

CA Client Automation

Linux および UNIX 用パッケージおよび インストーラ管理者ガイド

12.9



このドキュメント（組み込みヘルプシステムおよび電子的に配布される資料を含む、以下「本ドキュメント」）は、お客様への情報提供のみを目的としたもので、日本 CA 株式会社（以下「CA」）により随時、変更または撤回されることがあります。本ドキュメントは、CA が知的財産権を有する機密情報であり、CA の事前の書面による承諾を受けずに本書の全部または一部を複写、譲渡、変更、開示、修正、複製することはできません。

本ドキュメントで言及されている CA ソフトウェア製品のライセンスを受けたユーザは、社内でユーザおよび従業員が使用する場合に限り、当該ソフトウェアに関連する本ドキュメントのコピーを妥当な部数だけ作成できます。ただし、CA のすべての著作権表示およびその説明を当該複製に添付することを条件とします。

本ドキュメントを印刷するまたはコピーを作成する上記の権利は、当該ソフトウェアのライセンスが完全に有効となっている期間内に限定されます。いかなる理由であれ、上記のライセンスが終了した場合には、お客様は本ドキュメントの全部または一部と、それらを複製したコピーのすべてを破棄したことを、CA に文書で証明する責任を負います。

準拠法により認められる限り、CA は本ドキュメントを現状有姿のまま提供し、商品性、特定の使用目的に対する適合性、他者の権利に対して侵害のないことについて、黙示の保証も含めいかなる保証もしません。また、本ドキュメントの使用に起因して、逸失利益、投資損失、業務の中断、営業権の喪失、情報の喪失等、いかなる損害（直接損害か間接損害かを問いません）が発生しても、CA はお客様または第三者に対し責任を負いません。CA がかかる損害の発生の可能性について事前に明示に通告されていた場合も同様とします。

本ドキュメントで参照されているすべてのソフトウェア製品の使用には、該当するライセンス契約が適用され、当該ライセンス契約はこの通知の条件によっていかなる変更も行われません。

本書の制作者は CA および CA Inc. です。

「制限された権利」のもとでの提供：アメリカ合衆国政府が使用、複製、開示する場合は、FAR Sections 12.212、52.227-14 及び 52.227-19(c)(1)及び(2)、ならびに DFARS Section 252.227-7014(b)(3) または、これらの後継の条項に規定される該当する制限に従うものとします。

Copyright © 2014 CA. All rights reserved. 本書に記載されたすべての商標、商号、サービス・マークおよびロゴは、それぞれの各社に帰属します。

CA Technologies 製品リファレンス

このマニュアルセットで参照されている CA 製品は、以下のとおりです。

- CA Advantage® Data Transport® (CA Data Transport)
- CA Asset Intelligence
- CA APM (CA Asset Portfolio Management)
- CA Business Intelligence
- CA Common Services™
- CA DMM (CA Desktop Migration Manager)
- CA Embedded Entitlements Manager (CA EEM)
- CA MDM (CA Mobile Device Management)
- CA NSM (CA Network and Systems Management)
- CA Patch Manager
- CA Process Automation
- CA Service Desk Manager
- CA WorldView™

CA への連絡先

テクニカルサポートの詳細については、弊社テクニカルサポートの Web サイト (<http://www.ca.com/jp/support/>) をご覧ください。

目次

第 1 章：はじめに	11
Linux および UNIX のソフトウェア管理について	11
パッケージおよびインストーラの環境	12
パッケージ ユーザ インターフェース	13
複数のユーザ サポート (smgui)	16
リモート パッケージ登録	16
 第 2 章：パッケージとインストーラの開始	 17
パッケージのインストールの前提条件	17
パッケージのインストール	18
パッケージ インストール プログラム	19
パッケージおよびインストーラのインストール フォルダ	20
パッケージの削除	21
 第 3 章：Linux および UNIX 向けのパッケージとインストーラの使用	 23
PIF 製品	24
差分製品	24
パッチ製品	25
メインのパッチ パッケージ	26
パッチ パッケージの規則	26
仮想パッケージおよびエイリアス名	27
PIF 製品のコンポーネント	28
標準コンポーネント	28
プラットフォーム固有のコンポーネント	29
言語固有のコンポーネント	29
プリインストール コンポーネント	30
オプション コンポーネント	30
PIF 製品のファイル、ディレクトリ、およびシンボリック リンク	31
PIF 製品内のファイルのディスク領域の予約	32
製品のインストールおよび削除の範囲内のスクリプト	33

内部プロシージャの使用	33
シェル以外のスクリプトの使用	34
パラメータを指定したスクリプトの使用	35
スクリプト タイマの変更	35
スクリプトの進捗状況	36
PIF 製品の依存関係	36
製品のパラメータ	37
パラメータの種類	38
パラメータ値の設定方法	38
パラメータ値の種類	39
パラメータのデフォルト値	40
追加パラメータの説明	40
パラメータに関する重要事項	41
例：パラメータを使用したアクセス権の制御	41
PIF 製品用の応答ファイル	42
内部パラメータ	43
パラメータがインストーラによって解決される方法	45
PIF 製品のテンプレート	46
インストール ダイアログ ボックス	47
インストール ダイアログ ボックスの定義	49
標準インストール ダイアログ ボックス	49
タスク指向のインストール ダイアログ ボックス	50
動的に検出されたダイアログ ボックス	50
ダイアログ ボックスの検証スクリプト	51
アクション スクリプト	53
スプラッシュ画面	53
チェック ボックス ツリー	54
インストール ダイアログ ボックスのフォント指定	56
ダイアログ ボックスの入力順序の指定	57
インストール ダイアログ ボックス エディタ	59
ワークボックス	60
グラフィック領域	62
ツールボックスおよびダイアログ ボックス要素	63
プロパティ ボックス	72
さまざまなパッケージフォーマットのサポート	72
PKG、RPM、AIX、HP-SD、および OSX の各パッケージ統合の機能	73
RPM 固有の統合機能	73
RPM パッケージの使用に関する制限	74
RPM および PKG のネイティブ スイッチ	75
インストーラ機能の概要	75

解消された libncurses ライブラリへの依存関係	77
PIF 製品の自己インストール	78
自己インストール PIF 製品における JRE の統合	79
製品ファミリの概念	79
PIF 以外の製品のマイグレーション	80
インストーラの互換性についての考慮事項	80

第 4 章：パッケージ ユーザ インターフェースでの作業 83

新しい PIF パッケージの作成	83
PIF パッケージの保存	85
PIF パッケージのビルド	85
ライブラリまたはディスクへの PIF パッケージの登録	87
PIF パッケージの変更	88
PIF パッケージ用の暗号化方式の設定	89
PIF パッケージのアンロード	90
PIF パッケージの削除	90
PIF パッケージのインポート	91
多言語の PIF パッケージの作成	92
インストール ダイアログ ボックスの言語を選択します。	93
ダイアログ ボックス リソースのコピーおよび変更	95
パッチ製品作成ウィザード	95

第 5 章：サンプル PIF 製品のビルド 97

UNIX アプリケーション MERCHANT の要件および仕様	98
PIF 製品 MERCHANT のプロトタイプ ファイルの作成および製品情報の指定	98
PIF 製品 MERCHANT の依存関係の定義	100
PIF 製品 MERCHANT のファイルの指定	101
PIF 製品 MERCHANT へのプロシージャの追加	103
PIF 製品 MERCHANT のインストール パスの変更	106
PIF 製品 MERCHANT のインストール パスを問い合わせるダイアログ ボックスの追加	107
PIF 製品 MERCHANT への言語固有の製品ファイルの追加	109
PIF 製品 MERCHANT のインストール ダイアログ ボックスのローカライズ	111
コマンドラインを使用したインストール ダイアログ ボックスのローカライズ	112
GUI を使用したインストール ダイアログ ボックスのローカライズ	112
最終的な PIF 製品 MERCHANT のビルド	113

第 6 章：インストーラ タスク 117

プリインストール タスク	118
ソフトウェアの検証.....	118
プリインストール コンポーネントの一時フォルダへの抽出	118
マイグレーション スクリプトの実行	119
インストール ダイアログ ボックスの実行	120
ポストインタビュー スクリプトの実行	121
バックアップの作成.....	121
製品インストール中のタスク	121
環境からのパラメータ値の読み取り	121
プリインストール スクリプトの実行	122
コンポーネントのインストール	123
ポストインストール スクリプトの実行	123
製品情報の登録.....	123
ポストインストール タスク	123

第 7 章：参照 125

ファイルの命名規則.....	125
プロトタイプ ファイルのフォーマットと構造	126
製品情報セクション	127
コンポーネントセクション	129
デフォルトセクション (パラメータセクション)	129
ダイアログボックスセクション	130
Fontlist セクション.....	131
Desktop セクション	131
リソースセクション	132
プロトタイプ ファイルのキーワードとプロパティ	132
キーワード.....	132
プロパティ.....	144
パッケージおよびインストーラのコマンド	157
pifproto - PIF 製品のプロトタイプ ファイルの作成	158
pifmk - PIF 製品を生成.....	160
pifself--自己インストール PIF 製品を生成.....	162
pifextract - PIF 製品を抽出	164
pifdelta - 差分製品を作成	165
piflib - PIF ライブラリ ファイルを生成.....	167
pifpatch - パッチ製品を自動的に生成.....	168
sd_registerproduct - ソフトウェア パッケージ ライブラリに PIF 製品を登録.....	169

lsm - Linux および UNIX システムでソフトウェア インストールを管理	171
自己インストール PIF 製品をインストール	177
サポートされる Linux および UNIX 動作環境	179
言語識別子	180
テンプレート パラメータおよび pifproto コマンド スイッチ	181

第 8 章 : FAQ 183

ログ ファイル収集ツール dsminfo	184
自己インストール PIF 製品は分割可能	184
複数の PIF ファイルを 1 つの製品として登録	185
インストーラはカスタマ システム上のディレクトリを使用	186
パッケージはディレクトリにファイルを格納	186
X11 エミュレーションを設定する必要がある	187
パッケージおよびインストーラには特定タイプの Java Runtime Environment が必要	187
ログ ファイルに記録された製品インストール情報	188
選択的アンインストールのサポート	188
サイレント インストール中の Prepif スクリプトと応答ファイルのやり取り	189
Unicenter Software Delivery 4.0 エージェントにインストール可能な PIF SDK バージョン 4.1 のパッケージを使用して作成された PIF 製品	190
DVD ドライブのないコンピュータでのパッケージのインストール	190
パッケージをインストールする際に、警告メッセージが表示されます	191
Midnight Commander ソフトウェアがインストールされている場合、自己インストール PIF パッケージが失敗します	192

第 1 章：はじめに

このガイドでは、Linux および UNIX 向けの CA Technologies ソフトウェア管理のメインコンポーネントであるパッケージとインストーラの機能および使用方法について説明します。

ソフトウェア管理機能およびその目的についても大まかに紹介し、サンプル PIF 製品を作成するために、主要な機能も適用されます。この例は、別の章で取り上げ、サンプル PIF 製品の作成方法について手順ごとにわかりやすく説明します。

このガイドで説明するソフトウェア管理機能、およびパッケージとインストーラソフトウェアは、PIF SDK (Software Development Kit) バージョン 4.7 で使用できます。

このガイドは、Linux および UNIX ターゲットコンピュータにインストールするソフトウェア製品を開発する管理者および開発者を対象にしています。Java 環境や X エミュレーションなど、Linux および UNIX オペレーティングシステムに関する広範な知識が必要とされ、ソフトウェア製品のパッケージングとターゲットコンピュータ上の各種 Linux/UNIX 動作環境への製品のインストールについて充分に理解している必要があります。

ネットワーク経由で Linux/UNIX 製品を配布およびインストールする場合は、ソフトウェア管理の概念、使用方法、および用語について理解している必要があります。

Linux および UNIX のソフトウェア管理について

Linux および UNIX 向けの CA Technologies ソフトウェア管理ソリューションは、パッケージとインストーラという 2 つの主要コンポーネントで構成されています。Linux および UNIX 向けの CA Technologies ソフトウェア管理ツールおよび機能は、ネットワーク内のターゲットコンピュータ上の関連するすべての Linux および UNIX ベースの動作環境をサポートします。

ソフトウェア管理パッケージ（パッケージ）は、ソフトウェア製品の定義、設定、およびパッケージングに使用されます。CA Technologies では、Linux および UNIX システムのソフトウェア パッケージングの標準である **Product Interchange Format (PIF)** を提供しています。パッケージングプロセスにより作成されたものは **PIF 製品** と呼ばれます。PIF 製品は、PIF 製品 SDK（ソフトウェア開発キット）でサポートされるすべての Linux および UNIX 動作環境用に作成し、インストールすることが可能です。パッケージのすべての機能は、グラフィカル ユーザー インターフェース（Java GUI）およびコマンドライン インターフェース（CLI）の両方から使用できます。

インストーラはターゲット コンピュータ上に配置されます。インストーラでは、自動またはダイアログ ボックスを使用したインストールのいずれの場合でも、PIF SDK でサポートされるすべての動作環境に製品がインストールされます。インストール ダイアログ ボックスを使用すると、あらゆる種類のターゲット コンピュータに製品をインストールできます。

インストーラに備えられている機能を使用すると、PIF 以外のパッケージフォーマットの製品を簡単にインストールでき、ターゲット コンピュータにインストールされている Linux/UNIX 製品を管理することができます。

パッケージおよびインストーラの環境

ほとんどの場合、ソフトウェア管理はソフトウェア配信環境に統合されていますが、パッケージとインストーラをスタンドアロン モードで使用することもできます。

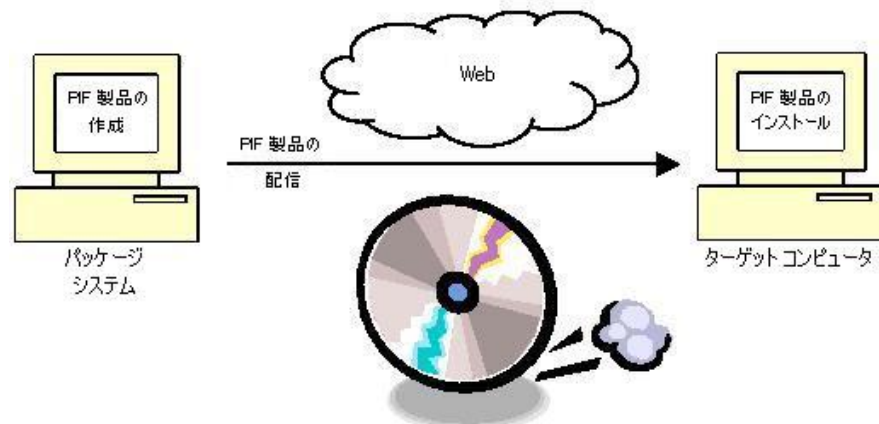
パッケージとインストーラをソフトウェア配信環境で使用する場合、パッケージを使用して、製品がパッケージング システム上で PIF フォーマットにパッケージ化されます。次に、PIF 製品がマネージャ システムのソフトウェア パッケージ ライブラリに登録され、インストールの注文を使用してソフトウェア配信機能によって配布されます。ターゲット コンピュータ上では、ソフトウェア配信 エージェント機能によってインストールの注文が受信および評価され、インストーラにより PIF 製品がインストールされます。

以下の図は、ソフトウェア配信環境でのパッケージおよびインストーラを示しています。



パッケージとインストーラをスタンドアロンモードで使用する場合、パッケージングシステムで作成された PIF 製品は、CD-ROM や DVD などのメディアを使用するか、または Web 経由でターゲットコンピュータに配布されます。ターゲットコンピュータ上で、インストーラによりメディアから PIF 製品がインストールされます。

以下の図は、スタンドアロンモードでのパッケージおよびインストーラを示しています。



パッケージ ユーザ インターフェース

ソフトウェア管理機能では、パッケージ用に使いやすいグラフィカル ユーザ インターフェース (Java GUI) が用意されていて、これを使用して、PIF 製品の定義、作成、および管理を行うことができます。

また、すべてのパッケージ機能は、コマンドライン インターフェース (CLI) でのコマンドを介して使用することができます。

パッケージの Java GUI を実行するには、パッケージング コンピュータに Java Runtime Environment (JRE) バージョン 1.5 または 1.6 がインストールされており、グラフィック表示対応モニターを使用し、\$DISPLAY 変数がローカル端末に設定されている必要があります。

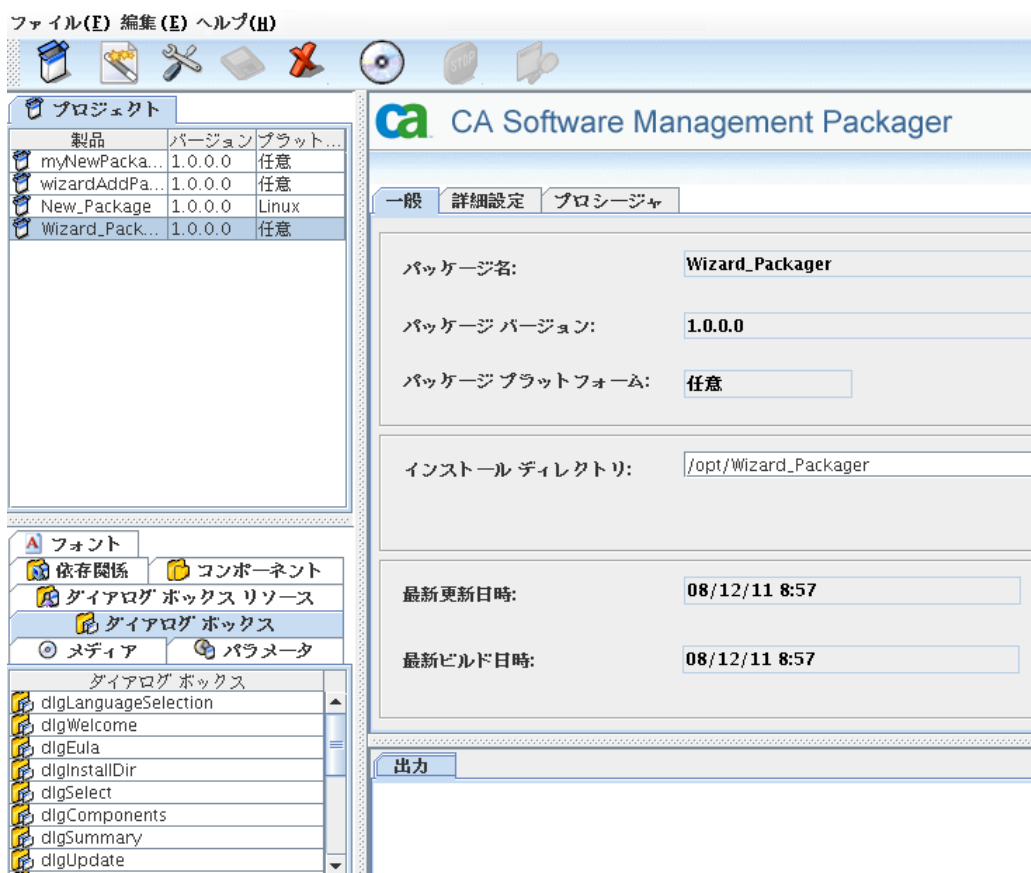
パッケージのコマンドを実行するには、コマンドラインで以下のいずれかのコマンドを入力し、新しいログイン シェルを開始する必要があります。

```
su - [user]
```

```
. /etc/profile
```

パッケージのインストール ディレクトリで smgui コマンドを実行し、パッケージ GUI を起動します。

以下に、パッケージ GUI の例を示します。



主なソフトウェア パッケージング タスクは、メニュー バーの [ファイル] メニューを選択するか、またはツールバー上のアイコンをクリックして実行することができます。

ツールバーには、[キャンセル] および [ダイアログ ボックスのテスト] 機能もあります。パッケージのオンライン ヘルプを表示するには、メニュー バーの [ヘルプ] - [トピックの検索] を選択するか、または F1 キーを押します。

以下の表に、ツールバーにあるアイコンとそれに関連する機能を示します。

アイコン	アイコン名	Description
	新規	新規 PIF 製品のプロトタイプ ファイルを作成します。
	保存	PIF 製品を保存します。
	削除	PIF 製品を削除します。
	ビルド	PIF 製品を作成します。
	キャンセル	パッケージング プロセスの実行中のサブプロセスを停止します。
	ダイアログ ボックスのテス ト	設計中のインストール ダイアログ ボックスのプレビューを表示 します。
	パッケージ ウィザード	パッケージ作成ウィザードを起動します。

プロジェクト ペインはウィンドウ左上部にあり、作成された製品およびインポートされた製品がすべて表示されます。製品を管理するには、プロジェクト ペインから対象の項目を選択する必要があります。

サブプロジェクト ペインはプロジェクト ペインの下にあり、プロジェクト ペインで選択されている製品の関連モジュール (サブプロジェクト) が表示されます。サブプロジェクト ペインの各タブには、サブプロジェクトの現在の内容が示されます。項目を選択し、作業ペインでプロパティを表示および変更することができます。

作業ペインは GUI ウィンドウの右上部にあり、選択された製品またはサブプロジェクト項目のプロパティが表示されます。

出力ペインには、パッケージ GUI から呼び出されたコマンドにより発行されたステータスおよびエラー メッセージが表示されます。出力ペインの別個のタブにエラー メッセージが表示されます。このエラー メッセージタブは、新しいビルドジョブが開始するまで表示され続けるので、エラー情報を残したままパッチ間で切り替えできます。

複数のユーザ サポート (smgui)

異なるユーザが GUI (smgui) に同時にログインできるようになりました。つまり、異なるユーザが異なるパッケージを同時に管理できます。ただし、異なるユーザが 1 つのパッケージを変更することはできません。

リモート パッケージ登録

Linux および UNIX コンピュータで、パッケージ GUI からのリモート パッケージの登録を有効にするために、または "registerproduct" コマンドを有効にするために、SD エージェントをインストールします。

第 2 章：パッケージとインストーラの開始

この章では、Linux および UNIX 向けのパッケージとインストーラの使用を開始する方法について説明します。ただし、この章の終わりには、パッケージの削除方法に関する情報も記載されています。

このセクションには、以下のトピックが含まれています。

[パッケージのインストールの前提条件](#) (P. 17)

[パッケージのインストール](#) (P. 18)

[パッケージ インストール プログラム](#) (P. 19)

[パッケージおよびインストーラのインストール フォルダ](#) (P. 20)

[パッケージの削除](#) (P. 21)

パッケージのインストールの前提条件

Linux および UNIX 向けのパッケージは、CA Client Automation エージェントがサポートするすべての 32 ビット版および 64 ビット版の Linux および UNIX プラットフォームにインストールできます。

パッケージのインストール プログラムは、パッケージをインストールするコンピュータのハードウェア設定およびソフトウェア設定に応じて、グラフィックモードまたは文字指向モードのいずれかで起動されます。

以下の前提条件を満たしている場合、パッケージのインストール ダイアログボックスが、グラフィック モードで実行されます。

- Java Runtime Environment (JRE) 1.5 または 1.6 がコンピュータにインストール済みであること
- グラフィック対応のモニタであること
- \$DISPLAY 変数がローカル ターミナルに設定されていること
- JRE が自己インストーラ ファイルに含まれていること

コンピュータがこれらの条件のいずれかを満たしていない場合、パッケージのインストール ダイアログ ボックスは、文字指向 VT100 モードで実行されます。

パッケージのインストール

ソフトウェアパッケージを作成するには、ローカルシステムにパッケージをインストールしておく必要があります。このシステムをパッケージングシステムと呼びます。

パッケージングシステムにパッケージをインストールするには、以下の手順に従います。

1. パッケージング コンピュータにユーザ `root` としてログオンします。
2. DVD/CD-ROM ドライブをマウントして、パッケージを収録したインストールメディアにアクセスできるようにします。オペレーティングシステムごとに、DVD/CD の `mount` コマンドを入力します (`mount` コマンドの使用法については、各オペレーティングシステムのマニュアルを参照)。
3. インストールメディアの該当するパッケージのディレクトリに移動します。
4. パッケージのランタイムコンポーネントのターゲットロケーションとして、デフォルト以外のロケーションを使用する場合はここで指定します。

パッケージインストールプログラムでは、そのランタイムコンポーネントの解凍先として、`/tmp` (一時) ディレクトリがデフォルトで使用されます。そのディレクトリには、ディスク空き領域として約 5 MB が必要です。

ランタイムコンポーネントの解凍先をデフォルトのディレクトリ以外にする場合は、以下のように `TMPDIR` 変数を目的のパスに設定します。

```
$TMPDIR=newpackagerpath
```

パッケージのインストールが正常に完了した後、解凍されたコンポーネントは自動的に削除されます。

5. パッケージインストールプログラム `install-sm-packager.sh` を実行します ([パッケージインストールプログラム](#) (P. 19) を参照)。
6. インストールウィザードの指示に従って、インストールを完了します。

注： インストールダイアログボックスが文字指向の VT100 の場合は、`Tab` キーを使用して一番下にあるナビゲーションボタンを選択し、キャリッジリターン (`CR`) キーを使用して選択ボタンをオンまたはオフにします。

パッケージのインストールが正常に完了した後、インストールプログラムのランタイムコンポーネントはすべて一時ディレクトリ (`/tmp` または `TMPDIR` で指定したディレクトリ) から自動的に削除されます。

パッケージ インストール プログラム

パッケージ インストール プログラム `install-sm-packager.sh` は、自己インストール型の PIF 製品です。 `install-sm-packager` を実行するときにはオプションを指定できます。

コマンドのフォーマットは、以下のとおりです。

```
sh install-sm-packager.sh [-t tracefile] [-p logfile] [-r responsefile] [-s] [-F] [-x]
```

-t *tracefile*

指定されたトレース ファイルにインストールをトレースします。

-p *logfile*

指定されたログ ファイルにインストールを記録します。

-r *responsefile*

指定された応答ファイルを使用して製品をインストールします。

-s

デフォルトのパラメータ値を使用して、自動（サイレント）モードで製品をインストールします。

-F

強制的に製品をインストールします。バックアップ エラーは無視されます。

-x

製品の `/tmp` ディレクトリへの抽出のみを行います。抽出された製品に使用されるファイル名は内部的に定義済みであるため、コマンドラインに自動的に表示されます。

-V *vt100 mode*

文字指向の VT100 モードで製品をインストールします。

-a *ask mode*

Ask モードで製品をインストールします。

-o *overwrite* (応答ファイル)

応答ファイルの生成時に、既存の応答を上書きします。

パッケージおよびインストーラのインストール フォルダ

CA Technologies のインストーラ標準では、パッケージのインストール フォルダを定義できます。

インストーラのホーム ディレクトリは、パラメータ `$INSTALLER_DIR` を使用して定義できます。このパラメータが PIF 製品の最初のインストールより前に設定されている場合、そのインストール フォルダとしてインストーラはこのパラメータの値を選択します。

パラメータ `$INSTALLER_DIR` が設定されていないときに、ほかの CA 製品が共有コンポーネント フォルダにインストールされており、参照パラメータ `$CASHCOMP` をエクスポートしてある場合は、`$CASHCOMP/installer` がインストーラのホーム ディレクトリとして使用されます。

インストーラでは、`$CASHCOMP` はグローバルに使用するものとして設定されません。つまり、ユーザは、最初の製品をインストールする前に、このパラメータを明示的に設定する必要があります。

パッケージの以前のバージョンに付属していた製品をインストールできるようにするには、以下の追加のファイルまたはリンクがシステムに必要です。

`/usr/bin/lsm`

(スクリプト) 以前のインストールで使用していたコマンドライン インターフェース (lsm) に固定します。

`/opt/CA/installer`

(リンク) 以前のインストールを新しいインストーラのホーム ディレクトリにリダイレクトします。

古いバージョンのインストーラをアップグレードする場合は、既存のインストーラのホーム ディレクトリが使用されます。新しいフォルダにマイグレーションされることはありません。

パッケージの削除

パッケージングシステムからパッケージを削除するには、以下の手順に従います。

1. `root` として、または `root` の権限を持つ ID を使用してサインオンします。たとえば、以下のコマンドを実行します。
`su - root`
2. 以下の `lsm` コマンドを実行して、パッケージを削除します。
`lsm -e SMPackager`

第 3 章：Linux および UNIX 向けのパッケージとインストーラの使用

この章では、パッケージとインストーラの特徴および機能の利用方法に関する概要について説明します。これらの機能の指定方法についてさらに理解するには、「[サンプル PIF 製品のビルド \(P. 97\)](#)」を参照してください。サンプル PIF 製品には、さまざまな機能が含まれています。

プロトタイプ ファイルの中で機能を記述するために使用されているすべてのキーワードおよびプロパティの詳細な説明については、「[キーワードとプロパティ \(P. 132\)](#)」セクションを参照してください。

このセクションには、以下のトピックが含まれています。

- [PIF 製品 \(P. 24\)](#)
- [差分製品 \(P. 24\)](#)
- [パッチ製品 \(P. 25\)](#)
- [仮想パッケージおよびエイリアス名 \(P. 27\)](#)
- [PIF 製品のコンポーネント \(P. 28\)](#)
- [PIF 製品のファイル、ディレクトリ、およびシンボリック リンク \(P. 31\)](#)
- [PIF 製品内のファイルのディスク領域の予約 \(P. 32\)](#)
- [製品のインストールおよび削除のスコープ内のスクリプト \(P. 33\)](#)
- [PIF 製品の依存関係 \(P. 36\)](#)
- [製品のパラメータ \(P. 37\)](#)
- [PIF 製品のテンプレート \(P. 46\)](#)
- [インストール ダイアログ ボックス \(P. 47\)](#)
- [インストール ダイアログ ボックス エディタ \(P. 59\)](#)
- [さまざまなパッケージ フォーマットのサポート \(P. 72\)](#)
- [インストーラ機能の概要 \(P. 75\)](#)
- [解消された libncurses ライブラリへの依存関係 \(P. 77\)](#)
- [PIF 製品の自己インストール \(P. 78\)](#)
- [自己インストール PIF 製品における JRE の統合 \(P. 79\)](#)
- [製品ファミリの概念 \(P. 79\)](#)
- [PIF 以外の製品のマイグレーション \(P. 80\)](#)
- [インストーラの互換性についての考慮事項 \(P. 80\)](#)

PIF 製品

パッケージ (Linux/UNIX) を使用してパッケージング プロセスにより作成されたものは PIF 製品と呼ばれます。

パッケージ GUI では、複数のツールやウィザードを使用して新しい製品の作成または既存の製品のコピーや編集がサポートされています。

PIF 製品の作成または修正は、ソフトウェア管理のコマンドライン インターフェース (CLI) から行うこともできます。作成したプロトタイプ ファイルは、エディタを使用して修正したり、プロトタイプ ファイルを複製し、複製を修正できます。その後、`pifmk` コマンドを実行して、変更されたプロトタイプ ファイルから新しい PIF 製品をビルドできます。

PIF 製品は、PIF SDK でサポートされるすべての Linux および UNIX 動作環境用に作成し、インストールすることが可能です。

差分製品

PIF 差分製品は、PIF パッケージ フォーマットのインストール可能な小さなユニットです。この製品には、同じ PIF 製品の 2 つのバージョン間の差分のみが含まれています。すべてのターゲット コンピュータに完全な新規の PIF 製品をインストールすることもできますが、以前のバージョンの PIF 製品がすでにインストールされているすべてのターゲット コンピュータには、小さな差分製品のみをインストールすることができます。たとえば、PIF 差分製品には、新規 PIF 製品のすべての変更されたファイルおよび新規ファイル、および新規 PIF 製品で使用されない以前の製品バージョンのファイルをアンインストールする追加の指示が含まれます。

PIF 差分製品は、標準的な PIF 製品です。

PIF 差分製品を作成するには、以下の例で示すように `pifdelta` コマンドを使用します。

例：2つの製品バージョンから差分製品を作成

以下のコマンドは、製品インストールディレクトリの `2100` および `3000` という名前のサブディレクトリにある PIF 製品 `MERCHANT` のバージョンを比較し、`deltas` という名前のサブディレクトリ内に製品バージョン `2.1.0.0` と `3.0.0.0` の差分のみを含む PIF 差分製品を作成します。

```
pifdelta -p $PATHmerchant¥2100¥MERCHANT.Any.@pif
-v $PATHmerchant¥3000¥MERCHANT.Any.@pif
-d $PATHmerchant¥deltas¥MERCHANT.Any.2.1.0.0-3.0.0.0.@pif
```

パッチ製品

PIF パッチ製品は、特定の PIF 製品を更新するためのパッチが含まれている標準の PIF 製品です。パッチ製品を使用すると、製品全体を再発行または配布することなくアプリケーションを修正できます。パッチは、**Software Delivery** 機能を使用するか、手動で適用できます。

依存関係およびインストールダイアログボックスなど、PIF 製品を定義する既存の機能のほとんどをパッチの定義に使用できます。PIF パッチ製品をインストールする場合、置換された、または削除された製品のファイルのすべては、製品のホームディレクトリに保存されます。パッチのインストールがロールバックされると、変更が取り消されます。特定の製品で、製品をアップデートしたり削除したりすると、インストール済みのパッチがすべて削除されます。

パッケージには、PIF パッチ製品を作成するためのウィザードが付いています。新規パッチ製品作成ウィザードを起動するには、更新される製品を右クリックし、コンテキストメニューから「新規パッチ」を選択します。

また、`pifpatch` コマンドを使用して、PIF パッチ製品を作成することもできます。パッチを適用する必要がある PIF 製品は、プロトタイプファイルの `#patchproduct` キーワードを使用して指定されます。`pifpatch` コマンドは、PIF デルタ製品作成メカニズムに基づき、2つのセットのパッケージを比較し、主なパッチ製品の依存構造を照会して、パッチを自動的に生成します。

累積パッチを適用する場合、現在のパッチ製品内で、現在のパッチ製品によって置換されるパッチ パッケージのリストを指定できます。`#replacepatches` キーワードを使用してプロトタイプ ファイルで置換されたパッチ パッケージのリストを指定します。

`lsm` コマンドを使用して、すべてのパッチ パッケージまたは特定の製品に適用するすべてのパッチをリストできます。

メインのパッチ パッケージ

以下のようなシナリオが考えられます。

元の **PIF** 製品は複数のパッケージから構成されていますが、この製品をインストールするときに、ユーザは特定のパッケージのサブセットのインストールを設定することにしました。つまり、元の製品は一部のみインストールされています。

このシナリオで **PIF** パッチ製品をインストールする必要がある場合には、製品は、適用する必要があるパッチ パッケージを自動的に評価する必要があります。

したがって、個別のパッチ パッケージに適切な依存関係を持った、個別のパッチ パッケージのアンカーとして動作する「メインのパッチ製品」を定義する必要があります。

メインのパッチ パッケージは、インストーラが常に必要なバージョンに更新される自己インストール パッケージとしてコンパイルできます。パッチの元のパッケージが存在する場合に限り、必要なパッチ パッケージがインストールされます。ターゲット コンピュータ上に元のパッケージがインストールされていない場合は、関連するパッチもインストールされず、エラーは返されません。このパッチ パッケージのアプリケーションは無視されます。

メインのパッチ パッケージが削除（ロールバック）されると、必要な個別のパッチ パッケージのすべても削除（ロールバック）されます。

パッチ パッケージの規則

パッチ パッケージには、以下の規則があります。

- パッチ パッケージの名前は、パッチするパッケージの名前とは異なるものにする必要があります。分かりやすくするために、パッチ パッケージの名前には発行番号またはサポート修正番号を使用できます。
- パッチ パッケージでは、[設定] ダイアログ ボックスおよび [削除] ダイアログ ボックスはサポートされていません。

- パッチパッケージのバージョンニングはサポートされていません。パッチは1回だけ適用されます。同じパッチをアップグレードすることはできません。この場合、パッチは別のパッチによって置き換えられる必要があります。
- パッチパッケージは、元のパッケージの1つのインスタンスに限り適用されます。元のパッケージの複数のインスタンスがインストールされている場合、パッチはそれぞれのインスタンスに個別に適用する必要があります。

仮想パッケージおよびエイリアス名

PROD1 などの PIF 製品内では、PROD1 製品によって機能またはサービスがカバーされる他の PIF または RPM パッケージ（仮想パッケージ）のエイリアス名を指定できます。エイリアス名には、たとえば、PROD1 の名前が最近変更された場合には、PROD1 の以前の名前を含めたり、製品 PROD1 にマージされたパッケージの名前を含めることができます。

エイリアス名の使用によって、物理パッケージの名前と論理サービスの名前が分離されます。これにより、依存関係情報を失うことなくパッケージの名前変更が可能になります。

PROD1 のインストール中に、依存関係チェックによってエイリアス名のリストが評価され、これらのパッケージへの依存関係が解決されます。そのため、このサービスやパッケージによって提供されるサービスがいずれかの製品でまだ必要である場合、インストーラではシステムからサービスやパッケージが削除されるのを防止します。

エイリアス名は、プロトタイプ ファイルの @PRODUCT セクションで #vname キーワードを使用するか、またはパッケージ GUI の [詳細設定] タブで指定します。

例：

「CAWIN」、「ca-cs-cawin」、および「CAI18N」の3つの異なる製品パッケージによって国際化サービスが提供されます。パッケージのそれぞれの内部でエイリアス名「CAWIN」を定義する場合、「CAWIN」国際化サービスを必要とする製品がある限り、これらのパッケージのいずれもシステムから削除されません。

PIF 製品のコンポーネント

PIF 製品は、1 つ以上のコンポーネントで構成されています。コンポーネント数に応じて、PIF 製品のプロトタイプファイルは、1 つ以上のコンポーネントセクションで構成されます。コンポーネントの内容は、プロトタイプファイルのコンポーネントセクションに定義されます。

コンポーネントは、その内容と目的に応じて、以下のいずれかのタイプになります。

標準コンポーネント

インストール後に製品の使用中に作成されるファイルに対するディレクトリ、製品ファイル、シンボリック リンク、およびディスク領域の予約が含まれます。

プラットフォーム固有のコンポーネント

コンポーネントに定義された動作環境にのみインストールされるディレクトリ、製品ファイル、およびシンボリック リンクが含まれます。

言語固有のコンポーネント

コンポーネントに指定された言語が、ターゲット コンピュータに定義された言語に一致する場合にのみインストールされるディレクトリ、製品ファイル、およびシンボリック リンクが含まれます。

プリインストール コンポーネント

メインの PIF 製品をインストールする前にインストールするすべてのファイルがバンドルされています。

設定コンポーネント

設定ダイアログ ボックスおよび製品の削除ダイアログ ボックスに必要なファイルをすべてバンドルします。

標準コンポーネント

標準コンポーネントには、ディレクトリ、製品ファイル、シンボリック リンク、および製品のインストール中にインストールされないファイルのディスク領域の予約など、PIF 製品のリソースのすべての定義が含まれます。これらの項目は、製品インストールの主要な部分において処理されます。標準コンポーネントは、ターゲット コンピュータにインストールされます。

プラットフォーム固有のコンポーネント

PIF 製品のコンポーネントは、通常、すべての UNIX および Linux 動作環境にインストール可能です。コンポーネントを特定の動作環境にのみインストールする場合は、プラットフォーム固有のコンポーネントを定義します。プロトタイプファイルで `#sys:` キーワードを使用して、必要な動作環境を指定します。コンポーネントは、指定されたプラットフォーム ID (PlatformID) パラメータに一致する動作環境のターゲット コンピュータにのみインストールされます。インストール時にプラットフォーム固有のコンポーネントが見つからない場合は、PlatformID = Any のデフォルトのコンポーネントがターゲット コンピュータにインストールされます。

プラットフォーム固有のコンポーネントの使用は、共有ライブラリのインストールなどの場合に適しています。共有ライブラリ名は、各種の UNIX 動作環境において異なります。共有ライブラリの名前は、Sun Solaris および Linux システムでは `.so` で終了し、HP-UX では `.sl` で終了し、IBM-AIX では `.o` または `.a` で終了します。たとえば、ネットワーク内の HP-UX および Solaris システム用に、これらの異なる命名規則を対象とするには、2 つのプラットフォーム固有のコンポーネントを PIF 製品に追加します。1 つ目のコンポーネントには拡張子が `.sl` の HP-UX 用ライブラリが格納され、2 つ目のコンポーネントには拡張子 `.so` の Solaris 用ライブラリが格納されます。

パッケージャにより、すべての UNIX/Linux 動作環境にインストール可能な製品 (PlatformID = Any) と特定の動作環境にインストールする製品が識別されます。すべての動作環境にインストール可能な PIF 製品には、プラットフォーム固有コンポーネントがすべて追加されます。指定された動作環境にのみインストール可能な PIF 製品の場合は、PlatformID パラメータに一致しないすべてのコンポーネントが削除されます。

言語固有のコンポーネント

言語固有のコンポーネントを使用すると、マルチリンガル対応のインストールが可能になります。言語固有のコンポーネントに含まれるファイルは、コンポーネントに定義された言語が、ターゲット コンピュータ上の `$LANG` パラメータで指定された言語に一致する場合にのみインストールされます。言語固有のコンポーネントには、通常、言語固有のテキストを使用したリソース ファイル、ヘルプ ファイル、マニュアル ページ、およびグラフィック ファイルが含まれています。言語固有のコンポーネントを指定するには、プロトタイプ ファイルのコンポーネント セクションで `#locale` キーワードを使用します。

言語固有のコンポーネントは、メインの PIF 製品内、または外部の言語固有の PIF ファイルに格納できます。外部の言語固有の PIF ファイルは、メインの PIF 製品と同じフォルダに格納する必要があります。

言語固有のコンポーネントを変換するには、**pifextract** コマンドを使用してこれらのコンポーネントを抽出します。変換された言語固有のコンポーネントは、新規の言語固有のコンポーネントとして PIF 製品に追加することができます。

言語固有のコンポーネントは、プラットフォーム固有ではありません。

デフォルトの製品言語を使用したコンポーネントは、随時インストールされます。また、ターゲット コンピュータの言語 (**\$LANG**) に一致する言語固有のコンポーネントがインストールされます。

プリインストール コンポーネント

プリインストール コンポーネントには、PIF 製品を実際にインストールする前 (インストール ダイアログ ボックスおよびスクリプトが実行される前) に使用されるすべてのファイルが含まれます。プリインストール コンポーネントは、**PIF_PREINSTALL_DIR** パラメータで指定された一時フォルダに抽出されます。

プリインストール コンポーネントには、通常、以下の項目が含まれます。

- Java ダイアログ ボックスで使用されるグラフィック ファイル
- インストール ダイアログ ボックスで使用されるスクリプト、検証スクリプト、および動的ダイアログ ボックス検出スクリプト
- PIF 以外の製品のマイグレーション スクリプト
- 製品およびコンポーネントの事前スクリプト
- インストール スクリプトで使用されるテキスト リソース ファイル

同じ名前のプリインストール コンポーネントを複数使用することができますが、異なる言語のものは使用できません。

プリインストール コンポーネントのロケールがシステムのロケールに一致しない場合、言語固有の PIF 製品が (ある場合は) 代わりに使用されます。

オプション コンポーネント

すべての種類のコンポーネントについて、必要に応じて、どのような場合でも製品と共にインストールするのではなく、必要とされる場合にのみ (インストール ダイアログ ボックスでコンポーネントを選択した場合、または応答ファイルのパラメータ値に応じて) インストールするように指定できます。要求に応じてインストールされるコンポーネントは、オプション コンポーネントと呼ばれます。たとえば、ドキュメント群をオプション コンポーネントとして定義できます。

PIF パラメータを使用すると、オプション コンポーネントを選択したり選択解除したりすることができます。PIF パラメータが **1** に設定されているとコンポーネントがインストールされますが、**0** に設定されているとコンポーネントはインストールされません。このパラメータを PIF パッケージに割り当てるには、GUI の「コンポーネント/詳細設定/インストール動作」を使用するか、または「`#install:`」というキーワードを使用して手動で割り当てます。チェック ボックスを追加して、一致した値にパラメータを提供することができます。このチェック ボックスをダイアログ ボックスに追加する必要があります。

PIF 製品のファイル、ディレクトリ、およびシンボリック リンク

PIF 製品には、ファイル、ディレクトリ、およびシンボリック リンクが含まれます。通常、これらの項目はすべて、製品のインストールパスの相対パスにインストールされます。これらをターゲット コンピュータの特定の位置にインストールするには、宛先パスを完全修飾名で指定します。

パッケージャおよびインストーラでは、以下のファイル タイプがサポートされます。

標準ファイル

標準ファイルは、ターゲット コンピュータへのインストールに必須の変更不可能な製品ファイルです。インストーラによって、標準ファイルの整合性がチェックされます。再インストールを行う場合は、再インストール時に標準ファイルが置き換えられます。

設定ファイル

設定ファイルには、すべての設定が保存されます。更新インストールの際に、設定ファイルが更新されることはありません。設定ファイルはサイズが変わることが多く、製品の整合性が反映されません。このため、PIF ファイルの整合性チェックでは、変更された設定ファイルは無視されます。設定ファイルに新しいパラメータを追加する場合は、プリインストール スクリプトまたはインストール後のスクリプト内で行う必要があります。PIF 製品が完全に削除されると、設定ファイルも削除されます（ただし、`lsm` オプション `-R` または内部パラメータ `PIF_RETAIN_CONFIGURATION` によって削除が抑止された場合を除く）。再インストールを行う場合、設定ファイルはインストーラによって変更されません。

一時ファイル

一時ファイルは、製品のインストール時にターゲット コンピュータにコピーされ、製品のインストールが終了するとターゲット コンピュータから削除されます。再インストールを行う場合、一時ファイルはインストーラによってチェックされません。

永続ファイル

ソフトウェア製品が削除されても、永続ファイルはシステムから削除されません。製品の削除後もシステムに残ります。永続ファイルが製品のインストール前からあり、既存の永続ファイルの変更日付が製品に梱包されたファイルの日付より新しい場合、元の永続ファイルがそのままシステムに残ります。

ディレクトリ (**#dir**) およびシンボリック リンク (**#slink**) は、ユーザおよびグループに割り当てられます。これらのユーザおよびグループは、製品をインストールする前に作成されている必要があります。

パスのすべてのディレクトリは、PIF 製品内にそれぞれエントリがある必要があります。たとえば、`/usr/local/bin/` の場合、製品に `/usr`、`/usr/local`、および `/usr/local/bin` の 3 つのエントリが必要となります。

PIF 製品内のファイルのディスク領域の予約

製品インストールの間にインストールされないファイル（仮想ファイル）用に、製品を作成するときにプロトタイプ ファイルの **#reservedspace** キーワードを使用してディスク領域を予約することができます。その場合、仮想ファイルの必須のディスク領域は、ディスク領域チェック プログラムによって考慮されます。

製品のインストールおよび削除のスコープ内のスクリプト

ターゲット コンピュータ上での PIF 製品のインストール/削除の前または後で実行するインストーラ スクリプトや実行可能ファイルを作成することができます。

デフォルトでは、**Bourne** 互換形式のシェル スクリプトが実行されるように想定されています。内部プロシージャ、およびほかのインタープリタによって実行される非標準のシェル スクリプトを実行することができ、パラメータを使用してスクリプトのカスタマイズおよび制御が可能です。

内部プロシージャの使用

内部プロシージャは、製品のインストール プロセスの設定および情報の交換に使用されるスクリプト ファイルです。内部プロシージャは製品の一部であり、各スクリプト ファイルがコンポーネントのファイル リストに定義されている必要があります。これらのスクリプト ファイルには、実行権限が指定されている必要があります。

（[新しいファイルを追加] の機能を選択したときに開く）内部エディタを使用して、新しいスクリプト ファイルをパッケージ GUI から作成できます。新しいスクリプト ファイルを保存すると、その名前が自動的にコンポーネントのファイル リストに追加されます。

既存のスクリプト ファイルは、PIF 製品のコンポーネントの [ファイルの挿入] 機能を使用して、コンポーネントのファイル リストに追加されます。

内部プロシージャを特定のコンポーネントに追加する場合、その特定のコンポーネント内で（GUI を使用するか、またはプロトタイプ ファイルの適切なコンポーネント セクションのエントリとして）プロシージャを定義する必要があります。一般的に、プロシージャを製品に追加するには、（GUI を使用するか、またはプロトタイプ ファイルの製品セクションのエントリとして）内部プロシージャをその製品用に定義できます。

内部プロシージャ用に指定可能なプロパティは、以下のとおりです。

実行時間

スクリプトの実行時間です。可能な値は、以下のとおりです。

- インストール前 - パッケージのインストール前にスクリプトを実行します。

- インストール後 - パッケージのインストール後にスクリプトを実行します。 `package installation`.
- 削除前 - パッケージの削除前にスクリプトを実行します。ゼロ以外の終了値を持つスクリプトが存在する場合、コンポーネントはシステムから削除されず、アンインストールプロセスが続行します。
- 削除後 - パッケージの削除後にスクリプトを実行します。
- 開始前 - 製品のインストールおよびインストール ダイアログ ボックスの開始前にスクリプトを実行します。
- PIF インストール前 - 製品をはじめてターゲット コンピュータにインストールする際にのみ実行する特殊なスクリプトです。このスクリプトのみ、製品のインストールパスの変更が可能です。
- インタビュー後 - (インストール ダイアログ ボックスの) インタビューフェーズが完了した後にスクリプトが実行されます。
- `post check` - インストーラの確認が実行された後に、スクリプトが実行されます。

[オプション]

コマンド スイッチを使用したスクリプト実行の設定が可能なスクリプトの場合は、製品に内部プロシージャを追加するときに、これらのスイッチを指定することができます。これらのコマンド スイッチは、スクリプトの実行時にスクリプトに適用されます。

インタープリタ

オプションとして、デフォルトのインタープリタ (シェル) を使用しない場合のインタープリタを指定できます。

シェル以外のスクリプトの使用

IPS/DMS スクリプトや Perl スクリプトなど、標準的なシェル以外のインタープリタで実行されるスクリプトを実行できます。これらの場合は、必要なインタープリタの名前をインストーラのプロパティとして指定する必要があります。

パラメータを指定したスクリプトの使用

スクリプトでパラメータを使用すると、製品のインストール中にスクリプトの値を設定したり、スクリプトの実行を制御することなどができます。スクリプトに渡すパラメータ値によって、スクリプトのどの部分を実行するかが判別されます。パラメータの値は、シェルパラメータ、スクリプトの引数、または応答ファイルとして、スクリプトに渡すことができます。

シェル スクリプトのパラメータを更新するには、応答ファイルを使用してパラメータ値をスクリプトに渡します。応答ファイルには、変更されたパラメータがすべて含まれている必要があります。インストーラによって応答ファイルが評価され、残りのインストール処理では変更されたパラメータが使用されます。対応する応答ファイルの名前は、内部パラメータ `$PIF_RESPONSE_FILE` を使用して、スクリプトで参照されます。更新したパラメータをこの応答ファイルに追加する必要があります。

シェル パラメータに行われた変更内容は無視されます。

重要： 製品が更新されているときは、インストール パス パラメータの値を変更できません。プリインストール スクリプトでは、コンポーネントのインストールフラグのみが変更できます。

スクリプト タイマの変更

デフォルトの PIF タイマである 10 分を超える実行時間の長いスクリプトの場合は、インストール スクリプト（プロトタイプ ファイルのキーワード `#timer` を使用）の要件を満たすよう製品のタイマを定義できます。

たとえば、PIF 製品が、`uncompress` を使用してインストールされ、圧縮されている PIF 以外の製品と依存関係にあり、PIF 以外の製品のインストール時間がデフォルトの制限時間である 10 分を超えるとします。この場合、主要な PIF 製品をインストールする前に、PIF 以外の製品がすべてインストールされていることを確認する必要があります。

重要： タイマの値を大きく設定すると、自動インストール時にタイムアウトを起こすメカニズムを回避できることがあります。

スクリプトの進捗状況

〔進捗状況〕ダイアログボックスでは、スクリプトの出力を使用して、〔進捗状況〕エリアにスクリプト実行の進捗状況が表示されます。この表示機能は、製品のインストール中に実行時間が長いスクリプトの進捗状況を表示するのに役立ちます。スクリプトの出力を表示するには、ダイアログボックス要素の〔進捗状況〕エリアを使用します。この表示機能は、プロトタイプファイルでは **#progressarea** キーワードを使用してサポートされています。

〔進捗状況〕ダイアログボックスでは、現在実行されているスクリプトの名前が〔アクション〕フィールドに表示されます。製品の進捗状況は、スクリプトが実行されると保留状態になります。

この機能は、RPM パッケージでは使用できません。

PIF 製品の依存関係

PIF 製品のインストールは、ほかの PIF 製品の有無やターゲット コンピュータ上の製品のバージョンに依存することがあります。このような依存関係は、PIF、PKG、または RPM パッケージフォーマットの別の必要な製品とインストールする PIF 製品との間に存在する可能性があります。

製品の依存関係の種類は、以下のとおりです。

必須

製品の製品 1 をインストールする前に製品 2 をターゲット コンピュータにインストールする必要がある場合、製品 1 は製品 2 と必須依存関係にあります。製品 2 がインストールされていないと、インストーラの特定アルゴリズムによって、自動的に製品 2 が検索され、検出後にインストールされます。製品 2 が見つからない場合、インストールは中止されます。製品 2 は必須製品です。

互換性なし

製品の製品 1 をインストールする前に製品 2 をターゲット コンピュータにインストールしてはいけない場合、製品 1 は製品 2 と互換性のない依存関係にあります。製品 2 がインストールされている場合は、製品 1 のインストールが中止されます。

オプション

製品 1 をインストールする前に必ずしも製品 2 をインストールする必要がない場合、製品 1 は製品 2 とオプション依存関係にあります。製品 2 は、ユーザが要求する場合にのみターゲット コンピュータにインストールされます。ターゲット コンピュータに製品 2 の前のバージョンがすでにインストールされている場合は、新しいバージョンに更新されます。

製品の依存関係は、キーワード `#pdep` (必須およびオプション)、キーワード `#dis` (互換性なし)、またはパッケージ `GUI` を使用して、プロトタイプ ファイルで定義されます。

製品のインストール時に、インストーラでは、製品で定義された別の PIF 製品およびそのコンポーネントに対して、必要に応じて複数の階層にわたってすべての依存関係が自動的に解決されます。実際の製品をインストールする前に、インストーラによって、すべての必要な製品がターゲット コンピュータにインストールされていることが確認されます。このため、必要な製品がすべて配布メディアに含まれていることを確認するだけですみます。

製品の削除中に、ほかの製品が必須 PIF 製品を必要としない場合、インストーラでは、必要に応じて複数の階層にわたり、必要な製品が自動的に削除されます。インストーラでは、製品と共に配布されインストールされた必須製品のうち削除する製品のみがチェックされます。

互換性のない製品は、自動的に削除されません。

製品のパラメータ

製品のパラメータを使用して、PIF 製品のインストールを設定します。このパラメータによって、スクリプトとインストールプロセスの間で情報が交換されます。パラメータは、インストーラにより PIF 製品のインストール時に解決されます。パラメータを使用すると、インストール パスを定義し、インストールプロセスからコンポーネントを追加または削除することができます。また、製品固有のパラメータをインストールに渡すこともできます。パラメータは、パラメータ名によって一意に指定されます。パラメータ名の先頭には、常にドル記号 (\$) が付きます。

パラメータは、パラメータの名前、種類、および値で定義されます。パラメータ値には、絶対 (静的な) 値、コマンドの実行結果、ほかのパラメータの結果、絶対値とほかのパラメータの組み合わせなどがあります。

パラメータの種類

パラメータには、製品インストールの PIF 製品の影響度および可視性によって、以下の種類があります。

専用

パラメータがその製品だけで有効であることを指定します。プライベート パラメータは、製品固有のインストール スクリプトでのみ変更できます。

発行済み

パラメータが、製品およびその製品に必要なすべての製品で有効であることを指定します。必要な製品では、アクセスは「読み取り専用」に制限されます。発行済みパラメータが上位レベルのパッケージで設定されていない場合は、このパラメータをメインの製品および後に続く PIF 製品で設定できます。

グローバル

パラメータがすべての製品のインストールで有効であることを指定します。設定されたグローバル パラメータは、そのパラメータを使用するすべての製品が削除されない限り、変更することができません。グローバル パラメータは、ほかのユーザが参照していない場合にのみ、削除できます。

重要: パラメータを使用して PIF 製品自体の設定を行う場合は、注意が必要です。この場合は、プライベート パラメータだけを使用する必要があります。これは、このパラメータがほかの製品のインストールによって変更されないからです。

パラメータ値の設定方法

PIF 製品のパラメータに値を設定する方法は、以下のとおりです。

デフォルトのパラメータ値

パラメータのデフォルト値は、プロトタイプ ファイルの **Default** セクションで定義されます。デフォルト値によって、確実に特定のパラメータに対する値が常に存在することになります。

環境から

インストーラでは、製品のインストール中に実行中のシェルの環境が調査されます。シェルにパラメータ値が割り当てられている場合は、製品で定義されたデフォルトのパラメータ値が上書きされます。

応答ファイルから

応答ファイルは、製品の自動インストールを設定するためにユーザが作成するファイルです。このファイルには、製品に定義されたデフォルトのパラメータ値を上書きするパラメータ値の割り当てが含まれています。

スクリプトから

プリインストール スクリプト（スクリプト パラメータ）に、パラメータ値の割り当てを定義できます。製品のインストールプロセス中にこのスクリプトが実行されると、製品にパラメータ値が挿入されます。

パラメータ値の種類

PIF 製品で使用するパラメータ値の種類は、以下のとおりです。

静的

静的値が入力され、直接パラメータに割り当てられます。

組み合わせ

パラメータに割り当てられた値は、静的な値と前に定義されたパラメータを組み合わせた値です。例は以下のとおりです。

```
#parameter: $installdir , /opt/CA/installpath ;  
#parameter: $my_install_path , $installdir/myproduct ;
```

パラメータの組み合わせの静的パラメータの終わりは、以下の文字のいずれかで判断します。

. ; : ¥t ¥n ¥¥ ¥" / , \$ () - { } 空白

動的

パラメータに設定された値は、コマンドを実行した結果の値です。このため、静的な値の代わりにたとえば以下のようなコマンドが指定されます。

```
#parameter: $my_date , `date` ;
```

パラメータのデフォルト値

PIF 製品で使用される各パラメータには、デフォルト値が割り当てられている必要があります。割り当てられていない場合、製品の作成は失敗します。デフォルト値を使用する目的は、以下のとおりです。

- インストール ダイアログ ボックスのタスクを実行するときに、初期値として使用します。
- 応答ファイルがない場合は、PIF 製品の自動インストールで使用します。

プロトタイプ ファイルでキーワード **#parameter** を使用して、**@DEFAULT** セクションにあるパラメータのデフォルト値を定義します。

重要： 製品の標準テンプレートの 1 つを使用して PIF 製品を作成するときには、すでにパラメータにデフォルト値が自動的に割り当てられています。

追加パラメータの説明

各製品パラメータに関する簡単な説明を入力するには、パラメータのプロパティとしてコメントを追加できます。

説明を追加するには、以下の手順に従います。

- [パラメータの追加/変更] ダイアログ ボックスの [コメント] フィールドに追加します（この説明は、**lsm -a** コマンドで作成する応答ファイルに追加されます）。
- プロトタイプ ファイルにあるキーワード **#parameter** の **paramcomment** プロパティとして追加します。

パラメータに関する重要事項

- オプションが必要な製品が選択された場合にのみ、その製品に定義されたグローバルパラメータがパラメータの共通セットに追加されます。
- 環境全体で設定されるグローバルパラメータの場合は、以下が適用されます。
インストーラでは、以前にインストーラによって不明とされたグローバルパラメータが、今回の環境ではすでに定義されているかどうかを確認されます。すでに定義された値がある場合、インストーラはその値をパラメータの設定値と見なし、このパラメータ値を変更するダイアログボックスを自動的にオフにします。
- 別のパラメータまたはパラメータの組み合わせ、および絶対値で構成されるパラメータ値は、ユーザがダイアログボックス間を移動するときに自動的に評価されます。
- 以下の文字は、パラメータ名には使用できません。
. ; : ¥t ¥n ¥¥ ¥" / , \$ () - { } 空白

例：パラメータを使用したアクセス権の制御

プロトタイプファイルのファイル（#file）またはディレクトリ（#dir）を定義する場合は、パラメータを使用して、ユーザ、グループおよびファイルアクセス権を制御したり、読み取り、書き込み、または実行の適切な権限を割り当てることができます。これは、異なる動作環境に異なるユーザの PIF 製品をインストールする場合に有効です。

以下に、プロトタイプ ファイルの **#dir** エントリにパラメータを使用した場合の例を示します。

```
@DEFAULT:
#sys: Any ;
#parameter: $OWNER , root ;
#parameter: $GROUP , sys ;
#parameter: $ACCESS , drwxrwxrwx ;
@ENDDFAULT:

@PRODUCT:
#phead: exampleOwnerParam , 1000 ;
#ppath: /tmp/exampleOwnerParam ;
#sys: Any ;
@ENDPROD:

@COMPONENT:
#chead: compl ;
#dir: mydir , $OWNER , $GROUP , $ACCESS ;
@ENDCOMP:
```

PIF 製品用の応答ファイル

PIF 製品用の応答ファイルには、パラメータに値を割り当てるレコードが含まれています。ターゲット コンピュータでは、インストーラ コマンド **lsm** で **-a** オプションを指定して、応答ファイルを対話式で作成できます。この **lsm** コマンドで入力された値を使用すると、インストール ダイアログ ボックスが実行され、応答ファイルが作成されますが、PIF 製品はインストールされません。

このパラメータの割り当て値の構文形式は、以下のとおりです。

parameter=value;

式の右辺の値は、**\$parameter**（静的割り当て）に直接割り当てられます。

例：

```
INSTALLDIR=/opt/CA/installpath;
```

parameter=`command`

コマンドが実行され、コマンドの出力が **\$parameter**（動的割り当て）に割り当てられます。コマンドはバッククォート（**`**）で囲む必要があります（一重引用符ではありません）。

例：

```
SYSTEMNAME=`uname -n`
```

応答ファイルのコメント行の先頭はハッシュ記号（#）です。

パラメータにプロトタイプ ファイル（#parameter キーワード）の Default セクションに割り当てられた説明がある場合、この説明は自動的に応答ファイルに追加されます。

内部パラメータ

PIF 製品のインストール中に内部パラメータが提供されます。内部パラメータは、インストール ダイアログ ボックスでの情報の表示にも使用できます。プレフィックス PIF_ は内部での使用で予約されているため、ユーザが定義するパラメータには使用できません。

以下の内部パラメータが使用されます。

PIF_BACKUP

現行バージョンの製品を更新または再インストールする前に、バックアップの作成を制御します。

PIF_CHANGECONTROL_dialogelement

ダイアログ ボックス要素を使用しない場合にグレー表示するかどうかを制御します。

PIF_CHANGEPARAM_parameter

ダイアログ ボックス要素を使用して指定されたパラメータを編集できるかどうかを制御します。PIF_CHANGEPARAM_parameter の値が 1（True）の場合、指定されたパラメータを変更するダイアログ ボックス要素が有効になります。この値が 0（False）の場合、指定されたパラメータにアクセスできません。

PIF_CONTROL_NAME

スクリプトの呼び出し前にアクティブだった（フォーカスが設定されていた）フィールドの名前を指定します。この情報に基づき、スクリプトは呼び出されたところからフィールドごとに異なる動作を実行できます。この内部パラメータは、PIF ダイアログ ボックス内の任意の検証スクリプトに渡されます。このスクリプトには、フィールド検証スクリプト、アクションスクリプト、ダイアログ ボックス検証スクリプト、および動的ダイアログ ボックス検出スクリプトが含まれます。

PIF_DIALOG_LANGUAGE

製品のインストールに使用する言語を指定します。このパラメータの値は、2 バイトまたは 3 バイトの言語識別子で、ENU などがあります。

PIF_DIALOG_MODE

インストール ダイアログ ボックスの種類を指定します。設定可能な値は以下のとおりです。

JAVA、VT100、および NONE。

PIF_DIALOG_NAME

選択されたボタンに対応するダイアログ ボックスの名前を指定します。

PIF_INSTALL_JOB

インストール ジョブを指定します。設定可能な値は以下のとおりです。

INSTALL、REINSTALL、UPDATE、および REMOVE。

PIF_INSTALL_MODE

インストール モードを指定します。設定可能な値は以下のとおりです。

ATTENDED、UNATTENDED、および ASK。

PIF_INSTALL_PATH

PIF パッケージのインストール パスを指定します。

PIF_LOCALE

PIF パッケージをインストールするロケールを示します。

PIF_LOG_FILE

ログ ファイル名を指定します。

PIF_MAINPRODUCT_DIR

メイン パッケージのインストール パスを指定します。

PIF_PREINSTALL_DIR

プリインストール コンポーネントの抽出先となる一時フォルダ名を指定します。

PIF_PRODUCT_NAME

PIF パッケージの名前を指定します。

PIF_PRODUCT_PLATFORM

PIF パッケージの動作環境を指定します。

PIF_PRODUCT_VERSION

PIF パッケージのバージョンを指定します。

PIF_RESPONSE_FILE

パラメータ値の指定に使用する応答ファイルの名前を指定します。

PIF_RETAIN_CONFIGURATION

設定ファイルの削除を制御します。

PIF_SUPPLIER

PIF パッケージのベンダー名を指定します。

PIF_INSTANCE_NAME

新しいインスタンスの名前を指定します。このパラメータはインストール時に使用されます。

PIF_CREATION_DATE

PIF 作成の日付を指定します。形式は DD/MM/YY です。

PIF_CREATION_TIME

PIF 作成の時刻を指定します。形式は、hh:mm:ss です。

パラメータがインストーラによって解決される方法

PIF 製品で使用するパラメータは、ターゲット コンピュータでの PIF 製品のインストール中にインストーラによって解決されます。インストーラを実行する手順は、インストールの種類によって異なります。手順は以下のとおりです。

新規インストール

1. PIF 製品で定義されているデフォルトのパラメータ値を読み取ります。
2. 応答ファイルからパラメータ値を読み取ります。
3. インストール ダイアログ ボックスからパラメータ値を読み取ります（インストール パスは変更できます）。
4. 既存環境からパラメータ値を読み取ります（自動モード）。
5. 製品固有のプリインストール スクリプトを実行した後に変更されたパラメータ値を読み取ります。

再インストール

以前に行ったインストールからパラメータ値を読み取ります（インストール パスは更新できません）。

更新インストール

1. PIF 製品で定義されているデフォルトのパラメータ値を読み取ります。
2. 応答ファイルからパラメータ値を読み取ります。
3. 以前に行ったインストールからパラメータ値を読み取ります。
4. インストール ダイアログ ボックスからパラメータ値を読み取ります（インストールパスは更新できません）。
5. 製品固有のプリインストール スクリプトを実行した後に変更されたパラメータ値を読み取ります。

PIF 製品のテンプレート

製品のテンプレートを使用すると、PIF 製品のインストールが統一され、PIF 製品を詳細に定義することができます。製品のテンプレートは、標準プロトタイプファイルです。このテンプレートには、あらかじめ定義されたパラメータのセットが含まれています。これらのパラメータは、PIF 製品の生成中に置き換えられます。テンプレートは `pifproto` コマンドで管理できます。

`$SDPcker/templates` フォルダのパッケージ インストールによって提供される PIF 製品の標準テンプレートは、以下のとおりです。

`standard.Any.@enu`

事前定義された基本インストールのダイアログ ボックス セットを使用して、パッケージを作成します。`standard.Any.@enu` は、テンプレートが指定されていない場合にデフォルトのテンプレートとして使用されます。

`advanced.Any.@enu`

事前定義されたインストールのダイアログ ボックス拡張セットを使用して、パッケージを作成します。

`unattended.Any.@enu`

インストール ダイアログ ボックスを使用せずに自動インストールでパッケージを作成します。

`multi_language.Any.@enu`

インストールに使用する言語を複数の言語から選択することができる、特定のダイアログ ボックスを使用してパッケージを作成します。

`multi_instance.Any.@enu`

複数のインスタンス サポートを使用して、パッケージの複数のインスタンスのインストールが可能なパッケージを作成します。「マルチ インストール」 (`#multiinst:1`) としてパッケージをマークする必要があります。

`$SDPcker/templates` ディレクトリでは、標準テンプレートのコピーや変更を行って、企業に固有の PIF 製品のインストールを定義できます。パッケージャでは、このディレクトリにある新しいファイルを個々に認識します。

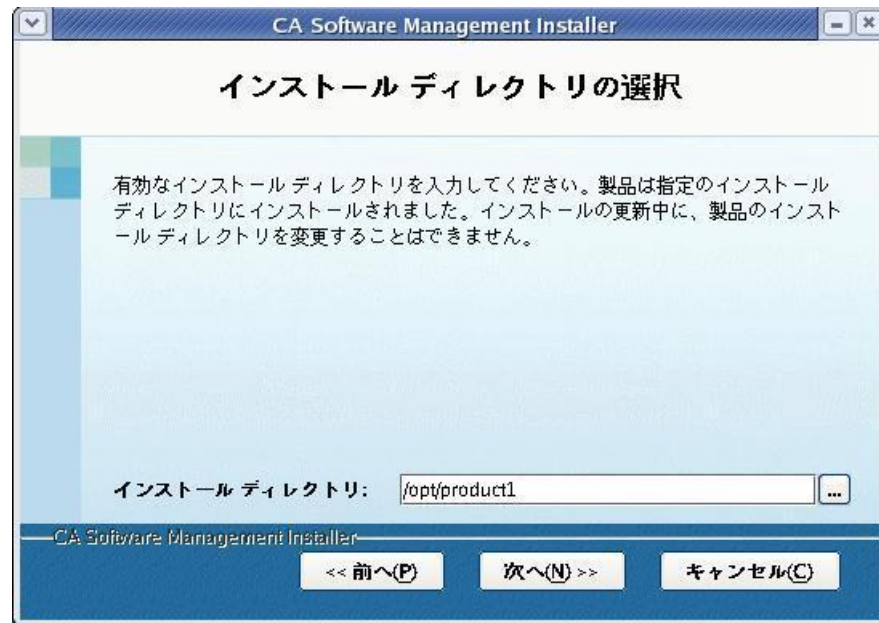
インストール ダイアログ ボックス

インストール ダイアログ ボックスを使用すると、ターゲット コンピュータでわかりやすく、拡張性のある設定を行うことができます。このダイアログ ボックスでは、パラメータを取得し、ターゲット コンピュータでの PIF 製品のインストールを設定します。インストーラでは、あらゆる種類のターゲット コンピュータに対応するインストール ダイアログ ボックスがサポートされています。

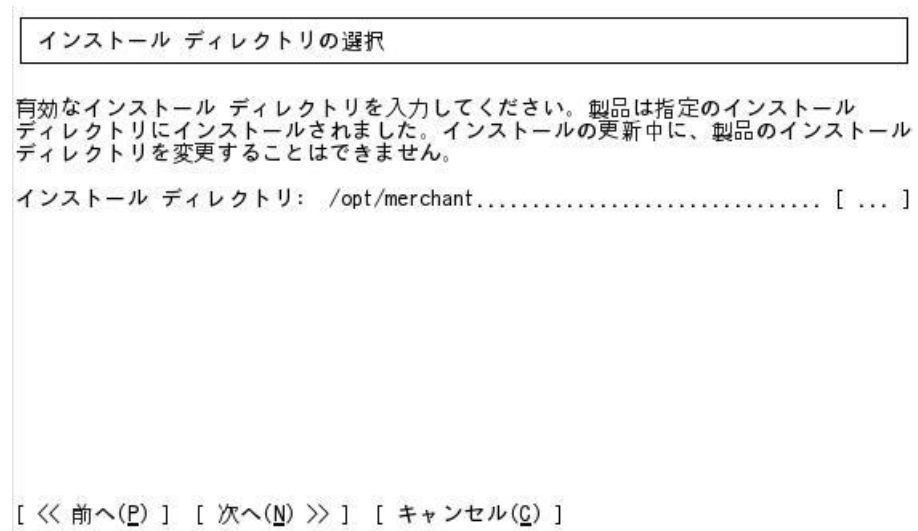
ターゲット コンピュータに **Java Runtime Environment (JRE) 1.5** または **1.6** がインストールされていて、モニタがグラフィック表示対応であると、インストール ダイアログ ボックスがグラフィカル モード (**Java GUI**) で表示されます。ターゲット コンピュータでこれらの前提条件が満たされない場合、インストール ダイアログ ボックスは **VT100** 互換モードで表示されます。

JRE を自己インストール PIF 製品に含めることができます。すると、含まれた JRE がインストールおよび使用され、ターゲット コンピュータでインストーラの **Java** ウィザードが実行されます。

以下の図に、グラフィック モードでのインストール ダイアログ ボックスの例を示します。



VT100 互換モードでの同じインストール ダイアログ ボックスの例は、以下のとおりです。



インストール ダイアログ ボックスの定義

プロトタイプ ファイルでは、適当なキーワード使用して、**Dialog** および **Resource** セクションにダイアログ ボックスのプロパティ、ボタン、およびテキストを定義できます。

各インストール ダイアログ ボックスは、一意の名前で識別されます。このため、ダイアログ ボックス間を移動ができ、ダイアログ ボックスの順番を定義できます。スクリプトを使用すると、次のダイアログ ボックスを開始する前にダイアログ ボックスに入力されたすべての値や単独の値を検証したり、次に開始するダイアログ ボックス名を決めることができます。

インストール ダイアログ ボックスには、テキスト入力フィールド、テキスト出力エリア、インストールディレクトリ入力フィールド、チェック ボックス、ナビゲーション ボタンなど、さまざまなダイアログ ボックス要素を含めることができます。インストールを開始したときにフォーカスされるダイアログ ボックス要素（「次へ」 ボタンなど）を設定することができます。

インストール ダイアログ ボックスに表示されるすべてのテキスト メッセージは、プロトタイプ ファイルの別々のセクションに格納されます。このため、テキスト メッセージをローカライズして同じ PIF 製品を別の言語で使用できます。

注： インストーラ（**IsM コマンド**）の **option -V** を使用して、強制的に **VT100** 互換モードにすることができます。

標準インストール ダイアログ ボックス

unattended.Any.@enu 以外の標準テンプレートを使用して、PIF 製品を作成する場合は、パッケージにより PIF 製品に標準インストール ダイアログ ボックスセットが自動的に追加されます。

パッケージ **GUI** では、サブプロジェクトのウィンドウにある「ダイアログ」タブにこれらのダイアログ ボックスの一覧が表示されます。プロトタイプ ファイルでは、それぞれのインストール ダイアログ ボックスは各「ダイアログ」セクションに記述されています。

標準インストール ダイアログ ボックスには、事前に設定されたダイアログ ボックスの要素が数多く含まれています。これらの要素は、パッケージのインストール ダイアログ ボックスのエディタを使用して、変更、拡張、およびコピーすることができます。

タスク指向のインストール ダイアログ ボックス

インストーラでは、新規インストール用のダイアログ ボックス（インストール ダイアログ ボックス）が、PIF 製品の更新インストール、再インストール、または削除用のダイアログ ボックス（更新ダイアログ ボックス）と区別されます。

製品の更新、再インストール、および削除用の各ダイアログ ボックスを開始するには、初期更新ダイアログボックスにユーザが使用できる別のインストール モードでメニューを表示する必要があります。

プロトタイプ ファイルでは、初めて PIF 製品をインストールするときに最初に呼び出されるインストール ダイアログ ボックスが、キーワード `#dlgpreinit` で指定されます。PIF 製品を更新、再インストール、または削除するときに最初に呼び出されるインストール ダイアログ ボックスは、キーワード `#dlgpreupd` で指定されます。

パッケージ GUI では、[ダイアログ] プロパティ ダイアログ ボックスを使用して、作成されたダイアログ ボックスが新規インストール用か更新インストール用かを判断します。

キャンセルアクションが関連付けられている各ナビゲーション ボタンに対して、1つ以上の [キャンセル] ダイアログ ボックスを定義する必要があります。

動的に検出されたダイアログ ボックス

次に表示するインストール ダイアログ ボックスを決定するには、シェル スクリプトを指定します。このスクリプトでは、固定のダイアログ ボックス名を指定する代わりに、次のインストール ダイアログ ボックスの名前を指定できます。固定のダイアログ ボックス名の代わりにシェル スクリプトを使用すると、順番に、指定されたダイアログ ボックスを動的に表示できます。

シェル スクリプトでは、標準出力に次のインストール ダイアログ ボックスの名前を印刷する必要があります。シェル スクリプトの名前はダイアログ ボックスのプロパティとして入力されます。このスクリプトは、ナビゲーション ボタン コントロールにリンクしています。

ダイアログ ボックスの検証スクリプト

インストール ダイアログ ボックスに入力された値の組み合わせを検証するには、ダイアログ ボックスの検証スクリプトを使用します。このスクリプトを使用するには、プリインストール コンポーネントに検証用のスクリプト ファイルを追加する必要があります。ダイアログ ボックス内で、ダイアログ ボックス全体または特定の入力フィールドのユーザ データを検証できます。

ダイアログ ボックスに入力されたすべてのユーザ データの検証

ダイアログ ボックスの検証スクリプトは、インストール ダイアログ ボックスの [次へ] ボタンをクリックすると実行されます。その後、インストール ダイアログ ボックスに入力されたすべてのパラメータ値がシェル環境に渡されます。検証が成功した場合、スクリプトによってゼロ (0) が返されます。エラーの場合は、スクリプトによって、プロトタイプ ファイルの **Resource** セクションに、一致するエラー テキストの番号が返されます。参照されているエラー テキスト番号は、1 ～ 255 の範囲内である必要があります。対応するエラー テキストがない場合、エラー メッセージにテキスト番号が表示されます。

入力された値 `$LOCALE` および `$DATE` の検証例は、以下のとおりです。英語のロケールが入力されている場合は、データ フィールドに日付が入らない（またはドイツ語での日付表示になる）ことがあります。

```
#!/bin/sh

# If one of the parameters $LOCALE or $DATE is not set,
# error message 1 is printed
[ ! "$LOCALE" -o ! "$DATE" ] && exit 1

# If an English locale has been entered, the date field
# must not contain any dot
[ "$LOCALE" = "en_US" -a `echo "$DATE" | grep "." 2>/dev/null 1>&2; echo $?` -eq 0 ] && exit 2

# exit without error message
exit 0
```

特定の入力フィールドに入力されたユーザ データの検証

テキスト フィールド (**#textfield**)、インストール ディレクトリ (**#instdir**)、およびパスワード フィールド (**#passwordfield**) の検証スクリプトを指定できます。これらの検証スクリプトは、ユーザが入力フィールドから移動したときに実行されます。検証スクリプトが実行される前に [次へ] ボタンをクリックすると、再度これらのスクリプトが呼び出されます。入力したパラメータ値は、スクリプトにシェル パラメータとして渡されます。検証が成功した場合、スクリプトによってゼロ (0) が返されます。エラーの場合は、スクリプトによって、プロトタイプ ファイルの **Resource** セクションに、一致するエラー テキストの番号が返されます。参照されているエラー テキスト番号は、1 ~ 255 の範囲内である必要があります。対応するエラー テキストがない場合、エラー メッセージにテキスト番号が表示されます。

以下に、ディレクトリ名が完全に正しいかどうかを確認 (パラメータ **\$INSTDIR**) する例を示します。

```
#!/bin/sh

# If the parameters $INSTDIR is not set, error message 1 is printed
[ ! "$INSTDIR" ] && exit 1

# If the parameters $INSTDIR does not start with a slash,
# error message 2 is printed
[ `expr "$INSTDIR" : '¥(¥)'\` != "/" ] && exit 2

# exit without error message
exit 0
```

検証スクリプトからのログ情報の提供

ログ情報は、以下のようにダイアログ ボックスの検証スクリプトにあるパラメータ **PIF_LOG_FILE** を使用して取得できます。

```
if [ $VERBOSE -eq 1 -a -w "$PIF_LOG_FILE" ]
then
    echo "LOG INFORMATION TEXT" >> "$PIF_LOG_FILE"
fi
```

アクション スクリプト

チェック ボックス、パスワード フィールド、インストール ディレクトリ、ファイル 選択 フィールド、コンボ ボックス、および ラジオ ボタン で使用する アクション スクリプト を定義 できます。 アクション スクリプト を使用 すると、フィールド のアクティブ と非アクティブ の切り替え など、インストール ダイアログ ボックス のダイアログ ボックス 要素 (フィールド) の属性 と内容を すぐに変更 することができます。

アクション スクリプト の例は、パッケージ をインストール した システム の `$SDPcker/examples/advanced/actionscript` ディレクトリ にあります。

スプラッシュ 画面

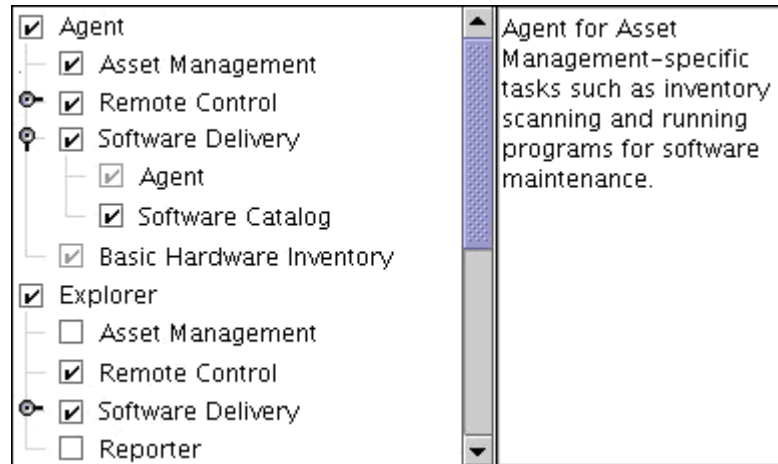
通常、インストール の開始 時に スプラッシュ 画面 が表示 され、製品、製品のバージョン、および 著作権 に関する グラフィック の情報が ポップアップ します。スプラッシュ 画面 は、インタビュー フェーズ のダイアログ ボックス の前に インタラクティブ なインストール で表示 されます。スプラッシュ 画面 は、インタビュー を開始 すると 自動的に 表示 されます。

GIF 形式 のグラフィック (スプラッシュ 画面 の背景 のグラフィック として 使用) とグラフィック の左下 の端 にある (それぞれの 枠から 10 ピクセル) オプション の文字列 を指定 できます。

プロトタイプ ファイル では、キーワード `#splashscreen` を使用 して スプラッシュ 画面 を定義 します。リソース セクション で文字列 を指定 して ローカライズ できます。スプラッシュ 画面 のテキスト の改行 を定義 するには、文字列 に `<br>` タグ を挿入 します。

チェック ボックス ツリー

チェック ボックス ツリーには、階層構造で製品コンポーネントがリストされます。以下のスクリーンショットは、チェック ボックス ツリーの例を示しています。



コンポーネントを選択すると、サブコンポーネントのすべてが自動的に選択されます。参照コンポーネントが選択される場合、必要なコンポーネントも選択されます。

コンポーネント リストは、スクロール バーを使用して解析することができます。

必要に応じて、選択したコンポーネントについての情報を示すテキスト エリアを定義することができます。

2つのキーワードのペアによって、チェック ボックス ツリーの制御を定義することができます。

#tree_begin、#tree_end

ツリーには、チェック ボックス のリストが含まれます。

#scrollarea_begin、#scrollarea_end

スクロール領域には、チェック ボックス ツリーまたはチェック ボックス のリストが含まれます。

例：チェック ボックス ツリーの作成および変更

この例では、選択したインストール ダイアログ ボックスにチェック ボックス ツリーを追加したり、それを変更する方法に関する情報を示しています。

この例では、以下の内容について説明します。

- チェック ボックス ツリーの追加
- 既存のチェック ボックス ツリーへのノードの追加
- 既存のチェック ボックス ツリーからのノードの削除

チェック ボックス ツリーを追加するには、以下の手順に従います。

1. プロジェクト ペインから、パッケージを選択します。
一般的なパッケージ オプションが表示されます。
2. 適切なダイアログ ボックスを [ダイアログ ボックス] タブ (サブプロジェクト ペイン) から選択します。
3. ツールボックスから、チェック ボックス ツリー アイコンをワークボックス内の目的の位置にドラッグアンドドロップします。
4. ワークボックスのチェック ボックス ツリー アイコンをクリックします。
空のエリアが開き、そこでチェック ボックス ツリーを作成および編集します。
5. このエリアで右クリックして、コンテキスト メニューから [チェック ボックスの追加] を選択します。
[入力] ダイアログ ボックスが表示されます。
6. チェック ボックスの名前を入力し、[OK] をクリックします。
チェック ボックスが編集エリアに表示されます。チェック ボックスの名前が、チェック ボックスの横に説明のテキストとして表示されます。

チェック ボックス ツリーにノードを追加するには、以下の手順に従います。

1. チェック ボックスを右クリックして、コンテキスト メニューから [子チェック ボックスの追加] を選択します。
[入力] ダイアログ ボックスが表示されます。
2. 子チェック ボックスの名前を入力し、[OK] をクリックします。
選択したチェック ボックスの 1 レベル下にチェック ボックスが表示されます。

チェック ボックス ツリーからノードを削除するには、以下の手順に従います。

1. チェック ボックス ツリーでノードを右クリックします。
コンテキスト メニュー オプションが表示されます。
2. コンテキスト メニューから [削除] を選択します。
選択したノードがツリーから削除されます。

インストール ダイアログ ボックスのフォント指定

PIF SDK では、すべてのダイアログ ボックスの要素に対して、Java インストール ダイアログ ボックス内のフォント サイズおよびフォントの表示を適合させることができます。以下のフォント プロパティを変更できます。

- サイズ
- 太字
- 斜体
- 色 (RGB 形式)
- Java 論理フォント : Serif、SansSerif、Monospaced、Dialog、および DialogInput
論理フォントは、Java ランタイム システムによって最適な物理フォントにマップされます。

フォント プロパティはダイアログ ボックスの要素とは別に保存されています。そのため、フォント プロパティをいったん定義すると、それらをインストール ダイアログ ボックスの任意のダイアログ ボックス要素に割り当てることができます。

フォントおよびそれらのプロパティは、[フォントの選択] ダイアログ ボックスを使用するか、プロトタイプ ファイルの中の **@FONTLIST** セクションで直接指定されます。

関連するすべてのダイアログ ボックス要素 (つまり、キーワード) には、**@FONTLIST** セクションのフォント指定を参照するオプションの *fontname* プロパティが含まれています。

例 : ダイアログ ボックスの要素のフォントの変更

この例では、選択したダイアログ ボックスの要素のフォントを変更する方法に関する情報を示しています。テキストを表示する各ダイアログ ボックスの要素のプロパティ ダイアログ ボックスに [フォント名] ドロップダウン リストが含まれます。

ダイアログ ボックスの要素のフォントを変更するには、以下の手順に従います。

1. ダイアログ ボックスで要素を選択します。
選択したダイアログ ボックスの要素のプロパティ ペインが表示されます。
2. [フォント名] ドロップダウン リストの隣にある参照ボタン (...) をクリックします。
[フォントの選択] ダイアログ ボックスが表示されます。
3. 次のパラメータに値を入力します。

フォント名

指定するフォントの名前を示します。

フォント タイプ、フォント サイズ

フォントのタイプとサイズを指定します。

効果

ボールド、イタリック、またはこれらの両方を指定します。

色

ドロップダウン リストから、使用できるフォント カラーを指定します。

注：指定したフォントのプレビューをプレビュー領域に表示できます。

4. 新しいフォントを追加するには、[追加] をクリックします
選択したダイアログ ボックスの要素に新しいフォントが追加されます。

注：また、ダイアログ ボックスの要素の新しいフォントを定義しない場合、[フォント名] ドロップダウン リストからフォントを選択することもできます。

ダイアログ ボックスの入力順序の指定

PIF パッケージのインストール時に、インストール ダイアログ ボックスの入力フィールドを処理する順序を指定できます。PIF パッケージを後にインストールする間にこのインストール ダイアログ ボックスが呼び出されたとき、このダイアログ ボックスの入力フィールドは指定の順序で処理されます。

インストール ダイアログ エディタのツールバーの [入力順序] チェック ボックスを使用したり、プロトタイプ ファイルの @Dialog セクションで直接指定することにより、入力順序を指定します。#inputorder ダイアログ ボックスのプロパティには、(入力) ダイアログ ボックスの要素の順序リストが格納されます。

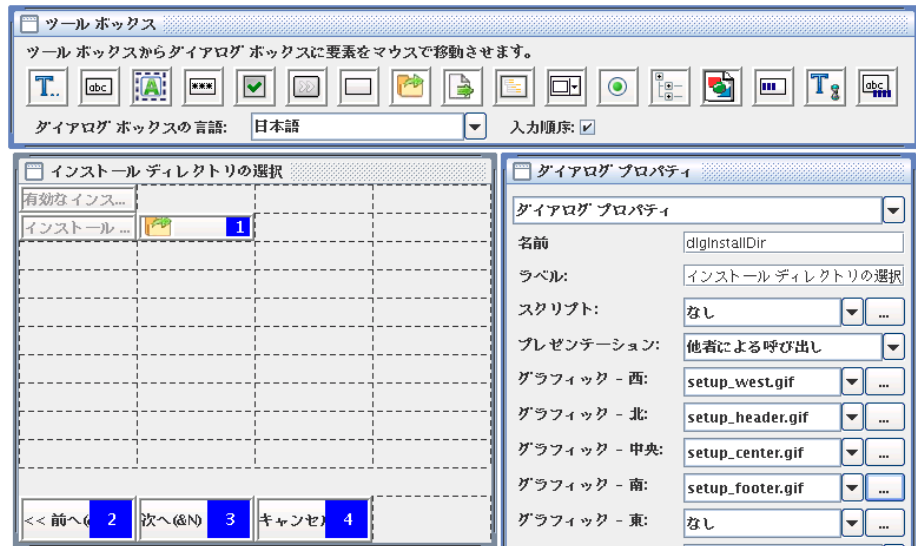
ダイアログ ボックスの入力順序を指定するには、以下の手順に従います。

1. [ダイアログ ボックス] タブ (サブプロジェクト ペイン) で、ダイアログ ボックスの要素の入力順序を指定するダイアログ ボックスを選択します。

右側のペインに、選択したダイアログ ボックスが表示されます。

2. インストール ダイアログ エディタのツールボックスで入力順序チェック ボックスをオンにします。

以下の図に示すように、ダイアログ ボックス作成エリアの入力フィールドに青いボックスが表示されます。



3. 入力フィールド (青いボックス) を 1 つずつクリックし、入力シーケンスを指定します。

PIF パッケージを後にインストールする間にこのインストール ダイアログ ボックスが呼び出されたとき、このダイアログ ボックスの入力フィールドは指定の順序で処理されます。

インストール ダイアログ ボックス エディタ

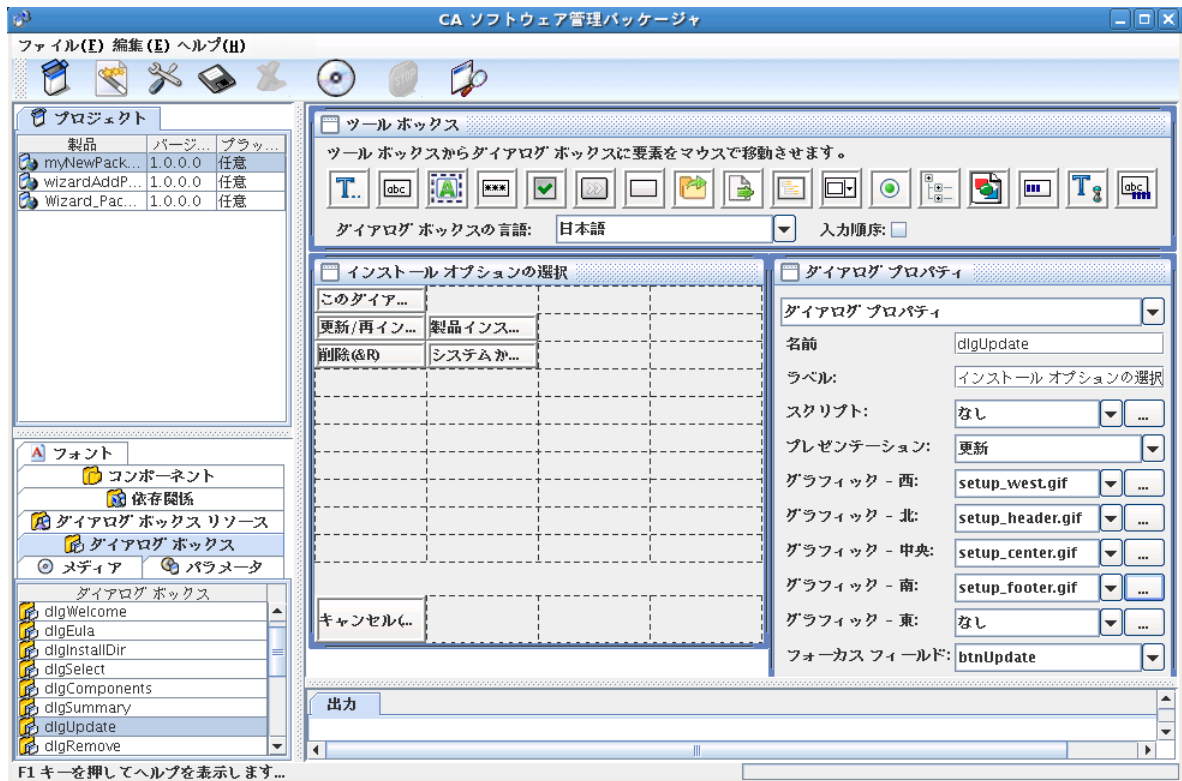
パッケージ GUI には、インストール ダイアログ ボックス エディタおよび複数のツールがあり、ニーズに合わせて標準のインストール ダイアログ ボックスを変更したり、個々のインストール ダイアログ ボックスを新規作成することができます。インストール ダイアログ ボックス エディタを使用すると、ツールボックスのダイアログ ボックス要素を作業エリアにドラッグ アンド ドロップし、プロパティ ボックスを使用してダイアログ ボックス要素をカスタマイズできます。

注：ドラッグ アンド ドロップのタスクには、次のステップが含まれています。

1. ツールボックスで要素を選択します。
2. マウスの右ボタンを押しながら、ダイアログ エディタ パネル（作業領域）に要素を移動します。
3. 要素を配置する場所でマウス ボタンを離します。

インストール ダイアログ ボックス エディタは、サブプロジェクト ペインの [ダイアログ] タブでダイアログ ボックスを選択するか、または [ダイアログ] タブを右クリックして新しいダイアログ ボックスの追加を選択すると開きます。

以下のスクリーンショットに、インストール ダイアログ ボックスのエディタ ウィンドウの例を示します。



ダイアログ ボックスの作成中などに選択したダイアログ ボックスの最新レイアウトを確認する場合は、パッケージ GUI のツールバーにある [ダイアログ ボックスのテスト] アイコンをクリックします。

ワークボックス

ワークボックスは、インストール ダイアログ ボックス エディタの作業面です。インストール ダイアログ ボックスの一般的なレイアウトは、以下のとおりです。

ヘッダ領域

インストール ダイアログ ボックスのヘッダ(タイトル)が含まれます。ヘッダ領域は作業ボックスには表示されません。ヘッダは、[ダイアログ ボックス] プロパティ ボックスのラベル フィールドで定義されます。

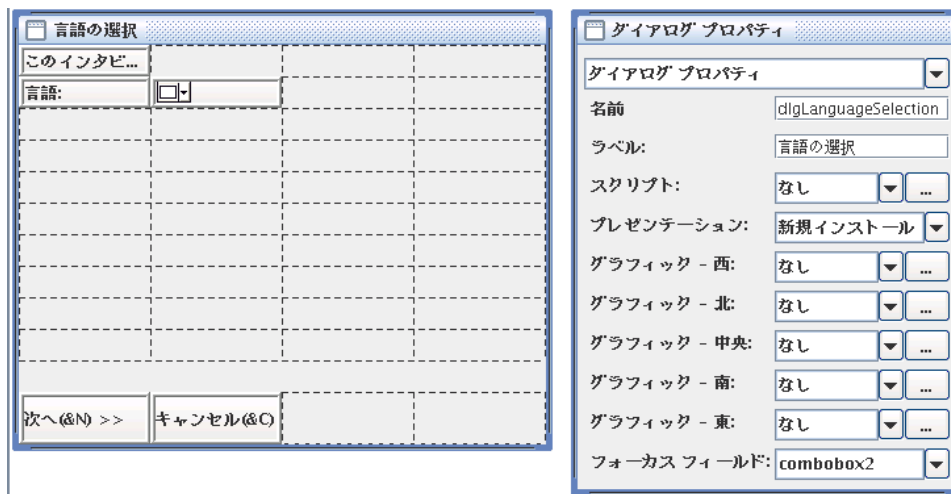
入力領域

インストール ダイアログ ボックスのメイン エリアを示します。入力領域は 10x4 グリッドで表されます。ナビゲーション ボタンを除くすべてのダイアログ ボックス要素は、ツール ボックスから入力領域にドラッグアンドドロップ機能を使用して移動できます。入力領域では、すべてのダイアログ ボックス要素がドラッグアンドドロップ機能を使用して移動できます。ダイアログ ボックス要素の位置を合わせるため、10 行 4 列の点線の仮想グリッドが入力領域に配置されます。グリッドのレイアウトは Java で制御できます。一部のダイアログ ボックス要素は、その位置や内容によって、ほかのダイアログ ボックス要素のサイズや位置に影響を与えることがあります。

ナビゲーション領域

ナビゲーション ボタンのみを定義できます。ナビゲーション ボタンはインストール ダイアログ ボックス間の移動に使用します。ナビゲーション領域は作業ボックスの最終行にあり、1x4 グリッドで表されます。

次の画面ショットには、ヘッダ、入力領域、およびナビゲーション領域が表示されています。



作成中のインストール ダイアログ ボックスの現在のレイアウトを表示するには、ツールバー上の [ダイアログ ボックスのテスト] アイコンをクリックします。

グラフィック領域

Java インストールのダイアログボックス内で、バックグラウンドグラフィックを定義できます。以下のように、コンパスの点を使用して、異なるグラフィック領域にグラフィックの位置を合わせることができます。

北		
西	中央	東
南		

バックグラウンドグラフィックの追加的な領域は、ダイアログボックスのヘッダ領域およびフッタ領域です。下の領域はナビゲーション領域として使用されます。

グラフィックおよびバックグラウンドの色に対してサポートされているファイル形式は GIF のみです。

プロパティパネルで、既存のグラフィックを選択したり、ダイアログボックスのプロパティに新しいグラフィックを割り当てたりできます。作業ボックス内の空のフィールドをクリックすると、ダイアログボックスのプロパティが表示されます。グラフィックを挿入するには、GIF ファイルをプリインストールコンポーネントに挿入しておく必要があります。これにより、[ダイアログ] プロパティダイアログボックスのドロップダウンリストからそのグラフィックを選択できます。

グラフィックのサイズはダイアログボックスの全体サイズに影響を与えるため、グラフィックが 800 x 600 ピクセル (SVGA) の画面解像度に対応する 400 x 200 ピクセル (概算) を超えないようにすることをお勧めします。

グラフィックの追加は、Java インストールダイアログボックス内でのみ可能です。VT100 モードダイアログボックスでは、これらの機能は無視されます。

以下の図に示されているように、作成されるインストールダイアログボックスの現在のレイアウトを表示するには、(スクリーンショットでハイライトされている) ツールバー上の [ダイアログボックスのテスト] アイコンをクリックします。



ツールボックスおよびダイアログ ボックス要素

ツールボックスには、インストール ダイアログ ボックスに表示されるすべての種類のダイアログ ボックス要素があります。

インストール ダイアログ ボックスで利用できるダイアログ ボックス要素は、パッケージ GUI 上のインストール ダイアログ ボックス エディタのツールボックスでアイコンとして表示されます。この要素は、ツールボックスからワークボックス領域内の目的の位置にドラッグ アンド ドロップすることができます。

プロトタイプ ファイルでは、インストール ダイアログ ボックスのダイアログ ボックス要素およびその位置は、適当なキーワードとプロパティを使用して [ダイアログ] セクションで指定できます。

特定のダイアログ ボックス要素はグレーアウトできるため、現在のコンテキストで入力フィールドを使用しない場合は、その入力フィールドをブロックできます。そのため、内部パラメータ `PIF_CHANGEPARAM_parameter` を使用します。

ボタン



ボタンには、新しいダイアログ ボックスを開く、スクリプトを実行する、またはインストール モード（標準、最小、またはユーザ固有）を選択するなど、ボタンをクリックしたときに実行されるアクションの仕様が含まれています。

プロトタイプ ファイルでは、キーワード `#button` を使用すると、ダイアログ ボックスの入力領域内のボタンを定義したり、ボタンに表示されるテキストまたはアイコン、パラメータ値の割り当て、アクション、またはボタンをクリックしたときに実行されるスクリプトを指定したりできます。仕様の詳細については、「[キーワード](#) (P. 132)」および「[プロパティ](#) (P. 144)」を参照してください。

ボタンに表示されるテキストは、別個の場所であるプロトタイプ ファイルの [Resource] セクションに格納されます。この区分によって、テキストのローカライズが容易になり、同じプロトタイプ ファイルにローカライズされたさまざまなバージョンを容易に保存することができます。

ボタンのテキスト文字列では、アンパサンド (&) 文字を使用してショートカットを定義できます。アンパサンド文字は、ショートカットとして使用する文字の前に置きます。たとえば、ボタンテキストとして **Ne&xt** と記述した場合は、小文字の **x** が定義済みショートカットになります。ボタンテキスト文字列には、最大で **1024** 文字を使用できますが、**CR**(キャリッジリターン)または**LF**(ラインフィード)などの制御文字は使用できません。

重要： HP-UX 11.0 および 11.i の共通デスクトップ環境 (CDE) セッションでインストール ダイアログ ボックスを実行すると、インストール ダイアログ ボックスにあるショートカット キーは機能しません。

チェック ボックス



チェック ボックスの論理値は、**True** (チェック ボックスがオン) または **False** (チェック ボックスがオフ) を取ることができ、インストール ダイアログ ボックスでのアイテムの選択 (**True**) またはアイテムの解除 (**False**) に使用できます。インストール ダイアログ ボックスでは、チェック ボックスはオプション要素の選択や製品プロパティの設定などに使用されます。チェック ボックスをオンにしたり、オフにすることで一連のアクティブなパラメータ値の割り当てを定義できます。

チェック ボックス ツリーをサポートするために、説明文、依存するチェック ボックスのダイアログ ボックス、およびアクション スクリプトを指定できます。アクション スクリプトを使用すると、インストール ダイアログ ボックスのフィールドの属性または内容がすぐに変更されます。

チェック ボックス ツリー



チェック ボックス ツリーとは、たとえば、選択用に製品コンポーネントを一覧表示するチェック ボックスの階層ツリー構造です。インストール ダイアログ ボックスでチェック ボックス ツリーの構造を定義するには、ツール バーから [チェック ボックス ツリー] アイコンを使用します。

詳細については、「チェック ボックス」の下も参照してください。

コンボ ボックス/リスト ボックス



このコントロールは、[入力可能] プロパティを選択しているかどうかに応じて、コンボ ボックスまたはリスト ボックスとして機能します。[入力可能] が選択されている場合、コントロールはコンボ ボックスとして機能し、[入力可能] が選択されていない場合はリスト ボックスとして機能します。

コンボ ボックス

コンボ ボックスには選択可能な値の一覧が示され、いずれかの値を選択できます。また、ユーザは別の値を入力することもできます。入力を検証するシェルスクリプトを指定できます。別のスクリプトを指定して、インストール ダイアログ ボックスのフィールドの属性や内容をただちに変更することもできます。**VT100** のダイアログ ボックスでは、矢印キーまたは+ と - キーを使用したナビゲーションが可能です。初めて+または-キーを押した場合、可能な値のリストがポップアップされます。さらに+または-キーを押すと、リスト内を上下に移動できます。キャリッジリターンによって、選択された値が確認され、リストが閉じられます。

リスト ボックス

リスト ボックスには選択可能な値の一覧が示され、いずれかの値を選択できます。**VT100** のダイアログ ボックスでは、矢印キーまたは+ と - キーを使用したナビゲーションが可能です。初めて+または-キーを押した場合、可能な値のリストがポップアップされます。さらに+または-キーを押すと、リスト内を上下に移動できます。キャリッジリターンによって、選択された値が確認され、リストが閉じられます。

ファイル エリア



ファイル エリアには、ファイルの内容が表示されます。この内容は、ブリインストール コンポーネントによって提供される必要があります。ファイル エリアは、エンドユーザ使用許諾契約、ヘルプ トピック、または **Readme** ファイルなどのサイズの大きいテキスト ファイルの表示に使用されます。また、製品で使用するすべてのパラメータ値の要約など、内部の管理目的で使用するファイルの内容もファイル エリアに表示できます。テキスト内のナビゲーションおよびスクロールが可能です。

パラメータを使用して、ファイル名を動的に設定できます。特殊用途のパラメータは、PIF_RESPONSE_FILE、PIF_ERROR_FILE、および PIF_LOG_FILE です。たとえば、内部変数 \$PIF_RESPONSE_FILE をファイル名として指定すると、内部的に使用される応答ファイルの内容を表示できます。

ファイルエリアには、以下の拡張子を持つファイルが表示されます。

- Java ユーザ インターフェース : .htm、.html、.rtf、および .txt
- VT100 ユーザ インターフェース : .txt ファイル

ファイルエリア機能を使用すると、1つのファイルエリアで異なるファイル形式を使用できます。

たとえば、次のファイルエリアの仕様では、Java ユーザ インターフェースの場合は readme.html が使用され、VT100 ユーザ インターフェースの場合は readme.txt が使用されます。

```
#filearea: 1 , 1 , MyFileArea , readme ;
```

VT100 モードのインストール ダイアログ ボックスのファイル エリア内を移動するには、以下のキーを使用します。

- プラス記号 (+) または Ctrl + F
次のページに進みます。
- マイナス記号 (-) または Ctrl + B
前のページに戻ります。
- タブ
ファイル エリアでの操作を終了します。

ファイルの選択



「ファイル選択」ボタンで PIF パッケージ用のファイルを選択できます。

イメージ



イメージとは、インストール ダイアログ ボックスに追加できるピクチャのことです。イメージは元のサイズで表示することもできますし、特定のサイズに拡大縮小することもできます。

インストール ディレクトリ フィールド



インストール ディレクトリ フィールドでは、PIF 製品またはそのいずれかのコンポーネントのインストールパスを指定します。Java インストール ダイアログ ボックスを作成するときに、ディレクトリ選択パネルを表示するための参照ボタン [...] が自動的に追加されます。インストール ディレクトリ フィールドに入力されたテキスト文字列にはパラメータが割り当てられ、これはインストール プロセスに渡されます。つまり、ディレクトリ情報を保管するパラメータを指定する必要があります。このパラメータは、製品またはコンポーネントのインストール ディレクトリ パラメータと一致する必要があります。入力フィールドの内容を検証するスクリプトを指定できます。

ラベル



ラベルには、インストール ダイアログ ボックスの別の要素について記述されています。ラベルは、通常、インストール ディレクトリの指定やチェック ボックスなどの入力フィールドの名前を示します。インストール ダイアログ ボックスでは、ラベルテキストは複数行にすることはできません（ダイアログ ウィンドウ内で折り返すことはできません）。

ナビゲーション ボタン



ナビゲーション ボタンでは、次に表示されるインストール ダイアログ ボックスを指定します。これにより、別の操作を開始できます。次のインストール ダイアログ ボックスを指定するシェルスクリプトを指定できます。ツールボックスからのナビゲーション ボタンのドラッグ アンド ドロップは、ワークボックスの下にあるナビゲーション エリアに対してのみ行うことができます。

以下の表は、ナビゲーション ボタンに関連付けることができるアクションを示します。

アクション	説明	アクション 番号
キャンセル	インストールを中止します。	0

アクション	説明	アクション 番号
インストール	製品をインストールします。	1
次のダイアログ ボックス	次のダイアログ ボックスを表示します。 注： 次のダイアログ ボックスの名前は、「次のダイアログ ボックス」フィールドで定義する必要があります。	2
削除	インストールされた製品を削除します。	3
前のダイアログ ボックス	前のダイアログ ボックスに戻ります。	4
動的ダイアログ ボックス	シェル スクリプトを実行して次のダイアログ ボックスの名 前を決定し、そのダイアログ ボックスを表示します。 注： スクリプトの名前は、「スクリプト」フィールドで定義 する必要があります。	5
スクリプトの起動	インストーラ GUI を終了して、ユーザ スクリプトを起動しま す。 注： スクリプトの名前は、「スクリプト」フィールドで定義 する必要があります。	6
移動	別のダイアログ ボックスに移動して、そのダイアログ ボッ クスを起点とします。（直前のダイアログ ボックスの）「戻 る」ボタンからは、その新しいダイアログ ボックスの前に使 用されたダイアログ ボックスに戻ります。 注： ターゲット ダイアログ ボックスの名前は、「次のダイ アログ ボックス」フィールドで定義する必要があります。	7
スクリプトを開始 して続行	現在のダイアログ ボックスを閉じ、指定したスクリプトを開 始して、最初のダイアログ ボックスを再度開始します。 注： スクリプトの名前は、「スクリプト」フィールドで定義 する必要があります。	8
スクリプトの実行 (ボタンのみ)	指定したスクリプトを実行し、指定した結果ペインに結果を 表示します。結果ペインは、進捗状況エリアまたはテキスト エリアになります。	9

インストール ダイアログ ボックスを作成する場合、ナビゲーション エリアには 4 つのボタンを定義できます。これらの機能を示すために、ボタン上に表示されるテキストまたはアイコンを指定できます。[動的] ダイアログ ボックスのアクションを選択した場合は、使用するスクリプトを指定する必要があります。

プロトタイプ ファイルでは、ダイアログ ボックスのセクションにキーワード **#navbutton** を使用して、ナビゲーション ボタンを定義します。パッケージ GUI では、[ナビゲーション ボタンのプロパティ] ダイアログ ボックスでナビゲーション ボタンを指定します。

ボタンのテキスト文字列では、アンパサンド (&) 文字を使用してショートカットを定義できます。アンパサンド文字は、ショートカットとして使用する文字の前に置きます。たとえば、ボタン テキストとして **Ne&xt** と記述した場合は、小文字の **x** が定義済みショートカットになります。デフォルトでは、テキスト文字列の最初の文字がショートカットになります。ボタン テキスト文字列には、キャリッジ リターン (**CR**) および行送り (**LF**) などの制御文字を使用することはできません。

キャンセル アクションが関連付けられている各ナビゲーション ボタンに対して、1 つ以上の [キャンセル] ダイアログ ボックスを定義する必要があります。

重要： HP-UX 11.0 および 11.i の共通デスクトップ環境 (CDE) セッションでインストール ダイアログ ボックスを実行すると、インストール ダイアログ ボックスにあるショートカット キーは機能しません。

パスワード フィールド



パスワード フィールドには、パスワードを入力することができます。パスワード情報は表示されず、アスタリスク (*) でマスクされます。入力データを検証するシェルスクリプトを指定できます。

進捗状況エリア



進捗状況エリアには、スクリプトの出力が表示されます。この機能は、PIF 製品のインストール中に長時間実行されるスクリプトの進捗状況を示す場合に使用できます。進捗状況エリアは、新規スクリプトが起動されるたびにリセットされます。現在処理中のスクリプトの名前は、[進捗状況] ダイアログ ボックスの [アクション] フィールドに表示されます。この機能は、RPM パッケージフォーマットのパッケージには使用できません。

プログレス バー



プログレス バーには、製品またはコンポーネントのインストール、またはインストール全体の進捗状況が表示されます。プログレス バー アイコンは、進捗状況ダイアログ ボックスの作成を指定した場合（ダイアログ ボックスのプロパティの「プレゼンテーション」フィールドが「進捗状況」に設定されている場合）にのみ、ツールボックスから作業領域にドラッグアンドドロップできます。

現在の製品またはコンポーネントのインストールの進捗状況は、以下の環境変数によって記述されます。

- PIF_PROGRESSBAR_PRODUCT
- PIF_PROGRESSBAR_COMPONENT
- PIF_PROGRESSBAR_COUNT

プログレス ラベル



プログレス ラベルには、インストールプロセスで変化したインストールの進捗状況に関する情報が表示されます。プログレス ラベルには、通常、バーに先立って情報が表示されます。ここには、現在のインストール アクション（インストール、バックアップ、削除など）、インストール フェーズ（オプション）、インストールされているファイルやコンポーネントの名前、またはインストールの経過時間が表示されます。プログレス ラベル アイコンは、進捗状況ダイアログ ボックスの作成を指定した場合（ダイアログ ボックスのプロパティの「プレゼンテーション」フィールドが「進捗状況」に設定されている場合）にのみ、ツールボックスから作業領域にドラッグアンドドロップできます。

現在実行中のインストール アクション、インストール中のファイル、またはコンポーネント名などは、以下の環境変数を介して記述されます。

- PIF_PROGRESSLABEL_ACTION
- PIF_PROGRESSLABEL_FILE
- PIF_PROGRESSLABEL_COMPONENT
- PIF_PROGRESSLABEL_ELAPSEDTIME
- PIF_PROGRESSLABEL_PHASE

ラジオ ボタン



ラジオ ボタンを使用すると、複数の値を切り替えることができます。ラジオ ボタンを選択するときに、**parameter=value** の割り当てリストを指定できます。シェルスクリプトを指定すると、インストール ダイアログ ボックスのフィールドの属性または内容がすぐに変更されます。

ラジオ ボタンは、スペース バーまたはキャリッジ リターンを使用して選択できます。ラジオ ボタンのグループは、ダイアログ内で横ではなく上下に並べる必要があります。

テキスト エリア



テキスト エリアには、インストール ダイアログ ボックス内のテキストが表示されます。テキストは複数行にすることができ、ダイアログ ボックスのウィンドウ内で自動的に折り返されます。

インストール ダイアログ ボックスの同じ行に、テキスト フィールドとテキスト 領域を使用することはお勧めできません。

テキスト フィールド



テキスト フィールドは、インストール ダイアログ ボックスの標準の入力フィールドです。入力したテキストはパラメータに割り当てられ、インストール スクリプトに渡された後、スクリプトで使用できるようになります。パラメータは、テキスト フィールドのプロパティとして定義する必要があります。テキスト フィールドに入力したテキストを検証するスクリプトを指定できます。

インストール ダイアログ ボックスの同じ行に、テキスト フィールドとテキスト 領域を使用することはお勧めできません。

プロパティ ボックス

プロパティ ボックスには、ワークボックスで選択したダイアログ ボックス要素のプロパティが表示され、プロパティの値を入力または変更できます。

また、プロパティ ボックスでは、現在インストール ダイアログ ボックスに定義されているダイアログ ボックス要素をドロップダウン リストから選択し、その要素のプロパティを表示または変更できます。たとえば、ダイアログ ボックスのタイトルを変更するには、ドロップダウン リストからダイアログ ボックスのプロパティを選択します。

さまざまなパッケージ フォーマットのサポート

PIF SDK によって管理される製品は、以下を含むさまざまなパッケージング形式でパッケージすることができます。

- PIF (CA Technologies Product Interchange Format)
- PKG (UNIX System V リリース 4 標準パッケージフォーマット)
- RPM (Red Hat パッケージ マネージャ フォーマット)
- IBM AIX パッケージング形式
- Hewlett Packard (HP-UX) パッケージング形式
- Apple Macintosh Mac OS X パッケージング形式

新製品は PIF 以外の形式では作成できませんが、PIF SDK の機能を使用して別のパッケージング形式で既存の製品パッケージの登録、インストール、または少なくとも一覧表示を行うことができます。

インストーラは、PIF 製品から、PKG、RPM、AIX、および HP-UX パッケージング形式を含む他のパッケージング形式の必須パッケージへの従属関係を管理します。

AIX および HP-UX パッケージは、lsm コマンドで一覧表示され、インストーラの依存関係チェッカーによって考慮されます。

System V リリース 4 システム (つまり、Sun Solaris、SCO UnixWare、および NCR) 上のパッチは、lsm コマンドで一覧表示され、インストーラの依存関係チェッカーによって考慮されます。

/Applications ディレクトリにインストールされている Apple Macintosh Mac OS X パッケージは、DSM エクスプローラの中で一覧表示され、登録されます。インストーラが提供するクエリおよび削除方式 (lsm コマンド) は、Mac OS X パッケージにも適用されます。Mac OS X パッケージのインストール中、インストーラによってすべてのアイコンおよびメニューが調節されます。

PIF パッケージは、Mac OS X を実行しているコンピューターに登録でき、OSX パッケージ マネージャ内でそれらのコンピュータ上に表示されます。

PKG、RPM、AIX、HP-SD、および OSX の各パッケージ統合の機能

次の PKG、RPM、AIX、HP-SD、および OSX の統合の機能は、インストーラから入手できます。

- PKG、RPM、AIX、HP-SD、および OSX の各パッケージの相互依存関係を定義します
- RPM、AIX、HP-SD、および OSX の各パッケージ階層をインストールします
- RPM、AIX、HP-SD、および OSX の各パッケージ階層を削除します
- PIF パッケージに渡された内部パラメータ (PIF_xxx) も RPM、AIX、HP-SD、および OSX の各パッケージに渡されます。

RPM 固有の統合機能

以下の機能は、RPM パッケージ統合に限り使用することができます。

- RPM にインストール パスを渡す (オプション `-prefix`)

RPM のインストールにインストール ディレクトリのパスを渡すため、特別な PIF パラメータが使用されます。このパラメータは、以下の名前を持ちます。

`PREFIX_rpm_packagename`

RPM パッケージに含まれる破線は、アンダースコアに置き換えられます。

RPM パッケージのインストール パスを含むパラメータの例は、以下のとおりです。

`$PREFIX_openoffice - openoffice` のインストール パスを含みます。

`$PREFIX_XFree86_libs - XFree86-libs` のインストール パスを含みます。

- RPM インストールの進捗状況を表示します (オプション **-percent**)。オプション **-percent** による RPM のインストールの進捗に関する情報が評価され、[進捗状況] ダイアログ ボックスの PIF 製品のプロGRESS バーが進められます (\$PIF_PROGRESSBAR_PRODUCT)。

注：RPM には、削除時の進捗情報は表示されません。

- RPM パッケージごとに RPM コマンドを設定します。これらのオプションは、スクリプト内でコメントとして定義されます。追加のインストール オプションを RPM プリインストール (**prein**) スクリプトに定義することができ、削除オプションを RPM の削除前 (**preun**) スクリプトに定義できます。これらスクリプトにおけるキーワードは、以下のとおりです。

prein スクリプト：

```
# PIF_RPM_INSTALL_OPTION=  
# PIF_RPM_REINSTALL_OPTION=  
# PIF_RPM_UPDATE_OPTION=
```

preun スクリプト：

```
# PIF_RPM_REMOVE_OPTION=
```

メソッド **REINSTALL** については、デフォルトは **--replacepkgs** に設定されます。メソッド **INSTALL**、**UPDATE**、および **REMOVE** については、デフォルトは設定されません。

注：手動モードでは、インストール、再インストール、および更新ジョブにオプション **-percent** が自動的に追加されます。

prein スクリプトの例は、以下のとおりです。

```
#!/bin/sh  
# following option is used to overwrite the RPM update installation switches:  
# PIF_RPM_UPDATE_OPTION=--replacepkgs --replacefiles  
exit 0
```

RPM パッケージの使用に関する制限

RPM パッケージ使用の場合は、以下の制限が適用されます。

- PIF 製品が依存する RPM 製品が別の PIF 製品に依存する場合など、依存関係を混在して設定することはできません。
- RPM パッケージへのアクセスには、**library.dct** のみが使用されます。

- RPM の共有ライブラリ、シェルなどへの自動要求はサポートされていません。
- (完全なトランザクションのための) ディスク領域の確認は使用できません。
- RPM パッケージのバックアップはサポートされていないため、ロールバックを行うことはできません。
- 該当する RPM 情報が提供されないため、RPM パッケージ削除の進捗状況は [進捗状況] ダイアログボックスに表示できません。
- RPM 操作の一貫性を保つために、RPM インストールのキャンセルは、特定の RPM インストールが完了すると期限が終了します。
- PIF と RPM の両方の種類のパッケージを、同じディレクトリに同じ製品名で含めることはできません。

RPM および PKG のネイティブ スイッチ

RPM および PKG インストール用にネイティブ スイッチを定義できます。

PKG パッケージは、「pkgadd」コマンドを使用して、それぞれのプラットフォーム (Solaris、UnixWare、NCR、SINIX など) にインストールされます。lsm インストーラは、スイッチを使用してコマンドを起動します。

コマンドラインで lsm オプションの `-nativeswitches` を使用して別のスイッチを追加できます。lsm は、これらのネイティブ スイッチを pkgadd コマンドに追加します。

例：

```
lsm -i MY_TEST_PKG -nativeswitches "¥-G"
```

ネイティブ スイッチ「-G」が pkgadd コマンドに追加されています。このスイッチによると、PKG パッケージ「MY_TEST_PKG」が、Solaris システムの「大域ゾーン」にのみインストールされます。

インストーラ機能の概要

インストーラは、さまざまなターゲット コンピュータの動作環境へのソフトウェアのインストールを管理します。インストーラのすべての機能は、コマンドラインから lsm コマンドおよび多数のオプションを使用することで実行できます。

関連するインストーラ機能の概要を以下に示します。

- PIF、PKG、AIX、HP-SD、OSX、または RPM 製品のインストールを、ダイアログ ボックスを使用して行うか、または自動 (サイレント) モードで行います。自動インストールをカスタマイズするために、応答ファイルを追加できます。
- ターゲット コンピュータにインストールされている PIF 製品を更新します。更新インストールでは、インストーラによってインストール済み PIF 製品のバックアップが作成されます。更新インストールでは、以前にインストールされた製品と新しい製品との差分が評価されます。使用されないファイルは、ターゲット コンピュータから自動的に削除されます。
- 製品およびコンポーネントに定義されたすべての依存関係を、必要に応じて複数の階層にわたって解決し、製品をインストールする前に依存関係にある製品がすべてインストールされるようにします。製品を削除するときに、可能な場合は依存関係にある製品を (複数の階層にわたって) すべて削除します (製品と共に配布されインストールされた依存製品にのみ適用)。
- 新しいインストール ダイアログ ボックスをテストし、ダイアログ ボックスに入力した値を使用して応答ファイルを作成します。
- ターゲット コンピュータのチェックサムを検証します。製品を作成するときに、パッケージによってプロトタイプ ファイルにチェックサムが追加されます。PIF 製品ファイルは、CRC チェックサム (CRC = 巡回冗長検査) によって保護されています。ターゲット コンピュータではインストーラがこの CRC チェックサムを知っていて、インストールされた PIF 製品のファイルが変更されていると警告が表示されます。PIF 製品ファイルが外部から変更された場合、インストールは失敗します。
- PIF 製品のインストール時に、共有ファイルを管理します (共有ファイルがインストールされ、複数の製品から同時に使用されます)。PIF 製品のアップデートに共有ファイルが含まれている場合は、PIF 製品をアップデートするときにその共有ファイルをアップデートするかどうか調べられ、既存ファイルと新しいファイルのチェックサムが異なっていて、新しいファイルの変更日が既存のファイルより新しい場合にのみ、その共有ファイルはアップデートされます。それ以外の場合、共有ファイルは変更されません。

以下のいずれかの条件があてはまる場合、PIF 製品のインストールは中止されます。

- 共有ファイルのファイル タイプが異なる場合 (標準ファイルがインストールされ、同じ名前のディレクトリが作成される場合など)
- 共有リンクのターゲットが異なる場合
- インストールされた製品の整合性をチェックします。製品ファイルの存在およびアクセス権がチェックされます。

- パッケージを削除した後でファイルをリストアします。新しい PIF のインストールが始まると、インストーラによって、その製品のインストールで書きされるすべてのファイルが書き留められます。これらのファイルは、**.notremove** という特別なファイルに保存されます。デフォルトで、**.notremove** ファイルがインストーラのホーム ディレクトリ (`/opt/CA/SharedComponents/installer`) に作成されます。このファイルはチェックサムで保護されており、ユーザはそれを変更できません。また、この中に、新しいインストールによって書きされたファイルがすべて含まれています。

パッケージが削除されて、それ以外に非リムーバブル ファイルを参照しているパッケージがない/なくなると、**.notremove** からそのエントリが削除されます。その結果、ファイルは削除されないで、最後のコンテンツのままシステム上に残ります。

- インストールされた製品または製品ファイルをクエリして、製品のプロパティおよび依存関係の階層を表示します。
- インストールされたすべての製品、または共有されたコンポーネントのみを一覧表示します。使用されるインストーラのバージョンを出力します。
- インストールされた製品を削除します（必要に応じて自動モード）。
- ターゲット コンピュータから PIF 製品を削除すると、インストーラ自体もターゲット コンピュータから削除されます（クリーンアップ）。
- インストール準備フェーズが長い大きな PIF 製品をインストールするときに、進捗状況と説明テキストをコンソールに表示します。
- パッチをインストールします。
- パッケージを複数回インストールします。
- パッケージインストールが失敗した後に、ファイルをリストアします。
- 最初にパッケージインストールを行う前に存在していたファイルを残します。

解消された libncurses ライブラリへの依存関係

lsn インストーラは libncurses ライブラリにリンクされなくなりました。そのため、自動インストールは libncurses ライブラリがシステムにインストールされていなくても成功します。このライブラリは、VT100 を使用して対話する場合にのみ必要です。

PIF 製品の自己インストール

パッケージを使用すると、自己インストール形式の PIF 製品を作成できます。これは、PIF 製品をインストールする前に、ターゲット コンピュータにインストーラをインストールする必要がない製品です。自己インストール PIF 製品では、PIF 製品とインストーラが 1 つの実行可能ファイルにバンドルされています。

自己インストール PIF 製品の構造は、以下のとおりです。

- PIF 製品を抽出するスクリプト
- オプション：JRE
- インストーラ（圧縮済み）
- PIF 製品（圧縮済み）

自己インストール PIF 製品は、シェル スクリプトとして実行されます。

インストーラでは、自己インストール PIF 製品によってインストールを行う場合、顧客システムで以下のディレクトリを使用または作成します。

- インストーラ ファイルは、デフォルトで `$CASHCOMP/installer` ディレクトリに保存されます。このディレクトリは、作成または更新されます。
- プログラム `lsm` が `/usr/bin` に追加されます。

自己インストール PIF 製品は、パッケージ GUI で作成するか、またはコマンドラインで `pifself` コマンドを使用して作成できます。

自己インストール PIF 製品は、起動スクリプト、実際の PIF 製品、およびインストーラを含む自己インストール用のシェル スクリプト (`ca-sm-installer`) の 3 つの部分に分けることができます。Software Delivery 環境で作業をしており、自己インストール機能を必要としない場合は、これらの部分に分けることをお勧めします。この場合、分割された実際の PIF 製品のみをソフトウェア パッケージライブラリに登録することができます。自己インストール機能を使用する場合は、通常、前述の 3 つの個別のユニットが同じフォルダ内にあることを確認してから、自己インストールシェル スクリプト (`ca-sm-installer`) を実行する必要があります。

自己インストール PIF 製品を作成する場合は、自己インストール製品をインストールする動作環境をオプションとして指定できます。このオプションを使用すると、任意のプラットフォーム用に作成された PIF 製品から自己インストール製品を作成できます。

自己インストール PIF 製品における JRE の統合

Linux および UNIX のインストーラのグラフィカル ユーザ インターフェースは、Java テクノロジーに基づいています。そのため、ダイアログを実行するには、ターゲット システムに JRE (Java Runtime Environment) インストールしている必要があります。JRE がターゲット システムにインストールされていない場合、予想された Java ダイアログの代わりに、英数字の VT100 インストール ダイアログが実行されます。

VT100 ダイアログ モードになってしまうことを避けるために、JRE を自己インストール PIF 製品に統合できます。そうすることで、この製品がターゲット システムにインストールされ、実行されます。

JRE を自己インストール PIF 製品に統合するために、`pifself` コマンドは `-j` オプションを提供します。このオプションは、コア JRE ファイルを自己インストール PIF 製品に追加します。ターゲット システムでは、インストーラがこの統合された JRE を使用してインストーラ用の Java ウィザードを実行します。

`-j` オプションを使用して、統合する JRE バージョンを指定したり、Linux 用にコンパクト化されたバージョンを使用したりできます。たとえば、`/usr/java/j2sdk.1.4_10/jre` など、「`jre`」サブディレクトリを含むパス名を入力します。

この Linux プラットフォーム用にコンパクト化された JRE バージョン (V1.4.2.10) には、インストーラの Java ウィザードを実行するのに必要なクラスおよびファイルのみを含みます。

他のすべてのプラットフォーム用には、JRE バージョン 1.5 または 1.6 を自己インストール PIF 製品に統合できます。

製品ファミリの概念

製品ファミリの概念を使用すると、同じ製品ファミリに属しているすべてのパッケージを最新のバージョンに同期して更新できます（「1 回のアップグレードですべてのアップグレードを行います」）。

キーワード `#productfamily` には、論理的に適合するすべてのパッケージを組み合わせるプロパティが記述されています（例、製品ファミリ = CCS）。すべてのパッケージが単一の製品ファミリに属することがあります。

インストーラ コマンド `lsm` で `-u` オプションを使用すると、キーワード `#productfamily` に基づいて、同じ製品ファミリのすべての製品が最新バージョンに更新されます。このコマンドを使用すると、インストーラではシステム上の一致するすべての製品がスキャンされ、最新のバージョンに更新されます。

すべてのパッケージが最新バージョンに更新されます。ローカル フォルダまたは「アップデート CD」で提供されているバージョンの方が古い製品は、ダウンロードされません。

「アップデート CD」またはローカル フォルダの構成には、すべてのパッケージがフラット フォルダで提供されているか、または別個に提供される `library.dct` ファイルに記述されている必要があります。

RPM パッケージの場合は、製品ファミリの識別に「`group`」タグが使用されます。

PIF 以外の製品のマイグレーション

更新時に、以前にインストールした PIF 以外の製品を PIF 製品に変換すると役立つ場合があります。

このため、インストーラにはマイグレーション メカニズムがあります。このメカニズムは、(ユーザ作成の) シェル スクリプトに基づいています。このスクリプトは、ターゲット コンピュータへの PIF 製品の最初のインストール時に起動されます。このスクリプトは、更新または再インストール時には起動されません。スクリプトでは、製品のインストール パスおよびインストール ジョブの内部パラメータ (`INSTALL` または `UPDATE`) を編集できます。これらのパラメータ値によって、デフォルトのパラメータ設定が上書きされます。

インストーラの互換性についての考慮事項

以下の表に、特定のパッケージ バージョンでパッケージされた PIF 製品 (表の 1 列目) について、ターゲット コンピュータ上のインストーラ バージョンごとにインストールが可能かどうかを示します。また、特定のパッケージ バージョンでパッケージされた PIF 製品をインストールするために、ターゲット コンピュータ上で必要なインストーラのバージョンも示します。

表内でインストール可能とされた項目にアスタリスク（インストール可*）が付いている場合は、自己インストール PIF 製品によって、PIF 製品をインストールする前にターゲット コンピュータに適切なインストーラが自動的にインストールされるか、または（必要に応じて）既存のインストーラが更新されることを示します。

インストールする PIF 製品をパッケージするパッケージバージョン	ターゲット コンピュータ上のインストーラバージョン： なし	ターゲット コンピュータ上のインストーラバージョン： 3.1	ターゲット コンピュータ上のインストーラバージョン： 4.0	ターゲット コンピュータ上のインストーラバージョン： 4.1	ターゲット コンピュータ上のインストーラバージョン： 4.2	ターゲット コンピュータ上のインストーラバージョン： 4.3
3.1	インストール不可	インストール可	インストール可	インストール可	インストール可	インストール可
4.0	インストール不可	インストール失敗	インストール可	インストール可	インストール可	インストール可
4.1	インストール不可	インストール失敗	インストール失敗	インストール可	インストール可	インストール可
4.2	インストール不可	インストール失敗	インストール失敗	インストール失敗	インストール可	インストール可
4.3	インストール不可	インストール失敗	インストール失敗	インストール失敗	インストール失敗	インストール可
4.0（自己インストールとして）	インストール可*	インストール可*	インストール可*	インストール可	インストール可	インストール可
4.1（自己インストールとして）	インストール可*	インストール可*	インストール可*	インストール可*	インストール可	インストール可
4.2（自己インストールとして）	インストール可*	インストール可*	インストール可*	インストール可*	インストール可*	インストール可

インストー ルする PIF 製品をパッ ケージする パッケージ ジャバー ジョン	ターゲット コンピュー タ上のイン ストーラ バージョン： なし	ターゲット コンピュー タ上のイン ストーラ バージョン： 3.1	ターゲット コンピュー タ上のイン ストーラ バージョン： 4.0	ターゲット コンピュータ 上のインス トローバー ジョン：4.1	ターゲット コンピュー タ上のイン ストーラ バージョン： 4.2	ターゲット コンピュー タ上のイン ストーラ バージョン： 4.3
4.2（自己イ ンストール として）	インストー ル可 *	インストー ル可 *	インストー ル可 *	インストール 可 *	インストー ル可 *	インストー ル可 *

例：パッケージャバージョン 4.3 でパッケージされた PIF 製品は、インストーラバージョン 4.2 が稼動するターゲット コンピュータ上で正しくインストールすることはできません。

第 4 章：パッケージ ユーザ インター フェースでの作業

この章では、PIF パッケージ ユーザ インターフェースを使用して、重要なタスクを実行する方法について説明します。

このセクションには、以下のトピックが含まれています。

[新しい PIF パッケージの作成](#) (P. 83)

[PIF パッケージの保存](#) (P. 85)

[PIF パッケージのビルド](#) (P. 85)

[ライブラリまたはディスクへの PIF パッケージの登録](#) (P. 87)

[PIF パッケージの変更](#) (P. 88)

[PIF パッケージ用の暗号化方式の設定](#) (P. 89)

[PIF パッケージのアンロード](#) (P. 90)

[PIF パッケージの削除](#) (P. 90)

[PIF パッケージのインポート](#) (P. 91)

[多言語の PIF パッケージの作成](#) (P. 92)

[パッチ製品作成ウィザード](#) (P. 95)

新しい PIF パッケージの作成

PIF パッケージ GUI を使用して、新しい PIF パッケージを作成できます。新しい PIF パッケージの作成とは、製品定義に関するすべての情報が含まれている新しいプロトタイプ ファイルを作成することです。

新しい PIF パッケージを作成するには、以下の手順に従います。

1. パッケージ GUI で、以下のいずれかの操作を行います。
 - メニュー バーから [ファイル] - [新規作成] を選択します。
 - ツールバーの [新規] アイコンをクリックします。
 - プロジェクト ペイン内を右クリックして、コンテキスト メニューから [新規] を選択します。

[新規パッケージ] ダイアログ ボックスが表示されます。

注：PIF パッケージ作成ウィザードを使用して、新しい PIF パッケージを作成することもできます。ウィザード起動するには、ツールバーのウィザードアイコンをクリック（またはプロジェクトペインを右クリックし、コンテキストメニューから「新規パッケージウィザード」を選択）し、ウィザードにステップに表示される説明に従います。

2. 次のフィールドに、PIF パッケージに関する情報を入力します。

パッケージ名

PIF パッケージの名前を示します。パッケージ名には、最高 128 文字まで含めることができます。

パッケージのバージョン

PIF パッケージのバージョンを示します。バージョンの形式には M.m.b.r があり、「M」はメジャー、「m」はマイナー、「b」はビルド、「r」はリビジョンをそれぞれ表します。

パッケージプラットフォーム

PIF パッケージがインストール可能なハードウェアおよびオペレーティングシステムを指定する UNIX または Linux プラットフォームを示します。

注：「部分」のオプションを選択すると、サポートされているすべての UNIX プラットフォームおよび Linux プラットフォームにパッケージがインストールされます。

テンプレート

個別のパッケージセットアップの作成に必要なフレームワークとして使用するテンプレートを示します。テンプレートは、プロトタイプファイルとして表示されます。要件に基づいて次のテンプレートのいずれかを選択します。

- standard.Any.@enu
- advanced.Any.@enu
- unattended.Any.@enu
- multi_language.Any.@enu
- multi_instance.Any.@enu

3. 「新規」をクリックします。

新しいパッケージが作成され、プロジェクトペインに表示されます。

PIF パッケージの保存

新しい PIF パッケージを作成した後、ビルドを開始する前に PIF パッケージを保存する必要があります。

PIF パッケージを保存するには、以下の手順に従います。

1. 保存するパッケージをプロジェクト ペインで選択します。
2. パッケージャ GUI で、以下のいずれかの操作を行います。
 - メニュー バーから [ファイル] - [保存] を選択します。
 - ツールバーの [保存] アイコンをクリックします。
 - パッケージを右クリックし、コンテキスト メニューから [保存] を選択します。

パッケージが保存されます。

PIF パッケージのビルド

PIF パッケージの設定が完了したら、それを保存してから PIF パッケージをビルドする必要があります。

PIF パッケージをビルドするには、以下の手順に従います。

1. プロジェクト ペインでパッケージを選択します。
2. 以下のいずれかの操作を実行します。
 - 選択したエントリを右クリックし、コンテキスト メニューから [ビルド] を選択します。
 - メニュー バーから [ファイル] - [ビルド] を選択します。
 - ツールバーの [ビルド] アイコンをクリックします。[製品のビルド] ダイアログ ボックスが表示されます。

3. 以下のビルド オプションを指定できます。

自動インストール

このチェック ボックスがオンの場合、自動インストール PIF パッケージ（インストーラを含む）が自動的にビルドされることを示します。ドロップダウン リストが表示され、パッケージプラットフォームが選択できます。以下のオプションが表示されます。

- インストーラと製品を分離：このチェック ボックスをオンにし、PIF とインストーラが分離されるように定義します。
- SDK の選択：このドロップダウン リストから該当するオプションを選択します。
- JRE ディレクトリ：パッケージに追加されるオプションの JRE を指定します。この JRE は自己インストール PIF パッケージに組み込まれています。その結果、ターゲット コンピュータに JRE がインストールされていない場合でも、パッケージインストールの間、グラフィカル ダイアログ ボックスはターゲット コンピュータのグラフィカル端末に表示できます。

削除

このチェック ボックスがオンの場合、PIF パッケージのすべてのバイナリ ファイルから、すべてのシンボル情報が削除されることを示します。これにより、PIF パッケージのサイズが小さくなります。

既存製品の上書き

すでにパッケージが存在する場合は、ダイアログ ボックスにこのチェック ボックスが表示されることを示します。このチェック ボックスはデフォルトでオンになっており、同名の既存の PIF パッケージが上書きされます。

4. 「ビルド」をクリックします。

[保存] ダイアログ ボックスは、PIF パッケージの保存場所を指定できる場所に表示されます。デフォルトでは PIF パッケージは、`$SDPcker/productlib` によって参照されるディレクトリに保存されます。

PIF パッケージが正常にビルドされると、対応する製品のエントリが [メディア] タブに表示されます。

ライブラリまたはディスクへの PIF パッケージの登録

PIF パッケージが正常にビルドされた後、マネージャ システムのソフトウェア パッケージ ライブラリ（「ライブラリへの登録」）、またはローカル ディスクのフォルダ（「ディスクへの登録」）にパッケージを登録する必要があります。

マネージャ システムで PIF パッケージを登録する方法（「ライブラリへの登録」）は次のとおりです。

1. プロジェクト ペインでパッケージを選択します。
2. サブプロジェクト ペインの [メディア] タブでパッケージの PIF ファイル (*.@pif) を選択します。
3. [メディア] タブの選択したエントリを右クリックし、コンテキスト メニューから [登録] - [ライブラリへ] を選択します。
[ライブラリに登録] ダイアログ ボックスが開きます。

注：ダイアログ ボックスを展開するには >> ボタンをクリックします。折りたたむには << ボタンをクリックします。

4. 以下のプロパティを入力します。

マネージャ

ソフトウェア パッケージ ライブラリがあるマネージャ システムのノード名を示します。使用されているすべてのマネージャがリストされます。

ユーザ

製品の登録に使用する UNIX ユーザの名前を示します。このユーザは、ソフトウェア パッケージ ライブラリにパッケージを登録する権限を持っている必要があります。

パスワード

ユーザ固有のパスワードを示します。

セキュリティ プロバイダ、ドメイン

ドメイン コントローラが使用されている場合のドメイン名を示します。有効なマネージャ名（[マネージャ] ドロップダウン リスト）に基づいて、すべてのセキュリティ プロバイダがスキャンされ、当てはまるセキュリティ プロバイダがドロップダウン リストに追加されます。

応答ファイル

(オプション) パッケージのインストールを設定するためのパラメータ設定を含む 1 つまたは複数のファイルを示します。ここに、1 つまたは複数の応答ファイル名を入力することができます。複数の応答ファイル名を入力する場合は、`respf1,respf2,respf3` のように、スペースを入れずにカンマで区切る必要があります。

5. [登録] をクリックします。

登録処理が開始します。

ローカル ディスクで PIF パッケージを登録する (「ディスクへの登録」) には、以下の手順に従います。

1. プロジェクト ペインでパッケージを選択します。
2. サブプロジェクト ペインの [メディア] タブでパッケージの PIF ファイル (*.@pif) を選択します。
3. [メディア] タブの選択したエントリを右クリックし、コンテキストメニューから [登録] - [ディスクへ] を選択します。

[ディスクへの登録] ダイアログ ボックスが表示されます。

4. [登録] をクリックします。

ディスク上の現在の製品フォルダに **reginfo** 構造が作成されます。この **reginfo** 構造はディスクに書き込まれ、**Software Delivery** メソッド (ドラッグアンドドロップ機能を含む) を使用して、ソフトウェア パッケージ ライブラリに配布され、登録されます。

PIF パッケージの変更

パッケージの作成後、パッケージを更新して変更できます。新しい要素を既存のダイアログ ボックスに追加する場合など、後で変更を組み込むために新しいパッケージを作成する必要はありません。また、これによりパッケージを最新の状態に保つことができます。

次の例では、ダイアログ ボックスに新しい要素を追加することにより、すでに作成されているパッケージを変更する方法について説明します。

既存のダイアログ ボックスに新しい要素を追加するには、以下の手順に従います。

1. プロジェクト ペインから、パッケージを選択します。
2. サブプロジェクト ペインの [ダイアログ ボックス] タブを選択します。
パッケージに含まれているすべてのダイアログ ボックスのリストが表示されます。
3. 新しい要素を追加するダイアログ ボックスを選択します。
右ペインに、（既存の要素も含む）ダイアログ ボックスが表示されます。
4. ダイアログ エディタのツールバーから新しい要素を選択し、それをダイアログ ボックスの適切な場所にドロップします。
既存のダイアログ ボックスに新しい要素が追加され、パッケージが変更されます。

PIF パッケージ用の暗号化方式の設定

PIF パッケージにパスワードが含まれていて、自動モードでインストールされる場合は、パスワードの暗号化および復号化に使用する方式およびアルゴリズムを指定する必要があります。

注：常に対話形式でインストールされるパッケージでは、パスワードがマスクされるので、パスワードの暗号化は必要ありません。

PIF パッケージ用の暗号化方式の設定方法

1. [プロジェクト] ペインからパッケージを選択します。
右ペインには、パッケージの特性が表示されます。
2. [セキュリティ] タブをクリックします。

注：このタブが有効になるのは、パッケージにパスワードが含まれている場合のみです。

3. 組み込みまたは外部暗号化方式を選択し、パスワードの暗号化および復号化に使用するコマンドを指定します。組み込み暗号化方式は、Blowfish 暗号化アルゴリズムを使用しているため、FIPS に準拠していません。FIPS 準拠の暗号化を使用するには、FIPS に準拠している外部暗号化方式を使用する必要があります。暗号化方式およびコマンドの詳細については、「*Software Packager for Linux and UNIX ヘルプ*」を参照してください。

`ls -a` コマンドを使用してパッケージの応答ファイルを作成する際に、パスワードは選択した方式およびアルゴリズムを使用して暗号化されます。

4. [OK] をクリックします。
暗号化方式が設定されます。

PIF パッケージのアンロード

PIF パッケージをプロジェクト ペインからアンロード（削除）できますが、ディスクからはパッケージは削除されません。アンロードしても、パッケージは製品ライブラリ（`productlib`）に含まれたままであり、ここからパッケージを再度インポートできます。同時に 1 つ以上のパッケージをアンロードできます。

PIF パッケージをアンロードするには、以下の手順に従います。

1. プロジェクト ペインでパッケージを選択します。
2. 選択したパッケージを右クリックし、コンテキスト メニューから [アンロード] を選択します。

選択したパッケージをアンロードするかどうかを確認するダイアログ ボックスが表示されます。

3. [はい] をクリックします。
選択したパッケージがプロジェクト ペインから削除されます。

PIF パッケージの削除

PIF パッケージが必要ない場合は削除できます。

PIF パッケージを削除するには、以下の手順に従います。

1. 削除するパッケージをプロジェクト ペインで選択します。
2. パッケージ GUI で、以下のいずれかの操作を行います。
 - メニュー バーから [ファイル] - [削除] を選択します。

- ツールバーの [削除] アイコンをクリックします。
- パッケージを右クリックし、コンテキストメニューから [削除] を選択します。

関連するすべてのパッケージプロトタイプおよび PIF ファイルがシステムから削除され、パッケージをリストアすることはできないというメッセージが表示されます。

3. [はい] をクリックします。

選択したパッケージが削除されます。

PIF パッケージのインポート

PIF パッケージのインポートとは、すでに作成されている PIF パッケージ (xxx@pif) または PIF プロトタイプ ファイル (xxx@prm) をインポートすることです。

すでに作成されている PIF パッケージをインポートする場合、インポートされた PIF パッケージを抽出するためのターゲットディレクトリを指定する必要があります。

インポートした PIF パッケージの名前がパッケージ GUI のプロジェクトペインに表示されるので、インポートした PIF 製品を変更できます。

プロトタイプ ファイルをインポートするには、以下の手順に従います。

1. [ファイル] メニューから [インポート] を選択するか、またはプロジェクトペインを右クリックしてコンテキストメニューから [インポート] を選択します。

[ファイルの選択] ダイアログボックスが表示されます。

2. 1つまたは複数のプロトタイプファイル (*.@prm) を参照するか、またはすべてのタイプを検索します。

または、[ファイル名] 入力フィールドでプロトタイプファイルの名前を指定することもできます。

注： インポートには有効な PIF プロトタイプファイルのみを使用できます。

3. [開く] をクリックします。

インポートされたパッケージがプロジェクトペインに表示されます。

既存の PIF パッケージをインポートするには、以下の手順に従います。

1. [ファイル] メニューから [インポート] を選択するか、またはプロジェクト ペインを右クリックしてコンテキスト メニューから [インポート] を選択します。
[ファイルの選択] ダイアログ ボックスが表示されます。
2. PIF パッケージ (*.@pif) を参照します。
または、[ファイル名] 入力フィールドで PIF パッケージの名前を指定します。
注：インポートに使用できるのは有効な PIF パッケージのみです。
3. [開く] をクリックします。
[パッケージのインポート] ダイアログ ボックスが表示されます。
4. インポートした PIF パッケージを抽出するフォルダの場所を指定します。
5. [インポート] をクリックします。
インポートした PIF パッケージの名前がパッケージ GUI のプロジェクト ペインに表示されるので、インポートした PIF パッケージを変更できます。

多言語の PIF パッケージの作成

multi_language.Any.@pif テンプレートを使用すると、多言語の PIF パッケージを作成できます。多言語の PIF パッケージを使用すると、製品のインストールで使用する言語を指定することができます。また、言語固有のコンポーネントをインストール用に選択（または選択解除）することができます。

多言語の PIF パッケージを作成するには、以下の手順に従います。

1. パッケージ GUI で、以下のいずれかの操作を行います。
 - メニュー バーから [ファイル] - [新規作成] を選択します。
 - ツールバーの [新規] アイコンをクリックします。
 - プロジェクト ペイン内を右クリックして、コンテキスト メニューから [新規] を選択します。
[新規パッケージ] ダイアログ ボックスが表示されます。
2. 次のフィールドに、PIF パッケージに関する情報を入力します。

パッケージ名

PIF パッケージの名前を示します。パッケージ名には、最高 128 文字まで含めることができます。

パッケージのバージョン

PIF パッケージのバージョンを示します。バージョンの形式には M.m.b.r があり、「M」はメジャー、「m」はマイナー、「b」はビルド、「r」はリビジョンをそれぞれ表します。

パッケージプラットフォーム

PIF パッケージがインストール可能なハードウェアおよびオペレーティングシステムを指定する UNIX または Linux プラットフォームを示します。

注：[部分] のオプションを選択すると、サポートされているすべての UNIX プラットフォームおよび Linux プラットフォームにパッケージがインストールされます。

テンプレート

個別のパッケージセットアップの作成に必要なフレームワークとして使用するテンプレートを示します。テンプレートは、プロトタイプファイルとして表示されます。ドロップダウンリストから [multi_language.Any.@enu] を選択します。

3. [新規] をクリックします。

新しいパッケージが作成され、プロジェクト ペインに表示されます。

インストール ダイアログ ボックスの言語を選択します。

多言語パッケージでは、インストール ダイアログ ボックスの言語を選択したり、選択した言語でダイアログ ボックスをプレビューしたりできます。

[インストール] ダイアログ ボックスの言語を選択するには、以下の手順に従います。

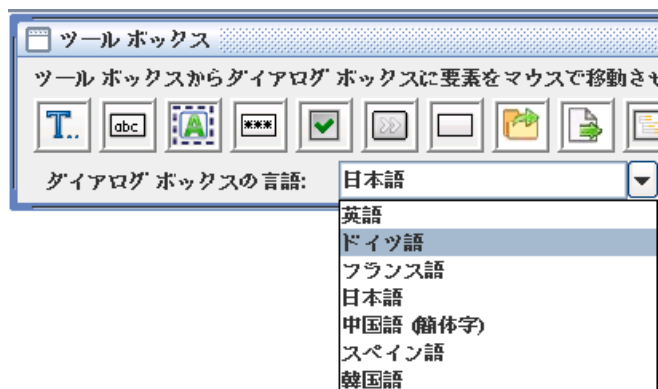
1. [プロジェクト] タブから多言語パッケージを選択します。
2. [ダイアログ ボックス] タブを選択します。

ダイアログ ボックスのリストが表示されます。

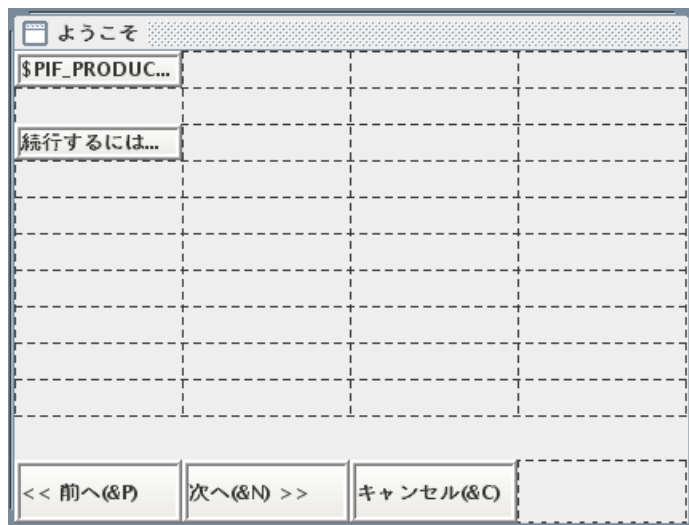
3. [ダイアログ ボックス] を選択します。

右側のペインに、ダイアログ エディタが表示されます。

4. 以下の図で示すように、[ダイアログ ボックスの言語] ドロップダウン リストから、適切な言語を選択します。



以下の図で示すように、すべてのインストール ダイアログ ボックスの言語が変更されます。



ダイアログ ボックス リソースのコピーおよび変更

多言語環境で異なる言語のテキストを使用する場合は、既存のダイアログ ボックス リソースをコピーして変更すると便利です。

ダイアログ ボックス リソースをコピーするには、以下の手順に従います。

1. プロジェクト ペインでパッケージを選択します。
2. サブプロジェクト ペインにある [ダイアログ ボックス リソース] タブの言語 エントリのいずれかを選択します。
3. 選択したリソースを右クリックし、コンテキスト メニューから [コピー] を選択します。

[リソース] ダイアログ ボックスが表示されます。

4. ドロップダウン リストから新しい言語を選択します。
5. [コピー] をクリックします。

サブプロジェクト ペインにある [ダイアログ ボックス リソース] タブに、新しいダイアログ ボックス リソースが表示されます。

ダイアログ ボックス リソースでテキストを変更するには、以下の手順に従います。

1. プロジェクト ペインでパッケージを選択します。
2. サブプロジェクト ペインの [ダイアログ ボックス リソース] タブで、変更するリソースを選択します。
3. GUI 作業ペインの [ダイアログ ボックスのテキスト] 表で、変更するエントリを選択します。
4. 選択したエントリを右クリックし、コンテキスト メニューから [変更] を選択します。

[テキストの追加/変更] ダイアログ ボックスが表示されます。

5. [テキスト] 入力ボックスでテキストを編集します。
6. [追加] をクリックします。

変更したテキストがダイアログ ボックス リソースに追加されます。

パッチ製品作成ウィザード

パッチ製品作成ウィザードを使用すると、既存の PIF 製品を更新するためのパッチを作成することができます。PIF 製品のパッチは、「通常の」PIF 製品と同じように、作成、ビルド、配信、およびインストールされます。

パッチ製品作成ウィザードで PIF パッチ製品を作成するには、以下の手順に従います。

1. プロジェクト ペイン内の製品を右クリックして、コンテキスト メニューから [新規パッチ] を選択します。

新規パッチ製品作成ウィザードが起動し、[パッチ名] ページが表示されます。

2. パッチ名およびバージョンに関する情報を適切なフィールド入力し、[次へ] をクリックします。

注： [パッチ名] ページには、パッチを作成している製品の名前およびバージョンに関する情報も表示されます。

[パッチ製品生成の選択] ページが表示されます。

3. [新規パッチ] をクリックします。

[ソース フォルダ] ページが表示されます。

4. パッチ製品の一部である、変更されたすべてのファイルが含まれているディレクトリの場所と名前を入力します。 [次へ] をクリックします。

[置換されるパッチ] ページが表示されます。

5. このパッチによって置き換えられる、以前割り当てられたすべてのパッチの名前を入力します。 [次へ] をクリックします。

注： 名前を区切るには、カンマ (,) を使用します。

[サマリ] ページが表示されます。

6. 製品のパッチを作成するには、[パッチの作成] をクリックします。

パッケージ GUI のプロジェクト ペインに PIF パッチ製品が表示されます。これは、プロジェクト ペインに一覧表示されているその他の「通常の」PIF 製品と同様に変更できます。

注： パッケージ GUI のタブには、PIF パッチ製品には適用されないためグレー表示されるものもあります。

第 5 章： サンプル PIF 製品のビルド

PIF 製品の定義とビルド方法、およびインストールの設定に従ったサンプルを示します。 PIF 製品は、プロトタイプの作成から始まるステップごとに作成され、「Linux および UNIX のソフトウェア管理」で説明されている多くの機能が適用されています。

製品、製品ファイルやディレクトリ、依存関係などを記述するプロトタイプ ファイルのエントリが記述され、仕様を入力する場所とその形式を確認できます。 記述順序は、プロトタイプ ファイルに仕様を入力する順序に従っています。

プロトタイプ ファイルの仕様を入力（および修正）するには、テキスト エディタを使用できます。 以下の記述では、この方法を前提としています。 また、製品作成のステップごとに、コマンドラインとパッケージャ GUI を使用して同じ仕様を入力する方法が記載されています。

使用するコマンドおよびプロトタイプ ファイルのエントリ（キーワード、プロパティ）については、「[キーワードおよびプロパティ \(P. 132\)](#)」を参照してください。 PIF 製品 MERCHANT の作成にパッケージャ GUI を使用する場合は、パッケージャのオンライン ヘルプも参照してください。

このセクションには、以下のトピックが含まれています。

[UNIX アプリケーション MERCHANT の要件および仕様 \(P. 98\)](#)

[PIF 製品 MERCHANT のプロトタイプ ファイルの作成および製品情報の指定 \(P. 98\)](#)

[PIF 製品 MERCHANT の依存関係の定義 \(P. 100\)](#)

[PIF 製品 MERCHANT のファイルの指定 \(P. 101\)](#)

[PIF 製品 MERCHANT へのプロシージャの追加 \(P. 103\)](#)

[PIF 製品 MERCHANT のインストールパスの変更 \(P. 106\)](#)

[PIF 製品 MERCHANT のインストールパスを問い合わせるダイアログ ボックスの追加 \(P. 107\)](#)

[PIF 製品 MERCHANT への言語固有の製品ファイルの追加 \(P. 109\)](#)

[PIF 製品 MERCHANT のインストール ダイアログ ボックスのローカライズ \(P. 111\)](#)

[最終的な PIF 製品 MERCHANT のビルド \(P. 113\)](#)

UNIX アプリケーション MERCHANT の要件および仕様

ここでは、アプリケーション MERCHANT バージョン 2.1.0.0 を PIF 製品にパッケージします。アプリケーションおよびその機能に関する詳細は省略しますが、アプリケーションの要件は以下のとおりです。

- アプリケーションは、すべての UNIX または Linux 動作環境で実行されます。
- Java アーカイブ ファイルとして提供される、Java GUI で構成されるアプリケーションは、`merchant.jar` と呼ばれます。シェル スクリプト `run_merchant.sh` によって GUI が起動します。このシェル スクリプトは、`/usr/bin` ディレクトリにリンクされています。
- アプリケーションデータは、すべて Ingres Enterprise Relational Database (Ingres) に格納されています。
- データベース テーブルは、製品のインストール後に作成されます。これは、ユーザ作成のスクリプト `create_tables.sh` で処理されます。
- ユーザがインストール ディレクトリを選択できるようにする必要があります。デフォルト パスは、値 `/home/merchant` が事前に設定されます。
- インストール ディレクトリを決めるために、インストール ダイアログ ボックスが使用されます。
- 共有ライブラリ `libi8n.so` では、ローカライズされたメッセージをプリントするためのインターフェースが提供されています。これは、一度インストールされると、さまざまな製品によって使用されます。
- 製品には、テキスト ファイル `merchant.enu` (英語メッセージ) および `merchant.deu` (ドイツ語メッセージ) で提供されているローカライズされたメッセージが表示されます。
- インストール ダイアログ ボックスでは、英語またはドイツ語のダイアログ ボックスが表示されます。

PIF 製品 MERCHANT のプロトタイプ ファイルの作成および製品情報の指定

PIF 製品 MERCHANT をビルドする最初のステップとして、完全な製品定義を含むプロトタイプ ファイルを作成します。プロトタイプ ファイルの命名規則とアプリケーションの要件に従うと、サンプル PIF 製品 MERCHANT のプロトタイプ ファイルの名前は以下のようになります。

`MERCHANT.Any.@prm`

プロトタイプ ファイルを作成するには、現在の作業ディレクトリでテキスト エディタを開き、この名前の（空の）ファイルを保存します。

製品情報のセクションを以下のように入力します。

```
@PRODUCT:
#phead: MERCHANT , 2.1.0.0 ;
#sys: Any ;
#locale: JPN ;
#comment: これはサンプルの PIF 製品です ;
#ppath: /home/merchant ;
@ENDPROD:
```

製品情報セクションでは、以下の製品プロパティが定義されます。

```
#phead: MERCHANT , 2.1.0.0 ;
```

PIF 製品名 MERCHANT および PIF 製品バージョン 2.1.0.0 を定義します。

```
#sys: Any ;
```

PIF 製品のインストールが可能な動作環境を定義します。この場合、Any は、PIF 製品 MERCHANT をすべての UNIX/Linux 動作環境にインストール可能であることを示しています。

```
#locale: JPN ;
```

PIF 製品 MERCHANT のデフォルトのロケールを定義します。デフォルト設定として、文字列 JPN（日本語）を使用します。

```
#comment: これはサンプルの PIF 製品です ;
```

キーワード #comment の後ろに、製品を説明するテキストを入力できます。

```
#ppath: /home/merchant ;
```

PIF 製品 MERCHANT は、ターゲット コンピュータの /home/merchant ディレクトリにインストールされます。このディレクトリは、製品のインストール時に自動的に作成されます。

製品情報を含む PIF 製品 MERCHANT のプロトタイプ ファイルの作成をコマンドラインから行うには、以下のように pifproto コマンドを実行します。

```
pifproto -b /home/merchant -a Any -n MERCHANT -v 2.1.0.0
```

パッケージ GUI を使用して製品情報を含む PIF 製品 MERCHANT のプロトタイプ ファイルを作成するには、以下の手順に従います。

1. smgui コマンドを実行し、パッケージ GUI を起動します。
2. メニュー バーから [ファイル] - [新規作成] を選択します。
[新規製品] ダイアログが表示されます。
3. 製品名 MERCHANT、バージョン 2.1.0.0、およびプラットフォーム ID Any を入力します。

PIF 製品 MERCHANT の依存関係の定義

要件に指定されているように、アプリケーション MERCHANT は、Ingres Enterprise Relational Database (Ingres) を使用するデータベース アプリケーションです。このため、PIF 製品 MERCHANT をインストールする前に、Ingres データベースをインストールする必要があります。

システムに適切な Ingres データベースが存在する場合にのみ製品 MERCHANT をインストールするため、プロトタイプ ファイルに依存関係を定義する必要があります。製品情報セクションにハイライトされた新しい行では、以下のように依存関係および必要な製品を指定します。

```
@PRODUCT:
#phead: MERCHANT , 2.1.0.0 ;
#sys: Any ;
#locale: JPN ;
#comment: これはサンプルの PIF 製品です ;
#ppath: /home/merchant ;

#pdep: Ingres , 4.0.0.0 , >= ;
@ENDPROD:

#pdep: Ingres , 4.0.0.0 , >= ;
```

必要な製品の名前は、Ingres です。バージョン番号 4.0.0.0 と後続の演算子 >= を組み合わせて、必要な Ingres 製品バージョンが 4.0.0.0 以上であることを指定します。インストーラでは、製品 MERCHANT をインストールする前に、ターゲット コンピュータに適切な Ingres のバージョンがインストールされているかどうかチェックされます。

コマンド行から依存関係を定義するには、テキスト エディタを使用して、プロトタイプ ファイルに新しく #pdep: 行を追加します。

パッケージ GUI での依存関係を定義するには、以下の手順に従います。

1. smgui コマンドを実行し、パッケージ GUI を起動します。
2. プロジェクト ペインで、PIF 製品 MERCHANT を選択します。
3. [依存関係] タブを選択します。
4. サブプロジェクト ペイン内を右クリックして、コンテキスト メニューから [追加] を選択します。
[製品の依存関係] ダイアログ ボックスが表示されます。

5. 以下の値を入力または選択します。

製品名 : Ingres

製品バージョン : 4.0.0.0

依存関係のタイプ : 必須

演算子 : >=

6. [追加] をクリックします。

依存関係の仕様がプロトタイプ ファイルに組み込まれます。

PIF 製品 MERCHANT のファイルの指定

最終的な PIF 製品を構成するファイルは、すべてプロトタイプ ファイルのコンポーネント セクションで定義されます。PIF 製品 MERCHANT の場合は、プロトタイプ ファイル MERCHANT.Any.@prm に以下のセクションを追加します。

```
@COMPONENT:
#thead: component1 ;
#dir: bin , root , sys , drwxrwxrwx ;
#file: bin/merchant.jar , /myhomepath/src/merchant.jar, root , sys , -r-r-r-- , 1 ;
#file: bin/run_merchant.sh , /myhomepath/scripts/run_merchant.sh, root , sys ,
-r-xr-xr-x , 1 ;
#slink: /usr/bin/run_merchant.sh , bin/run_merchant.sh ;
@ENDCOMP:
```

製品情報セクションでは、以下の製品プロパティが定義されます。

```
#thead: component1 ;
```

PIF 製品 MERCHANT の固有名（ここでは component1）を定義します。

```
#dir: bin , root , sys , drwxrwxrwx ;
```

PIF 製品 MERCHANT に bin ディレクトリを追加します。ユーザ root は、このディレクトリの所有者です。これは、グループ sys に割り当てられます。アクセス権は、everybody に対して「read-write-execute (rwx)」となります。

```
#file: bin/merchant.jar , /myhomepath/src/merchant.jar , root , sys , -r---r---r--- , 1 ;
```

```
#file: bin/run_merchant.sh , /myhomepath/scripts/run_merchant.sh , root , sys , -r-xr-xr-x , 1 ;
```

PIF 製品 MERCHANT の製品ファイルを定義します。最初のプロパティ値は、ファイルのターゲット名です。ターゲット ファイルには、ディレクトリ構造も追加する必要があります。この場合、各ファイルは bin ディレクトリにインストールされます。次のプロパティは、ソース ファイル名を指定します。これは、PIF 製品をビルドするときにファイルが置かれるパスです。ターゲットの所有者、グループ、およびアクセス権がこれに続きます。最後のプロパティ値 (1) では、ファイル タイプに標準ファイルを指定します。

注： ターゲット ファイル名の先頭にスラッシュを付けずに入力した場合、ファイルは製品のベース ディレクトリの相対パスにインストールされます。スラッシュ付けた場合、ファイルは絶対パス名を使用してインストールされます。

```
#slink: /usr/bin/run_merchant.sh , bin/run_merchant.sh ;
```

/usr/bin ディレクトリにシンボリック リンクを作成します。2 番目のパラメータは、ソース ファイルのパス名です。

PIF 製品のファイルの指定は、コマンドラインから pifproto コマンドを使用して自動化することができます。pifproto コマンドは、プロトタイプ ファイルおよび製品情報セクションを作成する以外に、製品ディレクトリに格納されているすべてのファイルのパスおよびプロパティをプロトタイプ ファイルに追加します。

コマンドラインからファイルを指定するには、以下の手順に従います。

1. 新しいフォルダ構造を作成します (例、/tmp/merchant および /tmp/merchant/bin)。
2. 追加するファイルを /tmp/merchant/bin フォルダにコピーします。
3. 以下の pifproto コマンドを実行します。

```
pifproto -n MERCHANT -v 2.1.0.0 -a Any -d /tmp/merchant -b /home/merchant
```

プロトタイプ ファイル MERCHANT.Any.@prm は、現在の作業ディレクトリに置かれます。

パッケージ GUI でファイルを指定するには、以下の手順に従います。

1. smgui コマンドを実行し、パッケージ GUI を起動します。
2. メニューバーから [ファイル] - [新規作成] を選択します。

[新規製品] ダイアログが表示されます。

3. 以下の値を入力します。

製品名 : MERCHANT

製品バージョン : 2.1.0.0

製品プラットフォーム : Any

テンプレート : standard.Any.@enu

4. [新規] をクリックします。

プロジェクトペインに製品 MERCHANT が表示されてハイライトされます。

GUI の作業領域に、現在の製品プロパティが表示されます。インストールディレクトリには、値 `/opt/merchant` があらかじめ設定されています。名前が `component1` の標準コンポーネントおよびプリインストールコンポーネントが自動的に作成され、サブプロジェクトペインの [コンポーネント] タブに表示されます。

5. プロジェクトペインで、製品 MERCHANT を選択します。
6. メニューバーから [ファイル] - [保存] を選択します。

現在のプロトタイプファイルが保存されます。

PIF 製品 MERCHANT へのプロシージャの追加

PIF 製品 MERCHANT では、この製品のファイルをインストールした後に、データベーステーブルを作成する必要があります。この処理は、製品 MERCHANT の仕様により、ユーザが作成するスクリプトファイル `create_tables.sh` で行われます。

コンポーネント セクションで指定されるスクリプト ファイルのプロパティは、以下のとおりです。

```
@COMPONENT:
#chead: component1 ;
#dir: bin , root , sys , drwxrwxrwx ;
#file: bin/merchant.jar , /myhomepath/src/merchant.jar, root , sys , -r-r-r-- , 1 ;
#file: bin/run_merchant.sh , /myhomepath/scripts/run_merchant.sh, root , sys ,
-r-xr-xr-x , 1 ;
#slink: /usr/bin/run_merchant.sh , bin/run_merchant.sh ;
#file: bin/create_tables.sh , /myhomepath/scripts/create_tables.sh , root , sys ,
-r-xr-xr-x , 1 ;
@ENDCOMP:

#file: bin/create_tables.sh , /myhomepath/scripts/create_tables.sh , root , sys ,
-r-xr-xr-x , 1 ;
```

スクリプト ファイル `create_tables.sh` は、ターゲット ディレクトリ `bin` にインストールする必要があります。スクリプト ファイルのソース ディレクトリは `/myhomepath/scripts` です。所有者は `root` で、グループは `sys` です。読み取り (`r`) および実行 (`x`) のアクセス権限がすべてのユーザに与えられています。このファイル タイプは、`1` (標準ファイル) です。

製品情報セクションで指定されるスクリプト実行時間は、以下のとおりです。

```
@PRODUCT:
#phead: MERCHANT , 2.1.0.0 ;
#sys: Any ;
#locale: JPN ;
#comment: これはサンプルの PIF 製品です ;
#ppath: /home/merchant ;
#pdep: Ingres , 4.0.0.0 , >= ;
#postinit: bin/create_tables.sh ;
@ENDPROD:

#postinit: bin/create_tables.sh ;
```

製品ファイルをインストールした後に、キーワード `#postinit:` の指定に従ってスクリプト `script create_table.sh` が実行されます。

コマンドラインでスクリプトファイル `create_table.sh` を指定するには、以下の手順に従います。

1. スクリプト ファイル `create_tables.sh` を `/tmp/merchant/scripts` フォルダにコピーします。
2. 以下の `pifproto` コマンドを実行します。

```
pifproto -n MERCHANT -v 2.1.0.0 -a Any -d /tmp/merchant -b /home/merchant -i scripts/create_tables.sh -o
```

`-i` オプションを使用して、インストール後のプロシージャを定義します。`-o` スイッチを使用すると、製品 **MERCHANT** の既存のプロトタイプ ファイルを上書きすることができます。

GUI を使用してプロシージャを追加するには、以下の手順に従います。

1. `smgui` コマンドを実行し、パッケージ GUI を起動します。
2. プロジェクト ペインで、製品 **MERCHANT** を選択します。
3. [コンポーネント] タブで `component1` を選択します。
4. [ファイル] リスト プロパティ シートを選択します。
5. [ファイル] リスト プロパティ シートを右クリックして、コンテキスト メニューから [挿入] を選択します。
[ソース ファイルの選択] ダイアログ ボックスが表示されます。
6. ファイル `create_tables.sh` を参照して、[選択] をクリックします。
7. ファイル リストで新しいファイルをダブルクリックします。表示されたダイアログ ボックスで [アクセス] プロパティ シートを選択し、実行チェック ボックスをすべてオンに、書き込みチェック ボックスをオフにして、[変更] をクリックします。
8. プロジェクト ペインで、製品 **MERCHANT** を選択します。
9. 作業ペインの [プロシージャ] タブを選択します。
10. プロシージャ リストを右クリックして、コンテキスト メニューから [内部プロシージャの追加] を選択します。
[内部プロシージャ] ダイアログ ボックスが表示されます。
11. 以下の値を指定または選択します。
実行時間：インストール後
プロシージャ ファイル： `create_tables.sh`
12. [追加] をクリックします。
プロシージャ リストにプロシージャが表示され、PIF 製品 **MERCHANT** のプロトタイプ ファイルにプロパティが組み込まれます。

PIF 製品 MERCHANT のインストールパスの変更

以前、PIF 製品 MERCHANT のインストールパスは固定で、インストール時に変更することができませんでした。現在は、パラメータ `$PATHmerchant` を定義すると、インストールパスを変更できます。

`$PATHmerchant` は、以下のように、プロトタイプ ファイルのパラメータ セクション（デフォルト セクション）で定義できます。

```
@DEFAULT:
#parameter: $PATHmerchant , /home/merchant ;
@ENDDFAULT:

#parameter: $PATHmerchant , /home/merchant ;
```

パラメータ名は `$PATHmerchant` で、デフォルト値は `/home/merchant` に設定されます。

以下のようにして、`#ppath` キーワードのインストールパスの絶対値 `/home/merchant` をプロトタイプ ファイルの製品情報セクションにあるパラメータ `$PATHmerchant` で置き換えます。

```
@PRODUCT:
#phead: MERCHANT , 2.1.0.0 ;
#sys: Any ;
#locale: JPN ;
#comment: これはサンプルの PIF 製品です ;
#ppath: $PATHmerchant ;
#pdep: Ingres , 4.0.0.0 , >= ;
#postinit: bin/create_tables.sh ;
@ENDPROD:
```

ここで、インストールパスを変更できます。

変更する方法の 1 つは、シェルパラメータを使用してデフォルト設定を上書きする方法で、以下に例を示します。

```
PATHmerchant=/opt/merchant; export PATHmerchant
```

もう 1 つの方法は、応答ファイルを使用して製品のインストールを再度設定する方法です。

コマンドライン インターフェイスおよびパッケージ GUI によって、PIF 製品に実際のインストールパスが自動的に追加します。アクションは必要ありません。

PIF 製品 MERCHANT のインストール パスを問い合わせるダイアログ ボックスの追加

プロトタイプファイルのダイアログ ボックス定義は、ダイアログ ボックスセクションとリソースセクションの 2 つの主要部分で構成されています。ダイアログ ボックスセクションでは、ダイアログ ボックスのレイアウトを定義します。リソースセクションには、ダイアログ ボックスの表示に必要なすべての文字列が格納され、その文字列には番号が付けられています。これらのテキスト番号のみがダイアログ ボックス定義に使用されます。

PIF 製品 MERCHANT の例では、インストールディレクトリを入力するダイアログ ボックスを作成する場合を考えます。このダイアログ ボックスは、製品 MERCHANT をインストールしたときに表示される最初のダイアログ ボックスです。

作成するダイアログ ボックスのダイアログ ボックスセクションの定義は、以下のとおりです。

```
@DIALOG:
#dhead: 600 , 400 , dlgInstallationDirectory , 1 ;
#label: 1 , 1 , lblInstallDirectory , 2 ;
#instdir: 1 , 2 , tfInstallationDirectory , $PATHmerchant ;
#navbutton: 1 , btnInstall , 3 , 1 ;
#navbutton: 2 , btnCancel , 4 , 0 ;
@ENDDIALOG:

#dhead: 600 , 400 , dlgInstallationDirectory , 1 ;
```

ダイアログ ボックスの幅は 600 ピクセルで、高さは 400 ピクセルです。ダイアログ ボックス名は `dlgInstallationDirectory` で、プロトタイプファイルのリソースセクションのテキスト番号 1 がダイアログ ボックスヘッダとして表示されます。

```
#label: 1 , 1 , lblInstallDirectory , 2 ;
```

インストールディレクトリの入力フィールドのラベルを定義します。ラベル名は `lblInstallationDirectory` で、リソースセクションのテキスト番号 2 が割り当てられます。ダイアログ ボックスに重なる仮想グリッドのラベル位置は、論理行 1、列 1 です。

```
#instdir: 1 , 2 , tflInstallationDirectory , $PATHmerchant ;
```

インストール ディレクトリの入力フィールドを定義します。このフィールドは、論理行 1 の カラム 2 に表示されます。インストール ディレクトリの入力フィールドに入力された値は、パラメータ \$PATHmerchant に割り当てられます。

```
#navbutton: 1 , btnInstall , 3 , 1 ;
```

```
#navbutton: 2 , btnCancel , 4 , 0 ;
```

インストール プロセス間の移動に使用するボタンを定義します。

最初のボタン **btnInstall** を選択すると、製品がインストールされます。ボタンのテキストは、リソース セクションのテキスト番号 3 に含まれます。ボタンを選択したときに実行されるアクションは、「製品のインストール」 (1) です。

2 番目のボタン **btnCancel** を選択すると、インストールは中止されます。ボタンのテキストは、リソース セクションのテキスト番号 4 に含まれます。ボタンを選択したときに実行されるアクションは、「インストールの中止」 (0) です。

プロトタイプ ファイルのリソース セクションにおけるこのダイアログ ボックスのテキストは、以下のとおりです。

```
@RESOURCE:
#locale: JPN ;
#text: 1 , Select installation directory ;
#text: 2 , Installation directory: ;
#text: 3 , &Install ;
#text: 4 , &Cancel ;
@ENDRESOURCE:

#locale: JPN ;
```

文字列の言語を日本語で定義します。([言語識別子](#) (P. 180) の説明を参照してください) 。

#text: 1 , Select installation directory ;

#text: ... , ... ;

ダイアログ ボックス定義で使用するテキスト番号を定義します。たとえば、テキスト「インストールディレクトリの選択」はテキスト番号 **1** に割り当てられ、この番号はプロトタイプファイルのダイアログ ボックス セクションでダイアログ ボックス ヘッダの定義に使用されます。

文字列の番号 **3** および 番号 **4** にアンパサンド (&) を使用すると、大文字 **I** および **C** がインストールおよびキャンセルのショートカットになります。

製品 MERCHANT をインストールするとき [インストールディレクトリの選択] ダイアログ ボックスを最初に表示するように指定するには、以下のように、プロトタイプファイルの製品情報セクションのキーワード **#dlgpreinit** を使用します。

```
@PRODUCT:
#phead: MERCHANT , 2.1.0.0 ;
#sys: Any ;
#locale: JPN ;
#comment: これはサンプルの PIF 製品です ;
#ppath: $PATHmerchant ;
#pdep: Ingres , 4.0.0.0 , >= ;
#postinit: bin/create_tables.sh ;
#dlgpreinit: dlgInstallationDirectory ;
@ENDPROD:
```

#dlgpreinit: dlgInstallationDirectory ;

dlgInstallationDirectory の名前が付いたダイアログ ボックスを最初のインストール ダイアログ ボックスとして表示することを指定します。

コマンドライン インターフェースおよびパッケージ GUI によって、PIF 製品にインストール ダイアログ ボックスが自動的に追加されます。アクションはありません。

PIF 製品 MERCHANT への言語固有の製品ファイルの追加

PIF 製品 MERCHANT では、ローカライズされたメッセージが英語とドイツ語で印刷されます。このため、テキスト ファイル **merchant.enu** および **merchant.deu** が製品に追加されます。

これらのテキスト ファイルは言語固有のコンポーネントに追加されます。言語固有のコンポーネントは、そのコンポーネントで指定する言語とターゲット システムのロケール (`$LANG`) が一致した場合にのみインストールされます。言語固有のコンポーネントでは、キーワード `#locale` (この例では `ENU` および `DEU`) のプロパティを使用して言語を指定します。

プロトタイプ ファイルの以下のセクションでは、英語のテキスト ファイルを含む言語固有のコンポーネントが指定されます。

```
@COMPONENT:
#thead: lang_spec ;
#locale: JPN ;
#file: bin/merchant.enu , /myhomepath/src/merchant.enu , root , sys , -r--r--r-- ,
1 ;
@ENDCOMP:
```

コンポーネントの名前は `lang_spec` で、`merchant.enu` テキスト ファイルに含まれています。 `lang_spec` は、ターゲット システムのロケールが英語を表す値 (`en_US` など) を持つ場合にインストールされます。

プロトタイプ ファイルの以下のセクションでは、ドイツ語のテキスト ファイルを含む言語固有のコンポーネントが指定されます。

```
@COMPONENT:
#thead: lang_spec ;
#locale: DEU ;
#file: bin/merchant.deu , /myhomepath/src/merchant.deu , root , sys , -r--r--r-- ,
1 ;
@ENDCOMP:
```

このコンポーネントは英語版と同じ `lang_spec` という名前ですが、キーワード `#locale` のプロパティによって異なります。この `lang_spec` は、`merchant.deu` テキスト ファイルに含まれており、ターゲット システムのロケールがドイツ語を表す値を持つ場合にインストールされます。

言語固有のリソースを外部製品の拡張機能として利用できます。最初に、製品はそのオリジナル テキスト リソースを使用してビルドされます。その後、以下のように `pifextract` コマンドを使用すると、言語固有のテキスト リソースを製品から抽出できます。

```
pifextract -f MERCHANT.Any.@pif -l ENU -L DEU -d german_folder
```

このコマンドを使用すると、すべての言語固有のコンポーネントが `german_folder` フォルダに抽出されて、適切なプロトタイプ ファイルが作成されます。テキスト リソース ファイルを置き換え、プロトタイプ ファイルの適切な `#file` エントリをカスタマイズした後に、`pifmk` コマンドを使用して外部の PIF 製品をビルドできます。その結果、PIF ファイル `MERCHANT.@deu` が作成されます。

`MERCHANT.@deu` はメインの PIF 製品のフォルダにコピーする必要があります。

PIF 製品 MERCHANT のインストール ダイアログ ボックスのローカライズ

実際のダイアログ ボックス テキストとプロトタイプ ファイルのダイアログ ボックス定義は分離されているため、ほかの言語の新しいリソース セクションを容易に作成することができます。既存のリソース セクションをコピーして変更するだけです。以下の例では、プロトタイプ ファイルのオリジナル リソース セクションのコピーと貼り付け、文字列の英語からドイツ語への変換、および `#locale` プロパティの `ENU` から `DEU` への変更を行って、新しい言語を指定します。

PIF 製品 MERCHANT のドイツ語のリソース セクションは、以下のとおりです。

```
@RESOURCE:
#locale: DEU ;
#text: 1 , Auswahl des Installationsverzeichnisses ;
#text: 2 , Installationsverzeichnis: ;
#text: 3 , &Installieren ;
#text: 3 , &Abbrechen ;
@ENDRESOURCE:
```

コマンド ラインを使用したインストール ダイアログ ボックスのローカライズ

エディタを使用して、プロトタイプ ファイルに言語固有のリソース セクションを追加します。

プロトタイプ ファイルにほかの言語のリソース セクションを作成する場合は、以下の手順を実行します。

1. 既存の元のリソース セクションをコピーします。
2. テキスト文字列を変換します。
3. `#locale` キーワードのプロパティを変更して、新規の言語を指定します。

GUI を使用したインストール ダイアログ ボックスのローカライズ

GUI を使用したインストール ダイアログ ボックスのローカライズ

インストール ダイアログ ボックスのローカライズされたテキスト文字列を作成するには、以下の手順に従います。

1. `smgui` コマンドを実行し、パッケージ GUI を起動します。
2. プロジェクト ペインで PIF 製品 MERCHANT を選択します。
3. サブプロジェクト ペインの[ダイアログ ボックス リソース]タブで[英語/US] エントリを選択して、ダブルクリックします。
4. 選択したリソースを右クリックし、コンテキスト メニューから [コピー] を選択します。

[プロシージャ リソース] ダイアログ ボックスが表示されます。

5. ドロップダウン メニューから目的の言語 [ドイツ語] を選択します。
6. [コピー] をクリックします。

[ドイツ語] 言語エントリが、サブプロジェクト ペインにある [ダイアログ ボックス リソース] タブに表示されます。

[ドイツ語] ダイアログ ボックス リソースのテキストを変換するプロセスは、以下のとおりです。

7. [ダイアログ ボックス リソース] タブの [ドイツ語] 言語エントリを選択します。
8. [ダイアログ] テキスト テーブルのテキストをダブルクリックして、ドイツ語に変換したテキストを直接その行に入力します。文字列を変換するには、この手順を繰り返して実行します。

最終的な PIF 製品 MERCHANT のビルド

前のセクションでビルドしたサンプル PIF 製品 MERCHANT のプロトタイプ ファイル MERCHANT.Any.@prm は、以下のとおりです。

```
@PRODUCT:
#phead: MERCHANT , 2.1.0.0 ;
#sys: Any ;
#locale: JPN ;
#comment: これはサンプルの PIF 製品です ;
#ppath: $PATHmerchant ;
#pdep: Ingres , 4.0.0.0 , >= ;
#postinit: bin/create_tables.sh ;
#dlgpreinit: dlgInstallationDirectory ;
@ENDPROD:

@COMPONENT:
#chead: component1 ;
#dir: bin , root , sys , drwxrwxrwx ;
#file: bin/merchant.jar , /myhomepath/src/merchant.jar , root , sys , -r--r--r-- ,
1 ;
#file: bin/run_merchant.sh , /myhomepath/scripts/run_merchant.sh , root , sys ,
-r-xr-xr-x , 1 ;
#file: bin/create_tables.sh , /myhomepath/scripts/create_tables.sh , root , sys ,
-r-xr-xr-x , 1 ;
#slink: /usr/bin/run_merchant.sh , bin/run_merchant.sh ;
@ENDCOMP:

@DEFAULT:
#parameter: $PATHmerchant , /home/merchant ;
@ENDDFAULT:
```

```
@DIALOG:
#dhead: 600 , 400 , dlgInstallationDirectory , 1 ;
#label: 1 , 1 , lblInstallDirectory , 2 ;
#instdir: 1 , 2 , tfInstallationDirectory , $PATHmerchant ;
#navbutton: 1 , btnInstall , 3 , 1 ;
#navbutton: 2 , btnCancel , 4 , 0 ;
@ENDDIALOG:

@RESOURCE:
#locale: JPN ;
#text: 1 , Select installation directory ;
#text: 2 , Installation directory: ;
#text: 3 , &Install ;
#text: 4 , &Cancel ;
@ENDRESOURCE:

@COMPONENT:
#thead: lang_spec ;
#locale: JPN ;
#file: bin/merchant.enu , /myhomepath/src/merchant.enu , root , sys , -r--r--r-- ,
1 ;
@ENDCOMP:

@COMPONENT:
#thead: lang_spec ;
#locale: DEU ;
#file: bin/merchant.deu , /myhomepath/src/merchant.deu , root , sys , -r--r--r-- ,
1 ;
@ENDCOMP:

@RESOURCE:
#locale: DEU ;
#text: 1 , Auswahl des Installationsverzeichnis: ;
#text: 2 , Installationsverzeichnis: ;
#text: 3 , &Installieren ;
#text: 4 , &Abbrechen ;
@ENDRESOURCE:
```

コマンドラインを使用して PIF 製品 MERCHANT をビルドするには、以下のように pifmk コマンドをプロトタイプ ファイルがある場所のディレクトリに入力します。

```
pifmk -f MERCHANT.Any.@prm
```

PIF 製品 MERCHANT は、現在の作業ディレクトリにビルドされます。PIF ファイル名は MERCHANT.Any.@pif です。

パッケージ GUI を使用して PIF 製品 MERCHANT を作成するには、以下の手順に従います。

1. smgui コマンドを実行し、パッケージ GUI を起動します。
2. プロジェクト ペインで PIF 製品 MERCHANT を選択します。
3. ツールバーから [ビルド] を選択します。

[製品のビルド] ダイアログ ボックスが表示されます。

この例の PIF 製品では、[削除および自己インストール] チェック ボックスは無視されます。

製品 MERCHANT がすでに存在している場合は、ほかのチェック ボックスが表示され、オンにすると、既存製品を上書することができます。

4. 「ビルド」をクリックします。

メディア テーブルに PIF 製品 MERCHANT.Any.@pif が表示されます。

第 6 章：インストーラ タスク

Linux および UNIX 用インストーラ (Installer) では、ターゲット コンピュータに PIF 製品を実際にインストールする前後に、多くのタスクが実行されます。PIF 製品をインストールする前のタスクをプリインストール タスクと呼び、実際に製品をインストールした後のタスクをポストインストール タスクと呼びます。

インストーラでは、以下のプリインストール タスクが実行されます。

- ソフトウェアが揃っているかどうかの検証
- プリインストール コンポーネントの一時フォルダへの抽出
- マイグレーション スクリプトの実行 (新しい製品をインストールする場合にのみ)
- ダイアログ ボックスによる必要なパラメータの検索
- インストール済み製品のバックアップの作成 (更新インストールの場合)

インストーラでは、以下のインストール タスクが実行されます (実際の製品インストール)。

- 環境からパラメータ値を読み取ります。
- 外部のプリインストール コンポーネントをインストールします。
- プリインストール スクリプトを実行します。
- PIF 製品のコンポーネントをインストールします。
- 外部のポストインストール コンポーネントをインストールします。
- ポストインストール スクリプトを実行します。
- ローカル データベースの製品情報を登録します。

PIF 製品の更新インストールが失敗した場合は、インストーラのポストインストール タスクとして、インストール済み PIF 製品のバックアップをリストアして、前のバージョンの製品を再度インストールします。

このセクションには、以下のトピックが含まれています。

[プリインストール タスク](#) (P. 118)

[製品インストール中のタスク](#) (P. 121)

[ポストインストール タスク](#) (P. 123)

プリインストール タスク

ここで説明するプリインストール タスクは、PIF 製品をインストールする前にインストーラによって自動的に実行されます。

規模の大きい PIF 製品をインストールする場合は、インストールの準備フェーズに多少の時間がかかる場合があります。インストーラでは、ユーザに対して目に見える情報は示されませんが、プリインストール スクリプトが実行され、製品状態やディスク構造の確認作業が行われています。このフェーズでは、インストールが継続していることをユーザに知らせることが重要です。

キーワード `#splashscreen` を使用すると、GIF フォーマットのグラフィックおよびオプションのテキストの表示を指定することができます。このテキストがコンソールに表示されます。たとえば、入力コマンドラインの次の行に、以下のとおり表示されます。

```
lsm -i NSM.Linux.@pif
```

```
Preparing installation: .....
```

進捗状況は、インストール インタビュー フェーズを実行する対話式インストールでのみ表示されます。

ソフトウェアの検証

インストーラでは、最初のタスクとして、PIF 製品のインストールに必要なすべてのソフトウェアが、インストール メディア (DVD、CD-ROM) の中に存在することが検証されます。たとえば、外部コンポーネントが参照されているが、まだインストールされていない場合、このコンポーネントを含む PIF 製品ファイルは、メインの PIF 製品と同じフォルダに存在する必要があります。また、インストールされていない必要な PIF 製品は、メインの PIF 製品と同じフォルダに存在する必要があります。

プリインストール コンポーネントの一時フォルダへの抽出

インストール ダイアログ ボックスおよびスクリプトを実行する前に、製品のプリインストール コンポーネントを一時フォルダに抽出します。

製品のスクリプトで、同じインストール トランザクションのほかに必要な製品のスクリプトを呼び出す必要がある場合は、この一時フォルダの名前を認識する必要があります。このため、パラメータ `$PIF_PREINSTALL_DIR` には、一時的なプリインストール フォルダの名前が設定されています。

プリインストール コンポーネントの保存場所は、以下のとおりです。

`$PIF_PREINSTALL_DIR/prodname.d/¥$PRODUCT_PATH`

`$PRODUCT_PATH` には、製品固有のインストール パスのパラメータが含まれます。

例：

`PIF_PREINSTALL_DIR` が `/tmp/EXTRACT12345/` に設定されている場合、パッケージ製品のスクリプトは以下の場所にあります。

`/tmp/EXTRACT12345/SMPACKAGER.d/¥$SDPcker/scripts`

注：製品インストールの中間ディレクトリを参照するマスクされたパラメータ（`abc¥$efg` など）は、この時点で保存されます。

マイグレーション スクリプトの実行

PIF 以外の製品のインストールをマイグレートするには、製品を最初にインストールする前にインストーラによって実行されるマイグレーション スクリプトを指定します。プロトタイプ ファイルの製品情報セクションにあるキーワード `#prepif` を使用して、PIF 製品用のマイグレーション スクリプトを指定する必要があります。

マイグレーション スクリプトを使用すると、`#ppath`（たとえば、`$MY_PIF_INSTALL_PATH`）およびインストール ジョブ（`$PIF_INSTALL_JOB`）を使用して定義した製品固有のインストール パス パラメータを変更することができます。

PIF パラメータは、シェル環境および一時応答ファイルを使用して渡されます。PIF 製品のインストール プロセスでは、更新済みパラメータが認識され、後で応答ファイルに追加されます。

例：マイグレーション スクリプト

以下のスクリプトは、古いインストールのベース ディレクトリを確認し、インストール ディレクトリを設定して、インストール モードを **INSTALL** から **UPDATE** に変更するスクリプトです。

```
## check if response file parameter is passed
[ ! "$PIF_RESPONSE_FILE" -o ! -s "$PIF_RESPONSE_FILE" ] && exit 1
## check non PIF installation: product is either installed under /home or /opt
directory
if [ -d /home/merchant ]
then
    echo "MY_PIF_INSTALL_PATH=/home/merchant" >> $PIF_RESPONSE_FILE
    echo "PIF_INSTALL_JOB=UPDATE" >> $PIF_RESPONSE_FILE
elif [ -d /opt/merchant ]
then
    echo "MY_PIF_INSTALL_PATH=/opt/merchant" >> $PIF_RESPONSE_FILE
    echo "PIF_INSTALL_JOB=UPDATE" >> $PIF_RESPONSE_FILE
fi
exit 0
```

インストール ダイアログ ボックスの実行

インストーラによって、PIF 製品用に定義されているインストール ダイアログ ボックスが実行されます。インストーラでは、このダイアログ ボックスから PIF 製品のインストールを設定するパラメータを取得します。

ターゲット コンピュータに **Java Runtime Environment (JRE) 1.5** または **1.6** がインストールされていて、モニタがグラフィック表示対応であると、インストール ダイアログ ボックスがグラフィカル モード (**Java GUI**) で表示されます。ターゲット コンピュータでこれらの前提条件が満たされない場合、インストール ダイアログ ボックスは **VT100** 互換モードで表示されます。

JRE を自己インストール PIF 製品に含めることができます。すると、含まれた **JRE** がインストールおよび使用され、ターゲット コンピュータでインストーラの **Java** ウィザードが実行されます。

ポストインタビュー スクリプトの実行

インストーラは、すべてのインストール ダイアログが実行された後にスクリプトを処理します。このスクリプトを使用して、現在設定されているインストール ジョブを変更できます。スクリプトは、各インストール フェーズ(インストール、更新、または再インストール) ごとに自動モードで処理されます。

このスクリプトは、プロトタイプ ファイルでは **#postinterview** キーワードを使用して指定されます。ポストインタビュー スクリプトは、プリインストール コンポーネントに追加する必要があります。

バックアップの作成

PIF 製品の更新中は、インストーラによって、インストール済み製品の一時バックアップが自動的に作成されます。このバックアップは更新インストールが失敗した場合にリストアされます。

内部パラメータ **PIF_BACKUP** では、現行バージョンの製品を更新または再インストールする前に、バックアップの作成を制御します。

バックアップの作成は、**Ism** コマンドで **-F** スイッチを指定して無効にできます。

製品インストール中のタスク

ここでは、PIF 製品をターゲット コンピュータにインストールするときのインストーラのタスクについて説明します。

環境からのパラメータ値の読み取り

PIF 製品のインストールの設定は、パラメータで行います。パラメータを使用すると、インストール パスを定義し、インストール プロセスからコンポーネントを追加または削除することができます。また、製品固有のパラメータをインストールに渡すこともできます。これらのパラメータは、インストール プロシージャのスクリプト内で使用します。

PIF 製品のインストールでは、以下の順序でパラメータが解決されます。

1. 応答ファイルによって渡された PIF パラメータが、読み取られます。ユーザは、lsm コマンドで -r スイッチを指定して、インストール時の応答を指定できます。また、インストール ダイアログ ボックスで集めたパラメータは、一時応答ファイルを経由してインストールプロセスに渡されます。
2. 自動インストールの場合は、環境変数によって応答ファイルのパラメータが上書きされます。環境変数は、PIF 製品のインストールが開始する前に設定する必要があります。

プリインストール スクリプトの実行

プロトタイプ ファイルのプリインストール スクリプトを指定した場合は、すぐにインストールされます。プリインストール スクリプトは、PIF 製品用または PIF 製品のコンポーネント用に定義されているため、ファイルおよびディレクトリのインストールが開始される前に実行する必要があります。プリインストールのスクリプトでは、インストール処理を確実に成功させるため、たとえば、デーモンプロセスの停止やインストール環境の確認を行います。(ディスクは共有ですか? スケーラビリティ サーバは使用可能ですか?)

パラメータは、すべてのプリインストール スクリプトおよびポストインストール スクリプトにシェル パラメータとして渡されます。新規または更新されたパラメータは、応答ファイルに追加する必要があります。

例：サーバの可用性を確認するスクリプト

サーバ（サーバの名前は \$SERVERNAME パラメータによって渡される）が使用可能かどうかを確認するスクリプトは、以下のとおりです。使用可能な場合、スクリプトによってコンポーネントのインストール フラグ（プロトタイプ ファイルのコンポーネント セクションにある #cinstall=\$INSTALL_SERVER）が起動されます。

```
## exit if response file has not been passed
[ ! "$PIF_RESPONSE_FILE" -o ! -s "$PIF_RESPONSE_FILE" ] && exit 1

## exit if the server name has not been passed
[ ! "$SERVERNAME" ] && exit 1

## server responds, activate the server component
if [ 'ping $SERVERNAME 2>/dev/null; echo $?' -eq 0 ]
then
    echo "INSTALL_SERVER=1" >> $PIF_RESPONSE_FILE
else
    echo "INSTALL_SERVER=0" >> $PIF_RESPONSE_FILE
fi
```

コンポーネントのインストール

インストーラのメイン タスクは、**PIF** 製品のコンポーネントをインストールすることです。ターゲット コンピュータにコンポーネントの内容、すなわち、ファイル、ディレクトリ、およびシンボリック リンクが以下の順序でインストールされます。

1. すべてのコンポーネントのすべてのディレクトリが作成されます。
2. すべてのファイルがインストールされます。
3. シンボリック リンクが作成されます。

ポストインストール スクリプトの実行

外部のポストインストール コンポーネントをインストールした後、プロトタイプ ファイルで指定されている場合は、インストーラによってポストインストール スクリプトが実行されます。ポストインストール スクリプトは、**PIF** 製品用または **PIF** 製品のコンポーネント用に定義されています。ポストインストール スクリプトは、通常、インストール済み **PIF** 製品を設定し、起動します。たとえば、デーモンプロセスを開始したり、データベース テーブルを作成します。

製品情報の登録

ターゲット コンピュータのローカル **PIF** データベースに **PIF** 製品を登録すると、インストーラの主要なインストール タスクは終了します。パラメータ値は **PIF** 製品と共に登録されます。**PIF** 製品を登録すると、**PIF** 製品の確認、クエリ、削除が正常に機能します。

ポストインストール タスク

PIF 製品の更新インストールがエラーのために失敗した場合は、インストーラによって、前にインストールされたバージョンの **PIF** 製品がバックアップから自動的にリストアされます。

第 7 章： 参照

規則、標準テンプレート、ファイルのフォーマットと構造、パッケージおよびインストーラのコマンド、サポート動作環境、言語などの詳細説明。

このセクションには、以下のトピックが含まれています。

[ファイルの命名規則](#) (P. 125)

[プロトタイプ ファイルのフォーマットと構造](#) (P. 126)

[プロトタイプ ファイルのキーワードとプロパティ](#) (P. 132)

[パッケージおよびインストーラのコマンド](#) (P. 157)

[自己インストール PIF 製品をインストール](#) (P. 177)

[サポートされる Linux および UNIX 動作環境](#) (P. 179)

[言語識別子](#) (P. 180)

[テンプレートパラメータおよび pifproto コマンドスイッチ](#) (P. 181)

ファイルの命名規則

PIF SDK を使用してプロトタイプ ファイルや PIF 製品を作成する場合は、以下に示すファイルの命名規則に従う必要があります。

プロトタイプ ファイル名

プロトタイプ ファイルのファイル名は、以下のとおりです。

`prodname.platform.@prm`

PIF 製品のファイル名

PIF 製品のファイル名は、以下のとおりです。

`prodname.platform.@pif`

言語リソースのプロトタイプ ファイル名

言語リソースのプロトタイプ ファイル名は、以下のとおりです。

`prodname.languageID.@prm`

このファイル名では、*languageID* は、ENU（英語（米国））、DEU（ドイツ語）、FRA（フランス語）のように、3 文字の[言語識別子](#) (P. 180) になります。

言語リソースの PIF 製品ファイル名

言語リソースの PIF 製品ファイル名は、以下のとおりです。

`prodname.@languageID`

このファイル名では、*languageID* は、**ENU**（英語（米国））、**DEU**（ドイツ語）、**FRA**（フランス語）のように、3 文字の言語識別子になります。

差分製品のファイル名

差分製品（同じ PIF 製品の前のバージョンと現在のバージョンとで異なるファイルが含まれる）のファイル名は、以下のとおりです。

`prodname.platform.priorversion-resultingversion.@pif`

パッケージ **GUI** から差分製品を作成する場合、ファイル名は自動的にこのフォーマットで生成されます。**pifdelta** コマンドで差分製品を作成する場合も、このファイル命名規則に従うことをお勧めしますが、必須ではありません。

プロトタイプ ファイルのフォーマットと構造

プロトタイプ ファイルは **ASCII** ファイルで、PIF 製品に関する情報を指定するために使用します。この情報はセクションの中にバンドルされており、目的や情報の内容によって異なります。PIF 製品に関係しない場合、セクションは省略されます。プロトタイプ ファイルには、以下のセクションを含めることができます。

- 製品情報セクション
- コンポーネント セクション
- パラメータ セクション（デフォルト セクション）
- ダイアログ ボックス セクション
- リソース セクション

プロトタイプ ファイルは、**pifproto** コマンドを使用するか、またはパッケージ **GUI** を使用して適当な値を入力して作成します。作成されるプロトタイプ ファイル名は、以下のとおりです。

`prodname.platform.@prm`

PIF 製品のプロトタイプ ファイルには、以下の表記規則が適用されます。

- プロトタイプ ファイルにおけるすべてのエントリの構造は、以下のとおりです。
`#keyword: property1 , property2 , ... ;`
 各キーワードの先頭にはハッシュ記号(#)が、終わりにはコロン (:) が付きます。
 各プロパティは、スペース、カンマ、スペースの順(,)で後続のプロパティと区切ります。
 角かっこ [] で囲まれたプロパティはオプションであり、省略できます。プロパティ リストでは、コンパイラの仕様により、スペース、カンマ、スペースの順で区切る必要があります。たとえば、プロパティ 2、プロパティ 4、プロパティ 5 がオプションの場合は、以下のようにします。
`#keyw: prop1 , , prop3 , , , prop6 ;`
- 各エントリは、スペース、セミコロン (;) で終了します。
- エントリは、複数行にわたり記述することはできません。
- 角かっこ [] で囲まれたエントリはオプションです。オプションのエントリは、値を指定せずに入力できます。エントリ全体を省略することもできます。
- コメントは、ダブルスラッシュ (//) またはスラッシュ、アスタリスク (/*) で始まり、アスタリスク、スラッシュ (*/) で終了します。
- プロトタイプ ファイルのエントリには、キャリッジリターン(CR)、ラインフィード (LF) などの制御キャラクタを使用することはできません。

製品情報セクション

製品情報セクションは、キーワード @PRODUCT および @ENDPROD でフレーム化されています。製品情報セクションには、PIF 製品や製品に関連するプロシージャの一般情報が記述されています。このセクションを構成するエントリの一覧は、以下のとおりです。エントリの詳細については、該当するキーワードの説明を参照してください。

@PRODUCT:

```
#phead: ..... ;
#sys: ..... ;
#ppath: ..... ;
#locale: ..... ;
[#build: ..... ;]
[#supplier: ..... ;]
[#comment: ..... ;]
[#productfamily: ..... ;]
```

```
[#frameheader: ..... ;]
[#patchproduct: ..... ;]
[#replacepatches: ..... ;]
[#vname: ..... ;]
[#admininst: ..... ;]
[#multiinst: ..... ;]
[#timer: ..... ;]
[#prepif: ..... ;]
[#prestart: ..... ;]
[#postinterview: ..... ;]
[#preinitcomp: ..... ;]
[#preventdowngrade: ..... ;]
[#pdep: ..... ;]
[#dis: ..... ;]
[#preinit: ..... ;]
[#postinit: ..... ;]
[#rmpreinit: ..... ;]
[#rmpostinit: ..... ;]
[#extpreinit: ..... ;]
[#extpostinit: ..... ;]
[#extrmpreinit: ..... ;]
[#extrmpostinit: ..... ;]
[#dlgcancel: ..... ;]
[#dlgend: ..... ;]
[#dlgendrm: ..... ;]
[#dlgerror: ..... ;]
[#dlgerrordisk: ..... ;]
[#dlgerrorpath: ..... ;]
[#dlgnextdisk: ..... ;]
[#dlgnextpath: ..... ;]
[#dlgpreinit: ..... ;]
[#dlgpreupd: ..... ;]
[#dlgprogress: ..... ;]
[#splashscreen: ..... ;]
[#postcheck: ]

@ENDPROD:
```

コンポーネント セクション

コンポーネント セクションは、キーワード **@COMPONENT** および **@ENDCOMP** でフレーム化されています。このセクションでは、PIF 製品の小さなインストール単位であるコンポーネントについて記述しています。コンポーネント セクションを構成するエントリの一覧は以下のとおりです。エントリの詳細については、該当するキーワードの説明を参照してください。

@COMPONENT:

```
#thead: ..... ;  
[#sys: ..... ;]  
[#locale: ..... ;]  
[#cinstall: ..... ;]  
[#cpath: ..... ;]  
[#preinit: ..... ;]  
[#postinit: ..... ;]  
[#rmpreinit: ..... ;]  
[#rmpostinit: ..... ;]  
[#extpreinit: ..... ;]  
[#extpostinit: ..... ;]  
[#extrmpreinit: ..... ;]  
[#extrmpostinit: ..... ;]  
[#file: ..... ;]  
[#dir: ..... ;]  
[#slink: ..... ;]  
[#reservedspace: ..... ;]
```

@ENDCOMP:

デフォルト セクション (パラメータ セクション)

デフォルト セクション (またはパラメータ セクション) は、キーワード **@DEFAULT** および **@ENDDEFAULT** でフレーム化されています。このセクションでは、以下に示すように、キーワード **#parameter** および **#globalparameter** を使用して、製品固有のパラメータとそのデフォルト値を定義します。エントリの詳細については、該当するキーワードの説明を参照してください。

インストーラではデフォルト値をインストール ダイアログ ボックスの初期値として使用したり、製品の自動インストールのために使用します。

```
@DEFAULT:

[#sys: ..... ;]
#parameter: ..... ;

@ENDDDEFAULT:
```

ダイアログ ボックス セクション

ダイアログ ボックス セクションは、キーワード **@DIALOG** および **@ENDDIALOG** でフレーム化されています。このセクションでは、製品の自動インストール中に実行するダイアログ ボックスのレイアウトについて定義します。ダイアログ ボックス セクションを構成するエントリの一覧は以下のとおりです。エントリの詳細については、該当するキーワードの説明を参照してください。

同じ定義は、**Java** および **VT100** のダイアログ ボックスで使用されます。インストーラでは、ターゲット コンピュータ上での適切な表示が管理されます。

```
@DIALOG:

#dhead: ..... ;
[#graphic: ..... ;]
[#textfield: ..... ;]
[#passwordfield: ..... ;]
[#instdir: ..... ;]
[#textarea: ..... ;]
[#filearea: ..... ;]
[#label: ..... ;]
[#checkbox: ..... ;]
[#button: ..... ;]
[#navbutton: ..... ;]
[#combobox: ..... ;]
[#radiobutton: ..... ;]
[#progressbar: ..... ;]
[#progresslabel: ..... ;]
[#progressarea: ..... ;]
[#fileselection: ..... ;]
[#scrollarea_begin: ..... ;]
[#scrollarea_end: ..... ;]
```

```
[#tree_begin: ..... ;]
[#tree_end: ..... ;]
[#image:]
[#fileselection:]
[#inputorder:]

@ENDDIALOG:
```

Fontlist セクション

Fontlist セクションは、キーワード **@FONTLIST** および **@ENDFONTLIST** でフレーム化されています。このセクションには、インストーラ ダイアログ用のフォント定義が含まれます。インストーラでは、サイズ、ボールド、イタリック、色などのフォントプロパティや、**Serif**、**SansSerif**、**Monospaced**、**Dialog**、および **DialogInput** などの Java 論理フォントを変更できます。**#font** エントリの詳細については、キーワードの説明を参照してください。

```
@FONTLIST:

#font: ..... ;

@ENDFONTLIST:
```

Desktop セクション

Desktop セクションは、キーワード **@DESKTOP** および **@ENDDESKTOP** でフレーム化されています。このセクションでは、PIF パッケージ用のメニューおよびデスクトップアイコンを指定できます。

エントリの詳細については、該当する説明を参照してください。

```
@DESKTOP:

[#desktopicon: ..... ;]

[#desktopmenu: ..... ;]

@ENDDESKTOP:
```

リソース セクション

リソース セクションは、キーワード `@RESOURCE` および `@ENDRESOURCE` でフレーム化されています。このセクションには、PIF 製品の言語固有の文字列が含まれています。この文字列は、たとえば、インストール ダイアログ ボックスで使用されます。文字列は、テキスト識別番号で参照されます。ロケール エントリでは、PIF 製品のデフォルト 言語を指定します。

`@RESOURCE:`

```
#locale: ..... ;  
#text: ..... ;  
#utf8encoded: ..... ;
```

`@ENDRESOURCE:`

注：文字列は、内部ではマルチバイト文字モードでフォーマットされています。PIF 製品の作成中、文字列は UTF8 コードに変換されます。インストーラでは、インストール時にこの文字列をマルチバイト文字モードに戻します。

プロトタイプ ファイルのキーワードとプロパティ

キーワードおよびプロパティは、PIF 製品のすべてのコンポーネント、機能、ファイル、およびパラメータを指定するために、プロトタイプ ファイルで使用されます。キーワードの先頭にはハッシュ記号(`#`) が付きます。

製品の詳細仕様は、各キーワードの後のプロパティによって定義されます。

キーワード

以下に、PIF 製品のプロトタイプ ファイルのさまざまなセクションで使用可能なすべてのキーワードをアルファベット順に示します。

`#admininstall: flag ;`

製品をルート ユーザ権限 (`flag 値 = True (1)`) で、または非ルート ユーザ権限 (`flag 値 = False (0)`) でインストールするかどうかを指定します。

#build: *[buildinformation]* ;

ユーザへの提供のみを目的とした製品ビルドについての情報、たとえば、同じ製品バージョンの異なるパッケージを識別する情報を示します。インストーラでは、この情報はチェックされません。

#button: *row , column , ctrlname , textID , action , [icon] , [nextdialog] , [paramset] , [fontname] , [resultpane]* ;

インストール ダイアログ ボックスの入力領域内のボタンおよびその位置を指定します。 *textID* オプションは、表示されるテキストを指定します。 *Nextdialog* は、次のダイアログ ボックス名、またはダイアログ ボックス名の動的な検出などのアクションを実行するスクリプトの名前を示します。 *icon* は、ボタンに対して指定することができます。 *paramset* オプションは、ボタンをクリックしたときに設定されるパラメータのリストを指定します。 *action* は、ボタンを押すと開始されるアクションを指定する番号です（例： 2 = 次のダイアログ ボックスの表示。 *action* の値の完全なリストは、 *action* プロパティの [Prototype File Properties セクション] にあります。）スクリプト実行の結果（ *action* = 9）を結果ペインに表示できます。また、 *resultpane* パラメータでは進捗状況エリアまたはテキスト エリアを指定できます。

#chead: *compname[, version]* ;

PIF 製品のコンポーネントの名前とバージョンを指定します。

#checkbox: *row , column , ctrlname , paramname , [actparamset] , [deactparamset] , [textID1] , [textID2] , [dependencylist] , [actionscript][, fontname]* ;

インストール ダイアログ ボックスの入力領域内のチェック ボックスおよびその位置を指定します。環境変数 *paramname* には、チェック ボックスのステータスが含まれています。チェック ボックスがオン（True）になっている場合、環境変数の値は（1）に設定され、それ以外の場合（False）は、変数はゼロ（0）に設定されます。カンマ区切りのパラメータ = 値の割り当ての 2 つのリストは、チェック ボックスをオンにする（ *actparamset* ）か、またはオフにした（ *deactparamset* ）場合に起動されるように指定することができます。 *textID1* オプションは、チェック ボックスの右側に表示されるラベルのテキストを指定します。 *textID2* の *dependencylist* および *actionscript* オプションは、チェック ボックス ツリーでのみ使用します。オプションの *textID2* ではツリーで選択された項目の追加の説明文を指定し、 *dependencylist* ではチェック ボックスの *ctrlnames* のカンマ区切りのリストを指定し、 *actionscript* ではインストール ダイアログ ボックスのフィールドの属性または内容をただちに変更するスクリプトを指定します。

#install: cinstparam ;

コンポーネントを製品と共にインストールするか、または必要に応じて（つまり、インストール ダイアログ ボックスでコンポーネントを選択して）インストールするかどうかを指定します。後者の場合、コンポーネントはオプションで呼び出されます。

#combobox: row , column , ctrlname , flag , listparam , resultparam , [dialogvalidationscript] , [actionscript][, fontname] ;

flag 値に応じて、コンボ ボックスまたはリスト ボックスを表示します。*flag* 値が **False (0)** の場合は、リスト ボックスが表示され、*flag* 値が **True (1)** の場合は、コンボ ボックスが表示されます。両方の種類のボックスにより、可能な値のリストが提供され、それらの値から **1** つを選択することができます。コンボ ボックスでは、別の値を入力することもできます。入力を検証するシェルスクリプトを指定したり、インストール ダイアログ ボックスのフィールド属性または内容をただちに変更する別のスクリプト (*actionscript*) を指定することができます。

#comment: textstring ;

製品の詳細な説明などのコメントを指定します。 *textstring* の最初の **127** バイトが、コメントとして **DSM** エクスプローラに表示されます。

#cpath: comppath ;

PIF 製品のコンポーネントのインストール パスを指定します。

#desktopicon: textID1 , targetpath , argument , iconpath , textID2[, userinstall][, allinstall] ;

PIF パッケージのデスクトップ アイコンを指定します。 *textID1* はアイコンと共に表示されるテキストの番号であり、*textID2* はローカライズされたコメントが含まれているテキストの番号です。 *userinstall* および *allinstall* のパラメータは、現在のユーザまたはすべてのユーザに対してアイコンがインストールされるかどうかを示します。

#desktopmenu: textID1 , targetpath , argument , iconpath , textID2[, userinstall][, allinstall] ;

PIF パッケージのデスクトップ メニューを指定します。 *textID1* はメニューと共に表示されるテキストの番号であり、*textID2* はローカライズされたコメントが含まれているテキストの番号です。 *userinstall* および *allinstall* のパラメータは、現在のユーザまたはすべてのユーザに対してメニューがインストールされるかどうかを示します。

#dhead: *width* , *height* ,
ctrlname , *textID* [, *dialogvalidationscript*] [, *fieldwithfocus*] ;

インストール ダイアログ ボックスの名前、位置、およびヘッドラインを指定します。入力を検証するシェルスクリプトを指定することができます。また、インストール ダイアログ ボックスを開いたときにフォーカスが設定されるダイアログ ボックス要素（フィールド）を定義することもできます。

#dir: *target* , *ownerID* , *groupID* , *access* ;

現在のコンポーネントに関連付けるディレクトリを指定します。

#dis: [*prodname* , *version* [, *operator*]] ;

互換性のない依存関係を指定します。この製品をインストールするときに、インストール先のコンピュータにインストールされないようにするすべての製品、およびそのバージョンを入力します。各依存ソフトウェア製品に対して 1 つの **#dis:** エントリを使用する必要があります。非互換性の依存関係がない場合は、このエントリを空白のままにするか、またはエントリを省略します。

#dlgcancel: *dlgref* ;

ユーザが PIF 製品のインストールをキャンセルすると表示されるダイアログ ボックスを指定します。

#dlgend: *dlgref* ;

製品のインストールが正常に完了した後に表示されるダイアログ ボックスを指定します。

#dlgendrm: *dlgref* ;

製品の削除が完了した後に表示されるダイアログ ボックスを指定します。

#dlgerror: *dlgref* ;

PIF 製品のインストール中にエラーが発生した場合に表示されるダイアログ ボックスを指定します。

#dlgerrordisk: *dlgref* ;

（複数ディスク サポートのみ）次に挿入されたディスクが一致していない場合に表示されるダイアログ ボックスを指定します。

#dlgerrorpath: *dlgref* ;

（複数ディスク サポートのみ）次に挿入されたパスが一致していない場合に表示されるダイアログ ボックスを指定します。

#dlnextdisk: *dlgrep* ;

(複数ディスク サポートのみ) 新しいディスクの挿入を要求する場合に表示されるダイアログ ボックスを指定します。

#dlnextpath: *dlgrep* ;

(複数ディスク サポートのみ) 新しいパスを要求する場合に表示されるダイアログ ボックスを指定します。

#dlgpreinit: *firstinstalldialog* ;

PIF 製品を最初にインストールするときに使用される最初のダイアログ ボックスを指定します。

#dlgpreupd: *firstupdatedialog* ;

PIF 製品を更新するときに使用するときに開くダイアログ ボックスを指定します。

#dlgprogress: *dlgrep* ;

PIF 製品のインストール中にインストールの進捗状況を視覚的に表示するダイアログ ボックスを指定します。

#extpostinit: *externalpostinstallprocedure* ;

PIF 製品のインストール後に実行する外部プロシージャを指定します。

#extpreinit: *externalpreinstallprocedure* ;

PIF 製品のインストール前に実行する外部プロシージャを指定します。

#extrmpostinit: *externalpostuninstallprocedure* ;

PIF 製品の削除後に実行する外部プロシージャを指定します。

#extrmpreinit: *externalpreuninstallprocedure* ;

PIF 製品の削除前に実行する外部プロシージャを指定します。

#file: *target , source , ownerID , groupID , access , filetype* ;

現在のコンポーネントに含まれるファイルを指定します。コンポーネントに含まれる各ファイルに対して、1 つの **#file:** エントリが必要となります。

#filearea: *row , column , ctrlname , filename[, fontname] ;*

参照先ファイルの内容が表示されるインストール ダイアログ ボックス内における領域を指定します。サポートされているファイル拡張子は、.htm、.html、.rtf、Java 用 .txt、および VT100 ユーザ インターフェース用 .txt です。ファイルの内容はファイル領域内で折り返されます。スクロール操作がサポートされています。パラメータを使用して、ファイル名を動的に設定できます。特殊用途のパラメータは、PIF_RESPONSE_FILE、PIF_ERROR_FILE、および PIF_LOG_FILE です。たとえば、内部変数 \$PIF_RESPONSE_FILE をファイル名として指定すると、内部的に使用される応答ファイルの内容を表示できます。

#fileselection: *row , column , ctrlname , paramname[, validationscript][, fontname] ;*

ファイル名入力フィールドの名前と位置、およびファイル名を格納する環境変数を指定します。入力を検証するシェルスクリプトを指定することができます。

#font: *fontname[, fontlogical][, fontsize][, fontbold][, fontitalic][, fontcolor] ;*

Java のインストール ダイアログ ボックスで使用されるフォントおよびフォントのプロパティを指定します。

#frameheader: *textID ;*

Java のインストール ダイアログ ボックスで、ウィンドウ フレーム ヘッダを指定します。textID は、Resource セクションで指定されたテキスト文字列の番号です。デフォルト値は、CA ソフトウェア管理インストーラです。

#graphic: *[GIFfilewest] , [GIFfilenorth] , [GIFfilecenter] , [GIFfilesouth][, GIFfileeast] ;*

Java のダイアログ ボックスの場合のみ。インストール ダイアログ ボックスのグラフィック領域に表示される 1 つ以上のグラフィックを指定します。グラフィック ファイルは、GIF 形式になっている必要があります。

#image: *row , column , ctrlname , imagefilename[, scalablewidth , scalableheight] ;*

インストール ダイアログ ボックスで表示されるイメージ (ピクチャ) を指定します。scalablewidth および scalableheight のパラメータはオプションです。これらが指定されている場合、定義されたサイズにイメージが調整されます。

#instdir: *row , column , ctrlname , paramname[, validationscript][, fontname] ;*

インストールディレクトリの入力フィールドの名前と位置、およびディレクトリを格納する環境変数を指定します。入力を検証するシェルスクリプトを指定することができます。

#inputorder: *ctrlname ;*

PIF パッケージのインストール時に、インストール ダイアログ ボックスの入力フィールドを処理する順序を指定します。

#label: *row , column , ctrlname , textID[, fontname] ;*

入力フィールド、チェック ボックス、またはインストールディレクトリに使用される説明のテキストを指定します。参照されるテキスト文字列には、最大 **1024** 文字を含めることができますが、ダイアログ ボックスにラベルが表示される場合は、**1** 行を超えないようにします。ラベルのテキストは、ダイアログ ボックス ウィンドウ内で折り返されません。

#locale: *localeID ;*

PIF 製品のデフォルト言語を指定します。localeID によって指定される言語は、デフォルト言語でインストール ダイアログ ボックスを開いたときに使用されます。(言語識別子 (P. 180)の説明を参照してください)。

#multiinst: *flag ;*

インストール先のコンピュータでの製品の複数インストールを有効または無効にするかを指定します。flag 値が **(0)** (デフォルト) の場合、インストール先のコンピュータでは、この製品バージョンのインスタンスはこれ以上作成することができません。flag 値が **True (1)** の場合、PIF 製品は、インストール先のコンピュータに複数回 (複数インスタンス) インストールすることができます。

#navbutton: *column , ctrlname , textID , action[, icon][, dlgref][, fontname] ;*

このナビゲーション ボタンをクリックすると表示される次のインストール ダイアログ ボックス **dlgref** を指定します。次のインストール ダイアログ ボックスを決定するシェルスクリプトを指定することができます。textID に、ナビゲーション ボタンに表示してその機能を視覚化するテキストを指定します。また、ナビゲーション ボタンに表示するアイコン (GIF ファイルとして) を指定することができます。action は、このナビゲーション ボタンを押すと開始されるアクションを指定する番号です (例: **2** = 次のダイアログ ボックスの表示)。インストール ダイアログ ボックスのナビゲーション領域には、最大 **4** つのナビゲーション ボタンを定義することができます。

#parameter: *paramname* , *paramvalue* , *paramcomment* , *paramtype* ;

製品インストールの設定に使用されるパラメータとそのデフォルト値を定義します。割り当てるパラメータ値には、以前定義したパラメータを含めることができます。

例 :

```
#parameter: $installdir , /opt/CA/installpath ;
```

```
#parameter: $my_install_path , $installdir/myproduct1 ;
```

paramcomment プロパティは、パラメータを記述する場合に使用されます。 **paramtype** の値により、パラメータが発行済み (値 = 0)、プライベート (値 = 1)、またはグローバル (値 = 2) かどうか制御されます。

#passwordfield: *row* , *column* , *ctrlname* , *paramname* [, *validationscript*] ;

パスワードの入力フィールドの名前と位置、およびパスワードを格納する環境変数を指定します。入力を検証するシェルスクリプトを指定することができます。

#patchproduct: *prodname* ;

この現在のパッチが変更する製品の名前を指定します。

例 : **#patchproduct:** common_utilities ;

#pdep: [*prodname* , *version* [, *operator*] [, *instparam*]] ;

製品の依存関係を指定します。現在の PIF 製品をインストールする前に、インストール先のコンピュータにインストールする必要がある製品およびそのバージョンを指定します。現在の製品が依存している各製品に対して、1 つの **#pdep:** エントリを使用する必要があります。製品の依存関係がない場合は、このエントリを空白のままにするか、または省略します。

#phead: *prodname* , *version* ;

PIF 製品の名前とバージョンを指定します。

#postinit: [*postinstallprocedure* [, *interpreter*]] ;

PIF 製品のインストール後に実行する内部プロシージャを指定します。シェル以外のインタプリタをプロシージャに対して指定することができます。

#postinterview: *script* ;

インタビュー フェーズ後 (つまり、すべてのダイアログ ボックスが実行された後) に実行されるスクリプトの名前を指定します。スクリプトをプリインストール コンポーネントに追加する必要があります。

#ppath: *prodpath* ;

PIF 製品のインストールパスを指定します。

#preinit: [*preinstallprocedure* [, *interpreter*]] ;

PIF 製品のインストール前に実行する内部プロシージャを指定します。シェル以外のインタプリタをプロシージャに対して指定することができます。

#preinitcomp: [*preinstallcomponent*] ;

PIF 製品より前にインストールするコンポーネントを指定します。

#prepif: [*migrationscript* [, *interpreter*]] ;

プロシージャ（PIF 製品に PIF 以外の製品をマイグレートするためにユーザが作成したスクリプト）を指定します。PIF 製品のインストールパスおよび PIF 製品のインストール モードは変更することができます（UPDATE、INSTALL）。prepif プロシージャは、PIF 製品の新規インストール前にのみ実行され、更新インストールまたは再インストール中は実行されません。シェル以外のインタプリタをスクリプトに対して指定することができます。

#prestart: [*prestartprocedure* [, *interpreter*]] ;

製品のインストール前、またはインストール ダイアログ ボックスを開いたときに、prepif プロシージャの直後に実行する内部プロシージャを指定します。このプロシージャは、最初のインストール ダイアログ ボックスを開き、インストール ダイアログ ボックスに必要な事前設定を定義するたびに実行されます。prestart プロシージャは、インストール ダイアログ ボックスが定義されていない場合にも実行されます。シェル以外のインタプリタをプロシージャに対して指定することができます。

#preventdowngrade: *flag* ;

製品の下位バージョンへのダウングレードを禁止するかどうかを指定します。flag 値が 1（True）の場合、製品をダウングレードすることはできません。flag 値が 0（False）の場合、製品をダウングレードすることができます。製品の依存関係があるトランザクションでは、主製品のみがチェックされ、インストールされたほかの必要な製品はすべて、最も高いバージョンのまま残されます。

#productfamily: *prodfamilyname* ;

PIF パッケージが属する製品ファミリ名を指定します。このキーワードを使用すると、同時に同じ製品ファミリ名のすべてのパッケージを、最新の使用可能な製品バージョンに更新することができます。すべてのパッケージが単一の製品ファミリに属することがあります。

#progressarea: *row , column , dlgref[, fontname]* ;

スクリプトからの出力テキストを使用して、スクリプト実行の進捗状況を表示します。長いテキスト行はダイアログボックスのウィンドウ内で折り返されません。進捗状況エリアは、新規スクリプトが起動されるたびにリセットされます。この機能は、RPM パッケージでは使用することができません。

#progressbar: *row , column , dlgref , progbartype , flag[, fontname]* ;

製品またはコンポーネントのインストールの進捗状況、または全体的なインストールの進捗状況（*progbartype* の値を使用して指定された）を表示します。Java のダイアログボックスでは、インストールの進捗状況を視覚的に示すために Java 標準要素が使用されます。VT100 モードのダイアログボックスのプログレスバーの文字数は、40 文字です。インストールの進捗状況は、パイプ文字と割合を使用して視覚的に示されます。プログレスバーのあるダイアログボックスは、オプションの *dlgref* によって指定されます。*flag* 値によって、プログレスバーで文字列の割合 (%) をオフ（値 = 0 (False)）またはオン（値 = 1 (True)）に切り替えられます。

#progresslabel: *row , column , dlgref , proglabelfontname]* ;

現在のインストールアクション（インストール、バックアップ、削除など）、および必要に応じてインストールフェーズ、現在インストール中のファイルやコンポーネントの名前、またはインストール経過時間（*proglabelfontname* の値により指定された）についての情報を通知します。プログレスラベルのあるダイアログボックスは、オプションの *dlgref* により指定されます。

#radiobutton: *row , column , ctrlname , textID , buttongroup , resultparamlist , flag[, actionscript][, fontname] ;*

2 つ以上の値を切り替えることができます。オプション **textID** は、ボタン内に表示されるテキストを指定します。ボタングループは、ラジオ ボタンを整理するグループの名前です。**Resultparamlist** は、ラジオ ボタンをクリックすると起動されるパラメータ=値割り当てのリストです。**flag** 値は、デフォルトでラジオ ボタンをオンにするか（値 = 1（True））オフにするか（値 = 0（False））を制御します。オプション **actionscript** は、インストール ダイアログ ボックスのフィールドの属性または内容をただちに変更するスクリプトを指定します。

#replacepatches: *prodlst ;*

現在のパスと置換するパッチ パッケージのカンマ区切りのリストを指定します。リスト内のカンマの前後にスペースを含めることはできません。

例 : **#replacepatches:** *common_utilities_patch1,network_patch7 ;*

#reservedspace: *virtualfilename , diskspace ;*

指定した仮想ファイル用のディスク領域を予約します。仮想ファイルに必要なディスク領域は、**diskspace** オプションにバイト数単位で指定します。

#rmpostinit: *[postuninstallprocedure[, interpreter]] ;*

PIF 製品の削除後に実行する内部プロシージャを指定します。標準シェル以外のインタプリタをプロシージャに対して指定することができます。

#rmpreinit: *[preuninstallprocedure[, interpreter]] ;*

PIF 製品の削除前に実行する内部プロシージャを指定します。標準シェル以外のインタプリタをプロシージャに対して指定することができます。

#scrollarea_begin: *row , column , ctrlname , [textarea][, scrollmode] ;*

インストール ダイアログ ボックスのチェック ボックス ツリー内のスクロール領域の先頭位置を指定します。スクロール領域の最後は、**#scrollarea_end** キーワードによって定義されます。現在のインストール ダイアログ ボックスに、スクロール領域の項目に関する追加情報を提供するテキスト領域を定義することができます。スクロール モードの仕様は、VT100 にのみ使用されます。

#scrollarea_end;

スクロール領域の最後を指定します。

#slink: *symboliclink* , *linksource* [, *ownerID* , *groupID*] ;

ファイルをほかのファイルにリンクさせるかどうかを指定します。

#splashscreen: *GIFfile* [, *textID*] ;

GIF モードでグラフィック、および手動モードで製品のインストールを開始したときに最初に表示される（つまり、インタビューフェーズのダイアログボックスの前に表示される）任意のテキストを指定します。名前に *GIFfile* が含まれるグラフィックは、GIF 形式である必要があります。*textID* は、リソースセクションで指定されたテキスト文字列の番号です。

#supplier:[*vendorname*] ;

製品ベンダーの名前を指定します。

#sys: *platform* ;

PIF 製品を実行することができる動作環境を指定します。

#text: *textID* , *textstring* ;

インストールダイアログボックスなどに表示するテキストリソースを指定します。プロトタイプファイルでは、テキスト文字列は *textID* の番号のみで参照されます。

#textarea: *row* , *column* , *ctrlname* , *textID* [, *fontname*] ;

参照先テキストが表示されるインストールダイアログボックスの領域を指定します。テキストはテキスト領域内で折り返されます。スクロール操作がサポートされています。

#textfield: *row* , *column* , *ctrlname* , *paramname* [, *validationscript*] [, *fontname*] ;

ユーザ固有の入力フィールドの名前と位置、および入力されたテキストを格納する環境変数を指定します。製品のインストールに渡される一時応答ファイルに、入力データが書き込まれます。必要に応じて、入力を検証するシェルスクリプトを指定することができます。

#timer: *timeout* ;

製品のインストールのタイムアウト値を指定します。デフォルトのタイム値 (10 分) を超える可能性がある長時間実行スクリプトの場合は、インストールスクリプトの要求を満たす大きいタイムアウト値を指定すると便利です。

#tree_begin: *ctrlname* , *flag* ;

スクロール領域内のツリーの先頭を指定します。ツリーの最後は、**#tree_end** キーワードによって指定されます。テキスト領域を指定して、スクロール領域の項目に関する追加情報を提供することができます。この場合は、テキスト領域を現在のインストール ダイアログ ボックスで定義する必要があります。**flag** はツリーの初期形状を指定し、**False = 0** の場合はツリーは最初から折りたたまれており、**True = 1** の場合はツリーは最初から展開されています。デフォルト値は、**False (0)** です。

#tree_end: ;

スクロール領域内のツリーの末尾を指定します。

#utf8encoded: 1 ;

テキスト文字列を **UTF-8** エンコード形式で入力する必要があることを指定します。キーワードが指定されていない場合、テキスト文字列は **MBCS** コードで入力されます。

#vname: *prodname* , *version* , *flag* ;

この現在のパッケージによって置換される仮想パッケージの **prodname** を指定します。指定されたバージョンは、仮想パッケージの最も下位のバージョンです。インストール中に、バージョン番号は **>=** (以上) 演算子により検証されます。フラグによって、インストール後に元のパッケージ (または別の仮想パッケージ) が削除されるかどうかを示されます。フラグ値 **0** は、「削除しない」を、フラグ値 **1** は、「削除する」を指定します。デフォルトは **1** です。

プロパティ

以下に、PIF 製品のプロトタイプ ファイルのキーワードに関連して指定することのできるすべてのプロパティをアルファベット順に示します。

アクセス

-rwxr-xr-x などの **10** バイトのアクセス権を示す文字列を指定します。

アクション

ボタンをクリックすると実行されたアクションのタイプが示されます。このプロパティは、以下の表に示す値の 1 つにすることができます。

数値	説明テキスト / 実行されるアクション
0	インストールをキャンセルします。
1	製品をインストールします。
2	次のダイアログ ボックス。 次のダイアログ ボックスが表示されます。
3	製品を削除します。
4	前のダイアログ ボックスを表示します。
5	動的ダイアログ ボックス。スクリプトを評価して、次のインストール ダイアログ ボックスを動的に決定します。
6	インストールを完了し、ユーザ スクリプトを起動します。
7	別のインストール ダイアログ ボックスに移動し、そのダイアログ ボックスの生成元を取得します。
8	スクリプトを開始して続行します。現在のダイアログ ボックスは閉じられ、指定のスクリプトが実行され、最初のダイアログ ボックスがふたたび開始されます。
9	スクリプトを実行します。指定のスクリプトが実行され、結果が指定の結果ペインに表示されます。このペインは進捗領域またはテキスト領域にできます。

actionsript

フィールドを起動したり、非アクティブしたりするなど、インストール ダイアログ ボックスのフィールドの属性または内容を変更するスクリプトの名前を指定します。

actparamset

チェック ボックスがオンになっている場合に起動される、パラメータ = 値の割り当てのカンマ区切りリスト。

buildinformation

FINAL BUILD など、最大 1024 の英数字で構成されるテキスト文字列。テキスト文字列には、キャリッジリターン (CR) および行送り (LF) などの制御文字を使用することはできません。

buttongroup

ラジオ ボタンを整理するグループの名前を指定します。

cinstparam

コンポーネントを製品と共にインストールするか、または必要に応じてのみ（オプションのコンポーネントとして）インストールするかどうかを指定するパラメータの名前を指定します。パラメータの値が **0** の場合、コンポーネントは製品と共にインストールされません。デフォルトでは、コンポーネントはインストールされるように設定されています（パラメータ名の構文の詳細については、プロパティ **paramname** を参照してください）。

column

4 列および **10** 行の仮想グリッド内のダイアログ ボックス要素の列番号を指定します。値の範囲は **1** ～ **4** になります。

compname

PIF 製品内のインストール可能な小さいユニットであるコンポーネントの名前を指定します。名前には、最大 **128** 文字まで含めることができます。

comppath

現在のコンポーネントのファイルが保存されているインストールパスを指定します。絶対パス名または環境変数を入力することができます。環境変数の先頭の文字には、常にドル記号（\$）を指定する必要があります。**2** 番目の文字にドル記号（\$）または数字を指定することはできません。パス名は、オペレーティングシステム規則に一致する必要があります。

ctrlname

ボタンまたはラベルなど、ダイアログ ボックス要素の名前を指定します。名前には、**1** ～ **32** 文字を含めることができます。

deactparamset

チェック ボックスがオフになっている場合に起動されるパラメータ=値割り当てのカンマ区切りリストを指定します。

dependencylist

（チェック ボックス ツリーにのみ使用）チェック ボックス **ctrlnames** のカンマ区切りのリスト。このリストにスペースを含めることはできません。

dialogvalidationscript

現在のダイアログ ボックスの入力の検証に使用することができるシェルスクリプト（ユーザ提供）の名前を指定します。入力されたすべての値は、シェルパラメータとしてダイアログ ボックスの検証スクリプトに渡されます。入力したそれぞれの値とすべての値の組み合わせが正しい場合は、スクリプトによって **0** が返されます。入力した値が間違っているの場合は、スクリプトによって、プロトタイプ ファイルのリソース セクションに、一致するエラー テキストの番号が返されます。このエラー テキスト番号は、**1 ～ 255** の範囲内である必要があります。

dlgref

製品インストール処理の次のダイアログ ボックスの名前を指定します。名前には、**1 ～ 32** 文字を含めることができます。特殊な方法では、シェルスクリプトをダイアログ ボックスの参照として指定することが可能です。このシェルスクリプトによって、ダイアログ ボックスの名前が標準出力に印刷されます。

externalpostinstallprocedure

現在の PIF 製品ファイルに含まれない、インストール後のプロシージャの名前を指定します。ターゲット コンピュータには特定のプロシージャがない場合があるため、外部プロシージャは慎重に使用してください。

externalpostuninstallprocedure

現在の PIF 製品ファイルに含まれない、アンインストール後のプロシージャの名前を指定します。ターゲット コンピュータには特定のプロシージャがない場合があるため、外部プロシージャは慎重に使用してください。

externalpreinstallprocedure

現在の PIF 製品ファイルに含まれない、インストール前のプロシージャの名前を指定します。ターゲット コンピュータには特定のプロシージャがない場合があるため、外部プロシージャは慎重に使用してください。

externalpreuninstallprocedure

現在の PIF 製品ファイルに含まれない、アンインストール前のプロシージャの名前を指定します。ターゲット コンピュータには特定のプロシージャがない場合があるため、外部プロシージャは慎重に使用してください。

fieldwithfocus

新しいインストール ダイアログ ボックスが開始されたときにフォーカスが設定されるダイアログ ボックス要素（フィールド）の名前を指定します。

filename

インストール ダイアログ ボックスに表示されるファイルの名前を指定します。 名前は、最大 255 文字で構成される文字列です。

filetype

ファイル タイプを指定します。値 1 を指定すると、標準ファイル（ファイル タイプ 1）が参照されます。 値 2 を指定すると、設定ファイル（ファイル タイプ 2）が参照されます。設定ファイルは、PIF 製品の更新中に変更されません。 値 3 を指定すると、一時ファイル（ファイル タイプ 3）が参照されます。

firstinstalldialog

PIF 製品の初期インストール用の開始ダイアログ ボックスの名前を指定します。 名前には、1 ～ 32 文字を含めることができます。

firstupdatedialog

PIF 製品の更新インストール用の開始ダイアログ ボックスの名前を指定します。 名前には、1 ～ 32 文字を含めることができます。

flag

ブールインジケータ。 フラグの値は、0（False）または 1（True）のいずれかになります。

fontbold

フォントがインストール ダイアログ ボックスで太字で表示されるかどうかを示すフラグ。 フラグの値が 1 の場合は太字を指定し、フラグの値が 0 の場合は一般的なフォント表示を指定します。

fontcolor

フォントの色（RGB）を指定します。 フォーマットは、R,G,B です。たとえば、255,0,0 は赤を指定します。

fontitalic

フォントがインストール ダイアログ ボックスでイタリックで表示されるかどうかを示すフラグ。 フラグの値が 1 の場合はイタリックを指定し、フラグの値が 0 の場合は一般的なフォント表示を指定します。

fontlogical

論理フォントの **Java** 特有の名前を指定します。

fontname

PIF 製品内のフォントの特定の名前を指定します。

fontsize

ブールインジケータ。フラグの値は、**0 (False)** または **1 (True)** のいずれかになります。

GIFfile

GIF グラフィック形式のグラフィックのファイル名を指定します。このグラフィックは、インストール ダイアログ ボックスのグラフィック領域に表示されます。GIF ファイルの名前には、**1 ~ 255** 文字を含めることができます。

グラフィックのサイズはダイアログ ボックス全体のサイズに影響を与えるため、画面解像度が **800 x 600** ピクセルの場合は、グラフィックが約 **400 x 200** ピクセルを超えないようにしてください。

groupID

たとえば、**sys** のようなグループを識別します。groupID の値は、最大 **32** バイトまで含めることができます。

height

ダイアログ ボックスのウィンドウの高さをピクセル単位で指定します。

注：ダイアログ ボックスの全体的な高さは、ダイアログ ボックスのセクションの **#graphic** エントリで指定したグラフィックの高さによって制御されます。

icon

ボタンに表示されるアイコンが含まれているファイルの名前を指定します。アイコンは、GIF グラフィック形式で提供する必要があります。アイコンが含まれる GIF ファイルは、インストール前のコンポーネントに追加する必要があります。GIF ファイルの名前には、**1 ~ 255** 文字を含めることができます。

installtime

外部コンポーネントがインストールされる時間を指定します。値 **0** の場合、外部コンポーネントは、PIF 製品のインストール前にインストールされます。値 **1** の場合、外部コンポーネントは、PIF 製品のインストール後にインストールされます。

instparam

依存関係にある製品を PIF 製品と共にインストールするか、または必要に応じてのみ（オプション）インストールするかどうかを指定するパラメータの名前を指定します。パラメータ *instparam* の値が **0** の場合、依存している製品はインストールされません。値が **1** の場合、依存している製品は PIF 製品と共にインストールされます。依存している製品は、デフォルトでインストールされます（パラメータ名の構文の詳細については、プロパティ **paramname** を参照してください）。

interpreter

標準シェル以外のインタプリタの名前を指定します。

linksource

リンクのソース ファイルの名前を指定します。名前は、前の **#file:** エントリの *target* パラメータの値と同じである必要があります。

listparam

リスト ボックスまたはコンボ ボックスで選択する値のリストを含むパラメータの名前を指定します。

localeID

たとえば、**ENU**（英語（米国））のように、3 文字のテキスト文字列として使用される言語を識別します。サポートされている[言語識別子](#) (P. 180)の 1 つを入力します。

migrationscript

PIF 以外の製品を PIF 製品にマイグレートするためのスクリプト（ユーザ提供）の名前を指定します。スクリプトは、コンピュータに新しく PIF 製品がインストールされるときにのみ起動されます。このスクリプトは、PIF 製品を更新または再インストールするときには実行されません。スクリプトをプリインストール コンポーネントに配置する必要があります。コンピュータで PIF でないインストールが検出された場合、次の PIF パラメータ、**PIF_INSTALL_JOB=UPDATE** および **PIF_INSTALL_PATH=existing installation path** を設定することをお勧めします。

nextdialog

製品インストール処理の次のダイアログ ボックスの名前を指定します。ダイアログ ボックスの名前には、1 ～ 32 文字を含めることができます。

operator

指定した製品バージョンとの関係を定義する演算子を指定します。演算子は、`=`、`>`、`>=` のいずれかになります。

たとえば、`= 2.1.0.3` はバージョン `2.1.0.3` と同製品を意味し、`>= 2.1.0.3` は `2.1.1.0` または `2.2.0.0` などのバージョン `2.1.0.3` およびそれ以降の製品を意味します。

ownerID

たとえば、`root` のようなユーザを識別します。 `ownerID` には、最大 32 バイトまで含めることができます。

paramcomment

パラメータについて説明するテキスト。

paramname

環境変数の名前を指定します。名前はドル記号 (`$`) で始まり、ドル記号を含めて最大 64 文字で構成されます。以下の文字は、パラメータ名には使用できません。

`. ; : ¥t ¥n ¥¥ ¥" / , $ ( ) - { }` 空白

paramset

パラメータ=値形式の割り当てのカンマ区切りのリスト。たとえば、以下のようになります。

`$PARAM1=Yes,$PARAM2=x,$INSTALLDIR=/dev/null`

リスト内のカンマは、スペースで区切ることができません。対応するボタンをクリックすると、値がパラメータに追加されます。

paramtype

パラメータのタイプを指定します。使用できる値は、`0` (= 発行済み)、`1` (= プライベート)、および `2` (= グローバル) です。

paramvalue

パラメータのデフォルト値を指定します。 `Paramvalue` は、最大 255 文字で構成される文字列です。パラメータ値は、静的値と以前に定義されたパラメータの組み合わせになります。組み合わせたパラメータ内で、静的値の最後は次の文字のいずれかマークが付けられます。

`. ; : ¥t ¥n ¥¥ ¥" / , $ ( ) - { }` 空白

platform

ターゲット コンピュータの動作環境を指定します。有効なプラットフォーム識別子の 1 つを入力します。

postinstallprocedure

PIF 製品のインストール後に実行するプロシージャの名前を指定します。このプロシージャは、ユーザによりスクリプトまたは実行可能ファイルとして提供される必要があります。 **#ppath:** エントリに指定したパスに相対パス名 (0 ～ 255 文字) を入力します。ファイルは、製品インストールディレクトリまたはサブディレクトリに保存する必要があります。このようなプロシージャがない場合は、このエントリを空白のままにするか、または省略します。

postuninstallprocedure

PIF 製品の削除後に実行するプロシージャの名前を指定します。ユーザは、このプロシージャをスクリプトまたは実行可能ファイルとして提供する必要があります。 **#ppath:** エントリに指定したパスに相対パス名 (0 ～ 255 文字) を入力します。このようなプロシージャがない場合は、このエントリを空白のままにするか、または省略します。

preinstallcomponent

PIF 製品のインストールの前に使用されるすべてのテキスト、スクリプト、およびリソース ファイルが含まれるコンポーネントの名前を指定します。通常、プリインストール コンポーネントには、GIF ファイル、使用許諾契約書、およびプリインストール スクリプトなどのダイアログ ボックスのリソース ファイルが含まれています。プリインストール コンポーネントは、PIF 製品ではインストールされません。

preinstallprocedure

PIF 製品のインストール前に実行するプロシージャの名前を指定します。ユーザは、このプロシージャをスクリプトまたは実行可能ファイルとして提供する必要があります。 **#ppath:** エントリに指定したパスに相対パス名 (0 ～ 255 文字) を入力します。ファイルは、製品インストールディレクトリまたはサブディレクトリに保存する必要があります。このようなプロシージャがない場合は、このエントリを空白のままにするか、または省略します。

prestartprocedure

製品のインストール前、または、たとえば、インストール ダイアログ ボックスで必要な事前設定を定義するために、インストール ダイアログ ボックスを開く前に実行するプロシージャの名前を指定します。ユーザは、このプロシージャを事前設定コンポーネントの一部として、スクリプトまたは実行可能なものとして提供する必要があります。 **#ppath:** エントリに指定したパスに相対パス名 (0 ～ 255 文字) を入力します。このようなプロシージャがない場合は、このエントリを空白のままにするか、または省略します。

preuninstallprocedure

PIF 製品の削除前に実行するプロシージャの名前を指定します。このプロシージャは、ユーザによりスクリプトまたは実行可能ファイルとして提供される必要があります。 **#ppath:** エントリに指定したパスに相対パス名 (0 ～ 255 文字) を入力します。このようなプロシージャがない場合は、このエントリを空白のままにするか、または省略します。

prodfamilyname

CCS などの PIF 製品ファミリの名前を指定します。名前には、最大 128 文字まで含めることができますが、スペースまたはスラッシュは含めることができません。製品ファミリの名前は、大文字小文字が区別されません。

prodlist

製品名のカンマ区切りのリスト。リスト内のカンマの前後にスペースを含めることはできません。

prodname

PIF 製品の名前を指定します。名前には、最大 128 文字まで含めることができます。

proddpath

ターゲット コンピュータの PIF 製品のインストールパスを指定します。製品をインストールするインストール先のコンピュータには絶対パス名を入力します。ルート ディレクトリから始まるパス名を入力します (/)。パス名の代わりに、インストール先のコンピュータのパス名を示す環境変数の名前を入力できます。環境変数名の先頭の文字には、常にドル記号 (\$) を指定する必要があります。2 番目の文字にドル記号 (\$) または数字を指定することはできません。パス名は、オペレーティング システム規則に一致する必要があります。

progbartype

現在の製品またはコンポーネントのインストールの進捗状況を示す環境変数。適切な値は、以下のとおりです。

```
$PIF_PROGRESSBAR_PRODUCT  
$PIF_PROGRESSBAR_COMPONENT  
$PIF_PROGRESSBAR_COUNT
```

proglabtype

現在実行中のインストール アクション、インストール中のファイル、またはコンポーネント名を示す環境変数。適切な値は、以下のとおりです。

```
$PIF_PROGRESSLABEL_ACTION  
$PIF_PROGRESSLABEL_FILE  
$PIF_PROGRESSLABEL_COMPONENT  
$PIF_PROGRESSLABEL_ELAPSEDTIME  
$PIF_PROGRESSLABEL_PHASE
```

resultparam

リスト ボックスまたはコンボ ボックスから選択された値を含むパラメータの名前を指定します。

resultparamlist

ラジオ ボタンをクリックすると起動されるパラメータ=値割り当てのリスト。

row

4 列および 10 行の仮想グリッド内のダイアログ ボックス要素の行番号を示します。値の範囲は 1 ～ 10 です。

script

スクリプト ファイル名を指定します。

scrollmode

(VT100 にのみ使用) 値 0 (フレーム (ボックス) を非表示) および 3 (フレーム (ボックス) を表示) を使用して、スクロール モードを指定します。

source

PIF 製品にパッケージするソース ファイルの名前を指定します。ソース ファイル名 (0 ～ 255 文字) は、相対名にすることができます。この場合、ソース ファイルパスの先頭は、環境変数となります。この変数は、PIF 製品を作成する前に設定しておく必要があります。

symboliclink

シンボリック リンクのファイル名 (0 ～ 255 文字) を指定します。シンボリック リンクとは、パス名が含まれているファイルのことです。シェルは、シンボリック リンクを表すファイル名を検出すると、その名前を指定されたパス名に置き換えます。このため、シンボリック リンクではなく、パス名が示すファイルへのアクセスが行われます。コンピュータの別のファイル システムにあるかどうかに関係なく、任意のファイルまたはディレクトリに、シンボリック リンクを設定することができます。

ターゲット ファイルに相対的なシンボリック リンクも作成できます (可能な場合)。ターゲット リンクは、完全修飾パスを使用してではなく、シンボリック リンクに相対的に決定されるパスを使用して参照されます。利点としては、シンボリック リンクを含むインストールされたフォルダのリンクを解除せずに移動できるということです。

以下の確認事項のいずれかに従って、シンボリック リンクを入力します。

- 指定されたシンボリック リンクがすでに絶対パス名の場合は、指定されたターゲットのパスを使用して、シンボリック リンクが作成されます。
- 製品パスが絶対パスであり、コンポーネントパスが相対パス (または空) の場合、シンボリック リンクは、製品の絶対パスおよびコンポーネントの相対パスの連結に対して作成されます。製品の絶対パスが環境変数によって表される場合、同じことが適用されます。
- 指定されたコンポーネントのパスが絶対パス名の場合、シンボリック リンクは、コンポーネントの絶対パスに対して作成されます。コンポーネントの絶対パスが環境変数によって表される場合、同じことが適用されます。

target

以下の確認事項のいずれかに従って、ターゲットのパス名 (0 ～ 255 文字) を指定します。

- 指定されたファイルパスがすでに絶対パス名の場合は、指定されたパスがインストールに使用されます。
- 製品パスが絶対パスであり、コンポーネントパスが相対パス (または空) の場合、ファイルは、製品の絶対パスおよびコンポーネントの相対パスを連結に対してインストールされます。製品の絶対パスが環境変数によって表される場合、同じことが適用されます。
- 指定されたコンポーネントのパスが絶対パス名の場合、ファイルは、コンポーネントの絶対パスに対してインストールされます。コンポーネントの絶対パスが環境変数によって表される場合、同じことが適用されます。

textarea

スクロール領域内の項目の追加情報を含むテキスト領域のダイアログボックス要素の名前を指定します。テキスト領域を現在のダイアログボックスで定義する必要があります。

textID

プロトタイプ ファイルの **Resource** セクションで定義されたテキスト文字列の参照番号を示します。参照番号の値の範囲は、**0** ～ **9999** です。

textstring

インストール ダイアログ ボックスなどで使用されるテキストを指定します。**1** ～ **1024** 文字まで入力できますが、テキストにはキャリッジリターン (CR) および行送り (LF) などの制御文字は使用できません。

timeout

長時間実行スクリプトに使用されるタイムアウト値を示します。この値は、秒単位で入力します。初期値は、**600** (10 分) です。

validationscript

テキスト フィールドからの入力を検証するシェル スクリプト (ユーザ提供) の名前を指定します。テキスト フィールドに入力される値は、最初の引数 (\$1) として検証スクリプトに渡されます。入力した値が有効な場合は、スクリプトによって **0** が返されます。入力した値が間違っている場合は、スクリプトによって、プロトタイプ ファイルのリソース セクションに、一致するエラー テキストの番号が返されます。このエラー テキスト番号は、**1** ～ **255** の範囲内である必要があります。

プロパティ **validationscript** は、最大 **255** 文字で構成される文字列です。

vendorname

製品ベンダーの名前を指定します。

version

PIF 製品のバージョンまたは PIF 製品のコンポーネントのバージョンを示します。バージョン番号の形式は、**M.m.b.r** です。**M** (メジャー)、**m** (マイナー)、**b** (ビルド)、および **r** (リビジョン) は、0 から 99,999 の範囲の数値です。**M**、**m**、**b**、および **r** の各値には、最大 5 桁を含めることができます (たとえば、10 または 00010 などの値を使用できます)。

width

インストーラ ダイアログ ボックスのウィンドウの幅をピクセル単位で指定します。

パッケージおよびインストーラのコマンド

コマンドラインで以下のコマンドを使用して、PIF 製品を作成および登録することができます。

pifproto

PIF 製品のプロトタイプ ファイルを作成します。

pifmk

PIF 製品ファイルを生成します。

pifself

自己インストーラ PIF 製品を生成します。

pifextract

PIF 製品ファイルを抽出します。

pifdelta

デルタ PIF 製品を作成します。

piflib

PIF ライブラリ ファイルを生成します。

pifpatch

PIF パッチ製品を自動的に生成します。

sd_registerproduct

ソフトウェア パッケージ ライブラリに PIF 製品を登録します。

以下のコマンドを使用して、PIF 製品をインストールし、ターゲット コンピュータにインストール済みの製品を管理することができます。

lsm

製品のインストール、削除、保存、クエリの実行、またはリストを行います。

pifproto - PIF 製品のプロトタイプ ファイルの作成

pifproto コマンドを使用して、PIF 製品を記述するプロトタイプ ファイルを作成することができます。プロトタイプ ファイルは、PIF 製品を生成する場合に **pifmk** コマンドによって使用されます。プロトタイプ ファイルは現在の作業ディレクトリに作成されます。プロトタイプ ファイルの名前は、以下のようになります。

prodname.platform.@prn

pifproto コマンドの実行直後に PIF 製品を作成するか、または後で作成するかを指定することができます。個々の設定ごとに、プロトタイプ ファイルを手動で変更することができます。

注： **pifproto** コマンドの実行中に、一時サブディレクトリ **.pif*** が作成されます。コマンドが正常に終了した場合、このサブディレクトリは自動的に削除されます。コマンドが失敗した場合、このサブディレクトリはディレクトリ ツリーに残されます。

このコマンドのフォーマットは、以下のとおりです。

```
pifproto -d proddir -a platform -n prodname -v version -b basedir
[-c comment] [-s vendorname] [-B buildID]
[-I preinstallprocedure]
[-i postinstallprocedure]
[-r preuninstallprocedure]
[-R postuninstallprocedure]
[-T template]
[-p | -S]
[-o]
```

-d proddir

項目が格納されている宛先ディレクトリの名前を指定します。現在の作業ディレクトリは、指定された **proddir** の外側にある必要があります。

-a platform

PIF 製品がインストールされているターゲット コンピュータのオペレーティング システム プラットフォーム名を指定します。

-n *prodname*

パッケージする PIF 製品の名前を指定します。製品名の小文字および大文字が異なると、異なる PIF 製品が作成されます。

-v *version*

作成する PIF 製品のバージョンを指定します。

-b *basedir*

PIF 製品を相対的にインストールするディレクトリ（`/usr/bin` など）または変数名（`$SDPATH` など）を指定します。

-c *comment*

製品固有のコメントを指定します。

-s *vendorname*

製品の供給元または製品ベンダーを指定します。

-B *buildID*

ビルドを指定します。*buildID* は、最大 1024 文字の英数字で構成され、キャリッジリターン（CR）、改行（LF）などの制御文字を含めることはできません。

-l *preinstallprocedure*

シェルスクリプトまたは実行可能ファイルを指定します。これらは、PIF 製品のインストール前に呼び出されます。

-i *postinstallprocedure*

シェルスクリプトまたは実行可能ファイルを指定します。これらは、PIF 製品のインストール後に呼び出されます。

-r *preuninstallprocedure*

シェルスクリプトまたは実行可能ファイルを指定します。これらは、PIF 製品が削除される前に呼び出されます。

-R *postuninstallprocedure*

シェルスクリプトまたは実行可能ファイルを指定します。これらは、PIF 製品が削除された後に呼び出されます。

-T *template*

各製品セットアップの作成に必要なフレームワークとして使用するテンプレートを指定します。テンプレートは、プロトタイプファイルとして表示されます。

-p

pifmk コマンドを内部的に実行して、**pifproto** コマンドの実行後に PIF 製品が自動的に作成されるように指定します。バイナリ ファイルのシンボルは、自動的に廃棄されます。

-S

pifmk と **pifself** コマンドを内部的に実行して、**pifproto** コマンドの実行後に自己インストール PIF 製品が自動的に作成されるように指定します。

-O

上書きを指定します。同じ名前の項目が宛先ディレクトリにすでに存在する場合は、その項目は上書きされます。

終了ステータス:

コマンド実行のステータスが表示されます。値ゼロ (0) は OK を意味し、ゼロ以外の値はエラーを示します。

例：プロトタイプ ファイルの作成

以下のコマンドを使用して、**test-pif.AIX.@prm** という名前のプロトタイプ ファイルを作成することができます。このファイルには、**/home/bin** ディレクトリにある複数のファイルが含まれています。

```
pifproto -a AIX -d /home/bin -b /usr/bin -v 1.0.0.0 -n test-pif
```

pifmk - PIF 製品を生成

pifmk コマンドでは、**pifproto** コマンドによって作成されたプロトタイプ ファイルを使用して、PIF 製品が生成されます。PIF 製品の生成には、十分なディスク領域が必要です。最小サイズは、作成する製品のサイズになります。デフォルトでは、PIF 製品ファイルは、**pifmk** コマンドによって **pifmk** の現在の作業ディレクトリ内に圧縮形式で作成されます。PIF 製品の名前は、**prodname.platform.@pif** になります。PIF 製品の格納先として、別の宛先ディレクトリを指定することができます。

注：圧縮された PIF 製品では、下位互換性が維持されません。つまり、この製品は、以前の Linux および UNIX 版 PIF SDK のインストーラを使用してインストールすることができません ([インストーラの互換性に関する考慮事項](#) (P. 80)の表を参照)。

このコマンドのフォーマットは、以下のとおりです。

Pifmk -v

-v

pifmk のバージョンを印刷します。

または

```
pifmk -f PIFprototypefile [-d proddir] [-a platform]
      [-v version] [-B buildID] [-o] [-s] [-t tracefile]
      [-c] [-e] [-A platform] [-c] [-V (40|42|43)]
```

-f *PIFprototypefile*

pifproto コマンドで作成したプロトタイプ ファイルの名前を指定します。

-d *proddir*

項目が格納されている宛先ディレクトリの名前を指定します。現在の作業ディレクトリは、指定された ***proddir*** の外側にある必要があります。

-a *platform*

PIF 製品がインストールされているターゲット コンピュータのオペレーティングシステム プラットフォーム名を指定します。

-v *version*

作成する PIF 製品のバージョンを指定します。

-B *buildID*

ビルドを指定します。***buildID*** は、最大 1024 文字の英数字で構成され、キャリッジリターン (CR)、改行 (LF) などの制御文字を含めることはできません。

-o

上書きを指定します。同じ名前の項目が宛先ディレクトリにすでに存在する場合は、その項目は上書きされます。

-s

ストリップを指定します。バイナリ ファイルからシンボルが廃棄されます。

-t *tracefile*

指定されたトレース ファイルにコマンド実行をトレースします。

-c *check only*

ファイルの存在を確認せずに、パッケージの整合性を確認します。

-e *write an empty package*

ファイルを含めずにパッケージに書き込みます。

-a *<platform>*

新しいプラットフォームを指定します。これにより、PIF プロトタイプファイルのデフォルトのプラットフォームが上書きされます。

-C *separate*

PIF と 自己インストーラを分けます。

-V *40/42/43*

PIF-SDK バージョン 4.7 互換のパッケージを作成します

終了ステータス:

コマンド実行のステータスが表示されます。値ゼロ (0) は OK を意味し、ゼロ以外の値はエラーを示します。

例：プロトタイプファイルからの PIF 製品の作成

以下のコマンドによって、プロトタイプファイル `test-pif.AIX.@prm` を使用して PIF 製品が作成されます。ビルド ID "Final build" を指定します。 `test-pif.AIX.@pif` という名前で PIF 製品を保存します。

```
pifmk -f test-pif.AIX.@prm -B "Final build"
```

pifself--自己インストール PIF 製品を生成

`pifself` コマンドでは、`pifmk` コマンドによって作成された PIF 製品ファイルを使用して、自己インストール PIF 製品が生成されます。ターゲットコンピュータでインストールプログラムを使用しなくても、PIF 製品をインストールすることができます。標準シェルによって、インストールプロシージャが呼び出されます。製品の生成には、十分な空きディスク領域が必要です。最小サイズは、パッケージする製品のサイズになります。

`-a` オプションを使用すると、製品が「任意」のプラットフォーム用に作成されている場合でも、自己インストール PIF 製品を作成できます。

-j オプションを使用すると、JRE (Java Runtime Environment) を自己インストール PIF 製品に統合できます。この JRE は、ターゲット システムでインストール ダイアログを実行するのに使用されます。

オプション-sを使用すると、自己インストール PIF 製品を、起動スクリプト、実際の PIF 製品、およびインストーラが含まれる自己インストール シェル スクリプトの 3 つのユニットに分割することができます。Software Delivery 環境で作業をしており、自己インストール機能を必要としない場合は、これらの部分に分けることをお勧めします。この場合、分割された実際の PIF 製品のみをソフトウェア パッケージライブラリに登録することができます。自己インストール機能を使用する場合は、通常、前述の 3 つのユニットが同じフォルダ内にあることを確認してから、自己インストール シェル スクリプトを実行する必要があります。

このコマンドのフォーマットは、以下のとおりです。

```
pifself -f PIFfile -d selfinstallingfile  
      [-a platform] [-j {JREpath|"default"}] [-s] [-o]
```

-f *PIFfile*

PIF 製品ファイル名を指定します。

-d *selfinstallingfile*

自己インストール PIF 製品の名前を指定します。

-a *platform*

PIF 製品がインストールされているターゲット コンピュータのオペレーティング システム プラットフォーム名を指定します。

-j {*JREpath* | "default"}

JRE (Java Runtime Environment) を指定したパスから、またはデフォルトの JRE を使用してインポートします。デフォルト JRE とは、Linux プラットフォーム用にコンパクト化された JRE バージョンです。*JREpath* の値は、`/usr/lib/jvm/jre-1.5sun` のように有効な JRE ディレクトリである必要があります。

-s

分割を指定します。-f オプションを使用して指定された PIF 製品ファイルが、自己インストール ファイルから分割されます。

-o

上書きを指定します。同じ名前の項目が宛先ディレクトリにすでに存在する場合は、その項目は上書きされます。

終了ステータス:

コマンド実行のステータスが表示されます。値ゼロ (0) は OK を意味し、ゼロ以外の値はエラーを示します。

例：自己インストール PIF 製品を作成

以下のコマンドでは、ファイル `test-pif.AIX.@pif` を使用して、自己インストール PIF 製品が作成されます。自己インストール PIF 製品を、`test-pif.AIX.sh` という名前で格納します。

```
pifself -f test-pif.AIX.@pif -d test-pif.AIX.sh
```

pifextract - PIF 製品を抽出

すでに作成済みの PIF 製品を変更するには（たとえば、製品定義の変更やファイルの追加、変更、削除など）、`pifextract` コマンドを使用することができます。変更後、`pifmk` コマンドを使用して、製品を再作成することができます。

`pifextract` コマンドを使用すると、すでに作成済みの PIF 製品全体、またはこの PIF 製品の一部を、指定した宛先ディレクトリに抽出することができます。宛先ディレクトリは、存在しないか、空である必要があります。製品を抽出するには、十分な空きディスク領域が必要です。プロトタイプファイルも、この宛先ディレクトリに追加されます。

このコマンドのフォーマットは、以下のとおりです。

```
pifextract ㉵f PIFfile ㉵d proddir [㉵l localeID ㉵L newlocaleID] [㉵p]
           [-t tracefile]
```

-f PIFfile

PIF 製品ファイル名を指定します。

-d proddir

PIF 製品を抽出する宛先ディレクトリの名前を指定します。ディレクトリ自体が存在しないか、ディレクトリにファイルが存在しない必要があります。

-l localeID

3 文字の[言語識別子](#) (P. 180)を指定します。指定した言語識別子に基づいて、PIF 製品から指定された言語固有の部分のみが抽出されます。

-L *newlocaleID*

希望の新しい言語を決定する 3 文字の[言語識別子](#) (P. 180)を指定します。

-p

ダイアログ ボックスに関連するすべてのセクション、およびプリインストール コンポーネントが一時フォルダに抽出され、PIF 製品が作成されます。新しい PIF 製品は、現在の作業ディレクトリに格納されます。新しい製品の名前は、*prodname.platform.@prf* になります。

-t *tracefile*

指定されたトレース ファイルにコマンド実行をトレースします。

終了ステータス:

コマンド実行のステータスが表示されます。値ゼロ (0) は OK を意味し、ゼロ以外の値はエラーを示します。

例：PIF 製品を抽出

以下のコマンドによって、PIF 製品 *test-pif.AIX.@pif* がディレクトリ */tmp/extractdir* に抽出されます。

```
pifextract -f test-pif.AIX.@pif -d /tmp/extractdir
```

pifdelta - 差分製品を作成

pifdelta コマンドを使用すると、同じ PIF 製品の 2 つのバージョン、*priorversionPIF* および *resultingversionPIF* の内容を比較し、差分を出力ファイル *deltaPIF* に書き込むことができます。

deltaPIF は、PIF パッケージフォーマットのインストール可能な小さいユニットです。PIF 製品全体 *resultingversionPIF* をインストールせずに、PIF 製品 *priorversionPIF* がすでにインストールされているすべてのターゲット コンピュータに、小さな *deltaPIF* ユニットのみをインストールすることができます。たとえば、*deltaPIF* には、*priorversionPIF* に含まれる、*resultingversionPIF* で使用されていないファイルをアンインストールする指示があります。また、*deltaPIF* には、*resultingversionPIF* で変更されたファイルおよび新規ファイルがすべて含まれています。

このコマンドのフォーマットは、以下のとおりです。

```
pifdelta -p priorversionPIF -v resultingversionPIF  
-d deltaPIF [-o] [-t tracefile]
```

-p *priorversionPIF*

PIF 製品が含まれるファイルの絶対パス名を指定します。この製品は、同じ PIF 製品の新バージョン *resultingversionPIF* に置き換えられます。

-v *resultingversionPIF*

あるバージョンの PIF 製品が含まれるファイルの絶対パス名を指定します。この PIF 製品が *priorversionPIF* と異なる場合、*priorversionPIF* はこの製品に置き換えられます。

-d *deltaPIF*

出力ファイルの絶対パス名を指定します。このファイルは、PIF パッケージフォーマットのインストール可能ユニットで、*priorversionPIF* と *resultingversionPIF* 間で異なるファイルのみが含まれています。

差分製品ファイル名には、パッケージ GUI で使用される命名規則を使用することをお勧めします。

prodname.platform.priorversion-resultingversion.@pif

このファイル名では、*prodname* は共通の製品名であり、*priorversion* と *resultingversion* はそれぞれ *priorversionPIF* および *resultingversionPIF* のバージョン番号です。たとえば、以下のようになります。

MERCHANT.Any.2.1.0.0-3.0.0.0.@pif

-O

上書きを指定します。同じ名前の項目が宛先ディレクトリにすでに存在する場合は、その項目は上書きされます。

-t *tracefile*

指定されたトレース ファイルにコマンド実行をトレースします。

Exit status:

コマンド実行のステータスが表示されます。値ゼロ (0) は OK を意味し、ゼロ以外の値はエラーを示します。

例：2つの製品バージョンから差分製品を作成

以下のコマンドは、製品インストールディレクトリの 2100 および 3000 という名前のサブディレクトリにある PIF 製品 **MERCHANT** のバージョンを比較し、**deltas** という名前のサブディレクトリ内に製品バージョン 2.1.0.0 と 3.0.0.0 の差分のみを含む PIF 差分製品を作成します。

```
pifdelta -p $PATHmerchant¥2100¥MERCHANT.Any.@pif
-v $PATHmerchant¥3000¥MERCHANT.Any.@pif
-d $PATHmerchant¥deltas¥MERCHANT.Any.2.1.0.0-3.0.0.0.@pif
```

piflib - PIF ライブラリ ファイルを生成

piflib コマンドを使用して、固定名 **library.dct** を持つ PIF ライブラリ ファイルを現在の作業ディレクトリに生成することができます。このライブラリ ファイルは、現在の作業ディレクトリに格納されているすべての PIF 製品を参照しています。インストーラがこのファイルを使用すると、共通の共有コンポーネントを含む PIF 製品が検出されます。ライブラリ ファイル（ディクショナリ）を生成する場合、**piflib** コマンドはシンボリック リンクをたどります。

このコマンドのフォーマットは、以下のとおりです。

```
piflib [-o] [ -d directory ] [-P]
```

-o

上書きを指定します。同じ名前の項目が宛先ディレクトリにすでに存在する場合は、その項目は上書きされます。

-d *directory*

オプションで、**library.dct** ファイルが格納されるフォルダの名前を指定します。

-P *scan PKG packages*

PKG 以外のプラットフォーム（Linux など）で PKG パッケージをスキャンします。デフォルトでは、PKG パッケージは PKG プラットフォーム（Solaris、UnixWare、または IRIX など）でのみスキャンされます。

終了ステータス:

コマンド実行のステータスが表示されます。値ゼロ（0）は OK を意味し、ゼロ以外の値はエラーを示します。

pifpatch - パッチ製品を自動的に生成

PIF 差分製品作成メカニズムに基づき、**pifpatch** コマンドによってパッチが自動的に生成されます。コマンドは、2 つのパッケージのセットを比較して、メインの製品の依存関係構造を照会します。

このコマンドのフォーマットは、以下のとおりです。

Pifpatch -v

-v

pifmpatch のバージョンを印刷します。

または

```
pifpatch -p originalpackage -f package_with_patch
        -d destinationfolder -I prestart_script
        -i postinstall_script -n anchorpatchPIF -v version
        [-c comment] [-o] [-t tracefile] [-x]
```

-p *originalpackage*

元の製品パッケージの絶対パス名を指定します。

-f *package_with_patch*

パッチを含むパッケージバージョンの絶対パス名を指定します。

-d *destinationdirectory*

パッチを含むパッケージが書き込まれる空のディレクトリの名前を指定します。

-I *prestartprocedure*

パッチのインストール前に実行するプロシージャまたはスクリプトを指定します。

-i *postinstallprocedure*

パッチのインストール後に実行するプロシージャまたはスクリプトを指定します。

-n *anchorpatchPIF*

パッチ製品の名前を指定します。

-v *version*

パッチのバージョン番号を指定します。

-c comment

コメントを指定します。

-o overwrite

パッチを上書きします。

-t tracefile

指定されたトレース ファイルにコマンド実行をトレースします。

-x extract only

必要なすべてのファイルを抽出し、PIF プロトタイプ ファイルに書き込みますが、パッケージのビルドは行いません。ビルドする前に、パッチ パッケージを変更できます。

終了ステータス:

コマンド実行のステータスが表示されます。値ゼロ (0) は OK を意味し、ゼロ以外の値はエラーを示します。

sd_registerproduct - ソフトウェア パッケージ ライブラリに PIF 製品を登録

sd_registerproduct コマンドを使用して、マネージャ システムのソフトウェア パッケージ ライブラリに、DEB、PIF、PKG、OSX、HPSD、BFF、または RPM パッケージ フォーマットのいずれかで製品を登録することができます。ファイル名が指定されている場合 (-f オプションを使用)、**sd_registerproduct** コマンドは、そのファイルのパッケージ フォーマットをチェックします。ディレクトリ名が入力されている場合 (-d オプションを使用)、パッケージ フォーマットは PKG ディレクトリ 構造と見なされます。-r オプションを使用して、DEB、PIF、PKG、OSX、HPSD、BFF、または RPM 製品の自動インストールの入力ファイルとして使用される応答 ファイルのリストを指定することができます。応答ファイルは、対応する製品と共にソフトウェア パッケージ ライブラリに登録されます。

このコマンドのフォーマットは、以下のとおりです。

```
sd_registerproduct {-f filename | -d directoryname}
                  -o remoteserver -u remoteuser
                  [-r responsefilelist] [-b] [-n] [-S] [-N package name] [-V version] [-A
platform]
```

```
sd_registerproduct -f filename -R
```

-f filename

PIF、PKG、または RPM 製品のファイル名を指定します。

-d *directoryname*

PKG 製品 (ディレクトリ フォーマット) を格納する場所を指定します。

-o *remoteserver*

ソフトウェア パッケージ ライブラリがあるマネージャ システムの名前を指定します。

-u *remoteuser*

ソフトウェア パッケージ ライブラリへの製品の登録に使用するユーザ アカウントを指定します。ユーザ アカウント *remoteuser* の形式は、以下のとおりです。

`[domain¥]user::password`

-r *responsefilelist*

カンマ区切りの応答ファイル名のリストを指定します。 リストでは、スペースが付かないカンマによって名前を区切る必要があります。

-b

システムのシャットダウン前にのみインストール可能な製品を登録します。

-R

現在のディレクトリで REGINFO 構造を作成します。この REGINFO 構造を使用して、ドラッグアンドドロップによってソフトウェア パッケージ ライブラリに製品を登録することができます。

-n

従属パッケージの登録を抑制します。

-S

ソフトウェア配信を有効にします。

-N *package name*

登録されているパッケージのパッケージ名を置き換えます。

-V *version*

登録されているパッケージのバージョンを置き換えます。

-A *platform*

登録されているパッケージのアーキテクチャを置き換えます。

終了ステータス:

コマンド実行のステータスが表示されます。値ゼロ (0) は OK を意味し、ゼロ以外の値はエラーを示します。

例：マネージャ システムで PIF 製品を登録

以下のコマンドを使用して、マネージャ システムでソフトウェア パッケージライブラリに PIF 製品 `test-pif.AIX.@pif` を登録することができます。マネージャ システムの名前は `daisy` で、指定されたユーザは `donald`、パスワードは `mypasswd` です。

```
sd_registerproduct -f /tmp/test-pif.AIX.@pif -o daisy -u donald::mypasswd
```

lsm - Linux および UNIX システムでソフトウェア インストールを管理

`lsm` コマンドを使用すると、ターゲット コンピュータ上の、さまざまな異なるパッケージング形式を持つ製品を管理することができます。これらのパッケージング形式には、以下のようなものがあります。

- PIF (CA Technologies Product Interchange Format)
- PKG (UNIX System V リリース 4 標準パッケージフォーマット)
- RPM (Red Hat パッケージ マネージャ フォーマット)
- IBM AIX パッケージング形式
- Hewlett Packard (HP-UX) パッケージング形式
- Apple Macintosh Mac OS X パッケージング形式

`lsm` コマンドでは、インストール済み製品をインストール、削除、リスト、バックアップ、確認、クエリしたり、PIF 製品ファイルをクエリしたり、インストール済み製品を更新することができます。このため、ターゲット コンピュータでルート権限が必要となります。

`lsm` コマンドの形式は、以下のとおりです。

- インストール

```
lsm -i prodfile [-r responsefile] [-s] [-F] [-V] [-R] [-p log file]
```

- 新しいインスタンスをインストールします

```
lsm -i prodfile [-I instance name] [-r responsefile] [-s] [-F] [-V] [-p log file]
```

- 削除

```
lsm -e prodname [-s] [-R]
```

- リスト

```
lsm -l [-O {pif | pkg | rpm | bff | osx | hpsd | patches}] [-g prodfamilyname] [-f filename]
```

- バックアップ

```
lsm -A prodname -d prodfile [-o]
```

- チェック

```
lsm -c prodname
```

- クエリ

```
lsm -q prodname [-l] [-ODEPLIST]
```

- パッケージのクエリ

```
lsm -Q prodfile [-l] [-ODEPLIST]
```

- 要求

```
lsm -a prodfile -r responsefile
```

- バージョンの出力

```
lsm -v
```

- 製品ファミリの更新

```
lsm -u prodfamilyname -d prodpath
```

- メッセージテキストの出力

```
lsm -M textID [-f prototypefile]
```

- エラー テキストの出力

```
lsm -E textID [param1] [param2] [param3] [param4]
```

- 内部スクリプトのエクスポート

```
lsm -x prodfile exportspec [-d directory] [-o] [-s]
```

- 設定

```
Lsm -c prodname
```

- システム情報の出力

`Lsm -L`

- すべて

`[ -t tracefile]`

`-i prodfile [-r responsefile] [-s] [-F] [-V] [-R]`

PIF、PKG、または RPM パッケージ製品をインストールします。`prodfile` には、製品のフルパス名を指定する必要があります。応答ファイルを追加して、自動インストールをカスタマイズすることができます。

`-s` オプションを使用すると、自動（サイレント）モードでインストールを実行することができます。

`-F` オプションを使用して、強制インストールを実行できます。この場合、既存製品のバックアップが失敗しても、インストールは続行されます。

`-V` オプションを使用すると、Java がインストールされている場合でもテキストモードを使用するように、インストーラに対して指示することができます。

`-R` オプションを使用すると、設定ファイルが保持されます。

`-e prodname [-s] [-R]`

指定されたインストール済み製品を削除します。

`-s` オプションを使用すると、自動（サイレント）モードで削除を実行することができます。

`-R` オプションを使用すると、システムのすべての設定ファイルが変更されません。

`-l [-O {pif | rpm | pkg | patches}] [-g prodfamilyname] [-f filename]`

すべてのインストール済み製品をリストします。

注：インストールプロセス中にリスト コマンドが使用された場合、詳しい情報は画面上に表示されます。

`-O` オプションを使用すると、指定したパッケージフォーマットの製品のみ、またはパッチ製品のみをリストすることができます。

`-g` オプションを使用すると、指定した製品ファミリに割り当てられているすべての製品をリストすることができます。

`-f` オプションを使用すると、指定したファイルをインストール済みの製品をリストすることができます。

-A *prodname* -d *prodfile* [-o]

インストール済みの PIF または PKG 製品 *prodname* のバックアップファイル *prodfile* を作成します。

-o オプションを使用すると、既存の *prodfile* は上書きされます。

-c *prodname*

指定されたインストール済み製品およびすべての依存関係のある製品の整合性を確認します。つまり、製品のファイルが存在するかどうか、および適切なアクセス権限、ユーザ権限および所有者権限があるかどうかを確認します。また、必須製品がインストール済みかどうかを確認します。

-C *prodname*

指定した製品を設定します。設定ウィザードを実行します（`#dlgconfig` キーワードによって、最初のダイアログ ボックスが指定されます）。

-q *prodname* [-I] [-ODEPLIST]

インストール済み製品 *prodname* に関するクエリが実行され、製品プロパティが表示されます。-ODEPLIST オプションを使用すると、指定した製品に関する依存階層（`INSTALL_ORDER`）をコンソールに印刷することができます。-I オプションを指定すると、インストールされているすべての製品ファイルを含む長いリストが提供されます。

-Q *prodfile* [-I] [-ODEPLIST]

PIF 製品ファイル *prodfile* に関するクエリを実行され、ファイルプロパティが表示されます。-ODEPLIST オプションを使用すると、指定した製品ファイルに関する依存階層（`INSTALL_ORDER`）をコンソールに印刷することができます。-I オプションを指定すると、インストールされているすべての製品ファイルを含む長いリストが提供されます。

-a *prodfile* -r *responsefile*

インストール ダイアログ ボックスが実行され、入力された値を使用して応答ファイルが作成されます。PIF 製品はインストールされません。

-v

使用されるインストーラのバージョンを出力します。

-u *prodfamilyname* -d *proddpath*

システムをスキャンして、同じ製品ファミリー名に一致する製品パッケージを検出するようインストーラに指示します。検出された製品は、最新バージョンのレベルまで更新されます。**-d** オプションを使用すると、製品の最新バージョンが格納される場所を指定することができます。製品の最新バージョンは、更新用 **CD** またはローカルフォルダを使用して提供できます。

すべての製品パッケージが最新バージョンに更新されます。宛先のパッケージよりも低いバージョンの製品パッケージが更新 **CD** に含まれている場合、更新中にその製品パッケージはスキップされます。

-M *textID1*[-*textID2*] -f *prototypefile*

指定した **PIF** 製品プロトコルファイルのリソースセクションで、単一のテキスト (*textID1* で指定) またはテキストの範囲 (*textID1*-*textID2* で指定) を参照します。**PIF** 製品の各スクリプトでこのコマンドオプションを使用すると、ローカライズ済みのテキストを印刷することができます。

-x *prodfile exportspec* [-d *directory*] [-o] [-s]

セットアッププロセスを実行し、スクリプトを実行およびテストするためのスクリプトおよび完了な環境を抽出します。ユーザがパラメータを変更できるようにするために、インタビューファイルと応答ファイルも抽出されます。

exportspec は、エクスポートするスクリプト環境を示し、次の値のいずれかを使用できます。

--exportprestart	prestart スクリプト環境をエクスポートします。
--exportpreinit	preinit スクリプト環境をエクスポートします。
--exportpostinit	postinstall スクリプト環境をエクスポートします。
--exportrmpreinit	preremove スクリプト環境をエクスポートします。
--exportallscripts	上記に一覧表示されているすべてのスクリプト環境を 1 コール以内にエクスポートします。

-d *directory* オプションは、ターゲットディレクトリを指定します。現在のディレクトリに対して、**-d pwd** または **-d .** (ドット) を入力する必要があります。

-o オプションは、ターゲット ディレクトリ内の既存のファイルが上書きされることを指定します。

-s オプションは、ユーザ操作がないサイレント モードであることを指定します。

終了ステータス:

コマンド実行のステータスが表示されます。値ゼロ (0) は OK を意味し、ゼロ以外の値はエラーを示します。

例：ローカル システムに PIF 製品をインストール

以下のコマンドによって、PIF 製品 `test-product.Any.@pif` がローカル システムにインストールされます。

```
lsm -i test-product.Any.@pif
```

例：ローカル システムから PIF 製品を削除

以下のコマンドを使用して、自動 (サイレント) モードで PIF 製品 `test-product.Any.@pif` をローカル システムから削除 (アンインストール) することができます。

```
lsm -e test-product -s
```

例：PIF 製品のバックアップ ファイルを作成

以下のコマンドによって、インストール済み PIF 製品 `test-product` のバックアップ ファイルが作成されます。バックアップ ファイルの名前は `test-product.bckp` であり、`/tmp` ディレクトリに作成されます。

```
lsm -A test-product -d /tmp/test-product.bckp
```

例：製品ファミリを更新

以下のコマンドを使用して、製品ファミリ名 `CCS` を持つパッケージがあるかどうかシステムをスキャンできます。検出されたパッケージは、ローカル フォルダ `/tmp/2.0.0.0` に格納されている製品パッケージを使用して更新されます。

```
lsm -u ccs -d /tmp/2.0.0.0/
```

例：スクリプトで lsm -M を使用

以下の例では、PIF 製品のスクリプトで **lsm -M** オプションを使用して、**Resource** セクションから数値 **100** のローカライズ済みテキストを印刷します。

```
@RESOURCE:
#text: 100 , Dieses ist ein lokalisierter Text ;
@ENDRESOURCE:
```

スクリプトは、以下のようになります。

```
#!/bin/sh
echo `lsm -M 100 2>/dev/null`
exit 0
```

例：prestart スクリプトおよびその環境をテスト目的で抽出

以下の **lsm** コマンドのバージョンは、PIF 製品 **test-product.Any.@pif** のセットアップを実行し、**prestart** スクリプトを抽出し、**prestart** スクリプトをテスト目的で実行するための完全な環境情報を含む「ラッパー」スクリプトを作成します。抽出された情報は、ユーザ操作および応答ファイル用のダイアログも含むので、テスト フェーズでパラメータを更新できます。

「ラッパー」スクリプトの名前は、インストーラ (**lsm command**) によって **start_prestart.sh** として作成されます。

コマンド出力のターゲットディレクトリは、現在の作業ディレクトリです。

```
lsm -x test-product.Any.@pif --exportprestart -d pwd
```

自己インストール PIF 製品をインストール

自己インストール PIF 製品は、シェルスクリプトとして実行されます。 コマンドオプションを使用して、ターゲット コンピュータに PIF 製品をインストールする方法を指定し、または応答ファイルのみを作成するかどうかを指定することができます。

このコマンドのフォーマットは、以下のとおりです。

```
sh selfinstallingfile [-t tracefile] [-p logfile] [-r responsefile]
                        [-s] [-F] [-x] [-a newresponsefile] [-v] [-I instance name] [-V] [-o] [-M textID]
```

selfinstallingfile

pifself コマンドなどを使用して定義された、自己インストール PIF 製品の名前。

-t tracefile

指定されたトレース ファイルにインストールをトレースします。

-p logfile

指定されたログ ファイルにインストールを記録します。

-r responsefile

指定された応答ファイルを使用して製品をインストールします。

-s

デフォルトのパラメータ値を使用して、自動（サイレント）モードで製品をインストールします。

-F

強制的に製品をインストールします。バックアップ エラーは無視されます。

-x

製品の **/tmp** ディレクトリへの抽出のみを行います。抽出された製品に使用されるファイル名は内部的に定義済みであるため、コマンドラインに自動的に表示されます。

-a newresponsefile

指定した名前で、応答ファイルのみを作成します。この応答ファイルを使用すると、後で PIF 製品をインストールすることができます。

-v

含まれているインストーラのバージョンをリストします。

-I instance name

（複数のインスタンスのインストールのための）新しい製品インスタンスの名前を指定します。

-V

VT100 ダイアログ ボックスを強制します。

-O

応答ファイルの生成時 (-a) に、既存の応答を上書きします。

-M textID1[-textID2] -f prototypefile

指定した PIF 製品プロトコル ファイルのリソース セクションで、単一のテキスト (textID1 で指定) またはテキストの範囲 (textID1-textID2 で指定) を参照します。PIF 製品の各スクリプトでこのコマンド オプションを使用すると、ローカライズ済みのテキストを印刷することができます。

例：自己インストール PIF 製品を実行、およびトレース情報を記録

以下のコマンドを使用して、自動 (サイレント) モードで自己インストール PIF 製品 test-pif.AIX.sh を実行し、対応する PIF 製品をインストールすることができます。トレース情報は、mytemp フォルダの指定されたトレース ファイル test-pif.AIX_tracef に書き込まれます。

```
sh test-pif.AIX.sh -s -t /mytemp/test-pif.AIX_tracef
```

サポートされる Linux および UNIX 動作環境

PIF 製品プロトタイプ ファイルでは、プラットフォーム識別子によって、PIF 製品をインストールできるハードウェア環境および動作環境が指定されます。以下の表は、サポートされる動作環境およびそのプラットフォーム識別子の概要を示します。プラットフォーム識別子 Any は、PIF 製品をすべてのサポート対象の動作環境にインストールできることを意味します。これはこの PIF 製品がオペレーティング システムには依存していないからです。

プラットフォーム識別子	動作環境
AIX	IBM AIX コンピュータ
すべて	すべてのサポート対象の動作環境
HPUX	Hewlett-Packard コンピュータ
Irix	Silicon Graphics ワークステーション
Linux	Linux (Intel) システム

プラットフォーム識別子	動作環境
LinuxS390	IBM Linux メインフレーム
MacOSX	Apple Macintosh OS システム (PPC)
MacOSXIntel	Apple Macintosh OS システム (Intel)
MacOSXUB	Apple Macintosh OS システム (汎用バイナリ)
NCR	NCR UNIX システム
SolarisSparc	Sun Solaris (Sparc) システム
SolarisIntel	Sun Solaris (Intel) システム
TRU64	DEC/HP Alpha サーバ
Unix	UNIX または Linux 動作環境
Unixware	Intel および AMD プロセッサ用の SCO UNIX OS

言語識別子

以下の表に、ソフトウェア管理で言語を指定（たとえば、変数 *localeID* を使用して）する場合に使用することができる 3 文字の言語識別子の概要を示します。

言語識別子	言語
すべて	すべての言語 / 言語に依存
CHS	中国語（簡体字）
CHT	中国語（繁体字）
CSY	チェコ語
DAN	デンマーク語
DEU	ドイツ語
ELL	ギリシャ語
ENU	英語（アメリカ）
ESP	スペイン語

言語識別子	言語
FIN	フィンランド語
FRA	フランス語
HUN	ハンガリー語
ITA	イタリア語
JPN	日本語
KOR	韓国語
NLD	オランダ語
NOR	ノルウェー語
PLK	ポーランド語
PTB	ポルトガル語 (ブラジル)
PTG	ポルトガル語
ROM	ルーマニア語
SVE	スウェーデン語
TRK	トルコ語

テンプレート パラメータおよび pifproto コマンド スイッチ

以下の表に、PIF SDK に事前定義済みのパラメータ、および対応する pifproto コマンド スイッチを示します。

テンプレート パラメータ	pifproto コマンド スイッチ
[\$PIF_PRODUCT_PLATFORM]	-a
[\$PIF_PRODUCT_NAME]	-n
[\$PIF_PRODUCT_VERSION]	-v
[\$PIF_INSTALL_PATH]	-d
[\$PIF_COMMENT]	-c
[\$PIF_SUPPLIER]	-s

テンプレート パラメータ	pifproto コマンド スイッチ
[\$PIF_PRODUCT_BUILD]	-B-
[\$PIF_PRE_INSTALL]	-I
[\$PIF_POST_INSTALL]	-i
[\$PIF_PRE_REMOVE]	-r
[\$PIF_POST_REMOVE]	-R

第 8 章 : FAQ

以下に、PIF パッケージおよびインストーラでの作業およびインストール時に発生する FAQ を示します。問題の解決方法および作業を続行するために必要なアクションを示しています。

このセクションには、以下のトピックが含まれています。

[ログ ファイル収集ツール dsminfo \(P. 184\)](#)

[自己インストール PIF 製品は分割可能 \(P. 184\)](#)

[複数の PIF ファイルを 1 つの製品として登録 \(P. 185\)](#)

[インストーラはカスタマ システム上のディレクトリを使用 \(P. 186\)](#)

[パッケージはディレクトリにファイルを格納 \(P. 186\)](#)

[X11 エミュレーションを設定する必要がある \(P. 187\)](#)

[パッケージおよびインストーラには特定タイプの Java Runtime Environment が必要 \(P. 187\)](#)

[ログ ファイルに記録された製品インストール情報 \(P. 188\)](#)

[選択的アンインストールのサポート \(P. 188\)](#)

[サイレントインストール中の Prepif スクリプトと応答ファイルのやり取り \(P. 189\)](#)

[Unicenter Software Delivery 4.0 エージェントにインストール可能な PIF SDK バージョン 4.1 のパッケージを使用して作成された PIF 製品 \(P. 190\)](#)

[DVD ドライブのないコンピュータでのパッケージのインストール \(P. 190\)](#)

[パッケージをインストールする際に、警告メッセージが表示されます \(P. 191\)](#)

[Midnight Commander ソフトウェアがインストールされている場合、自己インストール PIF パッケージが失敗します \(P. 192\)](#)

ログ ファイル収集ツール dsminfo

CA Technologies は、Client Automation がインストールされているシステムから、診断情報を収集する dsminfo ツールを提供しています。収集されたデータは、ログ ファイル、システム情報、ディレクトリ構造、レジストリおよび環境情報を含む単一のファイルに圧縮されます。この診断ツールは、Client Automation 製品インストール メディアの「DiagnosticTools」フォルダの下にあります。

Client Automation の問題が再現可能である場合は、以下のコマンドを実行してトレース レベルを **DETAIL** に変更してください。

```
cftrace -c set -l DETAIL
```

問題を再現し、dsminfo ツールを使って診断情報を収集してください。

注：

このツールの詳細については、製品インストール メディアの「DiagnosticTools」フォルダの下にある **DSMInfoReadMe.txt** ファイルを参照してください。

dsminfo ツールは、デフォルトでは「.7z」ファイルを作成します。これらのファイルは ZIP ファイルより高い圧縮を実行できます。そのため、CA へのアップロードが容易です。

自己インストール PIF 製品は分割可能

症状

ソフトウェア配信用に、自己インストール PIF 製品および PIF 製品の両方が必要です。パッケージを 2 つ提供する必要がありますか？

解決方法

2 つのパッケージ全体を提供する必要はありません。自己インストール PIF 製品を作成するときに、**-s** オプションを指定して **pifself** コマンドを使用します。このコマンドによって、同じ宛先ディレクトリに 3 つのファイルが作成されます。

- PIF 製品を抽出する起動スクリプト
- インストーラのバイナリを含む自己インストール シェル スクリプト (ca-sm-installer)
- 実際の PIF 製品

自己インストール PIF 製品を使用する場合、シェル スクリプトを実行します。実行時に、対応する（実際の）PIF 製品を同じディレクトリに置く必要があります。

ソフトウェア配信のために PIF 製品のみが必要な場合は、PIF 製品のみを使用して登録します。

複数の PIF ファイルを 1 つの製品として登録

症状

同じ製品に属する複数の PIF 製品ファイルを保有しています。複数のファイルを 1 つのソフトウェア配信製品として登録できますか？

解決方法

複数の PIF 製品ファイルを 1 つのソフトウェア配信製品として登録するには、以下の手順に従います。

1. PIF 製品に属するすべてのファイル（PIF 製品自体、外部コンポーネントを含む PIF ファイル、言語ソースを含む PIF ファイルなど）を同じディレクトリに置きます。
2. そのディレクトリ内で、オプション -R を使用して `sd_registerproduct` コマンドを実行し、必要な管理データを作成します。
そのディレクトリ内に「`reginfo`」構造が作成されます。
3. ディレクトリをエクスプローラにドラッグ アンド ドロップして、ソフトウェア パッケージ ライブラリに製品を登録します。エクスプローラのコンテキスト メニュー [PKG、PIF または RPM パッケージの登録] は、単一の PIF 製品ファイルしか登録されないため、この特殊なケースの場合には使用しないようにしてください。

インストーラはカスタマ システム上のディレクトリを使用

症状

自己インストール PIF 製品を使用して、インストーラをインストールします。インストーラではカスタマ システムのどのディレクトリが使用されますか? またはどのディレクトリが作成されますか?

解決方法

インストーラは、ファイルを `$CASHCOMP/installer` ディレクトリに格納します。このディレクトリは、顧客システム上で作成されるか、または更新されます。

プログラム `lsn` が `/usr/bin` に追加されます。

`/opt/CA/installer` から `$CASHCOMP/installer` へのリンクが作成されます。

パッケージはディレクトリにファイルを格納

症状

自己インストール PIF 製品を使用してパッケージをインストールします。パッケージでは、開発システムのどのディレクトリにファイルがコピーされますか?

解決方法

パッケージは、ファイルを `/opt/CA/packager` ディレクトリに格納します。ファイル `etc/.profile` は、以下の環境変数を使用して拡張されます。

- `$SDPcker` は、パッケージのホーム ディレクトリを指します。
- `$PATH`、`$LD_LIBRARY_PATH` および `$MANPATH` は、パッケージディレクトリに展開されます。

また、インストーラが存在しない場合はインストールされます。

X11 エミュレーションを設定する必要がある

症状

パッケージの GUI を使用するには、X11 エミュレーションを設定する必要がありますか？

解決方法

X11 は、クライアント/サーバアーキテクチャ向けに設計されたウィンドウ操作システムです。X11 サーバは、アプリケーションを起動するシステムです。X11 クライアントには、このグラフィカルアプリケーションが表示されます。クライアントおよびサーバは、ローカル コンソールなど、同じシステム上で実行できます。

- X11 サーバ側で、X11 クライアントに対するアクセスを以下のように設定する必要があります。

ローカル システムを X11 に登録します。

```
"xhost +client_name"
```

- クライアント側では、以下のように X11 クライアント ソフトウェアがインストールされ、起動されている必要があります。

\$DISPLAY 変数を X11 クライアントのノード名に設定します。

```
DISPLAY=client_name:0; export DISPLAY
```

パッケージおよびインストーラには特定タイプの Java Runtime Environment が必要

症状

システムには、gcj Java がインストールされています。Java モードのダイアログボックスで、これを使用できますか。

解決方法

gcj Java 環境では、標準 SUN JRE 1.2 以降で提供される複数の機能がサポートされていません。たとえば、libgcj には java.awt がありません。したがって、グラフィカルなダイアログボックスモードを使用する場合、SUN で提供される本物の標準 JRE をインストールする必要があります。

ログ ファイルに記録された製品インストール情報

症状

ログ ファイルで製品インストール情報を見ることができますか？

解決方法

はい。ソフトウェア管理では、インストール ログ ファイルが提供されます。

ログ ファイルの名前は `product_name.job.log` です。このファイルは、以下の場所にあります。

```
$CASHCOMP/installer/log/product_name.job.log
```

選択的アンインストールのサポート

症状

外部コンポーネントが含まれる PIF 製品をインストールしました。次に、インストーラおよび **Software Delivery** を使用せずに別の製品をインストールしました。後でインストールした製品のソフト リンクの 1 つが、PIF 製品の外部コンポーネントを指しています。その結果、**Software Delivery** では、この製品が外部コンポーネントを使用していることが認識されていません。この外部コンポーネントを削除すると、ほかの製品が無効になるという警告が表示されますか？ また、この外部コンポーネントを削除しないようにするオプションがありますか？

解決方法

はい、PIF SDK では、コンポーネントをターゲット コンピュータから削除できないように設定することができます。プロトタイプ ファイルで `#prerm` キーワードを使用して、削除前スクリプトを製品に追加することができます。この削除前スクリプトによって、コンポーネントがほかの製品で使用されていないかどうかチェックされ、スクリプトはゼロ以外のコードを返して終了します。これにより、インストーラによって、コンピュータ上のコンポーネントが変更されません。

サイレント インストール中の Prepif スクリプトと応答ファイルのやり取り

症状

独自の応答ファイルを作成、編集し、製品のインストール前に実行される **prepif** スクリプトを作成しました。サイレント インストールを実行する場合、応答ファイルは **prepif** スクリプトに渡されますか? その場合、応答ファイルに何らかの変更があると、インストールに渡された値は上書きされますか? 応答ファイルにインストールパスを設定し、**prepif** スクリプトでデフォルトの **installdir** を設定した場合、どちらの値が使用されますか?

解決方法

prepif スクリプトは、最初に **PIF** 製品をインストールするときのみ実行されます。更新インストールで、このスクリプトは実行されません。**prepif** スクリプトの目的は、**PIF** 以外のインストールをマイグレートすることです。マイグレーション後、**PIF_INSTALL_JOB** および製品固有の **PIF** インストールパス パラメータを変更することができます。応答ファイルのパスは上書きされます。上書きできるのは、プロトタイプファイルで **#ppath** を使用して定義済みの製品固有パスのみであることに注意してください。**PIF_INSTALL_PATH** は、通知にのみ使用され、上書きすることはできません。

Unicenter Software Delivery 4.0 エージェントにインストール可能な PIF SDK バージョン 4.1 のパッケージを使用して作成された PIF 製品

症状

PIF SDK バージョン 4.1 のパッケージを使用して、PIF 製品を作成しています。ターゲット コンピュータで実行中のバージョン 4.0 の **Software Delivery** エージェントを使用して、この PIF 製品をインストールできますか？

解決方法

PIF SDK バージョン 4.1 で機能が拡張されているため、このケースでは互換性を維持することができません。PIF 製品を自己インストール PIF 製品として作成し、**Software Delivery** 製品（「コマンド」タイプ）として登録することをお勧めします。この場合、ターゲット コンピュータにあるインストーラは、インストール時に、自己インストール PIF 製品に含まれている最新のインストーラ バージョンによって自動的に更新されます。

DVD ドライブのないコンピュータでのパッケージのインストール

症状

DVD ドライブのないコンピュータにパッケージ（Linux/UNIX）をインストールします。どのようにすればよいですか？

解決方法

Linux および UNIX のパッケージまたはコマンドライン インターフェース（Linux または UNIX から Windows 上のマネージャにパッケージを登録する際に必要です）をインストールする場合、（FTP または SFTP などを使用して）**Packager** ディレクトリ構造をダウンロードする必要があります。次に、ダウンロードされた **Packager** ディレクトリ全体を含む Linux コンピュータのディレクトリから `install-sm-packager.sh` を実行します。

パッケージをインストールする際に、警告メッセージが表示されます

SuSE 11 でのみ有効

症状

パッケージのインストール、設定、またはアンインストールの手順中に、以下の警告メッセージが表示されます。

libxcb: WARNING! Program tries to unlock a connection without having acquired a lock first, which indicates a programming error. There will be no further warnings about this issue.

解決方法

この警告は無視してください。インストール、構成またはアンインストール手順への影響はありません。

Midnight Commander ソフトウェアがインストールされている場合、自己インストール PIF パッケージが失敗します

SLED11SP3X64 でのみ有効です。

症状：

自己インストール PIF パッケージの作成中に、**SM** パッケージャで以下のエラーメッセージが表示されます。

```
sh: mc: line 1: syntax error: unexpected end of line
```

```
sh: Error importing function definition for 'mc'
```

解決方法：

コンピュータに **Midnight Commander** ソフトウェアがインストールされている場合に、これが発生します。**Midnight Commander** ソフトウェアをアンインストールし、自己インストール PIF パッケージの作成を続行します。