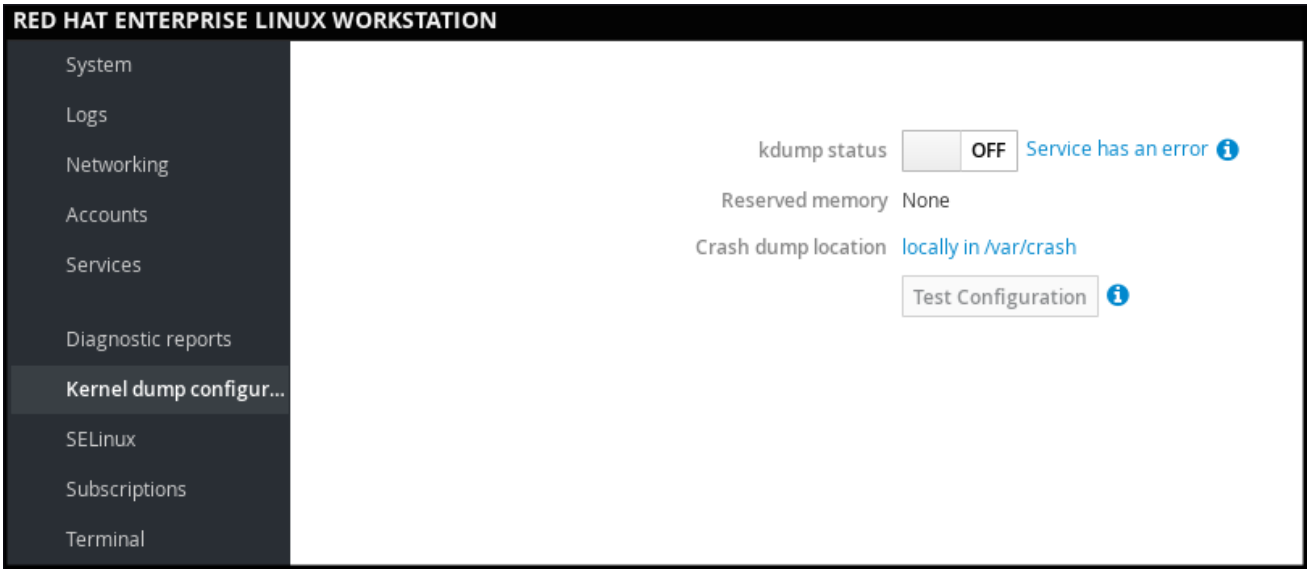


図1.5 Cockpit での kdump の設定

1.8.5. kdump に関するその他のリソース

kdump に関する詳細は 『Red Hat Enterprise Linux 7 カーネル管理ガイド』 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、システムをレスキューするためのメカニズムが必要です。システムのバックアップが保存されていることも役に立ちます。Red Hat は、これら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 と使用可能なタスクについて

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。

さらに、バックアップソフトウェアにより、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** ユーザーとして以下のコマンドを実行します。

```
~]# yum install rear genisoimage syslinux
```

/etc/rear/local.conf ファイルを使用して 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** ユーザーとして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」 と 「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、オペレーティングシステムに関する様々な情報およびメッセージが含まれたログファイルが利用できます。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、ログファイルに整理します。これは、オペレーティングシステムの監査および様々な問題のトラブルシューティングの際に役立ちます。

ログファイルの詳細は、*22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 - syslog のメッセージをタイプおよび優先順に分類し、**/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、含まれるメッセージやログの種類に応じて **/var/log** ディレクトリー下の様々なサブディレクトリーに保存されます。

- **var/log/messages** - 以下で言及したもの以外のすべての 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とエラー
- **var/log/maillog** - メールサーバーに関連するメッセージとエラー
- **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 の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すべてのものを提供しています。

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に関しては、「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 では、以下をご利用いただけます。

- 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
- 電話または電子メールによる Red Hat 専門スタッフへの問い合わせ

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 と利用できるタスクについて

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ー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、Red Hat Enterprise Linux システムから設定の詳細情報、システム情報、および診断情報を収集します。サポートケースを開く際は、SOS レポートを添付してください。

SOS レポート は、sos パッケージで提供しています。このパッケージは、Red Hat Enterprise Linux 7 のデフォルトである最小限のインストールには含まれていません。

sos パッケージをインストールするには、以下を実行します。

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

SOS レポート を生成するには、以下を実行します。

```
~]# sosreport
```

サポートケース用に **SOS レポート** を添付する方法は、Red Hat ナレッジケースの記事 「Ret Hat サポートチームにファイル (vmcore、rhev logcollector、sosreport、ヒープダンプ、ログファイルなど) を送付する 」 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る点に注意してください。

SOS レポート に関する詳細は、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 の役割と取得方法」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ー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ー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ー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、**systemd** デーモンが初期ブート時に読み取ります。**/etc/locale.conf** で設定されたロケール設定は、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一覧です。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については、man ページ **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	24時間表記か12時間表記かという現在の時間表記を変更します。
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 コマンドを使うと、システムロケールとキーボードレイアウト設定に対してクエリーを行ったり変更することができます。現行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用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一覧表示

ご使用のシステムで利用可能なロケールを一覧表示するには、以下を入力します。

```
localectl list-locales
```

例2.2 ロケールの一覧表示

英語ロケールをさらに細かく選択したい場合に、それがシステムで利用可能かどうかを確認するには、以下のコマンドを実行すると英語ロケールの一覧が表示されます。

```
~]$ 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、まず **list-locales** を使ってこのロケール名を見つけます。その後に、**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. Kickstart 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 Kickstart のインストール方法を使用して 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。

Kickstart ファイルの **%packages** セクションに **--instLang** オプションが含まれていると、**_install_langs** RPM マクロでこの値が使用され、それに応じてインストール済みのロケールが設定されます。ただしこの設定は今回のインストールのみに影響し、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を含む、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。

- Kickstart のインストールをまだ開始していない場合は、手順2.1「Kickstart のインストール中に RPM マクロの設定」 の手順に従って Kickstart 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ときは、手順2.2「RPM マクロのグローバルへの設定」 の手順に従って RPM マクロをシステムにグローバルに設定します。

手順2.1 Kickstart のインストール中に RPM マクロの設定

- Kickstart 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

awk '(NF==0amp amp!done){print
"override_install_langs='$LANG'";done=1}{print}' \
     < /etc/yum.conf > /etc/yum.conf.new
mv /ec/yum.conf.new /etc/yum.conf
```

手順2.2 RPM マクロのグローバルへの設定

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ー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一覧表示

システムに設定可能なキーボードレイアウトの一覧を表示するには、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 を使うと、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は、複数あることがよくあります。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** コマンド出力の中から選択したキーマップ名に置き換えます。**--no-convert** オプションを指定しないと、この設定は X11 キーボードマッピングに最も適合するものに変換された後、X11 ウィンドウシステムのデフォルト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
```

X11 レイアウトにこのコンソールレイアウトを設定したくない場合は、**--no-convert** オプションを使います。

```
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ースでは 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たい場合は、**root** で以下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* と *options* はキーボードのバリアントと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、**kbd(4)** man ページを参照してください。

2.3. 関連資料

Red Hat Enterprise Linux でキーボードレイアウトを設定する詳細方法は、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** コマンドおよび **sudo** コマンドを使用して管理者権限を取得する方法が説明されています。
- 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* と呼ばれます。これは通常、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ています。
- *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と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。

Red Hat Enterprise Linux 7 では、以下の 3 つのコマンドラインツールでシステムの日付および時間に関する情報を設定および表示できます。**timedatectl** ユーティリティーは Red Hat Enterprise Linux 7 で初めて導入されており、**systemd** の一部となります。**date** は従来のコマンドです。**hwclock** ユーティリティーは、ハードウェアクロックにアクセスするためのものです。

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の確認および変更ができます。このツールを使うと、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. 現在の日時の表示

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、RTC 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 現在の日時の表示

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場合は、以下のコマンドを実行します。

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

3.1.2. 現在の時刻の変更

現在の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 現在の時刻の変更

現在の時刻を午後 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 を使う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** として、**timedatectl** コマンドに **set-local-rtc** オプションを付けて実行します。

```
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. 現在の日付の変更

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。

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

例3.3 現在の日付の変更

現在の日付を 2017 年 6 月 2 日に変更し、現在の時間 (午後 11:26) を維持するには、**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一覧を表示するには、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

ユーザーの地域に最も近い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、ヨーロッパで利用可能なタイムゾーンの一覧を表示するには、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ことができます。NTP を有効にすると、**chronyd** サービスまたは **ntpd** サービスのどちらか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と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、現在の日時の表示および設定を可能とします。カスタマイズされた形式でシステムクロックに関する詳細情報を表示するためにスクリプト内で使われることが多くあります。

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と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. 現在の日時の表示

現在の日時を表示するには、**date** コマンドをオプションなしで実行します。

```
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。

デフォルトでは、**date** コマンドはローカルタイムを表示します。UTC で時間を表示するには、このコマンドに **--utc** オプションまたは **-u** のオプションを付けて実行します。

```
date --utc
```

表示される情報のフォーマットをカスタマイズするには、**+"*format*"** オプションをコマンドラインに追加します。

```
date +"format"
```

例3.6「現在の日時の表示」 にあるように、*format* 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。よく使わ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。また、オプションの完全なリストは **date**(1) man ページ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	*YYYY-MM-DD* フォーマットでの完全な日付 (例: **2016-09-16**)。このオプションは **%Y-%m-%d** と同じです。
%T	*HH:MM:SS* フォーマットでの完全な時間 (例: 17:30:24)。このオプションは **%H:%M:%S** と同じです。

例3.6 現在の日時の表示

現在の日付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 で現在の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. 現在の時刻の変更

現在の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** オプションを付けて実行します。

```
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。

デフォルトでは、**date** コマンド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 に設定するには、このコマンドに **--utc** オプションまたは **-u** オプションを付けて実行します。

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

例3.7 現在の時刻の変更

現在の時刻を午後 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. 現在の日付の変更

現在の日付を変更するには、**root** で **date** コマンドを **--set** オプションまたは **-s** オプションと共に実行します。

```
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。

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

例3.8 現在の日付の変更

現在の日付を 2017 年 6 月 2 日に変更し、現在の時間 (午後 11:26) を維持するには、**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の時刻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体系的なずれを補正する機能もあります。

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

hwclock コマンドは、Red Hat Enterprise Linux 6 ではシステムがシャットダウンまたは再起動する際に自動的に実行されますが、Red Hat Enterprise Linux 7 では実行されません。システムクロックが Network Time Protocol (NTP) または Precision Time Protocol (PTP) で同期される場合、カーネルは自動的に 11分ごとにハードウェアクロックをシステムクロックに同期します。

NTP の詳細は、17章*chrony スイートを使用した NTP 設定* および 18章*ntpd を使用した NTP 設定* を参照してください。PTP の詳細は、19章*ptp4l を使用した PTP の設定* を参照してください。**ntpdate** を実行した後のハードウェアクロックの設定方法は、「ハードウェアクロック更新の設定」 を参照してください。

3.3.1. 現在の日時の表示

root ユーザーでオプションなしに **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** または **--localtime** オプションを使用しても、ハードウェアクロックタイムが UTC またはローカルタイムで表示されるわけではありません。これらのオプションは、ハードウェアクロックの設定をいずれか変更するために使用されます。また、**hwclock --utc** または **hwclock --local** コマンドを使用しても、**/etc/adjtime** ファイルの記録は変更しません。したがって、このコマンドは、**/etc/adjtime** に保存された設定が誤っていても、その設定を変更したくない場合には役に立つかもしれません。ただし、このコマンドを間違った方法で使用すると、情報を誤って理解してしまう可能性があります。詳細は man ページ **hwclock**(8) を参照してください。

例3.9 現在の日時の表示

ハードウェアクロックから現在の日付および現在の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の日付と時刻を表示するほかに、特定の時刻に手動で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** オプションおよび **--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** オプションまたは **--localtime** オプションをそれぞれ追加することで、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** として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現在のシステム時間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現在のシステム時間に設定できます。

  ```
  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際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、**--utc** または **--localtime** オプションを追加して、ハードウェアクロックをローカルタイムまたは UTC で維持するかどうかを指定することもできます。**--set** を使用する場合と同様に、**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** に機能的に似ており、**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現在のシステム時間に設定し、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、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される必要があります。

3.4. 関連資料

Red Hat Enterprise Linux 7 で日時を設定する詳細情報は、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

- **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* 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* は、**su** および **sudo** コマンドを使っ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* は、systemd に関する詳細情報と、**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を説明しています。

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ー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、削除する方法について説明し、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であり、グループは共通の目的でユーザーをまとめようとする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) をサポートしています。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、システムユーザーとグループ用に、1000 より小さい数字のユーザー ID とグループ ID が予約されています。**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。予備のユーザーおよびグループ 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、以下のコマンドを使用します。

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

予約に使用される ID の範囲は将来広がる可能性があるため、5,000 以降の番号を ID に割り当てることが推奨されます。新規ユーザーへの割り当て ID がデフォルトで 5,000 から始まるようにするには、**/etc/login.defs** ファイルの **UID_MIN** と **GID_MIN** のディレクティブを変更します。

```
 ファイル内容を省略
UID_MIN                  5000ファイル内容を省略
GID_MIN                  5000ファイル内容を省略
```

注記

UID_MIN ディレクティブおよび **GID_MIN** ディレクティブ変更前に作成されたユーザーの UID については、デフォルトの 1000 から始まります。

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度に作成されます。ユーザープライベートグループは作成されたユーザーと同じ名前を持ち、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、新規作成されたファイルやディレクトリーに対し安全にデフォルトのパーミッションを設定できます。また、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。

新しく作成された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、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ず、**作成者のグループメンバーなど**、他のユーザーは変更できませんでした。しかし、UPG スキームでは全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」は必須ではありません。詳細は 「**umask** を使用した新しいファイルのデフォルト権限の設定」 を参照してください。

グループの一覧は **/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はシャドウパスワードをデフォルト設定で有効にしています。

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。詳細については、『Red Hat Enterprise Linux 7 セキュリティーガイド』 の「パスワードのセキュリティー」を参照してください。
- **/etc/group** ファイルを管理する **gpasswd** ユーティリティー。
- **-e**、**--expiredate** または **-f**、**--inactive** オプションを使用した **usermod** コマンド。
- **-e**、**--expiredate** 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 ユーティリティーを使用すると、グラフィカルユーザーインターフェースでローカルユーザーを参照、変更、追加、および削除できます。

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。**Users** 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 または Command キーになります。また、画面の右上隅にある自分のユーザー名をクリックし、**Settings (設定)** メニューから **Users** 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、**Account Type (アカウントタイプ)** を **Standard (標準)** から **Administrator (管理者)** 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)。

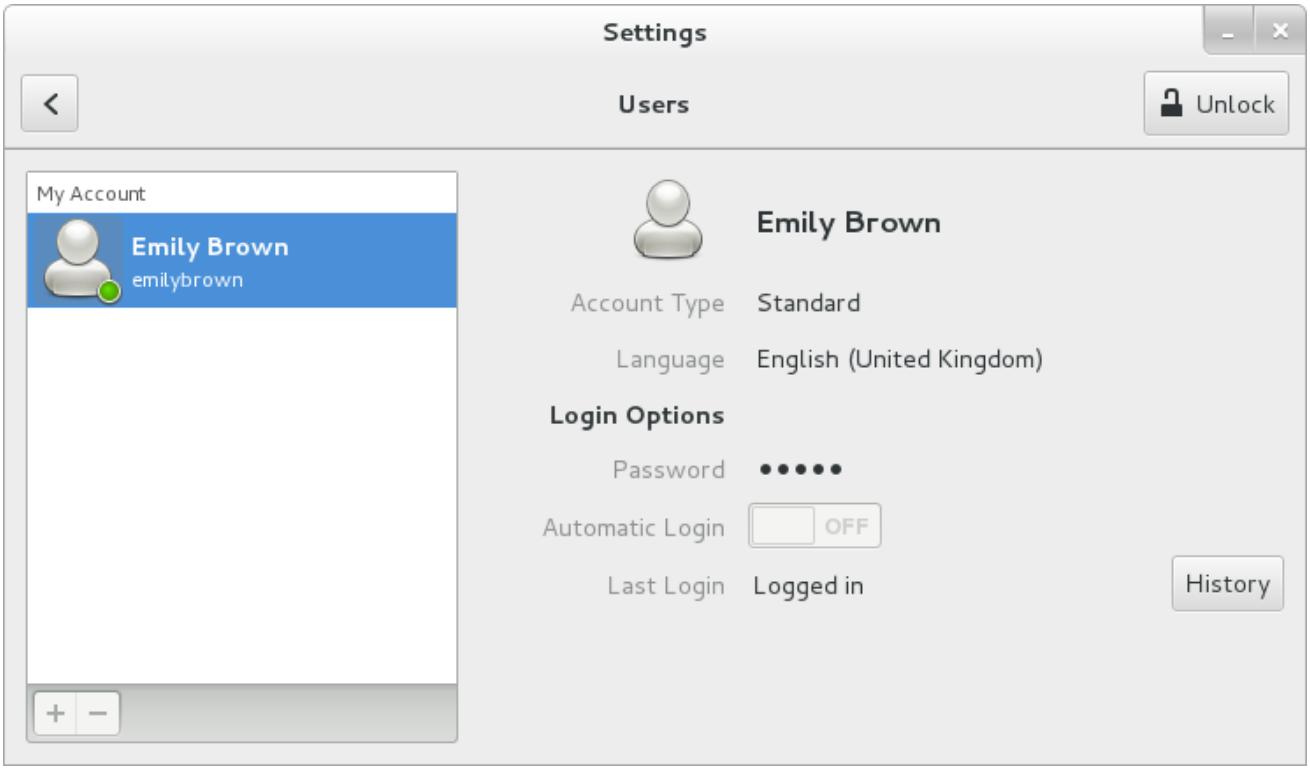


図4.1 ユーザー設定ツール

新しいユーザーが作成されたら、パスワードが設定され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	ユーザーおよびグループの 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	**newgrp** コマンドで使用する **/etc/gshadow** ファイル内のグループパスワードの修正に使用するユーティリティー。
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、または逆に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** としてシェルプロンプトに以下を入力します。

```
useradd [options] username
```

ここでの *options* は 表4.2「一般的な useradd コマンドラインオプション」 の説明にあ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。アカウントのロックを解除するには、**root** として以下のコマンドを実行し、パスワードを割り当てます。

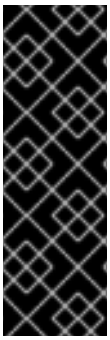
```
passwd username
```

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。『Red Hat Enterprise Linux 7 Security Guide (セキュリティーガイド)』 の 「パスワードのセキュリティー」 を参照してください。

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* はどのような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** を指定すると、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** を指定すると、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。
-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) のグループ名またはグループ番号の一覧で、コンマで区切ります。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。

オプション	
-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** で暗号化されたパスワードです。
-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。デフォルトでは **/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 範囲が以前のリリースから変更になりました。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の範囲が 1 ~ 999 となっています。この変更により、既存のユーザーの UID と GID が 500 ~ 999 の場合に Red Hat Enterprise Linux 7 に移行すると、問題が発生することがあります。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

以下のステップは、シャドウパスワードが有効なシステム上で **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. **/etc/passwd** に **juan** の新しい行が作成されます。

```
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** で始まっています。
- パスワードフィールドは **x** となっており、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 より大きい数字の UID が作成されます。Red Hat Enterprise Linux 7 では、1000 未満の UID はシステム使用のために確保されています。これらはユーザーに割り当てないことをお勧めします。
- 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 はシステム使用のために確保されています。これらはユーザーに割り当てないことをお勧めします。

- オプションの *GECOS* 情報は空白のままです。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は **/home/juan/** に設定されています。
- デフォルトのシェルは **/bin/bash** に設定されています。

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** で始まっています。
- 2 つの感嘆符 (**!!**) が **/etc/shadow** ファイルのパスワードフィールドに表示され、これがアカウントをロックします。

注記

-p フラグを使用して暗号化されたパスワードを渡している場合は、**/etc/shadow** ファイル内のユーザー用の新しい行に置かれます。

- パスワードは有効期限なしで設定されています。

3. **juan** というグループ用の新しい行が **/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 に作成された行には、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 **x** が表示され、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** 内の **juan** の プライマリーグループに対する GID と一致します。

4. **juan** というグループ用の新しい行が **/etc/gshadow** に作成されます。

```
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。
- 1 つの感嘆符 (**!**) が **/etc/gshadow** ファイルのパスワードフィールドに表示され、これがグループをロックします。
- 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** 用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** とグループ **juan** が所有しています。ユーザー **juan** に ***のみ*** *読み取り*、*書き込み* および *実行* の権限が与えられています。その他のパーミッションはすべて拒否されます。

6. **/etc/skel/** ディレクトリー内のファイル (デフォルトのユーザー設定を含む) 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、**juan** と呼ばれるロックされたアカウントがシステムに存在します。これをアクティブ化するには、管理者が **passwd** コマンドを使用して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

システムに新規グループを追加するには、**root** としてシェルプロンプトで以下を実行します。

```
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* と併用します。*gid* がすでに存在している場合は、**groupadd** がグループ用に別の一意の *gid* を選択します。
-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。

オプション	詳細
-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 の様々なオプションは、ユーザーのプライマリーグループと補助グループに様々な影響を与えます。

ユーザーのプライマリーグループを上書きするには、**root** として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** として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、ユーザーの以前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** として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、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、通常それぞれの主要なプロジェクトごとにグループを作成し、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人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、誰かがファイルを作成すると、ファイルはその人が属するプライマリグループに関連付けられるため、ファイル管理は困難です。1 人が複数のプロジェクトに関わっている場合、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。一方で、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内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することができます。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** としてシェルプロンプトで以下を入力して、**/opt/myproject/** ディレクトリーを作成します。

   ```
   mkdir /opt/myproject
   ```

2. システムに **myproject** グループを追加します。

```
groupadd myproject
```

3. **/opt/myproject/** ディレクトリーのコンテンツと **myproject** グループを関連付けます。

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

4. グループのユーザーがディレクトリー内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
```

この時点で、ユーザーが新規ファイルの書き込みをする度に管理者はファイルのパーミッションを変更する必要なく、**myproject** グループの全メンバーは **/opt/myproject/** ディレクトリー内の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. ユーザーを **myproject** グループに追加します。

```
usermod -aG myproject username
```

4.3.5. umask を使用した新しいファイルのデフォルト権限の設定

プロセスでファイルが作成されると、ファイルは **-rw-rw-r--** など、特定のデフォルト権限を持ちます。こうした初期権限は、*ファイル権限マスク* または *umask* とも呼ばれる *ファイルモード作成マスク* で部分的に定義されています。たとえば、デフォルトで **bash** は *umask* **0022** を持つなど、すべてのプロセスにそれぞれの umask があります。*umask* プロセスは変更できます。

umask を構成するもの

umask は、標準のファイル権限に対応するビットで構成されています。たとえば、*umask* が **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、実行および書き込みビットが設定されます
- **7** = 他の人の権限、実行、書き込み、読み取りビットが設定されます

umask は 2 進法、8 進法またはシンボリック表示で表されます。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 進法表示と反対で、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**禁止権限を設定**します (どの権限を設定できないかを指定します)。

- *umask* にビットを設定すると、ファイルの設定は解除されます。

- *umask* にビットが設定されていないと、他の要素にもよりますが、ファイルで設定できます。

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のかを示しています。

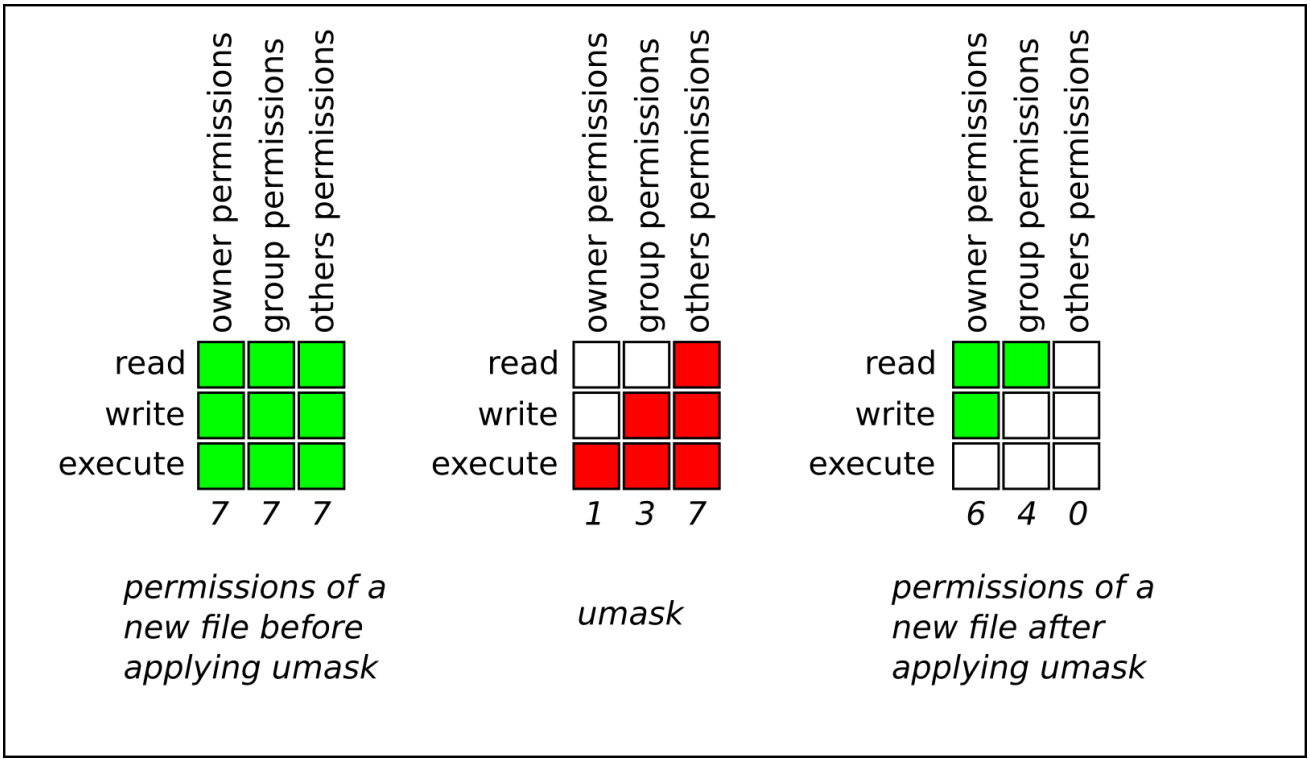
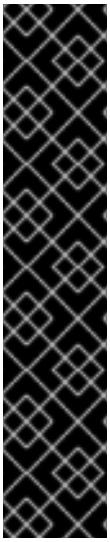


図4.3 ファイルの作成時に umask を適用



重要

セキュリティー上の理由から、レギュラーファイルはデフォルトで実行権限を持ちません。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. シェルでの umask の管理

bash、**ksh**, **zsh**、**tcsh** などの一般的なシェルでは、**umask** シェル **ビルトイン** を使用して *umask* を管理します。シェルから開始するプロセスは umask を継承します。

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

```
~]$ umask
0022
```

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

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

umask を使用したシェルでのマスク設定

8 進法を使用して、現在のシェルセッションの *umask* を設定するには、以下を実行します。

```
~]$ umask octal_mask
```

octal_mask を **0** から **7** の 4 桁以下の数字に置き換えます。3 桁以下の場合、コマンドに最初に 0 を含んでいるものとして権限が設定されます。たとえば、*umask* **7** は **0007** に変換されます。

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
```

シンボリック表示を使用して現在の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

シンボリック表示を使用して、*umask* を **0337** に設定するには、以下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 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、そのファイルは **/etc/bashrc** です。デフォルトの **bash** umask を表示するには、以下を実行します。

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

出力では、*umask* が **umask** コマンドまたは **UMASK** 変数のいずれかを使用して設定されていることが示されます。以下の例では、*umask* は **umask** コマンドを使用して **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 のデフォルトの *umask* を変更するには、**/etc/bashrc** で **umask** コマンドコールまたは **UMASK** 変数割り当てを変更します。この例では、デフォルトの *umask* を **0227** に変更しています。

```
   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 umask での作業
デフォルトで、新しいユーザーの **bash** *umask* は **/etc/bashrc** で定義されたものになっています。

特定のユーザーの **bash** *umask* を変更するには、そのユーザーの **$HOME/.bashrc** ファイルの **umask** コマンドにコールを追加します。たとえば、ユーザー **john** の **bash** *umask* を **0227** に変更します。

```
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のデフォルト権限設定
作成されたユーザーホームディレクトリ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. 関連資料

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

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、新規ユーザーの作成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、ユーザーの削除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、ユーザーの変更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、新規グループの作成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、グループの削除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、グループメンバーシップの変更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** ファイルの管理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** ファイルの整合性の検証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** と **/etc/shadow** ファイルの整合性の検証方法が説明されています。
- **pwconv**(8) - **pwconv**、**pwunconv**、**grpconv**、および **grpunconv** コマンドの man ページには、パスワードとグループのシャドウ化された情報を変換する方法が記載されています。
- **id**(1) - **id** コマンドの man ページには、ユーザー ID およびグループ ID の表示方法が記載されています。
- **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、システムグループの定義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、ユーザー情報の定義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、システムにおけるパスワードとアカウントの有効期限情報の設定方法が説明されています。

オンラインのドキュメント

- 『Red Hat Enterprise Linux 7 セキュリティーガイド』 - 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** コマンドおよび **sudo** コマンドを使って管理者権限を取得する方法が説明されています。

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収納されています。

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*

例:

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、そのアクセスに大して ACL が有効になっている場合は、ACL が認識されます。これは、Samba が **--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 をサポートしており、NFS クライアントが ACL を読み込める場合、ACL はクライアントシステムによって使用されます。

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 **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 が設定あれていない場合は、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 はオプションです。

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 (effective rights) 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 の追加または修正を実行できます。

```
# 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。コンマで区切って複数のルールを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、または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、以下を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 は上書きされます)。

```
# 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 を確認しています。

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一覧は 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	詳細
-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、または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	詳細
-/	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、または復元します。

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チェックすることができます。古いバージョンはこのチェックを拒否します。

5.7. ACL 参照情報

より詳しい情報については以下の 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を説明しています
- **man setfacl** - ファイルアクセス制御リストの設定方法を説明しています
- **man star** - **star** ユーティリティーとそのオプションを詳しく説明しています

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うと、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することができます。

管理者制御や潜在的な危険、管理者アクセスの不適切な使用によるデータ破損の回避方法は 『Red Hat Enterprise Linux 7 Security Guide』 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、認証後に **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。このアクセスは、SELinux が有効な場合はこれによる制限を受けますが、**root** になったユーザーは、 **su** コマンドを使用して、パスワードを求められることなくシステム上の他のユーザーに切り換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人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、以下のコマンドを **root** で実行します。

```
~]# 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、**ユーザー** 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** キーはキーボードやハードウェアによって異なりますが、通常は **スペースバー** の左側にある Windows または Command キーです。

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。

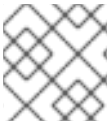
4. **アカウントの種類** を **Standard** から **Administrator** に変更します。これにより、ユーザーが **wheel** グループに追加されます。

ユーザー 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**、**/etc/pam.d/su** の PAM *(プラグ可能な認証モジュール)* 設定ファイル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、# 文字を削除して以下の行からコメントを削除します。

```
#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。



注記

root ユーザーはデフォルトで **wheel** グループの一員になっています。

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。信頼できるユーザーが、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** ユーザーが実行しているかのように実行されます。

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、**ユーザーの** シェルでコマンドが実行されます。つまり、『Red Hat Enterprise Linux 7 セキュリティガイド』 にあるように、**root** シェルを完全に無効化できます。

sudo コマンドを使用した正常な認証はすべて **/var/log/messages** ファイルに記録され、この発行者のユーザー名で発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、pam_tty_audit(8) man ページ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** コマンドを発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。

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あ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、パスワードをタイムアウトの 5 分間、保存します。この間に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** を使っ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、理論上は攻撃者に時間無制限の管理アクセスを与えることになります。

6.3. 関連資料

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

- **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。

オンラインのドキュメント

- 『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』 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ー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について説明しています。

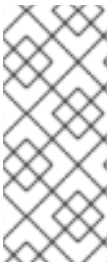
パート II. サブスクリプションおよびサポート

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) および有効になっている適切なリポジトリーに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、すでにインストールされた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。すべての **subscription-manager** コマンドは **root** で実行することになっている点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求めるプロンプトが出されます。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一覧を表示します。

   ```
   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、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一覧表示するには、**--all** オプションを追加します。プール ID は、**Pool ID** で始まる行に一覧表示されます。

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一覧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

システムが Red Hat コンテンツ配信ネットワークをサブスクライブしていると、リポジトリーファイルは **/etc/yum.repos.d/** ディレクトリーに作成されます。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一覧を表示します。

```
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一覧を表示するには、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**) であり、*variant* は Red Hat Enterprise Linux システムのバリアント (**server** または **workstation**) になります。以下に例を示します。

```
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。

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一覧表示し、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** に挙げられている番号です。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. 関連資料

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

- **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。

関連資料

- 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が説明されています。
- 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 パッケージ内の **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、サポートケースをオープンおよび更新したり、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。

1. ```
   ~]# 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。

2. ```
   ~]# 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 (使用できるコマンドは少なくなります) または **root** で実行できます。

? 文字を入力するとコマンドの一覧を表示できます。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一覧を表示できます。

```
~]# 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
```

手順8.1 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 **~/.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 (他の方法が設定されていない場合)。必要な場合は、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見ることができます。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
```

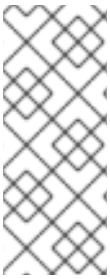
注記

-g, --global オプションを使用して設定をグローバルで保存するには、**Red Hat Support Tool** を **root** で実行する必要があります。通常のユーザーは **/etc/redhat-support-tool.conf** に書き込むのに必要なパーミッションを持っていないためです。

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消去および設定解除され、グローバル設定ファイル (利用可能な場合) の同等の設定が使用されます。



注記

非特権ユーザーで実行している場合、グローバル設定ファイルに保存された値は **-u**、**--unset** オプションを使用して削除できませんが、**-g**、**--global** オプションと **-u**、**--unset** オプションを併用してツールの現在の実行中インスタンスから消去および設定解除できます。**root** で実行している場合、値とオプションは **-g**、**--global** と **-u**、**--unset** オプションを併用してグローバル設定ファイルから削除できます。

8.5. インタラクティブモードでのサポートケースのオープンおよび更新

手順8.2 インタラクティブモードでの新しいサポートケースのオープン

インタラクティブモードで新しいサポートケースをオープンするには、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、サポートケースをオープン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. サポートケースをオープン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。

手順8.3 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
```

ここで、*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。

手順8.4 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
```

ここで、*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
```

ここで、*case-number* はダウンロードするケースの番号です。

8.7. 関連資料

Red Hat ナレッジベースの記事 『Red Hat Access の 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 システム上のすべてのソフトウェアは RPM パッケージとして分割されており、インストールやアップグレード、削除が可能です。ここここで、**Yum** を使って 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 パッケージマネージャーです。Yum は、パッケージ情報に関するクエリ、リポジトリ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とアンインストール、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を行うことができます。Yum は、パッケージの更新/インストール/削除を実行しているパッケージで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/インストールすることができます。

Yum は、新しい追加のリポジトリーまたは *パッケージソース* を使って設定することができ、機能を強化、拡張する多くのプラグインも提供します。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 を使うことで、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 で示されてるようにインストール中に 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) 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、移動中に変更されていないことが確信できます。Yum の署名認証を有効にする方法については、「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、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、これは習得する価値があります。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一覧表示してこれらを一度に更新することや、個別パッケージを選択して更新することができます。

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ると示されています。一覧の最初のパッケージは、**dracut** です。出力例の各行は複数の行から構成されます。**dracut** の場合、以下の構成になっています。

- **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** — インストールされる更新済みパッケージのバージョン
- **360.el7** — 更新されるパッケージのリリース
- **_2** — ビルドのバージョン。z-stream 更新の一部として追加
- **rhel-7-server-rpms** — 更新済みのパッケージがあるリポジトリ

また上記の出力は、**yum** コマンドを使ってカーネル (kernel パッケージ)、Yum および RPM (yum および rpm パッケージ)、さらにはそれらの依存関係 (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

1 回に更新するパッケージ数を 1 つ、複数、又は全てのパッケージから選択することができます。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、それらも更新されます。

単一パッケージの更新
単一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て下さい:

```
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]:
```

この出力には興味を引くアイテムがいくつかあります。

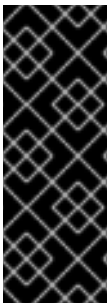
1. **Loaded plugins: langpacks, product-id, subscription-manager** — yum は、どの Yum プラグインがインストールされ有効であるかを常に通知します。Yum プラグインに関する一般的情報は 「Yum のプラグイン」 を参照してください。また、個別のプラグイ

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することができます。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 はデフォルトでは対話的に実行されます。**yum** コマンドがどのトランザクションを実行するかが事前にわかっている場合は、**-y** オプションを使用して、yum が尋ねるすべての質問に **yes** と自動回答できます (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。その結果、発生する可能性がある問題を簡単にトラブルシューティングできます。また、パッケージをインストールせずにダウンロードすることも選択もできます。この場合は、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されます。

   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閲覧することができます。



重要

yum update 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。

一方で **RPM** を使用する場合は、**rpm -u kernel** (現在のカーネルと **置換**) の代わりに、**rpm -i kernel** コマンド (新しいカーネルのインストール) 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することできます。**root** で以下を入力します。

```
yum group update group_name
```

ここでは、*group_name* を更新するパッケージグループに置き換えます。パッケージグループについての詳細は、「パッケージグループでの作業」 を参照してください。

Yum には **upgrade** コマンドもあります。これは、**obsoletes** 設定オプションを有効にした **update** と同様のものです (「[main] オプションの設定」 参照)。デフォルトでは、**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
すべてのパッケージとそれらの依存関係を更新するには、引数なしで **yum update** コマンドを使用します。

```
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、これら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** として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とこともできます。**root** として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更新としてリリースされている。
- kernel-3.10.0-3 がバグ修正の更新としてリリースされた。

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。

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。次の手順に進む前に作成してください。**root** で以下のコマンドを入力します。

   ```
   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入力します。

   ```
   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。ここでは、ファイルをブロックデバイスとしてマウントするために、**-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入力します。

```
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入力します。

```
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入力します。

```
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入力します。

```
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** として 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、前の手順でコピーされた **/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閲覧、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、サマリーを検索することができます。

```
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すべてを一覧表示するには、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[出力は省略されています]

============================== N/S matched: emacs
==============================
emacs.x86_64 : GNU Emacs text editor
emacs-auctex.noarch : Enhanced TeX modes for Emacs[出力は省略されています]

  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検索を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を返すことで、検索が速まります。より包括的な検索には **yum search all** を使用しますが、速度は遅くなります。

結果をフィルターする

Yum の一覧 コマンドではすべて、1 つ以上の *glob 表現* を引数として追加して結果をフィルターすることができます。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 *パス名の展開* 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が説明されます。

9.2.2. パッケージの一覧表示

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一覧表示するには、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一覧表示するには、以下のコマンドを使用します。

```
yum list glob_expression…
```

例9.5 ABRT 関連パッケージの一覧

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一覧表示するには、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
```

システムにインストール済みのパッケージすべてを一覧表示するには、**installed** キーワードを使用します。出力の右端の列には、パッケージが取得されたリポジトリーが表示されます。

```
yum list installed glob_expression…
```

例9.6 インストール済み krb パッケージの一覧表示

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すべてを一覧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一覧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
```

有効なリポジトリーすべてでインストール可能なパッケージを一覧表示するには、以下の形式のコマンドを使用します。

```
yum list available glob_expression…
```

例9.7 利用可能な gstreamer プラグインの一覧表示

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すべてを一覧表示するには、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
```

リポジトリーの一覧表示

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一覧表示するには、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についての詳細情報を一覧表示するには、**-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** コマンドを使っ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、以下のコマンドを使用します。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
```

disabled を最初の引数として渡すことで、コマンドの出力を無効なリポジトリーに制限できます。詳細な仕様については、リポジトリーの ID や名前、関連する glob 表現を引数として渡すことができます。引数と完全に一致するリポジトリー ID または名前がある場合は、**enabled** 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…
```

package_name をパッケージ名に置き換えます。

例9.8 abrt パッケージの情報を表示する

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することで、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わ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 でクエリーする

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/Pack
ages/yum-3.4.3-132.el7.noarch.rpm
     reason = user
     releasever = 7Server
     var_uuid = 147a7d49-b60a-429f-8d8f-3edb6ce6f4a1
```

yumdb コマンドの詳細については、**yumdb**(8) の man ページを参照してください。

9.2.4. パッケージのインストール

単一パッケージおよびそのパッケージのインストールされていない依存関係すべてをインストールするには、**root** で以下の形式で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、引数としてその名前を追加します。**root** で以下を入力します。

```
yum install package_name package_name…
```

AMD64 や Intel 64 マシンのような *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* を追加することで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 表現を使うと、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…
```

例9.11 すべての audacious プラグインをインストールする

似通っ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。すべての audacious プラグインを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することができます。インストールするバイナリ名が分かっていて、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** で以下を入力します。

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

yum はパッケージ一覧で検索を行い、**/usr/sbin/named** を提供するパッケージを探します。存在すれば、yum はインストールするかどうかを確認します。

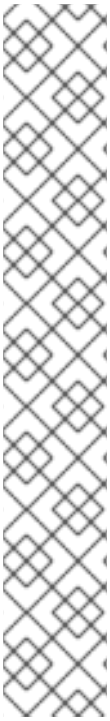
上記の例で分かるように、**yum install** コマンドは厳密に定義された引数を必要としません。様々な形式のパッケージ名や glob 表現を処理できるので、ユーザーによるインストールを容易にします。一方で、**yum** が入力を正確に分析するにはしばらく時間がかかります。多数のパッケージを指定した場合は、特にそうです。パッケージ検索を最適化するには、以下のコマンドを使って引数の分析方法を明示的に定義できます。

```
yum install-n name
```

```
yum install-na name.architecture
```

```
yum install-nevra name-epoch:version-release.architecture
```

install-n では、**yum** は *name* を正確なパッケージ名として判断します。**install-na** コマンドは **yum** に、その後に続く引数にピリオドで分けられたパッケージ名とアーキテクチャーが含まれていることを伝えます。**install-nevra** では、**yum** は引数が *name-epoch:version-release.architecture* の形式になっていることを予想します。同様に、**yum remove-n**、**yum remove-na**、および **yum remove-nevra** を使って削除するパッケージを検索することもできます。



注記

named バイナリを含むパッケージをインストールしたいものの、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* を含むパッケージを見つけるための便利な方法です。

例9.12 インストールプロセス

以下の例は、**yum** を使っ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 は既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** (はい) または **N** (いいえ) オプションの他に、**d** (ダウンロードのみ) を選択すると、パッケージのダウンロードはしますが、直接インストールはしません。**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!
```

システム上のローカルディレクトリーから既にダウンロード済みのパッケージを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へ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、インストールの確認を尋ね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 オプションを使用すると、yum によりパッケージのインストールなしでパッケージがダウンロードされます。このようなパッケージは、後でオフラインで **yum localinstall** コマンドを使ってインストールしたり、別のデバイスと共有したりできます。ダウンロードされ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。ダウンロードはバックグラウンドで行われるため、並行して **yum** を他の操作に使うことができます。

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 を使う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れに依存するすべてのパッケージをアン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 remove package_name…
```

複数のパッケージをインストールする場合と同様に、コマンドに複数のパッケージ名を追加することで一度に複数のパッケージを削除することができます。

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** は以下の引数を取ることができます。

- パッケージ名
- glob 表現

- ファイル一覧
- パッケージが提供する機能

警告

Yum では、パッケージに依存しているパッケージを削除せずにそのパッケージを削除することはできません。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、依存パッケージのセットもプルされるので時間が大幅に短縮できます。**yum groups** コマンドは、yum 内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一覧表示

summary オプションを使うと、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一覧表示するには、**list** オプションを追加します。コマンドの出力は、グループ名ごとにフィルターすることができます。

```
yum group list glob_expression…
```

このコマンドでは渡すことのできるオプションの引数がいくつかあります。**hidden** はユーザー表示可能とマークされていないグループも一覧表示し、**ids** はグループ ID を表示します。また、**language**、**environment**、**installed**、**available** などの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一覧表示するには、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を表示する

```
 ~]$ yum group info LibreOffice
Loaded plugins: langpacks, product-id, subscription-manager

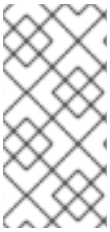
Group: LibreOffice
 Group-Id: libreoffice
 Description: LibreOffice Productivity Suite
 Mandatory Packages:
  =libreoffice-calc
   libreoffice-draw
  -libreoffice-emailmerge
   libreoffice-graphicfilter
  =libreoffice-impress
  =libreoffice-math
  =libreoffice-writer
  +libreoffice-xsltfilter
 Optional 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、パッケージグループの一部として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外でインストールされました。このため、**yum group remove** でこのパッケージを削除することはできません。

これらの区別は、**group_command** 設定パラメーターがデフォルト設定の **objects** になっている場合にのみ発生します。yum でパッケージがグループの一部としてインストールされたかまたは別個にインストールされたかを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 コマンドを使って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避けるには、**yum group mark blacklist** を使います。**yum group mark** の詳細な機能については、**yum**(8) man ページを参照してください。



注記

@^ 接頭辞を使う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** を使用する際は、**@group_name** を渡すとパッケージグループが、**@^group_name** を渡すと環境グループが指定され、**group_name** を渡すとこれら両方を含めることができます。

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。全パッケージグループの名前とそれらのグループ ID (括弧内に表示される) を一覧表示するには、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 を見つけるには、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、非表示グループを一覧表示できます。

```
~]$ 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** にグループ名全体 (groupid を含めない) を渡します。**root** で以下のコマンドを入力します。

```
yum group install "group name"
```

groupid で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 または引用符で囲んだグループ名を@ 記号の後に指定して、**install** コマンドに渡すことができます。これにより、**group install** を実行することが **yum** に通知されます。**root** で以下のコマンドを入力します。

```
yum install @group
```

group を groupid または引用符で囲んだグループ名で置き換えます。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には、4 つの異なる、ただし同等の方法があります。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

パッケージグループ名またはその ID を使って、**install** 構文に類似した構文でパッケージグループを削除することができます。**root** で以下のコマンドを入力します。

```
yum group remove group_name
```

```
yum group remove groupid
```

また、groupid または引用符で囲んだグループ名を@ マークに続けて、**remove** コマンドに渡すことも可能です。これで **group remove** を実行したいというメッセージが yum に伝わります。**root** で以下を入力します。

```
yum remove @group
```

group を groupid または引用符で囲んだグループ名で置き換えます。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には、4 つの異なる、ただし同等の方法があります。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することができます。さらに、このコマンドを使

うと、特定のトランザクションを元に戻す、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一覧表示

最近発生した 20 件のトランザクションを一覧表示するには、 **root** で引数なしで **yum history** を実行するか、シェルプロンプトで以下を入力します。

```
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一覧表示することも可能です。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…
```

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 コマンドはすべての形式で、以下のコラムで構成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** — ユーザー名で、この名前のログインセッションを使ってトランザクションが開始されました。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」 の説明のとおり、トランザクション中に実行された動作の一覧です。
- **Altered** — 表9.2「Altered フィールドの値」 の説明のとおり、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。
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。

記号	詳細
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使用され、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させるには、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ついての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することできます。

```
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一覧表示するには、**root** で以下のコマンドを実行します。

yum history package-list *glob_expression…*

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たい場合は、**root** で以下のコマンドを実行します。

```
yum history info id…
```

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、および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知りたい場合は、**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は、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には、**yum history addon-info** は以下の出力を返します。

```
~]# 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です。グローバルオプションの変更方法の詳細は、「[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については、「[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/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** としてシェルプロンプトで以下を入力します。

```
yum history undo id
```

特定のトランザクションを繰り返すには、**root** で以下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 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され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。トランザクションにより新しいパッケージがインストールされた場合、**yum history undo** コマンドはそれをアンインストールします。トランザクションによりパッケージがアンインストールされた場合、このコマンドは再度インストールします。またこのコマンドは、すべての更新済みパッケージを以前のバージョンにダウングレードする試みも実行します (古いパッケージが引き続き利用可能な場合)。

複数の同一システムを管理する場合、yum を使用すると、1 つのシステム上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上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格納するには、**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や rpmdb バージョンを無視するように **load-transaction** を設定することも可能です。これらの設定オプションについての詳細は、**yum.conf**(5) の 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格納します。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/** ディレクトリー内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お勧めします。

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、ここでは全体に影響を与える yum オプションを設定できます。また、1 つ以上の **[*repository*]** セクションを含め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/** ディレクトリー内にある、新規または既存の **.repo** ファイルで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]** セクションで設定された値よりも優先されます。

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** の **[*repository*]** セクションと **/etc/yum.repos.d/** ディレクトリー内の **.repo** ファイルを編集することで、個々のリポジトリーの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、1 つの **[main]** セクションが含まれます。この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 のリポジトリーの処理方法に影響を与えるものもあります。**/etc/yum.conf** 内にある **[main]** のセクション下に、多くのオプションを追加することができます。

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
[コメントは省略されています ]

# 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尋ねるかどうかを決定します。*value* を以下のどちらかに置き換えます。

0 — (**デフォルト**)。yum は実行する重要な動作の確認を尋ねます。

1 — **yum** が実行する重要な動作の確認を尋ねません。**assumeyes=1** に設定すると、yum はコマンドラインオプション **-y** および **--assumeyes** と同様の動作を実行します。

cachedir=*directory*

このオプションを使って、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。

$basearch および **$releasever** yum 変数の詳細については、「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** 値を大きい値に設定すると、yum はより詳細なデバッグ出力を表示します。**debuglevel=0** は、デバッグ出力を無効にします。デフォルト値は **debuglevel=2** です。

exactarch=*value*

このオプションを使うと、インストール済みのパッケージを更新の際に 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。この設定では、yum は 64 ビットアーキテクチャーのシステムにすでにインストール済みパッケージを更新するために 32 ビットアーキテクチャーのパッケージをインストールしません。

exclude=*package_name* [*more_package_names*]

exclude オプションを使うと、インストールまたはシステムの更新中にキーワードを使ってパッケージを除外することができます。除外する複数のパッケージを一覧表示したい場合は、パッケージをスペースで区切り、引用符で囲みます。ワイルドカードを使ったシェル glob 表現 (***** や **?** など) が利用できます。

gpgcheck=*value*

gpgcheck オプションを使うと、yum がパッケージに GPG 署名確認を実行するかどうかを指定できます。*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上での 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上で GPG 署名確認を有効にします。**gpgcheck** が有効だと、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、全リポジトリに対して GPG 確認ルールが設定されます。代わりに、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 確認を有効にしながら別のリポジトリでは無効にすることができます。個々のリポジトリーに対応する **.repo** ファイルで **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 オプションを使うと、**yum group install**、**yum group upgrade**、および **yum group remove** の各コマンドがパッケージグループを処理する方法を指定できます。*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。

objects — (**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を表示する」 を参照してください。

group_package_types=*package_type* **[***more_package_types***]**

このオプションでは、**yum group install** コマンドが呼び出された際にどのタイプのパッケージがインストールされるか (**optional**、**default**、または **mandatory**) を指定できます。**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うと、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。この情報は、**yum history** コマンドで表示できます。**history_record=1** がデフォルト値です。

yum history コマンドに関する詳細情報は、「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 は履歴記録を使って、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一覧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一覧については、**yum.conf**(5) の 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** ファイルに追加する場合は、**yum.conf**(5) の **installonlypkgs** セクション下に表示されているものも含め、インストールのみである **すべての** パッケージを一覧表示してください。特に、カーネルパッケージは常に **installonlypkgs** (デフォルトのとおり) に一覧表示するようにしてください。また、デフォルトのカーネルがブートに失敗した場合でもバックアップカーネルを常に利用できるように、**installonly_limit** は常に **2** より大きい値に設定することをお勧めします。

installonly_limit=*value*

このオプションは、**installonlypkgs** ディレクティブでリストされる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。*value* を **installonlypkgs** でリストされている単一パッケージについて同時インストールできるバージョンの最大数を表す整数で置き換えます。

installonlypkgs ディレクティブのデフォルトには複数の様々なカーネルパッケージが含まれているため、**installonly_limit** の値を変更すると、単一のカーネルパッケージのインストール済みバージョンの最大数にも影響が及ぶ点に注意してください。**/etc/yum.conf** に表示されているデフォルト値は、**installonly_limit=3** です。また、この値を低く、特に **2** より下に設定することは推奨されません。

keepcache=*value*

keepcache は、インストールの成功後に yum がヘッダーおよびパッケージのキャッシュを保持するかどうかを決定します。ここでは、*value* を以下のいずれかに置き換えます。

0 — (**デフォルト**)。インストールの成功後に、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、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 オプションは、インストールするパッケージに複数のアーキテクチャーバージョンがある場合にインストール動作を設定します。ここでの *value* は、以下のいずれかになります。

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。例えば 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-bit 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。例えば、AMD64 システムで **multilib_policy** を **all** に設定すると、yum はパッケージの i686 および AMD64 が利用可能であれば両バージョンをインストールします。

obsoletes=*value*

obsoletes オプションは、更新の実行時に obsoletes 処理ロジックを有効にします。あるパッケージがスペックファイル内で別のパッケージを *廃止する* ように宣言している場合、宣言しているパッケージがインストールされると、宣言されているパッケージはこのパッケージによって置き換えられます。例えば、パッケージ名が変更された場合などに obsoletes は宣言されます。*value* を以下のいずれかで置き換えます。

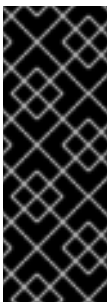
0 — 更新の実行時に yum の obsoletes 処理ロジックを無効にします。

0 — (**デフォルト**)。更新の実行時に yum の obsoletes 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** に設定した場合は、ある yum プラグインの設定ファイル内で **enabled=0** を設定することでそのプラグインを無効にすることも可能です。

yum の各種プラグインの詳細については、「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、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。 *value* は **0** 以上の整数で、**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]** オプションの完全なリストは、**yum.conf**(5) の man ページ 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することができます。ここで、*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*

ここでは、*repository_name* は、リポジトリーを説明するヒューマンリーダブルな文字列になります。

baseurl=*repository_url*

*repository_url*をリポジトリーの repodata ディレクトリーがあるディレクトリーへの URL で置き換えます。

- 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 にユーザー名とパスワードを先頭に追加して、指定することができます。たとえば、http://www.example.com/repo/ にある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** のリンクは、**http://user:password@www.example.com/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** 変数を展開する点に注意して下さい。yum 変数の詳細については、「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*

これで簡単に yum に特定のリポジトリーを使用するか無視するかを伝えられます。*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用に有効としたくないリポジトリーから、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*** もしくは **--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。*value* 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。全一覧については、**yum.conf**(5) の 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することができます。

$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することができます。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-release*product*** パッケージからバージョン番号を取得することで、正しい値を推測します。

$arch

この変数を使用して、Python の **os.uname()** 機能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 の両マシンは **i386** のベースアーキテクチャーを持っており、AMD64 および Intel 64 の両マシンは **x86_64** のベースアーキテクチャーを持っています。

$YUM0-9

これら 10 個の変数は、同じ名前を持つシェル環境変数の値でそれぞれ置換されます。これら変数のいずれかが (例えば **/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は、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. 現行設定の表示

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一覧表示するには、以下の形式でコマンドを実行します。

```
yum-config-manager section…
```

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…
```

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一覧表示するには、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[出力は省略されています]
```

9.5.5. Yum リポジトリーの追加、有効化、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お勧めします。

「[repository] オプションの設定」 では、yum リポジトリーの定義に使用可能な様々なオプションについて説明しました。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、有効、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/** ディレクトリー内の **.repo** ファイルに追加します。yum は、このディレクトリー内にある **.repo** ファイル拡張子が付いたすべてのファイルを読み取ることができます。リポジトリーは **/etc/yum.conf** 内ではなく、ここに定義することが推奨されます。

警告

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ソフトウェアパッケージを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。

通常 Yum リポジトリーにはそれぞれの **.repo** ファイルがあります。ある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
```

ここでの *repository_url* は、**.repo** ファイルへのリンクになります。

例9.27 example.repo を追加する

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、シェルプロンプトで **root** として以下を入力します。

```
yum-config-manager --enable repository…
```

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 を一覧表示するには **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…
```

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[出力は省略されています]
```

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[出力は省略されています]
```

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…
```

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 を一覧表示するには **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することができます。

```
yum-config-manager --disable glob_expression…
```

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[出力は省略されています]
```

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、以下のステップにしたがいます。

1. シェルプロンプトで **root** として以下を入力し、createrepo パッケージをインストールします。

   ```
   yum install createrepo
   ```

2. リポジトリーに配置するパッケージすべてを、**/mnt/local_repo/** などの 1 つのディレクトリーにコピーします。

3. このディレクトリーに移動して、以下のコマンドを実行します。

   ```
   createrepo --database /mnt/local_repo
   ```

   これにより yum リポジトリーに必要なメタデータ、さらには **sqlite** データベースが作成されるため yum の動作が迅速化します。

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の記事で説明されている手順に従ってください。

9.6. YUM のプラグイン

Yum は、動作を拡張し強化するプラグインを提供します。一部のプラグインは、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と常に、Yum はどのプラグインが読み込まれ、アクティブかを知らせます。例を示します。

```
~]# yum info yum
Loaded plugins: langpacks, product-id, subscription-manager[出力は省略されています]
```

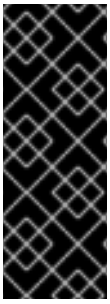
Loaded plugins に続くプラグインの名前は **--disableplugins=*plugin_name*** オプションに渡すことが可能な名前である点に注意して下さい。

9.6.1. Yum プラグインを有効、設定、無効にする方法

yum プラグインを有効にするには、**plugins=** で始まる行が **/etc/yum.conf** の **[main]** セクションにあり、値が **1** であるようにします。

```
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。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。

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。これらのファイルで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 ファイルと同様に、プラグイン設定ファイルには常に、**enabled=** オプションで **yum** コマンドの実行時にプラグインを有効にするかどうかを制御できる **[main]** セクションが含まれます。このオプションがない場合は、手動でファイルに追加できます。

/etc/yum.conf で **enabled=0** と設定してすべてのプラグインを無効にすると、すべてのプラグインは個々の設定ファイルで有効かどうかに関わらず無効になります。

1 回の **yum** コマンドで単にすべての yum プラグインを無効にする場合は、**--noplugins** オプションを使用します。

1 回の **yum** コマンドで 1 つ以上の yum プラグインを無効にしたい場合は、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= オプションに渡すプラグインの名前は、**yum** コマンドの出力内にある **Loaded plugins** の行の後に表示されている名前と同じです。複数のプラグインを無効にするには、名前をコンマで区切ります。さらに glob 表現を使用すると、複数のプラグイン名の適合や名前の短縮を行うことができます。

```
~]# yum update --disableplugin=aliases,lang*
```

9.6.2. 追加の Yum プラグインのインストール

通常 yum プラグインは **yum-plugin-*plugin_name*** パッケージの命名規則に準拠しますが、常にそうであるとは限りません。たとえば、**kabi** プラグインを提供するパッケージの名前は、**kabi-yum-**

plugins です。yum プラグインのインストールは、他のパッケージをインストールする場合と同じように実行できます。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以下を入力します。

```
~]# 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 プラグインのいくつかについて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** オプションとともに実行された場合、または **assumeyes** ディレクティブが **/etc/yum.conf** で有効な場合は、プラグインが確認を要求せずに無効なリポジトリーを一時的および永久的に有効にします。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]** セクションで利用できるディレクティブの一覧については、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、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設定すると、そうしたパッケージが決して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]** セクションで利用できるディレクティブの一覧は、以下の表に示されています。

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。

aliases (yum-plugin-aliases)

aliases プラグインは、**yum** コマンドでのエイリアスの設定および使用を可能にする **alias** オプションを追加します。

yum-changelog (yum-plugin-changelog)

yum-changelog プラグインは、更新の前後でパッケージ変更ログの表示を可能にする **--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 を受けてダウンロードし、これを 1 回のみのトランザクションに有効とする **--tmprepo** コマンドラインオプションを追加します。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** の各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うと、ロックされたパッケージを表示、編集することができます。

9.7. 関連資料

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 ページは、サポートされているオプションとコマンドの完全な一覧を提供します。
- **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、このツールを使ってクエリーを行い、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、利用可能な 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についての一覧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 カスタマーポータルラボ — Red Hat カスタマーポータルラボに 「Yum リポジトリー設定ヘルパー」 が含まれています。

関連項目

- *6章 権限の取得* では、**su** および **sudo** コマンドを使って管理者権限を取得する方法が説明されています。

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についての情報を提供します。

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、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、または依存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 では、*systemd units* という概念が導入されています。これらの unit は 表10.2「Systemd Unit 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ついての情報を要約します。利用可能な systemd unit タイプの完全な一覧については、表10.1「利用可能な systemd Unit タイプ」 を参照してください。

表10.1 利用可能な systemd Unit タイプ

Unit タイプ	ファイル拡張子	詳細
Service unit	**.service**	システムサービス
Target unit	**.target**	systemd unit のグループ
Automount unit	**.automount**	ファイルシステムの自動マウントポイント
Device unit	**.device**	カーネルが認識するデバイスファイル
Mount unit	**.mount**	ファイルシステムのマウントポイント
Path unit	**.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
Scope unit	**.scope**	外部作成のプロセス
Slice unit	**.slice**	システムプロセスを管理する階層的に組織された unit グループ
Snapshot unit	**.snapshot**	systemd マネージャーの保存状態
Socket unit	**.socket**	プロセス間の通信ソケット
Swap unit	**.swap**	スワップデバイスまたはスワップファイル
Timer unit	**.timer**	systemd タイマー

表10.2 Systemd Unit ファイルの場所

ディレクトリー	詳細
/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 unit ファイル
/run/systemd/system/	ランタイム時に作成された systemd unit 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 unit ファイルのディレクトリーに優先します。
/etc/systemd/system/	**systemctl enable** で作成された systemd unit ファイルおよびサービス拡張向けに追加された unit ファイル。このディレクトリーは、runtime unit ファイルのディレクトリーに優先します。

system.conf を使用したデフォルトの 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することができます。これらのデフォルトではなく、systemd ユニットへグローバルにデフォルト値を上書きしたいときは、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 秒に設定）を上書きしたいときは、***DefaultTimeoutStartSec*** パラメータを使って必要な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アクティベーションに *socket units* 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試みる初回にオンデマンドでスタートできます。Systemd は、バスベースのアクティベーションに *D-Bus サービスファイル* 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った際に、オンデマンドでスタートできます。Systemd は、デバイスベースのアクティベーションに *device units* 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、特定のファイルもしくはディレクトリーの状態が変更される際にオンデマンドでスタートできます。Systemd は、パスベースのアクティベーションに *path units* を使用します。

- **マウントおよびオートマウントポイントの管理** — Systemd は、マウントポイントおよびオートマウントポイントを監視、管理します。Systemd はマウントポイントには *mount units* を、オートマウントポイントには *automount units* 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。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、並行アクティベーションはシステム起動に必要となる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 スクリプトをサポートします。これにより、systemd サービス unit への更新パスが容易になります。

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、SysV init および Upstart とほぼ互換す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可能な訳ではないため、その結果、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 ターゲットとそれらのランレベルとの比較に関する詳細は、「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。**`start`**、**`stop`**、および **`status`** といった標準のコマンドに加えて、SysV init スクリプトのオーサーは、任意の多数のコマンドに対するサポートを実装して追加機能を提供することができます。たとえば、Red Hat Enterprise Linux 6 の **`iptables`** の init スクリプトは、**`panic`** コマンドとの実行が可能です。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すべての受信および送信パケットの切断が開始されます。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を受け付けます。

  `systemctl` ユーティリティーの詳細および以前の **`service`** ユーティリティーとの比較に関する情報は、「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 を保存してそれを追跡します。すると **`systemctl`** ユーティリティーはこの PID を使ってクエリーを行い、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`** は最新の状態を判断したり、これを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。以前は、シャットダウンシーケンスが開始されると、Red Hat Enterprise Linux 6 およびそれ以前のシステムのリリースは、**`/etc/rc0.d/`** ディレクトリーにあるシンボリックリンクを使ってシステムサービスのステータスに関係なく、利用可能なシステムサービスすべてを停止していました。systemd では、シャットダウン時に、実行中のサービスのみが停止されます。

- 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りできません。systemd がサービスを開始すると、標準入力を **/dev/null** に接続し、ユーザーとのインタラクションを防ぎます。
- システムサービスは、(**HOME** および **PATH** 環境変数といった) コンテキストを開始したユーザーやそのセッションから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されます。
- SysV init スクリプトを読み込む際に Systemd は、Linux Standard Base (LSB) ヘッダーにコード化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 5 分間でタイムアウトするデフォルト設定になっており、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別に長いタイムアウト値を指定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変更点の詳細なリストは、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 と配布されており、**/etc/rc.d/init.d/** にある *init スクリプト* を使用していました。これらの init スクリプトは通常 Bash で書かれており、システム管理者がシステムの状態とシステム内の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、これらの 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目的を果たします。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** および **chkconfig** はシステム内で利用可能でまだ機能しますが、これらは互換性のために含まれており、使用の回避をお勧めします。

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。
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。

サービス	systemctl	詳細
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チェックします。
chkconfig --list	**systemctl list-unit-files --type service**	サービスすべてを一覧表示し、それらが有効かどうかをチェックします。
chkconfig --list	**systemctl list-dependencies --after**	指定されたユニットの前に開始するよう命令されるサービスを一覧表示します。
chkconfig --list	**systemctl list-dependencies --before**	指定されたユニットの後に開始するよう命令されるサービスを一覧表示します。

サービスユニットの指定

明確化を図るため、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。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可能で、その場合、**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。これらの名前はユニット名よりも短いことがあり、ユニット名の代わりとして使用することができます。特定のユニットに使用可能な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 の動作
chroot コマンドを使って root ディレクトリを変更すると、ほとんどの **systemctl** コマンドは、アクションの実行を一切拒否します。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は、ファイルシステムに関して異なる見方を持っているからです。このような状況は、**systemctl** が **kickstart** 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。これらのコマンドは、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与えます。したがってこれらのコマンドは、**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/** ディレクトリの下位のシステムで **httpd** サービスを有効化するときは。

```
~]# 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. サービスの一覧表示

読み込み済みの最新サービスユニットをすべて一覧表示するには、シェルプロンプトで以下を入力します。

```
systemctl list-units --type service
```

このコマンドは、各サービスユニットファイルの完全な名前 (**UNIT**) を表示し、その後にユニットファイルが読み込み済みかどうか (**LOAD**)、高レベル (**ACTIVE**) および低レベル (**SUB**) のユニットファイルのアクティベーション状態、および簡単な説明 (**DESCRIPTION**) が続きます。

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。状態に関係なくすべての読み込み済みユニットを表示したい場合は、 **--all** または **-a** のオプションを付けてコマンドを実行します。

```
systemctl list-units --type service --all
```

また、すべての利用可能なサービスユニットを一覧表示させて、それらが有効かどうかをチェックできます。これには、以下を入力します。

```
systemctl list-unit-files --type service
```

このコマンドは各サービスユニットの完全な名前 (**UNIT FILE**) を表示し、その後に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については、「サービスステータスの表示」 を参照してください。

例10.1 サービスの一覧表示

現在読み込み済みの全サービスユニットを一覧表示するには、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全サービスユニットファイルを表示して、それらが有効かどうかを判断するには、以下を入力します。

```
~]$ 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ついての詳細情報を表示するには、シェルプロンプトで以下を入力します。

```
systemctl status name.service
```

name を調べたいサービスユニット名 (たとえば、**gdm**) に置き換えます。このコマンドでは、選択されたサービスユニット名の後に、その説明と 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には、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	詳細
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情報の後に、タイムスタンプが続きます。
Main PID	対応するシステムサービスの PID の後に、その名前が続きます。
Status	対応するシステムサービスについての追加情報。
Process	関連プロセスについての追加情報。
CGroup	関連する Control Group (cgroups) についての追加情報。

特定のサービスユニットが実行中かどうかチェックするだけの場合は、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判断するには、以下を入力します。

```
systemctl is-enabled name.service
```

systemctl is-active および **systemctl is-enabled** は両方とも、指定されたサービスユニットが実行中または有効な場合に、**0** の終了ステータス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すべてを一覧表示する方法については、「サービスの一覧表示」 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判断するには、シェルプロンプトで以下を入力します。

```
~]# 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ように命令されるサービスの表示

指定されたサービスの状態を確認するには、シェルプロンプトで次のコマンドを入力します。

```
~]# 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ように命令されるサービスの表示

指定されたサービスの後に開始するよう命令されるサービスを判別するには、シェルプロンプトで次のコマンドを入力します。

```
~]# 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には、**root** でシェルプロンプトに以下を入力します。

```
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、選択され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判断する方法については、「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 Server のサービスユニット名は **httpd.service** になります。このサービスユニットをアクティブ化して、現行セッションで **httpd** デーモンを開始するには、**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** でシェルプロンプトに以下を入力します。

```
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、選択されたサービスを現行セッションで停止します。起動時にサービスユニットを無効にして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判断する方法については、「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名は **bluetooth.service** になります。このサービスユニットを無効にして **bluetoothd** デーモンを現行セッションで停止するには、**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には、**root** でシェルプロンプトに以下を入力します。

```
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、選択されたサービスを現行セッションで停止し、即座に再起動します。重要な点は、選択されたサービスユニットが実行中でない場合、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 に指示するには、**root** で以下のコマンドを実行します。

```
systemctl try-restart name.service
```

特定のシステムサービスでは、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** で以下を入力します。

```
systemctl reload name.service
```

この機能をサポートしないシステムサービスは、このコマンド自体を無視することに注意してください。便宜上、代わりに **systemctl** コマンドは、この機能をサポートしないサービスを再起動する **reload-or-restart** 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10.7 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 では設定を再起動せずかつ処理されたリクエストをアクティブに妨害せずに、設定の編集および再読み込みができます。これを行うには、**root** でシェルプロンプトに以下を入力します。

```
~]# systemctl reload httpd.service
```

10.2.6. サービスの有効化

システムサービスに対応するサービスユニットを起動時に自動的に起動するように設定するには、**root** でシェルプロンプトに以下を入力します。

```
systemctl enable name.service
```

name を有効に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、選択され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の上書きはしません。シンボリックリンクが確実に再度作成されるようにするには、**root** で以下のコマンドを使用します。

```
systemctl reenable name.service
```

このコマンドは、選択され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判断する方法については、「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については、「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

Apache HTTP Server 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

システムサービスに対応するサービスユニットを起動時に自動的に起動しないようにするには、**root** でシェルプロンプトに以下を入力します。

```
systemctl disable name.service
```

name を無効に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、

選択され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ぐことができます。これを実行するには、**root** で以下のコマンドを実行します。

```
systemctl mask name.service
```

このコマンドは、**/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 にアクセスできないようにします。このアクセスを元に戻してサービスユニットをアンマスクするには、**root** で以下を入力します。

```
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判断する方法については、「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については、「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ています。このサービスユニットが起動時に開始しないようにするには、**root** でシェルプロンプトに以下を入力します。

```
~]# systemctl 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、1 つまたは複数の別のサービスの起動（正の依存関係）、あるいは 1 つまたは複数のサービスの停止（負の依存関係）が必要となることがあります。

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ますのでご注意ください。あるサービス実行しているときに、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、最初のサービスは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ときに **sendmail** サービスを実行しようとすると、**systemd** が **postfix** を自動的に停止させます。これら 2 つのサービスは競合しており、同じポートで実行させることができないためです。

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 もしくは Upstart と配布されており、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。これらのランレベルは 0 から 6 までの数字で表示され、システム管理者が特定のランレベルを有効にすると実行されるシステムサービスの選択によって定義されていました。Red Hat Enterprise Linux 7 では、ランレベルの概念は *systemd targets* に代わっています。

Systemd targets は *target units* で表されます。Target units 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 units 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される **graphical.target** ユニットは、GNOME Display Manager (**gdm.service**) や 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化します。同様に **multi-user.target** ユニットは、NetworkManager (**NetworkManager.service**) や D-Bus (**dbus.service**) といった他の必須のシステムサービスを開始して、かつ **basic.target** という別の target unit をアクティブ化します。

Red Hat Enterprise Linux 7 では、以前のシステムのリリースにおける標準ランレベルとほぼ似通った多くの定義済みターゲットが同梱されています。互換性目的で、これらのターゲットを SysV ランレベルに直接マッピングするエイリアスも提供されています。表10.6「SysV ランレベルと systemd ターゲットの比較」 では、SysV ランレベルの完全なリストと対応する systemd ターゲットが表示されています。

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**	非グラフィカルな複数ユーザーシステムを設定します。
3	**runlevel3.target**、**multi-user.target**	非グラフィカルな複数ユーザーシステムを設定します。
4	**runlevel4.target**、**multi-user.target**	非グラフィカルな複数ユーザーシステムを設定します。
5	**runlevel5.target**、**graphical.target**	グラフィカルな複数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で説明するように **systemctl** ユーティリティーを使用します。**runlevel** および **telinit** の各コマンドはシステムで利用可能なままで、期待どおりに機能しますが、これらが含まれているのは互換性が目的であり、使用は避けてください。

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

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、その結果を表示します。デフォルトのターゲットの変更方法については、「デフォルトターゲットの変更」 を参照してください。現在読み込まれているターゲットユニットを一覧表示する方法については、「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、以下を入力します。

```
~]$ systemctl get-default
graphical.target
```

10.3.2. 現在のターゲットの表示

読み込み済みの現在のターゲットユニットをすべて一覧表示するには、シェルプロンプトで以下のコマンドを入力します。

```
systemctl list-units --type target
```

このコマンドは、各ターゲットユニットの完全な名前 (**UNIT**) を表示し、その後にユニットが読み込み済みかどうか (**LOAD**)、高レベル (**ACTIVE**) および低レベル (**SUB**) のユニットのアクティベーション状態、および簡単な説明 (**DESCRIPTION**) が続きます。

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。状態に関係なくすべての読み込み済みユニットを表示したい場合は、**--all** または **-a** のオプションを付けてコマンドを実行します。

```
systemctl list-units --type target --all
```

デフォルトのターゲットの表示方法については、「デフォルトターゲットの表示」 を参照してください。現在のターゲットの変更方法については、「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全ターゲットユニットを一覧表示するには、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設定するには、**root** でシェルプロンプトに以下を入力します。

```
systemctl set-default name.target
```

name をデフォルトで使用したいターゲットユニット名で (たとえば、**multi-user**) 置き換えます。このコマンドは、**/etc/systemd/system/default.target** ファイルを **/usr/lib/systemd/system/*name*.target** へのシンボリックリンクで置き換えます。ここでの *name* は、使用するターゲットユニット名になります。現在のターゲットの変更方法については、「現在のターゲットの変更」 を参照してください。現在読み込まれているターゲットユニットを一覧表示する方法については、「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

システムがデフォルトで **multi-user.target** ユニットを使用するよう設定するには、**root** で以下のコマンドを使用します。

```
~]# 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

現行の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、シェルプロンプトで **root** として以下を入力します。

```
systemctl isolate name.target
```

name を使用したいターゲットユニット名で (たとえば、**multi-user**) 置き換えます。このコマンドは、*name* という名前の付いたターゲットユニットとすべての依存ユニットを開始し、即座にその他すべてを停止します。デフォルトのターゲットの変更方法については、「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すべてを一覧表示する方法については、「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ースをオフにして、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使用します。

```
~]# 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モードを提供し、通常の起動プロセスを完了できない状況における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ースのアクティブ化や、システムに同時に他のユーザーによる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。

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** でシェルプロンプトに以下を入力します。

```
systemctl rescue
```

このコマンドは **systemctl isolate rescue.target** と似ていますが、現在システムにログインしているすべてのユーザーに通知メッセージも送信します。systemd がこのメッセージを送信しないようにするには、このコマンドに **--no-wall** オプションを付けて実行します。

```
systemctl --no-wall rescue
```

緊急モードに入る方法については、「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の状態における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ー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、緊急モードに root パスワードを必要とします。

現在のターゲットを変更し、緊急モードに入るには、**root** でシェルプロンプトに以下を入力します。

```
systemctl emergency
```

このコマンドは **systemctl isolate emergency.target** と似ていますが、現在システムにログインしているすべてのユーザーに通知メッセージも送信します。systemd がこのメッセージを送信しないようにするには、このコマンドに **--no-wall** オプションを付けて実行します。

```
systemctl --no-wall emergency
```

レスキューモードに入る方法については、「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、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 システムで使わ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を理由にまだ利用可能ですが、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。

古いコマンド	新しい コマンド	詳細
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ための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** コマンドはシャットダウンを実行するために **systemctl** ユーティリティーを呼び出しますが、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、ユーザーがシステムシャットダウンの予定に関する警告に対応するための時間をより多く確保することができます。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てマシンの電源を切るには、 **root** でシェルプロンプトに以下を入力します。

```
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、このコマンドはどちらも systemd が 現在システムにログインしているすべてのユーザーに通知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、コマンドに **--no-wall** オプションを付けて実行します。例を示します。

```
systemctl --no-wall poweroff
```

shutdown コマンドの使用
特定の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** で以下のフォーマットでコマンドを使用します。

```
shutdown --poweroff hh:mm
```

ここで、*hh:mm* は 24 時間クロックフォーマットの時間になります。**/run/nologin** ファイルが、新たなログインを防ぐためにシステムがシャットダウンされる 5 分前に作成されます。time 引数が使用される場合、オプションのメッセージである *ウォールメッセージ* をコマンドに追加することができます。

マシンの電源を切らずにシステムを一定の遅延後にシャットダウンし、停止するには、**root** で以下のフォーマットでコマンドを使用します。

```
shutdown --halt +m
```

ここで、*+m* は遅延時間 (分単位) です。**now** キーワードは **+0** のエイリアスです。

保留中のシャットダウンは、以下のように **root** ユーザーでキャンセルできます。

```
shutdown -c
```

追加のコマンドオプションについては、**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、このコマンドは systemd が 現在システムにログインしているすべてのユーザーに通知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、コマンドに **--no-wall** オプションを付けて実行します。

```
systemctl --no-wall reboot
```

10.4.3. システムの一時停止

システムをサスペンドするには、**root** でシェルプロンプトに以下を入力します。

```
systemctl suspend
```

このコマンドは、システムの状態を RAM に保存し、RAM モジュールを除いてマシン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上ではなく RAM に保存されるので、システムのサスペンドモードからの復元は、休止状態からの復元よりも大幅に速くなります。ただし結果として、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については、「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** でシェルプロンプトに以下を入力します。

```
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上に保存されるので、マシンは RAM モジュールに電力を維持する必要がありません。ただしその結果、システムの休止状態からの復元は、サスペンドモードからの復元よりも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** で以下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については、「システムの一時停止」 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 プロトコルを使ってリモートマシン上で実行している systemd と対話操作することができます。**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** オプションを付けて実行すると、このマシンに接続できます。

```
systemctl --host user_name@host_name command
```

user_name をリモートのユーザー名で、*host_name* をマシンのホスト名で、**command** を上記の **systemctl** コマンドのいずれかでそれぞれ置き換えます。指定されたユーザーが SSH プロトコルを使っ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. システム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動作します。細かい調整を行うには、システム管理者は手動でユニットファイルを編集するか、または作成する必要があります。表10.2「Systemd Unit ファイルの場所」 は、システム上の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を一覧表示し、**/etc/systemd/system/** ディレクトリーは、システム管理者が作成するか、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
```

ここで、*unit_name* はユニットの名前を表し、*type_extension* はユニットタイプを特定します。ユニットタイプの詳細な一覧については、表10.1「利用可能な systemd Unit タイプ」 を参照してください。たとえば、通常システムには **sshd.service** 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** ファイルを追加し、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についての詳細情報は、「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/** 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。それら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)。さらに、それらのディレクトリーおよび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るユニットを生成するためのテンプレートとして機能する汎用ユニットファイルを作成できます。詳細は、「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 つのセクションで構成されています。

- [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。これらのオプションはユニットの説明を提供し、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。最もよく使われる [Unit] オプションの一覧については、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] オプションについては、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** および **disable** コマンドで使用されるインストールについての情報が含まれます。[Install] オプションの一覧については、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]	詳細
Description	ユニットの分かりやすい説明です。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 の一覧を提供します。
After[b]	ユニットが開始される順序を定義します。ユニットは、**After** で指定されたユニットがアクティブにされた後にのみ開始されます。**Requires** とは異なり、**After** は指定されたユニットを明示的にアクティブ化しません。**Before** オプションには、**After** とは反対の機能があります。
Requires	他のユニットに依存関係を設定します。**Requires** に一覧表示されるユニットは、該当ユニットと共にアクティブ化されます。必要なユニットのいずれかが開始しない場合、ユニットはアクティブ化されません。

オプション[a]	詳細
Wants	**Requires** よりも強度の弱い依存関係を設定します。一覧表示されるユニットのいずれかが正常に開始しない場合も、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** とは反対の、負の依存関係を設定します。

[a] [Unit] セクションで設定可能なオプションの詳細な一覧は、**systemd.unit(5)** man ページを参照してください。

[b] ほとんどの場合、**After** および **Before** ユニットファイルオプション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、依存関係の並び順は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]	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** – デフォルトの値です。**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** と同様ですが、プロセスは、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** と同様ですが、結果として生じるユニットは、メインプロセスが D-Bus 名を取得した後にのみ起動します。 • **notify** – このタイプは **simple** と同様ですが、結果として生じるユニットは、通知メッセージが sd_notify() 関数で送信された後にのみ起動します。 • **idle** – **simple** と同様ですが、サービスバイナリーの実際の実行はすべてのジョブが終了するまで遅延します。これにより、ステータスの出力とサービスのシェル出力とを混同することを防ぐことができます。
ExecStart	ユニットの起動時に実行され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** の前後に実行されるカスタムコマンドを指定します。**Type=oneshot** は、順次に実行される複数のカスタムコマンドの指定を可能にします。
ExecStop	ユニットの停止時に実行され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されるコマンドまたはスクリプトを指定します。

オプション[a]	詳細
Restart	このオプションが有効にされた状態で、サービスは **systemctl** コマンドによる完全な停止の例外と共に、そのプロセスの終了後に再起動します。
RemainAfterExit	True に設定される場合、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、**Type=oneshot** が設定されている場合にとくに役に立ちます。

[a] [Service] セクションで設定可能なオプションの詳細な一覧は、**systemd.service(5)** man ページを参照してください。

表10.11 [Install] セクションの重要なオプション

オプション[a]	詳細
Alias	ユニットの追加の名前のスペース区切りの一覧を提供します。**systemctl enable** を除くほとんどの **systemctl** コマンドは、実際のユニット名の代わりにエイリアスを使うことができます。
RequiredBy	該当ユニットに依存するユニットの一覧です。このユニットが有効な場合、**RequiredBy** に一覧表示されるユニットは、このユニットについての **Require** 依存関係を取得します。
WantedBy	該当ユニットに弱く依存するユニットの一覧です。このユニットが有効な場合、**WantedBy** に一覧表示されるユニットは、このユニットについての **Want** 依存関係を取得します。
Also	該当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一覧を指定します。
DefaultInstance	インスタンス化されたユニットに制限された状態で、このオプションは、ユニットが有効にされているデフォルト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] セクションで設定可能なオプションの詳細な一覧は、**systemd.unit(5)** man ページを参照してください。

ユニット設定を微調整するために使用できる様々なオプションがあります。例10.17「postfix.service ユニットファイル」 は、システムにインストールされた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/postifix.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] セクションでは、サービスについて説明し、競合するユニットと共に並び順依存関係を指定します。[Service] では、カスタムスクリプトのシーケンスが、ユニットのアクティベーション時、停止時、および再読み込み時に実行されるように指定されます。**EnvironmentFile** は、サービスの環境変数が定義されるロケーションを参照し、**PIDFile** はサービスのメインプロセスの安定した 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一覧表示します。

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。カスタムデーモンを実行したり、(例10.19「sshd サービスの 2 つ目のインスタンスの作成」 に説明されているように) 一部の既存サービスの 2 つ目のインスタンスを作成したり、または SysV init スクリプトをインポート (詳細は 「SysV Init スクリプトのユニットファイルへの変換」 を参照) したりできます。一方、既存ユニットの動作の変更または拡張のみを実行しようとする場合は、「既存のユニットファイルの変更」 の指示に従ってください。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について説明しています。

1. カスタムサービスで実行可能ファイルを用意します。これは、カスタムで作成されたスクリプトである場合も、ソフトウェアプロバイダーが提供する実行可能ファイルである場合もあります。必要な場合は、カスタムサービスのメインプロセスの一定の PID を保持するために 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。ソーススクリプトが (**chmod a+x** を実行して) 実行可能であり、インタラクティブではないことを確認します。

2. **/etc/systemd/system/** ディレクトリーにユニットファイルを作成し、これに正しいファイルパーミッションが含まれることを確認します。**root** で実行します。

   ```
   touch /etc/systemd/system/name.service
   chmod 664 /etc/systemd/system/name.service
   ```

   name を、作成されるサービスの名前に置き換えます。ファイルは実行可能でなくてもよいことに注意してください。

3. 直前のステップで作成された ***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使用できる様々なオプションがあります。「ユニットファイル構造の概要」 を参照してください。以下は、ネットワーク関連サービスのユ

ニット設定の例になります。

```
[Unit]
Description=service_description
After=network.target

[Service]
ExecStart=path_to_executable
Type=forking
PIDFile=path_to_pidfile

[Install]
WantedBy=default.target
```

ここで、

- *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、サービスがネットワークの実行後にのみ起動します。関連するサービスまたはターゲットのスペース区切りの一覧を追加します。
- *path_to_executable* は、実際のサービス実行可能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されます。サービスのメインプロセスは *path_to_pidfile* で指定される PID で作成されます。表10.10「[Service] セクションの重要なオプション」 で他の起動タイプを検索します。
- **WantedBy** は、サービスを起動する必要のあるターゲットを提示します。これらのターゲットは、ランレベルの古い概念の置き換え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して、systemd に対して、新規の ***name*.service** ファイルが存在することを通知します。

```
systemctl daemon-reload
systemctl start name.service
```

警告

新規のユニットファイルの作成または既存のユニットファイルの変更後に常に **systemctl daemon-reload** コマンドを実行します。実行しないと、**systemctl start** または **systemctl enable** コマンドは、systemd とディスク上の実際のサービスユニットファイルの状態の間にある不一致により失敗する可能性があります。

name.service ユニットは、「システムサービスの管理」 に説明されているコマンドでその他のシステムサービスとして管理することができます。

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、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のではなく、これをバックグラウンドで実行する方がスピードが早く、便利です。以下のステップでは、Emacs のユニットファイルを作成する方法を説明し、これをサービスのように処理できるようにします。

1. **/etc/systemd/system/** ディレクトリーにユニットファイルを作成し、これに正しいファイルパーミッションが含まれることを確認します。**root** で実行します。

```
~]# 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、**/usr/bin/emacs** 実行可能ファイルはサービスの起動時にデーモンモードで開始されます。SSH_AUTH_SOCK 環境変数は、ランタイムディレクトリーを表す "%t" ユニット指定子を使用して設定されます。さらにサービスは、予期せずに終了する場合に emacs プロセスを再起動します。

3. 設定を再読み込みし、カスタムサービスを起動するために以下のコマンドを実行します。

```
~]# 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ので、すべての標準的な **systemctl** コマンドが使用できます。例えば、**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することができます。

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、サービスの複数のインスタンスを設定し、実行しなければならないことが多くあります。これは、元のサービス設定ファイルのコピーを作成し、サービスの主なインスタンスとの競合を避けるために特定のパラメーターを変更することによって実行されます。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される **sshd_config** ファイルのコピーを作成します。

```
~]# cp /etc/ssh/sshd{,-second}_config
```

2. 直前のステップで作成された **sshd-second_config** ファイルを編集し、異なるポート番号と PID ファイルを 2 つ目のデーモンに割り当てます。

```
Port 22220
PidFile /var/run/sshd-second.pid
```

Port および **PidFile** オプションの詳細は、**sshd_config**(5) man ページを参照してください。選択するポートがその他のサービスで使用されていないことを確認します。PID ファイルはサービスの実行前になければならない訳ではありません。これ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. 直前のステップで作成された **sshd-second.service** を変更します。

   a. **Description** オプションを変更します。

```
Description=OpenSSH server second instance daemon
```

   b. **After** オプションで指定されたサービスに sshd.service を追加し、2 つ目のインスタンスが、最初のインスタンスが起動した後にのみ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. **-f /etc/ssh/sshd-second_config** パラメーターを **sshd** コマンドに追加し、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. 上記の変更後、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、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。そうしないと、sshd の 2 番目のインスタンスはポートへのバインドの際に拒否されます。

```
~]# semanage port -a -t ssh_port_t -p tcp 22220
```

6. sshd-second.service を有効にし、これがブート時に自動的に起動するようにします。

```
~]# systemctl enable sshd-second.service
```

systemctl status コマンドを使用して sshd-second.service が実行中であるかどうかを確認します。さらに、ポートがサービスに接続することによって適切に有効にされ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中の場合、sshd の 2 番目のインスタンスへの接続を許可するためにそれが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については、以下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については、Is there any useful information about writing unit files? を参照してください。

systemd が開始したサービスへの制限の設定については、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。これらの制限は、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された制限を無視する点に注意してください。これらのファイルに定義された制限は、ログインセッションの開始時に PAM によって設定されますが、**systemd** によって開始されたデーモンは、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、どこか他の場所で変換がすでに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」 に説明されるようにユニットファイルを作成することができます。init スクリプトはサービスのタイプによって大幅に異なるため、この章で概略されているよりも多くの設定オプションを変換に使用する必要があるかもしれません。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を示しています。

```
#!/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であるため、残りの部分は別の init ファイルにはない場合があります。**### BEGIN INIT INFO** および **### END INIT INFO** 行で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 ヘッダーが指定されている場合、これには、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しようと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。例10.17「postfix.service ユニットファイル」 で作成される postfix ユニットについて参照してください。

サービス説明の検索

#description で始まる行でスクリプトについての説明情報をを見つけてください。この説明をサービス名と共に、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 ヘッダーでは、**#Short-Description** 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。それらのほとんどは 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	詳細	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)。ただしこれは、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、不要になりました。	–

LSB オプション	詳細	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れら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 で 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構成します。失敗した Should-Start 依存関係はサービス起動に影響を与えません。	**After**、**Before**
Required-Stop、**Should-Stop**	負の依存関係を構成します。	**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 つの数値が含まれます。最も重要な値は、サービスが起動するデフォルトのランレベルを表す最初の番号です。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、それらのターゲットをユニットファイルの [Install] セクションの **WantedBy** オプションに一覧表示します。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。これらは Red Hat Enterprise Linux 7 の multi-user.target および graphical.target に変換されます。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」 で説明されているようにこれら両方を指定する必要はないことに注意してください。また、デフォルトおよび禁止されているランレベルについては、LSB ヘッダーの **#Default-Start** および **#Default-Stop** 行に情報がある場合があります。

#chkconfig 行で指定される他の 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。これらの値は、init スクリプトが読み込まれる場合は systemd によって解釈されますが、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機能ライブラリーを読み込み、設定、環境、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** で始まる行で指定され、これは **EnvironmentFile** ユニットファイルオプションに変換されます。**#pidfile** init スクリプト行で指定される 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、サービス実行可能ファイルへのパス、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**、または **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ために 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され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。さらに詳しく見ると、上記のコードでは複数の外部ファイルおよびディレクトリーが記述されています。主なサービス実行可能ファイルの **/usr/sbin/postfix**、**/etc/postfix/** および **/var/spool/postfix/** 設定ディレクトリー、および **/usr/sbin/postconf/** ディレクトリーです。

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、**ExecStart**、**ExecStartPre**、**ExecStartPost**、**ExecStop**、および **ExecReload** オプションでカスタム実行可能ファイルを実行することができます。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** ユニットファイルについては、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、それらを変更する必要はありません。一方、新規の環境では、サービス実行可能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れら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、パッケージの更新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。詳細は、「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、ユニットファイルに変更をすぐに適用するために必要な依存関係ツリー全体を再作成します。または、以下のコマンドを使って同じ結果を得ることができます。

```
init q
```

変更され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、このサービスは、新たな設定を受け入れるために再起動する必要があります。

```
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 (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、**/etc/rc.d/init.d/network** init スクリプトファイルは変更せず、代わりに新しいディレクトリ **/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。そして変更した値をドロップインファイルに入力します。注記: **systemd** は **network** サービスを **network.service** として認識します。作成したディレクトリを **network.service.d** と命名す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、まず **/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。

直前のステップで作成されたディレクトリーに設定ファイルを作成します。ファイル名は **.conf** という接尾辞で終了する必要があることに注意してください。以下を入力します。

```
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
```

ここで、*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。別の例は、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 つのタスクにのみフォーカスした小規模な設定ファイルを作成することを推奨します。これらのファイルは、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、リンクしたりすることができます。

ユニットに変更内容を適用するには、**root** で以下を実行します。

```
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、以下のステップを実行します。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。

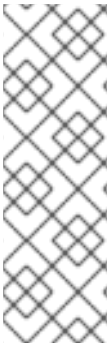
```
~]# mkdir /etc/systemd/system/httpd.service.d/
~]# touch /etc/systemd/system/httpd.service.d/custom_script.conf
```

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、以下のテキストを **custom_script.conf** ファイルに挿入します。

```
[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/** のファイルに優先されます。そのため、設定ファイルに **Description** または **ExecStart** などの 1 回のみ指定できるオプションが含まれる場合、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モニタリング」 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永続する変更を加えるには、まずファイルを **/etc/systemd/system/** ディレクトリーにコピーします。これを実行するには、**root** で以下のコマンドを実行します。

```
cp /usr/lib/systemd/system/name.service /etc/systemd/system/name.service
```

ここで、*name* は、変更する必要のあ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劇用されます。

テキストエディターでコピーされたファイルを開き、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止できます。指定しない場合、タイムアウトは通常のサービスには 90 秒、SysV と互換性のあるサービスには 300 秒に、それぞれデフォルト設定されています。

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。

1. **httpd** ユニットファイルを **/etc/systemd/system/** ディレクトリへコピーします。

   ```
   cp /usr/lib/systemd/system/httpd.service
   /etc/systemd/system/httpd.service
   ```

2. ファイル **/etc/systemd/system/httpd.service** を開き、 **TimeoutStartUSec** 値を **[Service]** セクションで指定します。

   ```
   ...
   [Service]...
   PrivateTmp=true
   TimeoutStartSec=10
   ```

```
[Install]
WantedBy=multi-user.target...
```

3. **systemd** デーモンを再読み込みします。

```
systemctl daemon-reload
```

4. **オプション。** 新しいタイムアウト値を検証します。

```
systemctl show httpd -p TimeoutStartUSec
```

注記

タイムアウト値をグローバルに変更するには、***DefaultTimeoutStartSec*** を **/etc/systemd/system.conf** ファイルに入力します。「systemd の概要」 を参照してください。

上書きされたユニットのモニタリング

上書きされたか、または変更されたユニットファイルの概要を表示するには、以下のコマンドを使用します。

```
systemd-delta
```

たとえば、上記のコマンドの出力は以下のようになります。

```
[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一覧表示します。ファイルが上書きされる場合、**systemd-delta** はデフォルトで、**diff** コマンドの出力に似た変更の要約を表示します。

表10.13 systemd-delta の相違タイプ

タイプ	詳細
[MASKED]	マスクされたユニットファイルです。ユニットのマスクについての説明は、「サービスの無効化」 を参照してください。
[EQUIVALENT]	元のファイルを上書きするが、コンテンツに相違のない変更されていないコピーです。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されるファイルです。
[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ないファイルです。

システムの更新後に、カスタム設定で上書きされているデフォルトユニットへの更新があるかどうかを確認するために **systemd-delta** を実行することができます。さらに、出力を特定の相違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

ランタイム時に単一の設定ファイルから複数のユニットをインスタンス化することができます。"@" 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** 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には以下の方法で名前が付けられます。

```
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を表します。*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は、ユニットのすべてのインスタンスに共通の設定オプションを持つ同一のテンプレートを参照できます。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
```

以下は、ユニットファイルの **Wants** 設定です。

```
Wants=getty@ttyA.service,getty@ttyB.service
```

最初に、systemd に指定されたサービスユニット検索させます。該当するユニットが見つからない場合、「@」とタイプ接尾辞の間の部分は無視され、systemd は **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、これらは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、テンプレートユニットでとくに役立つユニット指定子を一覧表示します。

表10.14 重要なユニット指定子

ユニット指定子	意味	詳細
%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	ユニット設定が読み込まれる時点の実行中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** か、または特権のないユーザーの XDG_RUNTIME_DIR 変数の値のいずれかになります。

ユニット指定子の詳細な一覧については、**systemd.unit(5)** man ページ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 がインスタンス化される場合、**Description=** は **Getty on ttyA** および **Getty on ttyB** として解決されます。

10.7. 関連資料

systemd および Red Hat Enterprise Linux 7 でのその使用については、以下に一覧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 ページでは、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 ユニットファイルについての詳細情報と、利用可能なすべての設定オプションが説明されています。

- **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フォーマットが説明されています。

- **systemd.target**(5) — **systemd.target** の man ページでは、ターゲットユニットファイルのフォーマットが説明されています。

- **systemd.kill**(5) — **systemd.kill** の man ページでは、プロセスを強制終了する手順の設定について記述しています。

オンラインのドキュメント

- 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』 は、このシステムにおけるネットワークインターフェースやネットワーク、ネットワークサービスの設定および管理に関する情報が説明されています。**hostnamectl** ユーティリティーの概要のほか、これを使ってコマンドラインでホスト名を表示、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ついての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。**logind** サービスの概要とその最重要機能を挙げているほか、**loginctl** ユーティリティーを使ってアクティブなセッションを一覧表示し、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、または OpenShift などの様々なサービスと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』 は、AMD64 および Intel 64 システム、64-bit IBM Power Systems サーバー、および IBM System z にシステムをインストールする方法を説明しています。また、キックスタートインストール、PXE インストール、および VNC プロトコルを使ったインストールなどの高度なインストール方法もカバーされています。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トについても説明されています。

- Red Hat Enterprise Linux 7 セキュリティガイド — Red Hat Enterprise Linux 7 の 『セキュリティガイド』 は、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ユーザーおよ

び管理の役に立ちます。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 Home Page — このプロジェクトのホームページでは、systemd についての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。**localectl** ユーティリティーを使って現在のロケールを表示、利用可能なロケールを一覧表示、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の表示、利用可能なキーマップの一覧表示、コマンドラインで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っ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** コマンドを使って管理者権限を取得する方法が説明されています。

- 12章*OpenSSH* は、SSH サーバーの設定方法や、**ssh**、**scp**、および **sftp** クライアントユーティリティーを使ってこのサーバーにアクセスする方法が説明されています。

- 22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ってログエントリーを表示する方法のドキュメント、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、システム管理者がどうすればこのシステムを視覚障害を持つユーザーに有用なシステムとして設定できるかを説明します。

Orca は画面から情報を読み取り、以下を使用しているユーザーにそれを伝達します。

- 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には、システム管理者は以下を行う必要があります。

- **brltty** サービスを 「**brltty** サービスの設定」 に説明されているとおりに設定します。
- **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 「**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」 に記載されているとおりに入れます。
- **Festival** 音声合成装置を 「**Festival Speech Synthesis System** の有効化」 に記載されているとおりに有効にします。

11.1. BRLTTY サービスの設定

点字ディスプレイは、**brltty** サービスを使っ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 サービスを有効にする

点字ディスプレイは **brltty** が実行され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
```

ユーザーに点字ディスプレイの使用を許可する

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。いずれも同じ結果が生じます。**/etc/brltty.conf** 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。

手順11.1 /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てください。
2. ユーザーを指定します。
   a. 1 人または複数のユーザーを個別に指定するには、以下の行にユーザーを列記します。

```
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、以下の行に名前を入力します。

```
api-parameters Auth=group:group  # Allow some local group
```

手順11.2 /etc/brlapi.keyを使って点字ディスプレイへのアクセスを設定する

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. 個別のユーザーを指定するには、以下を実行します。

```
~]# chown user_1 /etc/brlapi.key
```

b. グループを指定するには、以下を実行します。

```
~]# 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 にある **braille-driver** ディレクティブは、点字ディスプレイのドライバーの、2 文字から成るドライバー識別コードを指定します。

手順11.3 点字ドライバーの設定

- 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. 自動検知機能を使用する場合は **braille driver** をデフォルト設定の **auto** のままにしておきます。

```
braille-driver auto  # autodetect
```

警告

自動検知機能はすべてのドライバーに実行されます。したがって、時間がかかったり検出に失敗したりする可能性もあります。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. 自動検知機能を使用したくない場合は、**braille-driver** ディレクティブから必要な点字ドライバーの識別コードを指定してください。

/etc/brltty.conf に記載されている一覧から、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、

```
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ます。自動検知はそれらの中で実行されます。

点字装置の設定

/etc/brltty.conf にある **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」 を参照してください) 。

表11.1 点字装置の種類と対応する構文

点字装置の種類	種類の構文
シリアル装置	serial:path [a]
USB 装置	[serial-number] [b]
Bluetooth 装置	bluetooth:address

[a] 相対パスは **/dev** にあります。

[b] ここでのブラケット (**[ ]**) は、オプションであることを示しています。

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
```

複数の装置をコンマで区切って設定することもできます。それぞれの装置が順番に精査されます。

警告

装置を USB シリアルアダプターで接続している場合、**braille-device** から **usb:** への設定は動作しません。その場合は、カーネルがこのアダプター用に作成した仮想シリアル装置を指定します。仮想シリアル装置は、以下のように表示されます。

```
serial:ttyUSB0
```

実際の装置名は、以下のコマンドにより、装置プラグのカーネルメッセージから確認できます。

```
~]# dmesg | fgrep ttyUSB0
```

点字ディスプレイに特定のパラメーターを設定する

点字ディスプレイに特定のパラメーターを設定する必要がある場合は、**/etc/brltty.conf** にある **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へパスします。**/etc/brltty.conf** にある一覧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 にある **text-table** ディレクティブは、記号のエンコードに使用す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。

手順11.4 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ときに自動選択機能を使用するかどうか決定します。
2. a. 自動選択機能を使用する場合は **text-table** をデフォルト設定の **auto** のままにしておきます。

   ```
   text-table auto  # locale-based autoselection
   ```

   これで、**en-nabcc** へのフォールバックを備えたローカルベースの自動選択が実行されます。

   b. 自動選択機能を使用したくない場合は、**/etc/brltty.conf** の一覧から必要な **text-table** を選択してください。

   たとえば、米国英語のテキストテーブルを使用する場合は、以下のようになります。

   ```
   text-table en_US  # English (United States)
   ```

Contraction テーブルの設定

/etc/brltty.conf にある **contraction-table** ディレクティブは、略語のエンコードに使用す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 の一覧から必要な **contraction-table** を選択します。

たとえば、米国英語、グレード 2 の contraction テーブルを使用する場合は、以下のようになります。

```
contraction-table en-us-g2  # English (US, grade 2)
```

警告

指定しない場合、contraction テーブルは使用されません。

11.2. 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

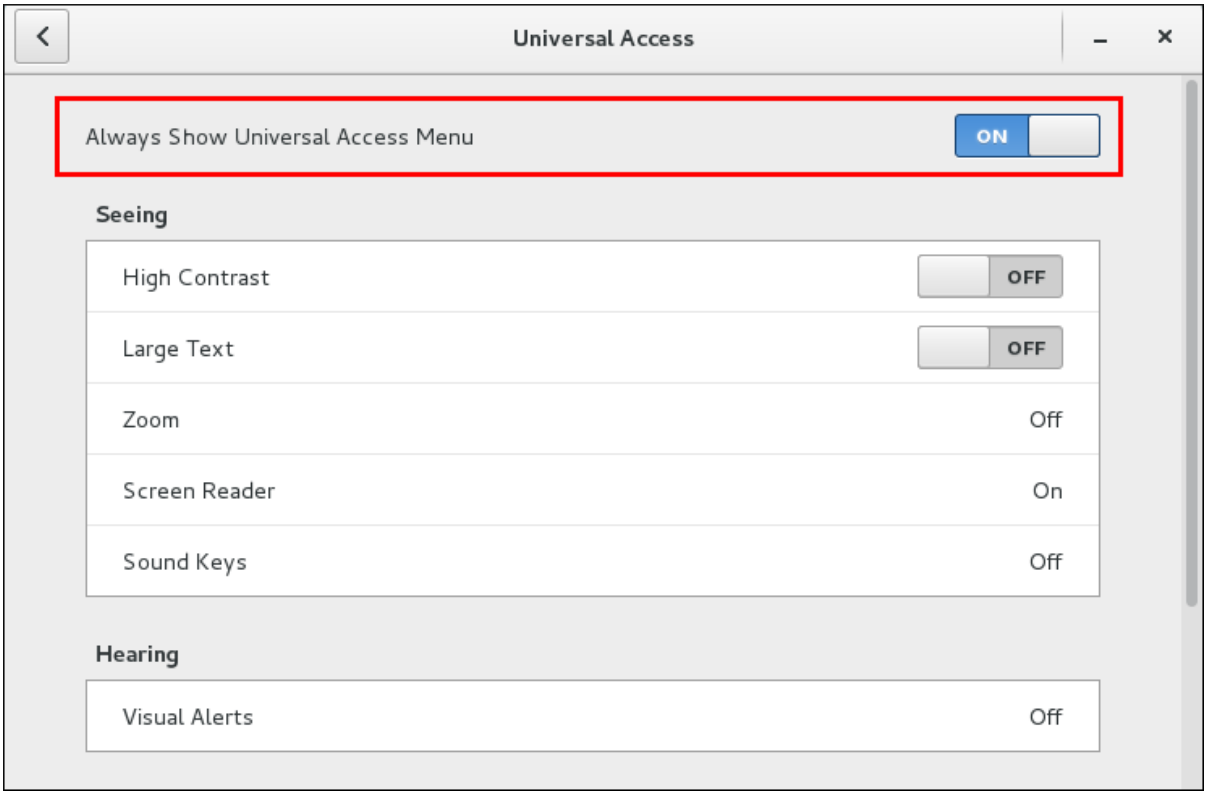
Orca スクリーンリーダーのスイッチを入れるには、**Super+Alt+S** を同時に押します。するとトップバーに **ユニバーサルアクセスメニュー (Universal Access Menu)** アイコンが表示されます。

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。アイコンが表示されないと、視覚障害を持つユーザーの利用に支障をきたします。システム管理者は**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れば、アイコンが利用できない状態を防げます。**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と、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、アイコンはトップバーに表示されます。

手順11.5 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 設定メニューを開き、**Universal Access** をクリックします。
2. **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** と **Festival Speech Synthesis System** (Festival) は、異なる方法で音声を合成します。ユーザーの中にはデフォルトの **eSpeak** よりも Festival を好む方がいるかもしれません。Festival を有効化する手順は以下のとおりです。

手順11.6 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. 新規 **festival.service** ファイルが存在していることを **systemd** に通知します。

```
~]# 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、以下の一覧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: セキュアシェル) 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** や **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、侵入者が暗号化されていないパスワードを入手するための接続が難しくなります。

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の低いターミナル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** のような旧式プログラムの代わりとなります。これら旧式アプリケーションはクライアントとサーバー間で送信するパスワードを暗号化しないので、可能な限り使用を避ける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 (openssh) と共に、OpenSSH サーバー (openssh-server) およびクライアント (openssh-clients) パッケージが含まれています。OpenSSH パッケージでは、いくつかの重要な暗号化ライブラリーをインストールし、OpenSSH の暗号化通信を可能にする OpenSSL パッケージ (openssl-libs) が必要になる点に注意してください。

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 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対象となる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* を使用して行われます。パケットスニファは、ネットワークを通過するパケットをキャプチャしてその内容を分析するかなり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。この戦略が成功すると、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ポイントします。後者の場合は、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、悲惨な結果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脅威を大幅に軽減することができます。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間の全通信は暗号化されます。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. 主な機能

SSH プロトコルは、以下のような保護手段を提供します。

対象のサーバーとして装うことができません

初回接続後に、クライアントは以前に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。

認証情報の取得はできません

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。

通信の傍受はできません

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、SSH プロトコルは以下のようなオプションも提供します。

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します

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することが可能です。

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します

SSH プロトコルは、送受信するものすべてを暗号化します。SSH サーバーは、*ポート転送* と呼ばれる手法を使用して、POP のようなセキュアでないプロトコルをセキュアにするための経路となり、システムとデータ全体のセキュリティーを強化することができます。

セキュアなチャンネルを作成できます

OpenSSH サーバーとクライアントは、サーバーとクライアントマシン間のトラフィックに対し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設定できます。

Kerberos 認証に対応します

OpenSSH の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: 汎用セキュリティサービス API) 実装を使用して認証を行うよう設定できます。

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 (新規) の 2 つの種類があります。Red Hat Enterprise Linux 7 の OpenSSH スイートでは、SSH バージョン 2 を使用します。バージョン 2 は、バージョン 1 での既知のエクスプロイトに対して脆弱性のない拡張された鍵交換アルゴリズムを使用します。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ために役立ちます。

1. 暗号化ハンドシェイクが行われ、クライアントは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間の接続のトランスポート層は、対称暗号方式を使用して暗号化されます。

3. クライアントは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役割は、認証時およびその後の通信中における 2 つのホスト間の安全でセキュアな通信を容易にすることです。トランスポート層は、データの暗号化と復号化を処理し、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役割を果たします。また、トランスポート層は、情報を圧縮して転送を高速化します。

SSH クライアントがサーバーにコンタクトすると、基本情報が交換されるため両システムはトランスポート層を適正に構築することができます。以下は、こうした基本情報の交換中に発生するステップです。

- 鍵が交換される。
- 公開鍵暗号化アルゴリズムが決定される。
- 対称暗号化アルゴリズムが決定される。
- メッセージ認証アルゴリズムが決定される。
- ハッシュアルゴリズムが決定される。

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、サーバーのホスト鍵はクライアントには未知であり、接続は成立しません。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は、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。この後に、ホスト鍵が一致しなくなった場合には、ユーザーは接続前にクライアントの保存してあるバージョンを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た後、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、認証と今後こ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、クライアントはサポートされているいずれかの方法を使っ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できるように設定可能なため、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き、サポートする暗号化方法を決定することができ、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した後は、*多重化*[1] と呼ばれる手法により複数のチャンネルが開かれます。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、各チャンネルに別々の番号が接続の両端に割り当てられます。クライアントが新しいチャンネルを開こうとする時には、クライアントは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ために使用されます。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* もサポートしているため規則的な方法でデータを送受信することができます。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、データはチャンネル上で送信されません。

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することができます。

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ために非常に重要となる、Diffie-Hellman 鍵交換に使用される Diffie-Hellman グループが格納されています。SSH セッションの始めで鍵が交換される時、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場合に使用されます。
/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** が存在する場合には、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** デーモン用の設定ファイルです。
/etc/ssh/ssh_host_ecdsa_key	**sshd** デーモンで使用する ECDSA 秘密鍵です。
/etc/ssh/ssh_host_ecdsa_key.pub	**sshd** デーモンで使用する ECDSA 公開鍵です。
/etc/ssh/ssh_host_rsa_key	**sshd** デーモンにより使用される 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** デーモンにより使用される 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** デーモン用の 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一覧が含まれています。クライアントがサーバーに接続する時、サーバーは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 秘密鍵を格納しています。
~/.ssh/id_ecdsa.pub	ユーザーの ECDSA 公開鍵です。
~/.ssh/id_rsa	**ssh** により使用される 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** により使用される SSH プロトコルのバージョン 2 用の RSA 公開鍵です。

ファイル	詳細
~/.ssh/known_hosts	ユーザーがアクセスする SSH サーバーのホスト鍵が格納されています。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ために非常に重要です。

警告

SSH サーバーを設定する場合、**/etc/ssh/sshd_config** ファイルの **UsePrivilegeSeparation no** ディレクティブを使用することで、**Privilege Separation** 機能をオフにしません。**Privilege Separation** をオフにすると、多くのセキュリティー機能が無効となり、サーバーは、セキュリティー上の潜在的な脆弱性にさらされ、攻撃対象となります。**UsePrivilegeSeparation** に関する詳細は、**sshd_config**(5) の 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。

SSH 設定ファイルに使用可能な各種ディレクティブについての情報は、**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については「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、シェルプロンプトで **root** として以下を入力します。

```
~]# 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
```

デーモンがブート時に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ており、これは静的設定のネットワークインターフェースやデフォルトの **ListenAddress 0.0.0.0** オプションの場合は十分なものです。**ListenAddress** ディレクティブで別のアドレスを指定して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上の依存関係を **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、以下のコマンドで **systemd** マネージャー設定をリロードします。

```
~]# systemctl daemon-reload
```

Red Hat Enterprise Linux でシステムサービスを管理する詳細情報については、*10章systemd によるサービス管理* を参照してください。

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。そのため、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。全一覧は 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、システムを再インストールする時には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お勧めします。さもないと、ユーザーのパスワードは SSH を使用した 1 回のセッションでは保護されても、その後に Telnet を使用してログインした時には結局傍受されてしまうためです。無効にするサービスは、**telnet**、**rsh**、**rlogin**、**vsftpd** などがあります。

vsftpd サービスの設定方法については、「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* を参照してください。

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、**/etc/ssh/sshd_config** の設定ファイルを **vi** や **nano** などのテキストエディターで開き、**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、以下のステップ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。

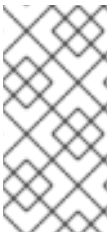
NFS がマウントされたホームディレクトリーで鍵ベースの認証を使うには、最初に **use_nfs_home_dirs** SELinux ブール値を有効にします。

```
~]# setsebool -P use_nfs_home_dirs 1
```

Red Hat Enterprise Linux 7 は、デフォルトでは SSH プロトコル 2 と 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)。

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した場合に、以前に生成された鍵ペアを維持したい時は、**~/.ssh/** ディレクトリーをバックアップします。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、**root** を含むシステム上の全ユーザーが実行できます。

12.2.4.1. 鍵ペアの生成

以下のステップに従って 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以下を入力します。

```
~]$ 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、新規作成された鍵用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、アカウントにログインする時と同じパスワードは使用しないでください。

   この後に、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だった MD5 鍵フィンガープリントを取得するには、**-E md5** オプションと共に **ssh-keygen** コマンドを使用します。

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。必要な場合は、以下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、以下のフォーマットでコマンドを発行します。

```
ssh-copy-id user@hostname
```

これで、インストールされていない場合は最近変更された **~/.ssh/id*.pub** 公開鍵がコピーされます。別の方法としては、公開鍵ファイル名を以下のように指定します。

```
ssh-copy-id -i ~/.ssh/id_rsa.pub user@hostname
```

これで **~/.ssh/id_rsa.pub** のコンテンツが接続先のマシン上にある **~/.ssh/authorized_keys** ファイルにコピーされます。このファイルがすでに存在する場合は、この鍵はファイル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、以下のステップにしたがいます。

1. ECDSA 鍵ペアを生成するには、シェルプロンプトで以下を入力します。

```
~]$ 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、アカウントにログインする時と同じパスワードは使用しないでください。

この後に、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。必要な場合は、以下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、以下のフォーマットでコマンドを発行します。

```
ssh-copy-id USER@hostname
```

これで、インストールされていない場合は最近変更された **~/.ssh/id*.pub** 公開鍵がコピーされます。別の方法としては、公開鍵ファイル名を以下のように指定します。

```
ssh-copy-id -i ~/.ssh/id_ecdsa.pub USER@hostname
```

これで **~/.ssh/id_ecdsa.pub** のコンテンツが接続先のマシン上にある **~/.ssh/authorized_keys** にコピーされます。このファイルがすでに存在する場合は、この鍵はファイル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ことが重要です。

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。インストールされていない場合には、「パッケージのインストール」 で Red Hat Enterprise Linux での新規パッケージのインストール方法について確認してください。

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一覧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 Windows または Command キーで、スペースバーの左側にあります。

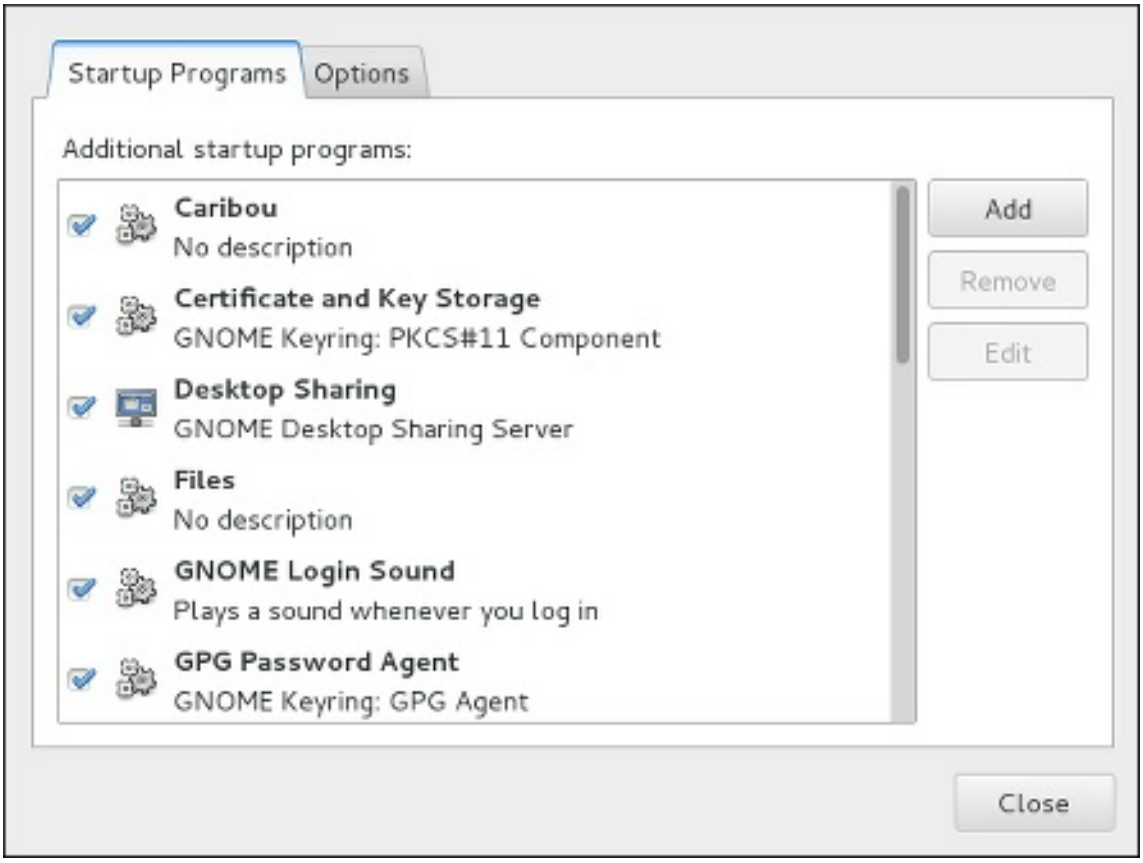


図12.1 自動起動するアプリの設定

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。

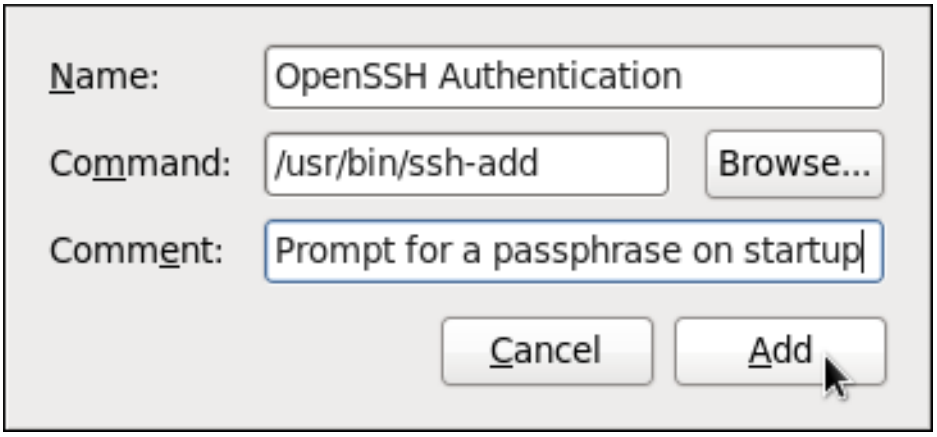


図12.2 新規アプリケーションの追加

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。

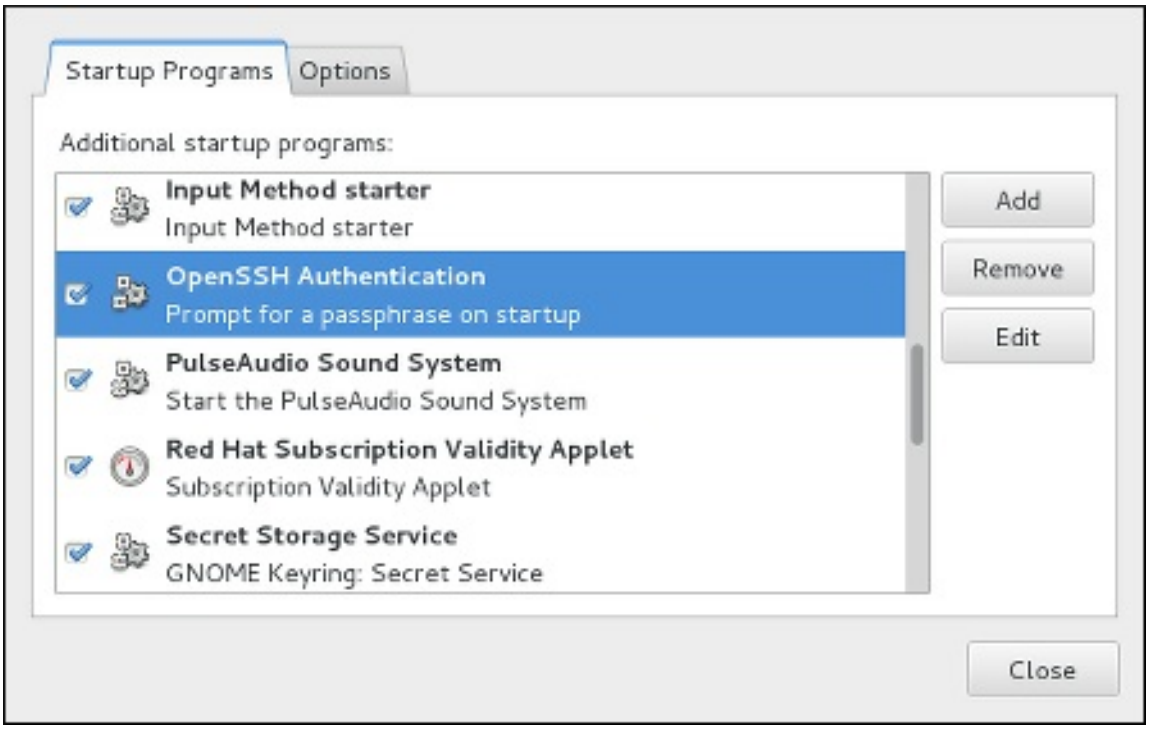


図12.3 アプリケーションの有効化

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。

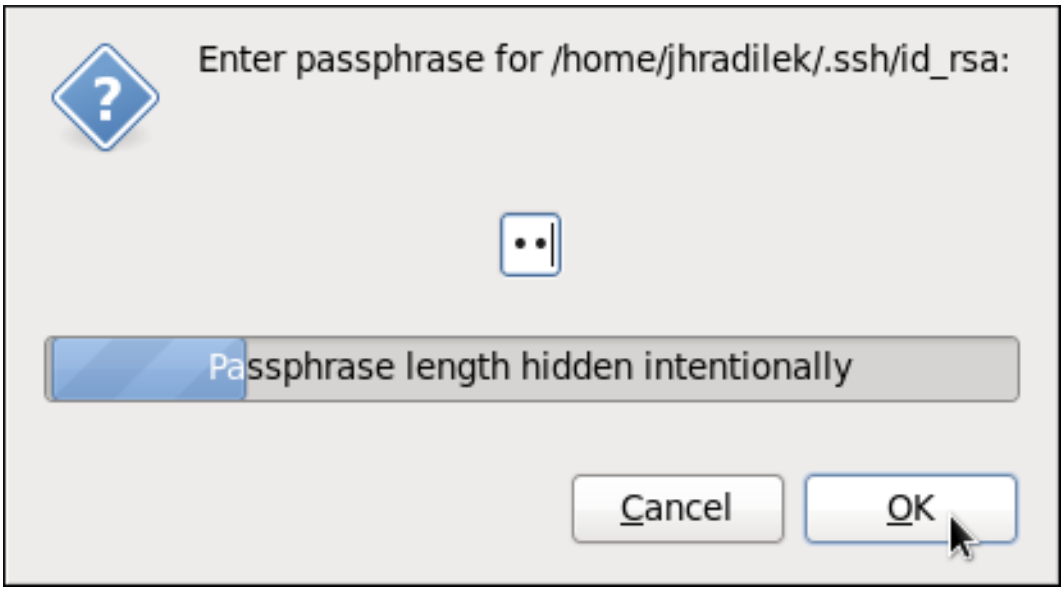


図12.4 パスフレーズの入力

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 (Red Hat Enterprise Linux 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」 を参照して下さい)。

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したい場合には、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。

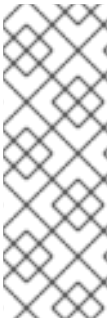
```
~]$ ssh USER@penguin.example.com
```

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だった MD5 鍵フィンガープリントを取得するには、**-E md5** オプションと共に **ssh-keygen** コマンドを使用します。以下に例を示します。

```
~]# 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一覧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。設計に関しては、暗号化されたセキュアな接続を使用する以外は **ftp** と似ています。

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* の内容を一覧表示します。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 *directory*	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。
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. セキュアシェルの高機能

セキュアなコマンドラインインターフェースは、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:
```

セキュアなシェルプロンプトから X プログラムが実行されると、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についての詳しい情報は 「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、**印刷設定** 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。

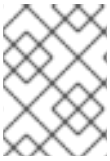
```
~]$ system-config-printer &
```

印刷設定 ツールが表示され、リモートユーザーはリモートシステム上で印刷の設定を安全に行うことができます。

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っ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。

システム管理者がポート転送に懸念がある場合は、**/etc/ssh/sshd_config** にある **AllowTcpForwarding** の行に **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により、サーバー上でこの機能を無効にすることができます。

12.5. 関連資料

Red Hat Enterprise Linux 上で OpenSSH を設定し、接続する詳細方法については、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

- **sshd**(8) — **sshd** デーモンの man ページは、利用可能なコマンドラインオプションを説明し、サポートされる設定ファイルおよびディレクトリーの完全な一覧を提供します。
- **ssh**(1) — **ssh** クライアントアプリケーション の man ページは、利用可能なコマンドラインオプションとサポートされる設定ファイルおよびディレクトリーの完全な一覧を提供します。
- **scp**(1) — **scp** ユーティリティーの man ページは、このユーティリティーの詳細な説明と使用方法を説明しています。
- **sftp**(1) — **sftp** ユーティリティーの man ページです。
- **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って **ssh** が使用する認証鍵を生成、管理、変換する詳細な方法を説明しています。
- **ssh_config**(5) — **ssh_config** の man ページでは、利用可能な SSH クライアントオプションが説明されています。
- **sshd_config**(5) — **sshd_config** のman ページは、利用可能な SSH デーモン設定オプションを詳細に説明しています。

オンラインのドキュメント

- OpenSSH Home Page — さらに詳しいドキュメントや FAQ、メーリングリストへのリンク、バグレポート、その他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 — さらに詳しいドキュメントや FAQ、メーリングリストへのリンク、その他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** コマンドを使って管理者権限を取得する方法が説明されています。
- 10章*systemd によるサービス管理* では systemd に関する詳細情報と、**`systemctl`** コマンドを使っ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。

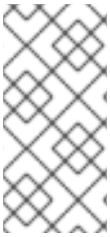
---

[1] 多重接続は、共有されている共通の媒体上で送信される信号で構成されます。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います。**/etc/sysconfig/vncserver** 設定ファイルは、**/etc/systemd/system/vncserver@.service** に代わりました。

13.1. VNC サーバー

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 が実行され、VNC デスクトップ上でウィンドウマネージャーが起動します。**vncserver** を使うと、どこからでもいくつものクライアントがアクセス可能なマシン上で独立したセッションを同時に実行できます。

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、以下のコマンドを **root** で実行します。

```
~]# yum install tigervnc-server
```

13.1.2. VNC サーバーの設定

VNC サーバーでは、表示設定、ネットワークアドレスおよびポート設定、セキュリティー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)。

手順13.1 単一ユーザー用の VNC 画面の設定

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'** は画面番号に置き換えられるため、ファイル名に画面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。詳細については、「ユーザー 2 人用に VNC サーバーを設定する」 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。**-geometry** 引数は、作成される VNC デスクトップのサイズを指定します。デフォルトでは、**1024x768** に設定されます。

```
ExecStart=/usr/sbin/runuser -l USER -c "/usr/bin/vncserver %i -
geometry 1280x1024"
PIDFile=/home/USER/.vnc/%H%i.pid
```

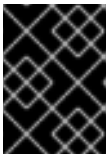
3. 変更を保存します。

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. 設定ファイルでユーザー用のパスワードを設定します。まず、**root** から *USER* に切り替わ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にアクセスがあれば、誰でもプレーンテキストでパスワードを見つけることができます。

「VNC サーバーの起動」 に進みます。

13.1.2.1. ユーザー 2 人用に VNC サーバーを設定する

同一マシン上で複数のユーザーを設定したい場合は、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。手順

13.1「単一ユーザー用の VNC 画面の設定」 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** は **systemd** が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、定義された画面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 サーバーを設定する

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することができます。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構成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。セットアップの条件は gdm、vnc、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
```

手順13.2 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 を有効化します。

```
[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、**-query localhost** オプションが xdmcp セッションの各 Xvnc インスタンスクエリー localhost を作成します。 **-depth** オプションは作成される VNC デスクトップのピクセル深度 (ビット) を指定します。使用可能な値は 8、15、16、24 で、他の値はアプリケーションの予期せぬ動作につながることがあります。

3. ファイル **/etc/services** を編集し、サービスを定義します。これを行うには、以下のスニペットを **/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、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てください。

```
# 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セットアップをテストします。

```
# vncviewer localhost:5950
```

コマンドは、パスワードを求められない 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。

セットアップのための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することができます。

```
~]# 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、サービスを停止することができます。

```
~]# 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画面 **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。

手順13.3 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** で以下のコマンドを入力します。

```
~]# 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画面 **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については、「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ースを表示し、**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や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** のパラメーターを一覧表示するには、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 サーバーが設定されると、VNC サーバーを任意の VNC ビューアーに接続できます。

手順13.4 SSH を使用した VNC サーバーへの接続

1. 引数なしで **vncviewer** コマンドを入力します。**VNC Viewer: Connection Details (VNC ビューアー: 接続の詳細)** 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** を押してメインメニューに戻ります。
3. 接続するアドレスおよび画面番号を入力します。

   ```
   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、画面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。

手順13.5 CLI を使用した VNC サーバーへの接続

1. アドレスと画面番号を引数として **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、画面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画面番号の合計を計算します。たとえば、2 つ目の画面は次のようになります。2 + 5900 = 5902

0 ～**3** の画面の場合は、以下で説明されているように **service** オプションを使用して VNC サービスの **firewalld** のサポートを使用します。**3** よりも大きい画面番号の場合は、手順13.7「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。

手順13.6 firewalld での VNC サービスの有効化

1. **firewalld** 設定について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
```

これらの変更は次回のシステム起動後に維持されないことに注意してください。ファイアウォールに永久的な変更を行うには、**--permanent** オプションを追加してコマンドを繰り返します。ファイアウォールリッチ言語コマンドの使用の詳細については、『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範囲を開くには、**firewall-cmd** コマンドラインツールに **--add-port** オプションを使用します。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。

手順13.7 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。

これらの変更は次回のシステム起動後に維持されないことに注意してください。ファイアウォールに永久的な変更を行うには、**--permanent** オプションを追加してコマンドを繰り返します。**firewalld** でのポートの開閉の詳細については、『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。

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。通信を安全にするには、**-via** オプションを使用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、以下のように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
暗号化された接続のみを使用したい場合は、 **systemd.service** ファイルの ExecStart 行で **-localhost** オプションを使うと、暗号化されていない接続をすべて防ぐことができます。

```
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とポート転送された接続以外の接続を受け入れなくなります。

SSH の使用の詳細については、12章*OpenSSH* を参照してください。

13.4. 関連資料

TigerVNC に関する詳細については、以下に示された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

- **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. サーバー関係

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について説明します。

第14章 WEB サーバー

*Web サーバー*は、Web 上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。通常これは Web ページを意味しますが、他のドキュメントも提供されます。Web サーバーは、*ハイパーテキストトランスポートプロトコル* (HTTP) を使用するため、HTTP サーバーとも呼ばれます。

14.1. APACHE HTTP サーバー

Red Hat Enterprise Linux 7 で利用可能な Web サーバーは、バージョン 2.4 の **Apache HTTP Server** である **httpd** です。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。

以前の Red Hat Enterprise Linux のリリースからアップグレードする際には、**httpd** サービスの設定も同様に更新する必要があります。本項では、新しく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. 注目すべき変更点

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、

```
apachectl graceful
```

に置き換えられます。**httpd** 用の **systemd** ユニットファイルは、以下のように init スクリプトとは異なる動作をします。

- サービスがリロードされると、デフォルトで正常な再起動が使用されます。
- サービスが停止されると、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

```
apachectl configtest
```

により置き換えられます。

Private /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 **httpd** デーモンを実行してプライベートの **/tmp** ディレクトリーを使用します。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 などの追加的にロードできる **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よりも前の **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含めるために使用されます。つまり、**conf.modules.d/** 内のすべての設定ファイルは **httpd.conf** のメイン部分よりも先に処理されます。**/etc/httpd/conf.d/** ディレクトリー内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** のメイン部分の後に処理されます。

httpd パッケージ自体は、追加の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、デフォルトではセキュリティーのために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は、**http://localhost/** で表示される「ようこそ」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は、デフォルトでは明示的に設定されることはなくなりました。代わりに、デフォルト設定はハードコーディングされています。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、マニュアルで指定されています。詳細は、「インストールできるドキュメント」 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複数の変更が行われため、変更が必要になります。http://httpd.apache.org/docs/2.4/upgrading.html のアップグレードの詳細については、以下の Apache ドキュメントを参照してください。

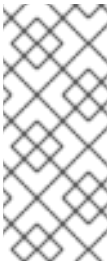
プロセスモデル

Red Hat Enterprise Linux の以前のリリースでは、異なる *マルチプロセスモデル* (MPM) が異なる **httpd** バイナリーとして利用可能となっていました。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、新サブパッケージ mod_session で提供されています。**mod_proxy_html** と **mod_xml2enc** の新モジュールは、新サブパッケージ 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 を確認してください。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 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 の記事で説明してあるステップにしたがってください。

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** 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 と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 (以前は **/var/www/icons/**) は **/usr/share/httpd/icons/** に変更されました。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/** に変更されました。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/** で利用可能です。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

認証や認可、さらには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、http://httpd.apache.org/docs/2.4/howto/auth.html の Apache ドキュメントを参照してください。

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** でインストールされなくなりました。代わりに、ファイルシステム機能ビットセットにより、限定的なパーミッションセットを許可できます。この変更と共に、**suexec** バイナリーは **/var/log/httpd/suexec.log** ログファイルを使わないようになりました。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ース

httpd モジュールインターフェースの変更のため、**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。このようなモジュールは、必要に応じて **httpd 2.4** モジュールインターフェース用に調整し、再構築する必要があります。バージョン **2.4** の API 変更の詳細な一覧は、http://httpd.apache.org/docs/2.4/developer/new_api_2_4.html で参照できます。

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** から **/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** モジュールは以下のとおりです。

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 はアップストリームでは、**httpd 2.4** で公式にサポートされていません。

mod_authz_ldap

httpd 2.4 は、**mod_authnz_ldap** を使用してサブパッケージ mod_ldap の LDAP サポートを提供します。

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、以下の手順にしたがいます。

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。名前変更されたモジュールそれぞれについて **LoadModule** ディレクティブを調節します。
2. サードパーティーのモジュールは読み込みを試行す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 **public_html**) を示す **UserDir** ディレクティブが備わっていることを確認します。
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、設定エラーの可能性をチェックすることができます。

```
~]# 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 に更新する方法の詳細については、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

このセクションでは、Apache HTTP サーバーの起動、停止、再起動、および現在のステータスのチェック方法を説明しています。**httpd** サービスを有効にするには、まず httpd がインストールされていることを確認します。以下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についての詳細は、10章*systemd によるサービス管理* を参照してください。

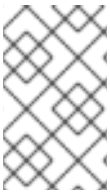
14.1.3.1. サービスの起動

httpd サービスを実効するには、シェルプロンプトで **root** として以下のコマンドを入力します。

```
~]# systemctl start httpd.service
```

起動時にサービスを自動的にスタートしたい場合には、以下のコマンドを入力します。

```
~]# 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、暗号化したプライベート SSL キーを使用すると、マシンが起動した後にパスワードを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、シェルプロンプトで **root** として以下を入力します。

```
~]# systemctl stop httpd.service
```

ブート時にサービスが自動的に起動しないようにする場合は、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** で以下のコマンドを入力します。

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

   これで実行中の **httpd** サービスが停止し、直ちに再起動します。このコマンドは、PHP のような動的に読み込まれたモジュールをインストールもしくは削除した後に使ってください。

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
```

これで実行中の **httpd** サービスが設定ファイルを再読み込みします。現在プロセス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リロードするには、**root** で以下のコマンドを入力します。

```
~]# apachectl graceful
```

これにより、実行中の **httpd** サービスは設定ファイルをリロードします。現在処理中の要求は、いずれも古い設定を使用し続けます。

Red Hat Enterprise Linux 7 でシステムサービスを管理する詳細情報については、*10章systemd によるサービス管理* を参照してください。

14.1.3.4. サービスステータスの確認

httpd サービスが実行中であることを確認するには、シェルプロンプトに以下を入力します。

```
~]# systemctl is-active httpd.service
active
```

14.1.4. 設定ファイルの編集

httpd サービスは開始後にデフォルトで、表14.1「httpd サービスの設定ファイル」 に一覧表示してある場所から設定を読み取ります。

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合しますが、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。変更はいかなるものでも、効果が反映されるには最初に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** サービスの再起動方法の詳細については 「サービスの再起動」 を参照してください。

設定でエラーの可能性をチェックするには、シェルプロンプトで以下を入力します。

```
~]# apachectl configtest
Syntax OK
```

失敗からの復旧をより簡単にするために、編集する前にオリジナルファイルのコピーを作成しておくことをお勧めします。

14.1.5. モジュールを使った作業

モジュラーアプリケーションである **httpd** サービスは、必要に応じて実行時に動的にロードまたはアンロードできる複数の *Dynamic Shared Objects (動的共有オブジェクト)* (DSO) と一緒に配布されます。Red Hat Enterprise Linux 7 では、これらのモジュールは **/usr/lib64/httpd/modules/** 内に配置されています。

14.1.5.1. モジュールのロード

特定の DSO モジュールをロードするには、**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 のロード

```
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** サービスの再起動の方法については 「サービスの再起動」 を参照してください。

14.1.5.2. モジュールの書き込み

新規の 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。確認するには、**root** で以下のコマンドを入力します。

```
~]# 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ています。

書き込みが終了したら、以下のコマンドでモジュールをビルドできます。

```
~]# apxs -i -a -c module_name.c
```

ビルドが成功すると、Apache HTTP サーバーで配布されている他のモジュールと同様にこのモジュールをロードできるはずです。

14.1.6. 仮想ホストのセットアップ

Apache HTTP サーバーに組み込まれている仮想ホストにより、サーバーは要求される IP アドレス、ホスト名、またはポートに基づいて異なる情報を提供できます。

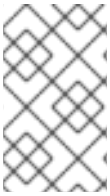
名前ベースの仮想ホストを作成するには、まず設定ファイルの例である **/usr/share/doc/httpd-*VERSION*/httpd-vhosts.conf** を **/etc/httpd/conf.d/** ディレクトリーにコピーし、**@@Port@@** と **@@ServerRoot@@** のプレースホルダーの値を置き換えます。例 14.2「仮想ホスト設定のサンプル」 にあるように、実際の要件に合わせてオプションをカスタマイズします。

例14.2 仮想ホスト設定のサンプル

```
<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ている有効な DNS 名である必要があることに注意してください。**<VirtualHost>** コンテナーは高度にカスタマイズ可能で、メインサーバー設定内で利用できるほとんどのディレクティブを受け付けます。このコンテナー内でサポート**されていない**ディレクティブには **User** と **Group** があり、これらは **SuexecUserGroup** に置き換えられています。



注記

仮想ホストがデフォルト以外のポートでリッスンするように設定する場合は、**/etc/httpd/conf/httpd.conf** ファイルのグローバル設定で **Listen** ディレクティブを適切に更新することを忘れないでください。

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については、「サービスの再起動」 を参照してください。

14.1.7. SSL サーバーのセットアップ

Secure Sockets Layer (セキュアソケットレイヤー) (SSL) は暗号化プロトコルであり、サーバーとクライアント間の安全な通信を可能にします。その拡張および改善バージョンである *Transport Layer Security (トランスポートレイヤーセキュリティ)* (TLS) と共に、SSL はプライバシーとデータの整合性を保証します。Apache HTTP Server は、SSL/TLS サポートを提供するために OpenSSL ツールキットを使用するモジュールである **mod_ssl** と共に、一般的に *SSL サーバー*と呼ばれます。また、Red Hat Enterprise Linux は、TLS 実装としての Mozilla NSS の使用をサポートします。Mozilla NSS のサポートは **mod_nss** モジュールにより提供されます。

傍受できる人は誰でも読み取りと修正が可能な HTTP 接続とは異なり、HTTP を介した SSL/TLS (HTTPS と呼ばれます) を使用すると、送信されたコンテンツの検査や修正を阻止できます。本項では、Apache HTTP Server 設定内でこのモジュールを有効にする方法についての基本情報を提供し、秘密鍵と自己署名の証明書の生成プロセスについて説明します。

14.1.7.1. 証明書とセキュリティーの概要

安全な通信は鍵の使用に基づいています。従来の *symmetric cryptography (対称型暗号化)* では、トランザクションの両方でお互いの通信の解読に使用可能な同一の鍵を備えています。それに対し、パブリックの *asymmetric cryptography (非対称型暗号化)* では、2 つの鍵が共存します。*秘密鍵* は秘密として守られて、*公開鍵* は通常、公共と共有されます。公開鍵でエンコードされたデータは秘密鍵でのみデコードされるのに対し、秘密鍵でエンコードされたデータは逆に公開鍵でのみデコードできます。

SSL を使用して安全な通信を提供するためには、SSL サーバーは *Certificate Authority* (CA) (認証局) で署名された電子証明書を使用する必要があります。証明書はサーバーの様々な属性 (サーバーのホスト名、企業の名前、その住所など) と CA の秘密鍵で生成した署名を一覧表示します。この証明書は特定の認証局が証明書に署名したこと、また証明書がいかなる方法でも修正されていないことを保証するものです。

Web ブラウザーは新規の SSL 接続を確立する際に、Web サーバーから提供される証明書をチェックします。証明書に信頼できる CA からの署名がない場合や、証明書に表記してあるホスト名が接続確立で使用されたホスト名と一致しない場合は、Web ブラウザーはそのサーバーとの交信を拒否して、通常

はユーザーに適切なエラーメッセージを表示します。

デフォルトでは、ほとんどのウェブブラウザーが広く使用されている認証局のグループを信用するように設定されています。そのため、セキュアサーバーをセットアップする時には適切な CA (認証局) を選択すべきです。そうすることで、対象ユーザーは接続を信頼できますが、そうでない場合はエラーメッセージが表示されて、手動で証明書を受理する必要があります。証明書のエラーメッセージを無視するようにユーザーに推奨することは攻撃者の侵略を許すことにつながるので、できる限り信頼できる CA を使用してください。この件に関する詳細情報は、表14.2「一般的な Web ブラウザーで使用される CA 一覧に関する情報」 を参照してください。

表14.2 一般的な Web ブラウザーで使用される CA 一覧に関する情報

ウェブブラウザー	リンク
Mozilla Firefox	Mozilla root CA 一覧
Opera	Opera により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 プロジェクトにより使用され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。認証局は証明書要求と識別を確認すると、サーバーで使用できる署名付きの証明書を送ってきます。その他の方法として、CA 署名のない自己署名の証明書を作成することができますが、これはテスト目的のみに使用すべきものです。

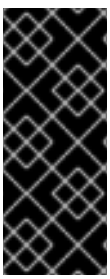
14.1.8. mod_ssl モジュールの有効化

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**。HTTPS のデフォルトポートはポート **443** です。

mod_ssl モジュールと OpenSSL ツールキットを使用して SSL サーバーをセットアップするには、mod_ssl および openssl パッケージをインストールします。**root** で以下のコマンドを入力します。

```
~]# yum install mod_ssl openssl
```

これにより、中心となる Apache HTTP サーバー設定ファイルにデフォルトで含まれている **/etc/httpd/conf.d/ssl.conf** に **mod_ssl** 設定ファイルが作成されます。このモジュールを読み込むには、「サービスの再起動」 にあるように **httpd** サービスを再起動します。



重要

『POODLE: SSLv3 脆弱性 (CVE-2014-3566)』 で示された脆弱性のため、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**でのみ**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。

手順14.1 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 以外のすべてを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、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. Apache デーモンを以下のように再起動します。

```
~]# systemctl restart httpd
```

   すべてのセッションが中断されることに注意してください。

手順14.2 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 以上を除くすべての SSL および TLS プロトコルバージョンを無効にするには、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、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
```

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. Apache デーモンを以下のように再起動します。

```
~]# systemctl restart httpd
```

   すべてのセッションが中断されることに注意してください。

手順14.3 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。コマンドの形式は以下のようになります。

```
openssl s_client -connect hostname:port -protocol
```

ここで、*port* はテストするポートであり、*protocol* はテストするプロトコルバージョンです。ローカルで実行されている SSL サーバーをテストするには、ホスト名として **localhost** を使用します。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。

1.
```
~]# 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:出力省略
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : SSLv3出力省略
```

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。

2.
```
~]$ 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出力省略
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出力省略
```

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 コマンドオプションは、**s_client(1)** man ページで文書化されています。

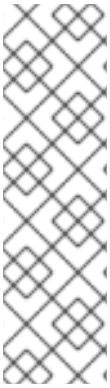
SSLv3 の脆弱性とそのテスト方法の詳細については、Red Hat ナレッジベースの記事 『POODLE: SSLv3 脆弱性 (CVE-2014-3566)』 を参照してください。

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 つのポートを所有できるモジュールは 1 つだけです。したがって、**mod_nss** と **mod_ssl** は、異なるポートを使用する場合のみ共存できます。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、各トークンには保護するためにパスワードを設定できます。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。

手順14.4 mod_nss の設定

手順14.4 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。モジュールをロードするには、「サービスの再起動」 で説明されているように **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、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内のすべての証明書を一覧表示するか、指定された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については、**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、**-n** オプションは証明書の名前を設定します。**-t CT,,** は、証明書が TLS クライアントおよびサーバーでの使用に信頼できることを意味します。**-A** オプションは既存の証明書を証明書データベースに追加します。データベースは、存在しない場合は作成されます。**-a** オプションは出入力での ASCII 形式の使用を可能にします。**-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。

他のコマンドラインオプションについては、**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. 以下のように行を **NSSPassPhraseDialog** ディレクティブで変更して、NSS 内部ソフトウェアトークンを使用するよう **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** パラメーターをコメント解除し、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** パラメーターがコメント解除され、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
```

以下の形式の行を追加します。

```
internal:password
```

password を、上記の手順 6 で NSS セキュリティーデータベースに適用されたパスワードに置き換えます。

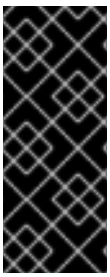
12. 適切な所有権とパーミッションを **/etc/httpd/password.conf** ファイルに適用します。

```
~]# 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」 で説明されているように Apache サーバーを再起動します。



重要

『POODLE: SSLv3 脆弱性 (CVE-2014-3566)』 で示された脆弱性のため、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**でのみ**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。

手順14.5 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 以上を除くすべての SSL および TLS プロトコルバージョンを無効にするには、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:出力省略
```

```
#   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、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され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. Apache デーモンを以下のように再起動します。

```
~]# service httpd restart
```

すべてのセッションが中断されることに注意してください。

手順14.6 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** で openssl パッケージをインストールします。

```
~]# yum install openssl
```

openssl s_client -connect コマンドの形式は以下のようになります。

```
openssl s_client -connect hostname:port -protocol
```

ここで、*port* はテストするポートであり、*protocol* はテストするプロトコルバージョンです。ローカルで実行されている SSL サーバーをテストするには、ホスト名として **localhost** を使用します。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。

1.
```
~]# openssl s_client -connect localhost:443 -ssl3
CONNECTED(00000003)
3077773036:error:1408F10B:SSL routines:SSL3_GET_RECORD:wrong version
number:s3_pkt.c:337:出力省略
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : SSLv3出力省略
```

   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。

2.
```
~]$ openssl s_client -connect localhost:443 -tls1
CONNECTED(00000003)
depth=1 C = US, O = example.com, CN = Certificate Shack出力省略
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出力省略
```

   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 コマンドオプションは、**s_client(1)** man ページで文書化されています。

SSLv3 の脆弱性とそのテスト方法の詳細については、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. 既存の鍵と証明書の使用

以前に鍵と証明書が作成されている場合は、新規で作成するのではなく、既存のファイルを使用するように SSL サーバーを設定することができます。ただし、これが実行できないケースが 2 つだけあります。

1. **IP アドレスまたはドメイン名を変更しているケース。**

   証明書は特定の IP アドレスとドメイン名のペアに発行されるものです。これらの値のどちらかが変更されると証明書は無効になります。

2. **VeriSign からの証明書があって、サーバーソフトウェアを変更しているケース。**

   広く使用されている認証局である VeriSign は、特定のソフトウェア製品、IP アドレス、およびドメイン名に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。

上記のいずれのケースでも、新しい証明書を取得する必要があります。このトピックに関する情報については、「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
```

そして、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
```

更新済みの設定を読み込むには、「サービスの再起動」 にあるように **httpd** サービスを再起動します。

例14.5 Red Hat セキュアウェブサーバー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。

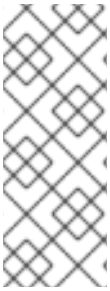
```
~]# yum install crypto-utils
```

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** を含みます。

重要

有効な証明書を既に所有していて、それを新規のものと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既に存在する場合は、**genkey** は開始を拒否します。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**) 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、以下の手順にしたがいます。

1. 鍵と証明書を保存する場所を確認します。

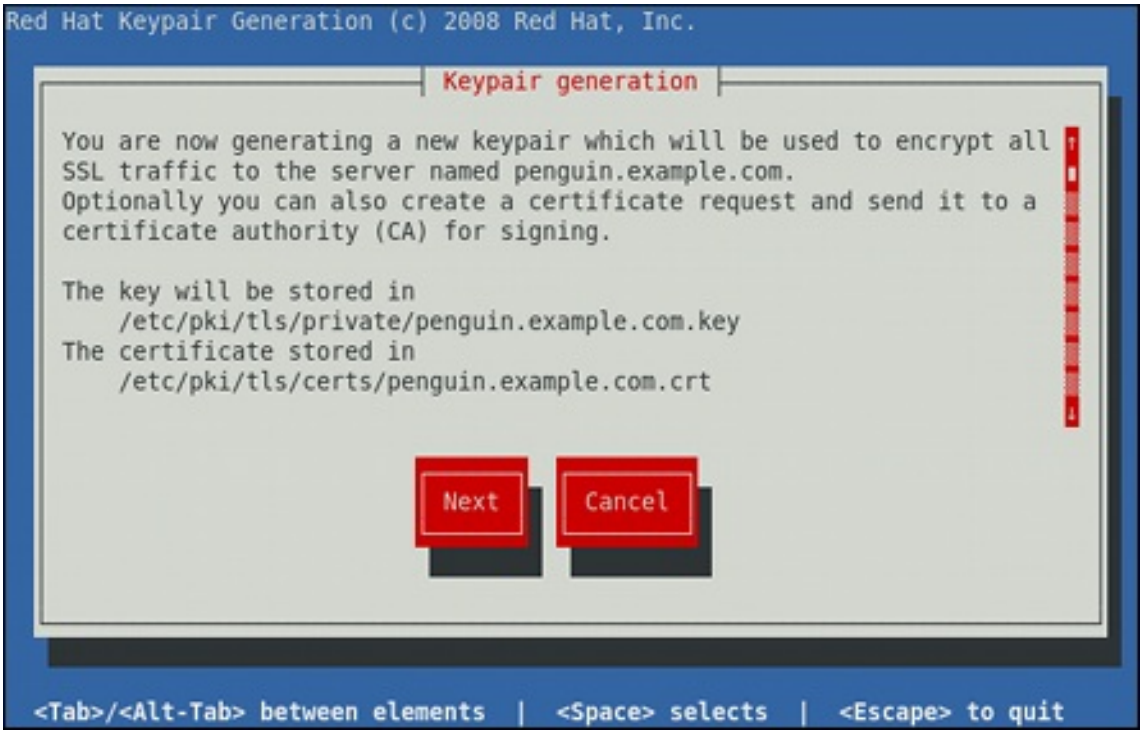


図14.1 genkey ユーティリティーの実行

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。大きな鍵はセキュリティーを向上させますが、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** の使用を推奨しています。『NIST Special Publication 800-131A』 を参照してください。

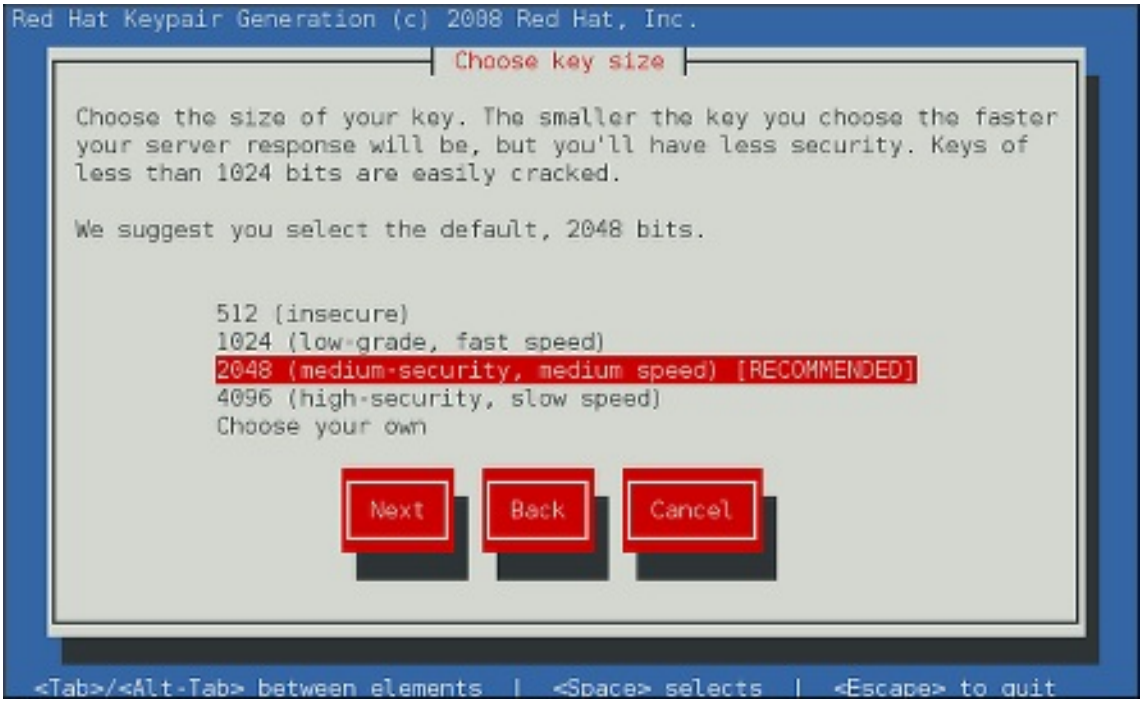


図14.2 鍵のサイズ選択

終了したら、**Tab** キーを使用して **Next (次へ)** ボタンを選択して **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、これに時間がかかりることもあります。

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

Tab 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、または **No (いいえ)** を選択して、自己署名の証明書を生成します。その後に、**Enter** を押して選択を確定します。

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。

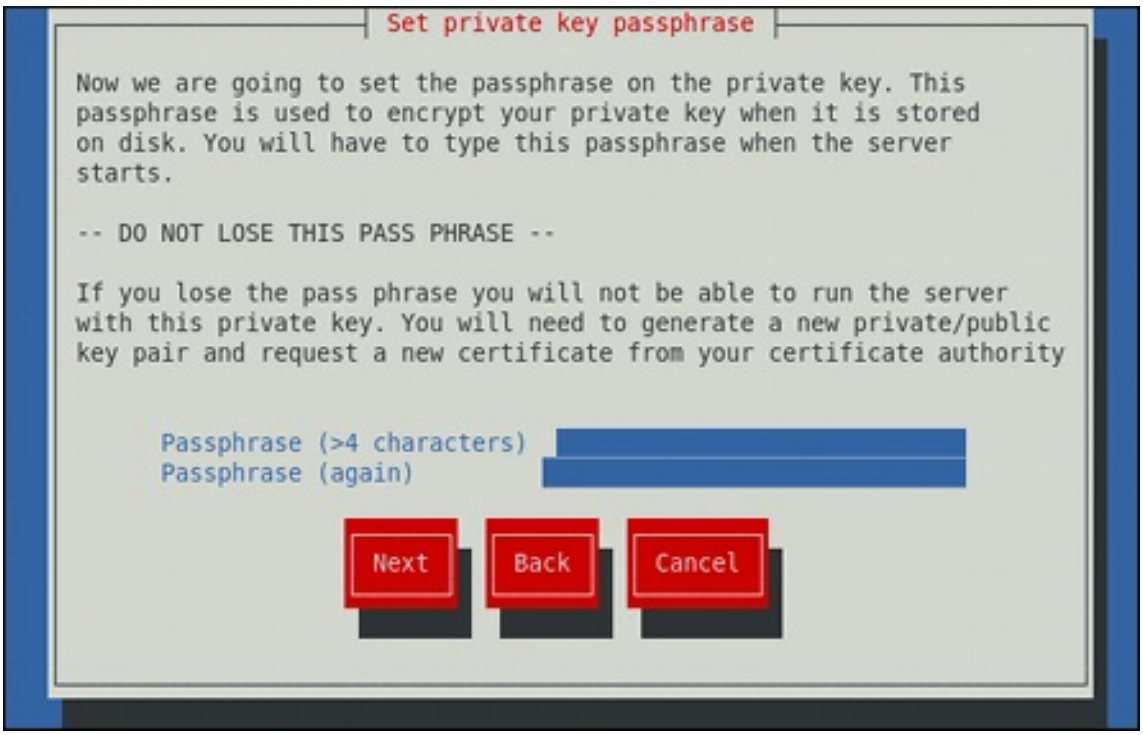


図14.5 パスフレーズの入力

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

サーバーを開始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しなければなりません。

6. 証明書詳細のカスタマイズ

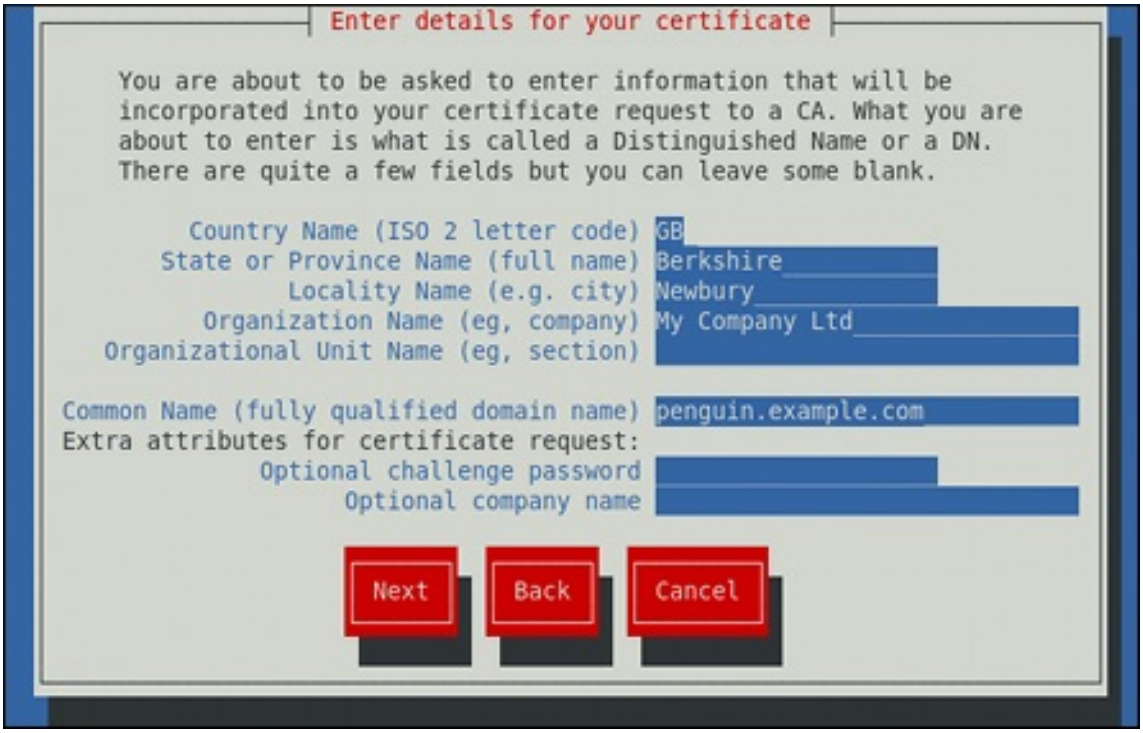


図14.6 証明書情報の指定

Tab キーを使用して **Next** ボタンを選択します。それから **Enter** を押すと鍵の生成が完了します。

7. 以前に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プロンプトされます。

図14.7 証明書要求を送信する方法の指示

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」 にあるように **httpd** サービスを再起動します。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た HTTP および HTTPS 用ファイアウォールの設定

Red Hat Enterprise Linux では、デフォルトで **HTTP** および **HTTPS** トラフィックが許可されません。システムが Web サーバーとして機能できるようにするには、**firewalld** のサポートされたサービスを使用して **HTTP** および **HTTPS** トラフィックが必要に応じてファイアウォールを通過できるようにします。

コマンドラインを使用して **HTTP** を有効にするには、**root** で以下のコマンドを発行します。

```
~]# firewall-cmd --add-service http
 success
```

コマンドラインを使用して **HTTPS** を有効にするには、**root** で以下のコマンドを発行します。

```
~]# firewall-cmd --add-service https
 success
```

これらの変更は次回のシステム起動後に維持されないことに注意してください。ファイアウォールに永久的な変更を行うには、**--permanent** オプションを追加してコマンドを繰り返します。

14.1.12.1. コマンドラインを使用した受信 HTTP および HTTPS 向けネットワークアクセスの設定

コマンドラインを使用してファイアウォールが許可するよう設定されてい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出力省略
```

デフォルトインストールのこの例では、ファイアウォールは有効ですが、**HTTP** および **HTTPS** は通過を許可されていません。

HTTP および **HTTP** ファイアウォールサービスが有効になると、以下のような **services** 行が現れます。

```
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。

14.1.13. 関連資料

Apache HTTP サーバーに関する詳細については、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

- **httpd(8)** — コマンドラインオプションの全一覧が含まれている **httpd** サービスの man ページです。
- **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 サーバー制御インターフェースの man ページです。

インストールできるドキュメント

- http://localhost/manual/ — ディレクティブと利用可能なモジュールの全説明を含む Apache HTTP サーバーの公式ドキュメントです。このドキュメントにアクセスするには、httpd-manual パッケージがインストールしてあり、Web サーバーが稼働している必要があります。

  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します。

  ```
  ~]# yum install httpd-manual
  ~]# apachectl graceful
  ```

オンラインのドキュメント

- http://httpd.apache.org/ — すべてのディレクティブとデフォルトのモジュールに関するドキュメントが収納された、Apache HTTP サーバーの公式 Web サイトです。
- http://www.openssl.org/: さらに詳しいドキュメントや FAQ、メーリングリストへのリンク、その他役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、電子メールを使用、アクセスするための多くの高度なアプリケーションを提供します。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. 電子メールプロトコル

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が、これは電子メールクライアントにとっても極めて重要となります。電子メールを送信するには、クライアントはメッセージを送信メールサーバーに送信します。送信メールサーバーはこれを受けて、転送先メールサーバーに配信を連絡します。このため、電子メールクライアントを設定する時には SMTP サーバーを指定する必要があります。

Red Hat Enterprise Linux では、ユーザーはメール配信を処理するようローカルマシン上の SMTP サーバーを設定することができます。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することができます。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ています。

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 パッケージにより提供されています。

注記

Dovecot を使用するには、まず使用中のシステムに dovecot パッケージがインストールされていることを確認するために、**root** で以下を実行します。

```
~]# yum install dovecot
```

yum を使ったパッケージのインストールについては 「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には不利になりますが、**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は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。

セキュリティーを強化するには、*SSL (Secure Socket Layer セキュアソケットレイヤ)* 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、**ipop3s** サービスまたは **stunnel** アプリケーションを使用して有効にすることができます。電子メール通信をセキュアにする方法の詳細については 「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 でのデフォルト **IMAP** サーバーは **Dovecot** で、dovecot パッケージが提供しています。**Dovecot** のインストール方法については 「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、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読み取り、削除を行うことができます。**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を作成、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っ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することができます。電子メール通信をセキュアにする方法の詳細については 「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** 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 **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します。デフォルトでは **dovecot** は、**SSL** を使用して **IMAP** と **POP3** をセキュアなバージョンとともに実行します。**POP** を使用するよう **dovecot** を設定するには、次のステップを実行します。

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。例えば以下のとおりです。

   ```
   protocols = imap pop3 lmtp
   ```

   protocols 変数がコメントアウトされている場合は、**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 とは違い、**IMAP** と **POP3** はユーザー名とパスワードを使用して接続するクライアントを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。詳細については、『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/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** 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役割を果たします。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く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、または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って制限されます。

最新の電子メールクライアントプログラムの多くは、電子メールを送信する際に MTA として機能することができます。ただし、このアクションが真の MTA の役割であると混同してはいけません。電子メールクライアントプログラムで MTA のような電子メール送信が可能である唯一の理由は、アプリケーションを稼働しているホストに独自の MTA がないためです。これは非 UNIX ベースのオペレー

ティングシステム上の電子メールクライアントプログラムに特に当てはまります。ただし、これらのクライアントプログラムは、使用を許可されている MTA に対して送信メッセージを送信するだけで、目的の受信者の電子メールサーバーには直接メッセージを配信することはありません。

Red Hat Enterprise Linux は *Postfix* と *Sendmail* という 2 つの MTA を提供しているので、電子メールクライアントプログラムは大抵、MTA として機能する必要がありません。Red Hat Enterprise Linux には、*Fetchmail* と呼ばれる特別用途の MTA も装備されています。

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 の役割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ー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 (メールユーザーエージェント) 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 プログラムにより、ユーザーは最低でも電子メールメッセージの読み取りと作成ができます。多くの MUA では、**POP** または **IMAP** プロトコルを介したメッセージ取得や、メッセージを保管するメールボックスの設定、MTA へのメッセージ送信ができます。

MUA は、**Evolution** のようなグラフィカルインターフェースの場合と、**Mutt** のようなシンプルなテキストベースのインターフェースの場合があります。

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って Sendmail に切り替えます。

```
~]# alternatives --config mta
```

以下のコマンドを使って、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詳細情報については、10章*systemd によるサービス管理* を参照してください。

15.3.1. Postfix

当初、IBM のセキュリティエキスパート/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 の設定ファイルはヒューマンリーダブルで、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。

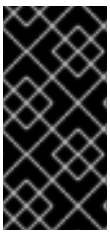
15.3.1.1. Postfix のデフォルトインストール

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一覧です。

- **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** — Postfix のグローバル設定ファイルです。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一覧です。



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。これを実行するには、**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 の以下の設定は、以前のリリースと異なります。

- **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** に変更された場合は、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** パラメータの詳細などの補足情報は http://www.postfix.org/ でオンラインで参照できます。

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。詳細については、『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

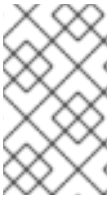
以下は **/etc/aliases** ファイルをルックアップする **LDAP** を使用する基本的な例です。**/etc/postfix/main.cf** ファイルに以下の内容が含まれていることを確認してください。

```
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、これを作成します。以下の内容を含めます。

```
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することができます。詳細は **ldap_table(5)** の 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 の使用が推奨されていることに注意してください。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。複数の役割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に思われるかもしれません。技術的にはこれは真実です。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。しかし、実際は大半のユーザーは単なるメール配信以上の機能を必要とします。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。あるいは、メールボックスへのアクセスに Web インターフェースを好むユーザーもいます。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 に関するリソースの一覧は「関連資料」 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** で以下を実行し、使用中のシステムに 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 を使ったパッケージのインストールについては「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 の切り替え方法については「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 の長い詳細設定ファイルは **/etc/mail/sendmail.cf** です。**sendmail.cf** ファイルは直接編集しないようにしてください。Sendmail の設定を変更するには、**/etc/mail/sendmail.mc** ファイルを編集して、元の **/etc/mail/sendmail.cf** ファイルをバックアップした上で、以下のような別の方法で新規設定ファイルを生成します。

- **/etc/mail** にある makefile を使用して新規の **/etc/mail/sendmail.cf** 設定ファイルを作成します。

  ```
  ~]# make all -C /etc/mail/
  ```

  /etc/mail にある生成された他のすべてのファイル (db ファイル) は、必要に応じて再生成されます。旧 makemap コマンドは現在も使用可能です。make コマンドは、**sendmail** サービスを起動もしくは再起動するたびに、自動的に使用されます。

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シング (aliasing) 形式を指定し、単一のマシン上における複数の仮想ドメインのホスティングを可能にします。

Sendmail が設定変更を使用可能となる前に、**access**、**domaintable**、**mailertable**、**virtusertable** など **/etc/mail/** ディレクトリーにある設定ファイルのいくつかが、実際にそれらの情報をデータベースファイルに保存する必要があります。これらの設定への変更をデータベースファイル内に含めるには、**root** で以下のコマンドを実行します。

```
~]# cd /etc/mail/
~]# make all
```

これで、**virtusertable.db**、**access.db**、**domaintable.db**、**mailertable.db**、**sendmail.cf**、および **submit.cf** が更新されます。

上記のデータベースファイルすべておよびカスタムのデータベースファイルを更新するには、以下のフォーマットでコマンドを使用します。

```
make name.db all
```

ここでの *name* は、更新するカスタムデータベースファイル名になります。

単一のデータベースを更新するには、以下のフォーマットでコマンドを使用します。

```
make name.db
```

ここでの *name.db* は、更新するデータベースファイル名になります。

変更を反映させるために、以下を実行して **sendmail** サービスを再起動することもできます。

```
~]# systemctl restart sendmail
```

たとえば、**example.com** ドメイン宛のすべての電子メールが **bob@other-example.com** に配信されるようにするには、**virtusertable** ファイルに以下の行を追加します。

```
@example.com bob@other-example.com
```

変更を完了するには、**virtusertable.db** ファイルを更新する必要があります:

```
~]# make virtusertable.db all
```

all オプションを使用すると、**virtusertable.db** と **access.db** が同時に更新されます。

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、全く新しい **/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 に追加したい場合は、**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 のデフォルトの設定ファイルは、大半の **SMTP** 専用サイトで機能します。ただし、*UUCP* (*UNIX-to-UNIX Copy Protocol*) サイトでは機能しません。UUCP メール転送を使用する場合は、**/etc/mail/sendmail.mc** ファイルを再設定して、新しい **/etc/mail/sendmail.cf** ファイルを生成する必要があります。

/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は、**/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. マスカレード

一般的な Sendmail の設定の 1 つとして、単一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_AS(`example.com.')dnl
MASQUERADE_DOMAIN(`example.com.')dnl
MASQUERADE_AS(example.com)dnl
```

m4 マクロプロセッサーを使用して新しい **sendmail.cf** ファイルを生成すると、この設定によりネットワーク内のメールはすべて **example.com** から送信されたかのように表示されます。

メールサーバー、**DNS** および **DHCP** サーバー、またプロビジョニングアプリケーションの管理者は、組織内で使用するホスト名の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については、 Red Hat Enterprise Linux 7 Networking Guide を参照してください。

15.3.2.5. スパムの防止

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定義することができます。これは、破壊的で高コストを伴う、広く蔓延したインターネット通信標準の悪用です。

Sendmail を使うと、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
```

しかし、ユーザーはインターネット上のサーバーからスパムを送信されることもあります。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 ファイルはデータベースであるため、**makemap** コマンドを使用して変更を更新します。これを行うには、**root** で以下のコマンドを使用します。

```
~]# makemap hash /etc/mail/access < /etc/mail/access
```

メッセージヘッダーの分析により、ヘッダーのコンテンツに基づいたメール拒否が可能となります。**SMTP** サーバーは、電子メールの送信経路に関する情報をメッセージヘッダー内に保管します。

メッセージがある MTA から別の MTA に送られると、その他すべての **Received** ヘッダーの上にそれぞれ **Received** ヘッダーを付けます。ただし、注意すべき重要なポイントは、この情報はスパム送信者が変更できるという点です。

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一保存することも可能です。SpamAssassin の使用に関する詳細は、「スパムフィルター」 を参照してください。

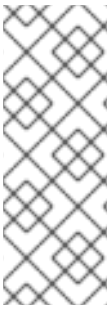
15.3.2.6. LDAP での Sendmail の使用

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。例えば、**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姓で検索することができます。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て、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** との非常に大規模な統合をサポートします。この統合では、Sendmail は **LDAP** を使用して、中規模からエンタープライズレベルの組織をサポートするために連携する異なるメールサーバー上の **/etc/aliases** や **/etc/mail/virtusertables** などの個別に管理されているファイルを置き換えます。一言でいうならば、**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多くの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** を使った非常に基本的な Sendmail の設定にすぎません。実際の設定は **LDAP** の実装により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。この方法については、「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** などの数々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、**root** で以下を実行し、まず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 を使ったパッケージのインストールについては 「パッケージのインストール」 を参照してください。

Fetchmail は、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各ユーザー用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上で渡し、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** ファイル内に配置し、それらのオプションが、**fetchmail** コマンドを発行する時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、コマンドライン上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 に与えます。
- **サーバーオプション** — ポーリングされるサーバーに関する必要情報を指定します。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の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** ファイルの最上部に表示されます。その次には 1 つまたは複数のサーバーオプションがあり、それぞれが Fetchmail が確認すべき異なる電子メールサーバーを指定します。サーバーオプションの次には、その電子メールサーバーを確認する各ユーザーアカウントのユーザーオプションがあります。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。

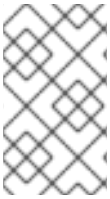
サンプルの **.fetchmailrc** ファイルは以下のとおりです:

```
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。次に、2 つの電子メールサーバーが指定されます。1 つは **POP3** 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。第 2 のサーバーオプションを使用して 2 人のユーザーが確認されますが、どのユーザー宛でも見つかった電子メールはすべて、**user1** のメールスプールに送信されます。これにより、単一の MUA 受信ボックスに表示させながら、複数のサーバー上で複数のメールボックスをチェックすることが可能となります。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した場合は、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、数々の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、非常に独特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** 内で、**poll** または **skip** アクションの後にそれぞれの行に配置される必要があります。

- **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 **kerberos_v5**、**kerberos_v4** および **ssh** を含む他のタイプの認証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 は最初にパスワードを必要としない方法を試みます。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*** の時間間隔でポーリングし、設定された全サーバー上の電子メールを確認します。このオプションは、通常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、サーバー上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 が接続の試行を打ち切った後でサーバーが非アクティブとなる秒数に置き換えます。この値が設定されていない場合は、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配置される場合と、サーバーオプションと同じ行に配置さ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) にしたがう必要があります。

- **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、停止する前に取得され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。このオプションは、大型のメッセージのダウンロードに時間がかかりすぎる場合の低速のネットワークリンクに有用です。

- **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 暗号化をアクティブ化します。執筆時点では、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** と **SSLv3** を**使用しない**よう設定できない **stunnel** を使用します。

- **sslproto** — 許可された SSL 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。デフォルトの値 (**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) 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** と **SSLv3** を**使用しない**よう設定できない **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することができます。 **.fetchmailrc** 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、大半のユーザー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一覧表示、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. 特別なオプション

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** オプションについては、**fetchmail** の man ページを参照してください。

15.3.4. MTA の設定

MTA は電子メールの送信に不可欠です。**Evolution** や **Mutt** などの *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、Sendmail と Postfix の 2 つの MTA を提供しています。どちらもインストールされている場合は、Postfix がデフォルトの MTA となります。Sendmail は Red Hat Enterprise Linux 7 では非推奨となることに注意してください。

15.4. メール配信エージェント(MDA)

Red Hat Enterprise Linux には Procmail と **mail** の 2 つの主要 MDA が装備されています。どちらのアプリケーションも LDA とみなされ、電子メールを 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 (**man mail**) を参照してください。

電子メールがローカルホストのメールスプールファイルに配置されると、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役割を果たします。

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配置すると、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** ファイルは存在せず、ユーザーのホームディレクトリーに **.procmailrc** ファイルは存在しません。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は、電子メールは特定のファイルに配置、削除、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。

大半の Procmail ユーザーが使用しない環境変数が多々あります。重要な環境変数の多くは、既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 レシピの一覧は、スパムのブロック、電子メール一覧の管理など異なる役割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。

  例えば、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
  ```

  電子メールの一覧の 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見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** の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。

環境変数すべてについての包括的な説明やデフォルト値については、**procmailrc** の man ページを参照してください。

15.4.2. Procmail のレシピ

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、レシピが適合するストリング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、例を使っ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 レシピは以下の書式を使用します:

```
: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配置して、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。

レシピは、メッセージと適合させる様々な条件を格納できます。条件がない場合は、あらゆる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、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配置すると、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アクションを実行するよう指定します。1 つのレシピにつき 1 つのアクションのみとなります。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. 配信レシピと非配信レシピの比較

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* レシピと見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、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見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** と全く同じです。

その他のフラグの詳細な一覧は、**procmailrc** の 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 (**:**) を配置します。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たバイト数内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た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 Procmail にメッセージを指定された電子メールアドレスに転送するように指示します。
- **$** — **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、様々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。
- **|** — メッセージを処理するための特定のプログラムを起動します。
- **{** 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することに加えて、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。正規表現についての詳細な説明は、本セクションの範囲外となります。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、調整方法は、これらのレシピ例を参照してください。また、基礎的な正規表現のルールに関する初歩的な情報は、**grep(1)** の man ページ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
```

1 行目では、ローカルロックファイルが作成されたことは指定していますが、名前は指定していません。そのため、Procmail は送信先のファイル名を使用して **LOCKEXT** 環境変数で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に適合し、**MAILDIR** 環境変数によって指定されたディレクトリー内にある **new-mail.spool** と呼ばれる単一のスプールファイルに配置されます。その後、MUA はこのファイル内のメッセージを閲覧できるようになります。

このような基本レシピは、**rc** ファイルの末尾に配置され、メッセージをデフォルトの場所に振り向けます。

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。これで、メールボックスを時折確認して、誤検知を探すことができます。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** メールボックスに配置されます。**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、「関連資料」 の Procmail に関する数々のオンライン資料を参照してください。

15.4.2.6. スパムフィルター

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することができます。

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 を使ったパッケージのインストールについては 「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配置することです。

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-default.rc
```

/etc/mail/spamassassin/spamassassin-default.rc には、シンプルな Procmail ルールが格納されており、受信するすべての電子メールに対して SpamAssassin をアクティブ化します。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て、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 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。

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配置します。システム全体の設定の場合は、**/etc/procmailrc** に配置してください。

```
INCLUDERC=/etc/mail/spamassassin/spamassassin-spamc.rc
```

15.5. メールユーザーエージェント (MAIL USER AGENTS)

Red Hat Enterprise Linux には、多数の利用可能な電子メールプログラムがあります。**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。

本セクションではこれ以降、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

Evolution や **Mutt** などの Red Hat Enterprise Linux に装備されている評判の高い 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 をクライアント側で有効にするのは簡単です。多くの場合は、MUA の設定ウインドウでボタンをクリックするか、MUA の設定ファイルのオプションで有効にすることができます。セキュアな **IMAP** と **POP** には既知のポート番号 (それぞれ **993** と **995**) があり、MUA はそれらを使用してメッセージの認証、ダウンロードを行います。

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** 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で可能です。

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お勧めします。実稼働環境で使用するサーバーは、CA により交付された 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を希望に応じて編集し、**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** または **POP** サービスへの標準的なセキュアでない接続の周りに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的な設定の作成方法については、Red Hat Enterprise Linux 7 Security Guide (セキュリティーガイド) のUsing stunnel (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
```

Security Guide (セキュリティーガイド) では、**stunnel** の起動および停止方法について説明しています。起動後は、**IMAP** または **POP** 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。

これらのリスクを避けるには、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。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。このため Sendmail は、今後のメジャーリリースではサポート対象外となる可能性があり、Red Hat は新規のデプロイメントでの使用を推奨していません。Fetchmail を MTA として使用することもできます。

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。

複数のスパム対策機能を持つ Sendmail を使用して、MTA サイドでスパムを止めることが可能です。Sendmail には、**header checks**、**relaying denial**、**access database**、および **sender information checks** などのスパム対策機能があります。詳細は、「スパムの防止」 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) に対応するので、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を設定

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には、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、「スパムフィルター」 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 」を参照してください。SpamAssassin に関する追加情報は、SpamAssassin project website 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については、「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 がこれらに提供するサポートレベルに関する詳細は、Red Hat ナレッジベースの記事「Red Hat グローバルサポートサービスは、サードパーティのソフトウェア、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。

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。これは、オープンソースのウイルス対策エンジンで、トロイの木馬、ウイルス、マルウェアなどの悪質なソフトウェアを検出します。ClamAV に関する追加情報は、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。

clamav、clamav-data、clamav-server、および clamav-update 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 リポジトリーに関する詳細は、「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 がこれらに提供するサポートレベルに関する詳細は、Red Hat ナレッジベースの記事「Red Hat グローバルサポートサービスは、サードパーティのソフトウェア、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。

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、the latest version of the epel-release package for Red Hat Enterprise Linux 7 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 Package Signing Keys を参照してください。

15.7. 関連資料

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一覧です。

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 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 **/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 の機能の全一覧を記載した **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について概説しています。
  - **procmailrc** — レシピの構築に使用される **rc** のファイル形式について説明しています。
  - **procmailex** — 実環境向けの役立つ Procmail レシピの例を多数記載しています。

- **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について説明しています。
- **/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. オンラインのドキュメント

- How to configure postfix with TLS? — postfix が TLS を使用するよう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 ナレッジベースの解答例です。
- 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 に関連したニュース、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 — Fetchmail に関する詳細な FAQ です。
- http://www.procmail.org/ — Procmail のホームページです。Procmail に特化した各種メーリングリストへのリンクや、様々な FAQ ドキュメントが記載されています。
- http://www.spamassassin.org/ — SpamAssassin プロジェクトの公式サイトです。

15.7.3. 関連資料

- 『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) — 主要な電メールプロトコルとそれらによ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、SMB (*Server Message Block* サーバーメッセージブロック) および **CIFS** (*common Internet file system*) プロトコルのオープンソース実装である **Samba**、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** のインストールと設定方法を紹介しています。また、プリンターを設定する **印刷設定** ツールの使用方法についても説明しています。

16.1. SAMBA

Samba は、Red Hat Enterprise Linux に SMB (Server Message Block サーバーメッセージブロック) プロトコルを実装します。SMB プロトコルは、ファイルの共有および共有されたプリンターなど、サーバー上のリソースへのアクセスに使われます。さらに Samba は、Microsoft Windows で使用される DCE RPC (Distributed Computing Environment Remote Procedure Call 分散コンピューティング環境リモートプロシージャコール) プロトコルを実装します。

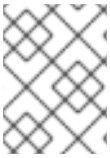
Samba は以下の環境で実行できます。

- 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 の Primary Domain Controller (PDC) または Backup Domain Controller (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、オプションでディレクトリーおよびプリンターを共有できます。これにより、Samba はファイルおよびプリントサーバーとして機能できます。



注記

Red Hat は、Samba を AD ドメイン制御 (DC) として実行することを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と印刷サービスを提供します。さらに、リソースのロックとユーザーの接続認証も、このサービスで提供しています。**smb systemd** サービスは、**smbd** デーモンを開始および停止します。

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、IP プロトコル上の NetBIOS 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** サービスは、ドメイン、ワークグループ、ホスト、ファイル共有、およびプリンターを探すために SMB ネットワークのブラウジングを有効にします。そのためにサービスは、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、またはローカルあるいはマスターブラウザーにこの情報を転送するかのいずれかを実行します。**nmb systemd** サービスは、**nmbd** デーモンを開始および停止します。

現在の SMB ネットワークは DNS を使用して、クライアントおよび IP のアドレスを解決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。

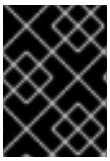
winbindd

winbindd サービスは、Name Service Switch (NSS) のインターフェースを提供し、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、ドメインユーザーは、Samba サーバーでホストされたサービスまたは他のローカルサービスの認証が可能となります。**winbind systemd** サービスは、**winbindd** デーモンを開始および停止します。

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。

16.1.2. testparm ユーティリティーを使用したsmb.conf ファイルの確認

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の検出だけでなく、ID マッピング用などの間違った設定も検出します。**testparm** に報告する問題がない場合、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、設定サービスが利用可能かどうか、または予想どおりに機能するかといったことを確認できない点に注意してください。



重要

Red Hat では、**/etc/samba/smb.conf** ファイルが修正されるたびに、**testparm** を使用してこのファイルを確認することを推奨しています。

/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、設定に間違ったパラメーター、値、その他のエラーがあると報告した場合は、問題を修正し、ユーティリティーを再度実行します。

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 ファイル内の ***[global]*** セクションの ***security*** パラメーターは、サービスに接続するユーザーを Samba が認証する方法を管理します。Samba にインストールするモードに応じて、パラメーターを設定する値は異なってきます。

- AD ドメインメンバー上で ***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セットアップする方法の詳細は、「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上で、***security*** = **user** を設定します。

  このモードでは、Samba はローカルデータベースを使用して接続する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セットアップする方法の詳細は、「Samba をスタンドアロンサーバーとしてセットアップ」 を参照してください。

- NT4 PDC または BDC で、***security*** = **user** を設定します。

  このモードで、Samba は ローカルまたは LDAP データベースにユーザーを認証します。

- NT4 ドメインメンバー上で ***security*** = **domain** を設定します。

  このモードで、Samba は NT4 PDC または BDC へ接続するユーザーを認証します。このモードは、AD ドメインメンバーでは使用できません。

  Samba をドメインメンバーとしてセットアップする方法の詳細は、「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) の 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セットアップ

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 はセントラル DC ではなく、ローカルデータベースでユーザーを認証します。さらに、ユーザーが認証なしに 1 つまたは複数のサービスに接続できるようゲストのアクセスを有効にすることができます。

16.1.4.1. スタンドアロンサーバーのサーバー設定のセットアップ

Samba をスタンドアロンサーバーとしてセットアップ

手順16.1 Samba をスタンドアロンサーバーとしてセットアップ

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、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
```

この設定は、**Example-WG** ワークグループ内の **Server** という名前のスタンドアロンサーバーを定義します。さらに、この設定は最低レベル (**1**) でのログオンを可能とし、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、接続するクライアントの NetBIOS 名の ***log file*** パラメーターに **%m** マクロを拡張します。これにより、各クライアントに個別のログファイルが可能となります。

詳細は、smb.conf(5) の 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。

3. ファイルまたはプリンター共有の設定。以下を参照してください。
   - 「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セットアップ 」
4. **/etc/samba/smb.conf** ファイルを確認します。

```
~]# testparm
```

詳細は、「**testparm** ユーティリティーを使用した**smb.conf** ファイルの確認」 を参照してください。

5. 認証が必要な共有をセットアップした場合は、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. 必要なポートを開き、**firewall-cmd** ユーティリティーを使用して、ファイアー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** サービスを開始します。

```
~]# systemctl start smb
```

8. あるいは、**smb** サービスを有効にして、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

共有への接続の際にユーザーが認証するよう有効化するには、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しなければなりません。Samba は、オペレーティングシステムのアカウントに対してファイルシステムオブジェクトのアクセス制御リスト (ACL) を確認するよう要求し、また、Samba アカウントに対して接続するユーザーを認証するよう要求します。

デフォルト設定の **passdb backend = tdbsam** を使用する場合、Samba は **/var/lib/samba/private/passdb.tdb** データベースにユーザーアカウントを保存します。

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。

手順16.2 Samba ユーザーの作成

1. オペレーティングシステムの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 の認証用にのみ使用される場合、アカウントがローカルでログインされることを防ぐため、**/sbin/nologin** コマンドをシェルとして割り当てます。

2. オペレーティングシステムのアカウントを有効化するためにパスワードを設定します。

```
~]# 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認証用として使用しません。しかし、アカウントを有効化するには自身でパスワードを設定する必要があります。アカウントが無効化されると、Samba はこのユーザーが接続した場合に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、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は、このパスワードを認証に使います。

4. Samba アカウントを有効化します。

```
~]# 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。使用できた場合は、以下が可能となります。

- 他のドメインメンバー上のドメインリソースへのアクセス
- **sshd** などのローカルサービスに対するドメインユーザーの認証
- ファイルおよびプリントサーバーとして機能するように、サーバー上にホストされたディレクトリーとプリンターの共有

16.1.5.1. ドメインへの参加

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。

手順16.3 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。

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. 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
```

このプラグインにより、Kerberos は DNS サービス記録を使用して AD サイトに基づいた Key Distribution Center (KDC) を探すことができます。

3. 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
```

4. ドメインに参加します。たとえば、**ad.example.com** という名前のドメインに参加するには、以下を実行します。

```
~]# realm join --client-software=winbind ad.example.com
```

realm ユーティリティーは、以前のコマンドを使用して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向けの **/etc/samba/smb.conf** ファイルの作成
- ユーザーおよびグループルックアップ向けの **winbind** モジュールを **/etc/nsswitch.conf** ファイルへ追加
- AD メンバーシップの **/etc/krb5.conf** ファイルに Kerberos クライアントを設定
- **/etc/pam.d/** ディレクトリーの Pluggable Authentication Module (PAM) 設定の更新
- **winbind** サービスを開始し、システム起動時のサービス開始を実現

realm ユーティリティーに関する詳細は、realm(8) の man ページおよび『Red Hat Windows 統合ガイド』の 『realmd を使用した Active Directory ドメインへの接続』 を参照してください。

5. オプションとして、**/etc/samba/smb.conf** ファイルに代替の ID マッピングバックエンドまたはカスタマイズ ID マッピングの設定を設定します。詳細は、「ID マッピングについて」 を参照してください。

6. オプションとして、設定を確認します。「Samba がドメインメンバーとして正常に参加したことを確認」 を参照してください。

16.1.5.2. Samba がドメインメンバーとして正常に参加したことを確認

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。以下を参照してください。

- 「オペレーティングシステムによるドメインユーザーアカウントとグループの取得が可能であることを確認」
- 「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を確認」
- 「利用可能なドメインの一覧表示」

オペレーティングシステムによるドメインユーザーアカウントとグループの取得が可能であることを確認

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** ドメイン内の **administrator** アカウントのクエリーには、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をクエリーするには、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、ファイルおよびディレクトリーにパーミッション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確認してください。たとえば、**/srv/samba/example.txt** ファイルのオーナーを **administrator** に、グループを **Domain Admins** に設定したい場合は、以下を実行します。

```
~]# chown administrator:"Domain Admins" /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を確認

AD 環境では、ユーザーは 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 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。

手順16.4 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一覧表示

winbindd サービスで利用可能なすべてのドメインを一覧表示するには、以下を入力します。

```
~]# wbinfo --all-domains
```

Samba がドメインメンバーとして正常に参加した場合、コマンドがビルトインおよびローカルホスト名を表示します。また、信頼されたドメインを含む Samba がメンバーのドメインも表示します。

例16.2 利用可能なドメインを表示します。

```
~]# 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、固有のセキュリティー識別子 (SID) でユーザーおよびグループを識別します。しかし、Linux は各ユーザーおよびグループに対して固有の UID と GID を要求します。ドメインメンバーとして Samba を実行する場合、**winbindd** サービスが、オペレーティングシステムにドメインユーザーおよびグループに関する情報を提供します。

winbindd サービスがユーザーおよびグループ固有の ID を Linux に提供できるようにするには、以下に対して **/etc/samba/smb.conf** ファイルに ID マッピングを設定しなければなりません。

- 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である AD または NT4 ドメイン

- 信頼された各ドメイン。ユーザーは、信頼された各ドメインからこの Samba サーバーのリソースにアクセス可能でなければならない

16.1.5.3.1. ID の範囲の計画

Linux UID および GID を AD に保存するかどうかに関わらず、またはこれらを生成するよう Samba を設定するかどうかに関わらず、各ドメイン設定は、他のドメインと重複 **してはならない** 固有の ID 範囲を要求します。

警告

ID の範囲を重複して設定した場合、Samba は正常に機能しません。

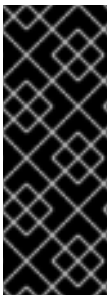
例16.3 一意の ID の範囲

以下は、重複しない ID マッピングの範囲のうち、デフォルト (*****)、**AD-DOM**、および **TRUST-DOM** のドメインを表示しています。

```
[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

ドメインごとに 1 つの範囲を割り当てることができます。したがって、ドメイン間の範囲と範囲の間には十分なスペースが必要です。これにより、後でドメインが広がった場合は範囲を広げることが可能となります。

後でドメインに別の範囲を割り当てた場合、これらのユーザーおよびグループが以前作成したファイルおよびディレクトリーの所有権は失われます。

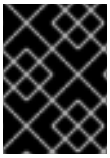
16.1.5.3.2. * デフォルトドメイン

ドメイン環境で、以下のそれぞれに対して 1 つの 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可能な信頼された各ドメイン

しかし、その他すべてのオブジェクトでは、Samba は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 ユーザーとグループ
- **BUILTIN\Administrators** などの Samba のビルトインアカウントおよびグループ



重要

Samba が正常に操作できるようにするには、本セクションで説明しているとおりにデフォルトドメインを設定する必要があります。

デフォルトのドメインバックエンドは、割り当てられた ID を永久に保存するために書き込み可能でなければなりません。

デフォルトのドメインでは、以下のバックエンドのいずれかを使用できます。

tdb

デフォルトドメインが **tdb** バックエンドを使用するように設定する場合は、将来的に作成されるオブジェクトのほか、定義されたドメイン ID マッピング設定外のオブジェクトを含めても十分な広さに ID 範囲を設定します。

たとえば、**/etc/samba/smb.conf** ファイルの ***[global]*** セクションに以下を設定します。

```
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

デフォルトドメインが **autorid** バックエンドを使用するよう設定する場合、オプションでドメイン向けに追加的な ID マッピング設定を追加できます。

たとえば、**/etc/samba/smb.conf** ファイルの ***[global]*** セクションに以下を設定します。

```
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、特定の設定向けに別の ID マッピングバックエンドを提供します。以下は、最もよく使われるバックエンドです。

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

バックエンド	ユースケース
rid	AD 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、バックエンドの使用箇所に関する推奨シナリオ、および設定方法について説明しています。

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、セキュリティー識別子 (SID)、UID、および GID マッピングテーブルを保存します。これには、ローカルユーザー、グループ、およびビルトインプリンシパルが含まれます。

* デフォルトドメインには、このバックエンドのみを使用します。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。

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、AD からのアカウントおよびグループ情報を読み込むために読み取り専用の API を実装します。これには、以下の利点があります。

- すべてのユーザーおよびグループ設定は、AD を中心に保存されます。
- ユーザーおよびグループ ID は、このバックエンドを使用するすべての Samba サーバー上で一貫性があります。
- ID は、破損の恐れのあるローカルデータベースに保存されないため、ファイルの所有権が不明になることはありません。

ad バックエンドは、AD から以下の属性を読み取ります。

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

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	オブジェクトに応じたユーザーまたはグループ名
uidNumber	ユーザー	ユーザー ID (UID)
gidNumber	グループ	グループ ID (GID)
loginShell [a]	ユーザー	ユーザーのシェルへのパス

AD 属性名	オブジェクトタイプ	マッピング先
unixHomeDirectory [a]	ユーザー	ユーザーのホームディレクトリーへのパス
primaryGroupID [b]	ユーザー	プライマリーグループの ID

[a] Samba は **idmap config *DOMAIN*:unix_nss_info = yes** に設定されている場合のみ、この属性を読み取ります。

[b] Samba は **idmap config *DOMAIN*:unix_primary_group = yes** に設定されている場合のみ、この属性を読み取ります。

16.1.5.4.2.1. 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には、以下に該当する必要があります。

- ユーザーおよびグループの両方は、AD に固有の ID セットを持つ必要があり、ID は **/etc/samba/smb.conf** ファイルに設定された範囲内でなければなりません。ID が範囲外のオブジェクトは、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は、必要とされるすべての属性のセットを AD に持つ必要があります。必要な属性がない場合は、ユーザーまたはグループは Samba サーバーでは利用できません。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。

16.1.5.4.2.2. 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。

手順16.5 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. 存在しない場合は、デフォルトドメイン (*) に ID マッピング設定を追加します。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 ドメイン内でユーザーおよびグループに割り当てられた 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

範囲は、このサーバー上の他のドメイン設定と重複しないようにします。さらに、範囲は将来的に割り当てられるすべての ID を含むことが可能な広さでなければなりません。詳細は、「ID の範囲の計画」 を参照してください。

d. Samba が AD からの属性を読み取る際に RFC 2307 スキーマを使用するよう設定します。

```
idmap config DOMAIN : schema_mode = rfc2307
```

e. Samba が、対応する AD 属性のユーザーホームディレクトリーのログインシェルとパスを読み取れるようにするには、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統一されたドメイン全体のホームディレクトリーパスとログインシェルの設定が可能です。

```
template shell = /bin/bash
template homedir = /home/%U
```

変数置換に関する詳細は、smb.conf(5) の 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 Linux 上のユーザーのプライマリーグループとして、ユーザーオブジェクトの ***primaryGroupID*** 属性を使用します。または、代わりに ***gidNumber*** 属性に設定した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** ファイルを確認します。

```
~]# testparm
```

詳細は、「**testparm** ユーティリティーを使用した**smb.conf** ファイルの確認」 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. 設定が予想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によるドメインユーザーアカウントとグループの取得が可能であることを確認」 を参照してください。

詳細は、smb.conf(5) および idmap_ad(8) の各 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 上に ID を生成できます。

注記

RID は、SID の最後の部分になります。たとえば、ユーザーの SID が **S-1-5-21-5421822485-1151247151-421485315-30014** の場合、RID は **30014** になります。Samba のローカル ID の算出方法に関する詳細は、idmap_rid(8) の 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 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たアカウントとグループ情報を算出するために読み取り専用の API を実装します。バックエンドを設定する際、**idmap config *DOMAIN* : range** パラメーターで RID の最小値と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た RID よりも小さい値または大きい値でユーザーやグループをマッピングすることはありません。

重要

rid は読み取り専用のバックエンドのため、**BUILTIN** グループ向けなどに新しい ID を割り当てることはできません。したがって、***** デフォルトドメインにこのバックエンドを使用しないでください。

16.1.5.4.3.1. rid バックエンドを使用する場合の利点と欠点

利点

- 設定範囲内に RID があるすべてのドメインユーザーおよびグループは、ドメインメンバー上で自動的に利用可能となります。
- 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。

欠点

- すべてのドメインユーザーは、同じログインシェルおよびホームディレクトリーを割り当てられます。しかし、変数を使用することも可能です。
- ユーザーおよびグループ ID は、すべてが同じ ID 範囲の設定で**rid** バックエンドを使用する場合にのみ、Samba ドメインメンバー全体で同じになります。
- ドメインメンバー上で利用可能な個々のユーザーまたはグループを除外することはできません。設定範囲外のユーザーおよびグループのみが除外されます。
- **winbindd** サービスが ID を算出するために使用する計算式に基づくと、別のドメインのオブジェクトが同じ RID を持つ場合、複数のドメイン環境では ID の重複が発生する恐れがあります。

16.1.5.4.3.2. 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。

手順16.6 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. 存在しない場合は、デフォルトドメイン (*) に ID マッピング設定を追加します。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。

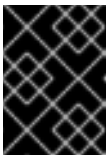
b. ドメイン向けに **rid** ID マッピングバックエンドを有効にします。

```
idmap config DOMAIN : backend = rid
```

c. 将来的に割り当てられるすべての RID を十分に含むことができる範囲を設定します。以下に例を示します。

```
idmap config DOMAIN : range = 2000000-2999999
```

Samba は、このドメインの RID が範囲外のユーザーおよびグループを無視します。



重要

範囲は、このサーバー上のその他のドメイン設定と重複してはいけません。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された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パスを設定します。

```
template shell = /bin/bash
template homedir = /home/%U
```

変数置換に関する詳細は、smb.conf(5) の 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** ファイルを確認します。

```
~]# testparm
```

詳細は、「**testparm** ユーティリティーを使用した**smb.conf** ファイルの確認」 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. 設定が予想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によるドメインユーザーアカウントとグループの取得が可能であることを確認」 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 [2]

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働きをしますが、別のドメインに自動的に ID を割り当てることができます。このため、以下のような状況では **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を対象。
- ***** デフォルトドメインおよび追加ドメインの場合は、ドメインが追加されるたびに ID マッピング設定を作成する必要がありません。
- 特定のドメインのみを対象。

16.1.5.4.4.1. `autorid` バックエンドを使用する場合の利点と欠点

利点

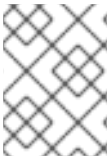
- 計算された UID および GID が設定範囲内にあるすべてのドメインユーザーおよびグループは、ドメインメンバー上で自動的に利用可能となります。
- ID、ホームディレクトリー、およびログインシェルを手動で割り当てる必要はありません。
- 複数のドメイン環境内の複数のオブジェクトが同じ RID を持つ場合でも、ID は重複しません。

欠点

- ユーザーおよびグループ ID は、Samba ドメインメンバー全体で同じではありません。
- すべてのドメインユーザーは、同じログインシェルおよびホームディレクトリーを割り当てられます。しかし、変数を使用することも可能です。
- ドメインメンバー上で利用可能な個々のユーザーまたはグループを除外することはできません。計算された UID および GID が設定範囲外のユーザーおよびグループのみが除外されます。

16.1.5.4.4.2. `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、オプションでドメイン向けに追加的な ID マッピング設定を追加できます。

手順16.7 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向けに **`autorid`** ID マッピングバックエンドを有効化します。

   ```
   idmap config * : backend = autorid
   ```

   b. 既存および将来的に割り当てられるすべてのオブジェクトに ID を割り当てるに十分なサイズ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、このドメイン内の計算された ID が範囲外のユーザーおよびグループを無視します。バックエンドの ID の計算方法に関する詳細は、idmap_autorid(8) の man ページの『THE MAPPING FORMULAS』 セクションを参照してください。

警告

範囲設定後に Samba が使用を開始すると、範囲の上限のみを増やすことができます。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. または、範囲の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た範囲のすべての ID が取得されるまで、各ドメインのオブジェクトに連続する ID の数字を割り当てます。詳細は、idmap_autorid(8) の 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された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パスを設定します。

```
template shell = /bin/bash
template homedir = /home/%U
```

変数置換に関する詳細は、smb.conf(5) の man ページの 『VARIABLE SUBSTITUTIONS』セクションを参照してください。

e. または、ドメイン向けに追加的な ID マッピング設定を追加します。 個々のドメイン用の設定が利用できない場合、Samba は以前設定された ***** 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

個々のドメイン用に追加的なバックエンドを設定する場合、すべての ID マッピング設定範囲は重複してはいけません。詳細は、「ID の範囲の計画」を参照してください。

2. **/etc/samba/smb.conf** ファイルを確認します。

```
~]# testparm
```

詳細は、「**testparm** ユーティリティーを使用した**smb.conf** ファイルの確認」 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. 設定が予想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によるドメインユーザーアカウントとグループの取得が可能であることを確認」 を参照してください。

16.1.6. IdM ドメインに Samba ファイルサーバーを統合

お使いの環境で Red Hat Identity Management (IdM) および Samba を実行する場合、IdM ユーザーが共有へアクセスする際に Kerberos を使用して認証するよう Samba サーバーを設定することができます。

詳細は、『Red Hat Windows 統合 ガイド』の該当セクションを参照してください。

16.1.7. Samba サーバーでのファイル共有の設定

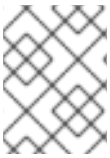
Samba をファイルサーバーとして使用する場合は、スタンドアロンまたはドメインメンバー設定の **/etc/samba/smb.conf** ファイルに共有を追加します。

以下のいずれかを使用する共有を追加できます。

- POSIX ACLs。「POSIX ACL を使用した共有のセットアップ 」 を参照してください。
- 粒度の細かい Windows ACL。「Windows ACL を使用した共有のセットアップ 」 を参照してください。

16.1.7.1. POSIX ACL を使用した共有のセットアップ [3]

Linux のサービスとして、Samba は POSIX ACL との共有をサポートします。これにより、**chmod** などのユーティリティーを使用して Samba サーバー上でパーミッションをローカルで管理できるようになります。共有が拡張属性をサポートするファイルシステムに保存されている場合、複数のユーザーおよびグループを持つ ACL を定義できます。



注記

粒度の細かい Windows ACL を代わりに使用する必要がある場合は、「Windows ACL を使用した共有のセットアップ 」 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。以下を参照してください。

- 「Samba をスタンドアロンサーバーとしてセットアップ」
- 「Samba をドメインメンバーとしてセットアップ」

16.1.7.1.1. 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。

手順16.8 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、**samba_share_t** コンテキストをディレクトリーに設定します。

```
~]# semanage fcontext -a -t samba_share_t "/srv/samba/example(/.*)?"
~]# restorecon -Rv /srv/samba/example/
```

3. ファイルシステム ACL をディレクトリーに設定します。詳細は 「ACL の設定」 を参照してください。

4. ***example*** 共有を **/etc/samba/smb.conf** ファイルに追加します。以下は、write-enabled の共有を追加する例です。

```
[example]
 path = /srv/samba/example/
 read only = no
```

注記

ファイルシステム ACL に関係なく、**read only = no** を設定しない場合、Samba は読み取り専用モードでディレクトリーを共有します。

5. **/etc/samba/smb.conf** ファイルを確認します。

```
~]# testparm
```

詳細は、「**testparm** ユーティリティーを使用した**smb.conf** ファイルの確認」 を参照してください。

6. 必要なポートを開き、**firewall-cmd** ユーティリティーを使用してファイアー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. あるいは、**smb** サービスを有効にして、起動時に自動的に起動するようにします。

```
~]# systemctl enable smb
```

16.1.7.1.2. ACL の設定

POSIX ACL サポートを使用する共有

- 標準 Linux ACL。詳細は、「標準 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。

16.1.7.1.2.1. 標準 Linux ACL の設定

Linux 上の標準 ACL は、1 人のオーナー、1 つのグループ、その他のすべての定義されていないユーザーに対する設定パーミッションをサポートします。ACL の更新には、**chown**, **chgrp** および **chmod**

ユーティリティーを使用できます。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。詳細は、「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

ディレクトリー上に set-group-ID (SGID) ビットを有効化することで、すべての新規ファイルとサブディレクトリーのデフォルトのグループをディレクトリーグループのものに自動的に設定します。これは、新規の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という通常の動作の代わりになります。

パーミッションに関する詳細は、chown(1) および chmod(1) の man ページを参照してください。

16.1.7.1.2.2. 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 をサポートする場合、これらを使用して複雑なパーミッションを設定できます。拡張 ACL には、複数のユーザーとグループへのパーミッションを含むことができます。

拡張 POSIX ACL により、複数のユーザーとグループを持つ複雑な ACL の設定が可能となります。しかし、設定るできるのは、以下のパーミッションに限定されます。

- アクセスなし
- 読み取りアクセス
- 書き込みアクセス
- 完全制御

Create folder / append data などの粒度の細かい Windows パーミッションが必要な場合、Windows ACL を使用するように共有を設定します。詳細は、「Windows ACL を使用した共有のセットアップ」 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。

手順16.9 共有における拡張 POSIX ACL の有効化

1. **/etc/samba/smb.conf** ファイルの共有のセクションにある以下のパラメーターを有効化し、拡張 ACL の ACL 継承を有効化します。

   ```
   inherit acls = yes
   ```

   詳細は、smb.conf(5) の 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。

2. **smb** サービスを再起動します。

   ```
   ~]# systemctl restart smb
   ```

3. あるいは、**smb** サービスを有効にして、起動時に自動的に起動するようにします。

```
~]# systemctl enable smb
```

4. ディレクトリーで 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

以下の手順では、**Domain Admins** グループの読み取り、書き込み、実行パーミッション、および**Domain Users** グループの読み取りと実行パーミッションを設定し、**/srv/samba/example/** ディレクトリー上のその他すべてへのアクセスを拒否します。

手順16.10 拡張 ACL の設定

1. ユーザーアカウントのプライマリーグループに対するパーミッションの自動付与を無効化します。

```
~]# 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、動的な **CREATOR GROUP** プリンシパルに追加でマッピングされます。Samba 共有で拡張 POSIX ACL を使用する場合、このプリンスパルは自動的に追加され、削除することはできません。

2. ディレクトリーでパーミッションを設定します。

a. 読み取り、書き込み、および実行のパーミッションを **Domain Admins** グループに付与します。

```
~]# setfacl -m group:"DOMAIN\Domain Admins":rwx
/srv/samba/example/
```

b. **Domain Users** グループに、読み取りおよび実行のパーミッションを与えます。

```
~]# setfacl -m group:"DOMAIN\Domain Users":r-x
/srv/samba/example/
```

c. **other** の ACL エントリーに対するパーミッションが、他の ACL エントリーと一致しないユーザーへのアクセスを拒否するよう設定します。

```
~]# setfacl -R -m other::--- /srv/samba/example/
```

これらの設定は、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 **This folder only** モードにマッピングされます。

3. 以前の手順で設定されたパーミッションを、このディレクトリーに作成された新規のファイルシステムオブジェクトに継承できるようにするには、以下を実行します。

```
~]# 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/
```

これらの設定により、プリンシパル用の **This folder only (このフォルダーのみ)** モードは現在、**This folder, subfolders, and files** に設定されています。

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	対象
DOMAIN\Domain Admins	完全制御	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 & 実行	このフォルダー、サブフォルダー、およびファイル
全員 [a]	なし	このフォルダー、サブフォルダー、およびファイル
owner (Unix User*owner*) [b]	完全制御	このフォルダーのみ
primary_group (Unix User*primary_group*) [c]	なし	このフォルダーのみ
CREATOR OWNER [d] [e]	完全制御	サブフォルダーおよびファイルのみ
CREATOR GROUP [f] [e]	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を **other** ACL エントリーからマッピングします。

[b] Samba は、このエントリーにディレクトリーのオーナーをマッピングします。

[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] 新規の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パーミッションを自動的に継承します。

[e] POSIX ACL を使用する共有でサポートされていない ACL からこれらのプリンシパルを設定または削除します。

[f] 新規のファイルシステムオブジェクトでは、作成者のプライマリーグループはこのプリンシパルのパーミッションを自動的に継承します。

16.1.7.1.3. 共有でパーミッションを設定

オプションとして、Samba 共有にアクセスを限定または付与するには、**/etc/samba/smb.conf** ファイルの共有のセクションで特定のパラメーターを設定できます。

注記

共有ベースのパーミッションは、ユーザー、グループ、またはホストが共有へアクセスできるかどうかを管理します。これらの設定がファイルシステム ACL へ影響を及ぼすことはありません。

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。

16.1.7.1.3.1. ユーザーおよびグループベースの共有アクセスの設定

ユーザーおよびグループベースのアクセス制御により、特定のユーザーまたはグループの共有へのアクセスを付与または拒否できます。たとえば、**Domain Users** グループのすべてのメンバーの **user** アカウントへのアクセスを拒否する一方で、共有へアクセスできるようにするには、以下のパラメーターを共有の設定に追加します。

```
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、**user** アカウントが **Domain Users** グループのメンバーの場合、以前の例を使用する場合は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) の 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。

16.1.7.1.3.2. 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 範囲をベースとする共有へのアクセスを付与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** ホストを有効化し、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。

手順16.11 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*** よりも優先度が高いです。たとえば、**client1.example.com** が、***hosts allow*** パラメーターに一覧表示された IP アドレスへと解決する場合、このホストへのアクセスは否定されます。

詳細は、smb.conf(5) の 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。

16.1.7.2. Windows ACL を使用した共有のセットアップ [4]

Samba は、Windows ACL を共有およびファイルシステムオブジェクトに設定するサポートをします。これにより、以下が可能となります。

- 粒度の細かい Windows ACL の使用
- Windows を使用した共有のパーミッションとファイルシステム ACL の管理

または、POSIX ACL を使用する共有の設定。詳細は、「POSIX ACL を使用した共有のセットアップ」 を参照してください。

16.1.7.2.1. SeDiskOperatorPrivilege 権限の付与

SeDiskOperatorPrivilege の特権を与えられたユーザーおよびグループのみが、Windows ACL を使用する共有にパーミッションを設定できます。たとえば、***DOMAIN*\Domain Admins** グループに特権を与える場合は、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

ドメイン環境にて、ドメイングループに **SeDiskOperatorPrivilege** を与えます。これにより、ユーザーのグループメンバーシップを更新することで、特権を中央管理できます。

SeDiskOperatorPrivilege を付与されたすべてのユーザーとグループを一覧表示するには、以下を実行します。

```
~]# net rpc rights list privileges SeDiskOperatorPrivilege \
       -U "DOMAIN\administrator"
Enter administrator's password:
SeDiskOperatorPrivilege:
  BUILTIN\Administrators
  DOMAIN\Domain Admins
```

16.1.7.2.2. Windows ACL サポートの有効化

Windows ACL をサポートする共有を設定するには、この機能を Samba で有効化する必要があります。すべての共有向けにグローバルに有効化するには、**/etc/samba/smb.conf** ファイルの ***[global]*** セクションに以下の設定を追加します。

```
vfs objects = acl_xattr
map acl inherit = yes
store dos attributes = yes
```

または、代わりに共有のセクションに同じパラメーターを追加することで、個別の共有に対して Windows ACL サポートを有効化できます。

16.1.7.2.3. 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。

手順16.12 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、**samba_share_t** コンテキストをディレクトリーに設定します。

```
~]# semanage fcontext -a -t samba_share_t "/srv/samba/example(/.*)?"
~]# restorecon -Rv /srv/samba/example/
```

3. ***example*** 共有を **/etc/samba/smb.conf** ファイルに追加します。以下は、write-enabled の共有を追加する例です。

```
[example]
 path = /srv/samba/example/
 read only = no
```

注記

ファイルシステム ACL に関係なく、**read only = no** を設定しない場合、Samba は読み取り専用モードでディレクトリーを共有します。

4. すべての共有の ***[global]*** セクションで Windows ACL サポートを有効化していない場合は、以下のパラメーターを ***[example]*** セクションに追加し、この共有向けにこの機能を有効化します。

```
vfs objects = acl_xattr
map acl inherit = yes
store dos attributes = yes
```

5. **/etc/samba/smb.conf** ファイルを確認します。

```
~]# testparm
```

詳細は、「**testparm** ユーティリティーを使用した**smb.conf** ファイルの確認」 を参照してください。

6. 必要なポートを開き、**firewall-cmd** ユーティリティーを使用してファイアー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
```

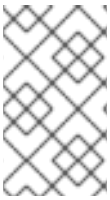
8. あるいは、**smb** サービスを有効にして、起動時に自動的に起動するようにします。

```
~]# systemctl enable smb
```

16.1.7.2.4. Windows ACL を使用する共有の共有パーミッションおよびファイルシステム 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** を使用した SMB 共有上での ACL の管理」 を参照してください。



注記

Windows のファイルシステムパーミッションを変更するには、特権を付与された **SeDiskOperatorPrivilege** のあるアカウントを使用する必要があります。詳細は、「**SeDiskOperatorPrivilege** 権限の付与」 を参照してください。

16.1.7.3. smbcacls を使用した SMB 共有上での ACL の管理

smbcacls ユーティリティーは、SMB 共有上に保存されたファイルおよびディレクトリーの ACL を一覧表示、設定、および削除できます。**smbcacls** を使用して、以下のようにファイルシステム ACL を管理できます。

- 詳細な Windows ACLs または POSIX ACL を使用したローカルまたはリモートの Samba サーバー上での管理。
- Windows でホストされる共有上の ACL をリモートで管理するための Red Hat Enterprise Linux での管理。

16.1.7.3.1. アクセス制御エントリーの理解

ファイルシステムオブジェクトの各 ACL エントリーには、以下のフォーマットで Access Control Entries (ACE) が含まれています。

```
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 **This folder, subfolders, and files** に適用される **Modify** パーミッションがある場合、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 のパーミッション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を付与または拒否するかを定義します。 値は **ALLOWED** または **DENIED** になります。

継承情報

以下の値が存在します。

表16.3 継承設定

値	詳細	マッピング先
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適用されません
ID	継承済み	ACE は親ディレクトリーから継承されました

さらに、値を以下のように組み合わせることが可能です。

表16.4 継承設定の組み合わせ

値の組み合わせ	Windows の **Applies to** 設定にマッピングします
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	サブフォルダーのみ
OI|IO	ファイルのみ

アクセス許可

この値は、1 つ以上の Windows パーミッションを表す 16 進数値または **smbcacls** エイリアスのいずれかになります。

- 1 つ以上の Windows パーミッションを表す 16 進数値

  以下の表は、 Windows パーミッションの詳細および対応する値を 16 進数値で表しています。

  表16.5 16 進数値での Windows パーミッションおよびそれに対応する smbcacls 値

Windows パーミッション	16 進数値
完全制御	`0x001F01FF`
フォルダーのトラバース / ファイルの実行	`0x00100020`
フォルダーの一覧表示 / データの読み取り	`0x00100001`
読み取り属性	`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`
サブフォルダーとファイルの削除	`0x00100040`
削除	`0x00110000`
パーミッションの読み取り	`0x00120000`
パーミッションの変更	`0x00140000`
所有権の取得	`0x00180000`

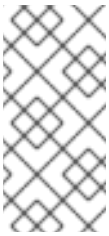
ビット単位の **OR** 操作を使用して、複数のパーミッションを単一の 16 進数値としてまとめることができます。詳細は、「ACE マスクの計算」 を参照してください。

- **`smbcacls`** エイリアス。以下の表は利用可能なエイリアスを表示します。

表16.6 既存の `smbcacls` エイリアスおよびそれに対応する Windows パーミッション

`smbcacls` エイリアス	Windows パーミッションへのマッピング
R	読み取り
READ	読み取り & 実行

smbcacls エイリアス	Windows パーミッションへのマッピング
W	特殊 • 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 • パーミッションの読み取り
D	削除
P	パーミッションの変更
O	所有権の取得
X	トラバース / 実行
CHANGE	変更
FULL	完全制御



注記

パーミッションを設定する場合、1 文字のエイリアスを複数組み合わせることができます。たとえば、Windows パーミッションの **Read** と **Delete** には、**RD** と設定することができます。しかし、2 文字以上のエイリアスを複数組み合わせたり、エイリアスと 16 進数値を組み合わたりしてはいけません。

16.1.7.3.2. smbcacls を使用した ACL の表示

--add などのオペレーションパラメーターなしに **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 root ディレクトリーの ACL を一覧表示するには、以下を実行します。

```
~]# 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、以下を表示します。

- **REVISION**: セキュリティー記述子の内部的な Windows NT ACL のリビジョン
- **CONTROL**: セキュリティー記述子制御
- **OWNER**: セキュリティー記述子のオーナーの名前または 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の理解」 を参照してください。

16.1.7.3.3. ACE マスクの計算

ほとんどの状況において、ACE を追加または更新する場合、表16.6「既存の **smbcacls** エイリアスおよびそれに対応する Windows パーミッション」 に一覧表示された **smbcacls** エイリアスを使用します。

しかし、表16.5「16 進数値での Windows パーミッションおよびそれに対応する **smbcacls** 値」 に一覧表示されているように詳細な Windows パーミッションを設定したい場合は、ビット単位の **OR** 操作を使用して正しい値を計算する必要があります。値の計算には、以下のシェルコマンドを使用することが可能です。

```
~]# 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。

- **Traverse folder / execute file** (**0x00100020**)
- **List folder / read data** (**0x00100001**)
- **Read attributes** (**0x00100080**)

以前のパーミッション用に 16 進数値を計算するには、以下を入力します。

```
~]# echo $(printf '0x%X' $(( 0x00100020 | 0x00100001 | 0x00100080 )))
0x1000A1
```

ACE を設定または更新した場合は返された値を使用します。

16.1.7.3.4. 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パスする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、および削除することができます。

ACL の追加

AD\Domain Users グループに、***This folder, subfolders, and files*** の ***CHANGE*** パーミッションを与える **//server/example** 共有の root に ACL を追加するには、以下を実行します。

```
~]# smbcacls //server/example / -U "DOMAIN\administrator \
       --add ACL:"AD\Domain Users":ALLOWED/OI|CI/CHANGE
```

ACL の更新
ACL の更新は、新規の ACL の追加に似ています。既存のセキュリティープリンシパルと共に ***--modify*** パラメーターを使用して ACL をオーバーライドすることで、ACL を更新します。**smbcacls** が ACL リストにセキュリティープリンシパルを見つけた場合は、ユーティリティーがパーミッションを更新します。そうでなければ、コマンドは以下のエラーを表示して失敗します。

```
SID principal_name の ACL が見つかりません
```

たとえば、**AD\Domain Users** グループのパーミッションを更新し、***This folder, subfolders, and files*** 向けに ***READ*** に設定する場合は、以下を実行します。

```
~]# 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
```

16.1.7.4. ユーザーによる Samba サーバーでのディレクトリー共有の有効化

Samba サーバー上では、**root** パーミッションなしに、ユーザーがディレクトリーを共有できるよう設定することが可能です。

16.1.7.4.1. ユーザー共有機能の有効化

ユーザーがディレクトリーを共有できるようになる前に、管理者は Samba でユーザー共有を有効化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を有効化し、ユーザー共有を作成するには、以下を実行します。

手順16.13 ユーザー共有の有効化

1. ローカルの **example** グループがない場合は、これを作成します。

   ```
   ~]# groupadd example
   ```

2. Samba のディレクトリーを用意し、ユーザー共有定義を保存してこのパーミッションを適切に設定します。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パーミッションを設定します。

```
~]# 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を変更したり、削除したりしないように 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. ユーザー共有定義を保存するために設定したディレクトリーへ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. Samba がこのサーバー上で作成を許可す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 **0** のデフォルトを使用する場合、ユーザー共有は無効化されます。

   c. オプションとして、絶対ディレクトリーパスの一覧を設定します。たとえば、Samba は **/data** および **/srv** ディレクトリーのサブディレクトリーのみの共有を許可すると設定する場合、以下のように設定します。

   ```
   usershare prefix allow list = /data /srv
   ```

   自身で設定可能なさらなるユーザー共有関連のパラメーターの一覧に関しては、smb.conf(5) の man ページの 『USERSHARES』 セクションを参照してください。

4. **/etc/samba/smb.conf** ファイルを確認します。

   ```
   ~]# testparm
   ```

   詳細は、「**testparm** ユーティリティーを使用した**smb.conf** ファイルの確認」 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
   ```

ユーザーは、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。

16.1.7.4.2. ユーザー共有の追加

「ユーザー共有機能の有効化」 に従って Samba を設定した後、ユーザーは **net usershare add** コマンドを実行することで **root** パーミッションがなくても Samba サーバー上でディレクトリーを共有できます。

net usershare add コマンドの概要

net usershare add share_name path [[comment] | [ACLs]] [guest_ok=y|n]

重要

ユーザー共有の作成時に ACL を設定する場合、ACL の前にコメント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、二重引用符の付いた空の文字列を使用します。

ユーザーが、ユーザー共有でゲストアクセスを有効化できるのは、**/etc/samba/smb.conf** ファイルの ***[global]*** セクションにおいて管理者が **usershare allow guests = yes** と設定した場合のみである点に注意してください。

例16.7 ユーザー共有の追加

Samba サーバー上で **/srv/samba/** ディレクトリーを共有したいユーザーがいるとします。共有の名前は **example** とする必要があり、コメントの設定はなく、ゲストユーザーがアクセス可能でなければなりません。さらに、**AD\Domain Users** グループは共有パーミッションにフルアクセスできるように設定し、その他のユーザーは読み取りパーミッションに設定する必要があります。この共有を追加するには、ユーザーとして以下を実行します。

```
~]$ net usershare add example /srv/samba/ "" \
       "AD\Domain Users":F,Everyone:R guest_ok=yes
```

16.1.7.4.3. ユーザー共有の設定の共有

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。詳細は、「ユーザー共有の追加」 を参照してください。

16.1.7.4.4. 既存のユーザー共有に関する情報の表示

ユーザーは、Samba サーバー上で **net usershare info** コマンドを入力して、ユーザー共有およびその設定を表示することができます。

あらゆる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一覧表示するには、***-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したい場合は、共有名またはワイルドカードをコマンドにパスします。たとえば、名前が ***share_*** から始まる共有に関する情報を表示するには、以下を実行します。

```
~]$ net usershare info -l share_*
```

16.1.7.4.5. ユーザー共有の一覧表示

Samba サーバー上で、設定なしで利用可能なユーザー共有のみを一覧表示したい場合は、**net usershare list** コマンドを使用します。

あらゆるユーザーが作成した共有を一覧表示するには、以下を実行します。

```
~]$ 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一覧表示するには、***-l*** パラメーターを省略します。

特定の共有のみを一覧表示するには、コマンドに共有名またはワイルドカードをパスします。たとえば、名前が ***share_*** から始まる共有のみを一覧表示するには、以下を実行します。

```
~]$ net usershare list -l share_*
```

16.1.7.4.6. 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
```

16.1.7.5. 共有へのゲストアクセスの有効化

特定の状況では、ユーザーが認証なしに接続できるディレクトリーを共有したい場合もあります。これを設定するには、共有上でゲストアクセスを有効化します。

警告

認証を必要としない共有はセキュリティーリスクとなる恐れがあります。

共有上でゲストアクセスが有効化された場合、Samba は ***guest account*** パラメーターに設定されたオペレーティングシステムアカウントにゲスト接続をマッピングします。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。

- アカウントは、ファイルシステム ACL に一覧表示されています。
- **other** ユーザー向けの POSIX パーミッションが許可します。

例16.8 ゲストの共有パーミッション

ゲストアカウントを **nobody** にマッピングするよう Samba を設定した場合 (デフォルト)、ACL は以下を実行します。

- ゲストユーザーが **file1.txt** を読み取ることを許可。

- ゲストユーザーが **file2.txt** を読み取り、変更することを許可。
- ゲストユーザーが **file3.txt** を読み取ること、または変更することを阻止。

```
-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化するには、以下を実行します。

手順16.14 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、このサーバーにセットアップする最初のゲスト共有の場合は、以下を実行します。
      i. ***[global]*** セクションで、**map to guest = Bad User** を設定します。

```
[global]
 ...
 map to guest = Bad User
```

      この設定で、Samba はユーザー名が存在しない限り、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存在せず、ゲストアクセスが共有で有効化されている場合は、Samba は接続をゲストログインとして扱います。

      ii. デフォルトでは、Samba はゲストアカウントを Red Hat Enterprise Linux 上の **nobody** アカウントにマッピングします。オプションで、別のアカウントを設定することができます。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されたアカウントは、Samba サーバー上で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 は有効なシェルが割り当てられていないアカウントの使用を推奨します。

   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** ファイルを確認します。

```
~]# testparm
```

   詳細は、「**testparm** ユーティリティーを使用した**smb.conf** ファイルの確認」 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
```

16.1.8. Samba プリントサーバーのセットアップ [5]

Samba をプリントサーバーとしてセットアップした場合、お使いのネットワーク内のクライアントは Samba を使用して印刷できます。さらに、設定されている場合は、Windows のクライアントは Samba サーバーからドライバーをダウンロードできます。

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セットアップ」
- 「Samba をドメインメンバーとしてセットアップ」

16.1.8.1. Samba spoolssd サービス

Samba **spoolssd** は、**smbd** サービスに統合されたサービスです。Samba 設定内の **spoolssd** を有効化し、多数のジョブまたはプリンターなど、プリントサーバーのパフォーマンスを大幅に向上させます。

spoolssd なしでは、Samba は **smbd** プロセスを fork し、各印刷ジョブの **printcap** キャッシュを初期化します。プリンターが多数の場合は、キャッシュが初期化されるまでの数秒間ほど、**smbd** サービスが反応しないことがあります。**spoolssd** サービスにより、遅延なしに印刷ジョブを処理する pre-fork の **smbd** プロセスを開始することが可能となります。主要な **spoolssd smbd** プロセスは、少量のメモリーおよび fork を使用し、子プロセスを終了します。

spoolssd サービスを有効化するには、以下を実行します。

手順16.15 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. オプションで、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	Samba は、新規の接続が確立された場合に、このパラメーターに設定された新規の子プロセスの数を ***spoolssd:prefork_max_children*** に設定された上限値まで fork します
spoolssd:prefork_max_allowed_clients	100	子プロセスが対応するクライアントの数
spoolssd:prefork_child_min_life	60	秒単位での子プロセスの最短寿命。最短は 60 秒

2. **/etc/samba/smb.conf** ファイルを確認します。

```
~]# testparm
```

詳細は、「**testparm** ユーティリティーを使用した**smb.conf** ファイルの確認」 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
```

サービスの再起動後、Samba は自動的に **smbd** 子プロセスを開始します。

```
~]# ps axf
...
30903 smbd
30912  \_ smbd
30913      \_ smbd
30914      \_ smbd
30915      \_ smbd
...
```

16.1.8.2. Samba でのプリントサーバーサポートの有効化

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。

手順16.16 Samba でのプリントサーバーサポートの有効化

1. Samba サーバー上で CUPS をセットアップし、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

CUPS が Samba プリントサーバー上にローカルにインストールされている場合、Samba が CUPS に転送できるのは印刷ジョブのみです。

2. **/etc/samba/smb.conf** ファイルを編集します。

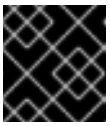
   a. **spoolssd** サービスを有効化したい場合、***[global]*** セクションに以下のパラメーターを追加してください。

   ```
   rpc_server:spoolss = external
   rpc_daemon:spoolssd = fork
   ```

   詳細は、「Samba **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 の共有名はハードコードされ、変更することはできません。

3. **/etc/samba/smb.conf** ファイルを確認します。

   ```
   ~]# testparm
   ```

   詳細は、「**testparm** ユーティリティーを使用した**smb.conf** ファイルの確認」 を参照してください。

4. 必要なポートを開き、**firewall-cmd** ユーティリティーを使用してファイアー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
   ```

サービスの再起動後、Samba は CUPS バックエンドに設定され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したい場合は、「特定のプリンターを手動で共有」 を参照してください。

16.1.8.3. 特定のプリンターを手動で共有

デフォルトで Samba をプリントサーバーとして設定した場合、Samba は CUPS バックエンドで設定されたすべてのプリンターを共有します。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。

手順16.17 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]*** セクションで以下を設定することで、自動プリンター共有を無効化します。

```
load printers = no
```

b. 共有したい各プリンターにセクションを追加します。たとえば、CUPS バックエンドで **example** と名付けられたプリンターを Samba で **Example-Printer** として共有する場合は、以下のセクションを追加します。

```
[Example-Printer]
 path = /var/tmp/
 printable = yes
 printer name = example
```

各プリンターには、個別のスプールディレクトリーは必要ありません。***[printers]*** セクションに設定するように、プリンターの ***path*** パラメーターに同じスプールディレクトリーを設定することが可能です。

2. **/etc/samba/smb.conf** ファイルを確認します。

```
~]# testparm
```

詳細は、「**testparm** ユーティリティーを使用した**smb.conf** ファイルの確認」 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
```

16.1.8.4. Windows クライアント向け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 [6]

Windows クライアント向けに Samba プリントサーバー実行している場合、ドライバーをアップロードしてプリンターを事前に設定できます。ユーザーがプリンターに接続した場合、Windows はローカルでクライアント上にドライバーを自動的にダウンロードおよびインストールします。インストールに際して、ローカル管理パーミッションはユーザーには必要ではありません。さらに、Windows はトレイの数など事前に設定された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セットアップ 」 を参照してください。

16.1.8.4.1. 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情報を提供します。

サポートされるドライバーモデルバージョン
Samba は、プリンタードライバーモデルバージョン 3 のみをサポートします。これは、Windows 2000 以降および Windows Server 2000 以降でサポートされています。Samba は、Windows 8 および Windows Server 2012 に導入されたドライバーモデルバージョン 4 をサポートしません。しかし、これ以降の Windows のバージョンも、バージョン 3 のドライバーをサポートしています。

package-aware ドライバー
Samba は、package-aware ドライバーをサポートしません。

アップロード向けの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できるようになる前に、以下を実行します。

- ドライバーが圧縮形式で提供される場合は、これを解凍します。
- ドライバーによっては、Windows ホスト上でローカルにドライバーをインストールするセットアップアプリケーションを開始する必要があります。特定の状況では、セットアップの実行中にインストーラーが個々のファイルをオペレーティングシステムの一時フォルダーに展開します。ドライバーファイルをアップロードに使用するには、以下を実行します。

  1. インストーラーを起動します。
  2. 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。
  3. インストールをキャンセルします。

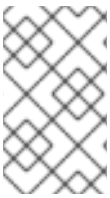
プリントサーバーへのアップロードをサポートするドライバーについて、プリンターの製造元に問い合わせしてください。

クライアントへのプリンターに 32 ビットと 64 ビットのドライバーを提供します。
32 ビットと 64 ビットの両方の Windows クライアントにプリンター用のドライバーを提供するには、両方のアーキテクチャーにまったく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、**Example PostScript** という名前の 32 ビットのドライバーと、**Example PostScript (v1.0)** という名前の 64 ビットのドライバーをアップロードする場合、名前が一致しません。その結果、プリンターには 1 つのドライバーのみを割り当てることになり、そのドライバーは両方のアーキテクチャーでは利用できません。

16.1.8.4.2. ユーザーによるドライバーのアップロードと事前設定の有効化

プリンタードライバーをアップロードして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** 特権を与える必要があります。たとえば、**printadmin** グループに特権を与える場合、以下を実行します。

```
~]# 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、ドメイングループに **SePrintOperatorPrivilege** を与えます。これにより、ユーザーのグループメンバーシップを更新することで、特権を中央管理できます。

SePrintOperatorPrivilege を与えられたすべてのユーザーおよびグループを一覧表示するには、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
```

16.1.8.4.3. print$ 共有のセットアップ

Windows オペレーティングシステムは、プリントサーバーの **print$** という名前の共有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 でハードコードされ、変更することはできません。

/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。

手順16.18 print$ 共有のセットアップ

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
   ```

   これらの設定の使用

   - **printadmin** グループのメンバーのみ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。
   - 新規ファイルのパーミッションは **664** に設定されます。
   - 新規のディレクトリーのパーミッションは、**2775** に設定されます。

2. プリンターに 64 ビットのドライバーのみをアップロードする場合は、この設定を ***[global]*** section in the **/etc/samba/smb.conf** に含めます。

   ```
   spoolss: architecture = Windows x64
   ```

   この設定なしでは、Windows は少なくとも 32 ビットバージョンをアップロードしたドライバーのみを表示します。

3. **/etc/samba/smb.conf** ファイルを確認します。

   ```
   ~]# testparm
   ```

   詳細は、「**testparm** ユーティリティーを使用した**smb.conf** ファイルの確認」 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、これを作成します。

```
~]# groupadd printadmin
```

6. **SePrintOperatorPrivilege** 権限を **printadmin** グループに与えます。

```
~]# net rpc rights grant "printadmin" SePrintOperatorPrivilege \
       -U "DOMAIN\administrator"
Enter DOMAIN\administrator's password:
Successfully granted rights.
```

詳細は、「ユーザーによるドライバーのアップロードと事前設定の有効化」 を参照してください。

7. SELinux を **enforcing** モードで実行する場合、**samba_share_t** コンテキストをディレクトリーに設定します。

```
~]# semanage fcontext -a -t samba_share_t
"/var/lib/samba/drivers(/.*)?"
~]# restorecon -Rv /var/lib/samba/drivers/
```

8. **/var/lib/samba/drivers/** ディレクトリーにパーミッションを設定します。

   - 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
CREATOR OWNER	完全制御	サブフォルダーおよびファイルのみ
Authenticated Users	読み取り & 実行、フォルダーコンテンツの一覧表示、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。

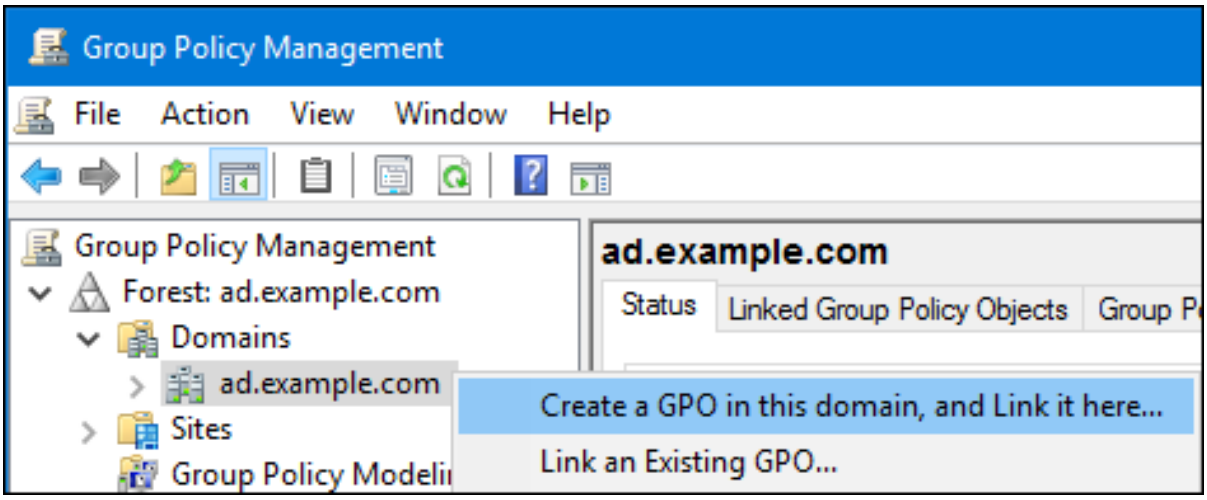
16.1.8.4.4. クライアントが Samba プリントサーバーを信頼できるように GPO を作成

セキュリティー上の理由から、最近の Windows オペレーティングシステムは、クライアントが信頼できないサーバーから non-package-aware 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ています。お使いのプリントサーバーが AD のメンバーの場合、Samba サーバーを信頼するためにドメインに Group Policy Object (GPO) を作成することができます。

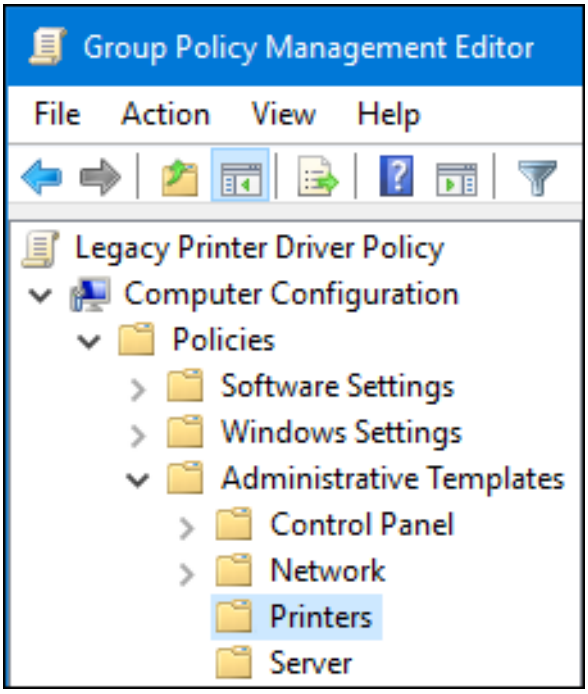
GPO を作成するには、お使いの Windows コンピューターに Windows Remote Server Administration Tools (RSAT) がインストールされていなければなりません。詳細は、Windows ドキュメントを参照してください。

手順16.19 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 GPO を作成

1. AD ドメイン **`Administrator`** ユーザーなど、グループポリシーの編集が許可された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** コンソールを開きます。
3. AD ドメインを右クリックして、**`Create a GPO in this domain, and Link it here (このドメインに GPO を作成し、ここにリンクします)`** を選択します。



4. **`Legacy printer Driver Policy`** など、GPO の名前を入力し、**`OK`** をクリックします。ドメインエントリーの下に、新しい GPO が表示されます。
5. 新規に作成された GPO を右クリックし、**`Edit`** を選択して **Group Policy Management Editor** を開きます。
6. **Computer Configuration** → **Policies** → **Administrative Templates** → **Printers** に移動します。

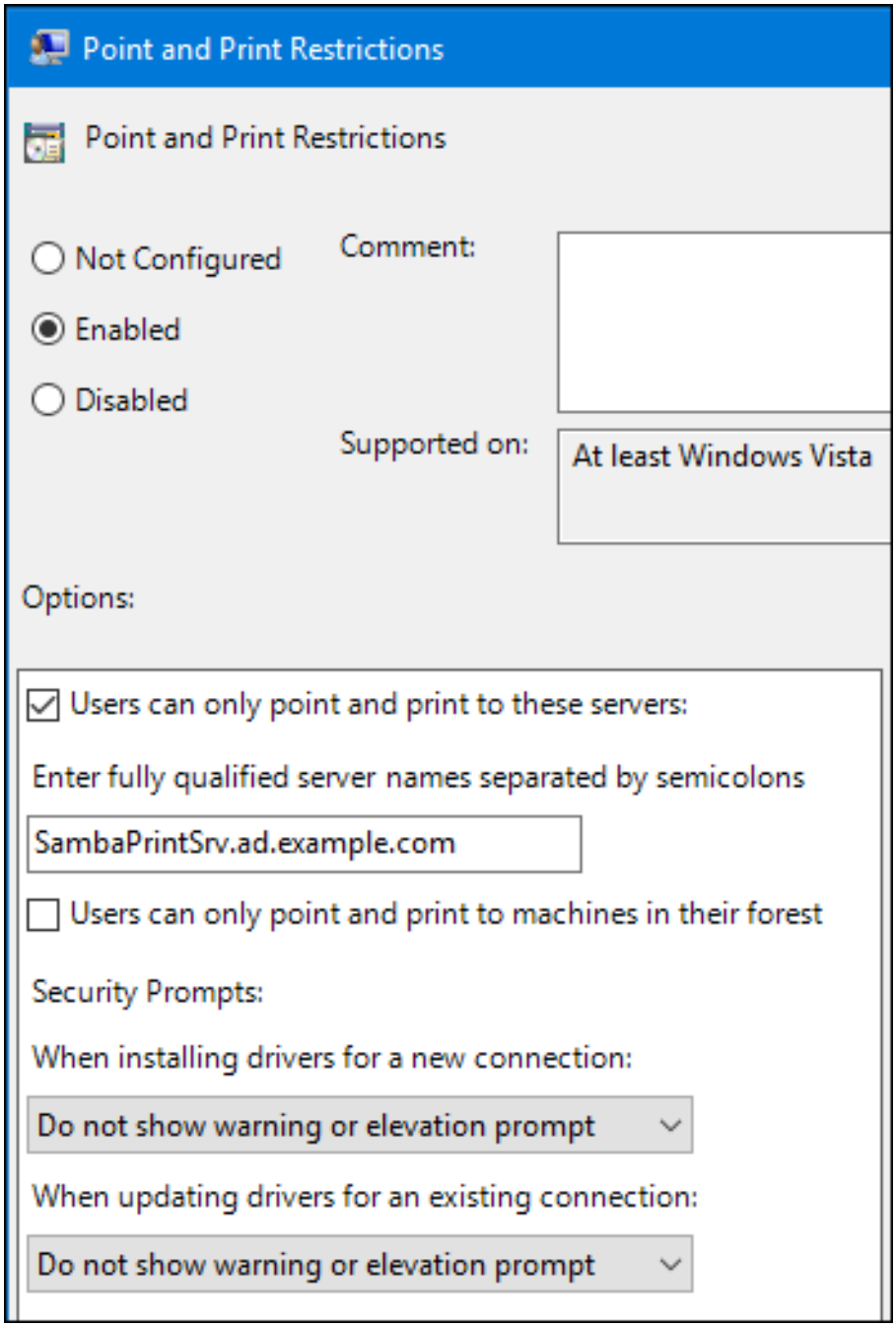


7. ウィンドウの右側で、**Point and Print Restriction (ポイントアンドプリント制限)** をダブルクリックしてポリシーを編集します。

   a. ポリシーを有効化し、以下のオプションを設定します。

      i. **Users can only point and print to these servers (ユーザーはこれらのサーバーにポイントアンドプリントのみできます)** を選択し、Samba プリントサーバーの完全修飾ドメイン名 (FQDN) をこのオプションの横にあるフィールドに入力します。

      ii. **Security Prompts (セキュリティープロンプト)** の下にあるチェックボックスは両方とも、**Do not show warning or elevation prompt (警告またはエレベーションプロンプトを表示しない)** を選択します。

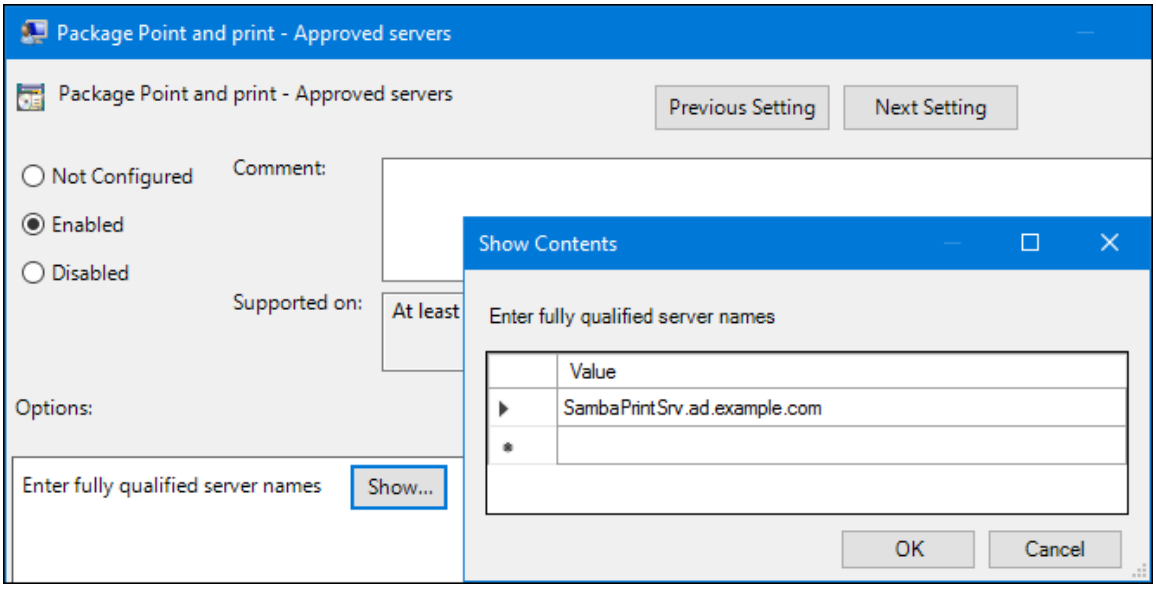


b. **OK** をクリックします。

8. ポリシーを編集するには、**Package Point and Print - Approved servers (ポイントアンドプリントをパッケージ - 承認済みサーバー)** をダブルクリックします。

   a. ポリシーを有効化し、**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** コンソールを閉じます。

Windows ドメインメンバーによるグループポリシーの適用後、ユーザーがプリンターに接続するとプリンタードライバーは 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。

16.1.8.4.5. 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 クライアントの **Print Management** アプリケーションを使用して、Samba プリントサーバー上でホストされるドライバーをアップロードし、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。

16.1.9. Samba サーバーのパフォーマンスのチューニング [7]

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について説明します。

16.1.9.1. SMB プロトコルバージョンの設定

SMB の新バージョンはそれぞれ、機能を追加してプロトコルのパフォーマンスを向上させます。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をサポートします。Samba が最新のプロトコルバージョンも使用する場合、Samba に接続する Windows クライアントはパフォーマンスの向上による恩恵を受けます。Samba では、***server max protocol*** のデフォルト値は、サポート対象の安定した最新の SMB プロトコルバージョンに設定されています。

最新の安定した SMB プロトコルバージョンを常に有効にするには、***server max protocol*** パラメーターを設定しません。手動でマニュアルを設定した場合、最新のプロトコルバージョンを有効化するために、SMB プロトコルを新バージョンごとに設定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。

16.1.9.2. 多数のファイルを含むディレクトリーでの共有のチューニング

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。

手順16.20 多数のファイルを含むディレクトリーでの共有のチューニング

1. 共有にあるすべてのファイル名を小文字に変更します。

注記

この手順の設定を使用して、小文字以外のファイル名は今後表示されません。

2. 共有のセクションで、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
```

パラメーターに関する詳細は、smb.conf(5) の man ページにある説明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
```

これらの設定の適用後、この共有にある新規に作成されたすべてのファイルの名前は、小文字を使用しています。これらの設定により、Samba はディレクトリーの大文字小文字をスキャンする必要がなくなり、パフォーマンスが向上します。

16.1.9.3. パフォーマンスを低下させる恐れのある設定

デフォルトで、Red Hat Enterprise Linux のカーネルのネットワークパフォーマンスは、高くチューニングされています。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** ファイルの ***socket options*** パラメーターを設定すると、これらのカーネル設定をオーバーライドします。その結果、ほとんどの場合、このパラメーターを設定すると Samba ネットワークのパフォーマンスは低下します。

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。

16.1.10. よく使われる 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について説明します。

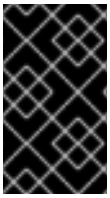
16.1.10.1. net ユーティリティーの使用

net ユーティリティーを使用すると、複数の管理タスクを Samba サーバーで実行することができます。本セクションでは、**net** ユーティリティーのサブコマンドで最もよく使うものについて説明します。

詳細は、net(8) の man ページを参照してください。

16.1.10.1.1. net ads join 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 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、手動で **/etc/samba/smb.conf** ファイルを作成する必要があります。また、オプションで PAM などの追加的な設定を更新します。



重要

Red Hat は、ドメインへの参加には **realm** ユーティリティーの使用を推奨しています。**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、「ドメインへの参加」 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。

手順16.21 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 ドメインのメンバーには、以下を実行してください。

```
[global]
workgroup = domain_name
security = ads
passdb backend = tdbsam
realm = AD_REALM
```

   - NT4 ドメインのメンバーには、以下を実行してください。

```
[global]
workgroup = domain_name
security = user
passdb backend = tdbsam
```

2. デフォルトドメインの * および参加したいドメインの 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** ファイルを確認します。

```
~]# testparm
```

   詳細は、「**testparm** ユーティリティーを使用した**smb.conf** ファイルの確認」 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、以下を実行してください。

```
~]# net ads join -U "DOMAIN\administrator"
```

   - NT4 ドメインに参加するには、以下を実行してください。

```
~]# net rpc join -U "DOMAIN\administrator"
```

5. **winbind** ソースを **/etc/nsswitch.conf** ファイルの ***passwd*** および ***group*** データベースエントリーに追加します。

```
passwd:     files winbind
group:      files winbind
```

6. **winbind** サービスを有効化し、開始します。

```
~]# 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 システムレベルの認証ガイド』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 環境はオプションで、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『Kerberos クライアントの設定』セクションを参照してください。

16.1.10.1.2. 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権限を割り当て、特別な操作を実行することができます。たとえば、共有に ACL を設定したり、プリンタードライバーをアップロードしたりできます。Samba サーバーでは、**net rpc rights** コマンドを使って権限を管理できます。

権限の一覧表示
利用可能な権限およびそのオーナーを一覧表示するには、**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
```

権限の付与
アカウントまたはグループに権限を付与するには、**net rpc rights grant** コマンドを使用します。

たとえば、***SePrintOperatorPrivilege*** 権限を ***DOMAIN\printadmin*** グループに付与する場合、以下を実行します。

```
~]# net rpc rights grant "DOMAIN\printadmin" SePrintOperatorPrivilege \
       -U "DOMAIN\administrator"
```

```
Enter DOMAIN\administrator's password:
Successfully granted rights.
```

権限の取り消し
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.
```

16.1.10.1.3. 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 あるいは Windows のサーバー上にある共有を一覧表示、追加、および取り消す機能を提供します。

共有の一覧表示
SMB サーバーの共有を一覧表示するには、**net rpc share list command** を使用します。オプションで、***-S server_name*** パラメーターをコマンドにパスして、リモートサーバーの共有を一覧表示します。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に **browseable = no** が設定された Samba サーバーにホストされた共有は、出力では表示されません。

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することができます。

たとえば、**C:\example** ディレクトリーを共有するリモートの Windows のサーバー上にある **example** という名前の共有を追加するには、以下を実行します。

```
~]# net rpc share add example="C:\example" -U "DOMAIN\administrator" -S
server
```

注記

Windows ディレクトリー名を指定する際、パスの末尾にあるバックスラッシュを省略しなければなりません。

Samba サーバーに共有を追加するためにコマンドを使用するには、以下を実行します。

- ***-U*** パラメーターで指定されたユーザーに、**SeDiskOperatorPrivilege** 権限を付与する必要があります。
- **/etc/samba/smb.conf** ファイルに共有のセクションを追加し、Samba を再度読み込むスクリプトを書く必要があります。このスクリプトは、**/etc/samba/smb.conf** の ***[global]*** セクションにおける ***add share command*** パラメーターに設定されなければなりません。詳細は、smb.conf(5) の man ページの ***add share command*** に関する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することができます。

たとえば、リモートの Windows のサーバーから **example** という名前の共有を削除する場合は、以下を実行します。

```
~]# net rpc share delete example -U "DOMAIN\administrator" -S server
```

Samba サーバーから共有を削除するためにコマンドを使用するには、以下を実行します。

- ***-U*** パラメーターで指定されたユーザーに、**SeDiskOperatorPrivilege** 権限を付与する必要があります。
- **/etc/samba/smb.conf** ファイルから共有のセクションを削除し、Samba を再度読み込むスクリプトを書く必要があります。このスクリプトは、**/etc/samba/smb.conf** の ***[global]*** セクションにおける ***delete share command*** パラメーターに設定されなければなりません。詳細は、smb.conf(5) の man ページの ***delete share command*** に関する説明を参照してください。

16.1.10.1.4. 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 上で、以下の動作を実行できるようになります。

- すべてのユーザーアカウントの一覧表示
- ユーザーの追加
- ユーザーの削除

注記

AD ドメインは ***ads***、NT4 ドメインは ***rpc*** など、接続方法を指定することは、ドメインユーザーアカウントを一覧表示した場合のみ必要です。他のユーザー関連のサブコマンドは、接続方法を自動検出できます。

-U user_name パラメーターをコマンドにパスして、要求された動作の実行を許可されたユーザーを指定します。

ドメインユーザーアカウントの一覧表示
AD ドメイン内のすべてのユーザーを一覧表示するには、以下を実行します。

```
~]# net ads user -U "DOMAIN\administrator"
```

NT4 ドメイン内のすべてのユーザーを一覧表示するには、以下を実行します。

```
~]# net rpc user -U "DOMAIN\administrator"
```

ドメインへのユーザーアカウントの追加
Samba ドメインメンバーでは、**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することができます。

たとえば、**user** アカウントをドメインに追加するには、以下を実行します。

手順16.22 ドメインへのユーザーアカウントの追加

1. アカウントの追加

```
~]# net user add user password -U "DOMAIN\administrator"
User user added
```

2. オプションで、Remote Procedure Call (RPC) シェルを使用して AD DC または NT4 PDC 上のアカウントを有効化します。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
```

ユーザーアカウントをドメインから削除
Samba ドメインメンバーでは、**net user delete** コマンドを使用して、ユーザーアカウントをドメインから削除することができます。

たとえば、**user** アカウントをドメインから削除するには、以下を実行します。

```
~]# net user delete user -U "DOMAIN\administrator"
User user deleted
```

16.1.10.1.5. net usershare コマンドの使用

「ユーザーによる Samba サーバーでのディレクトリー共有の有効化」 を参照してください。

16.1.10.2. 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 サーバー上で、クライアントサイドの Microsoft Remote Procedure Call (MS-RPC) 機能を手動で実行できるようになります。しかし、ほとんどの機能は Samba が提供する別のユーティリティーに統合されます。MS-PRC 機能をテストする際は、**rpcclient** のみを使用します。

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 Spool Subsystem (SPOOLSS) の管理。

例16.9 プリンターへのドライバーの割り当て

```
~]# rpcclient server_name -U "DOMAIN\administrator" \
       -c 'setdriver "printer_name" "driver_name"'
Enter DOMAIN\administrators password:
Successfully set printer_name to driver driver_name.
```

- SMB サーバーに関する情報の取得。

例16.10 すべてのファイル共有および共有されたプリンターの一覧表示

```
~]# rpcclient server_name -U "DOMAIN\administrator" -c
'netshareenum'
Enter DOMAIN\administrators 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た動作の実行

例16.11 SMB サーバー上のユーザーの一覧表示

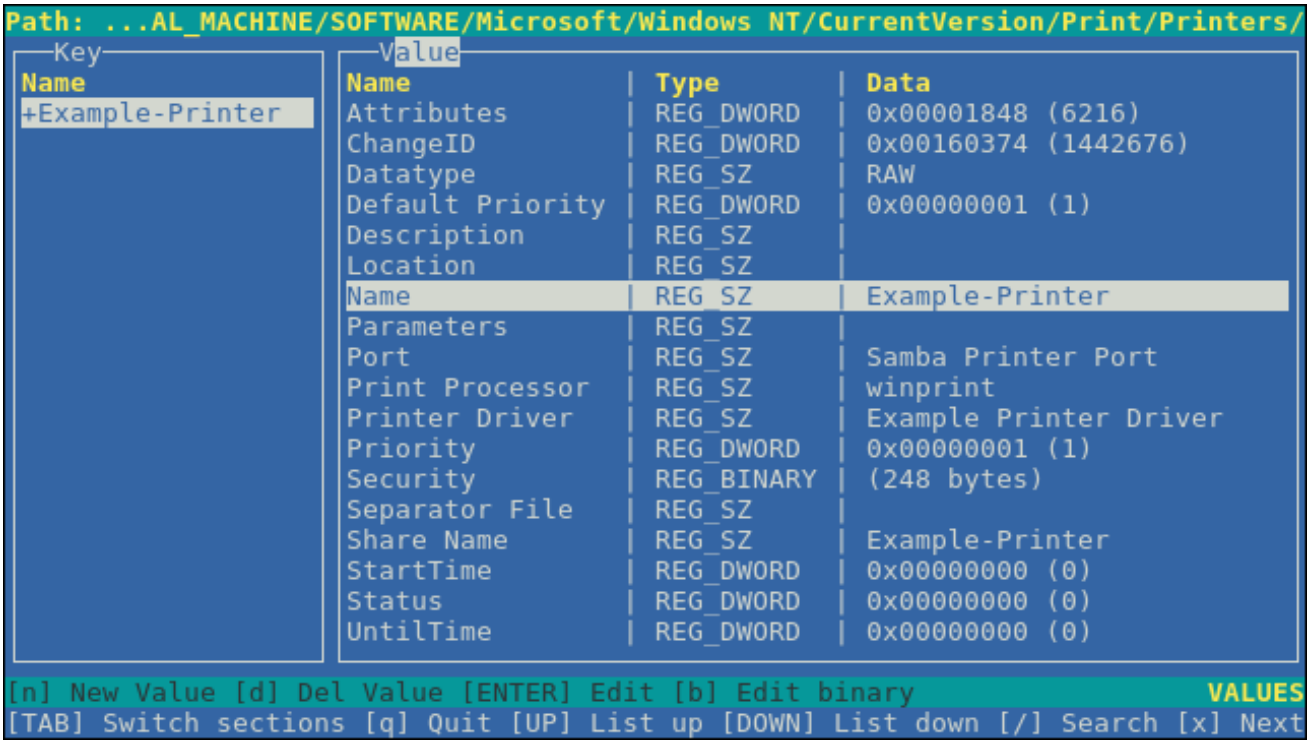
```
~]# rpcclient server_name -U "DOMAIN\administrator" -c
'enumdomusers'
Enter DOMAIN\administrators password:
user:[user1] rid:[0x3e8]
user:[user2] rid:[0x3e9]
```

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にユーザーを一覧表示します。AD DC または NT4 PDC に対してコマンドを実行すると、ドメインユーザーを一覧表示します。

サポート対象のサブコマンドの完全な一覧は、rpcclient(1) の man ページの 『COMMANDS』 セクションを参照してください。

16.1.10.3. `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ています。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開始するには、以下を入力します。

```
~]# samba-regedit
```

以下のキーを使用します。

- カーソルを上下に移動: レジストリーツリーと値の全体を移動
- **Enter**: キーを開くか、または値を編集します。
- **Tab**: **Key** と **Value** の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。

16.1.10.4. smbcacls ユーティリティーの使用

「**smbcacls** を使用した SMB 共有上での ACL の管理」 を参照してください。

16.1.10.5. 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ことで、コマンドラインの FTP クライアントと同様に SMB サーバー上のファイル共有へアクセスできるようになります。たとえば、ファイルの共有へのアップロード、または共有からのダウンロードに使用することができます。

たとえば、**DOMAIN\user** アカウントを使用して ***server*** 上にホストされた ***example*** 共有に対して認証するには、以下を実行します。

```
~]# smbclient -U "DOMAIN\user" //server/example
Enter domain\user's password:
Domain=[SERVER] OS=[Windows 6.1] Server=[Samba 4.6.2]
smb: \>
```

smbclient が正常に共有に接続された後、ユーティリティーはインタラクティブモードに入り、以下のプロンプトを表示します。

```
smb: \>
```

インタラクティブシェル内で利用可能なすべてのコマンドを表示するには、以下を入力します。

```
smb: \> help
```

特定のコマンドのヘルプを表示するには、以下を入力します。

```
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および説明については、smbclient(1) の man ページを参照してください。

16.1.10.5.1. インタラクティブモードでの smbclient の使用

-c パラメーターなしに **smbclient** を使用すると、ユーティリティーはインタラクティブモードに入ります。

以下の手順では、SMB 共有へ接続し、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示しています。

手順16.23 smbclient を使用して SMB 共有からファイルをダウンロード

1. 共有への接続

```
~]# 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. ディレクトリーのファイルを表示します。

```
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. 共有からの接続解除

```
smb: \example\> exit
```

16.1.10.5.2. スクリプトモードでの smbclient の使用

-c commands パラメーターを **smbclient** へパスした場合、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** を使用できるようになります。

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"
```

16.1.10.6. 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**、またはこれらすべてのサービスへコマンドメッセージを送信することができます。これらの制御メッセージは、たとえば、サービスに対して設定を再度読み込むなどの指示を出します。

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
```

詳細および利用可能なコマンドメッセージタイプの一覧については、smbcontrol(1) の man ページを参照してください。

16.1.10.7. 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 データベースのユーザーアカウントおよびパスワードを管理します。

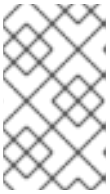
ユーザーとしてコマンドを実行すると、**smbpasswd** はユーザーの Samba パスワードを変更します。以下に例を示します。

```
[user@server ~]$ smbpasswd
New SMB password:
Retype new SMB password:
```

root ユーザーとして **smbpasswd** を実行した場合、ユーティリティーを使用して以下の例のようなことができます。

- 新規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 ユーザーを有効化します。

```
[root@server ~]# smbpasswd -e user_name
Enabled user user_name.
```

- Samba ユーザーを無効化します。

```
[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) の man ページを参照してください。

16.1.10.8. `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レポートします。

- 各 **smbd** デーモンの PID ごとの Samba サーバーへの接続。このレポートには、ユーザー名、プライマリーグループ、SMB プロトコルバージョン、暗号化、および署名情報が含まれています。
- Samba 共有ごとの接続。このレポートには、**smbd** デーモンの PID、接続するマシンの IP、接続が確立した時間のタイムスタンプ、暗号化、および証明情報が含まれます。
- ロックされたファイルの一覧。このレポートエントリーには、便宜的ロック (oplock) のタイプなど、さらなる詳細が含まれます。

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) の man ページを参照してください。

16.1.10.9. smbtar ユーティリティーの使用

smbtar ユーティリティーは、SMB 共有のコンテンツまたはそのサブディレクトリーをバックアップし、**tar** アーカイブにコンテンツを保存します。または、テープデバイスにコンテンツを書き込むことも可能です。

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
```

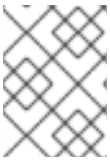
詳細は、smbtar(1) の man ページを参照してください。

16.1.10.10. 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** ファイルの確認」 を参照してください。

16.1.10.11. wbinfo ユーティリティーの使用

wbinfo ユーティリティーはクエリーを実行し、**winbindd** サービスによって作成され、使用された情報を返します。



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、以下のような目的で **wbinfo** を使用することができます。

- ドメインユーザーの一覧表示

```
~]# wbinfo -u
AD\administrator
AD\guest
...
```

- ドメイングループの一覧表示

```
~]# wbinfo -g
AD\domain computers
AD\domain admins
AD\domain users
...
```

- ユーザーの SID の表示

```
~]# 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の表示

```
~]# 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) の man ページを参照してください。

16.1.11. 関連資料

- Red Hat Samba パッケージには、この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 ページが含まれます。たとえば、**/etc/samba/smb.conf** ファイル内に設定可能なすべての設定パラメーターを説明するこのファイルの man ページを表示するには、以下を実行します。

```
~]# 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』』: GlusterFS ボリューム上に保存されたディレクトリーを共有するための Samba および Clustered Trivial Database (CDTB) のセットアップに関する情報を提供します。

- 『『Red Hat Enterprise Linux 6 クラスターの 管理 』』の「『Clustered Samba の設定』」セクションでは、Samba の高可用性インストールのセットアップ方法について説明しています。

  このドキュメントは現在、最新の Red Hat Enterprise Linux バージョンでは利用できません。

- 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たり、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 に標準装備されている優先的な **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い、**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** プロトコル独自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**スイーツからの **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 *アクティブ* か *パッシブ* のモードでデータを要求するかによって変化します。

これらのモードについて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

vsftpd (*The Very Secure FTP Daemon*) は、高速で安定性があり、(より重要なのは) セキュアであるようにゼロから設計されています。**vsftpd** は、多数の接続を効率的かつセキュアに処理できる能力があることから、Red Hat Enterprise Linux に標準装備されている、唯一のスタンドアロン型 **FTP** サーバーです。

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、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することができます。
- **ほとんどのプロセスを chroot jail で実行** — 可能な場合は常に、プロセスは共有ディレクトリーにルートディレクトリーを変更します。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、ディレクトリー **/var/ftp/** がプライマリー共有ディレクトリーの場合、**vsftpd** は **/var/ftp/** を **/** として知られる、新規 root ディレクトリーに再度割り当てます。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、潜在的な悪質ハッカー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します** — 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されます** — **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ません** — 子プロセスとの通信はソケット上で受信され、子プロセスからの情報の有効性は動作を実施する前にチェックされます。
- **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されます** — 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
```

このコマンドで **vsftpd** サービスを停止させた直後に再起動します。これが、**FTP** サーバーの設定ファイルを編集した後に変更を反映させる最も効率的な方法になります。別の方法では、**vsftpd** サービスが実行中の場合にのみ、以下のコマンドを使って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** 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詳細情報については、10章*systemd によるサービス管理* を参照してください。

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** を使っ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、まずすべての関連する **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、デバイスエイリアスの設定、さらにネットワーク設定スクリプトに関する詳細情報は、『Red Hat Enterprise Linux 7 ネットワークガイド』を参照してください。

次に、**FTP** ドメイン用の *DNS* サーバーが適切なマシンを参照するように設定する必要があります。Red Hat Enterprise Linux で使用される **DNS** プロトコル実装である **BIND** およびその設定ファイルについての情報は、『Red Hat Enterprise Linux 7 ネットワークガイド』を参照してください。

vsftpd が異なる **IP** アドレス上の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** サーバーの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** アドレスで置き換えます。このサイトが **IPv6** を使用している場合、代わりに **listen_address6** ディレクティブを使用してください。

複数の設定ファイルが **/etc/vsftpd/** ディレクトリーに格納されると、**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ことで開始可能になります。

```
~]# systemctl start vsftpd@configuration-file-name.service
```

上記のコマンドで、*configuration-file-name* を **vsftpd-site-2** などのリクエストしているサーバーの設定ファイルの一意の名前で置き換えます。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したい場合は、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。これは、vsftpd パッケージで提供されています。この systemd ターゲットにより、個別の **vsftpd** デーモンが **/etc/vsftpd/** ディレクトリー内で利用可能な **vsftpd** 設定ファイルごとに起動します。ターゲットを有効にするには、**root** で以下のコマンドを実行します。

```
~]# systemctl enable 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、systemd サービスマネージャーが **vsftpd** サービスを (設定済みの **vsftpd** サーバーインスタンスとともに) ブート時に起動するように設定します。システムを再起動することなく、サービスをただちに開始するには、**root** で以下のコマンドを実行します。

```
~]# systemctl start vsftpd.target
```

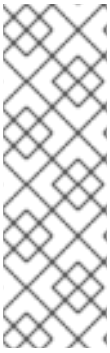
systemd ターゲットを使用してサービスを管理する方法については、「systemd ターゲットでの作業」を参照してください。

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

ユーザー名やパスワード、データを暗号化せずに送信するという **FTP** の元々の不安定な性質に対抗するために、**vsftpd** デーモンは **TLS** プロトコルを使って接続を認証し、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** には深刻なセキュリティー上の脆弱性があるため、新規のバージョン (**TLS**) のみを使用してください。**vsftpd** サーバーに組み込まれているドキュメントや **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、**ssl_enable** ディレクティブが **YES** に設定される場合はデフォルトでサポートされ、使用されます。

TLS サポートを有効にするには、**vsftpd.conf** ファイル内の **ssl_enable** 設定ディレクティブを **YES** に設定します。**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他の **TLS** 関連のディレクティブのデフォルト設定では、適切な **TLS** が提供されます。特にこれに含まれるのは、全接続に **TLS** v1 プロトコルの使用を必須とすることや (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されます) を必須とすることや、非匿名のログインすべて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 を設定

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
```

その設定を変更した後に **vsftpd** サービスを再起動します。

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

vsftpd での **TLS** の使用の微調整についての **TLS** 関連の接続ディレクティブは、vsftpd.conf(5) の man 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** プロセスと共に) **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、必要最小限の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、共有するファイルおよびディレクトリーに **public_content_t** ラベルを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、ファイルをアップロードするためのディレクトリーを設定するには、**public_content_rw_t** ラベルをそのディレクトリーに割り当てる必要があります。さら

に、**allow_ftpd_anon_write** SELinux ブール値オプションを **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** 経由で自分のホームディレクトリーにアクセスできるようにしたい場合 (Red Hat Enterprise Linux 7 ではこれがデフォルト設定)、**ftp_home_dir** のブール値オプションを **1** に設定する必要があります。**vsftpd** がスタンドアロンモードで実行できるようになっていると (これも Red Hat Enterprise Linux 7 でデフォルトで有効になっています)、**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についての詳細情報は、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 ユーザーおよび管理者のガイド』も参照してください。

16.2.3. 関連資料

vsftpd についての詳細情報は、以下のリソースを参照してください。

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的な情報を記載した **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についてのヒントが、また **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は数多くの man ページがあります。以下は、重要な man ページのリストです。

  サーバーアプリケーション

  - 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

  - 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一覧を格納しています。
  - hosts_access(5) — **hosts.allow** および **hosts.deny** の**TCP** ラッパーの設定ファイル内で利用可能なフォーマットとオプションについて説明しています。

  SELinux とのインタラクション

  - 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. オンラインのドキュメント

vsftpd および FTP 全般について

- 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 サイトは、FTP のアクティブモードとパッシブモードの相異点について簡潔に説明しています。

Red Hat Enterprise Linux のドキュメンテーション

- 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』 は、このシステムにおけるネットワークインターフェースやネットワーク、ネットワークサービスの設定および管理に関する情報が説明されています。`hostnamectl` ユーティリティーの概要のほか、これを使ってコマンドラインでホスト名を表示、設定する方法が説明されています。

- 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**、または **OpenShift** などの様々なサービスで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 Security Guide — Red Hat Enterprise Linux 7 の 『セキュリティガイド』 は、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ユーザーおよび管理の役に立ちます。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

- 『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。**vsftpd** が TLS 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

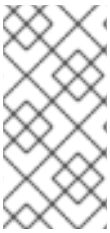
印刷設定 ツールを使うと、プリンター設定、プリンター設定ファイルの管理、印刷スプールディレクトリー、印刷フィルター、プリンタークラスの管理ができます。

このツールはコモンユニックスプリンティングシステム (Common Unix Printing System : CUPS) を基本にしています。CUPS を使用した 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は、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 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は許可しません。『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)』 で説明されている脆弱性により、Red Hat ではセキュリティー保護のためにこれに依存するのではなく、**stunnel** を使用してセキュアなトンネルを提供し、**SSLv3** を無効にすることを推奨します。**stunnel** の使用については、Red Hat Enterprise Linux 7 セキュリティガイド 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については、「X11 転送」 で説明されているように **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 ブラウザでアプリケーションにアクセスするには http://localhost:631/ を開いてください。CUPS マニュアルについてはこの Web サイトの **ホーム** タブのリンクを参照してください。

16.3.1. 印刷設定の設定ツールの起動

印刷設定 設定ツールを使うと、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することができます。CUPS を直接使用することも可能です (CUPS web アプリケーションにアクセスするには http://localhost:631/ をクリックします)。

コマンドラインから **印刷設定** 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。または GNOME デスクトップを使用している場合、**Super** キーを押してアクティビティーの概要に入り、**Print Settings (印刷設定)** 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、通常はWindows または Command キーであり、**スペースバー** 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。

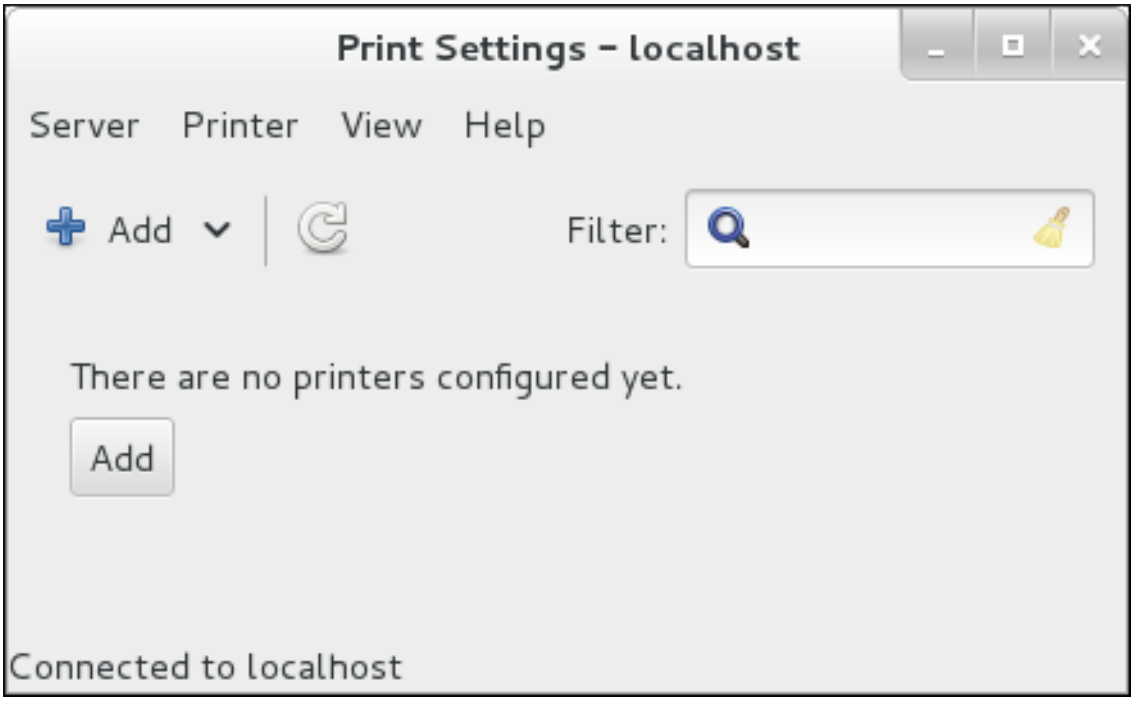


図16.1 印刷設定ウィンドウ

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 に接続したローカルプリンターを設定する場合は、プリンターは自動的に検出され、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. **サーバー** → **新規** → **プリンター** の順にクリックします。
3. **認証** 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、左側の一覧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** または **LPT #1** など)。
3. 右側で、接続プロパティーを入力します。

   その他 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

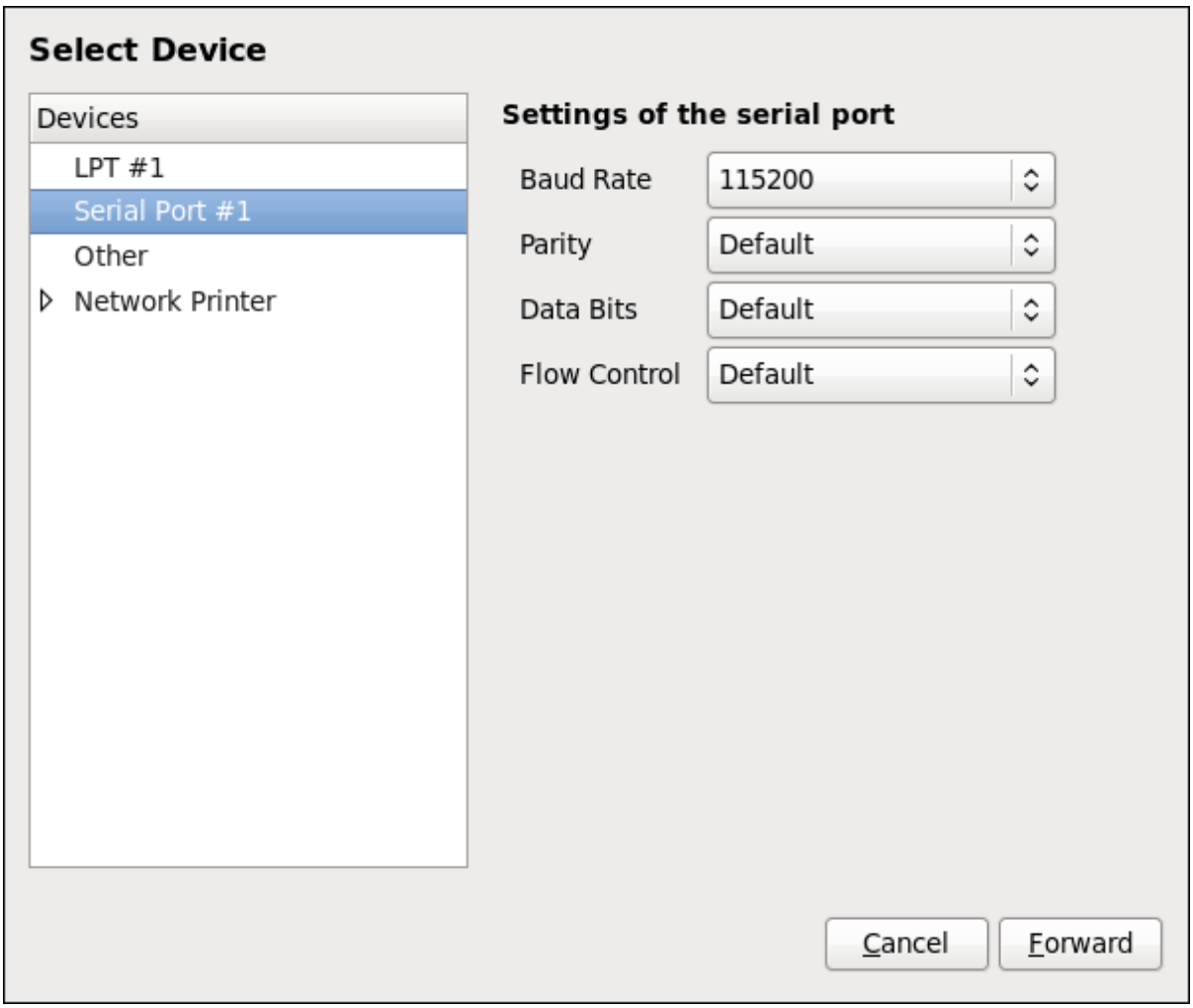


図16.2 ローカルプリンターの追加

4. **進む (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。詳細は 「プリンターモデルの選択と完了」 を参照して下さい。

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. 左側の一覧で **ネットワークプリンター** → **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。

ホスト名

プリンターホスト名または **IP** アドレス。

ポート番号

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**)

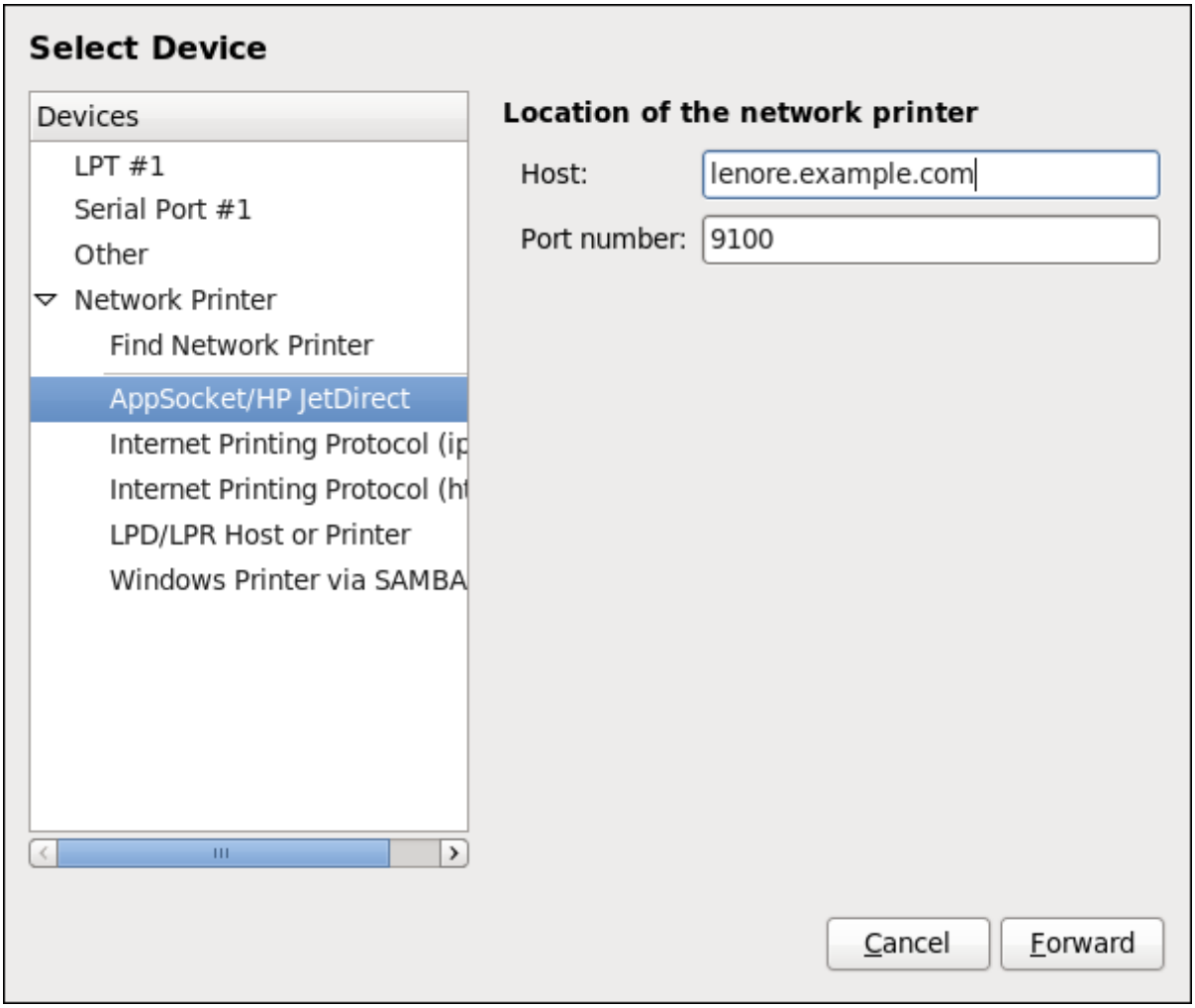


図16.3 JetDirect プリンターの追加

4. **進む (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。詳細は 「プリンターモデルの選択と完了」 を参照して下さい。

16.3.5. IPP プリンターの追加

IPP プリンターは同じ TCP/IP ネットワーク上の異なるシステムにつながっ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、または単に **IPP** を使用するよう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** 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. 左側のデバイスの一覧で、**ネットワークプリンター**、**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。

   ホスト

   IPP プリンターのホスト名。

   キュー

   新規のキューに与えるキューの名前です(このボックスが空白のままであると、デバイスノードを基にした名前が使用されます)。

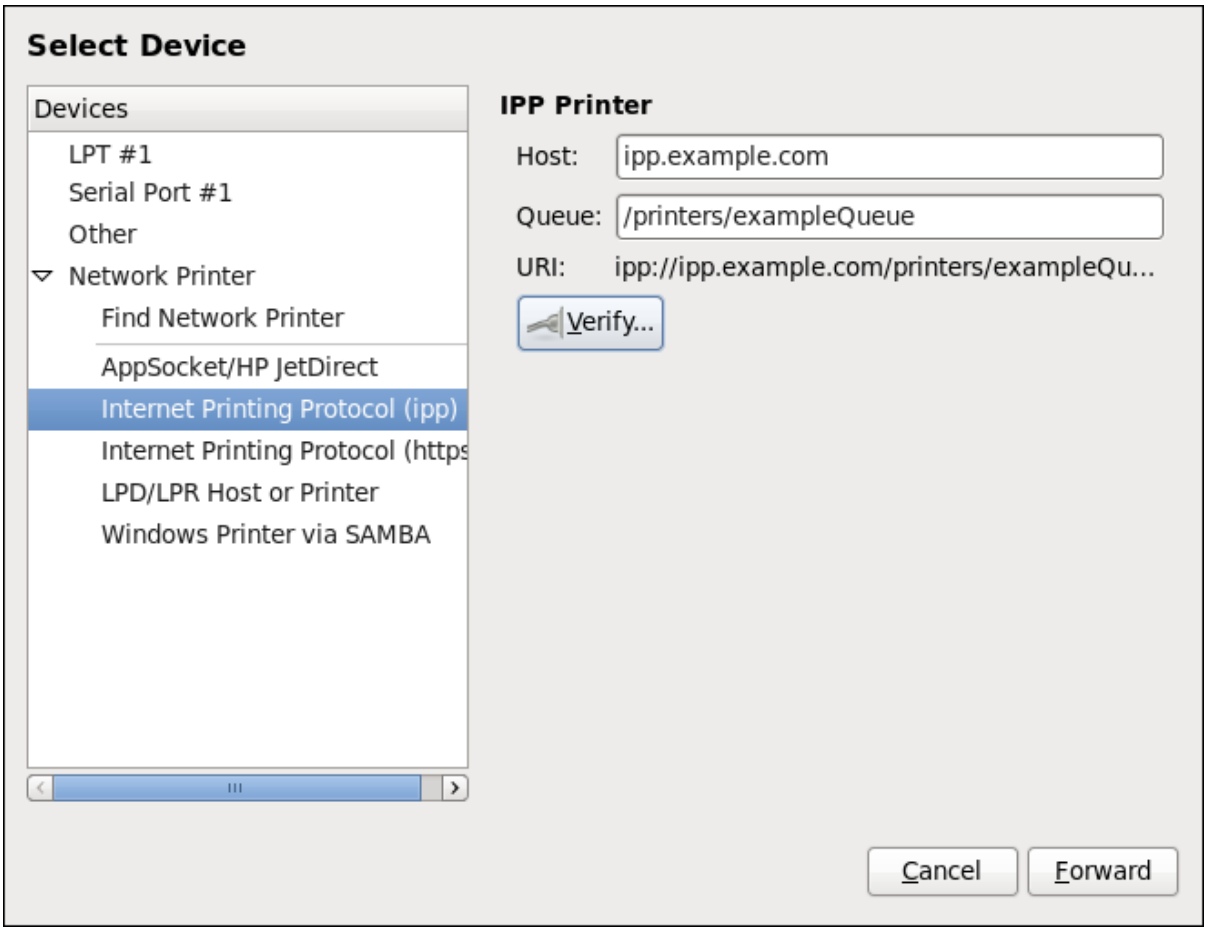


図16.4 IPP プリンターの追加

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。詳細は 「プリンターモデルの選択と完了」 を参照して下さい。

16.3.6. LPD/LPR Host or Printer の追加

以下の手順に従って LPD/LPR host or printer 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. 左のデバイス一覧から、**ネットワークプリンター** → **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。

   ホスト

   LPD/LPR printer or host のホスト名

   オプションで **Probe** をクリックすると、LPD ホスト上のキューを検索できます。

   キュー

   新規のキューに与えるキューの名前です(このボックスが空白のままであると、デバイスノードを基にした名前が使用されます)。

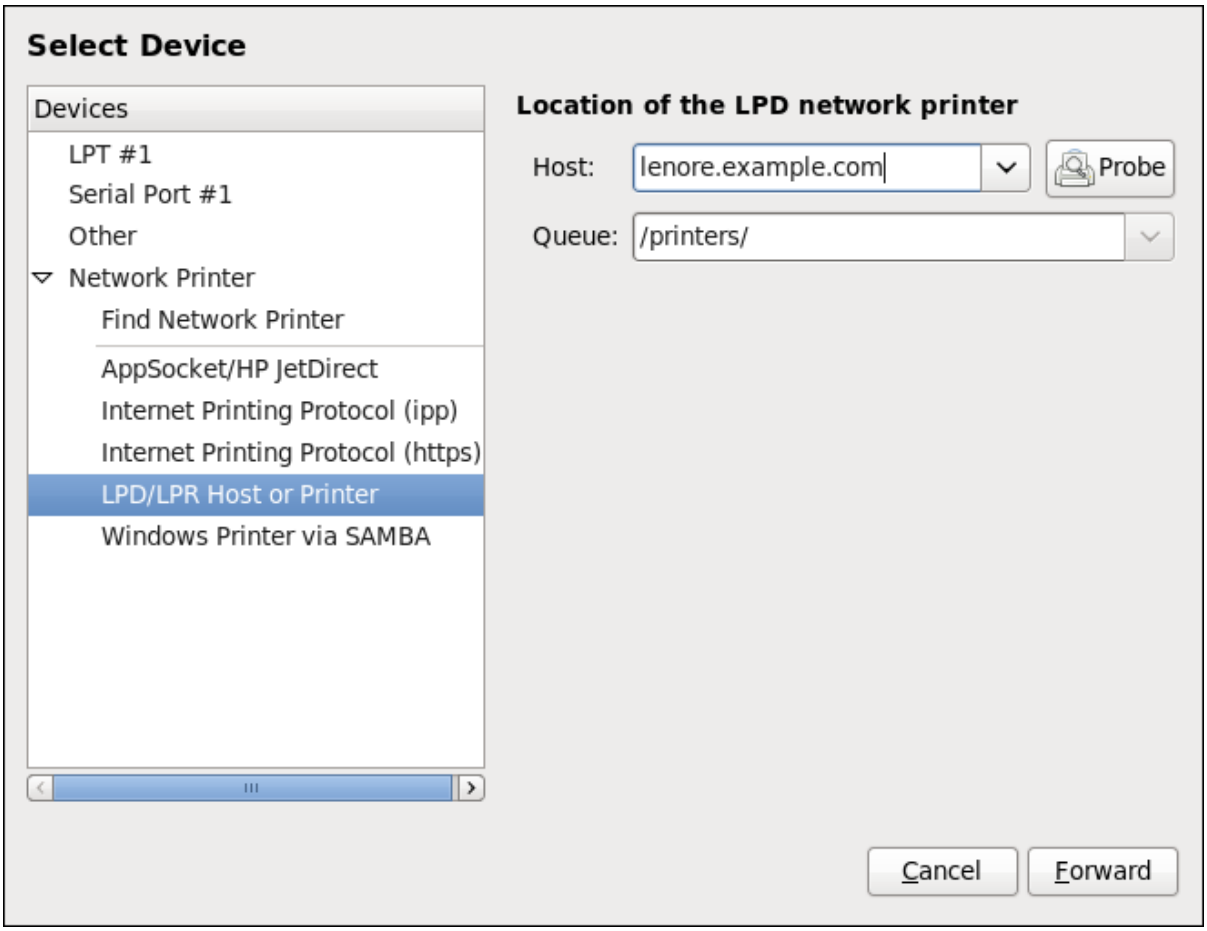
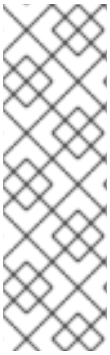


図16.5 LPD/LPR プリンターの追加

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。詳細は 「プリンターモデルの選択と完了」 を参照して下さい。

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。これを実行するには、**root** で以下を実行します。

```
yum install samba-client
```

Yum を使用したパッケージのインストールについての詳細は、「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. 左側の一覧で、**ネットワークプリンター** → **SAMBA 経由の Windows プリンター** 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。

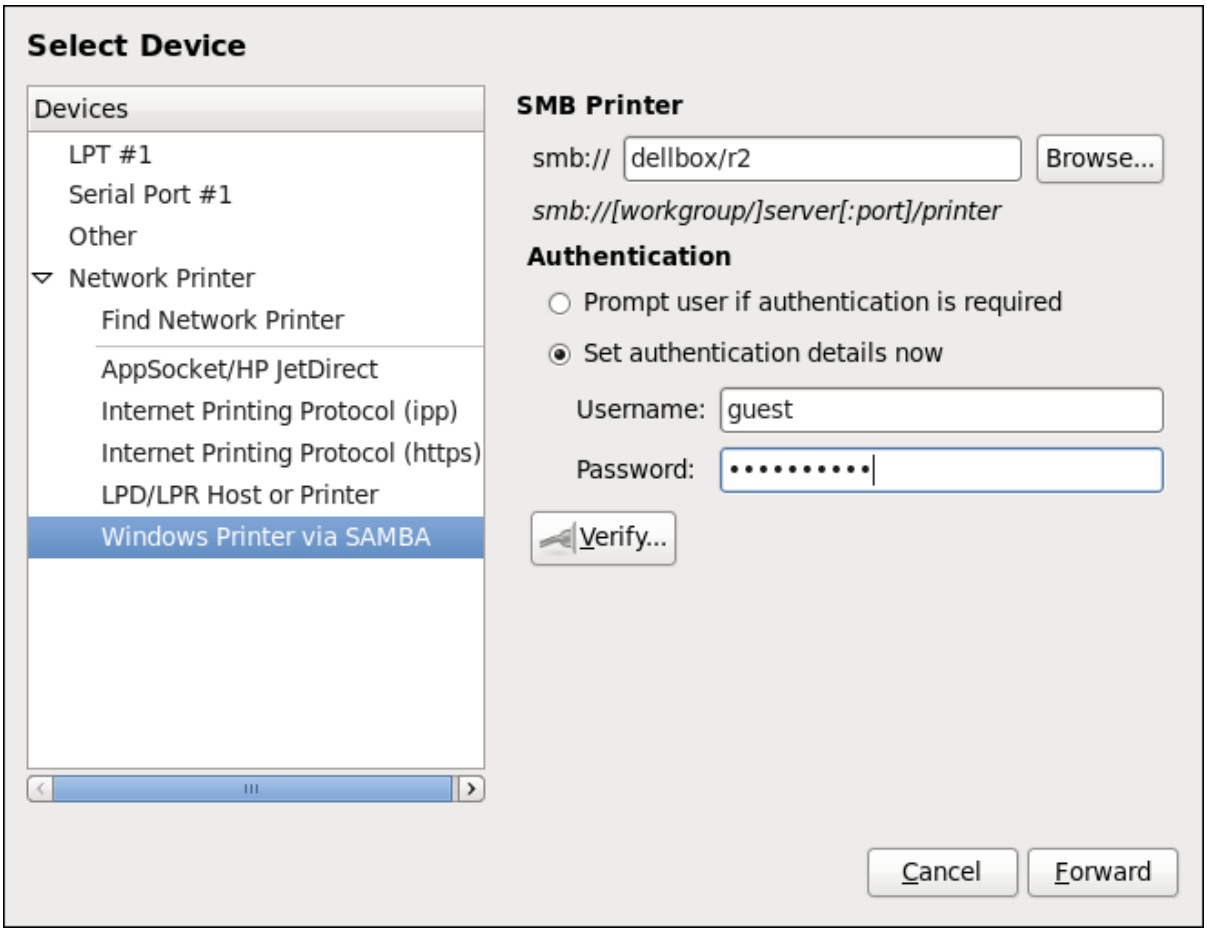


図16.6 SMB プリンターの追加

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どちらかを選択します。

- **認証が必要は場合にはユーザーに催促する (Prompt user if authentication is required)** を選択すると、文書を印刷する時にはユーザーはユーザー名とパスワードを提供することになります。

- **認証の詳細を今設定します (Set authentication details now)** を選択すると、今認証情報を提供するため後では入力が不必要となります。**ユーザー名** フィールドで、ユーザー名を入力してプリンターにアクセスします。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. **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** (必要な場合は) 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プリンターサーバーに **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お勧めします。

Samba プリント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. **進む (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。詳細は 「プリンターモデルの選択と完了」 を参照して下さい。

16.3.8. プリンターモデルの選択と完了

正しく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することができます。

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. 自動的にドライバーが検知され、ウィンドウが表示されます。以下のオプションのうちいずれかを選択します。

   - **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** の一覧からご使用のプリンターの選択した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一覧にない場合は、**Generic** を選択してください。

- **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デスクリプション* (PPD) を使用します。PPD ファイルは通常製造元が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、オプション詳細の下にあるブラウザバーを使用して、PPD ファイルを選択できます。

- **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)** — 製造元とご使用の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。

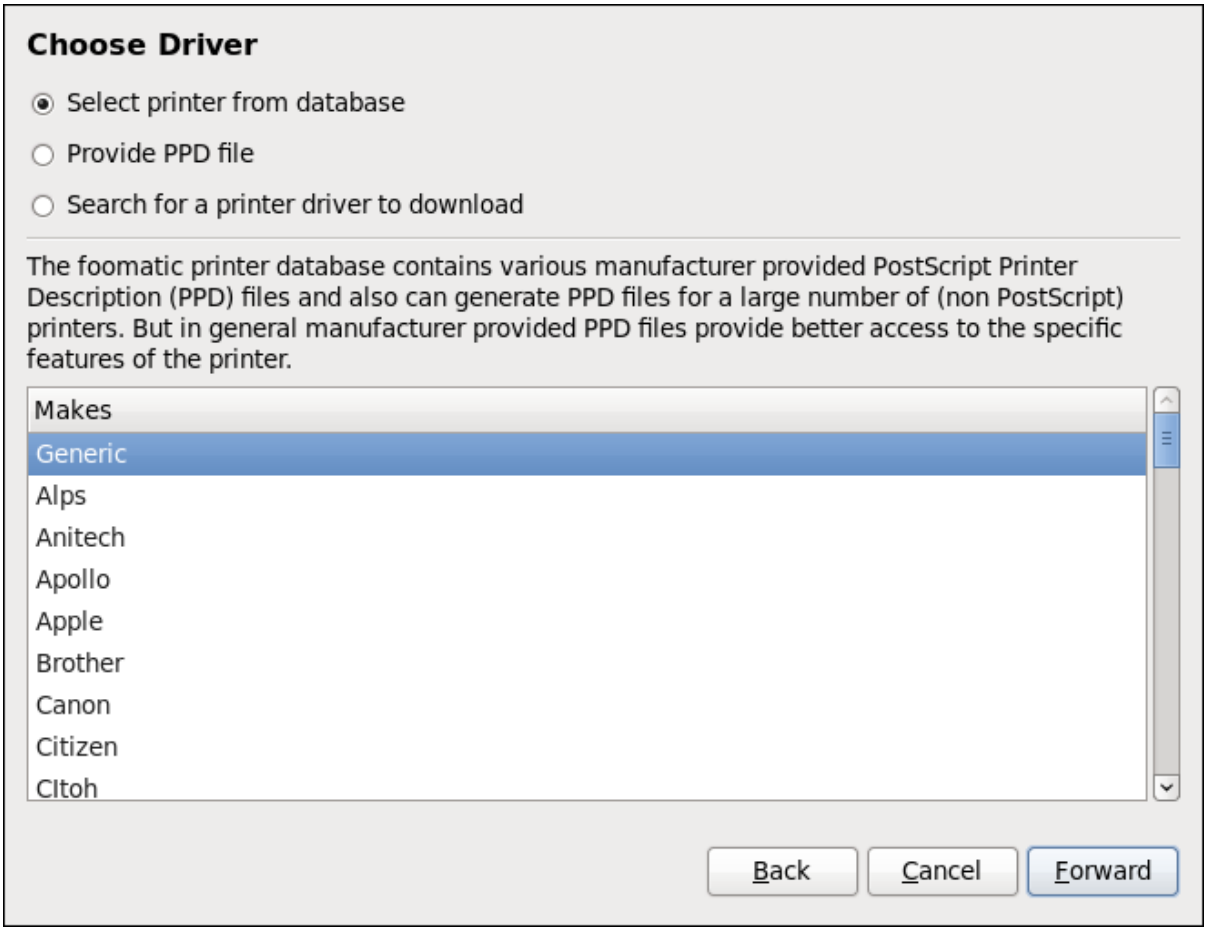


図16.7 プリンターブランドの選択

2. 以前に選択したものにより、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ぶことも可能です。印刷ドライバーは、プリンターが理解できる形式で印刷したいデータを処理します。ローカルプリンターは直接ご使用のコンピュータにつながっているため、プリンターに送られるデータを処理するプリンタードライバーが必要になります。

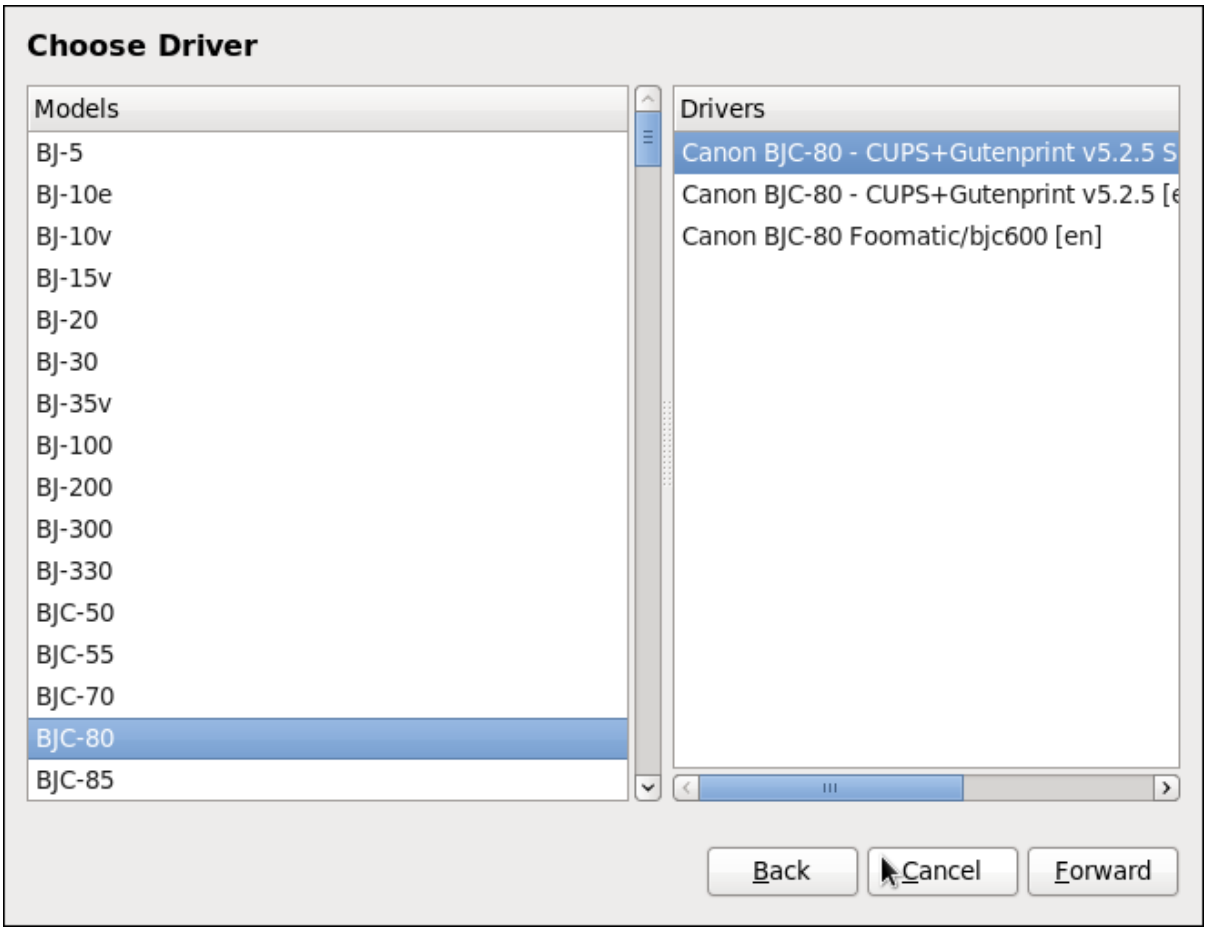


図16.8 プリンターモデルの選択

5. **進む (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、ダッシュ (-)、アンダースコア (_) を含むことができますが、スペースは含むことが **できません**。また、**説明 (Description)** と **場所 (Location)** フィールドに詳細情報を追加することも可能です。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。

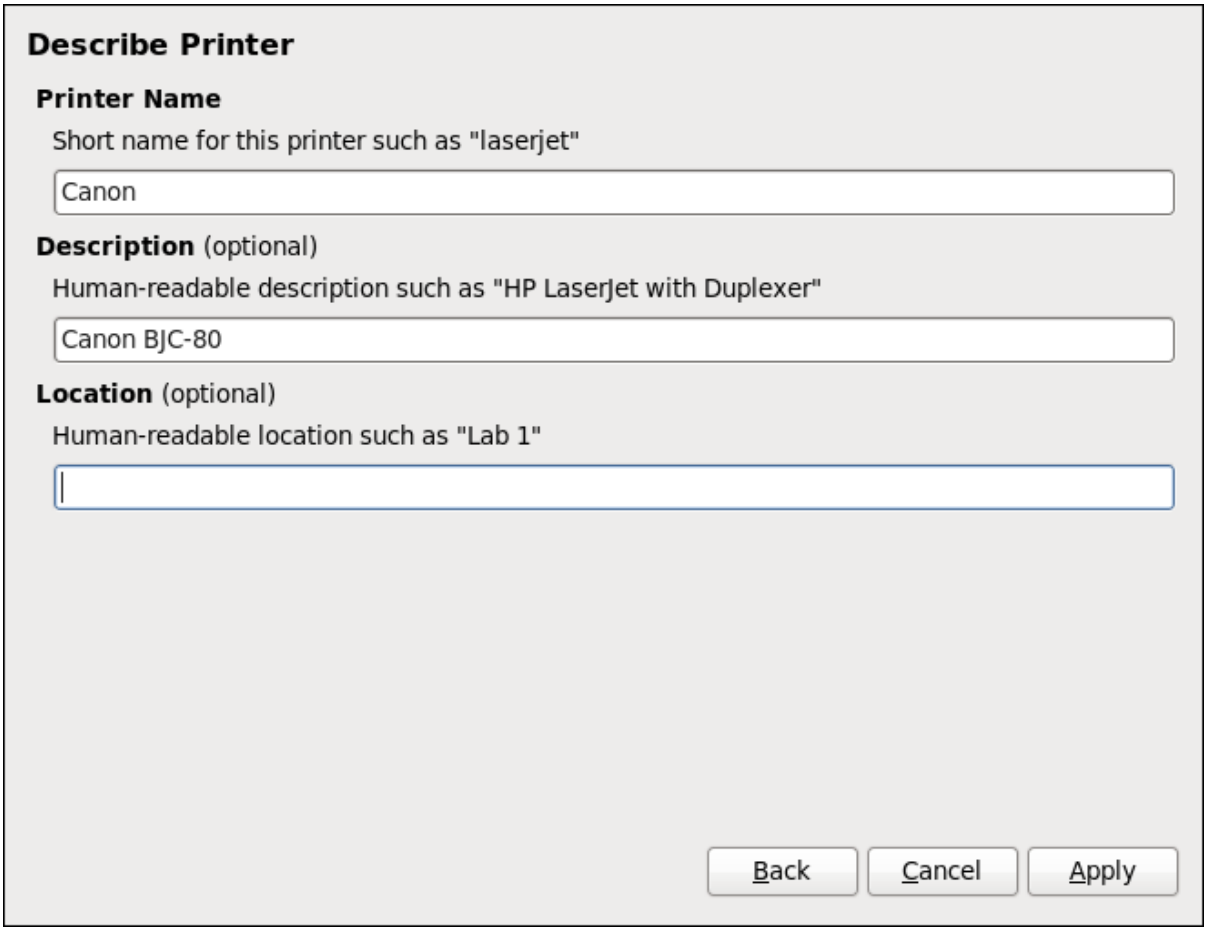


図16.9 プリンターの設定

7. 設定が正しければ、ご使用のプリンター設定を確認して、印刷キューを追加するために **適用 (Apply)** をクリックします。**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」 を参照して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 (Properties)** をクリックします。

2. プロパティーのウィンドウで、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、**プリンター → 削除** 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、プリンターの一覧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. 設定のページ

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** 一覧で該当する名前で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** ページを表示させます。

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。

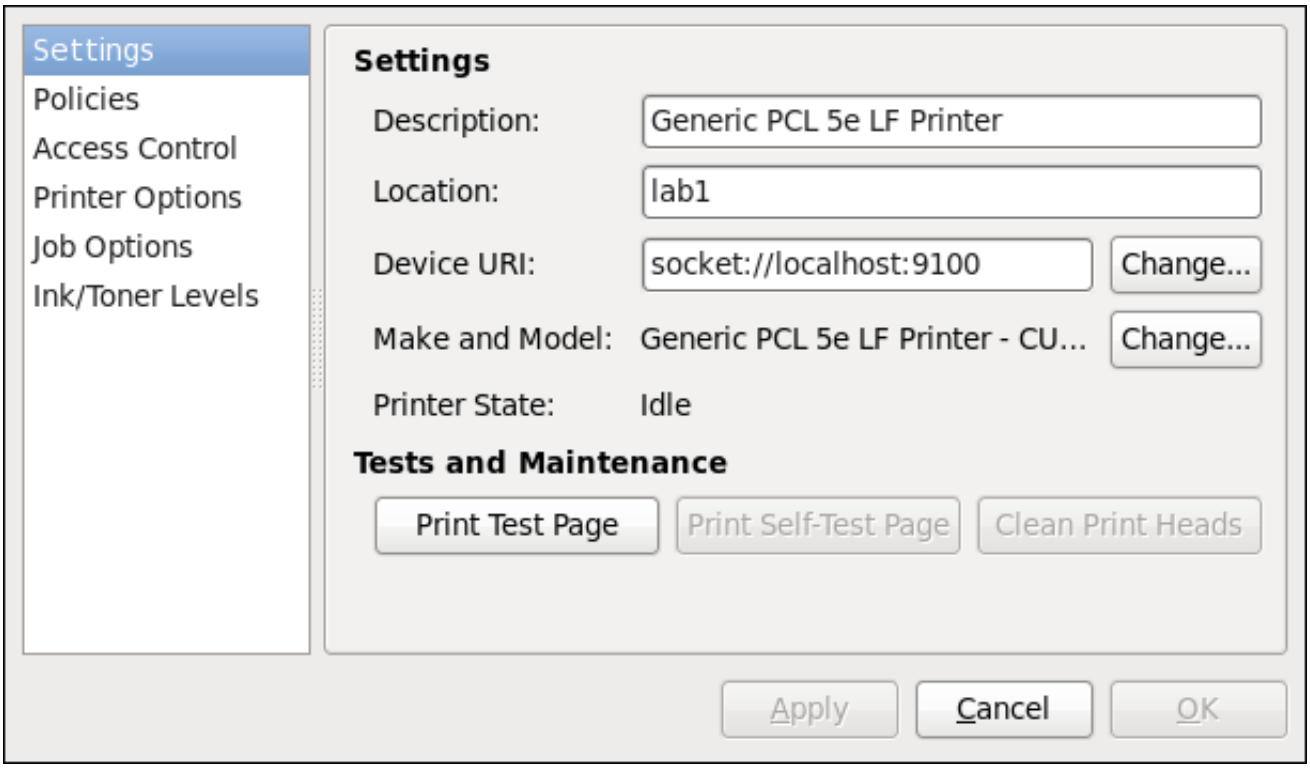


図16.10 設定のページ

16.3.10.2. ポリシーのページ

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)** を設定することができます (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、プリンターを共有すると印を付けることができます。プリンターを共有にすると、ネットワーク上で公開されているユーザーは使用できます。プリンターの共有機能を有効にするには、**サーバー → 設定** 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。

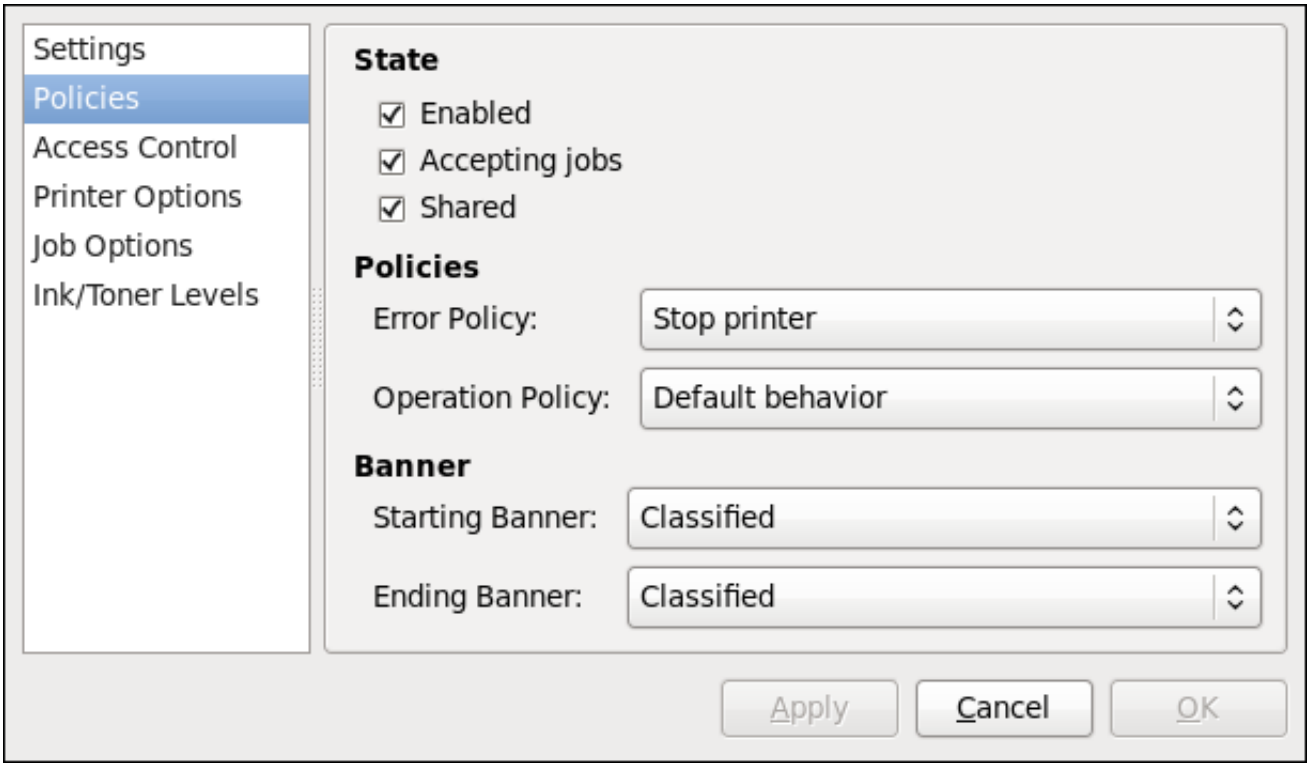


図16.11 ポリシーのページ

ファイアウォールがポート **631** への受信 **TCP** 接続を許可していることを確認してください。これは Network Printing Server (**IPP**) プロトコル用のポートです。Red Hat Enterprise Linux 7 のファイアウォールで **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。

手順16.24 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** と入力してから **Enter** を押します。**ファイアウォールの設定** 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っ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
   ```

   ファイアウォールの設定 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、**Configuration** 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 （ゾーン）** 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ー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 (インターフェース)** タブには、ゾーンに割り当てられたインターフェースが表示されます。

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** ツールを閉じます。

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 セキュリティガイド 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。

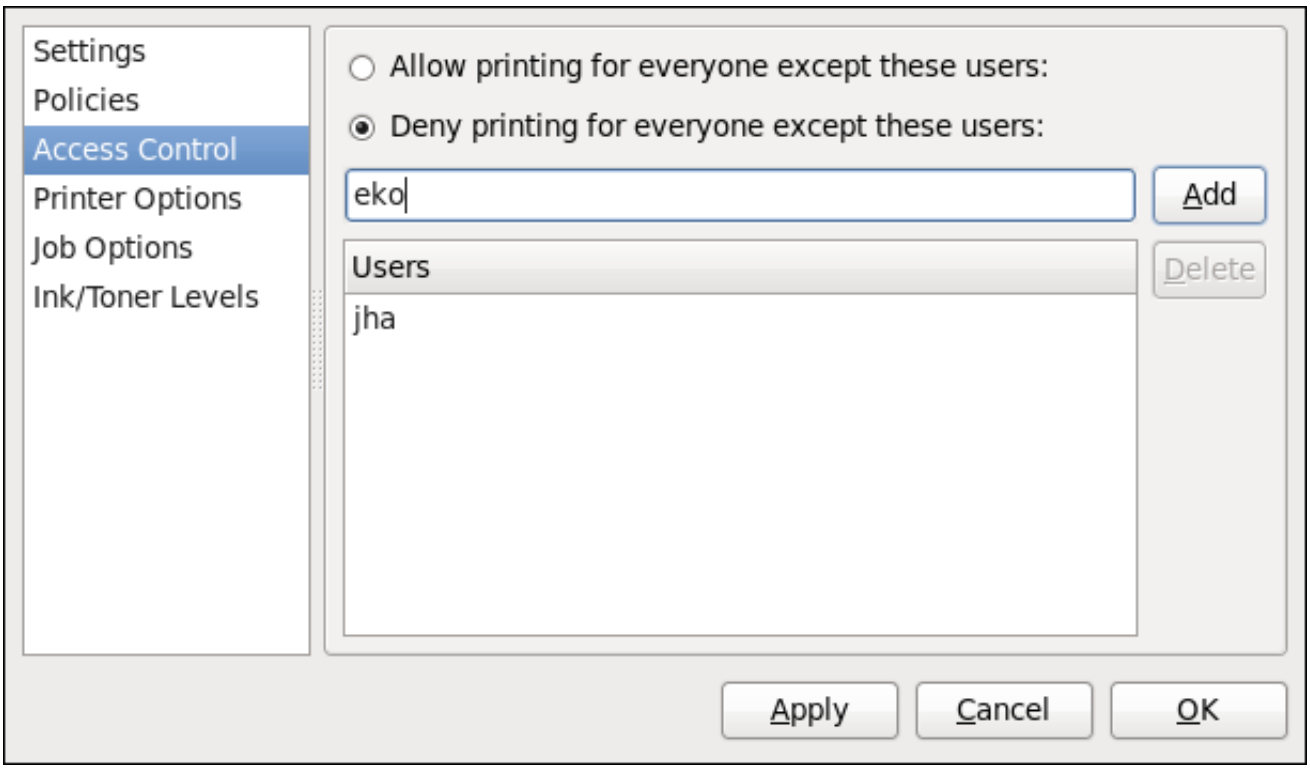


図16.12 アクセス制御のページ

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。

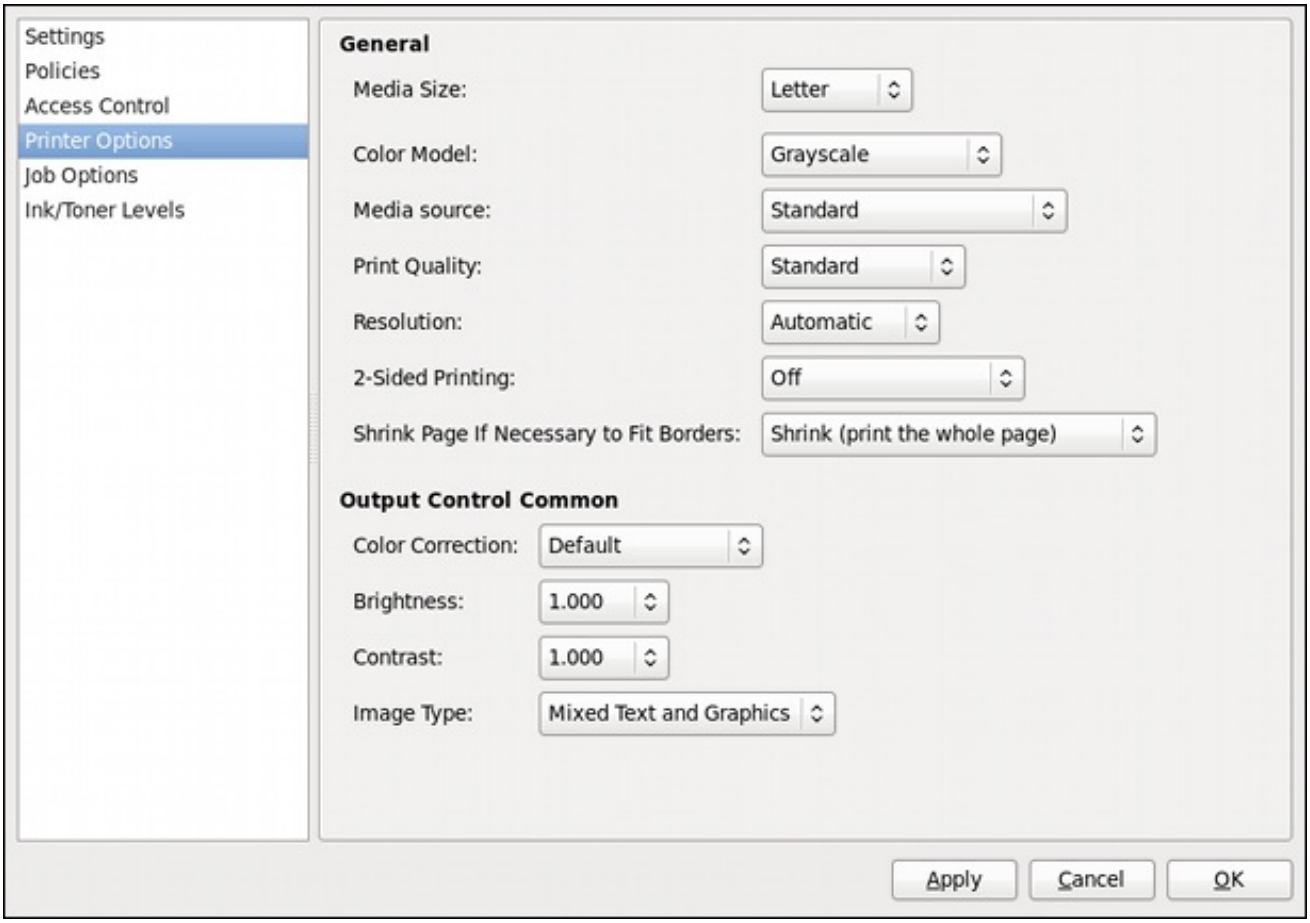


図16.13 プリンターオプションのページ

16.3.10.2.4. 依頼のオプションのページ

依頼のオプション (Job Options) ページでは、プリンターのジョブのオプションを詳細設定できます。左側の **依頼のオプション** 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する。この機能により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、カスタムの依頼のオプションを適用します。

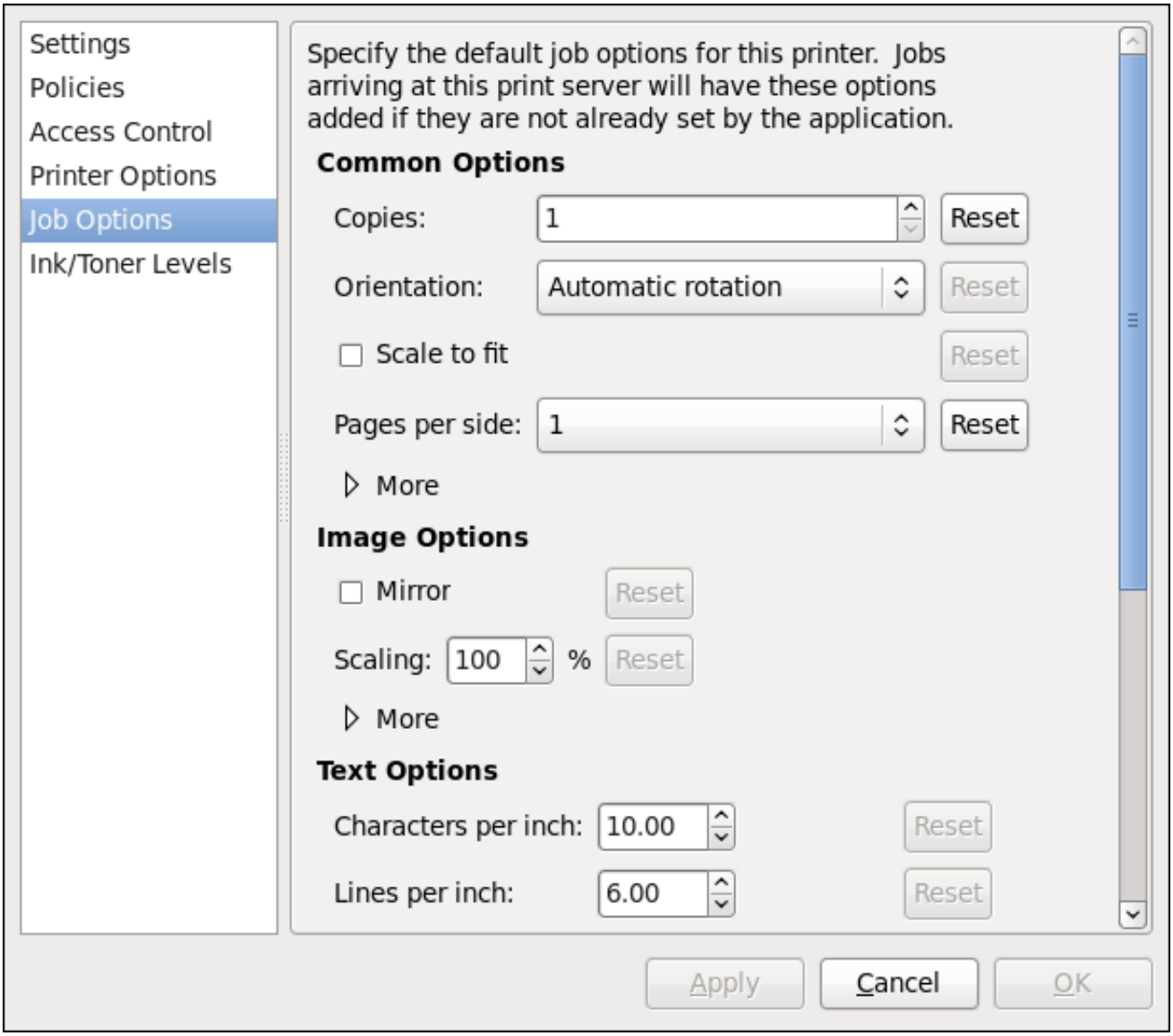


図16.14 依頼のオプションのページ

16.3.10.2.5. インク/トナーのレベルのページ

インク/トナーのレベル (Ink/Toner Levels) ページには該当する場合はトナーの状態の詳細、及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 **インク/トナーレベル** のラベルをクリックすると、ページが表示されます。

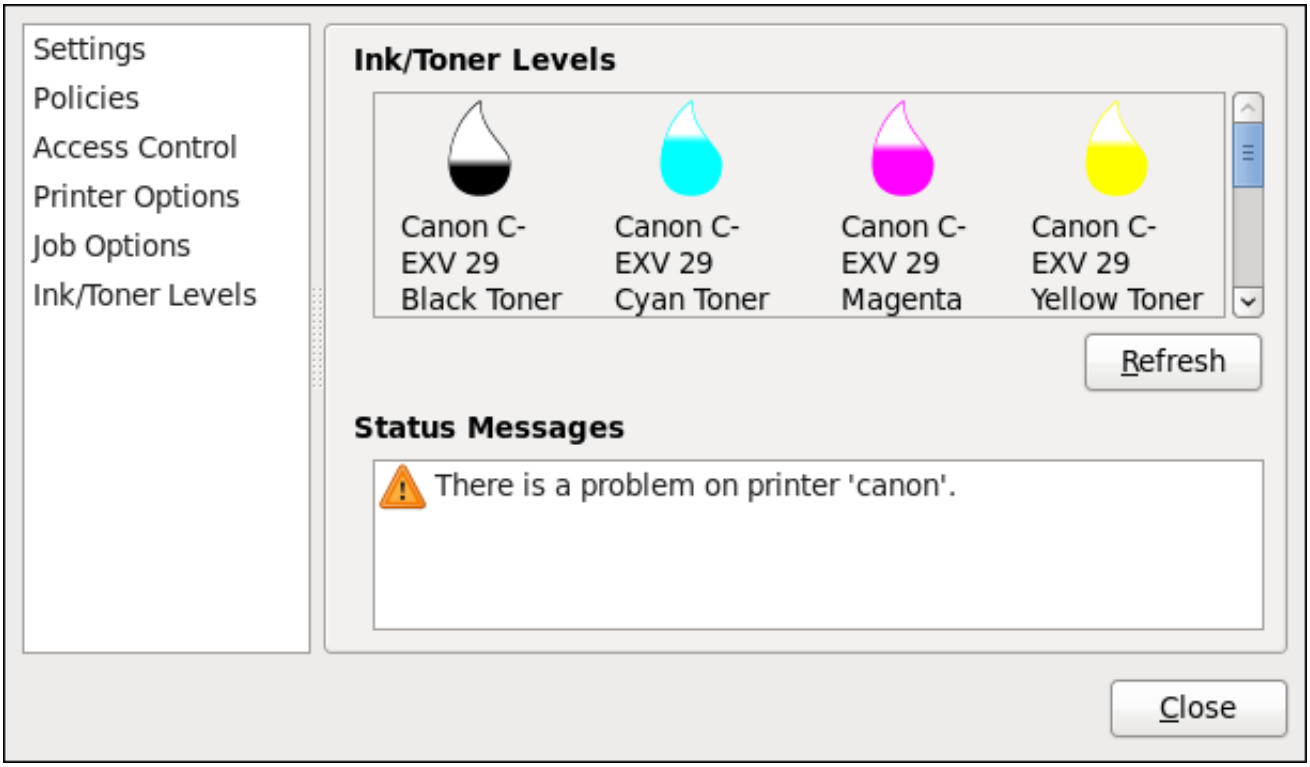


図16.15 インク/トナーのレベルのページ

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または **GIMP** からの画像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一覧で、各印刷要求に関する情報、例えば印刷要求の状態、ジョブ番号などを表示します。

印刷が処理されている間、**プリンター状態**のアイコンがパネルの **通知スペース** に表示されます。その**プリンター状態**をクリックすると、図16.16「GNOME 印刷の状態」 に似た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。

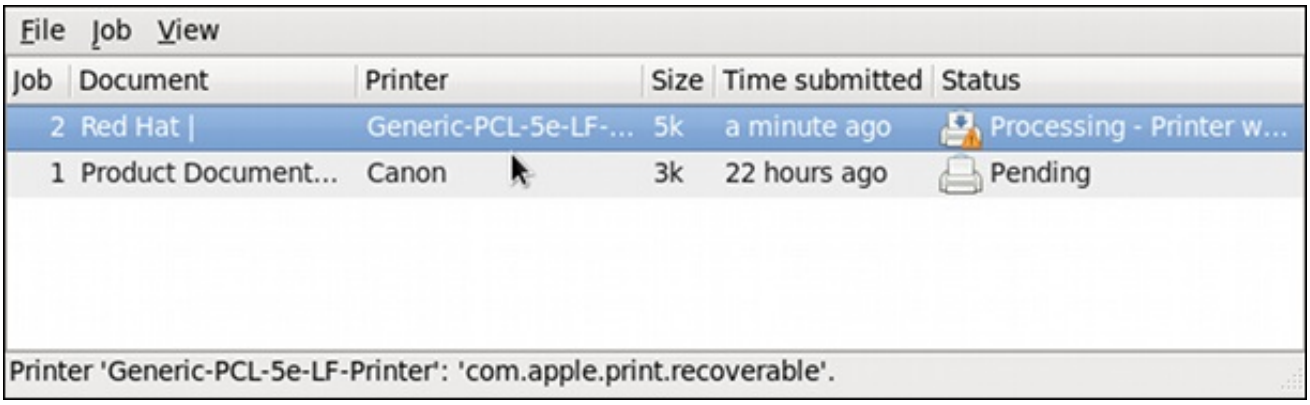


図16.16 GNOME 印刷の状態

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 印刷の状態 (GNOME Print Status)** でジョブを選択し、**ジョブ (Job)** 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一覧を表示させるには、`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したい場合は、**`lpstat -o`** コマンドを使って依頼したジョブ番号を見つけます。その後 **`cancel`** ***ジョブ番号*** コマンドを使用します。たとえば、**`cancel 60`** だと 例16.15「**`lpstat -o`** 出力の例」 の印刷ジョブを取り消します。他のユーザーによって開始された印刷ジョブを **`cancel`** コマンドでキャンセルすることはできません。ただし、**`cancel -U root`** ***`job_number`*** コマンドを使用すると、強制的にそうしたジョブを削除することは可能です。そうしたキャンセルを防ぐには、プリンターの操作ポリシーを **認証済** に変更することで **`root`** 認証を強制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. 関連資料

Red Hat Enterprise Linux の印刷に関する詳細は、以下のリソースを参照して下さい。

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。

オンラインのドキュメント

- 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。

---

[2] 本セクションの一部は、Samba Wiki に出版された 『idmap config autorid』 ドキュメントから掲載しています。ライセンス: CC BY 4.0。著者および寄稿者: Wiki ページの 履歴 タブを参照してください。

[3] 本セクションの一部は、Samba Wiki に出版された 『Setting up a Share Using POSIX ACLs (POSIX ACL を使用した共有のセットアップ)』 ドキュメントから掲載しています。ライセンス: CC BY 4.0。著者および寄稿者: Wiki ページの 履歴 タブを参照してください。

[4] 本セクションの一部は、Samba Wiki に出版された 『Setting up a Share Using Windows ACLs (Windows ACL を使用した共有のセットアップ)』 ドキュメントから掲載しています。ライセンス: CC BY 4.0。著者および寄稿者: Wiki ページの 履歴 タブを参照してください。

[5] 本セクションの一部は、Samba Wiki に出版された 『Setting up Samba as a Print Server (Samba をプリントサーバーとしてセットアップ)』 ドキュメントから掲載しています。ライセンス: CC BY 4.0。著者および寄稿者: Wiki ページの 履歴 タブを参照してください。

[6] 本セクションの一部は、Samba Wiki に出版された 『Setting up Automatic Printer Driver Downloads for Windows Clients (Windows クライアント向け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)』 ドキュメントから掲載しています。ライセンス: CC BY 4.0。著者および寄稿者: Wiki ページの 履歴 タブを参照してください。

[7] 本セクションの一部は、Samba Wiki に出版された 『Performance Tuning (パフォーマンスのチューニング)』 ドキュメントから掲載しています。ライセンス: CC BY 4.0。著者および寄稿者: Wiki ページの 履歴 タブを参照してください。

第17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、正確なタイムスタンプがパケットとログで必要になります。Linux システムでは、**NTP** プロトコルがユーザースペースで実行しているデーモンにより実装されます。

ユーザースペースのデーモンは、カーネルで実行され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様々なクロックソースを使用して時間を維持しています。一般に使用されるのは *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 レジスタです。非常に高速で高い分解能を持ち、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** と **chronyd** 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。

17.1. CHRONY スイートの概要

Chrony は、ネットワークタイムプロトコル (NTP) の実装です。**Chrony** を使うと以下のことが行えます。

- 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マニュアルの時間入力と同期する、
- 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) といった様々な条件下や、継続的に実行されない、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、ユーザースペースで実行されるデーモンの **chronyd** と、 **chronyd** のパフォーマンスを監視したり様々なオペレーティングパラメーターをその実行中に変更したりすることに使用できるコマンドラインプログラム、**chronyc** から構成されています。

17.1.1. ntpd と chronyd の違い

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、落ち着くまでに長時間かかる場合があります。

- デフォルトの設定では、**chronyd** は、実行中の他のプログラムを混乱させないようにするため、システムの起動時にクロックを同期した後は時間をステップしません。 **ntpd** も時間をステップしないように設定することは可能ですが、そうするとクロックの調整に異なる方法を用いなければならず、それによってクロックの精度にマイナスの影響が出るというデメリットがあります。

- **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。

- **chronyd** の方が小型でメモリ使用量が少なく、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。**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、この予測を使っ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ってシステムの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、一般的なサーバーやクライアントモードと比べて精度と安全性がもともと低いため、通常は避けることが推奨されます。

- **ntpd** は、公開鍵暗号でサーバーの承認を行う Autokey プロトコル (『RFC 5906』) に対応しています。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のでご注意ください。

- **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 (SHM) または Unix ドメインソケット (SOCK) を使っ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。

17.1.2. 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っ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上の問題があるため、使用を避けることが推奨されます。**Autokey** ではなく、対称鍵を使って認証を行ってください。この方法は、**chronyd** と **ntpd** の両方が対応しています。**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。

17.2. CHRONY の概要および設定

17.2.1. chronyd の概要

chrony のデーモンである **chronyd** は、コマンドラインユーティリティーの **chronyc** を使って監視と管理を行います。このユーティリティーは、**chronyd** の現在の状態にクエリを実行しその設定に変更を加えるための、多数のコマンドの入力を可能にするコマンドプロンプトを提供しています。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を設けることが推奨されます。

17.2.2. chronyc の概要

chrony のデーモンである **chronyd** は、コマンドラインユーティリティーを使って管理します。このユーティリティーは、**chronyd** の現状にクエリを実行しその設定に変更を加える、多数のコマンドの入力を可能にするコマンドプロンプトを提供しています。デフォルトの設定では、**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、制限を付けることが推奨されます。

17.2.3. chrony 設定コマンドの概要

chronyd のデフォルトの設定ファイルは **/etc/chrony.conf** です。**-f** オプションは、代替の設定ファイルのパスを指定するときに使用します。その他のオプションについては **chronyd** の man ページをご覧ください。使用できるディレクティブの一覧は『http://chrony.tuxfamily.org/manual.html#Configuration-file』 をご覧ください。

以下は、**chronyd** の設定オプションの一例です。

コメント

コメントの前には、#、%、;、または ! を付けます。

allow

オプションとして、**NTP** サーバーとして動作しているマシンへの接続が許可される **NTP** の接続元となるホスト、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:

```
1. allow 192.0.2.0/24
```

特定のネットワークへのアクセスを許可するときにこのコマンドを使用します。

2. ```
   allow 2001:0db8:85a3::8a2e:0370:7334
   ```

   IPv6 へのアクセスを許可するときにこのコマンドを使用します。

UDP ポート番号 123 は、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 を永続的に開きたいときは、**--permanent** オプションを使用します。

```
~]#  firewall-cmd --permanent --zone=public --add-port=123/udp
```

cmdallow

これは **allow** ディレクティブ (**allow** セクションを参照) と似ていますが、特定のサブネットやホストに (**NTP** クライアントアクセスではなく) 制御アクセスを許可します。(「制御アクセス」 とは、**chronyc** がこれらのホスト上で実行可能であり、このコンピューター上で **chronyd** に正常に接続できることを意味します。) 構文は同じです。また、 **cmdallow all** ディレクティブと動作が似通っている **cmddeny all** ディレクティブもあります。

dumpdir

chronyd の再起動にまたがる測定履歴を保存するディレクトリーへのパスです (この実行中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)。この機能が (設定ファイルの **dumponexit** コマンド、または **chronyc dump** コマンド経由で) 使用される場合は、**dumpdir** コマンドを使って測定履歴が保存され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常に記録された時間ソースの測定履歴を **chronyd** が保存すべきということを示しています (上記の **dumpdir** コマンドを参照)。

hwtimestamp

hwtimestamp ディレクティブは、ハードウェアタイムスタンプの非常に精度の高い同期を可能にします。詳細は **chrony.conf(5)** のマニュアルページをご覧ください。

local

local キーワードは、**chronyd** に現行の同期ソースがない場合でも、(クライアントがポーリングしているという観点で) 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、分離されたネットワーク上の 「master」 コンピューターで使用され、ここではいくつかのコンピューターが相互に同期する必要があり、「master」 は手動入力でほぼリアルタイムと一致します。

コマンド例:

```
local stratum 10
```

10 という大きな値が示しているのは、参照しているクロックからのホップ数が非常に多いので時間の信頼性がかなり低い、ということです。参照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のあ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、10 のように高い値を **local** コマンドに選ぶと、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乱することを防ぎます。

log

log コマンドは、特定情報がログ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** 測定と関連情報を **measurements.log** と呼ばれるファイルにログ記録します。

statistics

このオプションは、回帰処理についての情報を **statistics.log** と呼ばれるファイルにログ記録します。

tracking

このオプションは、システムが進むまたは遅れるレートの予測に対する変更、およびなされたスルーを **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ついての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参照クロックの測定を **refclocks.log** と呼ばれる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。コマンドの例は以下のようになります。

```
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。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、システムクロックが修正されるまでのこのスルーイングプロセス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** にシステムクロックを強制的にステップさせます。ただし、**chronyd** が起動した後に、特定の制限値を越える (負の数は制限値の無効化に使用されます) クロックの更新が行われなかった場合に限ります。**initstepslew** ディレクティブは **NTP** のソースのみに対応しているので、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、調整幅が 1000 秒よりも大きければシステムクロックは更新されますが、これは最初の 10 回のクロック更新のみです。

maxchange

このディレクティブは、クロック更新で修正される片寄りの最大値を設定します。指定された回数の更新の後でのみチェックが実行されることで、システムクロックの初回の調整が大幅なものにできます。指定された最大値よりも大きな片寄りが発生した場合、これは指定された回数無視され、その後に **chronyd** があきらめ、終了します (マイナス値を使うと、終了しないようになります)。どちらの場合も、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片寄りをチェックし、1000 秒よりも大きい調整を 2 回無視した後に、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参照ソースと比較してどれくらい早くまたは遅く動いているかを調べることです。さらに、予想される値のエラーの範囲を予測することです。エラーの範囲が広すぎる場合は、測定が落ち着いておらず、予想される進みまたは遅れのレートがあまり信頼性の高いものではないことを示すことになります。**maxupdateskew** パラメーターでは、予想の信頼性が高くないため使用すべきでないかどうかを判断するしきい値が使えるようになります。デフォルトでは、しきい値は 1000 ppm になります。構文は以下のようになります。

```
maxupdateskew skew-in-ppm
```

電話を使ったサーバーへのダイアルアップ接続の場合、通常の *skew-in-ppm* の値は 100 で、LAN 回線上のコンピューターの場合は 5 または 10 です。信頼性のない予測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 つからの別の測定の後に新たな予測が生成されると、加重組み合わせのアルゴリズムを使っ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、ローカルクロックが更新され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
```

デフォルトでは *number-of-sources* は 1 です。これより大きな数に設定すると、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

このディレクティブは引数を取らず、クライアントのアクセスをログ記録しないように指定します。通常、クライアントアクセスはログ記録され、**chronyc** でクライアントコマンドを使って統計数字が報告されます。

reselectdist

chronyd が利用可能なソースから同期ソースを選ぶ際には、同期距離が最短のものを優先します。しかし、同様の距離のソースが複数存在して再選択を繰り返すことを避けるため、現在選択されていないソースの距離に固定距離が追加されます。これを、**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 マイクロ秒になります。

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、より低い stratum を持つソースは、たとえ距離が著しく離れていても、一般的により高い stratum を持つソースを選びます。**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** が保存することが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、**writertc** コマンドが **chronyc** で発行されると、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レートです。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、リアルタイムクロックは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、その片寄りのレートを測定する **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。これは、カーネルにシステムクロックが同期されていることを知らせ、カーネルはリアルタイムクロックを 11 分ごとに更新します。

17.2.4. chronyc のセキュリティー

Chronyc は **chronyd** へ 2 通りの方法でアクセスすることができます。

- 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できます。

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。この接続に失敗した場合 (**chronyc** が非 root ユーザーの下で実行されているときに失敗する可能性がある) 、**chronyc** は 127.0.0.1 と then ::1 へ接続を試みます。

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 がこれらのコマンドを受け取るホストは、**chronyd** の設定ファイルにある **cmdallow** ディレクティブ、または、 **chronyc** にある **cmdallow** コマンドで設定できます。デフォルトでは、これらのコマンドはローカルホスト (127.0.0.1 または ::1) のみが受け取ります。

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。

手順17.1 chronyc を使った chronyd へのリモートアクセス

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。

   例17.1

   次の内容を **/etc/chrony.conf** ファイルへ追加します。

   ```
   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 を永続的に開きたいときは、**--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** のアクセス用であり、一方 **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。ローカルに実行されている **chronyc** を使うと、これら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編集します。

17.3. CHRONY の使用

17.3.1. 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。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。

17.3.2. chronyd ステータスの確認

chronyd のステータスを確認するには、以下のコマンドを発行します。

```
~]$ 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
```

17.3.3. chronyd の起動

chronyd を起動するには、**root** で以下のコマンドを発行します。

```
~]# systemctl start chronyd
```

chronyd をシステム起動時に自動的に起動するには、**root** で以下のコマンドを発行します。

```
~]# systemctl enable chronyd
```

17.3.4. chronyd の停止

chronyd を停止するには、**root** で以下のコマンドを発行します。

```
~]# systemctl stop chronyd
```

chronyd が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
```

17.3.5. chrony の同期を確認する

chrony が同期しているかどうかを確認するには、**tracking**、**sources**、および **sourcestats** のコマンドを使用します。

17.3.5.1. chrony トラッキングの確認

chrony のトラッキングを確認するには、以下のコマンドを発行します。

```
~]$ 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

コンピューターが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) (利用可能である場合) 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

参照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。このようなコンピューターは stratum-1 コンピューターなので、上記の例のコンピューターは 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

通常の操作では、**chronyd** はシステムクロックを更新しません。これは、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はこれ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ます。その後に、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、またはシェルの日付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、これを **NTP** クライアントに報告) と異なることになります。この行で報告される値は、この効果による差異です。

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

オフセット値の長期の平均です。

Frequency

「frequency」 は、**chronyd** が修正しない場合にシステムクロックが間違うレートです。これは、ppm (100 万分の 1) で表されます。たとえば、1 ppm という値が意味するのは、システムクロックが 1 秒進んだと考えると、実際の時間と比較して 1.000001 秒進んでいるということです。

Residual freq

現在選択されている参照ソースの 「residual frequency」 を示します。これは、参照ソースからの測定が示すあるべき周波数と現在使用されている周波数の差異を反映しています。これが常にゼロとは限らない理由は、補正する手順が周波数に提供されているためです。参照ソースからの測定が取得され、新たな剰余周波数が計算されるごとに、この剰余の予測される正確性は既存の周波数の値の予測される正確性 (次の **skew** を参照) と比較されます。新たな剰余の加重平均は、加重がこれらの正確性に依存して計算されます。参照ソース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されます。

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印刷されます。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、対称的なピア配置において発生しやすい)。

Root dispersion

コンピューターが最後に同期する、stratum-1 コンピューターへ戻るすべてのコンピューターの分散を累積した合値値です。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、ナノ秒の分解能で印刷されます。

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。

17.3.5.2. chrony ソースの確認

ソースコマンドは、**chronyd** がアクセスしている現在の時間ソースについての情報を表示します。オプションの引数 -v を指定すると、詳細情報になります。この場合、コラムの意味を表示する見出しの行が表示されます。

```
~]$ 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、**=** はピアを、# はローカル接続している参照クロックを意味します。

S

このコラムは、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示します。「+」 は、選択されたソースと結合される受け入れ可能なソースを示します。「-」 は、受け入れ可能なソースで結合アルゴリズムに除外されたものを示します。「?」 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示します。「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) クロックを示します。「~」 は、時間の変動性が大きすぎるように見えるソースを示します。「?」 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、もしくは参照クロックの参照 ID を表示します。

Stratum

最も間近に受信し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、ローカルで参照クロックに接続され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。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、64 秒ごとに測定が行われます。**chronyd** は、支配的な条件に対応して自動的にポーリングレートを変化させます。

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、最近の 8 回の通信全体についての有効な返信が受信されたことを意味します。

LastRx

このコラムは、ソースから最後のサンプルが受信されたのがいつだったかを表示します。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、日数、年を意味します。値が 10 年の場合、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

このコラムは、ローカルクロックと最後に測定されたソース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、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。

17.3.5.3. chrony ソースの統計情報の確認

sourcestats コマンドを使うと、**chronyd** が現在調査している各ソースの誤差のレートとオフセットの予測プロセスについての情報を表示できます。オプション引数 **-v** を使うと、詳細情報になります。この場合、コラムの意味を表示する見出しの行が表示されます。

```
~]$ 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

NTP サーバー (またはピア) の名前または **IP** アドレス、または行の残りの部分が関連する参照クロックの参照 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、これらのポイントを使っ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もの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。このケースでは、コンピューターのクロックはサーバーよりも 10^9 分の 1 遅く実行していると判断されています。

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。

17.3.6. システムクロックを手動で調整する

システムクロックを直ちにステップするには、進行中の調整をスルーイングによって迂回し、以下のコマンドを **root** として実行します。

```
~]# 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レートを測定する必要のある **chrony** に影響を与えます。

17.4. 異なる環境での CHRONY の設定

17.4.1. 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、もう 1 台のコンピューターをマスタータイムサーバ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って手動で設定します。マスターは、再起動すると周囲のシステムから時間を取得し、平均値を計算してシステムクロックの設定を行います。ドリフトファイルは **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上でテキストエディターを使って **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
```

ここでの **192.0.2.0** は、クライアントの接続が許可されるネットワークまたはサブネットアドレスになります。

マスターの直接のクライアントに選ばれたシステム上でテキストエディターを使って **root** で **/etc/chrony.conf** を以下のように編集します。

```
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ならないクライアントシステム上では、**local** および **allow** ディレクティブが省略される以外は、**/etc/chrony.conf** ファイルは同じものになります。

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。このモードは、NTP サーバーが同期されていないか、または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、NTP サーバーがリアルタイムに同期されたように見せる **chronyd** オペレーティングを可能にします。

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させそれらを互いに同期させるには、Orphan モードを有効にする **local** ディレクティブの **orphan** オプションを使います。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設定する必要があります。そうすると、最小の参照 ID を持つサーバーのみがローカル参照を有効化し、他のサーバーはそれに同期します。サーバーが有効化に失敗すると別のサーバーが引き継ぎます。

17.5. CHRONYC の使用

17.5.1. chronyc を使った chronyd の制御

インタラクティブモードで **chronyc** コマンドラインユーティリティーを使って **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入力します。

```
~]# chronyc
```

制限のあるコマンドを使用するには、**chronyc** は **root** で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>
```

コマンドすべてを一覧表示するには、**help** と入力します。

以下のようにコマンドと併せて呼び出すことで、インタラクティブモード以外でもユーティリティーを開始することができます。

```
chronyc command
```

注記

chronyc を使用した変更は永続的な変更ではなく、**chronyd** の再起動後に失われます。永続的な変更の場合は、**/etc/chrony.conf** を変更します。

17.6. HW タイムスタンプを使った CHRONY

17.6.1. ハードウェアタイムスタンプについて

ハードウェアタイムスタンプは、一部の Network Interface Controller (NIC) でサポートされている機能です。これは、送受信されるパケットの正確なタイムスタンプを提供します。**NTP** タイムスタンプは通

常、システムクロックを使用してカーネルおよび **chronyd** が作成します。しかし、HW タイムスタンプが有効になると、パケットがリンク層または物理層に出入りする際に、NIC は独自のクロックを使用してタイムスタンプを生成します。**NTP** と共に使用すると、ハードウェアタイムスタンプは、同期の精度を大幅に向上させることができます。 最高精度を実現するには、**NTP** サーバーと **NTP** クライアントの両方が、ハードウェアタイムスタンプを使用しなければなりません。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、**PTP** があります。**PTP** に関する詳細は、19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支援に依存します。最高精度の同期を実現したい場合は、**PTP** サポートのあるスイッチおよびルーターを持つネットワーク上で **PTP** を使用し、このようなスイッチおよびルーターを持たないネットワーク上では **NTP** を使用します。

17.6.2. ハードウェアタイムスタンプのサポートの確認

NTP でのハードウェアタイムスタンプがインターフェースによってサポートされていることを確認するには、**ethtool -T** コマンドを使用します。**ethtool** が、***SOF_TIMESTAMPING_TX_HARDWARE*** および ***SOF_TIMESTAMPING_TX_SOFTWARE*** 機能、そして ***HWTSTAMP_FILTER_ALL*** フィルターモードも一覧表示する場合、**NTP** を使ったハードウェアタイムスタンプにインターフェースを使用できます。

例17.2 特定のインターフェース上におけるハードウェア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)
```

17.6.3. ハードウェアタイムスタンプの有効化

ハードウェア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、単一のインターフェースを指定できます。または、ワイルドカードキャラクター (*) を使用して、ハードウェアタイムスタンプをサポートするすべてのインターフェース上で、ハードウェアタイムスタンプを有効にすることができます。linuxptp パッケージの **ptp4l** など、他のアプリケーションがインターフェース上でハードウェアタイムスタンプを使用していない場合に備えて、ワイルドカード仕様を使います。複数の **hwtimestamp** ディレクティブが、chrony 設定ファイルで許可されています。

例17.3 hwtimestamp のディレクティブを使用したハードウェアタイムスタンプの実現

```
hwtimestamp eth0
hwtimestamp eth1
hwtimestamp *
```

17.6.4. 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 内の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
```

17.6.5. 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、たとえば **chrony** など、ソフトウェアの **NTP** 実装を実行する一般目的のコンピューターである **NTP** サーバーは、パケット送信後にのみハードウェア転送先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にタイムスタンプを保存することができません。転送後に生成された転送先タイムスタンプを受信する **NTP** クライアントを有効にするには、**/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
```

17.6.6. 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、サーバー上でクライアントのアクセスログに割り当てられたメモリーの最大サイズを指定します。

```
clientloglimit 100000000
```

17.6.7. ハードウェアタイムスタンプの確認

インターフェースがハードウェアタイムスタンプを正常に有効化し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、各インターフェース向けに **chronyd** からのメッセージが、正常に有効化されたハードウェアタイムスタンプと共に含まれているはずです。

例17.4 ハードウェアタイムスタンプが有効化されたインターフェースでメッセージをログ

```
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、**chronyc ntpdata** コマンドにより、転送先と受信先タイムスタンプモードおよびインターリーブモードを各 **NTP** ソースに報告することができます。

例17.5 各 NTP ソースの転送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を報告

```
~]# 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17.6 NTP 測定の安定性を報告

```
# chronyc sourcestats
```

ハードウェアタイムスタンプが有効化されると、**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** コラムに報告されます。

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
```

17.6.8. PTP-NTP ブリッジの設定

非常に精度が高い Precision Time Protocol (**PTP**) 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 **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ースが必要なほか、グランドマスターに近いか、またはグランドマスターと直接接続されている必要があります。これにより、ネットーワークで非常に精度の高い同期が確実に実行されます。

linuxptp パッケージから **ptp4l** および **phc2sys** のプログラムを設定し、**PTP** を使用してシステムクロックを同期するためにインターフェースを 1 つ使用します。この設定は、19章*ptp4l を使用した PTP の設定* で説明しています。他のインターフェースを使用してシステムの時間を提供するために **chronyd** を設定します。

例17.7 他のインターフェースを使用してシステムに時間を提供するよう chronyd を設定

```
bindaddress 203.0.113.74
hwtimestamp eth1
local stratum 1
```

17.7. 関連資料

以下の情報資料は、**chrony** に関するその他のリソースを提供します。

17.7.1. インストールされているドキュメント

- **`chronyc(1)`** man ページ — コマンドおよびコマンドオプションを含む **chronyc** コマンドラインインターフェースツールを説明しています。
- **`chronyd(8)`** man ページ — コマンドとコマンドオプションを含む **chronyd** デーモンについて説明しています。
- **`chrony.conf(5)`** man ページ — **chrony** 設定ファイルを説明しています。

17.7.2. オンラインのドキュメント

- 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 をご覧ください。

第18章 NTPD を使用した NTP 設定

18.1. 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。GPS を構成する衛星には 2 つ以上の原子時計が含まれており、時間信号を非常に正確なものにしています。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。低周波の無線伝送や GPS 経由で日時を取得するデバイスは多く存在します。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を維持することは重要です。たとえば、ネットワーキングでは、パケットとログで正確なタイムスタンプが必要になります。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。詳細情報は、『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** ユーザースペースデーモンは **chronyd** であり、**ntpd** を使用する場合は、これを無効にする必要があります。**chrony** についての情報は、17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 はシステムクロックにソフトウェアクロックを使用しています。これは 「*リアルタイムクロック*」 (RTC) と呼ばれる通常の組み込みハードウェアクロックよりも精度が高いものです。ハードウェアクロックについての情報は、**rtc(4)** および **hwclock(8)** の man ページを参照してください。システムクロックは、様々なクロックソースを使用して時間を記録します。通常は *タイムスタンプカウンター* (TSC) が使われます。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。システムクロックはシステム起動時に RTC から時間と日付を読み取ります。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。このため、システムクロックは外部の時間参照と常に同期する必要があります。**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。

18.2. 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。原子時計はソースであることから Stratum 0 と呼ばれます。ただし、Stratum 0 パケットがインターネット上で送信されるはなく、stratum 0 原子時計すべては stratum 1 と呼ばれるサーバーに接続されています。これらのサーバーは、Stratum 1 とマークされているパケットを送信します。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。同一 stratum のサーバーは相互にパケットを交換できますが、これらは同期の最も良い参照先となる stratum の一階層下の stratum に所属するように指定されます。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。

18.3. 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、1 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。時計の誤差が計算され、修正されるようになっているためです。これは、以前の単純な時間プロトコルシステムでは行われていませんでした。64 ビットのタイムスタンプを使用することで、233 ピコ秒の精度が提供されます。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」 および 「Era Offset」 を提供します。これらは、68 年を超える長さの時間を処理する際に、ソフトウェアをより堅牢にするために使用できます。この問題を 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役割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。これをシステムクロックと比較して、タイムオフセットを判断します。つまり、システムクロックの時間と **ntpd** が判断した時間の差異を判別します。システムクロックは、使用中のカウンターの周波数を変更することで、最大でも 1 秒あたり 0.5 ミリ秒という非常にゆっくりしたペースで調整されてこのオフセットを減らします。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。このゆっくりした変更はスルーイング (slewing) と呼ばれ、後ろ向きに実行することはできません。クロックのタイムオフセットが 128 ミリ秒 (デフォルト設定) を超える場合は、**ntpd** はクロックを進めるか、または遅らす 「ステップ」 が可能です。システム起動時のタイムオフセットが 1000 秒を超える場合は、ユーザーが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。時間変更のステップに敏感なシステムでは、**-x** オプション (**-g** オプションとは無関係) を使うことでしきい値を 128 ミリ秒から 600 秒に変更できます。ただし、**-x** オプションを使ってステップの制限を 128 ミリ秒から 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。

18.4. 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。

18.5. 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、または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、LOCAL または UTC のいずれかで、RTC クロックの設定方法を示します。この設定は、**日付と時刻** のグラフィカル設定ツール内の **システムクロックで UTC を使用** のチェックボックスを使うと簡単に変更できます。このツールの使用方法については、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。「関連資料」 を参照してください。

18.6. 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。残念なことに、Autokey にはセキュリティ上の深刻な問題があることが後になって判明しました。したがって、Red Hat は対称鍵の使用を強く推奨します。**ntpd** の認証オプションとコマンドについては、man ページ **ntp_auth(5)** 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** 一覧内に 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、または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。

ブロードキャストとマルチキャストの各モードでは、デフォルトで認証が必要になります。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っ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っ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については、「鍵を使った対称認証の設定」 の **key** オプションを参照してください。

18.7. 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** のある Red Hat Enterprise Linux では、デフォルトのクロックソースは **kvm-clock** になります。『Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide』 の 『KVM guest timing management (KVM ゲストのタイミング管理)』 の章を参照してください。

18.8. 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。

18.9. NTPD 設定ファイルの概要

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

誤差ファイルへのパスが指定され、Red Hat Enterprise Linux で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るため、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** オプションを削除しないでください。

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。

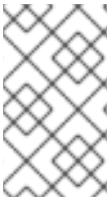
公開サーバーエントリー

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。特定のコマンドについての説明は、「NTP の設定」 を参照してください。必要な場合は、コマンドを例のすぐ下に追加します。



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。

18.10. 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 オプションを使うと、**ntpd** がオフセットの制限である 1000 秒を無視して、1000 秒を超える場合でも時間の同期を試みます。ただし、これはシステム起動時のみです。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た場合でも、サービスが再起動してオフセットが 1000 秒を超える場合は、システム起動後に終了します。

18.11. 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、以下のコマンドを発行します。

```
~]$ systemctl status chronyd
```

18.12. 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。

18.13. 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
```

18.14. 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
```

18.15. 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** パケットで構成されており、**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 **ファイアウォールの設定** ツールを使って、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** と入力してから **Enter** を押します。**ファイアウォールの設定** 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っ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
```

ファイアウォールの設定 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。

18.15.1. 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

実行時 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。

永続 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、または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 **Firewall の再読み込み** を選択します。

18.15.2. 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。**ポートとプロトコル** 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。

18.16. 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** および step-tickers リストの使用は非推奨とみなされているため、Red Hat Enterprise Linux 7 では **-g** オプションを付けた **ntpd** コマンドをデフォルトで使用し、**ntpdate** は使用しません。

Red Hat Enterprise Linux 7 の **ntpdate** サービスが役に立つのは、**ntpd** を使わずに単独で使用する場合のみです。サービスを並行して起動する **systemd** では、**ntpdate** サービスが提供する **time-sync.target** への並び順依存関係を指定しない限り、**ntpdate** サービスを有効にしても正確な時間を確保した後に他のサービスが起動することはありません。あるサービスが正確な時間を伴って起動されるようにするには、そのサービスに **After=time-sync.target** を追加し、ターゲット (**ntpdate** または **sntp**) を提供するサービスの 1 つを有効にします。Red Hat Enterprise Linux 7 上のサービスには、デフォルトでこの依存関係が含まれているものもあります (たとえば、**dhcpd**、**dhcpd6**、および **crond**)。

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 **0.rhel.pool.ntp.org** が含まれています。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。

18.17. 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** と共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。

18.17.1. 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 option
```

ここでの *option* は、以下のいずれか (1 つまたは複数) 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、タイムサーバー要求に応答しません。**ntpq** および **ntpdc** クエリーには影響ありません。**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** コマンドで **limited** オプションが必要になります。**ntpd** が **KoD** パケットで応答するには、**restrict** コマンドは **limited** および **kod** の両方のオプションが必要になります。

ntpq および **ntpdc** クエリーは増幅攻撃に使用可能 (詳細は 『CVE-2013-5211』 を参照) なので、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** オプションを削除しないでください。

18.17.2. NTP サービスへのレート制限アクセスの設定

システム上で実行中の **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」 の説明にあるように、**restrict** コマンドに **limited** オプションを追加します。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になります。

```
discard [average value] [minimum value] [monitor value]
```

- **average** — 許可される最小限の平均パケット間隔を指定します。$\log_2$ 秒の引数を取ります。デフォルト値は 3 です (2^3 は 8 秒と同等です)。

- **minimum** — 許可される最小限のパケット間隔を指定し、$\log_2$ 秒の引数を取ります。デフォルト値は 1 です (2^1 は 2 秒と同等です)。
- **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 コマンドの例を以下に示します。

```
discard average 4
```

```
discard average 4 minimum 2
```

18.17.3. 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。

18.17.4. 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。

18.17.5. 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。

18.17.6. 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。

18.17.7. 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。

18.17.8. 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。

18.17.9. 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。

18.17.10. 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。

18.17.11. 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** コマンドを使っ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。

18.17.12. 鍵を使った対称認証の設定

鍵を使っ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。

18.17.13. 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です。ここでは、間隔の秒数は 2 の *value* の累乗を計算して出されたものです。つまり、間隔は $\log_2$ 秒で表示されます。デフォルトの **minpoll** 値は 6 なので、2^6 は 64 秒となります。デフォルトの **maxpoll** 値は 10 なので、1024 秒になります。使用可能な値は、3 から 17 なので、8 秒から 36.4 時間までに相当します。これらのオプションは、**peer** または **server** で使用します。**maxpoll** を低く設定すると、クロックの精度が高まります。

18.17.14. 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。

18.17.15. 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。

18.17.16. 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。

18.18. 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。

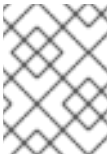
1. 以下の行を **/etc/sysconfig/ntpdate** ファイルに追加します。

   ```
   SYNC_HWCLOCK=yes
   ```

2. **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 経由の更新

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 として **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,
```

ソフトウェアクロックがハードウェアクロックと同期しているかを確認するには、出力の **status** 行を見ます (強調表示した部分) 。

最後から 3 桁目が 4 の場合、ソフトウェアクロックはハードウェアクロックと同期していません。

```
status 0x2401
```

最後の 4 桁の 2 桁目が 4 ではない場合、ソフトウェアクロックはハードウェアクロックと同期しています。

```
status 0x2001
```

18.19. 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一覧表示するには、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 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一覧も受け入れます。

GRUB メニューの変更方法についての詳細は、*25章GRUB 2 での作業* を参照してください。

18.20. 関連資料

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。

18.20.1. 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** コマンドを使っ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を説明しています。
- **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一覧表示します。
- **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。

18.20.2. 役に立つ Web ページ

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19章 PTP4L を使用した PTP の設定

19.1. 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。

19.1.1. 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。このアルゴリズムは、すべてのクロック上で実行されます。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、*ordinary clock* (OC: 通常のクロック) と呼ばれます。複数のポートがあるクロックではあるポートではマスターに、別のポートではスレーブになることができ、*boundary* clock (BC: 境界クロック) と呼ばれます。トップレベルのクロックは *グランドマスタークロック* と呼ばれ、*全地球測位システム* (GPS) の時間ソースを使って同期できます。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。

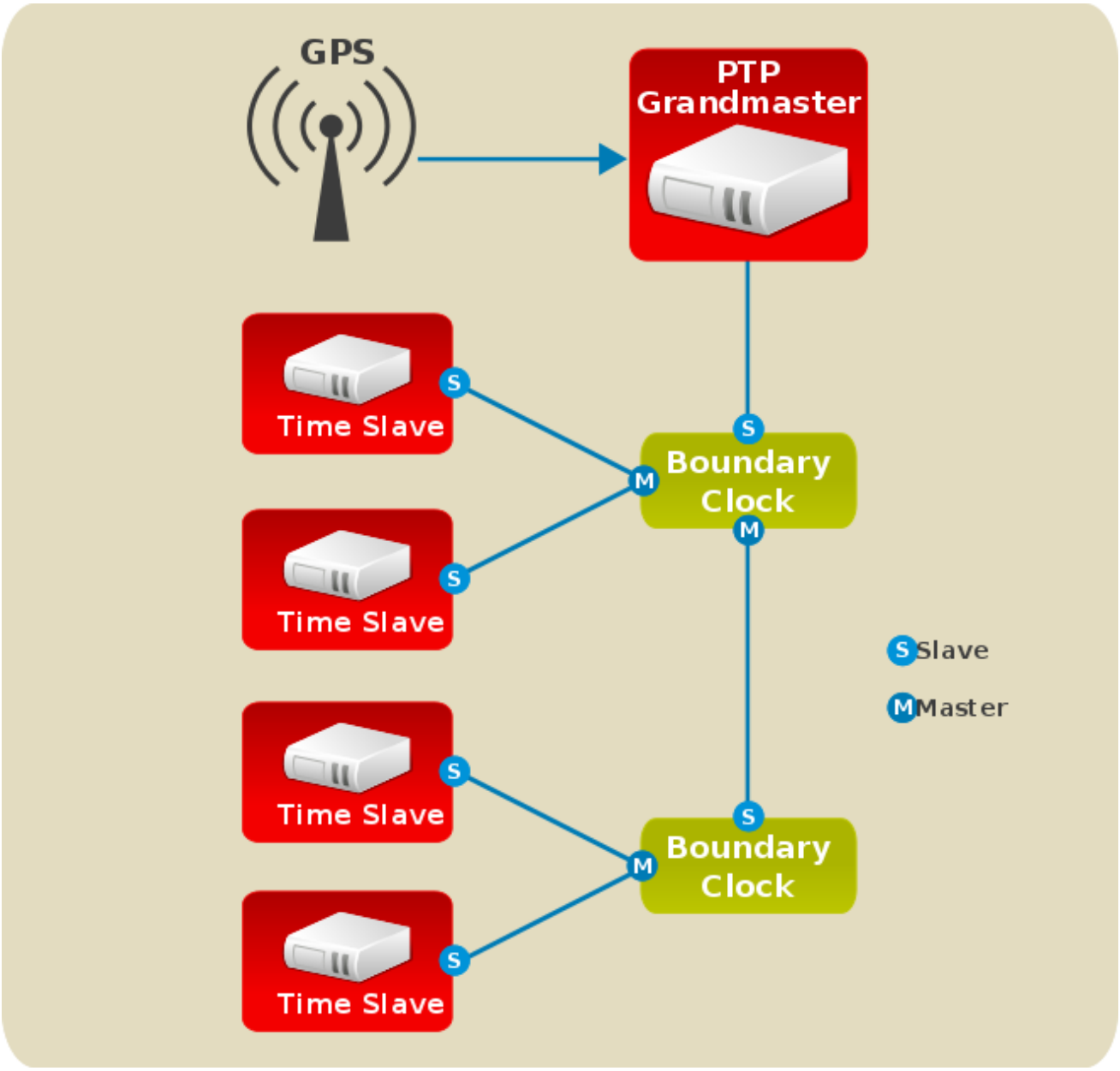


図19.1 PTP グランドマスター、境界、スレーブの各クロック

19.1.2. 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。通信パスで使用される PTP を認識しないコンポーネントが複数あると、これがさらに増幅します。最高の精度を達成するには、**PTP** クロック間のすべてのネットワークコンポーネントが **PTP** ハードウェア対応となっていることが推奨されます。ネットワークハードウェアがのすべてが **PTP** をサポートしているわけではない大型のネットワークにおける時間同期には、**NTP** が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 には自らのボード上のクロックが備わります。これは、送受信される **PTP** メッセージのタイムスタンプに使われます。**PTP** マスターに同期されるのはこのボード上のクロックで、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、システムクロックが **PTP** メッセージのタイムスタンプに使わ

れ、直接 **PTP** マスターに同期されます。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。

19.2. PTP の使用

PTP を使用するには、対象とするインターフェー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。

19.2.1. 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、以下のように **ethtool** ユーティリティーを使ってインターフェースにクエリーすることです。

```
~]# 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)
```

ここでの *eth3* は、チェックするインターフェースになります。

ソフトウェアタイムスタンプのサポートについては、パラメーター一覧に以下を含めます。

- **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、パラメーター一覧に以下を含めます。

- **SOF_TIMESTAMPING_RAW_HARDWARE**
- **SOF_TIMESTAMPING_TX_HARDWARE**
- **SOF_TIMESTAMPING_RX_HARDWARE**

19.2.2. 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。ユーザースペースのサポートは、**linuxptp** パッケージ内のツールが提供します。**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** と **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 を使った PTP 時間の実行」 を参照してください。

19.2.3. 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 でシステムサービスを管理する方法の詳細情報については、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ースに **-i** オプションを指定する必要があります。以下のコマンドを **root** で入力してください。

```
~]# ptp4l -i eth3 -m
```

ここでの *eth3* は、設定するインターフェースになります。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、**s1** はクロックステップ、**s2** はロックになります。サーボがロック状態 (**s2**) になると、(**ptp4l(8)** man ページの説明にあるように) **pi_offset_const** が設定ファイルでプラスの値に設定されている場合を除いて、クロックはステップされません (ゆっくり調整されるのみ)。**adj** の値は、10 億分の 1 (ppb) で表されるクロックの周波数調整です。path delay の値は、ナノ秒で表されるマスターから送信された同期メッセージの予想遅延です。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** インターフェースです (上記の例に基づく)。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。しかし、**-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
```

19.2.3.1. 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 オプションは *ピアツーピア* (P2P) 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

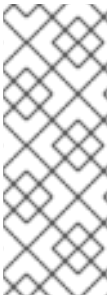
-E オプションは、*エンドツーエンド* (E2E) 遅延測定メカニズムを選択します。これがデフォルトです。

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 オプションは、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 モードで **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

単一の **PTP** 通信パス上にあるクロックはすべて、遅延測定で同一の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

19.3. 複数のインターフェースでの 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ースで使用する場合、*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、RFC 3704, Ingress Filtering for Multihomed Networks の Strict Reverse Path (厳密な逆パス) の推奨に従って *Strict Reverse Path Forwarding (厳密は逆パス転送)* を使用します。詳細は、Red Hat Enterprise Linux 7 セキュリティガイド 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ースごとに 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 インターフェースで **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** のインターフェースの場合は、以下のようになります。

```
~]# sysctl -w net.ipv4.conf.em1.rp_filter=2
```

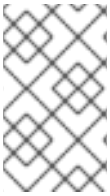
これらの設定を再起動しても保持するには、**/etc/sysctl.conf** ファイルを変更します。すべてのインターフェースのモード、または特定のインターフェースのモードを変更できます。

すべてのインターフェー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ー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ースと特定のインターフェースの設定を使用する場合、各インターフェー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ー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ースは、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、または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、または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。

19.4. 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、ランタイムで **-f** オプションを使って指定する必要があります。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]
```

19.5. 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、範囲がローカルで実行している **ptp4l** インスタンスに制限されます。範囲の値を大きくすると、ローカルクロックに加えて **PTP** ノードからも情報を取得します。取得可能な情報には、以下のものが含まれます。

- **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 コマンドの完全な一覧を表示するには、**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。

19.6. 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 **PTP** ハードウェアクロック (PHC) 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 オプションにより、**phc2sys** は **ptp4l** アプリケーションから同期されているクロックを読み取ります。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。システムクロックは、**-r** オプションも指定されない限り同期しません。システムクロックをタイムソースの候補とする場合は、**-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 オプションにより、**phc2sys** は **ptp4l** アプリケーションから同期されるクロックを読み取ります。システムクロックをタイムソースの候補にする場合は、**-r** オプションを 2 回指定します。

または **-s** オプションを使用して、システムクロックを特定インターフェースの **PTP** ハードウェアに同期します。以下は例になります。

```
~]# 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 TAI から UTC のオフセットを取得します。

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。通常、2、3 年ごとに起こります。**-O** オプションは、**-w** オプションを使用しない場合にこのオフセットを手動で設定する際に、以下のように使用する必要があります。

```
~]# phc2sys -s eth3 -O -36
```

phc2sys サーボがロック状態になると、**-S** オプションを使用しない限り、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** オプションを使うと、**ptp4l** によるクロックの同期を **phc2sys** が待機するため、これを必ずしも後に起動する必要はなくなります。

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。

19.7. 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** と **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印刷するときは、 **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、2 の **n** 乗として秒単位で指定します。 たとえば、出力を **1024** 秒ごとまで減らすときは、以下の行を **/etc/ptp4l.conf** ファイルに追加します。

```
summary_interval 10
```

ptp4l の出力例です。**summary_interval** は 6 に設定されています。

```
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 に設定されています。したがってメッセージは 1 秒ごとに印刷されます。これが最大頻度です。メッセージは LOG_INFO レベルでログに記録されます。メッセージを無効にするには、**-l** オプションを使って最大ログレベルを 5 以下に設定します。

```
~]# phc2sys -l 5
```

phc2sys の出力を減らすときは **-u** オプションを使用します。

```
~]# 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
```

これらのオプションを使うと、統計情報の更新間隔は 60 秒に設定され (**-u**) 、**phc2sys** は **ptp4l** が同期状態になるまで待機し (**-w**) 、メッセージは標準出力で印刷されます (**-m**) 。**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

19.8. NTP を使った 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、それらを **ntp.conf** に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。

19.9. PTP を使った NTP 時間の実行

NTP から **PTP** への逆方向の同期も可能です。**ntpd** を使ってシステムクロックを同期する場合、**ptp4l** を **priority1** オプションで (または最善のマスタークロックアルゴリズムに含まれている他のクロックオプションで) グランドマスタークロックに設定して **PTP** 経由でシステムクロックから時間を配布することができます。

```
~]# 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** を使って **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次のようになります:

```
OPTIONS="-a -r"
```

root で以下のように行を編集します。

```
~]# vi /etc/sysconfig/phc2sys
OPTIONS="-a -r -r"
```

ここで、**-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 クロックの短時間の周波数変更を避けるには、PI サーボでより小さい **P** (比例) および **I** (積分) の各定数を使用することでシステムクロックへの同期を緩やかにすることができます。

```
~]# phc2sys -a -r -r -P 0.01 -I 0.0001
```

19.10. 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、または **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* (システムで設定されている **NTP** デーモンに応じて異なる **chronyd** または **ntpd** へのSHM 参照クロック) により **phc2sys** and **ptp4l** で提供されます。**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、または **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ースを確認し、**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**、および **ptp4l** の設定ファイルは **/var/run/timemaster/** に書き込まれます。

19.10.1. 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 におけるシステムサービスの管理についての詳細は、10章*systemd によるサービス管理* を参照してください。

19.10.2. 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
```

以下のような名前のセクションに注目してください:

```
[ntp_server address]
```

これは、**NTP** サーバーセクションの例であり、「ntp-server.local」 はローカル LAN 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。セクション例の短いポーリング値は公開サーバーには適していないことに注意してください。適切な **minpoll** および **maxpoll** 値の説明については、18章*ntpd を使用した NTP 設定* を参照してください。

次のような名前のセクションに注意してください:

```
[ptp_domain number]
```

「PTP domain」 は、相互に同期する 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構成します。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

フェースには **PTP** クロック (PHC) が割り当てられます。ただし、NIC 上のインターフェースのグループが PHC を共有することは可能です。別々の **ptp4l** インスタンスは、同じ PHC を共有するインターフェースの各グループ、およびソフトウェアのタイムスタンプのみをサポートする各インスタンスに対して起動します。すべての **ptp4l** インスタンスはスレーブとして実行されるように設定されます。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ースが 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。

以下のような名前のセクションに注目してください:

```
[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。セクション見出しにホスト名または **IP** アドレスを組み込みます。
- **[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ースを指定します。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** 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。

19.10.3. timemaster オプションの設定

手順19.1 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。セクション例にある短いポーリング値は公開サーバーには適さないことに注意してください。適切な **minpoll** 値および **maxpoll** 値の説明については、18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必要のあるインタフェースを追加するには、**#[ptp_domain 0]** セクションを編集し、インターフェースを追加します。必要に応じて追加のドメインを作成します。以下は例になります。

```
[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、17章*chrony スイートを使用した NTP 設定* を参照してください。

5. **chronyd** をこのシステムの **NTP** サーバーとして使用している場合、**[chrony.conf]** セクションのデフォルトの**include /etc/chrony.conf** エントリーの下にオプションを追加します。**/etc/chrony.conf** のパスが変更されたことが認識される場合、デフォルトの **include** エントリーを編集します。

6. **ntpd** をこのシステムの **NTP** サーバーとして使用している場合、**[ntp.conf]** セクションのデフォルトの**include /etc/ntp.conf** エントリーの下にオプションを追加します。**/etc/ntp.conf** のパスが変更されたことが認識される場合、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、17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、18章*ntpd を使用した NTP 設定* を参照してください。

10. **[phc2sys]** セクションに、**timemaster** で呼び出される場合に **phc2sys** 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、**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
```

19.11. 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)。カーネルのティックレスモードは、**nohz=off** をカーネル起動オプションパラメーターに追加することで無効にできます。ただし、**kernel-3.10.0-197.el7** に加えられた最近の改良により、システムクロックの安定性が大幅に高まり、ほとんどのユーザーにとって **nohz=off** の有無によるクロックの安定性の違いは小さくなりました。

ptp4l および **phc2sys** アプリケーションは新規の適応型のサーボ (adaptive servo) 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。**ptp4l** にこれを使用するには、以下の行を **/etc/ptp4l.conf** ファイルに追加します。

```
clock_servo linreg
```

/etc/ptp4l.conf に変更を加えた後に、**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
```

phc2sys にこれを利用するには、以下の行を **/etc/sysconfig/phc2sys** ファイルに追加します。

```
-E linreg
```

/etc/sysconfig/phc2sys に変更を追加した後に、**root** で以下のコマンドを実行してコマンドラインから **phc2sys** サービスを再起動します。

```
~]# systemctl restart phc2sys
```

19.12. 関連資料

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。

19.12.1. インストールされているドキュメント

- **ptp4l(8)** 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)** 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 page — 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)** 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。

19.12.2. 役に立つ Web ページ

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について説明します。

第20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。

20.1. システムプロセスの表示

20.1.1. 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。静的な一覧、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一覧を表示させるには、**top** コマンドまたは **システムモニター** 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一覧表示するには、シェルプロンプトで以下を入力します。

```
ps ax
```

一覧表示された各プロセスについて **ps ax** コマンドが表示するのは次のとおりです。プロセス ID (**PID**)、それに関連付けされたターミナル (**TTY**)、現在のステータス (**STAT**)、累積 CPU 時間 (**TIME**)、実行ファイル名 (**COMMAND**) です。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][出力は省略されています]
```

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**) およびメモリ使用率 (**%MEM**)、キロバイト単位での仮想メモリサイズ (**VSZ**)、キロバイト単位での非スワップの物理メモリサイズ (**RSS**)、プロセスの開始日時を表示します。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][出力は省略されています]
```

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一覧は、**ps**(1) の man ページを参照してください。

20.1.2. top コマンドの使用

top コマンドは、システム上で実行中のプロセスの一覧をリアルタイムで表示します。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、一覧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
```

一覧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[出力は省略されています]
```

表20.1「インタラクティブな top コマンド」 では、**top** で使用可能なインタラクティブコマンドを表示しています。詳細は、**top**(1) の man ページを参照してください。

表20.1 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。
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 kill 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	一覧をユーザー別に並べ替えます。
M	一覧をメモリ使用率で並べ替えます。
P	一覧を 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。

20.1.3. システムモニターツールの使用

システムモニター ツールの **プロセス** 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ー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一覧を表示します。

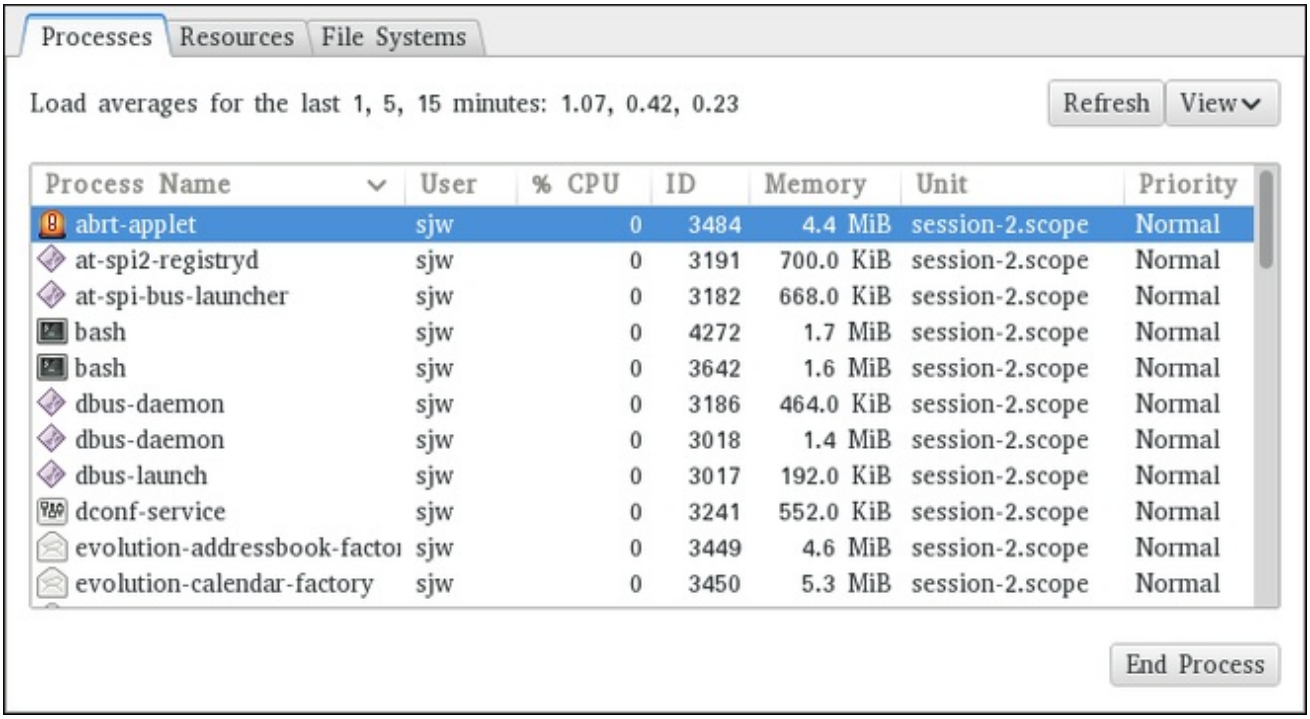


図20.1 システムモニター — プロセス

一覧表示された各プロセスについて **システムモニター** ツールが表示するのは次のとおりです。プロセス名 (**Process Name**)、現在の状態 (**Status**)、CPU 使用量のパーセンテージ (**% CPU**)、nice 値 (**Nice**)、プロセス ID (**ID**)、メモリ使用量 (**Memory**)、プロセスが待機しているチャンネル (**Waiting Channel**)、セッションについての補足情報 (**Session**) です。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。再度コラム名をクリックすると、昇順と降順が切り替わります。

デフォルトでは、**システムモニター** ツールは現在のユーザーが所有しているプロセスの一覧を表示します。**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。

- プロセスの一覧を更新する
- 一覧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

20.2. メモリ使用量の表示

20.2.1. free コマンドの使用

free コマンドは、システムのメモリ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。値をメガバイトで表示するには、**-m** コマンドラインオプションを指定してください。

```
free -m
```

例を示します。

```
~]$ 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一覧は、**free**(1) の man ページを参照してください。

20.2.2. システムモニターツールの使用

システムモニター ツールの **リソース** タブは、システムのメモリ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。

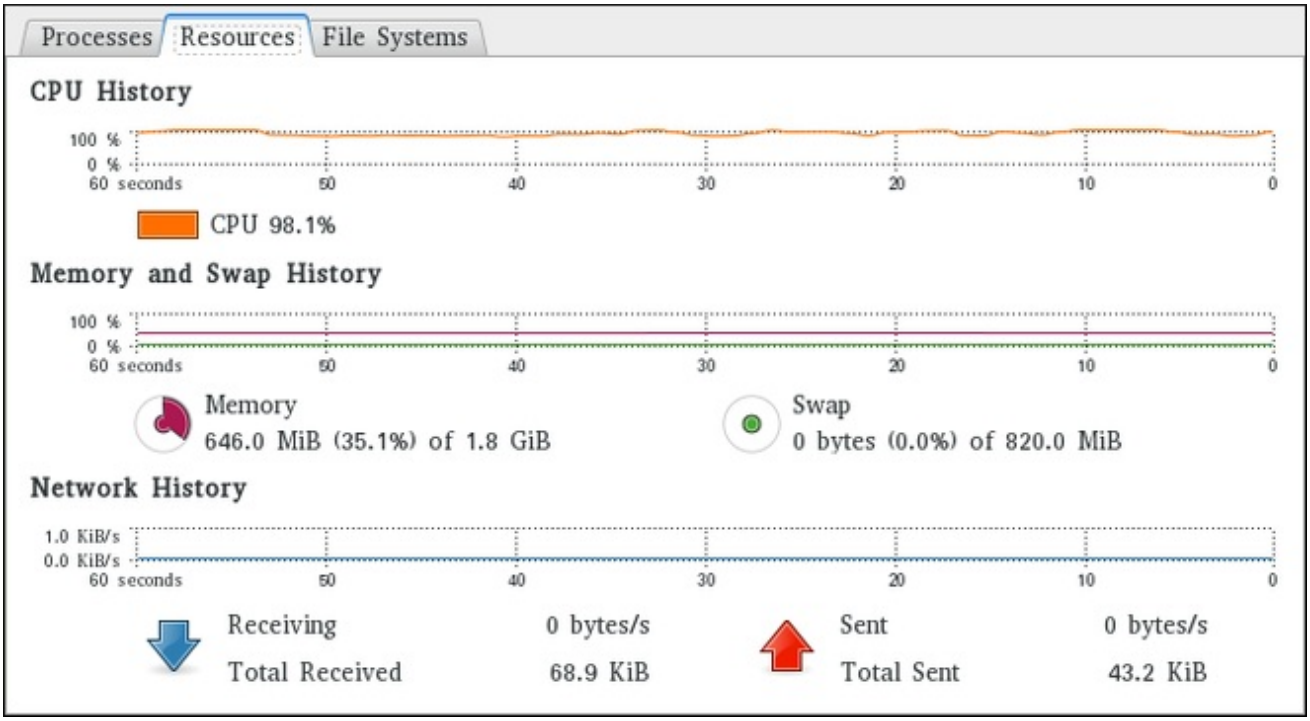


図20.2 システムモニター — リソース

システムモニター ツールは、**メモリとスワップの履歴** のセクションでメモリおよびスワップの使用履歴、さらには物理メモリの総容量 (**メモリ**)、スワップ領域 (**スワップ**)、使用量をグラフィカルに表示します。

20.3. CPU 使用率の表示

20.3.1. システムモニターツールの使用

システムモニター ツールの **リソース** 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 (リソース) タブをクリックしてシステムの CPU 使用率を表示します。

システムモニター ツールの **CPU 使用率の履歴** セクションでは、CPU 使用率の履歴をグラフィカルに表示するとともに現在使用中の CPU をパーセンテージで表示します。

20.4. ブロックデバイスとファイルシステムの表示

20.4.1. 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一覧を表示できます。**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。**lsblk** コマンドは **udev** から情報を読み取るため、**root** 以外のユーザーでも使用できます。ブロックデバイスの一覧を表示するには、シェルプロンプトで以下のコマンドを入力します。

```
lsblk
```

一覧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**)、メジャーおよびマイナーデバイス番号 (**MAJ:MIN**)、リムーバブルデバイスかどうか (**RM**)、そのサイズ (**SIZE**)、読み取り専用デバイスかどうか (**RO**)、そのタイプ (**TYPE**)、デバイスのマウント先 (**MOUNTPOINT**) です。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一覧表示します。通常の一覧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
```

例を示します。

```
~]$ 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一覧は、**lsblk**(8) の man ページを参照してください。

20.4.2. 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。この場合は、シェルプロンプトで **root** として以下のコマンドを入力します。

```
blkid
```

一覧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。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一覧表示します。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"
```

上記のコマンドに **-p** および **-o udev** のオプションを使用して、さらに詳しい情報を取得することも可能です。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
```

例:

```
~]# 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一覧は、**blkid**(8) の man ページを参照してください。

20.4.3. 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一覧を表示します。シェルプロンプトで以下を入力します。

```
findmnt
```

一覧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。マウントポイント先 (**TARGET**)、ソースデバイス (**SOURCE**)、ファイルシステムのタイプ (**FSTYPE**)、関連するマウントオプション (**OPTIONS**) です。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[出力は省略されています]
```

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一覧表示します。通常の一覧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
```

例を示します。

```
~]$ 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[出力は省略されています]
```

特定のタイプのファイルシステムのみを一覧表示するよう選択することも可能です。そのためには、**-t** コマンドラインオプション、その次にファイルシステムの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一覧表示するには、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一覧は、**findmnt**(8) の man ページを参照してください。

20.4.4. 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
```

一覧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。ファイルシステム名 (**Filesystem**)、サイズ (**1K-blocks** または **Size**)、使用領域 (**Used**)、利用可能な領域 (**Available**)、使用領域のパーセンテージ (**Use%**)、ファイルシステムのマウント先 (**Mounted on**) です。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。この情報をメガバイトとギガバイトで表示するには、**-h** オプションを指定すると **df** がヒューマンリーダブルな形式で値を表示します。

```
df -h
```

例を示します。

```
~]$ 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一覧は、**df**(1) の man ページを参照してください。

20.4.5. 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
```

例:

```
~]$ 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単位で表示するには、**-h** オプションを指定するとヒューマンリーダブルな形式で値を表示することができます。

```
du -h
```

例を示します。

```
~]$ du -h
```

```
15M     ./Downloads
4.0K    ./.mozilla/extensions
4.0K    ./.mozilla/plugins
12K     ./.mozilla
15M     .
```

du コマンドは、一覧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** オプションを使用します。

```
du -sh
```

例:

```
~]$ du -sh
15M     .
```

利用可能なコマンドラインオプションの一覧は、**du**(1) の man ページを参照してください。

20.4.6. システムモニターツールの使用

システムモニター ツールの **ファイルシステム** タブは、グラフィカルユーザーインターフェー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一覧を表示します。

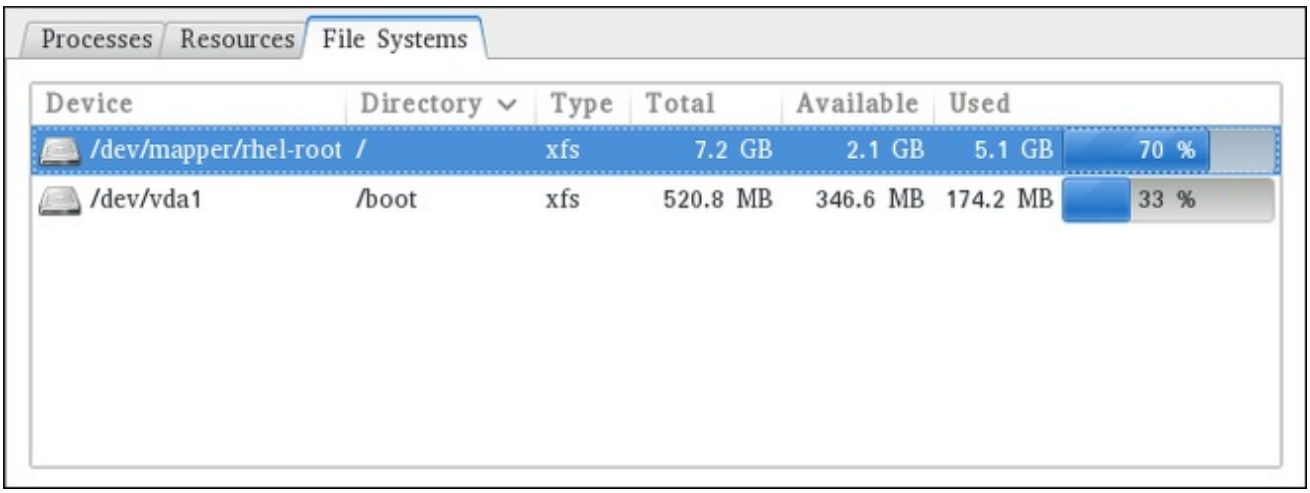


図20.3 システムモニター — ファイルシステム

一覧表示された各ファイルシステムについて **システムモニター** ツールが表示するのは次のとおりです。ソースデバイス (**Device**)、マウントポイント先 (**Directory**)、ファイルシステムのタイプ (**Type**)、さらにはそのサイズ (**Total**)、利用可能な領域 (**Available**)、使用領域 (**Used**) です。

20.5. ハードウェア情報の表示

20.5.1. 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 デバイスを一覧表示するには、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一覧を表示します。

```
~]$ 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)[出力は省略されています]
```

-v コマンドラインオプションを使用して、詳しい出力を表示することもできます。さらに詳細な出力を表示したい場合は、**-vv** を使用してください。

```
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、メモリサイズを確認するには、以下を入力します。

```
~]$ lspci -v[出力は省略されています]

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
[出力は省略されています]
```

利用可能なコマンドラインオプションの一覧は、**lspci**(8) の man ページを参照してください。

20.5.2. 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 デバイスを一覧表示するには、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一覧を表示します。

```
~]$ 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[出力は省略されています]
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
```

例を示します。

```
~]$ lsusb -v[出力は省略されています]

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[出力は省略されています]
```

利用可能なコマンドラインオプションの一覧は、**lsusb**(8) の man ページを参照してください。

20.5.3. lscpu コマンドの使用

lscpu コマンドは、システム内にある CPU に関する情報を一覧表示します。たとえば、CPU 数、アーキテクチャー、ベンダー、ファミリ、モデル、CPU キャッシュなどです。シェルプロンプトで以下を入力してください。

```
lscpu
```

例:

```
~]$ 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一覧は、**lscpu**(1) の man ページを参照してください。

20.6. ハードウェアエラーの確認

Red Hat Enterprise Linux 7 では、新しい HERM (Hardware Event Report Mechanism: *ハードウェアイベントレポートメカニズム*) 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一覧を表示します。

```
~]$ 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。

20.7. 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。

20.7.1. 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20.2「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20.2 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。
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…
```

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 での新しいパッケージのインストール方法の詳細は、「パッケージのインストール」 を参照してください。

20.7.2. 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、また特定のランレベルでこれを有効にする方法について説明します。Red Hat Enterprise Linux 7 におけるシステムサービスの管理方法の詳細については、10章*systemd によるサービス管理* を参照してください。

20.7.2.1. サービスの開始

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
```

20.7.2.2. 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
```

20.7.2.3. 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。

20.7.3. Net-SNMP の設定

Net-SNMP Agent Daemon の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、これを使っ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
```

20.7.3.1. 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)[出力は省略されています]
```

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** 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。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[出力は省略されています]
```

20.7.3.2. 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、X.509 証明書を用いた TLS 認証、Kerberos 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** 設定ファイルで **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** オプションを付けた **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[出力は省略されています]
```

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** に追加する **rwuser** ディレクティブ (または **-ro** のコマンドラインオプションが提供されている場合は **rouser**) の形式は、**rwcommunity** および **rocommunity** ディレクティブと似ています。

```
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** オプションを使うと、認証されていない要求を許可することができ、**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[出力は省略されています]
```

20.7.4. 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 パッケージの一覧、システム上で現在実行中のプロセス一覧、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」 で説明のとおりユーザーは 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。

20.7.4.1. ハードウェアの設定

Net-SNMP に含まれている **Host Resources MIB** は、ホストのハードウェアおよびソフトウェアの現行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20.3「利用可能な OID」 には、その MIB で利用可能な様々な OID を要約した内容が記載されています。

表20.3 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。

OID	詳細
HOST-RESOURCES-MIB::hrStorage	メモリ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一覧が含まれています。
HOST-RESOURCES-MIB::hrSWRun	実行中の全プロセス一覧が含まれています。
HOST-RESOURCES-MIB::hrSWRunPerf	HOST-RESOURCES-MIB::hrSWRun からのプロセステーブル上のメモリと CPU 統計が含まれています。
HOST-RESOURCES-MIB::hrSWInstalled	RPM データベースの一覧が含まれています。

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。

20.7.4.2. CPU とメモリ情報

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** および **ssCpuRawIdle** の OID は、システムがカーネル領域、ユーザー領域または I/O でプロセッサー時間の大半を費やしているかどうかを判断する際に役立つカウンターを提供します。**ssRawSwapIn** と **ssRawSwapOut** は、システムがメモリ消費不足かどうかを知る際にも有用な場合もあります。

さらなるメモリ情報は、**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一覧があります。

```
~]$ 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
```

20.7.4.3. 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
```

20.7.4.4. 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ースのエントリー、インターフェースの設定、インターフェー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ー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ー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
```

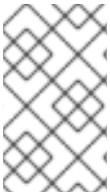
20.7.5. Net-SNMP の拡張

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ー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。本項では、Optional チャンネルのシェルスクリプトと Perl プラグインを使ったエージェントの拡張方法について説明します。本項では、net-snmp-utils 及び 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。

20.7.5.1. 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ってこの仕組みを説明しています。



注記

Net-SNMP Agent には、**proc** ディレクティブによりプロセステーブルを確認する組み込みメカニズムも備わっています。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 でこのスクリプトを利用可能にするには、システムパス上にある場所にスクリプトをコピー後、実行ビットを設定して、**extend** ディレクティブを **/etc/snmp/snmpd.conf** ファイルに追加します。**extend** ディレクティブの形式は、以下のとおりです。

```
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表示するには、**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。

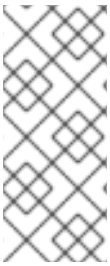
この最後の例では、システムの空きメモリと **httpd** プロセスの数のクエリーを示しています。このクエリーは、パフォーマンステスト中にメモリ負担に与えるプロセス数の影響を知るために使用することができます。

```
~]$ 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
```

20.7.5.2. Perl による Net-SNMP の拡張

extend ディレクティブを使用してシェルスクリプトを実行することは、SNMP を介して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制限された方法の 1 つです。Net-SNMP Agent は、カスタムオブジェクトを公開するための組み込み Perl インターフェースも提供します。Optional チャンネルの net-snmp-perl パッケージには、Red Hat Enterprise Linux に組み込み Perl プラグインを書き込むために使用される **NetSNMP::agent** Perl モジュールがあります。



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「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」の記事で説明されているステップに従ってください。

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一覧が含まれており、反復されデータが追加されるはずです。一覧の **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一覧については、**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一覧は、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。再起動されると、**snmpwalk** が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
```

20.8. 関連資料

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。

20.8.1. 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**(1) — `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 ページです。
- **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21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 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と配布され、ハードウェアや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ースを提供します。

21.1. OPENLMI の概要

OpenLMI は、物理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ー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構成されています。

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上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、インターフェースを提供します。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ー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 と配布されるインターフェースには、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ースを使っていても、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。

21.1.1. 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ースを提供します。
- 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 プロバイダーの集合とともに配布され、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ースを提供する 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。

21.1.2. 管理機能

OpenLMI の主な機能には、ストレージデバイスやネットワーク、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 と配布される CIM プロバイダーの完全な一覧については、表21.1「利用可能な CIM プロバイダー」 を参照してください。

表21.1 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。
openlmi-logicalfile	ファイルおよびディレクトリー読み取り用の CIM 。
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 として含まれています。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。

21.2. 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。

21.2.1. 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って監視および管理する予定のシステムです。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** として以下を入力し、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 と配布される CIM プロバイダーの一覧については、表21.1「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一覧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アクティブ化するには、**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化して OpenPegasus CIMOM を開始します。現行セッションで **tog-pegasus.service** ユニットをアクティブ化するには、**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予定の場合は、TCP 通信をポート **5989** (**wbem-https**) で有効にします。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って対話することができます。管理システム上で直接 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」 で説明されているステップも完了してください。

21.2.2. 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、以下のステップを実行します。

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
   ```

   これで 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」 に説明されているように、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することができます。

21.3. 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 (ウェブベースのエンタープライズ管理) 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。

安全な認証を確保するには、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) または TLS (Transport Layer Security) 証明書が必要になります。

システム上で 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。
- 認証局署名証明書はいったん設定されるとクライアントへの導入は容易ですが、必要な初期投資が大きくなる可能性があります。

認証局署名証明書を使用する場合は、クライアントシステム上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、その権限を使ってすべての管理システムの CIMOM 証明書に署名することが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ので、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、より高い認証局 (Verisign、CAcert、RSA などその他多数) によって署名される場合もあります。

ファイルシステム上のデフォルトの証明書と信頼の保存場所を、表21.2「証明書および信頼の保存場所」 に一覧表示します。

表21.2 証明書および信頼の保存場所

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 のみが知っている秘密鍵。
sslTrustStore	**/etc/Pegasus/client.pem**	信頼できる証明書機関の一覧を提供するファイルまたはディレクトリー。

重要

表21.2「証明書および信頼の保存場所」 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詳細情報については、10 章*systemd によるサービス管理* を参照してください。

21.3.1. 自己署名証明書の管理

自己署名証明書は、自身の秘密鍵を使って署名し、他のいかなる信頼チェーンにも接続されません。管理システム上では、**tog-pegasus** の初回起動時前に管理者が証明書を提供していない場合、システムのプライマリーホスト名を証明書の件名に使用して、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。これらの証明書を修正するには、このサブジェクトについて 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、以下のステップに従います。

1. **/etc/Pegasus/server.pem** 証明書を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/** ディレクトリーにコピーします。これを実行するには、**root** でシェルプロンプトに以下を入力します。

   ```
   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で、**root** ユーザーが SSH プロトコルを使っ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** サービスのインストールおよび設定、また **scp** コマンドを使って SSH プロトコルでファイルを送信する方法については、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上の証明書の整合性を検証します。管理システム上の **/etc/Pegasus/server.pem** ファイルのチェックサムを計算するには、そのシステム上で **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上の **/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、このシステム上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信頼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
```

21.3.2. Identity Management を使った認証局署名証明書の管理 (推奨)

Red Hat Enterprise Linux の Identity Management 機能は、ドメインにジョインしたシステム内での 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ジョインさせる方法については、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 ドキュメンテーション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、以下のステップが必要です。

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信頼ストアにコピーします。

```
cp /etc/ipa/ca.crt /etc/pki/ca-trust/source/anchors/ipa.crt
```

3. 信頼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。

ipa-admintools パッケージがインストール済みの Identity Management ドメイン内のシステムからであれば、このコマンドが実行できます。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上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、信頼ストアをアップデートするには、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信頼ストアにコピーします。

```
cp /etc/ipa/ca.crt /etc/pki/ca-trust/source/anchors/ipa.crt
```

3. 信頼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、以下のステップに従って信頼ストアを更新します。

1. **root**として、同一の Identity Management ドメインにジョインしている他のシステムから安全に **/etc/ipa/ca.crt** ファイルを信頼ストア **/etc/pki/ca-trust/source/anchors/** ディレクトリーにコピーします。
2. 信頼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
```

21.3.3. 認証局署名証明書を手動で管理する

認証局署名証明書を 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。

- デフォルトである認証局が信頼されている場合、これを達成するために実行が必要なステップはありません。
- 認証局がデフォルトで信頼されていない場合、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。
  1. **root** で以下のコマンドを入力して、証明書を信頼ストアにコピーします。

```
cp /path/to/ca.crt /etc/pki/ca-trust/source/anchors/ca.crt
```

  2. 信頼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、証明書についての情報を保管します。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、管理システム上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 信頼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
```

上記のステップ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、上記の手動のステップは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。

21.4. 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。

21.4.1. 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信頼ストアに対して確認します。この確認を無効にするには、**lmishell** コマンドを **--noverify** または **-n** オプションとともに実行します。

```
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** キーを使っ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やクラス、メソッド、オブジェクト属性が含まれます。

履歴閲覧

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することができます。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ています。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。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()
```

コマンド履歴に保存される行数は、**~/.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設定できます。さらに、同じ設定ファイルの **history_file** オプションの値を変更することで、履歴ファイルの場所を変えることができます。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)
```

例外処理を永続的に無効にするには、**~/.lmishellrc** の **use_exceptions** オプションを **True** に変更します。

```
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って 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)
```

接続オブジェクトの一時的なキャッシュを永続的に無効にするには、**~/.lmishellrc** の **use_cache** オプションを **False** に変更します。

```
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。この実行の後に LMIShell スクリプトを検査するには、**--interact** または **-i** の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。

21.4.2. CIMOM への接続

LMIShell では、同一システム上で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上の CIMOM に接続することができます。

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()** 関数を使っ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。パスワードが省略されると、LMIShell パスワードの入力をユーザーに要求します。この関数は **LMIConnection** オブジェクトを返します。

例21.1 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、インタラクティブのプロンプトで以下を入力します。

```
> 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()** 関数を使っ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21.2 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、インタラクティブのプロンプトで以下を入力します。

```
> c = connect("localhost")
>
```

CIMOM への接続の確認
connect() 関数は、接続が確立されないと **LMIConnection** オブジェクトまたは **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印刷します。

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で置き換えます。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21.3 CIMOM への接続の確認

例21.1「リモートの CIMOM への接続」 で作成された **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で以下を入力します。

```
> 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
>
```

21.4.3. 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一覧表示
利用可能なネームスペースすべてを一覧表示するには、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名で置き換えます。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印刷します。

利用可能なネームスペース一覧を取得するには、オブジェクト属性 **namespaces** にアクセスします。

```
object_name.namespaces
```

これは文字列の一覧を返します。

例21.4 利用可能なネームスペースの一覧表示

例21.1「リモートの CIMOM への接続」 で作成された **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で以下を入力します。

```
> c.root.print_namespaces()
cimv2
interop
PG_InterOp
PG_Internal
>
```

これらネームスペースの一覧を **root_namespaces** という名前の変数に割り当てるには、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で、*namespace_name* をアクセスするネームスペース名で置き換えます。これは **LMINamespace** オブジェクトを返します。

例21.5 ネームスペースオブジェクトへのアクセス

例21.1「リモートの CIMOM への接続」 で作成された **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、インタラクティブプロンプトで以下を入力します。

```
> ns = c.root.cimv2
>
```

21.4.4. 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やメソッド、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一覧表示することができ、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することができます。

利用可能なクラスの一覧表示
特定のネームスペースで利用可能なクラスすべてを一覧表示するには、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で置き換えます。このメソッドは、利用可能なクラスを標準出力に印刷します。

利用可能なクラス一覧を取得するには、**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一覧を返します。

例21.6 利用可能なクラスの一覧表示

例21.5「ネームスペースオブジェクトへのアクセス」 で作成された **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すべてを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> ns.print_classes()
CIM_CollectionInSystem
CIM_ConcreteIdentity
CIM_ControlledBy
CIM_DeviceSAPImplementation
CIM_MemberOfStatusCollection
...
>
```

これらクラスの一覧を **cimv2_classes** という名前の変数に割り当てるには、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、*class_name* をアクセスするクラス名に置き換えます。

例21.7 クラスオブジェクトへのアクセス

例21.5「ネームスペースオブジェクトへのアクセス」 で作成された **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てこれを **cls** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> cls = ns.LMI_IPNetworkConnection
>
```

クラスオブジェクトの検査

クラスオブジェクトはすべて、その名前と所属するネームスペースについての情報、および詳細なクラスのドキュメンテーションを保存しています。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ついての情報を取得するには、以下を使用します。

```
class_object.namespace
```

これは、ネームスペース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21.8 クラスオブジェクトの検査

例21.7「クラスオブジェクトへのアクセス」 で作成された **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で以下を入力します。

```
> 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一覧表示

特定のクラスオブジェクトの利用可能なメソッドすべてを一覧表示するには、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で置き換えます。このメソッドは、利用可能なメソッドを標準出力に印刷します。

利用可能なメソッド一覧を取得するには、**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一覧を返します。

例21.9 利用可能なメソッドの一覧表示

例21.7「クラスオブジェクトへのアクセス」 で作成された **cls** クラスオブジェクトを検査して利用可能なすべてのメソッドを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> cls.print_methods()
RequestStateChange
>
```

これらメソッドの一覧を **service_methods** という名前の変数に割り当てるには、以下を入力します。

```
> service_methods = cls.methods()
>
```

利用可能なプロパティーの一覧表示

特定のクラスオブジェクトの利用可能なプロパティーすべてを一覧表示するには、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で置き換えます。このメソッドは、利用可能なプロパティーを標準出力に印刷します。

利用可能なプロパティー一覧を取得するには、**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一覧を返します。

例21.10 利用可能なプロパティーの一覧表示

例21.7「クラスオブジェクトへのアクセス」 で作成された **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を一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> cls.print_properties()
RequestedState
HealthState
StatusDescriptions
TransitioningToState
Generation
...
>
```

これらクラスの一覧を **service_properties** という名前の変数に割り当てるには、以下を入力します。

```
> service_properties = cls.properties()
>
```

ValueMap プロパティーの一覧表示と閲覧

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 プロパティーすべてを一覧表示するには、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で置き換えます。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印刷します。

利用可能な ValueMap プロパティー一覧を取得するには、**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一覧を返します。

例21.11 ValueMap プロパティーの一覧表示

例21.7「クラスオブジェクトへのアクセス」 で作成された **cls** クラスオブジェクトを検査して利用可能なすべての ValueMap プロパティーを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> cls.print_valuemap_properties()
RequestedState
HealthState
TransitioningToState
DetailedStatus
OperationalStatus
...
>
```

これら ValueMap プロパティーの一覧を **service_valuemap_properties** という名前の変数に割り当てるには、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。

利用可能な定数値すべてを一覧表示するには、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印刷します。また、**values()** メソッドを使って利用可能な定数値の一覧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一覧を返します。

例21.12 ValueMap プロパティーへのアクセス

例21.11「ValueMap プロパティーの一覧表示」 では、**RequestedState** という名前の ValueMap プロパティーが出てきました。このプロパティーを検査して利用可能な定数値を一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> cls.RequestedStateValues.print_values()
Reset
NoChange
NotApplicable
Quiesce
Unknown
...
>
```

これら定数値の一覧を **requested_state_values** という名前の変数に割り当てるには、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 を定数値の名前で置き換えます。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判断するには、**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21.13 定数値へのアクセス

例21.12「ValueMap プロパティーへのアクセス」 では、**RequestedState** プロパティーが **Reset** という名前の定数値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で以下を入力します。

```
> cls.RequestedStateValues.Reset
11
```

```
> cls.RequestedStateValues.value("Reset")
11
>
```

この定数値の名前を判断するには、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。呼び出されたメソッドこのオブジェクトを実際に必要とする際にのみ LMIShell が CIMOM からオブジェクトを取り込むのはこのためです。**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。

21.4.5. 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、メソッドの一覧表示や呼び出し、ドキュメンテーション文字列の印刷、関連オブジェクト一覧の取得、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できます。

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すべてを取得するには、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で置き換えます。このメソッドは、**LMIInstance** オブジェクトの一覧を返します。

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() と **first_instance()** は、すべてのインスタンスを一覧表示したり最初のインスタンスを返したりするほか、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 を鍵と値のペアで構成される辞書で置き換えます。ここでの鍵はインスタンスプロパティーを表し、値はこれらプロパティーの必要な値を表します。

例21.14 インスタンスへのアクセス

例21.7「クラスオブジェクトへのアクセス」 で作成された **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 **device** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> device = cls.first_instance({"ElementName": "eth0"})
>
```

インスタンスの検査

インスタンスオブジェクトはすべて、そのクラス名と所属するネームスペースについての情報、およびプロパティーと値についての詳細なドキュメンテーションを保存しています。さらに、インスタンスオブジェクトを使うと、一意の ID オブジェクトを検索することができます。

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ついての情報を取得するには、以下を使用します。

```
instance_object.namespace
```

これは、ネームスペース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21.15 インスタンスの検査

例21.14「インスタンスへのアクセス」 で作成された **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で以下を入力します。

```
> 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することができます。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名で、*properties* を鍵と値のペアで構成される辞書で置き換えます。ここでの鍵はインスタンスプロパティーを表し、値はプロパティーの値を表します。このメソッドは、**LMIInstance** オブジェクトを返します。

例21.16 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21.5「ネームスペースオブジェクトへのアクセス」 で作成された **ns** ネームスペースオブジェクトを使用して、**pegasus** という名前のシステムグループと **lmishell-user** という名前のユーザー用にこれら 2 つのインスタンスを作成し、それらを **group** および **user** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> 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** クラスを使って **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21.17 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21.5「ネームスペースオブジェクトへのアクセス」 で作成された **ns** ネームスペースオブジェクトを使用して、**lmishell-user** という名前のユーザー用に **LMI_Account** クラスのインスタンスを作成し、それを **user** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> 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一覧表示とアクセス

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すべてを一覧表示するには、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で置き換えます。このメソッドは、利用可能なプロパティーを標準出力に印刷します。

利用可能なプロパティー一覧を取得するには、**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一覧を返します。

例21.18 利用可能なプロパティーの一覧表示

例21.14「インスタンスへのアクセス」 で作成された **device** インスタン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を一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> device.print_properties()
RequestedState
HealthState
StatusDescriptions
TransitioningToState
```

```
Generation
...
>
```

これらのプロパティーの一覧を **device_properties** という名前の変数に割り当てるには、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構成される 3 項目タプルを返します。

例21.19 個別プロパティーへのアクセス

例21.14「インスタンスへのアクセス」 で作成された **device** インスタンスオブジェクトを検査して、**SystemName** という名前のプロパティーの値を表示するには、インタラクティブプロンプトで以下を入力します。

```
> device.SystemName
u'server.example.com'
>
```

利用可能なメソッドの一覧表示と使用

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すべてを一覧表示するには、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で置き換えます。このメソッドは、利用可能なメソッドを標準出力に印刷します。

利用可能なメソッド一覧を取得するには、**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一覧を返します。

例21.20 利用可能なメソッドの一覧表示

例21.14「インスタンスへのアクセス」 で作成された **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すべてを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> device.print_methods()
RequestStateChange
>
```

これらメソッドの一覧を **network_device_methods** という名前の変数に割り当てるには、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で、*method_name* を呼び出すメソッド名で、*parameter* を設定するパラメーター名で、*value* をこのパラメーターの値で置き換えます。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構成される 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21.21 メソッドの使用

PG_ComputerSystem クラスは、システムを表します。例21.5「ネームスペースオブジェクトへのアクセス」 で作成された **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 **sys** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> 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 **acc** という名前の変数に割り当てるには、以下を入力します。

```
> acc = ns.LMI_AccountManagementService.first_instance()
>
```

システム内で **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、同時メソッド呼び出しをサポートします。このメソッドを使用すると、LMIShell は対応するジョブオブジェクトがその状態を 「finished」 に変更するまで待機し、その後にこのジョブの戻り値パラメーターを返します。特定のメソッドが以下のいずれかのクラスのオブジェクトを返す場合、LMIShell は同時メソッド呼び出しを実行できます。

- **LMI_StorageJob**
- **LMI_SoftwareInstallationJob**
- **LMI_NetworkJob**

LMIShell はまず、待機メソッドに indications 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。

同時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で、*method_name* を呼び出すメソッド名で、*parameter* を設定するパラメーター名で、*value* をこのパラメーターの値で置き換えます。同時メソッドではすべて、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構成される 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う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 パラメーターの一覧表示と閲覧

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。

特定メソッドの利用可能な ValueMap パラメーターすべてを一覧表示するには、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で置き換え、*method_name* を検査するメソッド名で置き換えます。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印刷します。

利用可能な ValueMap パラメーター一覧を取得するには、**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一覧を返します。

例21.22 ValueMap パラメーターの一覧表示

例21.21「メソッドの使用」 で作成された **acc** インスタンスオブジェクトを検査して **CreateAccount()** メソッドの利用可能なすべての ValueMap パラメーターを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> acc.CreateAccount.print_valuemap_parameters()
CreateAccount
>
```

これら ValueMap パラメーターの一覧を **create_account_parameters** という名前の変数に割り当てるには、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。

利用可能な定数値すべてを一覧表示するには、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印刷します。また、**values()** メソッドを使って利用可能な定数値の一覧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一覧を返します。

例21.23 ValueMap パラメーターへのアクセス

例21.22「 ValueMap パラメーターの一覧表示」 では、**CreateAccount** という名前の ValueMap パラメーターが出てきました。このパラメーターを検査して利用可能な定数値を一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> acc.CreateAccount.CreateAccountValues.print_values()
Operationunsupported
Failed
Unabletosetpasswordusercreated
```

```
Unabletocreatehomedirectoryusercreatedandpasswordset
Operationcompletedsuccessfully
>
```

これら定数値の一覧を **create_account_values** という名前の変数に割り当てるには、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 を定数値の名前で置き換えます。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判断するには、**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21.24 定数値へのアクセス

例21.23「 ValueMap パラメーターへのアクセス」 では、**CreateAccount** ValueMap パラメーターが **Failed** という名前の定数値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で以下を入力します。

```
> acc.CreateAccount.CreateAccountValues.Failed
2
> acc.CreateAccount.CreateAccountValues.value("Failed")
2
>
```

この定数値の名前を判断するには、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、これらが LMIShell のオブジェクトとの作業中に変更されると、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で置き換えます。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構成される 3 項目タプルを返します。

例21.25 インスタンスオブジェクトの更新

例21.14「インスタンスへのアクセス」 で作成された **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で以下を入力します。

```
> 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で置き換えます。このメソッドは、オブジェクトの MOF 表現を標準出力に印刷します。

例21.26 MOF 表現の表示

例21.14「インスタンスへのアクセス」 で作成された **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で以下を入力します。

```
> device.tomof()
instance of LMI_IPNetworkConnection {
        RequestedState = 12;
        HealthState = NULL;
        StatusDescriptions = NULL;
        TransitioningToState = 12;
...
```

21.4.6. 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** オブジェクトによって識別されます。利用可能なすべてのインスタンス名オブジェクト一覧を取得するには、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で置き換えます。このメソッドは、**LMIInstanceName** オブジェクトの一覧を返します。

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、すべてのインスタンス名オブジェクトを一覧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、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 を鍵と値のペアで構成される辞書で置き換えます。ここでの鍵は鍵プロパティーを表し、値はこれら鍵プロパティーの必要な値を表します。

例21.27 インスタンス名へのアクセス

例21.7「クラスオブジェクトへのアクセス」 で作成された **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 **device_name** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> device_name = cls.first_instance_name({"Name": "eth0"})
>
```

インスタンス名の検査

インスタンス名オブジェクトはすべて、そのクラス名と所属するネームスペースについての情報を保存しています。

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ついての情報を取得するには、以下を使用します。

```
instance_name_object.namespace
```

これは、ネームスペースを表す文字列を返します。

例21.28 インスタンス名の検査

例21.27「インスタンス名へのアクセス」 で作成された **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で以下を入力します。

```
> device_name.classname
```

```
u'LMI_IPNetworkConnection'
> device_name.namespace
'root/cimv2'
>
```

新規インスタンス名の作成

LMIShell では、リモートオブジェクトのプライマリーキーすべてが分かっている場合、新規のラップ済み **CIMInstanceName** オブジェクトが作成できます。すると、このインスタンス名オブジェクトを使ってインスタンスオブジェクト全体を取得することができるようになります。

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で、*key_properties* を鍵と値のペアで構成される辞書で置き換えます。ここでの鍵は鍵プロパティーを表し、値は鍵プロパティーの値を表します。このメソッドは、**LMIInstanceName** オブジェクトを返します。

例21.29 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21.5「ネームスペースオブジェクトへのアクセス」 で作成された **ns** ネームスペースオブジェクトを使用して、管理システム上の **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で以下を入力します。

```
> instance_name = ns.LMI_Account.new_instance_name({
...     "CreationClassName" : "LMI_Account",
...     "Name" : "lmishell-user",
...     "SystemCreationClassName" : "PG_ComputerSystem",
...     "SystemName" : "server"})
>
```

鍵プロパティーの一覧表示とアクセス

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すべてを一覧表示するには、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で置き換えます。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印刷します。

利用可能な鍵プロパティー一覧を取得するには、**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一覧を返します。

例21.30 利用可能な鍵プロパティーの一覧表示

例21.27「インスタンス名へのアクセス」 で作成された **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すべての鍵プロパティーを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> device_name.print_key_properties()
CreationClassName
SystemName
Name
SystemCreationClassName
>
```

これらの鍵プロパティーの一覧を **device_name_properties** という名前の変数に割り当てるには、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21.31 個別の鍵プロパティーへのアクセス

例21.27「インスタンス名へのアクセス」 で作成された **device_name** インスタンス名オブジェクトを検査して、**SystemName** という名前の鍵プロパティーの値を表示するには、インタラクティブプロンプトで以下を入力します。

```
> device_name.SystemName
u'server.example.com'
>
```

インスタンス名をインスタンスに変換する

インスタンス名はインスタンスに変換することが可能です。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21.32 インスタンス名をインスタンスに変換する

例21.27「インスタンス名へのアクセス」 で作成された **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、**device** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> device = device_name.to_instance()
>
```

21.4.7. 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全一覧を取得するには、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** — 返された各オブジェクトは、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** — 返された各オブジェクトは、このクラスのインスタンスかそのサブクラスの 1 つのインスタンスである必要があります。または、このクラスもしく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** — 返された各オブジェクトは、ソースオブジェクトが特定の役割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。ソース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。
- **ResultRole** — 返された各オブジェクトは、返されたオブジェクトが特定の役割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。返された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たすべてのプロパティー上の修飾子を含む) を QUALIFIER 要素として含めるかどうかを示すブール値。デフォルト値は、**False** です。

- **IncludeClassOrigin** — 返された各オブジェクトですべての適切な要素に 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** が空の場合、返されるオブジェクトにはプロパティーは含まれません。**None** の場合は、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21.33 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 21.5「ネームスペースオブジェクトへのアクセス」 で作成された **ns** ネームスペースオブジェクトを使用して、**/dev/vda** という名前のブロックデバイス用に **LMI_StorageExtent** クラスのインスタンスを作成し、これを **vda** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> vda = ns.LMI_StorageExtent.first_instance({
...     "DeviceID" : "/dev/vda"})
>
```

このブロックデバイス上のディスクパーティション全一覧を取得して、これを **vda_partitions** という名前の変数に割り当てるには、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の利用可能なインスタンス名一覧を取得するには、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の最初の関連するインスタンス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** — 返された各名前はオブジェクトを識別します。このオブジェクトは、このクラスもしく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** — 返された各名前はオブジェクトを識別します。このオブジェクトは、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 1 つである必要があります。デフォルト値は、**None** です。

- **Role** — 返された各名前は、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役割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。ソース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。

- **ResultRole** — 返された各名前は、オブジェクトを識別します。このオブジェクトは、返された名前オブジェクトが特定の役割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。返された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21.34 関連するインスタンス名へのアクセス

例21.33「関連するインスタンスへのアクセス」 で作成された **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一覧を取得し、**vda_partitions** という名前の変数に割り当てるには、以下を入力します。

```
> vda_partitions = vda.associator_names(ResultClass="LMI_DiskPartition")
>
```

21.4.8. 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一覧を取得するには、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** — 返された各オブジェクトは、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 1 つである必要があります。デフォルト値は、**None** です。

- **Role** — 返された各オブジェクトは、このパラメーターの値に合致する名前を持った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** — 応答に各オブジェクト (オブジェクトおよび返されたすべてのプロパティー上の修飾子を含む) を QUALIFIER 要素として含めるかどうかを示すブール値。デフォルト値は、**False** です。

- **IncludeClassOrigin** — 返された各オブジェクトですべての適切な要素に 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** が空の場合、返されるオブジェクトにはプロパティーは含まれません。**None** の場合は、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21.35 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ー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21.5「ネームスペースオブジェクトへのアクセス」 で作成された **ns** ネームスペースオブジェクトを使用して、eth0 という名前のネットワークインターフェースデバイス用に **LMI_LANEndpoint** クラスのインスタンスを作成し、これを **lan_endpoint** という名前の変数に割り当てるには、インタラクティブプロンプトで以下を入力します。

```
> 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 **bind** という名前の変数に割り当てるには、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ースデバイスの 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一覧を取得するには、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** — 返された各オブジェクト名は、このクラスのインスタンスかそのサブクラスの 1 つのインスタンスを識別します。または、このクラスまたはそのサブクラスの 1 つを識別します。デフォルト値は、**None** です。
- **Role** — 返された各オブジェクトは、このパラメーターの値に合致する名前を持った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21.36 関連付けインスタンス名へのアクセス

例21.35「関連付けインスタンスへのアクセス」 で作成された **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 **bind** という名前の変数に割り当てるには、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ースデバイスの 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
>
```

21.4.9. 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することが可能です。

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うこともできます。

```
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、*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すべて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21.37 Indication のサブスクライブ

例21.1「リモートの CIMOM への接続」 で作成された **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で以下を入力します。

```
> 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 の一覧表示

利用可能なサブスクライブしている indication すべてを一覧表示するには、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で置き換えます。このメソッドは、サブスクライブしている indication を標準出力に印刷します。

サブスクライブしている indication の一覧を取得するには、**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一覧を返します。

例21.38 サブスクライブしている indication の一覧表示

例21.1「リモートの CIMOM への接続」 で作成された **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 すべてを一覧表示するには、インタラクティブプロンプトで以下を入力します。

```
> c.print_subscribed_indications()
>
```

これらの indication の一覧を **indications** という名前の変数に割り当てるには、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すべて自動的に削除されます。個別のサブスクリプションをこれより前に削除するには、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で置き換え、*indication_name* を削除する indication の名前で置き換えます。

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21.39 Indication のサブスクライブ解除

例21.1「リモートの CIMOM への接続」 で作成された **c** 接続オブジェクトを使用して、例21.37「Indication のサブスクライブ」 で作成された 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で以下を入力します。

```
> c.unsubscribe_indication('cpu')
LMIReturnValue(rval=True, rparams=NocaseDict({}), errorstr='')
>
```

Indication ハンドラーの実装

subscribe_indication() メソッドを使うと、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一覧が含まれます。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で置き換えられます。ランダムな文字列を使用するには、indication リスナーを最初に開始してその次に indication をサブスクライブします。そうすることで、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*** の値が含まれます。

例21.40 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)
```

21.4.10. 使用例

本セクションでは、OpenLMI パッケージと配布される様々な CIM プロバイダーの例を示します。ここではすべての例で、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、*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。

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 プロバイダーを使って利用可能なシステムサービスを一覧表示する方法、およびそれらを開始、停止、有効化、および無効化する方法を示しています。

例21.41 利用可能なサービスの一覧表示

管理システム上で利用可能なすべてのサービスと、そのサービスが開始されているか (**TRUE**) または停止されているか (**FALSE**) についての情報、およびステータス文字列を一覧表示するには、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一覧表示するには、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 サービスについての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21.42 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21.43 サービスの有効、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 プロバイダーを使って特定のポート番号に関連付けられた IP アドレスを一覧表示する方法、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21.44 特定のポート番号に関連付けられている IP アドレスの一覧表示

eth0 ネットワークインターフェースに関連付けられた IP アドレスすべてを一覧表示するには、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、ある **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.DefaultGatewa
y:
        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 サーバーの一覧を取得するには、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、ある **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()** 関数を使って 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_NetworkRemoteAccessAvailableToEle
ment", ResultClass="LMI_NetworkRemoteServiceAccessPoint"):
            if rsap.AccessContext ==
ns.LMI_NetworkRemoteServiceAccessPoint.AccessContextValues.DNSServer:
                dnsservers.add(rsap.AccessInfo)
print "DNS:", ", ".join(dnsservers)
```

例21.45 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ースで静的 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_CreateIPSetting()** メソッドを **LMI_IPNetworkConnectionCapabilities** のインスタンス上で呼び出すことで、新規設定を作成します。これは、**LMI_IPNetworkConnectionElementCapabilities** で **LMI_IPNetworkConnection** に関連付けられています。また、**push()** メソッドを使って設定の修正も行います。

例21.46 接続のアクティブ化

設定をネットワークインターフェースに適用するには、**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。
- 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 プロバイダーを使ったボリュームグループおよび論理ボリュームの作成方法、ファ

イルシステムの構築およびマウント方法、システムが認知するブロックデバイスの一覧表示方法を説明します。

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21.47 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21.48 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21.49 ファイルシステムの作成

例21.48「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.FileSystemTypeVal
ues.EXT3,
        InExtents=[lv])
```

例21.50 ファイルシステムのマウント

例21.49「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21.51 ブロックデバイスの一覧表示

システムが認識するブロックデバイスすべてを一覧表示するには、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 プロバイダーを使って 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ついての情報を取得する方法を説明しています。

例21.52 CPU 情報の表示

CPU 名や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21.53 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的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21.54 シャーシ情報の表示

製造元やモデルなどのマシンについての基本的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21.55 PCI デバイスの一覧表示

システムが認識する PCI デバイスすべてを一覧表示するには、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
```

21.5. 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 モジュール上に構築されます。OpenLMI スクリプトのプロジェクトは、OpenLMI プロバイダーに数多くのインターフェース用の Python ライブラリーを提供します。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** と共に配布されます。

OpenLMI スクリプトをシステムにインストールするには、シェルプロンプトで以下を入力します。

```
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一覧は、Python website を参照してください。OpenLMI スクリプトについての詳細は、official OpenLMI Scripts documentation を参照してください。

21.6. 関連資料

OpenLMI およびシステム管理全般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

- 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。

オンラインのドキュメント

- 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ー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 ストレージ管理ガイド — Red Hat Enterprise Linux 7 の 『ストレージ管理ガイド』 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について説明しています。
- Red Hat Enterprise Linux 7 電力管理ガイド — Red Hat Enterprise Linux 7 の 『電力管理ガイド』 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 ドメイン ID、認証、およびポリシーガイド』 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のメインのユーザー資料となります。

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ースとコマンドラインを使ったシステムユーザーとグループの管理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って、コマンドラインでパッケージを検索、インストール、更新、アンインストールする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** コマンドを使っ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* では、SSH サーバーの設定方法や、**ssh**、**scp**、および **sftp** クライアントユーティリティーを使ってこのサーバーにアクセスする方法が説明されています。

第22章 ログファイルの表示と管理

ログファイル とは、システムで実行しているカーネル、サービス、アプリケーションなどのシステムに関するメッセージを格納しているファイルです。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る項目を説明します。

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。また、これらをインデックス化して、ユーザーが利用できるようにします。ネイティブのジャーナルファイル形式は構造化およびインデックス付きバイナリファイルで、これは検索を改善し、より速い操作を提供します。また、タイムスタンプやユーザー ID といったメタデータ情報も保存します。**journald** が生成するログファイルはデフォルトで永続的なものではなく、メモリーもしくは **/run/log/journal/** ディレクトリーの小型のリングバッファーに保存されるだけです。ログ記録されるデータ量は空きメモリーの量によります。容量の限界に達すると、一番古いエントリーが削除されます。しかし、この設定は変更可能です。「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については、「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。しかし、デフォルトでは **rsyslog** は **imjournal** 入力モジュール経由で実際の統合を行うので、上記のソケットは使用されません。また、**omjournal** モジュールを使って **rsyslogd** から **journald** に逆方向でデータを移動することもできます。詳細については、「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)。

22.1. 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一覧は **/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** や **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したが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。

22.2. 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* の部分で構成される *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。

22.2.1. 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 1 行内で、1 つ以上の空白かタブでそれらを区切って定義します。

rsyslog は、選択されたプロパティーにしたがって 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、**ファシリティー/優先度ベース**、**プロパティーベース**、さらに **式ベース** 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 メッセージのフィルター操作で最も使用されよく知られているのは、facility/priority ベースのフィルターを使用する方法です。これは、*facility* と *priority* の 2 つの条件をベースにして syslog メッセージをフィルター処理します。これらの条件は、ドットで区切られます。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
```

ここでは、

- *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)、**uucp** (8)、**cron** (9)、**authpriv** (10)、**ftp** (11)、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。

複数のファシリティーや優先度を定義するには、コンマ (**,**) でそれらを区切ります。複数のセレクターを 1 行内で定義するには、セミコロン (**;**) でそれらを区切ります。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22.1 ファシリティー/優先度ベースのフィルター

以下は簡単なファシリティー/優先度ベースのフィルターの例です。これは、**/etc/rsyslog.conf** で指定できます。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.*
```

crit およびそれ以上の優先度があるメール 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22.1「プロパティベースの比較処理　」 で一覧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"
```

ここでは、

- *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22.1「プロパティベースの比較処理　」 に一覧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22.1 プロパティベースの比較処理

比較処理	説明
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。大文字と小文字を区別しない比較を実行するには、***contains_i*** を使用します。

比較処理	説明
isequal	用意された文字列をプロパティーで提供されたテキストすべてに対して比較します。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。
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。大文字と小文字を区別しない比較を実行するには、***startswith_i*** を使用します。
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22.2 プロパティーベースのフィルター

以下は、プロパティーベースのフィルター例です。これは、**/etc/rsyslog.conf** で指定できます。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
```

ここでは、

- *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** 演算子を使うことで、単一フィルター内に複数の式を指定できます。
- *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** キーワードを使っ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22.3「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** フィルターを使うことができます。その一方、ここでは **property-based** フィルターは推奨されません。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22.3 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については、「オンラインのドキュメント」 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。

22.2.2. 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、*PATH* はターゲットファイルのパスを示します。

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構成されています。

```
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、**静的** でも **動的** でも構いません。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。テンプレートの詳細については、「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。

ネットワークを使った syslog メッセージの送信

rsyslog を使用すると、ネットワークを使って syslog メッセージを送受信できます。この機能により、1 台のマシン上で複数ホストの 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]
```

ここでは、

- 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22.4 ネットワークを使った 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** プロトコルを使ってメッセージを "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** プロトコルを使って **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われます。これは、ログファイルのローテーションに非常に便利なものです (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ついての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
```

ここでは、

- *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22.5 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22.7「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 は選択した syslog メッセージ用のプログラムを実行可能とし、**system()** 呼び出しを使用してシェル内でプログラムを実行します。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 caret 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22.6 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考慮してください。

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]
```

ここでは、

- *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** plug-in)。
- *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** データベースと **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確実に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 モジュールに関する詳細は、「Rsyslog モジュールの使用」 を参照してください。

別の方法として、**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ー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には、チルダ文字 (~) を使用します。

```
FILTER ~
```

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ーするために使用されます。破棄しなければログファイルを満たしてしまう繰り返されるメッセージを削除したい場合に、これは効果的です。破棄のアクションの結果は、設定ファイルのどこで指定されているかに左右されます。最善の結果を得るには、これらのアクションをアクションリストの最上部に置きます。メッセージは一旦破棄されると、設定ファイルの後ろ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、以下のルールはすべての cron syslog メッセージを破棄します。

```
cron.* ~
```

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することができます。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22.7 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、アクションにセミコロン (**;**) を末尾に付けてからテンプレートの名前を指定します。テンプレートについての詳細は、「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、それ以外の場合は無視されます。つまり、テンプレート定義は常に **/etc/rsyslog.conf** でルール定義の前にある必要があります。

22.2.3. 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** を使ってユーザーのニーズに応じて修正とフォーマットができます。テンプレートを作成するには、**/etc/rsyslog.conf** で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"])
```

ここでは、

- ***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われます。***TEMPLATE_NAME*** は固有のものである必要があります。
- ***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 ***sql*** と ***stdsql*** で、テキストを SQL クエリとして、またはテキストを JSON 処理に最適な形にフォーマットする JSON としてフォーマットするために使用されます。 ***casesensitive*** はプロパティ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*** 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。

詳細については、「アクション」 の項「『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

テンプレートを使って、動的なファイル名を生成することができます。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 メッセージ用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** 内で、クエスチョンマーク (**?**) をアクション定義に使って、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* を使って 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]%
```

ここでは、

- *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説明の一覧は、**rsyslog.conf(5)** man ページの **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** オプションのような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説明の一覧は、**rsyslog.conf(5)** 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いくつか以下に示します。

- 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%
  ```

- 以下のプロパティは、syslog メッセージを受信して 『RFC 3999』 日付基準にしたが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22.8「詳細な syslog メッセージのテンプレート」 は、syslog メッセージをフォーマット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、メッセージが受信された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22.8 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22.9「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22.9 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22.10「データベースフォーマットをしたメッセージのテンプレート」 は、syslog メッセージをフォーマット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。テンプレートオプションとして指定されている、テンプレートの最後にある ***sql*** オプションの使用に注目してください。これは、データベースライターに対してメッセージを MySQL **SQL** クエリーとしてフォーマットするように指示します。

例22.10 データベースフォーマットをし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。以下の一覧では、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 に類似したモダンスタイルのログファイルフォーマットですが、高精度のタイムスタンプとタイムゾーン情報を備えています。

```
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

精度の低いタイムスタンプを持つ旧式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を備えた転送フォーマット。

```
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持つ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")
}
```

22.2.4. 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”)
```

このディレクティブ用に定義されたデフォルトのサイズ (**10,000** メッセージ) は、別の値を指定することで上書きされます (上記の例を参照)。

/etc/rsyslog.conf 設定ファイル内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、アクションやキュー、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一覧は、「オンラインのドキュメント」 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。

22.2.5. 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、ログファイルすべて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、ログファイルは毎週交代され、交代されたログファイルは 4 週間保管されます。交代済みログファイルはすべて、**gzip** で **.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下に配置することもできます。ただし、**/etc/logrotate.d/** ディレクトリー内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。

設定ファイルが **/etc/logrotate.d/** ディレクトリーに配置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、交代された **/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一覧です。

- ***weekly*** — ログファイルの週毎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*** — 交代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回数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、交代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一覧は、**logrotate(5)** man ページを参照してください。

22.3. 新規設定フォーマットの使用

rsyslog パッケージの Red Hat Enterprise Linux 7 に対してデフォルトでインストールされる **rsyslog** バージョン 7 では、新しい設定構文が導入されました。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22.3「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()** 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、主に構造化が強化されているという点で異なります。パラメーターは、入力、アクション、テンプレート、モジュールロード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。また、パフォーマンスが大幅に増加する利点があります。一部の機能は両方の構文で公開され、一部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情報は、「オンラインのドキュメント」 を参照してください。

22.3.1. ルールセット

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構成される **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、ルールはすべて、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することができます。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することができます。**/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()** 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
      …
}
```

rulesetname を、使用する ruleset の識別子で置き換えます。ruleset 名は **RSYSLOG_** で開始することはできません。このネームスペースが **rsyslog** の使用のために確保されているためです。そして、メッセージに他の ruleset が割り当てられていない場合に実行されるデフォルトのルール一式を

RSYSLOG_DefaultRuleset が定義します。*rule* と *rule2* では、上記で説明したフィルター-アクションのフォーマット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する ruleset 名で置き換えます。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って ruleset を定義することもできます。詳細は、「オンラインのドキュメント」 を参照してください。

例22.11 ruleset 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。以下を **/etc/rsyslog.conf** に追加します。

```
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");
```

上記の例の ruleset は、2 つのポートからのリモート入力のログの行き先を定義しています。ポート **601** の場合、メッセージはファシリティーにしたがって分けられます。そして、TCP 入力が有効になり、ruleset 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。

22.3.2. 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 で存在しますが、バージョン 7 では存在しません。また、syslogd スタイルのコマンドラインオプションは非推奨となり、このコマンドラインオプションを使った **rsyslog** の設定も避けるべきです。ただし、複数のテンプレートとディレクティブを使って syslogd のような動作をエミュレートするために **rsyslogd** を設定することができます。

rsyslogd オプションについての詳細情報は、**rsyslogd(8)** man ページを参照してください。

22.4. RSYSLOG でのキュー (QUEUE) を使った操作

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することができます。**rsyslog** 内のデータフローは、以下のようになりま

す。

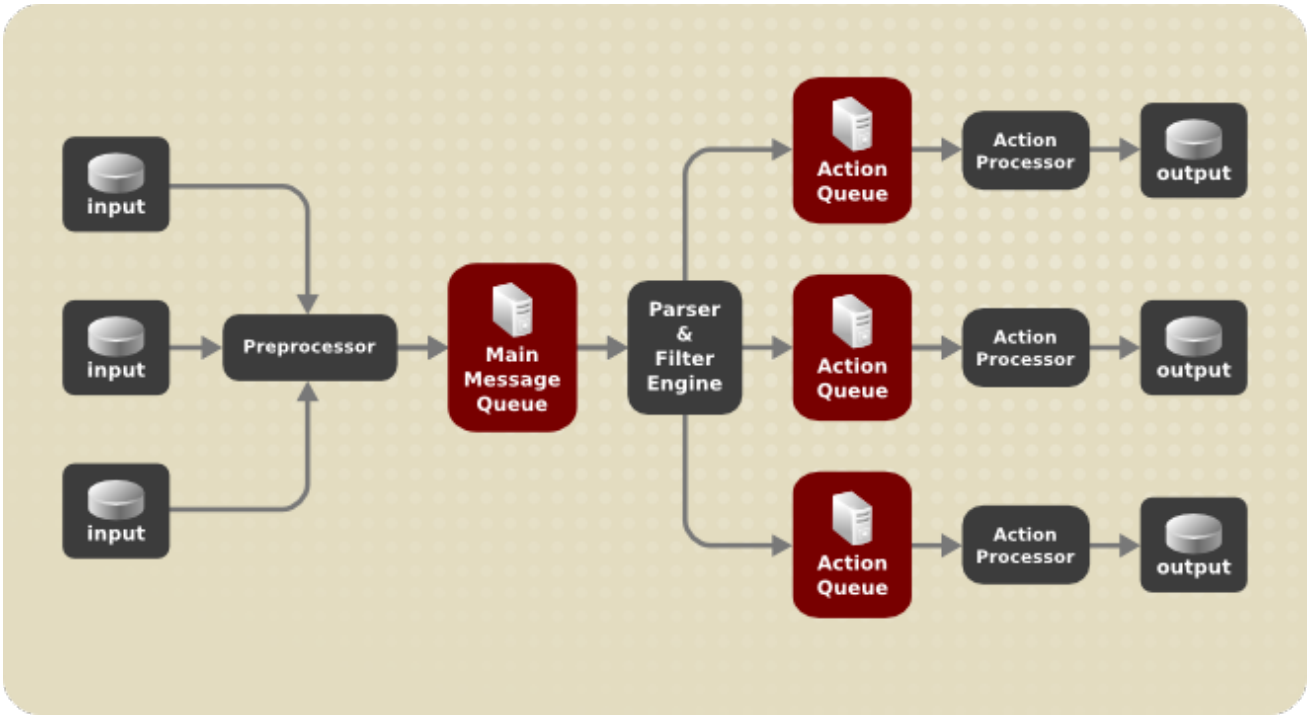


図22.1 Rsyslog 内のメッセージフロー

rsyslog はメッセージを受信すると常に、これをプリプロセッサーに渡します。そして、メインメッセージキュー (*main message queue*) に配置します。メッセージはそこでデキューされ、*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。アクションはそれぞれ、独自のアクションキューを持っています。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場合は、アクションプロセッサーがアクションキューに通知し、すると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** ではキューは 2 カ所に位置します。1 つは、rule processor の前に単一の **メインメッセージキュー** として。もう 1 つは、様々なタイプの出力アクションの前に **アクションキュー** として。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。

これら以外には、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I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、いっぱいでなくな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。

22.4.1. キューの定義

メッセージが保存される場所によって、キューにはいくつかタイプがあります。最も使われるのは、**direct**、**in-memory**、**disk**、および **disk-assisted in-memory** のキューです。これらのうちのいずれかを、メインメッセージキューおよびアクションキューに選択できます。以下を **/etc/rsyslog.conf** に追加します。

```
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 queues に設定されます。

ダイレクトキュー (Direct Queues)
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。キューの使用を避けるには、以下を使用します。

```
object(queue.type="Direct")
```

object を **main_queue**、 **action**、**ruleset** のいずれかに置き換え、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それぞれに使用します。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 (Disk Queues)
ディスクキューは、厳密にメッセージをハードドライブに保存します。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。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それぞれに使用します。ディスクキューは、デフォルトサイズの 10 MBで分けて書き込まれます。こ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、各ファイルごとにこれが増加します。

インメモリーキュー (In-memory Queues)
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。コンピュータの電源切断、投入または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 キューは稀に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 キューを使ってください。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。

インメモリーキューの設定には、以下の構文を使用します。

```
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それぞれに使用します。

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。そのためには、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補助のファイル名を定義します。このキューは **ディスク補助** 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なものです。

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** に置き換えて、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それぞれに使用します。*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され、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。不必要なディスク書き込みを最小限に抑えるため、ウォーターマークを正しく設定します。ただし、ディスクファイルへの書き込みは長いので、メッセージバースト用のメモリースペースを残してください。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。ハイウォーターマークとキューの全体サイズの差が、メッセージバースト用に確保されたスペアメモリーバッファーです。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22.12 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。この場合、送信に失敗したメッセージはサーバーが再び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。

手順22.1 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。

- **/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"
  arget="example.com" Port="6514" Protocol="tcp")
  ```

  ここで、

  - **queue.type** は LinkedList インメモリーキューを有効化し、
  - **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** プレフィックスを持つ **/var/lib/rsyslog/** ディレクトリーで作成され、

- **action.resumeRetryCount= “-1”** 設定は、サーバーの応答がない場合に再接続しようとしたときに rsyslog がメッセージをドロップすることを防止し、
- rsyslog がシャットダウンしたら、有効化された **queue.saveonshutdown** がインメモリーデータを保存し、
- 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が到達不能な場合にメッセージをメモリーに保持します。ディスク上のファイルは、rsyslog で設定済みメモリーキュー領域がなくなった場合、または rsyslog をシャットダウンする必要がある場合 (システムパフォーマンスが向上します) にのみ作成されます。

手順22.2 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。

- 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")
```

22.4.2. 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。

手順22.3 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。例を以下に示します。

```
~]# 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。

22.4.3. 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っ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することができます。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。「オンラインのドキュメント」 を参照してください。これらの設定のいくつかは一般的に使用されており、また、worker thread management のようにキューの動作を密接に管理する、高度なユーザー用に確保されているものもあります。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することができます。

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それぞれに使用します。*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、デキュー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することができます。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")
```

メインメッセージキューにこのオプションを使用する場合は *object* を **MainMsg** に、アクションキューに使用する場合は **Action** に置き換えます。ダイレクトキューでは、メッセージはプロデューサーからコンシューマーに直接かつ即座に渡されます。ここでの *number* は、破棄プロセスを開始する際のキューにあるメッセージ数です。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 を各プライオリティに対応する以下の数字のどれかに置き換えます: **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 **rsyslog** を設定することができます。このオプションを使うと、たとえば処理をオフピーク時に移すことが可能です。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。分数を使わずに、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届くと、3 つ目のワーカースレッドが起動する、といったようになります。しかし、多くのワーカースレッドが並列して実行されると効果的でなくなるので、以下を使用してこの最大数を制限できます。

```
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されると、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は、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することができます。

```
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。

22.4.4. 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()** オブジェクト内部で定義されます。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。例については、「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22.13 アクションキューの定義

出力アクションで、最大 10,000 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
     )
```

パラメーターが複数の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設定する最後の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以下の行を追加します。

```
global(workDirectory="/directory")
```

例22.14 新規構文を使用した単一サーバーへの転送

以下の例は 手順22.1「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"
     )
```

ここで、

- **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、先行するグローバルな **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** プレフィックスで作成されます。
- **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。
- **queue.saveOnShutdown="on"** が有効な場合に、rsyslog がシャットダウンすると、インメモリーデータが保存されます。
- 最後の行は、受信されたすべてのメッセージをロギングサーバーに転送します。ポートの指定はオプションです。

22.5. 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** 設定については、例22.12「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。Red Hat Enterprise Linux 7 では、rsyslog がデフォルトでインストールされます。確実にインストールする必要がある場合は、**root** で以下のコマンドを入力します。

```
~]# yum install rsyslog
```

syslog トラフィックのデフォルトのプロトコルとポートは、**/etc/services** ファイルで示されているように **UDP** と **514** です。ただし、デフォルトで **rsyslog** はポート **514** で **TCP** を使用するよう設定されます。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 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設定されます。

```
~]# 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** で許可するよう設定されたプロトコルを確認するには、以下のコマンドを入力します。

```
~]# semanage port -l | grep 514出力省略
rsh_port_t                     tcp      514
```

```
syslogd_port_t                 tcp      6514, 601
syslogd_port_t                 udp      514, 6514, 601
```

SELinux の詳細については、『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。

手順22.4 ポートで rsyslog トラフィックを許可するよう SELinux を設定

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場合は、以下のようになります。

1. ```
   ~]# 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、変更を反映するために **rsyslog** を再起動します。

   ```
   ~]# 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 コマンドの詳細については、**semanage-port(8)** man ページを参照してください。

手順22.5 firewalld の設定

firewalld が受信 **rsyslog** トラフィックを許可するよう設定します。たとえば、ポート **10514** で **TCP** を許可するには、以下の手順を実行します。

1. ```
   ~]# firewall-cmd --zone=zone --add-port=10514/tcp
   success
   ```

   ここで、*zone* は使用するインターフェースのゾーンです。これらの変更は次回のシステム起動後に維持されないことに注意してください。ファイアウォールに永久的な変更を行うには、**--permanent** オプションを追加してコマンドを繰り返します。**firewalld** でのポートの開閉の詳細については、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。

2. 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:
```

手順22.6 リモートログメッセージを受信してソートするよう rsyslog を設定

1. テキストエディターで **/etc/rsyslog.conf** ファイルを開き、以下の手順を実行します。

   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のログファイルを受信し、保管するためのログサーバーの設定が完了しました。

22.5.1. 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 には複数の異なるテンプレートスタイルがあります。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")
```

22.6. RSYSLOG モジュールの使用

モジューラーの設計上、**rsyslog** は、追加の機能性を装備する各種 *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 **入力モジュール** 参照) または出力 (以下の **出力モジュール** 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** を有効化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、**/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、様々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** や **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** や **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緊密に協力します。テンプレートに関する詳細情報は、「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** や **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。

すべての利用可能なモジュールとそれらの詳しい説明を含む総合一覧は、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。

22.6.1. 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** を使って、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することができます。**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です (「フィルター」 を参照)。

例22.15 テキストファイルのインポート

Apache HTTP サーバーはテキスト形式でログファイルを作成します。**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** モジュールを使っ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:")
```

22.6.2. 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**、または **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22.16 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、サーバー名やデータベース名などの追加情報は、最後の行で指定されています。

22.6.3. 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ースです。**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 を設定するには、「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化するには、**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。例えば **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。

上記一覧の最後の行で、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。設定が有効にするには、$InputGSSServerRun の前にすべての imgssapi 設定オプションを配置する必要があります。

例22.17 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
```

22.6.4. 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構成します。

   ```
   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 を使って 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。詳細な説明については、「新しい鍵と証明書の生成」 を参照してください。

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構成します。

   ```
   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* を証明書ファイルへ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")
   }
   ```

22.7. 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、構造化されたメッセージを作成して、ファイルデータベースに保存する場合 (「Rsyslog での構造化ロギング」 を参照) などです。この協力で必要となる通信インターフェースは、**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。また古いデータも **journald** からインポートできます (**IgnorePreviousMessages** オプションで禁止されていない場合)。**imjournal** の基本設定については、「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。ソケットを使って **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")
```

22.8. 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) フォーマットを使っ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-値のペアを使っ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** を使って **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することができます。解析が成功するには、**mmjsonparse** は入力メッセージが **Lumberjack** プロジェクトで定義された方法で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。**Lumberjack** はまた、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一覧も定義します。**Lumberjack** についての詳細情報は、「オンラインのドキュメント」 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** ライブラリーを用いて lumberjack 準拠の形式でメッセージを生成できます。**libumberlog** についての詳細情報は、「オンラインのドキュメント」 を参照してください。

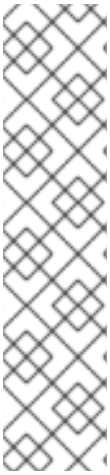
22.8.1. Journal からのデータのインポート

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することが可能です。

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。指定されたメッセージ数に達すると、毎回データが保存されます。
- *path* は、state ファイルへ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** で、現在ジャーナル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。指定された状態ファイルがない場合に使用されます。デフォルト設定は **off** です。この設定がオフで状態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。そのために、**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。これらメタデータエントリーの一部は、例22.19「詳細な journalctl 出力」 に一覧表示されています。journal フィールドの全一覧については、**systemd.journal-fields(7)** 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** にフォーカスすることができます。

22.8.2. 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 **@cee:** 文字列を JSON 文字列の前に付加し、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することができます。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")
```

22.8.3. 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。

22.8.4. MongoDB でのメッセージの保存

Rsyslog は、**ommongodb** 出力モジュールで JSON ログの MongoDB ドキュメントデータベースで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* を MongoDB サーバーの名前もしく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換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で置き換えます。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* を置き換えて、ログインの詳細を設定します。

テンプレートを使うと、最終的なデータベース出力の形式を形成できます。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。

22.9. 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。環境変数でデバッグを修正することができ、たとえばログファイルにデバッグ出力を保存することができます。**rsyslogd** を起動する前に、以下をコマンドラインに入力します。

```
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全一覧は、**rsyslogd(8)** man ページ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 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。

22.10. 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、様々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処理、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。

22.10.1. 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一覧で、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/** で使わ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22.18 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、エントリー一覧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のみに関連性があります。**journalctl** 出力を減らす簡単な方法は、**-n** オプションの使用です。これで、最新のログエントリーは指定された数のみ一覧表示されます。

```
journalctl -n Number
```

Number を表示する行数に置き換えます。数字が指定されない場合は、**journalctl** は最新の 10 エントリーを表示します。

journalctl コマンドを使うと、以下の構文で出力形式を制御することができます。

```
journalctl -o form
```

form を出力形式を指定するキーワードで置き換えます。**verbose** オプションを使うと、全フィールドをともなう完全構造化エントリーアイテムが返されます。**export** を使うと、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します。**json** を使うと、エントリーを JSON データ構造としてフォーマットします。キーワードの全一覧は、**journalctl(1)** man ページを参照してください。

例22.19 詳細な journalctl 出力

全エントリーについての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=78c81449c920439da57da7bd5c56
a770;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一覧表示しています。「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については、**systemd.journal-fields(7)** man ページを参照してください。

22.10.2. 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** で以下を入力します。

```
usermod -a -G adm username
```

username を **adm** グループに追加するユーザー名で置き換えます。すると、このユーザーは、root ユーザーと同様の **journalctl** コマンド出力を受け取るようになります。アクセス制御が機能するのは、**Journal** の永続ストレージが有効になっている場合のみであることに注意してください。

22.10.3. ライブビューの使用

journalctl はパラメーターなしで呼び出されると、収集されたうちで最も古いものから順にエントリーすべてを一覧表示します。ライブビューでは、新たなエントリーが現れるとこれが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することができます。**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。

22.10.4. メッセージのフィルタリング

パラメーターなしで実行された **journalctl** コマンドの出力は大規模なものになることが多いので、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することができます。

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 **debug** (7)、**info** (6)、**notice** (5)、**warning** (4)、**err** (3)、**crit** (2)、**alert** (1)、または **emerg** (0) のいずれかもしくはそれらの数字に置き換えます。

例22.20 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

現在のブートから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
```

システム再起動をたまにしか行わない場合は、**-b** は **journalctl** の出力を大幅に削減しません。この場合、時間ベースのフィルタリングが便利になります。

```
journalctl --since=value --until=value
```

--since および **--until** を使うと、特定の時間枠内に作成されたログメッセージのみを表示できます。これらのオプションに渡す *values* は、以下の例で示すように日付か時間、もしくはそれら両方の形式を取ることができます。

例22.21 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よって結果を絞ることができます。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、以下を入力します。

```
journalctl -p warning --since="2013-3-16 23:59:59"
```

高度なフィルタリング

例22.19「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一覧表示しています。**systemd** が保存可能なメタデータのすべての説明については、**systemd.journal-fields(7)** man ページを参照してください。このメタデータは、ユーザーが介入することなく、各ログメッセージについて収集されます。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一覧表示するには、以下の構文を使用します。

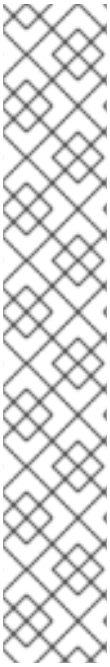
```
journalctl -F fieldname
```

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一覧表示されます。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一覧表示することもできます。以下を入力します。

```
journalctl fieldname=
```

そして **Tab** を 2 回押します。これは、**journalctl -F** *fieldname* の代わりとなります。

1 つのフィールドに複数の値を指定することもできます。

```
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22.22 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での最新の変更を追跡することもできます。

```
journalctl -f fieldname=value ...
```

22.10.5. 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。するとユーザーによっては、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。

- 長期的にトラブルシュートのための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っても、保存されるデータ量は空きのあるメモリーに依存することから、特定の期間をカバーす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** で以下を入力します。

```
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
```

22.11. 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。

22.11.1. 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、**Vi** や **Emacs** などのテキストエディターで表示することができます。一部のログファイルはシステム上の全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は読み取りに 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、**システムログ** を使用します。

注記

システムログ 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 を使ったパッケージのインストールについては、「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、**アプリケーション** → **システムツール** → **システムログ** 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、**システムログ** 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22.2「システムログ」 で表示されている一覧とは異なる場合があります。

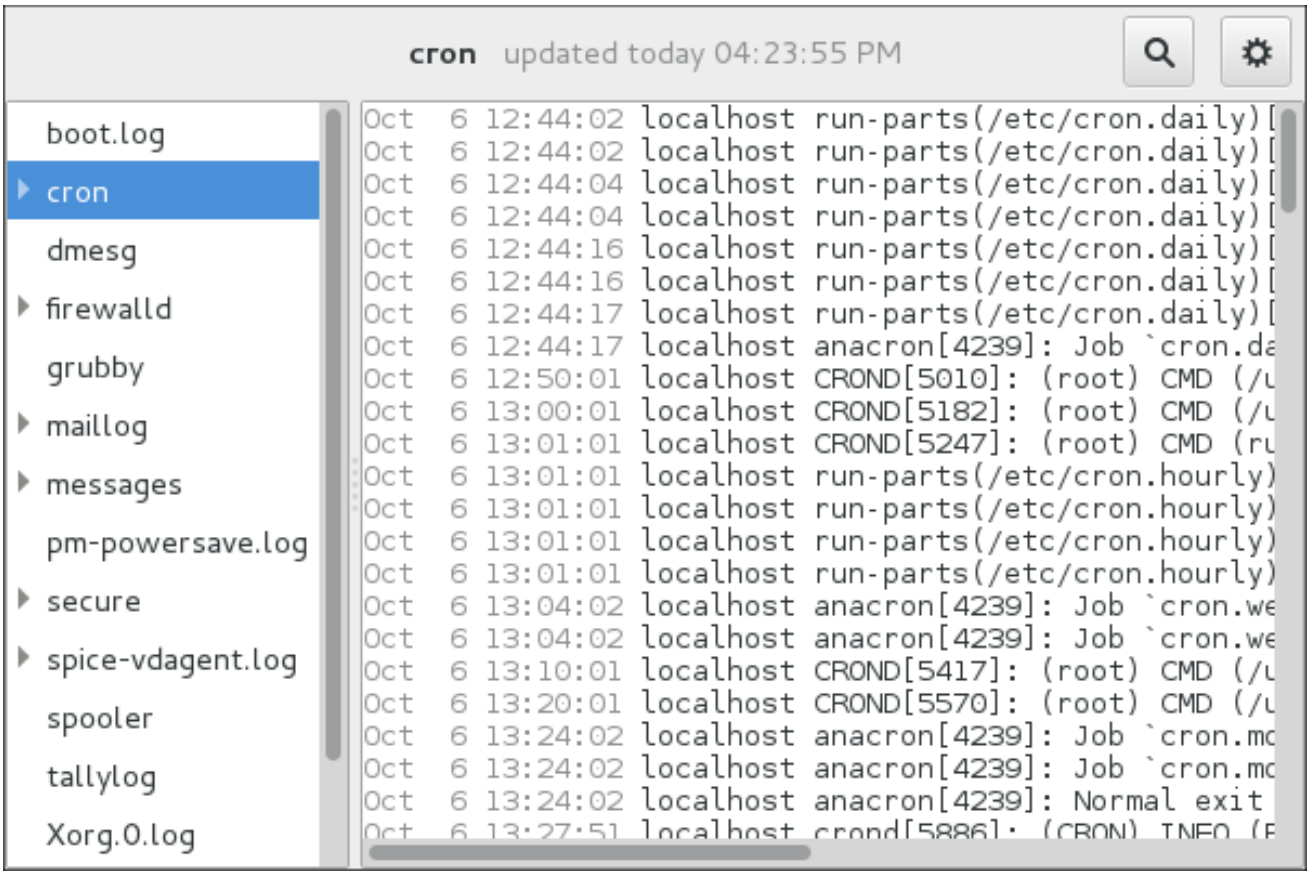


図22.2 システムログ

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 (フィルター)** → **Manage Filters (フィルターの管理)** を選択して必要なフィルターを定義または編集します。

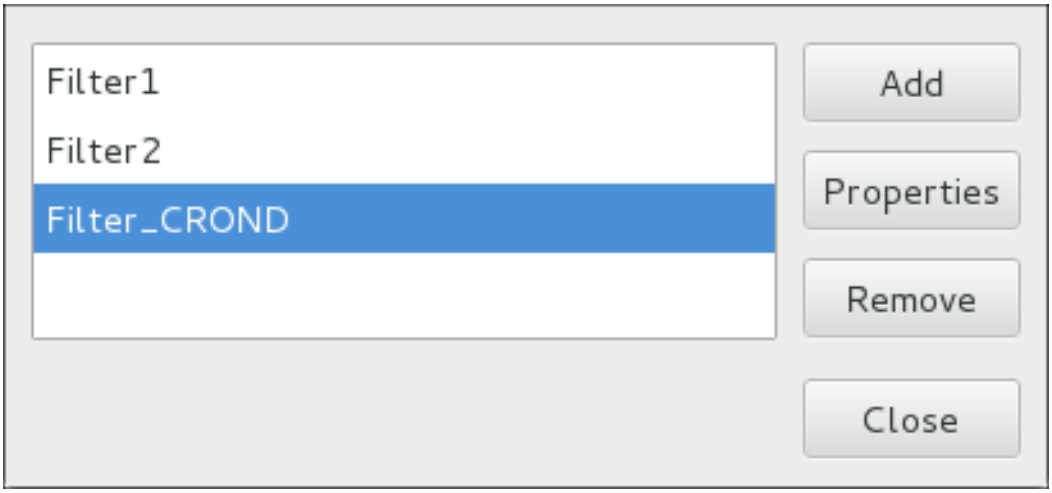


図22.3 システムログ - フィルター

フィルターを追加/編集することで、図22.4「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することができます。

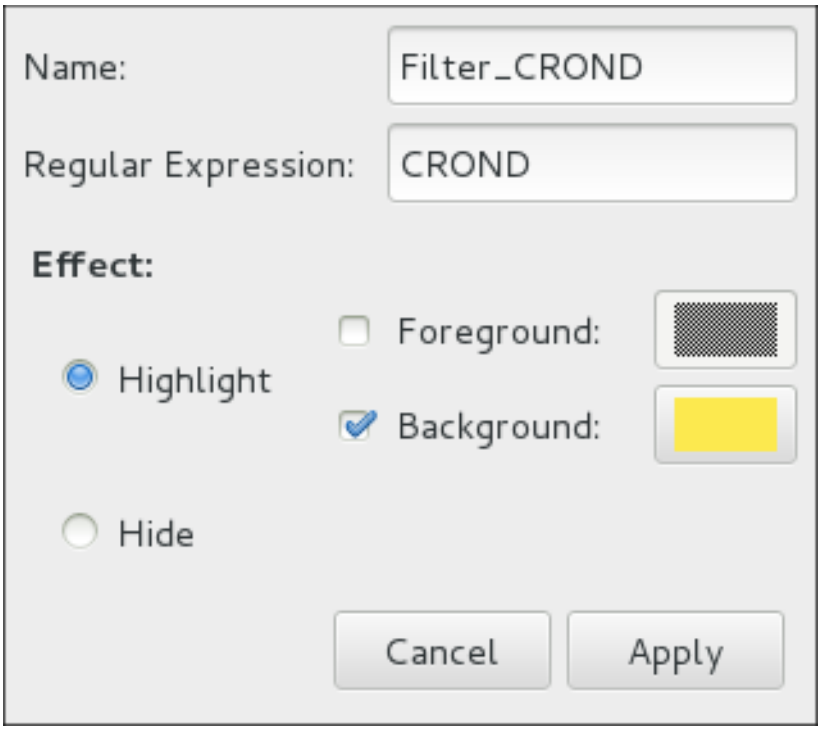


図22.4 システムログ - フィルターの定義

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。
- **効果**
  - **強調** — これが有効な場合は、検索結果は選択した色で強調されます。強調させる場所をテキストの背面/前面から選択できます。
  - **非表示** — 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。

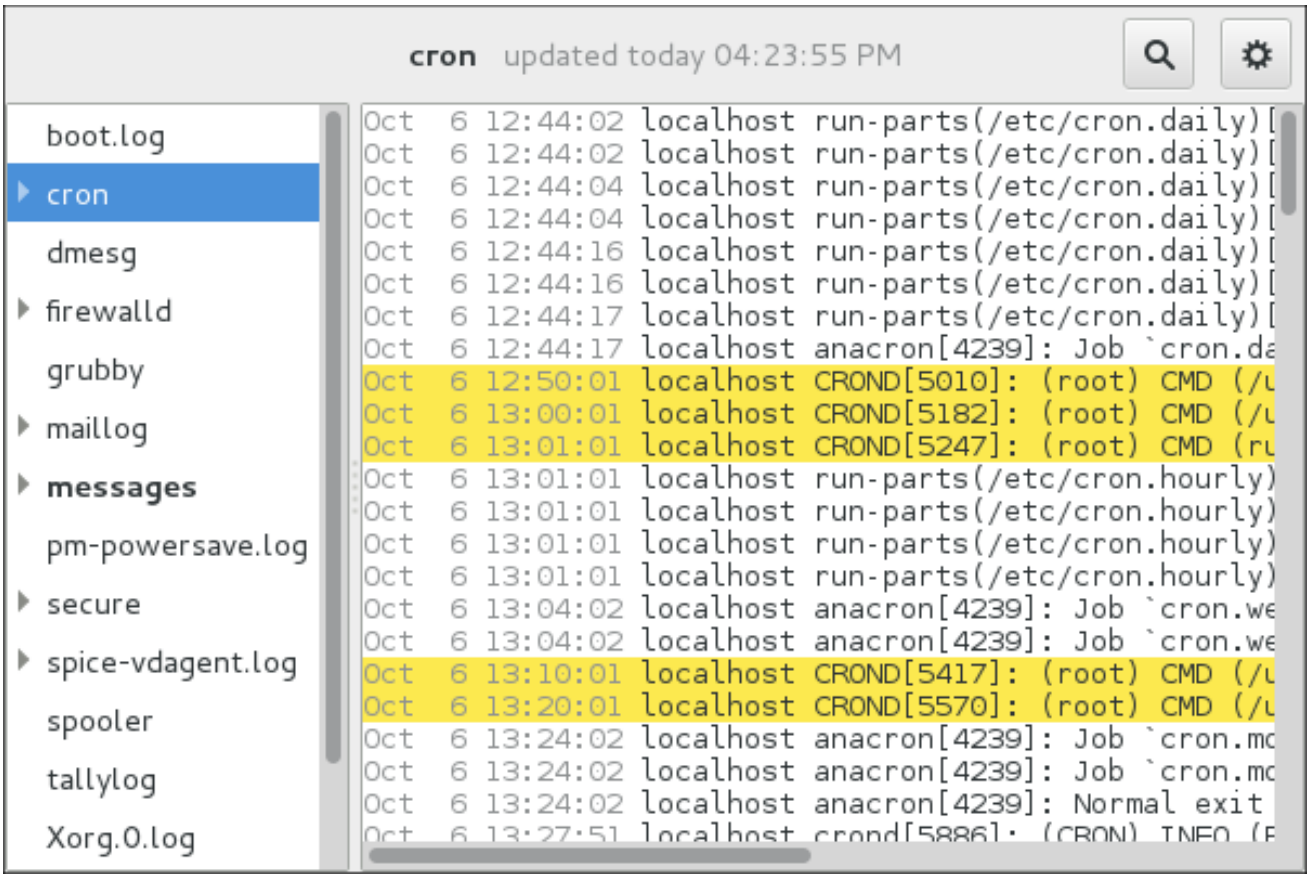


図22.5 システムログ - フィルターの有効化

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。

22.11.2. ログファイルの追加

一覧で閲覧したいログファイルを追加するには、**ファイル** → **開く** の順に選択します。その後、閲覧したいログファイルのディレクトリーとファイル名を選択できる **ファイルの選択** ウィンドウが表示されます。図22.6「システムログ - ログファイルの追加」 は、**ファイルの選択** ウィンドウを示しています。

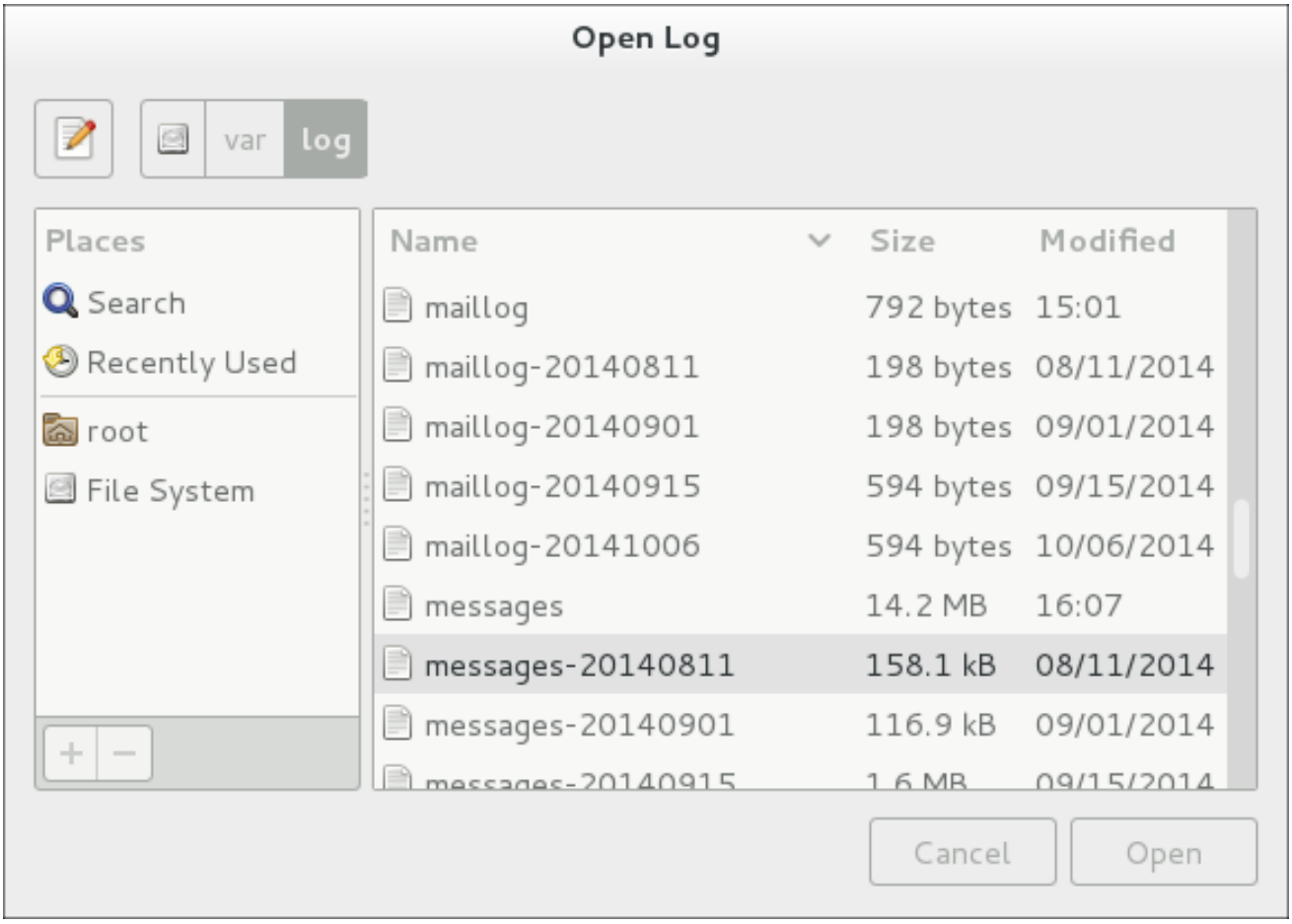


図22.6 システムログ - ログファイルの追加

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一覧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することができます。

注記

システムログ では、**.gz** 形式で圧縮されたログファイルを開くこともできます。

22.11.3. 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一覧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、ログファイルの最下部に新しい行が太字で表示されます。図22.7「システムログ - 新しいログ通知」 は、**cron** ログファイルおよび **messages** ログファイル内の新しい通知を示しています。**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)。

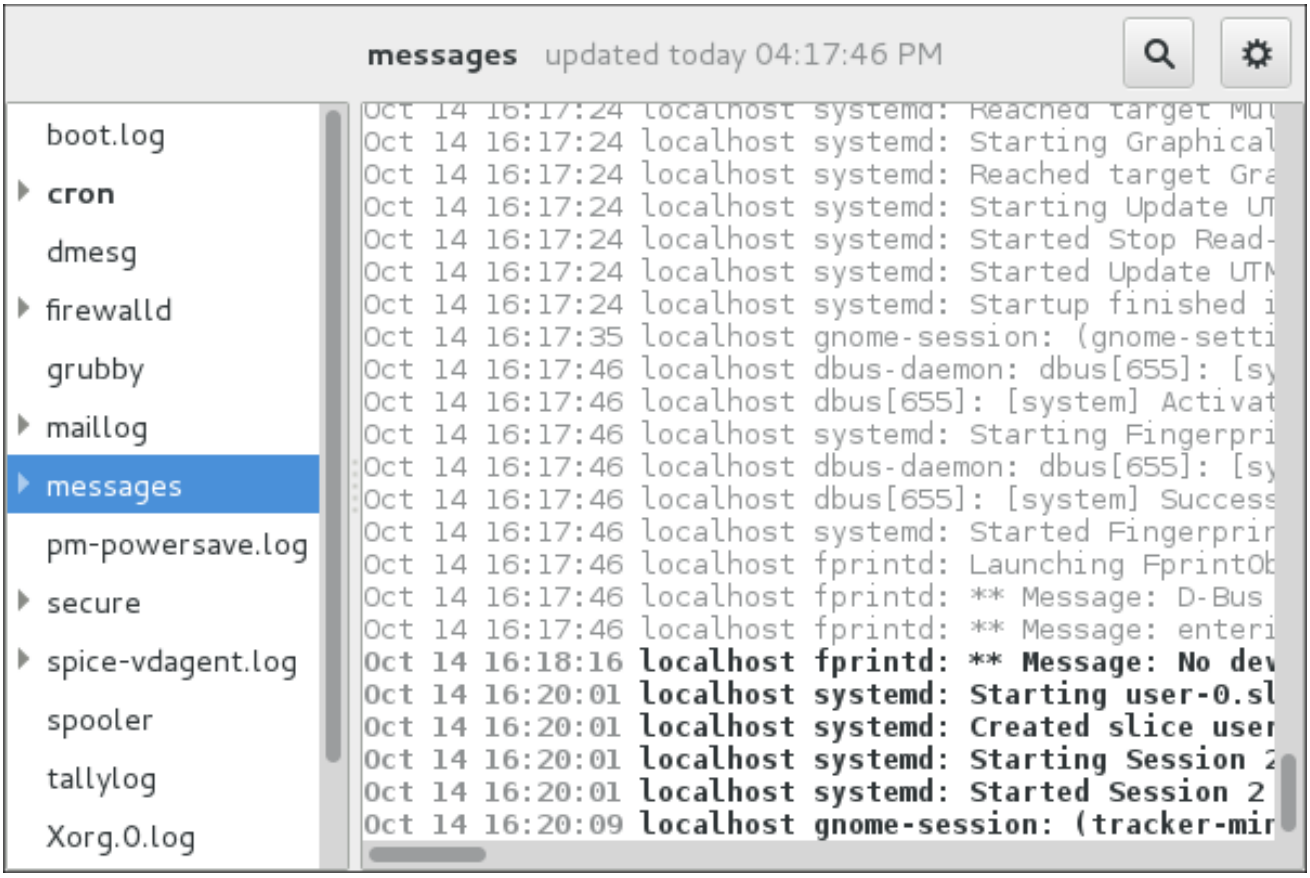


図22.7 システムログ - 新しいログ通知

22.12. 関連資料

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 ページは、特別な **Journal** フィールドを一覧表示しています。

インストールできるドキュメント

/usr/share/doc/rsyslog*version*/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
```

オンラインのドキュメント

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** コマンドを使って管理者権限を取得する方法が説明されています。
- 10章*systemd によるサービス管理* では systemd に関する詳細情報と、**`systemctl`** コマンドを使っ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23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** を使います。「Cron を使用した繰り返しジョブのスケジュール設定」 をご覧ください。
- 特定の日に非同期的に実行するには **anacron** を使います。「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」 をご覧ください。
- 特定の時間に 1 回実行するには **at** を使います。「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設定」 をご覧ください。
- 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** を使います。「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」 をご覧ください。
- 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」 をご覧ください。

本章では、これらのタスクの実行方法について説明します。

23.1. 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については、「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** サービスが読み取って、ジョブを実行します。

23.1.1. 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。

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化されており、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) のマニュアルページをご覧ください。

23.1.2. 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 テーブルを使用するか、より好ましくは **/etc/cron.d/** に cron テーブルファイルを作成します。**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. 選択:
   - 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、月の 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、より好ましくは **/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。

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) マニュアルページをご覧ください。基本的情報は **/etc/crontab** ファイルの冒頭部分をご覧ください。

```
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
```

非 root 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って **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。

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** 環境変数によって指定されたエディタを使って、ユーザー自身の **crontab** ファイルの編集が開始されます。

2. Scheduling a cron Job as root user にある方法と同じ方法でジョブを指定します。ただしユーザー名のフィールドは省略します。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。

23.2. 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、anacron テーブルファイル (**anacrontab** ファイルとも呼ばれます) に **anacron** ジョブを指定します。その後、これらのファイルは **crond** サービスが読み取って、ジョブを実行します。

23.2.1. 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。

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、cronie パッケージのサブパッケージなので、すでにインストールされている可能性があります。まだ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化されており、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 の設定に関する詳細は、anacrontab(5) のマニュアルページをご覧ください。

23.2.2. Anacron ジョブのスケジュール設定

root ユーザーとして anacron ジョブをスケジュール設定する

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。

手順23.1 root ユーザーとして anacron ジョブをスケジュール設定する

1. 選択:
   - 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。
   - 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. 完成した行を **/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** ファイルをご覧ください。ジョブの指定方法に関する詳細は、anacrontab(5) マニュアルページをご覧ください。

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** を使って日、週、月ごとにスケジュール設定できます。詳細は 「ジョブの時間、日、週、月ごとのスケジュール設定」 をご覧ください。

23.3. 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って　**at** ジョブを指定します。 このジョブはその後**atd** サービスによって実行されます。

23.3.1. 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。

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化されており、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 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
```

23.3.2. 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) マニュアルページと**/usr/share/doc/at/timespec** ファイルをご覧ください。

例23.1 At の時間を指定する

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
```

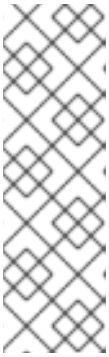
ジョブを 5 日後に実行するには、以下を実行します。

```
~]# 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。

```
警告: コマンドは /bin/sh を使って実行されます。
```

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. 空の行の上 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** と **batch** のいずれのジョブであるかを指定します。 **a** は **at** を、**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。

1. **atq** コマンドを使って、保留中のジョブを一覧表示します。

```
~]# atq
26      Thu Feb 23 15:00:00 2017 a root
28      Thu Feb 24 17:30:00 2017 a root
```

2. スケジュール設定した時間とユーザーを使って、削除したい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
```

23.3.2.1. 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することができます。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。
- いずれのファイルにも空白は使用しません。
- **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** 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が変更された場合でも、**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。

23.4. 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って**batch** ジョブを指定します。このジョブはその後**atd** サービスによって実行されます。

23.4.1. 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」をご覧ください。

23.4.2. 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。

```
警告: コマンドは /bin/sh を使って実行されます。
```

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、いずれか最初に発見されたものを使用します。

3. 空の行の上 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限界の変更
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。

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
ユーザーは、**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」 をご覧ください。

23.5. 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**、および **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については、「システム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** デーモンを再度読み込みます。

```
~]# systemctl daemon-reload
```

4. **systemd** サービスを有効にします。

```
~]# systemctl enable one-time.service
```

5. スクリプトを作成し以下を実行します。

```
~]# 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** ユニットを無効化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
```

23.6. 関連資料

Red Hat Enterprise Linux でシステムタスクを自動化する方法については、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** デーモンのマニュアルページは、**crond** の仕組みとその動作の変更方法について記載しています。
- crontab - **crontab** ユーティリティーのマニュアルページは、対応しているオプションの一覧について記載しています。
- 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は、 **crontab** ファイルのフォーマットについて記載しています。

第24章 自動バグ報告ツール (ABRT)

24.1. ABRT の概要

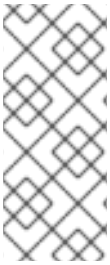
自動バグ報告ツール (Automatic Bug Reporting Tool) は通常 **ABRT** と呼ばれるツールセットで、ユーザーがアプリケーションクラッシュを検出し、報告する手助けとなるように設計されています。主な目的は、問題の報告と解決方法の発見のプロセスを容易にすることです。この場合の解決方法には、Bugzilla チケット、ナレッジベースの記事、修正を含むバージョンへの更新の提案などが含まれます。

ABRT は、**abrtd** デーモンと、検出された問題をプロセス、分析、報告するための数多くのシステムサービスとユーティリティーで構成されます。ほとんどの場合、このデーモンはバックグラウンドで何も表示せずに実行されますが、アプリケーションがクラッシュしたり、カーネル oops が検出されるとアクションを起こします。その後デーモンは、コアファイル (ある場合) などの関連する問題データや、クラッシュしているアプリケーションのコマンドラインパラメーター、さらに他のフォレンジックユーティリティーのデータなどを収集します。

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 oopses、およびカーネルパニックの検出をサポートしています。サポート対象の障害およびクラッシュのタイプおよびこれらが検出される方法についての詳細情報は、「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常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。このレポーティングツールは、問題データを Bugzilla データベースや Red Hat テクニカルサポート (RHTSupport) の web サイトに送信できるほか、**FTP**/**SCP** を使用したアップロード、メールでの送信、またはファイルへの書き込みなどにも対応しています。

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** と **libreport** の操作と設定は密接に統合されているため、本ガイドでも取り上げています。



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ので、**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。

24.2. ABRT のインストールとそのサービスの起動

ABRT を使用するためには、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ースを提供し、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。これら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全般的なワークフローは手順的には似通っており、同様のパターンになります。

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の一般的な情報については、「パッケージのインストール」 を参照してください。

24.2.1. ABRT GUI のインストール

ABRT の *グラフィカルユーザーインターフェース* 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
```

インストール時に、**ABRT** 通知アプレットは、グラフィカルデスクトップのセッション開始時に自動的に起動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
```

アプレットが実行されていない場合には、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することができます。

```
~]$ abrt-applet &
[1] 2261
```

24.2.2. コマンドライン用の ABRT のインストール

コマンドラインインターフェース 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、必要なパッケージをインストールできます。

```
~]# yum install abrt-cli
```

24.2.3. 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ついての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、報告することができます。デフォルトで **ABRT** は C および C++ アプリケーションのクラッシュなどの最も一般的な障害タイプのサポートと共にインストールされます。他の障害タイプのサポートは別個の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一覧は、「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すべての対応パッケージの一覧も含まれています。

24.2.4. ABRT サービスの起動

abrtd デーモンは、**/var/spool/abrt** ディレクトリでファイルシステムを管理するために、**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このユーザーが存在していない場合、**abrt** ユーザーは自動的に作成されます。UID と GID は 173 です。それ以外の場合、**abrt** ユーザーは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うと、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、デーモンは実行されていません。以下のコマンドを **root** として入力すると、現在のセッションにデーモンを開始することができます。

```
~]# systemctl start abrtd.service
```

同じコマンドを使って、関連するエラー検出サービスを開始する、またはステータスチェックを行うことが可能です。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一覧については、「ソフトウェア問題の検出」をご覧ください。

カーネルパニックまたはカーネルウップスが発生したときのみ開始される **abrt-vmcore** と **abrt-pstoreoops** のサービスを除き、すべての **ABRT** サービスは、各パッケージがインストールされたときのブート時に、自動的に有効化され開始されます。**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化または有効化できます。

24.2.5. 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** コマンドを使って 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** コマンドを実行した直後にクラッシュを検出します。グラフィカルセッションを実行している場合、GUI の通知アプレットが検出された問題をユーザーに通知します。コマンドライン上で、**abrt-cli list** コマンドを実行するか、**/var/tmp/abrt/** ディレクトリで作成されたクラッシュダンプを確認すると、クラッシュが検出されたことをチェックできます。検出されたクラッシュの対処方法については 「検出された問題の処理」 をご覧ください。

24.3. ABRT の設定

ABRT では、問題のライフサイクルは **イベント** によって決定されます。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。問題データディレクトリーは通常、以下のようなファイルで構成されます:
analyzer、**architecture**、**coredump**、**cmdline**、**executable**、**kernel**、**os_release**、**reason**、**time**、**uid**。

backtrace などの他のファイルは、問題の分析中に作成される場合があります。これは、使用するアナライザーメソッドやその設定によって異なります。これらのファイルには、それぞれシステムおよび問題自体についての固有の情報が格納されます。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ってその問題を処理することができます。記録された問題の処理に関する **ABRT** ツールについての詳細は、「検出された問題の処理」 を参照してください。

24.3.1. イベントの設定

ABRT のイベントは、*プラグイン* を使って実際のレポーティング操作を行います。プラグインはコンパクトなユーティリティーで、問題データディレクトリーのコンテンツを処理するためにイベントが呼び出します。プラグインを使うことで、**ABRT** は問題を様々な宛先に報告できます。また、ほとんどの報告先に関しては、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。

設定詳細の中にはデフォルト値を含めることが可能なものもありますが (Bugzilla URLなど)、適切なデフォルト値を設定できないものもあります (ユーザー名など)。**ABRT** は、このような設定を、システムワイドの設定には **/etc/libreport/events/** ディレクトリーの、ユーザー固有の設定には **$HOME/.cache/abrt/events/** ディレクトリーの **report_Bugzilla.conf** のような設定ファイルで探します。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。

これらのファイルは、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** ファイルに格納されています。このファイルは、ユーザーインターフェー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は、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/** ディレクトリーにコピーして、新規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な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一覧
- 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、それらのタイプ (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することができるかを判断し、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** ツールを使っ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の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 (設定)** を選びます。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中に選択可能な全イベントの一覧が表示されます。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することができます。

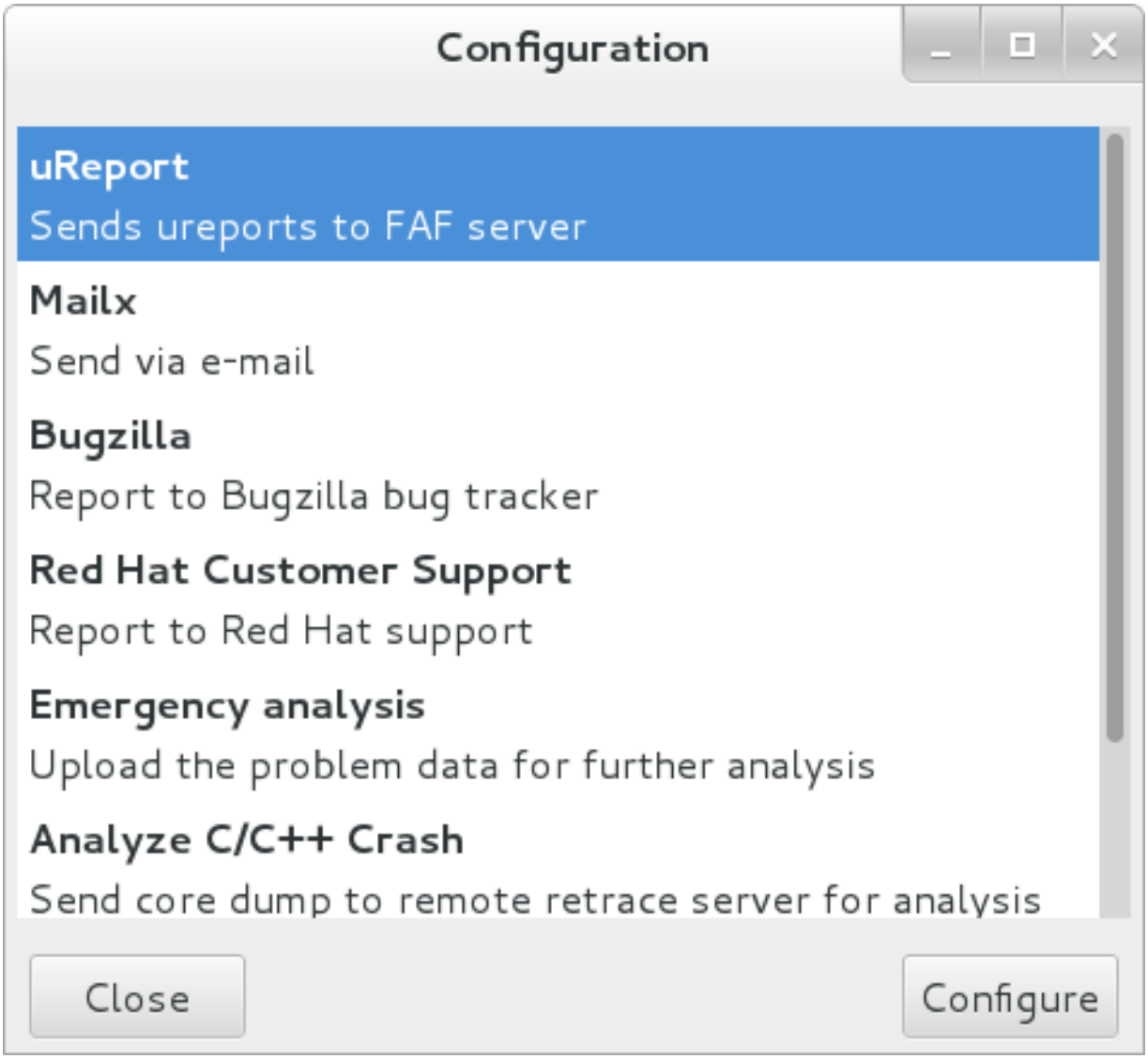


図24.1 ABRT イベントの設定

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一覧表示しています。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、すべてのイベントで GUI を使った設定ができるわけ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ついての情報は、「カスタムイベントの作成」 を参照してください。

表24.1 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。
Analyze VM core	analyze_VMcore **`vmcore_event.conf`**	カーネル oops の問題データに対して **GDB** (GNU デバッガ) 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 デバッガ) 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。
Logger	report_Logger **`print_event.conf`**	問題レポートを作成して、それを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。

24.3.2. 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って定義されます。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、主要設定ファイル **`/etc/libreport/report_event.conf`** によって読み込まれます。abrt

は **/etc/libreport/events.d/** に含まれているスクリプトを実行するので、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** は行頭に空白文字を入れない行で開始します。その後に続く、**空白** 文字または **タブ** 文字で始まる行は、すべてこのルールの一部とみなされます。各 **ルール** は、**条件** パートと **プログラム** パートの 2 部で構成されます。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*

ここで、

- *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構成されます。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の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** イベントが実行されると、**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、問題データをドロップす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されると、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*

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*

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。

24.3.3. 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** 設定ファイル内の **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。このアプリケーションを起動するには、**アプリケーション → 諸ツール → 自動バグ報告ツール (ABRT)** に移動します。

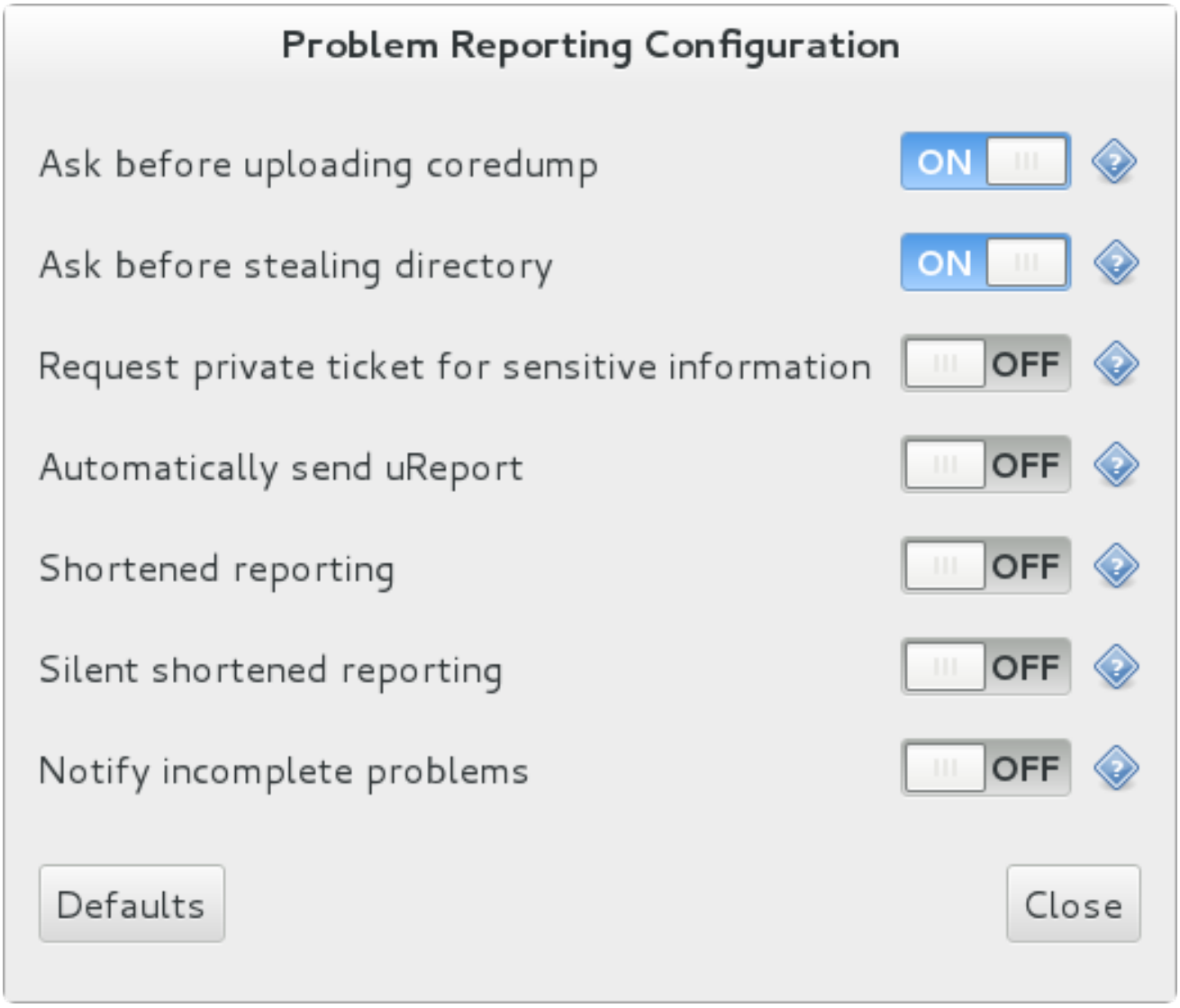
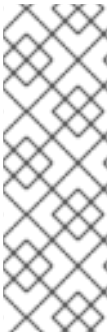


図24.2 ABRT 問題報告ツールの設定

デフォルトでは、**ABRT** はクラッシュを検出すると、問題の基本情報とともに 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、システム内部で使用でき、完全に匿名になるように設計されています。これらの特長があるために、これらのレポートを自動レポーティングで使用することができます。μReports を使用してバグの発生を追跡することはできますが、通常はエンジニアがバグを修正するのに必要な情報は十分に提供されません。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** 設定ファイル内の **AutoreportingEvent** ディレクティブの値を別の **ABRT** イベントに向けるように修正する必要があります。標準イベントの概要については、表24.1「標準 ABRT イベント」 を参照してください。

24.4. 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、プロセスすることができます。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** の主要パッケージのいずれか (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については、「ABRT のインストールとそのサービスの起動」 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一覧表示したものです。

表24.2 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

24.4.1. C および C++ クラッシュの検出

abrt-ccpp サービスは、自らのコアダンプハンドラーをインストールします。これが起動すると、カーネルの **core_pattern** 変数のデフォルト値を上書きするので、C および C++ クラッシュは **abrtd** が処理します。**abrt-ccpp** サービスを停止すると、以前に指定された **core_pattern** の値が回復されます。

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。つまり、カーネルは **core.*** という接頭辞の付いたファイルをクラッシュしたプロセスの現行ディレクトリーに作成します。**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。このプログラムは、**ABRT** のダンプの場所にそれを保存し、**abrtd** デーモンに新しいクラッシュを通知します。また、デバッグ目的で **/proc/*PID*/** ディレクトリー (*PID* はクラッシュしたプロセスの ID) から **maps**、**limits**、**cgroup**、**status** の各ファイルを保存します。これらのファイルのフォーマットおよび意味についての説明は、proc(5) を参照してください。

24.4.2. 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。このファイルは同様に、**abrt_exception_handler.py** をインポートします。これにより Python のデフォルトの **sys.excepthook** がカスタムハンドラーで上書きされ、未処理の例外がソケット 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わ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。

24.4.3. Ruby 例外の検出

rubygem-abrt パッケージは、**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されます。これにより、処理されていない可能性のある例外の確認が可能になります。未処理の例外が取得されるたびに、**ABRT** はバグレポートを作成します。これは標準の **ABRT** ツールを使ってRed Hat Bugzilla に提出することができます。

24.4.4. Java 例外の検出

ABRT Java コネクターは、取得されていない Java 例外を **abrtd** にレポートする JVM エージェントです。これは、いくつかの JVMTI イベントコールバックを登録します。またこれは **-agentlib** コマンドラインパラメーターを使って JVM に読み込む必要があります。登録済みコールバックの処理は、アプリケーションのパフォーマンスにマイナスの影響を与えることに注意してください。**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ことで、例外が **abrtd** に処理されるようになります。コネクターに例外を標準出力に出力させるには、このオプションを省略します。

24.4.5. X.Org クラッシュの検出

abrt-xorg は、**X.Org server** のクラッシュについての情報を **/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一覧は、**/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。

24.4.6. 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、これを処理することができます。この機能は、**abrt-oops** サービスが提供します。

ABRT は、**abrt-vmcore** サービスを使って、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することができます。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/tmp/abrt/** ディレクトリー内に作成し、コアダンプファイルをこの新規に作成された問題データディレクトリーに移動します。**/var/crash/** ディレクトリーの検索後に、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、**kdump** サービスがシステム上で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。この設定は、 **system-config-kdump** グラフィカルツールを使うか、または GRUB メニューのカーネルオプション一覧の **crashkernel** パラメーターを指定することで実行できます。**kdump** の有効化と設定の詳細については、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 メニューの変更方法については、 *25章GRUB 2 での作業* を参照してください。

abrt-pstoreoops サービスを使うと、**ABRT** はカーネルパニックについての情報を収集し、処理できるようになります。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。このプラットフォームに依存した pstore インターフェース (永続的なストレージ) は、システムの再起動時にデータを保存するメカニズムを提供するため、カーネルパニック情報の保存を可能にします。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。

24.5. 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 回のみですが、その問題のクラッシュダンプを削除してしまうと、その特定の問題が次回発生した際、**ABRT** はその問題を新規クラッシュとして処理することになります。このため、**ABRT** は警告を表示し、その問題の詳細を記入して報告するように要求します。**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。

24.5.1. 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 コマンドの出力で一覧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。

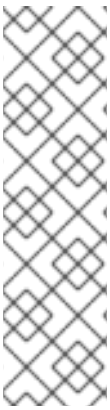
```
abrt-cli info [-d] directory_or_id
```

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。これでそれぞれの問題の **backtrace** ファイルが生成されている場合には、これらを含む問題の保存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することができます。**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** 変数と **EDITOR** 変数がチェックされます。これらの変数がいずれ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。優先するエディターは、**.bashrc** 設定ファイル内で設定できます。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提出されます。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っ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、その問題を削除することができます。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** コマンドについてのヘルプを表示するには、**--help** オプションを使用します。

```
abrt-cli command --help
```

24.5.2. 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、**アプリケーション → 諸ツール → 自動バグ報告ツール (ABRT)** 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、検出された問題の一覧を表示します。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構成されます。

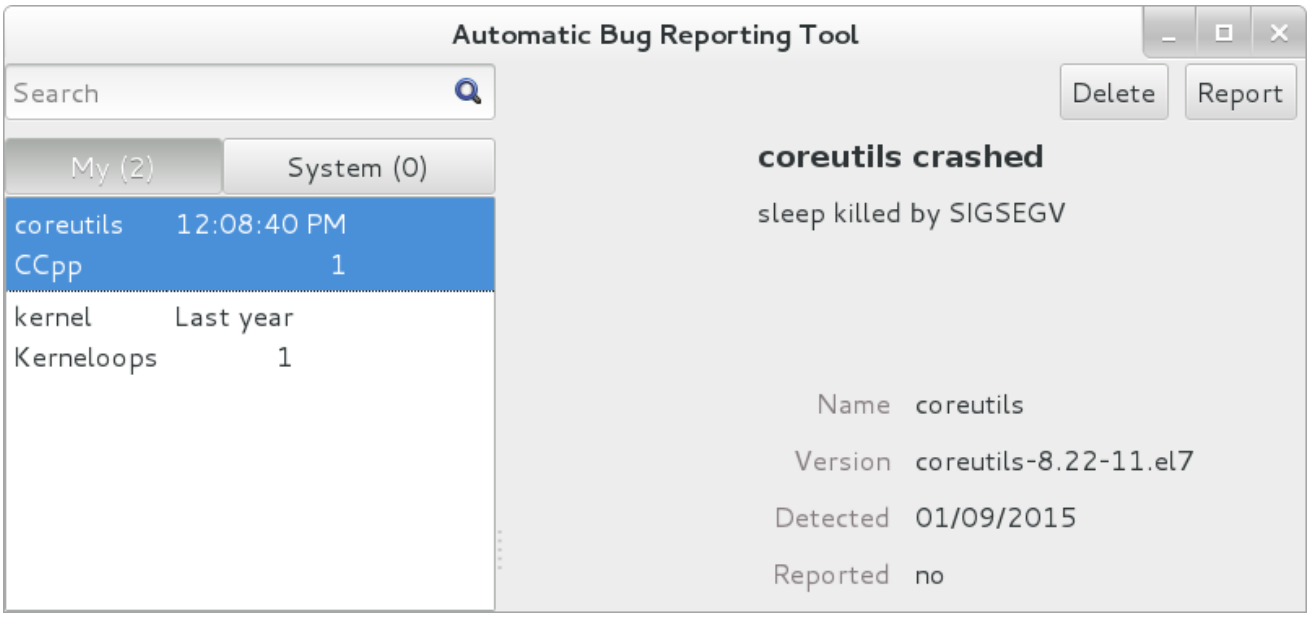


図24.3 ABRT GUI

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。次に、指示に従って問題の説明、分析方法の決定、問題の報告先に進みます。問題を破棄するには、**Delete** ボタンをクリックします。

24.6. 関連資料

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。

オンラインのドキュメント

- 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ー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 『カーネルクラッシュダンプガイド』 は、**kdump** クラッシュ回復サービスの設定し、テストし、使用する方法について説明します。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についての概要を説明します。

関連項目

- 22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って、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** コマンドを使っ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25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。

25.1. 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。これら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** を使用して手動で生成される場合、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) は **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** の詳細については、「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

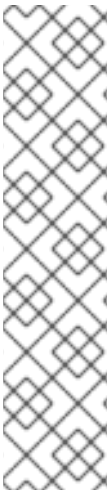
grub.cfg 設定ファイルには、多くのコードスニペットやディレクティブに加えて、1 つまたは複数の **menuentry** ブロックが含まれています。これらはそれぞれ、単一の GRUB 2 ブートメニューエントリーを表しています。これらのブロックは、常に **menuentry** キーワードで始まり、それにタイトル、オプションリスト、および中括弧が続きます。中括弧の中身は、インデントにします。たとえば、以下は Linux カーネル 3.8.0-0.40.el7.x86_64 Red Hat Enterprise Linux 7 の **menuentry** ブロック例です。

```
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** パーティションが作成されている場合は、カーネルと **initramfs** イメージへのパスは、**/boot** に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についての情報は、『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』の「初期 RAM ディスクイメージの検証」を参照してください。

25.2. 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行うには、「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については、「GRUB 2 設定ファイルのカスタマイズ」 を参照してください。

25.3. GRUB 2 メニューの一時的な変更

手順25.1 カーネルメニューエントリーの一時的な変更

単一の起動プロセス中に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**、または 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
   ```

   システムメッセージを有効にするには、**rhgb** と **quiet** のパラメーターを削除する必要があります。

   この設定は永続的なものではなく、単一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** の詳細については、「GRUB 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。

25.4. 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 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 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一覧表示
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 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一覧表示するには、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 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 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 オプションは、新しい引数を追加する **--args** および既存の引数を削除する **--remove-arguments** と組み合わせて使用する場合に、メニューエントリーを更新するために使用できます。これらのオプションには、引用符で囲まれたスペース区切りリストを使用できます。GRUB メニューエントリーに対して引数を同時に追加および削除するコマンドの形式は以下のとおりです。

```
grubby --remove-args="argX argY" --args="argA argB" --update-kernel
/boot/kernel
```

カーネルの GRUB 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、以下のように **--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については、**grubby(8)** man ページを参照してください。

25.5. 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。

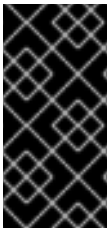
しかし、特定のエントリーを含むメニューの構築や、エントリーの特定の順番をユーザーが希望する場合もあります。GRUB 2 では基本的なブートメニューのカスタマイズが可能で、画面に表示されるものをユーザーが制御できます。

GRUB 2 は、メニューの構築に一連のスクリプトを使用します。これらは、**/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** — これは **user.cfg** ファイルからスーパーユーザー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することができます。



重要

/etc/default/grub ファイルで **GRUB_TIMEOUT** キーが **0** に設定されていると、GRUB 2 はシステム起動時に起動可能なカーネル一覧を表示しません。起動時にこの一覧を表示するようにするには、BIOS 情報が表示されている間に英数字のいずれかのキーを押し続けます。すると、GRUB 2 は GRUB メニューを表示します。

25.5.1. 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 GRUB 2 がロードするよう指示されます。**grub2-set-default** コマンドを使って別の GRUB 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** 値がパッケージタイプ kernel の最新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、タイプ kernel-debug 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 では、オペレーティングシステムが読み込まれるデフォルトの順番を **saved_entry** ディレクティブのキーに数値を使うことで変更できます。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。例を示します。

```
~]# grub2-set-default 2
```

一覧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。このため、上記の例では、3 番目のエントリーが読み込まれます。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一覧表示するには、**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** で以下のコマンド発行します。

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- UEFI ベースのマシンでは、**root** で以下のコマンド発行します。

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

25.5.2. 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 ブートメニューでパラメーターを追加する場合と同様に、**GRUB_CMDLINE_LINUX** キーには複数のパラメーターを指定できます。例を以下に示します。

```
GRUB_CMDLINE_LINUX="console=tty0 console=ttyS0,9600n8"
```

ここで、**console=tty0** は最初の仮想ターミナルであり、**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** で以下のコマンド発行します。

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- UEFI ベースのマシンでは、**root** で以下のコマンド発行します。

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

25.5.3. 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、同一パーティション上でインストール済み Linux カーネルを検索します。**/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自動的にブートメニューに追加されます。

/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
```

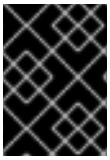
```
# This file provides an easy way to add custom menu entries.  Simply type
the
# menu entries you want to add after this comment.  Be careful not to
change
# the 'exec tail' line above.
```

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>
}
```

25.5.4. 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不要な行を **40_custom** ファイルに置いた場合は、消去してください。

   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)
   - **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、**chmod a-x <file_name>** コマンドを実行してこれらのファイルを実行可能ファイルにします。

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. 以下のように **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** で以下のコマンド発行します。

     ```
     ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
     ```

   - UEFI ベースのマシンでは、**root** で以下のコマンド発行します。

     ```
     ~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
     ```

25.6. パスワードによる 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だが、既存のメニューエントリーの起動には**必要ない**。
- 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 を使っ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のブロックから **--unrestricted** パラメーターを削除します。例を示します。

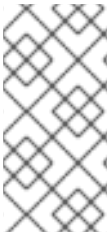
```
[file contents truncated]
menuentry 'Red Hat Enterprise Linux Server (3.10.0-
327.18.2.rt56.223.el7_2.x86_64) 7.2 (Maipo)' --class red --class
gnu-linux --class gnu --class os --unrestricted $menuentry_id_option
'gnulinux-3.10.0-327.el7.x86_64-advanced-c109825c-de2f-4340-a0ef-
4f47d19fe4bf' {
        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 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 パラメーターのないブートエントリーでは root パスワードが必要になりますが、GRUB 2 では、パスワードなしでこれらのエントリーを起動できる root 以外のユーザーを作成することもできます。エントリーの修正には root パスワードが必要になります。これら新規ユーザーの作成については、GRUB 2 Manual を参照してください。

25.7. 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。

25.7.1. 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
```

25.7.2. 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
```

25.7.3. 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. 以下のように **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** で以下のコマンド発行します。
     -

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- UEFI ベースのマシンでは、**root** で以下のコマンド発行します。

```
~]# 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。

25.8. 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 から 7 へインプレースアップグレードを行う場合、**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、アップデートを受信しません。
- **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。

アップグレードの要件

Red Hat Enterprise Linux 7 システムをアップグレードする前に、**GRUB Legacy** の手動でのバックアップを実行してください。**GRUB Legacy** は、grub パッケージで利用できる点に注意してください。

手順25.2 GRUB Legacy の手動でのバックアップを作成

1. grub パッケージをダウンロードします。

```
~]#  yum reinstall -y --downloadonly grub
```

2. ダウンロードパッケージを見つけます。

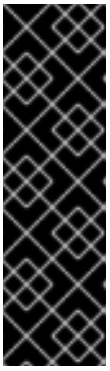
```
~]# find /var/cache/yum/ | grep "grub"
```

注記

yum のデフォルトキャッシュロケーションを変更していない場合、キャッシュは **/var/cache/yum/** ディレクトリーにあります。**yum** のデフォルトキャッシュロケーションを変更している場合は、設定を確認して見つけます。詳細は、Working with Yum Cache と Configuring Yum and Yum Repositories を参照してください。

3. **/root/** ディレクトリーなど、パッケージを安全な場所にコピーします。

```
~]# cp /var/cache/yum/x86_64/6Server/rhel/packages/grub-0.97-
99.el6.x86_64.rpm /root/
```



重要

grub パッケージを **/boot/** ディレクトリーにコピーしないでください。**/boot/** に十分な空きスペースがないと、RHEL 6 から RHEL 7 へのインプレースアップグレードに失敗することがあります。詳細は、プレアップグレードおよびアップグレードのドキュメントを参照してください。

- リリースノートドキュメント
- ナレッジベースの記事

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。

手順25.3 GRUB Legacy から GRUB 2 へのアップグレード

1. バックアップから grub パッケージをインストールします。

```
~]# rpm --install --force --nodeps grub-0.97-99.el6.x86_64.rpm
```

この手順は、**GRUB Legacy** から **GRUB 2** へのアップグレードが途中で失敗した場合のリカバリーオプションを確保します。パッケージには様々なバージョンがあるかもしれないので、バックアップしたパッケージの正確な名前を使用する必要がある点に注意してください。

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 がシステム上にない場合、以下を実行して手動でインストールできます。

```
~]# yum install grub2
```

起動可能なデバイスのファイルを決める

1. 起動可能なデバイス向けに **GRUB Legacy** の表示記号を探します。そのためには、**GRUB Legacy** 設定ファイルの **/boot/grub/grub.conf** を表示し、**root** 行を探します。

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to
this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/mapper/vg_rhel68-
lv_root
#          initrd /initrd-[generic-]version.img
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.32-642.4.2.el6.x86_64)
root (hd0,0)
kernel /vmlinuz-2.6.32-642.4.2.el6.x86_64 ro
root=/dev/mapper/vg_rhel68-lv_root rd_NO_LUKS  KEYBOARDTYPE=pc
KEYTABLE=us LANG=en_US.UTF-8 rd_NO_MD SYSFONT=latarcyrheb-sun16
```

```
crashkernel=auto rd_LVM_LV=vg_rhel68/lv_root
rd_LVM_LV=vg_rhel68/lv_swap rd_NO_DM rhgb quiet
initrd /initramfs-2.6.32-642.4.2.el6.x86_64.img
title Red Hat Enterprise Linux 6 (2.6.32-642.el6.x86_64)
root (hd0,0)
kernel /vmlinuz-2.6.32-642.el6.x86_64 ro root=/dev/mapper/vg_rhel68-
lv_root rd_NO_LUKS  KEYBOARDTYPE=pc KEYTABLE=us LANG=en_US.UTF-8
rd_NO_MD SYSFONT=latarcyrheb-sun16 crashkernel=auto
rd_LVM_LV=vg_rhel68/lv_root rd_LVM_LV=vg_rhel68/lv_swap rd_NO_DM
rhgb quiet
initrd /initramfs-2.6.32-642.el6.x86_64.img
```

各メニューエントリーについて、**root** 行は起動可能なデバイスを指定しています。この例では、**hd0,0** が起動可能なデバイスです。

2. **/boot/grub/device.map が正確でない場合のみ、この手順を実行します。この状況は、たとえば、ハードウェア設定の変更後に発生することがあります。**

   a. **/boot/grub/device.map** を再作成します。

   ```
   ~]# grub-install --recheck /dev/sda
   ```

   古い設定は **/boot/grub/device.map.backup** に自動的にバックアップが作成されます。

   b. 前の手順でデバイスマッピング設定が壊れてしまった場合は、バックアップを復元します。

   ```
   ~]# rm /boot/grub/device.map
   ~]# cp /boot/grub/device.map.backup /boot/grub/device.map
   ```

3. デバイスファイルに対する **GRUB Legacy** のデバイスの表示記号のマッピングを決定します。そのためには、手順 1 で見つけたデバイスをとり、**/boot/grub/device.map** ファイルで対応するエントリーを見つけます。

   ```
   # this device map was generated by anaconda
   (hd0)     /dev/sda
   (hd1)     /dev/sdb
   ```

   この例では、デバイス **hd0** のデバイスファイルは **/dev/sda** として一覧に表示されています。

   デバイスファイルをメモします。これは次の手順で使用します。

GRUB 2 設定ファイルの生成

元の **GRUB Legacy** 設定を削除せずに、 **GRUB 2** 設定を追加します。**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. **GRUB 2** ファイルを **/dev/sdX** ディスクの **/boot/grub/** ディレクトリーにインストールします。

   ```
   ~]# grub2-install --grub-setup=/bin/true /dev/sdX
   ```

/dev/sdX を 「起動可能なデバイスのファイルを決める」 で決めた起動可能なデバイスファイルに置き換えます。

--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。

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**
- **.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。

2. **/boot/grub2/grub.cfg** を生成します。

```
~]# grub2-mkconfig -o /boot/grub2/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** に直接するのではなく、**/etc/default/grub** にする必要があります。**/boot/grub2/grub.cfg** に直接変更を加えると、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 をインストールしたまま GRUB 2 をテスト

GRUB Legacy 設定を削除せずに **GRUB 2** をテストします。**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。設定を変更してしまうと、システムが起動できなくなることがあります。安全に **GRUB 2** 設定をテストするために、**GRUB 2** を **GRUB Legacy** から起動します。

1. 新しいセクションを **/boot/grub/grub.conf** に追加します。

```
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。

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、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。

GRUB Legacy の置き換えと削除

GRUB 2 が正常に機能したら、**GRUB Legacy** を置き換え、システムから削除します。

1. **GRUB Legacy** 起動セクターを **GRUB 2** ブートローダーで上書きします。

```
~]# grub2-install /dev/sda
```

2. grub パッケージをアンインストールします。

```
~]# yum remove grub
```

GRUB 2 へのアップグレードはこれで完了です。

25.9. シリアルコンソールでの GRUB 2

画面やキーボードのないコンピューターを使用している場合は、シリアル通信によるマシンの制御が非常に便利です。

25.9.1. GRUB 2 メニューの設定

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=ttyS0,115200
```

これらの設定は永続的ではなく、単一ブートでのみ適用されます。

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。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=ttyS0,115200 --
update-kernel=DEFAULT
```

--update-kernel パラメーターにはキーワード **ALL** またはカーネルインデックス番号のコンマ区切りリストを指定できます。**grubby** の使用の詳細については、「GRUB 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。

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 つの行を追加します。

```
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、その他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** ファイルでの変更後は、GRUB 2 設定ファイルの更新が必要になります。

以下のように **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** で以下のコマンド発行します。

```
~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- UEFI ベースのマシンでは、**root** で以下のコマンド発行します。

```
~]# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

注記

シリアル接続で GRUB ターミナルにアクセスするには、カーネルに定義に追加のオプションを追加して、特定のカーネルがシリアル接続を監視するようにします。以下に例を示します。

```
console=ttyS0,9600n8
```

ここで、**console=ttyS0** は使用するシリアルターミナル、**9600** はボーレート、**n** はパリティーなし、**8** はビットでの単語の長さになります。以下のログファイルのようなタスクには非常に大きいボーレート (例: **115200**) が推奨されます。

シリアルコンソール設定の詳細については、「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。

25.9.2. 画面を使っ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。

25.10. 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ースを使って行い、このインターフェー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** キーを押すと開始されます。**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ー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ースを読み込みます。

コマンドラインインターフェースは最も基本的な GRUB インターフェースですが、制御を一番多く付与するものでもあります。コマンドラインでは、関連する GRUB 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されます。このインターフェースには、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**、および **Delete** のキーは bash シェルでの機能と同様に作動します。

25.10.1. 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ースのアクティブ化や、システムに同時に他のユーザーによる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. ブート中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=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** を押して追加したパラメーターでシステムを起動します。

25.10.2. 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取り専用でマウントし、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ー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、緊急モードに **root** パスワードを必要とします。

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** を押して追加したパラメーターでシステムを起動します。

25.10.3. 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に **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。また、ログメッセージの数を増やすために、**debug** オプションをカーネルコマンド行に追加することもできます。**systemd** において、カーネルコマンドオプション **debug** は **systemd.log_level=debug** のショートカットになりました。

手順25.4 デバッグシェルコマンドの追加

このためにのみデバッグシェルをアクティブ化するには、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、編集のために **e** キーを押します。

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
```

オプションで、**debug** 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** を押して追加した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** の詳細については、「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。

手順25.5 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 (キーの送信)** → **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、次のようなプロンプトが TTY9 に表示されるはずです: **[root@localhost /]#**

3. 必要な場合は、デバッグシェルにいることを確認するために、以下のようにコマンドを入力します。

```
/]# 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。**systemd-debug-generator(8)** man ページでは、これらのオプションについて説明しています。

25.10.4. root パスワードの変更およびリセット

root パスワードのセットアップは、Red Hat Enterprise Linux 7 のインストールで必須です。**root** パスワードを忘れたり、失ったりした場合は、そのパスワードをリセットできます。ただし、wheel グループのメンバーであるユーザーは以下のように **root** パスワードを変更できます。

```
~]$ sudo passwd root
```

GRUB 2 でのパスワードの再設定は、Red Hat Enterprise Linux 6 での GRUB のようなシングルユーザモードでの実行はなされません。**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。

- 手順25.6「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 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。

- 手順25.7「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。GRUB メニューを編集し、時間がかかる可能性がある SELinux ファイルの再ラベル行うか、SELinux 強制モードを変更してブート完了時に **/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。

手順25.6 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。

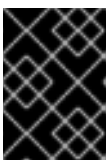
```
sh-4.2# rm -f /.autorelabel
```

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。

手順25.7 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. **linux16** 行または **linuxefi** 行 (UEFI システムの場合) の最後またはその付近から **rhgb** パラメーターと **quiet** パラメーターを削除します。

   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

システムメッセージを有効にするには、**rhgb** と **quiet** のパラメーターを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の場合は、この時点でパスワードが必要です。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。プロンプトを表示するには、**Backspace** キーを押します。ロギングメッセージを無視して、キーをリリースし、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 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** ツールは以下のエラーメッセージを表示して失敗します。

```
Authentication token manipulation error
```

8. パスワードファイルを更新すると、間違った SELinux セキュリティーコンテキストのファイルになります。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。

    暗号化された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。プロンプトを表示するには、**Backspace** キーを押したままにします。ロギングメッセージを無視して、キーをリリースし、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い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
```

    以下のコマンドを入力して、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
```

25.11. 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 1 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ために使用する 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ース

- 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 セキュアブートは、第 2 段階のブートローダーのインストールや削除を防ぎません。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。署名は、ブートローダーのインストール時や更新時ではなく、ブート時に検証されます。このため、UEFI セキュアブートは、ブートパス操作を停止せず、不正な変更の検出を可能にします。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。

25.11.1. Red Hat Enterprise Linux 7 における UEFI セキュアブートのサポート

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。この署名がないとシステムはドライバーを受け付けません。Red Hat により提供され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、カーネルの対応する Red Hat 公開鍵で認証されています。

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 (一部の制限が存在します)。

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。したがって、GRUB 2 モジュールのロードを許可すると、セキュアブートで定義するセキュリティー範囲内の信頼できないコードが実行されます。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。

詳細については、Red Hat ナレッジベースの記事 Restrictions Imposed by UEFI Secure Boot を参照してください。

25.12. 関連資料

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 の使い方や設定に関する情報が含まれています。**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なります。
- **`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 その使い方に関する 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 パッケージにより提供され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ています。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 — インストールガイドは、インストールや用語、インターフェース、コマンドなどの GRUB 2 に関する基本的な情報を提供しています。

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について説明します。

第26章 RELAX-AND-RECOVER (REAR)

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を復元する

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が、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。新規ハードウェア上でレスキューシステムを起動することで、**rear recover** コマンドが発行できるようになります。このコマンドは、復旧プロセスを開始します。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 の使い方について説明します。

26.1. 基本的な REAR の使用方法

26.1.1. ReaR のインストール

root で以下のコマンドを実行して rear パッケージと依存関係をインストールします。

```
~]# yum install rear genisoimage syslinux
```

26.1.2. 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で置き換えます。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26.1 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にようするには、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/
```

全オプション一覧については、rear(8) man ページの "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26.1「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 つの同等の出力ファイルができます。

- **/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 を出力先にしたいロケーションに置き換えます。

26.1.3. レスキューシステムの作成

以下の例では、詳細な出力とともに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26.1「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。

26.1.4. 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」 に詳細あり) に置き換えます。

例26.2 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
```

26.1.5. 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、以下の手順に従います。

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ースで "Recover" オプションを選択します。

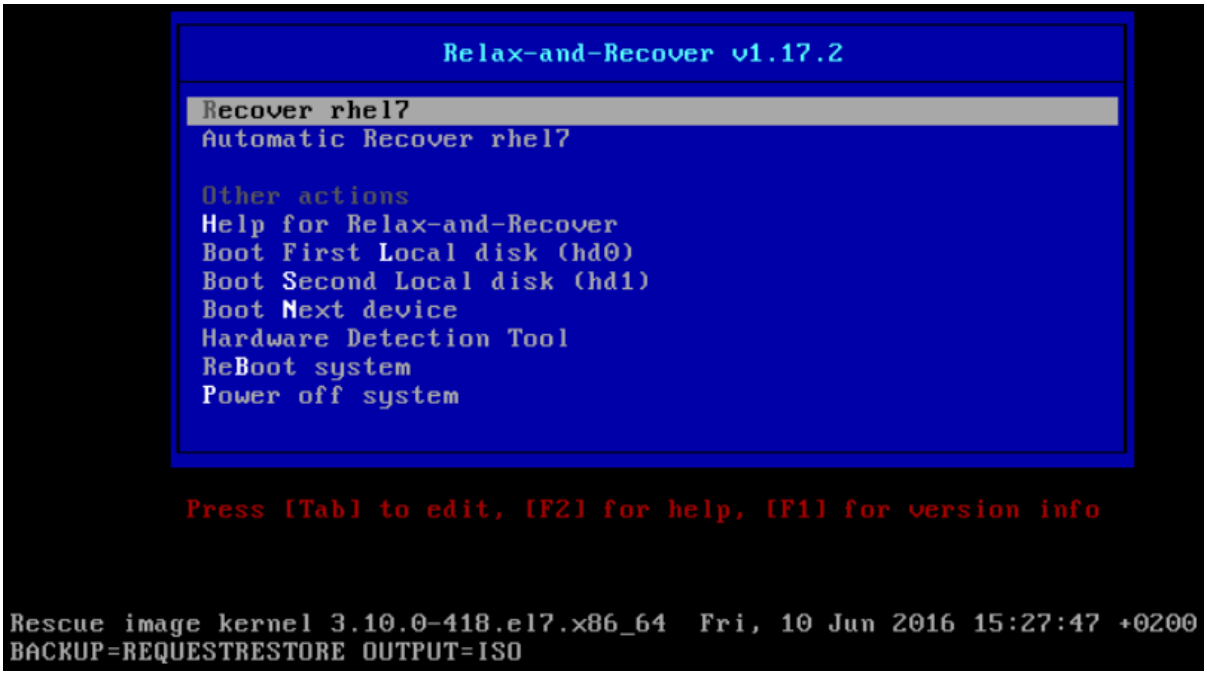


図26.1 レスキューシステムのメニュー

3. 以下のプロンプトが表示されます。

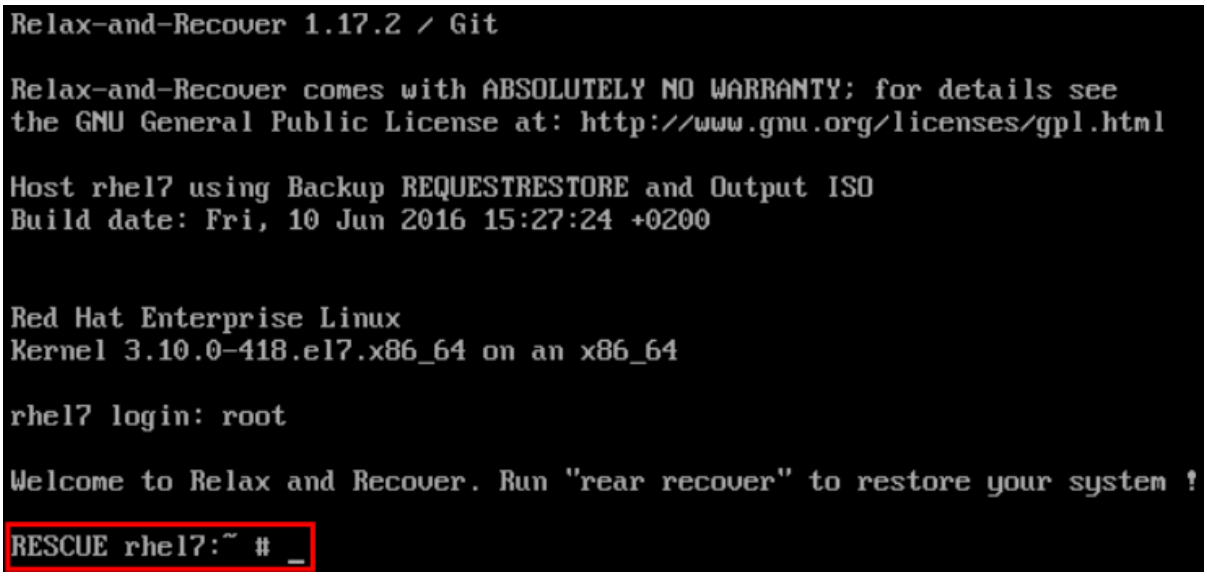


図26.2 レスキューシステムのプロンプト

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。

```
rhel7 login: root

Welcome to Relax and Recover. Run "rear recover" to restore your system !

RESCUE rhel7:~ # rear recover
Relax-and-Recover 1.17.2 / Git
Using log file: /var/log/rear/rear-rhel7.log
NOTICE: Will do driver migration
Comparing disks.
Disk configuration is identical, proceeding with restore.
Start system layout restoration.
Creating partitions for disk /dev/sda (msdos)
Creating LVM PV /dev/sda2
Restoring LVM VG rhel
Sleeping 3 seconds to let udev or systemd-udevd create their devices...
Creating xfs-filesystem / on /dev/mapper/rhel-root
meta-data=/dev/mapper/rhel-root  isize=256    agcount=4, agsize=524032 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0        finobt=0
data     =                       bsize=4096   blocks=2096128, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
Mounting filesystem /
Creating xfs-filesystem /boot on /dev/sda1
meta-data=/dev/sda1              isize=256    agcount=4, agsize=65536 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0        finobt=0
data     =                       bsize=4096   blocks=262144, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
Mounting filesystem /boot
Creating swap on /dev/mapper/rhel-swap
Disk layout created.
Please start the restore process on your backup host.

Make sure that you restore the data into '/mnt/local' instead of '/' because the
hard disks of the recovered system are mounted there.

Please restore your backup in the provided shell and, when finished, type exit
in the shell to continue recovery.
rear> _
```

図26.3 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26.3 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」 にある指示どおりに作成された **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/** に展開し、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、アーカイブと展開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/
```

```
Disk layout created.
Please start the restore process on your backup host.

Make sure that you restore the data into '/mnt/local' instead of '/' because the
hard disks of the recovered system are mounted there.

Please restore your backup in the provided shell and, when finished, type exit
in the shell to continue recovery.
rear> scp root@192.168.122.7:/srv/backup/rhel7/backup.tar.gz /mnt/local/
The authenticity of host '192.168.122.7 (192.168.122.7)' can't be established.
ECDSA key fingerprint is c9:a4:f2:89:f0:ef:3a:6b:bb:1b:b1:b7:08:f0:dd:1c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.122.7' (ECDSA) to the list of known hosts.
root@192.168.122.7's password:
backup.tar.gz                                 100% 1047MB  16.1MB/s   01:05
rear> tar xf /mnt/local/backup.tar.gz -C /mnt/local/
rear> rm -f /mnt/local/backup.tar.gz
rear> ls /mnt/local/
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin
srv  sys   tmp  usr  var
rear> _
```

図26.4 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** を入力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。その後にシステムを再起動します。

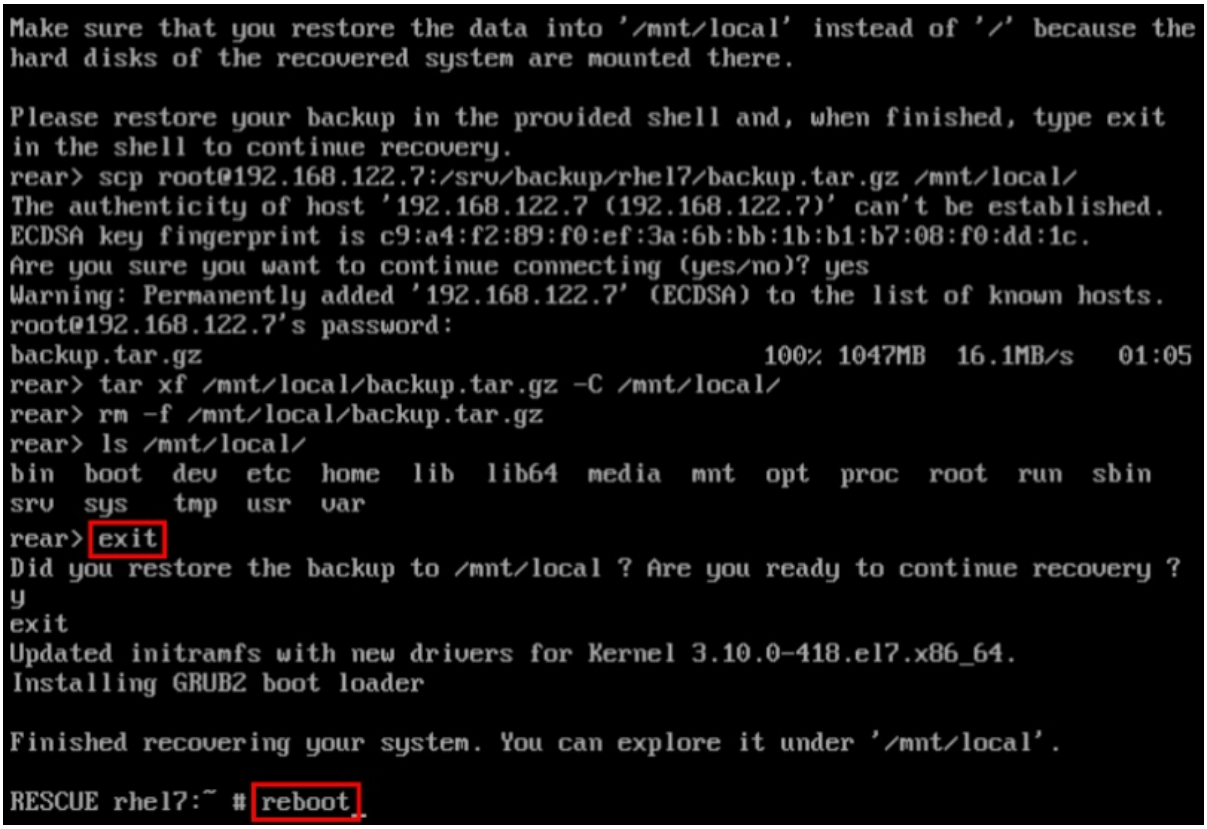


図26.5 レスキューシステム: リカバリーの終了

再起動すると、SELinux がファイルシステム全体に再度ラベル付けを実行します。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。

26.2. 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。

26.2.1. 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。

- 単一の **`rear mkbackup`** コマンドを使用して、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。

26.2.1.1. 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、rear(8) man ページの "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スペースがあるようにしてください。

例26.4 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。

- 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、単一の ISO バックアップは増分とすることができません。

  例26.5 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。

26.2.1.2. 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 と設定すると、ReaR はレスキューシステム、バックアップのいずれか、また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。

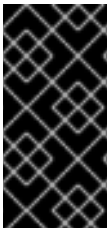
注記

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、バックアップファイルを **BACKUP_URL** で指定したディレクトリーにコピーします (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。

不必要に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 $?
```

ゼ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 設定が変更され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26.6 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
```

26.2.2. 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 を使っ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一覧および設定オプションについては、rear(8) man ページの "Backup Software Integration" セクションを参照してください。

26.2.3. 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っ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。設定例については、**/usr/share/rear/conf/default.conf** を参照してください。

26.2.4. 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)

rear コマンドの **-C** オプションを使用して個別バックアップを指定できます。引数は **/etc/rear/** ディレクトリ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、以下を実行します。

手順26.1 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** ディレクトリのファイルをバックアップします。

```
~]# 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が含まれている必要があります。

手順26.2 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。

1.
```
~]# rear -C basic_system recover
```

2.
```
~]# rear -C home_backup restoreonly
```

付録A 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 は様々な製品リリース間の、具体的な相互運用性や機能の互換性をまとめたインタラクティブなツールです。

Red Hat Enterprise Linux のバージョンごとに、各製品の具体的な機能への対応や互換性を比較するには、『comprehensive compatibility matrix』 をご覧ください。

付録B システム管理に関連する RED HAT カスタマーポータルラボ

Red Hat カスタマーポータルラボは、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 カスタマーポータルラボは以下のリンクから入手できます。Customer Portal Labs。

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構成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 を使って以下のセットアップを行います。

- 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表示します。
- **`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 を使って、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 およびそれ以降を使用する KVM 等のアプリケーションに、ブリッジ接続のネットワークインターフェースを設定するために作成されました。

NETWORK BONDING HELPER

Network Bonding Helper を用いることで、管理者はボンディングカーネルモジュールおよびボンディングネットワークインターフェースを使用して、複数のネットワークインターフェースコントローラーを単一のチャンネルにまとめることができます。

Network Bonding Helper を使用することにより、複数のネットワークインターフェースを 1 つのボンディングインターフェースとして機能させることができます。

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) を作成するための、最適なパラメータを決定します。

LVM RAID Calculator を使って、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 を使えば、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またはマウントの情報を指定すると、NFS 設定スクリプトがダウンロードできるようになります。

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最適なバランスを生成します。

Load Balancer Configuration Tool を使っ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について助言します。

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 を使って以下のセットアップを行います。

- ローカルの Yum リポジトリー
- a HTTP/FTP ベースの Yum リポジトリー

FILE SYSTEM LAYOUT CALCULATOR

ファイルシステムレイアウト計算プログラムは、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 ファイルの作成に最適なパラメータを決定します。

ファイルシステムレイアウト計算プログラムを使って、指定された RAID ストレージに所定のパラメータ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`**: 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無関係に、ブロックごとにファイルをコピー

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、設定するための bash スクリプトの生成に使用します。

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 は、ユーザーが提供する基本的な AD サーバー情報に基づいてスクリプトを生成するときに使用します。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 6.5/6.6/6.7/6.8 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よう設計されています。

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設計されています。

Kernel Oops Analyzer を使って、1 つ以上のカーネルウップス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 HELPER

Kdump Helper は、Kdump メカニズムをセットアップする設計となっています。

Kdump Helper を使ってスクリプトを生成し、メモリ内のデータを 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 DECODER

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

Multipath Helper は、Red Hat Enterprise Linux 5、6、7 にあるマルチパスデバイスに最適な設定を作成します。

Multipath Helper を使って、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、インストールスクリプトをダウンロードして、サーバーで実行します。

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** を使う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 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、または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。

付録C 改訂履歴

改訂 0.14-19.2 **Fri Sep 21 2018** **Keiko Moriguchi**
翻訳ファイルを XML ソースバージョン 0.14-19 と同期

改訂 0.14-19.1 **Wed May 30 2018** **Red Hat**
翻訳ファイルを XML ソースバージョン 0.14-19 と同期

改訂 0.14-19 **Tue Mar 20 2018** **Marie Doleželová**
7.5 GA 公開用ドキュメントの準備

改訂 0.14-17 **Tue Dec 5 2017** **Marie Doleželová**
Samba のセクションを更新。TLS を使用した RELP の設定についてのセクションを追加。GRUB Legacy から GRUB2 へのアップグレードについてのセクションを更新。

改訂 0.14-16 **Mon Aug 8 2017** **Marie Doleželová**
本ガイド全体を通じたマイナーな修正、「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)。

改訂 0.14-14 **Thu Jul 27 2017** **Marie Doleželová**
7.4 GA 公開用ドキュメントバージョン

改訂 0.14-8 **Mon Nov 3 2016** **Maxim Svistunov**
7.3 GA リリースのバージョン

改訂 0.14-7 **Mon Jun 20 2016** **Maxim Svistunov**
『Relax-and-Recover (ReaR)』 を追加。マイナーな修正。

改訂 0.14-6 **Thu Mar 10 2016** **Maxim Svistunov**
マイナーな修正および更新

改訂 0.14-5 **Thu Jan 21 2016** **Lenka Špačková**
マイナーな更新

改訂 0.14-3 **Wed Nov 11 2015** **Jana Heves**
7.2 GA リリース向けのバージョン

改訂 0.14-1 **Mon Nov 9 2015** **Jana Heves**
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。

改訂 0.14-0.3 **Fri Apr 3 2015** **Stephen Wadeley**
『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。

改訂 0.13-2 **Tue Feb 24 2015** **Stephen Wadeley**
7.1 GA リリース向けバージョン

改訂 0.12-0.6 **Tue Nov 18 2014** **Stephen Wadeley**
『TigerVNC』 の内容が改善されました。

改訂 0.12-0.4 **Mon Nov 10 2014** **Stephen Wadeley**
『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。

改訂 0.12-0 **Tue 19 Aug 2014** **Stephen Wadeley**
Red Hat Enterprise Linux 7.0 GA リリースのシステム管理者ガイド

索引

シンボル

.fetchmailrc, Fetchmail の設定オプション
    サーバーオプション, サーバーオプション
    ユーザーオプション, ユーザーオプション

.procmailrc, Procmail の設定
アクセシビリティーのためのシステム設定, アクセシビリティーのためのシステム設定
アクセス制御リスト (参照 ACL)
キーボードの設定
    レイアウト, キーボードレイアウトの変更

キーボード設定, システムロケールおよびキーボード設定
グループ (参照 グループ設定)
    GID, ユーザーとグループの管理
    ユーザープライベート, ユーザープライベートグループ
    以下を管理するためのツール
        groupadd, ユーザープライベートグループ, コマンドラインツールの使用

    共有ディレクトリー, グループディレクトリーの作成
    導入, ユーザーとグループの管理
    関連資料, 関連資料
        インストールされているドキュメント, 関連資料

グループ設定
    groupadd, 新規グループの追加
    グループの一覧を表示, グラフィカル環境でのユーザーの管理

サブスクリプション, システム登録およびサブスクリプション管理
システム
    サブスクリプション管理, システム登録およびサブスクリプション管理
    登録, システム登録およびサブスクリプション管理

システムの情報
    プロセス, システムプロセスの表示
        現在実行中, top コマンドの使用

    収集, システムモニタリングツール

システムモニター, システムモニターツールの使用, システムモニターツールの使用, システムモニターツールの使用, システムモニターツールの使用
システムログ
    フィルタリング, ログファイルの表示

更新レート, ログファイルの表示
検索, ログファイルの表示
監視, ログファイルのモニタリング

システム情報
CPU 使用率, CPU 使用率の表示
ハードウェア, ハードウェア情報の表示
ファイルシステム, ブロックデバイスとファイルシステムの表示
メモリ使用量, メモリ使用量の表示

シャドウパスワード
概要, シャドウパスワード

セキュリティー関連パッケージ
セキュリティー関連パッケージの更新, パッケージの更新

ハードウェア
表示, ハードウェア情報の表示

パスワード
シャドウ, シャドウパスワード

パッケージ, パッケージでの作業
yum を使用したインストール, パッケージのインストール
yum を使用したパッケージのアンインストール, パッケージの削除
yum を使用したパッケージのダウンロード, パッケージのダウンロード
yum を使用したパッケージの一覧表示
glob 表現, パッケージの検索
yum list available, パッケージの一覧表示
yum list installed, パッケージの一覧表示
yum repolist, パッケージの一覧表示
yum search, パッケージの一覧表示

yum を使用したパッケージの検索
yum search, パッケージの検索

yum を使用したパッケージの表示
yum info, パッケージ情報の表示

yum を使用した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のインストール
パッケージの表示
yum info, パッケージ情報の表示

パッケージグループ
yum を使用したパッケージグループの一覧表示

um groups, パッケージグループの一覧表示

ファイルシステム, ブロックデバイスとファイルシステムの表示
ブートローダー
GRUB 2 ブートローダー, GRUB 2 での作業

プリンター (参照 印刷設定)
プロセス, システムプロセスの表示
メモリ使用量, メモリ使用量の表示
メールユーザーエージェント, MTA の設定 (参照 電子メール)
メール転送エージェント (参照 MTA) (参照 電子メール)
メール配信エージェント (参照 電子メール)
ユーザー (参照 ユーザー設定)
UID, ユーザーとグループの管理
以下を管理するツール
useradd, コマンドラインツールの使用

導入, ユーザーとグループの管理
管理用ツール
ユーザー設定ツール, コマンドラインツールの使用

関連資料, 関連資料
インストールされているドキュメント, 関連資料

ユーザープライベートグループ (参照 グループ)
および共有ディレクトリー, グループディレクトリーの作成

ユーザー設定
コマンドラインの設定
passwd, 新規ユーザーの追加
useradd, 新規ユーザーの追加

ユーザーの一覧を表示, グラフィカル環境でのユーザーの管理

ユーザー設定ツール (参照 ユーザー設定)
ログファイル, ログファイルの表示と管理
(参照 システムログ)
rsyslogd デーモン, ログファイルの表示と管理
モニタリング, ログファイルのモニタリング
ローテーション, ログファイルの場所の特定
場所の特定, ログファイルの場所の特定
表示, ログファイルの表示
説明, ログファイルの表示と管理

仮想ホスト (参照 Apache HTTP サーバー)
印刷設定
CUPS, 印刷設定
IPP プリンター, IPP プリンターの追加
LDP/LPR プリンター, LPD/LPR Host or Printer の追加
Samba プリンター, Samba (SMB) プリンターの追加
セッティング, 設定のページ
プリンターの共有, プリンターの共有
ローカルプリンター, ローカルプリンターの追加
印刷ジョブ, 印刷ジョブの管理
新規のプリンター, プリンター設定の開始

情報
システム情報, システムモニタリングツール

自動化タスク, システムタスクの自動化
設定
基本設定, 環境の基本設定

追加
グループ, 新規グループの追加
ユーザー, 新規ユーザーの追加

電子メール
Fetchmail, Fetchmail
Postfix, Postfix
Procmail, メール配信エージェント(MDA)
Sendmail, Sendmail
スパム
フィルターによる除去, スパムフィルター

セキュリティ, 通信のセキュリティー保護
クライアント, セキュアな電子メールクライアント

セキュリティー
サーバー, 電子メールクライアントの通信のセキュリティー保護

タイプ
メールユーザーエージェント, メールユーザーエージェント (Mail User Agent)
メール転送エージェント, メール転送エージェント (Mail Transport Agent)
メール配信エージェント, メール配信エージェント (MDA)

プログラムの分類, 電子メールプログラムの分類
プロトコル, 電子メールプロトコル
IMAP, IMAP

POP, POP
SMTP, SMTP

メールサーバー
Dovecot, Dovecot

関連資料, 関連資料
インストールされているドキュメント, インストールされているドキュメント
オンラインのドキュメント, オンラインのドキュメント
関連書籍, 関連資料

A

ABRT, ABRT の概要
(参照 abrtd)
(参照 Bugzilla)
(参照 Red Hat テクニカルサポート)
CLI, コマンドラインツールの使用
GUI, GUI の使用
イベントの設定, イベントの設定
イベント作成, カスタムイベントの作成
インストール, ABRT のインストールとそのサービスの起動
クラッシュ検出, ABRT の概要
テスト, ABRT のクラッシュ検出テスト
問題
サポート対象, ソフトウェア問題の検出
処理, 検出された問題の処理
検出, ソフトウェア問題の検出

概要, ABRT の概要
標準イベント, イベントの設定
自動レポーティング, 自動レポーティングの設定
設定, ABRT の設定
起動, ABRT のインストールとそのサービスの起動, ABRT サービスの起動
関連資料, 関連資料

ABRT CLI
インストール, コマンドライン用の ABRT のインストール

ABRT GUI
インストール, ABRT GUI のインストール

abrtd
ステータス, ABRT サービスの起動

テスト, ABRT のクラッシュ検出テスト
再起動, ABRT サービスの起動
起動, ABRT のインストールとそのサービスの起動, ABRT サービスの起動
関連資料, 関連資料

ABRTツール
インストール, 補助 ABRT ツールのインストール

ACL
ext3 ファイルシステム上で, アクセス制御リスト
getfacl , ACL の取り込み
Samba で, アクセス制御リスト
setfacl , アクセス ACL の設定
でアーカイブ作成, 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
を使用した NFS 共有をマウント, NFS
を使用したファイルシステムのマウント, ファイルシステムのマウント
アクセス ACL, アクセス ACL の設定
デフォルト ACL, デフォルト ACL の設定
取り込み, ACL の取り込み
設定
アクセス ACL, アクセス ACL の設定

追加の参照, ACL 参照情報

Apache HTTP サーバー
SSL サーバー
公開鍵, 証明書とセキュリティーの概要
秘密鍵, 証明書とセキュリティーの概要, 既存の鍵と証明書の使用, 新しい鍵と証明書の生成
証明書, 証明書とセキュリティーの概要, 既存の鍵と証明書の使用, 新しい鍵と証明書の生成
認証局, 証明書とセキュリティーの概要

ステータスの確認, サービスステータスの確認
ディレクトリー
/etc/httpd/conf.d/ , 設定ファイルの編集
/usr/lib64/httpd/modules/ , モジュールを使った作業

バージョン 2.4
バージョン 2.2 から更新, 設定の更新
変更, 注目すべき変更点

ファイル
/etc/httpd/conf.d/nss.conf , mod_nss Module の有効化
/etc/httpd/conf.d/ssl.conf , mod_ssl モジュールの有効化
/etc/httpd/conf/httpd.conf , 設定ファイルの編集

モジュール
mod_ssl , SSL サーバーのセットアップ
mod_userdir, 設定の更新
ロード, モジュールのロード
開発, モジュールの書き込み

仮想ホスト, 仮想ホストのセットアップ
停止, サービスの停止
再起動, サービスの再起動
設定の確認, 設定ファイルの編集
起動, サービスの起動
関連資料
インストールされているドキュメント, 関連資料
インストールできるドキュメント, 関連資料
役立つ Web サイト, 関連資料

B

blkid, blkid コマンドの使用

C

ch-email .fetchmailrc
グローバルオプション, グローバルオプション

CPU 使用率, CPU 使用率の表示
createrepo, Yum リポジトリーの作成
cron, Cron を使用した繰り返しジョブのスケジュール設定
CUPS (参照 印刷設定)

D

df, df コマンドの使用
du, du コマンドの使用

E

ECDSA 鍵
生成, 鍵ペアの生成

F

Fetchmail, Fetchmail
コマンドオプション, Fetchmail のコマンドオプション
情報提供, 情報提供またはデバッグのオプション
特別, 特別なオプション

設定オプション, Fetchmail の設定オプション
グローバルオプション, グローバルオプション
サーバーオプション, サーバーオプション
ユーザーオプション, ユーザーオプション

関連資料, 関連資料

findmnt, findmnt コマンドの使用
free, free コマンドの使用
FTP, FTP
(参照 vsftpd)
アクティブモード, ファイル転送プロトコル (FTP)
コマンドポート, ファイル転送プロトコル (FTP)
データポート, ファイル転送プロトコル (FTP)
パッシブモード, ファイル転送プロトコル (FTP)
定義, FTP
紹介, ファイル転送プロトコル (FTP)

G

getfacl , ACL の取り込み
gnome-system-log (参照 システムログ)
gnome-system-monitor, システムモニターツールの使用, システムモニターツールの使用, システムモニターツールの使用, システムモニターツールの使用
GRUB 2
GRUB 2 のカスタマイズ, GRUB 2 での作業
GRUB 2 の再インストール, GRUB 2 での作業
GRUB 2 の設定, GRUB 2 での作業

H

HTTP サーバー (参照 Apache HTTP サーバー)
httpd (参照 Apache HTTP サーバー)

L

localectl (参照 キーボード設定)
logrotate, ログファイルの場所の特定
lsblk, lsblk コマンドの使用
lscpu, lscpu コマンドの使用
lspci, lspci コマンドの使用
lsusb, lsusb コマンドの使用

M

Mail Transport Agent Switcher, MTA の設定

MDA (参照 メール配信エージェント)
MTA (参照 メール転送エージェント)
Mail Transport Agent Switcher で切り替え, MTA の設定
デフォルトの設定, MTA の設定

MUA, MTA の設定 (参照 メールユーザーエージェント)

O

OpenSSH, OpenSSH, 主な機能
(参照 SSH)
ECDSA 鍵
生成, 鍵ペアの生成

RSA 鍵
生成, 鍵ペアの生成

ssh-add, ssh-agent の設定
ssh-agent, ssh-agent の設定
ssh-keygen
ECDSA, 鍵ペアの生成
RSA, 鍵ペアの生成

クライアント, OpenSSH クライアント
scp, scp ユーティリティーの使用
sftp, sftp ユーティリティーの使用
ssh, ssh ユーティリティーの使用

サーバー, OpenSSH サーバーの起動
停止, OpenSSH サーバーの起動
起動, OpenSSH サーバーの起動

鍵ベース認証の使用, 鍵ベース認証の使用
関連資料, 関連資料

OpenSSL
SSL (参照 SSL)
TLS (参照 TLS)
関連資料, 関連資料

P

Postfix, Postfix
デフォルトインストール, Postfix のデフォルトインストール

postfix, MTA の設定
Procmail, メール配信エージェント(MDA)

レシピ, Procmail のレシピ
SpamAssassin, スパムフィルター
フラグ, フラグ
ローカルロックファイル, ローカルロックファイルの指定
例, レシピの例
特別なアクション, 特別な条件とアクション
特別な条件, 特別な条件とアクション
配信, 配信レシピと非配信レシピの比較
非配信, 配信レシピと非配信レシピの比較
設定, Procmail の設定
関連資料, 関連資料

ps, ps コマンドの使用

R

RAM, メモリ使用量の表示
rcp, scp ユーティリティーの使用
ReaR
基本的な使用方法, 基本的な ReaR の使用方法

Red Hat Support Tool
コマンドラインでのサポートの取得, Red Hat Support Tool を使用したサポートへのアクセス

Red Hat サブスクリプション管理
サブスクリプション, システム登録およびサブスクリプションの割り当て

RSA 鍵
生成, 鍵ペアの生成

rsyslog, ログファイルの表示と管理
filters, フィルター
アクション, アクション
キュー, Rsyslog でのキュー (Queue) を使った操作
グローバルディレクティブ, グローバルディレクティブ
テンプレート, テンプレート
デバッグ, Rsyslog のデバッグ
モジュール, Rsyslog モジュールの使用
ルールセット, ルールセット
ログローテーション, ログローテーション
新しい設定形式, 新規設定フォーマットの使用
設定, Rsyslog の基本設定

S

Samba
Samba プリンター, Samba (SMB) プリンターの追加

scp (参照 OpenSSH)
security プラグイン (参照 セキュリティ)
Sendmail, Sendmail
LDAP との併用, LDAP での Sendmail の使用
UUCP を使用, Sendmail の一般的な設定変更
エイリアス, マスカレード
スパム, スパムの防止
デフォルトのインストール, Sendmail のデフォルトのインストール
マスカレード, マスカレード
一般的な設定変更, Sendmail の一般的な設定変更
制約, 用途と制約
用途, 用途と制約
関連資料, 関連資料

sendmail, MTA の設定
setfacl , アクセス ACL の設定
sftp (参照 OpenSSH)
SpamAssassin
Procmail との併用, スパムフィルター

ssh (参照 OpenSSH)
SSH プロトコル
X11 転送, X11 転送
セキュリティーリスク, SSH を使用する理由
セキュリティ保護されていないプロトコル, リモート接続に必要な SSH
バージョン 1, プロトコルのバージョン
バージョン 2, プロトコルのバージョン
ポート転送, ポート転送
リモートログインに必要, リモート接続に必要な SSH
層
チャンネル, チャンネル
トランスポート層, トランスポート層

接続のシーケンス, SSH 接続のイベントシーケンス
機能, 主な機能
設定ファイル, 設定ファイル
システム全体の設定ファイル, 設定ファイル
ユーザー固有の設定ファイル, 設定ファイル

認証, 認証

ssh-add, ssh-agent の設定
ssh-agent, ssh-agent の設定
SSL , SSL サーバーのセットアップ
    (参照 Apache HTTP サーバー)

SSL サーバー (参照 Apache HTTP サーバー)
star , 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
stunnel, 電子メールクライアントの通信のセキュリティー保護

T

TLS , SSL サーバーのセットアップ
    (参照 Apache HTTP サーバー)

top, top コマンドの使用

U

useradd コマンド
    次を使用したユーザーアカウントの作成, 新規ユーザーの追加

V

vsftpd
    SELinux, vsftpd 用の SELinux ポリシー
    TLS, TLS を使用した vsftpd 接続の暗号化
    ステータス, vsftpd の起動と停止
    マルチホーム設定, vsftpd の複数コピーの起動
    停止, vsftpd の起動と停止
    再起動, vsftpd の起動と停止
    安全化, TLS を使用した vsftpd 接続の暗号化, vsftpd 用の SELinux ポリシー
    暗号化, TLS を使用した vsftpd 接続の暗号化
    複数コピーの起動, vsftpd の複数コピーの起動
    起動, vsftpd の起動と停止
    関連資料, 関連資料
        インストールされているドキュメント, インストールされているドキュメント
        オンラインドキュメント, オンラインのドキュメント

W

Web サーバー (参照 Apache HTTP サーバー)

Y

Yum
    plug-ins

aliases, yum プラグインの使用方法
kabi, yum プラグインの使用方法
langpacks, yum プラグインの使用方法
product-id, yum プラグインの使用方法
search-disabled-repos, yum プラグインの使用方法
yum-changelog, yum プラグインの使用方法
yum-tmprepo, yum プラグインの使用方法
yum-verify, yum プラグインの使用方法
yum-versionlock, yum プラグインの使用方法

yum update, 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
yum を使用したパッケージのアンインストール, パッケージの削除
yum を使用したパッケージのダウンロード, パッケージのダウンロード
yum を使用したパッケージの一覧表示
yum list available, パッケージの一覧表示

Yum を使用したパッケージの表示
yum info, パッケージ情報の表示

Yum を使用した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のインストール
yum を使用したパッケージグループの一覧表示
yum groups list, パッケージグループの一覧表示

Yum プラグイン, Yum のプラグイン
[main] オプションの設定, [main] オプションの設定
[repository] オプションの設定, [repository] オプションの設定
パッケージ, パッケージでの作業
パッケージの表示
yum info, パッケージ情報の表示

プラグインの有効化, Yum プラグインを有効、設定、無効にする方法
プラグインの無効化, Yum プラグインを有効、設定、無効にする方法
プラグインの設定, Yum プラグインを有効、設定、無効にする方法
リポジトリー, Yum リポジトリーの追加、有効化、無効化, Yum リポジトリーの作成
変数, Yum 変数の使用

yum
yum と yum リポジトリーの設定, Yum と Yum リポジトリーの設定
yum を使用したインストール, パッケージのインストール
yum を使用したパッケージの一覧表示
glob 表現, パッケージの検索
yum list, パッケージの一覧表示
yum list installed, パッケージの一覧表示
yum repolist, パッケージの一覧表示

yum を使用したパッケージの検索
yum search, パッケージの検索

yumリポジトリー
yum と yum リポジトリーの設定, Yum と Yum リポジトリーの設定

Yum での更新
すべてのパッケージと依存関係の更新, パッケージの更新
セキュリティー関連パッケージの更新, パッケージの更新
パッケージの更新, パッケージの更新
単一パッケージの更新, パッケージの更新
更新の確認, 更新の確認