openSUSE Leap 15.7

AutoYaST ガイド

AutoYaST ガイド
openSUSE Leap 15.7

AutoYaST は openSUSE Leap を無人で一括導入するためのシステムです。AutoYaST のプロファイルには、インストールや設定に関するデータが含まれています。本マニュアルでは、自動インストールに関わる基本手順を説明しています。具体的には 準備とインストール、設定までの分野を説明しています。

発行日: 2026/11/05

SUSE LLC
1800 South Novell Place
Provo, UT 84606
USA
https://documentation.suse.com

Copyright © 2006– 2026 SUSE LLC and contributors. All rights reserved.

訳: SUSE LLC および貢献者が全権利を留保しています。

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or (at your option) version 1.3; with the Invariant Section being this copyright notice and license. A copy of the license version 1.2 is included in the section entitled 「GNU Free Documentation License」.

訳: この文書を、フリーソフトウェア財団発行の GNU フリー文書利用許諾契約書バージョン 1.2 または (希望すれば) 1.3 が定める条件の下で複製、頒布、あるいは改変することを許可します。ただし、この著作権とライセンス表記については変更不可部分とします。この利用許諾契約書の複製物は、「GNU フリー文書利用許諾契約書」という章に含まれています。

For SUSE trademarks, see https://www.suse.com/company/legal/ . All third-party trademarks are the property of their respective owners. Trademark symbols (®, ™ etc.) denote trademarks of SUSE and its affiliates. Asterisks (*) denote third-party trademarks.

訳: SUSE 社の商標については、https://www.suse.com/company/legal/ をご覧ください。その他の商標は各所有者の所有物です。商標シンボル (®, ™ など) は、それぞれ SUSE 社およびその関連会社の商標であることを示しています。また、アスタリスク (*) は第三者の商標を示しています。

All information found in this book has been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. Neither SUSE LLC, its affiliates, the authors nor the translators shall be held liable for possible errors or the consequences thereof.

訳: この書籍内にある全ての情報は、細部に至るまで最大限の注意を払って制作されていますが、完全に正確であることを保証するものではありません。SUSE LLC やその関連会社, 著者, 翻訳者のいずれも、本書籍内の誤りとそこから生じる結果について、一切の保証はいたしません。



注記

なお、本文書は原文 (英語) の翻訳文書であり、公式な文書ではありません。あらかじめご了承ください。

目次

前書き x
1 利用可能なドキュメンテーション x
2 ドキュメンテーションの改善 x
3 文書規約 xi
1 AutoYaST の紹介 1
1.1 AutoYaST の存在理由 1
1.2 概要と考え方 1
I AUTOYAST 制御ファイルの理解と作成 3
2 AutoYaST 制御ファイル 4
2.1 紹介 4
2.2 書式 4
2.3 構造 5
リソースとプロパティ 6 • 入れ子になったリソース設定 6 • 属性 7
3 AutoYaST 制御ファイルの作成 9
3.1 情報の収集 9
3.2 設定管理システム (Configuration Management System; CMS) の使用 9
新しい制御ファイルの作成 10
3.3 手作業による制御ファイルの作成および編集 11
3.4 XSLT を利用したスクリプト経由での制御ファイルの作成 13
3.5 制御ファイルのチェック 14
基本的なチェック 15 • 事前スクリプトの実行 15 • プロファイルの取り込み 15

II AUTOYAST の設定例 17

4 設定およびインストールのオプション 18

4.1 一般オプション 18
モードセクション 19 • インストール設定画面の設定 23 • 自己更新セクション 23 • 半自動化セクション 24 • 署名処理セクション 25 • 待機セクション 27 • general セクションの例 28

4.2 レポート 30

4.3 GRUB 2 ブートローダ 31
ブートローダの種類 32 • グローバル設定 32 • デバイスマップ 37

4.4 systemd ブートローダ 38
ブートローダの種類 38 • グローバル設定 38

4.5 パーティション設定 39
自動パーティション設定 39 • ガイド付きのパーティション設定 40 • 熟練者向けパーティション設定 41 • 高度なパーティション設定 57 • 論理ボリュームマネージャ (Logical Volume Manager (LVM)) 62 • ソフトウエア RAID 63 • マルチパスサポート 69 • bcache の設定 70 • マルチデバイス型 btrfs の設定 72 • NFS の設定 74 • tmpfs の設定 75

4.6 iSCSI イニシエータの概要 75

4.7 Fibre Channel over Ethernet (FCoE) の設定 76

4.8 国ごとの設定 77

4.9 ソフトウエア 78
パターンやパッケージの選択 79 • イメージの配置 79 • 追加のパッケージやカスタマイズしたパッケージ、もしくは独自の製品のインストール 80 • カーネルパッケージ 85 • 自動選択したパッケージの削除 86 • 推奨されるパッケージやパターンのインストール 86 • 第 2 ステージでのパッケージのインストール 87 • 第 2 ステージでのパターンのインストール 87 • 第 2 ステージでのオンライン更新 88

4.10 アップグレード 88

4.11 サービスとターゲット 90

4.12 ネットワークの設定 91
設定の概要 91 • ネットワークリソース 91 • インターフェイス 94 • 複数の IP アドレスの割り当て 99 • ネットワークインターフェイス名の固定 100 • ドメインネームシステム (DNS) 101 • ルーティング 102 • s390 オプション 103

4.13 プロキシ 104

4.14 NIS クライアントおよびサーバ 105

4.15 NIS サーバ 105

4.16 ホスト名の定義 108

4.17 Windows ドメインのメンバーシップ 108

4.18 Samba サーバ 109

4.19 認証クライアント 110

4.20 NFS クライアントおよびサーバ 111

4.21 NTP クライアント 112

4.22 メールサーバの設定 113

4.23 Apache HTTP サーバの設定 115

4.24 squid サーバ 123

4.25 FTP サーバ 129

4.26 TFTP サーバ 133

4.27 firstboot の処理手順 134

4.28 セキュリティ設定 134
パスワードの設定 135 • 起動の設定 135 • ログインの設定 136 • 新規ユーザの設定 (useradd の設定) 136 • Linux セキュリティモジュール (LSM) の設定 136 • OpenSCAP セキュリティポリシーの使用 137

4.29 Linux 監査フレームワーク (Linux Audit Framework (LAF)) 138

4.30 ユーザとグループ 139
ユーザ 139 • ユーザの既定値 145 • グループ 146 • ログインの設定 147

4.31 独自のユーザスクリプト 147
事前スクリプト 148 • パーティション設定後スクリプト 149 • chroot 環境スクリプト 149 • 事後スクリプト 149 • 準備スクリプト 150 • スクリプトの XML パラメータ 151 • スクリプト例 154

4.32 システム変数 (sysconfig) 156

4.33 設定ファイルの直接追加 157

4.34 インストール時におけるユーザへの値の確認 158
既定値スクリプト 164 • スクリプト 164

4.35 カーネルダンプ 168
メモリ予約 170 • ダンプの保存 171 • 電子メール通知 173 • Kdump カーネル設定 174 • 熟練者向け設定 175

4.36 DNS サーバ 176

4.37 DHCP サーバ 178

4.38 ファイアウオール設定 181
一般的なファイアウオール設定 181 • ファイアウオールのゾーン設定 183 • プロファイル内にある `firewalld` 設定が適用されるステージ 184 • 完全な例 185

4.39 その他のハードウエア／システムコンポーネント 186
プリンタ 186 • サウンドデバイス 187

4.40 SSH 鍵と設定の取り込み 188

4.41 設定管理 188
設定管理サーバへの接続 189 • スタンドアロン (単独) モードでの実行 191 • SUSE Multi-Linux Manager Salt Formulas サポートについて 192

III 動的プロファイルによる一括インストールの管理 193
5 動的プロファイルの使用 194
6 ルールとクラス 195
6.1 ルールベースの自動インストール 195
ルールファイルの詳細 196 • 独自のルール 199 • ルールで使用する比較の種類 199 • 属性の組み合わせ 200 • ルールファイルの構造 201 • 定義済みのシステム属性 201 • ダイアログルール 203
6.2 クラス 206
6.3 ルールとクラスの混合 208
6.4 ルールとクラスの合成 208
7 ERB テンプレート 211
7.1 ERB とは? 211
7.2 テンプレートヘルパー 212
boot_efi? 212 • disks 212 • network_cards 213 • os_release 214 • hardware 215
7.3 ERB ヘルパーの実行 215
7.4 ERB プロファイルの生成 216
7.5 ERB プロファイルのデバッグ 217
7.6 ERB とルール／クラスの比較 217
8 ERB テンプレートとスクリプトの組み合わせ 219
8.1 スクリプト内への ERB の組み込み 219
8.2 Ruby スクリプト内からの ERB ヘルパーへのアクセス 220
IV 自動インストール処理の理解 221
9 自動インストール処理 222
9.1 紹介 222
X11 インターフェイス (グラフィカル) 222 • シリアルコンソール 222 • テキストベースの YaST インストール 223

9.2 適切な起動メディアの選択 223
外付けストレージ (USB メモリなど) からの起動 223 • SUSE Linux Enterprise のインストールメディアからの起動 224 • ネットワーク経由での PXE 起動 224

9.3 自動インストール処理の開始 225
コマンドラインオプション 225 • 単一システムの自動インストール 232 • AutoYaST 制御ファイルと `linuxrc` `info` ファイルの組み合わせ 232

9.4 システム設定 233
インストール後およびシステムの設定 233 • システムのカスタマイズ 234

V インストール済みのシステムに対する AUTOYAST の使用 235

10 インストール済みのシステム内での AutoYaST の実行 236

VI 付録 238

A ルールの処理 239

B AutoYaST FAQ - よくある質問とその回答 240

C 高度な `linuxrc` オプション 244

C.1 `linuxrc` へパラメータを渡す方法 245

C.2 `info` ファイルの書式 245

C.3 高度なネットワーク設定 248

D GNU ライセンス 250

前書き

改訂履歴

2023-03-16

1 利用可能なドキュメンテーション

オンラインドキュメンテーション

ドキュメンテーションは https://doc.opensuse.org で公開されています。ここから直接読むこともできますし、様々な形式でダウンロードを行うこともできます。



注記: 最新の更新について

ドキュメンテーションの最新版は通常、英語版が先に作成されます。

SUSE ナレッジベース

何らかの問題に直面した場合は、https://www.suse.com/support/kb/ からアクセスできる技術情報文書 (TID) をご確認ください。ここからお客様からの問い合わせで発生した、様々な SUSE 社の知識情報が検索できます。

お使いのシステム内

オフライン環境での利用を想定して、お使いのシステム内の `/usr/share/doc/release-notes` ディレクトリにはリリースノートが保存されているほか、各パッケージに対するドキュメンテーションが `/usr/share/doc` 内に存在しています。
また、多くのコマンドに対して、マニュアルページ も用意されています。マニュアルページは `man` コマンドで表示することができます。このコマンドの後ろに参照したいコマンド名を指定してください。なお、`man` コマンドがインストールされていない場合は、`sudo zypper install man` を実行してインストールしてください。

2 ドキュメンテーションの改善

このドキュメンテーションに対するご意見だけでなく、改善や追記などの貢献をいただければ幸いです。それぞれ下記のチャンネル経由でお送りいただくことができます:

バグ報告

ドキュメンテーション内に誤記などを見つけた場合は、https://bugzilla.opensuse.org/ で問題を報告してください。

なお、この文書の HTML 版の各章には、問題を報告するための［Report a bug］というアイコンが用意されていますので、こちらをお使いのうえ報告をお願いいたします。これにより、Bugzilla で対象の製品や分類、そしてリンク先などをそれぞれ自動で設定するようになっています。あとは問題点の説明を記述するだけです。
なお、報告には Bugzilla のアカウントが必要となるほか、英語でのやり取りが必要となりますので、あらかじめご了承ください。

貢献

このドキュメンテーションに対して追記もしくは更新すべき具体的な内容をお持ちの場合は、この文書の HTML 版の各章に用意されている［EDIT SOURCE］のリンクをお使いください。これにより、GitHub 内にある 英語版原文の ソースコードを表示し編集することができますので、作業が終わり次第 pull request を送信してください。
なお、GitHub のアカウントが必要となります。



注記: ［Edit Source］は英語版のみに提供される件について

［EDIT SOURCE］のリンクは各文書の英語版原文を編集する目的でのみご利用いただけます。その他の言語については［Report a bug］でお知らせください。

なお、本文書で使用しているドキュメンテーション環境の情報については、リポジトリ内の README をお読みください。

電子メール

本製品のドキュメンテーションに対するフィードバックは、`doc-team@suse.com` でも受け付けております。なお、文書のタイトルと製品のバージョン、およびドキュメンテーションの発行日付をそれぞれご記入ください。また、問題点の報告や記述の追加に関するご提案は、それぞれ概要と対応するセクション番号、およびページ (もしくは URL) をご記入ください。

ヘルプ

openSUSE Leap に対するさらなる支援をご希望の場合は、https://ja.opensuse.org/Portal:Support をご覧ください。

3 文書規約

この文書内では、下記のような記述ルールを使用しています:

- `/etc/passwd` : ディレクトリ名やファイル名を示しています
- `PLACEHOLDER` : `PLACEHOLDER` の箇所は、実際の値に置き換えるべきものであることを示しています

- `PATH` : 環境変数であることを示しています
- `ls` , `--help` : コマンドやオプション、パラメータであることを示しています
- `user` : ユーザ名やグループ名であることを示しています
- **`パッケージ名`** : ソフトウエアのパッケージ名を示しています
- `Alt` , `Alt`–`F1` : キー入力や組み合わせキー入力を示しています; キーはキーボードに書かれているとおりに大文字で示されます
- [ファイル] , [ファイル] › [名前を付けて保存] : メニュー項目やボタンなどを示しています
- 第1章「章のタイトル」: 本ガイド内の他の箇所への参照を示しています。
- 下記は `root` ユーザの権限で実行しなければならないコマンドを示しています。一般ユーザから実行する場合は、これらのコマンドの前に `sudo` を付けることで、`root` で実行できるようになります:

```
# コマンド
> sudo コマンド
```

- 下記は一般ユーザで実行できるコマンドを示しています:

```
> コマンド
```

- また、行末にバックスラッシュ文字 (`\`) を付けることで、コマンドを複数行に分けて記述している場合もあります。バックスラッシュ文字は、シェルに対して、これ以降にもコマンドが続くことを示す文字になります:

```
> echo a b \
c d
```

- このほか、行頭にプロンプトが書かれたコマンド行に続いて、そのコマンドを実行した場合の出力例を示す場合もあります:

```
> コマンド
出力
```

- 各種の情報について



警告: 警告

実際に実施したりする前に、注意しておかなければならない、きわめて重要な情報を記述しています。セキュリティ面の問題のほか、データを失ってしまう可能性への告知、ハードウエアの損傷の可能性や物理的な障害が発生する可能性を示しています。



重要: 重要な情報

実際に実施する前に注意すべき点を説明しています。



注記: 一般的な情報

一般的な補足情報を示しています。たとえばソフトウエアバージョン間での違いなどを説明しています。



ヒント: その他のヒント

ガイドラインや実践的なアドバイスなど、ヒントとなる情報を示しています。

- 簡潔な補足情報



一般的な補足情報を示しています。たとえばソフトウエアバージョン間での違いなどを説明しています。



ガイドラインや実践的なアドバイスなど、ヒントとなる情報を示しています。

1 AutoYaST の紹介

改訂履歴

2022-04-05

1.1 AutoYaST の存在理由

openSUSE Leap の標準的なインストール方式はウイザード形式になっています。これは一般的なユーザにとっては分かりやすいもので、少数のマシンにインストールするだけであれば、十分に効率的な仕組みです。ところが、多数のマシンにインストールを行う場合、このようなウイザード形式は手間になってしまうばかりか、時間のかかるものになってしまいます。

このような問題を避けるには、特定のマシンにインストールしたものをハードディスクごと複製して配置する方法もあります。ただし、このような仕組みで大量配置を行う場合、配置後のさまざまな個別設定が面倒になってしまいます。たとえばそれぞれのマシンに対して固定の IP アドレスを割り当てるような場合、配置後にそれぞれの IP アドレスを設定する手間が生じることになってしまいます。

openSUSE Leap の通常のインストールであっても、半自動化が実現できています。インストールの冒頭で必要な情報 (通常は言語) を選択したあと、YaST はさまざまな要素とシステムのパラメータを考慮して提案を作成しますので、そのまま進めていくだけでインストールが完了します。提案内容を受け入れたあとは、残りの作業は自動的に実施されます。

AutoYaST はユーザ側での操作を一切不要にしたい場合や、カスタマイズが必要な場合に有用です。AutoYaST のプロファイルを使用することで、YaST は独自の方式でインストールを行い、プロファイル内に敢えて記述されている場合を除き、ユーザに一切尋ねることなく処理を進めることができます。

AutoYaST は GUI システムの自動化ではありません。そのため、通常は表示される多くの画面 (たとえば言語選択の画面など) が表示されないまま処理が行われます。AutoYaST では言語関連のユーザインターフェイスを表示することなく、サブシステムに言語パラメータを渡して処理を進めます。

1.2 概要と考え方

AutoYaST を使用することで、多数のシステムを同時並行で素早くインストールすることができるようになります。これらは同じ環境を共有する必要がありますが、ハードウエアについては全く同一でなくてもかまいません。インストール処理は XML 形式の設定ファイル (通常は `autoinst.xml` というファイル名です) 、「AutoYaST プロファイル」で制御します。プロファイルは既存の設定リソースから作成しておいて、後から必要に応じて調整することができます。

AutoYaST はシステムに完全に統合されているほか、インストールや設定に対してさまざまなオプションを提供しています。その他の自動インストールシステムと比較しても、既存のモジュールを利用してコンピュータを設定することができたり、通常はインストールの終了時に実行するスクリプトなどを設定できたりする点がメリットとなっています。

本文書では、自動インストールに関わる 3 種類のステップを説明しています:

- 準備: インストール先のシステムに関する全ての情報を収集し、プロファイル内の対応するディレクティブに変換します。プロファイルはインストール先のシステムに転送され、ディレクティブを処理したあと YaST に渡されます。
- インストール: YaST は AutoYaST プロファイルに書かれたデータをもとにして、基本的なシステムのインストールと設定 (パーティション、ネットワーク、ファイアウオールの各設定など) を行います。
- 事後設定: システムへのインストールと基本設定が完了すると、システムは残りの設定を行うために第 2 ステージに移行することができます。ここではインストール後スクリプトの動作やサードパーティ製モジュールの動作、そしてその他の YaST モジュールの処理などが含まれます。



注記: 第 2 ステージについて

openSUSE Leap 15.7 の通常のインストールは、単一のステージで実施されますが、自動インストールの場合にのみ 2 つのステージに分割して実行されます。インストールと基本的なシステムの設定が終わったあと、システムは第 2 ステージとしてインストール後のシステムを起動します。

第 2 ステージで自動インストールを正しく動作させるには、インストール先のシステムに `autoyast2` と `autoyast2-installation` のパッケージをインストールしておく必要があります。これらをインストールしておかないと、インストール済みのシステムを起動する際にエラーメッセージが表示されます。

なお、第 2 ステージはどうしても必要な場合にのみ動作します。第 2 ステージを無効化したい場合は、`second_stage` パラメータを設定してください:

```
<general>
  <mode>
    <confirm config:type="boolean">false</confirm>
    <second_stage config:type="boolean">false</second_stage>
  </mode>
</general>
```

I AutoYaST 制御ファイルの理解と作成

2 AutoYaST 制御ファイル 4

3 AutoYaST 制御ファイルの作成 9

2 AutoYaST 制御ファイル

改訂履歴

2025-03-28

2.1 紹介

制御ファイル は プロファイル とも呼ばれ、単一のシステムに対する設定を記述します。このファイルにはリストやレコード、ツリーや巨大組み込み形式や参照オブジェクトなど、複雑な構造に対応したプロパティリソースから構成されています。

2.2 書式

設定ファイルの形式に XML を採用していることにより、ファイル構造の一貫性が保たれているほか、新しいシステムを設定しようとする場合でも、学びやすく覚えやすい仕組みになっています。

AutoYaST の制御ファイルでは、XML を利用してシステムのインストールや設定を表現します。XML は汎用目的のマークアップ言語で、多くのユーザが慣れ親しんでいる構造であるほか、XML ファイルを処理するためのツールも多数用意されています。既存の制御ファイルを編集する場合でも、何もない状態から新しい制御ファイルを作成する場合でも、制御ファイルの構造が正しいことを検証しておくことを強くお勧めします。これは `xmllint` や `jing` のような検証型 XML パーサーを使用することで行うことができます (たとえば 3.3項「手作業による制御ファイルの作成および編集」をご覧ください) 。

下記は XML 形式での制御ファイルの例です:

例 2.1: AUTOYAST の制御ファイル (プロファイル)

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile
  xmlns="http://www.suse.com/1.0/yast2ns"
  xmlns:config="http://www.suse.com/1.0/configns">
  <partitioning config:type="list">
    <drive>
      <device>/dev/sda</device>
      <partitions config:type="list">
        <partition>
          <filesystem config:type="symbol">btrfs</filesystem>
          <size>10G</size>
```

```
          <mount>/</mount>
        </partition>
        <partition>
          <filesystem config:type="symbol">xfs</filesystem>
          <size>120G</size>
          <mount>/data</mount>
        </partition>
      </partitions>
    </drive>
  </partitioning>
  <scripts>
    <pre-scripts>
      <script>
        <interpreter>shell</interpreter>
        <filename>start.sh</filename>
        <source>
        <![CDATA[
#!/bin/sh
echo "Starting installation"
exit 0

]]>

        </source>
      </script>
    </pre-scripts>
  </scripts>
</profile>
```

2.3 構造

下記は制御ファイルのコンテナで、この中に実際の設定を書き込んでいくものです。

例 2.2: 制御ファイルのコンテナ

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile
  xmlns="http://www.suse.com/1.0/yast2ns"
  xmlns:config="http://www.suse.com/1.0/configns">
  <!-- リソース -->
</profile>
```

`<profile>` の要素 (ルートノード) には、1 つもしくは複数のリソース要素を追加します。指定可能なリソース要素は、スキーマファイル内で指定されています。

2.3.1 リソースとプロパティ

リソース要素には複数の重複しないプロパティ要素や子リソース要素を含めることができるほか、同じ子リソース要素を複数インスタンスとして追加することもできます。もちろん何も含めない指定も行うことができます。リソース要素内に含めることのできる要素は、スキーマファイル内で指定されています。

プロパティ要素には値としてリテラルを指定することができます。もちろん値を指定しなくてもかまいません。それぞれのリソース要素内で指定可能なプロパティ要素とその値についても、スキーマファイル内で指定がされています。

要素は他の要素に対するコンテナとする (リソース) ことができるほか、値としてリテラルを持つ (プロパティ) こともできますが、両方を同時に使用することはできません。この制限についても、スキーマファイル内に記述されています。1 つ以上の値を持つ設定のコンポーネントは、プロパティ値内で組み込みのリストとして表現するか、入れ子型のリソースとして表現します。

`<foo></foo>` や `<bar/>` のような空要素は、処理済みのデータモデル内に存在すべきでは *ありません* 。通常、これらには適切な既定値を設定すべきです。意図的に空要素を指定したい場合は、CDATA セクションを使用して `<foo><![CDATA[]]></foo>` のように指定し、空文字列を設定してください。

2.3.2 入れ子になったリソース設定

入れ子になったリソース要素を使用することで、任意の階層構造からなるツリー状の設定コンポーネントを作成することができます。

入れ子になったリソースとしては、マップとリストの 2 種類が存在します。マップ (連想配列やハッシュ、辞書と呼ばれることもあります) にはタグ名とそれに対応する値の対を含めることができます。リスト (配列と呼ばれることもあります) には、単純に値の羅列を含めることができます。

例 2.3: 入れ子になったリソース設定

```
...
<drive>
  <device>/dev/sda</device>
  <partitions config:type="list">
     <partition>
        <size>10G</size>
        <mount>/</mount>
     </partition>
     <partition>
        <size>1G</size>
        <mount>/tmp</mount>
     </partition>
  </partitions>
</drive>
```

```
....
```

上記の例では、`drive` リソースは `device` プロパティと `partitions` リソースから構成されるマップの形態になっています。`partitions` リソースは複数の `partition` から構成されるリストで、その `partition` リソースは `size` と `mount` のプロパティを含むマップの形態になっています。

入れ子になったリソースの場合、既定ではマップとして処理されます。リストを使用したい場合は、`config:type="list"` の属性を明示的に指定しなければなりません。



ヒント: 要素型の省略表記について

openSUSE Leap 15.3 およびそれ以降のバージョンでは、`config:type` の代わりに `t` という属性を使用することで、要素の型を指定することができます。

```
<mode t="boolean">true</mode>
```

2.3.3 属性

グローバル属性は、リソースやプロパティ内のメタデータを定義する際に使用します。属性はコンテキストの切り替えを定義するために使用します。これらは上記の章に示されているプロパティの名前付けや型指定で使用されます。属性については独自のネームスペース内に存在していることから、既定のネームスペース内で予約語として扱う必要はありません。

`config:type` 属性は処理済みのデータモデル内で、リソースやプロパティの型を設定するためのものです。リソースの場合、リスト型を指定したい場合は `list` を設定する必要がありますが、マップ型の場合は既定の型であるため、属性の設定は不要となります。ただし、中身無しの (空の) マップを指定する場合は、単純な文字列として処理されてしまうことの無いよう、マップ型を指定しておく必要があります。

例 2.4: 空のマップ

```
<general t="map" />
```

プロパティの場合、`boolean` (ブール値)、`symbol` (シンボル)、`integer` (整数) の各型を指定することができます。こちらの場合の既定値は `string` (文字列) となります。

上記で説明したとおり、マップと文字列の例外を除いて、属性は基本的に任意指定ではありません。ただし、スキーマ内のさまざまな場所でさまざまな指定がされていて一貫性が無いことから、中には任意指定の属性も存在します。また、場所によっては、値を列挙するのにシンボルで表す場合があるほか、文字列を必要とするような場合も存在します。また一方のリソースでは `config:type="integer"` の

指定が必要となるのに対して、他方では文字列属性で数値を処理する箇所もあります。またリソースによっては `config:type="boolean"` を使用するのに対して、他のリソースでは `yes` や `1` などを使用することもあります。詳しくはスキーマファイルをご覧ください。

3 AutoYaST 制御ファイルの作成

改訂履歴

2023-12-22

3.1 情報の収集

制御ファイルを作成するには、まずインストールしたいシステムに関して情報の収集を行う必要があります。この情報収集にはハードウエアデータやネットワーク情報などが含まれます。まずはインストール予定のマシンに対して、下記のような情報を収集してください:

- ハードディスクの種類とサイズ
- グラフィカルなインターフェイスと接続されているモニタ (もしあれば)
- 既知であればネットワークインターフェイスと MAC アドレスの情報 (特に DHCP を使用するような場合)

なお、`autoyast2-installation` および `autoyast2` の両パッケージがインストールされていることも確認しておいてください。

3.2 設定管理システム (Configuration Management System; CMS) の使用

1 台もしくは複数台のコンピュータに対して制御ファイルを作成する目的で、YaST をベースにした設定インターフェイスが提供されています。このシステムは、openSUSE Leap を通常の手順でインストールしたり設定したりする際に使用する、既存のモジュールをベースにして作られています。

設定管理システムを使用することで、制御ファイルを簡単に作成することができるほか、複数のクライアントからアクセスすることができるようにするため、ネットワーク環境内で設定のリポジトリを管理したりすることもできます。

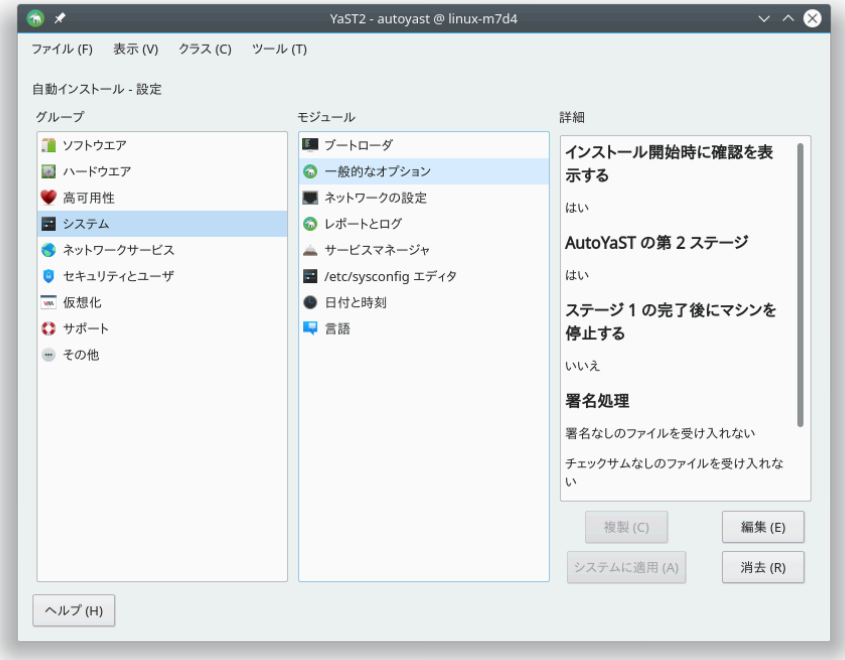

図 3.1: 設定システム

3.2.1 新しい制御ファイルの作成

AutoYaST のプロファイルを作成するのに最も簡単な方法は、既存の openSUSE Leap システムを雛形として使用することです。既にインストール済みのシステムから［YaST］›［その他］›［自動インストールの設定］を選択して、メニューバーから［ツール］›［参照プロファイルの作成］を選択してください。あとはプロファイル内に含めたいシステムコンポーネントを選択するだけです。それ以外にも、［YaST］›［その他］›［自動インストール向けのシステム複製］を選択するか、もしくはコマンドラインから `sudo yast clone_system` を実行すると、現在のシステムの設定全てを含むプロファイルを作成することもできます。

いずれの方法とも `/root/autoinst.xml` ファイルを作成することになります。システム複製でプロファイルを作成した場合は、作成したマシンと全く同じクローンを作成するプロファイルになります。ただし、通常は全く同じシステムを構成するのではなく、少しずつ異なるシステムを構築するのが一般的であるため、お使いの XML エディタなどで細かい微調整を行うことになります。



警告: プロファイル内の機密情報について

プロファイル内には、たとえばパスワードハッシュや登録キーなど、機密情報が含まれる可能性があることに注意してください。

また、出力されたプロファイルはよくご確認のうえ、ファイルのパーミッションを適切に設定するようにしてください。

いくつかの例外を除いて、ほぼ全ての制御ファイルのリソースは、設定管理システムを利用することで設定することができます。システムは柔軟に作られていて、設定を作成する際の手順も、通常の YaST コントロールセンターと同じユーザインターフェイスになっています。既存のものと同じユーザインターフェイスであるだけでなく、パーティション設定や汎用オプション、ソフトウエアなど、特殊で複雑な設定を作成するための新しいインターフェイスも提供されています。

これに加えて、CMS では生成された制御ファイルの正当性を確認する仕組みも備えていますので、生成された制御ファイルをそのまま自動インストールに使用することができます。

まずは設定管理システム (`autoyast2` パッケージ) がインストールされていることを確認し、YaST コントロールセンターから AutoYaST を起動するか、もしくは root で下記のコマンドを実行します (なお、グラフィカルな表示を行いたい場合は、`DISPLAY` 環境変数を正しく設定していることを確認してから実行してください):

```
/sbin/yast2 autoyast
```

3.3 手作業による制御ファイルの作成および編集

制御ファイルを手作業で編集する場合は、まず書式が正しいことを確認する必要があります。書式を確認するには、お使いのディストリビューションに既に用意されているツールをお使いください。たとえばファイルの書式が正しいこと (XML の構造に間違いがないこと) を確認したい場合は、`libxml2` パッケージ内に含まれる `xmllint` ユーティリティを使用します:

```
xmllint <制御ファイル>
```

制御ファイルの書式に問題が見つかると (たとえばタグが正しく閉じられていないなど) 、`xmllint` はエラーを報告します。

制御ファイルの内容を確認したい場合は、同じパッケージ内に含まれる `jing` ツールをお使いください。確認の際、タグ配置の誤りのほか、必須のタグや属性の確認や属性値の確認も行われます。

```
jing /usr/share/YaST2/schema/autoyast/rng/profile.rng <制御ファイル>
```

`/usr/share/YaST2/schema/autoyast/rng/profile.rng` ファイルは `yast2-schema-default` パッケージ内に含まれています。このファイルには、AutoYaST プロファイルの書式とクラスに関する情報が書かれています。



注記: スキーマの拡張について

AutoYaST は多の製品やモジュールを利用することで拡張を行うことができますが、スキーマ側にはそれらの拡張に対応するための仕様が含まれていません。そのため AutoYaST に対して拡張を使用するプロファイルを指定した場合、プロファイルが正しくないものとして扱われてしまう場合があります。

このような事情から、openSUSE Leap 15.3 およびそれ以降では、AutoYaST はトップレベルに未知のセクションが存在していても、それらを無視するようになっています。たとえば下記の例でいえば、`<sap-inst>` 内は検証されず無視されるものの、残りの箇所は通常通り検証が行われます。

```
<general>
  <mode>
    <confirm config:type="boolean">true</confirm>
  </mode>
</general>

<sap-inst>
  <!-- this section is not validated -->>
</sap-inst>
```

自動インストールを行う前に、これらのチェックで表示されたエラーを全て修正してください。制御ファイルの書式や形式が正しくない場合、自動インストールの処理を開始することはできません。

制御ファイルの編集にあたっては、お使いのシステムで利用可能な XML エディタを使用するか、もしくは XML に対応したテキストエディタ (例: Emacs, Vim) をお使いください。ただし、複数のマシンに対して制御ファイルを作成する場合は最適な選択であるとは言えないため、設定管理システム (CMS) を併用することをお勧めします。



ヒント: XML エディタとしての Emacs の使用について

Emacs に内蔵された nxml-mode を使用すると、タグの補完機能や検証機能も存在する、完全機能の XML エディタとして使用することができます。nxml-mode の設定方法について、詳しくは Emacs のヘルプをお読みください。

3.4 XSLT を利用したスクリプト経由での制御ファイルの作成

雛形となる制御ファイルが既に存在していて、その中の一部をスクリプトやコマンドラインで修正したい場合は、`xsltproc` のような XSLT プロセッサを使用することをお勧めします。たとえば AutoYaST の制御ファイル内のホスト名を、スクリプトを介して修正したいような場合がそれにあたります (ホスト名を設定するような場合は、それをスクリプト化することをお勧めします) 。

まずは XSL ファイルを作成します:

例 3.1: スクリプトでホスト名やドメイン名を置き換える場合の例

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:y2="http://www.suse.com/1.0/yast2ns"
  xmlns:config="http://www.suse.com/1.0/configns"
  xmlns="http://www.suse.com/1.0/yast2ns"
  version="1.0">
  <xsl:output method="xml" encoding="UTF-8" indent="yes" omit-xml-declaration="no" cdata-
section-elements="source"/>

  <!-- パラメータ名 -->
  <xsl:param name="hostname"/>
  <xsl:param name="domain"/>

  <xsl:template match="/">
    <xsl:apply-templates select="@*|node()"/>
  </xsl:template>

  <xsl:template match="y2:dns">
    <xsl:copy>
      <!-- where to copy the parameters -->
      <domain><xsl:value-of select="string($domain)"/></domain>
      <hostname><xsl:value-of select="string($hostname)"/></hostname>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>


  <xsl:template match="@*|node()" >
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```

このファイルでは、ホスト名とドメイン名をパラメータとして求めるようになっています。

```
<xsl:param name="hostname"/>
<xsl:param name="domain"/>
```

これらの情報は、同じ制御ファイル内の DNS セクション内に書かれているものと同じです。つまり、DNS セクション内に既にドメイン要素が存在する場合、ここにもう 1 つの設定を記述することになってしまいます。これでは期待通りの動作にならないことがあります。

XSLT に関する詳細については、公式 Web ページ www.w3.org/TR/xslt (https://www.w3.org/TR/xslt) (英語) をお読みください。

3.5 制御ファイルのチェック

用途にもよりますが、AutoYaST のプロファイルを作成する作業は、特にルールやクラス、ERB テンプレートや事前スクリプトを使用する必要が生じた場合は、複雑なものになってしまいます (動的なプロファイルに関する詳細は、パートIII「動的プロファイルによる一括インストールの管理」をお読みください) 。このような複雑なプロファイルのテストやデバッグの作業を簡単にするため、AutoYaST には `check-profile` と呼ばれるコマンドが用意されています。これは取得や構築のほか、必要であれば発生しうる問題を検出するためのプロファイル取り込みにも対応しています。

openSUSE Leap 15.3 およびそれ以降のバージョンでは、AutoYaST はインストール時にもプロファイルの検証を行い、その結果をユーザに報告します。この動作はそのままにしておくことが推奨されますが、無効化したい場合は `YAST_SKIP_XML_VALIDATION` という起動パラメータを `1` に設定してください。

これに加えて、テストやデバッグの作業を簡単にできるようにするため、AutoYaST では `check-profile` というコマンドを提供しています。このコマンドはプロファイルの取得や構築のほか、必要であれば取り込みの処理などを行って、潜在的な問題点を抽出することができます。

注記: 処理結果の違いについて

このコマンドはインストール処理と同じアプローチを使用して動作しますが、現在のシステムにインストールされている YaST のパッケージバージョンやアーキテクチャなどが、インストールメディアにインストールされているものと異なる場合、処理結果が異なる場合があります。

警告: 信頼できるプロファイルのみに限られる件について

また、事前スクリプトや ERB のコードは `root` ユーザとして実行されることから、信頼できるプロファイルのみを使用するようにしてください。

3.5.1 基本的なチェック

このコマンドの最も簡単な使用方法は、プロファイルを読み込んで検証する処理です:

```
> sudo  yast2 autoyast check-profile filename=autoinst.xml output=result.xml
```

`result.xml` ファイルには、プロファイルを評価した結果が書き込まれます。ただし、高度な機能を使用していなかった場合であっても、`autoinst.xml` と `result.xml` の内容は異なる場合があることに注意してください。これは、AutoYaST がプロファイルを処理する際に、いくつかのクリーンアップ処理を実施するためです。

`check-profile` はリモートのファイルを扱うこともできます:

```
> sudo  yast2 autoyast check-profile filename=http://192.168.1.100/autoinst.xml
 output=result.xml
```

3.5.2 事前スクリプトの実行

必要であれば、AutoYaST はプロファイル内にスクリプトを内蔵させて実行することができます。また、その実行時に検出されたエラーを報告することもできます。これは特に、事前スクリプト内でプロファイルを修正しているような場合に有用な仕組みです。この機能を有効化するには、`run-scripts` オプションを `true` にします。

```
> sudo  yast2 autoyast check-profile filename=http://192.168.1.100/autoinst.xml
 output=result.xml run-scripts=true
```



警告: スクリプトが root で実行される件について

なお、`run-scripts` オプションでスクリプトの実行を許可すると、スクリプトは root 権限で実行されることになりますので、現在お使いのシステムに悪い影響が発生する可能性があることに注意してください。

3.5.3 プロファイルの取り込み

プロファイルの内容が正しい場合であっても、プロファイルの取り込み時に何らかの問題が発生することがあります。これは AutoYaST がプロファイルを取得して構築し、検証する処理で、論理的なチェックまでは行っていないことによるものです。

このような問題の発生を予想させたい場合は、`check-profile` コマンドでプロファイルの取り込みを実行し、検出された問題を報告するようにしてください。この処理にはしばらくの時間がかかりますので、`import-all` オプションを `false` に設定して、この処理を無効化することもできます。

```
> sudo  yast2 autoyast check-profile filename=http://192.168.1.100/autoinst.xml
 output=result.xml import-all=false
```

なお、プロファイルの取り込みは安全な処理であり、取り込みを実行するシステムに影響を与えることはありません。

II AutoYaST の設定例

4 設定およびインストールのオプション 18

4 設定およびインストールのオプション

改訂履歴

2023-11-23

本章ではサービスや登録、ユーザやグループの管理、アップグレードやパーティション設定、各種設定の管理や SSH 鍵の管理、ファイアウオールの設定やその他のインストールオプションについて説明しています。

本章では、一般的な用途で使用する制御ファイル内の主要なパーツについて説明しています。その他の利用可能なオプションについて知るには、設定管理システム (CMS) をお使いください。

なお、オプションによっては追加のパッケージをインストールしなければならないものもあります。お使いのシステムをインストールする際、最小限のパッケージ選択を行っていると、必要なパッケージがインストールされていない場合がありますので、この場合は個別にパッケージを選択してインストールしてください。

YaST では、AutoYaST のインストール後フェーズの開始前に、インストールの第 2 フェーズで必要なパッケージをインストールします。しかしながら、システム側で必要な YaST モジュールがインストールされていないと、主要な設定ステップが飛ばされてしまいます。たとえば `yast2-security` パッケージがインストールされていない場合、セキュリティ関連の設定は行われません。

4.1 一般オプション

`general` セクションには、インストールの処理手順全体に影響する全ての設定が書かれています。このセクションの全体像は下記のとおりです:

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns"
 xmlns:config="http://www.suse.com/1.0/configns">
 <general>
  <ask-list>❶
   ...
  </ask-list>
  <cio_ignore>
   ...
  </cio_ignore>
  <mode>❷
   ...
  </mode>
  <proposals>❸
   ...
  </proposals>
```

```
  <self_update>❹
  ...
 </self_update>
  <self_update_url>
   ...
  </self_update_url>
  <semi-automatic config:type="list">❺
  ...
  </semi-automatic>
  <signature-handling>❻
   ...
  </signature-handling>
  <storage>❼
   ...
  </storage>
  <wait>❽
   ...
  </wait>
 </general>
<profile>
```

❶ 4.34項「インストール時におけるユーザへの値の確認」

❷ 4.1.1項「モードセクション」

❸ 4.1.2項「インストール設定画面の設定」

❹ 4.1.3項「自己更新セクション」

❺ 4.1.4項「半自動化セクション」

❻ 4.1.5項「署名処理セクション」

❼ 4.5項「パーティション設定」

❽ 4.1.6項「待機セクション」

4.1.1 モードセクション

モードセクションでは、ユーザに対する問い合わせや再起動に関する AutoYaST の動作を設定します。`mode` セクション内では下記の要素を設定することができます:

`activate_systemd_default_target`

この項目を `false` に設定すると、既定の `systemd` のターゲットが `systemctl isolate` で有効化されなくなります。この値の指定は任意で、既定値は `true` です。

```
<general>
 <mode>
  <activate_systemd_default_target config:type="boolean">
```

```
   true
  </activate_systemd_default_target>
 </mode>
 ...
</general>
```

`confirm`

既定では、インストール処理は［インストール設定］の画面で停止します。この時点では、まだシステムに対して何も変更が加えられておらず、必要であれば、この画面から各種の設定を変更することができます。ここから先に進むことで実際のインストールが始まることから、ここでユーザへの確認を行っています。この値を `false` にすると、設定内容は自動的に受け入れられ、インストールが始まることになります。そのため、完全な無人インストールを行いたい場合にのみ、`false` を指定してください。この値の指定は任意で、既定値は `true` です。

```
<general>
 <mode>
  <confirm config:type="boolean">true</confirm>
 </mode>
 ...
</general>
```

`confirm_base_product_license`

この値を `true` に設定すると、基本製品の EULA が表示されるようになります。ユーザ側では、表示されたライセンスを受け入れる必要があります。受け入れない場合、インストールはキャンセルされます。この値の設定は任意で、既定値は `false` です。この設定は基本製品のライセンスにのみ適用されます。追加のライセンスについては、`add-on` セクション内の `confirm_license` フラグをお使いください (詳しくは 4.9.3項「追加のパッケージやカスタマイズしたパッケージ、もしくは独自の製品のインストール」をお読みください) 。

```
<general>
 <mode>
  <confirm_base_product_license config:type="boolean">
   false
  </confirm_base_product_license>
 </mode>
 ...
</general>
```

`final_halt`

この値を `true` に設定すると、必要なパッケージのインストールと 第 2 ステージでの 設定が終わったあとに、マシンがシャットダウンするようになります。`final_halt` を設定した場合は、`final_reboot` オプションを `true` に設定する必要はありません。

```
<general>
```

```
 <mode>
  <final_halt config:type="boolean">false</final_halt>
 </mode>
 ...
</general>
```

`final_reboot`

この値を `true` に設定すると、必要なパッケージのインストールと 第 2 ステージでの 設定が終わったあとに、マシンが再起動するようになります。`final_reboot` を設定した場合は、`final_halt` オプションを `true` に設定する必要はありません。

```
<general>
 <mode>
  <final_reboot config:type="boolean">true</final_reboot>
 </mode>
 ...
</general>
```

`final_restart_services`

この項目を `false` に設定すると、インストールの最後 (全てのパッケージをインストールし、第 2 ステージの末尾での 設定が終わったあと) にサービス類が再起動 されなく なります。この値の設定は任意で、既定値は `true` です。

```
<general>
 <mode>
  <final_restart_services config:type="boolean">
   true
  </final_restart_services>
 </mode>
 ...
</general>
```

`forceboot`

openSUSE のリリースによっては、第 1 ステージ終了後の再起動を避けるため、Kexec を利用して居るバージョンがあります。これにより、インストール済みのシステムを即時に起動できるようになっています。この場合、この値を `true` に設定することで、Kexec ではなく再起動を行うようにすることができます。この値の設定は任意で、既定値は製品によって異なります。

```
<general>
 <mode>
  <forceboot config:type="boolean">false</forceboot>
 </mode>
 ...
</general>
```

重要: ドライバによっては再起動が必要となる件について

nVidia 社, ATI 社のグラフィックカード向けのプロプライエタリドライバなどを使用している場合、Kexec では正しく動作させることができず、必ず再起動が必要となります。そのため、openSUSE Leap 製品の既定では、必ず再起動を行うようになっています。

`halt`

第 1 ステージの終了後にマシンをシャットダウンします。この段階では、全てのパッケージとブートローダのインストールが完了し、chroot スクリプトの指定があれば、実行後の状態になります。これを指定すると、再起動を行って第 2 ステージを開始する代わりに、マシンの電源が落とされます。その後マシンの電源を投入すると、マシンは第 2 ステージの自動インストールが始まるようになります。この値の設定は任意で、既定値は `false` です。

```
<general>
 <mode>
  <halt config:type="boolean">false</halt>
 </mode>
 ...
</general>
```

`max_systemd_wait`

`systemd` が既定のターゲットの準備を整える際、AutoYaST が最大で待機する時間を秒単位で指定します。この値の設定は任意で、通常は設定不要です。既定値は `30` (秒) です。

```
<general>
 <mode>
  <max_systemd_wait config:type="integer">30</max_systemd_wait>
 </mode>
 ...
</general>
```

`ntp_sync_time_before_installation`

インストールを開始する前に時刻同期を行うための、NTP サーバを指定します。時刻同期は、このオプションを設定した場合にのみ行われます。なお、NTP サーバとの通信にはネットワークの接続と、NTP サーバへの経路情報の設定が必要となります。この値の設定は任意で、既定では時刻同期を行いません。

```
<general>
     <mode>
      <ntp_sync_time_before_installation>
       &ntpname;
      </ntp_sync_time_before_installation>
     </mode>
     ...
```

```
    </general>
```

`second_stage`

通常の openSUSE Leap のインストールは、単一のステージ内で行われますが、自動インストールの場合は、2 つのステージに分割して実行されます。自動インストールでは基本的なシステムのインストールを行ったあと、システムの設定を行うために第 2 ステージを起動します。このオプションを `false` に設定すると、第 2 ステージを無効化することができます。この値の設定は任意で、既定値は `true` です。

```
<general>
 <mode>
  <second_stage config:type="boolean">true</second_stage>
 </mode>
 ...
</general>
```

4.1.2 インストール設定画面の設定

AutoYaST では、インストールに関する設定の概要を表示する［インストール設定］の画面をカスタマイズすることができます。この画面では、実際のインストールが始まる前に、さまざまな設定を変更することができます。`proposal` タグを使用することで、インストール設定の画面に表示される設定 (「提案」) を制御することができます。表示することのできる提案項目の一覧については、インストールメディア内の `/control.xml` ファイルをご覧ください。この設定は任意で、既定では全ての設定オプションを表示します。

```
<proposals config:type="list">
 <proposal>partitions_proposal</proposal>
 <proposal>timezone_proposal</proposal>
 <proposal>software_proposal</proposal>
</proposals>
```

4.1.3 自己更新セクション

インストールの際、YaST は自分自身を更新して、リリース後に発見されたインストーラ自身のバグを解決することができます。この機能に関する詳細について、詳しくは 配置ガイド をお読みください。

YaST の自己更新機能を制御するには、下記のタグを設定してください:

`self_update`

このオプションを `true` もしくは `false` に設定することで、YaST の自己更新機能を有効化もしくは無効化することができます。この値の設定は任意で、既定値は `true` です。

```
<general>
 <self_update config:type="boolean">true</self_update>
 ...
</general>
```

カーネルのコマンドラインに対して、`self_update=1` の起動パラメータを設定しても、同じ動作になります。

`self_update_url`

YaST の自己更新機能を提供する更新リポジトリの場所を指定します。詳しくは『スタートアップ』、第3章「インストール手順」、3.2.2項「独自の自己更新リポジトリ」をお読みください。

重要: インストーラ自身の更新リポジトリのみを設定する必要がある件について

`self_update_url` パラメータでは、インストーラの自己更新を提供する URL のみを指定します。ソフトウエアの更新リポジトリなど、その他のリポジトリを指定してはなりません。

```
<general>
 <self_update_url>
  http://example.com/updates/$arch
 </self_update_url>
 ...
</general>
```

URL には `$arch` という変数を含めることができます。これはシステムのアーキテクチャに置き換えられる箇所を示すもので、たとえば `x86_64` , `s390x` などに置き換えられます。
それ以外の方法としては、カーネルのコマンドラインに対して、`self_update=1` と `self_update=URL` の起動パラメータを設定しても、同じ動作になります。

4.1.4 半自動化セクション

AutoYaST では、インストール時にいくつかの YaST モジュールを起動する機能を提供しています。これにより、いくつかの項目については手動で設定し、それ以外の全てを自動的に設定することができます。この半自動化セクションでは、下記のような YaST モジュールを起動することができます:

- ネットワーク設定モジュール (`networking`)
- パーティション設定モジュール (`partitioning`)
- 登録モジュール (`scc`)

下記の例は、インストール時に対応する 3 種類全てのモジュールを起動する場合の例です:

```
<general>
 <semi-automatic config:type="list">
  <semi-automatic_entry>networking</semi-automatic_entry>
  <semi-automatic_entry>scc</semi-automatic_entry>
  <semi-automatic_entry>partitioning</semi-automatic_entry>
 </semi-automatic>
</general>
```

4.1.5 署名処理セクション

既定では、AutoYaST は既知の GPG 鍵で署名されたソースからの署名付きパッケージのみをインストールします。本セクションでは、この既定値を変更することができます。



警告: 署名処理設定の変更に伴う危険性について

未署名のパッケージやチェックサムの確認が失敗したパッケージ、もしくは信頼していないソースからのパッケージを受け入れてしまうと、セキュリティ上のリスクとなります。このようなパッケージを受け入れてしまうと、パッケージが改ざんされていることを検出できなくなるほか、お使いのマシンに悪意のあるソフトウエアをインストールする可能性が生じてしまいます。ここで説明している既定値の上書きは、リポジトリとその中にあるパッケージの両方が信頼できるものである場合にのみ、設定してください。SUSE では、これらの正当性チェックを無効化した状態でソフトウエアをインストールし使用した場合、一切の保証を行いません。

下記に示す全オプションの既定値は false です。オプションが false に設定されていた場合、それに対応するパッケージやリポジトリのチェックが失敗すると、特にメッセージを表示することなくインストールを行わなくなります。

`accept_unsigned_file`

`true` に設定した場合、AutoYaST はコンテンツファイルなどが未署名であっても、受け入れるようになります。

```
<general>
 <signature-handling>
  <accept_unsigned_file config:type="boolean">
   false
  </accept_unsigned_file>
 </signature-handling>
 ...
<general>
```

`accept_file_without_checksum`

`true` に設定した場合、AutoYaST はコンテンツファイルなどのチェックサムが存在しない場合であっても、受け入れるようになります。

```
<general>
 <signature-handling>
  <accept_file_without_checksum config:type="boolean">
   false
  </accept_file_without_checksum>
 </signature-handling>
 ...
<general>
```

`accept_verification_failed`

`true` に設定すると、AutoYaST は署名の検証が失敗した場合でも、署名付きのファイルであればそれらを受け付けるようになります。

```
<general>
 <signature-handling>
  <accept_verification_failed config:type="boolean">
   false
  </accept_verification_failed>
 </signature-handling>
 ...
<general>
```

`accept_unknown_gpg_key`

`true` に設定した場合、AutoYaST はインストール元に新しい GPG 鍵が存在し、その鍵でコンテンツファイルに署名されている場合、それを受け付けるようになります。

```
<general>
 <signature-handling>
  <accept_unknown_gpg_key config:type="boolean">
   false
  </accept_unknown_gpg_key>
 </signature-handling>
 ...
<general>
```

`accept_non_trusted_gpg_key`

`true` に設定した場合、既知ではあるもののまだ信頼していない鍵を受け付けるようになります。

```
<general>
 <signature-handling>
  <accept_non_trusted_gpg_key config:type="boolean">
   false
  </accept_non_trusted_gpg_key>
```

```
 </signature-handling>
 ...
<general>
```

`import_gpg_key`

`true` に設定した場合、AutoYaST はインストール元にある新しい GPG 鍵を受け付け、データベース内に取り込むようになります。

```
<general>
 <signature-handling>
  <import_gpg_key config:type="boolean">
   false
  </import_gpg_key>
 </signature-handling>
 ...
<general>
```

4.1.6 待機セクション

インストールの第 2 ステージでは、ネットワークの設定などのさまざまなモジュールを動作させて、システムの設定を行います。`wait` セクションでは、特定のモジュールの動作前後で実行すべきスクリプトを指定することができます。また、各モジュールの前後でシステムを待機させる時間幅 (「スリープ」) を設定することもできます。

`pre-modules`

設定モジュールが起動する前に実行すべきスクリプトと、その待機時間をそれぞれ設定することができます。下記のコードでは、ネットワーク設定モジュールを起動する前に、10 秒間の待機と echo コマンドの実行を指定しています。

```
<general>
 <wait>
  <pre-modules config:type="list">
   <module>
    <name>networking</name>
    <sleep>
     <time config:type="integer">10</time>
     <feedback config:type="boolean">true</feedback>
    </sleep>
    <script>
     <source>echo foo</source>
     <debug config:type="boolean">false</debug>
    </script>
   </module>
  </pre-modules>
  ...
```

```
 </wait>
<general>
```

post-modules

設定モジュールが起動した後に実行すべきスクリプトと、その待機時間をそれぞれ設定することができます。下記のコードでは、ネットワーク設定モジュールを起動した後に、10 秒間の待機と echo コマンドの実行を指定しています。

```
<general>
 <wait>
  <post-modules config:type="list">
   <module>
    <name>networking</name>
    <sleep>
     <time config:type="integer">10</time>
     <feedback config:type="boolean">true</feedback>
    </sleep>
    <script>
     <source>echo foo</source>
     <debug config:type="boolean">false</debug>
    </script>
   </module>
  </post-modules>
  ...
 </wait>
<general>
```

4.1.7 general セクションの例

本章で説明してきた項目の設定例を示します。

例 4.1: 一般オプション

下記の例では、general セクション内で最もよく使用されるオプションを示しています。なお、pre および post の各モジュールセクションには、ダミーのスクリプトを埋め込んでいます。

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns"
 xmlns:config="http://www.suse.com/1.0/configns">
 <general>
  <mode>
   <halt config:type="boolean">false</halt>
   <forceboot config:type="boolean">false</forceboot>
   <final_reboot config:type="boolean">false</final_reboot>
   <final_halt config:type="boolean">false</final_halt>
   <confirm_base_product_license config:type="boolean">
    false
```

```
   </confirm_base_product_license>
   <confirm config:type="boolean">true</confirm>
   <second_stage config:type="boolean">true</second_stage>
  </mode>
  <proposals config:type="list">
   <proposal>partitions_proposal</proposal>
  </proposals>
  <self_update config:type="boolean">true</self_update>
  <self_update_url>http://example.com/updates/$arch</self_update_url>
  <signature-handling>
   <accept_unsigned_file config:type="boolean">
    true
   </accept_unsigned_file>
   <accept_file_without_checksum config:type="boolean">
    true
   </accept_file_without_checksum>
   <accept_verification_failed config:type="boolean">
    true
   </accept_verification_failed>
   <accept_unknown_gpg_key config:type="boolean">
    true
   </accept_unknown_gpg_key>
   <import_gpg_key config:type="boolean">true</import_gpg_key>
   <accept_non_trusted_gpg_key config:type="boolean">
   true
   </accept_non_trusted_gpg_key>
  </signature-handling>
  <wait>
   <pre-modules config:type="list">
    <module>
     <name>networking</name>
     <sleep>
      <time config:type="integer">10</time>
      <feedback config:type="boolean">true</feedback>
     </sleep>
     <script>
     <source>&gt;![CDATA[
echo "Sleeping 10 seconds"
      ]]&gt;</source>
     <debug config:type="boolean">false</debug>
     </script>
    </module>
   </pre-modules>
   <post-modules config:type="list">
    <module>
     <name>networking</name>
     <sleep>
      <time config:type="integer">10</time>
      <feedback config:type="boolean">true</feedback>
     </sleep>
     <script>
```

```
      <source>&gt;![CDATA[
echo "Sleeping 10 seconds"
      ]]&gt;</source>
      <debug config:type="boolean">false</debug>
     </script>
    </module>
   </post-modules>
  </wait>
 </general>
</profile>
```

4.2 レポート

`report` リソースでは、インストール時に表示される可能性のある、3 種類のポップアップを管理します:

- メッセージポップアップ (通常は致命的なものではなく、情報提供のみを目的としたもの)
- 警告ポップアップ (何らかの問題が発生していることを示すもの)
- エラーポップアップ (何らかのエラーが発生していることを示すもの)

例 4.2: レポートの動作

```
<report>
  <errors>
    <show config:type="boolean">true</show>
    <timeout config:type="integer">0</timeout>
    <log config:type="boolean">true</log>
  </errors>
  <warnings>
    <show config:type="boolean">true</show>
    <timeout config:type="integer">10</timeout>
    <log config:type="boolean">true</log>
  </warnings>
  <messages>
    <show config:type="boolean">true</show>
    <timeout config:type="integer">10</timeout>
    <log config:type="boolean">true</log>
  </messages>
  <yesno_messages>
    <show config:type="boolean">true</show>
    <timeout config:type="integer">10</timeout>
    <log config:type="boolean">true</log>
  </yesno_messages>
</report>
```

ご利用の方の知識レベルに応じて、これらのメッセージを非表示にしたり記録したり、もしくは (時間制限付きで) 表示したりすることができます。なお、全ての `messages` に対してタイムアウトを設定し、表示しておくことをお勧めします。警告は、その内容によっては読み飛ばすことができるものもありますが、通常は無視すべきものではありません。

自動インストールモードでの既定値では、全てのエラーを時間制限無しで表示し、それ以外の警告やその他のメッセージは 10 秒の時間制限を設定して全て表示します。



警告: 重要なシステムメッセージについて

`report` リソースでは、インストール時に表示される全てのメッセージを制御することはできません。パッケージのインストールやパーティション設定などで致命的な問題が発生した場合、`report` セクションの設定に関わらずメッセージが表示されることがあります。通常、これらのメッセージは［はい］または［いいえ］で回答する必要があるものです。

4.3 GRUB 2 ブートローダ

本章は `yast2-bootloader` で GRUB 2 を利用する場合の説明となります。なお、古いバージョンの GRUB を同梱している古い製品バージョンの場合は、そのバージョン内の `/usr/share/doc/packages/autoyast2/` ディレクトリ内にある文書をお読みください。

既定では、AutoYaST は起動メディアからインストールした場合と同じブートローダの選択になります。たとえば EFI で起動している場合は GRUB 2 for EFI がインストールされます。そのため、特別な要件がない限り、本章は省略してかまいません。また、EFI システムの起動には特別なパーティション設定が必要となるため、4.5.1項「自動パーティション設定」で説明している自動パーティション設定の使用もお勧めします。

なお、既定値を修正する場合は、`<bootloader>` セクションを設定してください。たとえば下記のようになります:

```
<bootloader>
  <loader_type>
    <!-- ブートローダの種類 (grub2 または grub2-efi) -->
  </loader_type>
  <global>
    <!--
      GRUB 2 のインストール設定と汎用ブートコードに関する設定
    -->
  </global>
  <device_map config:type="list">
    <!-- デバイスの順序に関する設定 -->
  </device_map>
```

```
</bootloader>
```

なお、全ての項目を指定する必要はありません。変更したい箇所のタグだけを設定してください。AutoYaST では、プロファイル内で設定のない箇所については既定値を適用します。

4.3.1 ブートローダの種類

ここでは使用したいブートローダの種類 (UEFI もしくは BIOS) を指定します。なお、アーキテクチャによっては、BIOS と UEFI の両方には対応しておらず、いずれか片方しか対応していないものもあります。インストーラ側で自動的に判断させたい場合は、最も安全な選択肢 (`default`) を指定してください。

```
<loader_type>種類</loader_type>
```

`種類` には下記のいずれかを指定します:

- `default` : インストーラ側で適切なブートローダを自動的に判断します。何も指定しない場合の既定値です。
- `grub2` : 古い BIOS 型のブートローダを使用します。
- `grub2-efi` : EFI ブートローダを使用します。
- `none` : インストーラ側では起動処理に関する設定を行わないようにします。

4.3.2 グローバル設定

この項目は任意指定ではありますが、重要な項目です。ここでは GRUB 2 のインストール先と起動処理の動作について設定を行います。こちらも同様に、何も設定しない場合は `yast2-bootloader` が自動的に設定を作成します。通常、AutoYaST の制御ファイルには本章のみを含めるものとし、その他の箇所は `yast2-bootloader` 側でインストール時に自動設定させるようにします。また、特別な要件がない限り、XML ファイル内にはブートローダの設定そのものを省略しておくことをお勧めします。



ヒント: ハイバネーションについて

この項目は任意指定ではありますが、重要な項目です。ここでは GRUB 2 のインストール先と起動処理の動作について設定を行います。こちらも同様に、何も設定しない場合は `yast2-bootloader` が自動的に設定を作成します。通常、AutoYaST の制御ファイルには本章のみを含めるものとし、その他の箇所は `yast2-bootloader` 側でインストール時に自動設定させるようにします。また、特別な要件がない限り、XML ファイル内にはブートローダの設定そのものを省略しておくことをお勧めします。

ヒント: ハイバネーションについて

ハイバネーションに関して特別な要件がある場合は、`append` 設定内に `resume` もしくは `noresume` を設定してください。

インストーラ側の判断に関わらずハイバネーションを無効化したい場合は、`append` セクション内のカーネルパラメータに `noresume` を設定してください。

その逆に、使用すべきハイバネーションデバイスを指定したい場合は、`resume` の値としてデバイスパスを設定してください。なお、デバイス名が変化してしまうことでデバイスの選択を誤ってしまう問題に対応するため、スワップデバイスをラベルで指定しておくことをお勧めします:

```
        <append>quiet resume=/dev/disk/by-label/my_swap</append>
```

`resume` や `noresume` のどちらも指定しない場合や、`resume` で指定したデバイスが存在しない場合は、インストーラ側で `resume` に指定すべき値を自動的に判断します。なお、環境によっては、インストーラ側の判断でハイバネーション関連のパラメータそのものを削除する場合もあります。

```
<global>
  <activate>true</activate>
  <timeout config:type="integer">10</timeout>
  <terminal>gfxterm</terminal>
  <gfxmode>1280x1024x24</gfxmode>
</global>
```

ブートローダのグローバル設定

activate

起動用のパーティションに対して起動 (boot) フラグを設定します。起動パーティションとは `/boot` ディレクトリにマウントされるパーティションで、それが無い場合は `/` ディレクトリにマウントされるパーティションになります。なお、起動パーティションが論理パーティションであった場合は、拡張パーティションに対して起動フラグが設定されます。

```
<activate>true</activate>
```

append

通常モードと復元モードで使用されるカーネルパラメータで、起動時にパラメータの末尾に追加するものを指定します。

```
<append>nomodeset vga=0x317</append>
```

boot_boot

GRUB 2 を `/boot` ディレクトリのパーティション内に書き込みます。`/boot` ディレクトリのパーティションがない場合、GRUB 2 は `/` ディレクトリのパーティションに書き込みます。

```
<boot_boot>false</boot_boot>
```

boot_custom

指定した独自のデバイスに GRUB 2 を書き込みます。

```
<boot_custom>/dev/sda3</boot_custom>
```

boot_extended

GRUB 2 を拡張パーティションに書き込みます。これは `/boot` パーティションが論理パーティション内に存在する場合に重要です。注意: なお、起動パーティションが論理パーティションである場合は、`generic_mbr` ではなく `boot_mbr` (つまり、GRUB 2 を MBR に書き込む) を使用すべきです。

```
<boot_extended>false</boot_extended>
```

boot_mbr

GRUB 2 を最初のディスクの MBR に書き込みます (ディスクの順序については `device.map` で設定します) 。

```
<boot_mbr>false</boot_mbr>
```

boot_root

GRUB 2 を `/` パーティションに書き込みます。

```
<boot_root>false</boot_root>
```

cpu_mitigations

CPU の脆弱性に対する緩和策となるカーネルの起動パラメータを選択します。これにより、セキュリティと性能のどちらを重視するのかを設定することにもなります。
設定できる値は下記のいずれかです:

auto

お使いの CPU モデルに応じた緩和策を全て有効化しますが、CPU を跨いだスレッド 攻撃については保護を行いません。この設定では、ご利用の負荷状況に応じて、性能面に幾分かの影響が発生します。

nosmt

利用可能な全ての緩和策を全て適用します。お使いの CPU モデルに応じた全ての 緩和策を適用し、かつ同時マルチスレッディング Simultaneous Multithreading (SMT) も無効化し、CPU スレッドを跨いだサイドチャネル攻撃を無効化します。ご利用の負荷状況にもよりますが、さらに性能面への影響が高くなります。

off

全ての緩和策を無効化します。お使いの CPU モデルに応じて、CPU に対するサイド チャネル攻撃が成立することになります。この設定では、性能面への影響はありません。

manual

緩和策のレベルを指定しません。カーネルのコマンドラインオプションを利用して、手作業で緩和策を個別に指定することができるようになります。

```
<cpu_mitigations>auto</cpu_mitigations>
```

AutoYaST で特に何も設定しない場合は、カーネルのコマンドラインで対応する設定を変更することができます。既定では、インストールメディア内の `/control.xml` に書かれた製品固有の設定を使用します。

generic_mbr

MBR に対して汎用のブートコードを書き込みます (なお、`boot_mbr` が `true` の場合は無視されます) 。

```
<generic_mbr config:type="boolean">false</generic_mbr>
```

gfxmode

GRUB 2 のグラフィカル画面の解像度を指定します (なお、<terminal> で `gfxterm` を指定する必要があります) 。
設定可能な値は `auto` , `水平解像度x垂直解像度` , `水平解像度x垂直解像度x色深` のいずれかになります。特定のシステムにおいて GRUB 2 で利用可能な解像度の一覧を取得したい場合は、システムの起動時に GRUB 2 のコマンドラインで `vbeinfo` コマンドを実行してください。

```
<gfxmode>1280x1024x24</gfxmode>
```

os_prober

`true` に設定すると、インストール済みの他のオペレーティングシステムを自動検出して、対応する起動項目を生成するようになります。

```
<os_prober>false</os_prober>
```

password

この項目を指定すると、ブートローダをパスワードで保護するようになります。ここで指定したパスワードを入力しない限り、システムは起動できなくなります。

なお、副要素として `value`，`encrypted`，`unrestricted` を指定することができます。

`value` 内にはパスワードそのものを指定します。パスワードは暗号化せずそのまま記述することもできます (この場合、YaST が自動的に暗号化して設定します) し、`grub-mkpasswd-pbkdf2` であらかじめ暗号化したものを設定することもできます。あらかじめ暗号化したものを設定する場合は、`encrypted` に `true` を指定してください。

`unrestricted` に `false` を指定すると、`value` で指定したパスワードを入力しない限り、起動も項目の編集 (項目を選んで `E` を押す) もできなくなりますが、`true` を指定すると、起動はパスワード入力無しで行うことができるものの、GRUB 2 の項目編集についてはパスワード入力が必要となるようになります。何も指定しない場合、`true` が設定されたものとみなされます。

```
<password><value>my_strong_password</value><encrypted>false</
encrypted><unrestricted>false</unrestricted></password>
```

suse_btrfs

古い形式であり、現在は使用されていません。btrfs のスナップショット機能は、自動的に有効化されます。

serial

GRUB 2 の端末モードを `serial` に設定した場合に、実行すべきコマンドを指定します。

```
<serial>serial --speed=115200 --unit=0 --word=8 --parity=no --stop=1</serials>
```

secure_boot

`false` に設定すると、UEFI Secure Boot が無効化されます。`grub2-efi` ブートローダを使用した場合にのみ意味があります。

```
<secure_boot>false</secure_boot>
```

terminal

GRUB 2 で使用すべき端末モードを指定します。設定可能な値は `console`，`gfxterm`，`serial` のいずれかとなります。`serial` を指定した場合は、<serial> でコマンドを指定する必要があります。

```
<terminal>serial</terminal>
```

timeout

既定の項目を自動起動するまでの待機時間を秒単位で指定します。

```
<timeout config:type="integer">10</timeout>
```

trusted_grub
: `true` を指定すると、Trusted GRUB を使用するようになります。Trusted GRUB は Trusted Platform Module (TPM) に対応しています。`grub2` ブートローダを使用した場合にのみ動作します。

```
<trusted_grub">true</trusted_grub>
```

update_nvram
: `true` を指定すると、ファームウエアの NVRAM 内に起動項目を追加するようになります。特殊な設定を行いたい場合や、ファームウエア側での問題を回避する必要がある場合を除いて、`true` を指定しておくことをお勧めします。

```
<update_nvram>true</update_nvram>
```

vgamode
: 起動時のカーネルパラメータに追加する `vga=値` の値部分を指定します。

```
<vgamode>0x317</vgamode>
```

xen_append
: Xen ゲスト向けの起動項目の末尾に追加すべきカーネルパラメータを指定します。

```
<xen_append>nomodeset vga=0x317</xen_append>
```

xen_kernel_append
: VM ホストサーバ 内での Xen カーネルに対して、末尾に追加すべきカーネルパラメータを指定します。

```
<xen_kernel_append>dom0_mem=768M</xen_kernel_append>
```

4.3.3 デバイスマップ

GRUB 2 では、BIOS ドライブと Linux デバイスの間での差異をなくすため、デバイスの ID 文字列 (UUID) やファイルシステムラベルでデバイスマップを指定することができます。通常は GRUB 2 ユーティリティがインストール時に自動作成しますので、ディスクが 1 台だけのシステムでは問題になりませんが、複数台のディスクが存在する場合など、自動的なデバイスマップでは問題がある場合、ここで独自のマッピングを設定することができます。

```
<device_map config:type="list">
  <device_map_entry>
    <firmware>hd0</firmware> <!-- マップ内でのデバイス順序  -->
    <linux>/dev/disk/by-id/ata-ST3500418AS_6VM23FX0</linux> <!-- デバイス（ディスク）名  -->
  </device_map_entry>
</device_map>
```

4.4 systemd ブートローダ

本章は `yast2-bootloader` で systemd-boot を利用する場合の説明となります。

AutoYaST ブートローダ設定は下記のような構造になっています:

```
<bootloader>
  <loader_type>
    systemd-boot
  </loader_type>
  <global>
    <!--
      systemd-boot のインストール設定と汎用ブートコードに関する設定
    -->
  </global>
 </bootloader>
```

4.4.1 ブートローダの種類

ここでは使用したいブートローダの種類 (systemd-boot) を指定します。

```
<loader_type>systemd-boot</loader_type>
```

4.4.2 グローバル設定

この項目は任意指定ではありますが、重要な項目です。ここでは systemd-boot のインストール先と起動処理の動作方法について設定を行います。なお、何も設定しない場合は `yast2-bootloader` が自動的に設定を作成します。特別な要件がない限り、XML ファイル内にはブートローダの設定を省略しておくことをお勧めします。

```
<global>
  <timeout config:type="integer">10</timeout>
  <secure_boot>false</secure_boot>
</global>
```

secure_boot

`false` に設定すると、UEFI Secure Boot が無効化されます。

```
<secure_boot>false</secure_boot>
```

timeout

既定の項目を自動起動するまでの待機時間を秒単位で指定します。

```
<timeout config:type="integer">10</timeout>
```

4.5 パーティション設定

パーティション設定に対して、AutoYaST では 3 つの種類に分類して扱っています:

- 自動パーティション設定。ユーザ側にはパーティション設定に対する問い合わせを行わず、AutoYaST 側で設定を全て賄うことを指します。
- ガイド付きのパーティション設定。ユーザ側で LVM を使用するかどうかなどの基本設定を行うものの、詳しいパーティション設定やボリュームグループの設定など、詳細箇所については行わないものを指します。
- 熟練者向けのパーティション設定。ユーザ側で詳細な配置の設定を行うことを指します。ただし、全ての設定までを指定する必要はなく、必要に応じて環境に合わせた既定値を提案するものとします。

これらの分類は、通常のインストーラでも選択することができます。[ガイド付き設定] を使用すれば詳細なパーティション設定画面を飛ばして YaST 側に設定を任せることができますし、[熟練者向けパーティション設定] を使用すれば、詳細な配置を設定することができます。

4.5.1 自動パーティション設定

AutoYaST では、ユーザに対して一切問い合わせを行わずに賢くパーティション設定を行うことができます。インストールする製品にも依存しますが、AutoYaST では通常、ルートファイルシステムに対して btrfs を、個別の `/home` パーティションに対して XFS をそれぞれ作成し、スワップパーティションを加えた 3 つのパーティションを作成します。これに加えて、アーキテクチャによっては起動に必要なパーティション (BIOS GRUB パーティションなど) を追加で作成することもあります。

しかしながら、これらの既定値はディスク領域などの状況によっても変化します。たとえばディスク容量が少ない場合、個別の `/home` パーティションが作成されなかったりします。

これらの既定値を調整したい場合は、4.5.2項「ガイド付きのパーティション設定」で説明している設定項目を調整してください。

4.5.2 ガイド付きのパーティション設定

AutoYaST には、一切ユーザに対して問い合わせを行うことなくパーティション設定を行う機能がありますが、いくつかの一般的なパラメータのみを指定しておいて、残りを AutoYaST に任せたほうが都合の良い場合があります。たとえば詳細なパラメータを指定せずに LVM を使用したい場合や、ファイルシステムの暗号化を行いたいような場合がそれにあたります。これは通常のインストール処理でも同様で、ガイド付きの設定を利用することで、基本的なパラメータのみを指定したパーティション設定を行うことができます。

例4.3「LVM ベースのガイド付きパーティション設定」で示している `storage` セクションでは、LVM を利用するとともに、必要かどうかにかかわらず全ての Windows パーティションを削除するような設定を示しています。

例 4.3: LVM ベースのガイド付きパーティション設定

```
<general>
   <storage>
     <proposal>
       <lvm config:type="boolean">true</lvm>
       <windows_delete_mode config:type="symbol">all</windows_delete_mode>
     </proposal>
   </storage>
</general>
```

lvm

LVM ベースの提案を作成するかどうかを指定します。既定値は `false` です。

```
<lvm config:type="boolean">true</lvm>
```

lvm_vg_reuse

提案にあたって、インストーラが既存の LVM ボリュームを再利用するかどうかを指定します。既定値は `true` です。

```
<lvm_vg_reuse config:type="boolean">false</lvm_vg_reuse>
```

resize_windows

`true` に設定した場合、AutoYaST はインストールに必要な領域を確保するため、Windows のパーティションのサイズ変更を行うようになります。

```
<resize_windows config:type="boolean">false</resize_windows>
```

windows_delete_mode

- `none` を指定すると、Windows パーティションを削除しないようになります。
- `ondemand` を指定すると、必要に応じて Windows パーティションを削除するようになります。
- `all` を指定すると、全ての Windows パーティションを削除するようになります。

```
<windows_delete_mode config:type="symbol">ondemand</windows_delete_mode>
```

linux_delete_mode

- `none` を指定すると、Linux パーティションを削除しないようになります。
- `ondemand` を指定すると、必要に応じて Linux パーティションを削除するようになります。
- `all` を指定すると、全ての Linux パーティションを削除するようになります。

```
<linux_delete_mode config:type="symbol">ondemand</linux_delete_mode>
```

other_delete_mode

- `none` を指定すると、その他のパーティションを削除しないようになります。
- `ondemand` を指定すると、必要に応じてその他のパーティションを削除するようになります。
- `all` を指定すると、全てのその他のパーティションを削除するようになります。

```
<other_delete_mode config:type="symbol">ondemand</other_delete_mode>
```

encryption_password

指定したパスワードで暗号化を行うようにします。既定では暗号化を行いません。

```
<encryption_password>some-secret</encryption_password>
```

4.5.3 熟練者向けパーティション設定

ガイド付きパーティション設定よりも詳しく設定を行いたい場合は、AutoYaST のプロファイル内に `partitioning` セクションを挿入して指定を行います。ただし、AutoYaST では全ての詳細までを指定する必要はありません。不完全な指定を行っていても、適切な既定値を割り当てて設定するようになっています。

`partitioning` セクション内には `drive` 要素の羅列を設定します。それぞれの `drive` 要素には、ディスクや LVM のボリュームグループ、RAID やマルチデバイス型の btrfs ファイルシステムなどの配置要素を設定します。

例4.4「/ , /home , swap の各パーティションを作成する場合の例」の例では、AutoYaST に対して、ディスク全体を使用して `/` , `/home` , `swap` の各パーティションを作成する設定例を示しています。なお、パーティション内で使用すべきファイルシステムの情報などの指定が省略されていることに注意してください。このような指定を行った場合は、AutoYaST 側で自動的に判断して設定を行います。

例 4.4: / , /home , swap の各パーティションを作成する場合の例

```
<partitioning config:type="list">
  <drive>
    <use>all</use>
    <partitions config:type="list">
      <partition>
        <mount>/</mount>
        <size>20GiB</size>
      </partition>
      <partition>
        <mount>/home</mount>
        <size>max</size>
      </partition>
      <partition>
        <mount>swap</mount>
        <size>1GiB</size>
      </partition>
    </partitions>
  </drive>
  <partitioning>
```



ヒント: 起動パーティションの提案

AutoYaST では、プロファイル内に指定されたパーティション設定で問題なく起動できるかどうかを自動的に判断します。もしも起動できないものと判断された場合は、必要なパーティションを自動的に追加します。そのため、起動用のパーティションが必要かどうかがわからないような状況でも、AutoYaST に設定を任せておけば正しい選択を行ってくれることになります。

4.5.3.1 ドライブの設定

下記のような XML 構造内に、必要な要素を追加していきます:

```
<profile>
  <partitioning config:type="list">
```

```
    <drive>
     ...
    </drive>
  </partitioning>
</profile>
```

属性／値／説明

device

任意指定です。ここでは設定対象のデバイスを指定しますが、何も指定しない場合は AutoYaST が自動的にデバイスを判断します。判断を調整する方法については、ヒント: ドライブを読み飛ばす設定について をお読みください。

値に `ask` を指定すると、AutoYaST はインストール時に使用するデバイスをユーザに対して尋ねるようになります。

また、デバイス名を ID で指定することで、システム構成の変更に影響を受けないようにすることもできます。たとえば `/dev/disk/by-id/ata-WDC_WD3200AAKS-75L9A0_WD-WMAV27368122` のように by-id で指定したり、`/dev/disk/by-path/pci-0001:00:03.0-scsi-0:0:0:0` のように by-path で指定したりすることができます。

```
<device>/dev/sda</device>
```

ボリュームグループやソフトウエア RAID 、`bcache` デバイスの場合は、インストール後のシステムで使用するデバイス名は別のものになります (既存のデバイスとの衝突を避けるためのものです) 。

マルチパスデバイスの扱いについて、詳しくは 4.5.7項「マルチパスサポート」をお読みください。

initialize

任意指定です。既定値は `false` で、`true` に設定すると、AutoYaST がパーティションの計算を行う前に、パーティションテーブルを消去するようになります。

```
<initialize config:type="boolean">true</initialize>
```

partitions

任意指定です。ここには <partition> 項目を入れます (詳しくは 4.5.3.2項「パーティションの設定」をお読みください) 。

```
<partitions config:type="list">
          <partition>...</partition>
          ...
          </partitions>
```

何もパーティションを指定しない場合、AutoYaST は適切なパーティション設定を自動で作成します (詳しくは 4.5.3.5項「不完全な設定の補完」をお読みください) 。

pesize

任意指定で、LVM でのみ意味のある項目です。LVM ボリュームグループの場合、4M が既定値となります。

```
<pesize>8M</pesize>
```

use

設定しておくことをお勧めします。AutoYaST がハードディスクに対してパーティション設定を行うにあたって、その大まかな方針を指定します。下記のいずれかを指定します:

`all`: 新しいパーティションを作成する際、ディスク全体を使用するようにします。

`linux`: 既存の Linux パーティションのみを使用します。

`free`: デバイス内の未使用領域のみを使用し、それ以外のパーティションについては何も変更しないようにします。

`1,2,3`: 使用すべきパーティションの番号をカンマ区切りで指定します。

type

任意指定です。`drive` の種類を指定します。通常の物理ハードディスクの場合、既定値は `CT_DISK` になります。下記のいずれかを指定してください:

`CT_DISK`: 物理的なハードディスク (既定値)

`CT_LVM`: LVM ボリュームグループ

`CT_MD`: ソフトウエア RAID デバイス

`CT_DMMULTIPATH`: マルチパスデバイス (廃止予定, CT_DISK と同じ意味になります)

`CT_BCACHE`: ソフトウエア `bcache` デバイス

`CT_BTRFS`: マルチデバイス型の btrfs ファイルシステム

`CT_NFS`: NFS

`CT_TMPFS`: `tmpfs` ファイルシステム

```
<type config:type="symbol">CT_LVM</type>
```

disklabel

任意指定です。既定では YaST 側で自動的に判断します。異なる種類のパーティションテーブルが既に存在している場合、そのままにしておくべきパーティションや再利用可能なパーティションが含まれている場合を除き、指定した種類のパーティションテーブルを再作成します。何もパーティションを作成せずにディスクを使用したい場合は、この要素を `none` にしてください。

`msdos`

`gpt`

`none`

```
<disklabel>gpt</disklabel>
```

keep_unknown_lv

任意指定です。既定値は `false` です。

type=CT_LVM のドライブでのみ意味のある項目です。論理ボリュームグループを再利用している場合、この値を `true` に設定すると、そのグループ内に存在する全ての既存論理ボリュームは、それらが <partitioning> で指定されている場合を除き、何も変更が加えられないようになります。そのため、そのままにしておくべき論理ボリュームについては、何も指定しなくて良いことになります。

```
<keep_unknown_lv config:type="boolean">false</keep_unknown_lv>
```

enable_snapshots

任意指定です。既定値は `true` です。

`/` にマウントされている btrfs ファイルシステムに対して、スナップショットを有効化します (その他のファイルシステムや、`/` にマウントされていない btrfs ファイルシステムには適用されません) 。

```
<enable_snapshots config:type="boolean">false</enable_snapshots>
```

quotas

任意指定です。既定値は `false` です。

btrfs サブボリュームに対するクォータサポートを有効化します。この要素を `true` に設定すると、ファイルシステムに対するクォータサポートを有効化します。なお、制限の設定は各サブボリューム側で行ってください。詳しくは 4.5.3.3項「btrfs サブボリューム」をお読みください。

```
<quotas config:type="boolean">true</quotas>
```



重要: データ損失への警告

`use` プロパティに設定した値に従って、既存のデータとパーティションの取り扱いが決まります。このプロパティに `all` を指定すると、ディスク全体が消去されます。いずれかのパーティション内に重要なデータが含まれる場合は、あらかじめバックアップを採取するものとし、`confirm` を指定して確認を行うようにしてください。確認を行うように設定しないと、何もメッセージが表示されないままパーティションが削除されてしまいます。



ヒント: ドライブを読み飛ばす設定について

AutoYaST の制御ファイル内で <device> 項目を指定せず、AutoYaST の自動デバイス判断に任せたい場合、その判断基準を設定することができます。通常は適切なデバイスのうち最初のデバイスを使用するようになっていますが、下記のように設定することで、条件に合致するデバイスを除外できるようになります:

```
<partitioning config:type="list">
  <drive>
    <initialize config:type="boolean">true</initialize>
    <skip_list config:type="list">
      <listentry>
        <!-- usb-storage ドライバを使用しているデバイスを読み飛ばす -->
        <skip_key>driver</skip_key>
        <skip_value>usb-storage</skip_value>
      </listentry>
      <listentry>
        <!-- 1GB より小さいデバイスを読み飛ばす -->
        <skip_key>size_k</skip_key>
        <skip_value>1048576</skip_value>
        <skip_if_less_than config:type="boolean">true</skip_if_less_than>
      </listentry>
      <listentry>
        <!-- 100GB より大きいデバイスを読み飛ばす -->
        <skip_key>size_k</skip_key>
        <skip_value>104857600</skip_value>
        <skip_if_more_than config:type="boolean">true</skip_if_more_than>
      </listentry>
    </skip_list>
  </drive>
</partitioning>
```

<skip_key> で設定できる値の一覧について、詳しくはインストール済みのシステムで `yast2 ayast_probe` を実行してください。

4.5.3.2 パーティションの設定

下記のような XML 構造内に、必要な要素を追加していきます:

```
<drive>
  <partitions config:type="list">
    <partition>
      ...
    </partition>
  </partitions>
</drive>
```

create

対象のパーティションや論理ボリュームを作成しなければならない場合、もしくは既に存在しているような場合に指定します。`false` に設定した場合、AutoYaST に対して使用すべきデバイスを指示するため、`partition_nr` , `lv_name`, `label`, `uuid` のいずれかを指定する必要があります。

```
<create config:type="boolean">false</create>
```

crypt_method

任意指定です。パーティションの暗号化方式を指定します。下記のいずれかを指定してください:

- `luks1` : 通常の LUKS1 形式での暗号化を行います。
- `luks2` : 通常の LUKS2 形式での暗号化を行います。
- `pervasive_luks2` : 普及型のボリュームの暗号化を行います。
- `protected_swap` : 暫定的な保護鍵を利用して暗号化します。
- `secure_swap` : 暫定的な機密鍵を利用して暗号化します。
- `random_swap` : 暫定的な乱数鍵を利用して暗号化します。

```
<crypt_method config:type="symbol">luks1</crypt_method>
```

暗号化方式の選択機能は openSUSE Leap 15.2 もしくはそれ以降のバージョンで使用することができます。以前のバージョンと同じ動作にしたい場合は、`luks1` を指定してください。
暗号化パスワードの指定方法については、`crypt_key` 要素の説明をお読みください。
通常の LUKS 暗号化を使用する場合、LUKS 内で指定する暗号化の種類に応じて `crypt_pbkdf` , `crypt_cipher` , `crypt_key_size` 等の設定を行って細かい調整を行うことができます。なお、使用する暗号化方式とその設定内容によっては、インストール処理を完了するのに必要なメモリ量を劇的に変化させる場合があることに注意してください。特に LUKS2 で細かいパラメータを指定せずに使用する場合、デバイスの暗号化を実施するのに数ギガバイト程度のメモリが必要となります。

crypt_fs

パーティションを暗号化するよう指定します (既定値は `false` です)。ただし、この要素は廃止予定になっています。代わりに `crypt_method` を使用してください。

```
<crypt_fs config:type="boolean">true</crypt_fs>
```

crypt_key

`crypt_method` でパスワードを設定する必要のある方式 (`luks1` , `luks2` , `pervasive_luks2` のいずれか) を選択した場合に必要となります。

```
<crypt_key>xxxxxxxx</crypt_key>
```

crypt_cipher

LUKS で使用する暗号化方式を指定します。この値は `crypt_method` の値が `luks1` または `luks2` である場合にのみ意味を持ちます。設定可能な値とその書式は、`cryptsetup` コマンドの `--cipher` パラメータの値と同じです。利用可能な暗号化方式を確認したい場合は、`/proc/crypto` ファイルをご覧ください。

```
<crypt_cipher>aes-xts-plain64</crypt_cipher>
```

crypt_key_size

LUKS で暗号化を行う際の鍵サイズをビット単位で指定します。この値は `crypt_method` の値が `luks1` または `luks2` である場合にのみ意味を持ちます。また、値は 8 の倍数でなければならないほか、使用する暗号方式によって異なる上限値があります。

```
<crypt_key_size config:type="integer">256</crypt_key_size>
```

crypt_pbkdf

LUKS2 暗号化で使用されるパスワードベースの鍵導出関数を指定します。この値は `crypt_method` の値が `luks1` または `luks2` である場合にのみ意味を持ちます。設定可能な値は `pbkdf2` , `argon2i` , `argon2id` のいずれかで、何も指定しない場合は `cryptsetup` コマンドの既定の関数が使用されます。なお、argon2 の 2 種類の形式はいずれも、暗号化処理時に大きなサイズのメモリを占有するように意図して設計されているため、これらの関数を直接指定した場合や、何も指定しなかった場合 (argon2 が使用される可能性があります) は、暗号化処理を行わない場合に比べてインストール処理時のメモリ要件が厳しくなることに注意してください。

```
<crypt_pbkdf config:type="symbol">argon2id</crypt_pbkdf>
```

crypt_label

暗号化デバイスに設定する LUKS ラベルを指定します。この値は `crypt_method` の値が `luks2` である場合にのみ意味を持ちます。

```
<crypt_label>crypt_home</crypt_label>
```

mount

少なくとも 1 つのルートパーティション (/) とスワップパーティションを設定する必要があります。

```
<mount>/</mount><mount>swap</mount>
```

fstopt

対象のパーティションに対して設定するマウントオプションを指定します。詳しくは `man mount` をお読みください。

```
<fstopt>ro,noatime,user,data=ordered,acl,user_xattr</fstopt>
```

label

パーティションのラベルを指定します。デバイスをフォーマットする場合 (特に `mountby` パラメータを `label` にする場合) に有用であるほか、既に存在するデバイスを識別するような場合にも使用します (上述の `create` もお読みください) 。指定例について、詳しくは `man e2label` をお読みください。

```
<label>mydata</label>
```

uuid

パーティションの UUID を指定します。既に存在するデバイスを識別する場合にのみ使用します (上述の `create` もお読みください) 。なお、新しく作成するデバイスの場合は UUID を事前に決めることができませんので、指定することもできません (詳しくは `man uuidgen` をお読みください)。

```
<uuid>1b4e28ba-2fa1-11d2-883f-b9a761bde3fb</uuid>
```

size

4G, 4500M などのように、パーティションのサイズを指定します。/boot パーティションとスワップパーティションに対しては、 `auto` を指定することもできます。この場合、AutoYaST 側で適切なサイズを設定します。また、いずれか 1 つのパーティションに対して `max` を指定すると、残りの領域全てを使用することができるようになります。
サイズは割合で設定することもできます。たとえば 10% と指定すると、ハードディスク全体もしくはボリュームグループの 10% 分を使用することができます。必要に応じて `auto` , `max` , `size` , 割合を組み合わせて使用することもできます。

```
<size>10G</size>
```

openSUSE Leap 15 以降では、全ての値 (`auto` と `max` を含む) をパーティションのサイズ変更時にも使用できるようになっています。

format

AutoYaST 側でパーティションのフォーマットを行うかどうかを指定します。 `create` を `true` に設定した場合は、この値についても `true` を設定しておいたほうが良いでしょう。

```
<format config:type="boolean">false</format>
```

filesystem

任意指定です。ルートパーティション (`/`) の場合、既定値は `btrfs` に、データパーティションの場合、既定値は `xfs` になります。指定可能な値は下記のとおりです:

- `btrfs`
- `ext2`
- `ext3`
- `ext4`
- `fat`
- `xfs`
- `swap`

```
<filesystem config:type="symbol">ext3</filesystem>
```

mkfs_options

任意指定です。`mkfs` に渡すオプション文字列を指定します。なお、この文字列はどうしても必要な場合にのみ設定してください (詳しくは、作成するファイルシステムの mkfs マニュアルページをお読みください) 。

```
<mkfs_options>-I 128</mkfs_options>
```

partition_nr

このパーティションのパーティション番号を指定します。`create=false` もしくは LVM を使用しようとしている場合、`partition_nr` でパーティションの番号を指定することができます。

```
<partition_nr config:type="integer">2</partition_nr>
```

partition_id

`partition_id` ではパーティションの ID を設定します。Linux パーティションでは 131 、スワップパーティションでは 130 が自動的に設定されますが、それ以外の ID を設定したい場合は、これで設定を行ってください。既定値は Linux パーティションの場合は `131` , スワップパーティションの場合は `130` です。

```
<partition_id config:type="integer">131</partition_id>
```

FAT16 (MS-DOS): `6`
NTFS (MS-DOS): `7`
FAT32 (MS-DOS): `12`

拡張 FAT16 (MS-DOS): `15`
DIAG, Diagnostics and firmware (MS-DOS, GPT): `18`
PPC PReP ブートパーティション (MS-DOS, GPT): `65`
スワップ (MS-DOS, GPT, DASD, implicit): `130`
Linux (MS-DOS, GPT, DASD): `131`
Intel Rapid Start Technology (MS-DOS, GPT): `132`
LVM (MS-DOS, GPT, DASD): `142`
EFI システムパーティション (MS-DOS, GPT): `239`
MD RAID (MS-DOS, GPT, DASD): `253`
BIOS ブート (GPT): `263`
Windows 基本データ (GPT): `258`
EFI (GPT): `259`
Microsoft 予約 (GPT): `261`

partition_type

任意指定です。指定可能な値は `primary` です。パーティションテーブルに `msdos` 形式を使用している場合、この要素でパーティションの種類を `primary` (プライマリ) に設定することができます。`gpt` パーティションテーブルを使用している場合、このような区別はありませんので、値を設定しても無視されます。

```
<partition_type>primary</partition_type>
```

mountby

パーティション番号の代わりに、`device` (デバイス), `label` (ラベル), `uuid` (UUID), `path` (udev パス), `id` (udev ID) などでパーティションをマウントするよう設定することができます。上述の `label` と `uuid` の説明をお読みください。既定値は YaST 側で規定されていますが、通常は `id` です。

```
<mountby config:type="symbol">label</mountby>
```

subvolumes

btrfs のファイルシステムで作成するサブボリュームの一覧を指定します。これは btrfs を使用する場合にのみ意味のある項目です。詳しくは 4.5.3.3項「btrfs サブボリューム」をお読みください。

パーティションの設定内に `subvolumes` セクションが存在しない場合、AutoYaST は指定したマウントポイント内に、既定のサブボリュームセットを作成します。

```
<subvolumes config:type="list">
  <path>tmp</path>
  <path>opt</path>
```

```
  <path>srv</path>
  <path>var</path>
  ...
</subvolumes>
```

create_subvolumes
: btrfs のサブボリュームを作成すべきかどうかを指定します。既定値は `true` ですが、`false` に設定した場合、サブボリュームは作成されなくなります。

subvolumes_prefix
: btrfs のサブボリュームに設定するプレフィクス (前置) 名を指定します。何もプレフィクスを指定したくない場合は、空の値を設定してください:

```
<subvolumes_prefix><![CDATA[]]></subvolumes_prefix>
```

既定では `@` が設定されています。

lv_name
: このパーティションがボリュームグループ内の論理ボリュームである場合、ここで論理ボリュームの名前を指定します (`drive` 設定内の `type` パラメータもお読みください) 。

```
<lv_name>opt_lv</lv_name>
```

stripes
: LVM のストライピングを設定するための整数です。どれだけの数のデバイスにデータを分散させるのかを指定します。

```
<stripes config:type="integer">2</stripes>
```

stripesize
: 各ブロックのサイズを KB 単位で指定します。

```
<stripesize config:type="integer">4</stripesize>
```

lvm_group
: このパーティションが LVM のボリュームグループの一部として使用されている物理パーティションである場合は、ここでボリュームグループの名前を指定する必要があります。

```
<lvm_group>system</lvm_group>
```

pool
: LVM 論理ボリュームが LVM の thin プール内に存在すべきである場合は、`pool` の値を `true` にしなければなりません。

```
<pool config:type="boolean">true</pool>
```

used_pool

この thin 論理ボリュームに対して、データストアとして使用する LVM thin プールの名前を指定します。ここに何らかの値を設定すると、このボリュームは thin 論理ボリュームとして使用されるようになります。

```
<used_pool>my_thin_pool</used_pool>
```

raid_name

この物理ボリュームが RAID アレイの一部である場合は、ここで RAID アレイの名前を指定します。

```
<raid_name>/dev/md/0</raid_name>
```

raid_options

RAID のオプションを指定します。なお、`partition` 内に RAID のオプションを設定するのは非推奨になっています。詳しくは 4.5.6項「ソフトウエア RAID」をお読みください。

bcache_backing_for

このデバイスを `bcache` のバッキングデバイスとして使用する場合は、ここで `bcache` のデバイス名を指定します。詳しくは 4.5.8項「bcache の設定」をお読みください。

```
<bcache_backing_for>/dev/bcache0</bcache_backing_for>
```

bcache_caching_for

このデバイスを `bcache` のキャッシュデバイスとして使用する場合は、ここで `bcache` のデバイス名を指定します。詳しくは 4.5.8項「bcache の設定」をお読みください。

```
<bcache_caching_for config:type="list"><listentry>/dev/bcache0</listentry></
bcache_caching_for>
```

resize

openSUSE Leap 15 およびそれ以降では、サイズ変更は物理ディスクパーティションと LVM ボリュームの両方に対して動作するようになっています。

```
<resize config:type="boolean">false</resize>
```

4.5.3.3 btrfs サブボリューム

4.5.3.2項「パーティションの設定」で説明しているとおり、それぞれの btrfs ファイルシステムに対して、必要なサブボリュームを定義することができます。最もシンプルな形態では、下記のようになります:

```
<subvolumes config:type="list">
```

```
  <path>usr/local</path>
  <path>tmp</path>
  <path>opt</path>
  <path>srv</path>
  <path>var</path>
</subvolumes>
```

なお、各サブボリュームに対する追加設定を行うこともできます。たとえばクオータの設定やコピーオンライト機構の有効化などがそれにあたります。具体的には、下記の例のような形式で設定してください:

```
<subvolumes config:type="list">
  <listentry>usr/local</listentry>
  <listentry>
    <path>tmp</path>
    <referenced_limit>1 GiB</referenced_limit>
  </listentry>
  <listentry>opt</listentry>
  <listentry>srv</listentry>
  <listentry>
    <path>var/lib/pgsql</path>
    <copy_on_write config:type="boolean">false</copy_on_write>
  </listentry>
</subvolumes>
```

`path`

サブボリュームのマウントポイントを指定します。

```
<path>tmp</tmp>
```

必須です。`path` 要素が指定されていない場合、AutoYaST はそのサブボリュームを無視するようになります。

`copy-on-write`

このサブボリュームに対して、コピーオンライト機構を有効化するかどうかを指定します。

```
<copy-on-write config:type="boolean">false</copy-on-write>
```

任意指定です。既定値は `false` です。

`referenced_limit`

サブボリュームに対するクォータを指定します。

```
<referenced_limit>1 GiB</referenced_limit>
```

任意指定です。既定値は `unlimited` (無制限) です。btrfs では、2 種類の制限 (`referenced` (参照サイズ) および `exclusive` (排他サイズ)) を設定することができますが、この要素では前者を指定します。

ディストリビューション側で既定のサブボリュームが設定されている場合 (たとえば openSUSE Leap では `@`) 、この既定のサブボリュームの名前がこの一覧の中に自動的に追加されます。このような動作を無効化したい場合は、4.5.3.1項「ドライブの設定」で説明しているとおりに `subvolumes_prefix` を設定してください。

```
<subvolumes_prefix><![CDATA[]]></subvolumes_prefix>
```

4.5.3.4 ディスク全体の使用

AutoYaST では、4.5.3.1項「ドライブの設定」で説明している `disklabel` の値を `none` に設定することで、パーティションを作成せずにディスク全体を使用することができます。このような場合、`drive` 内の最初の `partition` 内の設定が、ディスク全体に対して適用されるようになります。

下記の例では、2 番目のディスク (`/dev/sdb`) 全体を `/home` のファイルシステムとして使用しています。

例 4.5: ディスク全体をファイルシステムとして使用する設定

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <partitions config:type="list">
      <partition>
        <create config:type="boolean">true</create>
        <format config:type="boolean">true</format>
        <mount>/</mount>
        <size>max</size>
      </partition>
    </partitions>
  </drive>
  <drive>
    <device>/dev/sdb</device>
    <disklabel>none</disklabel>
    <partitions config:type="list">
      <partition>
        <format config:type="boolean">true</format>
        <mount>/home</mount>
      </partition>
    </partitions>
  </drive>
```

これに加えて、ディスク全体を LVM 物理ボリュームやソフトウエア RAID のメンバーとして使用することもできます。LVM やソフトウエア RAID に関する詳細は、4.5.5項「論理ボリュームマネージャ (Logical Volume Manager (LVM))」および 4.5.6項「ソフトウエア RAID」をお読みください。

後方互換性を確保する目的で、<partition_nr> の要素を 0 に設定しても、同じ結果になるようになっています。ただし <partition_nr> 要素は、openSUSE Leap 15 およびそれ以降で廃止予定とされていることに注意してください。

4.5.3.5 不完全な設定の補完

[熟練者向けパーティション設定] の方式を使用している場合、AutoYaST は不完全なプロファイルからでも適切なパーティションプランを作成することができます。下記のプロファイルでは、パーティション配置に関するいくつかの詳細のみを指定し、それ以外を AutoYaST 側に任せる設定を示しています。

例 4.6: 選択したドライブに対する自動パーティション設定

> 下記はドライブが 1 台だけ搭載されたシステムでの例で、事前には何もパーティションを作成していないため、指定されたパーティションプランに従って自動的にパーティションを作成するように指定しています。なお、何もドライブを指定しない場合、ドライブは自動的に検出されます。

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <use>all</use>
  </drive>
</partitioning>
```

さらに詳しい例では、既存のパーティションが存在する場合や複数のドライブが存在する場合の処理方法について示しています。

例 4.7: 複数ドライブへのインストール

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <use>all</use>
    <partitions config:type="list">
      <partition>
        <mount>/</mount>
        <size>10G</size>
      </partition>
      <partition>
        <mount>swap</mount>
        <size>1G</size>
      </partition>
    </partitions>
```

```
  </drive>
  <drive>
    <device>/dev/sdb</device>
    <use>free</use>
    <partitions config:type="list">
      <partition>
        <filesystem config:type="symbol">ext4</filesystem>
        <mount>/data1</mount>
        <size>15G</size>
      </partition>
      <partition>
        <filesystem config:type="symbol">xfs</filesystem>
        <mount>/data2</mount>
        <size>auto</size>
      </partition>
    </partitions>
  </drive>
</partitioning>
```

4.5.4 高度なパーティション設定

4.5.4.1 パーティションテーブルの消去

通常は AutoYaST が 1 つずつパーティションを削除していくことから、この設定を行う必要はありません。ですが、AutoYaST に対してパーティションを個別に削除するのではなく、敢えて新しいパーティションテーブルを作成させたい場合、下記のような設定が必要になります。

具体的には、`drive` セクション内に下記を追加してください:

```
<initialize config:type="boolean">true</initialize>
```

このような設定を行うことで、AutoYaST は既存のパーティションの分析やパーティションプランの作成を行う前に、パーティションテーブルの消去を行うようになります。この設定を行うと、ご存じのとおり既存のパーティション内にある全てのデータは消去され、保持できなくなります。

4.5.4.2 マウントオプション

既定では、マウントすべきファイルシステムは `/etc/fstab` 内にデバイス名で指定されます。この指定方法は UUID やボリュームラベルに変更することもできます。ただし、全てのファイルシステムが UUID やボリュームラベルでのマウントに対応しているわけではないことに注意してください。パーティションのマウント方法を指定するには、`symbol` 型で指定する `mountby` プロパティを設定してください。設定可能な値は下記のとおりです:

- `device` (既定値)
- `label`
- `UUID`

ラベルを利用して新しいパーティションをマウントする場合、`label` プロパティで指定した名前をボリュームラベルとして使用します。

また、`/etc/fstab` の 4 番目の列に指定することのできる、マウントオプションを追加することもできます。複数のオプションはカンマで区切ります。指定可能な値は下記のとおりです:

読み込み専用でマウント (`ro`)
: ファイルシステムに対して書き込みを行うことができなくなります。既定値は `false` です。

アクセス日時を更新しない (`noatime`)
: ファイルが読み込まれた際、アクセス日時を更新しないようにします。既定値は `false` です。

ユーザにマウントを許可 (`user`)
: 一般ユーザからファイルシステムをマウントできるようにします。既定値は `false` です。

データジャーナリングモード (`ordered` , `journal` , `writeback`)

`journal`
: データを書き込む際、実際のファイルシステムに書き込む前にジャーナルに書き込むようにします。

`ordered`
: メタデータをジャーナルに書き込む前に、全てのデータを直接実際のファイルシステムに書き込みます。

`writeback`
: データの書き込み順序を維持しません。

アクセス制御リスト (`acl`)
: ファイルシステムに対してアクセス制御リストを有効化します。

拡張ユーザ属性（ `user_xattr` ）

ファイルシステムに対して拡張ユーザ属性を使用できるようにします。

例 4.8: マウントオプション

```
<partitions config:type="list">
  <partition>
    <filesystem config:type="symbol">ext4</filesystem>
    <format config:type="boolean">true</format>
    <fstopt>ro,noatime,user,data=ordered,acl,user_xattr</fstopt>
    <mount>/local</mount>
    <mountby config:type="symbol">uuid</mountby>
    <partition_id config:type="integer">131</partition_id>
    <size>10G</size>
  </partition>
</partitions>
```



注記: ファイルシステムごとに対応するオプションが異なる件について

ファイルシステムの種類が異なると、使用できるオプションもことなります。オプションを設定する倍は、あらかじめ対応するドキュメンテーションをお読みください。

4.5.4.3 特定のパーティションの維持

状況によっては、特定のパーティションのみをフォーマットして、それ以外のパーティションを何もせずにそのままにしておきたいような場合があります。たとえば異なる Linux インストールを共存させたいような場合や、他のオペレーティングシステムをインストールしたままにしておきたいような場合、これらのパーティションは消去せずに維持したままにすることができます。もちろんパーティション内のデータについても維持することができます。

このような構成を作成する場合、インストール先のシステムとそのハードディスクに対して、さまざまな考慮を行う必要があります。構成内容にもよりますが、インストール先のハードディスクのパーティションテーブルとパーティションの ID 、サイズと番号をそれぞれ知っておく必要があります。これらのデータをもとにして、 AutoYaST に対して特定のパーティションの維持を設定し、必要であれば新しいパーティションの作成を行うことになります。

下記の例では、パーティション 1,2,5 をそれぞれ維持し、パーティション 6 を削除して新しい 2 つのパーティションを作成しています。また、パーティション 2 以外を全てフォーマットして使用しています。

例 4.9: パーティションの維持

```
<partitioning config:type="list">
```

```
  <drive>
    <device>/dev/sdc</device>
      <partitions config:type="list">
        <partition>
          <create config:type="boolean">false</create>
          <format config:type="boolean">true</format>
          <mount>/</mount>
          <partition_nr config:type="integer">1</partition_nr>
        </partition>
        <partition>
          <create config:type="boolean">false</create>
          <format config:type="boolean">false</format>
          <partition_nr config:type="integer">2</partition_nr>
          <mount>/space</mount>
        </partition>
        <partition>
          <create config:type="boolean">false</create>
          <format config:type="boolean">true</format>
          <filesystem config:type="symbol">swap</filesystem>
          <partition_nr config:type="integer">5</partition_nr>
          <mount>swap</mount>
        </partition>
        <partition>
          <format config:type="boolean">true</format>
          <mount>/space2</mount>
          <size>5G</size>
        </partition>
        <partition>
          <format config:type="boolean">true</format>
          <mount>/space3</mount>
          <size>max</size>
        </partition>
      </partitions>
    <use>6</use>
  </drive>
</partitioning>
```

上記の例では、既存のパーティションテーブルのほか、維持すべきパーティションの番号についても明示的に指定しています。ただし、さまざまな種類のハードディスクや設定が混在しているような環境の場合、これらを明示的に指定することができない場合もあります。下記のシナリオでは、非 Linux OS のシステムと Linux インストール用の専用領域を確保する構成を示しています。

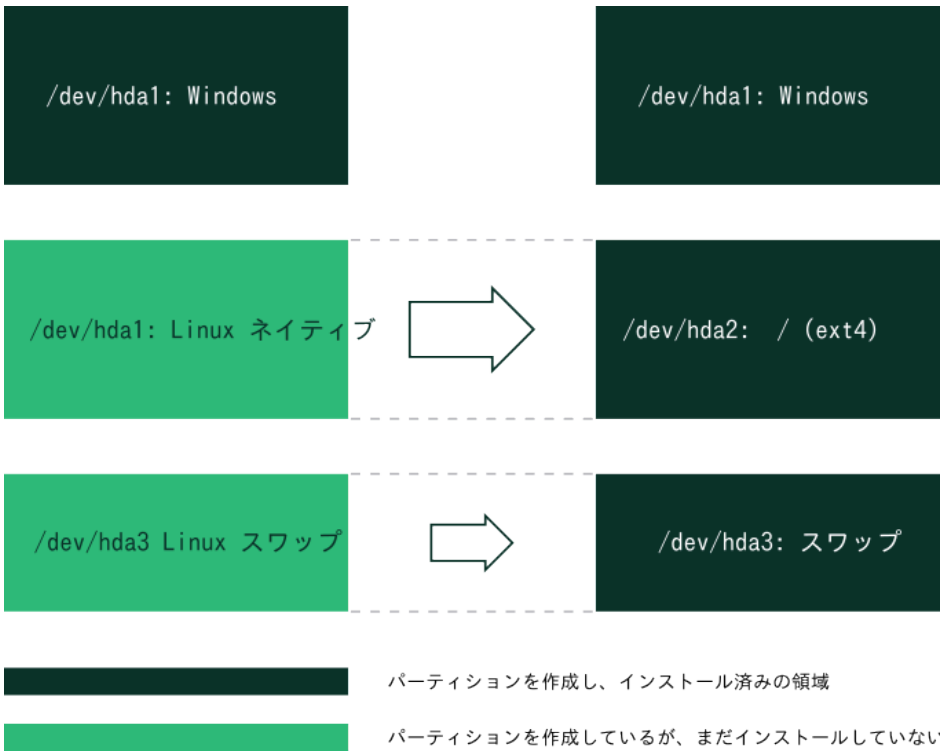

図 4.1: パーティションの維持

このシナリオでは、図4.1「パーティションの維持」の図に示しているとおり、AutoYaST では新しいパーティションを作成しないように設定します。その代わり、システム内に存在する特定の種類のパーティションを検出し、制御ファイルに書かれたパーティション設定プランに従ってそれらを使用するように構成しています。この場合、パーティションの番号は全く指定しておらず、マウントポイントとパーティションの種類のみを指定しています (たとえばファイルシステムのオプションや暗号化、ファイルシステムの種類などの追加データを設定することもできます) 。

例 4.10: 維持すべきパーティションの自動検出

```
<partitioning config:type="list">
  <drive>
    <partitions config:type="list">
      <partition>
        <create config:type="boolean">false</create>
        <format config:type="boolean">true</format>
        <mount>/</mount>
        <partition_id config:type="integer">131</partition_id>
      </partition>
      <partition>
        <create config:type="boolean">false</create>
        <format config:type="boolean">true</format>
        <filesystem config:type="symbol">swap</filesystem>
        <partition_id config:type="integer">130</partition_id>
        <mount>swap</mount>
      </partition>
```

```
    </partitions>
  </drive>
</partitioning>
```



注記: 暗号化されたデバイスの維持について

AutoYaST がストレージデバイスを検出する時点では、プロファイル内のパーティション設定セクションは解釈されていません。そのため、デバイスの暗号化を解除するために、どの鍵を使用すべきなのかを判別できません。たとえば、複数の暗号鍵を使用して暗号化を行っている場合がそれに該当します。このような問題を解決するため、AutoYaST では設定されている全ての鍵を利用して、デバイスの暗号化解除を試みるようになっています。

4.5.5 論理ボリュームマネージャ (Logical Volume Manager (LVM))

LVM を設定するには、上述の通常のパーティション作成を設定して、物理ボリュームを作成しておく必要があります。

例 4.11: LVM 物理ボリュームの作成

下記の例は、`partitioning` リソースで LVM 向けのパーティションを準備する場合の例です。この例では `/dev/sda1` を作成したあとフォーマットを行わず、`LVM` の種類を設定し、ボリュームグループ `system` に追加を行います。なお、パーティションのサイズはドライブ内の全空き領域分になります。

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <partitions config:type="list">
      <partition>
        <create config:type="boolean">true</create>
        <lvm_group>system</lvm_group>
        <partition_type>primary</partition_type>
        <partition_id config:type="integer">142</partition_id>
        <partition_nr config:type="integer">1</partition_nr>
        <size>max</size>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
</partitioning>
```

例 4.12: LVM 論理ボリューム

```
<partitioning config:type="list">
```

```
  <drive>
    <device>/dev/sda</device>
    <partitions config:type="list">
      <partition>
        <lvm_group>system</lvm_group>
        <partition_type>primary</partition_type>
        <size>max</size>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
  <drive>
    <device>/dev/system</device>
      <type config:type="symbol">CT_LVM</type>
      <partitions config:type="list">
        <partition>
          <filesystem config:type="symbol">ext4</filesystem>
          <lv_name>user_lv</lv_name>
          <mount>/usr</mount>
          <size>15G</size>
        </partition>
        <partition>
          <filesystem config:type="symbol">ext4</filesystem>
          <lv_name>opt_lv</lv_name>
          <mount>/opt</mount>
          <size>10G</size>
        </partition>
        <partition>
          <filesystem config:type="symbol">ext4</filesystem>
          <lv_name>var_lv</lv_name>
          <mount>/var</mount>
          <size>1G</size>
        </partition>
      </partitions>
      <pesize>4M</pesize>
    <use>all</use>
  </drive>
</partitioning>
```

論理ボリュームに対しても `size` に `max` を指定することができます。もちろん、いずれか 1 つ (!) の論理ボリュームにしか指定できません。また、1 つのボリュームグループ内で 2 つの論理ボリュームを `max` に設定することもできません。

4.5.6 ソフトウエア RAID

ソフトウエア RAID デバイスに対するサポートは、openSUSE Leap 15.2 で大幅に改善されています。

必要であれば、後方互換性維持の目的で残されているソフトウエア RAID の古い設定方法を利用することもできます。詳しくは 4.5.6.1項「古い書式の使用」をお読みください。

AutoYaST を使用することで、ソフトウエア RAID デバイスを作成したり構成したりすることができます。対応する RAID レベルは下記のとおりです:

RAID 0
: このレベルではディスク性能の強化が行われますが、冗長性は提供 されません 。いずれか 1 台のディスクがクラッシュすると、データの復元は不可能になります。

RAID 1
: このレベルでは最大限の冗長性を提供します。2 台以上のディスクを使用することができます。このレベルでは、全てのディスクに対して同じデータが書き込まれます。いずれか 1 台のディスクが正しく動作している限り、データが失われることはありません。この RAID レベルを使用する場合、使用するパーティションはほぼ同じサイズである必要があります。

RAID 5
: このレベルでは多数のディスクの管理と冗長性の両方を提供します。このレベルを使用するには、3 台以上のディスクが必要です。いずれか 1 台のディスクが壊れても、データが失われることはありませんが、2 台のディスクが同時に壊れてしまうと、データの復元は不可能になります。

マルチパス
: このモードでは、複数のコントローラを介して同じ物理デバイスにアクセスを行います。これはコントローラカードの障害に対して冗長性を提供するものです。このモードを使用するには、少なくとも 2 枚のカードが必要になります。

LVM と同様に、AutoYaST プロファイル内でのソフトウエア RAID 定義は、2 つの部品から構成されています:

- どのディスクやパーティションを RAID のメンバーとして使用するのかの設定。RAID メンバーとしての設定は、各デバイス要素内で `raid_name` を設定して行います。
- `drive` セクションによる RAID そのものの定義。

下記の例は、1 台目のディスクに存在するパーティションと、2 台目のディスクに存在するパーティションを RAID のメンバーとして、RAID10 を構成する場合の例です:

例 4.13: RAID10 設定

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <partitions config:type="list">
```

```
      <partition>
        <mount>/</mount>
        <size>20G</size>
      </partition>
      <partition>
        <raid_name>/dev/md/0</raid_name>
        <size>max</size>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
  <drive>
    <device>/dev/sdb</device>
    <disklabel>none</disklabel>
    <partitions config:type="list">
      <partition>
        <raid_name>/dev/md/0</raid_name>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
  <drive>
    <device>/dev/md/0</device>
    <partitions config:type="list">
      <partition>
        <mount>/home</mount>
        <size>40G</size>
      </partition>
      <partition>
        <mount>/srv</mount>
        <size>10G</size>
      </partition>
    </partitions>
    <raid_options>
      <chunk_size>4</chunk_size>
      <parity_algorithm>near_2</parity_algorithm>
      <raid_type>raid10</raid_type>
    </raid_options>
    <use>all</use>
  </drive>
</partitioning>
```

ソフトウエア RAID 内にパーティションを作成したくない場合は、通常のディスクと同様に `disklabel` を `none` に設定してください。下記の例では、単純化のために RAID の `drive` セクションのみを示しています:

例 4.14: パーティションを使用しない RAID10

```
<drive>
  <device>/dev/md/0</device>
```

```
  <disklabel>none</disklabel>
  <partitions config:type="list">
    <partition>
      <mount>/home</mount>
      <size>40G</size>
    </partition>
  </partitions>
  <raid_options>
    <chunk_size>4</chunk_size>
    <parity_algorithm>near_2</parity_algorithm>
    <raid_type>raid10</raid_type>
  </raid_options>
  <use>all</use>
</drive>
```

4.5.6.1 古い書式の使用

インストーラの自己更新機能が有効化されている場合、openSUSE Leap 15 でソフトウエア RAID のパーティションを作成することができます。ただし、それ以前のバージョンではこのような仕組みには対応していないため、ソフトウエア RAID の定義が少し異なることになります。

本章では、後方互換性のために提供されている古い形式の設定について説明しています。

古い形式で RAID を設定する場合、下記の点に注意してください:

- RAID デバイスは必ず `/dev/md` になります。
- `partition_nr` プロパティは、MD デバイスの番号を判断するために使用します。たとえば `partition_nr` に 0 を設定すると、`/dev/md/0` が設定されます。複数の `partition` セクションを設定すると、複数のソフトウエア RAID (`/dev/md/0` , `/dev/md/1` ,...) を作成する意味になります。
- RAID 関連の全てのオプションは、`raid_options` 内に含める必要があります。

例 4.15: 古い書式の RAID10 設定

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <partitions config:type="list">
      <partition>
        <partition_id config:type="integer">253</partition_id>
        <format config:type="boolean">false</format>
        <raid_name>/dev/md0</raid_name>
        <raid_type>raid1</raid_type>
        <size>4G</size>
      </partition>
```

```
        <!-- /dev/sda に配置する通常のパーティション設定（例: /, swap）
                をここに記述します -->

    </partitions>
    <use>all</use>
  </drive>
  <drive>
    <device>/dev/sdb</device>
    <partitions config:type="list">
      <partition>
        <format config:type="boolean">false</format>
        <partition_id config:type="integer">253</partition_id>
        <raid_name>/dev/md0</raid_name>
        <size>4gb</size>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
  <drive>
    <device>/dev/md</device>
    <partitions config:type="list">
      <partition>
        <filesystem config:type="symbol">ext4</filesystem>
        <format config:type="boolean">true</format>
        <mount>/space</mount>
        <partition_id config:type="integer">131</partition_id>
        <partition_nr config:type="integer">0</partition_nr>
        <raid_options>
          <chunk_size>4</chunk_size>
          <parity_algorithm>near_2</parity_algorithm>
          <raid_type>raid10</raid_type>
        </raid_options>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
</partitioning>
```

4.5.6.2 RAID オプション

下記のような XML 構造内に、必要な要素を追加していきます:

```
<partition>
     <raid_options>
     ...
     </raid_options>
     </partition>
```

chunk_size

単位付きの表記 (例: 32M) のほか、単純に数値のみを指定することもできます。数値のみを指定した場合は、キロバイト単位で指定したものと見なされます。なお、RAID1 の場合は `chunk_size` を指定してはいけません。また、RAID レベルを指定していない場合、`raid1` が既定値であることにも注意してください。

```
<chunk_size>4</chunk_size>
```

parity_algorithm

設定可能な値は下記のとおりです:
`left_asymmetric`, `left_symmetric`, `right_asymmetric`, `right_symmetric`, `first`, `last`, `first_6`, `left_asymmetric_6`, `left_symmetric_6`, `right_asymmetric_6`, `right_symmetric_6`, `near_2`, `offset_2`, `far_2`, `near_3`, `offset_3`, `far_3`.
以前のバージョンの AutoYaST との間での互換性維持のため、下記のような別名も指定することができます:
`parity_first`, `parity_last`, `parity_first_6`, `n2`, `o2`, `f2`, `n3`, `o3`, `f3`.
設定可能な値は RAID のレベル (例: `raid5`) と RAID 内のデバイス数によって異なります。なお、RAID0 と RAID1 はパリティによる冗長化を提供しないため、これらのデバイスに対してはオプションを指定してはなりません。

```
<parity_algorithm>left_asymmetric</parity_algorithm>
```

raid_type

値には `raid0`, `raid1`, `raid5`, `raid6`, `raid10` のいずれかを指定します。

```
<raid_type>raid1</raid_type>
```

既定値は `raid1` です。

device_order

この一覧には、物理デバイスの設定順序を指定します:

```
<device_order config:type="list"><device>/dev/sdb2</device><device>/dev/sda1</
device>...</device_order>
```

この設定は任意設定で、既定ではアルファベット順です。

4.5.7 マルチパスサポート

AutoYaST ではマルチパスデバイスを処理することができます。マルチパスデバイスを使用するには、まず 例4.16「マルチパスデバイスの使用」で説明しているように、マルチパスサポートの有効化を行う必要があります。それ以外にも、カーネルのコマンドラインに `LIBSTORAGE_MULTIPATH_AUTOSTART=ON` を追加してもかまいません。

例 4.16: マルチパスデバイスの使用

```
<general>
  <storage>
    <start_multipath config:type="boolean">true</start_multipath>
  </storage>
</general>
<partitioning>
  <drive>
    <partitions config:type="list">
      <partition>
        <size>20G</size>
        <mount>/</mount>
        <filesystem config:type="symbol">ext4</filesystem>
      </partition>
      <partition>
        <size>auto</size>
        <mount>swap</mount>
      </partition>
    </partitions>
    <type config:type="symbol">CT_DISK</type>
    <use>all</use>
  </drive>
</partitioning>
```

デバイスを指定したい場合は、マルチパスデバイスを表す World Wide Identifier (WWID) かデバイス名 (例: `/dev/dm-0`)、もしくは `/dev/disk` 以下のデバイスパスを指定してください。

たとえば 例4.17「マルチパスデバイスの列挙」のような `multipath` コマンドの出力があった場合、`/dev/mapper/14945540000000000f86756dce9286158be4c6e3567e75ba5` のほか、`/dev/dm-3` や `/dev/disk` 以下のデバイスパス (例: 例4.18「WWID を使用してマルチパスデバイスを指定する場合の例」) を指定することができるほか、いずれかのデバイスパス (例: `/dev/sda` or `/dev/sdb`) を指定してもかまいません。

例 4.17: マルチパスデバイスの列挙

```
# multipath -l
14945540000000000f86756dce9286158be4c6e3567e75ba5 dm-3 ATA,VIRTUAL-DISK
```

```
size=40G features='0' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=1 status=active
| `- 2:0:0:0 sda 8:0  active ready running
`-+- policy='service-time 0' prio=1 status=enabled
  `- 3:0:0:0 sdb 8:16 active ready running
```

例 4.18: WWID を使用してマルチパスデバイスを指定する場合の例

```
<drive>
  <partitions config:type="list">
    <device>/dev/mapper/14945540000000000f86756dce9286158be4c6e3567e75ba5</device>
    <partition>
      <size>20G</size>
      <mount>/</mount>
      <filesystem config:type="symbol">ext4</filesystem>
    </partition>
  </partitions>
  <type config:type="symbol">CT_DISK</type>
  <use>all</use>
</drive>
```

4.5.8 bcache の設定

`bcache` は複数の高速なドライブを利用して、1 つまたは複数の低速なドライブを高速化するための仕組みです。たとえば特定の高速なドライブをキャッシュとして使用し、巨大で遅いドライブの性能を改善したりすることができます。

openSUSE Leap での `bcache` に関する詳細は、https://documentation.suse.com/sles/15/html/SLES-all/cha-multitiercache.html#sec-multitiercache-bcache (英語) の記事をお読みください。

`bcache` のデバイスを設定するには、AutoYaST 側で下記のようなプロファイル設定が必要となります:

- (遅いほうの) ブロックデバイスを *バッキングデバイス* として設定するには、`bcache_backing_for` 要素を使用します。
- (早いほうの) ブロックデバイスを *キャッシュデバイス* して設定するには、`bcache_caching_for` 要素を使用します。複数のドライブの速度を上げるために、1 台のデバイスを使用するような設定も行うことができます。
- `bcache` デバイスの配置方法を指定するには、`drive` セクション内に `type` 要素が `CT_BCACHE` であるものを設定します。`bcache` デバイス内には、パーティションを含めることもできます。

例 4.19: bcache の定義

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <type config:type="symbol">CT_DISK</type>
    <use>all</use>
    <enable_snapshots config:type="boolean">true</enable_snapshots>
    <partitions config:type="list">
      <partition>
        <filesystem config:type="symbol">btrfs</filesystem>
        <mount>/</mount>
        <create config:type="boolean">true</create>
        <size>max</size>
      </partition>
      <partition>
        <filesystem config:type="symbol">swap</filesystem>
        <mount>swap</mount>
        <create config:type="boolean">true</create>
        <size>2GiB</size>
      </partition>
    </partitions>
  </drive>

  <drive>
    <type config:type="symbol">CT_DISK</type>
    <device>/dev/sdb</device>
    <disklabel>msdos</disklabel>
    <use>all</use>
    <partitions config:type="list">
      <partition>
        <!-- It can serve as caching device for several bcaches -->
        <bcache_caching_for config:type="list">
          <listentry>/dev/bcache0</listentry>
        </bcache_caching_for>
        <size>max</size>
      </partition>
    </partitions>
  </drive>

  <drive>
    <type config:type="symbol">CT_DISK</type>
    <device>/dev/sdc</device>
    <use>all</use>
    <disklabel>msdos</disklabel>
    <partitions config:type="list">
      <partition>
        <!-- It can serve as backing device for one bcache -->
        <bcache_backing_for>/dev/bcache0</bcache_backing_for>
      </partition>
    </partitions>
```

```
  </drive>

  <drive>
    <type config:type="symbol">CT_BCACHE</type>
    <device>/dev/bcache0</device>
    <bcache_options>
      <cache_mode>writethrough</cache_mode>
    </bcache_options>
    <use>all</use>
    <partitions config:type="list">
      <partition>
        <mount>/data</mount>
        <size>20GiB</size>
      </partition>
      <partition>
        <mount>swap</mount>
        <filesystem config:type="symbol">swap</filesystem>
        <size>1GiB</size>
      </partition>
    </partitions>
  </drive>
</partitioning>
```

現時点では、`bcache_options` で使用することのできるオプションは `cache_mode` だけとなります。

cache_mode

`bcache` のキャッシュモードを設定します。設定可能な値は下記のとおりです:

- `writethrough`
- `writeback`
- `writearound`
- `none`

```
<cache_mode>writethrough</cache_mode>
```

4.5.9 マルチデバイス型 btrfs の設定

btrfs では複数のデバイスにまたがった単一ボリュームを作成することができます。この仕組みにより、Linux カーネル内蔵の `mdraid` サブシステムのようなソフトウエア RAID に似た実装を使用することができるようになります。マルチデバイス型の btrfs には、他の RAID 実装では提供されていないいくつかの利点があります。たとえば一方の RAID レベルから他方の RAID レベルに動的に移行できる機能や、ファイル単位での RAID レベルの設定などがあります。ただし、openSUSE Leap 15 . 7 では、これら全ての機能をサポートしているわけではないことに注意してください。

AutoYaST では `CT_BTRFS` を指定することで、マルチデバイス型の btrfs を設定することができます。また、`device` 属性では任意の名前を設定して、マルチデバイス型の btrfs を識別することができます。

RAID として使用する場合は、まず必要な全てのブロックデバイス (パーティション、LVM 論理ボリュームなど) を作成しておいて、それらを btrfs ファイルシステムに割り当てておく必要があります。

下記の例では、シンプルなマルチデバイス型 btrfs 設定を示しています:

例 4.20: マルチデバイス型 BTRFS の設定

```
<partitioning config:type="list">
  <drive>
    <device>/dev/sda</device>
    <disklabel>none</disklabel>
    <partitions>
      <partition>
        <btrfs_name>root_fs</btrfs_name>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
  <drive>
    <device>/dev/sdb</device>
    <disklabel>gpt</disklabel>
    <partitions>
      <partition>
        <partition_nr>1</partition_nr>
        <size>4gb</size>
        <filesystem>ext4</filesystem>
        <btrfs_name>root_fs</btrfs_name>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
  <drive>
    <device>root_fs</device>
    <type config:type="symbol">CT_BTRFS</type>
    <partitions>
      <partition config:type="list>
        <mount>/</mount>
      </partition>
    </partitions>
    <btrfs_options>
      <raid_leve>raid1</raid_level>
      <metadata_raid_leve>raid1</metadata_raid_level>
    </btrfs_options>
  </drive>
</partitioning>
```

サポート対象となるデータおよびメタデータの RAID レベルは、`default`，`single`，`dup`，`raid0`，`raid1`，`raid10` のいずれかです。既定ではファイルシステムのメタデータは 2 つのデバイスに複製され、データは全てのデバイスに複製されます。デバイスが 1 つしか存在しない場合、メタデータは 1 つのデバイス内で複製されます。

マルチデバイス型の btrfs ファイルシステムを設定する際には、下記の点に注意してください:

- マルチデバイス型の btrfs ファイルシステム内に組み込まれるよう、必要なデバイスに対しては `btrfs_name` 属性を設定する必要があります。
- btrfs 関連のオプションを設定する場合は、`CT_BTRFS` ドライブの `btrfs_options` リソース内に記述します。

4.5.10 NFS の設定

AutoYaST では、openSUSE Leap を Network File System (NFS) の共有内にインストールすることもできます。これを実現するには、まず `CT_NFS` の種類でドライブを作成し、デバイス名として NFS の共有名を (`サーバ:パス`) の形式で指定します。マウントポイントに関する情報は、最初のパーティションセクション内に記述します。なお、NFS ドライブの場合、最初のパーティションしか使用されないことに注意してください。

システムをインストールした後の NFS クライアントやサーバの設定方法については、4.20項「NFS クライアントおよびサーバ」をお読みください。

例 4.21: NFS 共有の定義

```
<partitioning config:type="list">
  <drive>
    <device>192.168.1.1:/exports/root_fs</device>
    <type config:type="symbol">CT_NFS</type>
    <use>all</use>
    <partitions config:type="list">
      <partition>
        <mount>/</mount>
        <fstopt>nolock</fstopt>
      </partition>
    </partitions>
  </drive>
</partitioning>
```

4.5.11 tmpfs の設定

AutoYaST では type 要素に CT_TMPFS を指定することで、tmpfs 仮想ファイルシステムを設定することができます。この場合、1 つの partition セクションが 1 つの tmpfs ファイルシステムを表すように設定します。

例 4.22: tmpfs の設定例

```
    <partitioning config:type="list">
      <drive>
        <type config:type="symbol">CT_TMPFS</type>
        <partitions config:type="list">
          <partition>
            <mount>/srv</mount>
            <fstopt>size=512M</fstopt>
          </partition>
          <partition>
            <mount>/temp</mount>
          </partition>
        </partitions>
      <drive>
    <partitioning>
```

tmpfs は ext4 や btrfs などの通常のファイルシステムとは異なり、mount 要素 (必須) と fstopt 要素のみを指定します。fstopt 要素はサイズや制限、モードなどのファイルシステムの属性を設定します。設定可能な値について、詳しくは tmpfs のマニュアルページをお読みください。

4.6 iSCSI イニシエータの概要

iscsi-client リソースを使用することで、インストール先のマシンを iSCSI クライアントとして設定することができます。

例 4.23: ISCSI クライアント

```
<iscsi-client>
  <initiatorname>iqn.2013-02.de.suse:01:e229358d2dea</initiatorname>
  <targets config:type="list">
    <listentry>
       <authmethod>None</authmethod>
       <portal>192.168.1.1:3260</portal>
       <startup>onboot</startup>
       <target>iqn.2001-05.com.doe:test</target>
       <iface>default</iface>
    </listentry>
  </targets>
  <version>1.0</version>
```

```
  </iscsi-client>
```

ISCSI イニシエータ設定

initiatorname
: `InitiatorName` は `/etc/iscsi/initiatorname.iscsi` からの値を指定します。iBFT を使用している場合、この値は BIOS から設定されるため、BIOS セットアップで変更する必要があります。

version
: YaST モジュールのバージョンを指定します。既定値: 1.0

targets
: ターゲットの一覧を指定します。各項目には下記の値を設定することができます:

  authmethod
  : 認証方式: None/CHAP

  portal
  : ポータルアドレス

  startup
  : 値: manual/onboot

  target
  : ターゲット名

  iface
  : インターフェイス名

4.7 Fibre Channel over Ethernet (FCoE) の設定

`fcoe_cfg` リソースを使用することで、Fibre Channel over Ethernet (FCoE) の設定を行うことができます。

例 4.24: FCOE の設定

```
<fcoe-client>
  <fcoe_cfg>
    <DEBUG>no</DEBUG>
    <USE_SYSLOG>yes</USE_SYSLOG>
  </fcoe_cfg>
  <interfaces config:type="list">
    <listentry>
      <dev_name>eth3</dev_name>
```

```
        <mac_addr>01:000:000:000:42:42</mac_addr>
        <device>Gigabit 1313</device>
        <vlan_interface>200</vlan_interface>
        <fcoe_vlan>eth3.200</fcoe_vlan>
        <fcoe_enable>yes</fcoe_enable>
        <dcb_required>yes</dcb_required>
        <auto_vlan>no</auto_vlan>
        <dcb_capable>no</dcb_capable>
        <cfg_device>eth3.200</cfg_device>
      </listentry>
    </interfaces>
    <service_start>
      <fcoe config:type="boolean">true</fcoe>
      <lldpad config:type="boolean">true</lldpad>
    </service_start>
  </fcoe-client>
```

fcoe_cfg

値: `yes` もしくは `no`

`DEBUG` は、fcoe サービススクリプトや fcoemon から出力することのできるデバッグメッセージを、有効化するかどうかを指定します。

`USE_SYSLOG` を yes に設定すると、メッセージを syslog に送信するようになります。

interfaces

VLAN や FCoE の設定状態を含む、ネットワークカードの一覧を指定します。

service_start

値: `yes` もしくは `no`

`fcoe` と `lldpad` の各サービスを、システムの起動時に開始するかどうかを指定します。

`fcoe` サービスを開始すると、FCoE インターフェイスを制御し、`lldpad` デーモンとの通信を確立する Fibre Channel over Ethernet サービスデーモン `fcoemon` を起動します。

`lldpad` サービスは Link Layer Discovery Protocol agent daemon (`lldpad`) と呼ばれるサービスを提供するもので、このサービスは `fcoemon` に対して DCB (Data Center Bridging) 機能とインターフェイスの設定を通知します。

4.8 国ごとの設定

言語やタイムゾーン、キーボードの設定などを行います。

例 4.25: 言語

```
      <language>
        <language>en_GB</language>
        <languages>de_DE,en_US</languages>
```

```
    </language>
```

language
: 第一言語

languages
: 第二言語 (複数可、カンマ区切り)

利用可能な言語の一覧については、`/usr/share/YaST2/data/languages` ファイルをご覧ください。

第一言語に指定した値が正しくない場合、第一言語は既定値である `en_US` に設定されます。

例 4.26: タイムゾーン

```
    <timezone>
      <hwclock>UTC</hwclock>
      <timezone>Europe/Berlin</timezone>
    </timezone>
```

hwclock
: ハードウエアの時計がローカルの時刻 (localtime) を示しているのか、UTC を示しているのかを指定します。
値: `localtime` もしくは `UTC`

timezone
: タイムゾーンを指定します。
設定可能なタイムゾーンの一覧については、`/usr/share/YaST2/data/timezone_raw.ycp` ファイルをご覧ください。

例 4.27: キーボード

```
    <keyboard>
      <keymap>german</keymap>
    </keyboard>
```

keymap
: キーボードレイアウト
キーマップコードの値、もしくはキーマップの別名値を指定します。設定可能なキーボードレイアウトの一覧については、`/usr/share/YaST2/lib/y2keyboard/keyboards.rb` ファイルをご覧ください (例: `english-us, us, english-uk, japanese`)。

4.9 ソフトウエア

4.9.1 パターンやパッケージの選択

パターンやパッケージは下記のようにして設定します:

例 4.28: パターンやパッケージのセクションを利用した制御ファイル内でのパッケージ選択

```
<software>
  <patterns config:type="list">
    <pattern>directory_server</pattern>
  </patterns>
  <packages config:type="list">
    <package>apache</package>
    <package>postfix</package>
  </packages>
  <do_online_update config:type="boolean">true</do_online_update>
</software>
```



注記: パッケージとパターンの名前について

値にはパッケージ名もしくはパターン名を指定します。バージョンが変わることによってパッケージ名が変更されることもありますが、その場合はプロファイル内に設定した名前を変更する必要があります。

また、正規表現を利用してパッケージ名やパターン名を選択することもできます。この場合、AutoYaSTは指定した条件に合致する全てのパッケージをインストールしようとします。なお、正規表現はスラッシュで括らなければならないことに注意してください。たとえば 例4.29「正規表現を利用したパッケージ選択」の例では、`nginx` で始まる名前の全てのパッケージ (`nginx` や `nginx-macros` など) を選択することになります。

例 4.29: 正規表現を利用したパッケージ選択

```
<software>
  <packages config:type="list">
    <package>/nginx.*/</package>
  </packages>
</software>
```

4.9.2 イメージの配置

インストールにかかる時間を削減するため、イメージを利用してインストールすることができます。

例 4.30: イメージ配置の有効化

```
<!-- 注意! software セクション内には記述しません! -->
```

```
<deploy_image>
  <image_installation config:type="boolean">false</image_installation>
</deploy_image>
```

4.9.3 追加のパッケージやカスタマイズしたパッケージ、もしくは独自の製品のインストール

DVD-ROM 内に用意されているパッケージに加えて、必要であればカスタマイズしたカーネルなど、外部のパッケージをインストールすることもできます。なお、カスタマイズしたカーネルを使用する場合は、SUSE パッケージとの互換性があるものでなければならないほか、同じ場所にインストールしなければならないことにも注意してください。

従来のバージョンとは異なり、独自のパッケージや外部のパッケージをインストールするにあたって、特別なリソースを設定する必要はありません。その代わり、インストール元のリポジトリに必要なパッケージを追加したあと、パッケージデータベースを再作成する必要があります。

このような作業に対応するため、リポジトリ内に存在するパッケージを調べてパッケージデータベースを作成するためのスクリプトが用意されています。これは `/usr/bin/create_package_descr` というコマンドになります。このコマンドは openSUSE Build Service 内の `inst-source-utils` パッケージ内に含まれています。なお、データベースを作成する際、全ての言語は英語にリセットされます。

例 4.31: INST-SOURCE-UTILS.RPM 追加パッケージを利用したパッケージデータベースの作成

DVD 内のファイルを `/usr/local/DVDs/LATEST` に展開します。

```
> cp /tmp/inst-source-utils-2016.7.26-1.2.noarch.rpm /usr/local/DVDs/LATEST/suse/
noarch
> cd /usr/local/DVDs/LATEST/suse
> create_package_descr -d /usr/local/CDs/LATEST/suse
```

上記の例では、`/usr/local/CDs/LATEST/suse` 以下にアーキテクチャ (例: `x86_64`) に依存するパッケージと依存しないパッケージ (`noarch`) の両方が存在するものとします。他のアーキテクチャの場合は少し異なるかもしれません。

この方式を使用することで、あらかじめ決められた最新のパッケージを提供する最新のリポジトリを提供できるようになります。また、この方式を使用することで、独自の CD-ROM を簡単に作成することもできます。

たとえば SDK (SUSE Software Development Kit) のような独自モジュールを追加したい場合は、インストール元となるディレクトリのルートディレクトリに対して、`add_on_products.xml` ファイルを追加します。

下記の例では、基本製品のリポジトリ内に SDK モジュールを追加する方法を示しています。SDK のリポジトリそのものは `/sdk` 内にあるものとします。

例 4.32: add_on_products.xml

このファイルは、基本製品内に SDK モジュールを追加するためのファイルです。

```
<?xml version="1.0"?>
<add_on_products xmlns="http://www.suse.com/1.0/yast2ns"
   xmlns:config="http://www.suse.com/1.0/configns">
   <product_items config:type="list">
      <product_item>
         <name>SUSE Linux Enterprise Software Development Kit</name>
         <url>relurl:////sdk?alias=SLE_SDK</url>
         <path>/</path>
         <-- Users are asked whether to add such a product -->
         <ask_user config:type="boolean">false</ask_user>
         <-- Defines the default state of pre-selected state in case of ask_user
 used. -->
         <selected config:type="boolean">true</selected>
      </product_item>
   </product_items>
</add_on_products>
```

上記のような特別な例のほかにも、その他のモジュールや拡張、アドオン製品などをそれぞれ別々のディレクトリに配置して、AutoYaST のインストール時に追加するようなことも実現することができます。

製品やモジュールの情報を持たないリポジトリであっても、インストール時に追加を行うことができます。これらは **その他のアドオン** と呼ばれます。

例 4.33: SDK 拡張とユーザ定義リポジトリの追加

```
<add-on>
  <add_on_products config:type="list">
    <listentry>
      <media_url>cd:///sdk</media_url>
      <product>sle-sdk</product>
      <alias>SLE SDK</alias>
      <product_dir>/</product_dir>
      <priority config:type="integer">20</priority>
      <ask_on_error config:type="boolean">false</ask_on_error>
      <confirm_license config:type="boolean">false</confirm_license>
      <name>SUSE Linux Enterprise Software Development Kit</name>
    </listentry>
  </add_on_products>
  <add_on_others config:type="list">
    <listentry>
      <media_url>https://download.opensuse.org/repositories/YaST:/Head/openSUSE_Leap_15.2/
</media_url>
      <alias>yast2_head</alias>
      <priority config:type="integer">30</priority>
      <name>Latest YaST2 packages from OBS</name>
    </listentry>
```

```
  </add_on_others>
</add-on>
```

`add_on_others` と `add_on_products` のセクションでは、それぞれ同じ値に対応しています:

media_url
: 製品の URL を指定します。`cd:///` , `http://` , `ftp://` などのプレフィクスを使用することができます。この項目は必須です。
SUSE Linux Enterprise のパッケージ DVD などのように複数の製品が含まれるメディアを使用する場合は、URL パスが複数製品のメディアのルートディレクトリを指し示すように設定し、`product_dir` でそれぞれの製品を指定する必要があります (詳しくは下記をお読みください) 。

product
: アドオンが製品である場合、内部向けの製品名を指定します。`zypper products` を実行すると、インストール済みの製品名を表示することができます。

alias
: リポジトリの別名を指定します。ユーザ側で任意の名前を設定することができます。

product_dir
: 製品のサブディレクトリを指定します。この属性は、SUSE Linux Enterprise のパッケージ DVD など、複数の製品が含まれるメディアである場合にのみ使用します。

priority
: リポジトリに対して設定する libzypp の優先順位を指定します。1 が最も高い優先順位を表し、数値が大きくなればなるほど、優先順位が低くなります。既定値は 99 です。

ask_on_error
: AutoYaST に対して、指定した場所にアドオン製品やモジュール、拡張のリポジトリが存在しない場合、タイムアウトエラーを報告するのではなく、ユーザに対して尋ねるように設定するかどうかを指定します。ユーザに対して尋ねたい場合は、ask_on_error を `true` に設定します (既定値は `false` です) 。

confirm_license
: ユーザに対してライセンスへの同意を求めるかどうかを指定します。既定値は `false` です。

name
: リポジトリ名を指定します。`zypper lr` コマンドを実行することで、追加されたリポジトリの名前を表示することができます。

AutoYaST で未署名のインストールソースを使用するには、AutoYaST の制御ファイル内で下記のような内容を設定し、チェックを無効化してください。



注記: 未署名のインストールソースの制限について

ここでの署名チェックの無効化は、自動インストール処理内の第 1 ステージにのみ適用されます。第 2 ステージでのインストールは、AutoYaST の設定ではなく、インストール済みのシステムでの設定に従って動作します。

下記のような XML 構造内に、必要な要素を追加していきます:

```
<general>
  <signature-handling>
    ...
  </signature-handling>
</general>
```

下記に示す全てのオプションの既定値は `false` です。また、オプションが `false` に設定されていて、パッケージやリポジトリに対する対象のテストが失敗した場合、特に何もメッセージが表示されることなく無視され、パッケージがインストールされなくなります。なお、どの値を `true` に設定したとしても、そこには潜在的なセキュリティリスクが存在することに注意してください。また、サードパーティ製のインストール元を利用してインストールを行う場合、これらの設定を行ってはなりません。

accept_unsigned_file

`true` に設定した場合、AutoYaST はコンテンツファイルなどが未署名であっても、受け入れるようになります。

```
<accept_unsigned_file config:type="boolean" >true</accept_unsigned_file>
```

accept_file_without_checksum

`true` に設定した場合、AutoYaST はコンテンツファイルなどのチェックサムが存在しない場合であっても、受け入れるようになります。

```
<accept_file_without_checksum config:type="boolean" >true</
accept_file_without_checksum>
```

accept_verification_failed

`true` に設定した場合、AutoYaST は署名の検証が失敗した場合でもファイルを受け入れるようになります。

```
<accept_verification_failed config:type="boolean" >true</accept_verification_failed>
```

accept_unknown_gpg_key

`true` に設定した場合、AutoYaST はインストール元に新しい GPG 鍵が存在し、その鍵でコンテンツファイルに署名されている場合、それを受け付けるようになります。

```
<accept_unknown_gpg_key config:type="boolean" >true</accept_unknown_gpg_key>
```

accept_non_trusted_gpg_key
: `true` に設定した場合、既知ではあるもののまだ信頼していない鍵を受け付けるようになります。

```
<accept_non_trusted_gpg_key config:type="boolean" >true</accept_non_trusted_gpg_key>
```

import_gpg_key
: `true` に設定した場合、AutoYaST はインストール元にある新しい GPG 鍵を受け付け、データベース内に取り込むようになります。

```
<import_gpg_key config:type="boolean" >true</import_gpg_key>
```

それぞれのアドオン製品や拡張ごとに、署名の処理方法を設定することもできます。下記に示す要素は、個別のアドオン製品やモジュール、拡張などの中の `signature-handling` セクション内に設定しなければならないものです。いずれの設定も任意です。何も設定しない場合、`general` セクション内のグローバルな署名処理設定が適用されます。

accept_unsigned_file
: `true` に設定した場合、AutoYaST は対象のアドオン製品に対して、コンテンツファイルなどが未署名であっても、受け入れるようにします。

```
<accept_unsigned_file config:type="boolean" >true</accept_unsigned_file>
```

accept_file_without_checksum
: `true` に設定した場合、AutoYaST は対象のアドオン製品に対して、コンテンツファイルなどのチェックサムが存在しない場合であっても、受け入れるようにします。

```
<accept_file_without_checksum config:type="boolean" >true</
accept_file_without_checksum>
```

accept_verification_failed
: `true` に設定した場合、AutoYaST は対象のアドオン製品に対して、署名の検証が失敗した場合でもファイルを受け入れるようにします。

```
<accept_verification_failed config:type="boolean" >true</accept_verification_failed>
```

accept_unknown_gpg_key
: `all` を `true` に設定した場合、AutoYaST は対象のアドオン製品に対して、インストール元にある新しい GPG 鍵を受け付け、データベース内に取り込むようにします。

```
<accept_unknown_gpg_key> <all config:type="boolean">true</all> </
accept_unknown_gpg_key>
```

鍵を個別に指定することもできます:

```
<accept_unknown_gpg_key> <all config:type="boolean">false</all> <keys
 config:type="list"> <keyid>3B3011B76B9D6523</keyid> lt;/keys> </
accept_unknown_gpg_key>
```

accept_non_trusted_gpg_key

`true` に設定した場合、既知ではあるもののまだ信頼していない鍵を受け付けるようにします。

```
<accept_non_trusted_gpg_key> <all config:type="boolean">true</all> </
accept_non_trusted_gpg_key>
```

特定の鍵のみを信頼する場合は下記のように設定します:

```
<accept_non_trusted_gpg_key> <all config:type="boolean">false</all>
 <keys config:type="list"> <keyid>3B3011B76B9D6523</keyid> </keys> </
accept_non_trusted_gpg_key>
```

import_gpg_key

`all` を `true` に設定した場合、AutoYaST はインストール元にある全ての GPG 鍵を受け入れ、データベース内に取り込むようにします。

```
<import_gpg_key> <all config:type="boolean">true</all> </import_gpg_key>
```

鍵を個別に指定することもできます:

```
<import_gpg_key> <all config:type="boolean">false</all> <keys config:type="list">
 <keyid>3B3011B76B9D6523</keyid> </keys> </import_gpg_key>
```

4.9.4 カーネルパッケージ

カーネルパッケージはパッケージの選択の中に含めることができません。必要なカーネルはインストール時に自動判断されるためです。パッケージの選択内にカーネルパッケージを入れてしまうと、ほとんどの場合、矛盾が発生してインストールが失敗することになります。

特定の種類のカーネルを強制したい場合は、`kernel` プロパティで設定を行ってください。下記は既定のカーネル (kernel-default) を強制する場合の例です。下記のように設定することで、環境によらずに指定したカーネルをインストールするようになります。

例 4.34: 制御ファイル内でのカーネルセクション

```
<software>
  <kernel>kernel-default</kernel>
  ...
</software>
```

4.9.5 自動選択したパッケージの削除

依存関係にあるパッケージや推奨パッケージなど、いくつかのパッケージが自動的に選択されてインストールされることがあります。

これらのパッケージを削除してしまうと、システムの一貫性を損なうことになるほか、同類のサービスを提供するような代替パッケージをインストールしない限り、基本的なパッケージについても、削除を行わないことをお勧めします。最も分かりやすい例がメール転送エージェント (MTA) です。既定では `postfix` がインストールされますが、`sendmail` のような他の MTA をインストールすれば、postfix を一覧から削除することができます (ただし、sendmail は openSUSE Leap では提供されていません) 。下記の例では、このように postfix を sendmail に置き換える場合の例を示しています:

例 4.35: 制御ファイル内でのパッケージ選択

```
<software>
  <packages config:type="list">
    <package>sendmail</package>
  </packages>
  <remove-packages config:type="list">
    <package>postfix</package>
  </remove-packages>
</software>
```



注記: パッケージ削除の失敗について

パターン (詳しくは 4.9.1項「パターンやパッケージの選択」をお読みください) の一部として設定されているパッケージについては、削除を行うことができません。このようなパッケージを削除しようとすると、下記のようなエラーメッセージを表示してインストールが失敗してしまいます:

```
パッケージ解決器の実行に失敗しました。
        AutoYaST プロファイルのソフトウエアのセクションを確認してください。
```

4.9.6 推奨されるパッケージやパターンのインストール

AutoYaST では、推奨される パッケージやパターンをインストールするかどうかを制御することができます。制御には 3 種類の選択肢が用意されています:

- 推奨される全てのパッケージやパターンをインストールする
- 必要なパッケージやパターンのみをインストールする
- 推奨されるパッケージについてはインストールを行うが、推奨されるパターンについては無視する

設定ファイル内の `install_recommended` フラグを `true` に設定することで、全ての推奨パッケージと推奨パターンをインストールするようになります。

最小限のインストールに留め、必要な パッケージとパターンのみをインストールするようにしたい場合は、このフラグを `false` にしてください。

推奨されるパッケージのみをインストールし、推奨されるパターンについては無視しておきたい場合は、設定ファイル内でフラグを指定しないでください。ただし、この設定は新規インストールの場合にのみ適用されるものであり、アップグレードの場合には無視されることに注意してください。



注記: `install_recommended` フラグの効果について

このフラグはインストール処理時のパッケージ解決器にのみ効果があるフラグであり、`/etc/zypp/zypp.conf` ファイルの修正は行わないことに注意してください。そのため、この AutoYaST 設定はインストール後に影響を与えることはありません。

```
<software>
  <install_recommended config:type="boolean">false
  </install_recommended>
</software>
```

4.9.7 第 2 ステージでのパッケージのインストール

再起動後の第 2 ステージでパッケージをインストールしたい場合は、下記のようにして `post-packages` 要素を設定します:

```
<software>
  <post-packages config:type="list">
    <package>yast2-cim</package>
  </post-packages>
</software>
```

4.9.8 第 2 ステージでのパターンのインストール

第 2 ステージでパターンをインストールすることもできます。下記のようにして `post-patterns` 要素を設定してください:

```
<software>
  <post-patterns config:type="list">
    <pattern>apparmor</pattern>
  </post-patterns>
```

```
</software>
```

4.9.9 第 2 ステージでのオンライン更新

インストールの末尾にオンライン更新を行うこともできます。オンライン更新を行いたい場合は、`do_online_update` を `true` に設定してください。なおオンライン更新の処理は、たとえば suse-register/customer-center のセクションでオンライン更新用のリポジトリを設定していたり、post-script 内でリポジトリを設定したりしている場合にのみ意味があります。第 1 ステージの add-on セクションで既にオンライン更新用のリポジトリを設定している場合は、AutoYaST は当初の段階から最新のパッケージをインストールすることになります。なお、オンライン更新でカーネルを更新した場合、第 2 ステージの終了後に再起動処理が動作します。

```
<software>
  <do_online_update config:type="boolean">true</do_online_update>
</software>
```

4.10 アップグレード

AutoYaST はシステムのアップグレードを行う際にも使用することができます。パッケージのアップグレードだけでなく、下記のような機能のセクションも提供されています:

- `scripts/pre-scripts`: 全ての処理が行われるよりも前に動作するユーザスクリプトの指定。
- `add-on`: 追加のアドオン製品の定義。
- `language`: 言語の設定。
- `timezone`: タイムゾーンの設定。
- `keyboard`: キーボードの設定。
- `software`: 追加のソフトウエアやパターンのインストール、インストール済みのパッケージの削除。
- `suse_register`: 登録処理の実行。

アップグレード処理を制御する目的で、下記のようなセクションを定義することができます:

例 4.36: アップグレードとバックアップ

```
  <upgrade>
    <stop_on_solver_conflict config:type="boolean">true</stop_on_solver_conflict>
  </upgrade>
  <backup>
```

```
    <sysconfig config:type="boolean">true</sysconfig>
    <modified config:type="boolean">true</modified>
    <remove_old config:type="boolean">true</remove_old>
  </backup>
```

stop_on_solver_conflict
: パッケージの依存関係に問題が発生した場合、インストールを停止するようにします。

modified
: 変更されたファイルのバックアップを作成します。

sysconfig
: `/etc/sysconfig` ディレクトリのバックアップを作成します。

remove_old
: 以前の更新で作成されたバックアップを削除します。

AutoYaST のアップグレードモードを開始するには、下記のようにして行います:

手順 4.1: オフラインアップグレードモードでの AUTOYAST の起動

1. 作成した AutoYaST のプロファイルを、ファイルシステム内の `/root/autoupg.xml` にコピーします。
2. インストールメディアからシステムを起動します。
3. **アップグレード** を選択します。
4. コマンドラインには `autoupgrade=1` を設定します。
5. `Enter` を押すとアップグレード処理が始まります。

手順 4.2: オンラインアップグレードモードでの AUTOYAST の起動

1. インストールメディアからシステムを起動します。
2. **アップグレード** を選択します。
3. コマンドラインには `netsetup=dhcp autoupgrade=1 autoyast=http://192.169.3.1/autoyast.xml` を設定します。
   上記の設定では、DHCP を利用してネットワークを設定します。
4. `Enter` を押すとアップグレード処理が始まります。

4.11 サービスとターゲット

`services-manager` リソースを使用することで、既定の systemd ターゲットの指定のほか、個別のシステムサービスを開始したり停止したり、開始方法を指定したりすることができます。

`default-target` プロパティでは、システム起動時の既定の systemd ターゲットを指定します。設定可能な値は `graphical` (グラフィカルなログイン画面を表示する) もしくは `multi-user` (コンソールログイン) のいずれかです。

特定のサービスに対してシステムの起動時に開始するように設定したい場合は、`enable` と `disable` のリストを設定します。特定のサービスを開始したい場合は、`enable` のリスト内にその名前を設定します。逆にサービスをシステムの起動時に開始したくない場合は、`disable` のリスト内に名前を設定します。

サービスが `enable` と `disable` のどちらにも掲載されていない場合、既定の設定 (有効もしくは無効) が使用されます。

このほか、`cups` サポートなどのようにオンデマンドで有効化できる (ソケットで開始するサービス) ものもあります。このような機能を設定したい場合は、`enable` ではなく `on_demand` のリスト内に名前を記載してください。

例 4.37: サービスとターゲットの設定

```
<services-manager>
  <default_target>multi-user</default_target>
  <services>
    <disable config:type="list">
      <service>libvirtd</service>
    </disable>
    <enable config:type="list">
      <service>sshd</service>
    </enable>
    <on_demand config:type="list">
      <service>cups</service>
    </on_demand>
  </services>
</services-manager>
```

4.12 ネットワークの設定

4.12.1 設定の概要

ネットワークの設定は主に、ワークステーションをイーサネットベースの LAN に接続する際に使用します。ダイヤルアップ接続を設定することもできます。AutoYaST では通常、AutoYaST のプロファイルをネットワーク経由で取得することから、AutoYaST を動作させるよりも前に設定する必要がある場合もあります。このような場合は、`linuxrc` で設定を行います。



注記: `linuxrc` プログラムについて

`linuxrc` の仕組みと設定キーワードについてより詳しく知りたい場合は、付録C 高度な `linuxrc` オプション をお読みください。

既定の YaST の動作では、インストール時に使用したネットワーク設定をそのまま最終的な (インストール後の) 設定値として適用します。この設定値は、AutoYaST プロファイル内に書かれたものと組み合わせた形になります。

AutoYaST での設定は、既に存在する設定ファイルよりも高い優先順位で適用されます。YaST は `ifcfg-*` ファイルに書き込みを行う際、既存の内容を削除せずに行います。そのため、DNS やルーティングのセクションを設定しない場合、YaST は既存の値をそのまま保持するようになります。それ以外の場合は、プロファイル内の設定が適用されます。

また、ネットワークの設定やサービスの有効化については、`profile networking` のグローバルリソース内で設定します。

4.12.2 ネットワークリソース

例 4.38: ネットワークの設定

```
<networking>
  <dns>
    <dhcp_hostname config:type="boolean">true</dhcp_hostname>
    <hostname>linux-bqua</hostname>
    <nameservers config:type="list">
      <nameserver>192.168.1.116</nameserver>
      <nameserver>192.168.1.117</nameserver>
      <nameserver>192.168.1.118</nameserver>
    </nameservers>
    <resolv_conf_policy>auto</resolv_conf_policy>
    <searchlist config:type="list">
      <search>example.com</search>
```

```
      <search>example.net</search>
    </searchlist>
  </dns>
  <interfaces config:type="list">
    <interface>
      <bootproto>dhcp</bootproto>
      <name>eth0</name>
      <startmode>auto</startmode>
    </interface>
  </interfaces>
  <ipv6 config:type="boolean">true</ipv6>
  <keep_install_network config:type="boolean">false</keep_install_network>
  <managed config:type="boolean">false</managed>
  <net-udev config:type="list">
    <rule>
      <name>eth0</name>
      <rule>ATTR{address}</rule>
      <value>00:30:6E:08:EC:80</value>
    </rule>
  </net-udev>
  <s390-devices config:type="list">
    <listentry>
      <chanids>0.0.0800:0.0.0801:0.0.0802</chanids>
      <type>qeth</type>
    </listentry>
  </s390-devices>
  <routing>
    <ipv4_forward config:type="boolean">false</ipv4_forward>
    <ipv6_forward config:type="boolean">false</ipv6_forward>
    <routes config:type="list">
      <route>
        <destination>192.168.100.0/24</destination>
        <device>eth1</device>
        <extrapara>scope link src 192.168.100.100 table one</extrapara>
        <gateway>-</gateway>
      </route>
      <route>
        <destination>default</destination>
        <device>eth1</device>
        <gateway>192.168.100.1</gateway>
      </route>
      <route>
        <destination>default</destination>
        <device>lo</device>
        <gateway>192.168.5.1</gateway>
      </route>
    </routes>
  </routing>
</networking>
```

上述の例のとおり、`<networking>` 内にはいくつかのサブセクションが存在しています:

- `interfaces` 内にはネットワークインターフェイスの設定を記述します。ここには IP アドレスのほか、起動方法などを指定することができます。
- `dns` 内には DNS 関連の設定を記述します。ここにはホスト名のほか、ネームサーバの一覧などを指定します。
- `routing` 内にはルーティング (経路制御) ルールを記述します。
- `s390-devices` 内には z システム固有のデバイス設定を記述します。
- `net-udev` 内には、デバイス名を固定させるための udev ルールを記述します。

これに加えて、ネットワーク設定の適用方法に関する要素がいくつか提供されています:

backend
: 使用すべきネットワークバックエンドを指定します。設定可能な値は `wicked`, `network_manager`, `none` のいずれかで、`none` を指定するとネットワークサービスを無効化します。

```
<backend>network_manager</backend>
```

keep_install_network
: 4.12.1項「設定の概要」で説明しているとおり、既定では AutoYaST は動作中のシステムに設定されている項目とプロファイル内に設定されている項目を合成して設定を生成します。プロファイル側の設定のみを使用させたい場合は、この要素を `false` にしてください。`true` が既定値です。

```
<keep_install_network config:type="boolean">false</keep_install_network>
```

managed
: Wicked ではなく NetworkManager を使用するかどうかを設定します。
廃止予定です。代わりに `backend` を使用してください。

```
<managed config:type="boolean">true</managed>
```

start_immediately
: AutoYaST に対して、設定の書き込み直後にネットワークの再起動を強制させるかどうかを指定します。

```
<start_immediately config:type="boolean">true</start_immediately>
```

setup_before_proposal

インストール時に、プロファイル内で指定されたネットワーク設定を使用するかどうかを指定します。使用しない場合、AutoYaST は `linuxrc` で指定された設定に従って動作します。

```
<setup_before_proposal config:type="boolean">true</setup_before_proposal>
```

strict_IP_check_timeout

ネットワークを設定したあと、AutoYaST は設定した IP アドレスが重複していないかどうかを確認します。重複が確認された場合、ここで指定された秒数だけ警告メッセージが表示されます。`0` を指定すると、インストールが停止します。

```
<strict_IP_check_timeout config:type="integer">5</strict_IP_check_timeout>
```

virt_bridge_proposal

AutoYaST では、インストール時に仮想化パッケージ (Xen, QEMU, KVM) が選択されていると、ブリッジの設定を行います。この動作を変更したい場合は、この要素を `false` にしてください。

```
<virt_bridge_proposal config:type="boolean">false</virt_bridge_proposal>
```



ヒント: IPv6 アドレスのサポートについて

AutoYaST では IPv6 アドレスにも完全対応しています。IPv6 アドレスのサポートを無効化したい場合は、<ipv6 config:type="boolean">false</ipv6> を設定してください。

4.12.3 インターフェイス

`interfaces` セクションにはネットワークインターフェイス関連の設定を記述します。ここにはインターフェイスの起動方法や IP アドレス、サブネットマスクなどを記述します。それぞれ下記の要素は、`<interfaces>...</interfaces>` タグ内に記述しなければなりません。

`bootproto`

インターフェイスが使用する起動プロトコルを指定します。設定可能な値は下記のとおりです:

- 固定のアドレスを設定したい場合は、`static` を指定します。`ipaddr` 要素を利用して IP アドレスを割り当てたい場合は、これを指定してください。
- DHCP を利用して IP アドレスを割り当てたい場合は、`dhcp4` (IPv4), `dhcp6` (IPv6), `dhcp` (IPv4 もしくは IPv6) のいずれかを指定します。

- IPv4 の設定を Zeroconf から取得し、IPv6 の設定を DHCP から取得したい場合は、`dhcp+autoip` を指定します。
- IPv4 の設定を Zeroconf から取得したい場合は、`autoip` を指定します。
- iBFT プロトコルを利用して IP アドレスを取得したい場合は、`ibft` を指定します。
- アドレスの設定を行いたくない場合は `none` を指定します。ブリッジやボンディングに参加する場合の指定です。

必須です。

`broadcast`
: ブロードキャスト IP アドレスを指定します。
起動プロトコルで `static` を指定する必要があります。

`device`
: デバイス名を指定します。
廃止予定です。代わりに `name` を使用してください。

`name`
: デバイス名 (例: `eth0`) を指定します。
必須です。

`lladdr`
: リンクレイヤアドレス (MAC アドレス) を指定します。
任意指定です。

`ipaddr`
: インターフェイスに割り当てる IP アドレスを指定します。
起動プロトコルで `static` を指定する必要があります。また、`192.168.1.1/24` のように、ネットワークプレフィクスを指定することもできます。

`remote_ipaddr`
: Point-to-Point 接続で使用する対向の IP アドレスを指定します。
起動プロトコルで `static` を指定する必要があります。

`netmask`
: ネットマスクを指定します (例: `255.255.255.0`) 。
廃止予定です。代わりに `prefixlen` を使用して指定するか、`ipaddr` 要素でネットワークプレフィクスを指定してください。

`network`

ネットワーク IP アドレスを指定します。
廃止予定です。`ipaddr` と `prefixlen` 要素を使用してください。

`prefixlen`

ネットワークプレフィクスを指定します (例: `24`)。
起動プロトコルで `static` を指定する必要があります。

`startmode`

インターフェイスの起動タイミングを指定します。設定可能な値は下記のとおりです:

- USB 接続のネットワークカードなど、デバイスの接続時に起動を行いたい場合は、`hotplug` を指定します。
- システムの起動時に起動を行いたい場合は `auto` を指定します。`onboot` でも同じ意味になりますが、こちらは廃止予定となっています。
- `ifplugd` デーモンで管理したい場合は、`ifplugd` を指定します。
- デバイスを手作業で起動したい場合は、`manual` を指定します。
- `/` が NFS ボリューム内に存在する場合など、ルートファイルシステムをマウントする際に特定のネットワークデバイスが必要な場合は、`nfsroot` を指定します。
- デバイスを起動したくない場合は、`off` を指定します。

`ifplugd_priority`

`ifplugd` での優先順位を指定します。この優先順位の高い順にデバイスの有効化を行います。
起動タイミングの指定で `ifplugd` を指定する必要があります。

`usercontrol`

このパラメータは現在使用していません。
廃止予定です。

`bonding_slaveX`

ボンディングデバイスの名前を指定します。
ボンディングデバイスを設定する際には必ず指定します。なお、`X` の部分には 0 から始まる数値 (例: `bonding_slave0`) を指定します。また、それぞれのポートには別々の番号を指定する必要があります。

`bonding_module_opts`

ボンディングデバイスに対するオプションを指定します。
`bond` デバイスの場合にのみ指定することができます。

`mtu`

インターフェイスに対して設定する最大伝送単位 (MTU) を指定します。
任意指定です。

`ethtool_options`

デバイスの有効化時に使用する ethtool 向けのオプションを指定します。
任意指定です。

`zone`

インターフェイスの割当先となるファイアウオールのゾーン名を指定します。
任意指定です。

`vlan_id`

この VLAN の識別 ID を指定します。
`vlan` デバイスの場合にのみ指定することができます。

`etherdevice`

VLAN の接続先デバイスを指定します。
`vlan` デバイスの場合にのみ必ず指定します。

`bridge`

インターフェイスがブリッジである場合、`yes` を指定します。
廃止予定です。この設定は他の属性から自動的に判断するようになっています。

`bridge_ports`

ブリッジで使用するネットワークデバイスをスペース区切りで指定します (例: `eth0 eth1`) 。
`bridge` デバイスの場合にのみ必ず指定します。

`bridge_stp`

スパニングツリープロトコル (STP) を有効化するかどうかを指定します。`on` を指定すると有効に、`off` を指定すると無効になります。
`bridge` デバイスの場合にのみ指定することができます。

`bridge_forward_delay`

ブリッジに対する転送遅延 (例: `15`) を指定します。
`bridge` デバイスの場合にのみ指定することができます。設定可能な値は `4` から `30` までのいずれかです。

`aliases`

追加の IP アドレスを指定します。詳しくは 4.12.4項「複数の IP アドレスの割り当て」をお読みください。

例 4.39: ボンディングインターフェイスの設定

```
<networking>
  <setup_before_proposal config:type="boolean">false</setup_before_proposal>
  <keep_install_network config:type="boolean">false</keep_install_network>
  <interfaces config:type="list">
    <interface>
      <bonding_master>yes</bonding_master>
      <bonding_module_opts>mode=active-backup miimon=100</bonding_module_opts>
      <bonding_slave0>eth1</bonding_slave0>
      <bonding_slave1>eth2</bonding_slave1>
      <bootproto>static</bootproto>
      <name>bond0</name>
      <ipaddr>192.168.1.61</ipaddr>
      <prefixlen>24</prefixlen>
      <startmode>auto</startmode>
    </interface>
    <interface>
      <bootproto>none</bootproto>
      <name>eth1</name>
      <startmode>auto</startmode>
    </interface>
    <interface>
      <bootproto>none</bootproto>
      <name>eth2</name>
      <startmode>auto</startmode>
    </interface>
  </interfaces>
  <net-udev config:type="list">
    <rule>
      <name>eth1</name>
      <rule>ATTR{address}</rule>
      <value>dc:e4:cc:27:94:c7</value>
    </rule>
    <rule>
      <name>eth2</name>
      <rule>ATTR{address}</rule>
      <value>dc:e4:cc:27:94:c8</value>
    </rule>
  </net-udev>
</networking>
```

例 4.40: ブリッジインターフェイスの設定

```
<interfaces config:type="list">
  <interface>
    <name>br0</name>
    <bootproto>static</bootproto>
    <bridge>yes</bridge>
    <bridge_forwarddelay>0</bridge_forwarddelay>
    <bridge_ports>eth0 eth1</bridge_ports>
```

```
    <bridge_stp>off</bridge_stp>
    <ipaddr>192.168.1.100</ipaddr>
    <prefixlen>24</prefixlen>
    <startmode>auto</startmode>
  </interface>
  <interface>
    <name>eth0</name>
    <bootproto>none</bootproto>
    <startmode>hotplug</startmode>
  </interface>
  <interface>
    <name>eth1</name>
    <bootproto>none</bootproto>
    <startmode>hotplug</startmode>
  </interface>
</interfaces>
```

4.12.4 複数の IP アドレスの割り当て

AutoYaST では 1 つのインターフェイスに対して複数の IP アドレスを割り当てることができます。これらは `aliases` 要素内に複数の `aliasX` 項目を追加して、それぞれに 1 つずつのアドレスを指定する形態をとります。

各項目には下記の要素を指定することができます:

IPADDR
: 追加の IP アドレスを指定します。なお、`192.168.1.1/24` のように、ネットワークプレフィクスを含めて指定することもできます。

PREFIXLEN
: ネットワークプレフィクスを指定します (例: `24`)。

NETMASK
: アドレスに対するネットマスクを指定します。
廃止予定です。代わりに `PREFIXLEN` を使用して指定するか、`IPADDR` 要素でネットワークプレフィクスを指定してください。

LABEL
: アドレスに対するラベルを指定します。



注記: 大文字と小文字の区別について

過去の経緯から、`aliases` 要素内の `IPADDR`, `PREFIXLEN`, `LABEL`, `NETMASK` の各項目については、いずれも大文字と小文字を区別することに注意してください。

例 4.41: 複数の IP アドレス

```
<interfaces config:type="list">
  <interface>
    <name>br0</name>
    <bootproto>static</bootproto>
    <ipaddr>192.168.1.100</ipaddr>
    <prefixlen>24</prefixlen>
    <startmode>auto</startmode>
    <aliases>
      <alias0>
        <IPADDR>192.168.1.101</IPADDR>
        <PREFIXLEN>24</PREFIXLEN>
        <LABEL>http</LABEL>
      </alias0>
      <alias1>
        <IPADDR>192.168.2.100</IPADDR>
        <PREFIXLEN>24</PREFIXLEN>
        <LABEL>extra</LABEL>
      </alias1>
    </aliases>
  </interface>
</interfaces>
```

4.12.5 ネットワークインターフェイス名の固定

`net-udev` 要素では、インターフェイス名を固定させるために使用する udev のルールを指定します。

name
: ネットワークインターフェイス名 (例: `eth3`) を設定します (必須)。

rule
: MAC アドレスベースのルールを作成するには `ATTR{address}` を、バス ID ベースのルールを作成するには `KERNELS` を指定します (必須)。

value
: MAC アドレスベースのルールの場合は `f0:de:f1:6b:da:69` のように、バス ID ベースのルールの場合は `0000:00:1c.1` もしくは `0.0.0700` のように指定します (必須)。

ヒント: デバイス名の衝突時の処理について

不完全な udev ルールセットを作成してしまった場合、既存のデバイス名と指定したルールのデバイス名が衝突する場合があります。たとえばネットワークインターフェイスの名前を `eth0` に変更する場合、カーネル側で自動生成されたデバイス名と衝突することがあります。AutoYaST ではこのような場合でも、デバイス名をできる限り適切に処理しようとし、必要に応じて衝突したデバイスの名前を修正します。

例 4.42: MAC アドレスを使用したデバイス名の設定例

```
<net-udev config:type="list">
  <rule>
  <name>eth1</name>
  <rule>ATTR{address}</rule>
  <value>52:54:00:68:54:fb</value>
  </rule>
</net-udev>
```

4.12.6 ドメインネームシステム (DNS)

`dns` セクションでは、ホスト名やネームサーバなど、ネームサービスに関連する設定を行うことができます。

hostname

ドメイン名部分を除くホスト名を指定します。たとえば `foo.bar` ではなく、`foo` だけを指定します。Linux カーネルおよび YaST では、ホスト名として完全修飾ドメイン名 (FQDN) を使用することができますが、YaST の dns セクションの正しい使用方法ではありません。ここでドメイン名を指定せずに、解決器が FQDN を判別すべきです (FQDN の解決方法について、詳しくは `man 1 hostname` の "THE FQDN" セクションをお読みください) 。
ここでホスト名を指定しない場合で、DHCP サーバからもホスト名を取得できない場合 (`dhcp_hostname` についてもお読みください) は、AutoYaST が `install` というホスト名を設定します。

nameservers

ネームサーバの一覧を指定します。たとえば下記のように指定します:

```
<nameservers config:type="list">
  <nameserver>192.168.1.116</nameserver>
  <nameserver>192.168.1.117</nameserver>
</nameservers>
```

searchlist
: ホスト名の参照時に使用する検索リストを指定します。

```
<searchlist config:type="list">
  <search>example.com</search>
</searchlist>
```

: 任意指定です。

dhcp_hostname
: ホスト名を DHCP 経由で取得するかどうかを指定します。

```
<dhcp_hostname config:type="boolean">true</dhcp_hostname>
```

4.12.7 ルーティング

`routing` テーブルには経路情報の一覧を設定することができるほか、IPv4 や IPv6 での中継機能を有効化するかどうかを指定することができます。

ipv4_forward
: 任意指定: IPv4 での IP 中継機能を有効化するかどうかを指定します。

ipv6_forward
: 任意指定: IPv6 での IP 中継機能を有効化するかどうかを指定します。

routes
: 任意指定: 経路の一覧を記述します。

経路情報の設定方法を下記の表に示します。

destination
: 必須: 経路の宛先を指定します。`192.168.122.0/24` のように、プレフィクス付きのアドレスを指定することができます。
値として `default` を指定すると、同一アドレスファミリ (IPv4 もしくは IPv6) 内でのデフォルトゲートウエイを指定することができます。

device
: 必須: その経路に行き着くためのインターフェイスを指定します。

gateway
: 任意指定: ゲートウエイの IP アドレスを指定します。

netmask

(廃止予定) 経路の宛先のネットマスクを指定します。

`destination` でネットワークプレフィクスを指定するようにしてください。

extrapara

任意指定: `metric` , `mtu` , `table` のような追加のルーティングオプションを指定します。

例 4.43: ネットワークルーティングの設定

```
<routing>
  <ipv4_forward config:type="boolean">true</ipv4_forward>
  <ipv6_forward config:type="boolean">true</ipv6_forward>
  <routes config:type="list">
    <route>
      <destination>192.168.100.0/24</destination>
      <device>eth1</device>
      <extrapara>scope link src 192.168.100.100 table one</extrapara>
    </route>
    <route>
      <destination>default</destination>
      <device>eth1</device>
      <gateway>192.168.100.1</gateway>
    </route>
    <route>
      <destination>default</destination>
      <device>lo</device>
      <gateway>192.168.5.1</gateway>
    </route>
  </routes>
</routing>
```

4.12.8 s390 オプション

下記の要素は < `s390-devices` >... </ `s390-devices` > タグ内に記述しなければなりません。

type

`qeth` , `ctc` , `iucv` のいずれかを指定します。

chanids

コロン (推奨) もしくはスペース区切りでのチャンネル ID

```
<chanids>0.0.0700:0.0.0701:0.0.0702</chanids>
```

layer2

```
<layer2 config:type="boolean">true</layer2>
```

ブール値。既定は false

protocol
: 任意指定: CTC / LCS プロトコル (文字列型で小さな値)

```
<protocol>1</protocol>
```

router
: IUCV ルータ／ユーザ

上述のオプションに加えて、AutoYaST では設定ファイルの他のセクション内に IBM Z 固有のオプションを設定することもできます。特に論理リンクアドレス (LLADDR (イーサネットの場合 MAC アドレス)) を設定することもできます。設定を行うには、デバイスの定義内に `LLADDR` オプションを設定してください。



ヒント: VLAN 向けの LLADDR について

VLAN デバイスの場合、接続されている物理デバイスの LLADDR をそのまま継承します。VLAN デバイスに対して別のアドレスを設定するには、物理デバイス側に LLADDR のオプションを設定してください。

4.13 プロキシ

インターネットプロキシ (キャッシュサーバ) に関する設定を行います。

HTTP, HTTPS, FTP に対するそれぞれのプロキシサーバを、`http_proxy`, `https_proxy`, `ftp_proxy` で設定します。また、プロキシを経由せずに直接アクセスすべきアドレスやホストについては、`no_proxy` 内にスペース区切りで設定します。プロキシサーバに対して認証が必要な場合は、`proxy_user` (ユーザ名) および `proxy_password` (パスワード) にそれぞれ設定します。

例 4.44: ネットワーク設定: プロキシ

```
<proxy>
  <enabled config:type="boolean">true</enabled>
  <ftp_proxy>http://192.168.1.240:3128</ftp_proxy>
  <http_proxy>http://192.168.1.240:3128</http_proxy>
  <no_proxy>www.example.com .example.org localhost</no_proxy>
  <proxy_password>testpw</proxy_password>
  <proxy_user>testuser</proxy_user>
</proxy>
```



注記

プロキシ設定はインストール時のほか、提案を表示する前にネットワーク設定が必要となった場合や、`linuxrc` でプロキシ設定が指定されている場合にそれぞれ書き込みが行われます。

4.14 NIS クライアントおよびサーバ

`nis` リソースを使用することで、インストール先のマシンを NIS クライアントとして設定することができます。下記の例では、複数のドメインを利用した詳細な設定を行っています。

例 4.45: ネットワーク設定: NIS

```
<nis>
 <nis_broadcast config:type="boolean">true</nis_broadcast>
 <nis_broken_server config:type="boolean">true</nis_broken_server>
 <nis_domain>test.com</nis_domain>
 <nis_local_only config:type="boolean">true</nis_local_only>
 <nis_options></nis_options>
 <nis_other_domains config:type="list">
   <nis_other_domain>
     <nis_broadcast config:type="boolean">false</nis_broadcast>
     <nis_domain>domain.com</nis_domain>
     <nis_servers config:type="list">
       <nis_server>10.10.0.1</nis_server>
     </nis_servers>
   </nis_other_domain>
 </nis_other_domains>
 <nis_servers config:type="list">
   <nis_server>192.168.1.1</nis_server>
 </nis_servers>
 <start_autofs config:type="boolean">true</start_autofs>
 <start_nis config:type="boolean">true</start_nis>
</nis>
```

4.15 NIS サーバ

インストール先のマシンを NIS サーバとして構成することもできます。NIS マスターサーバ、NIS ワーカーサーバ、NIS マスター兼ワーカーサーバをそれぞれ設定することができます。

例 4.46: NIS サーバの設定

```
  <nis_server>
```

```
    <domain>mydomain.de</domain>
    <maps_to_serve config:type="list">
      <nis_map>auto.master</nis_map>
      <nis_map>ethers</nis_map>
    </maps_to_serve>
    <merge_passwd config:type="boolean">false</merge_passwd>
    <mingid config:type="integer">0</mingid>
    <minuid config:type="integer">0</minuid>
    <nopush config:type="boolean">false</nopush>
    <pwd_chfn config:type="boolean">false</pwd_chfn>
    <pwd_chsh config:type="boolean">false</pwd_chsh>
    <pwd_srcdir>/etc</pwd_srcdir>
    <securenets config:type="list">
      <securenet>
        <netmask>255.0.0.0</netmask>
        <network>127.0.0.0</network>
      </securenet>
    </securenets>
    <server_type>master</server_type>
    <slaves config:type="list"/>
    <start_ypbind config:type="boolean">false</start_ypbind>
    <start_yppasswdd config:type="boolean">false</start_yppasswdd>
    <start_ypxfrd config:type="boolean">false</start_ypxfrd>
  </nis_server>
```

属性／値／説明

`domain`
: NIS ドメイン名。

`maps_to_serve`
: サーバ側で利用できるようにするマップのリストを指定します。
値: auto.master, ethers, group, hosts, netgrp, networks, passwd, protocols, rpc, services, shadow

`merge_passwd`
: passwd ファイルと shadow ファイルの合成を行うかどうかを指定します (shadow ファイルが存在する場合にのみ行うことができます) 。
値: true (有効)/false (無効)

`mingid`
: ユーザマップ内での最小 GID を指定します。

`minuid`
: ユーザマップ内での最小 UID を指定します。

`nopush`

変更点をワーカーサーバにプッシュしないようにします (ワーカーサーバが存在しない場合に有用です) 。
値: true (有効)/false (無効)

`pwd_chfn`

YPPWD_CHFN - フルネームの変更を許可します
値: true (有効)/false (無効)

`pwd_chsh`

YPPWD_CHSH - ログインシェルの変更を許可します
値: true (有効)/false (無効)

`pwd_srcdir`

YPPWD_SRCDIR - passwd データのソースディレクトリ
既定値: `/etc`

`securenets`

NIS サーバに対して問い合わせることのできるホストの一覧
network のアドレスが、対向のアドレスと netmask の論理積のアドレスが一致した場合にのみ、アクセスが許可されます。
network 127.0.0.0 および netmask 255.0.0.0 の設定は、ローカルからの接続を許可するために必ず存在していなければなりません。
network 0.0.0.0, netmask 0.0.0.0 を指定すると、全てのホストからのアクセスを許可します。

`server_type`

NIS サーバをマスターサーバ／ワーカーサーバのどちらにするのか、もしくは NIS サーバを設定しないのかを選択します。
値: master, slave, none

`slaves`

NIS ワーカーサーバとして設定したいホスト名のリストを指定します。

`start_ypbind`

このホストを NIS クライアントとしても設定します (ただしクライアント側の設定をローカルで行う必要があります) 。
値: true (有効)/false (無効)

`start_yppasswdd`

パスワードデーモンも起動するようにします。
値: true (有効)/false (無効)

`start_ypxfrd`

マップ転送デーモンも起動するようにします。高速なマップ配布の仕組みにより、ワーカーに対するマップの転送も高速化されます。

値: true (有効)/false (無効)

4.16 ホスト名の定義

`host` リソースを使用することで、`/etc/hosts` 内に 1 つもしくは複数の項目を追加することができます。既に存在する項目については、削除されることはありません。下記に詳細例を示します。

例 4.47: /ETC/HOSTS

```
<host>
 <hosts config:type="list">
  <hosts_entry>
   <host_address>133.3.0.1</host_address>
   <names config:type="list">
    <name>booking</name>
   </names>
  </hosts_entry>
  <hosts_entry>
   <host_address>133.3.0.5</host_address>
   <names config:type="list">
    <name>test-machine</name>
   </names>
  </hosts_entry>
 </hosts>
</host>
```

4.17 Windows ドメインのメンバーシップ

`samba-client` リソースを使用することで、ワークグループや NT ドメイン、もしくは Active Directory ドメインのメンバーとして構成することができます。

例 4.48: SAMBA クライアント設定

```
<samba-client>
  <disable_dhcp_hostname config:type="boolean">true</disable_dhcp_hostname>
  <global>
    <security>domain</security>
    <usershare_allow_guests>No</usershare_allow_guests>
    <usershare_max_shares>100</usershare_max_shares>
    <workgroup>WORKGROUP</workgroup>
```

```
    </global>
    <winbind config:type="boolean">false</winbind>
  </samba-client>
```

属性／値／説明

`disable_dhcp_hostname`
: DHCP に対してホスト名の変更を許可しないようにします。
値: true (有効)/false (無効)

`global/security`
: 認証方式を設定します (NT ドメイン (domain) もしくは Active Directory サーバ (ADS)) 。
値: ADS/domain

`global/usershare_allow_guests`
: ゲストアクセスの共有を許可するかどうかを指定します。
値: No (いいえ)/Yes (はい)

`global/usershare_max_shares`
: `smb.conf` での最大共有数を指定します。
0 を指定すると、共有を無効化する意味になります。

`global/workgroup`
: ワークグループもしくはドメイン名を指定します。

`winbind`
: winbind を使用するかどうかを指定します。
値: true (有効)/false (無効)

4.18 Samba サーバ

シンプルな Samba サーバの設定を行うことができます。

例 4.49: SAMBA サーバ設定

```
  <samba-server>
    <accounts config:type="list"/>
    <backend/>
    <config config:type="list">
      <listentry>
        <name>global</name>
        <parameters>
          <security>domain</security>
```

```
          <usershare_allow_guests>No</usershare_allow_guests>
          <usershare_max_shares>100</usershare_max_shares>
          <workgroup>WORKGROUP</workgroup>
        </parameters>
      </listentry>
    </config>
    <service>Disabled</service>
    <trustdom/>
    <version>2.11</version>
  </samba-server>
```

属性／値／説明

accounts
: Samba アカウントのリスト。

backend
: 利用可能なバックエンドのリスト。
値: `true` / `false`

config
: `/etc/samba/smb.conf` 内に設定する追加のユーザ定義パラメータを指定します。
詳しくは `/etc/samba/smb.conf` ファイル内の `global` パラメータをお読みください。

service
: Samba サービスを起動時に開始するかどうかを指定します。
値: `Enabled` (有効) / `Disabled` (無効)

trustdom/
: 信頼するドメインを指定します。
2 つのマップを指定します (キー: `establish, revoke`) 。各マップには `key: domainname value: password` の形式で項目を設定します。

version
: Samba のバージョンを指定します。
既定値: 2.11

4.19 認証クライアント

設定ファイルは JSON 形式でなければなりません。まずは `autoyast2` と `autoyast2-installation` の各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。あとは YaST 内の［自動インストールの設定］で、必要な JSON 形式の設定ファイルを作成してください。YaST を起動

し、［その他］›［自動インストールの設定］を選択し、［ネットワークサービス］›［ユーザのログイン管理］を選択して［編集］を押し、設定作業を行ってください。設定が終わったら［OK］を押して閉じます。生成された設定ファイルを保存するには、［ファイル］›［保存］を選択します。



ヒント: ldaps:// の使用について

ネイティブ SSL (TLS ではなく) で LDAP を使用するには、`ldaps` リソースを追加します。

4.20 NFS クライアントおよびサーバ

設定システムを使用することで、NFS クライアントも NFS サーバも設定することができます。下記の例では、NFS クライアントと NFS サーバの両方を設定する場合の例を示しています。

openSUSE Leap 15.7 およびそれ以降では、NFS クライアントの設定構造が変更されています。それぞれ下記に示すグローバル設定オプションが追加されています: `enable_nfs4` (NFS4 サポートの有効化／無効化), `idmapd_domain` (rpc.idmapd 向けのドメイン名設定 (NFS4 が有効化されている場合にのみ効果があります)) 。注意: 古い構造は新しい構造とは互換性が無く、古いリリースで作成された NFS セクションを含む制御ファイルは、新しい製品では動作しません。

openSUSE Leap を NFS 共有にインストールする方法については、4.5.10項「NFS の設定」をお読みください。

例 4.50: ネットワーク設定: NFS クライアント

```
<nfs>
  <enable_nfs4 config:type="boolean">true</enable_nfs4>
  <idmapd_domain>suse.cz</idmapd_domain>
  <nfs_entries config:type="list">
    <nfs_entry>
      <mount_point>/home</mount_point>
      <nfs_options>sec=krb5i,intr,rw</nfs_options>
      <server_path>saurus.suse.cz:/home</server_path>
      <vfstype>nfs4</vfstype>
    </nfs_entry>
    <nfs_entry>
      <mount_point>/work</mount_point>
      <nfs_options>defaults</nfs_options>
      <server_path>bivoj.suse.cz:/work</server_path>
      <vfstype>nfs</vfstype>
    </nfs_entry>
    <nfs_entry>
      <mount_point>/mnt</mount_point>
      <nfs_options>defaults</nfs_options>
      <server_path>fallback.suse.cz:/srv/dist</server_path>
      <vfstype>nfs</vfstype>
```

```
    </nfs_entry>
  </nfs_entries>
</nfs>
```

例 4.51: ネットワーク設定: NFS サーバ

```
<nfs_server>
  <nfs_exports config:type="list">
    <nfs_export>
      <allowed config:type="list">
        <allowed_clients>*(ro,root_squash,sync)</allowed_clients>
      </allowed>
      <mountpoint>/home</mountpoint>
    </nfs_export>
    <nfs_export>
      <allowed config:type="list">
        <allowed_clients>*(ro,root_squash,sync)</allowed_clients>
      </allowed>
      <mountpoint>/work</mountpoint>
    </nfs_export>
  </nfs_exports>
  <start_nfsserver config:type="boolean">true</start_nfsserver>
</nfs_server>
```

4.21 NTP クライアント

重要: NTP クライアント設定の非互換性について

openSUSE Leap 15 およびそれ以降のバージョンでは、NTP クライアントの設定が新しい形式となり、以前のプロファイルとは互換性が なくなっています 。以前の openSUSE Leap バージョンで使用していたプロファイル内に NTP クライアントの設定が含まれている場合は、バージョン 15 もしくはそれ以降に対応するために更新を行う必要があります。

下記に NTP クライアントの設定例を示します:

例 4.52: ネットワーク設定: NTP クライアント

```
<ntp-client>
  <ntp_policy>auto</ntp_policy>❶
  <ntp_servers config:type="list">
    <ntp_server>
      <address>cz.pool.ntp.org</address>❷
      <iburst config:type="boolean">false</iburst>❸
      <offline config:type="boolean">false</offline>❹
    </ntp_server>
  </ntp_servers>
```

```
  <ntp_sync>15</ntp_sync> ❺
</ntp-client>
```

❶ `ntp_policy` には `/etc/sysconfig/network/config` 内の `NETCONFIG_NTP_POLICY` と同じ値を設定することができます。最もよく使用する値は 'static' と 'auto' (既定値) です。詳しくは `man 8 netconfig` をお読みください。

❷ タイムサーバもしくはタイムサーバのプールの URL を指定します。

❸ `iburst` を指定すると、`chronyd` の起動直後の初期時刻同期を高速化することができます。

❹ `offline` オプションを `true` に設定すると、`chronyd` の起動時にタイムサーバが利用できなかった場合に、問い合わせを行わないようになります。問い合わせは `chronyc online` コマンドでオンラインに戻すまで行われなくなります。ただし、このコマンドはシステムを再起動すると効果が失われることに注意してください。システム管理者側での作業を行うことなく常にタイムサーバに対して問い合わせを行いたい場合は、`false` を指定してください。

❺ `ntp_sync` に対しては、NTP デーモンを起動する場合は 'systemd' (既定値) を、cron を利用して定期的に同期する場合は 整数値 を、自動的な同期を行わない場合は 'manual' をそれぞれ指定します。

下記の例は IPv6 アドレスを使用する場合の例を示しています。ここではサーバの IP アドレスもしくはホスト名のほか、必要であればその両方を指定することもできます:

```
<ntp-server>
  <address>2001:418:3ff::1:53</address>
</ntp-server>

<ntp-server>
  <address>2.pool.ntp.org</address>
</ntp-server>
```

4.22 メールサーバの設定

メールのクライアント側の設定を行うには、このモジュールで詳細な設定を行います。このモジュールにはさまざまなオプションが含まれていますが、あくまでも初期設定として使用することをお勧めします。

例 4.53: メールの設定

```
<mail>
  <aliases config:type="list">
    <alias>
      <alias>root</alias>
      <comment></comment>
```

```
      <destinations>foo</destinations>
    </alias>
    <alias>
      <alias>test</alias>
      <comment></comment>
      <destinations>foo</destinations>
    </alias>
  </aliases>
  <connection_type config:type="symbol">permanent</connection_type>
  <fetchmail config:type="list">
    <fetchmail_entry>
      <local_user>foo</local_user>
      <password>bar</password>
      <protocol>POP3</protocol>
      <remote_user>foo</remote_user>
      <server>pop.foo.com</server>
    </fetchmail_entry>
    <fetchmail_entry>
      <local_user>test</local_user>
      <password>bar</password>
      <protocol>IMAP</protocol>
      <remote_user>test</remote_user>
      <server>blah.com</server>
    </fetchmail_entry>
  </fetchmail>
  <from_header>test.com</from_header>
  <listen_remote config:type="boolean">true</listen_remote>
  <local_domains config:type="list">
    <domains>test1.com</domains>
  </local_domains>
  <masquerade_other_domains config:type="list">
      <domain>blah.com</domain>
  </masquerade_other_domains>
  <masquerade_users config:type="list">
    <masquerade_user>
      <address>joe@test.com</address>
      <comment></comment>
      <user>joeuser</user>
    </masquerade_user>
    <masquerade_user>
      <address>bar@test.com</address>
      <comment></comment>
      <user>foo</user>
    </masquerade_user>
  </masquerade_users>
  <mta config:type="symbol">postfix</mta>
  <outgoing_mail_server>test.com</outgoing_mail_server>
  <postfix_mda config:type="symbol">local</postfix_mda>
  <smtp_auth config:type="list">
    <listentry>
      <password>bar</password>
```

```
      <server>test.com</server>
      <user>foo</user>
    </listentry>
  </smtp_auth>
  <use_amavis config:type="boolean">true</use_amavis>
  <virtual_users config:type="list">
    <virtual_user>
      <alias>test.com</alias>
      <comment></comment>
      <destinations>foo.com</destinations>
    </virtual_user>
    <virtual_user>
      <alias>geek.com</alias>
      <comment></comment>
      <destinations>bar.com</destinations>
    </virtual_user>
  </virtual_users>
</mail>
```

4.23 Apache HTTP サーバの設定

このセクションは、Apache HTTP サーバを設定するために使用します。

知識のあるユーザである場合を除き、まずは YaST の **HTTP サーバ** モジュールを利用して Apache サーバを設定しておくことをお勧めします。設定後、**AutoYaST 設定** モジュールを呼び出して、**HTTP サーバ** YaST モジュールを選択し、Apache の設定を複製するとよいでしょう。これらの設定は、**ファイル** メニューを利用することでエクスポートすることができます。

例 4.54: HTTP サーバの設定

```
  <http-server>
    <Listen config:type="list">
      <listentry>
        <ADDRESS/>
        <PORT>80</PORT>
      </listentry>
    </Listen>
    <hosts config:type="list">
      <hosts_entry>
        <KEY>main</KEY>
        <VALUE config:type="list">
          <listentry>
            <KEY>DocumentRoot</KEY>
            <OVERHEAD>
            #
            # Global configuration that will be applicable for all
            # virtual hosts, unless deleted here or overridden elsewhere.
            #
```

```
      </OVERHEAD>
      <VALUE>"/srv/www/htdocs"</VALUE>
    </listentry>
    <listentry>
      <KEY>_SECTION</KEY>
      <OVERHEAD>
      #
      # Configure the DocumentRoot
      #
      </OVERHEAD>
      <SECTIONNAME>Directory</SECTIONNAME>
      <SECTIONPARAM>"/srv/www/htdocs"</SECTIONPARAM>
      <VALUE config:type="list">
        <listentry>
          <KEY>Options</KEY>
          <OVERHEAD>
          # Possible values for the Options directive are "None", "All",
          # or any combination of:
          #   Indexes Includes FollowSymLinks SymLinksifOwnerMatch
          #   ExecCGI MultiViews
          #
          # Note that "MultiViews" must be named *explicitly*
          # --- "Options All"
          # does not give it to you.
          #
          # The Options directive is both complicated and important.
          #  Please see
          #  http://httpd.apache.org/docs/2.4/mod/core.html#options
          # for more information.
          </OVERHEAD>
          <VALUE>None</VALUE>
        </listentry>
        <listentry>
          <KEY>AllowOverride</KEY>
          <OVERHEAD>
          # AllowOverride controls what directives may be placed in
          # .htaccess files. It can be "All", "None", or any combination
          # of the keywords:
          #   Options FileInfo AuthConfig Limit
          </OVERHEAD>
          <VALUE>None</VALUE>
        </listentry>
        <listentry>
          <KEY>_SECTION</KEY>
          <OVERHEAD>
          # Controls who can get stuff from this server.
          </OVERHEAD>
          <SECTIONNAME>IfModule</SECTIONNAME>
          <SECTIONPARAM>!mod_access_compat.c</SECTIONPARAM>
          <VALUE config:type="list">
            <listentry>
```

```
                <KEY>Require</KEY>
                <VALUE>all granted</VALUE>
              </listentry>
            </VALUE>
          </listentry>
          <listentry>
            <KEY>_SECTION</KEY>
            <SECTIONNAME>IfModule</SECTIONNAME>
            <SECTIONPARAM>mod_access_compat.c</SECTIONPARAM>
            <VALUE config:type="list">
              <listentry>
                <KEY>Order</KEY>
                <VALUE>allow,deny</VALUE>
              </listentry>
              <listentry>
                <KEY>Allow</KEY>
                <VALUE>from all</VALUE>
              </listentry>
            </VALUE>
          </listentry>
        </VALUE>
      </listentry>
      <listentry>
        <KEY>Alias</KEY>
        <OVERHEAD>
        # Aliases: aliases can be added as needed (with no limit).
        # The format is Alias fakename realname
        #
        # Note that if you include a trailing / on fakename then the
        # server will require it to be present in the URL.  So "/icons"
        # is not aliased in this example, only "/icons/".  If the fakename
        # is slash-terminated, then the realname must also be slash
        # terminated, and if the fakename omits the trailing slash, the
        # realname must also omit it.
        # We include the /icons/ alias for FancyIndexed directory listings.
        # If you do not use FancyIndexing, you may comment this out.
        #
        </OVERHEAD>
        <VALUE>/icons/ "/usr/share/apache2/icons/"</VALUE>
      </listentry>
      <listentry>
        <KEY>_SECTION</KEY>
        <OVERHEAD>
        </OVERHEAD>
        <SECTIONNAME>Directory</SECTIONNAME>
        <SECTIONPARAM>"/usr/share/apache2/icons"</SECTIONPARAM>
        <VALUE config:type="list">
          <listentry>
            <KEY>Options</KEY>
            <VALUE>Indexes MultiViews</VALUE>
          </listentry>
```

```
            <listentry>
              <KEY>AllowOverride</KEY>
              <VALUE>None</VALUE>
            </listentry>
            <listentry>
              <KEY>_SECTION</KEY>
              <SECTIONNAME>IfModule</SECTIONNAME>
              <SECTIONPARAM>!mod_access_compat.c</SECTIONPARAM>
              <VALUE config:type="list">
                <listentry>
                  <KEY>Require</KEY>
                  <VALUE>all granted</VALUE>
                </listentry>
              </VALUE>
            </listentry>
            <listentry>
              <KEY>_SECTION</KEY>
              <SECTIONNAME>IfModule</SECTIONNAME>
              <SECTIONPARAM>mod_access_compat.c</SECTIONPARAM>
              <VALUE config:type="list">
                <listentry>
                  <KEY>Order</KEY>
                  <VALUE>allow,deny</VALUE>
                </listentry>
                <listentry>
                  <KEY>Allow</KEY>
                  <VALUE>from all</VALUE>
                </listentry>
              </VALUE>
            </listentry>
          </VALUE>
        </listentry>
        <listentry>
          <KEY>ScriptAlias</KEY>
          <OVERHEAD>
          # ScriptAlias: This controls which directories contain server
          # scripts. ScriptAliases are essentially the same as Aliases,
          # except that documents in the realname directory are treated
          # as applications and run by the server when requested rather
          # than as documents sent to the client.
          # The same rules about trailing "/" apply to ScriptAlias
          # directives as to Alias.
          #
          </OVERHEAD>
          <VALUE>/cgi-bin/ "/srv/www/cgi-bin/"</VALUE>
        </listentry>
        <listentry>
          <KEY>_SECTION</KEY>
          <OVERHEAD>
          # "/srv/www/cgi-bin" should be changed to wherever your
          # ScriptAliased CGI directory exists, if you have that configured.
```

```
            #
            </OVERHEAD>
            <SECTIONNAME>Directory</SECTIONNAME>
            <SECTIONPARAM>"/srv/www/cgi-bin"</SECTIONPARAM>
            <VALUE config:type="list">
              <listentry>
                <KEY>AllowOverride</KEY>
                <VALUE>None</VALUE>
              </listentry>
              <listentry>
                <KEY>Options</KEY>
                <VALUE>+ExecCGI -Includes</VALUE>
              </listentry>
              <listentry>
                <KEY>_SECTION</KEY>
                <SECTIONNAME>IfModule</SECTIONNAME>
                <SECTIONPARAM>!mod_access_compat.c</SECTIONPARAM>
                <VALUE config:type="list">
                  <listentry>
                    <KEY>Require</KEY>
                    <VALUE>all granted</VALUE>
                  </listentry>
                </VALUE>
              </listentry>
              <listentry>
                <KEY>_SECTION</KEY>
                <SECTIONNAME>IfModule</SECTIONNAME>
                <SECTIONPARAM>mod_access_compat.c</SECTIONPARAM>
                <VALUE config:type="list">
                  <listentry>
                    <KEY>Order</KEY>
                    <VALUE>allow,deny</VALUE>
                  </listentry>
                  <listentry>
                    <KEY>Allow</KEY>
                    <VALUE>from all</VALUE>
                  </listentry>
                </VALUE>
              </listentry>
            </VALUE>
          </listentry>
          <listentry>
            <KEY>_SECTION</KEY>
            <OVERHEAD>
            # UserDir: The name of the directory that is appended onto a
            # user's home directory if a ~user request is received.
            # To disable it, simply remove userdir from the list of modules
            # in APACHE_MODULES in /etc/sysconfig/apache2.
            #
            </OVERHEAD>
            <SECTIONNAME>IfModule</SECTIONNAME>
```

```
      <SECTIONPARAM>mod_userdir.c</SECTIONPARAM>
      <VALUE config:type="list">
        <listentry>
          <KEY>UserDir</KEY>
          <OVERHEAD>
          # Note that the name of the user directory ("public_html")
          # cannot simply be changed here, since it is a compile time
          # setting. The apache package would need to be rebuilt.
          # You could work around by deleting /usr/sbin/suexec, but
          # then all scripts from the directories would be executed
          # with the UID of the webserver.
          </OVERHEAD>
          <VALUE>public_html</VALUE>
        </listentry>
        <listentry>
          <KEY>Include</KEY>
          <OVERHEAD>
          # The actual configuration of the directory is in
          # /etc/apache2/mod_userdir.conf.
          </OVERHEAD>
          <VALUE>/etc/apache2/mod_userdir.conf</VALUE>
        </listentry>
      </VALUE>
    </listentry>
    <listentry>
      <KEY>IncludeOptional</KEY>
      <OVERHEAD>
      # Include all *.conf files from /etc/apache2/conf.d/.
      #
      # This is mostly meant as a place for other RPM packages to drop
      # in their configuration snippet.
      #
      #
      # You can comment this out here if you want those bits include
      # only in a certain virtual host, but not here.
      </OVERHEAD>
      <VALUE>/etc/apache2/conf.d/*.conf</VALUE>
    </listentry>
    <listentry>
      <KEY>IncludeOptional</KEY>
      <OVERHEAD>
      # The manual... if it is installed ('?' means it will not complain)
      </OVERHEAD>
      <VALUE>/etc/apache2/conf.d/apache2-manual?conf</VALUE>
    </listentry>
    <listentry>
      <KEY>ServerName</KEY>
      <VALUE>linux-wtyj</VALUE>
    </listentry>
    <listentry>
      <KEY>ServerAdmin</KEY>
```

```
            <OVERHEAD>
            </OVERHEAD>
            <VALUE>root@linux-wtyj</VALUE>
          </listentry>
          <listentry>
            <KEY>NameVirtualHost</KEY>
            <VALUE>192.168.43.2</VALUE>
          </listentry>
        </VALUE>
      </hosts_entry>
      <hosts_entry>
        <KEY>192.168.43.2/secondserver.suse.de</KEY>
        <VALUE config:type="list">
          <listentry>
            <KEY>DocumentRoot</KEY>
            <VALUE>/srv/www/htdocs</VALUE>
          </listentry>
          <listentry>
            <KEY>ServerName</KEY>
            <VALUE>secondserver.suse.de</VALUE>
          </listentry>
          <listentry>
            <KEY>ServerAdmin</KEY>
            <VALUE>second_server@suse.de</VALUE>
          </listentry>
          <listentry>
            <KEY>_SECTION</KEY>
            <SECTIONNAME>Directory</SECTIONNAME>
            <SECTIONPARAM>/srv/www/htdocs</SECTIONPARAM>
            <VALUE config:type="list">
              <listentry>
                <KEY>AllowOverride</KEY>
                <VALUE>None</VALUE>
              </listentry>
              <listentry>
                <KEY>Require</KEY>
                <VALUE>all granted</VALUE>
              </listentry>
            </VALUE>
          </listentry>
        </VALUE>
      </hosts_entry>
    </hosts>
    <modules config:type="list">
      <module_entry>
        <change>enable</change>
        <name>socache_shmcb</name>
        <userdefined config:type="boolean">true</userdefined>
      </module_entry>
      <module_entry>
        <change>enable</change>
```

```
        <name>reqtimeout</name>
        <userdefined config:type="boolean">true</userdefined>
      </module_entry>
      <module_entry>
        <change>enable</change>
        <name>authn_core</name>
        <userdefined config:type="boolean">true</userdefined>
      </module_entry>
      <module_entry>
        <change>enable</change>
        <name>authz_core</name>
        <userdefined config:type="boolean">true</userdefined>
      </module_entry>
    </modules>
    <service config:type="boolean">true</service>
    <version>2.9</version>
  </http-server>
```

リスト名／要素／説明

Listen
: ホストの `Listen` 設定のリストを指定します。

  PORT
  : ポート番号。

  ADDRESS
  : ネットワークアドレス。何も指定しない場合、全てのアドレスを指定したものとみなされます。

hosts
: 仮想ホスト設定のリストを指定します。

  KEY
  : ホスト名を指定します。`<KEY>main</KEY>` を指定すると、メインの仮想ホストの意味になります。例: `<KEY>192.168.43.2/secondserver.suse.de</KEY>`

  VALUE
  : そのホストに設定するさまざまな値を指定します。

modules
: モジュールのリストです。ユーザ側で必要なモジュールのみを指定します。

  name
  : モジュール名

  userdefined
  : 過去のバージョンの経緯から、この値は常に `true` に設定します。

change
過去のバージョンの経緯から、この値は常に `enable` に設定します。

要素／説明／コメント

version
使用する Apache サーバのバージョン
情報提供の意味で設定するだけです。既定値: 2.9

service
Apache サービスの有効化
任意指定です。既定値: false



注記: ファイアウオールについて

Apache サーバを正しく動作させるには、あわせてファイアウオールも設定する必要があります。

4.24 squid サーバ

squid はキャッシュ機能付きの Web プロキシサービスです。

例 4.55: SQUID サーバの設定

```
<squid>
  <acls config:type="list">
    <listentry>
      <name>QUERY</name>
      <options config:type="list">
        <option>cgi-bin \?</option>
      </options>
      <type>urlpath_regex</type>
    </listentry>
    <listentry>
      <name>apache</name>
      <options config:type="list">
        <option>Server</option>
        <option>^Apache</option>
      </options>
      <type>rep_header</type>
    </listentry>
    <listentry>
      <name>all</name>
      <options config:type="list">
        <option>0.0.0.0/0.0.0.0</option>
      </options>
```

```
        <type>src</type>
      </listentry>
      <listentry>
        <name>manager</name>
        <options config:type="list">
          <option>cache_object</option>
        </options>
        <type>proto</type>
      </listentry>
      <listentry>
        <name>localhost</name>
        <options config:type="list">
          <option>127.0.0.1/255.255.255.255</option>
        </options>
        <type>src</type>
      </listentry>
      <listentry>
        <name>to_localhost</name>
        <options config:type="list">
          <option>127.0.0.0/8</option>
        </options>
        <type>dst</type>
      </listentry>
      <listentry>
        <name>SSL_ports</name>
        <options config:type="list">
          <option>443</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>80</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>21</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>443</option>
        </options>
        <type>port</type>
      </listentry>
```

```
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>70</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>210</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>1025-65535</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>280</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>488</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>591</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>Safe_ports</name>
        <options config:type="list">
          <option>777</option>
        </options>
        <type>port</type>
      </listentry>
      <listentry>
        <name>CONNECT</name>
```

```
        <options config:type="list">
          <option>CONNECT</option>
        </options>
        <type>method</type>
      </listentry>
    </acls>
    <http_accesses config:type="list">
      <listentry>
        <acl config:type="list">
          <listentry>manager</listentry>
          <listentry>localhost</listentry>
        </acl>
        <allow config:type="boolean">true</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>manager</listentry>
        </acl>
        <allow config:type="boolean">false</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>!Safe_ports</listentry>
        </acl>
        <allow config:type="boolean">false</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>CONNECT</listentry>
          <listentry>!SSL_ports</listentry>
        </acl>
        <allow config:type="boolean">false</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>localhost</listentry>
        </acl>
        <allow config:type="boolean">true</allow>
      </listentry>
      <listentry>
        <acl config:type="list">
          <listentry>all</listentry>
        </acl>
        <allow config:type="boolean">false</allow>
      </listentry>
    </http_accesses>
    <http_ports config:type="list">
      <listentry>
        <host/>
        <port>3128</port>
        <transparent config:type="boolean">false</transparent>
```

```
      </listentry>
    </http_ports>
    <refresh_patterns config:type="list">
      <listentry>
        <case_sensitive config:type="boolean">true</case_sensitive>
        <max>10080</max>
        <min>1440</min>
        <percent>20</percent>
        <regexp>^ftp:</regexp>
      </listentry>
      <listentry>
        <case_sensitive config:type="boolean">true</case_sensitive>
        <max>1440</max>
        <min>1440</min>
        <percent>0</percent>
        <regexp>^gopher:</regexp>
      </listentry>
      <listentry>
        <case_sensitive config:type="boolean">true</case_sensitive>
        <max>4320</max>
        <min>0</min>
        <percent>20</percent>
        <regexp>.</regexp>
      </listentry>
    </refresh_patterns>
    <service_enabled_on_startup config:type="boolean">true</service_enabled_on_startup>
    <settings>
      <access_log config:type="list">
        <listentry>/var/log/squid/access.log</listentry>
      </access_log>
      <cache_dir config:type="list">
        <listentry>ufs</listentry>
        <listentry>/var/cache/squid</listentry>
        <listentry>100</listentry>
        <listentry>16</listentry>
        <listentry>256</listentry>
      </cache_dir>
      <cache_log config:type="list">
        <listentry>/var/log/squid/cache.log</listentry>
      </cache_log>
      <cache_mem config:type="list">
        <listentry>8</listentry>
        <listentry>MB</listentry>
      </cache_mem>
      <cache_mgr config:type="list">
        <listentry>webmaster</listentry>
      </cache_mgr>
      <cache_replacement_policy config:type="list">
        <listentry>lru</listentry>
      </cache_replacement_policy>
      <cache_store_log config:type="list">
```

```
        <listentry>/var/log/squid/store.log</listentry>
      </cache_store_log>
      <cache_swap_high config:type="list">
        <listentry>95</listentry>
      </cache_swap_high>
      <cache_swap_low config:type="list">
        <listentry>90</listentry>
      </cache_swap_low>
      <client_lifetime config:type="list">
        <listentry>1</listentry>
        <listentry>days</listentry>
      </client_lifetime>
      <connect_timeout config:type="list">
        <listentry>2</listentry>
        <listentry>minutes</listentry>
      </connect_timeout>
      <emulate_httpd_log config:type="list">
        <listentry>off</listentry>
      </emulate_httpd_log>
      <error_directory config:type="list">
        <listentry/>
      </error_directory>
      <ftp_passive config:type="list">
        <listentry>on</listentry>
      </ftp_passive>
      <maximum_object_size config:type="list">
        <listentry>4096</listentry>
        <listentry>KB</listentry>
      </maximum_object_size>
      <memory_replacement_policy config:type="list">
        <listentry>lru</listentry>
      </memory_replacement_policy>
      <minimum_object_size config:type="list">
        <listentry>0</listentry>
        <listentry>KB</listentry>
      </minimum_object_size>
    </settings>
  </squid>
```

属性／値／説明

`acls`

アクセス制御リスト (Access Control Settings (ACL)) のリストです。
リストの項目には名前や種類のほか、追加のオプションを設定することができます。設定可能な値の概要を知るには、YaST squid 設定モジュールをお使いください。

`http_accesses`

アクセス制御テーブル内で ACL グループに対してアクセスの許可または禁止を設定します。

1 つの定義内に複数の ACL グループが存在する場合、それぞれの ACL グループに属するメンバー全てに対して、一括で許可または禁止を設定することができます。
アクセス制御テーブルは、このリストの順序でチェックされ、最初に合致した項目のとおりにアクセスが許可もしくは禁止されます。

`http_ports`

squid がクライアントからの HTTP リクエストを待ち受ける全てのポートを指定します。
`Host` にはホスト名や IP アドレスを指定することができますが、何も指定しなくてもかまいません。
`transparent` を指定すると、PMTU discovery 機能が無効化されます。

`refresh_patterns`

squid がキャッシュ内のオブジェクトを取り扱う際の更新パターンを定義します。
更新パターンはこのリストの順序で解釈されます。最初に合致した項目のとおりに処理が行われます。
`Min` は明示的な有効期限が設定されていない場合に、オブジェクトが最新の状態であると判断する期間を分単位で指定します。`Max` は逆に、明示的な有効期限が設定されていない場合に、オブジェクトが最新の状態であると判断する上限の期間を指定します。`Percent` は直近の変更日時順に並べて、どこまでを保持するのかを指定します。明示的な有効期限が設定されていないオブジェクトについては、最新のものであると判断されます。

`settings`

設定可能な汎用パラメータとその既定値を設定します。
設定可能な項目について、詳しくは YaST の squid 設定モジュールをお使いください。

`service_enabled_on_startup`

システムの起動時に squid サービスを開始するかどうかを設定します。
値: true (有効)/false (無効)

4.25 FTP サーバ

FTP インターネットサーバの設定を行います。

例 4.56: FTP サーバの設定

```
<ftp-server>
  <AnonAuthen>2</AnonAuthen>
  <AnonCreatDirs>NO</AnonCreatDirs>
  <AnonMaxRate>0</AnonMaxRate>
  <AnonReadOnly>NO</AnonReadOnly>
```

```
    <AntiWarez>YES</AntiWarez>
    <Banner>Welcome message</Banner>
    <CertFile/>
    <ChrootEnable>NO</ChrootEnable>
    <EnableUpload>YES</EnableUpload>
    <FTPUser>ftp</FTPUser>
    <FtpDirAnon>/srv/ftp</FtpDirAnon>
    <FtpDirLocal/>
    <GuestUser/>
    <LocalMaxRate>0</LocalMaxRate>
    <MaxClientsNumber>10</MaxClientsNumber>
    <MaxClientsPerIP>3</MaxClientsPerIP>
    <MaxIdleTime>15</MaxIdleTime>
    <PasMaxPort>40500</PasMaxPort>
    <PasMinPort>40000</PasMinPort>
    <PassiveMode>YES</PassiveMode>
    <SSL>0</SSL>
    <SSLEnable>NO</SSLEnable>
    <SSLv2>NO</SSLv2>
    <SSLv3>NO</SSLv3>
    <StartDaemon>2</StartDaemon>
    <TLS>YES</TLS>
    <Umask/>
    <UmaskAnon/>
    <UmaskLocal/>
    <VerboseLogging>NO</VerboseLogging>
    <VirtualUser>NO</VirtualUser>
  </ftp-server>
```

要素／説明／コメント

`AnonAuthen`

匿名ユーザ、認証ユーザの有効／無効を切り替えます。
認証ユーザのみ: 1, 匿名ユーザのみ: 0, 両方: 2

`AnonCreatDirs`

匿名ユーザに対してディレクトリの作成を許可するかどうかを指定します。
値: YES (はい)/NO (いいえ)

`AnonReadOnly`

匿名ユーザに対してアップロードを許可するかどうかを指定します。
値: YES (はい)/NO (いいえ)

`AnonMaxRate`

匿名ユーザに対して適用する最大のデータ転送レートを指定します。
KB/s

`AntiWarez`

アップロードされているものの、ローカルの管理者が確認を行っていないファイルのダウンロードを禁止するかどうかを指定します。
値: YES (はい)/NO (いいえ)

`Banner`

サーバに接続した際に表示されるテキストを含むファイルの名前を指定します。

`CertFile`

SSL 暗号化接続で使用する DSA 証明書を指定します。
SSL 暗号化接続で使用する DSA 証明書の場所を指定します。

`ChrootEnable`

有効化すると、ローカルユーザがログインした際に既定でホームディレクトリ内に `chroot` されるようになります。
警告: このオプションを設定すると、セキュリティへの影響があります。値: YES (はい)/NO (いいえ)

`EnableUpload`

有効化すると、FTP ユーザがアップロードできるようになります。
匿名ユーザに対してアップロードを許可するには、`AnonReadOnly` を設定してください。値: YES (はい)/NO (いいえ)

`FTPUser`

匿名 FTP のユーザを指定します。

`FtpDirAnon`

匿名ユーザの場合の FTP ディレクトリを指定します。
匿名 FTP ユーザがログインした時に表示するディレクトリを指定します。

`FtpDirLocal`

認証ユーザの場合の FTP ディレクトリを指定します。
認証ユーザがログインした時に表示するディレクトリを指定します。

`LocalMaxRate`

ローカルで認証したユーザに対して適用する、最大のデータ転送レートを指定します。
KB/s

`MaxClientsNumber`

クライアント側からの接続最大数を指定します。

`MaxClientsPerIP`

1 つの IP アドレスあたりの最大接続数を指定します。
1 つの IP アドレスあたりに対して許可する、最大の同時接続数を制限することになります。

`MaxIdleTime`

クライアント側から送信される FTP コマンドの最大待機時間 (タイムアウト) を指定します。
分

`PasMaxPort`

パッシブ接続応答の場合のポート範囲のうち、最大値を指定します。
`PassiveMode` を YES に設定する必要があります。

`PasMinPort`

パッシブ接続応答の場合のポート範囲のうち、最小値を指定します。
`PassiveMode` を YES に設定する必要があります。

`PassiveMode`

パッシブモードを有効化します
値: YES (はい)/NO (いいえ)

`SSL`

セキュリティ設定
TLS/SSL を無効化する: 0, TLS/SSL を受け付ける: 1, TLS/SSL 無しでの接続を拒否する: 2

`SSLEnable`

有効化すると、SSL での接続を受け付けるようになります。
値: YES (はい)/NO (いいえ)

`SSLv2`

有効化すると、SSL バージョン 2 での接続を許可するようになります。
値: YES (はい)/NO (いいえ)

`SSLv3`

有効化すると、SSL バージョン 3 での接続を許可するようになります。
値: YES (はい)/NO (いいえ)

`StartDaemon`

FTP デーモンの起動方法を指定します。
手動: 0; システムの起動時: 1; systemd ソケット経由: 2

`TLS`

有効化すると、TLS での接続を受け付けるようになります。

値: YES (はい)/NO (いいえ)

`Umask`

ファイル作成時のマスク値を指定します。(ファイル向けの umask):(ディレクトリ向けの umask) の形式で指定します。
最も制限を厳しくするには `177:077` のように設定します。

`UmaskAnon`

匿名ユーザ向けのファイル作成時の umask 値を設定します。
8 進数で指定したい場合は、頭に "0" を付けてください。それ以外の場合は、10 進数として扱われます。

`UmaskLocal`

認証ユーザ向けの umask 値を設定します。
8 進数で指定したい場合は、頭に "0" を付けてください。それ以外の場合は、10 進数として扱われます。

`VerboseLogging`

有効化すると、全ての FTP 要求と応答がログに記録されます。
値: YES (はい)/NO (いいえ)

`VirtualUser`

仮想ユーザを使用することで、システムのアカウントとは別に FTP アカウントを管理できるようになります。
値: YES (はい)/NO (いいえ)



注記: ファイアウオールについて

FTP サーバを正しく動作させるためには、ファイアウオール側も設定を行う必要があります。

4.26 TFTP サーバ

TFTP インターネットサーバの設定を行います。

これを使用することで、TFTP (Trivial File Transfer Protocol) のサーバ側を設定することができます。サーバは `systemd` のソケットを利用して開始されます。

注意: TFTP と FTP は異なるものです。

例 4.57: TFTP サーバ設定:

```
<tftp-server>
  <start_tftpd config:type="boolean">true</start_tftpd>
```

```
    <tftp_directory>/tftpboot</tftp_directory>
  </tftp-server>
```

start_tftpd
: TFTP サーバを有効化するかどうかを指定します。`true` もしくは `false` の値を指定してください。

tftp_directory
: 起動イメージのディレクトリ: 提供するファイルの存在するディレクトリを指定します。
一般的には `/tftpboot` などのディレクトリを指定します。指定したディレクトリが存在しない場合、自動的に作成されます。TFTP サーバは、このディレクトリをルートとして使用 (`-s` オプション) します。

4.27 firstboot の処理手順

YaST の firstboot ユーティリティ (YaST 初期システム設定) はインストールの直後に動作する仕組みで、ここでは新規にインストールしたシステムに対して、必要な設定処理を行います。インストール後の初回起動では、ユーザはシステムを簡易に設定するための各種の手順を実施することになります。なお、YaST の firstboot は既定では動作しませんので、必要であれば設定を行って有効化します。

例 4.58: FIRSTBOOT の処理手順の有効化

```
    <firstboot>
      <firstboot_enabled config:type="boolean">true</firstboot_enabled>
    </firstboot>
```

4.28 セキュリティ設定

このモジュールの機能を利用することで、インストール先のシステムに対するローカルセキュリティを設定することができます。ローカルセキュリティには起動時の設定のほか、ログイン関連の設定やパスワードの設定、ユーザ追加時の設定やファイルパーミッションなどが含まれます。

セキュリティ関連の自動設定は、動作中のシステムで利用できるセキュリティモジュール内の設定に似た仕組みです。ここではさまざまな設定を行うことができます。

例 4.59: セキュリティ設定

> 下記の例で示されているキーと値の意味についてはリファレンスをお読みください。
>
> ```
> <security>
>   <console_shutdown>ignore</console_shutdown>
>   <displaymanager_remote_access>no</displaymanager_remote_access>
>   <fail_delay>3</fail_delay>
> ```

```
  <faillog_enab>yes</faillog_enab>
  <gid_max>60000</gid_max>
  <gid_min>101</gid_min>
  <gdm_shutdown>root</gdm_shutdown>
  <lastlog_enab>yes</lastlog_enab>
  <encryption>md5</encryption>
  <obscure_checks_enab>no</obscure_checks_enab>
  <pass_max_days>99999</pass_max_days>
  <pass_max_len>8</pass_max_len>
  <pass_min_days>1</pass_min_days>
  <pass_min_len>6</pass_min_len>
  <pass_warn_age>14</pass_warn_age>
  <passwd_use_cracklib>yes</passwd_use_cracklib>
  <permission_security>secure</permission_security>
  <run_updatedb_as>nobody</run_updatedb_as>
  <uid_max>60000</uid_max>
  <uid_min>500</uid_min>
  <selinux_mode>permissive</selinux_mode>
  <lsm_select>selinux</lsm_select>
</security>
```

4.28.1 パスワードの設定

最小パスワード長やパスワードの有効期限など、各種のパスワード設定を行いたい場合は、`<pass_*` リソースを使用します。

なお、`<encryption>` リソースを使用することで、現在サポートされている暗号化方式の中からいずれかを選択することができます。何も指定しない場合は `sha512` が設定されます。

暗号化方式としては下記のいずれかを指定することができます:

- `md5` — 128 ビットのハッシュ長による長いパスワードの暗号化
- `sha256` または `sha512` — 広く使用されているより安全なハッシュアルゴリズムです
- `des` — 指定可能ではありますが、安全性が不足するため、この方式は非推奨となります

4.28.2 起動の設定

security リソースでは、起動時の設定も変更することができます。

`Ctrl – Alt – Del` の解釈方法

コンソールを使用しているユーザが `Ctrl – Alt – Del` を押した際、通常はシステムの再起動が始まります。ただし、対象のマシンをサーバとしても使用しているような場合、このような処理を無効化することもできます。

GDM のシャットダウン時の動作
: GDM 経由でマシンをシャットダウンすることのできるユーザの一覧を設定します。

4.28.3 ログインの設定

さまざまなログイン関連の設定を行います。これらの設定は主に、`/etc/login.defs` ファイル内に保存されます。

4.28.4 新規ユーザの設定 (`useradd` の設定)

ユーザ ID およびグループ ID の最小値と最大値をそれぞれ設定します。

4.28.5 Linux セキュリティモジュール (LSM) の設定

openSUSE Leap 15.4 およびそれ以降のバージョンでは、インストール時の制御ファイルの設定として `<lsm_select>` が提供されるようになりました。これは主要な Linux セキュリティモジュール (LSM) を設定するための仕組みで、インストール後に既定で有効化されるモジュール (AppArmor, SELinux, なし) を選択するものです。

selinux_mode
: 任意指定です。SELinux のモードを設定します。`permissive`, `enforcing`, `disabled` のいずれかを指定してください。

lsm_select
: 任意指定です。インストール時に選択する主要な Linux セキュリティモジュールを選択します。`selinux`, `apparmor`, `none` (無し) のいずれかを指定してください。

4.28.6 OpenSCAP セキュリティポリシーの使用

YaST では、OpenSCAP のセキュリティポリシーを利用してシステムの強化をはかることができます。セキュリティポリシーのチェックと適用にあたっては、下記の 2 つの段階を踏むことになります:

- インストール時に YaST は、ファイルシステムの暗号化など、インストール後からは修正の難しい部分を中心にセキュリティポリシーのチェックを行います。プロファイル内のルールに適合しない箇所が検出されると、AutoYaST は問題を報告してインストールを中止します。
- これに加えて、AutoYaST は `ssg-apply` ツールをインストールして設定します。`ssg-apply` は、システムが選択したポリシーに準拠しているかどうかをチェックして、必要であれば修復を実施します。

`security_policy` セクションでは、セキュリティポリシーの選択と `ssg-apply` の設定を行うことができます。

policy
: チェックもしくは適用するセキュリティポリシーを指定します。現時点では Defense Information Systems Agency Security Technical Implementation Guide (DISA STIG) のみを指定することができます。このポリシーを使用したい場合は、`stig` を指定してください。この要素は必須です。

action
: 初回の起動時に `ssg-apply` がするべきことを指定します。

  - `scan` : 初回の起動時にシステムのスキャンを実施します。これが既定の動作です。
  - `remediate` : スキャンを実施したあと、選択したポリシーに準拠するようシステムを修復します。
  - `none` : 初回起動時に設定を行いますが、`ssg-apply` を実行しません。このオプションは、事前にセキュリティポリシーそのものを修正したい場合に使用します。

例 4.60: DEFENSE INFORMATION SYSTEMS AGENCY SECURITY TECHNICAL IMPLEMENTATION GUIDE の選択

> 下記は初回の起動時に DISA STIG ポリシーを利用してシステムをチェックし、必要であれば修正を行う場合の例です。
>
> ```
> <security>
>   <security_policy>
>     <policy>stig</policy>
>     <action>remediate</action>
>   </security_policy>
> ```

```
</security>
```

4.29 Linux 監査フレームワーク (Linux Audit Framework (LAF))

このモジュールを使用することで、監査デーモンの設定や監査サブシステムに対するルールの追加などを行うことができます。

例 4.61: LAF の設定

```
<audit-laf>
  <auditd>
    <flush>INCREMENTAL</flush>
    <freq>20</freq>
    <log_file>/var/log/audit/audit.log</log_file>
    <log_format>RAW</log_format>
    <max_log_file>5</max_log_file>
    <max_log_file_action>ROTATE</max_log_file_action>
    <name_format>NONE</name_format>
    <num_logs>4</num_logs>
  </auditd>
  <rules/>
</audit-laf>
```

属性／値／説明

`auditd/flush`
: ディスクへのデータの書き込み方法を設定します。
`INCREMENTAL` を指定すると、`auditd/freq` パラメータで設定されたレコード数ごとに、ディスクへの明示的なフラッシュ処理が発行されます。`NONE` を指定すると、データを書き込むにあたって特別な処理は行いません。`DATA` を指定するとデータ部分のみを同期し、`SYNC` を指定するとデータとメタデータの両方を同期するようになります。

`auditd/freq`
: このパラメータは、ディスクへの明示的な書き込み命令を発行するまでのレコード数を指定します。
このパラメータを使用するには、`flush` を `INCREMENTAL` にする必要があります。

`auditd/log_file`
: ログファイルのフルパスを指定します。

`auditd/log_fomat`
: 残しておくべき情報のサイズを指定します。

`RAW` を指定すると、(カーネルが送信したものと全く同じ形式で) 全てのデータを保存します。
`NOLOG` を指定すると、ディスクに書き込むことなく全てのデータを破棄します (ただしディスパッチャから送信されたデータはそのまま受け取ります) 。

`auditd/max_log_file`
残しておくべき情報のサイズを指定します。
単位: メガバイト

`auditd/num_logs`
ログファイルの数を指定します。
`max_log_file_action` を `ROTATE` に設定する必要があります。

`auditd/max_log_file_action`
ログ記録サイズの上限に達した場合の処理を指定します。
`ROTATE` を指定すると、`auditd/num_logs` で指定した数だけ過去のログファイルを保存するようになります。`SYSLOG` を指定すると、監査デーモンはシステムログに対して警告を書き込むようになります。`SUSPEND` を指定すると、デーモンからディスクへの書き込みが停止するようになります。`IGNORE` は何も行わず、`KEEP_LOGS` は `ROTATE` と似たような動作ですが、ログファイルへの上書きは行わなくなります。

`auditd/name_format`
ログファイルにコンピュータ名を書き込む際のコンピュータ名の書式を指定します。
`USER` を指定すると、ユーザ側で定義した名前を使用します。`NONE` を指定すると、コンピュータ名を書き込みません。`HOSTNAME` を指定すると、'gethostname' システムコールで取得できるホスト名を書き込みます。`FQD` を指定すると、完全修飾ドメイン名を使用します。

`rules`
auditctl 向けのルール
ルールを手作業で編集することもできますが、知識のあるユーザ向けの機能となります。全てのオプションに関する詳細は、`man auditctl` をお読みください。

4.30 ユーザとグループ

4.30.1 ユーザ

ユーザの一覧は `<users>` セクション内に記述します。ログインができるようにするには `root` に対する設定を記述するか、もしくは `linuxrc` のオプションで `rootpassword` を設定します。

例 4.62: 最小限のユーザ設定

```
<users config:type="list">
  <user>
    <username>root</username>
    <user_password>password</user_password>
    <encrypted config:type="boolean">false</encrypted>
  </user>
    <user>
    <username>tux</username>
    <user_password>password</user_password>
    <encrypted config:type="boolean">false</encrypted>
  </user>
</users>
```

下記の例では、さらに複雑なシナリオを構成しています。シェルやホームディレクトリの親となるディレクトリなど、`/etc/default/useradd` が示す既定値を適用しています。

例 4.63: 複雑なユーザ設定

```
<users config:type="list">
  <user>
    <username>root</username>
    <user_password>password</user_password>
    <uid>1001</uid>
    <gid>100</gid>
    <encrypted config:type="boolean">false</encrypted>
    <fullname>Root User</fullname>
    <authorized_keys config:type="list">
      <listentry> ssh-rsa
 AAAAB3NzaC1yc2EAAAADAQABAAABAQDKLt1vnW2vTJpBp3VK91rFsBvpY97NljsVLdgUrlPbZ/
L51FerQQ+djQ/ivDASQjO+567nMGqfYGFA/De1EGMMEoeShza67qjNi14L1HBGgVojaNajMR/
NI2d1kDyvsgRy7D7FT5UGGUNTOdlcSD3b85zwgHeYLidgcGIoKeRi7HpVDOOTyhwUv4sq3ubrPCWARgPeOLdVFa9clC8PTZdxSeKp4jp
PvMDa96DpxH1VlzJlAIHQsMkMHbsCazPNC0++Kp5ZVERiH root@example.net</listentry>
    </authorized_keys>
  </user>
  <user>
    <username>tux</username>
    <user_password>password</user_password>
    <uid>1002</uid>
    <gid>100</gid>
    <encrypted config:type="boolean">false</encrypted>
    <fullname>Plain User</fullname>
    <home>/Users/plain</home>
    <password_settings>
      <max>120</max>
      <inact>5</inact>
    </password_settings>
  </user>
</users>
```



注記: authorized_keys ファイルの上書きに関する注意

`authorized_keys` セクションでユーザに対する SSH の公開鍵を設定する場合、既存の `$HOME/.ssh/authorized_keys` ファイルの内容は上書きされます。存在しない場合は指定した内容がそのまま書かれます。`authorized_keys` ファイルを上書きしたくない場合は、AutoYaST の制御ファイル内に `authorized_keys` セクションを記述しないでください。



注記: rootpassword と root ユーザオプションの組み合わせについて

`linuxrc` での `rootpassword` と `root` に対する `user` セクションを同時に設定することもできます。`user` セクション内にパスワードの指定が書かれていない場合、パスワードは `linuxrc` 側に記述したものが使われますが、書かれている場合は `linuxrc` に書かれたものより優先されます。



警告: root 以外のスーパーユーザアカウントを作成してはなりません

技術的な観点では `root` 以外の名前でユーザ ID (`uid`) `0` のアカウントを作成することができますが、アプリケーションやスクリプト、サードパーティ製の製品によっては、`root` という名前で管理者ユーザが存在していることを前提にしているものもあります。これらの設定を変更するなどして `root` のアカウント名変更に対応できる場合もありますが、場合によってはパッケージ更新のたびに毎回対応を行わなければならないこともあります。特に複数のサードパーティ製アプリケーションが存在する複雑な環境の場合、それぞれのアプリケーションに対して対応の可否を確認しなければならなくなります。

このように `root` のアカウント名変更による影響が予測できない範囲に及ぶことから、SUSE では `root` のアカウント名変更をサポート対象外としております。

また `root` のアカウント名変更は一般に、アカウントの隠蔽や攻撃回避の目的で行われますが、`/etc/passwd` には `644` のパーミッションを設定して一般ユーザからも読めるように設定しなければならないことから、たとえアカウント名を変更したとしても、ユーザ ID 0 のアカウント名が何であるのかは容易に判別ができてしまう問題もあります。`root` アカウントの保護に関する詳細は、『セキュリティ強化ガイド』、第15章「ユーザ管理」、15.5項「root ログインの制限」および『セキュリティ強化ガイド』、第15章「ユーザ管理」、15.5.3項「SSH ログインの制限」をお読みください。



注記: ユーザ ID (uid) の指定について

Linux システム内のユーザには数値のユーザ ID が設定されます。AutoYaST の制御ファイル内でこれを指定したい場合は、`uid` タグを利用して直接指定することができますが、指定しない場合はシステム側で自動的に設定されます。

なお、ユーザ ID はシステム内で重複があってはなりません。重複が発生してしまうと、ログインマネージャ `gdm` のようなアプリケーションが、正しく動作しなくなってしまいます。

AutoYaST の制御ファイルでユーザを追加する場合は、ID を指定するユーザとしないユーザを混在させるのは避けておくことを強くお勧めします。混在させてしまうと、ユーザ ID の重複が発生する可能性があります。AutoYaST の制御ファイル内では、全てのユーザに対してユーザ ID を設定するか、もしくは全てのユーザに対してユーザ ID を自動設定させるのか、いずれかにしてください。

属性／値／説明

`username`
: テキスト

```
<username>lukesw</username>
```

必須です。正しいユーザ名である必要があります。詳しくは `man 8 useradd` をお読みください。

`fullname`
: テキスト

```
<fullname>Tux Torvalds</fullname>
```

任意指定です。ユーザのフルネームを指定します。

`forename`
: テキスト

```
<forname>Tux</forename>
```

任意指定です。ユーザの名前 (ファーストネーム) を指定します。

`surname`
: テキスト

```
<surname>Skywalker</surname>
```

任意指定です。ユーザの姓を指定します。

`uid`

番号

```
<uid>1001</uid>
```

任意指定です。ユーザ ID を指定します。システム内で重複していてはならず、かつ負の数であってはなりません。何も指定しない場合、AutoYaST はユーザ ID を自動的に選択します。さらに詳しい情報については、注記: ユーザ ID (`uid`) の指定について もお読みください。

`gid`

番号

```
<gid>100</gid>
```

任意指定です。初期グループ ID を指定します。重複していてはならず、かつ負の数であってはなりません。また、既存のグループ ID を指定しなければなりません。

`home`

パス

```
<home>/home/luke</home>
```

任意指定です。ユーザのホームディレクトリの絶対パスを指定します。既定では `/home/ユーザ名` が使用されます (例: ユーザ名が `alice` であれば `/home/alice` になります) 。

`home_btrfs_subvolume`

ブール値

```
<home_btrfs_subvolume config:type="boolean">true</home_btrfs_subvolume>
```

任意指定です。btrfs サブボリューム内にホームディレクトリを生成します。既定では無効化されます。

`shell`

パス

```
<shell>/usr/bin/zsh</shell>
```

任意指定です。既定では `/bin/bash` が設定されます。それ以外のものを指定する場合は、対応する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 (`software` セクション内でパッケージをインストールする必要があります) 。

`user_password`

テキスト

```
<user_password>some-password</user_password>
```

任意指定です。パスワードを単純なテキスト形式 (非推奨) か暗号化した形式で設定します。暗号化した形式でパスワードを設定したい場合は、`mkpasswd` コマンドをお使いください。パスワードは `/etc/shadow` (コロン区切りで 2 つめの項目) 内に書き込まれます。プロファイル内でパスワードの暗号化可否を設定したい場合は、`encrypted` パラメータを設定してください。パスワードの暗号化を有効化している場合、この項目にエクスクラメーションマーク (!) を指定すると、指定した値がそのまま `/etc/shadow` に書き込まれ、対象のアカウントがロック (施錠) されます。これにより、そのユーザはコンソールにログインできなくなります。

`encrypted`

ブール値

```
<encrypted config:type="boolean">true</encrypted>
```

任意指定です。指定しない場合、`false` が設定されたものとみなされます。ここでは、プロファイル内でユーザに対するパスワードを暗号化するかどうかを設定します。AutoYaST では標準的な暗号化アルゴリズム (詳しくは `man 3 crypt` をお読みください) に対応しています。

`password_settings`

パスワードの設定

```
<password_settings>
  <expire/>
  <max>60</max>
  <warn>7</warn>
</password_settings>
```

任意指定です。いくつかのパスワード関連の設定をカスタマイズすることができます: `expire` (`YYYY-MM-DD` 形式でのアカウント有効期限), `flag` (`/etc/shadow` のフラグ), `inact` (パスワードの有効期限が切れてから、アカウントを無効化するまでの日数), `max` (パスワードの有効期限日数), `min` (有効期限が切れたあと、ユーザがパスワードを変更できるようになるまでの日数), `warn` (パスワードの有効期限切れを予告警告する日数) 。

`authorized_keys`

認可済みの鍵の一覧

```
<authorized_keys config:type="list">
  <listentry>ssh-rsa ...</listentry>
</authorized_keys>
```

`$HOME/.ssh/authorized_keys` に書き込むべき認可済みのキーのリストを指定します。下記の例をご覧ください。

4.30.2 ユーザの既定値

プロファイル内には、パスワードの有効期限や初期に所属するグループ、ホームディレクトリのプレフィクスなど、さまざまな新規ユーザ向けの既定値を設定することができます。なお、プロファイル内で個別に指定していれば、既定値ではなく指定した値を利用してユーザを作成することができます。AutoYaST ではこれらの設定を `/etc/default/useradd` やその他の `useradd` 向けファイルに書き込みます。

属性／値／説明

`group`
: テキスト

```
<group>100</group>
```

任意指定です。既定の初期ログイングループを指定します。

`home`
: パス

```
<home>/home</home>
```

任意指定です。ユーザのホームディレクトリのプレフィクスを指定します。

`expire`
: 日付

```
<expire>2017-12-31</expire>
```

任意指定です。`YYYY-MM-DD` の形式で、既定のパスワード有効期限を指定します。

`inactive`
: 番号

```
<inactive>3</inactive>
```

任意指定です。アカウントの期限が切れてから無効化するまでの日数を指定します。

`shell`
: パス

```
<shell>/usr/bin/fish</shell>
```

既定のログインシェルを指定します。既定値は `/bin/bash` です。それ以外のものを指定する場合は、対応する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 (`software` セクション内でパッケージをインストールする必要があります) 。

`umask`

ファイル作成時のモードマスク

```
<umask>022</umask>
```

ホームディレクトリに対するファイル作成時のモードマスクを設定します。既定では `useradd` は `022` を使用します。詳しくは `man 8 useradd` および `man 1 umask` をお読みください。

4.30.3 グループ

グループの一覧は、例に示しているとおり `<groups>` 内に定義します。

例 4.64: グループの設定

```
<groups config:type="list">
  <group>
    <gid>100</gid>
    <groupname>users</groupname>
    <userlist>bob,alice</userlist>
  </group>
</groups>
```

属性／値／説明

`groupname`

テキスト

```
<groupname>users</groupname>
```

必須です。正しいグループ名である必要があります。詳しくは `man 8 groupadd` をお読みください。

`gid`

番号

```
<gid>100</gid>
```

任意指定です。グループ ID を指定します。重複していてはならず、かつ負の数であってはなりません。

`userlist`

ユーザリスト

```
<userlist>bob,alice</userlist>
```

任意指定です。グループに所属すべきユーザの一覧を指定します。ユーザ名はカンマで区切ります。

4.30.4 ログインの設定

AutoYaST のプロファイルでは、2 種類の特殊なログイン設定 (自動ログイン、パスワード無しログイン) を有効化することができます。いずれも既定では無効化されています。

例 4.65: 自動ログインおよびパスワード無しログインの有効化

```
<login_settings>
  <autologin_user>vagrant</autologin_user>
  <password_less_login config:type="boolean">true</password_less_login>
</login_settings>
```

属性／値／説明

`password_less_login`

ブール値

```
<password_less_login config:type="boolean">true</password_less_login>
```

任意指定です。パスワード無しでのログインを許可します。ただし、グラフィカルなログインにしか効果がありません。

`autologin_user`

テキスト

```
<autologin_user>alice</autologin_user>
```

任意指定です。指定したユーザに対する自動ログインを行います。

4.31 独自のユーザスクリプト

自動インストールの処理内にスクリプトを追加することで、必要に応じた独自のカスタマイズを実現することができるほか、インストール時のさまざまな段階で制御を行うことができます。

自動インストールの処理内では、全部で 5 箇所のスクリプトの実行タイミングが提供されています。

また、全てのスクリプトは <scripts> 内に配置しなければなりません。

- `pre-scripts`: いずれかの作業が発生するよりも前の、最も早い段階 (事前スクリプト)
- `postpartitioning-scripts`: パーティションの設定と `/mnt` へのマウントが完了しているが、RPM のインストールを行うよりも前の段階 (パーティション設定後スクリプト)

- `chroot-scripts`: パッケージのインストールが完了し、初回の起動が行われるよりも前の段階 (chroot 環境スクリプト)
- `post-scripts`: インストール済みのシステムの初回起動時で、サービスが全く動作していない状態 (事後スクリプト)

準備スクリプト: インストール済みのシステムの初回起動時で、全てのサービスを起動したあと (なお、このスクリプトだけは YaST から実行されないため、他のスクリプトとは記述方法が異なります。詳しくは 4.31.5項「準備スクリプト」を参照してください)

4.31.1 事前スクリプト

YaST がシステムに対して何らかの変更を加えるよりも前の段階で動作します (ハードウエアの検出が完了しているものの、パーティション設定やパッケージのインストールよりも前の段階) 。

事前スクリプトでは制御ファイルの修正を行い、AutoYaST に対して修正後のものを読み込み直すように指示することができます。制御ファイルは `/tmp/profile/autoinst.xml` に配置されますので、修正後の版を `/tmp/profile/modified.xml` に配置してください。AutoYaST では、事前スクリプトの動作完了後に修正後のファイルを読み込みなおします。

事前スクリプトでは、ストレージデバイスの編集を行うこともできます。たとえば新しいパーティションの作成や、マルチパスのような特定技術の設定変更などがそれにあたります。AutoYaST では、全ての事前スクリプトが実行された後に、必ず再度ストレージデバイスをチェックするようになっています。



注記: 確認表示付きの事前スクリプトについて

事前スクリプトはインストールの冒頭で動作します。インストール時に確認を求めるように設定した場合 (`profile/install/general/mode/confirm`) でも、これらのスクリプトは確認画面を表示するよりも前の段階で動作してしまいます。



注記: 事前スクリプトと zypper について

事前スクリプトで zypper を実行したい場合は、動作中の YaST プロセスと衝突しないようにするため、ZYPP_LOCKFILE_ROOT="/var/run/autoyast" という環境変数を設定する必要があります。

事前スクリプト の要素は下記のように配置しなければなりません:

```
<scripts>
  <pre-scripts config:type="list">
```

```
    <script>
      ...
    </script>
  </pre-scripts>
</scripts>
```

4.31.2 パーティション設定後スクリプト

YaST がパーティションの設定を行い、`/etc/fstab` 内に書き込みを行った後に動作するスクリプトです。`/mnt` 内には何もないシステムがマウントされているタイミングになります。

パーティション設定後スクリプト の要素は下記のように配置しなければなりません:

```
<scripts>
  <postpartitioning-scripts config:type="list">
    <script>
      ...
    </script>
  </postpartitioning-scripts>
</scripts>
```

4.31.3 chroot 環境スクリプト

chroot 環境スクリプトはインストールが完了したあと、マシンの再起動を行う前に動作します。chroot スクリプトはインストーラがインストール済みのシステムに chroot し、ブートローダの設定を行う前に実行するか、もしくはインストール済みのシステムに chroot した後に動作させることができます (詳しくは `chrooted` パラメータをお読みください) 。

chroot 環境スクリプト の要素は下記のように配置しなければなりません:

```
<scripts>
  <chroot-scripts config:type="list">
    <script>
      ...
    </script>
  </chroot-scripts>
</scripts>
```

4.31.4 事後スクリプト

これらのスクリプトは、AutoYaST がシステムの設定を完了し、システムの初回起動が完了したあとに実行されます。

事後スクリプト の要素は下記のように配置しなければなりません:

```
<scripts>
    <post-scripts config:type="list">
      <script>
        ...
      </script>
    </post-scripts>
  </scripts>
```

4.31.5 準備スクリプト

これらのスクリプトは YaST の処理が完了し、ネットワークの準備が完了したあと、初回の起動処理内で実行されます。これらの最終スクリプトは `/usr/lib/YaST2/bin/autoyast-initscripts.sh` を利用して動作するものであり、一度のみ動作します。準備スクリプトは init-scripts タグで設定します。

準備スクリプト の要素は下記のように配置しなければなりません:

```
<scripts>
  <init-scripts config:type="list">
    <script>
    ...
    </script>
  </init-scripts>
</scripts>
```

準備スクリプトは他の種類のスクリプトとは異なり、YaST 経由で実行されるものではなく、YaST の完了後に実行されます。このような仕組みであることから、この XML 表記は他とは異なる形になります。

準備スクリプトの XML 表記

`location`

スクリプトの取得元の場所を指定します。場所にはプロファイルと同じプロトコル (http, ftp, nfs, など) を指定することができます。

```
<location>http://10.10.0.1/myInitScript.sh</location>
```

<location> もしくは <source> を定義しなければなりません。

`source`

スクリプトそれ自身 (ソースコード) は CDATA タグ内にカプセル化して記述します。XML プロファイル内にシェルスクリプト全体を配置したくない場合は、location パラメータを指定してください。

```
<source>
<![CDATA[echo "Testing the init script" >/tmp/init_out.txt]]></source>
```

<location> もしくは <source> を定義しなければなりません。

`filename`

スクリプトのファイル名を指定します。一時ディレクトリ `/tmp` 内にファイル名となります。

```
<filename>mynitScript5.sh</filename>
```

準備スクリプトが 1 つだけである場合は任意です。この場合は、既定の名前 (`init-scripts`) が使用されます。複数のスクリプトを使用している場合は、それぞれのスクリプトに対して重複しない名前を設定しなければなりません。

`rerun`

`ayast_setup` を使用して XML ファイルを複数回実行するように設定しても、通常はスクリプトが一度だけ実行されます。この既定の動作を変更したい場合は、この値を `true` に設定します。

```
<rerun config:type="boolean">true</rerun>
```

任意指定です。既定値は `false` (スクリプトを一度だけ実行する) です。

制御ファイルに対して手作業で追加を行っている場合は、スクリプトには CDATA 要素を設定して、制御ファイル内でファイルの文法とその他のタグが混ざらないようにしてください。

4.31.6 スクリプトの XML パラメータ

下記に示す XML 要素のほとんどは、ここまでに説明してきた全ての種類のスクリプトに対して設定することができます。ただし 準備スクリプト については例外で、ここには一部の要素しか指定することができません。詳しくは 4.31.5項「準備スクリプト」をお読みください。



重要: 廃止予定の要素

`debug` は廃止予定の要素であり、将来のリリースでは削除される予定です。新しく設定するには、`interpreter` パラメータに、インタプリタごとの独自のデバッグパラメータを指定してください。たとえば <interpreter>shell</interpreter> の代わりに <interpreter>/bin/sh -x</interpreter> のように指定すると、`debug` フラグを指定した場合と同じ結果が得られます。

`location`

スクリプトの取得元の場所を指定します。場所には制御ファイルと同じプロトコル (http, ftp, nfs, など) を指定することができます。たとえば下記のようになります:

```
<location>http://10.10.0.1/myPreScript.sh</location>
```

場所の指定には、制御ファイルからの相対パスを使用することもできます。相対パスを指定する場合、`location` は下記のようになります:

```
<location>relurl://script.sh</location>
```

それ以外にも、`repo` URI スキームを使用することもできます。これはインストール元からの相対パスを指定するもので、たとえば下記のようになります:

```
<location>repo:/script.sh</location>
```

`location` もしくは `source` を定義しなければなりません。

`source`

スクリプトそれ自身 (ソースコード) は CDATA タグ内にカプセル化して記述します。XML プロファイル内にシェルスクリプト全体を配置したくない場合は、location パラメータを指定してください。

```
<source>
<![CDATA[
echo "Testing the pre script" > /tmp/pre-script_out.txt
]]>
</source>
```

`location` もしくは `source` を定義しなければなりません。

`interpreter`

スクリプトで使用するインタプリタを指定します。動作させる環境内で利用可能な任意のインタプリタを指定することができます。通常はフルパスでインタプリタを指定します (パラメータを付けることもできます) が、古い形式として "shell", "perl", "python" の各キーワードでインタプリタを指定することもできます (廃止予定の仕組みで、`debug` フラグでサポートされるものです) 。

```
<interpreter>/bin/bash -x</interpreter>
```

任意指定です。既定値は `shell` です。

`file name`

スクリプトのファイル名を指定します。一時ディレクトリ `/tmp` 内でのファイル名となります。

```
<filename>myPreScript5.sh</filename>
```

任意指定です。既定値はスクリプトの種類ごとに異なります (この例の場合は pre-scripts になります) 。複数のスクリプトを使用している場合は、それぞれのスクリプトに対して重複しない名前を設定しなければなりません。また、`filename` を指定しておらず、`location` を指定している場合は、指定したロケーションパスのファイル名を使用します。

`feedback`

このブール値を `true` に設定すると、スクリプトの出力メッセージとエラーメッセージ (STDOUT および STDERR) がポップアップとして表示されるようになります。ユーザ側では OK ボタンを押して確認する必要が発生します。

```
<feedback config:type="boolean">true</feedback>
```

任意指定です。既定値は `false` です。

`feedback_type`

`message` , `warning` , `error` のいずれかの値を指定します。これらのポップアップに対するタイムアウトは、<report> セクションで設定します。

```
<feedback_type>warning</feedback_type>
```

任意指定です。何も指定しない場合、常に動作を一時停止してポップアップを表示します。

`debug`

この値を `true` に設定すると、シェルスクリプト内の各行がログに記録されるようになります。Perl スクリプトの場合は警告表示が有効化されます。これは廃止予定のパラメータで、他の言語の場合は `interpreter` パラメータにそれぞれのデバッグ用オプションを指定します (例: "<interpreter>ruby -w</interpreter>") 。

```
<debug config:type="boolean">true</debug>
```

任意指定です。既定値は `true` です。

`notification`

スクリプトを裏で実行している間、ここで設定したテキストをポップアップとして表示します。

```
<notification>Please wait while script is running...</notification>
```

任意指定です。指定しない場合、通知ポップアップを表示しません。

`param-list`

スクリプトを呼び出す際のパラメータを指定します。`param` の項目は必要に応じて複数個設定することができます。複数個を指定した場合、それらはスペースで区切られてスクリプトのコマンドラインを構成します。パラメータそれ自身にスペースを含めたい場合は、シェルの引用符を指定してください。

```
<param-list config:type="list">
```

```
  <param>par1</param>
  <param>par2 par3</param>
  <param>"par4.1 par4.2"</param>
</param-list>
```

任意指定です。設定しない場合、スクリプトに対してパラメータを渡しません。

`rerun`

XML ファイル内に複数の `ayast_setup` を配置した場合であっても、スクリプトは通常、一度だけ実行します。この既定の動作を変更したい場合は、この値を `true` に設定してください。

```
<rerun config:type="boolean">true</rerun>
```

任意指定です。既定値は `false` で、スクリプトを一度だけ実行します。

`chrooted`

インストールの際、新しいシステムは `/mnt` 内にマウントされます。このパラメータを `false` に設定すると、AutoYaST は `chroot` を行わず、このステージではブートローダをインストールしなくなります。このパラメータを `true` に設定すると、AutoYaST は `chroot` を実行して `/mnt` 内に入り、ブートローダのインストールを行うようになります。つまり、新しくインストールしたシステムに対して何らかの変更を行いたい場合、`/mnt` のプレフィクスを指定せずに実行することができるようになります。

```
<chrooted config:type="boolean">true</chrooted>
```

任意指定です。既定値は `false` です。このオプションは chroot 環境スクリプトでのみ設定することができます。

4.31.7 スクリプト例

例 4.66: スクリプト設定

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://www.suse.com/1.0/
configns">
<scripts>
  <chroot-scripts config:type="list">
    <script>
      <chrooted config:type="boolean">true</chrooted>
      <filename>chroot-post.sh</filename>
      <interpreter>shell</interpreter>
      <source><![CDATA[
echo "Testing chroot (chrooted) scripts"
ls
]]>
      </source>
```

```
    </script>
    <script>
      <filename>chroot-pre.sh</filename>
        <interpreter>/bin/bash -x</interpreter>
        <source><![CDATA[
echo "Testing chroot scripts"
df
cd /mnt
ls
]]>
        </source>
      </script>
    </chroot-scripts>
    <post-scripts config:type="list">
      <script>
        <filename>post.sh</filename>
        <interpreter>shell</interpreter>
        <source><![CDATA[
echo "Running Post-install script"
systemctl start portmap
mount -a 192.168.1.1:/local /mnt
cp /mnt/test.sh /tmp
umount /mnt
]]>
        </source>
      </script>
      <script>
        <filename>post.pl</filename>
        <interpreter>perl</interpreter>
        <source><![CDATA[
print "Running Post-install script";
]]>
        </source>
      </script>
    </post-scripts>
    <pre-scripts config:type="list">
      <script>
        <interpreter>shell</interpreter>
        <location>http://192.168.1.1/profiles/scripts/prescripts.sh</location>
      </script>
      <script>
        <filename>pre.sh</filename>
        <interpreter>shell</interpreter>
        <source><![CDATA[
echo "Running pre-script"
]]>
        </source>
      </script>
    </pre-scripts>
    <postpartitioning-scripts config:type="list">
      <script>
```

```
        <filename>postpart.sh</filename>
        <interpreter>shell</interpreter>
        <debug config:type="boolean">false</debug>
        <feedback config:type="boolean">true</feedback>
        <source><![CDATA[
touch /mnt/testfile
echo Hi
]]>
        </source>
      </script>
    </postpartitioning-scripts>
  </scripts>
</profile>
```

インストール完了後は、スクリプトと出力されたログファイルが `/var/adm/autoinstall` 内に配置されます。スクリプトは `scripts` サブディレクトリ内、出力されたログは `log` サブディレクトリ内をそれぞれご覧ください。

出力されたログにはスクリプトを実行した際に生成された内容が書かれます。標準出力と標準エラー出力の両方が記録されます。

スクリプトが 0 以外の終了コードで終了した場合、ログ内には警告を示すメッセージが出力されます。これを出力したくない場合は、`feedback` オプションを指定してください。

4.32 システム変数 (sysconfig)

sysconfig リソースを使用することで、sysconfig リポジトリ (`/etc/sysconfig`) 内にある設定変数を直接定義することができます。sysconfig 変数はさまざまなシステムコンポーネントや環境変数を設定することのできる仕組みで、要件に正確に適合させることができるようになります。

下記に sysconfig リソースの設定例を示します。

例 4.67: SYSCONFIG の設定

```
<sysconfig config:type="list" >
  <sysconfig_entry>
    <sysconfig_key>XNTPD_INITIAL_NTPDATE</sysconfig_key>
    <sysconfig_path>/etc/sysconfig/xntp</sysconfig_path>
    <sysconfig_value>ntp.host.com</sysconfig_value>
  </sysconfig_entry>
  <sysconfig_entry>
    <sysconfig_key>HTTP_PROXY</sysconfig_key>
    <sysconfig_path>/etc/sysconfig/proxy</sysconfig_path>
    <sysconfig_value>proxy.host.com:3128</sysconfig_value>
  </sysconfig_entry>
  <sysconfig_entry>
```

```
    <sysconfig_key>FTP_PROXY</sysconfig_key>
    <sysconfig_path>/etc/sysconfig/proxy</sysconfig_path>
    <sysconfig_value>proxy.host.com:3128</sysconfig_value>
  </sysconfig_entry>
</sysconfig>
```

絶対パスと相対パスのどちらでも指定することができます。相対パスを指定した場合は、`/etc/sysconfig` ディレクトリからの相対パスとして扱われます。

4.33 設定ファイルの直接追加

さまざまなアプリケーションやサービスでは、インストール済みのシステム内に設定ファイルを配置することで、設定を行うことができます。たとえば Web サーバをインストールしているような環境では、(`httpd.conf`) のような設定ファイルが存在しています。

このリソースを使用してインストール済みのシステムにおけるパスを指定することで、制御ファイル内に設定ファイルの内容そのものを含めることができるようになります。YaST では、指定された場所にファイルをコピーします。

この機能を使用するには、インストール先のシステムに `autoyast2` パッケージをインストールする必要があります。このパッケージがインストールされていない場合、自動的にインストールを行います。

また、`file_location` を指定することで、ファイルの取得元を指定することもできます。取得元は HTTP サーバのようなネットワークリソースでもかまいません (例: `<file_location>http://my.server.site/issue</file_location>`) 。

このほか、`relurl://` プレフィクスを使用することで、ローカルファイルを指定することもできます (例: `<file_location>relurl://path/to/file.conf</file_location>` 。)

なお、ディレクトリを作成したい場合は、スラッシュで終わる `file_path` を指定してください。

例 4.68: インストール済みのシステムに対するファイルの出力

```
<files config:type="list">
  <file>
    <file_path>/etc/apache2/httpd.conf</file_path>
    <file_contents>

<![CDATA[
some content
]]>

    </file_contents>
  </file>
  <file>
    <file_path>/mydir/a/b/c/</file_path> <!-- create directory -->
  </file>
```

```
</files>
```

下記にはさらに複雑な例を示しています。この設定は `file_contents` 内で入れた内容をファイルに出力し、ファイルのパーミッションと所有権情報を設定してします。ファイルがシステムにコピーされたあと、スクリプトが実行されます。このような仕組みにより、ファイルをクライアントの環境に合わせて修正し、配置することができるようになります。

例 4.69: インストール済みのシステムに対するファイルの出力

```
<files config:type="list">
  <file>
    <file_path>/etc/someconf.conf</file_path>
    <file_contents>

<![CDATA[
some content
]]>

    </file_contents>
    <file_owner>tux.users</file_owner>
    <file_permissions>444</file_permissions>
    <file_script>
      <interpreter>shell</interpreter>
      <source>

<![CDATA[
#!/bin/sh

echo "Testing file scripts" >> /etc/someconf.conf
df
cd /mnt
ls
]]>

      </source>
    </file_script>
  </file>
</files>
```

4.34 インストール時におけるユーザへの値の確認

インストールの際、ユーザに対して制御ファイルに設定すべき値を問い合わせるように設定することもできます。この機能を使用した場合、インストールの途中でポップアップが表示され、ユーザに対して値を問い合わせます。インストールを完全に自動化したいものの、ユーザに対してローカルのアカウントに対するパスワードを尋ねたりしたいような場合は、制御ファイル内に `ask` ディレクティブを設定してください。

下記のような XML 構造内に、必要な要素を追加していきます:

```
<general>
  <ask-list config:type="list">
    <ask>
      ...
    </ask>
  </ask-list>
</general>
```

ユーザへの値確認: XML 表記

`question`

ユーザに対して提示する質問文を指定します。

```
<question>LDAP サーバのアドレスを入力してください</question>
```

既定値は要素のパスそのものです (これではユーザに対する問い合わせになりませんので、question については必ず設定しておくことをお勧めします) 。

`default`

ユーザに対して提示する既定値を指定します。テキスト入力の場合、表示されたタイミングでこの値が入力済みの状態になります。チェックボックスの場合は true もしくは false を指定することで、あらかじめ選択済みの状態か、そうでないかを指定することができます。

```
<default>dc=suse,dc=de</default>
```

任意指定です。

`help`

質問の左側に表示する、任意指定のヘルプテキストを指定します。

```
<help>LDAP サーバのアドレスを入力してください。</help>
```

任意指定です。

`title`

質問の上に表示する任意指定のタイトルを指定します。

```
<title>LDAP サーバ</title>
```

任意指定です。

`type`

入力すべきデータの形式を指定します。指定可能な値は `symbol` (シンボル), `boolean` (ブール値), `string` (文字列), `integer` (整数) のいずれかです。たとえばパーティション内でのファイルシステムの選択であれば `symbol` を、ユーザ設定内でのホームディレクトリの暗号化

可否設定であれば `boolean` になります。制御ファイル内の項目から形式を判断したい場合は、`config:type="...."` の箇所をご確認ください。また、`static_text` という形式を指定することもできます。この場合は、ユーザに対して入力を求めず、ヘルプテキストに含まれていない情報を表示するために使用することができます。

```
<type>symbol</type>
```

任意指定です。既定値は `string` です。`symbol` を指定した場合は、`selection` 要素も設定しなければなりません (下記参照) 。

`password`

このブール値を `true` に設定すると、単純なテキスト入力ではなく、パスワードダイアログを表示するようになります。なお、`type` の値が `string` の場合にのみ効果があります。

```
<password config:type="boolean">true</password>
```

任意指定です。既定値は `false` です。

`pathlist`

`path` (パス) 要素の一覧を指定します。パスはカンマ区切りのリストで、変更したい要素のパスを表すものです。たとえば制御ファイル内でのネットワーク設定の要素は `<networking>` セクション内にありますが、この場合は `networking` のように指定します。

```
<pathlist config:type="list">
  <path>networking,dns,hostname</path>
  <path>...</path>
</pathlist>
```

制御ファイル内に書かれている最初のユーザに対するパスワードを変更したい場合は、パスを `users,0,user_password` のように指定します。ここで `0` は、制御ファイルの対象となるセクション内にある最初の要素を意味します。たとえば下記の <users config:type="list"> の例で言うと、`root` が該当することになります。同様に `1` を指定すると、2 番目の項目を意味することになります。

```
<users config:type="list">
  <user>
    <username>root</username>
    <user_password>password to change</user_password>
    <encrypted config:type="boolean">false</encrypted>
  </user>
  <user>
    <username>tux</username>
    <user_password>password to change</user_password>
    <encrypted config:type="boolean">false</encrypted>
  </user>
</users>
```

上記のような <user> セクションがある場合、`root` のパスワード設定を行う場合は、<pathlist> は下記のようになります:

```
<pathlist config:type="list">
          <path>users,0,user_password</path>
        </pathlist>
```

この情報の設定は任意ですが、`path` もしくは `file` のいずれかを指定すべきです。

`file`

質問に対してユーザが入力した内容をファイルに保存することもできます。このようにすることで、スクリプトから読み込んで使用できるようになります。なお、`stage=initial` を設定した場合で、第 2 ステージで回答を使用したい場合は、`chrooted=false` を指定した chroot 環境スクリプトで、回答が書かれたファイルをコピーする必要があります。たとえば `cp /tmp/my_answer /mnt/tmp/` のように実行します。これは、第 1 ステージでは `/tmp` が RAM ディスク内に存在するため、再起動を行うと消えてしまうためです。このような仕組みから、`/mnt/` 以下にマウントされたインストール先のシステムにコピーを行うことで、この問題を解決することができます。

```
<file>/tmp/answer_hostname</file>
```

この情報の設定は任意ですが、`path` もしくは `file` のいずれかを指定すべきです。

`stage`

質問を表示するインストール時のステージを指定します。ここでは `cont` もしくは `initial` のいずれかを指定することができます。`initial` はインストールの冒頭で表示する意味となり、インストール前スクリプトの直後に表示されます。対する `cont` はシステムを初めて再起動した後に表示されます。なお、ディスク内の制御ファイルに対して回答を書き込みたい場合は、`initial` を指定してください。また、`initial` でパスワードを入力した場合、パスワードは暗号化されずに保存されることにも注意してください。また、`cont` を指定した場合、ファイルシステム関連の質問を表示しても意味がないことにも注意してください。これは、その時点では既にパーティションの設定が完了しているため、制御ファイルを書き換えても効果が無いためです。

```
<stage>cont</stage>
```

任意指定です。既定値は `initial` です。

`selection`

`selection` 要素には `entry` 要素のリストを指定します。それぞれの `entry` 要素は、ユーザ側に提示される選択肢を表します。この場合、テキストボックスには直接値を指定することができません。一覧から値を選択するのみとなります。

```
<selection config:type="list">
  <entry>
```

```
    <value>
        btrfs
    </value>
    <label>
        Btrfs File System
    </label>
  </entry>
  <entry>
    <value>
        ext3
    </value>
    <label>
        Extended3 File System
    </label>
  </entry>
</selection>
```

`type=string` の場合は任意、`type=boolean` の場合は設定不可、`type=symbol` の場合は必須です。

`dialog`

1 つのダイアログ内で複数の質問を表示することもできます。これを行うには、この要素で整数のダイアログ ID を指定してください。同じダイアログ ID の質問は、同じダイアログ内に表示されるようになります。また、ダイアログは ID 順に並べられます。

```
<dialog config:type="integer">3</dialog>
```

任意指定です。

`element`

1 つのダイアログ内で複数の質問を表示することもできます。この場合、`element` で整数の ID を指定する必要があります。ダイアログ内での質問は、この ID 順に並べられます。

```
<element config:type="integer">1</element>
```

任意指定です (`dialog` もご覧ください) 。

`width`

既定のダイアログ幅を広げることができます。ダイアログに対して複数の幅指定が存在する場合は、最も大きいものが使用されます。値は 1 文字分をおおよそ 1 とした幅で指定します。

```
<width config:type="integer">50</width>
```

任意指定です。

`height`

既定のダイアログの高さを広げることができます。ダイアログに対して複数の高さ指定が存在する場合は、最も大きいものが使用されます。値は 1 行分をおおよそ 1 とした高さで指定します。

```
<height config:type="integer">15</height>
```

任意指定です。

`frametitle`

1 つのダイアログ内で複数の質問を表示することもできます。この場合、ダイアログ内の各質問には枠が描かれ、枠タイトルと各質問に対するキャプションが表示されます。また 1 つの枠内に複数の要素を設定することもできます。この場合、同じ枠タイトルを設定する必要があります。

```
<frametitle>User data</frametitle>
```

任意指定です。既定では枠タイトルが設定されません。

`script`

質問に対する回答を入力したあと、スクリプトを実行することができます (スクリプトに関する詳細は 4.34.1項「既定値スクリプト」をお読みください) 。

```
<script>...</script>
```

任意指定です。既定ではスクリプトを動作させません。

`ok_label`

［Ok］ボタンのラベルを変更することもできます。1 つのダイアログに対しては、最後に指定された要素が使用されます。

```
<ok_label>Finish</ok_label>
```

任意指定です。

`back_label`

［Back］(戻る) ボタンのラベルを変更することもできます。1 つのダイアログに対しては、最後に指定された要素が使用されます。

```
<back_label>change values</back_label>
```

任意指定です。

`timeout`

ここではタイムアウトを秒単位で指定することができます。ここで指定した時間内に何も回答を行わなかった場合、既定値が回答されたものとして扱われます。なお、ダイアログ内のウイジェットに触れたり、何らかの変更を行ったりした場合は、タイムアウトの設定が解除され、［Ok］を押すまで待機するようになります。

```
<timeout config:type="integer">30</timeout>
```

任意指定です。値を指定しない場合は `0` として扱われ、タイムアウトの設定がないものとして処理されます。

`default_value_script`

質問に対する既定値を設定するために、スクリプトを動作させることができます (詳しい手順については 4.34.1項「既定値スクリプト」をお読みください) 。この機能は、何らかの理由で既定値を 計算 しなければならないような場合に有用で、`timeout` オプションと併用して使用します。

```
<default_value_script>...</default_value_script>
```

任意指定です。既定ではスクリプトを動作させません。

4.34.1 既定値スクリプト

質問に対する既定値を設定するために、スクリプトを動作させることができます。この機能は、何らかの理由で既定値を 計算 しなければならないような場合に有用で、`timeout` オプションと併用して使用します。

スクリプトは XML 構造内の要素として配置します。具体的には 4.31.6項「スクリプトの XML パラメータ」のようになります:

```
<general>
  <ask-list config:type="list">
    <ask>
      <default_value_script>
        ...
      </default_value_script>
    </ask>
  </ask-list>
</general>
```

`echo` コマンドで STDOUT (標準出力) に出力することで、ダイアログに対する既定値を設定することができます。なお、スクリプトの終了コードが 0 以外であった場合、通常の既定値が設定されます。なお、`\n` を省略するために `echo -n` を使用する必要がある点と、出力する値が適切なものでなければならない点、そしてブール値の場合は「okay」等を使用すべきではない点 (代わりに「true」を出力してください) にそれぞれ注意してください。

4.34.2 スクリプト

質問に対する回答が表示された後にスクリプトを実行することができます。

下記のような XML 構造内に、必要な要素を追加していきます:

```
<general>
```

```
  <ask-list config:type="list">
    <ask>
      <script>
        ...
      </script>
    </ask>
  </ask-list>
</general>
```

4.31.6項「スクリプトの XML パラメータ」内に示した要素以外にも、`<ask>` 要素内のスクリプトには下記を指定することができます:

スクリプト: XML 表記

`filename`

スクリプトのファイル名を指定します。

```
<filename>my_ask_script.sh</filename>
```

既定値は ask_script.sh です。

`environment`

スクリプトに対して、環境変数の形で問い合わせの回答を設定するかをブール値で指定します。設定される環境変数は `VAL` になります。

```
<environment config:type="boolean">true</environment>
```

任意指定です。既定値は `false` です。

`feedback`

スクリプトの実行時にフィードバックを表示するかどうかを指定します。STDOUT に何らかの値を出力すると、スクリプトの実行終了後にポップアップメッセージが表示されるようになります。

```
<feedback config:type="boolean">true</feedback>
```

任意指定です。既定値は `false` です。

`rerun_on_error`

スクリプトの終了コードが 0 (ゼロ) になるまでダイアログを開いたままにします。この機能を使用することで、ユーザからの入力を検証することができます。なお、スクリプト側では適切なエラーメッセージを表示するようにしてください。また、スクリプトからのエラーメッセージを表示するには、`feedback` オプションを `true` に設定する必要があることにも注意してください。このオプションの指定は任意で、既定値は `false` です。

なお、スクリプト内で `/tmp/next_dialog` ファイルを作成することもできます。このファイル内には、次に表示すべきダイアログの ID を指定します。-1 を書き込むと、この流れを終了することができます。

下記に `ask` 機能の使用例を示します。

```
<general>
  <ask-list config:type="list">
    <ask>
      <pathlist config:type="list">
        <path>ldap,ldap_server</path>
      </pathlist>
      <stage>cont</stage>
      <help>Choose your server depending on your department</help>
      <selection config:type="list">
        <entry>
          <value>ldap1.mydom.de</value>
          <label>LDAP for development</label>
        </entry>
        <entry>
          <value>ldap2.mydom.de</value>
          <label>LDAP for sales</label>
        </entry>
      </selection>
      <default>ldap2.mydom.de</default>
      <default_value_script>
        <source> <![CDATA[
echo -n "ldap1.mydom.de"
]]>
        </source>
      </default_value_script>
    </ask>
    <ask>
      <pathlist config:type="list">
        <path>networking,dns,hostname</path>
      </pathlist>
      <question>Enter Hostname</question>
      <stage>initial</stage>
      <default>enter your hostname here</default>
    </ask>
    <ask>
      <pathlist config:type="list">
        <path>partitioning,0,partitions,0,filesystem</path>
      </pathlist>
      <question>File System</question>
      <type>symbol</type>
      <selection config:type="list">
        <entry>
          <value config:type="symbol">ext4</value>
          <label>default File System (recommended)</label>
        </entry>
        <entry>
          <value config:type="symbol">ext3</value>
          <label>Fallback File System</label>
```

```
        </entry>
      </selection>
    </ask>
  </ask-list>
</general>
```

下記の例は AutoYaST の制御ファイルを選択させる場合の例です。AutoYaST は問い合わせダイアログの動作完了後に `modified.xml` を読み込みなおしますので、これにより制御ファイルを全く新しくすることができます。

```
<general>
  <ask-list config:type="list">
    <ask>
      <selection config:type="list">
        <entry>
          <value>part1.xml</value>
          <label>Simple partitioning</label>
        </entry>
        <entry>
          <value>part2.xml</value>
          <label>encrypted /tmp</label>
        </entry>
        <entry>
          <value>part3.xml</value>
          <label>LVM</label>
        </entry>
      </selection>
      <title>XML Profile</title>
      <question>Choose a profile</question>
      <stage>initial</stage>
      <default>part1.xml</default>
      <script>
        <filename>fetch.sh</filename>
        <environment config:type="boolean">true</environment>
        <source>
<![CDATA[
wget http://10.10.0.162/$VAL -O /tmp/profile/modified.xml 2>/dev/null
]]>
        </source>
        <debug config:type="boolean">false</debug>
        <feedback config:type="boolean">false</feedback>
      </script>
    </ask>tion>
  </ask-list>
</general>
```

下記のようなスクリプトを指定することで、質問に対する回答をチェックすることができます:

```
<general>
  <ask-list config:type="list">
```

```
    <ask>
      <script>
        <filename>my.sh</filename>
        <rerun_on_error config:type="boolean">true</rerun_on_error>
        <environment config:type="boolean">true</environment>
        <source><![CDATA[
if [ "$VAL" = "myhost" ]; then
    echo "Illegal Hostname!";
    exit 1;
fi
exit 0
]]>
        </source>
        <debug config:type="boolean">false</debug>
        <feedback config:type="boolean">true</feedback>
      </script>
      <dialog config:type="integer">0</dialog>
      <element config:type="integer">0</element>
      <pathlist config:type="list">
        <path>networking,dns,hostname</path>
      </pathlist>
      <question>Enter Hostname</question>
      <default>enter your hostname here</default>
    </ask>
  </ask-list>
</general>
```

4.35 カーネルダンプ



注記: 提供可否について

この機能は S/390 アーキテクチャでのみ利用可能です。

Kdump を使用することで、カーネル全体がクラッシュしてしまった際にクラッシュダンプファイルを作成することができるようになります。クラッシュダンプファイルには、システムがクラッシュした時点でのメモリ内容が含まれます。クラッシュダンプファイルはコアファイルとも呼ばれ、(カーネルの) 開発者がクラッシュの原因を調査する際に使用します。Kdump はクラッシュの再現が難しいものの、問題の修正が重要となるサーバ用途で特に有用です。

なお、Kdump にはマイナス面もあります。Kdump を有効化すると、64 MB から 128 MB 程度の追加のメモリが Kdump 向けに予約されます。この領域を利用してダンプファイルを動作させるためです。

本章では、AutoYaST で Kdump を設定する際の説明のみを行っています。Kdump の動作については、kdump(7) のマニュアルページなどをお読みください。

下記の例では、一般的な Kdump 設定を行っています。

例 4.70: KDUMP の設定

```
<kdump>
  <!-- メモリ予約 -->
  <add_crash_kernel config:type="boolean">true</add_crash_kernel>
  <crash_kernel>256M-:64M</crash_kernel>
  <general>

    <!-- ダンプ先の設定 -->
    <KDUMP_SAVEDIR>ftp://stravinsky.suse.de/incoming/dumps</KDUMP_SAVEDIR>
    <KDUMP_FREE_DISK_SIZE>64</KDUMP_FREE_DISK_SIZE>
    <KDUMP_KEEP_OLD_DUMPS>5</KDUMP_KEEP_OLD_DUMPS>

    <!-- フィルタと圧縮 -->
    <KDUMP_DUMPFORMAT>compressed</KDUMP_DUMPFORMAT>
    <KDUMP_DUMPLEVEL>1</KDUMP_DUMPLEVEL>

    <!-- 通知 -->
    <KDUMP_NOTIFICATION_TO>tux@example.com</KDUMP_NOTIFICATION_TO>
    <KDUMP_NOTIFICATION_CC>spam@example.com devnull@example.com</KDUMP_NOTIFICATION_CC>
    <KDUMP_SMTP_SERVER>mail.example.com</KDUMP_SMTP_SERVER>
    <KDUMP_SMTP_USER></KDUMP_SMTP_USER>
    <KDUMP_SMTP_PASSWORD></KDUMP_SMTP_PASSWORD>

    <!-- kdump カーネル -->
    <KDUMP_KERNELVER></KDUMP_KERNELVER>
    <KDUMP_COMMANDLINE></KDUMP_COMMANDLINE>
    <KDUMP_COMMANDLINE_APPEND></KDUMP_COMMANDLINE_APPEND>

    <!-- 熟練者向け設定 -->
    <KDUMP_IMMEDIATE_REBOOT>yes</KDUMP_IMMEDIATE_REBOOT>
    <KDUMP_VERBOSE>15</KDUMP_VERBOSE>
    <KEXEC_OPTIONS></KEXEC_OPTIONS>
  </general>
</kdump>
```

Kdump は既定で有効化されます。下記では Kdump の無効化方法を説明しています。

例 4.71: KDUMP を無効化するための設定

```
<kdump>
  <add_crash_kernel config:type="boolean">false</add_crash_kernel>
</kdump>
```

4.35.1 メモリ予約

Kdump ではまず、起動時にメモリを予約する処理を行います。このメモリは起動時のかなり早い段階で予約しなければならないものであることから、設定はカーネルのコマンドラインパラメータの `crashkernel` で行います。予約されたメモリは、通常使用しているカーネルがクラッシュした際に呼び出される「第二のカーネル」で使用されます。第二のカーネルには特殊な initrd が設定され、ここにはネットワークやディスクにダンプを保存する機能やメールを送信する機能、そして最後にシステムの再起動を行う機能がそれぞれ用意されています。

Kdump でメモリを予約するには、**メモリ量** (`64M` のように指定すると、64MB のメモリを予約する意味になります) と **オフセット値** を指定します。これらをあわせて `crashkernel=メモリ量@オフセット値` のように指定します。カーネルは適切なオフセット値を検出することができます (ただし、Xen ハイパーバイザを使用している場合は例外で、オフセット値に `16M` を指定する必要があります) 。メモリ量はお使いのアーキテクチャとメインメモリの量によって異なります。

システムのメモリ量に応じて予約するメモリ量を変更するようにするため、複雑なコマンドライン文法を使用することもできます。これは AutoYaST の制御ファイル 1 つで複数の環境に対応するような場合に有用であるほか、マシンのメインメモリを増やしたり減らしたりするような環境で使用することができます。書式は下記のとおりです:

```
範囲開始_1-範囲終了_1:メモリ量_1,範囲開始_2-範囲終了_2:メモリ量_2@オフセット
```

`範囲開始_1` には 1 つ目のシステムのメモリ範囲開始 (例: `0M`) を、`範囲終了_1` には 1 つ目のメモリ範囲終了をそれぞれ指定します。2 つ目以降も同様です。なお、範囲終了の値を省略すると、**無限大** の意味になります。たとえば `256M-2G:64M,2G-:128M` のように指定すると、システムのメモリ量が 256 MB から 2GB までの範囲であれば 64MB 分の予約が、それより大きければ 128MB 分の予約が行われる意味になります。

それとは別に、`crashkernel` パラメータに対して複数の値を設定することもできます。たとえば低メモリと高メモリで別々のセグメントを予約する必要がある場合は、`72M,low` と `256M,high` のように指定します。

例 4.72: 複数の値を設定した KDUMP メモリ予約

```
<kdump>
  <!-- メモリ予約（高/低） -->
  <add_crash_kernel config:type="boolean">true</add_crash_kernel>
  <crash_kernel config:type="list">
    <listentry>72M,low</listentry>
    <listentry>256M,high</listentry>
  </crash_kernel>
</kdump>
```

下記の一覧には、メモリ予約を行う際に必要な設定を示しています:

KDUMP メモリ予約設定: XML 表記

`add_crash_kernel`
: メモリを予約して Kdump を有効化する場合、`true` を指定します。

```
<add_crash_kernel config:type="boolean">true</add_crash_kernel>
```

必須

`crash_kernel`
: 上述の crashkernel コマンドラインの書式で設定を行います。

```
<crash_kernel>256M:64M</crash_kernel>
```

値のリストを指定することもできます。

```
<crash_kernel config:type="list">
  <listentry>72M,low</listentry>
  <listentry>256M,high</listentry>
</crash_kernel>
```

必須

4.35.2 ダンプの保存

本章では、クラッシュダンプの保存場所と保存方法を説明しています。

4.35.2.1 ターゲット

`KDUMP_SAVEDIR` では、ダンプの保存先 URL を指定します。指定可能なプロトコルは下記のとおりです:

- `file`: ローカルディスクへの保存
- `ftp`: FTP サーバへの保存 (暗号化無し)
- `sftp`: SSH2 SFTP サーバへの保存
- `nfs`: NFS 共有内への保存
- `cifs`: Samba もしくは Microsoft Windows が提供する CIFS/SMB 共有への保存

詳細は kdump(5) のマニュアルページをお読みください。たとえば `file:///var/crash` のように指定すると、FHS に準拠した既定の場所に保存することができますし、`ftp://user:password@host:port/incoming/dumps` のように指定することもできます。なお、指定したディレクトリ以下に名前とタイムスタンプを含むサブディレクトリが作成され、そのサブディレクトリ内にダンプが保存されます。

ダンプをローカルのディスクに保存する場合は、`KDUMP_KEEP_OLD_DUMPS` を指定することで、古いダンプを自動削除することができます。ここで指定する値は、保存しておくべきダンプ数になります。また、`KDUMP_FREE_DISK_SIZE` で指定した値よりも空き容量が少ない場合、ダンプを保存しないように設定することもできます。

4.35.2.2 フィルタリングと圧縮

カーネルのダンプは通常圧縮されておらず、フィルタもされていません。そのため、搭載されているメモリの分だけ巨大なファイルになります。より小さいファイルにしたい場合は、ダンプファイルを圧縮するように設定してください。なお、ダンプファイルを開く際には、あらかじめ展開する必要が生じるようになります。

crash(8) デバッグツールで使用できるように各ページを圧縮し、動的な展開を行うようページ圧縮を設定したい場合は、`KDUMP_DUMPFORMAT` に `compressed` (既定値) を指定します。

ゼロで埋められているページなど、全てのメモリページを保存したくない場合もあります。このようにダンプをフィルタしたい場合は、`KDUMP_DUMPLEVEL` を指定します。0 を指定すると完全なダンプを、31 を指定すると最も小さなダンプを生成することになります。それぞれの値の意味と保存されるページについて、詳しくは kdump(5) と makedumpfile(8) のマニュアルページをお読みください。

4.35.2.3 概要

ダンプターゲットの設定: XML 表記

`KDUMP_SAVEDIR`

ダンプと関連ファイルを保存する先を URL で指定します。

```
<KDUMP_SAVEDIR>file:///var/crash/</KDUMP_SAVEDIR>
```

必須

`KDUMP_FREE_DISK_SIZE`

ダンプを保存した後に残しておかなければならない空き容量を、メガバイト単位で指定します。十分な空き容量が存在しない場合、ダンプは保存されなくなります。

```
<KDUMP_FREE_DISK_SIZE>64</KDUMP_FREE_DISK_SIZE>
```

任意指定

KDUMP_KEEP_OLD_DUMPS

KDUMP_SAVEDIR がローカルのディレクトリを指し示している場合、削除せず残しておくべきダンプ数を指定します。0 を指定するとダンプの削除を無効化します。-1 を指定すると、最新のものを除いて全て削除するようになります。

```
<KDUMP_KEEP_OLD_DUMPS>4</KDUMP_KEEP_OLD_DUMPS>
```

任意指定

4.35.3 電子メール通知

マシンがクラッシュしてダンプが保存された際、電子メールによる通知を受け取りたい場合の設定です。

Kdump は initrd 内で動作する仕組みであることから、ローカルのメールサーバはメールを送信することができません。SMTP サーバを指定して送信する必要があります (下記参照) 。

なお、KDUMP_NOTIFICATION_TO には宛先のアドレスを 1 つだけ指定する必要があります。複数のアドレスを指定したい場合は、KDUMP_NOTIFICATION_CC に設定してください。なお、いずれの設定でも、電子メールアドレスだけを指定するものとし、名前などは含めないでください。

KDUMP_SMTP_SERVER のほか、メールサーバ側で認証を行う必要がある場合は、KDUMP_SMTP_USER と KDUMP_SMTP_PASSWORD も指定してください。TLS/SSL への対応は提供されていませんが、将来的に追加される予定です。

電子メール通知の設定: XML 表記

KDUMP_NOTIFICATION_TO

電子メールの送信先となるメールアドレスを 1 つだけ指定します。追加の受信者を設定したい場合は、KDUMP_NOTIFICATION_CC で設定してください。

```
<KDUMP_NOTIFICATION_TO
>tux@example.com</KDUMP_NOTIFICATION_TO>
```

任意指定 (何も指定しない場合、通知が無効化されます)

KDUMP_NOTIFICATION_CC

通知メールのコピー送信先となるメールアドレスを任意の数 (0 個以上) だけ指定します。

```
<KDUMP_NOTIFICATION_CC
```

```
>wilber@example.com suzanne@example.com</KDUMP_NOTIFICATION_CC>
```

任意指定

KDUMP_SMTP_SERVER

メールの配送に使用する SMTP サーバのホスト名を指定します。SMTP 認証を設定する必要である場合は、KDUMP_SMTP_USER と KDUMP_SMTP_PASSWORD を指定してください。TLS/SSL には未対応です。

```
<KDUMP_SMTP_SERVER>email.suse.de</KDUMP_SMTP_SERVER>
```

任意指定 (何も指定しない場合、通知が無効化されます)

KDUMP_SMTP_USER

SMTP 認証で使用するユーザ名を指定します。KDUMP_SMTP_PASSWORD についても設定を行ってください。

```
<KDUMP_SMTP_USER>bwalle</KDUMP_SMTP_USER>
```

任意指定

KDUMP_SMTP_PASSWORD

SMTP 認証で使用するパスワードを指定します。KDUMP_SMTP_USER についても設定を行ってください。

```
<KDUMP_SMTP_PASSWORD>geheim</KDUMP_SMTP_PASSWORD>
```

任意指定

4.35.4 Kdump カーネル設定

上述のとおり、ダンプを保存する際には特別なカーネルを起動します。どのカーネルを使用するのかを自動検出する仕組みを使用したくない場合 (詳しくは kdump(5) のマニュアルページをお読みください) は、KDUMP_KERNELVER でカーネルのバージョンを指定することができます。たとえば、この値を foo に設定すると、/boot/vmlinuz-foo もしくは /boot/vmlinux-foo (vmlinuz ファイルが存在するプラットフォームの場合の順序) を使用するようになります。

また、Kdump カーネルの起動する際のコマンドラインも設定することができます。通常は起動時のコマンドラインから、Kdump には無関係な設定 (たとえば crashkernel パラメータなど) を取り除き、Kdump で必要な設定 (詳しくは kdump(5) のマニュアルページをお読みください) を追加して設定することになります。追加のパラメータを指定したい場合は、KDUMP_COMMANDLINE_APPEND を指定してください。コマンドライン全体を設定したい場合は、KDUMP_COMMANDLINE を指定してください (この場合、必要なパラメータ全てを設定しなければならないことに注意してください) 。

カーネル設定: XML 表記

KDUMP_KERNELVER

Kdump で使用するカーネルのバージョン文字列を指定します。自動検出の仕組みを使用する場合は、指定しないでください (自動検出のほうを強くお勧めします) 。

```
<KDUMP_KERNELVER>6.4.0-default</KDUMP_KERNELVER>
```

任意指定 (何も指定しなければ自動検出)

KDUMP_COMMANDLINE_APPEND

Kdump カーネルに渡す追加のコマンドラインパラメータ。

```
<KDUMP_COMMANDLINE_APPEND>console=ttyS0,57600</KDUMP_COMMANDLINE_APPEND>
```

任意指定

KDUMP_Command Line

Kdump のコマンドラインとして生成されたものを上書きするための設定です。注意してお使いください。通常は KDUMP_COMMANDLINE_APPEND のほうを指定します。

```
<KDUMP_COMMANDLINE_APPEND>root=/dev/sda5 nr_cpus=1 irqpoll</KDUMP_COMMANDLINE>
```

任意指定

4.35.5 熟練者向け設定

熟練者向け設定: XML 表記

KDUMP_IMMEDIATE_REBOOT

true を指定すると、ダンプの保存完了後に自動的に再起動を行うようになります。false を指定すると再起動を行わなくなります。既定では自動的に再起動します。

```
<KDUMP_IMMEDIATE_REBOOT>true</KDUMP_IMMEDIATE_REBOOT>
```

任意指定

KDUMP_VERBOSE

Kdump の冗長出力度をビットマスクで指定します。詳しくは kdump(5) をお読みください。

```
<KDUMP_VERBOSE>3</KDUMP_VERBOSE>
```

任意指定

KEXEC_OPTIONS

Kdump カーネルを読み込む際に kexec に渡す追加のオプションを指定します。通常は何も指定しません。

```
<KEXEC_OPTIONS>--noio</KEXEC_OPTIONS>
```

任意指定

4.36 DNS サーバ

`dns-server` リソースを指定することで、bind DNS サーバを設定することができます。下記に示す 3 種類の直感的なプロパティでは、1 が有効、0 が無効を表しています。

属性	値	説明
`chroot`	0 / 1	DNS サーバを chroot 内に閉じこめておくかどうか。
`start_service`	0 / 1	bind を有効にするかどうか (システムの起動時に開始するかどうか) 。
`use_ldap`	0 / 1	直接的な設定ファイルではなく、LDAP 内に設定を保存するかどうか。

例 4.73: 基本的な DNS サーバの設定

```
<dns-server>
  <chroot>0</chroot>
  <start_service>1</start_service>
  <use_ldap>0</use_ldap>
</dns-server>
```

これらの基本的な設定のほかにも、下記のリストに示す 3 種類の追加プロパティが提供されています。下記はいずれも、サービスの設定を細かく調整するためのものです。

リスト	説明
`logging`	DNS サーバのログ機能に関するオプションです。
`options`	使用すべきファイルやディレクトリ、フォワーダの一覧やその他の設定に関するオプションです。

リスト	説明
zones	サーバ側に設定する DNS ゾーンとそれらの設定、およびレコード類のオプションです。

例 4.74: DNS サーバのゾーン設定と高度な設定

```
<dns-server>
  <logging config:type="list">
    <listentry>
      <key>channel</key>
      <value>log_syslog { syslog; }</value>
    </listentry>
  </logging>
  <options config:type="list">
    <option>
      <key>forwarders</key>
      <value>{ 10.10.0.1; }</value>
    </option>
  </options>
  <zones config:type="list">
    <listentry>
      <is_new>1</is_new>
      <modified>1</modified>
      <options config:type="list"/>
      <records config:type="list">
        <listentry>
          <key>mydom.uwe.</key>
          <type>MX</type>
          <value>0 mail.mydom.uwe.</value>
        </listentry>
        <listentry>
          <key>mydom.uwe.</key>
          <type>NS</type>
          <value>ns.mydom.uwe.</value>
        </listentry>
      </records>
      <soa>
        <expiry>1w</expiry>
        <mail>root.aaa.aaa.cc.</mail>
        <minimum>1d</minimum>
        <refresh>3h</refresh>
        <retry>1h</retry>
        <serial>2005082300</serial>
        <server>aaa.aaa.cc.</server>
        <zone>@</zone>
      </soa>
      <soa_modified>1</soa_modified>
      <ttl>2d</ttl>
```

```
      <type>master</type>
      <update_actions config:type="list">
        <listentry>
          <key>mydom.uwe.</key>
          <operation>add</operation>
          <type>NS</type>
          <value>ns.mydom.uwe.</value>
        </listentry>
      </update_actions>
      <zone>mydom.uwe</zone>
    </listentry>
  </zones>
</dns-server>
```

4.37 DHCP サーバ

`dhcp-server` リソースを使用することで、DHCP サーバの全ての設定を行うことができます。下記の 6 種類のプロパティをお使いください。

要素	値	説明
`chroot`	0 / 1	値に 1 を指定すると、DHCP サーバを chroot 環境内に閉じこめて動作するようになります。
`start_service`	0 / 1	値に 1 を指定すると、DHCP サーバを有効化します (システムの起動時に開始するようにします) 。
`use_ldap`	0 / 1	値に 1 を指定すると、設定を設定ファイルではなく LDAP に保存するようになります。
`other_options`	テキスト	DHCP サーバの起動時に、実行ファイルに渡すコマンドラインパラメータを文字列で指定します。たとえば "-p 1234" のように指定すると、非標準の 1234 ポートでサービスを待ち受けるようになります。設定可能なオプションの一覧については、

要素	値	説明
		dhcpd のマニュアルページをお読みください。何も指定しない場合、既定値が適用されます。
`allowed_interfaces`	リスト	DHCP サーバのサービスを提供するネットワークカードのリストを指定します。設定方法については下記の例をご覧ください。
`settings`	リスト	DHCP サーバの動作を設定するための設定リストを指定します。設定はツリー構造で、ルート要素がグローバルオプションを表し、そこからサブネットやホストを定義する形になっています。また、`children`，`parent_id`，`parent_type` の各プロパティを指定することで、入れ子になった構造を表します。正確な書式については下記の例をご覧ください。

例 4.75: DHCP-SERVER セクションの例

```
<dhcp-server>
  <allowed_interfaces config:type="list">
    <allowed_interface>eth0</allowed_interface>
  </allowed_interfaces>
  <chroot>0</chroot>
  <other_options>-p 9000</other_options>
  <start_service>1</start_service>
  <use_ldap>0</use_ldap>

  <settings config:type="list">
    <settings_entry>
      <children config:type="list"/>
      <directives config:type="list">
        <listentry>
          <key>fixed-address</key>
          <type>directive</type>
```

```
          <value>192.168.0.10</value>
        </listentry>
        <listentry>
          <key>hardware</key>
          <type>directive</type>
          <value>ethernet d4:00:00:bf:00:00</value>
        </listentry>
      </directives>
      <id>static10</id>
      <options config:type="list"/>
      <parent_id>192.168.0.0 netmask 255.255.255.0</parent_id>
      <parent_type>subnet</parent_type>
      <type>host</type>
    </settings_entry>
    <settings_entry>
      <children config:type="list">
        <child>
          <id>static10</id>
          <type>host</type>
        </child>
      </children>
      <directives config:type="list">
        <listentry>
          <key>range</key>
          <type>directive</type>
          <value>dynamic-bootp 192.168.0.100 192.168.0.150</value>
        </listentry>
        <listentry>
          <key>default-lease-time</key>
          <type>directive</type>
          <value>14400</value>
        </listentry>
        <listentry>
          <key>max-lease-time</key>
          <type>directive</type>
          <value>86400</value>
        </listentry>
      </directives>
      <id>192.168.0.0 netmask 255.255.255.0</id>
      <options config:type="list"/>
      <parent_id/>
      <parent_type/>
      <type>subnet</type>
    </settings_entry>
    <settings_entry>
      <children config:type="list">
        <child>
          <id>192.168.0.0 netmask 255.255.255.0</id>
          <type>subnet</type>
        </child>
      </children>
```

```
      <directives config:type="list">
        <listentry>
          <key>ddns-update-style</key>
          <type>directive</type>
          <value>none</value>
        </listentry>
        <listentry>
          <key>default-lease-time</key>
          <type>directive</type>
          <value>14400</value>
        </listentry>
      </directives>
      <id/>
      <options config:type="list"/>
      <parent_id/>
      <parent_type/>
      <type/>
    </settings_entry>
  </settings>
</dhcp-server>
```

4.38 ファイアウオール設定

openSUSE Leap 15.0 およびそれ以降のバージョンでは、SuSEfirewall2 が `firewalld` に置き換えられています。SuSEfirewall2 プロパティを使用していたプロファイルについては、`firewalld` のプロファイル形式に変換されます。ただし、全てのプロファイル設定が変換されるわけではありません。

重要: SuSEFirewall2 を利用したプロファイルとの後方互換性の制限について

SuSEfirewall2 ベースのプロファイルの使用は、多くのオプションが `firewalld` で使用できなくなっていることから、部分的なサポートにとどまっています。そのため、移行できなかった設定によって、ネットワークセキュリティに影響がある場合があります。

4.38.1 一般的なファイアウオール設定

`firewalld` では、一般的な設定は少数のプロパティでのみ提供されていて、ほとんどの設定はゾーン内で行うようになっています。

属性	値	説明
`start_firewall`	ブール値	設定を適用したあと、`firewalld` をすぐに起動するかどうかを指定します。
`enable_firewall`	ブール値	`firewalld` をシステムの起動時に開始するかどうかを指定します。
`default_zone`	ゾーン名	特にゾーンを割り当てない場合に使用する既定のゾーンを指定します。
`log_denied_packets`	ログに記録すべきパケットの種類	ドロップしたパケットのうち、ログに記録すべきものを選択します。値: `off` (無し), `unicast` (ユニキャスト), `multicast` (マルチキャスト), `broadcast` (ブロードキャスト), `all` (全て)
`name`	ゾーンの識別子	ゾーンを識別するために使用するものです。ゾーンが既知のものではない場合、新しいゾーンを作成します。
`short`	ゾーンに対する短い説明	ゾーンの目的を大まかにまとめたものです。既に存在するゾーンの場合は無視されます。何も指定しない場合、名前をそのまま説明に適用します。
`description`	ゾーンの説明	ゾーンの目的を詳しく説明したものです。既に存在するゾーンの場合は無視されます。何も指定しない場合、名前をそのまま説明に適用します。

属性	値	説明
`target`	既定のアクション	どのルールにも該当しない場合に取るべきアクションを指定します。設定可能な値は `ACCEPT` , `%%REJECT%%` , `DROP` , `default` のいずれかです。何も指定しない場合は `default` が指定されたものとみなされます。値の意味についての詳細は、https://firewalld.org/documentation/zone/options.html をお読みください。

4.38.2 ファイアウオールのゾーン設定

`firewalld` の設定はゾーンの仕組みをベースにしています。ゾーンとは接続やインターフェイス、接続元アドレスに適用する信頼レベルを意味するもので、各ゾーンの動作はいくつかの方法で制御することができるものの、全てのプロパティを設定できるというわけではありません。

属性	値	説明
`interfaces`	インターフェイス名のリスト	このゾーンに割り当てるインターフェイスの一覧を指定します。インターフェイスもしくはソースのみをゾーン内に設定することができます。
`services`	サービスの一覧	このゾーンで許可するサービスの一覧を指定します。
`ports`	ポートの一覧	このゾーンで開くポート、もしくはポート範囲の一覧を指定します。

属性	値	説明
protocols	プロトコルの一覧	このゾーンで開く、もしくはアクセスできるようにするプロトコルの一覧を指定します。
masquerade	マスカレード処理の可否	このゾーンに対してネットワークアドレス変換 ((NAT)) を有効化するかどうかを指定します。

4.38.3 プロファイル内にある firewalld 設定が適用されるステージ

openSUSE Leap 15 .3 およびそれ以降のバージョンでは、firewalld のプロファイルは、通常はインストールの第 1 ステージの終了時に適用されるようになっています (インストールのステージについては 1.2項「概要と考え方」をお読みください) 。ただし、場合によっては第 2 ステージで適用される場合もあります。下記に firewalld のプロファイルが適用されるステージの条件を示します:

- AutoYaST で firewalld セクションを設定している場合で、かつ openSUSE Leap を SSH や VNC 経由でインストールしていない場合、ファイアウオールは第 1 ステージで設定されます。
- AutoYaST で firewalld セクションを設定している場合で、かつ openSUSE Leap を SSH や VNC 経由でインストールしていて第 2 ステージが必要ない場合、ファイアウオールは第 1 ステージで設定されます。
- AutoYaST で firewalld セクションを設定している場合で、かつ openSUSE Leap を SSH や VNC 経由でインストールしていて第 2 ステージが必要な場合、ファイアウオールは第 2 ステージで設定されます。
- AutoYaST で firewalld セクションを設定していない場合は、ファイアウオールは既定の製品の提案に従って第 1 ステージで設定されます。
- AutoYaST で firewalld セクションが設定されていてもいなくても、ネットワークアクセスが必要なカスタムスクリプトが存在する場合は、プロファイルや製品の提案に従って第 1 ステージ内でファイアウオールが設定されます。ただし、スクリプトが必要とするネットワークへのアクセスを設定するため、ファイアウオールの設定を調整しなければなりません。

4.38.4 完全な例

`firewall` セクションの例です。ここには一般的なプロパティのほか、ゾーン固有のプロパティも書かれています。

例 4.76: FIREWALL セクションの例

```
<firewall>
  <enable_firewall config:type="boolean">true</enable_firewall>
  <log_denied_packets>all</log_denied_packets>
  <default_zone>external</default_zone>
  <zones config:type="list">
    <zone>
      <name>public</name>
      <interfaces config:type="list">
        <interface>eth0</interface>
      </interfaces>
      <services config:type="list">
        <service>ssh</service>
        <service>dhcp</service>
        <service>dhcpv6</service>
        <service>samba</service>
        <service>vnc-server</service>
      </services>
      <ports config:type="list">
        <port>21/udp</port>
        <port>22/udp</port>
        <port>80/tcp</port>
        <port>443/tcp</port>
        <port>8080/tcp</port>
      </ports>
    </zone>
    <zone>
      <name>dmz</name>
      <interfaces config:type="list">
        <interface>eth1</interface>
      </interfaces>
    </zone>
  </zones>
</firewall>
```

4.39 その他のハードウエア／システムコンポーネント

AutoYaST では、ネットワーク認証やセキュリティなどの中枢コンポーネントの設定に加えて、さまざまなハードウエアやシステムの設定オプションを提供しています。この仕組みは対話的な手段でシステムを手作業でインストールし、設定する場合と同じ機能となります。具体的には、YaST で提供されているプリンタやサウンドデバイス、TV カードやその他のハードウエアコンポーネントなどを設定することができます。

また、YaST 側に新しい設定オプションが追加されると、AutoYaST 側でも利用できるようになります。

4.39.1 プリンタ

AutoYaST でのプリンタのサポートは、ネットワーク経由で印刷を行う場合、それを CUPS でどのように取り扱うかといった、基本的な設定項目に限られます。

なお、AutoYaST ではローカルの印刷キューを設定する機能は用意されていません。新しいプリンタであれば通常は USB 経由で接続しますが、CUPS からは `usb://ACME/FunPrinter?serial=1a2b3c` 等のように、型式ごとに異なる URI を使用することになります。この URI は CUPS のバックエンドである `usb` が検出の際に設定するものであることから、あらかじめ USB のデバイス URI を判断するのは難しいことになります。このような背景から、AutoYaST ではローカルの印刷キューを設定する機能は提供されていません。

ネットワーク経由で印刷を行う CUPS のクライアント側は、下記のような構造になっています:

クライアント側のワークステーションでは、アプリケーションプログラムが印刷キューを CUPS デーモンプロセス (`cupsd`) に送信します。`cupsd` は印刷ジョブを、実際に処理を行う CUPS 印刷サーバに転送します。CUPS 印刷サーバは、プリンタ固有のデータをプリンタデバイスに送信します。

ネットワーク内に CUPS 印刷サーバが 1 台だけ存在するような場合は、各クライアント側のワークステーションで CUPS デーモンを動作させる必要はありません。その代わりに、CUPS サーバの場所を `/etc/cups/client.conf` ファイルで直接指定してアクセスするように設定します (この設定ファイルには CUPS サーバを 1 つしか設定できません) 。この場合、クライアント側のアプリケーションプログラムは、指定した CUPS 印刷サーバに対して、直接ジョブを送信することになります。

例4.77「プリンタの設定」には `printer` セクションの構造を示していますが、この中の `cupsd_conf_content` には、`cupsd` の設定ファイル `/etc/cups/cupsd.conf` の内容全体を記述します。また `client_conf_content` には同様に、`/etc/cups/client.conf` の内容全体を記述します。なお、`printer` セクションには `cupsd` の設定を記述しますが、cupsd を動作させるべきかどうかについては指定しません。

例 4.77: プリンタの設定

```
  <printer>
```

```
    <client_conf_content>
      <file_contents><![CDATA[
... verbatim content of /etc/cups/client.conf ...
]]></file_contents>
    </client_conf_content>
    <cupsd_conf_content>
      <file_contents><![CDATA[
... verbatim content of /etc/cups/cupsd.conf ...
]]></file_contents>
    </cupsd_conf_content>
  </printer>
```



注記: /etc/cups/cups-files.conf について

CUPS バージョン 1.6 もしくはそれ以降のバージョンでは、CUPS の設定ファイルが `cupsd.conf` と `cups-files.conf` の 2 つのファイルに分割されています。ただし、通常の印刷設定では `cups-files.conf` を既定値のままで使用すれば十分であることから、openSUSE Leap 15.7 では AutoYaST は `cupsd.conf` の修正のみに対応しています。

4.39.2 サウンドデバイス

設定システムを利用してサウンド関連の設定を行った場合の例を書きに示します。

例 4.78: サウンド設定

```
<sound>
  <autoinstall config:type="boolean">true</autoinstall>
  <modules_conf config:type="list">
    <module_conf>
      <alias>snd-card-0</alias>
      <model>M5451, ALI</model>
      <module>snd-ali5451</module>
      <options>
        <snd_enable>1</snd_enable>
        <snd_index>0</snd_index>
        <snd_pcm_channels>32</snd_pcm_channels>
      </options>
    </module_conf>
  </modules_conf>
  <volume_settings config:type="list">
    <listentry>
      <Master config:type="integer">75</Master>
    </listentry>
  </volume_settings>
</sound>
```

4.40 SSH 鍵と設定の取り込み

YaST では以前のインストールから SSH の鍵とサーバの設定を取り込むことができます。この機能は AutoYaST プロファイル側でも使用することができます。

例 4.79: /DEV/SDA2 からの SSH 鍵と設定の取り込み

```
<ssh_import>
  <import config:type="boolean">true</import>
  <copy_config config:type="boolean">true</copy_config>
  <device>/dev/sda2</device>
</ssh_import>
```

属性	値	説明
`import`	true / false	SSH 鍵を取り込みます。`false` を指定すると、鍵を取り込まなくなります。
`copy_config`	true / false	SSH のサーバ設定もあわせて取り込むようにします。なお、この設定は `import` が `false` である場合には効果がありません。
`device`	パーティション	鍵や設定の取り込み元を指定します。何も指定しない場合、アクセス日時が最も新しいものを取り込みます。

4.41 設定管理

AutoYaST では、Salt のような 設定管理ツール に対して、設定の一部を代行させることができます。この場合、AutoYaST では基本的なシステムのインストール (パーティション設定やネットワーク設定など) を行い、残りの設定作業を代行させる形になります。

注記: Salt のみをサポートする件について

本文書内には Puppet に関する説明も書かれていますが、サポート対象となるのは Salt のみであることに注意してください。ただし、Puppet をご利用の場合で何か問題があれば、報告をお願いいたします。

AutoYaST では 2 種類の異なるアプローチに対応しています:

- 設定管理サーバを使用する方法。この場合、AutoYaST は設定管理ツールを構成します。ここからマスターサーバへの接続を行い、システムの設定に関する情報を取得します。
- どこか別の場所 (たとえば HTTP サーバや USB メモリなど) から設定を取得する方法。この場合、設定管理ツールはスタンドアロン (単独) モードで動作することになります。

4.41.1 設定管理サーバへの接続

このアプローチは、設定管理サーバ (Salt や Puppet の用語では、マスター と呼びます) が既に用意されている場合、特に有用となります。この場合、構築時に最も難しい箇所は、適切な認証方式の設定となります。

Salt および Puppet では下記の認証方式に対応しています:

- その場での手動認証。AutoYaST がクライアントを起動すると、新しい認証要求が生成されます。管理者はその要求を、サーバ側で受け付けることになります。AutoYaST では接続を再試行しますので、その間にキーを受け付けてもらうことができれば、AutoYaST がインストールを続けることができるようになります。
- 事前シード鍵の使用。事前シード鍵の生成方法について、詳しくはお使いの設定管理システムのドキュメンテーションをお読みください。AutoYaST 内での事前シード鍵の参照先は、`keys_url` オプションで指定します。

下記に設定例を示します。AutoYaST はクライアントを起動して認証要求を生成します。最大で 3 回、15 秒間隔で再試行します。

例 4.80: 手動認証によるクライアント／サーバ

```
<configuration_management>
    <type>salt</type>
    <master>my-salt-server.example.net</master>
    <auth_attempts config:type="integer">3</auth_attempts>
    <auth_time_out config:type="integer">15</auth_time_out>
</configuration_management>
```

ただし、下記に示す設定例では、AutoYaST はキーを外付けのストレージ (USB メモリなど) から取得し、それを利用してマスターサーバへの接続を行っています。

例 4.81: 事前シード鍵によるクライアント／サーバ

```
<configuration_management>
    <type>salt</type>
    <master>my-salt-server.example.net</master>
    <keys_url>usb:/</keys_url>
</configuration_management>
```

下記の表には、これらのシナリオで使用しているオプションの意味について説明しています。

属性	値	説明
`type`	文字列	設定管理方式の名称です。現時点では `salt` のみをサポート対象としています。
`master`	文字列	設定管理サーバのホスト名もしくは IP アドレス。
`auth_attempts`	整数	サーバに対して接続を試す際の最大試行回数。既定値は 3 です。
`auth_time_out`	整数	サーバへの接続を待機する時間 (秒単位) 。既定値は 15 秒です。
`keys_url`	使用する鍵の URL	`default.key` および `default.pub` の存在する HTTP サーバやハードディスク、USB メモリなどの URL を指定します。この鍵は、設定管理マスター内で既知のものでなければなりません。
`enable_services`	True/False	インストール後にクライアント側で設定管理サービスを有効化するかどうかを指定します。既定値は `true` です。

4.41.2 スタンドアロン (単独) モードでの実行

シンプルな構成を目的とした場合、設定管理サーバの配置は不要となります。その代わり、Salt や Puppet を スタンドアロン (単独) (もしくは マスターレス) モードで動作させます。

サーバが存在しないことから、AutoYaST 側には設定の取得元を設定する必要があります。設定ファイルを TAR 形式の書庫にまとめて、このファイルをどこか (USB メモリや HTTP/HTTPS サーバ、もしくは NFS/SMB 共有) に配置します。

なお、TAR 形式の書庫は Salt サーバ内での `/srv` と同じ構造になっていなければなりません。具体的には、Salt の状態に関する情報は `salt` ディレクトリ内に配置するほか、formula については個別の `formulas` ディレクトリ内に配置します。

これに加えて、pillar データを含む `pillar` ディレクトリが存在していてもかまいません。このほか、`pillar_url` オプションを利用して個別の TAR 書庫を指定して、その中に含めることもできます。

例 4.82: スタンドアロンモード

```
<configuration_management>
    <type>salt</type>
    <states_url>my-salt-server.example.net</states_url>
    <pillar_url>my-salt-server.example.net</pillar_url>
</configuration_management>
```

属性	値	説明
`type`	文字列	設定管理方式の名称です。現時点では `salt` のみをサポート対象としています。
`states_url`	URL	Salt states の TAR 書庫の場所を指定します。formula や pillar を含んでいてもかまいません。また、ファイルは `salt` ディレクトリ内に配置しなければなりません。
`pillar_url`	URL	pillar を含む TAR 書庫の場所を指定します。
`modules_url`	URL	Puppet モジュールの場所を指定します。

4.41.3 SUSE Multi-Linux Manager Salt Formulas サポートについて

AutoYaST ではスタンドアロン (単独) モードで SUSE Multi-Linux Manager Salt Formulas を動作させる環境をサポートしています。この場合、formula は state 用の TAR 書庫内に存在する必要があります。AutoYaST では、適用すべき formula を選択し、設定するための画面を表示します。

ただし、この機能を利用してしまうと、AutoYaST がユーザからの入力を待機してしまうことになりますので、AutoYaST の目的である無人インストールの意味が薄くなってしまうことに注意してください。

III 動的プロファイルによる一括インストールの管理

5 動的プロファイルの使用 194

6 ルールとクラス 195

7 ERB テンプレート 211

8 ERB テンプレートとスクリプトの組み合わせ 219

5 動的プロファイルの使用

改訂履歴

2022-02-11

複数のシステムに対してインストールを行う場合、インストール先のシステムに合わせて自動調整を行ってくれるようなプロファイルがあると便利です。AutoYaST では、3 種類の仕組みでインストール時にプロファイルを調整することができます。

ルールとクラス

ルールとクラスを使用することで、複数の制御ファイルを 1 つにまとめることができるようになります。詳しくは 第6章「ルールとクラス」をお読みください。

ERB テンプレート

AutoYaST では組み込み Ruby (ERB) テンプレート文法と呼ばれる仕組みに対応し、インストール時にプロファイルを変更することができるようになっています。使用方法に関する詳細は、第7章「ERB テンプレート」をお読みください。

インストール前スクリプト

インストール前スクリプトを使用することで、インストール時にプロファイルを修正することができるばかりか、全く新しいプロファイルを生成させることもできます。詳しくは 4.31.1項「事前スクリプト」をお読みください。

インストール時のユーザへの問い合わせ

動的なプロファイルの代替として AutoYaST では、プロファイルの実行時にユーザに対して問い合わせを行うことができます。この場合、インストールは全くの無人では動作しないことになりますが、ユーザ名やパスワード、IP アドレスなどを設定する場合には有用です。この機能に関する詳細は、4.34項「インストール時におけるユーザへの値の確認」をお読みください。

6 ルールとクラス

改訂履歴

2025-02-28

ルールとクラスを使用することで、様々な方法でマシンのインストールをカスタマイズすることができます:

- ルールを使用することで、システムの属性に応じた設定を行うことができます。
- クラスはインストール先のシステムの設定グループを構成するものです。クラスはシステムに対して割り当てることができます。



注記: autoyast 起動オプションのみの使用について

ルールとクラスを使用する場合、起動パラメータでは `autoyast=URL` のみを指定してください。

これは、`autoyast2=URL` を使用してしまうと、単一の AutoYaST 制御ファイルのみをダウンロードしてしまうことから、ルールとクラスに対応できなくなってしまうためです。

6.1 ルールベースの自動インストール

ルールを使用することで、インストール時に複数の制御ファイルを組み合わせて、システムの属性に応じた設定を行うことができます。ルールベースのインストールは、ルールファイルを元に制御を行います。

ルールベースのインストールはたとえば、2 つの部署に対するシステムのインストールをまとめて行いたいような場合に有用です。たとえば部署 A ではオフィスデスクトップをインストールする要件があり、部署 B では開発者向けのワークステーションを構築する必要がある場合、2 つの異なるルールファイルを作成することになります。それぞれのルールではそれぞれ異なるシステムパラメータを設定しますし、そこからそれぞれの部署に対応するためのプロファイルへのリンクも作成することになります。

ルールファイルは XML 形式のファイルで、自動インストールを行うシステム内の各グループに対するルールを記述します。ここには 1 つもしくは複数のルールが書き込まれ、システムの属性を元にシステムのグループを判断します。全てのルールの処理が終わると、システム内のグループは特定の制御ファイルに辿り着きます。ルールファイルと制御ファイルは、いずれもあらかじめ指定されたアクセス可能な場所に配置します。

ルールファイルは `autoyast` キーワードで制御ファイルを直接指定しない場合にのみ取得します。たとえば下記のような指定を行ってしまうと、ルールファイルが評価されなくなってしまいます:

```
autoyast=http://10.10.0.1/profile/myprofile.xml
autoyast=http://10.10.0.1/profile/rules/rules.xml
```

その代わりに、下記のように指定してください:

```
autoyast=http://10.10.0.1/profile/
```

これにより、`http://10.10.0.1/profile/rules/rules.xml` ファイルを読み込むようになります (ディレクトリ名の末尾のスラッシュが重要です) 。

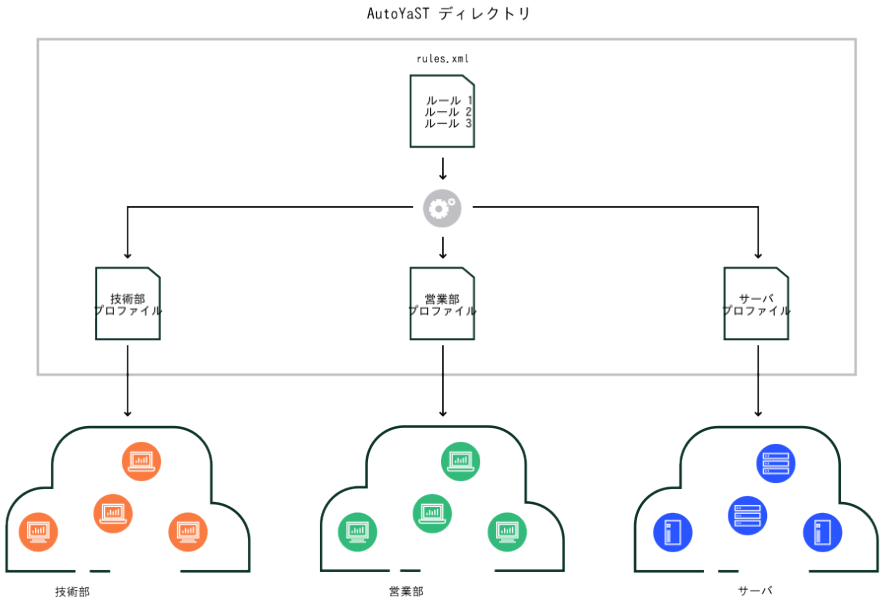

図 6.1: ルール

複数のルールを適用する場合、各グループの最後でスクリプトを介して制御ファイルが合成され生成されます。この合成処理はルールの順序をもとにして動作するもので、後ろのほうにあるルールが前のほうにあるルールを上書きします。なお、合成された XML ファイルの冒頭セクションの名前は、合成が成功するようアルファベット順である必要があります。

ルールのファイルの使用は任意です。ルールファイルが何も見つからない場合、システムのインストールは指定された制御ファイルを元にした標準的な方法で行われるか、システムの MAC アドレスや IP アドレスを元にして制御ファイルを検索して実施されます。

6.1.1 ルールファイルの詳細

例 6.1: シンプルなルールファイル

> 下記のシンプルな例は、既知のハードウエアでクライアントが設定を取得する際、どのようなルールファイルを設定することができるのかを示すための例です。

```
<?xml version="1.0"?>
<!DOCTYPE autoinstall>
<autoinstall xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://
www.suse.com/1.0/configns">
  <rules config:type="list">
    <rule>
       <disksize>
            <match>/dev/sdc 1000</match>
            <match_type>greater</match_type>
       </disksize>
       <result>
            <profile>department_a.xml</profile>
            <continue config:type="boolean">false</continue>
        </result>
    </rule>
    <rule>
       <disksize>
            <match>/dev/sda 1000</match>
            <match_type>greater</match_type>
       </disksize>
       <result>
            <profile>department_b.xml</profile>
            <continue config:type="boolean">false</continue>
        </result>
    </rule>
  </rules>
</autoinstall>
```

上記の例では 2 つのルールが設定され、それぞれに対して別々の制御ファイルが設定されています。この場合、ルールの判断基準になっているのは `disksize` (ディスクサイズ) です。ルールファイルを処理したあと、YaST は `rules.xml` ファイル内に書かれたルールを利用して、システムの判断を行います。インストール先のシステムが、ルール内に書かれた全ての属性条件に合致すると、対応するリソースが制御ファイルのスタックに追加され、最終的な制御ファイルを作成するために使用されるようになります。なお `continue` プロパティは、AutoYaST に対してルールに合致した際、それ以外のルールも参照するかどうかを制御するものです。

なお、最初のルールに合致しなかった場合、合致するルールが見つかるまで次のルールへの移動を繰り返します。

`disksize` 属性を使用することで、ハードディスクのサイズに応じて設定を変更することができるようになります。上記の例では、最初のルールは `/dev/sdc` に対するルールで、これが 1 GB よりも大きい (`match` プロパティ) かどうかを判断しています。

ルールには少なくとも 1 つ以上の属性を指定しなければなりません。複数の属性 (たとえばメモリやアーキテクチャなど) を指定する場合は、`rule` リソース内に複数の属性を指定することになります (次の例をご覧ください) 。

例 6.2: シンプルなルールファイル

下記のシンプルな例は、既知のハードウエアでクライアントが設定を取得する際、どのようなルールファイルを設定することができるのかを示すための例です。

```
<?xml version="1.0"?>
<!DOCTYPE autoinstall>
<autoinstall xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://
www.suse.com/1.0/configns">
  <rules config:type="list">
    <rule>
       <disksize>
            <match>/dev/sdc 1000</match>
            <match_type>greater</match_type>
       </disksize>
       <memsize>
            <match>1000</match>
            <match_type>greater</match_type>
       </memsize>
       <result>
            <profile>department_a.xml</profile>
            <continue config:type="boolean">false</continue>
        </result>
    </rule>
    <rule>
       <disksize>
            <match>/dev/sda 1000</match>
            <match_type>greater</match_type>
       </disksize>
       <memsize>
            <match>256</match>
            <match_type>greater</match_type>
       </memsize>
       <result>
            <profile>department_b.xml</profile>
            <continue config:type="boolean">false</continue>
        </result>
    </rule>
  </rules>
</autoinstall>
```

ルールディレクトリは、起動時に `autoyast` キーワードで指定したものと同じディレクトリ内に配置しなければなりません。たとえばクライアントを `autoyast=http://10.10.0.1/profiles/` で起動する場合、AutoYaST はルールファイルを検索する際、http://10.10.0.1/profiles/rules/rules.xml を使用します。

6.1.2 独自のルール

AutoYaST が提供する属性では目的を達成できないような場合、独自のルールを作成することもできます。独自のルールはシェルスクリプトで記述し、スクリプトの出力 (STDOUT のみ、STDERR は無視されます) を判断結果として使用します。

下記に独自のルールの設定例を示します:

```
<rule>
  <custom1>
    <script>
if grep -i intel /proc/cpuinfo > /dev/null; then
echo -n "intel"
else
echo -n "non_intel"
fi;
    </script>
    <match>*</match>
    <match_type>exact</match_type>
  </custom1>
  <result>
    <profile>@custom1@.xml</profile>
    <continue config:type="boolean">true</continue>
  </result>
</rule>
```

このルール内のスクリプトは STDOUT に `intel` もしくは `non_intel` を出力します (grep コマンドの出力は /dev/null に送られています) 。このルールスクリプトの出力は 2 つの '@' 文字で挟まれた箇所に当てはめられ、取得すべき制御ファイルの名前が `intel.xml` もしくは `non_intel.xml` になるように設定されています。これらのファイルには、ソフトウエアの選択を行うための AutoYaST のプロファイルの一部を含めることができます。この場合、Intel ハードウエアかどうかによって、インストールするソフトウエアを変える動作になります。

独自のルールは最大で 5 つまでに制限されています。そのため、`custom1` から `custom5` までを設定することができます。

6.1.3 ルールで使用する比較の種類

match_type には 5 種類の比較のうちの 1 つを選択することができます:

- `exact` (既定値; 厳密一致)
- `greater` (より大きい)
- `lower` (より小さい)

- `range` (範囲指定)
- `regex` (正規表現; bash の `=~` 演算子のようなシンプルな仕組みです)

`exact` を指定した場合、文字列として厳密に一致した場合にのみ合致したとみなされます。`regex` は単純な部分文字列としても使用することができますので、たとえば `ntel` のように指定すると、Intel, intel, intelligent のいずれにも合致することになります。`greater` と `lower` は `memsize` や `totaldisk` など、数値同士を比較する際に使用します。ただし、整数値のみを扱うことができます。`range` も整数値の範囲を指定する場合にのみ使用することができるもので、たとえば `512-1024` のように、`最小-最大` のような形式で値を指定します。

6.1.4 属性の組み合わせ

論理演算子を使用することで、複数の属性を組み合わせて使用することもできます。たとえば `disksize` (ディスクサイズ) が 1GB 以上であるか、もしくは `memsize` (メモリサイズ) が 512MB ちょうどである場合にのみ合致するルールを作成することができます。

論理演算子を指定するには、ルール内に `operator` 要素を指定します。この要素には `and` (いずれのルールにも合致した場合に全体で合致したものとみなす) もしくは `or` (いずれかのルールに合致した場合に全体で合致したものとみなす) を指定することができます。既定値は `and` です。下記に例を示します:

```
<rule>
  <disksize>
    <match>/dev/sda 1000</match>
    <match_type>greater</match_type>
  </disksize>
  <memsize>
    <match>256</match>
    <match_type>greater</match_type>
  </memsize>
  <result>
    <profile>machine2.xml</profile>
    <continue config:type="boolean">false</continue>
  </result>
  <operator>or</operator>
</rule>
```

6.1.5 ルールファイルの構造

`rules.xml` は下記の条件全てに合致する必要があります:

- 少なくとも 1 つ以上のルールが存在すること。
- `rules.xml` というファイル名になっていること。
- プロファイルのリポジトリ内の `rules` ディレクトリ内に配置されていること。
- ルール内の合致について、少なくとも 1 つ以上の属性が存在すること。

6.1.6 定義済みのシステム属性

下記の表には、ルールファイル内で合致に使用することのできる、定義済みのシステム属性の一覧を示しています。

お使いのシステムでどのような値になるのかを知りたい場合は、`/sbin/yast2 ayast_probe ncurses` を実行してください。スクロール可能なテキストボックスが表示され、その中に検出された値が表示されます。ただし、このコマンドは排他ロックを取得する必要があるため、他の YaST プロセスが動作している (例: インストーラ) 場合には動作しません。そのため、インストール時に実行することはできません。

表 6.1: システム属性

属性	値	説明
`hostaddress`	ホストの IP アドレス	この属性には `exact` による厳密一致を指定しなければなりません。
`hostname`	ホスト名	この属性には `exact` による厳密一致を指定しなければなりません。
`domain`	ホストのドメイン名	この属性には `exact` による厳密一致を指定しなければなりません。
`installed_product`	インストールする製品の名前	この属性には `exact` による厳密一致を指定しなければなりません。

属性	値	説明
`installed_product_version`	インストールする製品のバージョン	この属性には `exact` による厳密一致を指定しなければなりません。
`network`	ホストのネットワークアドレス	この属性には `exact` による厳密一致を指定しなければなりません。
`mac`	ホストの MAC アドレス	この属性には `exact` による厳密一致を指定しなければなりません (MAC アドレスは `0080c8f6484c` のような形式になります) 。
`linux`	システムに設定されている Linux パーティションの数	この属性は 0 以上の値になります。
`others`	システムに設定されている Linux 以外のパーティションの数	この属性は 0 以上の値になります。
`xserver`	グラフィックアダプタが必要な X サーバ	この属性には `exact` による厳密一致を指定しなければなりません。
`memsize`	メガバイト単位でのホスト内のシステムメモリ量	全ての種類の比較を使用することができます。
`totaldisk`	メガバイト単位でのホスト内のディスク容量合計	全ての種類の比較を使用することができます。
`hostid`	IP アドレスの 16 進数表記	`exact` による厳密一致が必要です
`arch`	ホストのアーキテクチャ	`exact` による厳密一致が必要です

属性	値	説明
karch	ホストのカーネルアーキテクチャ (SMP カーネルや Xen カーネルなど)	exact による厳密一致が必要です
disksize	ドライブデバイスとサイズ (単位: メガバイト)	全ての種類の比較を使用することができます。
product	SMBIOS 内で設定されているハードウエアの製品名	exact による厳密一致が必要です
product_vendor	SMBIOS 内で設定されているハードウエアの製造元名	exact による厳密一致が必要です
board	SMBIOS 内で設定されているシステムボードの名前	exact による厳密一致が必要です
board_vendor	SMBIOS 内で設定されているシステムボードの製造元名	exact による厳密一致が必要です
custom1-5	シェルスクリプトを利用した独自ルール	全ての種類の比較を使用することができます。

6.1.7 ダイアログルール

チェックボックス付きのポップアップを表示して、合致させたいルールを選択することもできます。

下記に示す要素は rules.xml ファイル内の XML 構造内に配置すべきものです:

```
<rules config:type="list">
  <rule>
    <dialog>
      ...
    </dialog>
  </rule>
</rules>
```

属性／値／説明

`dialog_nr`

同じ `dialog_nr` の値が設定されたルールは、同一のダイアログ内に表示されるようになります。そのため、`dialog_nr` は複数のルール内に設定することができます。

```
<dialog_nr config:type="integer">3</dialog_nr>
```

この要素は任意指定で、値を指定しない場合の既定値は `0` です。全てのルールで 1 つのポップアップだけを表示する場合は、`dialog_nr` を指定しなくてもかまいません。

`element`

重複しない ID を指定します。複数のダイアログを設定している場合は、同じ ID のものが現れてはなりません。ただしダイアログ 1 と 2 で同じ `1` を指定してもかまいません (この動作は `ask` とは逆で、ダイアログが異なる場合であれば同じ ID でもかまいません) 。

```
<element config:type="integer">3</element>
```

任意指定です。何も指定しない場合、AutoYaST 側で ID を自動設定します。矛盾したルールは設定できません (下記をお読みください) 。

`title`

ポップアップダイアログのタイトルを指定します。

```
<title>Desktop Selection</title>
```

任意指定

`question`

チェックボックスの隣に表示する文章を指定します。

```
<question>GNOME Desktop</question>
```

任意指定です。何もテキストを指定しない場合、このルールを起動した XML ファイルの名前が表示されます。

`timeout`

ダイアログの制限時間を秒単位で指定します。制限時間を超過すると、［Ok］ボタンが自動的に「押された」として扱われます。ルール内の問い合わせで止まったままにしたくない場合に使用します。

```
<timeout config:type="integer">30</timeout>
```

任意指定です。制限時間を指定しない場合、ユーザ側で操作を行ってダイアログを終了しない限り、インストールが停止したままになります。

`conflicts`

このルールと競合する要素 ID (ルール) のリストを指定します。このルールに合致した場合、もしくはこのルールをユーザ側で選択した場合、競合関係にあるルールの選択は外され、ポップアップ内で無効化されます。デッドロックを発生させることがないように注意してお使いください。

```
<conflicts config:type="list">
  <element config:type="integer">1</element>
  <element config:type="integer">5</element>
  ...
</conflicts>
```

任意指定

下記はルール内でダイアログを使用する場合の例です:

```
<rules config:type="list">
  <rule>
    <custom1>
      <script>
echo -n 100
      </script>
      <match>100</match>
      <match_type>exact</match_type>
    </custom1>
    <result>
      <profile>rules/gnome.xml</profile>
      <continue config:type="boolean">true</continue>
    </result>
    <dialog>
      <element config:type="integer">0</element>
      <question>GNOME Desktop</question>
      <title>Desktop Selection</title>
      <conflicts config:type="list">
        <element config:type="integer">1</element>
      </conflicts>
      <dialog_nr config:type="integer">0</dialog_nr>
    </dialog>
  </rule>
  <rule>
    <custom1>
      <script>
echo -n 100
      </script>
      <match>101</match>
      <match_type>exact</match_type>
    </custom1>
```

```
    <result>
      <profile>rules/gnome.xml</profile>
      <continue config:type="boolean">true</continue>
    </result>
    <dialog>
      <element config:type="integer">1</element>
      <dialog_nr config:type="integer">0</dialog_nr>
      <question>Gnome Desktop</question>
      <conflicts config:type="list">
        <element config:type="integer">0</element>
      </conflicts>
    </dialog>
  </rule>
  <rule>
    <custom1>
      <script>
echo -n 100
      </script>
      <match>100</match>
      <match_type>exact</match_type>
    </custom1>
    <result>
      <profile>rules/all_the_rest.xml</profile>
      <continue config:type="boolean">false</continue>
    </result>
  </rule>
</rules>
```

6.2 クラス

クラスはインストール先のシステムのグループに対する設定を表します。ルールとは異なり、クラスは制御ファイル内に設定する必要があります。クラスを設定した後、目的のシステムに割り当てを行います。

下記にクラス定義の例を示します:

```
<classes config:type="list">
  <class>
    <class_name>TrainingRoom</class_name>
    <configuration>Software.xml</configuration>
  </class>
</classes>
```

上記の例では、`Software.xml` は `classes/TrainingRoom/` サブディレクトリ内に配置しなければなりません。AutoYaST の制御ファイルやルールと同じ場所から取得が行われます。

複数の制御ファイルが存在していて、それらの内容のうち共通する部分がある場合は、共通する部分をクラスとして設定すると良いでしょう。このほか、XIncludes を使用することもできます。

設定管理システムを使用することで、クラスのセットを作成することもできます。クラスの定義では下記のような変数を使用することができます:

- Name: クラス名
- Description:
- Order: クラスの順序 (優先順位)

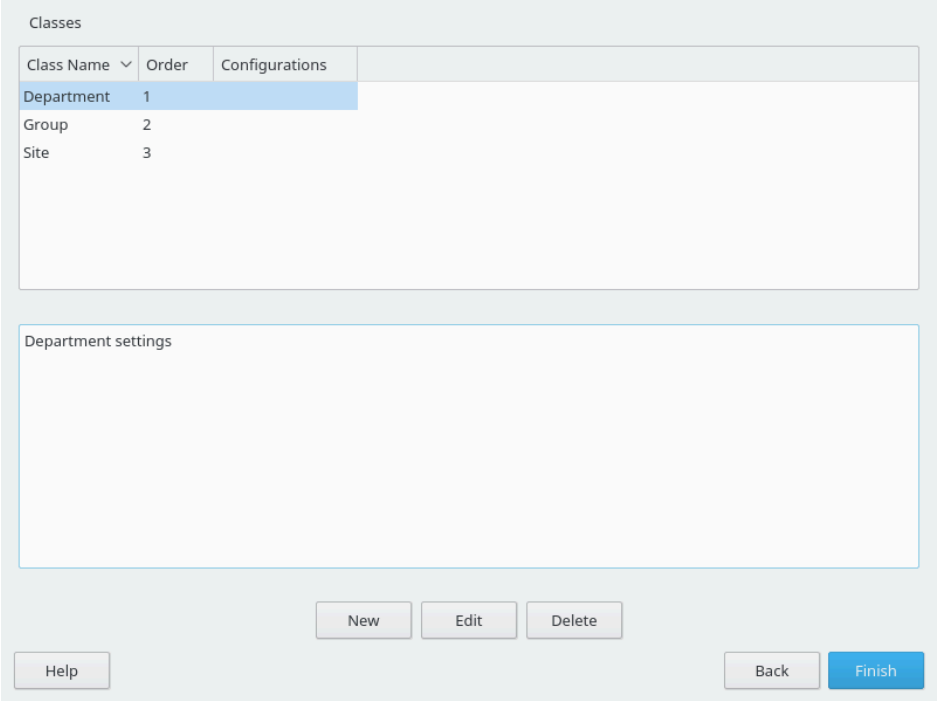

図 6.2: クラスの定義

クラスは必要な数だけ作成することができますが、設定をできるかぎり簡潔にするため、できるだけ少ないクラス数にしておくことをお勧めします。たとえば下記のようなクラスのセットを使用することができます:

- site: 物理的な配置をもとにしたクラス
- machine: マシンの種類をもとにしたクラス
- role: マシンの機能をもとにしたクラス
- group: 部署や場所をもとにしたクラス

class ディレクトリ内に保存したファイルは通常の制御ファイルと同じ書式ですが、設定の一部分のみを表すように記述します。たとえば特定のネットワークインターフェイスを持つコンピュータに対して新しい制御ファイルを作成する場合、ネットワークの設定のみを設定した制御ファイルリソースを作成します。複数の種類のネットワークがある場合は、他のクラスファイルを利用してそれらの種類に対応させ、それらを組み合わせて新しい制御ファイルを作り上げる形になります。

6.3 ルールとクラスの混合

自動インストールのセッションでは、ルールとクラスを混在させることができます。たとえばクラス定義を含むルールを利用してシステムを識別したりすることができます。流れは 図A.1「ルールの取得処理」で示しています。

ルールを取得してそれらを合成したら、生成された制御ファイルが処理され、クラスの定義が存在していないかどうかを確認します。クラスが定義されている場合、元のリポジトリからクラスファイルを取得し、さらに合成処理を行います。

6.4 ルールとクラスの合成

クラスとルールの両方を使用することで、複数の XML ファイルから 1 つの XML ファイルを合成することになります。この合成処理は、通常期待される処理とは異なる動作になっていることから、しばしば混乱の原因となります。まず注意しておくべきことは、合成処理を成功させるには、合成対象の XML ファイル内のトップセクションは、アルファベット順に並んでいなければならない、という点です。

たとえば下記のような 2 種類の合成すべき XML ファイルがあるものとします:

```
<partitioning config:type="list">
  <drive>
    <partitions config:type="list">
      <partition>
        <filesystem config:type="symbol">swap</filesystem>
        <format config:type="boolean">true</format>
        <mount>swap</mount>
        <partition_id config:type="integer">130</partition_id>
        <size>2000mb</size>
      </partition>
      <partition>
        <filesystem config:type="symbol">xfs</filesystem>
        <partition_type>primary</partition_type>
        <size>4Gb</size>
        <mount>/data</mount>
      </partition>
    </partitions>
```

```
  </drive>
</partitioning>
```

```
<partitioning config:type="list">
  <drive>
    <initialize config:type="boolean">false</initialize>
    <partitions config:type="list">
      <partition>
        <format config:type="boolean">true</format>
        <filesystem config:type="symbol">xfs</filesystem>
        <mount>/</mount>
        <partition_id config:type="integer">131</partition_id>
        <partition_type>primary</partition_type>
        <size>max</size>
      </partition>
    </partitions>
    <use>all</use>
  </drive>
</partitioning>
```

これらを合成することによって、通常は 3 種類のパーティション設定になるかと思われますが、残念ながらそうではありません。これらを合成すると 2 種類のパーティション設定となり、最初のパーティションはスワップとルートパーティションを混在させたものになります。また、`mount` や `size` など、両方のパーティション設定内に存在するものは、いずれも 2 つ目のファイルのものを使用することになります。いずれか片方にしか存在しない設定については、設定されているほうの値が適用されます。

また上記の例では、2 つ目の `drive` については不要となり、両方を 1 つにまとめる必要があります。このような場合、3 つの個別のパーティション設定を行う必要があります。また、`dont_merge` メソッドを使用することで、合成時の問題を解決することができます:

```
<classes config:type="list">
  <class>
    <class_name>swap</class_name>
    <configuration>largeswap.xml</configuration>
    <dont_merge config:type="list">
      <element>partition</element>
    </dont_merge>
  </class>
</classes>
```

```
<rule>
  <board_vendor>
    <match>ntel</match>
    <match_type>regex</match_type>
  </board_vendor>
  <result>
    <profile>classes/largeswap.xml</profile>
    <continue config:type="boolean">true</continue>
    <dont_merge config:type="list">
```

```
      <element>partition</element>
    </dont_merge>
  </result>
  <board_vendor>
    <match>PowerEdge [12]850</match>
    <match_type>regex</match_type>
  </board_vendor>
  <result>
    <profile>classes/smallswap.xml</profile>
    <continue config:type="boolean">true</continue>
    <dont_merge config:type="list">
      <element>partition</element>
    </dont_merge>
  </result>
</rule>
```

7 ERB テンプレート

改訂履歴

2025-07-07

ERB テンプレートとは、プロファイル内に Ruby 言語のコードを埋め込むことのできる機能で、インストール時にプロファイルを修正する機能を提供します。この仕組みを利用することで、システムの状態を検知してプロファイル内の設定値を調整したり、セクションの追加や削除などを行ったりすることができます。

ERB 機能を有効化するには、プロファイルのファイル名に `.erb` という拡張子を追加します (例: `autoyast.xml.erb`) 。このような仕組みが存在することから、ルールやクラスと ERB テンプレートを同時に使用することはできません。

7.1 ERB とは?

ERB とは Embedded Ruby の略で、Ruby プログラミング言語の力を借りて様々なコンテンツ生成を行うための仕組みです。ERB を利用してプロファイル内に Ruby コードを含めることで、インストール先のシステムに依存した様々なプロファイル調整を行うことができるようになります。

ERB を使用する場合、Ruby 言語のコードは `<%` 記号から `%>` 記号までの間に記述します。また、コマンドの出力をそのままプロファイルに入れ込みたい場合は、イコール記号 (`=`) で設定します。

例 7.1: ERB を利用したファイルの取り込み

```
<bootloader>
  <% require "open-uri" %>
  <%= URI.open("http://192.168.1.1/profiles/bootloader-common.xml").read %>
</bootloader> <!-- this line gets replaced with the content of bootloader-common.xml -->
```

Ruby の仕組みを利用することで、任意のコマンドを実行することもできます。たとえばコマンドの出力結果を取得したい場合は、コマンドの前後をバッククオートで括ります。また、コマンドの実行が成功したかどうかを調べたい場合は、 `system` 関数を使用してください。

例 7.2: RUBY におけるコマンドの実行

```
<% files = `ls` %> <!-- files にはコマンドの実行結果が入ります (例: "file1
file2
file3") -->
<% success = system("dmidecode | grep some-model") %> <!-- success には true もしくは false
 のいずれかが入ります -->
```

このほか条件句やループなど、高度な Ruby 言語構造を含めることもできます。

例 7.3: RUBY 言語構造の使用

```
<% ip_forward = File.read("/proc/sys/net/ipv4/ip_forward").strip %>
<% if ip_forward == "1" %>
  <!-- 何らかの処理 -->
<% end %>

<% files = `ls /tmp/config/*.xml` %>
<% files.split.each do |file| %>
  <%= file.read %>
<% end %>
```

AutoYaST では ヘルパー関数 と呼ばれる仕組みも提供しています。これは `disks` (ディスク) や `network_cards` (ネットワークカード) など、インストール先のシステムに関する様々な情報を取得することができる仕組みです。ヘルパーの一覧と値の意味について、詳しくは 7.2項「テンプレートヘルパー」をお読みください。

7.2 テンプレートヘルパー

テンプレートヘルパーは Ruby メソッド集であり、プロファイル内で使用することで様々なインストール先システムの情報を収集することができます。

7.2.1 boot_efi?

`boot_efi?` はブール値のヘルパーで、システムが EFI で起動されているかどうかを表します。下記の例では、現在の起動モードに合わせてブートローダを設定しています。

例 7.4: ブートローダの設定

```
<% if env.boot_efi? %>
  <loader_type>grub2-efi</loader_type>
<% else %>
  <loader_type>grub2</loader_type>
<% end %>
```

7.2.2 disks

`disks` ヘルパーは、検出されたディスクの一覧を返却するヘルパー関数です。一覧の各要素には、デバイス名やサイズなどの基本的な情報が含まれます。

キー	型	値
:device	String	デバイスのカーネル名 (例: sda)
:model	String	ディスクの型番
:serial	String	シリアル番号
:size	Integer	ディスクサイズ (セクタ数)
:udev_names	Array<String>	ディスクの udev 名の一覧 (プロファイル内でデバイスを指定する際、これらのいずれかを使用することができます)
:vendor	String	製造元の名前

下記のプロファイル例は、最も容量の大きいディスクに対してインストールを行う場合の例を示しています。検出されたディスクの一覧をサイズで並び替え、末尾のデバイスを選択します。あとは device 要素に対して :device キーで取得できた値を代入して使用します。

例 7.5: 最も容量の大きいディスクの使用

```
<partitioning t="list">
  <drive>
    <% disk = disks.sort_by { |d| d[:size] }.last %> <!-- find the largest disk -->
    <device><%= disk[:device] %></device> <!-- print the disk device name -->
    <initialize t="boolean">true</initialize>
    <use>all</use>
  </drive>
</partitioning>
```

7.2.3 network_cards

network_cards ヘルパーはネットワークカードの一覧を返す関数です。ここにはネットワークカードの名前のほか、状態に関する情報 (接続されているかどうかなど) も含まれています。

キー	型	値
:device	String	デバイス名 (例: eth0 , enp3s0 など)

キー	型	値
:mac	String	MAC アドレス
:active	Boolean	デバイスが有効化されているかどうか
:link	Boolean	デバイスが接続されているかどうか
:vendor	String	製造元の名前

下記の例では、ネットワークに接続されている最初のネットワークカードを検出し、そのネットワークカードで DHCP を使用するように設定します。

例 7.6: 接続されているネットワークカードの設定

```
<interfaces t="list">
  <% with_link = network_cards.sort_by { |n| n[:name] }.find { |n| n[:link] } %>
  <% if with_link %>
    <interface>
      <device><%= with_link[:device] %></device>
      <startmode>auto</startmode>
      <bootproto>dhcp</bootproto>
      </interface>
  <% end %>
</interfaces>
```

7.2.4 os_release

os_release ヘルパーは、オペレーティングシステムに関する情報を返す関数です。これは /etc/os-release ファイルから情報を読み込みます。

キー	型	値
:id	String	ディストリビューション ID (例: sles, opensuse-tumbleweed)
:name	String	ディストリビューション名 (例: SLES, openSUSE Tumbleweed)

キー	型	値
`:version`	String	ディストリビューションのバージョン (例: `15.2`)

この情報を使用することで、インストールすべき製品を判断することができます。たとえば SLE と openSUSE で異なる設定を適用することができるようになります。

例 7.7: 異なるディストリビューションに対して同じプロファイルを適用する設定

```
<products t="list">
  <% if os_release[:id] == 'sle' %>
  <product>SLES</product>
  <% else %>
  <product>openSUSE</product>
  <% end %>
</products>
```

7.2.5 hardware

`hardware` ヘルパーは追加のハードウエア情報を提供する仕組みです。ここには `hwinfo` コマンドからの全ての情報が含まれていて、その他のヘルパーでは不十分な場合の回避策として使用することができます。たとえば下記の例では、`hardware` ヘルパーを利用して USB デバイスのフィルタリングを行っています。`hardware` ヘルパーで提供される情報についての詳細は、7.3項「ERB ヘルパーの実行」をお読みください。

例 7.8: USB デバイスのフィルタ

```
<% usb_disks = hardware["disk"].select { |d| d["driver"] != "usb-storage" } %>
```

7.3 ERB ヘルパーの実行

AutoYaST の ERB ヘルパーを実行するにあたって Ruby コンソールを使用して結果を確認することができます。ERB ヘルパーは `Y2Autoinstallation::Y2ERB::TemplateEnvironment` クラスのインスタンスを介してアクセスできます。下記のようにして root で Ruby インタプリタを起動して確認してください: `irb -ryast -rautoinstall/y2erb`

例 7.9: ヘルパーの実行

```
irb > env = Y2Autoinstallation::Y2ERB::TemplateEnvironment.new  # env 変数にはヘルパーのインスタンスが代入されます
```

```
irb > env.disks
=>
[{:vendor=>"WDC", :device=>"sda", ...},
 {:vendor=>"TOSHIBA", :device=>"sdb", ...},
...]

irb > env.hardware.keys
=>
["architecture",
 "bios",
 "bios_video",
 ...]

irb > env.hardware["architecture"]
=>
"x86_64"
```

7.4 ERB プロファイルの生成

AutoYaST のコマンドラインでは `check-profile` コマンドが提供されています。これは ERB ファイルからプロファイルを生成するための仕組みで、このコマンドは AutoYaST に対して ERB コードの処理と実行を依頼して、プロファイルの生成を行います。これを利用することで、期待通りにプロファイルが生成されるかどうかを確認することができます。サポートされている全てのオプションを表示するには、`autoyast check-profile --help` のように入力して実行してください。たとえば下記の例では、AutoYaST に対してプロファイルのダウンロードと解釈を依頼して、ERB コードの実行と事前スクリプトの実行までを行っています。実行結果は `result.xml` ファイルに出力されます。

例 7.10: プロファイルの生成

```
> sudo yast2 autoyast check-profile filename=http://192.168.1.100/autoinst.erb
output=result.xml run-scripts=true run-erb=true
```

警告: check-profile の許可について

ほとんどの場合、`check-profile` には root のアクセス権限が必要となります。そのため、インストール前スクリプトや ERB プロファイルを root で動作させることになることに注意してください。信頼できるプロファイルのみを指定してください。

7.5 ERB プロファイルのデバッグ

ERB の処理や動作を 1 つずつ確認していきたい場合は、YaST の提供する `byebug` デバッガをお使いください。具体的には `rubygem(byebug)` パッケージをインストールしたあと、`Y2DEBUGGER` 環境変数を 1 に設定します。

例 7.11: デバッグ環境の準備

```
> sudo zypper --non-interactive in "rubygem(byebug)"
> sudo Y2DEBUGGER=1 yast2 autoyast check-profile ...
```

ブレークポイントを追加したい場合は、その箇所に <% byebug %> を追加するだけです。`byebug` に関する詳細は、https://github.com/deivid-rodriguez/byebug (英語) をお読みください。

例 7.12: ブレークポイントの追加

```
<% byebug %>
<% if system("dmidecode | grep some-model") %>
  <!-- 何らかの処理 -->
%<% end %>
```

7.6 ERB とルール／クラスの比較

ERB とルール／クラスのどちらもプロファイルを動的に生成する仕組みですが、一般的には ERB のほうが読みやすく理解しやすい仕組みです。また、大きく異なる点として、ルールとクラスがプロファイルの合成に対応しているのに対して、ERB は対応していないことがあげられます。プロファイルの合成に関する詳細は、第6章「ルールとクラス」をお読みください。もう一方の ERB では、Ruby という高度な言語の威力を存分に発揮することができます。たとえば下記の 2 つの例では、`/dev/sdb` が存在した場合、`/home` をそのデバイス内に作成するよう指示しています。

例 7.13: ルールとクラス

```
<rule>
  <custom1>
    <script>
if blkid | grep /dev/sdb > /dev/null; then
echo -n "yes"
else
echo -n "no"
fi;
    </script>
    <match>yes</match>
    <match_type>exact</match_type>
  </custom1>
```

```
  <result>
    <profile>classes/sdb_home.xml</profile>
    <dont_merge config:type="list">
      <element>partition</element>
    </dont_merge>
  </result>
</rule>
```

例 7.14: ERB

```
<% home_in_sdb = disks.map { |d| d[:device] }.include?("sdb") %>

<partitioning config:type="list">
  <drive>
    ...
  </drive>
  <% if home_in_sdb %>
  <drive>
    <device>sdb</device>
    <disklabel>none</disklabel>
    <partitions t="list">
      <partition>
        <format t="boolean">true</format>
        <filesystem t="symbol">xfs</filesystem>
        <mount>/home</mount>
      </partition>
    </partitions>
  </drive>
  <% end %>
</partitioning>
```

8 ERB テンプレートとスクリプトの組み合わせ

改訂履歴

2022-03-16

4.31.1項「事前スクリプト」で示しているとおり、プロファイルの内容を変更するためのインストール前スクリプトを設定することができます。このスクリプトは簡単に言うと、スクリプトが `/tmp/profile/modified.xml` ファイルを生成すると、AutoYaST はその生成されたプロファイルを読み込み、元のプロファイルを廃棄する仕組みです。

これは比較的柔軟な仕組みですが、インストールメディア内に存在する言語とライブラリのみが使用できるという点にご注意ください。

8.1 スクリプト内への ERB の組み込み

ルール とは異なり、ERB テンプレートの仕組みとスクリプトを組み合わせて使用することもできます。AutoYaST では、対象のスクリプトを実行する前に、ERB タグの評価が行われます。この動作はプロファイル内に組み込まれたスクリプトに対してのみ有効で、外部のスクリプトに対しては適用されません。

下記の例では、MAC アドレスをベースにしてプロファイルをダウンロードしています。ファイルは `/tmp/profile/modified.xml` 内に保存されますので、AutoYaST はそれを読み込んで使用するようになります。

例 8.1: プロファイルを取得するために MAC アドレスを使用する方法

```
<scripts>
  <pre-scripts config:type="list">
    <script>
      <interpreter>shell</interpreter>
      <filename>load_profile.sh</filename>
      <% mac = network_cards.first >
      <source><![CDATA[#!/bin/bash
wget -O /tmp/profile/modified.xml http://myserver/<%= network_cards.first[:mac] %>.xml
]]></source>
    </script>
  </pre-scripts>
</scripts>
```

8.2 Ruby スクリプト内からの ERB ヘルパーへのアクセス

Ruby スクリプト内から ERB ヘルパーを使用することもできます。ヘルパーを使用するには require で `yast` と `autoinstall/y2erb` ライブラリを指定してください。また、ERB ヘルパーへのアクセスには `Y2Autoinstall::Y2ERB::TemplateEnvironment` クラスを指定してください。

例 8.2: RUBY スクリプト内からの ERB ヘルパーへのアクセス

```
<scripts>
  <pre-scripts config:type="list">
    <script>
      <interpreter>/usr/bin/ruby</interpreter>
      <filename>load_profile.rb</filename>
      <source><![CDATA[#!/usr/bin/env ruby
require "yast"
require "autoinstall/y2erb"
helpers = Y2Autoinstallation::Y2ERB::TemplateEnvironment.new
# TemplateEnvironment のインスタンスを利用してヘルパーを呼び出すことができます
disk_devices = helpers.disks.map { |d| d[:device] }
File.write("/root/disks.txt", disk_devices.join("
"))
]]></source>
    </script>
  </pre-scripts>
</scripts>
```

IV　自動インストール処理の理解

9　自動インストール処理 222

9 自動インストール処理

改訂履歴

2023-04-17

9.1 紹介

システムが自動インストールを起動して制御ファイルを取得すると、YaST は制御ファイル内に書かれた情報をもとにして、システムの設定を始めます。全ての設定は既定で表示されるウインドウ内にまとめられますが、完全にインストールを自動化したい場合には無効化することもできます。

YaST が設定の概要を表示したタイミングでは、YaST はハードウエアの検出を行って自動インストールのための準備を行っただけの状態になります。この時点ではまだシステムに対して、何も変更を加えていない状態になります。ここまでで何らかのエラーが発生しても、処理を安全に中止することができます。

システムを自動インストールする場合は、グラフィックアダプタやモニタを使用せずに行うこともできます。ただし、クライアントマシンに自動インストールするような場合は、進捗状況やエラー時の対応を行う必要があることから、モニタを接続しておくことをお勧めします。また、グラフィカルなインターフェイスだけでなく、テキストベースの ncurses インターフェイスを使用することもできます。このほか、キーボードやマウス、ディスプレイの存在しないクライアントであれば、シリアルポートをコンソールとして使用することもできます。

9.1.1 X11 インターフェイス (グラフィカル)

自動インストールを行う場合、これが既定のインターフェイスとなります。このインターフェイスを使用する場合には、特に何も設定を行う必要はありません。

9.1.2 シリアルコンソール

シリアルコンソールを利用してインストールを行いたい場合は、カーネルのコマンドラインに対して `console` キーワード (例: `console=ttyS0`) を追加します。これにより `linuxrc` がコンソールモードで動作するようになり、YaST についてもシリアルコンソール経由で表示されるようになります。

9.1.3 テキストベースの YaST インストール

このオプションについても、コマンドラインから有効化することができます。YaST をテキストモードで起動したい場合は、コマンドラインに `textmode=1` を指定してください。

YaST のテキストモードによる起動は、64 MB よりも少ないメモリしか搭載していないクライアントや、ディスプレイの接続がなく、X11 を設定すべきではないクライアントを使用しているような場合にお勧めです。

9.2 適切な起動メディアの選択

クライアントの起動に際してはさまざまな方式を使用することができます。コンピュータによっては搭載されているネットワークインターフェイスカード (NIC) を利用して、DHCP 経由もしくは TFTP 経由で起動イメージを取得できるものもあります。このほか、USB メモリなどからカーネルと initrd イメージを読み込んだり、起動可能な DVD-ROM から起動を行ったりすることもできます。

YaST はその起動時、起動メディアもしくは initrd 内のルートディレクトリに、`autoinst.xml` ファイルが存在していないかどうかを確認します。存在していた場合には、自動インストール処理に移行します。制御ファイルが異なるファイル名であったり、どこか別の場所に配置したりしているような場合は、カーネルのコマンドラインパラメータに `AutoYaST=URL` を指定してください。

それ以外にも、`autoinst.xml` ファイルを保存した物理デバイスや論理デバイスに `OEMDRV` というラベルを設定する方式があります。この場合、カーネルのコマンドラインで `autoinst.xml` の場所を指定する必要はありません。なお、`autoinst.xml` ファイルはデバイスのルートディレクトリに配置しなければなりません。

9.2.1 外付けストレージ (USB メモリなど) からの起動

テストやシステムの救出を目的とする場合、もしくはお使いのネットワークインターフェイスカードに PROM や PXE が搭載されていないような場合でも、AutoYaST を使用することができます。フラッシュメモリであれば制御ファイルを含めることもできます。



ヒント: インストールメディアのイメージをリムーバブルメディアにコピーする場合について

下記のコマンドを実行することで、インストールメディアのイメージをリムーバブルメディアに書き込むことができます。

```
> sudo dd if=イメージ of=リムーバブルメディアのパス bs=4M && sync
```

イメージ には SLE-15-SP7-Online-ARCH-GM-media1.iso もしくは SLE-15-SP7-Full-ARCH-GM-media1.iso のイメージファイルのパスを、リムーバブルメディアのパス にはリムーバブルメディアのパスをそれぞれ指定します。デバイス名がわからない場合は、下記のように入力して実行してください:

```
# grep -Ff <(hwinfo --disk --short) <(hwinfo --usb --short)
      disk:
      /dev/sdc             General USB Flash Disk
```

なお、デバイスのサイズがイメージのサイズに対して十分かどうかを確認しておいてください。デバイスのサイズを確認したい場合は、下記のように入力して実行します:

```
# fdisk -l /dev/sdc | grep -e "^/dev"
      /dev/sdc1   *     2048 31490047 31488000  15G 83 Linux
```

上記の例では、デバイスのサイズは 15 GB あることになります。この場合、SLE-15-SP7-Full-ARCH-GM-media1.iso を書き込む際のコマンドは下記のようになります:

```
dd if=SLE-15-SP7-Full-ARCH-GM-media1.iso of=/dev/sdc bs=4M && sync
```

なお、dd コマンドを実行する際には、デバイスがマウントされていてはなりません。また、デバイス内の全てのデータが消去されることにも注意してください。

9.2.2 SUSE Linux Enterprise のインストールメディアからの起動

SUSE Linux Enterprise のインストールメディア (SLE-15-SP7-Online-ARCH-GM-media1.iso もしくは SLE-15-SP7-Full-ARCH-GM-media1.iso) と他のメディアを併用することもできます。たとえば制御ファイルを USB メモリやネットワーク内の場所に配置しておき、DVD-ROM から起動して場所を指定することもできます。また、インストールメディアそのものを修正して、制御ファイルを入れておくこともできます。

9.2.3 ネットワーク経由での PXE 起動

PXE 起動を行うには、ネットワーク内に DHCP サーバと TFTP サーバを用意する必要があります。これにより、物理メディアを使用せずに起動を行うことができます。

なお PXE 経由でインストールを行う場合、インストールを繰り返し続けてしまうという問題が発生することがあります。これはインストール後の再起動で、通常は第 2 ステージのインストール処理が動作すべきはずのところ、PXE 経由の起動が再度行われてしまい、第 1 ステージに戻ってきてしまうことによるものです。

このような問題を解決する方法はいくつかあります。1 つは HTTP サーバ内に AutoYaST の制御ファイルを配置する方法で、通常のファイルではなく CGI スクリプトとして AutoYaST の制御ファイルを提供するように構成し、2 度目の起動では TFTP サーバの設定を変更して、ハードディスクから起動するように設定します。

もう 1 つの方法としては、AutoYaST の制御ファイル内で新しい PXE 起動の設定をアップロードするようにする方法があります:

```
<pxe>
  <pxe_localboot config:type="boolean">true</pxe_localboot>
  <pxelinux-config>
    DEFAULT linux
    LABEL linux
    localboot 0
  </pxelinux-config>
  <tftp-server>192.168.1.115</tftp-server>
  <pxelinux-dir>/pxelinux.cfg</pxelinux-dir>
  <filename>__MAC__</filename>
</pxe>
```

上記のように設定すると、初回の再起動が行われる直前に、インストール先のホストから TFTP サーバに対して新しい設定ファイルがアップロードされるようになります。ほとんどのインストール環境では TFTP デーモンが `nobody` で動作していますので、このユーザから `pxelinux.cfg` ディレクトリに書き込むことができるように設定します。また、アップロードするファイルのファイル名を指定することもできます。上記の例では `__MAC__` と指定していますが、これは `01-08-00-27-79-49-ee` のような実際の MAC アドレスに置き換えられます。なお、ファイル名の設定が無い場合は、ファイル名に IP アドレスが設定されます。

同じマシンに対して自動インストールを再度行わせたい場合は、TFTP サーバ内で該当のファイルを削除してください。

9.3 自動インストール処理の開始

9.3.1 コマンドラインオプション

コマンドライン変数 `autoyast` を設定することで、`linuxrc` を自動モードにすることができます。その場合、`linuxrc` プログラムは設定ファイル (制御ファイルとは別のものです) を検索します。そのときの場所は下記のとおりです:

- システムを起動する際に使用していた初期 RAM ディスクのルートディレクトリ
- 起動メディアのルートディレクトリ

`linuxrc` の設定ファイルには、複数のキーワードが用意されています。`linuxrc` の詳しい説明とキーワードの一覧については、付録C 高度な `linuxrc` オプション をお読みください。主なものは下記のとおりです:

autoupgrade
: AutoYaST を利用した自動アップグレードの開始については、4.10項「アップグレード」をお読みください。

autoyast
: 自動インストールの際に使用する制御ファイルの場所を指定します。詳しくは AutoYaST の制御ファイルの場所 をお読みください。

ifcfg
: ネットワークを設定し、起動するようになります。AutoYaST に対して、ネットワーク上離れた場所から取得を行う必要がある場合に必要となります。詳しくは C.3項「高度なネットワーク設定」をお読みください。

insmod
: 読み込むべきカーネルモジュールを指定します。

install
: インストールに使用するディレクトリを指定します (例: `install=nfs://192.168.2.1/CDs/`) 。



注記: SSL チェックの無効化について

HTTPS を使用している場合、既定では SSL のチェックが有効化されます。必要であれば、HTTPS の URL 内に `ssl_verify=no` を追加して、SSL のチェックを無効化してください。たとえば下記のようになります:

```
install=https://192.168.2.1/CDs/?ssl_verify=no
```

なお、複数のクエリ文字列を指定している場合は、それらをアンパサンドで区切ってください:

```
install=https://192.168.2.1/CDs/?foo=bar&ssl_verify=no
```

詳しくは `man 8 zypper` で表示されるマニュアルページ内の、"FTP/HTTP/HTTPS directory tree" セクションをお読みください。

instmode
: インストールモードを指定します。たとえば `nfs` , `http` などを指定します (`install` が指定されていれば不要です) 。

rootpassword
: AutoYaST プロファイル内で指定されていない場合、ここで root のパスワードを設定することができます。

server
: ソースディレクトリとして使用するサーバ (NFS) を指定します。

serverdir
: NFS サーバ内のディレクトリを指定します

y2confirm
: 制御ファイル内に <confirm>no</confirm> が設定されていても、確認メッセージを表示するようにします。

これらの値やキーワードはシステムを起動する際に使用され、YaST がメインの制御ファイルを得るまでの間使用されます。現時点ではソースメディアは自動検出されるようになっていますので、場合によっては `linuxrc` に特段の設定を行わなくても、自動インストールの処理を開始することができるようになっています。

従来の `linuxrc` の設定ファイル (`info`) には、インストールサーバやソースの場所など、クライアントに対して十分な情報を提供する機能が用意されてきました。通常はこのファイルを用意する必要はありませんが、DHCP も BOOTP も使用していないような環境であったり、特殊なカーネルモジュールを読み込む必要があったりするような環境では、必要となる場合もあります。

`linuxrc` にキーワードを渡したい場合は、カーネルのコマンドラインを使用することができます。これはいくつかの方法で設定することができます。通常のメディアからの起動時に、その他のカーネルパラメータとともに `linurc` キーワードを指定することができるほか、ネットワーク経由で起動するディスクイメージを作成して、そこにカーネルパラメータを設定する方法もあります。また、Etherboot や PXE との組み合わせで、DHCP サーバ側でカーネルのパラメータを設定することもできます。



注記: autoyast ではなく autoyast2 のオプションを使用すべき点について

`autoyast2` オプションと `autoyast` オプションは似通った仕組みですが、`linuxrc` は指定された値を処理するだけでなく、必要に応じてネットワークの設定も行おうとします。このオプションについては本文書内では説明していません。AutoYaST での違いと `linuxrc` URI の書式について、詳しくは 付録C 高度な `linuxrc` オプション にある `linuxrc` の付録をお読みください。ここでは AutoYaST のルールやクラスには 対応していません 。

コマンドライン変数 `autoyast` には、下記の表に示すような書式を使用することができます。

URI の書式

使用したい制御ファイルの場所を指定するために `autoyast` で指定する URI の書式は、少し混乱を生むものになってしまっています。書式は一般に `スキーマ://ホスト/ファイルへのパス` の形式で記述しますが、スキーマの後ろに付与するスラッシュ記号の数は、使用するスキーマによって異なります。たとえば NFS サーバ内を指定する場合は、`autoyast=nfs://SERVER/PATH` のようにスラッシュを 2 つ並べます。

ローカルのファイルシステム内に制御ファイルを配置する場合は、たとえば `autoyast=usb:///profile.xml` のような書式になります。これは `autoyast=usb://localhost/profile.xml` と同じ意味になります。このとき、`localhost` を省略するためにスラッシュ記号を 3 回繰り返していることに注意してください。もしも `autoyast=usb://profile.xml` のように記述してしまうと、`profile.xml` がホスト名として扱われてしまいます。

制御ファイルの場所を指定する必要がない場合

アップグレード: 自動オフラインアップグレードの場合は、`autoyast` 変数を指定する必要はありません。詳しくは 手順4.1「オフラインアップグレードモードでの AutoYaST の起動」をお読みください。

新規インストール: `autoinst.xml` ファイルが下記 3 箇所のうちのいずれかの場所に存在すると、AutoYaST が開始されます:

1. インストールに使用する外付けストレージ (USB メモリなど) のルートディレクトリ
2. インストールメディアのルートディレクトリ
3. システムを起動する際に使用する初期 RAM ディスクのルートディレクトリ

`autoyast=file:///パス`

指定したパス内の制御ファイルを読み込みます。この場合、ソースメディアのルートディレクトリからの相対パスを指定します。たとえば CD や USB メモリから起動している場合、それが外付けのドライブであったとしても、`file:///autoinst.xml` のように指定すると、任意のローカルファイルシステム内のルートディレクトリにあるファイルを読み込むようになります (これは `file://localhost/autoinst.xml` と同じ意味になります) 。

`autoyast=device://デバイス/ファイル名`

指定したストレージデバイス内の制御ファイルを読み込みます。この場合、デバイスに対するフルパスではなく、デバイス名のみを指定してください (例: `vda1/autoyast.xml`) 。また、デバイス名を省略する (例: `autoyast=device://localhost/autoinst.xml` , もしくは `autoyast=device:///autoinst.xml`) と、AutoYaST は全てのデバイスを検索するようになります。

`autoyast=nfs://サーバ/パス`
: NFS サーバ内の制御ファイルを読み込みます。

`autoyast=http://[ユーザ名:パスワード@]サーバ/パス`
: HTTP プロトコルを利用して、Web サーバから制御ファイルを取得します。ユーザ名とパスワードの指定は任意です。

`autoyast=https://[ユーザ名:パスワード@]サーバ/パス`
: HTTPS プロトコルを利用して、Web サーバから制御ファイルを取得します。ユーザ名とパスワードの指定は任意です。

`autoyast=tftp://サーバ/パス`
: TFTP 経由で制御ファイルを取得します。

`autoyast=ftp://[ユーザ名:パスワード@]サーバ/パス`
: FTP プロトコルを利用して制御ファイルを取得します。ユーザ名とパスワードの指定は任意です。

`autoyast=usb://パス`
: USB デバイスから制御ファイルを取得します (AutoYaST は接続されている全ての USB デバイスを検索します) 。

`autoyast=relurl://パス`
: インストール元から制御ファイルを取得します。これは既定のインストール元のほか、`install=インストール元のパス` で指定したものでもかまいません。

`autoyast=repo:/パス`
: 指定したパスから制御ファイルを取得します。パスはインストール元からの相対パスでなければなりません。

`autoyast=cifs://サーバ/パス`
: CIFS サーバ内にある制御ファイルを読み込みます。

`autoyast=label://ラベル/パス`
: 指定したラベルのデバイス内にある制御ファイルを読み込みます。

自動インストールでは、異なる種類のインフラストラクチャとソースメディアを使用するさまざまなシナリオが考えられます。最もシンプルな方法は openSUSE Leap のインストールメディア (`SLE-15-SP7-Online-ARCH-GM-media1.iso` もしくは `SLE-15-SP7-Full-ARCH-GM-media1.iso`) を使用する方法ですが、この場合は自動インストールを開始するのにコマンドライン変数を入力する必要があるほか、YaST からアクセス可能な場所に制御ファイルを配置する必要もあります。

スクリプトという観点では、マシンに対してシリアルコンソールを使用することで、テキストモードによるインストールを行う方法もあります。この場合、必要なパラメータを expect スクリプトなどで問い合わせることができます。

それぞれの起動方法に対して利用可能な、制御ファイルの提供方法を示します:

openSUSE Leap のインストールメディアを使用する方法

インストールメディア (`SLE-15-SP7-Online-ARCH-GM-media1.iso` もしくは `SLE-15-SP7-Full-ARCH-GM-media1.iso` が必要です) を使用する場合、制御ファイルは外付けストレージ (USB メモリなど) やネットワーク内に存在している必要があります:

外付けストレージ (USB メモリなど): `autoyast=usb://パス` オプションを指定して制御ファイルにアクセスさせてください。

ネットワーク: 下記のいずれかのコマンドで制御ファイルにアクセスすることができます: `autoyast=nfs://..` , `autoyast=ftp://..` , `autoyast=http://..` , `autoyast=https://..` , `autoyast=tftp://..` , or `autoyast=cifs://..` 。ネットワークへのアクセスを使用する場合は、linuxrc の起動オプションで設定を行う必要があります。たとえば DHCP を使用する場合、`netsetup=dhcp autoyast=http://163.122.3.5/autoyast.xml` のようになります。

独自のインストールメディアを使用する方法

この方法では、制御ファイルをインストールメディア内に直接含めることができます。制御ファイルはルートディレクトリ内に配置し、`autoinst.xml` というファイル名にしておいてください。これにより、インストール時に自動検出され、使用されるようになります。もちろん `autoyast=file:///パス` のように指定すれば、他のパスにある制御ファイルを指定して使用することもできます。

ネットワーク経由でインストールする方法

自動インストールではこのインストール方法が最も多いでしょう。複数のマシンにインストールを行う場合、SLP や NFS のサーバのほか、BOOTP や DHCP などのネットワークサービスを併用することが多いためです。この場合、制御ファイルの配置にあたって最も簡単な方法は、インストールソースとして使用するディレクトリのルートに、制御ファイルを配置する方法です。ファイル名は `autoinst.xml` というファイル名にしてください。この場合、インストール時に自動検出され、自動インストールが行われるようになります。制御ファイルは下記のような場所に配置することもできます:

外付けストレージ (USB メモリなど): `autoyast=usb://パス` オプションを指定して制御ファイルにアクセスさせてください。

ネットワーク: 下記のいずれかのコマンドで制御ファイルにアクセスすることができます: `autoyast=nfs://..` , `autoyast=ftp://..` , `autoyast=http://..` , `autoyast=https://..` , `autoyast=tftp://..` , `autoyast=cifs://..`



注記: ネットワークと DHCP の無効化について

インストール時にネットワークが不要である場合や利用できない場合、たとえば DVD-ROM から自動インストールを行うような場合、ネットワークを無効化するには、`linuxrc` オプションに `netsetup=0` を指定することで、ネットワークの設定を無効化することができます。

全ての AutoYaST オプションでは、下記のような方式で制御ファイルの場所を指定することができます:

1. 制御ファイルの正確な場所を指定する方法:

```
autoyast=http://192.168.1.1/control-files/client01.xml
```

2. 複数の制御ファイルが配置されているディレクトリを指定する方法:

```
autoyast=http://192.168.1.1/control-files/
```

この場合、対応する制御ファイルは IP アドレスの 16 進数表記によるファイル名で取得することになります (後述) 。
ディレクトリであることを表すため、パスは `/` で終わる必要があります。
また、このディレクトリ内のファイルには、`.xml` などの拡張子を付けてはなりません。つまり、ファイル名は IP アドレスや MAC アドレスそのものになります。

```
> ls -r control-files
C00002 0080C8F6484C default
```

パスのプレフィクスのみを指定した場合、YaST は下記の流れで制御ファイルへのアクセスを試みます:

1. まずは自分自身の IP アドレスを 16 進数に変換したファイル名の制御ファイルを検索します。たとえば `192.0.2.91 -> C000025B` のようになります。
2. このファイルが見つからない場合、YaST は 16 進数表記から 1 桁を削って再試行します。この処理は、ファイルが見つかるまで繰り返されます。それでも見つからない場合は、クライアント側の MAC アドレスからファイル名を生成してアクセスしようとします (MAC アドレスは `0080C8F6484C` のような書式になります) 。それでも見つからない場合は、`default` というファイル名 (全て小文字) でアクセスを試みます。

たとえば 192.0.2.91 であれば、HTTP クライアントは下記の順序でアクセスしようとします:

```
C000025B
C000025
C00002
C0000
C000
C00
C0
C
0080C8F6484C
default
```

"¥n ¥n"

クライアントの IP アドレスを 16 進数に変換するには、`syslinux` パッケージに含まれる `/usr/bin/gethostip` ユーティリティをお使いください。

例 9.1: IP アドレスの 16 進数表記への変換

```
> /usr/bin/gethostip 10.10.0.1
10.10.0.1 10.10.0.1 0A0A0001
```

9.3.2 単一システムの自動インストール

ネットワーク接続がない場合、自動インストールを行うのに最も簡単な方法は、openSUSE Leap の DVD-ROM と外付けストレージ (USB メモリなど) を使用する方法です。この場合、インストールサーバもネットワーク環境も用意する必要がありません。

制御ファイルを作成して `autoinst.xml` というファイル名で保存します。あとは USB メモリに `autoinst.xml` ファイルをコピーします。

9.3.3 AutoYaST 制御ファイルと `linuxrc` info ファイルの組み合わせ

`info` ファイルや起動オプションを利用して `linuxrc` に情報を渡す場合、XML の制御ファイルから設定を行うことができます。下記に示す例のようにして、`info_file` セクションを設定してください。このセクションには 1 行に 1 つずつ、コロン区切りでキーワードと値の対を指定します。

例 9.2: AUTOYAST 制御ファイル内での `linuxrc` オプション

```
....
  <install>
```

```
....
    <init>
        <info_file>

install: nfs://192.168.1.1/CDs/full-x86_64
dud: https://example.com/driver_updates/filename.dud
upgrade: 1
textmode: 1
      </info_file>
    </init>
......
  </install>
  ....
```

ただし、`autoyast2` キーワードは同じファイルを参照するようにしなければならないことに注意してください。たとえば外付けのストレージ (USB メモリなど) であれば、`usb:///` を指定する必要があります。また、`info` ファイルが初期 RAM ディスク内に配置されている場合は、`file:///` を指定する必要があります。

9.4 システム設定

自動インストール時のシステム設定は、処理全体の中で最も重要な部分となります。これまでの章で説明してきたとおり、インストール先のシステムはほぼ全て自動設定することができます。また、あらかじめ定義されているディレクティブに加えて、さまざまな事後スクリプトを使用することで、さまざまなシステム設定を調整することができます。必要であればシステムの変数を変更したり、インストール先のシステムに設定ファイル全体をコピーしたりするようなことも実現できます。

9.4.1 インストール後およびシステムの設定

インストール後の処理やシステムの設定処理は、インストール先のシステムへのパッケージインストールが完了すると即時に実行され、初回の再起動後にも続けて行われます。

システムの初回起動よりも前に、AutoYaST はインストール時に収集した全てのデータと、ブートローダの設定を指定された場所に保存します。これらの通常の処理に加え、AutoYaST では制御ファイル内に指定された chroot 環境スクリプトも実行されます。ただし、これらのスクリプトはシステムがまだマウントされていない状態で実行されます。

既定とは異なるカーネルをインストールしている場合は、ハードウエアレベルでの再起動が必要となります。ハードウエアレベルでの再起動は、インストールされているカーネルとは無関係に強制することもできます。ハードウエアレベルでの再起動を強制したい場合は、`general` リソース内の `reboot` を設定してください (詳しくは 4.1項「一般オプション」をお読みください) 。

9.4.2 システムのカスタマイズ

インストールの第 2 ステージでは、ほとんどのシステムのカスタマイズを完了させます。AutoYaST リソースでは実現できないようなカスタマイズが必要となる場合は、インストール後スクリプトを設定して、その中でカスタマイズを実施してください。

制御ファイル内には独自のスクリプトを必要な数だけ定義することができます。また、この設定を行うにあたっては、制御ファイルの編集のほか、設定システムを使用して行うこともできます。

V インストール済みのシステムに対する AutoYaST の使用

10 インストール済みのシステム内での AutoYaST の実行 236

10 インストール済みのシステム内での AutoYaST の実行

改訂履歴

2024-04-08

既に稼働中のシステムであっても、AutoYaST を実行したほうが便利な場合があります。なお、この場合は `partitioning` セクションが無視されることに注意してください。

たとえば下記の例では、追加のソフトウエアパッケージ (`foo`) をインストールします。このソフトウエアを動作させるには、NTP クライアントをインストールして設定する必要もあります。

下記の AutoYaST プロファイルでは、パッケージのインストール (4.9.7項「第 2 ステージでのパッケージのインストール」) のほか、ユーザの追加 (4.30.1項「ユーザ」) や NTP クライアントの設定 (4.21項「NTP クライアント」) をそれぞれ行うセクションが記述されています:

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://www.suse.com/1.0/
configns">
  <ntp-client>
    <peers config:type="list">
      <peer>
        <address>us.pool.ntp.org</address>
        <comment/>
        <options> iburst</options>
        <type>server</type>
      </peer>
    </peers>
    <start_at_boot config:type="boolean">true</start_at_boot>
    <start_in_chroot config:type="boolean">false</start_in_chroot>
    <sync_interval config:type="integer">5</sync_interval>
    <synchronize_time config:type="boolean">false</synchronize_time>
  </ntp-client>
  <software>
    <post-packages config:type="list">
      <package>ntp</package>
      <package>yast2-ntp-client</package>
      <package>foo</package>
    </post-packages>
  </software>
  <users config:type="list">
    <user>
      <encrypted config:type="boolean">false</encrypted>
      <fullname>Foo user</fullname>
      <gid>100</gid>
      <home>/home/foo</home>
      <password_settings>
```

```
        <expire/>
        <flag/>
        <inact/>
        <max>99999</max>
        <min>0</min>
        <warn>7</warn>
      </password_settings>
      <shell>/bin/bash</shell>
      <uid>1001</uid>
      <user_password>linux</user_password>
      <username>foo</username>
    </user>
  </users>
</profile>
```

上記のファイルを `/tmp/install_foo.xml` というパスに保存した場合、下記のようなコマンドを実行することで AutoYaST の処理を開始することができます:

```
> sudo yast2 ayast_setup setup filename=/tmp/install_foo.xml dopackages="yes"
```

詳しくは `yast2 ayast_setup longhelp` を実行して表示される内容をお読みください

VI 付録

A ルールの処理 239

B AutoYaST FAQ - よくある質問とその回答 240

C 高度な `linuxrc` オプション 244

D GNU ライセンス 250

A ルールの処理

改訂履歴

2022-02-11

下記の図には、ルールの処理と取得、そして合成に至るまでの流れが書かれています。

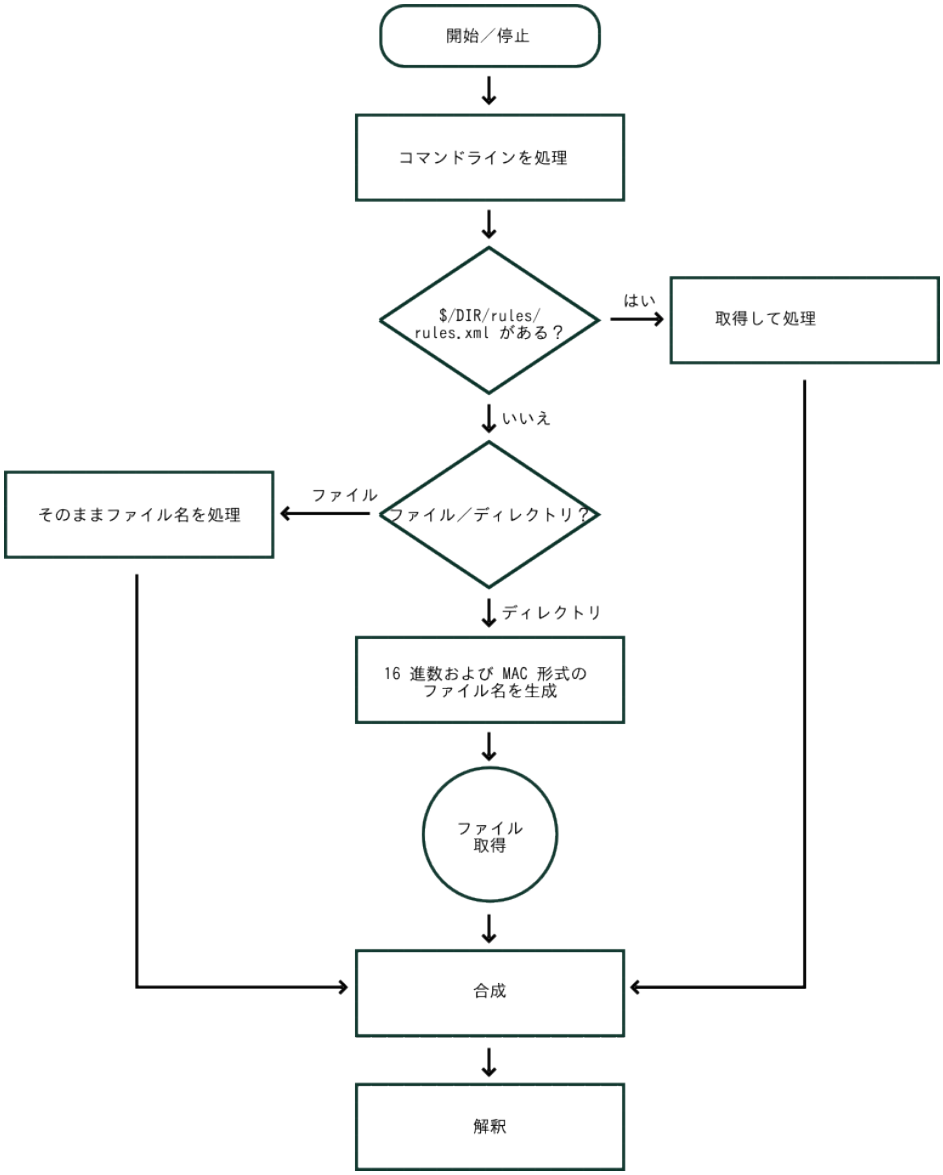

図 A.1: ルールの取得処理

B AutoYaST FAQ - よくある質問とその回答

改訂履歴

2026-09-01

問: 1. AutoYaST のインストールを開始するには?

openSUSE Leap では、どのバージョンであっても、カーネルのコマンドラインに `autoyast=<プロファイルの_URI>` を指定することで、自動インストールを開始することができます。たとえば `autoyast=http://MYSERVER/MYCONFIG.xml` のように指定すると、`myserver` という Web サーバから AutoYaST の制御ファイルを取得して、自動インストールを開始する意味になります。詳しくは 9.3項「自動インストール処理の開始」をお読みください。

問: 2. AutoYaST プロファイルとは?

プロファイルとは AutoYaST 設定ファイルや制御ファイルとも呼ばれ、このファイルの中にはシステムの設定に関する詳細や、インストールすべきパッケージの一覧などが書かれています。それ以外にも、パーティションの設定やネットワークの設定、ソフトウエアのインストール元などが含まれています。ほとんど全ての設定は、動作中のシステムで YaST を利用することで作成することができます。プロファイルは ASCII XML 形式で書かれています。

問: 3. AutoYaST プロファイルを作成するには?

最も簡単な方法は、既にインストール済みの openSUSE Leap のシステムを雛形として使用する方法です。既にインストールの完了しているシステムで [YaST] › [その他] › [自動インストールの設定] を選択し、メニューから [ツール] › [参照プロファイルの作成] を選びます。あとはプロファイル内に含めたいシステムコンポーネントを選択します。このほか、コマンドラインから `sudo yast clone_system` を実行することでも、完全なシステム設定を含むプロファイルを作成することができます。

いずれの方法であっても、ファイルは `/root/autoinst.xml` 内に保存されます。コマンドラインから作成した場合は、プロファイルを作成したシステムと全く同一の環境になるようにプロファイルが作られます。しかしながら、通常はいくつかの箇所に対して調整を行う必要があるはずです。調整作業はテキストエディタや XML エディタで実施してください。

問: 4. 作成した AutoYaST プロファイルの文法をチェックするには?

作成した AutoYaST プロファイルをチェックするための最も効率的な方法は、`jing` もしくは `xmllint` を使用する方法です。

詳しくは 3.3項「手作業による制御ファイルの作成および編集」をお読みください。

問: 5.使用できる最小限の AutoYaST プロファイルが知りたい。

AutoYaST プロファイル内に何もセクションを定義していない場合、一般的な YaST の提案内容がそのまま適用されますが、少なくともインストール後にログインができなければならないはずですので、`root` のパスワードのみ指定した下記のプロファイルが最小となります。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://
www.suse.com/1.0/configns">
  <users config:type="list">
    <user>
      <encrypted config:type="boolean">false</encrypted>
      <user_password>linux</user_password>
      <username>root</username>
    </user>
  </users>
</profile>
```

問: 6.サウンドカードを自動検出して自動インストールするには?

プロファイル内に下記の `sound` セクションを設定してください:

```
<sound>
  <autoinstall config:type="boolean">true</autoinstall>
  <configure_detected config:type="boolean">true</configure_detected>
</sound>
```

問: 7.DVD のみを使用してインストールを行いたい。AutoYaST プロファイルにはどのように設定すればいいのか?

DVD のルートディレクトリにプロファイルを配置し、`file:///プロファイル名.xml` としてアクセスしてください。

問: 8.コマンドラインで合成処理をテストするには?

たとえば `a.xml` と `base.xml` という、2 つのプロファイルを合成する場合は、下記のようなコマンドを実行します:

```
> /usr/bin/xsltproc --novalid --param replace "'false'" \
--param dontmerge1 "'package'" --param with "'a.xml'" --output out.xml \
/usr/share/autoinstall/xslt/merge.xslt base.xml
```

上記を実行するには、両方のプロファイルがアルファベット順に並んでいる必要があります (たとえば <software> セクションが <add-on> セクションの後ろに存在していなければなりません) 。YaST でプロファイルを作成していれば、プロファイルはアルファベット順に並べられます。

なお、`dontmerge1` パラメータは任意で指定するもので、プロファイル内に `dont_merge` を使用している場合にどのようにすべきなのかを例示しているものです。詳しくは 6.4項「ルールとクラスの合成」をお読みください。

問: 9. スクリプトから zypper コマンドを呼び出せるか?

zypper は AutoYaST の準備スクリプト (init-script) からのみ呼び出すことができます。これは、インストール後スクリプトの段階では、YaST がまだ RPM データベースに対する排他ロックを取得したままの状態であるためです。

準備スクリプト以外 (たとえばインストール後スクリプト) でどうしても使用したい場合は、下記のようにして自己責任でロックを外してください:

```
<post-scripts config:type="list">
  <script>
    <filename>yast_clone.sh</filename>
    <interpreter>shell</interpreter>
    <location/>
    <feedback config:type="boolean">false</feedback>
    <source><![CDATA[#!/bin/sh
mv /var/run/zypp.pid /var/run/zypp.sav
zypper in foo
mv /var/run/zypp.sav /var/run/zypp.pid
]]></source>
  </script>
</post-scripts>
```

問: 10. AutoYaST プロファイル内では、セクションの順序に注意する必要があるか?

実際のところ、順序は重要ではありません。プロファイル内におけるセクションの順序は、AutoYaST の処理手順には何も影響しません。しかしながら、異なるプロファイル同士を 合成 したい場合は、セクションはアルファベット順に並べる必要があります。

問: 11. `linuxrc` で `File not signed` と表示されてしまいインストールが進まない。どのように対応すれば自動化できるか?

`linuxrc` はドライバ更新などのファイルに対して、署名されているかどうかのチェックを行います。未署名のファイルを受け付けるには、`linuxrc` のパラメータリスト内に `insecure=1` を設定してください (自動インストールなので `autoyast=...` パラメータも必要です) 。

問: 12. DVD/USB/HD のみを使用してインストールを行いたいが、XML ファイルだけはネットワークから取得したい。どのように設定すればよいのか?

`linuxrc` に対して `ifcfg` のパラメータを設定する必要があります。これはネットワークの設定を行う際には必須のパラメータで、指定を行わないと、プロファイルをリモートからダウンロードできなくなってしまいます。詳しくは C.3項「高度なネットワーク設定」をお読みください。

問: 13. NFS をルートディレクトリとして使用する環境の自動インストールには対応しているのか?

対応していますが、他の方式よりは複雑な構成になります。環境 (DHCP, TFTP など) は注意して構築しなければなりません。また、AutoYaST のプロファイルは下記のようになります:

```
<?xml version="1.0"?>
<!DOCTYPE profile>
<profile xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://
www.suse.com/1.0/configns">
  <partitioning config:type="list">
    <drive>
      <device>/dev/nfs</device>
      <initialize config:type="boolean">false</initialize>
      <type config:type="symbol">CT_NFS</type>
      <partitions config:type="list">
        <partition>
          <filesystem config:type="symbol">nfs</filesystem>
          <fstopt>nolock</fstopt>
          <device>10.10.1.53:/tmp/m4</device>
          <mount>/</mount>
        </partition>
      </partitions>
      <use>all</use>
    </drive>
  </partitioning>
</profile>
```

問: 14. ここに掲載されていない質問に対する回答を得るには?

AutoYaST のメーリングリスト (英語のみ) で質問を受け付けております。詳しくは https://lists.opensuse.org/opensuse-autoinstall/ をお読みください。

C 高度な linuxrc オプション

改訂履歴

2022-02-11

linuxrc はカーネルが読み込まれた直後、AutoYaST やその他のステージが動作するよりも前に動作する小さなプログラムです。このプログラムはインストール向けにシステムの準備を行います。このプログラムはモジュールの読み込みのほか、インストール済みのシステムの起動やレスキュー (救出) システムの起動、そして YaST の操作ガイドなどの機能を提供します。



注記: AutoYaST と linuxrc の設定が等価ではない件について

linuxrc の設定項目によっては、autoyast.xml ファイル内で記述する AutoYaST の設定項目と偶然に名前が一致しているものがあります。ですが、これは単なる偶然であって、linuxrc と AutoYaST の設定が同一というわけではありませんし、取り得る値も場合によっては異なります。たとえば AutoYaST には self_update という設定がありますが、これを 1 に設定すると、もう 1 つの設定である self_update_url が読み込まれ、解釈されます。linuxrc にも同じ self_update という設定がありますが、こちらの場合は 0 もしくは URL を指定します。

このようなことから、AutoYaST のパラメータを linuxrc にそのまま渡してはなりません。このようにしてしまうと、ほとんどの場合において予期しない結果になってしまいます。

linuxrc がマシンにインストールされていれば、/usr/share/doc/packages/linuxrc/ ディレクトリ内に linuxrc に関する情報が書かれたファイルがあります。それ以外にも、https://ja.opensuse.org/SDB:Linuxrc でも情報が提供されています。



注記: インストール済みのシステムにおける linuxrc の実行について

linuxrc をインストール済みのシステムで動作させた場合は、インストール済みのシステムを破壊することがないよう、少し異なる動作になります。そのため、インストール済みのシステムでは全ての機能をテストすることができません。

なお、linuxrc のバイナリファイルをできる限り小さくまとめたい目的により、ライブラリとその他のファイルはバイナリファイル内に全て組み込まれています。このような構造により、初期 RAM ディスク (initrd) 内で動作させるような場合でも、共有ライブラリを別途インストールする必要がなくなっています。

C.1 linuxrc へパラメータを渡す方法

linuxrc を手動モードで動作させた場合を除いて、linuxrc を動作させると info ファイルを検索する処理を行います。info ファイルはまず外付けストレージ (USB メモリなど) 内のルートディレクトリ (/info) を検索し、そこに存在しなかった場合は initrd 内のルートディレクトリを検索します。その後、カーネルのコマンドラインを処理してパラメータを取得します。なお、linuxrc が読み込むべき info ファイルは、info というコマンドラインパラメータで変更することができます。また、linuxrc でカーネルのコマンドラインを読み込まないように設定したい場合 (たとえば linuxrc 側でも解釈されてしまうようなパラメータをカーネルに指定したい場合) は、linuxrc=nocmdline を設定してください。

linuxrc は /linuxrc.config と呼ばれるファイルを常に検索して処理します。このファイルには必要な既定値を設定しておいてください。また、一般的には info ファイルを使用したほうがよいでしょう。なお、/linuxrc.config ファイルは info ファイルよりも先に読み込まれるほか、手動モードであっても読み込まれます。

C.2 info ファイルの書式

で始まる行はコメント行です。また、項目は下記の書式で記述します:

```
キー: 値
```

なお、値 は行の終わりまでを読み込む仕組みであることから、行末にスペースが入っていてもかまいません。また、キー は大文字と小文字を区別しません。

linuxrc の設定はカーネルのコマンドラインからも設定することができます。カーネルのコマンドラインで設定する場合は、キー=値 の書式となります。ただし、カーネルのコマンドラインではスペースはパラメータの終わりとみなされます。

下記に主なキーと値の例を示しています。全ての linuxrc パラメータの一覧を読みたい場合は、https://ja.opensuse.org/SDB:Linuxrc をお読みください。

表 C.1: 高度な linuxrc のキーワード

キー: 値の例	説明
addswap: 0|3|/dev/sda5	0 を指定すると、スワップ領域に関する問い合わせを行いません。正の整数 n を指定すると、n 番目のスワップパーティションを有効化します。パーティション名を指定すると、指定したスワップパーティションを有効化します。

キー: 値の例	説明
`autoyast: ftp://AutoYaST_ファイル`	自動インストールファイルの場所を指定し、自動インストール機能を有効化します。詳しくは AutoYaST の制御ファイルの場所 をお読みください。
`bootptimeout: 10`	BOOTP リクエストに対するタイムアウト (この例では 10 秒) を指定します。
`bootpwait: 5`	ネットワークの有効化から BOOTP の開始を行うまでの待機時間 (この例では 5 秒) を指定します。
`display: color|mono|alt`	メニューの色スキームを指定します。
`exec: コマンド`	`コマンド` を実行します。
`forceinsmod: 0|1`	`insmod` コマンドを実行する際、`-f` オプションを付けて強制的に実行します。
`forcerootimage: 0|1`	インストールシステムを RAM ディスク内に読み込むようにします。
`ifcfg: ネットワーク設定`	ネットワークを設定して開始します。詳しくは C.3項「高度なネットワーク設定」をお読みください。
`insmod: モジュール`	`モジュール` で指定したモジュールを読み込みます。
`install: URL`	`URL` で指定したリポジトリからインストールを行います。`URL` の書式については https://ja.opensuse.org/SDB:Linuxrc#パラメーター一覧 (https://ja.opensuse.org/SDB:Linuxrc#.E3.83.91.E3.83.A9.E3.83.A1.E3.83.BC.E3.82.BF.E4.B8.80.E8 をお読みください。
`keytable: jp106`	読み込むべき仮想コンソールのキーボードマップを指定します。
`language: ja_JP`	インストール時にあらかじめ選択する言語を設定します。
`loghost: 10.10.0.22`	syslog (UDP ポート番号 514) を介してリモートにログを書き込むようにします。

キー: 値の例	説明
`loghost: @10.10.0.22`	syslog (TCP ポート番号 514) を介してリモートにログを書き込むようにします。
`memloadimage: 50000`	空きメモリ量が指定した値 (単位: キロバイト, この例では 50000 KB) より多い場合、インストールシステムを RAM ディスクに読み込むようにします。
`memlimit: 10000`	空きメモリ量が指定した値 (単位: キロバイト, この例では 10000 KB) より少ない場合、スワップ領域を追加すべきかどうかを尋ねるようにします。
`memYaST: 20000`	空きメモリ量が指定した値 (単位: キロバイト, この例では 20000 KB) より少ない場合、YaST をテキストモードで動作させるようにします。
`memYaSTText: 10000`	空きメモリ量が指定した値 (単位: キロバイト, この例では 10000 KB) より少ない場合、YaST を開始する前にスワップ領域を追加すべきかどうかを尋ねるようにします。
`proxy: http://10.10.0.1:3128`	HTTP プロキシサーバを指定します。書式に関する詳細は https://ja.opensuse.org/SDB:Linuxrc をお読みください。
`rescue: 1|nfs://server/dir`	レスキュー (救出) システムを読み込みます。URL を指定する場合は、レスキューイメージの場所を明示的に指定します。
`rescueimage: /suse/images/rescue`	レスキュー (救出) システムのイメージの場所を指定します。
`rootimage: /suse/images/root`	インストールシステムのイメージの場所を指定します。
`textmode: 1`	YaST をテキストモードで動作させるかどうかを指定します。
`usbwait: 4`	USB モジュールの読み込み後に待機する時間を秒単位で指定します。
`y2confirm`	制御ファイル内の `confirm` パラメータを上書きするための指定で、インストールの提案時に確認を行うように設定します。

C.3 高度なネットワーク設定

`linuxrc` に対して `hostip`，`nameserver`，`gateway` のようなパラメータを指定した場合でも、ネットワークは必要な場合 (たとえば SSH や VNC 経由でインストールを行う指定をしている場合など) にしか開始されません。これは、`autoyast` は `linuxrc` のパラメータではないためで、`linuxrc` から YaST にそのまま渡されるためです。そのため、AutoYaST のプロファイルとしてリモートの場所を指定しても、ネットワークが自動的に有効化されることは ありません 。

そのため、ネットワーク機能を使用する場合は、明示的に有効化する必要があります。ネットワークを明示的に有効化するには `ifcfg` パラメータを使用します。`ifcfg` は `/etc/sysconfig/network/ifcfg-*` ファイルに含まれる内容を直接制御することができます。

DHCP を使用する場合の設定

DHCP を使用する場合は、下記のような書式を使用します:

```
ifcfg=インターフェイス名=DHCP*,オプション_1=値_1,オプション_2=値_2
```

ここで、`インターフェイス名` にはインターフェイスの名前を指定します。`eth0` のように明示的に指定することができるほか、`eth*` のように指定すると、全てのインターフェイスをまとめて指定することができます。また、`DHCP*` には `dhcp` (IPv4 および IPv6 での DHCP), `dhcp4` (IPv4 のみでの DHCP), `dhcp6` (IPv6 のみでの DHCP) のいずれかを指定します。
たとえば `eth0` に対して DHCP を設定するには:

```
ifcfg=eth0=dhcp
```

全てのインターフェイスに対して DHCP を設定するには:

```
ifcfg=eth*=dhcp
```

固定のアドレスを使用する場合の設定

固定アドレスを使用する場合は、下記のような書式を使用します:

```
ifcfg=インターフェイス名=IP_リスト,ゲートウエイリスト,ネームサーバリスト,ドメイン検索リスト,\
オプション=値_1,...
```

ここで、`インターフェイス名` にはインターフェイスの名前を指定します。`eth0` のように明示的に指定することができるほか、`eth*` のように指定すると、最初に見つかったデバイスを使用します。その他のパラメータは上記の順序で指定する必要があります。たとえば下記のようになります:

```
ifcfg=eth0=192.168.2.100/24,192.168.5.1,192.168.1.116,example.com
```

パラメータに複数のアドレスを指定したい場合は、アドレス間をスペースで区切り、かつ文字列全体を引用符で括ってください。たとえば下記の例では、2 つのネームサーバと 2 つのドメイン検索リストを指定しています。

```
ifcfg="eth0=192.168.2.100/24,192.168.5.1,192.168.1.116 192.168.1.117,example.com
 example.net"
```

詳しくは https://ja.opensuse.org/SDB:Linuxrc をお読みください。

D GNU ライセンス

本付録には、GNU Free Documentation License バージョン 1.2 とその日本語訳 (八田真行氏 (mhatta@gnu.org) による翻訳) を収録しています。

GNU Free Documentation License

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See https://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled “GNU
Free Documentation License”.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

GNU フリー文書利用許諾契約書

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. この利用許諾契約書を、一字一句そのままに複製し頒布することは許可する。しかし変更は認めない。

0. はじめに

この利用許諾契約書の目的は、この契約書が適用されるマニュアルや教科書、その他機能本位で実用的な文書を(無料ではなく)自由という意味で「フリー」とすること、すなわち、改変の有無あるいは目的の営利非営利を問わず、文書を複製し再頒布する自由をすべての人々に効果的に保証することです。加えてこの契約書により、著者や出版者が自分たちの著作物に対して相応の敬意と賞賛を得る手段も保護されます。また、他人が行った改変に対して責任を負わずに済むようになります。

この利用許諾契約書は「コピーレフト」的なライセンスの一つであり、この契約書が適用された文書から派生した著作物は、それ自身もまた原本と同じ意味でフリーでなければなりません。この契約書は、フリーソフトウェアのために設計されたコピーレフトなライセンスであるGNU一般公衆使用許諾契約書を補足するものです。

この利用許諾契約書は、フリーソフトウェア用のマニュアルに適用することを目的として書かれました。フリーソフトウェアはフリーな文書を必要としており、フリーなプログラムはそのソフトウェアが保証するのと同じ自由を提供するマニュアルと共に頒布されるべきだからです。しかし、この契約書の適用範囲はソフトウェアのマニュアルに留まりません。対象となる著作物において扱われる主題が何であれ、あるいはそれが印刷された書籍として出版されるか否かに関わらず、この契約書は文字で書かれたいかなる著作物にも適用することが可能です。私たちとしては、主にこの契約書を解説や参照を目的とする著作物に適用することをお勧めします。

1. この利用許諾契約書の適用範囲と用語の定義

著作物がこの利用許諾契約書の定める条件の下で頒布される旨の告知を、著作権者がその中に書いたすべてのマニュアルあるいはその他の著作物は、いかなる媒体上にあってもこの契約書の適用対象となる。そのような告知を置くことで、全世界において、著作権使用料を必要とせず、許可の存続期間を限定されること無く、この契約書の中で述べられている条件の下で当該著作物を利用できるという許可を与えることとする。以下において、「『文書』(Document)」とはそのような告知が記載されたマニュアルないし著作物すべてを指す。公衆の一員ならば誰でも契約の当事者となることができ、この契約書中では「あなた」と表現される。あなたは、著作権法の下で許可を必要とするような方法で著作物を複製や改変、あるいは頒布することにより、この契約書を受諾することになる。

『文書』の「改変版 (Modified Version)」とは、一字一句忠実に複製したか、あるいは改変や他言語への翻訳を行ったかどうかに関わらず、その『文書』の全体あるいは一部分を含む著作物すべてを意味する。

「補遺部分 (Secondary Section)」とは、『文書』中でその旨指定された補遺ないし本文に先だって前付けとして置かれる一部分であり、『文書』の出版者あるいは著者と、『文書』全体の主題(あるいはそれに関連する事柄)との関係のみを論じ、全体としての主題の範疇に直接属する内容を全く含まないものである(たとえば、『文書』の一部が数学の教科書だった場合、補遺部分では数学について何も解説してはならない)。補遺部分で扱われる関係は、その主題あるいは関連する事柄との歴史的なつながりのことかも知れないし、それらに関する法的、商業的、哲学的、倫理的、あるいは政治的立場についてかも知れない。

「変更不可部分 (Invariant Sections)」とは補遺部分の一種で、それらが変更不可部分であることが、『文書』をこの利用許諾契約書の下で発表する旨述べた告知中においてその部分の題名と共に明示されているものである。ある部分が上記のような「補遺」性の定義にそぐわない場合は、その部分を「変更不可」として指定することは認められない。『文書』は、変更不可部分を全く含まなくても良い。『文書』において変更不可部分が全く指定されていなければ、その『文書』に変更不可部分は存在しないということである。

「カバーテキスト(Cover Texts)」とは、『文書』がこの利用許諾契約書の指定する条件の下で発表される旨述べた告知において、「表カバーテキスト」あるいは「裏カバーテキスト」として列挙された短い文章のことを指す。表カバーテキストは最大で5語、裏カバーテキストは最大で25語までとする。

『文書』の「透過的」複製物とは、機械による読み取りが可能な『文書』の複製物のことを指す。透過的な複製物の文書形式は、その仕様が一般の人々に入手可能で、『文書』の内容を一般的なテキストエディタ、または(画素で構成される画像ならば)一般的なペイントプログラム、あるいは(図面ならば)いくつかの広く入手可能な製図エディタで簡単に改訂するのに適しており、なおかつテキストフォーマッタへの入力に適する(あるいはテキストフォーマッタへの入力に適する諸形式への自動的な変換に適する)ものでなければならない。透過的なファイル形式への複製であっても、マークアップ、あるいはマークアップの不在が読者によるそれ以降の改変をわざと邪魔し阻害するように仕組まれたものは透過的であるとは見做されない。ある画像形式が、相当量のテキスト文章を表現するために使われた場合、それは透過的ではない。透過的ではない複製は「非透過的」複製と呼ばれる。

透過的複製に適した形式の例としては、マークアップを含まないプレーンなASCII形式、Texinfo入力形式、LaTeX入力形式、一般に入手可能なDTDを用いたSGMLあるいはXML、または人間による改変を想定して設計された、標準に準拠したシンプルなHTMLやPostScript、PDFなどが挙げられる。透過的な画像形式の例には、PNGやXCF、JPGが含まれる。非透過な形式としては、独占的なワードプロセッサでのみ閲覧編集できる独占的なファイル形式、普通には入手できないDTDまたは処理系を使ったSGMLやXML、ある種のワードプロセッサが生成する、出力のみを目的とした機械生成のHTMLやPostScript、PDFなどが含まれる。

「題扉 (Title Page)」とは、印刷された書籍に於いては、実際の表紙自身のみならず、この利用許諾契約書が表紙に掲載することを義務づける文章や図などを、読みやすい形で載せるのに必要なだけの、表紙に引き続く数ページをも意味する。表紙に類するものが無い形式で発表される著作物においては、「題扉」とは本文の始まりに先だって、その著作物の題名が最も目立つ形で現れる場所の近くに置かれる文章のことを指す。

「XYZと題された (Entitled XYZ)」部分とは、『文書』において「XYZ」と名付けられた一部分であり、その題名は正確に「XYZ」であるか、「XYZ」を他の言語に翻訳した上でその後ろに「XYZ」をそのまま括弧で括ったものを含む記述のどちらかである(ここでの「XYZ」とは、この利用許諾契約書において以下で言 及される特定の部分名を意味している。例えば「謝辞 (Acknowledgements)」、「献辞 (Dedications)」、「推薦の辞 (Endorsements)」、「履歴 (History)」)。あなたが『文書』を改変する場合、そのような部分の「題名を保存する (Preserve the Title)」とは、「XYZと題された」部分として、ここでの定義に従い題名を残すということである。

『文書』は、「保証否認警告 (Warranty Disclaimers)」を、この利用許諾契約書が『文書』に適用されると述べた告知の次に含んでも良い。この種の保証否認警告は、この契約書からの言及という形で利用条件に含まれるものと解されるが、保証の否認に関することについてのみ有効とする。こういった保証否認警告で示しうるその他のいかなる含意も無効であり、この契約書の効能には何ら影響を持たない。

2. 逐語的に忠実な複製

この利用許諾契約書、著作権表示、この契約書が『文書』に適用される旨述べた告知の三つがすべての複製物に複製され、かつあなたがこの契約書で指定されている以外のいかなる条件も追加しない限り、あなたはこの『文書』を、商用であるか否かを問わずいかなる形でも複製頒布することができる。あなたは、あなたが作成あるいは頒布する複製物に対して、閲覧や再複製を技術的な手法によって妨害、規制してはならない。しかしながら、複製と引き換えに代価を得てもかまわない。あなたが相当量の複製物を頒布する際には、本契約書第3項で指定される条件にも従わなければならない。

またあなたは、上記と同じ条件の下で、複製物を貸与したり複製物を公に開示することができる。

3. 大量の複製

もしあなたが、『文書』の印刷された(あるいは通常は印刷された表紙を持つ媒体における)複製物を100部を超えて出版し、また『文書』の利用許諾告知がカバーテキストの掲載を要求している場合には、指定されたすべてのカバーテキストを、表カバーテキストは表表紙に、裏カバーテキストは裏表紙に、はっきりと読みやすい形で載せた表紙の中に複製物本体を綴じ込まなければならない。また、両方の表紙において、それらの複製物の出版者としてのあなたをはっきりとかつ読みやすい形で確認できなければならない。表表紙では『文書』の完全な題名を、題名を構成するすべての語が等しく目立つようにして、視認可能な形で示さなければならない。それらの情報に加えて、表紙に他の文章や図などを加えることは許可される。表紙のみを変更した複製物は、それが『文書』の題名を保存し上記の条件を満たす限り、ほかの点では逐語的に忠実な複製物 として扱われる。

もしどちらかの表紙に要求されるカバーテキストの量が多すぎて読みやすく収めることが不可能ならば、あなたはテキスト先頭の一文(あるいは適切に収まるだけ)を実際の表紙に載せ、続きは隣接したページに載せるべきである。

あなたが『文書』の「非透過的」複製物を100部を超えて出版あるいは頒布する場合、それぞれの非透過な複製物と一緒に機械で読み取り可能な透過的複製物を添付するか、それぞれの非透過な複製物(あるいはそれに付属する文書)中で、公にアクセス可能なコンピュータネットワーク上の所在地を記述しなければならない。その場所には、非透過な複製物と内容的に寸分違わず、余計なものが追加されていない完全な『文書』の透過的複製物が置かれ、またそこから、ネットワークを利用する一般公衆が、一般に標準的と考えられるネットワークプロトコルを使ってダウンロードすることができなければならない。もしあなたが後者の選択肢を選ぶならば、その版の非透過な複製物を公衆に(直接、あるいはあなたの代理人ないし小売業者が)最後に頒布してから最低1年間は、その透過的複製物が指定の場所でアクセス可能であり続けることを保証するよう、非透過な複製物の大量頒布を始める際に十分に慎重な手順を踏まなければならない。

これは要望であり必要条件ではないが、『文書』の著者に、『文書』の更新された版をあなたに提供する機会を与えるため、透過非透過を問わず大量の複製物を再頒布し始める前には彼らにきちんと連絡しておいてほしい。

4. 改変

『文書』の改変版を、この利用許諾契約書と細部まで同一の契約の下で発表する限り、すなわち原本の役割を改変版で置き換えた形での頒布と改変を、その複製物を所有するすべての人々に許可する限り、あなたは改変版を上記第2項および第3項が指定する条件の下で複製および頒布することができる。さらに、あなたは改変版において以下のことを行わなければならない。

A. 題扉に(もしあればその他の表紙にも)、『文書』お よび『文書』のそれ以前の版と見分けがつく題名を載せること(もし以前の 版があれば、『文書』の「履歴 (History)」の部分に列記されているはずである)。もし元の版の出版者から許可を得たならば、以前の版と同じ題名を 使っても良い。

B. 題扉に、改変版における改変を行った1人以上の人物か団体名を列記すること。あわせて元の『文書』の著者として、最低5人(もし5人以下ならばすべて)の主要著者を列記すること。ただし元の著者たちがこの条件を免除した場合は除く。

C. 題扉に、改変版の出版者名を出版者として記載すること。

D. 『文書』にあるすべての著作権表示を残すこと。

E. 他の著作権表示の近くに、あなたの改変に対する適当な著作権表示を追加すること。

F. 著作権表示のすぐ後に、改変版をこの契約書の条件の下で利用することを公衆に対して許可する告知を含めること。その形式はこの契約書の末尾にある付記で示されている。

G. 元の『文書』の利用許諾告知に書かれた、変更不可部分の完全な一覧と、要求されるカバーテキストとを、改変版の利用許諾告 知でもそのまま残すこと。

H. この契約書の、変更されていない複製物を含めること。

I. 「履歴 (History)」と題された部分とその題名を保存し、そこに改変版の、少なくとも題名、出版年、新しく変更した部分の著 者名、出版者名を、題扉に掲載するのと同じように記載した一項を加えること。もし『文書』中に「履歴」と題された部分が存在しない場合には、『文書』の題名、出版年、著者、出版者を題扉に掲載するのと同じように記載した部分を用意し、上記で述べたような、改変版を説明する一項を加えること。

J. 『文書』中に、『文書』の透過的複製物への公共的アクセスのために指定されたネットワーク的所在地が記載されていたならば、それを保存すること。同様に、その『文書』の元になった以前の版で指定されていたネットワーク的所在地も載っていたならば、それも保存すること。これらの情報は「履歴(History)」の部分に置いても良い。ただし、それが『文書』自身より少なくとも4年前に出版された著作物の情報であったり、あるいは改変版が参考にしている版の元々の出版者から許可を得たならば、その情報を削除してもかまわない。

K. 「謝辞 (Acknowledgement)」あるいは「献辞 (Dedication)」等と題されたいかなる部分も、その部分の題名を保存し、その部分の内容(各貢献者への謝意あるいは献呈の意)と語調を保存すること。

L. 『文書』の変更不可部分を、その本文および題名を変更せずに保存すること。章番号やそれに相当するものは部分の題名の一部とは見做さない。

M. 「推薦の辞 (Endorsement)」というような章名が題された部分はすべて削除すること。そのような部分を改変版に含めてはならない。

N. すでに存在する部分を「推薦の辞 (Endorsement)」と題されるように改名したり、題名の点で変更不可部分のどれかと衝突するように改名してはならない。

O. 保証否認警告を保存すること。

もし改変版に、補遺部分としての条件を満たし、かつ『文書』から複製物された文章や図などをいっさい含んでいない、前書き的な章あるいは付録が新しく含まれるならば、あなたは希望によりそれらの部分の一部あるいはすべてを変更不可と宣言することができる。変更不可を宣言するためには、それらの部分の題名を改変版の利用許諾告知中の変更不可部分一覧に追加すれば良い。これらの題名は他の章名とは全く別のものでなければならない。

含まれる内容が、さまざまな集団によるあなたの改変版に対する推薦の辞のみである限り、あなたは、「推薦の辞 (Endorsement)」と題された章を追加することができる。推薦の辞の例としては、ピアレビューの陳述、あるいは文書がある標準の権威ある定義としてその団体に承認されたという声明などがある。

あなたは、5語までの一文を表カバーテキストとして、25語までの文を裏表紙テキストとして、改変版のカバーテキスト一覧の末尾に加えることができる。一個人ないし一団体が直接(あるいは団体内で結ばれた協定によって)加えることができるのは、表カバーテキストおよび裏カバーテキストとしてそれぞれ一文ずつのみである。もし以前すでにその文書において、表裏いずれかの表紙にあなたの(またはあなたが代表する同じ団体内で為された協定に基づく)カバーテキストが含まれていたならば、あなたが新たに追加することはできない。し かしあなたは、その古い文を加えた以前の出版者から明示的な許可を得たならば、古い文を置き換えることができる。

『文書』の著者あるいは出版者は、この利用許諾契約書によって、彼らの名前を利用することを許可しているわけではない。彼らの名前を改変版の宣伝に使ったり、改変版への明示的あるいは黙示的な保証のために使うことを許可するものではない。

5. 文書の結合

あなたは、上記第4項において改変版に関して定義された条件の下で、この利用許諾契約書の下で発表された複数の文書を一つにまとめることができる。その際、原本となる文書にある変更不可部分を全て、改変せずに結合後の著作物中に含め、それらをあなたが統合した著作物の変更不可部分としてその利用許諾告知において列記し、かつ原本にある全ての保証否認警告を保存しなければならない。

結合後の著作物についてはこの契約書の複製物を一つ含んでいればよく、同一内容の変更不可部分が複数ある場合には一つで代用してよい。もし同じ題名だが内容の異なる変更不可部分が複数あるならば、そのような部分のそれぞれの題名の最後に、(もし分かっているならば)その部分の原著者あるいは出版者の名前で、あるいは他と重ならないような番号を括弧で括って記載することで、それぞれ見分けが付くようにしなければならない。結合後の著作物の利用許諾告知における変更不可部分の一覧においても、章の題名に同様の調整をすること。

結合後の著作物においては、あなたはそれぞれの原本の「履歴 (History)」と 題されたあらゆる部分をまとめて、「履歴 (History)」と題された一章にしなければならない。同様に、「謝辞 (Acknowledgements)」あるいは「献辞 (Dedications)」と題されたあらゆる部分もまとめなければならない。あなたは「推薦の辞 (Endowsements)」と題されたあらゆる部分も削除しなければならない。

6. 文書の収集

あなたは、この利用許諾契約書の下で発表された複数の文書で構成される収集著作物を作ることができる。その場合、それぞれの文書が逐語的に忠実に複製されることを保障するために他のすべての点でこの契約書の定める条件に従う限り、さまざまな文書中のこの契約書の個々の複製物を、収集著作物中に複製物を一つ含めることで代用することができる。

あなたは、このような収集著作物から文書を一つ取り出し、それをこの契約書の下で頒布することができる。ただしその際には、この契約書の複製物を抽出された文書に挿入し、またその他すべての点でこの文書の逐語的に忠実な複製に関してこの契約書が定める条件に従わなければならない。

7. 独立した著作物の集積

『文書』あるいはその派生物を、他の別の独立した文書あるいは著作物と一緒にし、一巻の記憶装置あるいは頒布媒体に収めた編集著作物は、編集に起因する著作権が編集著作物に含まれる個々の著作物がその利用者に許可した法的権利を制限するよう行使されない限り、「集積」著作物と呼ばれる。『文書』が集積著作物に含まれる場合、この契約書は、『文書』と共にまとめられた他の独立した著作物には、それら自身が『文書』の派生物で無い限り適用されることにはならない。

このような『文書』の複製物において、この利用許諾契約書の第3項によりカバーテキストの掲載が要求されている場合、『文書』の量が集積著作物全体の2分の1以下であれば、『文書』のカバーテキストは集積著作物中で『文書』そのものの周りを囲む中表紙、あるいは『文書』が電子的形式である場合には表紙の電子的等価物にのみ配置するだけでよい。その場合以外は、カバーテキストは集積著作物全体を取り巻く印刷された表紙に掲載されなければならない。

8. 翻訳

翻訳は改変の一種と見做すので、あなたは『文書』の翻訳をこの利用許諾契約書の第4項の定める条件の下で頒布することができる。変更不可部分を翻訳によって置き換えるには著作権者の特別許可を必要とするが、元の変更不可部分に追加する形で変更不可部分の全てないし一部の翻訳を含めることはかまわない。この契約書や『文書』中の利用許諾告知、保証否認警告すべての英語原本も含める限り、あなたはこの契約書、告知、警告の翻訳を含めることができる。契約書や告知、警告に関して翻訳と英語原本との間に食い違いが生じた場合、英語原本が優先される。

典型的な例として、『文書』のある部分が原文で「Acknowledgements」、「Dedications」、あるいは「History」と題されていた場合、実際の題名を変更するには、題名を保存する(この契約書の第1項)ための条件(同第4項)を満たすことが必要となる。

9. 契約の終了

この利用許諾契約書の下で明確に提示されている場合を除き、あなたは『文書』を複製、改変、サブライセンス、あるいは頒布してはならない。このライセンスで指定されている以外の、『文書』の複製、改変、サブライセンス、頒布に関するすべての企ては無効であり、この契約書によって保証されるあなたの権利を自動的に終結させることとなる。しかし、この契約書の下であなたから複製物ないし諸権利を得た個人や団体に関しては、そういった人々がこの契約書に完全に従ったままである限り、彼らに与えられた許諾は終結しない。

10. 将来における本利用許諾契約書の改訂

フリーソフトウェア財団は、時によってGNUフリー文書利用許諾契約書の新しい改訂版を出版することができる。そのような新版は現在の版と理念においては似たものになるであろうが、新たに生じた問題や懸念を解決するため細部においては違ったものになるだろう。詳しくは https://www.gnu.org/copyleft/ を参照せよ。

GNUフリー文書利用許諾契約書のそれぞれの版には、新旧の区別が付くようなバージョン番号が振られている。もし『文書』において、この契約書のある特定の版か「それ以降のどの版でも」適用して良いと指定されている場合、あなたはフリーソフトウェア財団から発行された(草稿として発表されたものを除く)指定の版かそれ以降の版のうちどれか一つを選び、その条項や条件に従うことができる。もし『文書』がこの契約書のバージョン番号を指定していない場合には、あなたはフリーソフトウェア財団から今までに出版された(草稿として発表されたものを除く)版のうちからどれか一つを選ぶことができる。

ADDENDUM: この利用許諾契約書をあなたの文書に適用するには

この利用許諾契約書をあなたが書いた文書に適用するには、この契約書の複製物一つを文書中に含め、以下に示す著作権表示と利用許諾告知を題扉のすぐ後に置いて下さい:

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled “GNU
Free Documentation License”.
```

(訳: Copyright (C) 西暦年 あなたの名前. この文書を、フリーソフトウェア財団発行のGNU フリー文書利用許諾契約書(バージョン1.2かそれ以降から一つを選択)が定める条件の下で複製、頒布、あるいは改変することを許可する。変更不可部分、表カバーテキスト、裏カバーテキストは存在しない。この利用許諾契約書の複製物は「GNU フリー文書利用許諾契約書」という章に含まれている。)

もし変更不可部分や表カバーテキスト、裏カバーテキストがあれば、「変更不可部分…は存在しない。」というところを以下で置き換えてください:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

(訳: (章の題名を列記)は変更不可部分であり、(表カバーテキストを列記)は表カバーテキスト、(裏カバーテキストを列記)は裏カバーテキストである。)

変更不可部分はあるがカバーテキストは存在しないなど、その他の三者の組み合せに関しては、状況に合わせて上記二つの選択肢を混ぜてください。

あなたの文書に、他に類を見ない独自のプログラムコードのサンプルが含まれる場合、フリーソフトウェアにおいてそのコードを利用することを許可するために、そういったサンプルに関してはこの利用許諾契約書と同時にGNU一般公衆許諾契約書のようなフリーソフトウェア向けライセンスのうちどれか一つを選択して適用してもよい、というような条件の下で発表することを推奨します。