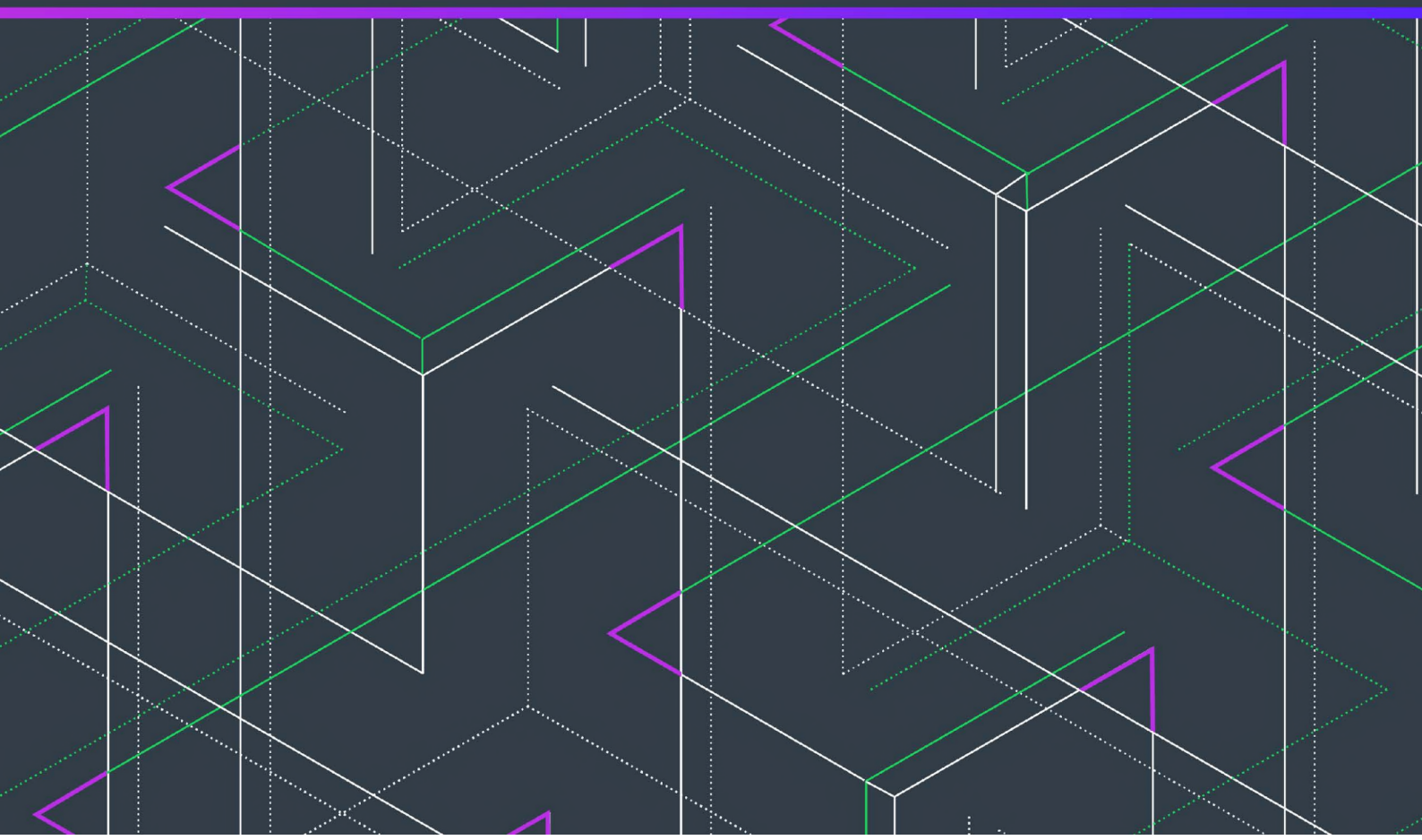
revenera

InstallShield 2025

ユーザー ガイド



法的情報

文書名: InstallShield 2025 ユーザー ガイド

部品番号: ISP-3100-UG00

製品のリリース日: 2025 年 7 月

著作権情報

Copyright © 2025 Flexera.All Rights Reserved.

この出版物には、Flexera およびそのライセンサーによって所有されている機密情報、創造的な製作物が含まれています。本出版物の一部または全部を、Flexera からの事前の書面による明示的許可なしに、使用、複製、出版、配布、表示、改変または転載することはいかなる形態または手段を問わず厳重に禁止いたします。Flexera によって書面で明示されている場合を除き、この出版物の所有は、禁反言、黙示などによっても、Flexera が所有するいかなる知的財産権の下、ライセンスまたは権利を一切付与するものではありません。

本技術およびそれに関する情報のすべての複製は、Flexera より許可されている場合に限り、著作権および所有権に関する通知を完全な形で表示しなければなりません。

知的財産

Flexera が所有する商標および特許の一覧は、https://www.flexera.com/producer/company/about/intellectual-property/ を参照してください。Flexera 製品、製品ドキュメント、およびマーケティング資料で言及されているその他すべてのブランドおよび製品名は、各社の商標または登録商標です。

(米国内向け) 制限付権利に関する表示

本ソフトウェアは商業用コンピュータ ソフトウェアです。本ソフトウェアのユーザーまたはライセンス許可対象者が米国政府の代理、部署、その他の関連機関の場合、ソフトウェアまたは技術データおよびマニュアルを含むすべての関連文書の使用、複写、複製、開示、変更、公開、または譲渡に関して、ライセンス契約または本契約の条項ならびに民生機関については連邦調達規則第 12.212 条または軍事機関については国防連邦調達規則補遺第 227.7202 条による制限が適用されます。本ソフトウェアは完全に自費で開発されたものです。その他一切の使用は禁止されています。

目次

1 InstallShield 2025 ヘルプ ライブラリ 69

InstallShield 2025 の新しい機能 **71**

InstallShield 2025 R1 の新しい機能 71

InstallShield の以前のバージョンの新しい機能 **72**

InstallShield 2024 の新しい機能 73

InstallShield 2023 の新しい機能 75

InstallShield 2022 の新しい機能 79

InstallShield 2021 の新しい機能 80

InstallShield 2020 の新しい機能 84

InstallShield 2019 の新しい機能 92

InstallShield 2018 R2 の新しい機能 100

InstallShield 2018 SP1 の新しい機能 100

InstallShield 2018 の新しい機能 103

InstallShield 2016 SP2 の新しい機能 113

InstallShield 2016 SP1 の新しい機能 115

InstallShield 2016 の新しい機能 115

InstallShield 2015 SP1 の新しい機能 127

InstallShield 2015 の新しい機能 128

InstallShield 2014 SP1 の新しい機能 143

InstallShield 2014 の新しい機能 143

InstallShield 2013 SP1 の新しい機能 156

InstallShield 2013 の新しい機能 161

InstallShield 2012 Spring SP1 の新しい機能 174

InstallShield 2012 Spring の新しい機能 176

InstallShield 2012 SP1 の新しい機能 191

InstallShield 2012 の新しい機能 191

InstallShield 2011 の新しい機能 199

InstallShield 2010 の新しい機能 219

InstallShield 2010 SP1 の新しい機能 221

InstallShield 2010 の新しい機能 223

InstallShield 2009 SP2 の新しい機能 243
InstallShield 2009 SP1 の新しい機能 244
InstallShield 2009 の新しい機能 245
InstallShield 2008 の新しい機能 261
InstallShield 12 SP1 の新しい機能 282
InstallShield 12 の新しい機能 283
ターゲット システムの要件 289
64 ビット オペレーティング システムをターゲットにする 290
64 ビットのオペレーティング システムを基本の MSI および InstallScript MSI インストールを使ってターゲットする 291
64 ビットのオペレーティング システムを 32 ビットInstallScript セットアップ ランチャーを使ってターゲットする 294
64 ビット オペレーティング システムをスイート/アドバンスト UI およびアドバンスト UI インストールでターゲットにする 295
32 ビットのオペレーティング システムを 64 ビットInstallScript セットアップ ランチャーを使ってターゲットする 296
InstallShield を管理者権限を使って、または管理者権限を持たずに起動する違い 297
32 ビットと 64 ビット システムにおけるインストールの開発およびビルドの違い 298
ヘルプの使い方 301
お問い合わせ先 305

2 スタート ガイド 307
インストールの基本 308
アプリケーション ライフサイクル 309
InstallShield の起動 310
InstallShield スタート ページ 311
プロジェクトについて 312
インストール プロジェクトの概要 312
ニーズに合ったインストール プロジェクトを選択する 313
プロジェクトの種類 314
アドバンスト UI およびスイート/アドバンスト UI プロジェクト 317
MSIX プロジェクト 318
パッケージ情報 318
パッケージ ペイロード 318
メディア 319
基本の MSI インストール プロジェクト 319
InstallScript カスタム アクションを使った、基本の MSI インストールのランタイム動作 319
DIM プロジェクト 320
InstallScript インストール プロジェクト 321
InstallScript MSI インストール プロジェクト 322
InstallScript MSI インストールのランタイム動作 323
マージ モジュール プロジェクト 324
MSI データベースおよび MSM データベース プロジェクト 325
InstallScript オブジェクト プロジェクト 325
QuickPatch プロジェクト 325
トランスフォーム プロジェクト 326

プロジェクトの使用 326
新しいプロジェクトの作成 326
プロジェクトを開く 327
ソース コード管理からプロジェクトを開く *327*
ダイレクト編集モードでパッチ作成プロパティ ファイルを開く *327*
Windows Installer パッケージを開く 328
マージ モジュールを開く 328
オブジェクト プロジェクトを開く 329
[再配布可能ファイル] ビューにマージ モジュールを配置する *329*
プロジェクトの保存 329
新しい名前と場所でプロジェクトを保存する *329*
プロジェクトを別の種類のプロジェクトに変換する 330
基本の MSI プロジェクトを InstallScript MSI プロジェクトへ変換する *331*
InstallScript MSI プロジェクトから InstallScript プロジェクトへ変換する *332*
GUID 332
デフォルトのプロジェクトの場所を変更する 333
プロジェクト要素を再利用する 334
リポジトリを使用してプロジェクトの要素を共有する 335
ネットワーク リポジトリを設定する *335*
プロジェクト テンプレート 335
プロジェクト テンプレートの作成 *336*
テンプレートを使用した新規プロジェクトの作成 *337*
プロジェクト テンプレートをリポジトリにパブリッシュする *337*
サンプル プロジェクト 338
プロジェクト アシスタント **338**
プロジェクト アシスタントを使用する 339
プロジェクト アシスタントの [他のオプション]、[他の場所]、および [ヘルプ リンク] を使用する 340
プロジェクト アシスタント内を移動する 340
インストール デザイナーを開く 341
プロジェクト アシスタントの表示または非表示 341
[アプリケーション情報] ページ 342
コントロール パネルの [プログラムの追加と削除] 342
インストールの会社名と製品名 342
[インストール要件] ページ 343
プロジェクト アシスタントでオペレーティング システムを指定する 343
インストールによる要件確認のタイミング 344
ソフトウェア要件のランタイム メッセージを変更する 344
カスタム インストール要件の作成 344
[インストール アーキテクチャ] ページ 345
プロジェクト アシスタントで機能を追加する 345
複数機能インストールを作成するかどうかを判断する 346
複数の機能を持つインストールを作成する 346
デフォルトの機能 347
機能の階層を定義する 348
[アプリケーション ファイル] ページ 348
プロジェクト アシスタントで機能にファイルを追加する 348

プロジェクト アシスタントで機能からファイルを削除する 349
固定のフォルダーの場所へファイルを追加する 349
追加の定義済みフォルダーを表示する 350
[アプリケーションのショートカット] ページ 350
ファイル拡張子 351
インストールに含まれていないファイルへのショートカットを作成する 351
プロジェクト アシスタントでデフォルトのショートカットを変更する 352
プロジェクト アシスタントでショートカットのターゲットにファイル拡張子を関連付ける 352
[アプリケーション レジストリ] ページ 353
レジストリの更新 353
プロジェクト アシスタントでレジストリ データを構成する 354
プロジェクト アシスタントでレジストリ データの値を変更する 354
機能にレジストリ データを関連付ける 355
レジストリ データで変数データ型を使用する 355
アプリケーション パス 356
[インストール インタビュー] ページ 356
プロジェクト アシスタントでインストールに使用するダイアログを指定する 356
エンドユーザーによるインストール先の変更を許可する 357
使用許諾契約書 357
インストールが選択できるインストールを作成する 358
[インストール ローカリゼーション] ページ 358
プロジェクト アシスタントから文字列データをローカライズする 359
インストールでローカライズされた文字列データの使用のされ方 360
[インストールのビルド] ページ 360
プロジェクト アシスタントからインストールをビルドする 360
プロジェクト アシスタントの完了後: 次のステップ 361
InstallShield インターフェイスを使って作業する 362
ビュー リストを表示する 363
InstallShield ユーザー インターフェイスでビューを開く 363
様々なビューで、[グループ ボックス] 領域を使って作業する 363
ツールバーの表示または非表示 365
ツールバーにボタンおよびメニューを追加する 365
ツールバーからボタンおよびメニューを削除する 366
カスタム ツールバーの作成 366
[出力] ウィンドウを固定する/取り外す 366
様々なビューで、[スクリプト エディター] ペインを使って作業する 367
スクリプト エディターでブックマークを使用する 367
スクリプト エディターでオートコンプリート機能を有効または無効にする 368
スクリプト エディターでのコードの書き込みにオートコンプリート機能を使う 369
[スクリプト エディター] 内で InstallScript 関数の呼び出しについてのヒントを有効または無効にする 370
スクリプト エディター内で InstallScript 関数について関数呼び出しのヒントを表示する 371
スクリプト エディターで構文ハイライト機能を有効または無効にする 371
スクリプト エディターで構文ハイライトの色を変更する 372
スクリプト エディターでテキストを表示するためのフォントを変更する 373
スクリプト エディターで行番号を表示または非表示にする 374
スクリプト エディター内で表示されているスクリプトの行番号へジャンプする 374

スクリプト エディターで自動インデント機能を有効または無効にする 374
スクリプト エディターでタブ幅を設定する 375
スクリプト エディターで構文の折りたたみ機能を有効または無効にする 376
スクリプト エディターで空白スペースを表示または非表示にする 376
スクリプト エディター内でテキストをドラッグ アンド ドロップする 377
[スクリプト エディタ] 内からスクリプト ファイルを印刷する 378
スクリプト エディターのキーボード ショートカット 378
InstallShield の詳細設定を構成する 379
デジタル署名のタイムスタンプ サーバーを変更する 380
Setup.exe と ISSetup.dll にストリームされるファイルの圧縮レベルを構成する 381
.cab ファイルの最大サイズを構成する 383
仮想メディア メモリの最大サイズを構成する 385
Standalone Build のポータブル実行可能ファイルの一覧を変更する 386
XML エンコード オプションのサポートを追加する 387
すべての仮想パッケージをビルドする場所を指定する 388
セットアップのデフォルト Clone_wait パラメーターを変更する 390
デジタル署名のプラットフォーム アーキテクチャを構成する 391
ランタイム中の MSI の検証を構成する 392
イベント設定コマンドの実行失敗に対するエラー表示の管理 392
リリースの配布用の仮想マシン設定を共有する 393
以前のバージョンの InstallShield からアップグレードする 395
InstallShield 2014 以前のプロジェクトをアップグレードする 395
InstallShield 2013 以前のプロジェクトをアップグレードする 399
InstallShield 2012 Spring 以前のプロジェクトをアップグレードする 399
InstallShield 2012 以前のプロジェクトをアップグレードする 403
InstallShield 2011 以前のプロジェクトをアップグレードする 404
InstallShield 2010 以前のプロジェクトをアップグレードする 407
InstallShield 2009 以前のプロジェクトをアップグレードする 411
InstallShield 2008 以前のプロジェクトをアップグレードする 422
プロジェクトを InstallShield 12 以前からアップグレードする 433
InstallShield 11.5 以前のプロジェクトをアップグレードする 436
InstallScript カスタム アクションを含む InstallShield 11.5 または以前の基本の MSI プロジェクトをアップグレードする 439
基本の MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール 441
InstallShield 11.5 以前の InstallScript MSI プロジェクトをアップグレードする 444
InstallScript MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール 447
InstallShield 11.5 以前の InstallScript プロジェクトをアップグレードする 452
InstallScript カスタム アクションを含む InstallShield 11.5 または以前の QuickPatch プロジェクトをアップグレードする 454
InstallShield 11.5 以前の InstallScript MSI プロジェクトに標準のパッチを作成する 454
InstallShield 11.5 以前の InstallScript MSI オブジェクト プロジェクトまたはこの種類のオブジェクトを含むプロジェクトをアップグレードする 454
InstallShield Express で作成したプロジェクトをアップグレードする 455
InstallShield Professional で作成したプロジェクトをアップグレードする 456
InstallShield Professional 6.x からの移行 456

InstallShield Professional 5.x からの移行 460
スクリプトの変更: 語彙変換 466
プロジェクト アシスタント ダイアログ サポートを InstallShield Professional プロジェクトからアップグレードされたプロジェクトに追加する 470
InstallShield—Windows Installer Edition で作成したプロジェクトをアップグレードする 470
プロパティ マネージャーでのアップグレード後の変更 471
InstallShield の他のエディションからアップグレードする 471
Express Edition から Professional Edition または Premier Edition にアップグレードする 476
Professional Edition から Premier Edition にアップグレードする 477
Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートする 477
InstallShield のアップデートを取得する 481
InstallShield 実行時の言語サポート 481
言語 サポートのコード ページ要件 482
ビルド マシン上に補足言語サポートをインストールする 482
サポートされているアプリケーション プログラミング言語 483

3 チュートリアル 485

InstallScript プロジェクト チュートリアル 487
ステップ 1: プロジェクトの作成、ビルドおよびテスト 487
プロジェクト アシスタントを使ったプロジェクトの作成 488
アプリケーション情報を入力する 489
インストール アーキテクチャのカスタマイズ 489
プロジェクトへファイルを追加する 490
ショートカットの作成 490
レジストリ データを構成する 490
インストール インタビューを使ってダイアログを選択する 491
インストール用の言語を選択する 491
インストールをビルドする 491
インストールを実行する 492
InstallShield インターフェイスを使って作業する 492
機能のプロパティを設定する 492
"セットアップの種類" プロパティの設定 493
コンポーネントおよびファイルリンクを作成する 493
リリースのビルド 494
リリースに名前を付ける 494
メディアタイプと一般オプションの選択 495
パスワードとサポートするプラットフォームを指定する 495
セットアップ言語と含める機能を指定する 495
メディア レイアウトとダイアログの外観を定義する 496
インターネット オプションの指定およびアプリケーションのデジタル署名 496
アップデートとポスト ビルド情報を指定する 497
設定を確認する 497
インストールのトラブルシューティング 497
ステップ 2: ショートカットとレジストリ データ 498
ショートカットの作成 498

レジストリ データを作成する 499
ステップ 3: COM サーバーの登録 500
ステップ 4: 条件とプロパティ 501
ステップ 5: スクリプトでの作業 502
ステップ 6: ユーザー インターフェイスの変更 504
ユーザー入力の処理 504
表示ダイアログを変更する 505
ダイアログ エディターを使用する 505
基本の MSI プロジェクト チュートリアル 507
ステップ 1: プロジェクトの作成、ビルドおよびテスト 507
新規基本の MSI プロジェクトの作成 508
アプリケーション情報を入力する 509
インストール要件の設定 510
インストール アーキテクチャのカスタマイズ 511
プロジェクトへファイルを追加する 511
ショートカットの作成 512
レジストリ データを構成する 512
インストール インタビューを使ってダイアログを選択する 512
インストール用の言語を選択する 513
インストールをビルドする 513
インストールを実行する 514
IDE で作業する 515
機能のプロパティを設定する 515
コンポーネントおよびファイルリンクを作成する 516
リリースのビルド 516
製品構成とリリースに名前を付ける 517
フィルター設定と言語の指定 517
メディアの種類とディスク分割のオプションを選択する 518
圧縮の設定とセットアップランチャーのオプションを指定する 518
デジタル署名とパスワード保護を追加する 519
.NET Framework のサポートを含める/詳細設定の選択 519
設定を確認する 520
インストールのトラブルシューティング 520
ステップ 2: ショートカットとレジストリ データ 521
ショートカットの作成 521
レジストリ データを作成する 522
ステップ 3: COM サーバーの登録 524
ステップ 4: 条件とプロパティ 525
ステップ 5: エンドユーザー インターフェイスを変更する 527
新しいダイアログの追加 527
ダイアログ エディターでダイアログのレイアウトを変更する 528
グローバリゼーション チュートリアル 529
プロジェクトファイルを開く 529
ターゲット言語の選択 529
言語固有の文字列エントリを編集する 530
文字列エントリの作成 531

言語別ファイルとコンポーネントを含める 532
コンポーネント インストール条件を指定する 532
文字列の翻訳 533
インストールのビルド 534
インストールの実行 535
インストールのテスト 535

4 インストールの作成 537

まず始めに 538
Windows ロゴ プログラムの要件 539
Windows Installer 入門 540
インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する 540
アプリケーションのディスクの使用量を設定する 546
NSTALLDIR と TARGETDIR の違い 546
現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ 547
非管理者パッチのインストールを準備する 548
プラットフォームとプラットフォーム スイートの設定 550

インストール情報を指定する 553
一般的なプロジェクト設定を構成する 553
InstallShield プロジェクト ファイル (.ism) を XML またはバイナリ形式で保存する 554
製品名の指定 554
製品バージョンを指定する 555
InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトにおける製品バージョン番号 555
Windows Installer ベースのプロジェクトで製品コードを設定する 557
InstallScript ベースのプロジェクトで製品コードを設定する 557
アップグレード コードを設定する 558
製品の条件を設定する 558
デフォルトの製品インストール先フォルダー (INSTALLDIR) の設定 560
ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護 561
プロジェクトで、ロックダウン環境でのアクセス許可タイプを選択する 564
Windows Installer インストールをログ記録するかどうかを指定する 565
InstallScript インストールを複数回実行する 566
"メンテナンスの有効化" 設定を構成する 567
InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い 568
InstallScript MSI インストールの InstallScript ユーザー インターフェイス タイプを指定する 574
概要情報ストリーム データを入力する 574
[概要情報ストリーム パネル] へのアクセス 575
概要情報ストリームのプロパティを設定する 575
"テンプレート概要" プロパティを使用する 576
[プログラムの追加と削除] 情報を構成する 577
Readme ファイルを指定する 578
製品のソフトウェア識別タグを含める 579
製品の Microsoft System Center Configuration Manager アプリケーション モデル データを含める 580

インストールのファイルを編成する 583

インストールをデザインする 583
アプリケーションをコンポーネントに分ける 584
アプリケーションを機能に分ける 585
パス変数を使用する 585
パス変数の種類 586
定義済みパス変数 *587*
標準パス変数 *588*
レジストリ パス変数 *589*
環境パス変数 *590*
カスタム パス変数の作成と定義 591
リリースのカスタム パス変数の値をオーバーライドする 592
プロジェクト内の絶対パスの参照をパス変数に変換する 593
パス変数の削除 594
ファイルとフォルダーを含める 594
プロジェクト ファイルに追加してターゲット システムにインストールする 595
コンテキスト メニューを使ってファイルをドラッグ アンド ドロップする 597
64 ビット ソース マシンの 64 ビット System32 フォルダーからファイルを追加する 598
新しいインストール先フォルダーの作成 599
空フォルダーの作成 599
ハードコード化されたインストール先ディレクトリを指定する 600
ターゲット システムからファイルとフォルダーを削除する 601
プロジェクトでファイルとフォルダーを管理するときのヒント 602
プロジェクトでファイルとフォルダーを検索する 603
[ファイルとフォルダー] ビューにコンポーネントを表示する 604
[ファイルとフォルダー] ビューを使ってコンポーネントを変更する 604
ファイルの設定を構成する 604
ファイルの関連付け 605
新しいファイル拡張子の追加 *606*
ファイル拡張子の削除 *606*
ファイル拡張子のプロパティを設定する *607*
コマンド動詞の指定 607
ファイル拡張子から動詞を削除する 608
新しい MIME タイプを追加する 608
MIME タイプを削除する 608
ProgID の作成 609
ProgID の削除 610
ダイナミック ファイル リンク 610
ダイナミック ファイル リンクの制限事項 *611*
ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する *612*
ダイナミック リンクの作成 *614*
ダイナミック ファイル リンクをコンポーネントに追加する *615*
ダイナミック ファイル リンクにキー ファイルを設定する *616*
ターゲット システム上でファイルとコンポーネントを上書きする 617
[ファイル] ビューで定義済みフォルダーを表示する 618
コンパニオン ファイル 619
ファイルとフォルダーのアクセス許可を構成する 620
ファイルの追加時に COM データを抽出する 621

自己登録ファイルのインストール 621
.NET アセンブリのプロパティおよび依存関係を識別する 622
依存関係の 64 ビット .NET アセンブリをスキャンする 624
.NET 依存関係スキャナー結果の確認 625
InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトでフォントをインストールする 626
Windows Fonts Folder にフォントをインストールする 626
グローバル フォント登録を有効または無効にする 627
フォント ファイルの登録を有効または無効にする 627
コンポーネントを使用する 628
セットアップ ベスト プラクティス 628
コンポーネントの作成 630
[セットアップのデザイン] ビューまたは [コンポーネント] ビューを使ってコンポーネントを作成する 630
コンポーネント ウィザードのベスト プラクティス オプションを使用する 632
コンポーネント作成におけるベスト プラクティス規則 632
InstallShield による COM サーバー コンポーネントの作成 633
InstallShield によるフォント コンポーネントの作成 634
コンポーネント ウィザードの [コンポーネント タイプ] オプションを使用する 634
COM サーバーをインストールする 635
フォントのインストール 636
他のプロジェクトへコンポーネントをエクスポートする 636
プロジェクトのコンポーネントを削除する 637
コンポーネントと機能の関連付け 638
新しいコンポーネントを機能に関連付ける 638
既存コンポーネントを機能に関連付ける 639
コンポーネントと機能の関連付けを解除する 639
データをコンポーネントに追加する 640
ファイルをコンポーネントに追加する 640
コンポーネントのファイルの [リンク] 列で値を変更する 641
コンポーネントからファイルを削除する 642
レジストリ データをコンポーネントに追加する 642
[コンポーネント] ビューでショートカットを作成する 643
静的にリンクされたコンポーネントにサブフォルダーを追加する 643
コンポーネントのキーファイル 643
コンポーネントのキーファイルを設定する 644
コンポーネントからキー ファイルをクリアする 645
コンポーネントの設定 645
コンポーネントのインストール先と機能のインストール先の違い 646
コンポーネントのファイルのインストール先フォルダーを設定する 647
スクリプトからコンポーネントのインストール先を指定する 649
大文字のディレクトリ識別子とコンポーネントのインストール先 650
アンインストール時にコンポーネントのファイルと他の関連データをアンインストールするかどうかを指定する 650
コンポーネント条件を構成する 651
再インストール中にコンポーネント条件を再評価する 652
共有ファイルの参照カウントを管理する 653
インストール前にファイルのバージョンを確認する 655
同じ名前のファイルをインストールする 655
コンポーネントの "リモート インストール" 設定を構成する 656

コンポーネントの "リモート インストール" 設定と機能の "リモートインストール" 設定の違い....657
ビルド時に COM 登録データを抽出する....657
.NET 依存関係とプロパティを確認するためのスキャン....658
.NET アプリケーション ファイルを指定する....660
.NET Installer クラスへ渡されたプロパティを読み込む....660
レジストリ リフレクションの有効化と無効化....662
コンポーネントの "共有コンポーネントのパッチ" を有効にするかどうかを指定する....663
コンポーネントの詳細設定....665
COM 登録の設定を手動で構成する....666
COM 登録の COM クラスを手動で構成する....667
COM 登録の ProgID を手動で構成する....668
COM 登録のタイプ ライブラリを手動で構成する....669
ファイル拡張子を登録する....669
アセンブリのインストール....670
アセンブリを追加する....671
アセンブリを削除する....672
ターゲット システムで .NET アセンブリ サポートをテストする....672
コンポーネントのアプリケーションパスを指定する....673
デバイスドライバー設定を構成する....674
コンポーネントをパブリッシュする....674
パブリッシュ情報を指定する....675
パブリッシュされるコンポーネントのコンポーネント ID を追加する....676
ComponentID を削除する....676
componentID へ修飾子を追加する....677
componentID から修飾子を削除する....678
修飾子の設定を構成する....678
機能の定義....679
機能の作成....680
機能の設定を構成する....681
機能のインストール先を設定する....681
機能の条件を設定する....682
機能をエンドユーザーへ表示する....684
機能の条件付き選択....685
機能の条件付き非表示....686
条件付きで実行時にスイート機能の脆弱性を設定する....687
機能のインストールを必須にする....688
機能のアドバタイズ....688
機能の "インストール レベル" 設定を構成する....689
機能の "リモート インストール" 設定を設定する....690
機能にリリース フラグを使用する....691
リリース フラグを機能に割り当てる....691
リリース フラグを削除する....691
機能の並べ替え....692
"必要な機能" 設定を使用する....692
セットアップの種類について....693
セットアップの種類を追加する....694
セットアップの種類を編集する....694
セットアップの種類の名前を変更する....695

セットアップの種類を削除する . . . 695
インストールに再配布可能ファイルを含める . . . 696
再配布可能ファイルの出荷 . . . 698
再配布可能ファイルギャラリーを管理する . . . 700
再配布可能ファイルをコンピューターにダウンロードする . . . 702
再配布可能ファイルギャラリーに InstallShield 前提条件を追加する . . . 704
再配布可能ファイルギャラリーから InstallShield 前提条件を削除する . . . 704
マージ モジュールを参照する . . . 705
マージ モジュールを参照した場合に起きること . . . 705
再配布可能ファイル ギャラリーにマージ モジュールを追加する . . . 706
再配布可能ファイルギャラリーからマージ モジュールを削除する . . . 707
InstallScript プロジェクトでオブジェクトを登録する . . . 707
InstallShield 前提条件、マージ モジュール、およびオブジェクトをプロジェクトに組み込む . . . 708
InstallShield 前提条件、マージ モジュール、オブジェクトを基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加する . . . 708
InstallShield 前提条件、マージ モジュール、およびオブジェクトを InstallScript プロジェクトに追加する . . . 710
InstallShield 前提条件をプロジェクトから削除する . . . 711
プロジェクトからマージ モジュールとオブジェクトを削除する . . . 712
InstallShield 前提条件、マージ モジュールおよびオブジェクトのファイルを決定する . . . 712
インストール プロジェクトに含まれている InstallShield 前提条件を利用する . . . 713
セットアップ前提条件と機能前提条件の違い . . . 714
基本の MSI プロジェクトで、InstallShield 前提条件を機能に関連付ける . . . 715
基本の MSI プロジェクトで、機能から InstallShield 前提条件の関連付けを解除する . . . 716
InstallShield 前提条件のインストール順を指定する . . . 717
前提条件を含むリリースを構成する . . . 718
InstallShield 前提条件を含むディレクトリを指定する . . . 719
特定の InstallShield 前提条件の実行時の場所を指定する . . . 720
リリース フラグを InstallShield 前提条件に割り当てる . . . 721
InstallShield 前提条件を含むリリースをビルドする . . . 721
InstallShield 前提条件を含むインストールの実行時の動作 . . . 722
インストールに InstallShield 前提条件が含まれていたアプリケーションをアンインストールする . . . 726
インストール プロジェクトに含まれているマージ モジュールを使用する . . . 726
マージ モジュールを含むディレクトリを指定する . . . 727
マージ モジュールのインストール先をオーバーライドする . . . 728
マージ モジュールに関するトラブルシューティング . . . 729
マージ モジュールをインストール プロジェクト内からリポジトリにパブリッシュする . . . 729
Windows Installer 再配布可能ファイルをプロジェクトに追加する . . . 730
Microsoft Windows Installer の前提条件を含める . . . 731
.NET Framework 再配布可能ファイルをプロジェクトへ追加する . . . 733
Microsoft .NET Framework および Microsoft .NET Framework 言語パックの前提条件を含める . . . 734
.NET Framework 再配布可能ファイルをプロジェクトへ追加する . . . 735
Microsoft .NET Core 前提条件を含める . . . 736
MySQL Connector ODBC 前提条件を含める . . . 737
Oracle 11g Instant Client の InstallShield 前提条件を含める . . . 738
DirectX 9.0 オブジェクトを含める . . . 738
アプリケーションの依存関係を識別する . . . 739

スタティック スキャン . . . 740
ダイナミック スキャン . . . 741
64 ビット依存関係のスキャン . . . 742
依存関係スキャナー結果の確認 . . . 742
依存関係スキャナーでファイルをフィルターする . . . 743
COM サーバーの登録 . . . 745
従来型の COM 登録 . . . 745
COM サーバーが自己登録をサポートするかどうかを判断する . . . 746
COM 情報を COM サーバーから抽出する . . . 746
COM 抽出のレジストリ変更をフィルターする . . . 748
COM サーバーの自己登録 . . . 750
自己登録メソッド . . . 751
InstallShield 自己登録 (ISSelfReg) . . . 752
レジストリ フリー COM 登録 . . . 752
インストールに Reg-Free COM ファイルを作成および変更する . . . 753
Ref-Free COM ファイルのサンプル マニフェスト ファイル . . . 754
プロジェクトに DIM を追加する . . . 754
インストール プロジェクトに DIM を統合するための適切な方法を判別する . . . 755
プロジェクトで DIM ファイルを参照する . . . 756
DIM リファレンスを機能に関連付ける . . . 757
ビルド時に発生する基本の MSI プロジェクトと DIM リファレンス間の競合を解決する . . . 757
DIM リファレンスのビルドの手順を表示する . . . 759
DIM リファレンスのインストール先のオーバーライド . . . 759
DIM リファレンスからのカスタム アクションとダイアログをスケジュールする . . . 760
インストール プロジェクト内から参照された DIM プロジェクトを開く . . . 760
DIM をプロジェクトにインポートする . . . 761
DIM を基本の MSI プロジェクトにインポート中、デザイン時の競合を解決する . . . 761
インストール プロジェクトで DIM の要素を識別する . . . 762
インストール プロジェクトで使用する DIM プロジェクト内のパス変数をオーバーライドする . . . 762
ターゲット システムの構成 . . . 765
ショートカットおよびプログラム フォルダーの作成 . . . 765
ショートカットの種類 . . . 766
ショートカットの作成 . . . 768
ショートカットの設定を構成する . . . 769
ショートカットのアイコンを指定する . . . 769
インターネット ショートカットの作成 . . . 771
フォルダーにショートカットを作成する . . . 771
ショートカットにアクセスできるキーボード ショートカットを指定する . . . 773
ショートカット名の変更 . . . 773
ショートカットのシェル プロパティを設定する . . . 774
Windows 8 スタート画面への初回のショートカットのピン留めを抑制する . . . 775
エンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めできるかどうかを指定する . . . 776
[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐ . . . 778
カスタム アクション シェルのプロパティを設定する . . . 780
基本の MSI プロジェクトのアンインストール ショートカットを作成する . . . 781

InstallScript と InstallScript MSI プロジェクトのアンインストール ショートカットを作成する 782
スタート画面上のデスクトップ アプリのタイルの外観を構成する 783
レジストリの編集 784
コンポーネントまたは機能によるレジストリ エントリのフィルタリング 785
[レジストリ] ビューのリフレッシュ 786
レジストリ キーの作成 787
レジストリ エントリをドラッグアンドドロップしてレジストリ キーを作成する 788
レジストリ キーの削除 789
レジストリ値の作成 790
レジストリ値データの変更 791
レジストリ値の削除 792
レジストリ フラグ 793
プロジェクト内のレジストリ エントリの検索 794
レジストリ キーのアンインストール動作を設定する 795
レジストリ値で環境変数を使用する 796
レジストリにプロパティ値を書き込む 797
レジストリ ファイルをインポートする 797
レジストリ ファイルのエクスポート 798
コンポーネントのキーパスを設定する 798
レジストリ キーのアクセス許可を構成する 799
レジストリ テーブルのプライマリ キー 800
Registry のプライマリ キーを指定する *800*
レジストリ キーに複数行の文字列値を入力する 801
ユーザーごとのインストールで HKCU の下にエントリを書き込む 802
レジストリで複数行文字列の設定または取得を行う 802
レジストリ関数の使い方 803
レジストリからデータを読み込む 804
.ini ファイル データの変更 805
.ini ファイルのリファレンスを作成する 806
既存の .ini ファイルをインポートする 807
.ini ファイルのセクションを指定する 807
.ini ファイルのキーワードを指定する 808
.ini ファイルからデータを読み込む 809
.ini ファイルからすべてのキー名を取得する 810
ODBC リソースの構成 810
ODBC リソースのインストール 811
追加の ODBC リソースを含める 811
ODBC リソースを機能に関連付ける 812
ODBC リソース属性の設定 813
環境変数を使用する 813
環境変数の設定 814
環境変数の例 815
XML ファイルの変更 816
XML ファイルの変更の概要 817
XML ファイルの変更の実行時の要件 818
XML ファイル リファレンスの作成 819

XML ファイルの場所を指定する 820
XML ファイルの変更のリファレンスを機能に関連付ける 821
ルート要素の追加 822
新しい要素の追加 822
要素を .NET 構成ファイルに追加する 823
属性を要素に追加する 824
属性の値を編集する 825
コンテンツを要素に追加する 825
XPath 式を使って、XML ファイル内の XML データを検索する 826
Windows Installer のプロパティを使用して、XML ファイルを動的に変更する 830
InstallScript テキスト置換を使用して、XML ファイルを動的に変更する 832
要素内で予約されている文字 (<、>、&、'、") を使用する 833
XML ファイルで名前空間を使用する 834
XML ファイルに名前空間のマッピングを宣言する *834*
名前空間プレフィックスを要素に追加する *835*
名前空間プレフィックスを属性に追加する *836*
名前空間プレフィックスを属性から削除する *836*
名前空間プレフィックスを要素から削除する *837*
XML ファイルから名前空間のマッピングを削除する *837*
XML ファイルに加えられるインストール時の変更をテストする 838
XML ファイルに加えられるアンインストール時の変更をテストする 839
[XML ファイルの変更] ビューから要素または XML ファイルを削除する 840
テキスト ファイルの変更 841
テキスト 変更セットを作成する 842
テキスト ファイルの変更を指定する 843
テキスト ファイルの変更が行われる順番を変更する 844
Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する 844
ANSI テキスト ファイルを開く時に使用するコード ページを指定する 847
[テキスト ファイルの変更] ビューから変更アイテムまたは変更セットを削除する 848
スケジュール タスク 848
スケジュール タスクの追加 849
Windows Installer プロパティを使って、スケジュール タスクをダイナミックに構成する 849
スケジュール タスクの削除 851
Windows サービスのインストール、制御、および構成 851
実行時に既定のユーザー アカウントを作成する 853
ユーザーごとのインストールとマシンごとのインストールの違い 854
管理者権限なしに InstallScript インストールを実行する 857
インストール動作のカスタマイズ 859
InstallScript を使用する 859
ISSetup.dll の概要 860
スクリプト ファイル 861
InstallScript ファイルの作成 *861*
InstallScript ファイルを開く *862*
スクリプト ファイルの挿入とインポート *862*
スクリプトのコンパイル *864*
スクリプトのデバッグ *864*

プリプロセッサ ステートメントを使用してスクリプトをデバッグする....865
InstallScript ファイルの名前を変更する....866
スクリプト中で文字列エントリを使用する....866
プロジェクトから InstallScript ファイルを削除する....866
スクリプト ライブラリ (.obl ファイル) を作成する....867
InstallScript ファイル (.rul および .h) をリポジトリにパブリッシュする....868
InstallScript リスト....869
リストの作成と破棄....869
リストへ要素を追加する....870
空白リストへ要素を追加する....871
現在の要素の前に要素を追加する....871
現在の要素の後に要素を追加する....872
現在の要素の前後に要素を追加する....872
リスト内にある既存の要素を変更する....873
リストから要素を削除する....874
リストの特定要素を検索する....875
リストの 1 番目と 2 番目の要素を取得する....875
リストへファイルを読み込む....876
リストのインデックスを設定する....876
リストのスキャン....877
ファイルへリストを書き込む....878
InstallScript ファイルの保存....878
システムの復元....878
プロパティの取得と設定....879
ビット フラグの使い方....881
文字列比較....882
ヌル区切り文字列の使用....882
相対パス....883
長いファイル名....883
長いファイル名と 16 ビット アプリケーション....883
長いファイル名と二重引用符....884
長いファイル名の形式....884
コンパイラで定数を定義する....884
スクリプトで Windows 定数を使用する....884
長い文字列リテラルのコーディング....885
絶対パス....885
関数のビルド....885
関数の呼び出し....886
DLL ファイル関数の呼び出し....886
DLL ファイル関数に配列を渡す....888
関数の宣言....888
関数から値を戻す....889
サポートされていない関数....890
エントリポイント関数の書き込み....890
関数ウィザードで関数の呼び出しを追加する....891
宣言済みの Windows API 関数....891
デフォルト以外の機能イベント ハンドラー関数を指定する....892
グローバル変数へのアクセス....892
初期化 (.ini) ファイル エントリのアンインストール....893

初期化 (.ini) ファイル エントリのアンインストールについての一般制限事項 *893*
AddProfString の初期化 (.ini) ファイル エントリをアンインストールする *894*
ReplaceProfString の初期化 (.ini) ファイル エントリをアンインストールする *895*
WriteProfString の初期化 (.ini) ファイル エントリをアンインストールする *895*
COM オブジェクトを使用してインストールを拡張する 896
関数呼び出しで機能やサブ機能を指定する 899
Windows API 関数を呼び出す 899
カスタム ファイルの転送操作を埋め込む 900
InstallScript で大きい数値を表現する 902
デバイス ドライバーのインストール 903
コンパイラ バージョンの確認 903
_ISCRIPT_ISDEV と _ISCRIPT_ISPRO を使用してオーサリング環境を確認する 904
別の InstallScript インストールからインストールを起動する 904
InstallScript の言語サポート 905
色化けの防止 907
カスタム アクションを使用 908
カスタム アクションウィザードを使用する 909
[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でカスタム アクションを作成する 910
カスタム アクションのクローン 910
カスタム アクションを別のプロジェクトにエクスポートする 911
カスタム アクションの設定を構成する 912
カスタム アクションの種類についての概要 912
カスタム アクションの戻り値 913
スクリプト内実行 914
カスタム アクションの動作をドキュメント化する 916
Windows ロゴ プログラムの要件を満たすカスタム アクションを作成するときのガイドライン 917
カスタム アクションをリリース フラグに基づいて条件付きで起動する 918
.msi データベースにファイルを配置し、実行時に抽出する 918
遅延/コミット/ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う 920
InstallScript カスタム アクション 921
標準 DLL ファイルの関数を呼び出す 922
Windows Installer DLL ファイル内の関数を呼び出す 923
カスタム アクションで DLL ファイル関数にパラメーターを渡す 925
マネージ アセンブリのパブリック メソッドを呼び出す 926
マネージ コード ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す *926*
マネージコード カスタム アクションの実行時要件 927
32 ビット マネージ コード カスタム アクションと 64 ビット マネージ コード カスタム アクションの違い 928
.NET Core ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す *928*
.NET Core カスタム アクションの実行時要件 929
32 ビットと 64 ビット .NET Core カスタム アクションの違い 929
アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する *930*
プロセスの強制終了カスタム アクションの呼び出し 932
PowerShell カスタム アクションの呼び出し 933
実行可能ファイルを起動する 935
Msiexec.exe を使用して 2 番目の Windows Installer インストールを起動する 938
ネスト インストール 940

ネスト インストール カスタム アクションを作成する *941*
ネストされたインストール カスタム アクションをシーケンスへ挿入する *942*
子製品を削除する *942*
インストールで MessageBoxA を呼び出す 944
ターゲット システム上のファイルを検索する 946
インストールが完了した後にアプリケーションを起動する 948
InstallScript カスタム アクション内部からインストールを終了する 948
スクリプトを通して ODBC プロパティを変更する 948
VBScript カスタム アクションで INSTALLDIR プロパティを使用する 949
アクション テキストを使う 949
アクションの説明とアクション データのテンプレートを指定する 950
進行状況ダイアログにアクションの説明を表示する 951
進行状況ダイアログにアクション データを表示する 952
アクションの説明またはアクション データのテンプレートを削除する 953
シーケンスを定義する 954
[インストール] シーケンス 954
[アドバタイズ] シーケンス 967
[管理] シーケンス 969
[ユーザー インターフェイス] シーケンス 971
[実行] シーケンス 971
アクションをシーケンスに挿入する 972
カスタム アクションを別のシーケンスへコピーする 973
シーケンスの順序を変更する 974
アクションをシーケンスから削除する 975
ロールバック カスタム アクションのシーケンス 976
.exe ファイルを起動するカスタム アクションをシーケンスする 976
DLL 内の関数を呼び出すカスタム アクションをシーケンスする 977
スクリプトを呼び出すカスタム アクションをシーケンスする 979
プロパティまたはディレクトリのプロパティを設定するカスタム アクションのシーケンス 980
2 番目の .msi パッケージを起動するカスタム アクションをシーケンスする 981
ISSetAllUsers カスタム アクション 982
サポート ファイルを使用する 983
サポート ファイルを追加する 983
ライセンス ファイルの追加 984
サポート ファイルの並べ替え 985
Disk1 フォルダーへファイルとフォルダーを追加する 985
Disk1 フォルダーからファイルまたはフォルダーを削除する 986
最後の Disk フォルダーにファイルとフォルダーを追加する 986
最後の Disk フォルダーからファイルとフォルダーを削除する 987
その他の Disk フォルダーにファイルとフォルダーを追加する 987
他の Disk フォルダーからファイルとフォルダーを削除する 988
サポート ファイルを削除する 988
エンドユーザー インターフェイスを定義する **989**
ダイアログの使い方 989
プロジェクトでダイアログを使用する 990
ダイアログ ウィザードを使用して新しいダイアログを作成する *990*

既存のユーザー アカウントを作成または設定する機能を追加する *991*
他のプロジェクトで使用するダイアログを .isd ファイルへエクスポートする *993*
.isd ファイルからダイアログをインポートする *993*
ダイアログ ファイル (.isd) をリポジトリにパブリッシュする *994*
すべてのダイアログを .rc ファイルへエクスポートする *994*
リソース .dll ファイルからダイアログをインポートする *996*
他のプロジェクトへダイアログをエクスポートする *996*
プロジェクトからダイアログを削除する *997*
カスタム ダイアログの作成と構成 *998*
ダイアログ エディターで変更を元に戻す *998*
InstallScript および InstallScript MSI プロジェクトでダイアログを使用する 999
InstallScript および InstallScript MSI インストールでダイアログを表示する *1000*
InstallScript と InstallScript MSI プロジェクトのダイアログに含まれるテキストを変更する *1001*
InstallScript と InstallScript MSI プロジェクトのダイアログに含まれるデフォルト イメージについての背景情報 *1002*
InstallScript および InstallScript MSI プロジェクトで新しいカスタム ダイアログを作成する *1002*
新規ダイアログ ウィザードを使って、InstallScript または InstallScript MSI プロジェクトに新しいカスタム ダイアログを追加する *1003*
InstallScript または InstallScript MSI プロジェクトのダイアログ レイアウトを編集する *1003*
InstallScript または InstallScript MSI プロジェクトのダイアログにコントロールを追加する 1004
コントロールのプロパティを設定する 1005
ビットマップの上部にコントロールを表示する 1006
InstallScript を使用してカスタム ダイアログを実装する *1006*
InstallScript を利用してダイアログ コントロールを処理する *1009*
InstallScript または InstallScript MSI プロジェクトでダイアログの動作を編集する *1012*
"Sd ダイアログ スタティック テキスト" フィールドに製品名、製品バージョン、または、インストール済みバージョンを含むフィールドを追加する *1012*
ダイアログで HTML コントロールを使用する *1013*
デフォルトのダイアログに戻す *1015*
リソース コンパイラとリソース リンカ *1016*
ダイアログ サンプラー *1016*
ダイアログ スキン *1017*
ダイアログ スキンの指定 1018
基本の MSI プロジェクトでダイアログを使用する 1019
基本の MSI インストール中にダイアログを表示する *1019*
基本の MSI プロジェクトで新しいダイアログを作成する *1020*
基本の MSI プロジェクトでダイアログ レイアウトを編集する *1021*
ダイアログ エディターを開く 1021
基本の MSI プロジェクトでダイアログにコントロールを追加する 1022
コントロールの設定を構成する 1022
ビットマップ上にコントロールを表示する 1023
基本の MSI プロジェクトでダイアログ レイアウトを編集する *1024*
基本の MSI ダイアログでコントロール イベントを使用する *1024*
基本の MSI ダイアログでサブスクリプションを使用する *1028*
基本の MSI ダイアログでコントロール イベントをトリガーする *1030*
ダイアログ コントロールに新しいイベントを追加する 1030
ダイアログ コントロールのイベントを並べ替える 1031
イベントをダイアログ コントロールから削除する 1031
ダイアログ ボタンからカスタム アクションを起動する *1032*

[カスタム アクションとシーケンス] ビューでエンドユーザー ダイアログの順序の表示する (基本の MSI プロジェクト) *1032*
CustomSetup ダイアログのオプション *1034*
ダイアログのテーマ *1036*
幅の広いダイアログ テーマにおける実行時の要件 1036
ダイアログ テーマのプレビュー 1036
ダイアログ テーマの選択または変更 1037
テーマをカスタム外部ダイアログに適用する 1038
ロゴまたは他のイメージを外部ダイアログを追加する 1039
提供されているテーマと対応するダイアログのサイズ 1040
Circles テーマ (大) 1041
Classic テーマ 1042
Cooperation テーマ (大) 1043
Filmstripテーマ (大) 1044
Global テーマ 1046
InstallShield Blue テーマ 1047
InstallShield Blue テーマ (大) 1048
InstallShield Silver テーマ 1049
Monitor テーマ 1050
Pastel Wheat テーマ 1051
Theater テーマ (大) 1053
右から左に記述される言語のダイアログ サポート *1054*
[ファイルを開く] ダイアログを起動する *1055*
LicenseAgreement ダイアログでエンドユーザーが EULA を最初から最後までスクロールするのを必須にする *1058*
[印刷] ボタンをダイアログに追加する *1060*
Windows Vista 以降のシステムの再起動を最小限にする *1062*
ダイアログのコントロール 1063
ビルボード コントロール *1067*
ビットマップ コントロール *1069*
[チェック ボックス] ボタン *1074*
コンボ ボックス コントロール *1079*
ダイアログ コントロール *1084*
[ディレクトリ コンボ] コントロール *1087*
[ディレクトリ リスト] コントロール *1090*
[編集フィールド] コントロール *1093*
[グループ ボックス] コントロール *1098*
ハイパーリンク コントロール *1101*
アイコン コントロール *1105*
行コントロール *1108*
リスト ボックス コントロール *1111*
リスト ビュー コントロール *1116*
マスク編集コントロール *1121*
パス編集コントロール *1123*
[進行状況バー] コントロール *1126*
[プッシュ ボタン] コントロール *1129*
[ラジオ ボタン] コントロール *1134*
[ラジオ ボタン グループ] コントロール *1137*
[スクロール可能テキスト] コントロール *1142*
[選択ツリー] コントロール *1144*

[テキスト領域] コントロール *1148*
[ボリューム コスト リスト] コントロール *1153*
[ボリューム選択コンボ] コントロール *1156*
コントロールのコピーと貼り付け *1159*
コントロールの切り取りと貼り付け *1159*
ウィザード インターフェイスを使って作業する 1159
ウィザード インターフェイスの要素 1160
スタイル、ブラシ、コントロール テーマ、および最小ウィザード ページを使ってウィザード インターフェイスをカスタマイズする 1162
カスタム スタイルをウィザード インターフェイスに定義する *1163*
テキスト スタイルをウィザード インターフェイスのテキストに適用する *1164*
カスタム ブラシをウィザード インターフェイスに定義する *1165*
ウィザード インターフェイスのヘッダー、本文、およびナビゲーション領域に背景ブラシを指定する *1166*
ウィザード ページまたは 2 番目のウィンドウのデフォルト 背景ブラシをオーバーライドする *1168*
カスタム コントロール テーマをウィザード インターフェイスに定義する *1168*
テーマをウィザード インターフェイスのコントロールに適用する *1169*
最小ウィザード ページのカスタマイズ *1171*
ウィザード インターフェイスのフォーマットを選択する 1172
アドバンスト UI またはスイート/アドバンスト UI プロジェクトで新しい空白のウィザード ページを作成する 1173
アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、定義済みのウィザード ページを追加する 1174
アドバンスト UI またはスイート/アドバンスト UI プロジェクトで新しい 2 番目のウィンドウを作成する 1175
シーケンス ウィザード ページ 1176
ウィザード ページまたは 2 番目のウィンドウのレイアウトおよび動作を編集する 1176
ウィザード ページまたは 2 番目のウィンドウにコントロールを追加する *1177*
ウィザード ページまたは 2 番目のウィンドウに含まれるコントロールのタブの順番を変更する *1178*
ウィザード インターフェイスにおけるキーボード ショートカットの指定とアンパサンド (&) の使用 *1178*
ウィザード ページでナビゲーション ボタンを使う *1180*
ウィザード ページのナビゲーション ボタンのデフォルト動作をオーバーライドする 1182
ウィザード インターフェイスのコンボ ボックスおよびリスト ボックス コントロールを設定する *1183*
ウィザード ページまたはウィンドウに含まれるコントロールの検証を構成する *1185*
ウィザード インターフェイス内の要素のアクションを構成する *1188*
ウィザード インターフェイスにおけるイメージ、テキスト ファイル、その他のオブジェクトの使用 1194
ウィザード インターフェイスにおける DPI 対応 1194
エンドユーザー インターフェイスのローカライズ 1195
複数言語をサポートするプロジェクトで文字列エントリを処理する 1196
InstallShield で文字列エントリを使用する 1197
文字列エントリを追加する 1199
文字列エントリを編集する 1200
文字列 ID のインスタンスを検索する 1201
文字列エントリの検索と置換 1202
文字列エントリを削除する 1203
ローカライズ可能な設定から文字列 ID を削除する 1203
ビルボードを表示する 1204
基本の MSI インストールでビルボードを表示する 1204
基本の MSI プロジェクトにおけるビルボード ファイルの種類 *1205*
基本の MSI プロジェクトにおけるビルボードの種類 *1205*
基本の MSI プロジェクトで使用するビルボードの種類を指定する 1208

基本の MSI プロジェクトに Adobe Flash アプリケーション ファイル ビルボードを追加する *1208*
基本の MSI プロジェクトにイメージ ビルボードを追加する *1209*
基本の MSI プロジェクトでビルボードの設定を構成する *1209*
リリースをビルドまたは起動せずにビルボードをプレビューする *1210*
基本の MSI プロジェクトでビルボードの順序を設定する *1210*
ビルボードを含む基本の MSI インストールにおける実行時の動作 *1211*
基本の MSI プロジェクトからビルボードを削除する *1212*
InstallScript および InstallScript MSI インストールでビルボードを表示する 1212
InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルおよびファイルの種類 *1212*
InstallScript または InstallScript MSI プロジェクトにおけるビルボード ファイルの名前 *1214*
InstallScript または InstallScript MSI プロジェクトに Adobe Flash アプリケーション ファイル ビルボードを追加する *1215*
InstallScript または InstallScript MSI プロジェクトにイメージ ビルボードを追加する *1216*
InstallScript または InstallScript MSI プロジェクトでコードを追加または変更してビルボードを表示する *1217*
InstallScript または InstallScript MSI プロジェクトでビルボードの順序を設定する *1217*
InstallScript または InstallScript MSI インストール中に特殊効果を持つビルボードを表示する *1218*
InstallScript または InstallScript MSI プロジェクトでビルボードを画面上の異なる場所に移動させる *1218*
InstallScript または InstallScript MSI プロジェクトでビルボードの順序を設定する *1219*
InstallScript および InstallScript MSI インストールで背景ウィンドウを表示する 1219
実行時にリスト ボックスへデータを挿入する 1220
ファイル参照ダイアログを表示する 1221
InstallScript インストールでネットワーク参照ダイアログを表示する 1221
サーバーの構成 **1223**
SQL サポートの構成 1223
SQL ログインの設定に Windows Installer プロパティを使用する 1224
新しい SQL 接続が同じ Windows Installer のプロパティを共有するかどうかを指定する *1226*
InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する 1227
新しい SQL 接続の追加 1228
SQL Server データベースのデフォルトの TCP/IP ネットワーク ライブラリを別のプロトコルで上書きする *1228*
SQL Server Express LocalDB のインスタンスに接続するときの要件 *1229*
新しい SQL スクリプトの追加 1230
SQL スクリプトの挿入およびインポート 1231
SQL Server データベースのインポートと SQL スクリプト ファイルの生成 1232
SQL スクリプト ファイルの編集 1233
SQL スクリプト ファイルのバージョン番号を指定する 1234
接続に関連付けられている複数 SQL スクリプトの実行順序を指定する 1235
デフォルトの SQL ランタイム動作をオーバーライドする 1236
SQL ランタイム エラーの処理 1238
SQL スクリプトにデータベース サーバー タイプの条件を設定 1238
Windows Installer ベースのインストールで SQL スクリプトを条件付きで起動する 1240
InstallScript インストールで SQL スクリプトを条件付きで起動する 1241
SQL スクリプトが完全 SQL サーバーに対してのみ実行されるように要求する 1242
Windows Installer プロパティを使って SQL スクリプトの文字列を動的に置換する 1242
InstallScript テキスト置換を使って SQL スクリプトの文字列を動的に置換する 1244
Windows Installer ベースのインストールで SQL サーバー側インストールを要求する 1246
InstallScript プロジェクトのサーバー側インストールを要求する 1247

SQL スクリプト ファイルをリポジトリにパブリッシュする 1247
MySQL ODBC ドライバーをインストールする 1248
アンインストール中に、SQL データベースを削除する 1249
Microsoft SQL Azure Database サーバーへの接続と SQL スクリプトの実行 1250
Oracle のインスタンスに接続して SQL スクリプトを実行する 1250
Oracle Instant Client をダウンロードして、そのための .msi パッケージを作成する 1251
Oracle Basic Instant Client および Oracle ODBC Instant Client をダウンロードして、両方の .msi パッケージ および InstallShield 前提条件を作成する 1253
Oracle 11g Instant Client をインストールする 1256
Oracle Database Server 上で Unicode 文字を含む SQL スクリプトを実行する 1257
カスタマイズした SQL スクリプトを実行して SQL Server データベースを作成するサンプル プロジェクトを作成する 1258
カスタマイズした SQL スクリプトを実行して Oracle スキーマを作成するサンプル インストールを作成する 1261
スイート/アドバンスト UI プロジェクトで、SQLLogin 定義済みウィザード ページを追加する 1263
COM+ アプリケーションとコンポーネントの管理 1265
COM+ サーバー アプリケーションの管理 1265
COM+ アプリケーション プロキシの管理 1266
サーバーとクライアント マシンをターゲットにした COM+ アプリケーションを含める 1268
インターネット インフォメーション サービス 1268
InstallShield における IIS サポートのバージョン固有情報 1269
インストールする IIS のバージョンを判別する 1270
IIS サポートの実行時要件 1270
Web サーバーで CMD コマンドが SSI #exec ディレクティブに使用されるのを許可するかどうかを指定する 1271
Web サイトの作成とアプリケーションまたは仮想ディレクトリの追加 1272
IIS Web サイトのスキャンをして、その設定を InstallShield プロジェクトにインポートする 1273
Web サイトおよびその設定を InstallShield プロジェクトにインポートするときに IIS データをフィルターする1275
ネスト仮想ディレクトリの作成 1277
TCP ポート番号とサイト番号の構成 1278
Web サイトの IIS ホスト ヘッダー名を指定する 1281
Web サイトの SSL 証明書を指定する 1281
実行時に IIS 証明書ファイルを設定する 1282
ファイルを IIS 仮想ディレクトリに追加する 1283
[IIS 構成] ビューからアプリケーションと仮想ディレクトリを削除する 1284
アプリケーション プールの追加 1284
[IIS 構成] ビューからアプリケーション プールを削除する 1285
Web サービス拡張の追加 1286
[IIS 構成] ビューから Web サービス拡張を削除する 1286
IIS サポートの機能とコンポーネントの関連付け 1287
Web サイト、アプリケーション、および仮想ディレクトリのアンインストール 1288
Web サービス拡張とアプリケーション プールのアンインストール 1289
Web サイトまたはアプリケーションの ASP.NET バージョンを設定する 1290
繰り返しまたは 1 回限りの ASP.NET 登録を行うかどうかを指定する 1291
64 ビット プラットフォーム上で IIS 6 をサポートする際の考慮 1291
Web サイト、アプリケーション、または仮想ディレクトリのアプリケーションのマッピングを定義する 1294
Web サイト、アプリケーション、または仮想ディレクトリのアプリケーションのマッピングを定義する 1295
ターゲット システムに IIS 6 以前が搭載されているか、または IIS 6 メタベースの互換性機能があるかどうかを判別する 1295

IIS の詳細設定を構成する ... 1297
Web サイト、アプリケーション、または仮想ディレクトリのカスタム エラー メッセージを定義する ... 1298
Windows Installer のプロパティを使用して、IIS 設定を動的に変更する ... 1298
InstallScript テキスト置換を使用して、IIS 設定を動的に変更する ... 1300
ターゲット マシン上で Web サービスを配布する ... 1301
Web アプリケーションのフォーム認証を有効化する ... 1302
IISROOTFOLDER サポートの追加 ... 1302
IIS_WEBSITE_NAME プロパティ ... 1303
IIS_PORT_NUMBER プロパティ ... 1303
メンテナンスおよびアンインストールのためのインストールを作成する ... 1305
アンインストールのための InstallScript インストールを作成 ... 1306
アンインストール時またはインストール時のみにスクリプト コードを実行する ... 1307
アンインストール用にログされた InstallScript 関数 ... 1308
メンテナンス/アンインストール機能 ... 1309
製品が作成したファイルの削除 ... 1309
製品が作成したレジストリデータの削除 ... 1310
トランザクション処理を使って複数パッケージ インストールを構成する ... 1313
トランザクション処理を使った複数パッケージ インストールについての概要 ... 1313
新しい連鎖 .msi パッケージをプロジェクトに追加する ... 1314
リリース フラグを連鎖 .msi パッケージに割り当てる ... 1316
プロジェクトから連鎖 .msi パッケージを削除する ... 1316
インストールのビルド、テスト、および配布 ... 1319
リリースの作成とビルド ... 1319
製品構成の使用 ... 1320
ビルドに適切なアーキテクチャ検証の種類を選択する ... 1320
リリースに関する作業 ... 1323
リリース ウィザードを使用して、リリースの作成およびビルドを行う ... 1324
[リリース] ビューを使用して、リリースの作成およびビルドを行う ... 1324
リリースの場所の設定 ... 1326
コマンドラインからのリリースのビルド ... 1326
ISCmdBld.exe を使用して、コマンドラインからリリースをビルドする ... 1327
.ini ファイルでコマンドライン ビルド パラメーターを渡す ... 1327
ISBuild.exe を使用してコマンドラインからのビルドする ... 1333
コマンドラインからの自己展開型実行可能ファイルをビルドする ... 1333
リリースの再ビルド ... 1333
クイック ビルドを実行する ... 1334
バッチ ビルドを実行する ... 1335
ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する ... 1336
ファイルの準備イベントを使って、署名されていない InstallShield ファイルに署名を行う ... 1338
ビルドエラーと警告を解決する ... 1339
コマンドラインからプロジェクトの設定を変更する ... 1340
ビルドのキャンセル ... 1340
Standalone Build ... 1341
Standalone Build をビルド マシンにインストールする ... 1341
Standalone Build 用に再配布可能ファイルをビルド マシンへ追加する ... 1344
スタンドアロン コマンドライン ビルド ... 1346
スタンドアロン オートメーション インターフェイス ... 1346
Standalone Build と InstallShield を同一のマシン上にインストールする ... 1346

Microsoft ビルド エンジン (MSBuild) *1347*
MSBuild を使用して、コマンドラインからリリースをビルドする 1355
リリースの設定を構成する 1356
配布する単一実行可能ファイルの作成 *1356*
ディスク イメージにファイルを非圧縮のまま残す *1357*
Windows Vista 以降のプラットフォームでのセットアップランチャーの必要実行レベルを指定する *1358*
InstallShield 前提条件が昇格された権限で実行されるときに、製品をアドバタイズするかどうか指定する *1360*
インストールのパスワード保護 *1362*
リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する *1362*
リリース レベルで、アドバンスト UI またはスイート/アドバンスト UI パッケージのランタイムの場所を指定する *1363*
スイート でパッケージのアンインストール順を指定する *1365*
デジタル署名とセキュリティ 1366
ビルド時にリリースとそのファイルにデジタル署名を行う *1368*
コマンドラインからビルドされたリリースにデジタル署名を行う *1369*
リリース フラグ 1370
製品構成フラグとリリース フラグの違い *1371*
リリース フラグに基づくフィルター *1372*
1 つのプロジェクトから複数の製品構成のインストールを作成する 1374
標準 Web インストールと One-Click Install の違い 1374
InstallScript プロジェクトの One-Click Install インストール *1375*
One-Click Install オブジェクト *1376*
Player オブジェクト *1377*
LastError メソッド 1377
Open メソッド 1378
Ether オブジェクト *1379*
GetLastError メソッド 1379
IsPlaying メソッド 1381
Play メソッド 1381
SetProperty メソッド 1382
One-Click Install インストールのデジタル署名 *1382*
古いプロパティとメソッドの置換 *1383*
デフォルトの Web ページ (Setup.htm) *1384*
One-Click Install インストール *1385*
Setup Player の別の呼び出し方法 *1385*
HTTPS と One-Click Install インストール *1385*
One-Click Install インストールの作成 *1386*
Setup Player を使用する 1387
起動された ClickOnce Install インストールにデータを渡す 1387
インターネット インストールをサイレントで実行する 1387
ランタイム言語の設定 1387
デバッグ モードで One-Click Install インストールを実行する 1388
InstallScript からユーザーを認証する 1388
Setup.ini 1389
[Startup] セクション *1390*
[Mif] セクション *1392*
リリースのクローンを作成する 1393
Windows アプリ パッケージの作成 1393
インストールのテストと実行 1395

インストールのテスト実行 1395
InstallShield インターフェイスからインストールを実行する 1396
サイレント モードでインストールを実行する 1396
InstallScript MSI と InstallScript のサイレント インストール *1397*
サイレント インストールの作成 *1398*
応答ファイルを作成する *1398*
応答ファイルのサイレント ヘッダー 1400
応答ファイルのアプリケーション ヘッダー 1400
応答ファイルのダイアログ シーケンス 1401
応答ファイルのダイアログ データ 1402
サンプル応答ファイル 1411
サイレント インストールの実行 *1412*
Setup.log ファイルを使用してエラーを確認する *1413*
InstallScript MSI プロジェクトの MSI サイレント モード インストール 1415
コマンドラインを使ったサイレント アンインストール (InstallScript MSI のみ) 1416
スイート インストールをコマンドラインから最小ユーザー インターフェイスを使って実行する 1416
コマンドラインからアップデートのダウンロードを行わずにスイートのサイレント インストールを実行する 1417
プロジェクトの検証 1417
インストール パッケージまたはマージ モジュールを検証する 1418
ビルド時に検証を行うかどうかを指定する 1419
検証中に実行する ICE、ISICE、ISVICE および ISBP を指定する 1419
インストール パッケージまたはマージ モジュールをオン デマンドで検証する 1420
検証結果を表示する 1420
ICE 1421
ISICE 1421
ISICE01 . . . . . *1424*
ISICE02 . . . . . *1425*
ISICE03 . . . . . *1426*
ISICE04 . . . . . *1426*
ISICE05 . . . . . *1427*
ISICE06 . . . . . *1428*
ISICE07 . . . . . *1429*
ISICE08 . . . . . *1430*
ISICE09 . . . . . *1431*
ISICE10 . . . . . *1431*
ISICE11 . . . . . *1432*
ISICE12 . . . . . *1434*
ISICE16 . . . . . *1434*
ISICE17 . . . . . *1435*
ISICE18 . . . . . *1436*
ISICE19 . . . . . *1437*
ISICE20 . . . . . *1439*
ISICE21 . . . . . *1440*
ISICE22 . . . . . *1441*
ISICE23 . . . . . *1441*
ISICE24 . . . . . *1442*
ISICE25 . . . . . *1443*

ISICE26 . . . . . *1443*
InstallShield 仮想化整合性スイート 1444
ISVICE01 . . . . . *1447*
ISVICE02 . . . . . *1447*
ISVICE03 . . . . . *1448*
ISVICE04 . . . . . *1449*
ISVICE05 . . . . . *1450*
ISVICE06 . . . . . *1450*
ISVICE07 . . . . . *1451*
ISVICE08 . . . . . *1452*
ISVICE09 . . . . . *1452*
ISVICE10 . . . . . *1453*
ISVICE11 . . . . . *1454*
ISVICE12 . . . . . *1454*
ISVICE13 . . . . . *1455*
ISVICE14 . . . . . *1456*
ISVICE15 . . . . . *1457*
ISVICE16 . . . . . *1457*
ISVICE17 . . . . . *1458*
ISVICE18 . . . . . *1459*
ISVICE19 . . . . . *1459*
InstallShield MSIX/UWP アプリ適合性スイート 1460
ISUWP01 . . . . . *1462*
ISUWP02 . . . . . *1462*
ISUWP03 . . . . . *1463*
ISUWP04 . . . . . *1463*
ISUWP05 . . . . . *1463*
ISUWP06 . . . . . *1464*
ISUWP07 . . . . . *1464*
ISUWP08 . . . . . *1465*
ISUWP09 . . . . . *1465*
ISUWP10 . . . . . *1466*
ISUWP11 . . . . . *1466*
ISUWP12 . . . . . *1467*
ISUWP13 . . . . . *1467*
ISUWP14 . . . . . *1468*
ISUWP15 . . . . . *1468*
ISUWP16 . . . . . *1468*
ISUWP17 . . . . . *1469*
ISUWP18 . . . . . *1469*
ISUWP19 . . . . . *1470*
ISUWP20 . . . . . *1470*
InstallShield ベスト プラクティス スイート 1470
ISBP01 . . . . . *1472*
ISBP02 . . . . . *1473*
ISBP03 . . . . . *1473*

ISBP04. . . . . *1474*
ISBP05. . . . . *1475*
ISBP06. . . . . *1476*
ISBP07. . . . . *1477*
ISBP08. . . . . *1477*
ISBP09. . . . . *1478*
ISBP10. . . . . *1479*
ISBP11. . . . . *1480*
ISBP12. . . . . *1480*
ISBP13. . . . . *1481*
ISBP14. . . . . *1482*
ISBP15. . . . . *1483*
ISBP16. . . . . *1483*
ISBP17. . . . . *1484*
ISBP18. . . . . *1485*
ISBP19. . . . . *1485*
ISBP20. . . . . *1486*
インストールまたはアンインストールがターゲット システムを再開するタイミングについて 1487
Windows Installer ベースのインストールのデバッグ. 1488
MSI デバッガーを開始する 1488
MSI デバッガーでブレークポイントを設定する. 1489
ブレークポイントを削除する. 1490
MSI デバッガーでプロパティの表示と設定を行う. 1490
MSI デバッガーでアクションにステップインする 1491
MSI デバッガーでアクションを順番に実行する 1491
MSI デバッガーを停止する 1492
インストールを複数のディスクまたは CD に分割する 1492
複数ディスク インストールを圧縮する 1493
ディスク分割規則. 1493
カスタム ディスク分割 1494
セットアップランチャーの作成 1495
セットアップランチャーのファイルのプロパティをカスタマイズする 1497
セットアップランチャーのファイルのアイコンを指定する 1504
インストールを配布する. 1505
フォルダーまたは FTP サイトにリリースを自動的に配布する 1505
ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する. 1506
インターネットで配布するインストールを準備する. 1508
InstallScript インストールと共に配布される再配布可能ファイル 1509
64 ビット セットアップ ランチャーのビルド 1510
Windows アプリ パッケージの配布. 1512
トランスフォームの作成. 1515
2 つの .msi パッケージを比較してトランスフォームを作成する 1516
単一の .msi パッケージを基にトランスフォームを作成する. 1516
トランスフォームの適用 1517
トランスフォームの編集 1518
既存のトランスフォーム ファイルを InstallShield トランスフォーム ウィザードで開く 1518

.msi ファイルをトランスフォームとして保存する. 1519
トランスフォームを .msi ファイルとして保存する 1519
応答トランスフォーム. 1520
応答トランスフォームの作成 1520
応答トランスフォームの変更 1521
応答トランスフォームのテスト 1521
サーバーの場所の構成. 1521
サーバーの場所の追加 1522
サーバーの場所の変更 1522
サーバーの場所の削除 1523
サーバーの場所の順番を変更する 1523

5 アドバンスト UI およびスイート/アドバンスト UI インストールの作成 1525

アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い. 1527

アドバンスト UI またはスイート/アドバンスト UI インストールで機能、パッケージ、前提条件、およびファイルを編成する 1530

パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン. 1532
.msi パッケージ、.msp パッチ、トランザクションをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する 1534
InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する 1536
アドバンスト UI およびスイート/アドバンスト UI プロジェクトの InstallScript パッケージについての特別考慮 1537
スイート/アドバンスト UI プロジェクトに InstallShield プロジェクト (.ism) をパッケージとして追加する 1539
実行可能パッケージ (.exe) をスイート/アドバンスト UI プロジェクトに追加する 1540
Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する 1541
サイドローディング Windows アプリ パッケージ (.appx| msix) をスイート/アドバンスト UI プロジェクトに追加する 1542
サイドローディング Windows アプリ バンドル (.appxbundle | .msixbundle) をスイート/アドバンスト UI プロジェクトに追加する 1544
Windows パッケージ マネージャー パッケージをスイート/アドバンスト UI プロジェクに追加する. 1545
アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージにおけるスタティック ファイルとダイナミック ファイルの違い. 1546
アドバンスト UI またはスイート/アドバンスト UI プロジェクトでダイナミック リンクを作成する 1547
アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ダイナミック リンク フォルダーのフィルターを定義する 1549
InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める 1550
アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるプライマリ パッケージと依存パッケージの違い. 1552
アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、パッケージとトランザクションのインストール順序を指定する. 1553
アドバンスト UI またはスイート/アドバンスト UI プロジェクトの設定を構成する 1553
アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージを機能に関連付ける 1554
異なるアドバンスト UI およびスイート/アドバンスト UI インストール間で、共通パッケージを共有する. 1555
アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに対してランタイムの場所を指定する 1557
アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージ操作を構成する 1558
アドバンスト UI または スイート/アドバンスト UI インストールのパッケージにカスタム フォルダー名を使用する 1560

スイート/アドバンスト UI インストールの動作をカスタマイズする 1561

スイート/アドバンスト UI インストール中に Windows 役割と機能を有効化する 1561

スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する 1564
スイート/アドバンスト UI インストールに含まれるアクションの .exe ファイルでの作業について 1565
スイート/アドバンスト UI インストールに含まれるアクションの DLL ファイルでの作業について 1566
スイート/アドバンスト UI インストールに含まれるアクションの PowerShell スクリプトでの作業について 1570
スイート/アドバンスト UI インストールに含まれるプロパティを設定するアクションでの作業について 1572
スイート/アドバンスト UI インストールに含まれる InstallScript コードを実行するアクションでの作業について 1573
スイート/アドバンスト UI インストールでマネージ コード アクションを使用する 1575
スイート/アドバンスト UI インストールで .NET Core アクションを使用する 1578
アクションをスイート/アドバンスト UI プロジェクトに追加する 1582
スイート/アドバンスト UI アクションの設定を構成する 1582
スイート/アドバンスト UI アクションをスケジュールする 1583
スイート/アドバンスト UI インストールにおけるイベントの種類 1585
アドバンスト UI またはスイート/アドバンスト UI インストールの終了条件を定義する 1587
アドバンスト UI またはスイート/アドバンスト UI インストールのインストール モードまたはメンテナンス モードをトリガする 1589
アドバンスト UI またはスイート/アドバンスト UI インストールの [プログラムの追加または削除] エントリ 1590
コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す 1591
アドバンスト UI およびスイート/アドバンスト UI 形式の式を使用して、コマンドラインを動的に構成する 1594
アドバンスト UI またはスイート/アドバンスト UI インストールのエンドユーザー インターフェイスを定義する 1596
アドバンスト UI およびスイート/アドバンスト UI インストールでハイコントラストモードをサポート 1597
アドバンスト UI またはスイート/アドバンスト UI プロジェクトの UI で機能ステータスおよびその他の機能データを参照する 1598
アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する 1600
アドバンスト UI およびスイート/アドバンスト UI プロジェクトでオブジェクト式を書いて、ターゲット システムを検索する 1602
アドバンスト UI およびスイート/アドバンスト UI インストール中に、ユーザー アカウント制御のプロンプト回数を最小限に抑える 1603
アドバンスト UI またはスイート/アドバンスト UI パッケージにおけるターゲット システムの再起動 1605
再起動が必要なアドバンスト UI またはスイート/アドバンスト UI パッケージの動作を指定する 1606
アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート 1609
[新しいアップデート利用可能] ダイアログのカスタマイズ 1615
Windows タスク スケジューラでのアプリケーション更新スケジュール タスクの管理 1616
アドバンスト UI またはスイート/アドバンスト UI インストールをパスワードで保護する 1617
アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング 1618
アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート 1623
64 ビット セットアップ EXE のビルド 1625

6 InstallShield 前提条件およびその他の再配布可能ファイルをデザインする 1629
InstallShield 前提条件を定義する 1633
InstallShield 前提条件を作成する 1634
既存の InstallShield 前提条件を変更する 1634
InstallShield 前提条件のプロパティを設定する 1634
nstallShield Windows Package Manager 前提条件の作成 1635

.prq ファイルの代替 URL を指定する 1635
InstallShield 前提条件のインストール条件を設定する 1636
InstallShield 前提条件の新しいインストール条件を追加する 1637
InstallShield 前提条件のオペレーティング システム条件を追加する 1637
InstallShield 前提条件のインストール条件を変更する 1638
InstallShield 前提条件からインストール条件を削除する 1639
InstallShield 前提条件のファイルを指定する 1639
InstallShield 前提条件をダウンロードする URL を指定する 1640
InstallShield 前提条件をインストールするためのパラメーターを指定する 1641
InstallShield 前提条件のコマンドライン パラメーターを指定する 1642
InstallShield 前提条件のインストールでターゲットマシンを再起動する 1645
InstallShield 前提条件のインストール動作を指定する 1645
InstallShield 前提条件に管理者権限が必要であることを指定する 1646
InstallShield 前提条件のインストールの要否をエンド ユーザーが選択できるようにする 1646
ターゲットシステムで [インストールされるセットアップ前提条件] リストに、前提条件の名前を含めるかどうかを指定する 1646
実行時に、InstallShield 前提条件のインストールの進行状況を表示するかどうかを指定する 1648
InstallShield 前提条件のインストールに伴う可能性のある問題の対処方法について 1649
再起動を必要とする InstallShield 前提条件の動作を指定する 1649
InstallShield 前提条件の依存関係を指定する 1649
InstallShield 前提条件の依存関係を追加する 1650
InstallShield 前提条件の依存関係を変更する 1650
InstallShield 前提条件の依存関係を削除する 1650
InstallShield 前提条件を保存する 1650
マージ モジュールのデザイン 1653
マージ モジュール プロジェクトを作成する 1654
マージ モジュールのデフォルトのインストール先フォルダーを指定する 1654
マージ モジュール言語を選択する 1655
マージ モジュールに依存関係を追加する 1655
マージ モジュールへ除外を追加する 1656
マージ モジュールをマージ モジュール プロジェクト内からリポジトリにパブリッシュする 1657
InstallScript オブジェクトを作成する 1659
オブジェクトの設計 1660
オブジェクト用機能の設計 1662
オブジェクト ファイルの管理 1663
オブジェクトのウィザードの設計 1663
カスタム オブジェクト ウィザード 1664
オブジェクトのデザインタイム環境のローカライズ 1665
インターナショナル化のためのオブジェクトの作成 1666
実行時に、オブジェクトとインストールとの互換性を持たせる 1667
プロパティとメソッド 1668
プロパティの作成 1668
構造評価プロパティの作成 1670
オブジェクトとインストールの間で文字列配列を渡す 1672
オブジェクトのプロパティにアクセスする 1673
メソッドの作成 1674

メソッドを使用する *1675*
オブジェクトステータスのプロパティ *1675*
オブジェクトの実行時のユーザー インターフェイスを作成する 1679
オブジェクトでサポートされていない機能 1680
オブジェクトのビルド 1681
ユーザー インターフェイスからオブジェクトをビルドする 1681
コマンドラインからオブジェクトのコンパイルとビルドを行う 1682
オブジェクトのテストとデバッグ 1682
オブジェクトの配布 1684
オブジェクトのスクリプト 1685
オブジェクト プロジェクトでサポートされていない関数 1685
オブジェクト プロジェクトでサポートされていない定数 1685
オブジェクト プロジェクトでのみサポートされている関数 1686
ScriptDefinedVar プロパティをオブジェクトに追加する 1686
オブジェクトでシステム変数を使用する 1688
スクリプトを使用してオブジェクトのインストール先を設定する 1688

7 開発プロセスを再利用/分担するためのインストール プロジェクトのモジュール化 1689

DIM のインストール情報を指定する 1690
プロジェクト ファイルを XML またはバイナリ形式で保存する 1690
DIM のデフォルトのインストール先フォルダーを指定する 1690
DIM でパス変数を使用する 1691
DIM のファイルを編成する 1695
DIM のターゲット システムの構成 1695
DIM のインストール動作をカスタマイズする 1696
DIM のサーバーを構成する 1696

8 アプリケーションのアップデート 1697

アップグレードの概要 1697
メジャーアップグレード 1699
マイナーアップグレード 1700
スモール アップデート 1700
パッチの適用 1701
QuickPatch プロジェクト 1701
差分リリース 1702
完全リリース 1703
最適なアップグレード ソリューションの決め方 1703
メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い 1706
自動アップグレード 1709
パッチと QuickPatch プロジェクトの違い 1709
アップグレードのパッケージ オプション 1712
差分リリースと完全リリース 1713
アップグレード、パッチ、および QuickPatch プロジェクトを使用する 1714
ファイルの上書き規則について 1715
アップグレード関する考慮事項 1715

InstallShield を構成してアップグレードの種類を自動判別する 1717
自動アップグレード アイテムの追加 1717
自動アップグレード アイテムの設定を構成する 1717
自動アップグレードからメジャーまたはマイナーアップグレードへ変換する 1718
自動アップグレード アイテムを削除する 1718
メジャー アップグレードを作成する 1718
メジャー アップグレード アイテムを追加する 1719
メジャー アップグレードの設定を構成する 1719
メジャー アップグレード アイテムを削除する 1720
マイナー アップグレードを作成する 1720
マイナー アップグレード アイテムを追加する 1721
マイナー アップグレードの設定を構成する 1721
マイナー アップグレードを構成してインストール済みのデータを削除する 1722
マイナー アップグレード アイテムを削除する 1723
マイナー アップグレードの実行時の動作 1723
スモール アップデートを作成する 1724
パッチ時の考慮事項 1725
アップデート起動ツールのファイルのプロパティをカスタマイズする 1727
アップデート起動ツールのアイコンを指定する 1728
パッチ シーケンス 1729
パッチのアンインストール 1732
非管理者パッチ 1733
パッチの作成と適用 1734
新しいパッチ構成アイテムを追加する 1735
新しい最新セットアップ アイテムを追加する 1735
新しい以前のセットアップ アイテムを追加する 1735
外部ファイルを追加する 1736
パッチの設定を構成する 1736
パッチの識別情報を指定する 1737
パッチのアンインストールを有効にする 1737
パッチのシーケンス 1738
パッチ パッケージに署名する 1739
パッチ パッケージをパスワードで保護する 1739
パッチに Update.exe アップデート ランチャをビルドをするかどうかを指定する 1740
最新セットアップ アイテムを別のパッチ構成にコピーする 1740
パッチ構成アイテムを削除する 1741
最新セットアップ アイテムを削除する 1741
以前のセットアップ アイテムを削除する 1741
パッチのビルド 1742
パッチの適用 1742
QuickPatch プロジェクトを作成する 1743
既存の QuickPatch をパッチする QuickPatch プロジェクトを作成する 1744
QuickPatch パッケージを簡素化するかどうかを指定する 1744
パッチのターゲット リリースを指定する 1745
QuickPatch を実行するカスタム アクションを指定する 1746
QuickPatch へファイルを追加する 1746

QuickPatch の識別情報を指定する 1747
QuickPatch のアンインストールを有効にする 1747
QuickPatch パッケージをシーケンスする 1747
QuickPatch パッケージに署名する 1748
QuickPatch パッケージをパスワードで保護する 1748
QuickPatch パッケージに Update.exe アップデート起動ツールをビルドをするかどうかを指定する 1749
[パッチするファイル] エクスプローラーからファイルを削除する 1750
QuickPatch を使用したインストール済みファイルの変更と削除 1750
QuickPatch を使用したレジストリデータの追加、変更および削除 1750
QuickPatch プロジェクトのファイル形式を指定する: XML またはバイナリ 1751
アップグレード、パッチ、および QuickPatch パッケージを検証する 1751
検証ツール 1752
Val0001 . . . . . *1753*
Val0002 . . . . . *1754*
Val0003 . . . . . *1755*
Val0004 . . . . . *1757*
Val0005 . . . . . *1758*
Val0006 . . . . . *1759*
Val0007 . . . . . *1760*
Val0008 . . . . . *1760*
Val0009 . . . . . *1762*
Val0011 . . . . . *1763*
Val0012 . . . . . *1764*
Val0013 . . . . . *1764*
Val0014 . . . . . *1765*
Val0015 . . . . . *1766*
グローバル アセンブリ キャッシュのアセンブリをパッチする 1768
差分リリースと完全リリース 1769
以前のバージョンをアップデートする InstallScript リリースを作成する 1769

9 追加のインストール オプション 1773

複数言語インストールの作成 1775
グローバリゼーションのヒント 1776
デフォルトのプロジェクト言語の設定 1777
言語の設定 1778
インストール言語の選択 1781
コンポーネントを言語依存としてマークする 1782
言語基づいてコンポーネントをインストールする 1783
リリースへの言語の組み込み 1784
文字列エントリの翻訳 1785
文字列エントリのエクスポートとインポート 1785
各言語ごとのダイアログを変更する 1787
インストールがユーザー インターフェイスで使用する言語を判別する方法 1789
言語サポートのカスタマイズ 1790
言語識別子 1790

製品の複数のインスタンスをインストールする 1795

複数インスタンス サポートの実行時要件 1795
InstallShield で複数インスタンスを構成する 1796
製品構成にインスタンスを追加する 1797
インスタンスの名前付け 1797
1 つ以上のインスタンスの製品コードを更新する 1798
インスタンスのプロパティを設定します 1799
製品構成からインスタンスを削除する 1800
複数インスタンス サポートに関する特別考慮 1800
複数インスタンスのサポートを含むリリースの構成とビルド 1801
複数インスタンスのサポートを含むメジャー アップグレードの構成とビルド 1802
製品の複数インスタンスをインストールするためのサポート 1803
製品のインスタンスを複数インストールするときの実行時の動作 1804
ターゲット システムの条件を検出する 1807
条件ステートメントのビルド 1807
条件ステートメントの構文 1810
管理者権限と昇格された権限を検出する 1813
初回インストール、メンテナンス モード、およびアンインストールを検出する 1814
初回インストール時にのみ動作をトリガーする 1815
エンド ユーザーが特定の機能を選択したかどうかを検出する 1815
エンド ユーザーが特定のオペレーティング システムを実行しているかどうか検出する 1816
インストールが仮想マシン上で実行されているかどうかを検出する 1817
アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド 1819
アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類 1822
"プラットフォーム" 条件の設定 1826
"ファイルの存在" 条件の設定 1829
"ファイルの比較" 条件の設定 1831
"レジストリの存在" 条件の設定 1835
"レジストリの比較" 条件の設定 1836
"プロパティの比較" 条件の設定 1839
"MSI パッケージ" 条件の設定 1842
"MSI アップグレード" 条件の設定 1844
"対象パッケージ" 条件の設定 1846
"InstallScript パッケージ" 条件の設定 1847
"ロケール" 条件の設定 1849
"インストール済み" 条件の設定 1850
「Windows アプリ パッケージの条件" の設定 1853
"UWP タイプの存在" 条件の設定 1854
"パッケージの操作" 条件の設定 1855
"機能の操作" 条件の設定 1856
"拡張条件" の設定 1857
"モード条件" の設定 1858
アドバンスト UI またはスイート/アドバンスト UI プロジェクトで条件を定義するときのガイドライン 1859
特定のバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールが既にインストールされているかどうかを判別する 1861
アドバンスト UI およびスイート/アドバンスト UI プロジェクトでロケール条件を使用する 1862
アドバンスト UI またはスイート/アドバンスト UI インストールのカスタム条件を作成する 1864

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、拡張条件 DLL を作成する *1865*
アドバンスト UI またはスイート/アドバンスト UI プロジェクトに拡張条件 DLL を追加する *1868*
インストールされたデータの検索 **1871**
定義済みシステム検索の追加 1871
システム検索の追加 1872
システム検索の変更 1873
実行時にレジストリを検索する場合の特別考慮 1873
システム検索をリポジトリにパブリッシュする 1875
システム検索を削除する 1876
XML データの検索 1876
インストールおよびプロジェクトのテーブルを編集する **1879**
プロジェクト テーブルを編集する 1880
ダイレクト エディターで新しいテーブルを追加する 1881
テーブルへレコードを追加する 1881
ダイレクト エディターでの検索と置換 1882
テーブル レコードを編集する 1883
テーブルのフィールド編集 1884
ダイレクト エディターからテーブルをエクスポートする 1885
ダイレクト エディターを使ってテーブルをインポートする 1885
テーブルからレコードを削除する 1886
ダイレクト エディターでテーブルを削除する 1887
Windows Installer データベースとプロジェクト ファイルのインプレース差分比較 1887
テーブルへバイナリデータを入力する 1889
親のないディレクトリ エントリ 1889
InstallShield ツールの使用 **1891**
InstallShield MSI ツール 1891
InstallShield MSI Diff 1892
トランスフォーム (.mst ファイル) によって行われる変更を判別する *1892*
パッチが .msi パッケージに対して有効であるかどうかを判別する *1893*
パッチによる実行時の影響を判別する *1893*
2 つの .msi パッケージ間の違いを判別する *1894*
InstallShield MSI Diff をコマンドラインから実行して、差分のログ ファイルを生成する *1894*
2 つの InstallShieldプロジェクト間の違いを判別する *1894*
InstallShield MSI Query 1895
特定の Windows Installer ベースに対して SQL Query を実行する *1895*
InstallShield MSI Sleuth 1896
ターゲット システムで実行された .msi パッケージについての詳細を表示する *1896*
特定のコンポーネントを含むすべてのインストール済みのパッケージを検索する *1896*
InstallShield MSI Grep 1897
.msi パッケージで特定の文字列を検索する *1897*
InstallShield キャビネット & ログ ファイル ビューアー 1898
InstallScript キャビネット ファイル (.cab) およびヘッダー ファイル (.hdr) を表示する 1898
InstallScript .cab および .hdr ファイルの概要 *1899*
InstallScript .cab および .hdr ファイルを開く *1900*
InstallScript .cab または .hdr ファイルからファイルを抽出する *1901*
InstallScript .cab または .hdr ファイルからファイルについてのレポートを生成する *1901*

InstallScript ログ ファイル (.ilg) を参照する ... 1902
InstallScript ログ ファイルの概要 ... *1902*
InstallScript ログ ファイルを開く ... *1903*
InstallScript ログ ファイルを検索する ... *1904*
InstallScript ログ ファイルをテキスト ファイルに変換する ... *1904*
InstallShield MSIX バンドル ユーティリティ ... 1904
Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方 ... 1907
Windows Installer プロパティの概要 ... 1908
Windows Installer プロパティ リファレンス ... 1909
SETUPEXEDIR ... 1934
アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス ... 1935
Windows Installer ベースのプロジェクトにおけるプロパティの作成 ... 1942
アドバンスト UI およびスイート/アドバンスト UI プロジェクトでプロパティを作成する ... 1944
既存プロパティを変更する ... 1945
ローカライズ可能なプロパティを作成する ... 1945
既存プロパティをローカライズ可能にする ... 1946
プロパティ値が Windows Installer ログ ファイルに記録されないように防ぐ ... 1947
アドバンスト UI およびスイート/アドバンスト UI デバッグ ログ ファイルへのプロパティ値の書き込みを防ぐ ... 1947
パブリック プロパティが制限付きパブリック プロパティである必要があることを指定する ... 1948
InstallScript で Windows Installer プロパティを取得または設定する方法 ... 1949
プロパティから値を削除する ... 1949
プロパティを削除する ... 1950
.msi および .msm データベースを直接編集する ... 1953
Windows Installer パッケージを開く ... 1953
ダイレクト編集モードで .msi/.msm データベースを編集する ... 1954
ダイレクト編集モードでファイルを追加する ... 1954
ダイレクト編集モードでマージ モジュールを追加する ... 1955

10 InstallShield と外部アプリケーションの統合 ... 1957
ソース コード管理を使用する ... 1957
ソース コード管理統合を使用する ... 1958
プロジェクトをソース管理へ追加する ... 1958
ソース管理からプロジェクトをチェックアウトする ... 1959
ソース管理にプロジェクトをチェックインする ... 1960
ソース管理からのプロジェクトのリンクの解除 ... 1960
Microsoft Visual Studio との統合 ... 1960
Microsoft Visual Studio で InstallShield プロジェクトを作成する ... 1961
Microsoft Visual Studio で InstallShield プロジェクトを開く ... 1962
Visual Studio ソリューションで VSSolutionFolder パス 変数を使用する ... 1962
Visual Studio ソリューションにリファレンスを追加する ... 1963
ツールボックスを使ってダイアログ コントロールを追加する ... 1963
InstallShield ツールバーまたはコマンドを Visual Studio ツールバーに追加する ... 1964
Microsoft Visual Studio でリリースをビルドする ... 1965
.NET アセンブリをインストール プロジェクトに追加する ... 1967
Web サービスまたは Web アプリケーション プロジェクトからプロジェクト出力を追加する ... 1968

Microsoft Visual Studio Team Foundation Server との統合 1968
InstallShield プロジェクトを Team Explorer に追加する 1969

11 インストールの開発およびビルド プロセスの自動化 1971
VBScript を使用したサブフォルダーの機能へのリンク 1971
オートメーション インターフェイスを使用した文字列エントリのインポートとエクスポート 1974
オートメーション インターフェイスを使用したプロジェクト レポートの実行 1976
コマンドラインでソース コード管理を使用する 1978
ビルドのステータス イベント 1980
64 ビット システム上でオートメーション インターフェイスを使用する 1981

12 InstallShield プロジェクトを DevOps でビルドする 1983
Azure DevOps ビルド拡張 1983
InstallShield StandAlone Build と Docker 1984

13 リファレンス 1985
メニュー、ツールバー、および ウィンドウのリファレンス 1987
メニュー 1987
[ファイル] メニュー 1988
[ホーム] メニュー 1989
[編集] メニュー 1991
[表示] メニュー 1993
[移動] メニュー 1994
[プロジェクト] メニュー 1997
[ビルド] メニュー 2000
[ツール] メニュー 2005
[ウィンドウ] メニュー 2008
[ヘルプ] メニュー 2009
ツールバー 2010
標準ツールバー 2011
[コントロール] ツールバー 2014
レイアウト ツールバー 2016
[MSI デバッグ] ツールバー 2018
出力ウィンドウ 2018
ダイアログ ボックス リファレンス 2021
[.NET インストーラー クラス引数] ダイアログ ボックス 2025
[引数/値ペアの追加] ダイアログ ボックス 2025
[既存のメディアを追加] ダイアログ ボックス 2026
[このパッケージのファイルを追加する] ダイアログ ボックス 2026
[MIME の種類を追加] ダイアログ ボックス 2028
[新しいメソッドの追加] ダイアログ ボックス 2029
[新しいプロパティの追加] ダイアログ ボックス 2029
[定義済み検索の追加] ダイアログ ボックス 2030
[ソース管理に追加] ダイアログ ボックス 2031

[アプリケーション拡張子マッピング] ダイアログ ボックス 2031
[アプリケーションのマッピング] ダイアログ ボックス 2032
[コンポーネントの関連付け] ダイアログ ボックス 2033
[DIM の関連付け] ダイアログ ボックス 2033
[バッチ ビルド] ダイアログ ボックス 2033
[ディレクトリの参照] ダイアログ ボックス 2034
ディレクトリ/ショートカットのターゲットを参照するダイアログ ボックス 2035
[ファイルの参照] ダイアログ ボックス 2037
[実行するファイルの参照] ダイアログ ボックス 2038
[ショートカット アイコンの参照] ダイアログ ボックス 2039
[タイトル ターゲットの参照] ダイアログ ボックス 2040
[証明書の選択] ダイアログ ボックス 2040
[チェックイン] ダイアログ ボックス 2043
[チェックアウト] ダイアログ ボックス 2043
[コンポーネントのプロパティ] ダイアログ ボックス 2043
[全般] タブ 2044
[ファイルのリンク] タブ 2046
[機能] タブ 2048
[条件ビルダー] ダイアログ ボックス 2048
[コンテンツ ソース パス] ダイアログ ボックス 2049
[InstallScript MSI の変換] ダイアログ ボックス 2050
[新しいコンポーネントの作成] ダイアログ ボックス 2052
[新しい機能の作成] ダイアログ ボックス 2052
[カスタム エラー処理] ダイアログ ボックス 2053
[カスタム エラー] ダイアログ ボックス 2053
[検証設定のカスタマイズ] ダイアログ ボックス 2054
[データ言語] ダイアログ ボックス 2054
[プロパティの削除] ダイアログ ボックス 2054
[依存関係] ダイアログ ボックス 2055
[ダイアログのイメージ] ダイアログ ボックス 2055
[セットアップのデジタル署名] ダイアログ ボックス 2056
[差分] ダイアログ ボックス 2058
[ダイナミック ファイル リンクの設定] ダイアログ ボックス 2059
[データの編集] ダイアログ ボックス 2061
[IIS Metabase 値の編集] ダイアログ ボックス 2061
[オプション リストの編集] ダイアログ ボックス 2062
[レジストリ データの編集] ダイアログ ボックス 2063
[仮想マシンの構成を編集] ダイアログ ボックス 2064
[エラー マッピングのプロパティ] ダイアログ ボックス 2067
[言語の除外] ダイアログ ボックス 2067
[テーブルのエクスポート] ダイアログ ボックス 2068
[機能条件ビルダー] ダイアログ ボックス 2068
[ファイルの詳細] ダイアログ ボックス 2070
[ファイルのプロパティ] ダイアログ ボックス 2070
[ファイルのプロパティ] ダイアログ ボックス (InstallScript インストール ロジェクト) 2074
ファイル 削除の [プロパティ] ダイアログ ボックス 2075

[検索/置換] ダイアログ ボックス 2077
[プロジェクト内の文字列 ID を検索する] ダイアログ ボックス 2079
[フォルダーのプロパティ] ダイアログ ボックス 2079
[全般] タブ 2080
[共有] タブ 2080
[関数シグネチャ] ダイアログ ボックス 2081
[一般オプション – 詳細] ダイアログ ボックス 2084
[一般オプション – 別のディスク ファイル] ダイアログ ボックス 2086
[履歴] ダイアログ ボックス 2086
[ダイアログのインポート] ダイアログ ボックス 2086
[InstallScript ファイルのインポート] ダイアログ ボックス 2087
[SQL スクリプト ファイルのインポート] ダイアログ ボックス 2087
[アクションの挿入] ダイアログ ボックス 2088
[InstallShield前提条件のプロパティ] ダイアログ ボックス 2090
[言語] ダイアログ ボックス 2092
[リンク タイプ] ダイアログ ボックス 2092
[Windows Installer 4.0 以降のログ記録オプション] ダイアログ ボックス 2094
小文字のコンポーネントディレクトリ – 警告 2095
[メディア ファイルのプロパティ] ダイアログ ボックス 2095
[マージ モジュールの構成可能な値] ダイアログ ボックス 2096
[マージ モジュールのプロパティ] ダイアログ ボックス 2096
[メディア] タブ 2097
[インストール先] タブ 2097
[メソッドの参照] ダイアログ ボックス 2098
[メソッド シグネチャ] ダイアログ ボックス 2098
[MIME の種類] ダイアログ ボックス 2100
[ダイナミック リンクの変更] ダイアログ ボックス 2101
[プロパティの変更] ダイアログ ボックス / リリース ウィザード – [プラットフォーム] パネル 2102
[モジュール依存関係] ダイアログ ボックス 2104
[除外モジュール] ダイアログ ボックス 2105
[MSI 値] ダイアログ ボックス 2105
[MST SIS の設定] ダイアログ ボックス 2105
[複数行文字列値] ダイアログ ボックス 2107
[新規依存関係] ダイアログ ボックス 2109
[新規ファイル] ダイアログ ボックス 2109
[新規プロジェクト] ダイアログ ボックス 2109
[オペレーティング システム] ダイアログ ボックス 2112
[オプション] ダイアログ ボックス 2112
[全般] タブ 2113
[ファイルの場所] タブ 2114
[パス変数] タブ 2115
[ユーザー インターフェイス] タブ 2115
[プリファレンス] タブ 2117
[.NET] タブ 2118
[ファイル ビュー] タブ 2119
[ファイルの拡張子] タブ 2121

[前提条件] タブ 2123
[ソース管理] タブ....2123
[ディレクトリ] タブ 2125
[リソース] タブ 2125
[検証] タブ 2126
[リポジトリ] タブ 2127
[SQL スクリプト] タブ 2127
[マージ モジュール] タブ 2129
その他の IIS プロパティ ダイアログボックス 2130
[その他のウィンドウ スタイル] ダイアログ ボックス....2130
ダイアログのその他のウィンドウ スタイル 2131
ビットマップ、アイコン、およびテキスト領域コントロールのその他のウィンドウ スタイル 2133
ボタンコントロールのその他のウィンドウスタイル....2134
コンボ ボックスコントロールのその他のウィンドウ スタイル....2136
リスト ビュー コントロールのその他のウィンドウ スタイル....2138
編集フィールドとリスト ボックス コントロールのその他のウィンドウ スタイル 2140
行コントロールのその他のウィンドウ スタイル 2142
進行状況バー コントロールのその他のウィンドウ スタイル....2144
選択ツリー コントロールのその他のウィンドウスタイル 2145
[出力] ダイアログ ボックス....2146
[上書き] ダイアログ ボックス 2147
[パッチ シーケンス] ダイアログ ボックス....2149
[パス変数のオーバーライド] ダイアログ ボックス 2149
[パス変数の推奨] ダイアログ ボックス....2150
ファイルとディレクトリの [アクセス許可] ダイアログ ボックス....2151
レジストリ キーの [アクセス許可] ダイアログ ボックス....2154
[プラットフォーム スイート] ダイアログ ボックス....2157
[プラットフォーム] ダイアログ ボックス 2157
[ビルド後のイベント] ダイアログ ボックス....2160
[ビルド前のイベント] ダイアログ ボックス....2161
[圧縮前のイベント] ダイアログ ボックス 2161
[ファイルの準備イベント] ダイアログ ボックス....2162
[前提条件設定] ダイアログ ボックス....2162
[特権] ダイアログ ボックス....2170
[製品条件ビルダー] ダイアログ ボックス....2171
[プロジェクト設定] ダイアログ ボックス 2173
[プロジェクト] タブ....2173
[アプリケーション] タブ (Windows Installer ベースのプロジェクト)....2173
[アプリケーション] タブ (InstallScript プロジェクト) 2175
[プラットフォーム] タブ 2177
[言語] タブ 2178
[メンテナンス] タブ 2178
[言語非依存オブジェクトのプロパティ] タブ 2179
[言語固有オブジェクトのプロパティ] タブ 2180
[キー名の変更] ダイアログ ボックス....2180
[必要な機能] ダイアログ ボックス 2181

[競合の解決] ダイアログ ボックス 2181
[名前を付けて保存] ダイアログ ボックス 2182
[スクリプト変換] ダイアログ ボックス 2183
[スクリプト エディターのプロパティ] ダイアログ ボックス 2184
[SCM シャットダウンの遅延] ダイアログ ボックス 2188
[セキュリティ記述子] ダイアログ ボックス 2188
[ファイルの選択] ダイアログ ボックス 2189
[メディアの場所選択] ダイアログ ボックス 2189
[SQL Server の選択] ダイアログ ボックス 2189
[文字列の選択] ダイアログ ボックス 2190
シリアル番号違反ダイアログ ボックス 2191
[サーバーの場所] ダイアログ ボックス 2192
自動開始サービスの遅延に関する [サービスのオプション] ダイアログ ボックス 2193
回復操作の実行のための [サービスのオプション] ダイアログ ボックス 2193
[サービス SID] ダイアログ ボックス 2194
[ファイルの URL の設定] ダイアログ ボックス 2195
[設定] ダイアログ ボックス 2195
[コンパイル/リンク] タブ 2196
[実行/デバッグ] タブ 2200
[MSI ログファイル] タブ 2201
[セットアップ言語] ダイアログ ボックス 2201
[SQL Server 要件] ダイアログ ボックス 2203
[優先] ダイアログ ボックス 2203
[UI 言語] ダイアログ ボックス 2204
[マージ モジュール検索パスのアップデート] ダイアログ ボックス 2204
ウィザード リファレンス 2205
コンポーネント ウィザード 2206
[ようこそ] パネル 2207
セットアップ ベスト プラクティス 2208
[インストール先] パネル 2209
[ファイル] パネル 2210
[概要] パネル 2211
[コンポーネント タイプ] パネル 2211
COM サーバー コンポーネントの種類 2212
[COM サーバー ファイル] パネル 2213
[クラス] パネル 2215
[コンテキストの種類] パネル 2217
[タイプ ライブラリ] パネル 2217
[AppID 一般情報] パネル 2218
[AppID 詳細情報] パネル 2218
[概要] パネル 2220
サービスのコントロール コンポーネントの種類 2220
[サービスの指定] パネル 2221
[インストール イベント] パネル 2222
[アンインストール イベント] パネル 2223
[待機のタイプ] パネル 2224
[概要] パネル 2224
サービスのインストール コンポーネントの種類 2225

[サービス実行可能ファイル] パネル ... 2226
[サービスの種類の情報] パネル ... 2227
[サービス開始タイプ情報] パネル ... 2228
[サービスのロード順] パネル ... 2229
[エラー コントロール] パネル ... 2229
[サービス ログオン] パネル ... 2230
[概要] パネル ... 2231
フォント コンポーネント タイプ ... 2231
[インストールされているフォントの追加] パネル ... 2232
[新規フォントの追加] パネル ... 2233
[概要] パネル ... 2233
コンポーネント ウィザード [一般概要] パネル ... 2234
ソース パス変換ウィザード ... 2235
[ようこそ] パネル ... 2235
[検索] パネル ... 2235
[検索結果と推奨] パネル ... 2235
[プロジェクトのアップデート] パネル ... 2236
[アップデート完了] パネル ... 2236
新規 QuickPatch 作成ウィザード ... 2236
[ようこそ] パネル ... 2237
[QuickPatch プロジェクト ベース] パネル ... 2237
[元のセットアップ パッケージ] パネル ... 2238
[既存の QuickPatch の場所] パネル ... 2239
テーブル作成ウィザード ... 2240
[ようこそ] パネル ... 2240
[列のプロパティ] パネル ... 2241
[概要] パネル ... 2242
カスタム アクション ウィザード ... 2242
[ようこそ] パネル ... 2244
[基本情報] パネル ... 2244
[アクションの種類] パネル ... 2245
[関数定義] パネル ... 2251
[アクション パラメーター] パネル ... 2254
[マネージ メソッドの定義] パネル ... 2259
[シーケンス内スクリプト] パネル ... 2261
[追加のオプション] パネル ... 2261
[応答オプション] パネル ... 2264
[シーケンスに挿入] パネル ... 2265
[概要] パネル ... 2266
データベース インポート ウィザード ... 2266
[ようこそ] パネル ... 2267
SQL Server パネル ... 2267
[SQL データベース] パネル ... 2267
[データベース テーブル] パネル ... 2268
[スクリプト化するオブジェクト] パネル ... 2268
[スクリプト作成オプション] パネル ... 2269
[スクリプト作成詳細オプション] パネル ... 2271
[概要] パネル ... 2272

デバイス ドライバー ウィザード....2273
[ようこそ] パネル....2273
[デバイス ドライバー パッケージ] パネル....2274
[デバイス ドライバー ファイル] パネル....2274
[デバイス ドライバー情報] パネル....2274
デバイス ドライバー タイプ....*2275*
プロジェクト全体の [デバイスドライバー情報] パネル....2276
[概要] パネル....2277
ダイアログ ウィザード....2277
[ようこそ] パネル....2277
[ダイアログ テンプレート] パネル....2278
[ユーザー インターフェイス シーケンス] パネル....2280
[ダイアログの位置と条件] パネル....2281
DirectX オブジェクト ウィザード....2281
[ようこそ] パネル....2282
[オブジェクトの設定] パネル....2282
[概要] パネル....2283
ダイナミック スキャン ウィザード....2283
[ようこそ] パネル....2284
[ファイルのフィルター] パネル....2284
[実行可能ファイルの指定] パネル....2285
[アプリケーション ファイルの指定] パネル....2285
アプリケーションの起動....2286
[アプリケーションが実行中です] パネル....2286
[ファイル選択] パネル....2287
[スキャン結果] パネル....2287
[ダイナミック スキャン ウィザード完了] パネル....2288
コンポーネントのエクスポート ウィザード....2288
[ようこそ] パネル....2289
[コンポーネントの選択] パネル....2289
[ターゲット ファイルの選択] パネル....2290
[ターゲット ファイル情報] パネル....2290
[概要] パネル....2291
関数ウィザード....2291
[関数名] パネル....2291
[関数のパラメーター] パネル....2292
DIM のインポート ウィザード....2292
[ようこそ] パネル....2293
[ソース ファイルの選択] パネル....2293
[インポート] パネル....2293
REG ファイルのインポート ウィザード....2294
[ようこそ] パネル....2294
[レジストリ ファイルのインポート] パネル....2295
[競合オプションのインポート] パネル....2295
[インポート進行状況] パネル....2295
XML 設定のインポート ウィザード....2295

[ようこそ] パネル 2296
[XML ファイル] パネル 2297
[XML 要素] パネル 2297
[XML 進行状況] パネル 2297
[XML 結果] パネル 2297
Microsoft .NET Framework オブジェクト ウィザード 2298
[ステップ 1] パネル 2298
[ステップ 2] パネル 2299
MSI からMSIX への変換ウィザード 2300
[ようこそ] パネル 2301
新しい言語ウィザード 2303
[ようこそ] パネル 2303
プロジェクト言語パネル 2304
[プロジェクトファイル] パネル 2304
[概要] パネル 2305
MSI/MSM オープン ウィザード 2306
[ようこそ] パネル 2306
[ファイル の場所] パネル 2307
MSP オープン ウィザード 2307
[ようこそ] パネル 2307
[基本の MSI パッケージ] パネル 2307
トランスフォーム オープン ウィザード 2308
[ようこそ] パネル 2309
[トランスフォーム情報] パネル 2309
[追加トランスフォーム] パネル 2309
[応答トランスフォームの作成] パネル 2310
パブリッシュ ウィザード 2311
[ようこそ] パネル 2312
[リポジトリ] パネル 2312
[パブリッシュ情報] パネル 2312
[概要] パネル 2312
再配布可能ファイル ダウンローダー ウィザード 2312
Reg-Free COM ウィザード 2313
[ようこそ] パネル 2313
[EXE ファイルの選択] パネル 2314
[マニフェスト ファイルの選択] パネル 2314
[ファイルの選択] パネル 2314
[概要] パネル 2314
リリース ウィザード 2315
[ようこそ] パネル 2315
[製品構成] パネル 2316
[リリースの指定] パネル 2316
[フィルターの設定] パネル 2316
[マージ モジュールのオプション]パネル 2317
[セットアップ言語] パネル 2317
[メディアの種類] パネル 2319

[ディスク分割オプション] パネル 2320
[ディスク分割詳細設定] パネル 2321
[Web タイプ] パネル 2322
[ダウンローダーのオプション]パネル 2323
[Web からインストールする オプション] パネル 2324
One-Click Install パネル 2325
[ローカル マシン] パネル 2326
[リリース構成] パネル 2326
[カスタム圧縮の設定] パネル 2327
[セットアップランチャ] パネル 2328
[InstallShield 前提条件] パネル 2329
[デジタル署名] パネル 2330
[デジタル署名のオプション] パネル 2332
[パスワードと著作権] パネル 2336
[詳細設定] パネル 2337
[リリース設定の概要] パネル 2341
[一般オプション] パネル 2341
[機能] パネル 2343
[メディア レイアウト] パネル 2343
[カスタム メディア レイアウト] パネル 2344
[ユーザー インターフェイス] パネル 2345
[インターネット オプション] パネル 2346
[アップデート] パネル 2347
[オブジェクト差分] ダイアログ ボックス 2349
[ポストビルドのオプション] パネル 2349
セットアップ ベスト プラクティス ウィザード 2351
[ようこそ] パネル 2351
[準拠] パネル 2352
スタティック スキャン ウィザード 2352
[ようこそ] パネル 2353
[ファイルのフィルター] パネル 2353
[スキャンの進行状況] パネル 2353
[ファイル選択] パネル 2354
[スキャン結果] パネル 2355
[スタティック スキャン ウィザードの完了] パネル 2355
システム検索ウィザード 2355
[ようこそ] パネル 2356
検索する対象を指定してくださいパネル 2356
検索方法パネル (システム検索メソッドの定義) 2357
[ファイルの詳細の指定] パネル (ファイル検索オプション) 2359
検索方法パネル (フォルダー検索オプション) 2360
検索方法パネル(特定のフォルダーオプション) 2360
検索方法パネル (レジストリ検索オプション) 2361
検索方法パネル (.ini ファイル検索 オプション) 2362
検索方法パネル (コンポーネント検索オプション) 2363
[XML ファイル] パネル (XML ファイル オプション) で検索するデータを指定します。 2363

この値の処理方法を指定してくださいパネル 2364
この値の処理方法を指定してくださいパネル 2365
トランスフォーム ウィザード 2366
[ようこそ] パネル 2367
[ファイルの指定] パネル 2367
[検証設定] パネル 2367
[エラー条件の抑制] パネル 2369
[出力ファイル名の指定] パネル 2370
[概要] パネル 2370
[トランスフォーム ウィザードの完了] パネル 2371
アップグレード検証ウィザード 2371
[ようこそ] パネル 2371
[設定] パネル 2372
[概要] パネル 2372
ユーザー インターフェイス ウィザード 2373
[定義済みタスク ページ] パネル 2373
[タスク構成] パネル 2376
Visual Basic .NET、C# .NET、および Visual C++ .NET 用の Visual Studio .NET ウィザード 2377
[ようこそ] パネル 2378
[ソリューション] パネル 2378
Visual Studio デプロイメント プロジェクト インポート ウィザード 2378
[ようこそ] パネル 2379
[プロジェクトファイル] パネル 2379
[オプション] パネル 2379
[概要] パネル 2385
ビュー リファレンス 2387
[インストール情報] ビュー 2387
[一般情報] ビュー 2388
[一般情報] ビューの設定 2389
[パッケージ情報] ビュー 2433
[一般情報] ビュー 2433
[一般情報] ビューの設定 2434
[ビジュアル資産] ビュー 2437
パッケージ ロゴの設定 2437
既定のビジュアル資産の設定 2438
ユーザー定義の設定 2443
[アプリ機能] ビュー 2443
デバイスのアプリ機能 2443
全般アプリ機能 2444
制限付きアプリ機能 2447
[宣言] ビュー 2449
[アプリケーションの宣言セット] 2449
パッケージの宣言 2499
[コンテンツ URI] ビュー 2503
[編成] ビュー 2504
[セットアップのデザイン] ビュー 2506

[機能] ビュー 2507
機能の設定 *2508*
[コンポーネント] ビュー 2522
コンポーネントの設定 *2523*
コンポーネントの詳細設定 *2545*
アプリケーション パスの設定 2547
アセンブリの設定 2547
COM 登録の設定 2549
ファイル タイプの設定 2555
サービスの設定 2557
パブリッシュの設定 2558
デバイス ドライバーの設定 2559
その他のデータの設定 2562
[セットアップの種類] ビュー 2563
[パッケージ] ビュー 2564
共通 *2565*
Features *2598*
ダイナミック リンクの設定 *2599*
[DIM リファレンス] ビュー 2601
[全般] タブ *2601*
[手順] タブ *2602*
[ビルド オプション] タブ *2602*
[機能] タブ *2603*
[シーケンス] タブ *2604*
[アプリケーション データ] ビュー 2604
[ファイルとフォルダー] ビュー 2605
インストール先フォルダー *2608*
[再配布可能ファイル] ビュー 2612
[前提条件] ビュー 2616
[オブジェクト] ビュー 2618
[システム検索] ビュー 2619
[ショートカット] ビュー 2623
ショートカットの設定 *2625*
フォルダーの設定 *2644*
[タイル構成] の設定 *2646*
[MSIX アプリケーション] ビュー 2650
アプリケーションの設定 *2650*
[レジストリ] ビュー 2656
[ODBC リソース] ビュー 2657
ODBC リソースの設定 *2657*
[INI ファイルの変更] ビュー 2660
.ini ファイルの設定 *2661*
.ini ファイルのセクションの設定 *2662*
.ini ファイルのキーワードの設定 *2662*
[環境変数] ビュー 2663
環境変数の設定 *2664*
[XML ファイルの変更] ビュー 2666
XML ファイルの [全般] タブ *2667*

XML ファイルの [詳細] タブ 2668
XML ファイルの [名前空間] タブ 2670
XML 要素の [全般] タブ 2670
XML 要素の [詳細] タブ 2672
[テキスト ファイルの変更] ビュー 2673
変更セットの設定 2673
変更の設定 2675
[スケジュール タスク] ビュー 2679
スケジュール タスクの設定 2679
[サービス] ビュー 2683
[サービス] ビューの設定 2684
[サーバー構成] ビュー 2698
[IIS 構成] ビュー 2699
Web サイトの設定 2700
"アプリケーション" と "仮想ディレクトリ" の設定 2709
アプリケーション プールの設定 2718
Web サービス拡張の設定 2724
[コンポーネント サービス] ビュー 2725
[インストール] タブ 2726
[SQL スクリプト] ビュー 2729
[全般] タブ 2730
[要件] タブ 2733
[詳細] タブ 2735
SQL スクリプト レベル 2738
[全般] タブ 2739
[スクリプト] タブ 2740
[ランタイム] タブ 2740
[データベースのインポート] タブ 2743
[テキスト置換] タブ 2743
[動作とロジック] ビュー 2744
[InstallScript] ビュー 2747
[InstallScript] ビューのスクリプト エディター 2748
[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) 2749
カスタム アクションの種類 2752
カスタム アクションの設定 2756
アクション テキストの設定 2771
[サポート ファイル] ビュー (アドバンスト UI、基本の MSI、InstallScript オブジェクト、およびスイート/アドバンスト UI プロジェクト) 2773
[サポート ファイル/ビルボード] ビュー (InstallScript および InstallScript MSI プロジェクト) 2775
[システム検索] ビュー 2776
[プロパティ マネージャー] ビュー 2777
[イベント] ビュー 2780
[イベント] ビューの設定 2781
[ユーザー インターフェイス] ビュー 2789
[ダイアログ] ビュー (InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクト) 2791
ダイアログ ビュー (基本の MSI、その他の Windows Installer ベースのプロジェクト) 2792
[ウィザード インターフェイス] ビュー 2792

[ウィザード インターフェイス] ビューのツールバー *2795*
[ビルボード] ビュー 2799
ビルボード設定 *2800*
Adobe Flash アプリケーション ファイル ビルボードとイメージ ビルボードの設定 *2801*
[文字列エディター] ビュー 2805
[メディア] ビュー 2808
[パス変数] ビュー 2809
[アップグレード] ビュー 2814
[共通] タブ *2816*
[詳細] タブ *2818*
メジャー アップグレード アイテム: [共通] タブ *2820*
メジャー アップグレード アイテム: [詳細] タブ *2822*
マイナー アップグレード/スモール アップデート アイテムの設定 *2826*
自動アップグレード アイテムの設定 *2826*
[リリース] ビュー 2827
[製品構成] の設定 *2828*
製品構成の [全般] タブ 2828
製品構成の [複数インスタンス] タブ 2840
リリースの設定 *2840*
リリースの [ビルド] タブ 2841
リリースの [Setup.exe] タブ 2854
リリースの [署名] タブ 2870
リリースの [インターネット] タブ 2878
リリースの [イベント] タブ 2881
リリースの [Appx/MSIX] タブ 2894
リリースの [アップデート] タブ 2905
MSIX リリースの [全般] タブ 2908
MSIX リリースの [アプリ インストーラー] タブ 2919
連鎖 .msi パッケージの設定 *2922*
[パッチのデザイン] ビュー 2927
パッチの構成 *2928*
[共通] タブ 2928
[識別] タブ 2930
[デジタル署名] タブ 2931
[シーケンス] タブ 2933
[詳細] タブ 2934
最新セットアップ *2943*
以前のセットアップ *2944*
[共通] タブ 2944
[詳細] タブ 2945
追加の外部ファイル *2947*
外部ファイル 2947
[追加ツール] ビュー 2947
依存関係スキャナー 2949
MSI デバッガー 2949
ダイレクト エディター 2950
Windows Installer テーブル リファレンス *2956*
InstallShieldテーブル リファレンス *2956*
ISAlias テーブル 2958
ISCOMCatalogAttribute テーブル 2959

ISCOMCatalogCollection テーブル 2960
ISCOMCatalogCollectionObject テーブル 2960
ISCOMCatalogObject テーブル 2961
ISCOMPlusApplication テーブル 2961
ISCustomActionReference テーブル 2962
ISIISItem テーブル 2963
ISIISProperty テーブル 2963
ISProductConfigurationInstance テーブル 2965
ISSelfReg テーブル 2966
InstallShield標準 Windows Installer テーブルの列 2967
QuickPatch プロジェクト 2969
[パッチの設定] ビュー 2969
[一般情報] ビュー 2970
製品のプロパティ 2970
ビルドの設定 2971
[共通] タブ 2971
[識別] タブ 2972
[デジタル署名] タブ 2972
[詳細] タブ 2975
履歴 2983
カスタム アクション 2983
[ファイル] ビュー 2983
新規ファイルの設定 2983
変更/削除されたファイルの設定 2984
[レジストリ] ビュー 2986
InstallShield 前提条件エディター リファレンス 2989
[プロパティ] タブ 2990
[Windows パッケージ マネージャー] タブ 2990
[条件] タブ 2992
[含めるファイル] タブ 2993
[実行するアプリケーション] タブ 2993
[動作] タブ 2996
[依存関係] タブ 3001
エラーと警告 3003
ビルド エラーと警告 3004
メディア ビルド エラーと警告 3101
エラー 118 3103
エラー 129 3103
警告 -5000 3104
警告 -5006 3104
警告 -5020 3104
警告 -5021 3105
警告 -5032 3105
警告 -5050 3105
警告 -5051 3105
エラー -7040 3106
エラー -7041 3106
エラー -7043 3106
エラー -7044 3107

エラー –7045 . . . 3107
エラー –7126 . . . 3107
エラー –7127 . . . 3107
エラー –7275 . . . 3108
エラー –7276 . . . 3108
エラー –7380 . . . 3108
エラー –9008 . . . 3108
MSI/MSM 変換エラー . . . 3109
Windows Installer 実行時のエラー . . . 3113
Setup.exe 戻り値および実行時のエラー (InstallScript プロジェクト) . . . 3123
Setup.exe 戻り値および実行時のエラー (基本の MSI および InstallScript MSI プロジェクト) . . . 3126
Setup.exe 戻り値および実行時のエラー (アドバンスト UI およびスイート/アドバンスト UI) . . . 3131
Visual Studio プロジェクトのインポート エラーと警告 . . . 3141
アップグレード エラー (InstallShield Professional からのアップグレード) . . . 3160
アップグレードの警告 (InstallShield Professional からのアップグレード) . . . 3168
更新の警告 289: 動的にリンクされたファイルのファイルリンクが存在しません . . . 3168
更新の警告 –305: InstallShield Professional コンポーネントの "ビルドに含める" プロパティを直接移行できませんでした . . . 3169
更新の警告 –480: ファイル グループ中のファイルの共有 .dll 参照カウントのインクリメント . . . 3169
更新の警告 –481: ファイルグループ内のダイナミックリンクファイルの共有 DLL 参照カウントのインクリメント . . . 3170
更新の警告 –486: ショートカット アイコンの指定 . . . 3170
更新の警告 –487: インストール時のファイルの上書き . . . 3171
更新の警告 –488: スクリプト ファイルに古い形式のイベントがあります . . . 3172
更新の警告 495: ODBC コアがデフォルトでインストールされていません . . . 3173
アップグレード エラーと警告 (InstallShield—Windows Installer Edition からのアップグレード) . . . 3173
Windows Installer ランタイムエラーの HRESULT 値 . . . 3175
DIFxAPI エラー (InstallScript プロジェクト) . . . 3176
"文字列 PRODUCT_NAME が文字列テーブルで見つかりません" エラー . . . 3178
InstallScript エラー情報 . . . 3179
InstallScript 構文またはコンパイラ エラー . . . 3179
エラー C8001 . . . *3183*
エラー C8002 . . . *3184*
エラー C8003 . . . *3184*
エラー C8004 . . . *3184*
エラー C8005 . . . *3185*
エラー C8006 . . . *3185*
エラーC8007 . . . *3186*
エラー C8008 . . . *3186*
エラー C8009 . . . *3186*
エラー C8010 . . . *3187*
エラー C8011 . . . *3187*
エラー C8012 . . . *3187*
エラー C8013 . . . *3187*
エラー C8014 . . . *3188*
エラー C8015 . . . *3188*
エラー C8016 . . . *3189*

エラー C8017. 3189
エラー C8018. 3189
エラー C8019. 3190
エラー C8020. 3190
エラー C8021. 3191
エラー C8022. 3191
エラー C8023. 3191
エラー C8024. 3191
エラー C8025. 3192
エラー C8026. 3193
エラー C8027. 3193
エラー C8028. 3193
エラー C8031. 3194
エラー C8032. 3194
エラー C8033. 3194
エラー C8034. 3195
エラー C8035. 3195
エラー C8036. 3195
エラー C8037. 3196
エラー C8038. 3196
エラー C8039. 3196
エラー C8040. 3197
エラー C8042. 3197
エラー C8043. 3197
エラー C8044. 3198
エラー C8045. 3198
エラー C8046. 3198
エラー C8047. 3199
エラー C8048. 3199
エラー C8049. 3199
エラー C8050. 3200
エラー C8051. 3200
エラー C8052. 3200
エラー C8053. 3200
エラー C8054. 3201
エラー C8055. 3201
エラー C8057. 3201
エラー C8058. 3202
エラー C8059. 3202
エラー C8062. 3202
エラー C8063. 3202
エラー C8064. 3203
エラー C8065. 3203
エラー C8066. 3203
エラー C8067. 3204
エラー C8068. 3204

エラー C8069. . . . . *3204*
エラー C8070. . . . . *3205*
エラー C8071. . . . . *3205*
エラー C8072. . . . . *3205*
エラー C8073. . . . . *3206*
エラー C8074. . . . . *3206*
エラー C8075. . . . . *3206*
エラー C8076. . . . . *3207*
エラー C8077. . . . . *3207*
エラー C8078. . . . . *3207*
エラー C8079. . . . . *3208*
エラー C8080. . . . . *3208*
エラー C8081. . . . . *3208*
エラー C8082. . . . . *3209*
エラー C8083. . . . . *3209*
エラー C8084. . . . . *3209*
エラー C8085. . . . . *3210*
エラー C8086. . . . . *3210*
エラー C8087. . . . . *3210*
エラー C8088. . . . . *3210*
エラー C8089. . . . . *3211*
エラー C8090. . . . . *3211*
エラー C8091. . . . . *3211*
エラー C8092. . . . . *3212*
エラー C8093. . . . . *3212*
エラー C8097. . . . . *3212*
エラー C8098. . . . . *3213*
エラー C8099. . . . . *3213*
エラー C8100. . . . . *3213*
エラー C8101. . . . . *3214*
エラー C8112. . . . . *3214*
エラー C8113. . . . . *3215*
エラー C8114. . . . . *3215*
エラー C8115. . . . . *3215*
エラー C8126. . . . . *3215*
エラー C8127. . . . . *3216*
エラー C8128. . . . . *3216*
エラー C8522. . . . . *3216*
InstallScript の致命的エラー. 3217
エラー F8501. . . . . *3218*
エラー F8502. . . . . *3219*
エラー F8503. . . . . *3219*
エラー F8504. . . . . *3220*
エラー F8505. . . . . *3220*
エラー F8506. . . . . *3221*
エラー F8508. . . . . *3221*

エラー F8509. 3221
エラー F8510. 3222
エラー F8511. 3222
エラー F8512. 3223
エラー F8513. 3223
エラー F8514. 3223
エラー F8515. 3223
エラー F8516. 3224
エラー F8517. 3224
エラー F8518. 3224
エラー F8519. 3225
InstallScript 内部エラー 3225
エラー C9001. 3225
InstallScript 警告. 3225
警告 C7501 3226
警告 C7502 3226
Warning C7503. 3227
Warning C7505. 3227
仮想化変換のエラーと警告 3229
エラー -9000: 不明な例外 3229
エラー -9001: 不明な COM. 3229
エラー -9002: パッケージを開くときにエラーが発生しました. 3229
エラー -9003: パッケージを保存するときにエラーが発生しました. 3230
エラー -9004: ユーザーによってプロセスがキャンセルされました 3230
エラー -9005: 一時フォルダーの作成中にエラーが発生しました 3231
エラー -9006: パッケージの圧縮中にエラーが発生しました. 3231
エラー -9007: 拡張子を持つファイルが見つかりませんでした 3232
エラー -9008: アイコンの抽出中にエラーが発生しました 3232
エラー -9009: 不明のプロバイダー 3232
エラー -9010: ターゲット ファイル名が正しくありません 3233
エラー -9011: MSI テーブルの読み込み中にエラーが発生しました. 3233
エラー -9012: メソッドで予期しないエラーが発生しました 3234
エラー -9013: タイプ ライブラリが見つかりませんでした 3234
エラー -9014: ShellExecute が失敗しました 3234
エラー -9015: ドライバーの完全パスを判別できませんでした 3235
警告 -9016: テーブルのコンテンツが無視されました 3236
警告 -9017: .NET 1.x アセンブリはサポートされていません 3236
警告 -9018: カスタム アクションが無視されました 3236
警告 -9019: 条件付きコンポーネント. 3237
エラー -9020: ディレクトリの親がヌルです 3238
エラー -9021: COM データを抽出できませんでした. 3239
エラー -9022: Complus テーブル 3239
エラー -9024: FileSFPCatalog 3240
警告 -9026: LaunchCondition テーブル 3240
警告 -9027: LockPermissions テーブル 3241
エラー -9028: MoveFile テーブル 3242

エラー –9029: MsiDriverPackages テーブル 3242
警告 –9030: ODBCTranslator テーブル 3243
警告 –9031: RemoveFile テーブル 3243
警告 –9032: RemoveIniFile テーブル 3244
警告 –9033: RemoveRegistry テーブル 3244
エラー –9036: ISCEInstall テーブル 3245
エラー –9037: ISComPlusApplication テーブル 3245
エラー –9038: ISPalmApp テーブル 3246
エラー –9039: ISSQLScriptFile テーブル 3246
エラー –9040: ISVRoot テーブル 3247
エラー –9041: ISXmlFile テーブル 3248
エラー –9051: パッケージの圧縮中がキャンセルされました 3248
エラー –9100: パッケージ オブジェクトのインスタンスの作成が失敗しました 3248
エラー –9101: パッケージ オブジェクトの作成操作が失敗しました 3249
Error –9102: ヘッダー情報の書き込みに失敗しました 3249
エラー –9103: Citrix の最終処理が失敗しました 3250
エラー –9104: Citrix の保存に失敗しました 3250
エラー –9105: Citrix ライターの初期化中にエラーが発生しました 3250
エラー –9106: Citrix パッケージの初期化中にエラーが発生しました 3251
エラー –9107: Citrix ファイル エントリの書き込み中にエラーが発生しました 3251
エラー –9108: ソース ファイル パスの判別中にエラーが発生しました 3252
エラー –9109: Citrix フォルダー エントリの書き込み中にエラーが発生しました 3252
エラー –9110: Citrix レジストリ エントリの書き込み中にエラーが発生しました 3252
エラー –9113: Citrix INI ファイル エントリの書き込み中にエラーが発生しました 3253
エラー –9114: Citrix ショートカットの書き込み中にエラーが発生しました 3253
エラー –9115: Citrix プロファイルを保存するときにエラーが発生しました 3254
エラー –9116: 空の Citrix プロファイルを作成するときにエラーが発生しました 3254
エラー –9117: 中間フォルダーの作成中にエラーが発生しました 3254
エラー –9118: Citrix プロファイルの初期化中にエラーが発生しました 3255
エラー –9119: Citrix プロファイルにデフォルト ターゲットを作成するときにエラーが発生しました 3255
エラー –9120: プロファイルからファイルを削除中にエラーが発生しました 3256
エラー –9121: ファイルを Citrix プロファイルにコピーできませんでした 3256
エラー –9122: ターゲットが Citrix プロファイルに存在しません 3256
エラー –9124: このプロファイルに作成されたショートカットがありません 3257
エラー –9125: Citrix ファイルの種類の関連付けの書き込み中にエラーが発生しました 3257
エラー –9126: 証明書を使ってプロファイルを署名できませんでした 3258
エラー –9127: スクリプトの実行を作成できませんでした 3258
警告 –9128: ショートカットが重複しています 3258
警告 –9129: ショートカットの名前が重複しています 3259
警告 –9130: ショートカットのターゲットが重複しています 3259
警告 –9131: インストーラーの変数を解決できませんでした 3260
警告 –9132: 16 色ショートカット アイコンが見つかりませんでした 3260
警告 –9133: ショートカット アイコンが見つかりませんでした 3260
警告 –9134: 実行可能ファイルからアイコンを抽出できませんでした 3261
エラー –9135: ショートカットのターゲットが 16 ビットです 3261
警告 –9136: 一部のファイルが圧縮されていない可能性があります 3262

警告 -9137: 対象となるディレクトリが見つかりませんでした. 3262
Warning -9138: DuplicateFile テーブル エントリを無視しています 3263
警告 -9150: Windows Installer パッケージをビルド中に警告が発生しました 3263
エラー -9151: Windows Installer パッケージをビルド中にエラーが発生しました 3264
Error -9200: ThinApp のインストールが必要です. 3264
警告 -9201: ショートカット ファイルの拡張子は".exe" である必要があります. 3265
エラー -9202: アプリケーションが作成されませんでした. 3265
Error -9203: ThinApp ツールがありません 3265
エラー -9204: ショートカットが重複しています 3266
エラー -9205: 同じ名前のショートカットが既に存在しますが、パラメーターが異なります 3266
エラー -9206: 同じ名前のショートカットが既に存在しますが、ターゲットが異なります 3267
エラー -9207: ビルド プロセス中にエラーが発生しました (vregtool.exe). 3267
エラー -9208: ビルド プロセス中にエラーが発生しました (vftool.exe). 3267
エラー -9209: ビルド プロセス中にエラーが発生しました (tlink.exe) 3268
エラー -9300: AdviseFile 操作で、未処理の例外が発生しました 3268
エラー -9301: AdviseRegistry 操作で、未処理の例外が発生しました 3269
エラー -9302: コマンド操作で、未処理の例外が発生しました. 3269
エラー -9303: ファイルの変更操作で、未処理の例外が発生しました 3269
エラー -9304: レジストリの変更操作で、未処理の例外が発生しました. 3270
エラー -9305: 作成操作で、未処理の例外が発生しました 3270
エラー -9306: 規則エンジンの実行時に、未処理の例外が発生しました. 3270
エラー -9401: App-V ライターの初期化中に、エラーが発生しました 3271
エラー -9402: App-V パッケージの初期化中に、エラーが発生しました 3271
エラー -9403: App-V ファイル エントリを書き込み中にエラーが発生しました 3271
エラー -9404: App-V フォルダー エントリを書き込み中にエラーが発生しました. 3272
エラー -9405: App-V レジストリ エントリを書き込み中にエラーが発生しました. 3272
エラー -9406: App-V INI ファイル エントリを書き込み中にエラーが発生しました 3272
エラー -9407: App-V ショートカットを書き込み中にエラーが発生しました 3273
エラー -9408: App-V ファイル型のデータを書き込み中にエラーが発生しました 3273
エラー -9409: App-V データの保存中にエラーが発生しました 3273
エラー -9410: ソース ファイル パスの判別中にエラーが発生しました. 3274
エラー -9411: OSD ファイルのテンプレートを抽出できませんでした. 3274
エラー -9412: OSD ファイルを保存できませんでした 3274
エラー -9413: App-V OSD の保存 3275
警告 -9414: プライマリ アプリケーションの依存関係として指定されたローカル App-V アプリケーション 3275
エラー -9415: 依存関係のアプリケーションが見つかりませんでした. 3275
警告 -9416: プライマリ アプリケーション ディレクトリが無効です 3276
エラー -9417: 依存アプリケーションの OSD ファイルに、無効な HREF の値が含まれています。 3276
エラー -9418: サイドバイサイド アセンブリをプライベート化中にエラー 3277
エラー -9419: ウォーターマークを挿入中に、エラーが発生しました 3277
エラー: -9424: App-V 5.x パッケージをビルドする 3277
警告 -9500: ショートカットが不足しています. 3278
エラー -10000: ユーザーによってプロセスがキャンセルされました 3278
警告 -10001: スイート ファイルが不足しています 3278
警告 -10002: スイート ファイルが重複しています 3279
警告 -10003: アプリケーション ファイルが不足しています. 3279

警告 -10004: INI ファイルが不足しています 3279
修正 11000: Excluding TCPIP レジストリ エントリを除外しています 3280
致命的エラー 11001: VMware が失敗しました 3280
警告 11003: コントロール パネル アプレット - Citrix 3281
修正 11004: コントロール パネル アプレット - ThinApp 3281
致命的エラー 11005: QuickTime 7.4.1 が原因で、致命的エラーが発生しました 3281
修正 11006: Adobe Distiller によって AdobePDFSettings が除外されました 3282
修正 11007: URL ショートカットの除外 3282
テクニカル サポートにお問い合わせの前に 3282
変換前および後の操作が必要なアプリケーションの機能 3282
オートメーション インターフェイス 3285
オートメーション オブジェクト 3285
ISWiProject オブジェクト 3286
AddAdvancedFile メソッド 3303
AddAutomaticUpgradeEntry メソッド 3304
AddChainPackage メソッド 3305
AddComponent メソッド 3306
AddCustomAction メソッド 3307
AddFeature メソッド 3308
AddLanguage メソッド 3310
AddPatchConfig Method 3310
AddPathVariable メソッド 3311
AddProductConfig メソッド 3312
AddProperty メソッド 3312
AddSetupFile メソッド 3313
AddSetupType メソッド 3314
AddSQLServerConnection メソッド 3315
AddUpgradeTableEntry メソッド 3316
BuildPatchConfiguration メソッド 3317
BuildPCPFile メソッド 3319
CloseProject メソッド 3320
CreatePatch メソッド 3320
CreateProject メソッド 3321
DeleteAdvancedFile メソッド 3322
DeleteChainPackage メソッド 3323
DeleteComponent メソッド 3323
DeleteCustomAction メソッド 3324
DeleteMergeModule メソッド 3325
DeletePatchConfig メソッド 3325
DeletePathVariable メソッド 3326
DeleteProductConfig メソッド 3327
DeleteProperty メソッド 3327
DeleteSetupFile メソッド 3328
DeleteSetupType メソッド 3329
DeleteSQLConnection メソッド 3330
DeselectLanguage メソッド 3331

ExportProject メソッド 3332
ExportStrings メソッド 3332
ForceUpgrade メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3333
GenerateGUID メソッド 3334
ImportProject メソッド 3335
ImportStrings メソッド 3335
OpenProject メソッド 3337
RemoveFeature メソッド 3339
SaveProject メソッド 3340
ISWiAdvancedFile オブジェクト 3340
ISWiAutomaticUpgradeEntry オブジェクト 3342
Delete メソッド 3342
ISWiChainPackageData オブジェクト 3343
ISWiChainPackage オブジェクト 3344
AddChainPackageData メソッド 3345
DeleteChainPackageData メソッド 3346
ISWiCustomAction オブジェクト 3347
ISWiComponent オブジェクト 3348
AddComponentSubFolder メソッド 3366
AddDynamicFileLinking メソッド 3367
AddEnvironmentVar メソッド 3368
AddFile メソッド 3369
AddRemoveFile メソッド 3370
DeviceDriverFlagsEx プロパティ 3371
DeviceDriverSetRedist プロパティ 3372
ImportINIFile メソッド 3373
ImportRegFile メソッド 3374
ISWiRemoveFiles メソッド 3375
RemoveComponentSubFolder メソッド 3376
RemoveDynamicFileLinking メソッド 3377
RemoveEnvironmentVar メソッド 3378
RemoveFile メソッド 3378
RemoveRemoveFile メソッド 3379
ISWiComponentSubFolder オブジェクト 3380
DeleteFile メソッド 3381
ISWiCondition オブジェクト 3382
ISWiDynamicFileLinking オブジェクト 3382
ISWiEnvironmentVar オブジェクト 3385
ISWiFeature オブジェクト 3387
AddCondition メソッド 3394
AddMergeModule メソッド 3395
AddObject メソッド 3395
AttachComponent メソッド 3396
DeleteCondition メソッド 3397
RemoveComponent メソッド 3397
RemoveMergeModule メソッド 3398

RemoveObject メソッド 3399
ISWiFile オブジェクト 3399
ISWiFolder オブジェクト 3405
AddShortcut メソッド 3408
AddSubFolder メソッド 3408
DeleteShortcut メソッド 3409
DeleteSubFolder メソッド 3410
ISWiLanguage Object 3410
AddStringEntry メソッド 3411
DeleteStringEntry メソッド 3412
ISWiLatestSetup オブジェクト 3413
AddPreviousSetup メソッド 3415
DeletePreviousSetup メソッド 3415
ISWiMsiPatchSequence オブジェクト 3416
ISWiObject オブジェクト 3418
ISWiPatchConfig オブジェクト 3418
AddLatestSetup メソッド 3430
DeleteLatestSetup メソッド 3431
ISWiPatchExternalFile オブジェクト 3431
ISWiPatchWholeFile オブジェクト 3432
ISWiPathVariable オブジェクト 3433
ISWiPreviousSetup オブジェクト 3435
ISWiProductConfig オブジェクト 3436
AddRelease メソッド 3441
DeleteRelease メソッド 3442
ISWiProperty オブジェクト 3442
ISWiRelease オブジェクト 3443
ビルド メソッド 3503
BuildTablesOnly メソッド 3503
BuildTablesRefreshFiles メソッド 3504
CubFile プロパティ 3504
SignMedia プロパティ 3505
ISWiRemoveFile オブジェクト 3506
ISWiSequenceRecord オブジェクト 3507
ISWiSetupFile オブジェクト 3508
ISWiSetupType オブジェクト 3510
ISWiShellProperty オブジェクト 3511
ISWiShortcut オブジェクト 3511
AddShellProperty メソッド 3518
DeleteShellProperty メソッド 3519
ISWiSISProperty オブジェクト 3520
ISWiSQLConnection オブジェクト 3521
AddSQLRequirement メソッド 3524
AddSQLScript メソッド 3525
AddSQLScriptEx メソッド 3525
DeleteSQLScript メソッド 3526

GetDatabaseServer メソッド 3527
InsertSQLScript メソッド 3527
RemoveSQLRequirement メソッド 3528
ISWiSQLDatabaseServer オブジェクト 3528
ISWiSQLReplace オブジェクト 3530
ISWiSQLRequirement オブジェクト 3531
SetPredefinedRequirement メソッド 3532
ISWiSQLScript オブジェクト 3534
AddSQLScriptError メソッド 3537
AddSQLScriptReplacement メソッド 3537
DeleteSQLReplace メソッド 3538
DeleteSQLScriptError メソッド 3538
ISWiSQLScriptError オブジェクト 3539
ISWiStringEntry オブジェクト 3539
ISWiUpgradeTableEntry オブジェクト 3540
Delete メソッド 3541
オートメーション コレクション 3542
ISWiAdvancedFiles コレクション 3542
ISWiAutomaticUpgradeEntries コレクション 3543
ISWiChainPackageDatas コレクション 3544
ISWiChainPackages コレクション 3545
ISWiComponents コレクション 3546
ISWiComponentSubFolders コレクション 3548
ISWiConditions コレクション 3549
ISWiCustomActions コレクション 3550
ISWiDynamicFileLinkings コレクション 3551
ISWiEnvironmentVars コレクション 3552
ISWiFeatures コレクション 3552
ISWiFiles コレクション 3554
ISWiFolders コレクション 3555
ISWiLanguages コレクション 3556
ISWiLatestSetups コレクション 3557
AddMsiPatchSequence メソッド 3558
DeleteMsiPatchSequence メソッド 3558
ISWiMsiPatchSequences コレクション 3559
BuildPatch メソッド 3560
ISWiObjects コレクション 3562
ISWiPatchConfigs コレクション 3563
ISWiPatchExternalFiles コレクション 3564
AddPatchWholeFile メソッド 3565
DeletePatchWholeFile メソッド 3566
ISWiPatchWholeFiles コレクション 3566
ISWiPathVariables コレクション 3567
ISWiPreviousSetups コレクション 3568
AddPatchExternalFile メソッド 3569
DeletePatchExternalFile メソッド 3570

ISWiProductConfigs コレクション 3570
ISWiProperties コレクション 3572
ISWiReleases コレクション 3573
ISWiRemoveFiles コレクション 3574
ISWiSequence コレクション 3575
InsertCustomAction メソッド *3577*
RemoveSequenceRecord メソッド *3577*
ISWiSetupFiles コレクション 3578
ISWiSetupTypes コレクション 3579
ISWiShellProperties コレクション 3580
ISWiShortcuts コレクション 3581
ISWiSISProperties コレクション 3582
ISWiSQLConnections コレクション 3583
ISWiSQLDatabaseServers コレクション 3584
ISWiSQLReplaces コレクション 3585
ISWiSQLRequirements コレクション 3586
ISWiSQLScriptErrors コレクション 3587
ISWiSQLScripts コレクション 3588
ISWiStringEntries コレクション 3589
ISWiUpgradeTableEntries コレクション 3590
» アドバンスト UI およびスイート/アドバンスト UI プロジェクトのオートメーション オブジェクト 3593
ISWiProject オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3593
AddExitCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3601*
AddLanguage メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3602*
AddPathVariable メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3603*
AddProperty メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3603*
AddSetupFile メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3604*
AddSuiteAction メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3605*
AddSuiteFeature メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3606*
AddSuitePackage メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3608*
AddSuiteRelease メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3610*
AddSuiteTransaction メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3610*
CloseProject メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3611*
CreateProject メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3612*
DeleteExitCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3613*
DeleteLanguage メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3613*
DeletePathVariable メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3614*
DeleteProperty メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3615*
DeleteSetupFile メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3616*
DeleteSuiteAction メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3617*
DeleteSuiteFeature メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3617*
DeleteSuitePackage メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3618*
DeleteSuiteRelease メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3619*
ForceUpgrade メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3619*
GenerateGUID メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3620*
ISWiSuiteEvent メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3620*

OpenProject メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3623*
SaveProject メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3624*
ISWiLanguage オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3625
AddStringEntry メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3626*
DeleteStringEntry メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3627*
ISWiPathVariable オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3628
ISWiProperty オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3631
ISWiSetupFile オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3632
ISWiSuiteAction オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3633
ISWiSuiteActionRef オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3641
MoveTo メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3642*
ISWiSuiteCondition オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3643
AddCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3645*
AddExtensionCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3649*
AddGroup メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3649*
AddParameter メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3650*
ChangeGroupType メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3651*
DeleteCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3652*
DeleteParameter メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3653*
Move メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3654*
ISWiSuiteConditionParameter オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3654
ISWiSuiteDFLFilter オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3655
CanMove メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3656*
MoveDown Method メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3657*
MoveUp Method メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3658*
ISWiSuiteDynamicFileLink オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3658
AddFilter メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3659*
DeleteFilter メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3660*
ISWiSuiteEvent オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3661
AddSuiteActionRef メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3662*
DeleteSuiteActionRef メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3663*
ISWiSuiteExitCondition オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3664
ISWiSuiteFeatureオブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3664
AddCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3668*
AddSuiteSubFeature メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3669*
AttachPackage メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3669*
DeleteCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3670*
DeleteSuiteSubFeature メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3670*
RemovePackage メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3671*
ISWiSuiteFile オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3672
ISWiSuiteFolder オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3673
AddDynamicFileLink メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3674*
AddFile メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3675*
AddFolder メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3676*
DeleteDynamicFileLink メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3677*
DeleteFile メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3677*

DeleteFolder メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3678*
ISWiSuiteOperation オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3679
ISWiSuitePackage オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3684
AddDetectCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3706*
AddEligibleCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3707*
AddSuitePackage メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3707*
DeleteCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3709*
MoveDown Method メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3710*
MoveUp Method メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3711*
ISWiSuiteRelease オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3711
Build メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3731*
» アドバンスト UI およびスイート/アドバンスト UI プロジェクトのオートメーション コレクション 3732
ISWiLanguages コレクション 3733
ISWiPathVariables コレクション 3734
ISWiProperties コレクション 3734
ISWiSetupFiles コレクション 3735
ISWiSuiteActionRefs コレクション 3736
ISWiSuiteActions コレクション 3736
ISWiSuiteConditionParameters コレクション 3737
ISWiSuiteConditions コレクション 3738
ISWiSuiteDFLFilters コレクション 3738
ISWiSuiteDynamicFileLinks コレクション 3739
ISWiSuiteEvents コレクション 3740
ISWiSuiteExitConditions コレクション 3741
ISWiSuiteFeatures コレクション 3742
ISWiSuiteFiles コレクション 3743
ISWiSuiteFolders コレクション 3744
ISWiSuitePackages コレクション 3745
ISWiSuiteReleases コレクション 3746
アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用可能な式のオブジェクト リファレンス 3747
Feature オブジェクト 3748
File オブジェクト 3750
Parcel オブジェクト 3751
Platform オブジェクト 3753
Registry オブジェクト 3755
InstallShield カスタム アクション リファレンス 3759
コマンドライン ツール 3771
Compile.exe 3773
ISCmdBld.exe 3779
ISBuild.exe 3789
IISscan.exe 3790
iSign.exe 3791
ReleasePackager.exe 3792
SetupIni.exe 3793
MsiExec.exe コマンドラインのパラメーター 3794
Setup.exe および Update.exe コマンドライン パラメーター 3799

アドバンスト UI およびスイート/アドバンストの Setup.exe コマンドラインのパラメーター 3819
Setup.exe (InstallScript プロジェクト) 3824

14 よく寄せられる質問 (FAQ) 3825

用語集 3829

1 索引 1

目次

1

InstallShield 2025 ヘルプ ライブラリ

InstallShield を使って、Windows ベースのシステムをターゲットにするインストールを素早くビルド、テストおよび配布することができます。

InstallShield ヘルプ ライブラリには、InstallShield の機能性および機能についての情報が掲載されています。ヘルプ ライブラリは、次のセクションに分かれています。

テーブル 1-1 ・ ヘルプ ライブラリのセクション

セクション	説明
InstallShield 2025 の新しい機能	InstallShield 2025 の変更点について案内します。
InstallShield の以前のバージョンの新しい機能	InstallShield の以前のバージョンで加えられた変更に関する情報です。
ターゲット システムの要件	ターゲット システムの要件を一覧にします。
InstallShield を管理者権限を使って、または管理者権限を持たずに起動する違い	管理者権限を持たずに InstallShield を実行している場合に利用できない機能についてアラートします。また、管理者と非管理者コンテキストを切り替えた場合で、プロジェクトでマップされたドライブを使用した際に発生する可能性のある問題についても説明します。
32 ビットと 64 ビット システムにおけるインストールの開発およびビルドの違い	InstallShield を 32 ビット システムで使用している場合に現れる可能性がある、64 ビット システムとの相違点をハイライトします。
ヘルプの使い方	InstallShield ドキュメントに関する情報が提供されています。
スタート ガイド	InstallShield に慣れていただき、インストール プロジェクトの作成の開始、および InstallShield ユーザー インターフェイスのカスタマイズに役立つ情報が掲載されています。

テーブル 1-1 • ヘルプ ライブラリのセクション (続き.)

セクション	説明
チュートリアル	InstallScript および基本の MSI インストール プロジェクトの処理手順、およびグローバル インストールの作成方法について、順を追って説明します。
インストールの作成	ユーザーフレンドリーで信頼できるインスールの作成方法、およびその作成手順がステップごと ([プログラムの追加と削除] のための情報指定からインストールのビルド、テストおよび配布まで) に説明されています。
アドバンスト UI およびスイート/アドバンスト UI インストールの作成	スイート/アドバンスト UI およびアドバンスト UI プロジェクトについての概要および詳しい使用方法が説明されています。 InstallShield Premier Edition で提供されているスイート/アドバンスト UI プロジェクト タイプでは、複数の .msi パッケージ、.msp パッケージ、InstallScript パッケージ、.exe パッケージ、サイドローディング Windows アプリ パッケージ (.appx)、および Windows Installern トランザクションだけでなく、複数の InstallShield 前提条件も、モダンでカスタマイズ可能なユーザー インターフェイスを持つ単一のアプリケーションにパッケージできます。 InstallShield Professional Edition で提供されているアドバンスト UI プロジェクト タイプでは、単一の .msi パッケージ、.msp パッケージ、または InstallScript パッケージに対して、最新のカスタマイズ可能なユーザー インターフェイスを提供することができます。
InstallShield 前提条件およびその他の再配布可能ファイルをデザインする	他のインストール プロジェクトで使用したり、他のインストール開発者が使用できるように配布できる InstallShield 前提条件、マージ モジュール、InstallScript オブジェクトを自分自身で初めて設計するときに役に立つ基本的な概念が紹介されています。
開発プロセスを再利用/分担するためのインストール プロジェクトのモジュール化	デベロッパー インストール マニフェスト (DIM) の紹介。DIMは、製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、およびインストール パッケージの論理的に分かれている部分を別個に構成している要素など、関連するアイテムを集めたものです。サイズは、機能とほぼ同じです。
アプリケーションのアップデート	製品アップデートのために用意されている様々なタイプのアップグレードおよびパッチの計画および実装の仕方をステップごとに見ていくことができます。また、FlexNet Connect を利用して、どのようにエンドユーザーに入手可能なアップグレードおよびパッチについて通知するかについても説明されています。
カスタム仮想アプリケーションの作成	InstallShield を使ってカスタマイズされた仮想アプリケーションを作成する方法について説明します。
追加のインストール オプション	InstallShield で幅広く提供されているオプションの説明で、多言語インストールの作成、1 つの製品の複数インスタンスのインストール、条件ステートメントのビルド、インストール済みデータの検索、インストール テーブルの編集、その他多くのトピックが掲載されています。

テーブル 1-1・ヘルプ ライブラリのセクション (続き.)

セクション	説明
InstallShield と外部アプリケーションの統合	InstallShield と ソース コード管理ソフトウェア、Microsoft Visual Studio や Microsoft Visual Studio Team Foundation Server (TFS) などのサードパーティ ツールとの統合について詳細を説明します。
インストールの開発およびビルド プロセスの自動化	InstallShield ユーザー インターフェイスを直接開くことなくインストール プロジェクトの作成プロセスを自動化することができる InstallShield オートメーション インターフェイスについての詳細が載っています。
リファレンス	次の項目に関する総合リファレンス情報です: InstallShield ユーザー インターフェイス、InstallScript 言語、インストールの作成、コンパイル、ビルド、実行時に発生する可能性があるエラーおよび警告、リリースのビルドおよびインストールの実行等のタスクを実行するためにコマンドラインから利用することができるツール、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの様々な設定にオブジェクト式を埋め込んでターゲット システムを検索するときに使用できるオブジェクト、プロジェクトに追加される InstallShield カスタム アクション、オートメーション インターフェイスを使ってインストール プロジェクトを変更するときに使われるオブジェクト、メソッド、プロパティ およびコレクション。
よく寄せられる質問 (FAQ)	InstallShield およびプロジェクトの作成について頻繁に寄せられる質問に答えるヘルプ トピックを見つけることができます。
用語集	用語およびその解説を集めたものです。

***メモ・** InstallShield ヘルプライブラリは、InstallShield とインタラクトするよう設計されているので、InstallShield 内からヘルプを開くことをお勧めします。ヘルプファイルを別のフォルダーやシステムにコピーすると、多くの機能が正常に機能しないことがあります。*

InstallShield に関してよく寄せられる質問や、ドキュメントに記載されていない新規の情報については ナレッジベースを参照してください。

InstallShield 2025 の新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

- InstallShield 2025 R1 の新しい機能

InstallShield 2025 R1 の新しい機能

InstallShield 2025 R1 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- アドバンスト UI およびスイート/アドバンストUIインストールでハイコントラストモードをサポート
- 64 ビット InstallScript ベースのセットアップ ランチャーのビルド サポート

アドバンスト UI およびスイート/アドバンストUIインストールでハイコントラストモードをサポート

以前のバージョンでは、アドバンスト UI およびスイート/アドバンスト UI インストーラーのインストール中に、ウィザード ページ上の本文テキストやヘッダーテキスト、ナビゲーション ボタンのラベルなどのテキスト要素は、ターゲット システムでハイコントラスト モードが有効になっている場合には読み取れませんでした。

InstallShield 2025 R 1では、アドバンスト UI およびスイート/アドバンストUIインストールが、ターゲット システムのハイコントラスト モードをサポートするように拡張されています。この機能強化により、ターゲットシステムにハイコントラスト モードを設定すると、各ウィザードページの本文テキストやヘッダー テキスト、ナビゲーション ボタンのラベルなどのテキスト要素が明確に読み取れるようになり、アクセシビリティとユーザー エクスペリエンスが向上します。

詳細については、「アドバンスト UI およびスイート/アドバンスト UI インストールでハイコントラストモードをサポート」を参照してください。

64 ビット InstallScript ベースのセットアップ ランチャーのビルド サポート



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield 2025 R 1では、**[メディア]** ビューの **[リリース]** エクスプローラーにあるリリースの **[ビルド]** タブに、**[アーキテクチャ]** という名前の新しい設定が導入されています。この設定により、デプロイメント要件に応じて、32 ビット (x 86) および 64 ビット (x 64) Windowsプラットフォームの両方、または 64 ビット (x 64) Windows プラットフォーム専用の InstallScript ベースのセットアップ ランチャーをビルドできるようになり、柔軟性が向上します。

この設定を導入することで、適切なシステム アーキテクチャを簡単にターゲットにして、アプリケーションのランタイム環境に合わせることができます。次のオプションを選択できます:

- **x86**—このオプションを選択すると、32 ビットと 64 ビットの両方の Windows プラットフォームで動作する 32 ビット セットアップ ランチャーがビルドされます。
- **x64**—このオプションを選択すると、64 ビット Windows プラットフォーム上で動作する 64 ビット セットアップ ランチャーがビルドされます。

デフォルトで、この設定は **x86** になっています。

InstallShield の以前のバージョンの新しい機能

このセクションでは、InstallShield の以前のバージョンでリリースされた機能と強化点が説明されています。

- InstallShield 2024 の新しい機能
- InstallShield 2023 の新しい機能
- InstallShield 2022 の新しい機能

- InstallShield 2021 の新しい機能
- InstallShield 2020 の新しい機能
- InstallShield 2019 の新しい機能
- InstallShield 2018 R2 の新しい機能
- InstallShield 2018 SP1 の新しい機能
- InstallShield 2018 の新しい機能
- InstallShield 2016 SP2 の新しい機能
- InstallShield 2016 SP1 の新しい機能
- InstallShield 2016 の新しい機能
- InstallShield 2015 SP1 の新しい機能
- InstallShield 2015 の新しい機能
- InstallShield 2014 SP1 の新しい機能
- InstallShield 2014 の新しい機能
- InstallShield 2013 SP1 の新しい機能
- InstallShield 2013 の新しい機能
- InstallShield 2012 Spring SP1 の新しい機能
- InstallShield 2012 Spring の新しい機能
- InstallShield 2012 SP1 の新しい機能
- InstallShield 2012 の新しい機能
- InstallShield 2011 の新しい機能
- InstallShield 2010 の新しい機能
- InstallShield 2010 SP1 の新しい機能
- InstallShield 2010 の新しい機能
- InstallShield 2009 SP2 の新しい機能
- InstallShield 2009 SP1 の新しい機能
- InstallShield 2009 の新しい機能
- InstallShield 2008 の新しい機能
- InstallShield 12 SP1 の新しい機能
- InstallShield 12 の新しい機能

InstallShield 2024 の新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

- InstallShield 2024 R2 の新しい機能
- InstallShield 2024 R1 の新しい機能

InstallShield 2024 R2 の新しい機能

InstallShield 2024 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- InstallScript パッケージで ARM をサポート

InstallScript パッケージで ARM をサポート

InstallShield 2024 R2 には、ARM プロセッサ アーキテクチャを使用した Windows 11 上での InstallScript インストールのサポートが含まれています。

InstallShield 2024 R1 の新しい機能

InstallShield 2024 R1 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- .NET Core カスタム アクションをサポート
- SQL Scripts の データベース インポート ウィザード機能の強化

.NET Core カスタム アクションをサポート

InstallShield 2024 R1 では、プロジェクトの **[カスタム アクションとシーケンス]** ビューに .NET Core タイプのカスタムアクションを追加して、.NET Core ベースのアセンブリ ファイルをサポートすることができます.NET Core タイプのカスタム アクションは、Visual Basic .NET または C#のようなマネージ コードで書かれた .NET アセンブリ内のパブリック メソッドを呼び出します。

この機能は、基本の MSI、InstallScript MSI、および スイート/アドバンスト UI プロジェクトで使用できます。

詳細は、次を参照してください:

- カスタム アクションの種類
- カスタム アクションの設定
- .NET Core ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す
- .NET Framework 再配布可能ファイルをプロジェクトへ追加する
- Microsoft .NET Core 前提条件を含める
- スイート/アドバンスト UI インストールで .NET Core アクションを使用する

SQL Scripts の データベース インポート ウィザード機能の強化

[SQL スクリプト] ビューの **データベース インポート ウィザード**を使ってデータベース設定をインポートして、構成された設定およびオプションに基づいて SQL スクリプト ファイルを生成することができます。

InstallShield 2024 R1 の**データベース インポート ウィザード**の機能強化として、SQL Server 管理オブジェクト (SMO) が含まれています。SQL Server 管理オブジェクトを組み込むことで、データベース オブジェクトのインポートおよび SQL スクリプト ファイルの生成がより正確で効率的になります。
この強化機能によって、カスタマーから報告されたレガシー スクリプト問題を解決することができます。

InstallShield 2023 の新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

- InstallShield 2023 R2 の新しい機能
- InstallShield 2023 R1 の新しい機能

InstallShield 2023 R2 の新しい機能

InstallShield 2023 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- 内部ファイルにデジタル署名を行う機能
- デジタル署名のカスタム署名ソリューションを構成できる機能
- 新しいスケジュール タスクの時間単位のスケジューリング設定
- Standalone Build をビルド中に前提条件をダウンロードするコマンドラインをサポート
- EV デジタル署名用トークン パスワードの保存機能

内部ファイルにデジタル署名を行う機能

InstallShield 2023 R2 Premier Edition では、**[メディア]** ビューの **[リリース]** エクスプローラーに表示される **[イベント]** タブに **"ファイルの準備イベント"** および **"ファイルの準備でバッチファイルを実行する"** という設定が追加されています。こららの設定を使って、バッチ ファイル (.bat または . cmd) または実行可能ファイル (.exe) を使ったカスタム スクリプト/ コマンドを指定し、署名されていない InstallShield 内部ファイルにデジタル署名を行うことができます。

InstallShield のビルトイン署名ステップを使用する場合、内部ファイルを含むすべてのバイナリの署名はデフォルトの署名動作で管理されているため、これらの設定は不要です。

これらの設定は、InstallShield の標準署名ソリューションの代わりにカスタム署名ソリューションをご利用のお客様向けに提供されています。

詳細は、次を参照してください:

- リリースの [イベント] タブ
- [ファイルの準備イベント] ダイアログ ボックス

デジタル署名のカスタム署名ソリューションを構成できる機能

InstallShield 2023 R2 では、ビルドで生成されたファイルにデジタル署名を行うカスタム署名ソリューションを選択および構成できる新しい "署名の種類" 設定が導入されました。

これらの設定は、InstallShield の標準署名ソリューションの代わりにカスタム署名ソリューションをご利用希望のお客様向けに提供されています。**[カスタム]** 署名タイプを選択すると、カスタム署名ユーティリティ パスおよび引数を構成できる追加フィールドが有効化されます。カスタム署名ソリューションを構成するには、**[メディア]** ビューの **[リリース]** エクスプローラーに表示されているリリースの **[署名]** タブで次の新しい設定を使用してください::

- **署名の種類** ー この設定を使って、ビルドが生成したファイルにデジタル署名行う方法を指定します。この設定では、次のオプションを使用できます:
  - **標準** ー InstallShield デフォルト署名ツールを使って、ビルドが生成したファイルに署名を行います。

  - **カスタム** ー 独自のカスタマイズ署名ツールを使って、ビルドが生成したファイルに署名を行います。このオプションを設定すると、**"パス"** および **"引数"** 設定が有効になります。

- **パス** ー ビルドが生成したファイルにデジタル署名行うために使用する署名ツールの場所を指定します。署名ツールの場所を指定するには、この設定で省略記号ボタン (...) をクリックします。
- **引数** ー 署名ツール構成のコマンドライン引数を指定します。たとえば、次のコマンドライン引数を使って、Microsoft ビルトイン署名ツールがバイナリに署名を行うカスタム オプションとして構成します。

  sign /fd SHA256 /f "<ProgramFilesFolder>¥testCA.pfx" /t http://timestamp.digicert.com /p MyPassword [filename]

  ***メモ・** カスタム署名ツールへのパスは、<ProgramFilesFolder>¥Windows Kits¥10¥bin¥<WinSDKVer>¥x86¥signtool.exe を使用します。*

  [filename] 変数は、署名を行う完全修飾パスのプレースホルダーです。ビルド時に、これはバイナリ ファイルの完全修飾パスを解決して署名を行います。デフォルトで、引数の終わりにファイル パスが追加され、カスタム署名ツールに渡されます。ハードコード化されたパスを使用する代わりに、プロジェクトで定義されている任意のパス変数および環境変数を使用することができます。

詳細は、次を参照してください:

- リリースの [署名] タブ
- [デジタル署名] タブ ([パッチのデザイン] ビューのパッチ構成用)
- [デジタル署名] タブ (QuickPatch プロジェクト)

***メモ・** デフォルトで、**"署名の種類"** 設定は **[標準]** になっています。*

新しいスケジュール タスクの時間単位のスケジューリング設定

InstallShield 2023 R2 では、1 時間単位のスケジュール タスクの繰り返し頻度と、各スケジュール タスクの期間を指定できるようになりました。以下の設定は、**[システム構成]** ビューの **[スケジュールされたタスク]** エクスプローラで、新しいタスクの **"スケジュール"** 設定に追加されています:

- **繰り返し間隔 (分)** ー この設定を使って、指定した時間の間に連続するタスクを実行する頻度を決定づける時間間隔を分数で指定します。この間隔は、前のスケジュール タスクの開始時点から計算されます。

  たとえば、タスクが 8:00 AM から 5:00 PM まで 1 時間間隔で実行する場合、このフィールドには 60 と入力します。

- **時間間隔 (分)** ー この設定を使って、アクティブなトリガでスケジュール タスクを繰り返す期間を決定づける時間間隔を分数で指定します。ここに指定する間隔は、**"繰り返し間隔 (分)"** 設定で指定された間隔値と等しいか、それよりも長く無くてなりません。

  たとえば、8:00 AM から5:00 PM までスケジュール タスクを実行するには、540 と入力します。

詳細については、「スケジュール」を参照してください (**[スケジュール タスク]** ビューのスケジュール タスクの設定)。

***メモ・** この設定は、**"繰り返し間隔 (分)"** 設定に間隔値が指定されているときに、"時間間隔 (分)" 設定が有効化されます。*

Standalone Build をビルド中に前提条件をダウンロードするコマンドラインをサポート



プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- 基本の MSI
- InstallScript
- InstallScript MSI

InstallShield 2023 R2 では、-prqdownload という新しいコマンドラインパラメーターが導入されました。このパラメーターは、ISCmdBld.exe を使用してコマンドラインからプロジェクトをビルドするときに、不足している InstallShield の前提条件バイナリ ファイルを該当するプロジェクトのディレクトリに自動的にダウンロードするように Standalone Build に指示します。
詳細については、「ISCmdBld.exe」を参照してください。



メモ・Iscmdbld.exe を使ったスイート/アドバンスト UI プロジェクトがこのパラメーター無しでビルドされると、ISM パッケージに必要な InstallShield 前提条件のバイナリ ファイルが自動的にプロジェクト ファイルにダウンロードされます。そのため、このパラメーターはスイート/アドバンスト UI プロジェクトには適用しません。

EV デジタル署名用トークン パスワードの保存機能

InstallShield 2023 R2 では、EV トークンのパスワードを暗号化してプロジェクト ファイルに保存するオプションが提供されています。InstallShield の以前のバージョンでは、**[証明書ストアを使用する]** オプションを使った EV 証明書がサポートされていました。ただしこのオプションを使用すると、ファイルが EV 証明書で署名されるたびにトークン パスワードが要求されます。

また、EV ベンダーに付属する [シングルログオンを有効にする] というオプションもあります。このオプションでは、1 回のトークン パスワード要求でセッションごとのユーザー介入を制限できます。InstallShield の [署名] タブでトークン パスワードを構成すると、パスワードの有効期限が切れるまで機能し続けます。トークンのパスワードは、InstallShield IDE または InstallShield オートメーション レイヤーを使って変更することができます。

詳細は、次を参照してください:

- リリースの [署名] タブ
- MSIX リリースの [全般] タブ

InstallShield 2023 R1 の新しい機能

InstallShield 2023 R1 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- スイート/アドバンスト UI プロジェクトにおける Windows パッケージ マネージャー パッケージのサポート
- スイート/アドバンスト UI プロジェクトに新しく導入された、アプリのアップデート確認機能
- InstallShield 2022 R1 における Visual Studio 2022 の統合

スイート/アドバンスト UI プロジェクトにおける Windows パッケージ マネージャー パッケージのサポート

InstallShield 2023 R1 では、スイート/アドバンスト UI プロジェクトで新しく Windows パッケージ マネージャー (WPM) パッケージがサポートされています。Windows パッケージ マネージャー は、Windows 10 と Windows 11 上にアプリケーションをインストールするためのコマンドライン ツールおよび一連のサービスで構成される総合的なパッケージ マネージャー ソリューションです。

[編成] ビューの **[パッケージ]** エクスプローラーを使って、Windows パッケージ マネージャー (WPM) パッケージをスイート/アドバンスト UI プロジェクトに追加できます。

Windows パッケージ マネージャー (WPM) パッケージをインストールするには、ターゲット システム上に Windows パッケージ マネージャー (WPM) ソリューションがインストールされている必要があります。Windows パッケージ マネージャー(WPM) ソリューションは、アプリ インストーラー MSIX バンドルでインストールされます。
Windows パッケージ マネージャー (WPM) パッケージが スイート/アドバンスト UI プロジェクトに追加されると、InstallShield は、アプリ インストーラー MSIX バンドル パッケージを依存ファイルとして追加します。ターゲット マシン上でアプリ インストーラーが使用できない場合、InstallShield はアプリ インストーラー MSIX バンドル パッケージをインストールします。Windows パッケージ マネージャー は、Windows 10 1809 からアプリ インストーラーを通してサポートされています。

詳細については、次を参照してください。

- Windows パッケージ マネージャー パッケージをスイート/アドバンスト UI プロジェクに追加する
- [パッケージ] ビュー

スイート/アドバンスト UI プロジェクトに新しく導入された、アプリのアップデート確認機能

InstallShield 2023 R1 では、スイート/アドバンスト UI セットアップ ランチャーのアップデートの設定を構成する **[アプリのアップデートを確認する]** という名前の新しい機能が、**[メディア]** ビューの **[リリース]** エクスプローラーにある **[アップデート]** タブに追加されました。

ビルドを行う前にこの新しい機能を使って、アプリケーションがターゲットマシン上に配置される時に使用可能なアプリケーションのアップデートを定期的にチェックして表示する追加設定を構成できます。最初のアップデートの確認ともない、後に続くアップデートがインストール開発環境 (IDE) で定義された間隔で表示されます。

たとえば、ビルド前に **[アプリのアップデートを確認する]** 機能の追加設定が構成されている場合、任意のアプリケーションで使用可能なアップデートがあるとき、バージョン アップデート情報と共に次の 3 つのボタンを含むダイアログ ボックスが表示されます。

- **後で通知** – 最近のアップデート情報を設定された間隔でもう一度ダイアログボックスに表示するには、このボタンをクリックします。
- **アップデートをスキップ** – 特定のアップデート情報を再び表示するダイアログ ボックスを抑制するには、このボタンをクリックします。
- **ダウンロード** – アップデートをダウンロードするには、このボタンをクリックします。

詳細については、「リリースの [アップデート] タブ」を参照してください。

InstallShield 2022 R1 における Visual Studio 2022 の統合

InstallShield 2023 R1 から、Visual Studio 2022 内で InstallScript プロジェクトの作成、編集およびビルドがサポートされています。

InstallShield 2022 の新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

- InstallShield 2022 R2 の新しい機能
- InstallShield 2022 R1 の新しい機能

InstallShield 2022 R2 の新しい機能

InstallShield 2022 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

InstallShield 2022 R2 における Visual Studio 2022 の統合

InstallShield Express 2022 R2 では Visual Studio 2022 でプロジェクト作成、編集、アップグレードおよびビルドを行うことができます。

Visual Studio 2022 と InstallShield 2022 R2 との統合についての概要は、つぎのとおりです:

- Visual Studio 2022 内で新しい InstallShield プロジェクトを作成することができます。
- プロジェクトを Visual Studio 2022 内で編集することができます。
- プロジェクトを最新版の InstallShield にアップグレードすることができます。
- Visual Studio 2022 IDE および x64 MSBuild からのプロジェクトをビルドすることができます。



***メモ** - Visual Studio 2022 からの InstallScript プロジェクトのサポートは、InstallShield の将来的なリリースで予定されています。*

InstallShield 2022 R1 の新しい機能

InstallShield 2022 R1 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- MSIX でカスタム宣言をサポート
- InstallShield 2022 R1 における Visual Studio 2022 の統合

MSIX でカスタム宣言をサポート

MSIX プロジェクトにカスタム宣言のサポートが追加されました。

MSIX プロジェクトの [宣言] ビューで、多くの組み込みアプリケーションおよびパッケージ拡張タイプがデフォルトでサポートされています。また、カスタム宣言機能を使って、InstallShield では提供されていない MSIX 拡張機能を定義することもできます。[カスタム宣言] メニューでは、名前空間およびコンテンツを指定して必要な拡張タイプを定義できるオプションが提供されています。InstallShield ビルド プロセスは、MSIX マニフェスト ファイルの準備中にこれらのカスタマイズを考慮に入れます。詳細については、InstallShield 製品マニュアルを参照してください。

InstallShield 2022 R1 における Visual Studio 2022 の統合

InstallShield は Visual Studio 2022 との統合をサポートします。この統合は段階的な提供が予定されています。InstallShield 2022 R1 では Visual Studio 2022 でプロジェクトを開き、アップグレードおよびビルドを行うことができます。

Visual Studio 2022 と InstallShield 2022 R1 との統合についての概要は、つぎのとおりです:

- 既存するプロジェクトを Visual Studio 2022 で開くことができます。
- プロジェクトを最新版の InstallShield にアップグレードすることができます
- Visual Studio 2022 IDE および x64 MSBuild からのプロジェクトをビルドすることができます。



***メモ・** Visual Studio 2022 は以前のリリースの 32 ビットとは異なり、64 ビット アプリケーションとしてリリースされているため、段階的な統合が行なわれています。*



***メモ・** InstallScript プロジェクトおよび Visual Studio 2022 から新しい InstallShield プロジェクトを作成できる機能は、InstallShield の将来的なリリースでサポートされます。。*

InstallShield 2021 の新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

- InstallShield 2021 R2 の新しい機能
- InstallShield 2021 R1 の新しい機能

InstallShield 2021 R2 の新しい機能

InstallShield 2021 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

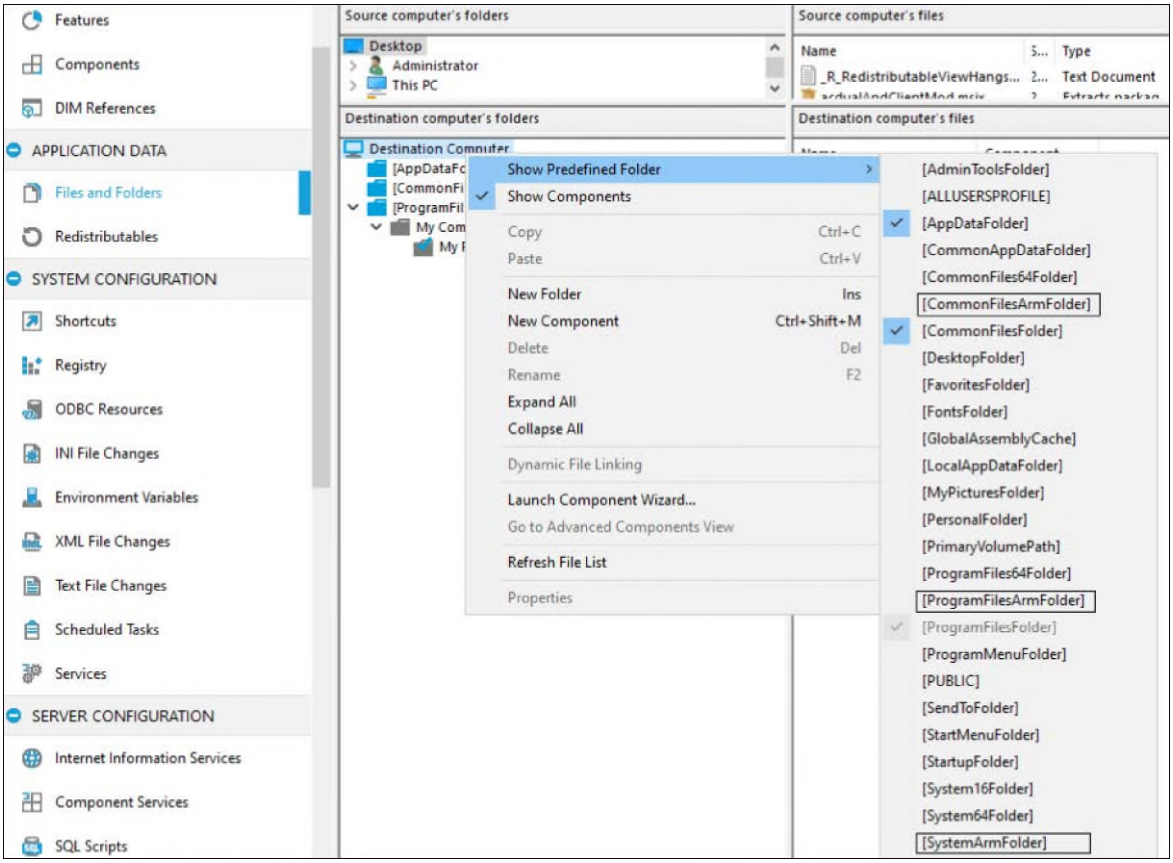
- Windows 11 および Windows Server 2022 のサポート強化
- 完全な ARM サポート

Windows 11 および Windows Server 2022 のサポート強化

Windows 11 および Windows Server 2022 をターゲットに特定してインストール条件を構成することができます。

完全な ARM サポート

ARM64 ベースのデバイスをターゲットとするソフトウェア インストーラで、今回より ARM マシン上で実行中の Windows 10 のネイティブ ARM の場所にファイルをインストールすることができます。InstallShield を使って、ARM 固有のフォルダーを定義済みフォルダーから InstallShield プロジェクトに追加することができます。実行時に ARM 固有のフォルダーにファイルをコピーするためには、Windows Installer バージョン 5.0 が必要で、またテンプレート概要が Arm64 をポイントしなくてはなりません。このため、[一般情報] ビューでスキーマ バージョンを 500、およびテンプレート概要を Arm64 に変更する必要があります。



InstallShield 2021 R1 の新しい機能

InstallShield 2021 R1 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- Cloud License Server (CLS) を使ったライセンシング
- 前提条件における Windows Package Manager のサポート
- MSIX アップデート向けのアプリ インストーラー ファイルのサポート
- シンプルでモダンなインストーラー
- Windows 11 のサポート

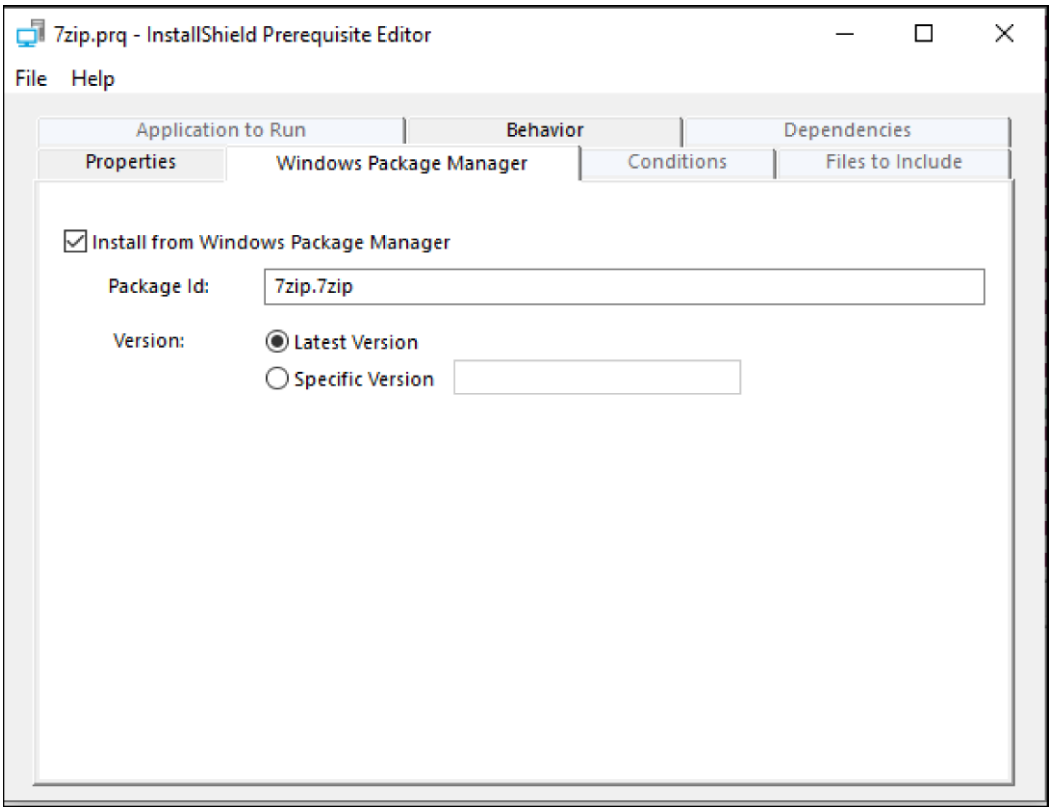
Cloud License Server (CLS) を使ったライセンシング

InstallShield は、Cloud License Server (CLS) を使った同時接続ライセンスを提供します。

Cloud License Server (CLS) は、Revenera FlexNet Cloud Licensing Service でホストされ、InstallShield が使用するライセンス権利を保管します。Cloud License Server (CLS) を使用することで、ローカル ライセンス サーバーをセットアップおよび管理するのに比べて時間と労力が削減されるだけでなく、複数のマシンからライセンスをチェックアウトできる同時接続ライセンスを使用でき、また DepOps ビルド パイプラインからのビルドを簡単に構成することができます。

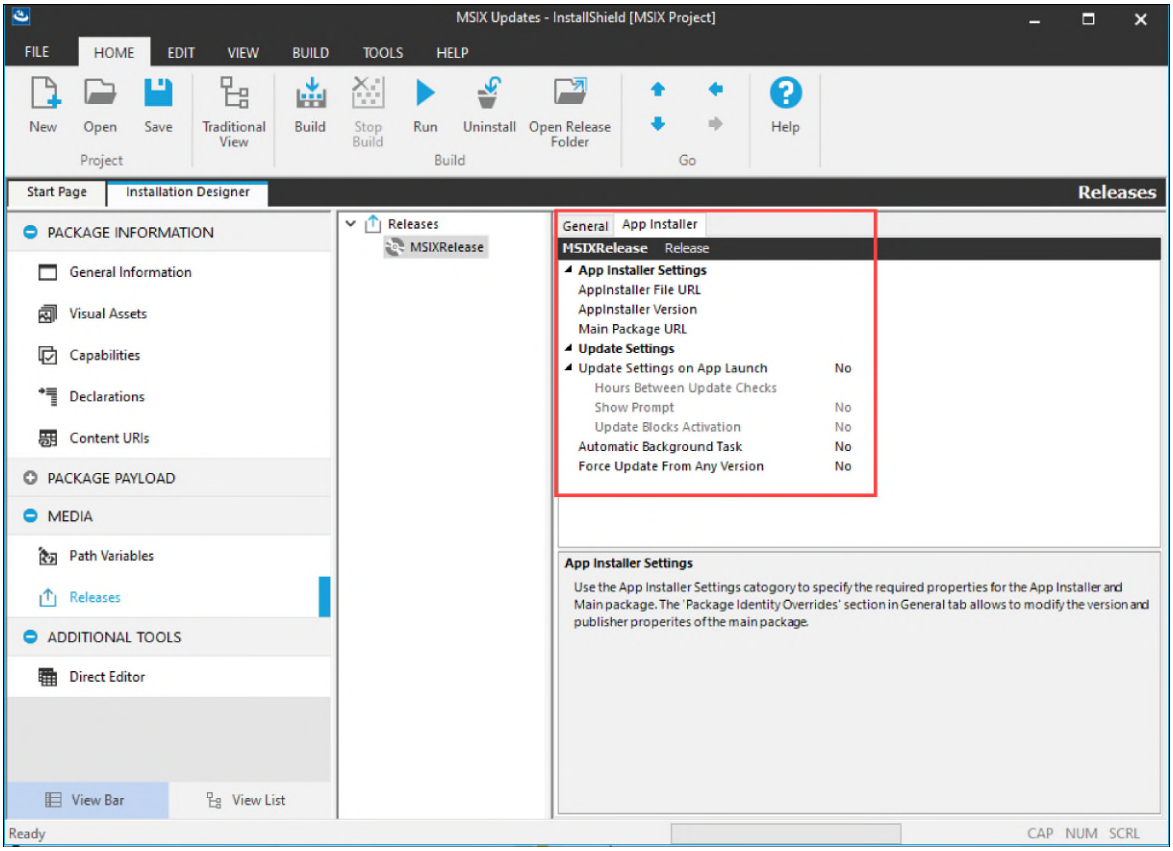
前提条件における Windows Package Manager のサポート

InstallShield 2021 R1 では、InstallShield 前提条件エディターに [Windows Package Manager] タブが追加されています。[Windows Package Manager] タブを使って、製品に前提条件としてインストールしたい Windows Package Manager (WPM) パッケージを指定することができます。



MSIX アップデート向けのアプリ インストーラー ファイルのサポート

今回より、アプリ インストーラー ファイルを使って MSIX アップデートを構成することができます。



シンプルでモダンなインストーラー

[ウィザード インターフェイス] ビューの強化機能として [最小ウィザード ページ] が追加されました。このページを使ってフレームを使わない画面でエンド ユーザーにモダンなインストール/アンインストール エクスペリエンスを提供することができます。

制限

スイート プロジェクトにおける最小ウィザード ページのサポートに関する制限事項:

- 最小ウィザード ページのレイアウトのカスタマイズはサポートされていません。ページおよびコントロールの追加/削除を行うことはできません。
- 最小ウィザード ページ上のボタン クリックのアクションを指定することはできません。
- インストールのパスワード保護は、現在サポートされていません。
- セットアップのインストール中、タスクバーに表示される進行状況は、このリリースではサポートされていません。
- Windows 7 上の InstallShield IDE で最小ウィザード ページをプレビューすることはできません。

Windows 11 のサポート

InstallShield 2021 R1 で作成されたセットアップは、Windows 11 バージョン上で実行可能です。

InstallShield 2020 の新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

- InstallShield 2020 R3 SP1 の新しい機能
- InstallShield 2020 R3 の新しい機能
- InstallShield 2020 R2 の新しい機能
- InstallShield 2020 の新しい機能

InstallShield 2020 R3 SP1 の新しい機能

- InstallShield 前提条件で MSIX パッケージおよびバンドルをサポート

InstallShield 前提条件で MSIX パッケージおよびバンドルをサポート

InstallShield 前提条件エディターは、次の MSIX パッケージおよびバンドルをサポートします。

- InstallShield 前提条件に MSIX パッケージおよびバンドルを追加または構成する。
- 検出条件に基づいて、選択された MSIX パッケージまたはバンドルをインストールする。

InstallShield 2020 R3 の新しい機能

InstallShield 2020 R3 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- InstallShield Azure DevOps ビルド拡張
- [ファイルとフォルダー] ビューのショートカット オプション
- MSIX フォント

InstallShield Azure DevOps ビルド拡張

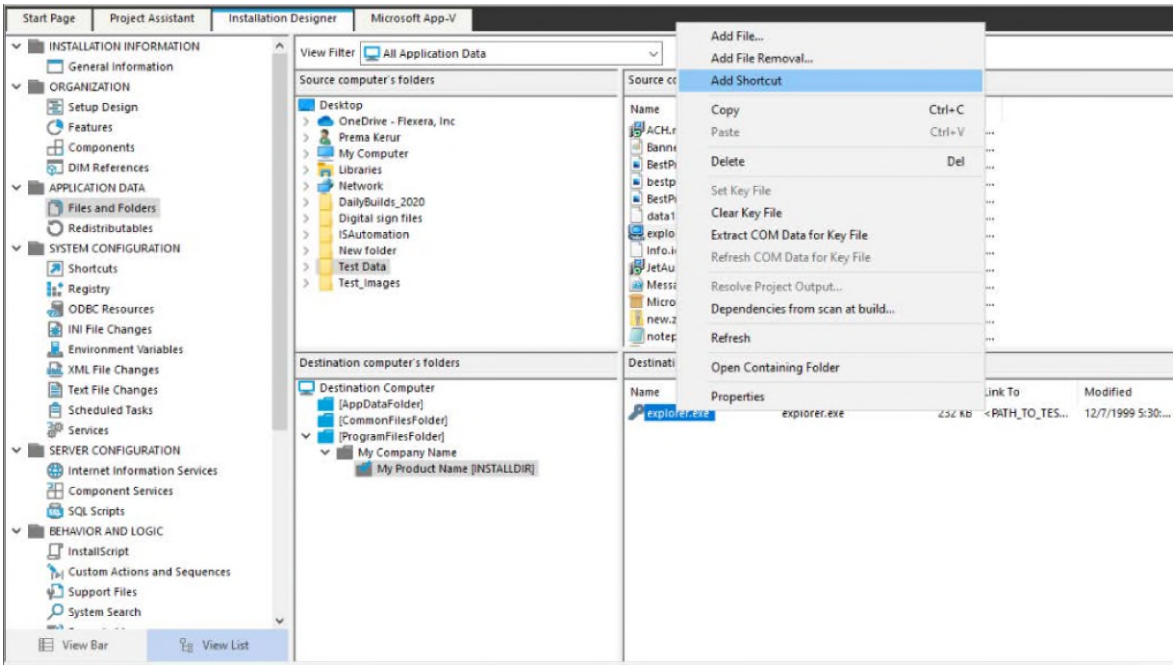
InstallShield 2020 R3 では、Azure DevOps Pipelines で InstallShield プロジェクトをビルドするビルトイン拡張が追加されました。タスクを構成して InstallShield プロジェクトをビルドするには、「InstallShield Azure DevOps Build Extension (InstallShield Azure DevOps ビルド拡張)」ナレッジベース記事 (英語) を参照してください。

[ファイルとフォルダー] ビューのショートカット オプション

基本の MSI プロジェクトで、[ファイルとフォルダー] ビューにショートカットの作成オプションが追加されました。

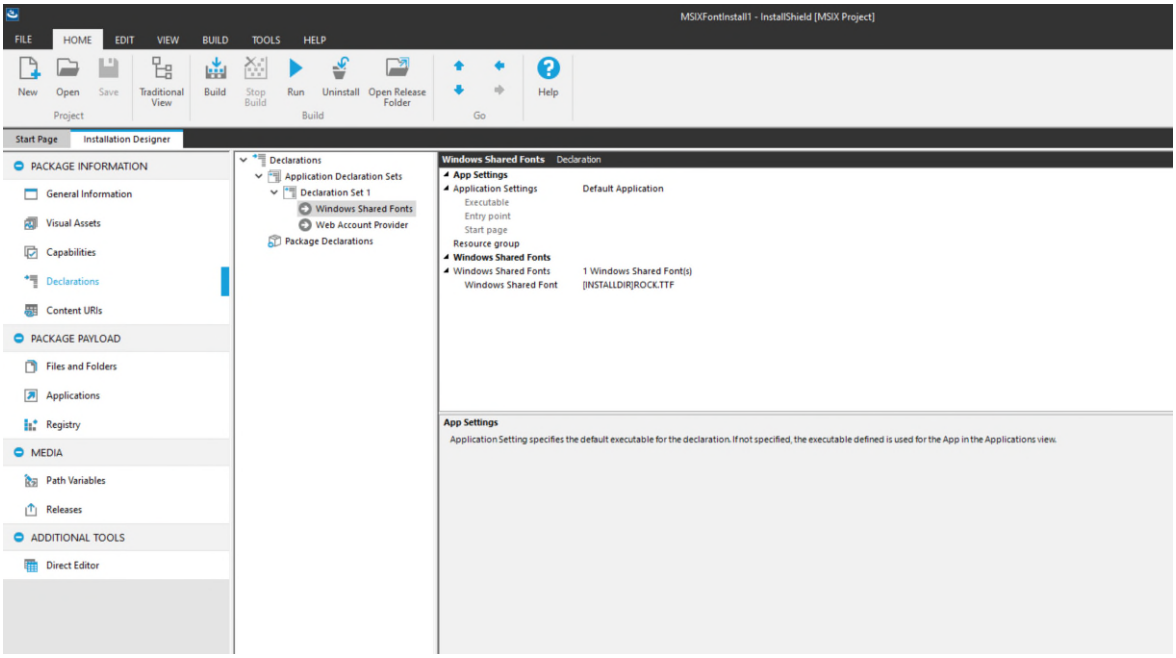
[ファイルとフォルダー] ビューからショートカットを作成するには、次の手順に従います。

1. 実行可能ファイルを右クリックします。
2. [ショートカットの追加] オプションをクリックします。



MSIX フォント

InstallShield では、MSIX プロジェクトで「Windows 共通フォント」宣言がサポートされています。この宣言は、[宣言] ビューから MSIX プロジェクトに追加することができ、カスタム フォントをインストールし、システム上のその他のアプリケーションとそれを共有する事ができます。



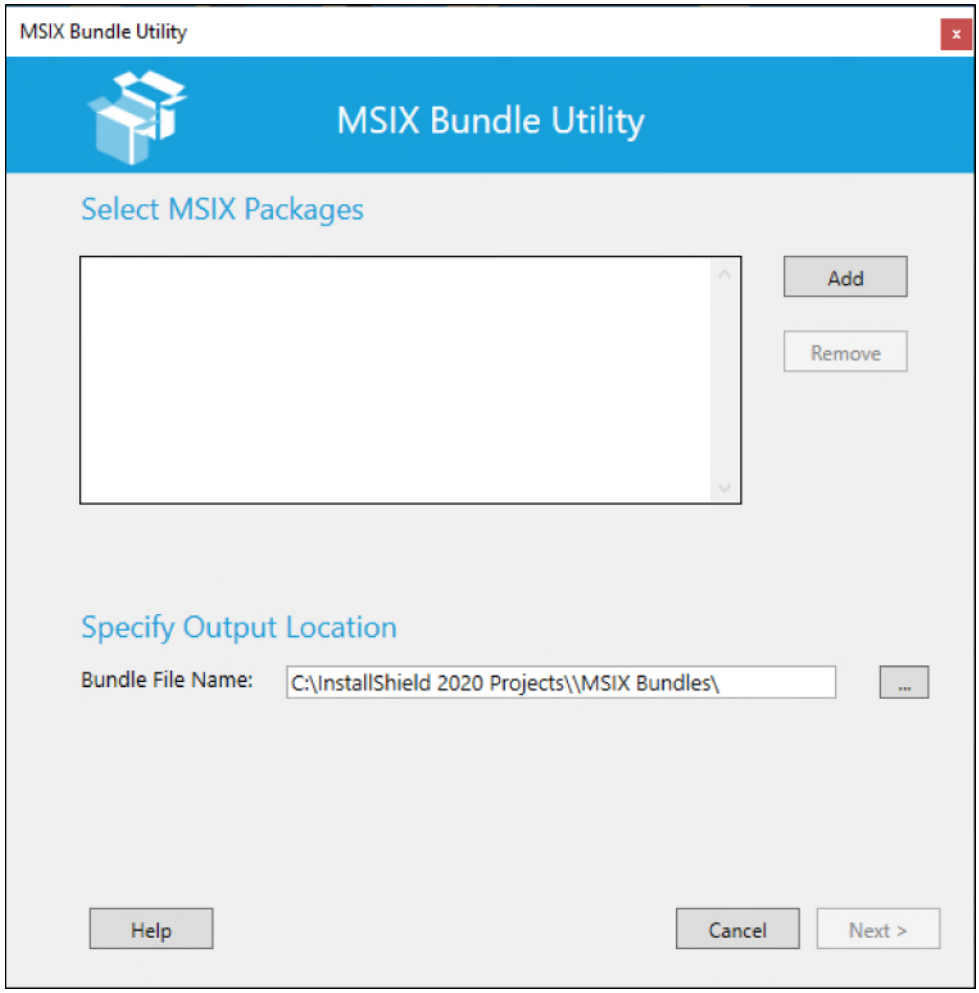
InstallShield 2020 R2 の新しい機能

InstallShield 2020 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- MSIX バンドル
- MSIX – サービスのサポート
- スイート/アドバンスト UI 向けの MSIX バンドルまたは AppX バンドル パッケージ

MSIX バンドル

新しい MSIX バンドル ユーティリティを使って MSIX バンドルを作成できます。アーキテクチャ固有のパッケージをユーティリティに追加し、署名情報を入力して MSIX バンドルを作成します。



MSIX – サービスのサポート

今回より、新しい宣言をパッケージに追加することで、サービスを含む MSIX パッケージを作成できるようになりました。また、基本の MSI プロジェクトに既存するサービスを MSIX パッケージに変換することもできます。

スイート/アドバンスト UI 向けの MSIX バンドルまたは AppX バンドル パッケージ

MSIX バンドルを作成できる機能に加えて、それをスイート/アドバンスト プロジェクトに追加し、他のパッケージと共にインストールすることも可能になりました。

InstallShield 2020 の新しい機能

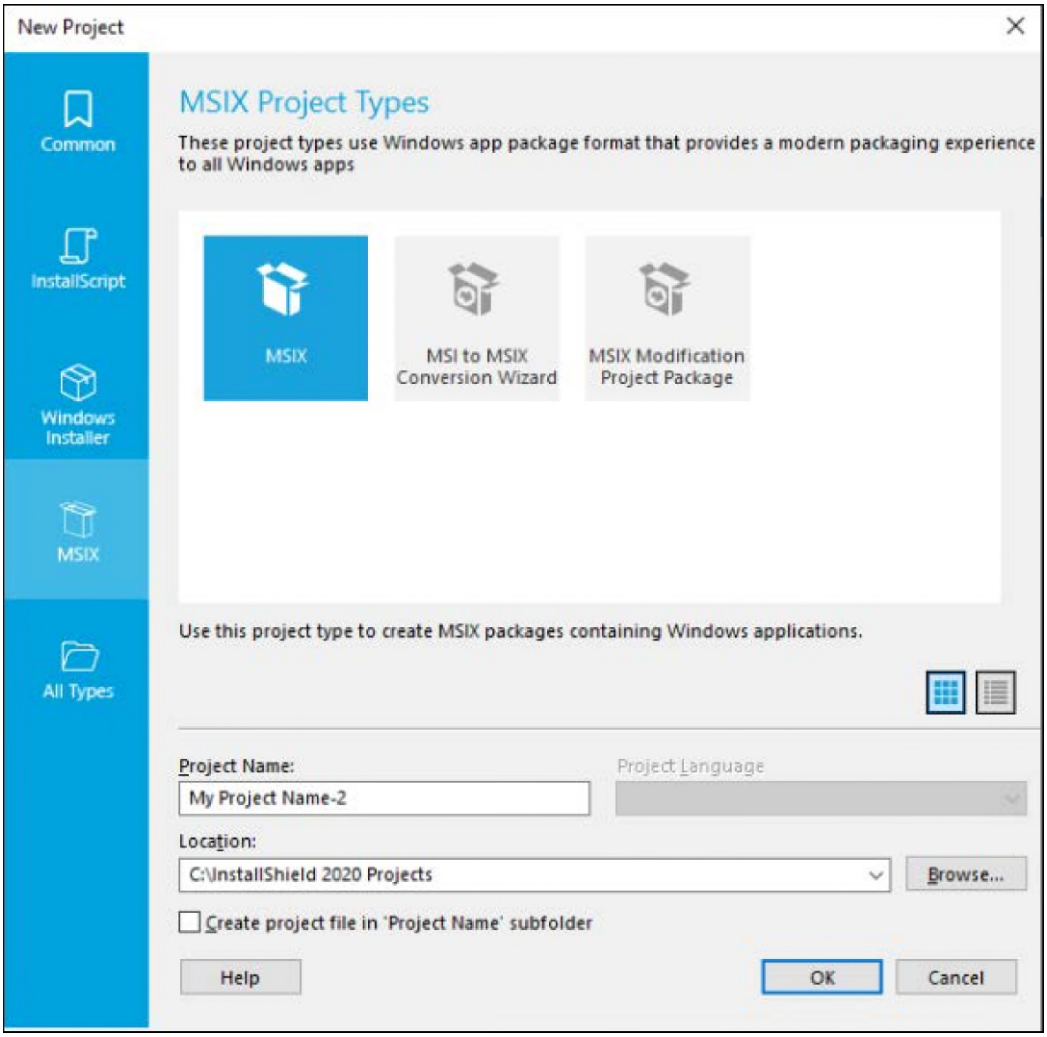
InstallShield 2020 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- MSIX 変更パッケージ
- MSIX コアのサポート
- ピュア 64 ビット インストーラー
- AWS CloudHSM ベースのデジタル署名をサポート

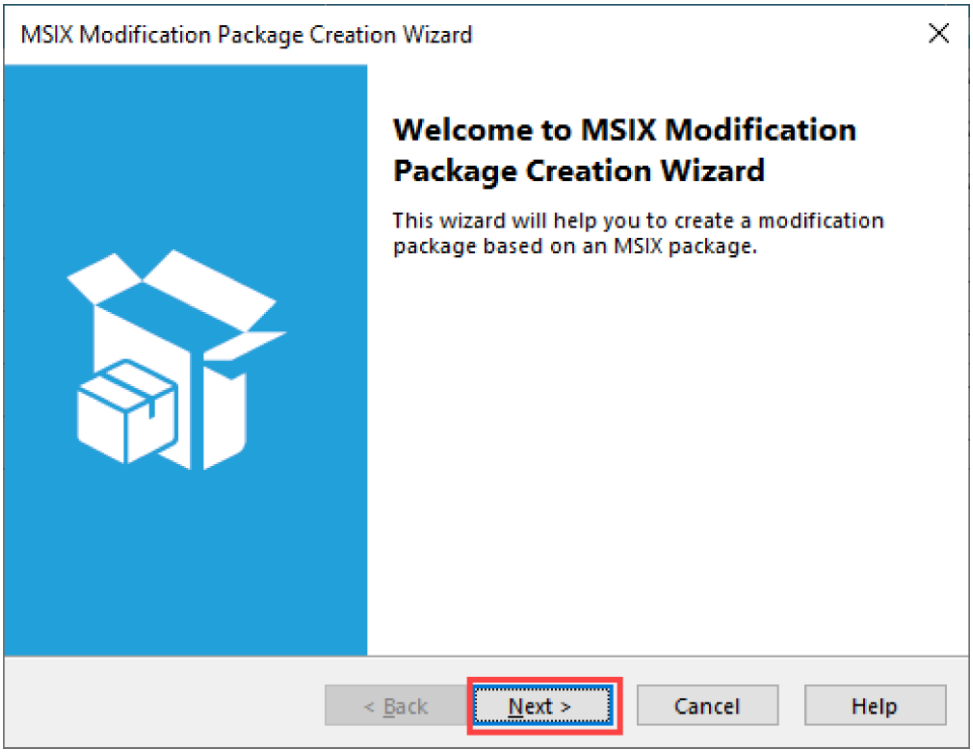
MSIX 変更パッケージ

今回より InstallShield を使って、任意の MSIX パッケージの変更パッケージを作成することができます。次の方法で変更パッケージを作成します:

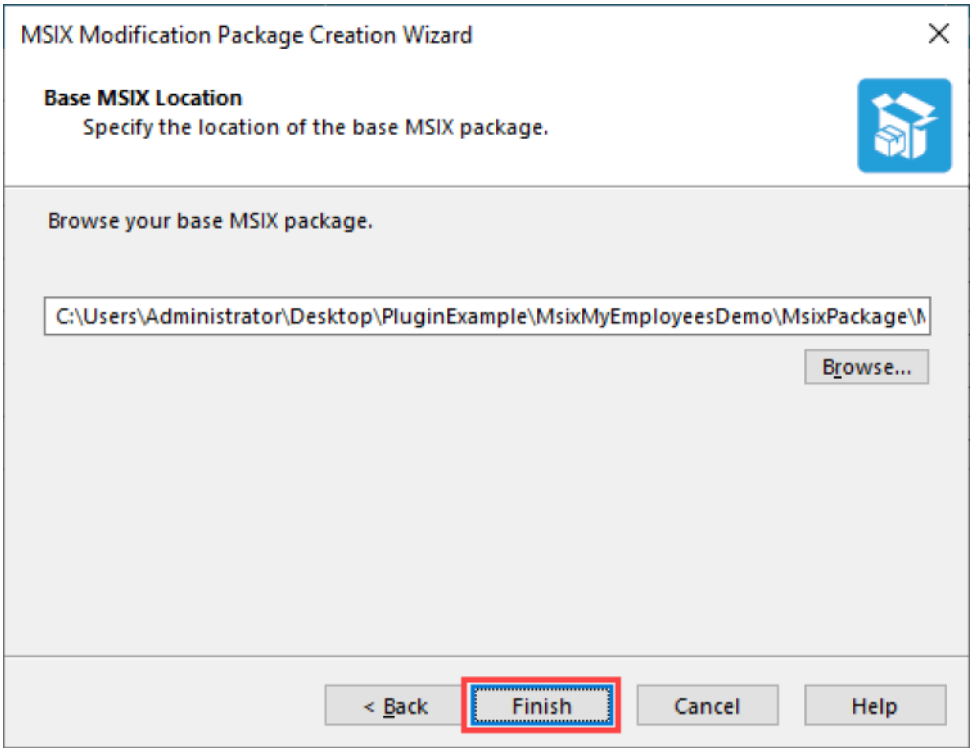
1. [新規プロジェクト] ダイアログから新しい変更パッケージを作成する。



2. MSIX 変更パッケージ作成ウィザードで [次へ] をクリックします。



3. ベース MSIX パッケージを参照します。



4. [ファイルとフォルダー] ビューで、更新パッケージに追加したい新しいファイルをすべて追加します。
5. [レジストリ] ビューでは、変更パッケージに含める任意のエントリを追加または変更することができます。

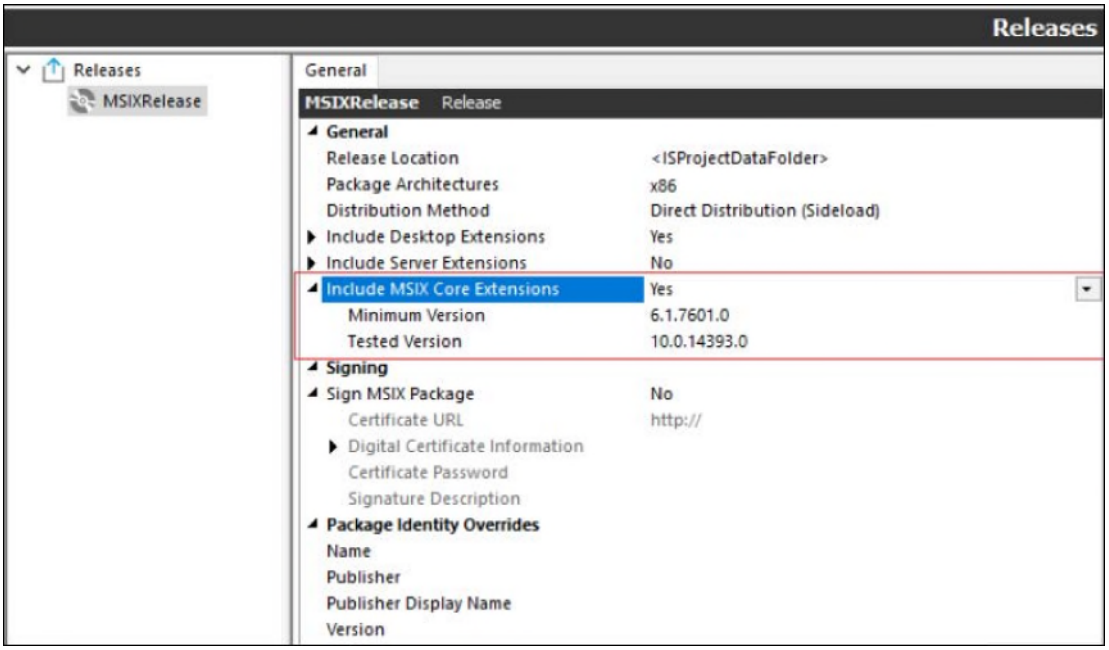
6. [リリース] ビューでは、デジタル署名情報を追加します。

7. プロジェクトをビルドします。
これは変更パッケージを生成します。

MSIX コアのサポート

InstallShield 2020 R1 より、Windows 7 および Windows 10 の古いバージョン (1709 以前) をターゲットとする MSIX パッケージを作成およびインストールすることができます。さらに、サポートされているすべてのプラットフォームに MSIX パッケージをインストールする単一のスイート インストーラーを作成することも可能です。

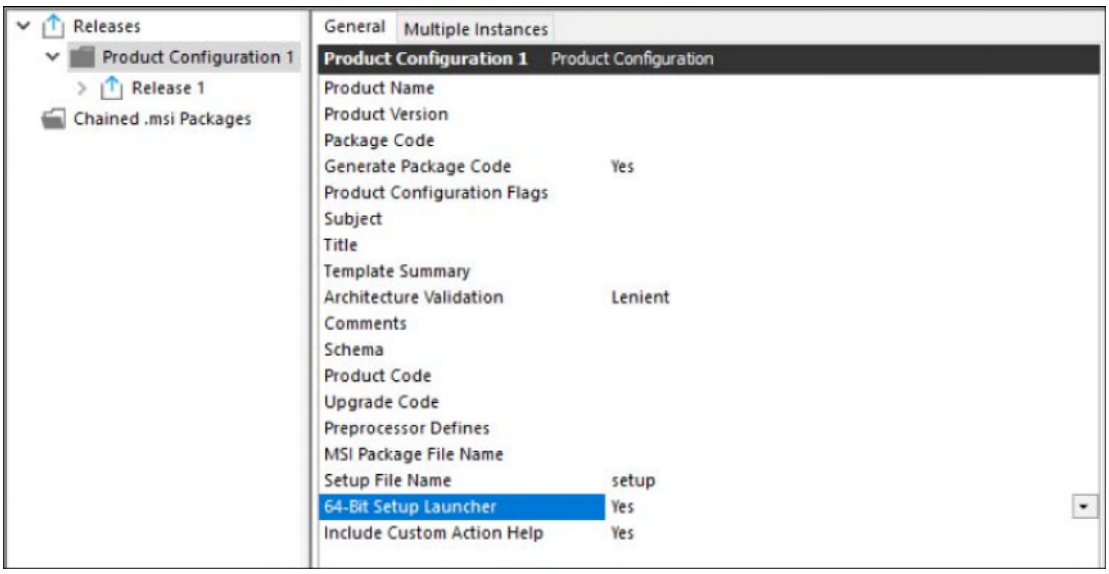
基本の MSI プロジェクトまたは MSIX プロジェクトで MSIX パッケージを作成するとき、[リリース] ビューの値を次のように設定してください:



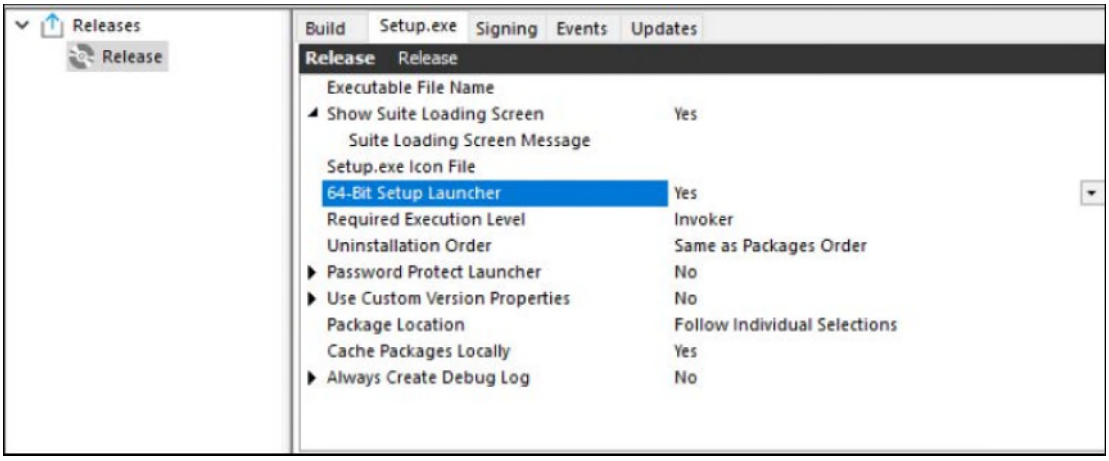
ピュア 64 ビット インストーラー

2020 R1 より、基本の MSI プロジェクトおよびスイート プロジェクトで 64 ビット ランチャーを使ったインストーラーを作成することができます。

1. 基本の MSI プロジェクトでピュア 64 ビット インストーラーを作成するには、[製品の構成] ビューに移動して [64 ビット セットアップ ランチャー] を選択します。



2. スイート プロジェクトでピュア 64 ビット インストーラーを作成するには、[リリース] ビューに移動して [64 ビット セットアップ ランチャー] を選択します。



AWS CloudHSM ベースのデジタル署名をサポート

InstallShield を使って、今回より AWS CloudHSM ベースのデジタル署名を使ってインストーラーにデジタル署名を行うことができます。この機能を有効化するには、次のプロパティを <<InstallShield_Location>/Support/<0409¥0411>Settings.xml ファイルに追加してください。

```
<!-- Specify Platform = X86 | X64 for Digital Signing -->
<DigitalSignature Platform="X64"/>
```

InstallShield 2019 の新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

- InstallShield 2019 R3 の新しい機能
- InstallShield 2019 R2 の新しい機能
- InstallShield 2019 の新しい機能

InstallShield 2019 R3 の新しい機能

InstallShield 2019 R3 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- Docker コンテナ上の Standalone Build
- ファイル リダイレクト fixup のサポート
- OLE DB Driver for SQL Server の使用
- データベース サーバーの SQL ドライバーを選択する
- オートメーション インターフェイスを使って [パッチ デザイン] のパッチを構成できる機能

Docker コンテナ上の Standalone Build

今回より、InstallShield Standalone Build (SAB) を Docker コンテナ内にインストールおよび実行することが可能です。

Docker Hub 上に InstallShield 2019 R3 Standalone Build をセットアップする方法について、または Docker 上にセットアップ済みの InstallShield 2019 R3 Standalone Build をダウンロードする方法については、「InstallShield StandAlone Build with Docker (Docker を使った InstallShield スタンドアロン ビルド)」ナレッジベース記事に掲載されています。

ファイル リダイレクト fixup のサポート

以前のリリースで、InstallShield に MSIX のファイル リダイレクトおよびカスタム fixup を含むパッケージ サポート フレームワークのサポートが追加されました。

InstallShield 2019 R3 リリースでは、ファイル リダイレクト fixup パラメーターの実行可能ファイル名をカスタマイズすることができるように機能が強化されています。この強化は、複数プロセスの起動を行う実行可能ファイルに適用します。

以前のリリースでは、リダイレクト fixup パラメーターの実行可能ファイルは、プライマリ プロセス名と同じ名前が使用されました。InstallShield 2019 R3 には、MSIX プロジェクト タイプの **[アプリケーションの設定]** ビューにある **[パッケージ サポート フレームワーク]** 領域に、新しい **"実行可能ファイル"** フィールドが追加されました。ここでリダイレクト fixup パラメーターの実行可能ファイル名を指定することができます。デフォルトで、値としてプライマリ プロセス名が入力されます。

OLE DB Driver for SQL Server の使用

以前の Microsoft OLE DB Provider for SQL Server (SQLOLEDB) および SQL Server Native Client OLE DB provider (SQLNCLI) は、引き続き使用できません。

現在、Microsoft から Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) と呼ばれる新しい OLE DEB プロバイダーがリリースされています。

InstallShield 2019 R3 では今回より、Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) を使って TLS 1.2 のみの環境をサポートします。



***上級者向けメモ・** Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) は、InstallShield 2019 R3 に前提条件として含まれています。*

データベース サーバーの SQL ドライバーを選択する

InstallShield 2019 R3 では今回より、使用するデータベース サーバー用のドライバー一覧から、ドライバーを選択することができます。

デフォルトで、新しい SQL 接続には、[要件] タブで Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) が表示されます。

オートメーション インターフェイスを使って [パッチ デザイン] のパッチを構成できる機能

以前は、オートメーション インターフェイスを使って [パッチ ザイン] のパッチを構成することができませんでした。オートメーションでは、既に作成済みのパッチのみをビルドすることができました。

InstallShield 2019 R3 リリースから、[パッチ デザイン] のパッチをオートメーション インターフェイスを使って構成することができます。

InstallShield 2019 R2 の新しい機能

InstallShield 2019 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- MSIX – パッケージ サポート フレームワーク

MSIX – パッケージ サポート フレームワーク

パッケージ サポート フレームワークは、ソース コードにアクセスできない場合でも既存する win32 アプリケーションに修正を適用できるキットなので、MSIX コンテナ内でこれを実行することができます。

これを使って、アプリケーションにモダンな実行環境のベスト プラクティスを適用できます。パッケージ サポート フレームワークには、実行可能ファイル、ランタイム マネージャー DLL、およびランタイム修正プログラムのセットが含まれます。

ユーザーがアプリケーションを起動すると、パッケージ サポート フレームワーク ランチャーが最初に実行されます。ランチャーによって構成ファイルが読み取られ、ランタイム修正プログラムとランタイム マネージャー DLL がアプリケーション プロセスに挿入されます。ランタイム マネージャーは、MSIX コンテナー内で実行するアプリケーションで必要になったときに修正プログラムを適用します。

InstallShield 2019 R2 では、ファイル リダイレクト fixup およびカスタム fixup などのランタイム修正を含むパッケージ サポート フレームワークを MSIX で使用できます。

- ファイル リダイレクト fixup – ファイル リダイレクト fixup を使って、MSIX コンテナ内で実行するアプリケーションからアクセスできないディレクトリ内にデータを書き込みまたは読み取りしようとする場合にリダイレクトを行います。
- カスタム fixup – カスタム リダイレクト fixup を使って、アプリケーションに適用する修正を指定する構成ファイルとアプリケーションの互換性問題を解決できます。

InstallShield 2019 の新しい機能

InstallShield 2019 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- MSIX プロジェクト タイプを使って MSIX パッケージをビルドする
- MSI からMSIX への変換ウィザード
- 基本の MSI の MSIX 検証スイートをビルド
- スイートへの MSIX の追加
- 新しいプロジェクトウィザード

MSIX プロジェクト タイプを使って MSIX パッケージをビルドする

MSIX は Windows アプリ パッケージ形式で、すべての Windows アプリにモダンなパッケージ エクスペリエンスを提供します。MSIX パッケージ形式は、既存アプリ パッケージおよび/またはファイルのインストール機能をはそののままに、Win32、WPF、および WinForm アプリに新しくモダンなパッケージ作成およびデプロイメント機能を追加します。

InstallShield 2019 では、 MSIX プロジェクト タイプを使って MAIX パッケージをビルドすることができます。サイドロードまたは Windows ストア経由で配布を行う Windows アプリケーションが含まれます。

- パッケージ情報
- パッケージ ペイロード
- メディア

パッケージ情報

今回より、次を使ってプロジェクトを説明および識別することができます:

- **[一般情報]** ビューでは、プロジェクト ファイルの名前やフォーマットなどの詳細を指定します。
- **[MSIX ビジュアル資産]** エクスプローラーでは、統合的に視覚的な方法で、MSIX アプリのビジュアル的な側面を説明することができます。
- **[アプリ機能]** ビューは、特定の API または画像、音楽、またはカメラやマイクなどのデバイスといったリソースにアクセスできるよう、アプリのパッケージ マニフェスト内で有効化さている必要があります。
- **[宣言]** ビューは、アプリケーション、パッケージ レベルの宣言を作成および管理できるビジュアル ツールを提供し、それらのプロパティを構成することができます。
- **[コンテンツ URI]** ビューでは、window.external.notify を使って Script-Notify イベントをアプリに送る URI を指定します。

パッケージ ペイロード

今回より、MSIX アプリケーションをデザインするファイルを指定することができます:

- **[ファイルとフォルダー]** ビューで、InstallShield プロジェクトにファイルを追加することができます。ターゲット上にあるフォルダーへこれらのファイルを整理することができます。
- **[アプリケーション]** ビューを使って、パッケージで提供される機能の一部またはすべてを構成する Windows アプリのプロパティを視覚的および統合的にデザインすることができます。
- **[レジストリ]** ビューでは、インストールが作成するレジストリキーおよびレジストリ値を定義することができます。

メディア

今回より、インストールを配布するのに利用するファイルとフォルダーをカスタマイズできます:

- **[パス変数]** ビューでは、変数を使って開発システム間で簡単にインストールを移動させることができます。
- **[リリース]** ビューには、プロジェクトに含まれる各製品のビジュアルツールがあります。

MSI からMSIX への変換ウィザード

MSI からMSIX への変換ウィザードを使って、既存の Windows Installer ファイル（.msi ファイル）を基本の MSI プロジェクトを通して MSIX パッケージに変換することができます。

基本の MSI の MSIX 検証スイートをビルド

今回より、Microsoft MSIX 標準に準拠する必要がある既存の InstallShield プロジェクトを識別することができます。また、MSIX で動作するために必要な MSI プロジェクト要素の変換に関する情報や手順も提供します。

今回より InstallShield には、InstallShield MSIX 適合性スイートという名前の 1 セットの検証ツールが含まれています。

このスイートに含まれる InstallShield MSIX 適合性検証ツール は、.msi パッケージ内で MSIX パッケージ（..msix）フォーマットに適さないアイテムの存在をスキャンします。

スイートへの MSIX の追加

InstallShield 2019 では、スイートに MSIX パッケージを追加して、他のデプロイメント パッケージ タイプと共にインストールすることが可能です。

新しいプロジェクトウィザード

新しいプロジェクト ウィザードを使って、異なるプロジェクト タイプを容易に選択できます。

新しいプロジェクト ウィザードには、次の異なるタブがあります:

- 良く使うプロジェクト タイプ – よく使用されるプロジェクト タイプを使用します。
- InstallScript プロジェクト タイプ – InstallShield Professional で InstallShield セットアップ エンジンを使ってビルドされたプロジェクトを使用します。
- Windows Installer プロジェクト タイプ – InstallShield で Microsoft Windows Insatller（MSI）セットアップ エンジンを使ってビルドされたプロジェクトを使用します。
- MSIX プロジェクト タイプ – Windows アプリ パッケージ形式を使用します。
- すべてのプロジェクト タイプ – 使用可能なすべてのプロジェクト タイプおよび任意のカスタム テンプレートを使います。

InstallShield 2019 R2 強化機能

InstallShield 2019 R2 には、次の強化機能が含まれています:

- 連続する署名の遅延
- IsCmdBld.exe パラメーターを使ってデジタル証明書パスワードを指定する
- スイート プロジェクトで、ISM パッケージへのリンクに相対パスを使用可能にする
- SQL 2012 Native Client 前提条件のアップデート

- PRQ 条件に Windows Server 2019 を追加

連続する署名の遅延

InstallShield 2019 R2 では、連続するデジタル署名の間に遅延を設定することができます。この機能は、タイムスタンプ サーバーが連続する署名要求の処理に失敗した場合のみ必要になります。

Settings.xml の <DevStudio/Build> ノードの下にある <DelayBetweenSigning default="1500"/> ノードで、ミリ秒単位で指定する必要があります。

InstallShield と共にインストールされている Settings.xml ファイルを検出します。Settings.xml は、使用している InstallShield の言語バージョンに応じて、次のいずれかの場所にインストールされています:

- 英語—InstallShield Program Files Folder¥Support¥0409
- 日本語—InstallShield Program Files フォルダー¥Support¥0411

IsCmdBld.exe パラメーターを使ってデジタル証明書パスワードを指定する



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*
- *アドバンスト UI*

InstallShield 2019 R2 には、デジタル証明書パスワードを指定するための新しいパラメーター (cert_password) が含まれています。これはオプション パラメーターで、コマンドライン ビルド (IsCmdBld.exe) で使用できます。

指定されなかった場合、ファイルの署名にはプロジェクトで構成されたパスワードが使用されます。

スイート プロジェクトで、ISM パッケージへのリンクに相対パスを使用可能にする



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*

InstallShield 2019 R2 では今回より、変更済みのプロジェクトをビルドするとき、相対パスを使って ISM パッケージにリンクすることができます。

SQL 2012 Native Client 前提条件のアップデート



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*

- *InstallScript MSI*

InstallShield 2019 R2 には、最新バージョンの 2012 Native Client 用の Microsoft SQL Server 2012 Native Client 前提条件 (x86 および x64) が含まれています。

PRQ 条件に Windows Server 2019 を追加



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*

InstallShield 2019 R2 では、前提条件の条件を Windows Server 2019 の特定のプラットフォーム上で実行するように設定できます。オプションを選択/選択解除して、前提条件エディターで適切な PRQ 条件を設定することができます。

InstallShield 2019 の強化機能

InstallShield 2019 には、以下の強化機能が含まれています:

- 選択された証明書の詳細を表示する
- オペレーティング システム要件として Windows Server 2019 を追加
- 定義済みインストール条件の追加
- デジタル署名におけるタイムスタンプ サーバーの更新
- MSBuild を使ったスイート プロジェクトのサポート
- デバッグ ログ ファイル名のカスタマイズ
- カスタム リターン コードのカスタマイズ
- セットアップ ファイル名のカスタマイズ
- InstallShield Standalone Build のサポート

選択された証明書の詳細を表示する



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*
- *QuickPatch*
- *MSIX*

InstallShield 2019 では、証明書の全般的な情報や、セキュリティ、証明書パスなどの情報が **[証明書の選択]** ダイアログ ボックスの **[詳細表示]** オプションに一覧表示されます。

オペレーティング システム要件として Windows Server 2019 を追加



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

InstallShield 2019 では、オペレーティング システム要件セクションに Windows Server 2019 のオプションを追加できるようになりました。オプションを選択/選択解除して、プロジェクトに適切なインストール条件を設定することができます。

定義済みインストール条件の追加



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

InstallShield に新しい定義済みシステム検索が追加されました:

- Microsoft .NET Framework 4.6.2

インストールで上記が必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、このシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

デジタル署名におけるタイムスタンプ サーバーの更新



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallShield 2019 で

- SHA-1 を使ってパッケージに署名を行うと、パッケージは次を使ってタイムスタンプを取得します:

  <DigitalSignature Timestamp="http://timestamp.verisign.com/scripts/timstamp.dll"/>

- SHA-256 を使ってパッケージに署名を行うと、パッケージは次を使ってタイムスタンプを取得します:

  <DigitalSignature TimestampRFC3161="http://sha256timestamp.ws.symantec.com/sha256/timestamp"/>

MSBuild を使ったスイート プロジェクトのサポート



***プロジェクト・**この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

InstallShield では、.NET Framework に含まれている Microsoft ビルド エンジン (MSBuild) がサポートされています。MSBuild サポートを使って、Visual Sutudio がビルド ラボ環境にインストールされていない場合でも、InstallShield で Visual Studio ソリューションをビルドすることができます。

InstallShield 2019 では今回より、MSBuild を使用することで、Visual Studio で作成されたスイート プロジェクト (.sln) をビルドすることができます。

デバッグ ログ ファイル名のカスタマイズ



***プロジェクト・**この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトで **[リリース]** ビューの **[Setup.exe]** タブに新しいオプション **[常にデバッグ ログを作成する]** が追加されました。

InstallShield 2019 では今回より、生成されるデバッグ ログ ファイル名を指定することができます。

カスタム リターン コードのカスタマイズ



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*
- *InstallScript*

InstallShield 2019 では今回より、実行時に Setup.exe のリターン コードを独自のカスタム値で上書きすることができます。

セットアップ ファイル名のカスタマイズ



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*
- *InstallScript*
- *スイート/アドバンスト UI*

InstallShield 2019 では、Setup.exe の元のファイル名プロパティをカスタマイズすることができます。

InstallShield Standalone Build のサポート

今回より、InstallShield Standalone Build をインストールした場合、オートメーション インターフェイス機能がデフォルトで選択されています。

これによって、手動で選択せずに、オートメーション インターフェイス機能をインストールすることができます。

追加された前提条件

InstallShield 2019 には、以下の前提条件が追加されています:

- Microsoft .Net Core 2.1 ランタイム 前提条件

Microsoft .Net Core 2.1 ランタイム 前提条件

InstallShield の [再配布可能ファイル] ビューには、今回より Microsoft .Net Core 2.1 ランタイム 前提条件が追加されています。

InstallShield 2018 R2 の新しい機能

InstallShield 2018 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- InstallShield IDE を使った MSIX パッケージのビルド

InstallShield IDE を使った MSIX パッケージのビルド



プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *基本の MSI*
- *InstallScript MSI*

InstallShield 2018 R2 では、InstallShield IDE を使って MSIX パッケージをビルドすることができます。

InstallShield 2018 SP1 の新しい機能

InstallShield 2018 SP1 で追加された強化機能についての詳細は、次のセクションを参照してください:

- スイート/アドバンスト UI インストールでダウンロード可能なアップデートを構成する
- SHA1 & SHA256 を使用するデュアル署名のサポート
- スイート読み込み画面のメッセージを構成する
- アップデート ランチャー名のカスタマイズ
- PowerShell カスタム アクションで文字スペースを指定する
- 定義済みパブリック フォルダーの指定

スイート/アドバンスト UI インストールでダウンロード可能なアップデートを構成する

InstallShield 2018 SP1 では今回より、ソフトウェアのダウンロード可能なアップデートを有効化および構成することができます。ユーザーはより新しいバージョンをダウンロードするか、スキップして現在のインストール処理を続行することができます。

- ダウンロード可能なアップデートのサポートを有効化する
- ダウンロード可能なアップデートのより新しいバージョンを構成する
- 確認メッセージを構成する
- ダウンロード可能なアップデートのサポートを追加する

ダウンロード可能なアップデートのサポートを有効化する

今回より、エンド ユーザーがアドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのダウンロード可能なアップデートのサポートを確認できるようにアップデートを有効化することができます。

ダウンロード可能なアップデートのより新しいバージョンを構成する

- エンド ユーザーがより新しいバージョンをダウンロードするか、スキップして現在のインストール処理を続行するかどちらかのオプションを受け取ります。
- エンド ユーザーはスイート パッケージ (使用可能な場合) の新しいバージョンを強制的にダウンロードおよびインストールすることができます。

確認メッセージを構成する

エンド ユーザーがより新しいバージョンをダウンロードするか、スキップして現在のインストール処理を続行するためのメッセージを入力できるようになりました。

新しいメッセージを作成するか、文字列一覧からローカライズ済みのメッセージを選択できます。

***メモ・** より新しいバージョンをダウンロードするか、スキップして現在のインストール処理を続行するための [はい] または [いいえ] オプションを表示する確認メッセージがエンド ユーザーに表示されます。*

ダウンロード可能なアップデートのサポートを追加する

以前のリリースでは、インストーラーの **[Setup.exe]** タブでターゲット システムへのダウンロードを可能にするアップデート セットアップ ランチャーのパスとして今後使用する、アドバンスト UI およびスイート/アドバンスト UI インストールの**絶対パス**を入力しました。

InstallShield 2018 SP1 では、アドバンスト UI またはスイート/アドバンスト UI インストールと同じ**絶対パス**を、インストーラーの **[アップデート]** タブに入力することができます。

SHA1 & SHA256 を使用するデュアル署名のサポート

以前のリリースでは、署名ダイジェスト ハッシュ アルゴリズムの選択肢は次に限られていました:

- 証明書ハッシュ
- SHA-1
- SHA-256

InstallShield 2018 SP1 では、これらの署名ダイジェスト ハッシュ アルゴリズム以外にもデュアル署名 – (SHA-1 および SHA-256) ダイジェストを選択できます。

***メモ・** 両方のダイジェスト (SHA1 およびSHA256) の使用は、.msi のデジタル署名でサポートされていません。*

スイート読み込み画面のメッセージを構成する

***プロジェクト・** この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では、アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのスイート読み込み画面にメッセージ追加することができます。

スイート読み込み画面メッセージの長さは 35 文字までです。

アップデート ランチャー名のカスタマイズ



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

InstallShield では、アップデート ランチャーの名前をカスタマイズするための新しい設定が追加されています。InstallShield はアップデート ランチャーの名前として、デフォルトで Update.exe を使用します。今回より、名前を指定してアップデート ランチャーを作成することができます。

PowerShell カスタム アクションで文字スペースを指定する



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallShield では、PowerShell カスタム アクションで前後に移動させる場合の文字スペース数を指定するための新しい設定が追加されています。

定義済みパブリック フォルダーの指定



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield では、ユーザーのパブリック フォルダーへの完全パスを保持する新しい定義済みフォルダーが追加されました。

InstallShield 2018 の新しい機能

InstallShield 2018 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- スイート プロジェクトでパッケージのアンインストール順を指定する
- 最小 UI を使ったスイート インストールを実行するメソッド
- 条件付きで実行時に機能の表示/非表示を設定する
- 再帰的または非再帰的な IIS 登録を行う
- Web アプリケーションのフォーム認証を設定する
- アプリケーション プール エンティティのユーザー プロファイルを読み込むかどうかを制御する新しいオプション
- トランスフォーム プロジェクトにプロセスの強制終了および PowerShell カスタム アクションを追加する
- 強化機能

スイート プロジェクトでパッケージのアンインストール順を指定する



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では今回より、**[リリース]** ビューの **[Setup.exe]** タブにある新しい **"アンインストール順"** プロパティを使って、スイート プロジェクトに含まれるパッケージのアンインストール順を指定することができます。

この設定を使って、次のオプションから一つを選択し、スイート プロジェクトに含まれるパッケージのアンインストール順を指定できます。

- **パッケージ順と同じ** – (プロジェクトで定義されている) パッケージのインストール順と同じ順番でパッケージをアンインストールします。
- **パッケージ順と逆** – (プロジェクトで定義されている) パッケージのインストール順と逆の順番でパッケージをアンインストールします。

オートメーション インターフェイスにおける新しい UninstallOrder メソッド

オートメーション インターフェイスで UninstallOrder メソッドを使って **"アンインストール順"** プロパティを設定することができます。次の値のうち 1 つを指定します。

- euoForward(0) – (プロジェクトで定義されている) パッケージのインストール順と同じ順番でパッケージをアンインストールします。
- euoReverse(1) – (プロジェクトで定義されている) パッケージのインストール順と逆の順番でパッケージをアンインストールします。

最小 UI を使ったスイート インストールを実行するメソッド



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では今回より、進行状況パネルを表示するのみの最小 UI モードでスイートインストールを実行する新しいコマンドライン パラメーターを使用できます。

スイート インストールを最小 UI モードで実行するには、コマンドラインで /passive パラメーターを使用します:

Setup.exe /passive

最小 UI モードを使ってアンインストールするには、次のコマンドを使用します :

Setup.exe /passive /remove

条件付きで実行時に機能の表示/非表示を設定する



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

以前のリリースでは、アドバンスト UI またはスイート/アドバンスト UI インストールで機能の **"表示"** プロパティを **[はい]**または **[いいえ]** に設定して、インストーラーの **InstallationFeatures** ウィザード ページで表示するかどうかを指定することができました。

InstallShield 2018 では、インストール デザイナーの **[機能]** ビューで **"表示"** プロパティの下にある **[条件]** オプションを使って、実行時のプロパティに基づいて条件付きで機能を表示または非表示することができます。

"条件" 設定を使って、**InstallationFeatures** ウィザード ページで機能をインストール可能としてデフォルトで表示するかどうかを評価するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する 1 つ以上の条件を指定できます。

たとえば、Windows の特定バージョンが搭載されたターゲット システム上では特定の機能をデフォルトで表示する場合、その Windows バージョンを指定する条件を作成します。



タスク ***アドバンスト UI またはスイート/アドバンスト UI インストールで条件付きで機能を表示する場合、以下の手順に従います:***

1. **[機能]** ビューで **"表示"** プロパティの下にある **[条件]** 行をクリックします。緑色の + 記号 (**[新しい条件]** ボタン) が行の終わりに表示されます。
2. **[新しい条件]** ボタンをクリックします。**[条件]** 行の下に新しい行が追加されます。
3. **[新しい条件]** ボタンの横にある下矢印をクリックして、一覧から適切なオプション (**[All]**、**[Any]**、または **[None]**) を選択します。
4. 次に同じ行で、**[新しい条件]** ボタンをクリックし、適切なオプションを選択してから条件ステートメントのビルドを続行します。

1 つ以上の条件ステートメントが構成されると、**"条件"** プロパティには **(条件)** と表示されます。何も構成されていない場合、**"条件"** プロパティには **(空白)** と表示されます。

条件付きの表示をサポートするための、オートメーション インターフェイスに追加された新しいメソッド

実行時に条件付きで機能の表示を設定するため、次の新しいメソッドがオートメーション インターフェイスに追加されました:

メソッド	構文
AddVisibleCondition	AddVisibleCondition() As ISWiSuiteCondition
DeleteVisibleCondition	DeleteVisibleCondition()
VisibleCondition	読み取り専用オブジェクト プロパティ

再帰的または非再帰的な IIS 登録を行う



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

[IIS 構成] ビューの **"アプリケーション"** 設定に **[ASP.NET 登録]** という名前の新しいオプションが追加され、再帰的または非再帰的な ASP .NET 登録を行うことができます。この機能を使って、同じ Web サイトに ASP.NET アプリケーションおよび ASP.NET コア アプリケーションの両方をインストールすることができます。

ASP.NET アプリケーション登録オプションを IIS (インターネット インフォメーション サービス) を使って設定するには、**"ASP.NET 登録"** プロパティを次のオプションの 1 つに設定します:

- **再帰的** — 指定されたアプリケーションおよびすべてのサブアプリケーションのスクリプト マップとアプリケーション プールの割り当てを更新します。
- **非再帰的** — 特定のアプリケーションのスクリプト マップとアプリケーション プールの割り当てを更新します。サブアプリケーションは変更されません。

Web アプリケーションのフォーム認証を設定する



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

InstallShield 2018 に、Web アプリケーションでフォーム認証を設定するための新しいオプションが追加されました。この新しいオプション **[フォーム認証]** は、Web サイトの **[IIS 構成]** ビューで **[認証済みアクセス]** セクションの下に表示されます。

[フォーム認証] オプションを **[はい]** に設定してフォーム認証を有効化します。ASP.NET フォーム ベースの認証は、多くの要求を受け取るパブリック Web サーバー上にあるサイトまたはアプリケーションに適しています。この認証モードでは、オペレーティング システムが提供している認証方法に頼ることなく、アプリケーション レベルでのクライアント登録および認証を管理することができます。



重要 - *フォーム認証は、Web サーバーにユーザー名とパスワードをテキスト形式で送信します。ホームページ以外のアプリケーション内のすべてのページ、およびログオン ページには、Secure Sockets Layer (SSL) 暗号化を使用してください。*

アプリケーション プール エンティティのユーザー プロファイルを読み込むかどうかを制御する新しいオプション



プロジェクト - *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

InstallShield 2018 では、アプリケーション プール エンティティのユーザー プロファイルをロードするかどうかを制御する新しい **"アプリケーション プール"** 設定が **[IIS 構成]** ビューに追加されました。

"ユーザー プロファイルの読み込み" プロパティに次のオプションの 1 つを設定します:

- **はい** – IIS がアプリケーション プールのユーザー プロファイルをロードします。
- **いいえ** – IIS がアプリケーション プールのユーザー プロファイルをロードしません。これは IIS 6.0 のときと同じ動作です。

トランスフォーム プロジェクトにプロセスの強制終了および PowerShell カスタム アクションを追加する



プロジェクト - *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*
- *トランスフォーム*

以前のリリースでは、トランスフォーム プロジェクトにプロセスの強制終了または PowerShell カスタム アクションを追加することができませんでした。InstallShield 2018 では今回より、**[カスタム アクションとシーケンス]** ビューでトランスフォーム プロジェクトに **[新しいプロセスの強制終了]** または **[新しい PowerShell]** カスタム アクションを追加することができます。

強化機能

InstallShield 2018 には、以下の強化機能が含まれています:

- QuickPatch プロジェクトを XML 形式で保存する
- スイート プロジェクトにおける製品名プロパティのローカライズ
- 製品構成のセットアップ ファイル名にプロパティの値を含める
- 概要情報ストリーム コメントおよびパッケージ ファイル名を設定するための新しい MSBuild パラメーター
- テキスト ファイルの変更で改行およびタブ文字の指定
- スイート読み込み画面を削除または非表示にする
- スイート インストールで常にデバッグ ログを作成する設定
- 実行時に SSL 証明書の IIS 証明書ファイルを設定する新しい簡易ダイアログ
- XML ファイルで新しい子要素を作成するときに絶対パスまたは相対パスを指定する
- スイート プロジェクトで、ダイアログ ボックス コントロールのデフォルト キーボード フォーカスを設定する
- 基本の MSI プロジェクトの PowerShell スクリプト エディター
- InstallShield トランスフォーム ウィザードで既存のトランスフォーム ファイルを開く新しいオプション
- 追加された前提条件

QuickPatch プロジェクトを XML 形式で保存する



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *QuickPatch*

InstallShield 2018 では、QuickPatch プロジェクトを XML 形式で保存することができます。また XML 形式で保存されているプロジェクトから QuickPatch プロジェクトを作成することもできます。以前のリリースでは、QuickPatch プロジェクト はバイナリ形式でのみ保存することが可能でした。

スイート プロジェクトにおける製品名プロパティのローカライズ



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では、今回より、スイート プロジェクトで **"製品名"** プロパティのローカライズをサポートします。

スイート プロジェクトで "製品名" プロパティをローカライズする場合、次の手順に従います。

タスク *スイート プロジェクトでプロパティをローカライズするには:*

1. スイート プロジェクトを開いて**[ユーザー インターフェイス > 文字列エディター]** ビューに移動します。
2. スイート プロジェクトでサポートされている各言語の翻訳済みテキストを含む新しい文字列を作成します。例、ID_STRING2
3. **[インストール情報 > 一般情報]** ビューを開きます。
4. **"製品名"** フィールドの横にある参照ボタンをクリックして、**[文字列の選択]** ダイアログ ボックスを開きます。
5. 翻訳済みテキストを含む作成した文字列の名前を選択します。

製品構成のセットアップ ファイル名にプロパティの値を含める

プロジェクト・*この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

InstallShield 2018 では今回より、製品リリース構成のセットアップ名およびパッケージ ファイル名に Property テーブルからのプロパティの値を含めることができます。

たとえば、**[リリース > 製品構成]** ビューの **[全般]** タブにある "セットアップ ファイル名" または "MSI パッケージ ファイル名" フィールドに次の任意のプロパティを入力することができます:

setup[ProductVersion]
setup[CustomVersion]
setup[ProductCode]
setup[ProductCode][ProductVersion]

たとえば、**"セットアップ ファイル名"** フィールドに **setup[ProductVersion]** と入力した場合、セットアップ名が setup14.10.1234.exe となります。

概要情報ストリーム コメントおよびパッケージ ファイル名を設定するための新しい MSBuild パラメーター

InstallShield 2018 で今回追加された MSBuild パラメーターを使って、インストーラーにコメントの追加を設定、およびインストーラーのパッケージ ファイル名の設定を行うことができます。

- 概要情報ストリーム コメントを設定する新しいパラメーター
- パッケージ ファイル名を設定する新しいパラメーター

概要情報ストリーム コメントを設定する新しいパラメーター

プロジェクト・*この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

- *マージ モジュール*

[一般情報] ビューの **"概要情報ストリーム コメント"** フィールドでインストーラーにコメントを追加することができます。

InstallShield 2018 ではまた、ビルド時にコメントを入力できるオプションも追加されています。MSBuild.exe タスクに SummaryInfoComments という名前の新しいパラメーターが追加されました。これを使って、次の例のようにビルド番号を含めるなど、ビルド時に **[概要情報ストリーム コメント]** を設定することができます。

```
MSBuild.exe c:¥installers¥Setup.sln /Property:SummaryInfoComments="ここにコメントを入力"
```

SummaryInfoComments プロパティを使って追加されたコメントは、ビルド済みのインストーラーの **[プロパティ]** ダイアログ ボックスで参照することができます。

パッケージ ファイル名を設定する新しいパラメーター



***プロジェクト** - この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

[リリース] ビューで **"製品構成"** フィールドの **[全般]** タブにある **"MSI パッケージ ファイル名"** フィールドで、インストーラーのパッケージ ファイル名を指定できます。

InstallShield 2018 ではまた、パッケージ ファイル名を設定できるオプションも追加されています。MSBuild.exe タスクに MSIPackageFileName という名前の新しいパラメーターが追加されました。これを使って、次の例のようにビルド時にインストーラーのパッケージ ファイル名を設定することができます。

```
MSBuild.exe c:¥installers¥Setup.isproj /Property:MSIPackageFileName="MySetup"
```

MSIPackageFileName パラメーターの値を入力するとき、InstallShield が .msi ファイルに使用する (ピリオドまたはファイル拡張子を含まない) ファイル名を入力する必要があります。

テキスト ファイルの変更で改行およびタブ文字の指定



***プロジェクト** - この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

ターゲット システム上で実行時に変更するテキスト ファイルの内容について、インストーラーの検索/置換処理を構成することができます。そのためには、**[システム構成 > テキスト ファイルの変更]** ビューを開いて実行時に検索するテキスト ファイルを識別するためのテキスト **[変更セット]** を追加し、さらに検索するテキスト (**検索後の文字列**) および置換するテキスト (**置換後の文字列**) を指定します。

InstallShield 2018 では、テキストの変更セットを追加したとき、改行またはタブを指定するために **"置換後の文字列"** フィールドにエスケープ シーケンスを入力できるようになりました。

文字	エスケープシーケンス
改行	¥r¥n
タブ	¥t



***メモ・** Windows オペレーティング システムの場合、改行には **¥r¥n** の両方を入力する必要があります。*

実行時に検索/置換が行われた場合、**"置換後の文字列"** フィールドの **¥r¥n** が入力された場合に改行が、**¥t** が入力された場合にタブが挿入されます。

これらの文字がエスケープ シーケンスとして認識されるように、**[エスケープ シーケンスの解析]** オプションを **[はい]** に設定する必要があります。

スイート読み込み画面を削除または非表示にする



***プロジェクト・** この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では今回より、インストール中にスイート読み込み画面を表示するかどうかを制御する機能が追加されました。

この画面が表示されるかどうかを制御するため、**[リリース]** ビューの **[Setup.exe]** タブに **[スイート読み込み画面を表示]** という名前の新しいプロパティが追加されました。アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのスイート 読み込み画面を非表示にする場合は、このプロパティを **[いいえ]** に設定します。

スイート読み込み画面のメッセージを構成する

InstallShield 2018 では、アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのスイート読み込み画面にメッセージ追加することができます。

スイート読み込み画面メッセージの長さは 35 文字までです。

オートメーション インターフェイスにおける新しい ShowSuiteLoadingScreen メソッド

オートメーション インターフェイスで ShowSuiteLoadingScreen メソッドを使って、**[リリース]** ビューの **[Setup.exe]** タブにある **"スイート読み込み画面を表示"** 設定を指定することができます。デフォルト値は True です。

スイート インストールで常にデバッグ ログを作成する設定



***プロジェクト・** この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では今回より、コマンドラインで debuglog を渡さずにスイート プロジェクトのログ記録を有効にするオプションを選択できます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトで **[リリース]** ビューの **[Setup.exe]** タブに新しいオプション **[常にデバッグ ログを作成する]** が追加されました。

アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのデバッグ ログを常に作成する場合は **[常にデバッグ ログを作成する]** オプションに **[はい]** を選択します。

オートメーション インターフェイスにおける新しい CreateDebugLog メソッド

オートメーション インターフェイスで CreateDebugLog メソッドを使って、**[リリース]** ビューの **[Setup.exe]** タブにある **"常にデバッグ ログを作成する"** 設定を指定することができます。デフォルト値は False です。

実行時に SSL 証明書の IIS 証明書ファイルを設定する新しい簡易ダイアログ



プロジェクト・この情報は、基本の MSI プロジェクトに適用します。

InstallShield 2018 には、エンド ユーザーが SSL 証明書に提供する IIS 証明書ファイルを参照し、インストール実行時にパスワードを入力できる、インストーラー用の新しい簡易ダイアログ（IISBrowseSSLCertificate）が含まれています。

インストーラーに **[IIS の SSL を構成する]** ダイアログ を追加するには、次の手順に従います：



タスク ***インストーラーに [IIS の SSL を構成] ダイアログ を追加するには：***

1. **[サーバー構成]** の下の **ビュー リスト** にある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーを右クリックして、**[Web サイトの追加]** をクリックします。InstallShield が新しい Web サイトを追加します。
3. 新しい Web サイトを選択して **[セキュリティ > セキュリティで保護された通信]** の下にある **"SSL 証明書"** および **"SSL 証明書パスワード"** プロパティに移動します。
4. **"SSL 証明書"** および **"SSL 証明書パスワード"** プロパティを次の値に設定します:

Property	Value
SSL 証明書	[IS_IIS_WEBCERTPATH]
SSL 証明書のパスワード	[IS_IIS_WEBCERTPASSWORD]

5. **[ユーザー インターフェイス > ダイアログ]** ビューを開いて、ダイアログ シーケンスに **IISBrowseSSLCertificate** ダイアログを追加します。

   [編集] ボックス（**IISWebCertPassword** と **IISWebCertPath**）およびプッシュボタン（**BrowseCertificate**）イベントの **IISBrowseSSLCertificate** ダイアログで、ユーザーが構成した SSL 証明書およびパスワードのプロパティ名がプロパティ名を更新する必要があります。

XML ファイルで新しい子要素を作成するときに絶対パスまたは相対パスを指定する



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

以前のリリースでは、**[システム構成 > XML ファイルの変更]** ビューを使って既存する親要素内の子要素と同じ名前を持つ XML ファイルに新しい子要素を追加すると、XML ファイルの変更が失敗しました。

XML ドキュメント内のノードのパスには、絶対パスまたは相対パスの両方が使用できます。絶対パスはルートから始まります。既存の親要素内にある子要素と同じ名前を持つ XML ファイルに新しい子要素を追加するときは、絶対パスを使用する必要があります。

InstallShield 2018 では、子要素を作成するときに絶対パスを使用することを指定するための "絶対 XPath を使用する" という名前の新しい設定が **[XML ファイルの変更]** ビューに追加されました。

"絶対 XPath を使用する" 設定によって異なる子要素作成時の動作:

- **選択** – このオプションが選択されている場合、子要素が追加されるときに絶対 XPath が使用されます。
- **非選択** – このオプションが選択されていない場合、子要素が追加されるときに汎用 XPath が使用されます。デフォルトでは、**"絶対 Xpath を使用する"** オプションは選択されていません。

スイート プロジェクトで、ダイアログ ボックス コントロールのデフォルト キーボード フォーカスを設定する



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では今回より、スイート プロジェクトでウィザード ページを定義するとき、ウィザード ページ上のどのコントロールにデフォルトのキーボード フォーカスを配置するのかを指定することができます。

[ウィザード インターフェイス] ビューで、**[外観]** の下に **"デフォルト フォーカス"** という名前の新しいプロパティが追加されていて、そのウィザード ページで定義されているすべてのコントロールが一覧表示されます。コントロールを選択して、デフォルト キーボード フォーカスをそのコントロールに設定します。

基本の MSI プロジェクトの PowerShell スクリプト エディター



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

InstallShield 2018 では、基本の MSI プロジェクトで **[カスタム アクションとシーケンス > カスタム アクション]** ビューの新しい **[スクリプト]** タブで PowerShell スクリプト エディターを使用できます。以前のリリースでは、PowerShell スクリプト エディターはスイート/アドバンスト のみ使用可能でした。

InstallShield トランスフォーム ウィザードで既存のトランスフォーム ファイルを開く新しいオプション



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *トランスフォーム*

InstallShield 2018 では今回より、InstallShield トランスフォーム ウィザードで、既存のトランスフォーム ファイルを開くことができます。トランスフォーム ウィザードで初めて開いたときと同様に、トランスフォーム ファイルのベースとなる MSI パッケージを選択するためのプロンプトが表示されます。これによって、複数の MSI パッケージに共通の汎用トランスフォーム ファイルを使用することができます。

InstallShield トランスフォーム ウィザードで既存のトランスフォーム ファイルを開くには、Windows エクスプローラーでトランスフォーム ファイルを右クリックしてから、コンテキスト メニューから **[InstallShield トランスフォーム ウィザードで開く]** を選択します。

追加された前提条件

InstallShield 2018 には、以下の前提条件が追加されています:

- Visual C++ 2017 x86 および x64 前提条件
- Microsoft SQL Server 2014 SP1 および SP2 前提条件
- Microsoft .NET Framework 4.7 前提条件

Visual C++ 2017 x86 および x64 前提条件

Microsoft Visual Studio 2017 がリリースされたため、InstallShield は今回より、Visual C++ 2017 x86 および x64 の前提条件を含みます。

Microsoft SQL Server 2014 SP1 および SP2 前提条件

Microsoft SQL Server 2014 には 2 つのサービス パックがリリースされているため、InstallShield は今回より、Microsoft SQL Server 2014 SP1 および SP2 両方の前提条件を含みます。

Microsoft .NET Framework 4.7 前提条件

InstallShield は今回より、Microsoft .NET Framework 4.7 の前提条件を含みます。

InstallShield 2016 SP2 の新しい機能

FlexNet Code Aware との統合

InstallShield には、今回より FlexNet Code Aware との統合が含まれています。FlexNet Code Aware は、製品のセキュリティおよび知的財産 (IP) コンプライアンス リスクを迅速にスキャンする、自動オープン ソース リスク評価およびパッケージ検出ソリューションです。

FlexNet Code Aware の現在のリリースでは、次のファイル分析がサポートされています:

- Java パッケージ
- Node パッケージ
- Nuget パッケージ
- RPM パッケージ
- Ruby パッケージ
- EXE & DLL ファイル

セキュリティ脆弱性は、National Vulnerability Database (NVD) と照合されます。

FlexNet Code Aware を実行する

FlexNet Code には InstallShield とは別のライセンスが必要です。トライアル/評価版もあります。詳細については、Revenera Web サイトの [FlexNet Code Aware 製品] ページ を参照してください。

InstallShield 内部から FlexNet Code Aware を実行するには、InstallShield の **[プロジェクト]** メニューから **[FlexNet Code Aware を使ってプロジェクトをスキャン]** をクリックします。このメニュー オプションは、InstallShield プロジェクトが現在開いていない場合は無効です。FlexNet Code Aware アイコンも、InstallShield 標準ツールバーに表示されます。

FlexNet Code Aware がプロジェクトのスキャンを完了したとき概要にはキャン済みファイルの数、オープン ソース パッケージ、ならびに検出された脆弱性の数が表示されます。FlexNet Code Aware のフル ライセンス版では、**[View report (レポートの表示)]** ボタンが使用できます。このレポートで提供される詳細については、「FlexNet Code Aware レポートの読み方」を参照してください。

FlexNet Code Aware レポートの読み方



***メモ・** FlexNet Code Aware レポートは、トライアル/評価版モードでは使用できません。FlexNet Code Aware のフル ライセンス版が必要です。*

FlexNet Code Aware レポートを参照するためには、FlexNet Code Aware がプロジェクトをスキャンした後に表示される概要ダイアログで **[View repoert (レポートの表示)]** をクリックします。

FlexNet Code Aware レポートは、いくつかのセクションから構成されています:

- 最初の [Summary (概要)] ビューには、**Scan Summary (スキャン概要)**、**Operational Risk (運用リスク)評価**、**Security Vulnerability Exposure (セキュリティ脆弱性の公開)**、および **License Exposure (ライセンスの公開)** が提示されます。
  - **[Scan Summary (スキャン概要)]** セクションは、ファイル タイプの内訳、分析済みファイルの割合、および検出数を含む、スキャン済みのコードベースについての詳細が表示されます。
  - **[Operational Risk (運用リスク)]** セクションには、パッケージと知的財産 (IP) との問題、およびパッケージとセキュリティ脆弱性の組み合わせに基づいた合成リスク評価が表示されます。
  - **[Security Vulnerability Exposure (セキュリティ脆弱性の公開)]** および **[License Exposure (ライセンスの公開)]** セクションには、識別された問題の種類とカテゴリーの内訳が表示されます。
- [Package Inventory (パッケージ インベントリ)] ビューは **[Scan Summary (スキャン概要)]** セクションで **[view full package inventory (フル パッケージ インベントリの表示)]** をクリックすると表示され、検出されたオープンソースおよびサードパーティ パッケージおよび関連ライセンス、セキュリティ脆弱性、依存関係、ならびに検出された著作権ステートメントの完全リストを表示します。

[Package Inventory (パッケージ インベントリ)] ビューには、ターゲットのクエリを実行して、一覧を様々なパッケージの種類に絞り込むためのフィルターが用意されています。

パッケージの詳細を確認する

[Package Inventory (パッケージ インベントリ)] レポート ページの [Vulnerabilities (脆弱性)] 列にある脆弱性カウントをクリックして、各パッケージの詳細を確認します: パッケージ インベントリ レポートの一部が含まれた **[Vulnerabilities detail (脆弱性の詳細)]** が表示されます:

InstallShield 2016 SP1 の新しい機能

Microsoft Visual Studio 2017 のサポート

InstallShield は、Visual Studio 2017 をサポートします。このバージョンの Visual Studio 内部から InstallShield プロジェクトを作成できます。

InstallShield 2016 の新しい機能

InstallShield 2016 には、以下のよう新機能が追加されています:

- Windows オペレーティング システムの最新版リリースをサポート
- Microsoft SQL Server 2016 サポート
- Windows アプリ パッケージを作成するためのサポート
- スイートに追加された SQL サポート
- タイルの構成
- 新しい Microsoft Visual C++ 2015、.NET Framework 4.6 その他用の InstallShield 前提条件
- Adobe Reader、Microsoft Office、および .NET Framework の前提条件システム検索

Windows オペレーティング システムの最新版リリースをサポート

InstallShield 2016 は、Windows オペレーティング システムの最新版リリースをサポートします。

- Windows 10 Anniversary Update
- Windows Server 2016

InstallShield をこれらのオペレーティング システムにインストールできるだけでなく、これらのオペレーティング システムをターゲットにするインストールを作成することができます。

Microsoft SQL Server 2016 サポート



***プロジェクト** - Microsoft SQL Server 2016 サポートは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*

- *InstallScript MSI*

InstallShield に、SQL Server 2016 Database 上で SQL スクリプトを実行できるサポートが追加されました。また、InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストには、SQL Server 2016 が含まれています。

インストールで SQL Server 2016 をターゲットするとき、エンドユーザーがデータベース サーバーを参照する選択をしたときに表示される SQLBrowse ランタイム ダイアログで、SQL Server 2016、SQL Server 2016 Express、および SQL Server 2014 Express LocalDB のインスタンスが表示できるようになりました。また、エンドユーザーがデータベース カタログを参照する選択をしたときに表示される SQLBrowse ランタイム ダイアログで、指定された SQL Server 2016 サーバー上のカタログが表示されるようになりました。

InstallShield に追加された、新しい InstallShield 前提条件の完全リストは「新しい Microsoft Visual C++ 2015、.NET Framework 4.6 その他用の InstallShield 前提条件」を参照してください。

***メモ・** Microsoft SQL Server 2016 は 64-bit (x64) のみです。*

Windows アプリ パッケージを作成するためのサポート

***プロジェクト・** Windows アプリの作成機能は、基本の MSI プロジェクトで使用できます。*

***重要・** デスクトップ拡張 (デスクトップ ブリッジ) を含む Windows アプリ パッケージ (.appx) のインストールおよびテストを行うには、Windows 10 Anniversary Update が必要です。Windows アプリ パッケージにデジタル署名を行う場合、InstallShield を Windows 10 または Windows 10 SDK がインストールされているマシン上にインストールする必要があります。*

Windows 8.x および 10 上にアプリを配布およびインストールする為に使用される Windows アプリ パッケージ (.appx) は、シンプルでセキュリティ保護されたパッケージ フォーマットで、UWP (ユニバーサル Windows プラットフォーム) で使用可能な唯一のフォーマットです。Windows アプリ パッケージの利点:

- 高い可用性、信頼性、および耐久性によって、アプリケーションが長期間にわたってエラーなしで継続的に動作し続けます。
- 必要最小限の構成とカスタマイズ不要な UI によるスタティック ビルドを使ったスムーズなインストール経験
- Windows ストアを使ってアプリケーションを販売または提供できるオプション
- UWP API を使用できる機能だけでなく、ライブ タイルなどの UWP 機能を活用
- Windows Nano Server 上で サポートを持つ唯一のパッケージ フォーマット

InstallShield は今回より、代替ビルド出力を通して Windows アプリ パッケージ フォーマット (.appx) およびそのデスクトップ / サーバー拡張機能の作成をサポートし、Windows アプリ パッケージ フォーマットに適合しないアイテムを識別するための適合性テストを提供します。Windows アプリ パッケージの作成をサポートする InstallShield に新しく追加された機能の詳細については、次のサブセクションをご覧ください。

- [リリース] ビュー内の Windows アプリの設定
- [ショートカット] ビューにおける Windows アプリ ロゴのカスタマイズ
- MSIX/UWP アプリ適合性のテスト

- スイートに追加された新しい UWP 条件のチェック

[リリース] ビュー内の Windows アプリの設定

[リリース] ビューでリリースを選択するとき、**[Appx/MSIX]** という名前の新しいリリースごとに提供されるタブに、Windows アプリ パッケージを作成する設定が追加されました。ここで、Windows アプリ パッケージのビルドプロセスに影響する様々な主要な設定を指定できます。特に **[配布方法]**、**[デスクトップ拡張を含む]**、または **[サーバー拡張を含む]** オプションは、特定の種類のインストーラー プロジェクト データにどの警告またはエラーが発生するかに影響します。

これらの新しい設定についての総合情報は、「リリースの [Appx/MSIX] タブ」を参照してください。

[ショートカット] ビューにおける Windows アプリ ロゴのカスタマイズ

[リリース] ビューに追加された新しい Windows アプリ設定以外にも、Windows アプリ パッケージで作成されたタイルを構成するための新しい設定があります。これらの設定は、[ショートカット] ビューの **[Windows アプリ パッケージ タイルのオーバーライド]** 領域にあります。

これらの新しい設定についての総合情報は、「ショートカットの設定」を参照してください。

MSIX/UWP アプリ適合性のテスト

InstallShield には、.msi パッケージ内で Windows アプリ パッケージ フォーマットに適さないアイテムの存在をスキャンする、新しい **InstallShield MSIX/UWP アプリ適合性スイート**が追加されました。アクセスするには、**[ビルド]** メニューから **[検証]** をポイントしてから、**[InstallShield MSIX/UWP アプリ適合性スイート]** をクリックします。

InstallShield MSIX/UWP アプリ適合性スイートは、問題が見つかったすべてのテストを [リリース] ビューに表示し、各問題に関連付けられた列に既知の Windows アプリ形式への適用性を示します。従来型の CUB の場合、これらの列は空白のままです。このレポートは、[リリース] ビューで、リリースの下にある [検証] フォルダーを選択すると表示されます。新しい ISUWP 検証の説明その他の総合情報は、「InstallShield MSIX/UWP アプリ適合性スイート」を参照してください。

スイートに追加された新しい UWP 条件のチェック

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、機能、またはウィザード インターフェイス条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、ターゲット システム上で評価するチェックを様々な異なる種類から選択できます。スイートに次の条件チェックが追加されました。

- **Windows アプリ パッケージの対象**—ターゲット システムで Windows アプリ パッケージの実行時に依存ファイルの存在を確認して、これをサポートしない Windows または Windows Server バージョンに Windows アプリ パッケージがインストールされることのないように防ぎます。

  ***メモ・** この条件は、Windows アプリ パッケージの対象条件のみで使用可能です。別のパッケージ タイプで使用した場合、正しく機能しません。*

- **UWP タイプの存在**—ターゲット システムをチェックして、UWP 機能の存在を確認します。たとえば、デスクトップ ブリッジの存在を確認する条件ステートメントを作成するには、Windows.ApplicationModel.FullTrustProcessLauncher タイプを確認します。これは、条件付でインストールをブロックする、または .msi および Windows アプリ パッケージ (.appx) のどちらをインストールするかを選択する場合に使用できます。



メモ・これは、Windows 10 以前のオペレーティング システムで常に False 評価されます。"Windows.ApplicationModel.FullTrustProcessLauncher " タイプ名サブ設定を使用するには、Windows 10 Anniversary Update 以降が必要です。

スイートに追加された SQL サポート



***プロジェクト・**この情報は、スイート/アドバンスト UI プロジェクトに適用します。*



***エディション・**スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。*

SQL サーバーは、特に InstallShield スイート インストールで提供される複数パッケージ サポートを活用する多くのアプリケーションで不可欠です。以前、InstallShield SQL サポートが使用できるのは基本の MSI、InstallScript、および InstallScript MSI プロジェクトのみでした。今回より、SQL サポートがスイート/アドバンスト UI プロジェクトに追加されており、以下の機能を使用することができます:

- 新しい SQLLogin 定義済みウィザード ページを追加
- SQL ステートメントを直接スイートから実行

詳しくは、次のトピックを参照してください:

- スイート/アドバンスト UI プロジェクトで、SQLLogin 定義済みウィザード ページを追加する
- ウィザード インターフェイス内の要素のアクションを構成する
- [定義済みタスク ページ] パネル

新しい SQLLogin 定義済みウィザード ページを追加

InstallShield では、新しい**SQLLogin** 定義済みウィザード ページを使ってスイート/アドバンスト UI プロジェクトに SQL サポートを追加します。以前、スイート/アドバンスト UI プロジェクトに SQL サポートを含む .msi パッケージを追加すると、アドバンスト UI またはスイート/アドバンスト UI セットアップ起動プログラムが Windows Installer のユーザー インターフェイスを自動的に抑制しました。スイート/アドバンスト UI プロジェクト インストールにカスタム SQLLogin ウィザード ページを手動で作成する必要がありました。

新しい定義済みのページをプロジェクトに追加するとき、**[データベース サーバーのログイン情報を入力]** タスク ページを選択して、必要に応じてウィザードのパネルを完了させます。その後、SQLLogin 定義済みウィザード ページを、プロジェクトに追加します。この SQLLogin ウィザード ページを使って、エンド ユーザーはデータベース サーバー ログイン情報 (データベース サーバー名、認証資格情報、データベース カタログ名など) を入力し、スイートに含まれる 1 つ以上の .msi パッケージがターゲットとするデータベース サーバーへの接続を設立することができます。

SQLLogin ウィザード ページをプロジェクトに追加すると、以下が可能となります:

- スイート プロジェクト内の SQL ログインの設定を識別するプロパティを指定してから、これらのプロパティを受け取る .msi パッケージを選択する
- .msi パッケージ内で SQL ログインの設定を識別するプロパティを指定する
- データベース テクノロジ (Microsoft SQL Server、Microsoft Windows Azure、MySQL、または Oracle) を選択して、ターゲットにする ODBC ドライバーを選択する

SQL ステートメントを直接スイートから実行

スイート/アドバンスト UI プロジェクトでは今回より、ユーザー インターフェイスから SQL データベース サーバー上の SQL ステートメントを直接実行することができます。これによって、インストール続行前に SQL データベース サーバーを調査することができます。

これによって、スイート プロパティで SQL クエリの結果にアクセスが可能となります。このサポートを使用するための **[SQL 文字列を実行]** オプションが、UI イベントの [新しいアクション] メニューに追加されています。SQL ステートメントは、UI イベントの [新しいアクション] メニューで使用できる新しい追加オプション ([**データベース メタデータの構成**] および [**SQL ログイン プロパティのオーバーライド**]) を使って指定されたプロパティおよびデータベース メタデータ を使って実行されます。

タイルの構成



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- 基本の MSI
- InstallScript MSI
- InstallScript

Windows 8 からアプリケーション タイルのグリッドをスタート画面に表示できるようになりました。これは、今までのショートカットの一覧に取って代わるもので、ショートカットの代わりにタイルを配置します。
InstallShield は、スタート画面上のデスクトップ アプリのタイルの外観をカスタマイズすることができます。次のタイル構成設定が使用できます:

- アプリケーション名を中サイズ (150x150) のタイルに含めるとき、明色または暗色のテキストを切り替える
- タイル背景色を選択
- カスタム タイル イメージ (小: 70x70、中:150x150) の使用オプション
- アプリケーション名を中サイズ タイルに表示または非表示を選択

[タイルの構成] ノードは、メインの **[ショートカット]** ビューおよび各コンポーネントの **[ショートカット]** サブビューに表示されます。適用するすべてのタイル構成が一覧表示されます。

詳しくは、次のトピックを参照してください:

- スタート画面上のデスクトップ アプリのタイルの外観を構成する
- [タイル構成] の設定

新しい Microsoft Visual C++ 2015、.NET Framework 4.6 その他用の InstallShield 前提条件



***プロジェクト・**InstallShield 前提条件は、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加できます。*

InstallShield には、以下の InstallShield 前提条件が含まれています。

- Microsoft .NET Framework 4.6.2 (完全版)
- Microsoft .NET Framework 4.6.1 (完全版)

- Microsoft .NET Framework 4.6.1 (Web インストーラー)
- Microsoft ReportViewer 2015
- Microsoft SQL Server 2014 Express System CLR Types (x86)
- Microsoft SQL Server 2016 Express RTM (x64)
- Microsoft SQL Server 2016 Express RTM LocalDB (x64)
- Microsoft Visual C++ 2015 Update 3 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2015 Update 3 再配布可能パッケージ (x64)
- Windows Management Framework 4.0 for Windows 7 SP1 および Server 2008 R2 SP1 (x64)
- Windows Management Framework 4.0 for Windows Server 2012 (x64)
- Windows Management Framework 5.0 for Windows 7 SP1 (x86)
- Windows Management Framework 5.0 for Windows 7 SP1 および Server 2008 R2 SP1 (x64)
- Windows Management Framework 5.0 for Windows 8.1 (x86)
- Windows Management Framework 5.0 for Windows 8.1 および Server 2012 R2 (x64)
- Windows Management Framework 5.0 for Windows Server 2012 (x64)

これらの前提条件は、サポートされているターゲット システムに適切なテクノロジをインストールします。



***メモ・** .NET Framework の* **Web** *前提条件には、インターネット接続が必要です。この前提条件は、必要に応じて、必須の再配布可能ファイルをダウンロードします。.NET Framework の* **完全** *な前提条件は、インターネットへの接続が不要なスタンドアロン インストールです。*

Adobe Reader、Microsoft Office、および .NET Framework の前提条件システム検索



***プロジェクト・** 前提条件システム検索は、基本の MSI および InstallScript MSI プロジェクトに適用します。*

InstallShield に新しい定義済みシステム検索が追加されました:

- Adobe Reader 11
- Adobe Reader DC
- Microsoft Office 2013
- Microsoft Office 2016
- Microsoft .NET Framework 4.5.1
- Microsoft .NET Framework 4.5.2
- Microsoft .NET Framework 4.6
- Microsoft .NET Framework 4.6.1
- Microsoft .NET Framework 4.6.2

インストールでこれらの 1 つまたは両方が必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、これらのシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

強化機能

InstallShield 2025には、以下のような強化機能が搭載されています。

- [ダイレクト エディター] ビューの強化機能
- スイート – UI 機能強化
- [プロセスの強制終了] カスタム アクションの強化内容
- コンポーネントの属性に使用するデフォルト値を設定できる機能
- 追加のビューで、機能ごとに項目をフィルターできる機能
- デジタル署名の更新

[ダイレクト エディター] ビューの強化機能



***プロジェクト・**ダイレクト エディターは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

InstallShield には今回より、テーブル、スキーマ情報、および検証エラーの詳細を確認できるいくつかのダイレクト エディター強化内容が追加されています。これにより、ダイレクト エディターを使って高度な問題を識別および解決するためにトラブルシューティングを行うセットアップ作成者またはパッケージ作成者の生産性が飛躍的に高まります。これらの強化内容について、次のセクションに分けて説明されています:

- Directory テーブルに、解決されたターゲット ディレクトリ パスが表示される
- 列ヘッダー スキーマ情報のツールヒント
- テーブル レコード参照の追跡
- 破損した参照のインジケータ

総合的な情報は、「ダイレクト エディター」を参照してください。

Directory テーブルに、解決されたターゲット ディレクトリ パスが表示される

Directory テーブルを表示するとき、InstallShield は各行についてディレクトリの場所の解決済みパスを表示する読み取り専用の灰色列を表示します。この列は実際、プロジェクト ファイルには保存されません。表示されるテキストごとに並べ替えることができますが、その値を挿入、更新、または削除することはできません。

列ヘッダー スキーマ情報のツールヒント

InstallShield は今回より、使用可能な列データの種類を示すスキーマ情報を表示する列ヘッダーに、ツールヒントを表示します。

- **ヌル可能**—列は空白のままに残すことが可能です。
- **必須**—列には、必ず空白以外の値が必要です。
- **文字(*nn*)**—固定文字数 *nn* の文字列。
- **小さい整数**—整数値 (小数点なし)、–32767 から +32767 までの値を含む。
- **長い整数**—整数値 (小数点なし)、–2147483647 から ++2147483647 までの値を含む。
- **ローカライズ可能**—翻訳可能な文字列を含む列。このマーカーが付いていない列は、ローカライズ不可能です。
- **ストリーム**—ファイルのコンテンツなどの、バイナリ ストリーム。



*ヒント・**Directory**、**Binary**、および **CustomAction** ダイレクト エディター テーブルには、これらの種類の列のいくつかが表示されます。*

テーブル レコード参照の追跡

ダイレクト エディターには今回より、テーブル レコードの関係性を簡単に確認することができる [参照の追跡] ペインが追加されています。ダイレクト エディター上部に追加された **[参照の追跡を表示]** ボタンを使って、ペインの表示/非表示を切り替えることができます。

各レコードは、1 つ以上のレコードを参照、または 1 つ以上のレコードによって参照されている可能性があります。レコードが強調表示されている場合、別のレコードを参照しているか、別のレコードから参照されていることを示し、[参照の追跡] ペインには、参照が存在するテーブルを表示する **[参照テーブル]** セクション、および実際のレコードの参照を表示する追加セクションが含まれます。レコードの参照セクションには、参照の方向を示す矢印アイコンが表示されます。

- 右向きの緑色矢印は、ダイレクト エディター テーブルで選択されたレコードが [参照の追跡] ペインに表示されているレコードを**参照する**ことを示します。
- 左向きの青色矢印は、ダイレクト エディター テーブルで選択されたレコードが [参照の追跡] ペインに表示されているレコードによって**参照されている**ことを示します。
- 両方向を指す 2 つの矢印 (右向きの緑色矢印と左向きの青色矢印) は、ダイレクト エディター テーブルで選択されたレコードが [参照の追跡] ペインに表示されているレコードを**参照する**、*および*、**参照されている**ことを示します。



*メモ・複数のダイレクト エディター レコードが選択されている場合、フォーカスされているレコードのみの参照が表示されます。さらに、複数のテーブルが **[参照テーブル]** セクションに表示された場合、ダイレクト エディター テーブルで選択されたレコードが参照する、または複数のテーブルのレコードによって参照されることを示*

*します。関連するレコードの参照を表示するには、**[参照テーブル]** セクションにある任意のテーブルをクリックします。*



***ヒント・**[参照の追跡] ペインでは、セル内をダブルクリックしてレコードの参照間を簡単に移動することができます。*

破損した参照のインジケータ

ダイレクト エディターのテーブル レコードが、もう存在しない外部キー レコードを参照している場合があります。InstallShield は今回より、このような破損した参照を持つセルに注意が向くように、赤い背景色を使用します。

たとえば、Component テーブルの **Directory_** 列が、Directory テーブルで見つからないディレクトリ名を参照する場合、**Directory_** 列が赤い背景色で表示されます。



***メモ・**ダイレクト エディターの破損した参照インジケータは、[オプション] ダイアログ ボックスの [プリファレンス] タブにある **[参照の整合性を維持]** チェックボックスには関連していません。**"参照の整合性を維持"** 設定は、プライマリ キーを変更したときに、外部キーを更新することを目的とし、破損した参照インジケータは、親の無いレコードを簡単に識別できるよう、破損した参照を表示します。このため、破損した参照は **"参照の整合性を維持"** 設定で行った選択に関係なく表示されます。*

スイート – UI 機能強化



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



***エディション・**アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

様々な用途を広くサポートするために、InstallShield ではアドバンスト UI およびスイート/アドバンスト UI プロジェクトに次の機能が追加されています:

- 新しい [Windows を閉じる] および [イベントの停止] UI アクション
- スイートでスプラッシュ スクリーンのロード追加

新しい [Windows を閉じる] および [イベントの停止] UI アクション

InstallShield には、以下のような新しい UI アクションが搭載されています。

テーブル 1-2 ・ 新しいコントロール アクション

アクションの種類	説明
ウィンドウを閉じる	この種類のアクションは、メインのウィザード ページまたは 2 番目のウィンドウを閉じるか、場合によっては 2 番目のウィンドウを条件付きで閉じます。 [ウィンドウを閉じる] アクションには、次の定義済みリターン コード ID に対応するパラメータを使用できます: IDOK、IDCANCEL、IDABORT、IDRETRY、IDIGNORE、IDYES、IDNO、および IDCLOSE [ウィンドウを閉じる] アクションの動作は、ウィザード ページおよび 2 番目のウィンドウで次のように多少異なります: ・ ウィザード ページの場合、[ウィンドウを閉じる] アクションはそのリターン コード パラメータが IDCANCEL に設定されている場合、エンド ユーザーがウィザードをキャンセルできるプロンプトを表示 (また、エンド ユーザーが [はい] を指定した場合にウィザードを中断します)。その他のリターン コード ID の場合、ウィザードがすぐに閉じます。 ・ 2 番目のウィンドウの場合、[ウィンドウを閉じる] アクションは 2 番目のウィンドウを閉じます。また特殊な場合、たとえば 2 番目のウィンドウが ISRMFilesInUse および ISRMFileInUse の場合は指定されたリターン コード値が戻されます。 InstallShield では現在、指定されたリターン コード ID によって異なるカスタム動作を含む、次の 2 番目のウィンドウが提供されています: ・ ISDownloadProgress ・ ISPromptForSourceMedia ・ ISFilesInUse ・ ISRMFilesInUse ・ ISUpgradeParcel ・ ISSQLBrowse
イベントの停止	この種類のアクションは、後に続くアクションの処理を条件付きで停止します。たとえば、このアクションを使ってボタンのデフォルト動作を抑制することができます。

詳細については、「ウィザード インターフェイス内の要素のアクションを構成する」を参照してください。

スイートでスプラッシュ スクリーンのロード追加

ロード処理に 0.5 秒以上を要するスイート インストール中、InstallShield は今回より、[インストールようこそ] ダイアログが表示される前に、プログラムが起動済みで、ロード処理を完了する必要があることを示す、スプラッシュ スクリーンを表示します。スプラッシュ スクリーンに、InstallShield は提供されている中から一番大きい setup.exe アイコンを使用し、進行状況バーも含みます。

[プロセスの強制終了] カスタム アクションの強化内容



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

[プロセスの強制終了] カスタム アクションの設定に新しい **"プロセス"** 設定が追加されました。この設定を使って、[プロパティ マネージャー] を使用してプロパティを作成、およびアクションが正しく動作するようその値を正しくフォーマットする必要なしに、強制終了する実行可能ファイルの名前またはプロセスの PID を直接入力することができます。

詳細については、「プロセスの強制終了カスタム アクションの呼び出し」を参照してください。



***ヒント・"プロセス"** 設定の値は、ISTerminateProcesses プロパティに書き込まれる場合があります。InstallShield 2015 以前から移行したプロジェクトなど、**"プロセス"** 設定に値が指定されていない追加の kill-process カスタム アクションがある場合、ISTerminateProcesses プロパティを共有して使用すると予期しない動作が発生する可能性があります。*

コンポーネントの属性に使用するデフォルト値を設定できる機能

InstallShield テーブルで、コンポーネント属性に使用されるデフォルト値を設定するための新しいプロパティのサポートが追加されました。InstallShield テーブルに MsiComponentAttributes のプロパティが存在する場合、その値は Component テーブル内のデフォルト Attributes 列値を 8 から指定の値にオーバーライドします。

たとえば、新しいコンポーネントを 64 ビットとする場合、MsiComponentAttributes 値に 256 を追加します。264 (64 ビット共有の場合)、または256 (64 ビット非共有の場合) を指定できます。これによって、"64 ビット コンポーネント" および "共有" 設定 ([コンポーネント] ビューの [全般] 領域) がそれぞれ [はい] または [いいえ] に更新されます。

Component テーブルの Attributes 列を計算する時に使用されるビット値についての詳細は、MSDN ライブラリの「Component Table」ページを参照してください。



***メモ・**コンポーネント属性に使用されるデフォルト値を設定するには、各プロジェクトの **ダイレクト エディタ** にある InstallShield テーブルに、手動で MsiComponentAttributesプロパティを更新する必要があります。この場合、製品構成の **"テンプレートの概要"** 設定は無視されます。*

追加のビューで、機能ごとに項目をフィルターできる機能

次のビューには今回より、プロジェクトに含まれる任意の機能ごとにビュー リストをフィルターすることができる [ビュー フィルター] が用意されています。

- **[環境変数] ビュー**—このビューの上部にある [ビュー フィルター] リストを使って、プロジェクトに含まれる特定機能に関連付けられた環境変数を表示/非表示にすることができます。[ビューリスト] から機能を選択して、その機能のみを後に続くイベント (たとえば、環境変数の作成、変更、または削除) に関連付けることができます。最後に、プロジェクトに含まれるすべての環境変数を表示するには、[ビュー フィルター] リストで [すべてのアプリケーション データ] オプションを選択します。詳細については、「[環境変数] ビュー」を参照してください。
- **[テキスト ファイルの変更] ビュー**—このビューの上部にある [ビュー フィルター] リストを使って、プロジェクトに含まれる特定機能に関連付けられたテキスト ファイルの変更セットを表示/非表示にすることができます。[ビューリスト] から機能を選択して、その機能のみを後に続くイベント (たとえば、変更セットの作成、変更、並べ替え、または削除」) に関連付けることができます。結果となる変更は、機能がインストールされるときにターゲット システム上で実行時に行われます。最後に、プロジェクトに含まれるすべてのテキスト ファイルの変更セットを表示するには、[ビュー フィルター] リストで [すべてのアプリケーション データ] オプションを選択します。詳細については、「[テキスト ファイルの変更] ビュー」を参照してください。
- **[INI ファイルの変更] ビュー**—このビューの上部にある [ビュー フィルター] リストを使って、プロジェクトに含まれる特定機能に関連付けられた初期化 (.ini) ファイルを表示/非表示にすることができます。[ビューリスト] から機能を選択して、その機能のみを後に続くイベント (たとえば、.ini ファイルの作成、インポート、変更、または削除」) に関連付けることができます。結果となる変更は、機能がインストールされるときにターゲット システム上で実行時に行われます。最後に、プロジェクトに含まれるすべての .ini ファイルを表示するには、[ビュー フィルター] リストで [すべてのアプリケーション データ] オプションを選択します。詳細については、「[INI ファイルの変更] ビュー」を参照してください。

デジタル署名の更新

InstallShield 2015 より、インストールおよびファイルをビルド時に署名する際、SHA-256 ハッシュ アルゴリズムを使ったデジタル証明書を使用できるサポートが追加されました。

InstallShield 2016 では、Windows Installer および InstallScript プロジェクトの SHA-256 デジタル証明書サポートが次のように強化されています:

- **[証明書の選択]** ダイアログ ボックスの新しい [署名ダイジェスト] ドロップダウンを使って、ダイジェストの種類を指定できる機能
- 今回より、RFC3161 タイムスタンプがサポートされていて、settings.xml で次の記述を含んで指定できます:
  - .msi、.exe、および .dll ファイルでは、**DigitalSignature/@Timestamp** は **Authenticode** または **RFC3161 サーバー**
  - Windows アプリ パッケージファイルに使用される **DigitalSignature/@TimestampRFC3161** ファイルは **RFC3161 サーバー** でなくてはなりません
- 証明書ストアにある類似した名前の証明書も処理します



***重要**・2016 年 1 月以降に作成またはタイムスタンプが付けられたすべての新しい署名は、SHA-256 に基づく必要があります。SHA-1 証明書を使って署名されているすべてのファイルを継続してサポートするためには、2016 年 1 月以前の日時を使ったタイムスタンプを含める必要があります。これらのファイルは、すべての現在のバージョンの Windowsですべての SHA-1 サポートが停止される 2020 年 1 月 14 日まで、MOTW (Mark of the web) システムを使って引き続き使用することができます。*

InstallShield 2015 SP1 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

Windows 10 のサポート

InstallShield は、Windows 10 をサポートします。

Microsoft Visual Studio 2015 のサポート

InstallShield は、Visual Studio 2015 をサポートします。このバージョンの Visual Studio 内部から InstallShield プロジェクトを作成できます。

Microsoft App-V 5.1 のサポート

InstallShield および InstallShield の Microsoft App-V アシスタントには、Microsoft App-V 5.1 クライアント上で実行できる仮想アプリケーションを作成するためのサポートが含まれています。

新しい App-V 5.1 用 InstallShield 前提条件

InstallShield には、生成された App-V 5.x パッケージに setup.exe を含めるときに使用できる、新しい InstallShield 前提条件が含まれています。InstallShield 前提条件をリリースに含める必要がある場合、Setup.exe セットアップランチャーが必要になることに注意してください。

- Microsoft App-V 5.1 Desktop Client

この InstallShield 前提条件用の再配布可能ファイルは Microsoft から入手しなくてはならないため、InstallShield 内部からダウンロードすることはできません。Microsoft から再配布可能ファイルを入手した後、InstallShield 前提条件エディターで前提条件を編集するときに表示される場所に配置してください。必要な前提条件についての詳細は、https://technet.microsoft.com/ja-jp/library/mt346482.aspxを参照してください。

App-V アシスタントの [オペレーティング システム] オプションについて、App-V 5.x バージョン用に Windows 10 を追加

App-V 5.1 より、Windows 10 32 ビット版および 64 ビット版のオペレーティング システムがサポートされています。これに伴い、Microsoft App-V アシスタントの [パッケージ情報] ページで、App-V 5.x が選択されている際に選択可能なオペレーティングシステムとして次が追加されました:

- Windows 10 (32 ビット版)
- Windows 10 (64 ビット版)

Microsoft Visual C++ 2015 および .NET Framework 4.6 用の新しい InstallShield 前提条件

InstallShield には、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加可能な新しい前提条件が含まれています:

- Microsoft Visual C++ 2015 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2015 再配布可能パッケージ (x64)
- Microsoft .NET Framework 4.6 Full
- Microsoft .NET Framework 4.6 Web

これらの前提条件は、サポートされているターゲット システムに適切なテクノロジをインストールします。

DialogSetInfo 関数の新しい定数

InstallScript 関数 DialogSetInfo の nInfoType パラメーターに新しい定数が使用できます:

DLG_INFO_ALTIMAGE_HIDPI—この定数は、ダイアログで高 DPI イメージを指定します。高 DPI イメージは BMP、GIF、JPEG、PNG、および TIFF などをサポートします。透明化が必要な場合、それをサポートする PNG などのイメージ タイプを使い、ダイアログ内で szInfoString が表示するイメージの名前 (オプションでパスを含むことが可能) を指定します。このパラメーターは、ダイアログの左側にある標準インストール イメージを表示する ダイアログすべてに適用されます。

DLG_INFO_ALTIMAGE_HIDPI が nInfoType で渡される場合、次のパラメーター値が必要です:

- szInfoString—表示するイメージ ファイルの名前。オプションでパスを含めることもできます。ファイルが指定されなかった場合、SUPPORTDIR に存在するものと見なされます。ファイルが存在しない場合は、DialogSetInfo が ISERR_FILE_NOT_FOUND を返します。
- nParameter—DPI 拡大/縮小率。たとえば、200% 拡大する場合 200、150% の場合 150 など。サポートされている最小縮小値は 25 です。この値に 0 が渡された場合、イメージは何も表示されません。If DLG_INFO_ALTIMAGE_REVERT_IMAGE が渡されると、以前に使用されたイメージが表示されます。

この関数は、InstallScript プロジェクトおよび InstallScript MSI プロジェクトの InstallScript イベントで使用できます。

詳細については、「DialogSetInfo」を参照してください。

アーキテクチャの検証を無効化するオプションの追加

[リリース] エクスプローラーの [アーキテクチャの検証] オプションに **[なし]** オプションが追加され、ビルド時にアーキテクチャの検証をバイパスすることを指定できるようになりました。

詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。

InstallShield 2015 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

Windows 10 ベース システムのサポート

InstallShield は、Windows 10 をサポートします。

Windows 10 をターゲットにする

Windows 10 が搭載されているシステム上では、Windows Installer プロパティ VersionNT および VersionNT64 が 603 を示します。これは、もともと Windows 8.1 のバージョン番号として使用されたものです。従って、Windows 10 をターゲットに特定して .msi パッケージの条件を作成することは不可能です。

Windows Installer 5.0 および Windows 7 より、.msi パッケージの DLL アクションには shim が適用されるため、オペレーティング システム バージョンの取得が阻止されます。API の **GetVersion**、**GetVersionEx**、および **RtlGetVersion** は、もともと Windows Vista のバージョン番号である 6.0.6000 を返します。従って、DLL カスタム アクション、または InstallScript カスタム アクション (DLL として実装される) から実際の Windows のバージョン番号を取得することはできません。

前述の Windows Installer 動作のため、.msi パッケージが実行中のバージョンを検出することは容易ではありません。ターゲット システムの OS 要件を指定できる領域、たとえば基本の MSI および InstallScript MSI プロジェクトでは、プロジェクト アシスタントの [インストール要件] ページで、新しい実行時の動作を反映するように、[Windows 8.1] オプションの名前が **「Windows 8.1 または Windows 10」** に変更されました。

InstallScript、アドバンスト UI、およびスイート/アドバンスト UI プロジェクトでは、ターゲット システム上に存在する Windows のバージョン (Windows 10 を含む) を正しく検出できる機能が提供されています。したがって、インストールが Windows 10 をターゲットとする、または除外する必要がある場合、これらのプロジェクトの種類を使って、実際のターゲット システム プラットフォームに基づいて .msi パッケージを実行する条件を作成できます。

Windows 10 にインストール可能な InstallShield 前提条件は、必要に応じて、これらのシステムにインストールされるように更新されています。以前これらのシステムでは、前提条件がデフォルトで実行されない場合がありました。

Windows 10 の InstallScript 言語サポート

次の構造メンバーと定義済み定数が InstallScript 言語に追加されました:

- **SYSINFO.WINNT.bWin10** – 新しい SYSINFO 構造メンバーです。オペレーティング システムが Windows 10 の場合、この値は TRUE です。(これは、InstallScript イベント ドリブン型コードに適用し、カスタム アクションには適用しません。)
- **ISOSL_WIN10** – **FeatureFilterOS** 関数と SYSINFO 構造変数と共に使用できる新しい定義済み定数です。これは、ターゲット システムが Windows 10 を実行中であることを示します。

詳細については、「FeatureFilterOS 関数と SYSINFO 構造変数」を参照してください。

Microsoft Visual Studio 2015 のサポート

InstallShield は、Visual Studio 2015 プレビュー版をサポートします。このバージョンの Visual Studio 内部から InstallShield プロジェクトを作成できます。

デジタル署名の強化機能

InstallShield には、ビルド時にインストールおよびファイルにデジタル署名を行うための、いくつかの強化機能が含まれています。

SHA-256 デジタル証明書のサポート

InstallShield では、インストールおよびファイルをビルド時に署名する際、HA-256 ハッシュ アルゴリズムを使ったデジタル証明書を使用できます。

SHA-1 はセキュリティの脆弱性があるため、SHA-256 の使用が推奨されます。Microsoft は、Windows では 2016 年 1 月以降に SHA-1 証明書を使って署名およびタイムスタンプが追加されているアイテムを信頼しないことを発表しました。さらに、証明書を発行する組織である証明機関では、SHA-1 証明書が段階的に廃止されます。したがって、InstallShield プロジェクトに含まれる任意の SHA-1 証明書は、SHA-256 証明書と差し替えることが推奨されます。最新情報および特定の詳細については、証明機関にお問い合わせください。

InstallShield でリリースに署名するための SHA-1 証明書を SHA-256 証明書に置き換えるには、[リリース] ビューの [署名] タブを使って、現在の証明書への参照を SHA-256 証明書と置き換えます。

プロジェクトで、SHA-256 証明書を使った署名が構成されている場合、InstallShield はビルド時に署名を行うファイルの署名に SHA-256 ハッシュを使用します。プロジェクトで SHA-1 証明書を使った署名が構成されたままになっている場合、InstallShield は SHA-1 ハッシュを使用します。また、SHA-1 証明書を使用すると、今回より SHA-1 の使用についてアラートするビルド警告 -7346 が発生します。

InstallShield の以前のバージョンでは、SHA-1 または SHA-256 証明書のいずれかを使って署名が行なわれる際、ファイルの署名に SHA-1 ハッシュが使用されました。

詳細については、「デジタル署名とセキュリティ」を参照してください。

証明書を参照するための証明書ストアを使用できる機能

ファイルおよびインストールの署名に使用するデジタル署名情報を指定するとき、InstallShield では今回より、使用する証明書を含む証明書ストアを参照することができます。このサポートは、マシン上の .pfx 証明書ファイルを指定する方法の代替として提供されています。

証明書ストアまたは .pfx 証明書を使うかどうかを指定するには、[リリース] ビューの [署名] タブにある "デジタル証明書ファイル" 設定を使用します。この設定で省略記号ボタン (...) をクリックすると、新しい [証明書の選択] ダイアログ ボックスが開きます。このダイアログ ボックスを使って、ストア名 (Personal、Trusted Root Certification Authorities、Enterprise Trust、Intermediate Certification Authorities)、ストアの場所 (ユーザー、マシン)、および特定の証明書を識別するためのサブジェクトといった情報を指定することができます。別の方法として、このダイアログ ボックスで使用する .pfx ファイルの名前とパスを指定することもできます。

ストアにパスワード付きでインポートされた証明書をプロジェクトで使用するように構成すると、ビルド時、InstallShield がプロジェクトのファイルに署名を行うときに、Windows がパスワードをプロンプトします。Windows が使用する強力なキー保護のため、InstallShield がパスワードを暗号化サービス プロバイダーに提供することはできません。

証明書ストアを、パッチおよび QuickPatch パッケージの署名に使用することもできます。

- パッチの証明書ストアまたは .pfx 証明書情報を指定するには、[パッチのデザイン] ビューでパッチの構成の [デジタル署名] タブを使用します。
- QuickPatch パッケージで証明書ストアまたは .pfx 証明書情報を指定するには、QuickPatch プロジェクトで [一般情報] ビューの [ビルドの設定] 領域を使用します。この領域には、新しいサポートを含む [デジタル署名] タブがあります。

さらに、ビルド済みの InstallScript リリースの署名に使用できるコマンドライン ツール iSign.exe が更新され、証明書ストアにある証明書を使用できるようになりました。

詳しくは、次を参照してください:

- デジタル署名とセキュリティ
- [証明書の選択] ダイアログ ボックス
- リリースの [署名] タブ ([リリース] ビュー内のリリース)
- [デジタル署名] タブ ([パッチのデザイン] ビューのパッチ構成用)
- [デジタル署名] タブ (QuickPatch プロジェクト)
- iSign.exe

InstallShield では、今回より .spc および .pvk ファイルを使った署名がサポートされていません。これらのファイルを .pfx ファイルに変換する方法については、「デジタル署名とセキュリティ」を参照してください。

UAC ダイアログ ボックスにプログラム名を指定できる機能

[リリース] ビューの [署名] タブには、"署名の説明" 設定があります。この設定を使って、ビルド時に InstallShield が署名を行なう Setup.exe ファイル、.msi ファイル、その他のインストール ファイルの UAC ダイアログ ボックスで "プログラム名:" ラベルの右側に表示するテキストを指定します。UAC ダイアログ ボックスは、エンド ユーザーが署名されたファイルを起動したとき、昇格された権限が必要な場合に開きます。

"署名の説明" 設定を空白のままに残すと、InstallShield は UAC ダイアログ ボックスのテキストとして、ファイル名を拡張子なしで使用します。

詳細については、「リリースの [署名] タブ」を参照してください。

.pfx ファイルまたは証明書ストアにある証明書を使って、メディア ヘッダー ファイルにデジタル署名できる機能

InstallShield では今回より、.pfx ファイルを使って、メディア ヘッダー ファイル (.hdr ファイル) に署名することができます。.hdr ファイルは、InstallScript プロジェクトの One-Click Install インストールに使用されます。別の方法として、証明書ストア内の証明書を使って、メディア ヘッダー ファイルに署名することもできます。

以前は、.pfx ファイルの代わりに .spc および .pvk ファイルを使って署名しなくてはなりませんでした。

デジタル署名情報を指定できるオートメーション インターフェイス サポート

オートメーション インターフェイスには、.pfx ファイルおよび署名ストア情報の指定がサポートが含まれています。また、署名の説明を指定することもサポートされています。

基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、およびマージ モジュール プロジェクトには、ISWiReleases オブジェクトに 2 つの読み書き文字列プロパティが含まれています。

- **DigitalCertificateInfo**—このプロパティは、マシン上の .pfx ファイルへのパスまたは証明書ストアを示す文字列を取得または設定します。
- **SignatureDescription**—このプロパティは、署名の説明を取得または設定します。

詳細については、「ISWiRelease オブジェクト」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの ISWiSuiteReleases オブジェクトは、これらの同じプロパティを含みます。詳細については、「ISWiSuiteRelease オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)」を参照してください。

64 ビット開発システム上で、ソースマシンのレジストリの 32 ビットおよび 64 ビット領域の両方を表示できる機能

InstallShield を 64 ビット開発システム上で使用する場合、InstallShield が表示する [レジストリ] ビューは、使用中のマシンのレジストリの 32 ビットおよび 64 ビット領域の両方を表示します:

- HKEY_LOCAL_MACHINE¥Software
- HKEY_LOCAL_MACHINE¥Software¥Wow6432Node

このサポートによって、このビューのインストール先ペインの適切な領域ににソース領域からのエントリをドラッグ アンド ドロップできるようになるため、64 ビット マシン上でのインストールの開発が容易になります。

以前、64 ビット開発システム上で InstallShield を使用した場合、InstallShield の [レジストリ] ビューのソース ペインにはレジストリの HKLM¥Software 部分にある 64 ビットデータが表示されませんでした。さらに、ソース ペインには、マシンの HKLM¥Software¥Wow6432Node 領域の 32 ビットデータが HKLM¥Software 領域に表示されました。

インストールがレジストリ データを、32 ビット領域にリダイレクトせずに、64 ビット ターゲット システム上のレジストリの 64 ビット領域にインストールする場合、レジストリ データを 64 ビットとしてマークされたコンポーネントに配置します。[レジストリ] ビューのソースペインから 64 ビット データを、ビュー内のインストール先ペインにドラッグするだけでは、そのコンポーネントが 64 ビットであるとマークされません。

この機能は、基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- 32 ビットと 64 ビット システムにおけるインストールの開発およびビルドの違い
- レジストリ エントリをドラッグアンドドロップしてレジストリ キーを作成する
- [レジストリ] ビュー

アドバンスト UI およびスイート/アドバンスト UI プロジェクトで形式化された式を埋め込んで実行時に解決できる機能

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの様々な領域では、ファイル、レジストリ エントリ、オペレーティング システム、その他の詳細情報をターゲット システムで照会するオブジェクト式を埋め込むことができます。これによって、ターゲット システム固有の条件に基づいて、実行時にアドバンスト UI またはスイート/アドバンスト UI の多くの設定をダイナミックに構成することができます。オブジェクト式には、この規則を使用します:

[@Object(Parameters, ...).Property(Parameters, ...)]

各オブジェクト式には、オブジェクト固有プロパティの集まりである、オブジェクトへの参照が含まれています。オブジェクトとプロパティにはパラメーターを含めることができます。

たとえば、次の Platform オブジェクト式は、アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のマシンのアーキテクチャ (x86、x64、IA64、ARM、または不明) を取得します:

[@Platform.Architecture]

次の Registry オブジェクト式は、HKLM¥Software¥My Company Name¥My Product Name レジストリ キーの RegisteredOwner 値のデータを取得します:

[@Registry(HKLM¥Software¥[COMPANY]¥[PRODUCT], true).KeyValue(RegisteredOwner)]

プロパティ式その他のオブジェクト式のような、別の形式化された式内にオブジェクト式を埋め込むことができます。次の式では、Registry オブジェクト式が File オブジェクト式にパラメーターの一部として埋め込まれています。

[@File([@Registry(HKLM¥Software¥MyProduct).KeyValue(MyProductPath)]¥MyProduct.exe).Version]

MyProduct.exe ファイルが MyProductPath 値データに指定された場所にあるとき、File オブジェクト式はファイルのバージョンを返します。ファイルがその場所に見つからなかった場合、またはレジストリ値が存在しなかった場合、File オブジェクト式は空白文字列を返します。

InstallShield ではまた、アドバンスト UI またはスイート/アドバンスト UI インストール内のパッケージにコマンドラインを使って、文字列の一部としてリテラル角括弧を渡すこともできます。たとえば、[¥[]Text[¥]] のようなコマンドラインは実行時に [Text] として解決され、角括弧と共にパッケージに渡されます。以前、角括弧で囲まれた文字列はプロパティとして扱われたため、インストールが実行時に解決しようとしました。唯一のワークアラウンドは、実行時に角括弧を含むプロパティ値に解決する形式化されたプロパティ式 (たとえば、[PropertyForSquareBracketString]) を使う方法でした。

詳しくは、次を参照してください:

- アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトでオブジェクト式を書いて、ターゲット システムを検索する
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用可能な式のオブジェクト リファレンス

- Feature オブジェクト
- File オブジェクト
- Parcel オブジェクト
- Platform オブジェクト
- Registry オブジェクト

共通パッケージを異なるアドバンスト UI およびスイート/アドバンスト UI インストール間で、共通パッケージを共有できる機能

アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージを構成するとき、今回より、これを共有とマークすることができます。共有パッケージ機能を使って、2 以上のアドバンスト UI またはスイート/アドバンスト UI インストールが 1 つのパッケージを共有している場合に、アドバンスト UI およびスイート/アドバンスト UI 製品のすべてが削除されるまで、ターゲット システムにそのパッケージが保持されるようにします。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [パッケージ] ビュで選択されているパッケージをマークするには、新しい "共有" 設定に [はい] を選択します。さらに、パッケージを共有するすべてのアドバンスト UI およびスイート/アドバンスト UI プロジェクトで必ず同じ パッケージ GUID を使用してください。[パッケージ] ビューの "パッケージ GUID" 設定を使って、パッケージの GUID を確認および変更することができます。

このビルトイン共有サポートが有効になる前には、様々な状況下で予期しない結果が発生する可能性がありました。たとえば、一部の状況下で、ターゲットシステム上に 2 つの異なるアドバンスト UI またはスイート/アドバンスト UI インストールが共有パッケージをインストールした場合、1 つだけがアンインストールされたとき、共有パッケージも削除されました。共有パッケージが不足すると、残りのアドバンスト UI またはスイート/アドバンスト UI 製品が使用不可能となる可能性があります。別の状況下では、アドバンスト UI またはスイート/アドバンスト UI 製品が無いのにもかかわらず、共有パッケージが誤ってターゲット システムに残りました。

この機能は、アドバンスト UI およびスイート/アドバンスト UI プロジェクトに含まれるパッケージの種類 (.msi、.msp、.exe、.appx、InstallScript、基本の MSI および InstallScript MSI プロジェクト) で使用できます。

詳細については、次を参照してください。

- 異なるアドバンスト UI およびスイート/アドバンスト UI インストール間で、共通パッケージを共有する
- [パッケージ] ビュー

アドバンスト UI およびスイート/アドバンスト UI プロジェクトのリリースで、製品名、製品バージョン、およびスイート GUID をオーバーライドできる機能

InstallShield では今回より、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [一般情報] ビューで指定されたスイート GUID 値を、選択されたリリースの新しい値でオーバーライドすることができます。[一般情報] ビューの値をオーバーライドするには、[リリース] ビューでリリースの [ビルド] タブで使用できる新しい設定 (製品名、製品バージョン、およびスイート GUID) を使用します。

詳細については、「リリースの [ビルド] タブ」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトのリリースで、パス変数の値をオーバーライドできる機能

InstallShield では今回より、プロジェクト内の各リリースでプロジェクトのカスタムパス変数 (標準パス変数、環境パス変数、およびレジストリ パス変数) の値をオーバーライドすることができます。この機能を使って、ビルドする特定のリリースごとに、ビルド時にプロジェクト内の特定のファイルとフォルダーを別のファイルとフォルダーに置換することができます。

たとえば、この機能を使って、プロジェクト内の異なるリリースでイメージや EULA などの UI 要素を置換することができます。これによって、プロジェクトの異なるエディションまたは異なるブランド バージョンのインストールを簡単に生成することができます。

プロジェクトで 1 つ以上のパス変数をオーバーライドする場合、[リリース] ビュー内のリリースの [ビルド] タブに追加された "パス変数のオーバーライド" 設定を使います。

詳しくは、次を参照してください:

- リリースのカスタム パス変数の値をオーバーライドする
- "パス変数のオーバーライド" 設定

アドバンスト UI およびスイート/アドバンスト UI プロジェクトのオートメーション インターフェイス サポート

InstallShield では、InstallShield インターフェイスを直接開いて異なるビューで変更を加えることなく、オートメーション インターフェイスを利用してアドバンスト UI およびスイート/アドバンスト UI プロジェクト (.issuite ファイル) のほとんどの開発およびビルド プロセスを自動化することができます。オートメーション インターフェイスを通して、アドバンスト UI およびスイート/アドバンスト UI プロジェクトの様々な領域に、プログラムによるアクセスが可能です。オートメーション インターフェイスでは、多くの言語から呼び出してアドバンスト UI またはスイート/アドバンスト UI プロジェクトを作成、編集およびビルドできる COM インターフェイスが公開されます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトのトップ階層のオートメーション オブジェクトは、ISWiProject オブジェクトです。プロジェクトを開いて、変更、保存およびこれをと閉じるサンプル VBScript コードの始まりと終りは次の通りです:

```
Dim pProj As ISWiProject
Set pProj = CreateObject("ISWiAutoSuite22.ISWiProject")
pProj.OpenProject "C:¥InstallShield 2015 Projects¥Project1.issuite"
' 変更をここで行う
pProj.SaveProject
pProj.CloseProject
```

メソッドを呼び出し、プロパティを取得および設定、およびコレクションにアクセス、またプロジェクトに機能やパッケージを追加、条件の構成、その他を行うことができます。詳細については、ドキュメントの次のセクションを参照してください:

- ≫ アドバンスト UI およびスイート/アドバンスト UI プロジェクトのオートメーション オブジェクト
- ≫ アドバンスト UI およびスイート/アドバンスト UI プロジェクトのオートメーション コレクション

アドバンスト UI およびスイート/アドバンスト UI オートメーション インターフェイスの ProgID は ISWiAutoSuiteAutomation Interface Version.ISWiProject です。その他のプロジェクト タイプの ProgID は ISWiAutoAutomation Interface Version.ISWiProject です。

新しい Microsoft Visual C++ 2015、.NET Framework 4.6 その他用の InstallShield 前提条件

InstallShield には、プロジェクトに追加することができる新しい InstallShield 前提条件が含まれています:

- Microsoft Visual C++ 2015 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2015 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2013 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2013 再配布可能パッケージ (x64)
- Microsoft .NET Framework 4.6 Full
- Microsoft .NET Framework 4.6 Web
- Microsoft .NET Framework 4.5.2 (完全版)
- Microsoft .NET Framework 4.5.2 (Web インストーラー)
- Microsoft SQL Server 2012 Express SP2 (x86)
- Microsoft SQL Server 2012 Express SP2 (x86 & x64Wow)
- Microsoft SQL Server 2012 Express SP2 (x64)
- Microsoft SQL Server 2012 Express SP2 LocalDB (x86)
- Microsoft SQL Server 2012 Express SP2 LocalDB (x64)
- Microsoft SQL Server 2012 Express SP2 Management Objects (x86)
- Microsoft SQL Server 2012 Express SP2 Management Objects (x64)
- Microsoft SQL Server 2012 Express SP2 System CLR Types (x86)
- Microsoft SQL Server 2012 Express SP2 System CLR Types (x64)
- Windows 7 (x86) 用 Internet Explorer 11.0
- Windows 7 と Windows Server 2008 R2 (x64) 用の Internet Explorer 11.0
- Microsoft ReportViewer 2012

これらの前提条件は、サポートされているターゲット システムに適切なテクノロジをインストールします。

Microsoft SQL Server 2012 Express SP2 前提条件は、Microsoft SQL Server 2012 Express SP1 前提条件にとって代わります。

Internet Explorer 10 および 11 用の新しい定義済みシステム検索

InstallShield には、ターゲット システム上で Internet Explorer 10 または Internet Explorer 11を確認する新しい定義済みのシステム検索が含まれています。インストールまたは製品でこれらのバージョンのどちらかが必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、これらのシステム検索の 1 つをプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

この変更は、基本の MSI および InstallScript MSI プロジェクトに含まれています。

Microsoft App-V 5.0 SP3 の仮想アプリケーションを作成するためのサポート、追加の App-V パッケージの強化機能

InstallShield および InstallShield の Microsoft App-V アシスタントには、Microsoft App-V 5.0 SP3クライアント上で実行できる仮想アプリケーションを作成するためのサポートが含まれています。また、Microsoft App-V アシスタントには、App-V 5.0 SP3 または App-V の以前のバージョンに適用する、新しい設定と機能が含まれています。

新しい pp-V 5.0 SP3 用 InstallShield 前提条件

InstallShield には、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加可能な新しい InstallShield 前提条件が含まれています:

- Microsoft App-V 5.0 SP3 Desktop Client (x86)
- Microsoft App-V 5.0 SP3 Desktop Client (x64)

これらの InstallShield 前提条件用の再配布可能ファイルは Microsoft から入手しなくてはならないため、InstallShield 内部からダウンロードすることはできません。Microsoft から再配布可能ファイルを入手した後、InstallShield 前提条件エディターで前提条件を編集するときに表示される場所に配置してください。

プライマリ アプリケーション ディレクトリの代わりに、仮想ファイル システムにファイルをマップできる機能

Microsoft App-V アシスタントを使って、今回より、仮想ファイル システム (VFS) にファイルをマップするように App-V パッケージを構成することができます。このサポートは、App-V 4.x および 5.xパッケージで使用できます。

VFS にファイルをマップするか、プライマリ アプリケーション ディレクトリを使用するかを指定するには、Microsoft App-V アシスタントの [ファイル] ページを使用します。このページのその他の [オプション] 領域には、新しいファイルマッピング リンクが含まれています。この新しいリンクをクリックすると、新しい [ファイル マッピング] ダイアログ ボックスが開き、適切なオプションを選択することができます。

[ファイル マッピング] リンクとダイアログ ボックスは、[プライマリ アプリケーション ディレクトリ] リンクとダイアログ ボックスを置換します。ここでは、プライマリ アプリケーション ディレクトリを指定することができます。

詳細については、[ファイル マッピング] ダイアログ ボックス」を参照してください。

仮想ファイル システムに完全な書き込みアクセス権を持つ App-V 5.x パッケージを作成できる機能

Microsoft App-V アシスタントを使って、今回より、作成中の App-V 5.x パッケージに仮想ファイル システム (VFS) の完全な書き込みアクセス権を持たせるかどうかを指定することができます。このサポートを使用するかどうかを指定するには、新しい [VFS の完全な書き込みアクセス権を許可する] チェックボックスを使います。このチェックボックスは、[詳細オプション] 領域にある [ファイル マッピング] リンクをクリックすると表示される Microsoft App-V アシスタントの [ファイル] ページから使用できる [ファイル マッピング] ダイアログ ボックスにあります。

詳細については、[ファイル マッピング] ダイアログ ボックス」を参照してください。

App-V5.x の詳細 COM 分離の設定を構成できる機能

Microsoft App-V アシスタントを使って、今回より、COM 分離の詳細設定を構成できます。このサポートは、App-V 5.x パッケージで使用できます。

新しい設定を構成するには、Microsoft App-V アシスタントの [パッケージ情報] ページを使用します。このページの [詳細オプション] 領域には、新しい [分離の設定] リンクが含まれています。この新しいリンクをクリックすると、新しい [分離の設定] ダイアログ ボックスが開きます。このダイアログ ボックスを使って、ローカル シ

ステムから COM オブジェクトを分離するかどうか、またはCOM オブジェクトとローカル システムの対話を許可するかどうかを指定できます。このダイアログ ボックスを使って、ローカル システムから名前付きオブジェクトを分離するかどうか、または名前付オブジェクトとローカル システムの対話を許可するかどうかを指定できます。

詳細については、パッケージの [分離オプション] ダイアログ ボックスを参照してください。

Microsoft App-V アシスタントは、InstallShield Premier Edition で使用できます。

強化機能

[ファイルとフォルダー] ビューのパフォーマンス強化

サイズが大きいプロジェクトの [ファイルとフォルダー] ビューををより短時間でロードできるように、InstallShield が強化されています。

機能ごとに [ファイルとフォルダー] ビューのアイテムをフィルターできる機能

[ファイルとフィルター] ビューには今回より、[すべてのアプリケーション データ] オプションの他に、プロジェクト内の各機能を一覧表示するビュー フィルターが追加されました。このフィルターを使って、このビューのインストール先ペインのファイルとフォルダーを表示または非表示にします。

- このビューで特定の機能に属するファイルとフォルダーのみを表示するには、[ビュー フィルター] リストで機能を選択します。
- 特定の機能にファイルまたはフォルダーを追加するには、[ビュー フィルター] リストで機能を選択します。次に、**[インストール先コンピューターのフォルダー]** ペインの適切な場所にファイルまたはフォルダーを追加します。
- プロジェクトに含まれるすべてのファイルとフォルダーを表示するには、[ビュー フィルター] リストで [すべてのアプリケーション データ] オプションを選択します。

新しいビュー フィルターは、以前の **[機能に新しいコンポーネントを追加する]** フィルターの代わりとなります。これまでは、選択された機能によってビュー内のファイルとフォルダーを表示または非表示にする機能がありませんでした。

詳細については、「[ファイルとフォルダー] ビュー」を参照してください。

基本の MSI インストールおよび InstallScript MSI インストールにおける PowerShell スクリプトからインストーラー セッションにアクセスできる機能

PowerShell カスタム アクション サポートが強化されました。今回より、実行中の基本の MSI インストールまたは InstallScript MSI インストールとの対話操作を可能にする、いくつかの cmdlet をサポートします: cmdlet を使って Windows Installer プロパティを取得および設定し、形式化された式の値を展開し、情報をログ ファイルに書き込むことができます。

この改訂された実装により、Windows Installer プロパティ IS_CLR_VERSION を利用して、カスタム アクションがダウンロードして PowerShell スクリプトを実行するセミコロンで区切られた .NET Framework のバージョン一覧を識別することができます。

詳細については、「PowerShell カスタム アクションの呼び出し」を参照してください。

トランスフォーム プロジェクトおよび MSI データベースでメジャー アップグレードを構成できる機能

トランスフォーム プロジェクトと MSI データベース プロジェクトには、今回より、メジャー アップグレードの作成に使用できる [アップグレード] ビューが含まれています。トランスフォーム プロジェクトまたは MSI データベース プロジェクトの間接編集モードでアップグレード エントリを追加するには、[アップグレード] ビューを開きます。[Windows Installer セットアップのアップグレード] ノードを右クリックしてから、[メジャー アップグレード アイテムの追加] を選択します。必要に応じて、右側のペインで設定を構成します。

詳しくは、次を参照してください:

- [アップグレード] ビュー
- メジャー アップグレード アイテムを追加する
- メジャーアップグレード
- アップグレード関する考慮事項
- 現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ

アドバンスト UI およびスイート/アドバンスト UI ログ ファイルへのパスワードの書き込みを防ぐ ISHiddenProperties プロパティの強化

ISHiddenProperties プロパティは、大文字と小文字を区別するプロパティ名のリストをセミコロン区切りで格納し、それらの値をデバッグ ログ ファイルに書き込まないようにできます。このプロパティを使って、パスワードその他の機密情報を含むプロパティのログ記録を防ぐことができます。ISHiddenProperties プロパティが強化されました。このプロパティを使って、次の状況下で値がログ記録されるのを防ぐのことができます:

- アドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルを起動したときに、エンド ユーザーが、コマンドラインを使ってプロパティの値を設定する場合。
- アドバンスト UI またはスイート/アドバンスト UI インストールがパッケージに渡すコマンドラインを使ってプロパティが構成される場合。これは [パッケージ] ビューの [共通] タブ、[操作] 領域で構成できます。

以前、ISHiddenProperties は、プロパティ値を変更するとログ記録される値に限ってログ記録を防ぎました。

詳細については、「アドバンスト UI およびスイート/アドバンスト UI デバッグ ログ ファイルへのプロパティ値の書き込みを防ぐ」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI インストールのウィザード インターフェイスで使用されているアンパサンドの解釈に関する強化

アドバンスト UI およびスイート/アドバンスト UI プロジェクトのウィザード インターフェイスの特定の領域で、アンパサンドの解釈が更新されました。

インストールでウィザード インターフェイスの次の任意の領域でアンパサンド (&) を使用する場合、今回より、インストールはアンパサンドをリテラル文字として表示します。以前はアンパサンドがキーボード ショートカットの先頭の文字として解釈されました。この変更はまた、これらの文字列にアンパサンドを含む値に解決するプロパティが含まれている場合にも適用します。

- ウィザード ページまたは 2 番目のウィンドウのヘッダー領域にある文字列 — これは [ウィザード インターフェイス] ビューでウィザード ページまたはウィンドウの "タイトル" 設定で構成されます。
- ウィザード ページのキャプション バー — これは [ウィザード インターフェイス] ビューで [ウィザード ページ] ノードの "ウィザード キャプション" 設定で構成されます。
- コマンド リンク コントロールの [補足説明] 領域 — これはウィザード ページまたはウィンドウの "メモ" 設定で構成されます。

- イメージ コントロールの代替テキスト – これはウィザード ページまたはウィンドウにあるイメージ コントロールの "代替テキスト" 設定で構成されます。

ほとんどのラベル コントロールでは、今回より、"スタイル" 設定の下にある SS_NOPREFIX サブ設定のデフォルトの選択が True となりました。以前はデフォルトで False が選択されていました。このサブ設定が True 値に設定されていることで、これらのコントロールの文字列エントリに含まれるアンパサンドが誤ってキーボード ショートカットとして解釈されないようにし、コントロールの文字列に含まれるアンパサンドをウィザード インターフェイスのアンパサンドとして正確に表示します。これは、定義済みウィザード ページだけでなく、ビルトイン デフォルト ウィザード ページとウィンドウにも適用します。この SS_NOPREFIX サブ設定の変更に関する唯一の例外は、キーボード ショートカットを含むラベル コントロールです。たとえば、デフォルトの定義済み [Web 配布] ウィザード ページには、対応するラベルにキーボード ショートカットが含まれている可能性があるいくつかのテキスト ボックスがあります。したがって、このようなコントロールの SS_NOPREFIX サブ設定のデフォルト値は False のままとなります。

InstallShield ヘルプ ライブラリには、ウィザード インターフェイスにおけるアンパサンドおよびキーボード ショートカットの使用をサポートするためのプロジェクト構成方法について説明する、新しい「ウィザード インターフェイスにおけるキーボード ショートカットの指定とアンパサンド (&) の使用」ヘルプ トピックが追加されました。

InstallShield 2014 以前のアドバンスト UI または スイート/アドバンスト UI プロジェクトを InstallShield 2025 にアップグレードすると、アンパサンドの解釈に関する前述の変更の多くが自動的に組み込まれますが、すべてが変更されるわけではありません。詳細については、「InstallShield 2014 以前のプロジェクトをアップグレードする」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI インストールにおけるファイル連関連タイプの条件のチェックで複数のターゲット システム パスを検索できる機能強化

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、機能、またはウィザード インターフェイス条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、次のファイル関連タイプの条件チェックを使用できます:

- **ファイルの存在**—ターゲット システムをチェックして、特定のファイルの有無を確認します。
- **ファイルの比較**—ターゲット システムをチェックして、特定のファイルの特定の情報 (日付、バージョン番号、またはコンポーネント) を確認します。

これら両方の条件チェックが拡張され、ターゲット システム上の複数のパスでファイルを確認する条件を定義できるようになりました。以前、各条件はターゲット システム上にある特定の場所を 1 つだけ確認することができました。

このサポートを有効にするため、これらの条件に既存する "パス" 設定がファイルの名前だけを入力できるように拡張されました (つまり、オプションでパスが不要となりました)。パスを入力しない場合、新しい "検索パス" 設定を使って、インストールが実行時に特定のファイルを確認するセミコロン区切りのパス一覧を指定することができます。この設定にプロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールがこれらの式の値を拡張します。

たとえば、ターゲット システム上でレジストリで定義されたパスだけでなく、PATH 環境変数で定義されたすべてのディレクトリで指定されたファイルを検索するには、"検索パス" 設定に次の値を入力します:

```
[%PATH];[@Registry(HKLM¥SOFTWARE¥MyPath).KeyValue(MyValue)]
```

詳細については、次を参照してください。

- "ファイルの存在" 条件の設定

- "ファイルの比較" 条件の設定

アドバンスト UI およびスイート/アドバンスト UI プロジェクトで、特定の種類のファイル関連条件のチェックと共に InstallShield 前提条件をインポートする際の機能強化

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、今回より、InstallShield 前提条件でサポートされている次の条件がサポートされています:

- ある前提条件で、[**ファイルが存在しているかしていないか**] または [**特定の日付を持つファイルが存在する**] という条件を使用していて、かつ、そのファイルのパスの一部に対して、角括弧で囲まれたレジストリ キーを参照している場合、基本の MSI、InstallScript、InstallScript MSI インストールでは、レジストリ キーがレジストリ値で解決されます。アドバンスト UI およびスイート/アドバンスト UI インストールにおける条件の形式化された式のサポートが拡張されたため、これらの種類の前提条件は、アドバンスト UI およびスイート/アドバンスト UI プロジェクトに適切にインポートされ、実行時に解決されます。以前、角括弧に囲まれたレジストリ キーが実行時に解決されず、これらの条件が常に False と評価されました。
- ある前提条件で、[**ファイルが存在しているかしていないか**] または [**特定の日付を持つファイルが存在する**] という条件を使用していて、そのファイルのパスを省略した場合、基本の MSI、InstallScript、InstallScript MSI インストールでは、ターゲット システムで PATH 環境変数で定義されているすべてのディレクトリで、指定されたファイルが検索されます。アドバンスト UI およびスイート/アドバンスト UI インストールにおける条件の形式化された式のサポートが拡張されたため、これらの種類の前提条件は、アドバンスト UI およびスイート/アドバンスト UI プロジェクトに適切にインポートされ、実行時に解決されます。以前、ファイルへのパスが解決されなかったため、この種類の条件は常に False と評価されました。

Oracle Instant Client 用の InstallShield前提条件は、ファイル関連の条件を使用する前提条件の例です。これらの種類の InstallShield 前提条件をアドバンスト UI またはスイート/アドバンスト UI プロジェクトにインポートすると、今回より、[パッケージ] ビューの "検出条件" 設定でファイル関連の条件が正しく構成されます。

詳細については、「InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める」を参照してください。

オートメーション インターフェイスを使ってプロジェクトをアップグレードできる機能

オートメーション インターフェイスには、InstallShield の以前のバージョンから現在のバージョンへのプロジェクトのアップグレードがサポートされています。オートメーション インターフェイスを使ってプロジェクトをアップグレードするには、ISWiProject オブジェクトの ForceUpgrade メソッドを使用します。

この強化機能は、アドバンスト UI、基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、および スイート/アドバンスト UI プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- ISWiProject オブジェクト
- ISWiProject オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

オートメーション インターフェイスを使って、リリースに仮想マシン構成を選択できる機能

オートメーション インターフェイスでは、指定のリリースに仮想マシン構成を選択することができます。これによって、ビルド時に選択されたリリースを仮想マシン (VM) に配布する際に使用する構成を指定することができます。

基本の MSI、InstallScript、および InstallScript MSI プロジェクトでは、ISWiReleases オブジェクトに新しい読み書きプロパティが含まれています。スイート/アドバンスト UI プロジェクトでは、ISWiSuiteReleases オブジェクトに同じ新しい読み書きプロパティが含まれています。

- VMConfig—このプロパティは、リリースを VM に配布するときに使用する VM 構成設定グループの名前を取得または設定します。
- VMSnapShot—このプロパティはオプションで、リリースの配布に使用するスナップショットの名前を取得または設定します。

  VM 構成にスナップショットが指定されていない場合、InstallShield は特定のスナップショットには戻りません。InstallShield は特定のスナップショットに戻さずに VM の電源をオンにして、VM にインストールをコピーします。
- VMMachineUserName—このプロパティは VM 構成のユーザー名を取得または設定します。
- VMMachinePassword—このプロパティは VM 構成のパスワードを取得または設定します。
- VMStageMachineCopyPath—このプロパティは、リリースの配布先となる VM 上の場所を取得または設定します。パスの最後のサブフォルダーとして、InstallShield が VM 上に作成するパスを使用できます。その他のパスは既存していなくてはなりません。
- VMStagePostBuild—このブール型プロパティは、ビルドが成功するたびに選択されたリリースを、InstallShield で自動的に配布するかどうかを示します。

この機能強化は、InstallShield の Premier Edition で使用できます。

詳しくは、次を参照してください:

- ISWiRelease オブジェクト
- ISWiSuiteRelease オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

ISICE 11 に追加された強化機能

InstallShield 内部性合成評価プログラム 11 (ISICE11) が改良されました。プロジェクトに含まれる実行可能ファイルに trustInfo 要素の asm.v2 エントリが含まれている有効なマニフェストが含まれているとき、検証中に ISICE11 が発生することがなくなりました。以前、asm.v3 エントリはチェックされましたが、asm.v2 エントリはチェックされませんでした。

詳細については、「ISICE11」を参照してください。

基本の MSI、DIM、およびマージ モジュールプロジェクトで強化された [ダイアログ] ビュー: 各コントロールのすべての "動作" 設定を単一のグリッドで表示

[ダイアログ] ビューが強化されています。実行時ダイアログ上の各ユーザー インターフェイス コントロールの動作 (イベント、サブスクリプション、および条件) を構成できるすべての設定が、今回より、単一のグリッドで表示されます。以前は、ビューの右下に表示される個別の [イベント]、[サブスクリプション]、および [条件] タブに設定が表示されました。

イベント、サブスクリプション、または条件をコントロールに追加するには、[ダイアログ] ビューで、コントロールを含むダイアログの下にある [動作] ノードを選択します。ダイアログで使用されているコントロールが中央ペインに表示されます。ここで構成するコントロールを選択します。次に、右側ペインに表示される "イベント"、"サブスクリプション"、および "条件" 設定を使って適切な動作を構成します。

この強化機能は、基本の MSI、DIM、およびマージ モジュール プロジェクト タイプで使用できます。

詳細は、次を参照してください:

- 基本の MSI プロジェクトでダイアログ レイアウトを編集する
- 基本の MSI ダイアログでコントロール イベントを使用する

- 基本の MSI ダイアログでサブスクリプションを使用する

強化されたフォルダー ビューで、現在のプロジェクトに含まれる各アイテムの合計数を表示

InstallShield のフォルダー ビューでは、今回より、プロジェクトに含まれるコンテンツの概要を確認することができます。

たとえば、一部のプロジェクト タイプで [ファイルとフォルダー] ビューと [再配布可能ファイル] ビューを含むフォルダービューである [アプリケーション データ] ビューは、現在のプロジェクトに含まれるファイルの数、マージ モジュールの数、および InstallShield 前提条件の数を示します。[ショートカット] や [レジストリ] などのビューを含むフォルダー ビューである [システム構成] ビューは、ショートカットの数、レジストリ キーの数、およびレジストリ値の数といった概要データを表示します。

未使用のコンポーネントを自動的に削除する、新しいマシン全体の設定

InstallShield の [オプション] ダイアログにある [ファイル ビュー] タブに、新しく **[未使用のコンポーネントをクリーンアップする]** チェックボックスが追加されました。このチェック ボックスを使って、InstallShield でプロジェクトから未使用のコンポーネントを自動的に削除するかどうかを指定することができます。

このチェックボックスが選択されているときに、コンポーネントのすべてのファイルを削除して、そのコンポーネントが別の領域で必要でない場合、そのコンポーネントは自動的に削除されます。

この新しいチェックボックスは、マシン全体に影響する設定です。このチェックボックスはデフォルトでクリアの状態です。したがって、デフォルトで未使用のコンポーネントは自動的に削除されません。

詳細については、「[ファイル ビュー] タブ」を参照してください。

Binary テーブルのファイルを使用するカスタム アクションを削除する強化された機能

Binary テーブルのファイルを使用するカスタム アクションをプロジェクトから削除するとき、InstallShield は、Binary テーブルのファイルがプロジェクト内の別のカスタム アクションで参照されているかどうか判別します。別のカスタム アクションで参照されていない場合、InstallShield はカスタム アクションを削除すると同時に、Binary テーブルからエントリを削除するかどうかを指定できるプロンプトを表示します:

- デフォルトの選択である [いいえ] ボタンは、カスタム アクションのみを削除できます。
- [はい] ボタンは、カスタム アクションと Binary テーブル エントリの両方を削除できます。
- [キャンセル] ボタンは、カスタム アクションと Binary テーブル エントリの両方が削除されないようにします。

プロジェクト内の 1 つ以上のアクションが Binary テーブル内のファイルを参照する場合、InstallShield は削除が指定されているカスタム アクションのみを削除します。Binary テーブル エントリは削除されません。このシナリオでは、InstallShield はカスタム アクションを削除するときにプロンプトを表示しません。

以前、前述の状況下でカスタム アクションを削除すると、Binary テーブルのエントリがプロジェクト内の別のエントリで参照されているかどうかにかかわらず、常に Binary テーブルのエントリを削除するかどうかを問い合わせるプロンプトが表示されました。このプロンプトのデフォルト ボタンは [はい] で、カスタム アクションと Binary テーブルのエントリの両方が削除されます。このデフォルトの動作によって、**Binary** テーブルのエントリの削除が誤って選択されることがありました。

この強化機能は、基本の MSI、DIM、InstallScript MSI、マージ モジュール および トランスフォーム プロジェクト タイプに適用します。

新しいスタンドアロン ビルド レジストリ エントリが InstallShield バージョン情報を表示

スタンドビルド インストールは、今回より、InstallShield インストールが HKEY_LOCAL_MACHINE¥SOFTWARE¥InstallShield¥*VersionNumber*¥Professional の下にインストールするのと同じ InstallShield バージョン関連レジストリ値をインストールします。これによって、使用中の環境内のすべての開発マシンとビルド マシンで、製品の同じバージョンを使用しているかどうかを素早く検証することができます。関連するレジストリ値:

- **製品コード**—このレジストリ値は、InstallShield IDE またはスタンドアロン ビルドのどちらがインストールされているかによって異なります。これらのツールのメジャー アップグレードの際に更新されます。
- **MPIndicator**—適切な場合、このレジストリ値はメンテナンスパックを示します。たとえば、*サービス パック 1*。
- **SP**—適切な場合、このレジストリ値はサービス パックの番号を示します (たとえば、*1*)。

InstallShield 2025 および InstallShield 2025 スタンドアロン ビルドの場合、レジストリ値は次のキーの下にインストールされます:

HKEY_LOCAL_MACHINE¥SOFTWARE¥InstallShield¥Automation Interface Version¥Professional

InstallShield 2014 SP1 の新しい機能

InstallShield には、以下のような強化機能が搭載されています。

ポルトガル実行時文字列の更新

ポルトガル語 (ブラジル) および ポルトガル語 (ポルトガル) のデフォルトの実行時文字列は、1990 年のポルトガル語新正書法に基づいて更新されています。この新正書法は、2009 年以来 6 年間の移行期間を経て、2014 年の終りに完全実施されます。

InstallShield 2014 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

新しい .NET Framework 4.5.1 用 InstallShield 前提条件

InstallShield には、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加可能な新しい InstallShield 前提条件が含まれています:

- Microsoft .NET Framework 4.5.1 (完全版)
- Microsoft .NET Framework 4.5.1 (Web インストーラー)

これらの前提条件は、サポートされているターゲット システムに、.NET Framework 4.5.1 をインストールします。

Microsoft SQL Server 2014 サポート

InstallShield に、SQL Server 2014 Database 上で SQL スクリプトを実行できるサポートが追加されました。また、InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストには、SQL Server 2014 が含まれています。

インストールで SQL Server 2014 をターゲットするとき、エンドユーザーがデータベース カタログを参照する選択をしたときに表示される SQLBrowse ランタイム ダイアログで、SQL Server 2014、SQL Server 2014 Express、および SQL Server 2014 Express LocalDB のインスタンスが表示できるようになりました。また、エンドユーザーがデータベース カタログを参照する選択をしたときに表示される SQLBrowse ランタイム ダイアログで、指定された SQL Server 2014 サーバー上のカタログが表示されるようになりました。

このサポートは、基本の MSI、DIM、InstallScript、および InstallScript MSI プロジェクト タイプで利用できます。

Microsoft SQL Server 2014 の前提条件

InstallShield には、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加可能な新しい SQL Server 2014 関連の InstallShield 前提条件が含まれています:

- Microsoft SQL Server 2014 Express RTM (x64)
- Microsoft SQL Server 2014 Express RTM (x86 & x64Wow)
- Microsoft SQL Server 2014 Express RTM (x86)
- Microsoft SQL Server 2014 Express RTM LocalDB (x64)
- Microsoft SQL Server 2014 Express RTM LocalDB (x86)

これらの InstallShield 前提条件は、サポートされているターゲット システムにテクノロジーがインストールされます。

新しい App-V 5.0 SP2 用 InstallShield 前提条件

InstallShield には、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加可能な新しい InstallShield 前提条件が含まれています:

- Microsoft App-V 5.0 SP2 Desktop Client (x86)
- Microsoft App-V 5.0 SP2 Desktop Client (x64)

これらの InstallShield 前提条件用の再配布可能ファイルは Microsoft から入手しなくてはならないため、InstallShield 内部からダウンロードすることはできません。Microsoft から再配布可能ファイルを入手した後、InstallShield 前提条件エディターで前提条件を編集するときに表示される場所に配置してください。

Microsoft Visual C++ 2012 Update 4 用の新しい InstallShield 前提条件

InstallShield には、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加可能な新しい InstallShield 前提条件が含まれています:

- Microsoft Visual C++ 2012 Update 4 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2012 Update 4 再配布可能パッケージ (x64)

これらの前提条件は、サポートされているターゲット システムに様々なテクノロジをインストールします。

アドバンスト UI およびスイート/アドバンスト UI インストールのメンテナンス モードからアップデートをチェックできる機能

アドバンスト UI およびスイート/アドバンスト UI インストールでは、ユーザーがメンテナンス ウィザード ページにある新しい [アップデート] ボタンをクリックして製品のアップデートをチェックできる機能がサポートされています。Web サイトでアップデートが利用可能なとき、エンド ユーザーがそのアップデートの取得を選択した場合、インストールがそれをダウンロードし、デジタル署名を検証してから起動します。

この機能を使用するには、アドバンスト UI およびスイート/アドバンスト UI プロジェクトで自動アップデートがサポートされていなくてはなりません。ビルド時に InstallShield は isupdate.xml という名前のメタ ファイルを作成して、プロジェクトのビルド フォルダー内の UpdateMetadata サブフォルダーに配置します。このメタ ファイルは、アドバンスト UI またはスイート/アドバンスト UI インストールの異なるバージョンとそのダウンロード場所を識別します。製品のアップデートのリリース準備が完了したら、アップデート済みのアドバンスト UI またはスイート/アドバンスト UI インストールと共に、この isupdate.xml ファイルもサイトにアップロードします。

実行時、エンド ユーザーがメンテナンス モードを実行すると、スイート エンジンがメタデータ ファイルのアップデートをチェックします。ダウンロード可能なアップデートがある場合、スイート エンジンが新しい ISUpdateAvailable プロパティを 1 に設定して、[アップデート] ボタンが含まれるメンテナンス ウィザードの新しいページ MaintenanceUpdateWelcome を表示します。また、スイート エンジンは新しい ISUpdateVersion プロパティをダウンロード可能なスイート/アドバンスト UI またはアドバンスト UI インストールのバージョンと等しく設定します。

アップグレードが使用できないとき、スイート エンジンは [アップデート] ボタンが含まれていない標準の MaintenanceWelcome ウィザード ページを表示します。

詳細については、次を参照してください。

- アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート
- アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス

サブフォルダーにサポート ファイルを含める機能

InstallShield では、今回より、サポート ファイル用のカスタム フォルダー構造を指定することができます。[サポート ファイル] ビューの語固有ノードの 1 つまたは言語非依存ノードの 1 つの下にサブフォルダーを追加するには、そのノードを右クリックしてから [新しいフォルダー] をクリックします。サブフォルダーが追加され、必要に応じてその名前を変更することができます。ネストされたフォルダー構造を追加することもできます。ファイルをサブフォルダーに追加するには、右側のペインを選択してから、[ファイルの挿入] をクリックします。実行時、インストールがサポート フォルダーとファイルをターゲット システム上の一時ディレクトリである SUPPORTDIR にコピーして、製品のインストール処理中に使用できるようにします。サポート フォルダーとファイルは、インストールが完了すると削除されます。

強化された [サポート ファイル] ビューは、アドバンスト UI、基本の MSI、スイート/アドバンスト UI、InstallScript、および InstallScript MSI プロジェクト タイプで利用できます。

詳細については、「サポート ファイルを追加する」を参照してください。

オートメーション インターフェイスには、サポートファイルのサブフォルダー機能も含まれています。ISWiSetupFile オブジェクトには、SUPPORTDIR 内でサポートファイルのターゲット ディレクトリのパスを指定または取得するのに使われる新しい読み取り-書き込み Target プロパティが含まれています。詳細については、「ISWiSetupFile オブジェクト」を参照してください。

IIS サーバーおよびクラウドに Web 配置パッケージを配置するためのサポート

スイート/アドバンスト UI インストールには、今回より、IIS サーバーおよびクラウドに Web 配置パッケージを展開するためのビルトイン サポートが含まれています。Web 配置パッケージは、Visual Studio などの IIS または Web アプリケーション開発環境を使って作成できます。

Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加するには、[パッケージ] ビューを使います。[パッケージ] ビューにパッケージを追加および選択すると、右側のペインにパッケージに関連する設定が表示されます。右側のペインの [共通] タブにある [構成] 領域には、Web 配置の構成関連の設定、たとえばインストール先 (多くの場合はリモート サーバー) や資格情報などが含まれています。さらに、[構成] 領域には Web 配置パッケージのパラメーター XML ファイルで定義されるパラメーターの設定も含まれています。デフォルトでは、スイート/アドバンスト UI プロパティが構成設定すべての構成設定に使用され、エンド ユーザーはウィザード インターフェイスまたはコマンドラインを使って実行時にそれらを設定することができます。

Web 配置パッケージをプロジェクトに追加すると、そのパッケージ用の定義済みの Web 配置ウィザード ページがプロジェクトに追加されます。Web 配置ウィザード ページを使って、エンド ユーザーは特定のパッケージをローカル IIS サーバー、リモート サーバー、またはクラウドベースのサーバーのいずれに配置するのかを指定することができます。また、パブリッシャー プロファイル ファイル (.publishsettings) から構成設定をロードできるようにもなります。エンド ユーザーが Web 配置パッケージのパラメーター XML ファイルで定義されているパラメーターを構成できるようにするには、必要に応じてウィザード インターフェイスにコントロールを追加して、それらのコントロールを [パッケージ] ビューの対応するパラメーター設定に指定されているプロパティに関連付けます。

Web 配置パッケージは一般的に、初回インストールと同様、または既存の古いバージョンあるいは同一バージョンを上書きする場合と同様に動作するように設計されています。その結果、Web 配置パッケージではアンインストールを行うことができません。つまり、スイート/アドバンスト UI インストールはこの種類のパッケージのメンテナンスを行うことができません。1 つ以上の Web 配置パッケージ、または 1 つ以上の従来型パッケージ (メンテナンス オプションが含まれている) をプライマリ パッケージとして含むスイート/アドバンスト UI インストールの作成を避けることが推奨されます。

この機能は、InstallShield の Premier Edition で提供されています。

詳細については、次を参照してください。

- Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する
- [パッケージ] ビュー

スイート/アドバンスト UI プロジェクトに InstallShield プロジェクトをパッケージとして含める機能のサポート

新しい InstallShield では、InstallScript インストールを、パッケージとして、スイート/アドバンスト UI またはアドバンスト UI プロジェクトに追加することができます。InstallShield プロジェクトとして、基本の MSI プロジェクトまたは InstallScript プロジェクトのどちらかを使用できますが、これらのプロジェクトはスイート/アドバンスト UI プロジェクトと同じバージョンの InstallShield で保存されている必要があります。

1 つ以上の InstallShield プロジェクト パッケージを含むスイート/アドバンスト UI プロジェクトのリリースをビルドすると、関連する InstallShield プロジェクト内の指定されたリリースが最初にビルドされて、生成されるスイート/アドバンスト UI インストールにそれらがパッケージとして含まれます。InstallShield は基本の MSI プロジェクトおよび InstallScript プロジェクトの出力を .msi パッケージおよび .hdr パッケージとしてスイート/アドバンスト UI インストールに追加します。

[パッケージ] ビューを使って、スイート/アドバンスト UI プロジェクトに InstallShield プロジェクトをパッケージとして追加できます。このビューではまた、スイート/アドバンスト UI プロジェクトにパッケージとして含めるのが、どの製品構成 (基本の MSI プロジェクト パッケージ) か、またはどのリリース (基本の MSI パッケージと InstallScript プロジェクト パッケージの両方) かを指定します。

この機能は、InstallShield の Premier Edition で提供されています。

詳細については、次を参照してください。

- スイート/アドバンスト UI プロジェクトに InstallShield プロジェクト (.ism) をパッケージとして追加する
- [パッケージ] ビュー

スイート/アドバンスト UI インストールにおける InstallScript アクションのサポート

スイート/アドバンスト UI インストールでは、今回より、インストールに含めるパッケージの範囲を超えた様々な実行時タスク行うための InstallScript アクションを起動できるビルトインサポートが提供されています。スイート/アドバンスト UI ベースの InstallScript アクションは、基本の MSI プロジェクトの InstallScript カスタム アクションとほぼ同じ要領で機能します。

スイート/アドバンスト プロジェクトで今回から使用可能な [InstallScript] ビューを使って、プロジェクトに InstallScript ファイル (.rul) を追加し、InstallScript 言語を使って関数を書き込むことができます。スイート/アドバンスト UI アクションで InstallScript 関数を呼び出す場合、基本の MSI プロジェクトで InstallScript カスタム アクションをプロトタイプ化するときと同様に、export キーワードを使って関数をプロトタイプ化します。

InstallScript コードをプロジェクトに追加してから、[イベント] ビューを使ってコードを呼び出す InstallScript を追加します。つぎに、実行時にアクションを起動するタイミングをスケジュールします。たとえば [イベント] ビューを使って、スイート/アドバンスト UI エンジンが管理する 1 つ以上のビルトイン イベントの実行中に InstallScript アクションをスケジュールできます。

スイート/アドバンスト UI プロジェクトで InstallScript を使用するとき、いくつかの制限があります。たとえば、InstallScript UI 関連の関数と InstallScript 実行時パス変数 (PROGRAMFILES および WINDIR) をスイート/アドバンスト UI プロジェクトで使用することはできません。

この機能は、InstallShield の Premier Edition で提供されています。

詳しくは、次を参照してください:

- スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する
- スイート/アドバンスト UI インストールに含まれる InstallScript コードを実行するアクションでの作業について
- スイート/アドバンスト UI アクションの設定を構成する
- スイート/アドバンスト UI アクションをスケジュールする
- スイート/アドバンスト UI およびアドバンスト UI の対話関数。

スイート/アドバンスト UI インストールにおける マネージコード アクションのサポート

スイート/アドバンスト UI インストールでは、今回より、インストールに含めるパッケージの範囲を超えた様々な実行時タスク行うためのマネージコード アクションを起動するビルトイン サポートが提供されています。このアクションの種類は、Visual Basic .NET または C# などのマネージ コードで書かれた、。NET アセンブリのパブリック メソッドを呼び出します。これらのスイート/アドバンスト UI ベースの InstallScript アクションは、ス

イート/アドバンスト UI インストールに含まれる DLL アクションと同じ機能を果たします。つまり、スイートエンジンは呼び出しを行うマネージ関数で 1 つのパラメーターを見つけます。このパラメーターは ISuiteExtension インターフェイスで、System.Object として渡されます。戻り値の型は 32 ビット整数型でなくてはなりません。

マネージコード アクションをスイート/アドバンスト UI プロジェクトに追加するには、[イベント] ビューを使います。

この機能は、InstallShield の Premier Edition で提供されています。

詳細については、次を参照してください。

- スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する
- スイート/アドバンスト UI インストールでマネージ コード アクションを使用する
- スイート/アドバンスト UI アクションの設定を構成する
- スイート/アドバンスト UI アクションをスケジュールする

ウィザード インターフェイス コントロールを使用したときにエンド ユーザーがトリガするスイート/アドバンスト UI イベント アクションを起動できる機能

スイート/アドバンスト UI インストールでは、ウィザード インターフェイス コントロール上でエンド ユーザーがクリックその他の操作をおこなったときに起動される、[イベント] ビューで定義される実行時のアクションがサポートされています。たとえばこの機能を使って、ウィザード インターフェイス上のボタンをエンド ユーザーがクリックしたときに、インストールで実行可能ファイルを実行、InstallScript コードを実行、マネージ アセンブリのマネージ メソッドを呼び出し、PowerShell スクリプトを実行、またはスイート/アドバンスト UI プロパティの設定を行うことができます。

プロジェクトにアクションを追加するには、[イベント] ビューを使います。

ウィザード インターフェイス コントロールがクリックまたは使用されたときに実行するアクションをスケジュールするには、[ウィザード インターフェイス] ビューで構成するコントロールを選択してから、"クリック" 設定または "イベント" 設定の下にあるアクション関連の設定を使って起動するアクションを選択します。

この機能は、InstallShield の Premier Edition で提供されています。

詳細については、次を参照してください。

- ウィザード インターフェイス内の要素のアクションを構成する
- スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する

ウィザード ページとウィンドウが表示/非表示のときにスイート/アドバンスト UI アクションを起動できる機能

スイート/アドバンスト UI インストールは、ウィザード ページまたは 2 番目のウィンドウが開いているまたは閉じているときに起動された実行時アクションをサポートします。この機能によって、コンボボックスの値を動的に取得するなど、インストールで任意の初期設定処理が可能となります。また、実行時にウィザード インターフェイスが表示される前に実行されるアクションをスケジュールすることも可能です。

プロジェクトにアクションを追加するには、[イベント] ビューを使います。

ウィザード ページが開くまたは閉じるときに実行するアクションをスケジュールするには、[ウィザード インターフェイス] ビューで構成するウィザード ページを選択してから "開始ページ" 設定または "終了ページ" 設定で適切なアクションを選択します。

2 番目のウィンドウが開くまたは閉じるときに実行するアクションをスケジュールするには、[ウィザード インターフェイス] ビューで構成する 2 番目のウィンドウを選択してから "開始ウィンドウ" 設定または "終了ウィンドウ" 設定で適切なアクションを選択します。

この機能は、InstallShield の Premier Edition で提供されています。

詳細については、次を参照してください。

- ウィザード インターフェイス内の要素のアクションを構成する
- スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する

アドバンスト UI およびスイート/アドバンスト UI インストールのログ ファイルへのパスワードの書き込みを防ぐ新しいプロパティ

/debuglog パラメーターを使って アドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルを起動すると、スイート エンジンがデバッグ ファイルを生成します。デフォルトで、デバッグ ログ ファイルには、アドバンスト UI またはスイート/アドバンスト UI プロパティの値が含まれます。

場合によって、スイート エンジンが特定のプロパティの値をデバッグ ログ ファイルに書き込むことを防ぐ必要があります。たとえば、パスワードその他の機密情報を含むプロパティのログ記録を防ぐ必要があります。この状況に対応するため、アドバンスト UI およびスイート/アドバンスト UI インストールでは ISHiddenProperties という新しいプロパティがサポートされています。このプロパティに大文字と小文字を区別するプロパティ名のリストをセミコロン区切りで設定して、それらの値をデバッグ ログ ファイルに書き込まないようにできます。

ISHiddenProperties は、プロパティ値を変更するとログ記録される値に限ってログ記録を防ぐのに使用できます。その他の方法でプロパティがログ記録される場合 (ISuiteExtension::LogInfo を使う場合など)、スイート エンジンでログ記録を防ぐことはできません。したがって、ログ ファイルにプロパティ値を書き込むために作成するすべてのコードは、ISHiddenProperties を読み込んで、そのプロパティ値がログ記録されるべきかを判別する必要があります。

詳細については、「アドバンスト UI およびスイート/アドバンスト UI デバッグ ログ ファイルへのプロパティ値の書き込みを防ぐ」を参照してください。

強化されたアドバンスト UI およびスイート/アドバンスト UI プロジェクトの置換構文

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの様々な領域では、プロパティ名を角括弧で囲むことができます。角括弧で囲まれたプロパティ名は、実行時に適切な値で置換されます。実行時の構文では、新しい種類の置換がサポートされています:

- **特殊文字** ― 特殊文字の前に円記号を付けて、結果の文字列を角括弧で囲むと、実行時に特殊文字が解決されます。たとえば、[¥[]Text[¥]] と入力すると [Text] に置換されます。
- **ヌル文字** ― 角括弧で囲まれたチルダ (つまり [~]) は、文字列に埋め込まれたヌル文字として解決します。
- **環境変数** ― 環境変数の名前の前にパーセント記号を付けて、結果の文字列を角括弧で囲むと、実行時に環境変数が解決されます。たとえば、[%PATH] は PATH 環境変数の値として解決します。
- **再帰的なプロパティの解決** ―プロパティの解決をさらに角括弧で囲むと、解決済みのプロパティの値が解決されます。たとえば [[PROPERTY1]] は、PROPERTY1 にその名前が格納されているプロパティの値に解決し、PROPERTY1 が PROPERTY2 を含む場合、解決された値は PROPERTY2 に格納されている値となります。

詳細については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

ファイルとフォルダーの削除に関するサポート

InstallShield では、実行時にターゲット システムから削除するファイルとフォルダーを簡単に指定することができるビルトイン サポートが提供されています。このファイルとフォルダーの削除機能は、アプリケーションによって作成されるファイルの削除など、インストールが追跡を行わない処理に使用すると便利です。

ファイルまたはフォルダーの削除は、次のイベントの 1 つにスケジュールできます:

- ファイルまたはフォルダーのコンポーネントがインストールされるとき
- ファイルまたはフォルダーのコンポーネントがアンインストールされるとき
- ファイルまたはフォルダーのコンポーネントがインストールまたはアンインストールされるとき

削除されるアイテムがフォルダーの場合、そのフォルダーが空の場合のみ削除されます。

InstallShield では、プロジェクトからのファイルまたはフォルダーのを削除を構成するさまざまな方法が提供されています:

- [ファイルとフォルダー] ビューで、削除するファイルまたはフォルダーを含むフォルダーを選択します。次に、**[インストール先コンピューターのファイル]** ペインを右クリックしてから [ファイルの削除を追加] をクリックします。
- [セットアップのデザイン] ビュー (インストール プロジェクト) または [コンポーネント] ビューで、削除するファイルまたはフォルダーが含まれているコンポーネントのノードを拡張してから、[ファイル] サブノードをクリックします。次に、[ファイル] ペインを右クリックしてから [ファイルの削除を追加] をクリックします。

どちらの方法も、使用可能な削除の設定を構成できる [プロパティ] ダイアログ ボックスが表示されます。

以前は、削除するファイルとフォルダーを示す方法は、[ダイレクト エディター] ビューを使って RemoveFile テーブルに手動でエントリを入力する方法のみでした。

この機能は、基本の MSI、DIM、 InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細は、次を参照してください:

- ターゲット システムからファイルとフォルダーを削除する
- ファイル 削除の [プロパティ] ダイアログ ボックス

オートメーション インターフェイスに、ファイルとフォルダーを削除するための新しい ISWiRemoveFile オブジェクトと新しい ISWiRemoveFiles コレクションが追加されました。また、ISWiComponent オブジェクトで新しい 3 つのメソッドが使用できます: AddRemoveFile は現在のコンポーネントにファイルの削除エントリを追加し、RemoveRemoveFile は現在のコンポーネントからファイルの削除エントリを削除し、ISWiRemoveFiles は現在のコンポーネントと関連付けられたファイルの削除エントリすべてを含む ISWiRemoveFiles コレクションを返します。

これらのオートメーション インターフェイスの強化機能は、基本の MSI、InstallScript、InstallScript MSI、およびマージ モジュール プロジェクト タイプで利用できます。

詳細については、次を参照してください。

- ISWiRemoveFile オブジェクト
- ISWiRemoveFiles コレクション
- AddRemoveFile メソッド
- RemoveRemoveFile メソッド

- ISWiRemoveFiles メソッド
- ISWiComponent オブジェクト

DPI 対応インストールのサポート

InstallShield では、今回より、様々な高 DPI ディスプレイ環境のターゲット システムにおいて、実行時のユーザー インターフェイス要素を適切に調整できる DPI 対応インストールを作成して、インストールのユーザー インターフェイスにより統一性のある魅力的な外観を実現できます。使用可能なサポートおよび強化内容は、使用するプロジェクトの種類によって異なります。

アドバンスト UI およびスイート/アドバンスト UI プロジェクト

アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける DPI 対応には、2 つの基本事項があります:

- エンジンが使用するアイテムは、システムの DPI 設定にしたがって調整されます。つまり、ターゲット システムで DPI 200% が設定されている場合、チェックボックス コントロールも同様に調整されます。
- エンジンは、ウィザード インターフェイスに含まれたイメージその他のリソースを表示するとき、スケール因子と言語を考慮します。

たとえば、エンジンがスケール因子を 150 と判断したとき、適切な言語が英語である場合は、ファイル名またはパス名に *scale-150* の文字列を含むリソースを探します。ターゲット システム上で image.png を検索するとき、次にリストされている順番でパスを検索します:

1. [SUPPORTDIR]0409¥**scale-150**¥image.png
2. [SUPPORTDIR]0409¥image.**scale-150**.png
3. [SUPPORTDIR]0409¥image.png
4. [SUPPORTDIR]**scale-150**¥image.png
5. [SUPPORTDIR]image.**scale-150**.png
6. [SUPPORTDIR]¥image.png

正しい DPI よりも一致する言語が優先します。

InstallShield はアドバンスト UI およびスイート/アドバンスト UI プロジェクトに含まれるビルトイン デフォルト イメージに、scale-150 および scale-200 イメージを含みます。プロジェクトにカスタム リソースを含める場合、[サポート ファイル] ビューを使って適切なスケールのイメージを追加します。

InstallScript および InstallScript MSI プロジェクト

InstallScript プロジェクトと InstallScript MSI プロジェクトにおける DPI 対応には、次の点が含まれています:

- ビルトイン デフォルト ダイアログのすべてのイメージは、200% に上手く調整可能なイメージに差し替えられています。
- 今回より、コントロール ID 1200 を持つスタティック ダイアログ コントロールは、透明色の使用が可能な .png イメージをサポートします。これは、ウェルカム ダイアログのような、外部スキン ダイアログの左側にデフォルトで表示されるコンピューター イメージに適用します。内部ダイアログのバナー イメージにも適用します。

  さらに、これらの同じコントロールは、各イメージとそれに関連するスケール因子を識別するための新しいフォーマットをサポートします。

@@ResourceID;ScaleFactor

ResourceID は、.png イメージ (リソース タイプ PNG として格納される) またはビットマップ イメージ (リソース タイプ RT_BITMAP として格納される) のいずれかを検出するためのリソース識別番号を示します。*ScaleFactor* は、イメージが意図されている DPI スケール率を示します。

たとえば、100% (96 DPI)、125% (120 DPI)、150% (144 DPI)、200% (192 DPI) となります。イメージに指定されているスケール因子が 200 の場合、200% DPI スケール以下で実行中のターゲット システム上で、イメージは縮小表示されます。200% ターゲットシステム上で、これは 1:1 の割合で表示されます。イメージに指定されているスケール因子が 100 の場合、200% DPI スケールで実行中のターゲット システム上で、イメージは拡大表示されます。

ビットマップ イメージ (.bmp) を識別する以前のフォーマットも使用できますが、スケーリングまたは .png イメージがサポートされていません:

@BitmapResourceID;TransparentFlag;3-DFlag;<unused>;TransparentColorKey

- ダイアログ ヘッダーのバナーおよび外部パネルのアイコン イメージ用のすべての ID 1200 スタティック コントロールには、今回より新しいイメージ文字列フォーマットが使用されます。InstallShield プロジェクトまたは InstallScript MSI プロジェクトを InstallShield 2025 にアップグレードするとき、以前のバージョンの InstallShield でダイアログを編集した場合、プロジェクトでサポートされているすべての言語のイメージ文字列を新しいフォーマットに編集する必要があります。
- InstallScript 関数 **SdOptionsButtons** は、イメージを識別する新しい文字列フォーマットをサポートするようアップデートされています:

  @@ResourceID;ScaleFactor

  以前は、次のフォーマットが使用されていました:

  @BitmapResourceID;BitmapIconFlag;TransparentColor

- InstallScript 関数 **GetSystemInfo** の nItem パラメーターは、2 つの新しい定数をサポートします:
  - **SYSTEM_DPI** — nvResult パラメーターのシステム DPI 値を返します。
  - **SYSTEM_DPI_SCALING** — nvResult パラメーターの DPI スケール値 を返します。

詳細は、次を参照してください:

- GetSystemInfo
- SdOptionsButtons
- InstallScript と InstallScript MSI プロジェクトのダイアログに含まれるデフォルト イメージについての背景情報

基本の MSI プロジェクト

Setup.exe 起動プログラムが実行時に表示するダイアログに表示されるビルトイン イメージが、より大きな領域に上手く調整されるイメージに差し替えられています。これには、初期設定ダイアログに表示されるイメージも含まれています。

ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにインストールを配布できる機能

インストールのビルドが成功するたびに InstallShieldが仮想マシン (VM) を指定のスナップショットに戻し、VM の電源をオンにしてインストールを VM にコピーし、テストが可能な状態になるようプロジェクトを構成できます。また、これらのテスト準備を必要なときにオンデマンドで行うこともできます。このテスト準備機能により、手作業で行う処理を減らし、テストにかかる時間を短縮できます。

VM は Microsoft Hyper-V Server、VMware ESX または ESXi Server、または VMware Workstation を指定できます。

VM 情報を指定するには、[リリース] ビューにあるリリースの [イベント] タブにある [仮想マシン] 領域を使用します。設定のうちの 1 つを使って、ビルドが成功するたびに指定の VM にリリースを配布するかどうかを指定します。

必要なときにリリースを VM に配布するには、[リリース] ビューで該当するリリースを右クリックしてから、[VM に配布] をクリックします。

[イベント] タブにある "構成" 設定を使用して VM の詳細を構成すると、InstallShield によって 指定されたデータが VMConfigurations.xml という名前のファイルに書き込まれます。この VMConfigurations.xml ファイルは、マシン全体に適用するファイルで、一度構成を行うと他の InstallShield プロジェクトでも使用できます。また、他のチームメンバーと共有することも可能です。また、このファイルを Standalone Build で使用することも可能です。

サポートされている VM の1 つに配布する場合、InstallShield では必要な仮想化テクノロジとの対話が必要です。VMWare 仮想化テクノロジ (VMware ESX、ESXi Server oまたは VMware Workstation) を使用する場合、InstallShield と同じマシン上に VMware VIX API が インストールされていなくてはなりません。マシン上に VMware Workstation をインストールするか、http://www.vmware.com/support/developer/vix-api から VMware VIX API をダウンロードおよびインストールすることができます。

VMware Workstation を使用する場合、InstallShield と同じマシン上に VMware Workstation をインストールすることが推奨されます。これによって、InstallShield が特定の VMware Workstation バージョン用に設計された VIX API のバージョンを使用します。VIX API の新しいバージョンも使用できる場合がありますが、現在使っている VMware Workstation にバンドルされている VIX API のバージョンを InstallShield で使用する方法が最も推奨されます。

この機能は、基本の MSI、InstallScript、InstallScript MSI、および スイート/アドバンスト UI プロジェクト タイプで使用できます。

この機能は、InstallShield の Premier Edition で提供されています。

詳しくは、次を参照してください:

- ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する
- リリースの [イベント] タブ
- [仮想マシンの構成を編集] ダイアログ ボックス
- リリースの配布用の仮想マシン設定を共有する

強化機能

InstallShield には、以下のような強化機能が搭載されています。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトで、コンボボックスおよびリストボックス コントロールの設定におけるユーザビリティーの向上

InstallShield では、コンボボックス コントロールとリストボックス コントロールをアドバンスト UI およびスイート/アドバンスト UI プロジェクトのウィザード ページおよび 2 番目のウィンドウに追加することができます。これらのどちらかの種類のコントロールを作成するプロセスでは、2 つのプロパティ (コントロールで表示するオプションのリストを定義するプロパティ、および実行時にエンド ユーザーが選択するオプションを格納するためにインストールが使用するプロパティ) を使用します。これまでは [ウィザード インターフェイス] ビューと [プロパティ マネージャー] ビューを使用する必要がありましたが、これらのプロパティの両方を InstallShield の [ウィザード インターフェイス] ビュー内部から構成することができます。

[ウィザード インターフェイス] ビューで、コンボボックスまたはリストボックス コントロールの "コンテンツ プロパティ" 設定には省略記号ボタン (...) が含まれていて、これをクリックしてコントロールに表示するオプションのリストを定義することができます。このボタンをクリックすると、シンプルな [リスト オプションを編集] ダイアログ ボックスが開いて、選択されたコントロールに表示するオプションのリストおよび関連する値を指定できます。

以前は、[プロパティ マネージャー] ビューを使ってオプションのリストが含まれるプロパティおよび対応するプロパティ値を定義する必要がありました。プロパティを定義するのに必要なフォーマット (例、ID_Option1¥rValue1¥nID_Option2¥rValue2¥nID_Option3¥rValue3) は覚えにくく、間違いやすいものでした。

詳細は、次を参照してください:

- ウィザード インターフェイスのコンボ ボックスおよびリスト ボックス コントロールを設定する
- [オプション リストの編集] ダイアログ ボックス

アドバンスト UI およびスイート/アドバンスト UI インストールでファイルまたはフォルダーの参照における実行時の機能強化

アドバンスト UI およびスイート/アドバンスト UI インストールでは、今回より、エンド ユーザーがファイルまたはフォルダーを参照できる機能が強化されています:

- ウィザード ページと 2 番目のウィンドウのコントロールで使用できる [新しいファイルを参照] アクションが追加されました。このアクションをボタンなどのコントロールの Click イベントに追加すると、エンド ユーザーがそのボタンをクリックしたときに参照ダイアログ ボックスが開き、新しいファイルとその場所を指定することができます。

  この種類のアクションをプロジェクト内のコントロールに追加するとき、ダイアログ ボックスのタイトル バーにテキストを指定できます。また、選択可能なファイル拡張子のフィルター オプションや、ファイル作成時に使用するデフォルトのファイル拡張子を指定することもできます。

- 既存の [フォルダーの参照] アクションが強化されました。このアクションが起動する [フォルダーの参照] ダイアログ ボックスは、対応するファイル システムを持つ場所のみに使用することができます。つまり、エンド ユーザーが [この PC] または [マイコンピューター] といった場所を選択した場合、このダイアログ ボックスの [OK] ボタンは無効化されます。以前、エンド ユーザーがこれらの場所を選択することが可能で、その場合に関連プロパティの値がクリアされました。

  この種類のアクションをプロジェクトに追加するとき、この [フォルダーの参照] ダイアログ ボックスに表示する支持を指定することができます。

- 既存の [ファイルの参照] アクションが強化され、[ファイルの参照] ダイアログ ボックスで選択可能なファイル拡張子のフィルター オプションを指定できるようになりました。

詳細については、「ウィザード インターフェイス内の要素のアクションを構成する」を参照してください。

実行時に LicenseAgreement ダイアログで [印刷] の選択肢を提供できる機能

LicenseAgreement ダイアログの [印刷] ボタンの機能が強化されました。エンド ユーザーが [印刷] ボタンをクリックしたときにデフォルト プリンターに直接印刷する代わりに、今回より、印刷選択ダイアログ ボックスが開きます。

この強化は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで使用できます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、以前からこの機能の使用が可能です。

テキスト ファイルにテキストを挿入できる機能

[テキスト ファイルの変更] ビューでは、ターゲット システム上で実行時にテキスト ファイルに行う変更を構成することができます。テキスト ファイルは、インストールの一部またはシステム上に既存するファイルのどちらでも構いません。[テキスト ファイルの変更] ビューには、1 つ以上のファイルの最初または最後にテキストを挿入できる機能が追加されています。また今回より、指定したテキストの前後にテキストを挿入することもできます。このビューでテキスト ファイルの変更を構成するとき、既存テキストを置換するのか、既存テキストの前後にテキストを挿入するのか、またはファイルの最初または最後にテキストを挿入するのかを指定するための新しい "アクション" 設定を使用します。

以前、[テキスト ファイルの変更] ビューでは、実行時にテキスト ファイル内の既存テキストを置換するためのサポートが含まれていました。

この強化機能は、基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- テキスト ファイルの変更
- [テキスト ファイルの変更] ビュー

エンド ユーザーがメジャー アップグレードを適用する製品の以前のバージョンのインストール済みインスタンスを選択できる機能をサポート

基本の MSI プロジェクトにおける複数インスタンス サポートは、今回より、メジャー アップグレードのサポートを含みます。実行時、Setup.exe セットアップ ランチャーが表示するインスタンスの選択ダイアログを通して、エンドユーザーは製品の現在インストール済みのインスタンスのすべてをリストで参照できます。リストには、インストール済みの製品の異なるメジャーバージョンも含まれます。エンド ユーザーは、今回より、この強化されたインスタンス ダイアログを使用して新しいインスタンスのインストール、既存インスタンスの保持、またはインスタンスのメジャー アップグレードの適用を行うことができます。

以前、インスタンスの選択ダイアログには、実行中の複数インスタンス インストールと同じ製品コードを持つインスタンスのみがリストされました。

詳しくは、次を参照してください:

- 複数インスタンスのサポートを含むメジャー アップグレードの構成とビルド
- インスタンスのプロパティを設定します
- 製品のインスタンスを複数インストールするときの実行時の動作

追加プラットフォームで Microsoft App-V 5x パッケージを作成できる機能

InstallShield の Microsoft App-V アシスタントには、Microsoft App-V 5 フォーマットで仮想アプリケーションを作成するためのサポートが含まれています。このアシスタントの [パッケージ情報] ビューを使って、App-V 5 または 4.x のどちらをターゲットにするかを指定できます。InstallShield がサポートされている任意の Windows バージョンで、App-V 5x パッケージをビルドすることができます。以前は Windows 8 または Windows Server 2012 が必要でした。

詳細については、次を参照してください:

- App-V パッケージのコンポーネント
- App-V パッケージのビルド出力

基本の MSI インストールの [ファイルを開く] 実行時ダイアログの機能強化

InstallShield は、基本の MSI インストールのダイアログの 1 つから [ファイルを開く] ダイアログを起動するサポートを含みます。エンド ユーザーがダイアログの 1 つから [参照] ボタンをクリックすると、[ファイルを開く] ダイアログが起動します。[ファイルを開く] ダイアログを使って、エンド ユーザーはファイルを参照できます。今回よりオプションで、新しいプロパティ IS_BROWSE_INITIALDIR を使用して、実行時にこのダイアログに表示するデフォルト パスを指定できます。以前、ダイアログにはエンド ユーザーが最後に参照したパスが表示されました。

詳細については、「[ファイルを開く] ダイアログを起動する」を参照してください。

新しい FlexNet Connect 13.06 再配布可能ファイル

InstallShield は、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで FlexNet Connect 13.06 をサポートします。InstallShield の [アップデート通知] ビューで、2 つの FlexNet Connect 13.06 マージ モジュール (Common Software Manager が含まれているマージ モジュールと、含まれていないマージ モジュール) のどちらかを含みます。

InstallShield 2013 SP1 の新しい機能

InstallShield 2013 Service Pack 1 (SP1) には、Windows 8.1、Windows Server 2012 R2、および Visual Studio 2013 をサポートするための変更が含まれています。

Windows 8.1 および Windows Server 2012 R2 システムをターゲットにできる機能

InstallShield では、インストールに Windows 8.1 または Windows Server 2012 R2 が必要であることを指定できます。また、これらのオペレーティング システムに対する機能条件およびコンポーネント条件をビルドすることができます。

Windows 8.1 および Windows Server 2012 R2 にインストール可能な InstallShield 前提条件は、必要に応じて、これらのシステムにインストールされるように更新されています。以前これらのシステムでは、前提条件がデフォルトでは実行されませんでした。これは、次の InstallShield 前提条件に適用します:

- FSharp Redistributable Package 2.0
- JRE_SE 1.7.0_02 (x64)
- JRE_SE 1.7.0_02 (x86)
- Microsoft .NET Framework 3.0 OS Component
- Microsoft .NET Framework 3.5 SP1 (Windows 機能)
- Microsoft .NET Framework 4.5 Full
- Microsoft .NET Framework 4.5 Web
- Microsoft App-V 5.0 SP1 Desktop Client (x64)
- Microsoft App-V 5.0 SP1 Desktop Client (x86)
- Microsoft ReportViewer 2010
- Microsoft SQL CE 3.5 SP2
- Microsoft SQL Server 2005 Express SP3 (x86 & x64Wow)

- Microsoft SQL Server 2005 Express SP3 (x86)
- Microsoft SQL Server 2008 Express SP1 (x64)
- Microsoft SQL Server 2008 Express SP1 (x86 & x64Wow)
- Microsoft SQL Server 2008 Express SP1 (x86)
- Microsoft SQL Server 2008 Management Objects 10.00.2531 (IA64)
- Microsoft SQL Server 2008 Management Objects 10.00.2531 (x64)
- Microsoft SQL Server 2008 Management Objects 10.00.2531 (x86)
- Microsoft SQL Server 2008 Native Client 10.00.2531 (IA64)
- Microsoft SQL Server 2008 Native Client 10.00.2531 (x64)
- Microsoft SQL Server 2008 Native Client 10.00.2531 (x86)
- Microsoft SQL Server 2008 R2 Express RTM (x64)
- Microsoft SQL Server 2008 R2 Express RTM (x86 & x64Wow)
- Microsoft SQL Server 2008 R2 Express RTM (x86)
- Microsoft SQL Server 2008 R2 Express SP2 (x64)
- Microsoft SQL Server 2008 R2 Express SP2 (x86 & x64Wow)
- Microsoft SQL Server 2008 R2 Express SP2 (x86)
- Microsoft SQL Server 2008 R2 Native Client 10.50.1600.1 (IA64)
- Microsoft SQL Server 2008 R2 Native Client 10.50.1600.1 (x64)
- Microsoft SQL Server 2008 R2 Native Client 10.50.1600.1 (x86)
- Microsoft SQL Server 2012 Express LocalDB RTM (x64)
- Microsoft SQL Server 2012 Express LocalDB RTM (x86)
- Microsoft SQL Server 2012 Express RTM (x64)
- Microsoft SQL Server 2012 Express RTM (x86 & x64Wow)
- Microsoft SQL Server 2012 Express RTM (x86)
- Microsoft SQL Server 2012 Native Client (x64)
- Microsoft SQL Server 2012 Native Client (x86)
- Microsoft SQL Server Compact 4.0 (x64)
- Microsoft SQL Server Compact 4.0 (x86)
- Microsoft SQL Server Native Client 9.00.4035 (IA64)
- Microsoft SQL Server Native Client 9.00.4035 (x64)
- Microsoft SQL Server Native Client 9.00.4035 (x86)
- Microsoft SQL Server System CLR Types 10.00.2531 (IA64)
- Microsoft SQL Server System CLR Types 10.00.2531 (x64)

- Microsoft SQL Server System CLR Types 10.00.2531 (x86)
- Microsoft Visual C++ 2005 SP1 再配布可能パッケージ MFC のセキュリティ更新プログラム KB2538242(x64)
- Microsoft Visual C++ 2005 SP1 再配布可能パッケージ MFC のセキュリティ更新プログラム KB2538242(x86)
- Microsoft Visual C++ 2005 SP1 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2005 SP1 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2008 SP1 再配布可能パッケージ MFC のセキュリティ更新プログラム KB2538243(x64)
- Microsoft Visual C++ 2008 SP1 再配布可能パッケージ MFC のセキュリティ更新プログラム KB2538243(x86)
- Microsoft Visual C++ 2008 SP1 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2008 SP1 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2010 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2010 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2010 RTM 再配布可能パッケージ MFC のセキュリティ更新プログラム KB2467173 (x64)
- Microsoft Visual C++ 2010 RTM 再配布可能パッケージ MFC のセキュリティ更新プログラム KB2467173 (x86)
- Microsoft Visual C++ 2010 SP1 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2010 SP1 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2012 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2012 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2012 Update 1 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2012 Update 1 再配布可能パッケージ (x86)
- Microsoft VSTO 2010 Runtime (x64)
- Microsoft VSTO 2010 Runtime

Windows 8.1 および Windows Server 2012 R2 用の InstallScript 言語の強化内容

次の構造メンバーと定義済み定数が InstallScript 言語に追加されました:

- SYSINFO.WINNT.bWin81 – 新しい SYSINFO 構造メンバーです。オペレーティング システムが Windows 8.1 または Windows Server 2012 R2 の場合、この値は TRUE です。
- ISOSL_WIN81 – **FeatureFilterOS** 関数と SYSINFO 構造変数と共に使用できる新しい定義済み定数です。これは、ターゲット システムが Windows 8.1 または Windows Server 2012 R2 を実行中であることを示します。

詳細については、「FeatureFilterOS 関数と SYSINFO 構造変数」を参照してください。

オートメーション インターフェイスの強化機能: Windows 8.1 と Windows Server 2012 R2 用の OSFilter プロパティの値

オートメーション インターフェイスで、ISWiComponent と ISWiRelease オブジェクトの OSFilter メンバーと共に、以下の定数を使用できます:

eosWin81 = &H8000000 (134217728)—これらの定数は Windows 8.1 および Windows Server 2012 R2 用です。

また、eosAll 定数の値が &7D100D0 (131137744) から &HFD100D0 (265355472) に変更されました。

OSFilter メンバーは、InstallScript、および InstallScript MSI プロジェクトで ISWiComponent オブジェクトに適用します。OSFilter メンバーは、InstallScript プロジェクトで ISWiReleaseISWiRelease オブジェクトに適用します。

詳細は、次を参照してください:

- ISWiComponent オブジェクト
- ISWiRelease オブジェクト

Microsoft Visual Studio 2013 のサポート

InstallShield は、Visual Studio 2013 をサポートします。このバージョンの Visual Studio 内部から InstallShield プロジェクトを作成できます。

新しい Microsoft SQL Server 2012 Express SP1 用の InstallShield 前提条件

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる、いくつかの新しい SQL Server 関連の InstallShield 前提条件が含まれています:

- Microsoft SQL Server 2012 Express SP1 LocalDB (x64)
- Microsoft SQL Server 2012 Express SP1 LocalDB (x86)
- Microsoft SQL Server 2012 Express SP1 (x64)
- Microsoft SQL Server 2012 Express SP1 (x86 & x64Wow)
- Microsoft SQL Server 2012 Express SP1 (x86)
- Microsoft SQL Server 2012 Express SP1 Management Objects (x64)
- Microsoft SQL Server 2012 Express SP1 Management Objects (x86)
- Microsoft SQL Server 2012 Express SP1 System CLR Types (x64)
- Microsoft SQL Server 2012 Express SP1 System CLR Types (x86)

これらの InstallShield 前提条件は、サポートされているターゲット システムにテクノロジーがインストールされます。

スイート/アドバンスト UI およびアドバンスト UI インストールで、ウィザード ページおよびウィンドウのコントロールに使用するカスタム テーマを定義できるサポート

スイート/アドバンスト UI およびアドバンスト UI プロジェクトでは、今回より、ウィザード インターフェイスのコントロールに使用するテーマを定義できます。コントロールのテーマは、コントロールの様々な部分の色 (テキストの色、背景の色、および境界線の色) や、クリック済みまたはホバー状態といった、コントロールの様々な状態に使用する色を集めたものです。デフォルトで、ユーザー インターフェイスのコントロールに使用するコントロールのテーマを指定できます。ユーザー インターフェイス内のウィザード ページやウィンドウに含まれる特定のコントロールのテーマをオーバーライドすることもできます。コントロールのテーマを使用すると、インストールのユーザー インターフェイス内のその他のアイテムが 3D 効果なしで平面的に表示されます。これは Windows Store アプリケーションのスタイルに似ています。

次の種類のコントロールで、コントロールのテーマを使用できます:

- ボタン (ナビゲーション ボタンを含む)

- チェック ボックス
- ラジオ ボタン
- コマンド リンク

プロジェクトにコントロールのテーマを追加するには、[ウィザード インターフェイス] ビューで [コントロールのテーマ] ノードを右クリックしてから [コントロールのテーマを追加] を選択します。必要に応じて、テーマの設定を構成します。

ユーザー インターフェイスのコントロールに、デフォルトで使用するテーマを指定するには、[ウィザード ページ] ノードが選択されているときに [ウィザード インターフェイス] ビューに表示される、新しい "デフォルト コントロールのテーマ" 設定で適切なテーマを選択します。デフォルトで、この設定は空白です。

特定のコントロールについて、コントロールのテーマをオーバーライドするには、そのコントロールの "コントロールのテーマ" 設定で適切なテーマを選択します。

詳細については、次を参照してください:

- スタイル、ブラシ、コントロール テーマ、および最小ウィザード ページを使ってウィザード インターフェイスをカスタマイズする
- カスタム コントロール テーマをウィザード インターフェイスに定義する
- テーマをウィザード インターフェイスのコントロールに適用する

新しい、Visual Studio ソリューション フォルダーの定義済みパス変数

ハイレベルなベース ディレクトリを参照する、VSSolutionFolder と呼ばれる新しい定義済みパス変数をプロジェクトで使用できます。このサポートを使うと、InstallShield プロジェクトで Visual Studio ソリューション フォルダー内にある姉妹プロジェクトのファイルへのスタティック リンクを含めることができます。異なるマシン上のプロジェクトで作業を行う場合、VSSolutionFolder パス変数を使用するスタティック リンクは、姉妹プロジェクトのファイルへの正しいパスを参照することができます。

VSSolutionFolder パス変数は、InstallShield が Visual Studio ソリューション内で開かれたときに自動的に定義されます。また、MSBuild を使って InstallShieldプロジェクトを含むソリューションをビルドするときにも、自動的に定義されます。ただし、Visual Studio ソリューションなしで InstallShieldプロジェクトを開いた場合、VSSolutionFolder が自動的に定義されることはありません。たとえば、InstallShieldプロジェクトを、Visual Studio を開かずに InstallShieldインターフェイスで直接開いた場合、VSSolutionFolder は定義されません。同様に、コマンドライン ツール IsCmdBld.exe や、MSBuild で .isproj ファイルを使用する場合、VSSolutionFolder は定義されません。IsCmdBld.exe を使って InstallShield プロジェクトのリリースをビルドするには、-L コマンドライン パラメーターを使って、VSSolutionFolder の値を設定します。MSBuild を使う場合は、PathVariables パラメーターを使って VSSolutionFolder の値を設定します。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPathVariableOverrides として公開されます。

InstallShieldプロジェクトで VSSolutionFolder パス変数を含むパスを持つソース ファイルを含み、それを VSSolutionFolder パス変数がサポートされていない環境でビルドすると、次のようなビルド エラーが発生する可能性があります:

- -6103: ファイル <VSSolutionFolder>¥MyFile.exe が見つかりません
- -6271: ファイル <VSSolutionFolder>¥MyFile.exe が見つかりませんでした。このファイルの MsiFileHash テーブルをビルド中にエラーが発生しました。指定した場所にファイルが存在することを確認します。

この機能は、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、およびスイート/アドバンスト UI プロジェクト タイプで使用できます。

詳細は、次を参照してください:

- Visual Studio ソリューションで VSSolutionFolder パス 変数を使用する
- 定義済みパス変数
- Microsoft ビルド エンジン (MSBuild)
- ISCmdBld.exe

複数インスタンス プロジェクトでインスタンスの製品コードを簡単に更新できる機能

複数インスタンス サポートを含むプロジェクトのメジャー アップグレードを作成している場合、各インスタンスの製品コードを更新する必要があります。InstallShield では、今回より、プロジェクトで定義されている特定インスタンスの製品コードを簡単に更新することができます。InstallShield ではまた、プロジェクト内の製品構成で定義されているすべてのインスタンスの製品コードを簡単に更新することもできます。

特定インスタンスの製品コードを更新するには、[リリース] ビューの製品構成で [複数インスタンス] タブで定義されているインスタンスの製品コード設定の値を右クリックすると使用できる、[新しい GUID を生成する] コマンドを使用します。

プロジェクトに含まれる製品構成で定義されている各インスタンスについて、その製品コードを更新するには、[リリース] ビューにある製品構成で、[複数インスタンス] タブにある [インスタンス] ノードを右クリックしたときに使用できる、新しい [各インスタンスの ProductCode を変更する] コマンドを使用します。

この機能は、基本の MSI プロジェクトで使用できます。

詳細については、「1 つ以上のインスタンスの製品コードを更新する」を参照してください。

InstallShield 2013 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

スイート/アドバンスト UI インストールの動作を拡張する実行時アクションを作成できる機能

スイート/アドバンスト UI インストールで、インストールに含めるパッケージの範囲を超えた様々な実行時タスクが必要な場合があります。たとえば、スイート/アドバンスト UI インストールで、以下の 1 つ以上の操作が必要な状況が考えられます。

- スイート/アドバンスト UI インストールのユーザー インターフェイスが表示される前にアプリケーションをインストールする。

  この状況が必要な例として、スイート/アドバンスト UI インストール内のパッケージの 1 つが Oracle データベースをインストールする SQL スクリプトを実行する時があります。このパッケージを実行する前に、スイート/アドバンスト UI インターフェイスで、エンド ユーザーがネットワーク上で使用可能なサーバーの一覧から適切なサーバーを選択できるウィザード ページを表示したい場合があります。この種類の UI サポートを使用するには、ターゲット システム上に ODBC ドライバーがインストールされていることが必要です。

- 特定の製品、テクノロジ、フォルダー、ファイル、レジストリ エントリ、その他のアイテムの存在の有無をターゲット システム上で検索する。
- スイート/アドバンスト UI インストール内のパッケージを実行する前後にターゲット システムを構成する。

スイート/アドバンスト UI インストールの機能を拡張して、前述のタスクその他を実行できるようにするには、スイート/アドバンスト UI プロジェクトに新しく追加された [イベント] ビューを使って、実行時に実行可能ファイルの実行、DLL 関数の呼び出し、PowerShell スクリプトの実行、またはスイート/アドバンスト UI プロパティの設定を行うアクションを作成できます。

プロジェクトにアクションを追加するとき、それを実行時に起動するタイミングをスケジュールできます。

- [イベント] ビューにある [イベント] エクスプローラーを使って、スイート/アドバンスト UI が管理する 1 つ以上のビルトイン イベントの実行中にアクションをスケジュールできます。ビュー内のこの領域には、各イベント中に発生するイベントとアクションが時系列で上から下にリストされます。
- [パッケージ] ビューの [イベント] 領域にある設定を使って、スイート/アドバンスト UI エンジンのインストール、削除、変更、またはパッケージの修復の前後に実行するアクションをスケジュールします。

アクションをスケジュールするとき、アクションを実行するかどうかを制御する条件ステートメントをビルドできます。新しく追加された、次の 2 種類の条件チェック以外にも、スイート/アドバンスト UI プロジェクトの別の領域で、同じ種類の条件チェックを利用できます:

- **パッケージの操作**—スイート/アドバンスト UI インストール内の特定のパケージの実行状態をチェックします (例、インストールまたは削除)。
- **機能の操作**—スイート/アドバンスト UI インストール内の特定の機能の実行状態をチェックします (例、インストールまたは削除)。

これらの条件の種類は、[イベント] ビューのイベント、または [パッケージ] ビューのパッケージに関連付けられたアクションでのみ使用可能です。

PowerShell アクションをサポートするには、ターゲット システム上に PowerShell 2.0 以降が必要です。

この機能は、InstallShield Premier Edition で提供されています。

詳細については、次を参照してください。

- スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する
- スイート/アドバンスト UI インストールに含まれるアクションの .exe ファイルでの作業について
- スイート/アドバンスト UI インストールに含まれるアクションの DLL ファイルでの作業について
- スイート/アドバンスト UI インストールに含まれるアクションの PowerShell スクリプトでの作業について
- アクションをスイート/アドバンスト UI プロジェクトに追加する
- スイート/アドバンスト UI アクションの設定を構成する
- スイート/アドバンスト UI アクションをスケジュールする
- スイート/アドバンスト UI インストールにおけるイベントの種類
- [イベント] ビュー
- [パッケージ] ビュー
- "パッケージの操作" 条件の設定
- "機能の操作" 条件の設定

スイート/アドバンスト UI インストール中に Windows の役割と機能を有効化できるサポート

スイート/アドバンスト UI インストール内の特定のパッケージが、ターゲット システム上で 1 つ以上の Windows の役割と機能が有効化されていることを必要とする場合、スイート/アドバンスト プロジェクトでパッケージを構成中に、[パッケージ] ビューに追加された新しい "Windows 機能" 設定を使って、この要件を指定できます。実行時、インストールされる対象のパッケージが無効化されている 1 つ以上の Windows の役割と機能を必要とする場合、スイート/アドバンスト UI インストールは、パッケージを起動する前にこれらの役割と機能を有効化します。

たとえば、スイート/アドバンスト UI プロジェクトで Windows のインターネット インフォメーション サービス機能が有効であることを必要とする IIS Web サイトをインストールパッケージがあります。同じプロジェクト内の別のパッケージでは、PowerShell 機能が有効であることが必要です。プロジェクトのこれらのパッケージの設定を構成するとき、必要な Windows 機能を指定します。実行時、パッケージがターゲット システム上でインストールの対象となっているが、必要な Windows 機能のいずれかが無効である場合、スイート/アドバンスト UI インストールは、起動前にこれらの無効化されている Windows 機能を有効にします。パッケージがインストールの対象ではない場合、必要な Windows 機能は有効化されません。

InstallShield には、いくつかの Windows 機能のビルトインサポートが搭載されています:

- インターネット インフォメーション サービス
- PowerShell
- .NET Framework 3.x

InstallShield ではまた、パッケージに必要な上記以外の Windows 役割と機能を指定することもできます。

この機能は、InstallShield Premier Edition で提供されています。

詳しくは、次を参照してください:

- スイート/アドバンスト UI インストール中に Windows 役割と機能を有効化する
- [パッケージ] ビュー

アドバンスト UI およびスイート/アドバンスト UI プロジェクト テンプレートを作成できる機能

InstallShield では、アドバンスト UI およびスイート/アドバンスト UI プロジェクトをテンプレートとして使用する機能がサポートされています。プロジェクト テンプレートは、新しいアドバンスト UI またはスイート/アドバンスト UI プロジェクトの作成を開始するときに使用できる、デフォルトの設定とデザイン要素を含みます。

任意のアドバンスト UI またはスイート アドバンスト UI プロジェクトをテンプレートに指定できます。プロジェクト ファイルとテンプレートファイルは、同じファイル拡張子 (.issuite) を持ちます。

.issuite プロジェクト ファイルがテンプレート ファイルであり、[新規プロジェクト] ダイアログ ボックスでこのテンプレートの選択を可能にするためには、[新規プロジェクト] ダイアログ ボックスの [すべてのタイプ] タブ内で右クリックして [新規テンプレートの追加] を選択します。テンプレートとして使用する .issuite ファイルを参照して、[すべてのタイプ] タブに新しく追加したテンプレートのアイコンが表示されます。

.issuite ファイルをテンプレートとして新しいアドバンスト UI またはスイート/アドバンスト UI プロジェクトを作成するには、新しいプロジェクトを作成する際に [新規プロジェクト] ダイアログ ボックスの [すべてのタイプ] タブでテンプレートのアイコンを選択します。

手順については、次を参照してください:

- プロジェクト テンプレートの作成
- テンプレートを使用した新規プロジェクトの作成

多層アプリケーションをインストールするためのインストール作成用の新しいビルトイン テンプレート

InstallShield には、クラウド ベースの多層アプリケーション向けのスイート/アドバンスト UI インストールの作成に役立つ、新しい多層アプリケーション用のテンプレートが含まれています。エンド ユーザーが Web サイト、データベース、およびアプリケーションをインストールするインストール階層を簡単に管理できる、この種類のインストールを作成できます。エンド ユーザーは、スイート/アドバンスト UI インストールのウィザード ページを使って、インストールしたい層を選択するか、コマンドライン サポートを使って、適切な機能を選択できまうす。

新しい多層アプリケーション テンプレートには、「機能」として表示される 3 つの層があります。これらの機能をカスタマイズして、製品の層を含みます。機能は必要に応じて追加および削除が可能です。テンプレートにはまた、[リリース] ビューに 4つのリリース (3 つの定義済み機能のそれぞれにフラグが付いているテンプレートと、すべての機能を含むテンプレート) が含まれています。これによって、各層に分かれたインストールと、エンド ユーザーがすべての使用可能な層をインストールできるインストールを個別にビルドすることができます。

この機能は、InstallShield の Premier Edition で提供されています。

.NET Framework 3.5 SP1、Microsoft Visual C++ 2012、および SQL Server 2008 R2 Express SP2 用の新しい InstallShield 前提条件

InstallShield には、「アドバンスト UI」、「基本の MSI」、「InstallScript」、「InstallScript MSI」、および「スイート/アドバンスト UI」 プロジェクトに追加可能な次の前提条件が含まれています:

- Microsoft .NET Framework 3.5 SP1 (Windows 機能)
- Microsoft SQL Server 2008 R2 Express SP2 (x64)
- Microsoft SQL Server 2008 R2 Express SP2 (x86 & x64Wow)
- Microsoft SQL Server 2008 R2 Express SP2 (x86)
- Microsoft Visual C++ 2012 Update 1 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2012 Update 1 再配布可能パッケージ (x86)

これらの前提条件は、サポートされているターゲット システムに様々なテクノロジをインストールします。

新しいアプリケーション仮想化適合性テスト

InstallShield では、製品を仮想化するための準備が整っているかどうかを判断するのに役立つ、新しい検証スイートが提供されています。これらのスイートで提供されている InstallShield 仮想化内部整合性検証ツール (ISVICE) を使って、Microsoft App-V、Microsoft Server App-V、VMware ThinApp、および Citrix XenApp の適合性を確認できます。検証スイートを使用することで、顧客に仮想バージョンの提供を考慮する場合に、製品をどのようにビルドするかについて、豊富な情報に基づいて意思決定を行うことができます。

InstallShield を使って、基本の MSI リリースが正しくビルドされるたびに、これらの検証スイートを実行するように構成する場合: [ツール] メニューから [オプション] を選択します。[検証] タブで、適切なチェック ボックスを選択します。

ビルド処理とは別に検証を行う場合: [ビルド] メニューから [検証] をポイントして、適切な新しいスイートを選択します。

仮想化スイートは、InstallShield Premier Edition で使用できます。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- InstallShield 仮想化整合性スイート
- プロジェクトの検証

Microsoft App-V 5 パッケージを作成するためのサポート

InstallShield の Microsoft App-V アシスタントには、Microsoft App-V 5 フォーマットで仮想アプリケーションを作成するためのサポートが含まれています。このアシスタントの [パッケージ情報] ビューを使って、App-V 5 または 4.x のどちらをターゲットにするかを指定できます。

App-V 5.x をターゲットにする場合、App-V 5.x パッケージを配布サーバーへ移動する前に、新しいバージョンの App-V ランチャー ツールを使って、それらをテストできます。このツールの新しいバージョンは、App-V パッケージの仮想環境内からコマンド プロンプト ウィンドウを起動することができます。そのため、App-V 5.x 以降の場合、Cmd.exe または Regedit.exe の診断ツール ショートカットを App-V パッケージに直接挿入する必要がなくなりました。

InstallShield が App-V パッケージをビルド時に保存するディレクトリは、ターゲットとする App-V のバージョンによって異なります。App-V 5 パッケージの場合、ディレクトリは ProductName で、App-VPackage という名前のフォルダーに配置されます。App-V 4.x パッケージの場合、InstallShield が ProductName フォルダーに製品バージョン番号を ProductName_v*N*（ここで *N* は製品バージョン番号）の形式で含みます。

InstallShield がサポートする任意の Windows バージョンで App-V 5.x パッケージの設定を構成できますが、InstallShield 内部から App-V 5.x パッケージをビルドするには Windows 8 または Windows Server 2012 が必要です。以前の Windows バージョンで App-V 5.x パッケージをビルドしようとすると、ビルド エラー -9424 が発生します。

Microsoft App-V アシスタントは、InstallShield Premier Edition で使用できます。

詳細については、次を参照してください:

- App-V パッケージのコンポーネント
- App-V パッケージのビルド出力

条件チェックの新しいプロパティの比較の種類 – スイート/アドバンスト UI および アドバンスト UI プロジェクトのインストール モードを判別するための新しいプロパティ

スイート/アドバンスト UI またはアドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクションの条件ステートメントをビルドするとき、ターゲット システム上で評価するチェックを様々な種類から選択できます。条件チェックに新しく追加された [プロパティの比較] 型を使って、特定のビルトイン アドバンスト UI またはスイート/アドバンスト UI プロパティ、あるいは [プロパティ マネージャー] ビューで定義されたプロパティの値をチェックできます。

さらに、ISInstallMode という新しい読み取り専用プロパティも使用できます。このプロパティは、スイート/アドバンスト UI またはアドバンスト UI インストールを実行しているモードを示す値を格納します（初回インストール、メンテナンス/UI メンテナンスの選択、変更、削除、修復、またはステージングのみ）。このプロパティは、プロジェクトで構成する条件ステートメントで使用できます。

詳細については、次を参照してください。

- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類
- "プロパティの比較" 条件の設定
- アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス

64 ビット専用 .msi および .msm パッケージをビルドできる機能 – 新しいビルド時のアーキテクチャの検証

Windows Server Core は、32 ビットの Windows-on-Windows (WOW64) サポートの無効化をサポートします。この構成がより一般化するのに伴い、64 ビットのアプリケーションが 32 ビットの機能に全く依存しないでインストールできるようにすることが必要となります。これを実現するため、InstallShield では 64 ビット専用 .msi パッケージのビルドが可能です。これらは、WOW64 機能を搭載していない 64 ビット版の Windows ベースのシステム上で実行します。また、ピュア 64 ビット マージ モジュールをビルドして、それを 64 ビット サポートを含む InstallShieldプロジェクトに含めることができます。インストールまたはマージ モジュール プロジェクトが 64 ビット専用環境をターゲットとし、任意のビルトイン InstallShield カスタム アクション DLL を必要とするサポートを含む場合、InstallShield はビルド時に、リリースにこれらの DLL の新しい 64 ビット バージョンを含めます。

今回より、InstallShield でリリースをビルドする時に使用できるアーキテクチャの検証が搭載されています。アーキテクチャの検証を使って、インストールがターゲット システムのアーキテクチャと一致しない製品ファイルをインストールしようとしたり、ランタイム バイナリを使用したりする問題を引き起こす可能性のある状況を検出することができます。

たとえば、エンド ユーザーが WOW64 サポートが搭載されていない 64 ビットのターゲット システムでインストールを実行する場合、アーキテクチャの検証を使って、インストールに含まれるすべての 32 ビット の製品ファイルおよび 32 ビットのカスタム アクション ファイルを識別できます。また、単一のプロジェクトに 64 ビットと 32 ビットの製品ファイル (または 64 ビットと32 ビットのカスタム アクション) が混在し、個別の 32 ビット .msi パッケージと 64 ビット .msi パッケージを生成する場合、アーキテクチャの検証を使って、32 ビット リリースに含まれる任意の 64 ビット製品またはカスタム アクション ファイルを識別することができます。これらのファイルを 32 ビット ターゲット システムでロードすることはできません。

使用するアーキテクチャの検証の種類を指定し、32 ビット専用パッケージ、または 64 ビット専用パッケージのどちらをビルドするのかを識別するには、[リリース] ビューの製品構成を選択したときに [全般] タブに表示される新しい "アーキテクチャの検証" 設定を使用します。2 種類の検証タイプ (厳密な検証をする、厳密な検証をしない) があります。

厳密な検証を行うオプションは、"テンプレートの概要" プロパティで指定されているアーキテクチャが、リリースに含まれている 1 つ以上のカスタム アクション ファイルのアーキテクチャと一致しない場合に、ビルド エラーをトリガする場合があります。この種類の検証ではまた、"テンプレートの概要" プロパティで指定されているアーキテクチャが、リリースに含まれている 1 つ以上の製品ファイルのアーキテクチャと一致しない場合にビルド警告をトリガする場合もあります。

デフォルトの検証の種類である、アーキテクチャの厳密な検証をしない場合、"テンプレートの概要" プロパティで指定されたアーキテクチャとリリースに含まれる 1 つ以上の製品ファイルまたはカスタム アクション ファイルのアーキテクチャが一致しないとき、ビルド エラーまたは警告がトリガされません。

オートメーション インターフェイスには、新しいアーキテクチャの検証サポートが含まれています。ISWiProductConfig オブジェクトに含まれている、新しい読み書きプロパティ ArchitectureValidation を使って、製品構成で使用するアーキテクチャの検証の種類を指定できます。

この機能は、基本の MSI およびマージ モジュール プロジェクト タイプで使用できます。

詳細は、次を参照してください:

- ビルドに適切なアーキテクチャ検証の種類を選択する
- 64 ビット オペレーティング システムをターゲットにする
- "テンプレート概要" プロパティを使用する
- ISWiProductConfig オブジェクト

Windows 8 ロゴ プログラムの検証

InstallShield では、新しい 2 つの検証スイート (InstallShield 検証スイート – Windows 8 および InstallShield マージ モジュール検証スイート - Windows 8) が提供されています。これらの検証スイートには、いくつかの新しい InstallShield 内部これらの検証スイートには、新しい ISICE (内部整合性検証ツール) ISICE21 から ISICE26 までが含まれています。いれらのスイートは、Windows Installer ベースのインストール、またはマージ モジュールが Windows 8 デスクトップ アプリケーション認証プログラムに対応するインストール要件を満たすかどうかの検証に役立ちます。Windows 8 ロゴを使用する場合、アプリケーションのインストールがプログラムの要件を満たしている必要があります。これらのスイートはまた、インストールまたは マージ モジュールが Windows Server 2012 システム上で動作することを検証するのにも役立ちます。

InstallShield を使って、リリースが正しくビルドされるたびに、これらの検証スイートを実行するように構成する場合: [ツール] メニューから [オプション] を選択します。[検証] タブで、適切なチェック ボックスを選択します。

ビルド処理とは別に検証を行う場合: [ビルド] メニューから [検証] をポイントして、適切な新しいスイートを選択します。

詳細は、次を参照してください:

- プロジェクトの検証
- ISICE
- Windows ロゴ プログラムの要件

最新プラットフォームにおけるショートカットの構成に関する強化内容 – ショートカット作成における InstallScript 言語の強化

InstallShield では、プロジェクトのショートカットを構成するための強化が提供されています。

Windows 8 スタート画面にショートカットをピン留めしないように防ぐサポート

InstallShield では、Windows 8 ターゲット システム上で、インストールに含まれる各ショートカットをデフォルトでスタート画面にピン留めするかどうかを指定できます。インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを無効化したい場合があります。ショートカットのピン留めを無効化した場合でも、システム上のすべてのショートカットを含むアプリケーション一覧には、そのショートカットが表示されます。

ショートカットをスタート画面にピン留めするのを防ぐには、[ショートカット] ビューでショートカットの新しい "Windows 8 スタート画面にピン留めする" 設定を使用します。

この機能は、基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

オートメーション インターフェイスでは、Windows 8 スタート画面にショートカットをピン留めするかどうかを指定できるサポートを提供します。ISWiShortcut オブジェクトには、Windows 8 ターゲット システム上で、デフォルトでスタ—ト画面にショートカットをピン留めするように構成されているかどうかを示すブール型 EnableWin8StartPin プロパティが含まれています。

詳細は、次を参照してください:

- Windows 8 スタート画面への初回のショートカットのピン留めを抑制する
- [ショートカット] ビュー
- ISWiShortcut オブジェクト

エンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めするのを防ぐサポート

インストールに含まれる各ショートカットについて、エンド ユーザーがショートカットをタスクバーまたは [スタート] メニューにピン留めできるようにするかどうかを指定できます。この種類のピン留めを防ぐ場合、インストールはショートカットをタスクバーおよび [スタート] メニューにピン留めするコンテキスト メニュー コマンドを非表示にするプロパティを設定します。インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを防ぎたい場合があります。

エンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めできないように防ぐサポートは、使用するプロジェクトの種類によって異なります。

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、およびトランスフォーム プロジェクトの場合: ショートカットのタスクバーおよび [スタート] メニューへのピン留めを防ぐには、[ショートカット] ビューでショートカットの "シェル プロパティ" 設定を使用します。この設定には、選択されたショートカットのピン留めを無効化できる、新しい [ピン留めを防ぐ] オプションが追加されました。Windows Installer 5 で、このショートカットの設定がサポートされています。以前、同じ動作を構成するためには、適切なプロパティ名と値を手入力する必要がありました。

InstallScript プロジェクトの場合: ショートカットのタスクバーおよび [スタート] メニューへのピン留めを防ぐには、[ショートカット] ビューに新しく追加された "ピン留めを防ぐ" 設定を使用します。Windows 7 以降で、このショートカットの設定がサポートされています。以前、InstallScript プロジェクトで、この機能の構成はサポートされていませんでした。

オートメーション インターフェイスでは、エンド ユーザーが製品をインストールした後、ショートカットをタスクバーおよび [スタート] メニューにピン留めするためのコンテキスト メニューを表示するかどうかを指定できるサポートが提供されています。ISWiShortcut オブジェクトには、ショートカットのピン留めを行うコンテキスト メニュー コマンドが非表示となるように構成されているかどうかを示す、読み取り専用のブール型 PreventPinning プロパティが含まれています。

詳細については、次を参照してください。

- エンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めできるかどうかを指定する
- [ショートカット] ビュー
- ISWiShortcut オブジェクト

[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐサポート

オプションで InstallShield では、エンド ユーザーが製品をインストールした後に、[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐことができます。これは、ターゲット システム上で個別のアイテムに対して [[スタート] メニューのカスタマイズ] ダイアログ ボックスで **[新しくインストールされたプログラムを強調表示する]** チェック ボックスをクリアした場合と同じ効果を持ちます。インストールの一部であるツールまたは従属的な製品のショートカットのプロパティを設定したい場合があります。

強調表示の動作は、使用するプロジェクトの種類によって異なります。

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合: ショートカットの強調表示を防ぐには、[ショートカット] ビューでショートカットの "シェル プロパティ" 設定を使用します。この設定には、選択されたショートカットの強調表示を無効化できる、新しい [新規として強調表示しない] オプションが追加されました。Windows Installer 5 で、このショートカットの設定がサポートされています。以前、同じ動作を構成するためには、適切なプロパティ名と値を手入力する必要がありました。

InstallScript プロジェクトの場合: ショートカットの強調表示を防ぐには、[ショートカット] ビューで新しい "新規として強調表示しない" 設定を使います。Windows 7 以降で、このショートカットの設定がサポートされています。以前、InstallScript プロジェクトで、この機能の構成はサポートされていませんでした。

オートメーション インターフェイスには、エンド ユーザーが製品をインストールした後に、[スタート] メニューで新規項アイテムとして強調表示するかどうかを指定できるサポートが含まれています。ISWiShortcut オブジェクトには、この強調表示を阻止するかどうかを示す新しい読み取り専用のブール型 DoNotHighlightAsNew プロパティが含まれています。

詳しくは、次を参照してください:

- [スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐ
- [ショートカット] ビュー
- ISWiShortcut オブジェクト

ショートカットの追加 Windows シェル プロパティのビルトイン サポート

InstallShield では、実行時に Windows シェルによる設定が必要な 1 つ以上の追加ショートカット プロパティを指定できます。シェルが設定できるプロパティは、Windows SDK に含まれている propkey.h で定義されています。

ショートカットの追加プロパティを構成するには、[ショートカット] ビューでショートカットの "シェル プロパティ" 設定を使用します。この設定には、選択されたショートカットに追加プロパティとそれに対応する値を指定できる、新しい [カスタム プロパティ] オプションが追加されています。

Windows Installer 5 では、シェル プロパティの構成がサポートされています。

オートメーション インターフェイスには、ショートカットにカスタム Windows シェル プロパティ用の ISWiShellProperty オブジェクトが含まれています。さらに、ISWiShortcut オブジェクトには、ショートカットにシェル プロパティを追加する (AddShellProperty)メソッド、シェル プロパティを削除する (DeleteShellProperty) メソッド 、およびシェル プロパティのコレクションを取得する (ISWiShellProperties) が含まれています。

このサポートは、基本の MSI、DIM、 InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細は、次を参照してください:

- カスタム アクション シェルのプロパティを設定する
- [ショートカット] ビュー
- ISWiShellProperty オブジェクト
- AddShellProperty メソッド
- DeleteShellProperty メソッド
- ISWiShellProperties コレクション
- ISWiShortcut オブジェクト

ショートカット プロパティの設定、その他のショートカット機能における InstallScript 言語の強化

SetShortcutProperty という名前の新しい InstallScript 関数が追加されました。InstallScript コードでこの関数を使って、ショートカットの Windows シェル プロパティを設定します。この関数は、エンド ユーザーが Windows 8 スタート画面にショートカットをピン留めすることを防ぎ、Windows 7 以降のシステムでは、タスクバーまたは [スタート] メニューにショートカットをピン留めすることを防ぎ、Windows 7 以降のシステム上では [スタート] メ

ニューのショートカットが新しくインストールされたアイテムとして強調表示されないように防ぎます。またこの関数を使って、Windows SDK に含まれている propkey.h で定義されている追加シェル プロパティを指定することもできます。

InstallShield キャビネット & ログ ファイル ビューアは、前述のシェル プロパティの状態を反映して更新されています。

現在のオペレーティング システムに関する用語を反映して、既存のショートカット関数の名前が変更されています。古い関数名も、InstallShield の以前のバージョンと同様にコンパイルおよびビルドすることが可能です。新しい InstallScript 関数は、次の通りです:

- **CreateShortcut**—この関数は AddFolderIcon に優先します。**CreateShortcut** は、**AddFolderIcon** と同じパラメーターを受け付けますが、nFlag パラメーターの以下のオプションを追加または置換します:
  - CS_OPTION_FLAG_REPLACE_EXISTING
  - CS_OPTION_FLAG_RUN_MAXIMIZED
  - CS_OPTION_FLAG_RUN_MINIMIZED
  - CS_OPTION_FLAG_PREVENT_PINNING
  - CS_OPTION_FLAG_NO_NEW_INSTALL_HIGHLIGHT
  - CS_OPTION_FLAG_NO_STARTSCREEN_PIN
- **CreateShortcutFolder** — この関数は **CreateProgramFolder** に優先します。どちらの関数も、同じパラメーターを使用します。
- **DeleteShortcut** — この関数は **DeleteFolderIcon** に優先します。どちらの関数も、同じパラメーターを使用します。
- **DeleteShortcutFolder** — この関数は **DeleteProgramFolder** に優先します。どちらの関数も、同じパラメーターを使用します。
- **GetShortcutInfo** — この関数は **QueryProgItem** に優先します。どちらの関数も、同じパラメーターを使用します。
- **ReplaceShortcut** — この関数は **ReplaceFolderIcon** に優先します。**ReplaceShortcut** は、**ReplaceFolderIcon** と同じパラメーターを受け付けますが、nFlag パラメーターに **CreateShortcut** と同じオプションを追加または置換します。

詳細については、次を参照してください。

- CreateShortcut
- CreateShortcut の例
- CreateShortcutFolder
- CreateShortcutFolder の例
- DeleteShortcut
- DeleteShortcut の例
- DeleteShortcutFolder
- DeleteShortcutFolder の例
- GetShortcutInfo

- GetShortcutInfo の例
- ReplaceShortcut
- ReplaceShortcut の例
- SetShortcutProperty
- SetShortcutProperty の例

アドバンスト UI およびスイート/アドバンスト UI プロジェクトで、ウィザード インターフェイスのカスタマイズにおけるユーザビリティーの向上

混乱しやすい構文要件の削除

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの [ウィザード インターフェイス] ビューにある様々な UI 設定が改良され、アドバンスト UI およびスイート/アドバンスト UI インストールのウィザード インターフェイスのカスタマイズがより簡単になりました。たとえば、[ウィザード インターフェイス] ビューでウィザード ページまたはウィザード コントロールを選択したときに表示される次の設定が改良されています:

- **表示**—この設定を使って、選択されたページまたはコントロールを表示するかどうかを評価するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する 1 つ以上の条件を指定できます。
- **有効**—この設定を使って、選択されたコントロールを有効化するかどうかを評価するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する 1 つ以上の条件を指定できます。
- **検証**—この設定を使って、エンド ユーザーによるコントロールへの応答を評価するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する 1 つ以上の検証ステートメントを指定できます。たとえば、この設定を使ってエンド ユーザーがテキスト ボックス コントロールにシリアル番号を入力するのに使用するフォーマットを指定できます。
- **アクション**—この設定の各インスタンスは、選択されたコントロールに適切な特定の種類のトリガを反映した、より相応しい名前に変更されています。たとえば、リスト ボックス コントロールの "アクション" 設定の名前は、"選択範囲の変更" に変更されています。ボタンの "アクション" 設定の名前は、"クリック" に変更されています。名前が変更されている設定を使って、エンド ユーザーが選択されたコントロールを使ったときにトリガする 1 つ以上のアクションを選択できます。

これらの設定には、今回より、アドバンスト UI およびスイート/アドバンスト UI プロジェクトのその他の領域で使用できるドロップダウン リストとボタンと同じような要素が含まれていて、素早く条件ステートメントをビルドすることができます。これらの設定を使って、UI の様々な領域で状態の表示/非表示、状態の有効化/無効化、検証、およびアクションの応答を素早く構成して、適切な場合、その動作に対する条件ステートメントをビルドできます。以前、これらの設定は、覚え難い構文を使ったステートメントを手作業で入力しなくてはならない編集フィールドでした。

詳細については、次を参照してください。

- ウィザード ページまたはウィンドウに含まれるコントロールの検証を構成する
- ウィザード インターフェイス内の要素のアクションを構成する
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

ウィザード インターフェイスのヘッダー、本文、およびナビゲーション領域に単一の背景を使用できるビルトイン サポート

[ウィザード インターフェイス] ビューの [ウィザード ページ] ノードに、新しい "完全なウィザード背景" 設定が追加されました。この設定を使って、ウィザード ページのヘッダー、本文、およびナビゲーション領域のすべてにわたって使用する単一の背景を指定できます。以前は、InstallShield で背景を指定するためのビルトインされた方法は、"デフォルトの本文背景"、"ヘッダー背景"、および "ナビゲーション背景" 設定で個別の背景スタイルを使用する方法しかありませんでした。ウィザード インターフェイスの 3 つの領域すべてで単一の背景を使用したい場合、InstallTalk ブログ記事に記載されているように、テキスト エディターで InstallShield プロジェクト ファイル (.issuite) を変更する処理を行う必要がありました。

この新しい "完全なウィザード背景" 設定で背景を選択する場合、その他の背景設定 ("デフォルトの本文背景"、"ヘッダー背景"、および "ナビゲーション背景") からすべての値を削除しなくてはなりません。

詳細については、「ウィザード インターフェイスのヘッダー、本文、およびナビゲーション領域に背景ブラシを指定する」を参照してください。

すべてのナビゲーション ボタンにグローバル テキスト スタイルを指定できる機能

[ウィザード インターフェイス] ビューの [ウィザード ページ] ノードに、新しい "ナビゲーションのテキスト スタイル" 設定が追加されました。この設定を使って、すべてのナビゲーション ボタン上のテキストに使用する単一のテキスト スタイルを選択できます。必要な場合、個別のナビゲーション ボタンのグローバル テキスト スタイルをオーバーライドできます。

論理的に編成された設定グリッド

[ウィザード インターフェイス] ビューでは、ウィザード インターフェイスをカスタマイズするのに必要な設定を見つけやすくするため、ウィザード ページのグリッドの設定、2 番目のウィンドウ、およびウィザード コントロールが再編成されています。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでフォント セットを定義し、オプションで言語固有の選択を行う機能のサポート

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、フォント セットという、新しい種類のウィザード インターフェイス スタイルがサポートされています。フォント セット とは、フォント名、サイズ、太さなどの属性を含むフォントのコレクションです。フォント セットに含まれる各フォントについて、それが適用される言語を指定できます。これによって、プロジェクトでサポートする各言語に異なるフォントを選択できます。

フォント セットは、テキスト スタイルと組み合わせて使用します。テキスト スタイル (テキストの色や配置などのテキスト属性を定義するスタイル) は、今回より、フォント セットを参照しますが、オプションでフォント セットのレベルで定義されている様々なフォント属性をオーバーライドすることが可能です。

たとえば、ウィザード インターフェイスのテキスト本文にフォント情報を指定する時、BodyFonts と呼ばれるフォント セットを使用して、プロジェクトでサポートされている各言語に異なるフォントを設定できます。その後、様々なテキスト スタイル (本文、ヘッダー、および BodyBold) に BodyFonts フォントセットを選択できますが、必要に応じて小さいサイズや太字を特別にオーバーライドすることができます。これによって、可能なフォントの大きなセットを一度指定しておいて、特定のウィザード インターフェイスが使用するテキスト サイズや色などの属性を変更することができます。

詳しくは、次を参照してください:

- スタイル、ブラシ、コントロール テーマ、および最小ウィザード ページを使ってウィザード インターフェイスをカスタマイズする
- カスタム スタイルをウィザード インターフェイスに定義する

強化機能

InstallShield には、以下のような強化機能が搭載されています。

新しい [サービス] ビュー

新しい [サービス] ビュー ([ビュー] リストの [システム構成] ノードの下) では、実行時にインストール、開始、構成、停止、または削除する Windows サービスについて必要な情報を指定できます。

この新しい [サービス] ビューは、新しいユーザーがサービスを構成しやすいように追加されました。このビューは次の既存領域と同じ機能を提供します:

- (インストール プロジェクトの) [セットアップのデザイン] ビューでコンポーネントの [詳細設定] ノードの下にある [サービス] 領域
- [コンポーネント] ビューでコンポーネントの [詳細設定] ノードの下にある [サービス] 領域

3 つのビュー ([サービス] ビュー、[セットアップのデザイン] ビュー、または [コンポーネント] ビュー) のいずれかでサービス情報を構成すると、対応するビューも自動的に更新されます。

[サービス] ビューは、基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- Windows サービスのインストール、制御、および構成
- [サービス] ビュー

スイート/アドバンスト UI およびアドバンスト UI ウィザード インターフェイスの新しいウィザード フォーマット

スイート/アドバンスト UI およびアドバンスト UI インストールのウィザード インターフェイスで使用できる新しい Glass ウィザード フォーマットこの新しいフォーマットは、既存の Aero フォーマットに似ています。どちらのフォーマットでも、インストールのユーザーが (プロジェクトで定義されたブラシ スタイルの代わりに) ユーザー選択のシステム色を使ってウィザード インターフェイスのキャプション バー、ヘッダー、およびナビゲーション領域を表示できます。Windows Vista から使用できる透明感サポートが有効なターゲット システム上では、ウィザード インターフェイスのこれらの同じ領域にガラス効果 (半透明) を使用することもできます。

一部の状況下で、インストールは Glass または Aero フォーマットの代わりに Wizard 97 フォーマットを使用します:

- Windows 8 および Windows Server 2012 システムは、透明感をサポートしません。これらのシステムでは、Glass フォーマットを持つウィザード インターフェイスは Wizard 97 フォーマットを使用します。ただし、Aero フォーマットを持つウィザード インターフェイスは (ユーザー選択のシステム色を含む) Aero を使用しますが、ガラス効果はありません。
- Windows 7、Windows Vista、Windows Server 2008 R2、および Windows Server 2008 では、エンド ユーザーが有効化/無効化できる透明感サポートが使用できます。これらのシステムでは、透明感が無効化されている場合、Glass フォーマットを使ったウィザード インターフェイスと Aero フォーマットを使ったウィザード インターフェイスでは、Wizard 97 フォーマットが使用されます。また、これらのウィザード インターフェイスでは、ターゲット システム上のデスクトップ コンポジション機能が無効化されている場合、Wizard 97 フォーマットが使用されます。
- Windows XP および Windows Server 2003 上では、Glass フォーマットのウィザード インターフェイスと Aero フォーマットのウィザード インターフェイスは Wizard 97 フォーマットを使用します。

フォーマットを構成するには、[ウィザード ページ] ノードを選択したときに [ウィザード インターフェイス] ビューに表示される "ウィザード フォーマット" 設定を使用します。すべての新しいスイート/アドバンスト UI およびアドバンスト UI プロジェクトでは、Glass フォーマットがデフォルト オプションです。

詳細については、「ウィザード インターフェイスのフォーマットを選択する」を参照してください。

オートメーション インターフェイスの強化内容: マイナー アップグレードおよびスモール アップデートで使用できる新しい OnUpgrade プロパティ

オートメーション インターフェイスで、新しい読み書きプロパティ OnUpgrade が ISWiProject オブジェクトに追加されました。このプロパティをサポートされている値の 1 つに設定して、マイナー アップグレードまたはスモール アップデートをインストールする前にプロンプトを表示するかどうかを指定できます:

このプロパティは、[アップグレード] ビューの [共通] タブにある "スモール/マイナー アップグレード" 設定オプションに対応します。

詳細については、「ISWiProject オブジェクト」を参照してください。

InstallShield 2012 Spring SP1 の新しい機能

InstallShield 2012 Spring Service Pack 1 (SP1) には、Windows 8、Windows Server 2012、および Visual Studio 2012 の最終版をサポートするための変更が含まれています。また、その他の変更も含まれています。



重要 *• InstallShield 2012 SP1 で InstallShield 2012 Spring アドバンスト UI または スイート/アドバンスト UI プロジェクト (.issuite) を開くとき、プロジェクトを InstallShield 2012 Spring SP1 にアップグレードすることを許可する必要があります。InstallShield 2012 SP1 アドバンスト UI およびスイート/アドバンスト UI プロジェクトには、InstallShield 2012 Spring でこれらのプロジェクト タイプでは使用できなかった機能のサポートが提供されていますが、このサポートはアップグレード中に追加する必要があります。InstallShield 2012 Spring SP1 アドバンスト UI またはスイート/アドバンスト UI プロジェクトを以前のバージョンの InstallShield (SP1 を適用する前の InstallShield 2012 Spring を含む) で開くことはできませんので、ご注意ください。このため、複数のユーザーが InstallShield プロジェクトを開いて編集する場合、すべてのユーザーが同時に SP1 パッチを適用するようにしてください。*

InstallShield 2012 Spring アドバンスト UI またはスイート/アドバンスト UI プロジェクトを InstallShield Spring SP1、InstallShield 2012 Spring SP1 で開くと、そのプロジェクトを新しいバージョンに変換するかどうかをたずねるメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前にプロジェクトのバックアップ コピーが作成されます。

サイドローディング Windows アプリ パッケージ (.appx) をスイート/アドバンスト UI プロジェクトに追加するためのサポート

InstallShield では、今回よりサイドローディング Windows アプリ パッケージ (.appx) をスイート/アドバンスト UI プロジェクトに追加できます。その他の種類のパッケージをプロジェクトに追加する処理と同様に、このパッケージ タイプをスイート アドバンスト UI の [パッケージ] ビューを使って追加できます。

アプリのサイドローディングとは、Windows Store を介さずにアプリケーションをインストールするプロセスです。この種類のアプリケーションは、企業環境で配布されることがあります。Windows 8 および Windows Server 2012 では、サイドローディング アプリケーションがサポートされています。

このサポートは、InstallShield の Premier Edition で提供されています。

サイドローディング Windows アプリ パッケージを含むスイート/アドバンスト UI インストールを作成およびビルドするには、InstallShield または Standalone Build が搭載されているマシン上に Windows Vista 以降または Windows Server 2008 以降が必要です。

詳細については、次を参照してください。

- サイドローディング Windows アプリ パッケージ (.appx| msix) をスイート/アドバンスト UI プロジェクトに追加する
- [パッケージ] ビュー

スイート/アドバンスト UI およびアドバンスト UI プロジェクトにおける、条件チェックに追加された Windows アプリ パッケージ タイプ

スイート/アドバンスト UI またはアドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、ターゲット システム上で評価するチェックを様々な種類から選択できます。新しい Windows アプリ パッケージ タイプの条件チェックを使うと、ターゲット システムで特定 Windows アプリ パッケージの存在をチェックできます。条件は特定のアプリケーション名をチェックし、製品バージョンなどのその他の情報もチェックできます。

詳細については、次を参照してください。

- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類
- 「Windows アプリ パッケージの条件" の設定
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

Visual Studio 2012、.NET Framework 4.5、および Visual C++ 2012 のサポート

InstallShield Visual Studio 2012 の最終版をサポートするための変更が含まれており、このバージョンの Visual Studio インターフェイス内部でインストールおよび製品の開発が可能です。

さらに、InstallShield には 2 つの .NET Framework 用 InstallShield 前提条件および新しい 2 つの Visual C++ 用 InstallShield 前提条件が提供されています:

- Microsoft .NET Framework 4.5 Full
- Microsoft .NET Framework 4.5 Web
- Microsoft Visual C++ 2012 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2012 再配布可能パッケージ (x64)

これらの前提条件は、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加できます。

.NET Framework の Web 前提条件には、インターネット接続が必要です。この前提条件は、必要に応じて、必須の再配布可能ファイルをダウンロードします。完全な前提条件は、インターネットへの接続が不要なスタンドアロン インストールです。

追加の変更

InstallShield 2012 Spring SP1 で解決されている問題の一覧は、リリース ノートをご覧ください。リリース ノートは、InstallShield の [ヘルプ] メニューからご覧になることができます。

InstallShield 2012 Spring の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

Windows 8 および Windows Server 2012 システムをターゲットできる機能

InstallShield では、インストールに Windows 8 または Windows Server 8 が必要であることを指定できます。また、これらのオペレーティング システムに対する機能条件およびコンポーネント条件をビルドすることができます。

Windows 8 および Windows Server 2012 にインストール可能な InstallShield 前提条件は、必要に応じて、これらのシステムにインストールされるように更新されています。以前これらのシステムでは、前提条件がデフォルトでは実行されませんでした。これは、次の InstallShield 前提条件に適用します:

- FSharp Redistributable Package 2.0
- Microsoft ReportViewer 2010
- Microsoft SQL CE 3.5 SP2
- Microsoft SQL Server 2005 Express SP3
- Microsoft SQL Server 2008 Express SP1
- Microsoft SQL Server 2008 Management Objects 10.00.2531
- Microsoft SQL Server 2008 Native Client 10.00.2531
- Microsoft SQL Server 2008 R2 Express RTM
- Microsoft SQL Server 2008 R2 Native Client 10.50.1600.1
- Microsoft SQL Server Native Client 9.00.4035
- Microsoft SQL Server System CLR 型 10.00.2531
- Microsoft Visual C++ 2005 SP1 Redistributable MFC セキュリティ更新プログラム KB2538242
- Microsoft Visual C++ 2005 SP1 Redistributable Package
- Microsoft Visual C++ 2008 SP1 Redistributable MFC セキュリティ更新プログラム KB2538243
- Microsoft Visual C++ 2008 SP1 Redistributable Package
- Microsoft Visual C++ 2010 再配布可能パッケージ
- Microsoft Visual C++ 2010 RTM Redistributable MFC セキュリティ更新プログラム KB2467173
- Microsoft Visual C++ 2010 SP1 Redistributable Package
- Microsoft VSTO 2010 Runtime

InstallShield Profesioanl Edition で最新のカスタマイズ可能なエンド ユーザー インターフェイスをサポート

新しいアドバンスト UI プロジェクト タイプでは、Windows Installer パッケージまたは InstallScript パッケージ用に、デザインが一新された最新のウィザード ページを持つ新しいエンドユーザー インターフェイスを作成することができます。InstallShield Professional Edition で提供されているこの新しいプロジェクト タイプでは、以前 InstallShield Premier Edition で紹介されたスイートプロジェクト タイプ (現在はスイート/アドバンスト UI プロジェクト タイプと呼ばれています) と同じテクノロジーが使用されています。

新しいアドバンスト UI プロジェクト タイプでは、アドバンスト UI インストールに含めることができるカスタマイズ可能なビルトイン ウィザード ページが使用されています。このプロジェクト タイプで使用できるウィザード ページ エディターでは、必要に応じてページを追加、シーケンス、または削除することができます。また、様々な異なる種類のコントロールを追加または削除することで任意のページのレイアウトを編集することもできます。以前 Premier Edition でのみ使用可能だったすべての新しい UI 機能は、アドバンスト UI プロジェクトで使用可能になっています。

スイート/アドバンスト UI プロジェクト同様、アドバンスト UI プロジェクトでは、ターゲット システムでパッケージを起動する次世代セットアップ ランチャー (Setup.exe) を使用しています。また、スイート/アドバンスト UI プロジェクト同様、アドバンスト UI プロジェクトでは、アドバンスト UI インストールに含まれるパッケージのランタイム ソースの場所を指定できる柔軟なオプションが提供されています。以下は、選択可能な場所です:

- Web 上で、必要に応じて Setup.exe によるダウンロードが可能
- Setup.exe に埋め込み、必要に応じてターゲット システムに展開
- 非圧縮ファイルでスイートのアドバンスト UI ソース メディアに格納

エンド ユーザーはコンパクトなアドバンスト UI Setup.exe ファイルを短時間でダウンロードすることができ、Setup.exe ファイルが必要に応じて、Windows Installer ベースまたは InstallScript のパッケージをダウンロードおよび起動します。

アドバンスト UI プロジェクト タイプでは、プライマリ パッケージとして含めることができるパッケージは、.msi パッケージ、.msp パッケージ、または InstallScript パッケージの中からいずれか 1 つのみです。InstallShield Premier Edition で提供されているスイート/アドバンスト UI プロジェクト タイプでは、.exe パッケージを含む複数のプライマリ インストールを単一インストールにパッケージ化できる機能が提供されています。

詳しくは、次を参照してください:

- アドバンスト UI およびスイート/アドバンスト UI インストールの作成
- アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い
- .msi パッケージ、.msp パッチ、トランザクションをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する
- ウィザード インターフェイスを使って作業する

Microsoft Windows Azure SQL データベース サポート

InstallShield では、Microsoft Windows Azure SQL データベース サーバーで SQL スクリプトを実行できるサポートが提供されています。さらに、InstallShield の [SQL スクリプト] ビューには、製品がサポートするターゲットのデータベース サーバーを指定するときに選択可能な定義済みのデータベース サーバーのリストに Microsoft Windows Azure が含まれています。

インストールが Windows Azure をターゲットとする場合、エンド ユーザーがデータベース カタログの参照を選択したときに表示される SQLBrowse ランタイム ダイアログに、SQL データベース サーバーで指定されたカタログの一覧が含まれています。

このサポートは、基本の MSI、DIM、InstallScript、および InstallScript MSI プロジェクト タイプで利用できます。

詳細は、次を参照してください:

- Microsoft SQL Azure Database サーバーへの接続と SQL スクリプトの実行
- SQL サポートの構成
- [SQL スクリプト] ビュー

このサポートは、基本の MSI、DIM、InstallScript、および InstallScript MSI プロジェクト タイプで利用できます。

Microsoft Visual Studio 2012 のベータ サポート

InstallShield には Visual Studio 2012 ベータのサポートが含まれています。このバージョンの Visual Studio 内部から InstallShield プロジェクトを作成できます。

Microsoft .NET Framework 4.5 前提条件

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる .NET 関連の新しい 2 つの InstallShield 前提条件が含まれています:

- Microsoft .NET Framework 4.5 Full
- Microsoft .NET Framework 4.5 Web

これらの InstallShield 前提条件は、サポートされているターゲット システムに、ベータ バージョンの .NET Framework 4.5 をインストールします。

Web 前提条件には、インターネットへの接続が必要です。この前提条件は、必要に応じて、必須の再配布可能ファイルをダウンロードします。完全な前提条件は、インターネットへの接続が不要なスタンドアロン インストールです。

Microsoft SQL Server 2012 サポート

InstallShield に、SQL Server 2012 Database 上で SQL スクリプトを実行できるサポートが追加されました。また、InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストには、SQL Server 2012 が含まれています。

インストールで SQL Server 2012 をターゲットするとき、エンドユーザーがデータベース カタログを参照する選択をしたときに表示される SQLBrowse ランタイム ダイアログで、SQL Server 2012、SQL Server 2012 Express、および SQL Server 2012 Express LocalDB のインスタンスが表示できるようになりました。また、エンドユーザーがデータベース カタログを参照する選択をしたときに表示される SQLBrowse ランタイム ダイアログで、指定された SQL Server 2012 サーバー上のカタログが表示されるようになりました。

詳細は、次を参照してください:

- SQL Server Express LocalDB のインスタンスに接続するときの要件
- SQL サポートの構成
- [SQL スクリプト] ビュー

このサポートは、基本の MSI、DIM、InstallScript、および InstallScript MSI プロジェクト タイプで利用できます。

Microsoft SQL Server 2012 の前提条件

InstallShield に、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加できる SQL Server 2012 関連の新しい InstallShield 前提条件がいくつか追加されました:

- Microsoft SQL Server 2012 Express
- Microsoft SQL Server 2012 Express LocalDB
- Microsoft SQL Server 2012 Native Client

InstallShield には、Microsoft SQL Server 2012 Express の依存関係である Microsoft .NET Framework 3.5 SP1 Update KB956250 をインストールする InstallShield 前提条件も含まれています。

これらの InstallShield 前提条件は、サポートされているターゲット システムにテクノロジーがインストールされます。

App-V 4.6 SP1、SQL Server Compact 4.0 および JRE SE 1.7 の新しい InstallShield 前提条件

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる、新しい InstallShield 前提条件が含まれています:

- Java Runtime Environment Second Edition (JRE SE) 1.7
- Microsoft App-V 4.6 SP1 Desktop Client (これらの InstallShield 前提条件の再配布可能ファイル (32 ビット バージョンおよび 64 ビット バージョン) は、InstallShield 内部からはダウンロードできません。これらは、Microsoft から取得する必要があります。Microsoft からそれらのファイルを入手した後、InstallShield 前提条件エディターで前提条件を編集するときに表示される場所に配置してください。)
- SQL Server Compact 4.0

これらの InstallShield 前提条件は、サポートされているターゲット システムにテクノロジーがインストールされます。

Configuring System Center 2012 Configuration Manager のアプリケーション データの構成のサポート

配置メタデータを正確に識別することは、アプリケーションを System Center 2012 Configuration Manager アプリケーション モデルに移行する上で必須です。InstallShield の [一般情報] ビューには、ソフトウェア ID タグを使ってアプリケーション モデル メタデータの一部を指定するための、いくつかの新しい設定が含まれています。AdminStudio のユーザーが AdminStudio のアプリケーション カタログにパッケージをインポートしたとき、パッケージの要素が解析されて、検出方法、依存関係、要件などの配置データ、およびソフトウェア識別タグ内の情報が確認されます。AdminStudio では、この情報を Microsoft System Center 2012 Configuration Manager に公開する前に、確認およびテスト用として、ユーザーに提供しています。

この機能は、基本の MSI プロジェクトで使用できます。

詳細については、「製品の Microsoft System Center Configuration Manager アプリケーション モデル データを含める」を参照してください。

PowerShell カスタム アクションのサポート

Windows PowerShell は、構成タスクのオートメーション化を可能にする .NET Framework ベースのコマンドライン シェルおよびスクリプト言語です。InstallShield では、PowerShell スクリプトを実行するカスタム アクションをサポートします。この種類のカスタム アクションは、インストールの実行時にシステムの構成タスクを実行するプロジェクトに追加することができます。

インストールで、PowerShell カスタム アクションを実行するには、Windows PowerShell がターゲット システムにインストールされている必要がありますので注意してください。InstallShield には、ターゲット システムで PowerShell の有無を確認する新しい定義済み PowerShell システム検索が追加されました。このシステム検索をプロジェクトに含めると、PowerShell カスタム アクションが、PowerShell がインストールされていることが判別されたときのみ実行されるように構成できます。

ターゲット システムで PowerShell スクリプトを実行できるかどうかを判別する PowerShell 実行ポリシーは、デフォルトで、PowerShell スクリプトの実行が許可していない制限付きになっています。インストールで、ターゲット システムの実行ポリシーを、インストールの PowerShell カスタム アクションに適切なポリシーでオーバーライドする場合、Windows Installer プロパティ IS_PS_EXECUTIONPOLICY を使用して、適切な実行ポリシーを指定できます。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

詳細については、「PowerShell カスタム アクションの呼び出し」を参照してください。

スイート/アドバンスト UI およびアドバンスト UI インストールのアップデートの自動確認とダウンロードのサポート

スイート/アドバンスト UI およびアドバンスト UI インストールでは、Web サイトでホストする更新済みのスイート/アドバンスト UI またはアドバンスト UI の Setup.exe ファイルを自動確認し、アップデートが存在する場合、それをダウンロードし起動する機能を使用できます。更新されたスイート/アドバンスト UI またはアドバンスト UI の Setup.exe ファイルは、最新のスイート/アドバンスト UI およびアドバンスト UI パッケージ用のアップグレードやパッチを配信するために使用できます。

スイート/アドバンスト UI およびアドバンスト UI プロジェクトの [リリース] ビューの [アップデート] タブに、更新されたスイート/アドバンスト UI またはアドバンスト UI セットアップ ランチャーの絶対パス (ファイル名を含む) を指定できる "アップデート URL" 設定が新しく追加されました。ベースのスイート/アドバンスト UI またはアドバンスト UI セットアップ ランチャーによってメンテナンス以外の操作が実行される場合、アップデート URL をチェックして、ダウンロード可能なアイテムがあるかどうかが確認されます。ダウンロード可能なアイテムが存在する場合、ダウンロードが実行され、デジタル署名が確認されます。アップデート セットアップ ランチャーのデジタル署名がベースのセットアップ ランチャの署名と一致すると、アップデート セットアップ ランチャーが自動的に実行されます。デジタル署名が一致しなかった場合、または、ベース セットアップ ランチャーがデジタル署名されていない場合、エンドユーザーがアップデート セットアップ ランチャーの実行を続けるかどうかを選択できるセキュリティ警告が表示されます。

詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート」を参照してください。

特定のバージョンのスイート/アドバンスト UI またはアドバンスト UI インストールが既にインストールされているかどうかを判別できる機能

スイート/アドバンスト UI およびアドバンスト UI プロジェクトに、特定のバージョンのスイート/アドバンスト UI またはアドバンスト UI インストールが既にインストールされているかどうかを判別するためのサポートが追加されました。このタイプの条件チェックは "インストール済みスイート" 条件と呼ばれます。

デフォルトで、スイート/アドバンスト UI およびアドバンスト UI インストールには、それぞれ、2 つの "インストール済みスイート" 条件が含まれています:

- 新しいインストール済みスイートの終了条件は、エンドユーザーが、スイート/アドバンスト UI またはアドバンスト UI インストールの現在のバージョンによって、同じスイート/アドバンスト UI またはアドバンスト UI インストールの将来のバージョンが上書きインストールされるのを防ぎます。
- 同じバージョンのスイート/アドバンスト UI またはアドバンスト UI インストールが既にインストールされている場合、"インストール済みスイート" のモード条件によって、インストールが初回インストール モードで実行されるのが禁止されることがあります。

これらの新しいデフォルトの条件は、スイート/アドバンスト UI およびアドバンスト UI プロジェクトで使用できます。InstallShield 2012 スイート プロジェクトを InstallShield 2025 にアップグレードした場合、これらのデフォルト条件は自動的にプロジェクトに追加されます。

デフォルトの "インストール済みスイート" 条件は、必要に応じて編集が可能です。また、必要に応じて、プロジェクトに自作のインストール済みスイート条件を追加することもできます。

詳細については、次を参照してください。

- 特定のバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールが既にインストールされているかどうかを判別する
- アドバンスト UI またはスイート/アドバンスト UI インストールのインストール モードまたはメンテナンス モードをトリガする
- "インストール済み" 条件の設定
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

インストールのフォルダーのネットワーク共有を構成できる機能

InstallShield では、ネットワーク上でファイルの共有を有効にするネットワーク共有の作成、変更、および削除が可能になりました。プロジェクトでフォルダーを構成するとき、ネットワーク共有を有効にするかどうかを指定します (デフォルトでは無効になっています)。また、共有の名前、共有にアクセスできる同時ユーザーの最大数など、その他のオプションも構成できます。

共有を有効にするには、[プロパティ] ダイアログ ボックスに追加された新しい [共有] タブ ([ファイルとフォルダー] ビューでフォルダーを右クリックすると表示されます) を使用し、[プロパティ] をクリックします。

この機能は、基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、および MSM データベース プロジェクト タイプで使用できます。

詳細については、「[フォルダーのプロパティ] ダイアログ ボックス」を参照してください。

特定のプロセスを終了する新しいビルトイン カスタム アクション

InstallShield に、新しい Kill-Process タイプのカスタム アクションが追加されました。このタイプのカスタム アクションをプロジェクトに追加すると、実行時に強制終了する 1 つまたは複数のプロセスの名前または ID を指定することができます。また、即時または遅延モードのカスタム アクションをスケジュールすることもできます。

このタイプのカスタム アクションを作成する手順には、プロジェクトの [カスタム アクションとシーケンス] ビューにカスタム アクションを追加して構成する作業と、プロパティ マネージャーを使って、適切なプロセスの名前または PID で、プロパティを定義する作業が含まれます。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

詳しい手順については、「プロセスの強制終了カスタム アクションの呼び出し」を参照してください。

実行時に既定のユーザー アカウントとグループを作成できる機能

InstallShield に、実行時に、複数のユーザー アカウントと対応するグループを、ログイン ダイアログを使わずに作成できるビルトイン サポートが追加されました。アカウントおよびグループを構成するには、プロジェクトのプロパティ ISNetApiLogonUsername、ISNetApiLogonGroup、および ISNetApiLogonPassword の値をそれぞれユーザー名、グループ、パスワードで定義します。複数の名前、グループ、またはパスワードは、角かっこを使ったチルダ [~] で囲んで区切ります。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

詳細については、「実行時に既定のユーザー アカウントを作成する」を参照してください。

スイート/アドバンスト UI およびアドバンスト UI プロジェクトに InstallShield 前提条件をパッケージとして含める機能

新しい InstallShield で、InstallShield 前提条件を、.msi や .exe パッケージとしてスイート/アドバンスト UI およびアドバンスト UI プロジェクトにインポートすることができるようになりました。インポートできる前提条件は、InstallShield に含まれている InstallShield 前提条件だけでなく、自分で作成したカスタム InstallShield 前提条件もインポートすることができます。[パッケージ] ビューにある [パッケージ] エクスプローラーを右クリックし、新しい 前提条件 (.prq) のインポート] コマンドをクリックすると、プロジェクトに、前提条件に対して実行されるように設定されたファイルの種類に応じて、.msi パッケージまたは .exe パッケージが追加されます。また、.prq ファイルで構成された設定に基づいて、前提パッケージの各設定にデフォルトの値が自動的に設定されます。これらの設定は、スイート/アドバンスト UI またはアドバンスト UI プロジェクトでパッケージの設定を変更するのと同様、必要に応じて変更が可能です。

InstallShield 前提条件に依存関係がある場合 (つまり、1 つまたは複数の .prq ファイルが、スイート/アドバンスト UI またはアドバンスト UI プロジェクトに追加する InstallShield 前提条件で、依存関係として指定されている場合)、依存関係にある前提条件が [パッケージ] エクスプローラーに別のパッケージとして自動的に追加されます。

以前、スイート プロジェクトでは、.msi および .exe インストールをパッケージとして追加し、パッケージごとに、すべての条件と設定を手動で構成する必要がありました。

詳細については、「InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める」を参照してください。

スイート/アドバンスト UI またはアドバンスト UI プロジェクトに InstallScript インストールをパッケージとして含める機能; 新しいスイート/アドバンスト UI またはアドバンスト UI 固有 InstallScript イベントと関数

新しい InstallShield では、InstallScript インストールを、パッケージとして、スイート/アドバンスト UI またはアドバンスト UI プロジェクトに追加することができます。スイート/アドバンスト UI またはアドバンスト UI インストールで InstallScript パッケージが起動されたとき、それ自身のユーザー インターフェイス (UI) が表示され、InstallScript パッケージの UI は自動的に抑制されます。これにより、インストールで、最新の UI 使用感を提供できます。スイート/アドバンスト UI またはアドバンスト UI インストールでは、InstallScript パッケージに対して、進行状況も表示されます。

これらの変更を可能にするために、スイート/アドバンスト UI およびアドバンスト UI インストールでは、デフォルトで、いくつかのスイート/アドバンスト UI およびアドバンスト UI 固有の InstallScript イベントおよび関数が使用され、一部の標準 InstallScript イベントおよび関数は無視されます。

スイート/アドバンスト UI またはアドバンスト UI プロジェクトにおける新しい InstallScript パッケージのサポート

InstallShield では、InstallScript パッケージが次の要件を満たしたとき、InstallScript パッケージをスイート/アドバンスト UI またはアドバンスト UI プロジェクトに追加することができます。

- InstallScript パッケージは非圧縮です。
- InstallShield 2012 Spring 以降が、InstallScript パッケージまたはスイート/アドバンスト UI またはアドバンスト UI インストールを作成するために使われている。
- InstallScript パッケージで、イベント ベースのスクリプトが使用される。program...endprogram スタイルのスクリプトは使用しない。

詳細については、「InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する」を参照してください。

新しいスイート/アドバンスト UI およびアドバンスト UI 固有の InstallScript イベントと関数

InstallScript Setup.exe ファイルから起動される (スイート/アドバンスト UI またはアドバンスト UI インストールから起動されない) 標準の InstallScript インストールでは、ほとんどのイベントは OnShowUI イベントから直接呼ばれます。InstallScript パッケージを起動するアドバンスト UI またはスイート/アドバンスト UI またはアドバンスト UI インストールでは、OnShowUI がOnSuiteShowUI によって置き換えられます。インストール状態 (初回インストール、メンテナンス、またはアップデート) に応じて、OnSuiteShowUI では、OnFirstUIBefore や OnFirstUIAfter などの UI イベントが無視され、次のイベントが呼び出されます:

- **初回インストール**—OnSuiteInstallBefore、OnSuiteInstallAfter
- **メンテナンス**—OnSuiteMaintBefore、OnSuiteMaintAfter
- **アップデート**—OnSuiteUpdateBefore、OnSuiteUpdateAfter

InstallScript 言語には、InstallScript パッケージとそれを実行するアドバンスト UI またはスイート/アドバンスト UI インストールの間の対話を可能にする新しいスイート/アドバンスト UI およびアドバンスト UI 固有の関数がいくつか追加されました。たとえば、InstallScript には、InstallScript パッケージの情報をスイート/アドバンスト UI およびアドバンスト UI のデバッグ ログに記録できる関数、スイート/アドバンスト UI およびアドバンスト UI のプロパティを設定および取得する関数、およびスイート/アドバンスト UI およびアドバンスト UI インストールからのデータを InstallScript パッケージに渡す関数が新しく含まれています。

InstallScript インストールが、スイート/アドバンスト UI またはアドバンスト UI インストール内の InstallScript パッケージとして実行されているかどうかを判別するには、InstallScript コードで新しい SUITE_HOSTED 変数を使用します。

スイート/アドバンスト UI およびアドバンスト UI プロジェクトにおける、新しい InstallScript パッケージの条件の種類の確認

スイート/アドバンスト UI またはアドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、ターゲット システム上で評価するチェックを様々な種類から選択できます。新しい InstallScript パッケージの条件チェックを使うと、ターゲット システムで特定の InstallScript パッケージによってインストールされた製品の存在をチェックできます。特定の製品コードの条件チェックでは、製品バージョンなどのその他の情報もチェックできます。

詳しくは、次を参照してください:

- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類
- "InstallScript パッケージ" 条件の設定
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

スイート/アドバンスト UI およびアドバンスト UI プロジェクトにおける、パッケージ ファイルのダイナミック ファイル リンクのサポート

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、.msi、.msp、または .exe パッケージを追加または構成するとき、パッケージに、パッケージ ファイルの近くに保存されている追加ファイルが必要かどうかを示すことができます。たとえば、追加するパッケージが非圧縮の .msi パッケージである場合、その他のファイル (近くのサブフォルダーに保存されている .cab ファイルや非圧縮データ ファイルなど) をパッケージとファイルに含める必要が、場合によって、あります。

InstallShield では、追加のパッケージ ファイルにダイナミック リンクを使うことができます。ダイナミック リンクは、パッケージに必要な追加ファイルのリストがビルドとビルドの間に変更する可能性があるときに便利です。ソース フォルダーは常にビルドの前にスキャンされ、すべての新規または変更されたパッケージ ファイルが自動的にリリースへ組み込まれます。

InstallShield ではまた、ビルド時にダイナミック リンクに含めるまたは除外する追加ファイルを制御するフィルターを定義できます。ダイナミック リンクに対して定義したフィルター基準は、InstallShield で評価されるときの順番を変更することができます。アドバンスト UI またはスイート/アドバンスト UI インストールをビルドすると、ダイナミック リンクのフィルターを基に、常に適切な追加ファイルが含められます。

以前、追加ファイルには、スタティック リンクのみ使用することができました。ビルド間で追加ファイルのリストに変更が発生した時、パッケージ ファイルのリストは手動で更新する必要がありました。

詳細については、次を参照してください。

- アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージにおけるスタティック ファイルとダイナミック ファイルの違い
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトでダイナミック リンクを作成する
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ダイナミック リンク フォルダーのフィルターを定義する

スイート/アドバンスト UI またはアドバンスト UI のインストール中にエンドユーザー入力を検証したときの拡張フィードバックの生成

スイート/アドバンスト UI およびアドバンスト UI インストールでは、実行時、エンドユーザー入力の検証で、拡張フィードバックを生成することができるようになりました。スイート/アドバンスト UI またはアドバンスト UI プロジェクトの [ウィザード インターフェイス] ビューで提供されている様々なインターフェイス コントロールに、既存の "テキスト スタイル" 設定の下に新しいサブ設定 ("デフォルト"、"有効"、および "無効") が追加されました。これらのサブ設定を構成して、スイート/アドバンスト UI またはアドバンスト UI インストールで、異なる状況に応じて使用するテキスト スタイルを選択することができます。

スイート/アドバンスト UI またはアドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート

スイート/アドバンスト UI またはアドバンスト UI プロジェクトのパッケージ設定を構成するとき、スイート/アドバンスト UI またはアドバンスト UI インストールを /log コマンドライン パラメーターを使ってコマンドラインから起動する場合、新しい "ログの有効化" 設定を使って、ログ ファイルを生成するかどうかを指定できます。パッケージの種類 (.msi パッケージ、.msp パッケージ、またはその他の種類のパッケージ) に応じて、提供されている設定を構成して、その他の情報 (ログ ファイルが作成されるときにパッケージに渡すログ オプションなど) も指定できます。

スイート/アドバンスト UI またはアドバンスト UI の Setup.exe ファイルの新しい /log コマンドライン パラメーターを使用して、パッケージのログ ファイル用のディレクトリを指定することができます。パスが /log パラメーターを使用して指定されなかった場合、パッケージ ログ ファイルは %TEMP% ディレクトリに作成されます。

新しいスイート/アドバンスト UI またはアドバンスト UI インストールのプロパティ ISLogDir は、パッケージ ログ ファイルを含むディレクトリへのパスを格納します。

以前、スイート インストールの Windows Installer ベースのパッケージに対してログを有効化するには、ログ システム ポリシーを使用するか、または、MsiLogging プロパティを使用する方法しかありませんでした。

詳細については、次を参照してください。

- アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート
- アドバンスト UI およびスイート/アドバンストの Setup.exe コマンドラインのパラメーター
- アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス

InstallScript インストールでの 64 ビット コンポーネントのサポート

InstallScript プロジェクトで、InstallScript コードを変更せずに、ファイルを WINSYSDIR64 (64 ビット System32 フォルダーにマップする InstallScript 変数) にインストールするサポートと、レジストリ データを 64 ビットのレジストリの場所にインストールできるサポートが追加されました。これらの 64 ビットの場所にインストールする必要があるファイルまたはレジストリ データがある場合、それらのファイルとレジストリ データをコンポーネントに追加して、コンポーネントの新しい "64 ビット コンポーネント" 設定で [はい] を選択します。実行時に、コンポーネントの System32 ファイルに対して、ファイル システムのリダイレクトが自動的に無効になり、コンポーネントの 64 ビットのレジストリ データのリダイレクトが防止されます。

以前、ファイルを WINSYSDIR64 にインストールするには、その場所にインストールされたコンポーネントを含む機能に対して Installing と Installed のイベントを上書きする必要がありました。Installing イベントでは、WOW64FSREDIRECTION 定数を **Disable** 関数と一緒に使って、ファイル システムのリダイレクトを無効にし、Installed イベントでは、WOW64FSREDIRECTION を **Enable** 関数と一緒に使って、インストールの他の場所でのリダイレクトを再有効にする必要がありました。また、アンインストールでも、これらのファイルが正しく削除されるように、UnInstalling と UnInstalled イベントで、同じ無効と有効の操作が必要でした。

ファイル システムのリダイレクトが、WINSYSDIR64 へのインストールで無効になっていなかった場合、64 ビット Windows では、ファイル転送は自動的に 32 ビットの System32 フォルダー (SysWOW64) にリダイレクトされます。

また、以前のバージョンでは、レジストリ データを 64 ビット領域のレジストリにインストールする場合、InstallScript のレジストリ関数を使って、REGDB_OPTIONS に REGDB_OPTION_WOW64_64KEY が設定されているレジストリ データを作成する必要がありました。そしてこの後、REGDB_OPTION_USE_DEFAULT_OPTIONS を REGDB_OPTIONS と共に使用して、インストールの他の箇所に対して、レジストリのリダイレクトを再有効にする必要がありました。

64 ビットのレジストリの場所 (HKEY_LOCAL_MACHINE¥Software) へのインストールでレジストリのリダイレクトが無効になっていない場合、64 ビットの Windows では、レジストリ変更が、相当する 32 ビットのレジストリの場所 (HKEY_LOCAL_MACHINE¥Software¥Wow6432Node) へ自動的にリダイレクトされます。

今回より、製品のインストール時に、InstallScript インストールによって作成された InstallScript ログ ファイル (.ilg) では、新しい OPTYPE_FILE64 タイプを使って、ファイル システムのリダイレクトが無効のときにインストールされたファイルを識別するために使われています。InstallScript ログ ファイルでは、既存の OPTYPE_REGISTRY64 タイプが使われて、レジストリのリダイレクトが無効のときにインストールされたレジストリ データが識別されます。OPTYPE_FILE64 と OPTYPE_REGISTRY64 タイプを見るには、InstallShield のキャビネット & ログ ファイル ビューアーを使って .ilg ファイルを表示します。

詳細は、次を参照してください:

- 64 ビットのオペレーティング システムを 32 ビットInstallScript セットアップ ランチャーを使ってターゲットする
- コンポーネントの設定
- WINSYSDIR64
- REGDB_OPTIONS
- InstallShield キャビネット & ログ ファイル ビューアー

スイート/アドバンスト UI およびアドバンスト UI プロジェクトの新しいロケール タイプの条件チェック

スイート/アドバンスト UI またはアドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、ターゲット システム上で評価するチェックを様々な種類から選択できます。新しいロケール タイプの条件チェックを使うと、ターゲット システムで、1 つまたは複数のロケール関連の設定に一致するものがあるかどうかを確認することができます。

詳しくは、次を参照してください:

- アドバンスト UI およびスイート/アドバンスト UI プロジェクトでロケール条件を使用する
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類
- "ロケール" 条件の設定
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

スイート/アドバンスト UI およびアドバンスト UI におけるレイアウト編集用の新しいウィザード インターフェイス ツールバーと、スイート/アドバンスト UI プロジェクトにおける言語の切り替え機能のサポート

スイート/アドバンスト UI プロジェクトの [ウィザード インターフェイス] ビューでウィザード ページまたは二次ウィンドウを選択すると、ウィザード インターフェイスのプレビュー ペインのすぐ上に表示されるツールバーに、選択したページまたはウィンドウを変更できるボタンやコントロールが追加表示されます。また、ウィザード ページまたは 2 番目のウィンドウでコントロールを選択したときも、[ウィザード インターフェイス] ビューにこのツールバーが表示されます。

新しいツールバーには、インストールのユーザー インターフェイスにラベル、テキスト ボックス、チェックボックス、その他のコントロールを追加できるボタンが提供されています。また、ツールバーには、選択したコントロールの整列、サイズ調整、位置調整を簡単に行うことができるボタンも提供されています。スイート/アドバンスト UI プロジェクトの新しいツールバーにある [デフォルト言語] リストでは、このビューのウィザード ページおよび二次ウィンドウで表示する文字列を、プロジェクトで使用する別の言語の文字列に切り替えることができます。

詳細については、次を参照してください。

- ウィザード ページまたは 2 番目のウィンドウにコントロールを追加する
- [ウィザード インターフェイス] ビューのツールバー

スイート/アドバンスト UI プロジェクトに言語を追加するサポート

InstallShield Premier Edition には、35 ヶ国語のデフォルト ランタイム文字列がサポートされています。サポート対象の言語をスイート/アドバンスト UI プロジェクトに追加すると、その言語は InstallShield 内の様々な言語関連設定で使用可能となります。さらに、InstallShield はプロジェクトにその言語に翻訳された文字列エントリを追加します。それらは、デフォルト ウィザード ページ、メッセージ、およびその他のエンド ユーザー インターフェイス要素の文字列エントリです。

新しい InstallShield のスイート/アドバンスト UI プロジェクトでは、新しい言語ウィザードを使って 35 ヶ国語以外のサポートされていない言語も追加することができるようになりました。サポートされていない言語とは、デフォルトのランタイム文字列が全く翻訳されていない言語です。サポートされていない言語をスイート/アドバンスト UI プロジェクトに追加すると、その言語は、プロジェクト内の様々な言語関連設定で使用可能となります。さらに、InstallShield は新しく追加されたサポートされていない言語の文字列用のプレースホルダーとして、プロジェクトのデフォルト言語の文字列を使用します。サポートされていない言語に翻訳済みの文字列を提供するには、[文字列エディター] ビューを使用します。

スイート/アドバンスト UI プロジェクトで新しい言語ウィザードを起動するには、ツール メニューで、[新しい言語の追加] をクリックします。

詳しくは、次を参照してください:

- 新しい言語ウィザード
- InstallShield 実行時の言語サポート

実行時にターゲット システムでスケジュール タスクを作成および構成できる機能

InstallShield に、ターゲット システムで、実行時に、Windows のタスク スケジューラを使って作成するタスクを構成できる [スケジュール タスク] ビューが追加されました。ビューでは、タスクに対して起動するファイルや、開始日付と時刻などの情報を指定できます。起動するファイルは、インストールに含まれているファイルでも、ターゲット システムに既に存在するファイルでも可能です。

この機能は、基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- スケジュール タスク
- [スケジュール タスク] ビュー

新しい FlexNet Connect 13.03 再配布可能ファイル

InstallShield は、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで FlexNet Connect 13.03 をサポートします。InstallShield の [アップデート通知] ビューで、2 つの FlexNet Connect 13.03 マージ モジュール (Common Software Manager が含まれているマージ モジュールと、含まれていないマージ モジュール) のどちらかを含みます。

強化機能

InstallShield には、以下のような強化機能が搭載されています。

オペレーティング システムに関する InstallScript 言語の強化

次の構造メンバーと定義済み定数が InstallScript 言語に追加されました:

- **SYSINFO.WINNT.bWin8** – 新しい SYSINFO 構造メンバーです。オペレーティング システムが Windows 8 または Windows Server 8 の場合、この値は TRUE です。
- **ISOSL_WIN8** – **FeatureFilterOS** 関数と SYSINFO 構造変数と共に使用できる新しい定義済み定数です。これは、ターゲット システムが Windows 8 または Windows Server 8 を実行中であることを示します。

詳細については、「FeatureFilterOS 関数と SYSINFO 構造変数」を参照してください。

オートメーション インターフェイスの強化機能: Windows 8 と Windows Server 8 用の OSFilter プロパティの値

オートメーション インターフェイスで、ISWiComponent と ISWiRelease オブジェクトの OSFilter メンバーと共に、以下の定数を使用できます:

eosWin8 = &H4000000 (67108864) È これらの定数は Windows 8 および Windows Server 8 用です。

また、eosAll 定数の値が &3D100D0 (64028880) から &7D100D0 (131137744) に変更されました。

OSFilter メンバーは、InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトで ISWiComponent オブジェクトに適用します。OSFilter メンバーは、InstallScript、および InstallScript オブジェクト プロジェクトで ISWiRelease オブジェクトに適用します。

詳細は、次を参照してください:

- ISWiComponent オブジェクト
- ISWiRelease オブジェクト

スイート/アドバンスト UI およびアドバンスト UI プロジェクトで条件を簡単に移動できる機能

スイート/アドバンスト UI または アドバンスト UI プロジェクトにおいて、終了、検出、対象、または機能条件の 1 つ以上の条件ステートメントがある場合、条件ステートメントを移動して、条件を並べ替えたり、条件ツリーの階層を変更したりできます。たとえば、None 条件グループにプラットフォーム条件ステートメントがある場合で、None 条件グループが All 条件グループの中にあるとき、プラットフォーム条件ステートメントを左に移動させて、All 条件グループの一部となるように変更できます。

条件ステートメントまたはグループを移動するには、移動させるアイテムの設定にある新しい [条件を移動] ボタンをクリックしてから、適切なオプション (上に移動、下に移動、左に移動、または右に移動) をクリックします。

以前は、手作業で新しい条件ステートメントを新しい場所に作成してから、古い場所にある条件ステーメントを削除しなくてはなりませんでした。あるいは、テキスト エディターの .issuite ファイルを編集して、条件ステートメントの順番を変更する方法もあります。

詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。

スイート/アドバンスト UI およびアドバンスト UI プロジェクトにおける新しい拡張条件のサポートと機能強化された条件設定

スイート/アドバンスト UI およびアドバンスト UI プロジェクトで、新しい拡張条件機能が使用できるようになりました。このタイプの条件を利用すると、ターゲット システム上で、作成した C/C++ DLL を参照して、自作のカスタム条件があるかどうかを確認できます。

また、スイート/アドバンスト UI およびアドバンスト UI プロジェクトで提供されている条件に関連する設定の多くが機能強化され、条件ステートメントを簡単にビルドできるようになりました。たとえば、いくつかの条件設定の 1 つで、手動で製品コード、アップグレード コード、パッチ コード、またはその他のタイプのデータの入力を試みる代わりに、設定にある新しい省略記号ボタン (...) を使うことができます。ボタンをクリックすると、適切なパッケージを参照し選択できる参照ダイアログ ボックスが表示されます。パッケージを選択すると、スイート/アドバンスト UI またはアドバンスト UI プロジェクトの設定に、そのパッケージの情報が正しく設定されます。

対象となるパッケージ条件を構成するとき、適切なパッケージ GUID を手動で入力する代わりに、スイート/アドバンスト UI またはアドバンスト UI プロジェクトのパッケージ リストから選択できます。

詳細については、次を参照してください。

- アドバンスト UI またはスイート/アドバンスト UI インストールのカスタム条件を作成する
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、拡張条件 DLL を作成する
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトに拡張条件 DLL を追加する
- "拡張条件" の設定

スイート/アドバンスト UI またはアドバンスト UI プロジェクトのウィザード ページまたはウィンドウのコントロールに対して行う検証とアクションを構成における機能強化

スイート/アドバンスト UI またはアドバンスト UI プロジェクトで、ウィザード インターフェイスで表示できる様々なコントロールの "検証" 設定と "アクション" 設定が強化され、検証の定義と様々のアクションのトリガをより簡単に行えるようになりました。これらの設定に、これらの設定を使って入力可能なサンプル ステートメントのドロップダウン リストが追加されました。また、依然同様、これらの設定をテキスト ボックスとして使用し、ステートメントを手入力することも可能です。たとえば、"検証" 設定では、エンドユーザーがコントロールでシリアル番号を入力したときに合致する必要があるフォーマットを指定することができます。検証ステートメントをすべて手入力する代わりに、"検証" 設定でマスク タイプの検証を選択して、この設定のデフォルト フォーマットをオーバーライドすることもできます。

同様に、"アクション" 設定では、例として、ボタン コントロールの印刷アクションを定義できます。つまり、エンド ユーザーが [印刷] ボタンをクリックすると、[印刷] ダイアログ ボックスが開いて、エンド ユーザーが使用許諾契約を印刷できるようになります。今回より、アクション ステートメントをすべて手入力する代わりに、"アクション" 設定で印刷アクションを選択して、そのファイル名を、印刷されるファイルの名前でオーバーライドすることができます。

"検証" 設定と "アクション" 設定のドロップダウン リストに、自分で作成し、[サポート ファイル] ビューを使ってプロジェクトに追加した C/C++ DLL ファイルが表示されるようになりました。今回より、この機能によって、ウィザード インターフェイスの様々なコントロールに対して、自作の検証やアクションをトリガできます。

詳しくは、次を参照してください:

- ウィザード ページまたはウィンドウに含まれるコントロールの検証を構成する
- ウィザード インターフェイス内の要素のアクションを構成する

スイート/アドバンスト UI およびアドバンスト UI リリースのパッケージにカスタム フォルダー名を使用できる機能

スイート/アドバンスト UI またはアドバンスト UI インストールをビルドしたとき、インストールに含まれている各パッケージに対してフォルダーが 1 つ作成されます。これらのフォルダーは、スイート/アドバンスト UI またはアドバンスト UI のセットアップ ランチャーと同じフォルダー内で作成されます。デフォルトで、各フォルダーの名前には、InstallShield で生成された GUID が使用されます。

InstallShield の [パッケージ] ビルドでは、今回から、これらのパッケージ フォルダーに対して、この GUID 名をオーバーライドして、ユーザーに分かりやすい名前を使用できるようになりました。このビューでフォルダーのカスタム名を入力するには、フォルダーの名前をカスタマイズするパッケージの下で、[パッケージ ファイル] フォルダーを見つけます。フォルダーを右クリックし、[名前を変更] をクリックして、新しい名前を入力します。複数のフォルダー名をカスタマイズする場合、各フォルダー名に異なる名前を使用してください。

以前、各フォルダーの名前には GUID のみが使用され、名前のカスタマイズはサポートされていませんでした。

手順については、「アドバンスト UI または スイート/アドバンスト UI インストールのパッケージにカスタム フォルダー名を使用する」を参照してください。

スイート/アドバンスト UI およびアドバンスト UI プロジェクトで他のプロパティを参照する値を持つプロパティにおける新しい "フォーマット済み" サポート

スイート/アドバンスト UI またはアドバンスト UI プロジェクトの [プロパティ マネージャー] ビューで定義された各プロパティに、新しい "フォーマット済み" チェック ボックスが追加されました。この新しいチェック ボックスを使って、実行時に、プロパティの値で参照されるプロパティを解決し、これらのプロパティの値で置換するかどうかを示すことができます。

実行時に、角かっこで囲まれたプロパティ ([PropertyName] など) を置き換える場合、このチェック ボックスを選択します。角かっことそのコンテンツをそのままにしておく場合、このチェック ボックスをクリアします。

詳細については、「[プロパティ マネージャー] ビュー」を参照してください。

起動者マニフェストがあるスイート/アドバンスト UI およびアドバンスト UI インストールに対して権限の昇格が必要なダウンロードされたパッケージの UAC プロンプトのタイミングが改善されました

起動者マニフェストがあるアドバンスト UI またはスイート/アドバンスト UI インストールをビルドして、そのインストールに含まれているターゲット システムでダウンロードし起動する必要があるパッケージの中に、"昇格された権限が必要" 設定で [はい] が選択されているものが存在するとき、UAC プロンプトは、エンドユーザーが [インストール] ボタンをクリックした直後 (ダウンロードが開始される前) に表示されます。

以前、UAC プロンプトは、ダウンロードが発生した後に表示されていました。このため、パッケージのステージングに時間がかかっていた場合 (パッケージのダウンロードに時間がかかっていた場合など)、エンドユーザーが [インストール] ボタンをクリックした瞬間から、Windows でパッケージの昇格を求める UAC プロンプトが表示される瞬間のまでに大きなギャップが開いていた可能性があります。エンドユーザーが速やかに同意しなかった場合、または認証情報を入力しなかった場合、UAC プロンプトがタイムアウトになり、インストールが失敗していました。

以前の一部のケースでは、UAC プロンプトがエンドユーザーが [インストール] ボタンをクリックした直後に表示されていたため、管理マニフェストを使うことで、この問題を回避することができていましたが、この方法では、インストール全体に昇格された権限が使用されていました。

スイート/アドバンスト UI およびアドバンスト UI インストールのウィザード インターフェイスで使用されているテキストの配置を指定できるサポート

InstallShield には、スイート/アドバンスト UI およびアドバンスト UI プロジェクトのウィザード インターフェイスで使用されているテキスト属性 (テキストの色、サイズ、フォント名など) を定義する多くのビルトイン テキスト スタイルが含まれています。アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [ウィザード インターフェイス] ビューを使って、これらのビルトイン スタイルの任意の設定を編集、または独自のスタイルを定義することができます。

各ビルトインまたはカスタム テキスト スタイルには、特定のスタイルを使用するコントロール内のテキストに対して配置のタイプを選択できる "テキスト配置" 設定が新しく追加されています。

詳細については、「スタイル、ブラシ、コントロール テーマ、および最小ウィザード ページを使ってウィザード インターフェイスをカスタマイズする」を参照してください。

スイート/アドバンスト UI およびアドバンスト UI インストールのウィザード インターフェイスの機能拡張されたコンボ ボックス コントロール

スイート/アドバンスト UI およびアドバンスト UI プロジェクトで、ウィザード ページまたは二次ウィンドウに、コンボボックス コントロールを追加することができます。このタイプのコントロールには、デフォルトで 2 つのコントロールがあります:

- 定義された値を既に持つドロップダウン リストを含むボックス
- エンドユーザーがカスタム値を入力できるテキスト ボックス

以前のバージョンでは、新しいコンボボックス コントロールを追加したとき、コントロールにドロップダウン リストが含まれていましたが、テキストボックスは提供されていなかったため、エンドユーザーはカスタム値を入力することができませんでした。

このコントロールをテキストボックスのないドロップダウン リストに変更するには (つまり、エンドユーザーが定義済みの値を選択できるが、カスタム値を入力できるようにする場合)、このコントロールの CBS_DROPDOWNLIST スタイルを False に設定します。

スイート/アドバンスト UI およびアドバンスト UI ウィザード インターフェイスの Aero フォーマットの拡張と変更

スイート/アドバンスト UI およびアドバンスト UI インストールの Aero フォーマットのウィザード ページが、一部拡張および変更されました。

- ウィザード ページのヘッダーおよびナビゲーション領域が、Aero ガラス効果 (半透明) で表示されるようになりました。以前、ウィザード ページのキャプション バーのみ、Aero ガラス効果で表示されていました。
- Aero フォーマットのウィザード ページでは、今回より、Wizard 97 フォーマットのウィザード ページと同じレイアウトが使用されています。つまり、Aero フォーマットのウィザード ページのキャプション バーの縦幅が低くなり、Wizard 97 フォーマット ウィザード ページのキャプション バーと同じ縦幅が適用されています。また、Aero フォーマットのウィザード ページの [戻る] ボタンが、Wizard 97 フォーマットのウィザード ページ同様、ナビゲーション領域で表示されるようになりました。以前、[戻る] ボタンは、Aero フォーマットのウィザード ページのキャプション バーの左上に表示されていました。

設定から文字列値とその ID を削除できる機能

InstallShield のビュー内にある設定で値を入力するときに、その値がエンド ユーザーに表示されるテキスト文字列である場合、自動的にその設定の文字列 ID が使用されます。InstallShield は、文字列値の前に中括弧で囲まれた文字列 ID を配置します。これらの種類の設定には、**[この文字列参照を削除]** ボタンが追加されました。

設定から文字列 ID とその値を削除する場合、この新しいボタンをクリックします。このボタンを使って、設定のエントリをクリアできます。文字列 ID とその値をプロジェクトから削除するには、[文字列エディター] ビューを使います。

InstallShield 2012 SP1 の新しい機能

強化機能

InstallShield には、以下のような強化機能が搭載されています。

ソフトウェア識別タグのデジタル署名サポート

プロジェクトにソフトウェア ID タグを含めて [リリース] ビューで .pfx ファイルを使ってリリースに署名を行うことを設定すると、InstallShield がビルド時にタグをデジタル署名します。タグ ファイルに署名するためには、.NET Framework 2.0 以降をビルドマシンにインストールする必要があります。

詳細については、「製品のソフトウェア識別タグを含める」を参照してください。

InstallShield 2012 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

複数のパッケージを実行するスイート インストールを作成できる機能、最新のカスタマイズ可能なエンド ユーザー インターフェイス、ハイブリッド 32 ビット/64 ビット インストールをビルドできる機能

今回より InstallShield Premier Edition では、条件付きで複数のインストールを実行し、必要に応じてターゲット システムに Windows Installer パッチ (.msp) を適用することが可能な次世代のセットアップランチャー (Setup.exe) を使ったスイート インストールをビルドすることができます。この機能は、新しいスイート プロジェクト タイプで使用できます。スイート インストールは Windows XP 以降および Windows Server 2003 以降が搭載されたシステムで実行します。

機能の主な特徴は以下の通りです。

複数のインストールを単一のインストールとしてパッケージできる機能

新しいスイート プロジェクト タイプには、以下の種類のパッケージから 1 つ以上を指定できる [パッケージ] ビューが含まれています。

- ターゲット システムで実行する Windows Installer ベースおよび Windows Installer ベース以外のインストールを含む実行可能ファイル (.exe)
- ターゲット システムで実行する Windows Installer パッケージ (.msi)
- ターゲット システムに適用する Windows Installer パッチ (.msp)

また [パッケージ] ビューを使って、トランザクション処理を使って実行する .msi および .msp パッケージを含めることもできます。これは Windows Installer 4.5 以降の機能です。パッケージを連鎖させることによって、単一のトランザクションとして処理します。各スイート インストールには、複数の個別のトランザクションを含めることができます。トランザクションに含まれる 1 つまたは複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は現在のトランザクションに含まれる連鎖パッケージ全てのロールバックを開始して、システムを以前の状態に復元します。

スイート インストールは定義された条件および [パッケージ] ビューのパッケージでリストされている順番に従って、実行時に適切なパッケージを起動します。

最新のカスタマイズ可能なインストールのユーザー インターフェイス、およびスイート Setup.exe ウィザード ページをカスタマイズできる新しいエディター

InstallShield のスイート プロジェクト タイプでは、カスタマイズが可能な全く新しい、最新デザインのビルトイン ウィザード ページを使った、エンド ユーザー インターフェイスをインストールに含めることができます。このプロジェクトの種類で使用できる新しいウィザード ページ エディターでは、必要に応じてページを追加、シーケンス、または削除することができます。また、様々な異なる種類のコントロールを追加または削除することで任意のページのレイアウトを編集することもできます。

64 ビットおよび 32 ビットの Windows Installer パッケージを単一のインストールに組み合わせるサポート

より多くのユーザーが Windows 64 ビット バージョンに移行する中、32 ビット システム上では 32 ビットの場所に、また 64 ビット システムでは 64 ビットの場所にインストールする単一のインストールを、今すぐまたは近い将来に配布する必要が出てくるでしょう。スイート プロジェクト タイプを使うと、32 ビット パッケージと 64 ビット パッケージの両方を単一のスイート インストールに含めて、各ターゲット システム上に適切なパッケージのみを実行することができます。以前は、2 つの個別のインストール (32 ビット システム用および 64 ビット システム用) を配布するか、カスタム起動ツール、ブートストラップ アプリケーション、または InstallShield インストールを作成する必要がありました。

スイート パッケージ全体のステータスを表示する単一の進行状況バーを表示するサポート

スイート インストールの進行状況ウィザードページで表示される進行状況バーには、スイート パッケージ全体のステータスが表示されます。この統合された進行状況バーによって、エンド ユーザーはスイート インストール全体の進行状況をはっきりと視覚的に確認することができます。エンド ユーザーに対して統合された進行状況バーのみが表示されるように、ユーザー インターフェイスを非表示に (つまり、サイレントで実行) するコマンドライン パラメーターを指定した .exe インストール、.msi パッケージ、および .msp パッチのみを含めてください。

スイート インストールにおけるオプションの [プログラムの追加と削除] エントリ

スイート プロジェクトでは、スイート インストールに [プログラムの追加と削除] を含めるかどうかを指定できます。このエントリを使って、エンド ユーザーは必要に応じてスイートの管理、変更、または削除を行うことができます。スイート プロジェクトの [一般情報] ビューには、適切な動作を指定できる "[プログラムの追加と削除] エントリの表示" 設定があります。

スイート全体に対して単一のエントリのみを表示する場合、スイート プロジェクトに含めるパッケージからのエントリを非表示にしてください。

ユーザー インターフェイス無しでスイート インストールを実行するサポート

エンド ユーザーは、ユーザー インターフェイスを使って、またはユーザー インターフェイスなしのサイレントでスイート インストールを実行することができます。サイレント インストールは、エンド ユーザーによる操作が不要で、実行時にランタイム ウィザード ページを使った情報の入力や監視が不要です。

35 すべてのサポート言語に新しく追加された "デフォルト Setuup.exe ユーザー インターフェイス" 設定を使った、スイート ランタイム文字列の編集機能

スイート プロジェクトのビルトイン ウィザード ページに表示されるすべてのデフォルト文字列について、InstallShield でサポートされているすべての 35 ランタイム言語に対してその翻訳が用意されています。これらすべてのスイート文字列は、スイート プロジェクトの [文字列エディター] ビューに表示されます。[文字列エディター] ビューでは、その他のプロジェクトの種類で提供されているのと同じ堅牢な機能を使用することができるため、実行時、スイート インストールの処理中に表示されるテキスト文字列すべてを 1 ヶ所でまとめて制御することができます。

必要なときに必要なパッケージのみをダウンロードできる機能を搭載したコンパクトなベース Setup.exe ファイル

スイート プロジェクトには、スイート インストールに含まれる各パッケージのランタイム ソースの場所を指定できる柔軟なオプションが含まれています。スイート プロジェクトに含めるパッケージを定義するとき、個別のパッケージの場所を指定できます。選択可能なオプションは、以下のとおりです。

- Web 上で、必要に応じて Setup.exe によるダウンロードが可能
- Setup.exe に埋め込み、必要に応じてターゲット システムに展開
- 非圧縮ファイルでスイートのソース メディアに格納

スイート インストールで使用されるベース Setup.exe ファイルは、基本の MSI、InstallScript MSI、および InstallScript インストールで使用されるベース Setup.exe ファイルよりもはるかにサイズが小さいです。このため、エンド ユーザーはコンパクトなスイート Setup.exe ファイルを短時間でダウンロードすることができ、Setup.exe ファイルが必要に応じて 1 つ以上のパッケージをダウンロードおよび起動します。

詳細については、次を参照してください。

- アドバンスト UI およびスイート/アドバンスト UI インストールの作成
- .msi パッケージ、.msp パッチ、トランザクションをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する

- 実行可能パッケージ (.exe) をスイート/アドバンスト UI プロジェクトに追加する
- ウィザード インターフェイスを使って作業する
- アドバンスト UI およびスイート/アドバンストの Setup.exe コマンドラインのパラメーター
- アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

開発プロセスを再利用/分担するためのインストール プロジェクトのモジュール化が新しく更新され、サポートが拡張されました

InstallShield に新しいプロジェクト タイプ、DIM (*デベロッパー インストール マニフェスト*) が追加されました。DIM プロジェクトは、機能サイズのプロジェクトで、インストール パッケージの別個に分かれている部分を構成する製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、および要素など、関連するアイテムを集めたものです。以下は、DIM を利用した時のいくつかの利点です:

- DIM では、基本の MSI プロジェクトで提供されている機能とほぼ同じ機能がサポートされています。このため、DIM の作成者には、インストールの部分開発を行うために必要な柔軟性がすべて提供されています。
- リリース エンジニアは、DIM を複数の基本の MSI プロジェクトで繰り返し使用できるため、非常に効率的です。
- DIM を利用することにより、複数のチーム メンバーが、インストールの開発を同時に携わることができます。各ソフトウェア開発者またはチームメンバーは、異なる DIM について個別で作業することができ、リリース エンジニアは、1 つまたは複数の基本の MSI プロジェクトでそれらを参照することができます。

DIM を作成したあと、2 つある方法のいずれかで基本の MSI プロジェクトに追加することができます:

- **参照** – [セットアップのデザイン] ビューまたは [DIM リファレンス] ビューから、DIM プロジェクトの参照を基本の MSI プロジェクトに追加することができます。この方法を使用すると、DIM の要素は、ビルド時に基本の MSI プロジェクトにマージされます。基本の MSI インストールがビルドされるたびに、DIM プロジェクトの最新バージョンが参照され、生成されたインストールに含められます。この方法は、最も頻繁に使われる方法です。
- **インポート** – 新しい DIM のインポート ウィザードを使用して、DIM プロジェクトを基本の MSI プロジェクトにインポートできます。この方法は、デザイン時に、DIM データを基本の MSI プロジェクトにマージするときに行う目的で一回切り行われ、一度行われると元に戻すことはできません。

この新しくデザインが変わり拡張された DIM のサポートにより、ユーザーは、InstallShield Collaboration という別のツールを使って DIM を作成し、その DIM ファイルを InstallShield の基本の MSI プロジェクトにインポートしなくてはならないという手間がなくなりました。新しい DIM のサポートは、これまでのサポートに比べより堅牢になりました。新しい DIM プロジェクトによって、基本の MSI プロジェクトで使用可能な柔軟性の高いオーサリング機能が事実上すべて利用可能になりました。たとえば、新しい DIM プロジェクトでは、コンポーネントの作成においてフル制御が可能になり、これにより、コンポーネントを新しい DIM プロジェクトに追加したり、コンポーネントのキー ファイルを設定したり、コンポーネントの設定を構成できるようになりました。新しい DIM プロジェクトではまた、IIS Web サイトの構成も可能になりました。InstallShield Collaboration では、コンポーネントのデザインにおいて、同レベルの柔軟性が提供されておらず、また、IIS Web サイトを構成するためのビルトイン サポートも提供されていませんでした。

DIM プロジェクトの作成機能は、InstallShield Premier Edition で使用できます。このサポートは、新しいコラボレーション アドオン、InstallShield Developer Installation Manifest Editor でも提供されています。DIM ファイルを基本の MSI プロジェクトに追加する機能は、InstallShield Premier Edition で提供されています。

詳細は、次を参照してください:

- 開発プロセスを再利用/分担するためのインストール プロジェクトのモジュール化
- プロジェクトに DIM を追加する
- インストール プロジェクトに DIM を統合するための適切な方法を判別する
- DIM でパス変数を使用する
- [DIM リファレンス] ビュー
- DIM のインポート ウィザード

Internet Explorer 9、SQL Server 2008 R2 Native Client、Windows Identity Foundation、その他の再配布可能ファイル用の新しい InstallShield 前提条件

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる、いくつかの新しい InstallShield 前提条件が含まれています:

- Internet Explorer 9.0
- Microsoft SQL Server 2008 R2 Native Client 10.50.1600.1
- Windows Identity Foundation
- Microsoft VSTO 2010 Runtime (x64)
- Microsoft Office 2010 PIA (この前提条件は Microsoft Office 2010 Primary Interop Assemblies をインストールします。この前提条件を使用するには、マイクロソフトの Web サイトから PrimaryInteropAssembly.exe ファイルをダウンロードおよび実行して、.msi ファイルを抽出します。)

64 ビット依存関係スキャンのサポート

InstallShield の依存関係スキャナー (スタティック スキャン ウィザードおよびダイナミック スキャン ウィザード) には、プロジェクトに含まれる 64 ビット ファイルの 64 ビット依存関係を識別できるサポートが追加されました。Windows Vista 以降の 64 ビット バージョン、または Windows Server 2008 以降の 64 ビット バージョンで InstallShield を使用する場合、スキャナーは 64 ビット 依存関係を検出することができます。ウィザードでは、プロジェクト内で検出された依存関係の可能性がある各ファイルについて、それらを含めるかどうかを指定することができます。

また、Windows Vista 以降の 64 ビット バージョン、または Windows Server 2008 以降の 64 ビット バージョンで InstallShield を使用していて、以下のどちらかのビルトイン処理によって依存関係を検出する場合、InstallShield はプロジェクトに含まれる 64 ビット .NET アセンブリの 64 ビット 依存関係をスキャンすることができます。

- スタティック スキャン ウィザードを使って、64 ビット .NET アセンブリの依存関係の可能性があるファイルをオンデマンドで識別できます。このウィザードで検出された依存関係のリストが表示され、それぞれをプロジェクトに含めるかどうかを指定することができます。
- コンポーネントの "ビルド時に .NET スキャン" 設定を使って、プロジェクトをビルドする度に InstallShield が 64 ビット .NET アセンブリの依存関係を識別するかどうかを指定できます。InstallScript プロジェクトの場合、コンポーネントの ".NET アセンブリ" 設定を [ローカル アセンブリ] に設定しなくてはなりません。InstallShield がビルド時に不足している可能性のある依存関係を検出した場合、InstallShield が生成するリリースにはそれが組み込まれます。

64 ビット依存関係の 64 ビット ファイルをスキャンするためには、InstallShield が、64 ビット オペレーティング システムにインストールされていなくてはなりません。Windows の 32 ビット バージョンで InstallShield を使用する場合、これらのビルトイン スキャンはプロジェクトに含まれる 32 ビット ファイルの 32 ビット依存関係のみを検出できます。プロジェクトに 64 ビット ファイルが含まれている場合、必要に応じてプロジェクトに依存関係を手動で追加することができます。

詳しくは、次を参照してください:

- 32 ビットと 64 ビット システムにおけるインストールの開発およびビルドの違い
- 64 ビット依存関係のスキャン
- アプリケーションの依存関係を識別する
- 依存関係の 64 ビット .NET アセンブリをスキャンする
- .NET アセンブリのプロパティおよび依存関係を識別する

64 ビットの場所にあるファイル、フォルダー、およびレジストリ キーのアクセス許可を設定するサポート

InstallShield では 64 ビットの場所にあるファイル、フォルダー、およびレジストリ キーのアクセス許可を設定するためのサポートが提供されています。サポートは、使用するプロジェクトの種類によって異なります。

Windows Installer ベースのプロジェクトにおけるカスタム InstallShield 処理の使用

カスタム InstallShield 処理を使ってファイル、フォルダー、およびレジストリ キーのアクセス許可を設定する場合、今回より、これらのアイテムが 64 ビットの場所にある場合にアクセス許可を設定できます。これには、レジストリの 64 ビットの場所および 64 ビット システム上の 64 ビット System32 フォルダも含まれます。フォルダーまたはレジストリ キーは、64 ビットとマークされたコンポーネントに含めなくてはなりません (つまり、"64 ビット コンポーネント" 設定で [はい] が選択されている)。以前、コンポーネントが 64 ビットとマークされている場合、アクセス許可を設定することはできず、ランタイム エラーが表示されました。

このカスタム InstallShield 処理サポートは、基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。[一般情報] ビューの "ロックダウンの設定方法" 設定で、アクセス許可の設定に使用する方法 (カスタム InstallShield 処理または従来型の Windows Installer 処理のいずれか) を指定できます。

InstallScript ベースのプロジェクトおよび Windows Installer ベースのプロジェクトにおける InstallScript 関数 SetObjectPermissions の使用

REGDB_OPTION_WOW64_64KEY オプションが有効な場合に、InstallScript 関数 **SetObjectPermissions** を使って 64 ビット レジストリ キーのアクセス許可を設定したとき、関数がそのアクセス許可を正しく設定します。REGDB_OPTION_WOW64_64KEY オプションの有効/無効に関わらず、**SetObjectPermissions** で強制的に 64 ビット レジストリ キーのアクセス許可を設定するには、新しい定数 IS_PERMISSIONS_OPTION_64BIT_OBJECT を使用します。定数 IS_PERMISSIONS_OPTION_64BIT_OBJECT は **SetObjectPermissions** の nOptions パラメーターに渡します。この定数を 32 ビット ターゲット システムで渡すことはできません。

SetObjectPermissions を呼び出して 64 ビットの System32 フォルダーにあるファイルまたはフォルダーのアクセス許可を設定するとき、WOW64FSREDIRECTION 定数を使ってシステム リダイレクトを無効とした場合、関数がアクセス許可を正しく設定します。ファイル システム リダイレクトが無効でない場合、アクセス許可は設定できません。

InstallScript および InstallScript MSI プロジェクトでは InstallScript イベントの **SetObjectPermissions** 関数を使用できます。この関数は、InstallScript、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクト タイプで、InstallScript カスタム アクションでも使用できます。

詳細については、「SetObjectPermissions」を参照してください。

COM 抽出機能の強化

InstallShield では、COM 抽出時に新しい監視方式をサポートします。Windows Vista 以降のシステムまたは Windows Server 2008 以降のシステム上で、InstallShield を使用している場合、この新しい方式がデフォルトとなります。この方法は、カーネル ドライバーを使って、ビルド時のダイナミック COM 抽出中、およびデザイン時のスタティック COM 抽出中に変更されたレジストリ領域を監視します。この新しい方式は、DLL が既存のレジストリ エントリを読み込んでビルド マシンへの変更を妨げる以前の方法の利点を組み合わせたものです。

必要な場合、UseAPIRegistryHooks レジストリ値 (32 ビット マシンの場合は HKEY_LOCAL_MACHINE¥SOFTWARE¥InstallShield¥RegSpy レジストリ キーに含まれる、64 ビット マシンの場合は HKEY_LOCAL_MACHINE¥SOFTWARE¥Wow6432Node¥InstallShield¥RegSpy レジストリ キーに含まれる) の値データを設定して 3 つの異なる COM 抽出方式を切り替えることができます。使用可能な REG_DWORD 値データ:

- 0—API フックを使って、既存 DLLのレジストリ エントリを読み取ります。
- 1—レジストリのリダイレクトを使って、ビルド マシン上の登録済み DLL への変更を防止します。値を設定しなかった場合、これが Windows XP および Windows Server 2003 システム上でのデフォルト動作となります。
- 2—新しいカーネル モードの監視を使って、2 つのメソッドの両方の利点を組み合わせます。値が設定されていない場合、これが Windows Vista 以降および Windows Server 2008 以降のシステム上でのデフォルト動作となります。

この機能は、基本の MSI、DIM、InstallScript MSI、および マージ モジュール プロジェクト タイプに適用します。

Adobe Reader 10、Internet Explorer 9、および Microsoft Office 用の定義済みシステム検索

InstallShield に新しい定義済みシステム検索が追加されました:

- Adobe Reader 10
- Internet Explorer 9
- Microsoft Office 2010
- Microsoft Office 2007
- Microsoft Office 2003

インストールでこれらの 1 つ以上が必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、これらのシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

この機能は、基本の MSI および InstallScript MSI プロジェクトに適用します。

ソフトウェア識別タグ機能のサポート

ISO/IEC 19770-2 は、ソフトウェア識別タグを作成するための国際規格です。ソフトウェア識別タグは、製品名、製品エディション、パブリッシャーなど、ソフトウェアに関する説明的な情報を含む XML ベースのファイルです。ソフトウェア資産管理ツールは、企業でインストールされているソフトウェアについての正確なアプリケーション ID を提供する目的で、タグ内のデータを収集するツールです。

ソフトウェア識別タグ機能は、業界標準として現在進化しつつあるもので、この機能により、独立系ソフトウェア ベンダーは、ソフトウェア資産管理およびライセンス最適化イニシアチブに役立つ情報を提供する、より洗練されたアプリケーションを作成することができます。製品のインストール パッケージに識別タグを持たせることで、顧客は、インストールした製品の内部的使用状況を監視できるツールが使えるようになります。これにより、顧客がソフトウェア会社などから入手して精神のライセンスの数を管理および最適化することができるようになり、ライセンス契約内容に違反するリスクがなくなります。

InstallShield の [一般情報] ビューには、製品の識別タグを作成するために必要な情報を指定するための新しい設定がいくつかあります。また、ビルド時にタグを自動生成して、それをインストールに含めるかどうかを指定できる新しい "ソフトウェア識別タグの使用" 設定も追加されました。この設定のデフォルト値は [はい] です。

"ソフトウェア識別タグの使用" 設定で [はい] が選択されている時に、4 つの必須識別設定 ([一般情報] ビューの "一意な ID"、"一意な登録 ID"、"タグ作成者"、および "タグ作成者 ID" 設定) のうち最低 1 つの値が設定されていなかった場合、空白の設定ごとに、ビルド警告 –7235 が一回発生します。このビルド警告では、特定の必須タグが空白であるために、ソフトウェア識別タグが作成されず、インストールに含まれなかったことが通知されます。この警告を解決するには、各設定に適切な値を入力するか、または "ソフトウェア識別タグの使用" 設定で [いいえ] を選択します。

オートメーション インターフェイスには、新しいタグの設定が含まれています。ISWiProject オブジェクトに、プロジェクトで、ソフトウェア識別タグの作成を有効または無効にできる新しい EnableSwidtag プロパティが追加されました。また、[一般情報] ビューでも構成可能な様々なタグ関係の設定を構成できる新しい SwidtagProperty プロパティも追加されました。

この機能は、基本の MSI プロジェクトに適用します。

詳細については、次を参照してください。

- 製品のソフトウェア識別タグを含める
- [一般情報] ビューの設定
- ISWiProject オブジェクト

強化機能

InstallShield には、以下のような強化機能が搭載されています。

マージ モジュール プロジェクトに、IIS、テキスト ファイルの変更、および XML ファイルの変更のためのビルトイン サポートが追加されました

いくつかの既存のビューが、InstallShield のマージ モジュール プロジェクトでも使用可能になりました:

- **IIS 構成** – このビューでは、新しい IIS Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張を作成および管理することができます。
- **テキスト ファイルの変更** – このビューを使って、ターゲット システム上で実行時に変更を行うテキスト ファイル (たとえば、.txt、.htm、.xml、.config、.ini、および .sql) 内の検索/置換処理を構成できます。
- **XML ファイル変更** – このビューでは、実行時に変更する XML ファイルのノードとノード セットの参照を追加します。

マージ モジュール プロジェクトのこれらのビューでは、IIS を構成したり、テキスト ファイルの変更を指定したり、XML ファイルの変更を指定したりできます。インストール プロジェクトでこれらの変更を構成するために使用されていた手順が、マージ モジュール プロジェクトでもサポートされるようになりました。これらのビューで

プロジェクト情報を含むマージ モジュールをビルドし、そのマージ モジュールをインストール プロジェクトに追加して、そのインストール プロジェクトでリリースをビルドした場合、適切なランタイム サポートがインストールに含められます。

以前、これらのビューは、インストール プロジェクトでのみ使用可能でした。

詳しくは、次を参照してください:

- インターネット インフォメーション サービス
- テキスト ファイルの変更
- XML ファイルの変更

IIS 7.x の新しいアプリケーション プール ID オプション

新しい ApplicationPoolIdentity オプションは、[IIS 構成] ビューで構成されたアプリケーション プールの "識別" 設定で使用できます。アプリケーション プールのワーカプロセスを実行するために選択されたアプリケーション プールに一意な仮想 ID を使用する場合は、このオプションを選択します。このオプションのサポートは、IIS 7 以降が搭載されたターゲット システムで使用できます。IIS 6 が搭載されているシステム上で実行されるインストールでこの新しいオプションが選択された場合、アプリケーション プールの ID には NetworkService アカウントが代わりに使用されます。

この機能は、基本の MSI、InstallScript、InstallScript MSI、および マージ モジュール プロジェクト タイプに適用します。

詳細については、「アプリケーション プールの設定」を参照してください。

オートメーション インターフェイスの機能強化: 新しく追加された Setup.exe の必須実行レベルを指定するための RequiredExecutionLevel プロパティ

ISWiRelease オブジェクト に読み取り RequiredExecutionLevel プロパティが追加されました。このプロパティは、[リリース] ビューにあるリリースで開かれている [Setup.exe] タブにある "必要実行レベル" 設定に対応しています。このプロパティは、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで使用できます。

InstallShield 2011 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

InstallScript における Unicode サポート

InstallShield を使って、ランタイム文字列、ファイル、パス、レジストリ エントリ、その他のインストール データで Unicode を使用できる InstallScript インストールおよび InstallScript カスタム アクションを作成できます。さらに、InstallScript コンパイラおよび InstallScript エンジンでは、今回より、スクリプト コード外部に実装された関数に Unicode 文字列へのポインターを渡すことができます。また、スクリプト コード外部に渡される構造体に Unicode 文字列を格納することもできます。そのほか、InstallScript にはモダン言語の多言語インストールを完全にサポートするための機能が追加されています。

その結果、すべての言語は、そのサポートがインストールされているシステム上で正しく表示されます。エンド ユーザーは、システム上で Unicode 対応ではないプログラム用に使用される言語とインストールに使用される言語とを一致させる必要がなくなりました。ただし、ターゲット システムには、フォントをインストールする必要

があります。Windows の一部のバージョンでは、そのフォントがデフォルトでインストールされない言語もあります。たとえば、Windows XP 英語版システムでは、日本語のフォントがデフォルトではインストールされません。その場合、インストールが日本語の文字を使用できるように、フォントをインストールする必要があります。

InstallScript および InstallScript MSI プロジェクトにおける Unicode セットアップランチャー

今回より、InstallShield ではすべての Setup.exe と Update.exe ファイルが Unicode でビルドされます。以前、InstallScript および InstallScript MSI プロジェクトでは、すべての Setup.exe と Update.exe ファイルが ANSI でビルドされました。

Unicode セットアップランチャーは、ターゲット システムで適切なコード ページが実行されているいないにかかわらず、セットアップランチャーのユーザー インターフェイスで文字を正しく表示することができます。ANSI セットアップランチャーは、ターゲット システムで適切なコード ページが実行されている場合のみ、セットアップランチャーのダイアログで文字を正しく表示します。しかし、ターゲット システムで適切なコード ページが実行されていない場合、文字化けすることがありました。

エンド ユーザーは、ターゲット システムの言語にかかわらず、Setup.exe および Update.exe ファイルを Unicode パス内部から起動することができます。たとえば、今回より、エンド ユーザーは、英語版システム上で C:¥Users¥*日本語文字*¥Desktop¥Setup.exe からインストールを起動することができます。以前、このインストールは失敗しました。

ファイル、フォルダー、レジストリ エントリ、およびサポート ファイルにおける Unicode サポート

インストール ランタイムの主要な部分に Unicode サポートが追加されました。これによって、ファイル名、フォルダー名、レジストリ エントリ、およびサポート ファイル名に任意の言語の文字を同時に使用できるようになりました。たとえば、ファイル名またはターゲット パスに日本語文字が含まれるファイルを、英語版システムにインストールすることが可能です。組み合わされた言語は、ターゲット システムの現在の言語設定に関わらず正しく表示されます。

ポインター サポート

InstallScript エンジンとコンパイラーは、今回より、WPOINTER と呼ばれる新しいポインターをサポートします。対応する名前、wpointer および LPWSTR も使用可能です。たとえばパラメーターで Unicode 文字列へのポインターを受け付ける DLL 関数がある場合、この新しい種類のポインターを使うことができます。実行時にスクリプトで DLL 関数が呼び出されると、InstallScript エンジンはポインターを ANSI バージョンではなく、文字列の Unicode コピーに渡します。以前、ポインターは文字列の ANSI コピーにのみ渡すことができました。

InstallShield の以前のバージョンで使用される WSTRING データ タイプを使って Unicode 文字列を渡す機能は、引き続きサポートされています。

詳細については、次を参照してください。

- ポインター
- データ型および定義済み構造

構造体のサポート

InstallScript の構造体には、文字列、ポインター、その他の構造体をはじめとする任意の基本データ タイプを含めることができます。構造体に Unicode 文字列を含む必要があり、その構造体が外部 DLL に渡される場合、InstallScript エンジンはその構造内の文字列メンバー タイプを区別して、構造体サイズとメンバー オフセットを正しく計算します。Unicode として保存して渡す必要がある文字列メンバーは、WSTRING タイプを使って宣言できます。

以前、構造体に Unicode 文字列を含む必要があり、その構造が外部 DLL に渡される場合、InstallScript エンジンはその構造内に含まれる文字列がすべて ANSI であると想定しました。その結果、構造体のサイズとメンバーのオフセットに誤りが生じることがあり、DLL がその構造体に関連するデータの読み取りまたは書き込みを正しく行いませんでした。構造体の文字列メンバーに WSTRING を使用しても、効果がありませんでした。

既存の文字列を STRING タイプのままに残すことができます。InstallScript エンジンは、スクリプト コード外部で文字列を渡すとき、引き続きこれらを ANSI 文字列として処理します。

構造体のポインター メンバーも、今回より、WPOINTER として宣言できます。これによって、構造体で Unicode 文字列へのポインターを格納できます。

詳細については、次を参照してください。

- データ構造体
- データ型および定義済み構造

文字列テーブルのサポート

InstallShieldは、今回より、InstallScript プロジェクトの文字列テーブルをビルドする際に Unicode エンコードを使用します。このサポートにより、今回より、InstallScript プロジェクトの文字列テーブルに複数言語を含めることが可能となり、それらの言語がターゲット システムのコードページ設定に影響を受けることがなくなりました。さらに今回より、文字列テーブルに、ヒンディー語のようなコード ページを持たない言語の文字列を含めることができます。

インストールのユーザー インターフェイスに表示されるすべての文字列を、文字列テーブルに格納することが推奨されます。InstallScript コード内に直接文字列を書き込むこともできますが、その場合 Unicode として格納されません。したがって、正しいコード ページが設定されたシステム上でインストールが実行された場合にのみ、言語が正しく表示されます。文字列テーブルに文字列を格納して、InstallScript コードからそれらを参照することで、この問題を回避することができます。

詳細については、次を参照してください。

- InstallShield で文字列エントリを使用する
- 文字列定数演算子 (@)

Unicode を使った InstallScript ダイアログ

InstallScript ダイアログは、Unicode サポートを含みます。これによって、たとえば InstallScript ダイアログで日本語とドイツ語、またはロシア語とポーランド語を同時に使用できます。組み合わされた言語は、ターゲット システムの現在の言語設定に関わらず正しく表示されます。

この機能は、InstallScript および InstallScript MSI プロジェクト タイプで使用できます。

InstallScript デバッガーにおける Unicode サポート

InstallScript デバッガーは、今回より Unicode をサポートします。たとえば、szMsg = @ID_MSG という InstallScript コードの行をデバッグするとき、今回より、使用中のオペレーティング システム、または文字列に使用している言語にかかわらず、szMsg の値が表示されます。ID_MSG の値に中国語文字、または複数の言語からの文字が含まれている場合、InstallScript デバッガーのあらゆる領域 (スクリプト ウィンドウ内の変数に対して表示される変数ウィンドウ、ウォッチ ウィンドウ、およびツールヒント) で、疑問符の代わりに適切な文字が表示されます。

InstallScript デバッガについての情報は、*InstallScript デバッガー ユーザー ガイド*の「InstallScript ベース インストールのデバッグ」を参照してください。

InstallShield キャビネット ビューアーおよびログ ファイル ビューアーにおける Unicode サポート

InstallShield キャビネット ビューアーおよび InstallShield ログ ファイル ビューアーが、新しく InstallShield キャビネット & ログ ファイル ビューアー という 1 つのツールにまとまりました。この新しいツールを使って、InstallScript キャビネット ファイル (.cab)、InstallScript ヘッダー ファイル (.hdr) および InstallScript ログ ファイル (.ilg) を開いて参照することができます。このツールには Unicode サポートが含まれているため、使用中のオペレーティング システムの言語や文字列に使用されている言語にかかわらず、各ファイル名、レジストリ キー、ショートカット、および .cab、.hdr、または .ilg ファイルに含まれるその他のデータを正しく表示することができます。以前、一部の状況において一部の文字が疑問符で表示される場合がありました。

キャビネットおよびヘッダー ファイル サポートは、InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクト タイプに適用します。

ログ ファイル サポートは、InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクト タイプに適用します。 InstallScript および InstallScript MSI プロジェクト

このツールについては、次を参照してください:

- InstallShield キャビネット & ログ ファイル ビューアー
- InstallScript キャビネット ファイル (.cab) およびヘッダー ファイル (.hdr) を表示する
- InstallScript ログ ファイル (.ilg) を参照する

Team Foundation Server (TFS) との統合

InstallShield では、Team Foundation Server (TFS) 2010 との統合サポートが強化されています。

Visual Studio 2010 内から InstallShield を使用する場合、Source Control Explorer にアクセスして、InstallShield プロジェクトを Team Foundation バージョン コントロールと統合し、InstallShield プロジェクトと Visual Studio ソリューションへの変更を管理することができます。

Team Foundation Build を使って、InstallShield プロジェクトと Visual Studio ソリューションを定期的、またはオンデマンドでコンパイル、テスト、およびデプロイすることもできます。インストールはソリューションがビルドされる度に、自動的に最新のソース ファイルで更新されます。

さらに、InstallShield と Visual Stuio が搭載された同じマシン上に Team Explorer をインストールした場合、Visual Studio で開かれている InstallShield プロジェクト内から Team Explorer を使用できます。これで、次のようなタスクを行うことができます:

- InstallShield プロジェクトでの作業中にソース管理エクスプローラーを使用。
- InstallShield プロジェクトと Visual Studio ソリューションのビルドを構成。
- 新しいビルドをキューに配置。
- InstallShield プロジェクトと Visual Studio ソリューションのバグやタスクなどの作業項目を追跡する。

詳細については、次を参照してください。

- Microsoft Visual Studio Team Foundation Server との統合
- InstallShield プロジェクトを Team Explorer に追加する

Microsoft SQL Server 2008 R2 サポート

InstallShield は、今回より、SSQL Server 2008 R2 上で SQL スクリプトを実行するためのサポートを含みます。また、InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストには、SQL Server 2008 R2 が含まれています。

この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプで使用できます。

SQL Server 2008 R2 Express、SQL Server Native Client、Visual C++ 2010 用の新しい InstallShield 前提条件およびその他の再配布可能ファイル

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる多くの InstallShield 前提条件が含まれています。

- Microsoft SQL Server 2008 R2 Express
- Microsoft SQL Server 2008 Native Client 10.00.2531
- Microsoft SQL Server Native Client 9.00.4035
- Microsoft SQL Server System CLR 型 10.00.2531
- Microsoft SQL Server 2008 Management Objects 10.00.2531
- Microsoft Visual C++ 2010 再配布可能パッケージ
- Microsoft Visual C++ 2008 SP1 Redistributable Package
- Microsoft Visual C++ 2005 SP1 再配布可能パッケージ (x64)
- Windows Installer 3.1 – 日本語
- MSXML 6.0 SP1 – 日本語
- Microsoft .NET Framework 4.0 Client Language Pack – 日本語
- Microsoft .NET Framework 4.0 Full Language Pack – 日本語

InstallScript コード、SQL スクリプト、VBScript カスタム アクション、および JScript カスタム アクションの作成および変更を行うスクリプト エディターの強化

InstallShield のいくつかのビューで、プロジェクトのコードを書き込むためのスクリプト エディターが強化されました。スクリプト エディターは、以下の点で強化されています:

- **オートコンプリート機能の拡張** – スクリプト エディターに文字を入力するとき、InstallShield はその文字で始まる関数、キーワード、および定数をアルファベット順にリストするポップアップを表示します。用語全体を手入力する代りに、そのポップアップ リストから選択すると、InstallShield によってその用語がスクリプトに追加されます。

  [InstallScript] ビューでスクリプト エディターを使用する場合、文字列定数演算子 (@) を入力すると、使用可能な文字列識別子のポップアップ リストが表示されます。リストから適切な文字列識別子を選択すると、InstallShield がそれをスクリプトに追加します。

  ローカル変数のオートコンプリート機能も有効な場合、[InstallScript] ビューのスクリプト エディターで表示されるポップアップ リストには、ローカル変数も含まれています。

  オートコンプリートを使うと、コードを手入力する手間が省けるため、作業効率が上がります。また、コードのスペル ミスを回避することができます。

  InstallShield の以前のバージョンでは、[InstallScript] ビューのスクリプト エディターでのみオートコンプリートがサポートされていました。そのため、オートコンプリート機能は SQL スクリプト、VBScript コード、または JScript コードでは使用できませんでした。さらに、ポップアップ リストも InstallScript 関数に限られており、キーワード、定数、ローカル変数、または文字列 ID はオートコンプリートで使用できませんでした。

- **詳細な InstallScript 関数呼び出しのヒント** ― 関数呼び出しのヒントが有効な場合、[InstallScript] ビューでスクリプトに関数の呼び出しを入力しているときに、InstallScript 関数呼び出しのヒント (一種のツールヒント) が表示されます。関数呼び出しのヒントは、ビルトイン関数のパラメーター情報を表示します。その他、ビルトイン関数の説明や、入力しているパラメーターについての説明も表示します。詳細な呼び出しのヒントを使うと、スクリプト エディターからヘルプ ライブラリへ、そしてまたスクリプト エディターへと切り替える必要がなく、関数に関するヘルプ情報を表示できます。

  InstallShield の以前のバージョンの [InstallScript] ビューでも、呼び出しのヒントをサポートしますが、その情報量が限られています。呼び出しのヒントには、関数呼び出しとその関数のすべてのパラメーターが表示されましたが、呼び出しのヒントには、関数の説明、または入力しているパラメーターの説明は含まれませんでした。

- **構文の折りたたみ機能** ― InstallShield の様々なビューで、スクリプト エディターの構文の折りたたみ機能をサポートするかどうかを指定できます。構文の折りたたみ機能が有効な場合、スクリプトで展開可能または折りたたみ可能なブロックで始まるコードの各行の横のマージンに、プラス (+) またはマイナス (−) 記号が追加されます。プラス記号をクリックして非表示となっているコードを展開したり、マイナス記号をクリックして表示されているコードを非表示にしたりできます。

  構文の折りたたみ機能を使って、長いスクリプトを縮小することで、現在行っている作業に関連のあるコードに焦点を当てることができます。また、スクリプトの全体的な構造を確認するのに便利です。

強化されたスクリプト エディターを含むビューは、[InstallScript] ビュー、[SQL スクリプト] ビュー、および [カスタム アクションとシーケンス] ビュー (このビューで VBScript または JScript ファイルを参照する場合) です。

デフォルトで、スクリプト エディタにおける構文の折りたたみ機能は無効になっていますが、オートコンプリート、ローカル変数のオートコンプリート、および関数呼び出しのヒントは有効です。これらの機能を有効または無効にするには、[スクリプト エディタのプロパティ] ダイアログ ボックスを使います。このダイアログを使って、フォント、構文の色、行番号などのスクリプト エディターのその他の設定も変更できます。このダイアログ ボックスにアクセスするには、スクリプト エディター内を右クリックしてから、[プロパティ] を選択します。

詳細は、次を参照してください:

- 様々なビューで、[スクリプト エディター] ペインを使って作業する
- スクリプト エディターでのコードの書き込みにオートコンプリート機能を使う
- スクリプト エディター内で InstallScript 関数について関数呼び出しのヒントを表示する
- スクリプト エディターで構文ハイライトの色を変更する
- スクリプト エディターで構文の折りたたみ機能を有効または無効にする
- スクリプト エディターのキーボード ショートカット
- [スクリプト エディターのプロパティ] ダイアログ ボックス

64 ビットの Microsoft App-V パッケージを作成できる機能

InstallShield の Microsoft App-V アシスタントには、64 ビットの Windows Installer パッケージを 64 ビットの Microsoft App-V パッケージに変換するためのサポートが搭載されています。64 ビットの Microsoft App-V パッケージは、64 ビットの Microsoft App-V 4.6 Client を搭載した 64 ビット Windows システム上に展開することができます。リパッケージャーに追加された新しい 64 ビット リパッケージ サポートとこの機能を組み合わせて、任意の 64 ビット インストールを 64 ビット App-V パッケージに変換することができます。

64 ビット Windows Installer パッケージを App-V パッケージに変換する場合は、64 ビット Windows システム上で行うことをお勧めします。32 ビット システム上で変換しようとすると、64 ビット バイナリ ファイルの COM 情報抽出エラーが発生する場合があります。また、一部の状況において、Windows Installer パッケージはパッケージそ

のものに含まれていない実行可能ファイルをターゲットとするショートカットを含みます。ショートカットが、64 ビットの場所にある実行可能ファイルをターゲットとする場合、これらのショートカットは、32 ビット システム上で正しく処理されません。

Microsoft App-V アシスタントは、InstallShield Premier Edition で使用できます。

InstallShield 前提条件の検索パスを指定できる機能 (前提条件ファイルのソースの場所におけるパス変数および相対パスのサポート)

InstallShield では、今回より、InstallShield 前提条件ファイル (.prq ファイル) を検索するフォルダを指定できます。この機能によって、複数の開発者の間で InstallShield 前提条件を共有し、それらをソース コード管理システムで保管することが容易になります。以前は、InstallShield が .prq ファイルを検索するのは、*InstallShield Program Files フォルダー*¥SetupPrerequisites のみでした。

InstallShield でフォルダーを指定するとき、いくつかの方法があります:

- InstallShield 内部から編集またはビルドを行う場合、[ツール] メニューで [オプション] をクリックすると表示される [オプション] ボックスにある新しい [前提条件] タブを使って、マシン共通および現在のユーザーのフォルダをコンマ区切りのリストで指定できます。このタブは、マージ モジュールの検索パスを指定できる、[オプション] ダイアログ ボックスの [マージ モジュール] タブと似ています。
- ISCmdBld.exe を使ってコマンドラインからビルドを行う場合、新しい -prqpath パラメーターを使って、コンマ区切りのフォルダーのリストを指定できます。

  .ini ファイルを使って ISCmdBld.exe パラメーターを指定する場合、.ini ファイルの [Mode] セクションで新しい PrerequisitePath パラメーターを使用して、フォルダのコンマ区切りのリストを指定できます。
- MSBuild または Team Foundation Server (TFS) を使ってビルドする場合、InstallShield タスクで新しい PrerequisitePath パラメーターを使います。このパラメーターは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPrerequisitePath として露出されます。複数のパスを指定するには、順序指定されたパスの配列を使用します。

InstallShield 前提条件エディターの [含めるファイル] タブを使って InstallShield 前提条件にファイルを追加すると、エディターは今回より、適切な場合に <WindowsFolder> および <ISProductFolder> などの定義済みパス変数を使用します。また、追加するファイルが InstallShield 前提条件の .prq ファイルと同じフォルダ (または .prq ファイルが含まれているフォルダのサブフォルダ) に格納されている場合、InstallShield 前提条件エディターは .prq ファイル内のファイルに相対パスを使用します。[含めるファイル] タブでファイルのパスを参照したとき、InstallShield 前提条件エディターは相対パスではなく、完全パスをリストします。

InstallShield 前提条件エディターで [名前を付けて保存] コマンドを使用して .prq ファイルの場所を変更すると、InstallShield 前提条件エディターは、適切な場合に前提条件のファイルのパスを更新します。

詳しくは、次を参照してください:

- InstallShield 前提条件を含むディレクトリを指定する
- [前提条件] タブ
- ISCmdBld.exe
- .ini ファイルでコマンドライン ビルド パラメーターを渡す
- Microsoft ビルド エンジン (MSBuild)

Windows Installer ベースのダイアログにおける新しいハイパーリンク コントロール

コントロール ツールバーに、Windows Installer ベースのダイアログで [ダイアログ] ビューに使用できる、新しいハイパーリンク コントロールが追加されました。ハイパーリンク コントロールは HTML リンクを表示します。このリンクを実行時にクリックすると、ターゲット システム上のデフォルト ブラウザでページが開きます。

Windows Installer 5 は、この新しいハイパーリンク コントロールをサポートします。このコントロールを Windows Installer の以前のバージョンを持つシステムで表示されるダイアログに使用すると、ランタイム エラー 2885 が発生して、インストールが中止されます。このため、ダイアログ上でハイパーリンク コントロールを使用するとき、インストールが Windows Installer 4.5 以前をターゲットとする場合、プロジェクトにハイパーリンク コントロールを含むバージョンと、含まないバージョンの 2 種類のダイアログを含めることをお勧めします。ダイアログに条件を追加して、ターゲット マシンの Windows Installer バージョンによってダイアログを表示または非表示とします。

この機能は、基本の MSI およびマージ モジュール プロジェクト タイプで使用できます。

詳細については、「ハイパーリンク コントロール」を参照してください。

Setup.exe および Update.exe においてカスタム アイコンとカスタム バージョン リソースのプロパティを指定できる機能

InstallShield では、今回より、ビルド時に作成する Setup.exe ファイルでカスタム アイコンおよびカスタムバージョン リソースのプロパティを使用できます。アイコンとバージョン リソースのプロパティは、Setup.exe の [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Setup.exe ファイルを右クリックして、[プロパティ] をクリックしたときに表示されます。エンド ユーザーが Windows Explorer で Setup.exe ファイルを参照したときにも、このアイコンが表示されます。このサポートは、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで使用できます。

今回より、同じ機能 (カスタム アイコンとカスタム バージョン リソースのプロパティを指定できる機能) が、基本の MSI、InstallScript MSI、および QuickPatch プロジェクトで作成された Update.exe ファイルでも使用できます。

Setup.exe および Update.exe のカスタム アイコン

[リリース] ビューの [Setup.exe] タブには、Setup.exe セットアップランチャーに使用するアイコンを指定できる "Setup.exe アイコン ファイル" 設定があります。アイコンには、.exe、.dll、または .ico ファイルを使用できます。アイコンを指定しなかった場合、InstallShield は Setup.exe ファイルのデフォルト アイコンを使用します。

以前は、ビルド中の Setup.exe ファイルが自己展開型の単一セットアップランチャーの場合に、InstallScript プロジェクトでカスタム Setup.exe アイコンの指定がサポートされていました。これは、基本の MSI または InstallScript MSI プロジェクトではサポートされていませんでした。また、圧縮された InstallScript インストールにおいてもサポートされていませんでした。

新しいアイコン設定を使って、Update.exe ランチャーに使用するアイコンを指定できます。この設定は、基本の MSI および InstallScript MSI プロジェクトの [パッチのデザイン] ビュー内にあるパッチの構成の [詳細] タブで行います。また、QuickPatch プロジェクトの場合、[一般情報] ビューの [ビルドの設定] 領域にある [詳細] タブで設定できます。アイコンを指定しなかった場合、InstallShield は Update.exe ファイルのデフォルト アイコンを使用します。

以前は、InstallShield は Update.exe ファイルのカスタム アイコンの指定を一切サポートしませんでした。

Setup.exe および Update.exe のカスタム バージョン リソースのプロパティ

InstallShield がビルド時に Setup.exe ランチャーの以下のバージョン リソースを構成するとき、今回より、[一般情報] ビューおよび [リリース] ビューで入力されたカスタム情報を使用します。

- 会社名

- 製品名
- 製品バージョン
- 著作権情報
- ファイル バージョン
- ファイルの説明

カスタム著作権情報およびカスタム ファイルの説明を使用する場合、[リリース] ビューで "カスタム バージョンのプロパティを使用する" 設定で [はい] を選択してください。

以前、InstallShield は多くの場合にカスタム情報を使用しませんでした。たとえば、InstallShield が以前に作成した InstallScript Setup.exe ファイルには、その Setup.exe ファイルをビルドした InstallShield のバージョン固有の詳細が含まれました。そのため、Setup.exe プロパティ ダイアログ ボックスに表示される製品名は、インストールする製品の名前ではなく、InstallShield と表示されました。

"会社名"、"製品名"、"製品バージョン"、"説明"、"著作権情報" といった新しい設定を使って、InstallShield で Update.exe ファイルをビルドするときに使用するカスタム情報を指定できます。これらの設定は、基本の MSI および InstallScript MSI プロジェクトの [パッチのデザイン] ビュー内にあるパッチの構成の [詳細] タブで行います。また、QuickPatch プロジェクトの場合、[一般情報] ビューの [ビルドの設定] 領域にある [詳細] タブで設定できます。

以前、InstallShield は Update.exe ファイルにカスタム バージョン リソース情報を使用しませんでした。

詳しくは、次を参照してください:

- セットアップランチャーのファイルのアイコンを指定する
- セットアップランチャーのファイルのプロパティをカスタマイズする
- リリースの [Setup.exe] タブ
- アップデート起動ツールのアイコンを指定する
- アップデート起動ツールのファイルのプロパティをカスタマイズする
- [詳細] タブ ([パッチのデザイン] ビュー)
- [詳細] タブ (QuickPatch プロジェクト)

ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する機能

InstallShield Premier Edition には、ビルド プロセスの様々な段階で実行するコマンドを指定することができる、新しいリリースの設定が追加されました。これらの新しい設定は、[リリース] ビューでリリースを選択したときに、新しい [イベント] タブに表示されます。

- **ビルド前のイベント** — この設定を使って、InstallShield がリリースのビルドを開始する前に実行するコマンドを指定します。このイベントは InstallShield がリリース フォルダとログ ファイルを作成した後、リリースのビルドを開始する前に実行します。

  この設定は、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトで使用できます。

- **圧縮前のイベント** — この設定を使って、製品のデータ ファイルを .cab ファイルに格納する場合、InstallShield が .msi パッケージと .cab ファイルをビルドした後に実行するコマンドを指定します。このイベントは .cab ファイルが .msi パッケージにストリームされた後、.msi パッケージにデジタル署名が行われて Setup.exe ファイルにストリームされる前に発生します。

  この設定は、基本の MSI および InstallScript MSI プロジェクトで使用できます。

- **ビルド後のイベント** – この設定を使って、InstallShield がリリースをビルドして署名を行った後に実行するコマンドを指定します。

  この設定は、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトで使用できます。

新しい [イベント] タブは、以前の [ポストビルド] タブに取って代わります。以前 [ポストビルド] タブで提供されていた設定は、新しい [イベント] タブに表示されます。また、以前マージ モジュール プロジェクトの [ビルド] タブにあったパブリッシュ関連の設定は [イベント] タブに移動しました。

オートメーション インターフェイスは、この新しいビルド イベントの設定をサポートします。ISWiRelease オブジェクトには、ビルド プロセスの様々な段階におけるコマンドを指定できる 3 つの新しいプロパティが含まれてます。

- PrebuildEvent
- PrecompressionEvent
- PostbuildEvent

詳しくは、次を参照してください:

- ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する
- リリースの [イベント] タブ
- ISWiRelease オブジェクト

Setup.exe に有効期限日を設定する機能

InstallShield では、Setup.exe の有効期限日と有効期限切れメッセージを設定できます。エンド ユーザーが Setup.exe をプロジェクトで指定された日付以降に実行すると、有効期限切れメッセージが表示されて、インストールが終了します。

Setup.exe ファイルの有効期限日とメッセージの設定は、[リリース] ビューで選択されたリリースの [Setup.exe] タブに新しく追加された "有効期限日と有効期限メッセージ" 設定で行います。デフォルトで、Setup.exe に有効期限日は設定されていません。有効期限日を指定すると、デフォルトの有効期限切れメッセージまたは独自のカスタムメッセージを使用できます。

オートメーション インターフェイスには、これらの新しい設定のサポートが含まれています。ISWiRelease オブジェクトには、有効期限日とメッセージを設定できる 2 つの新しいプロパティ (ExpirationDate および ExpirationMessage) が含まれています。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- リリースの [Setup.exe] タブ
- ISWiRelease オブジェクト

Visual Studio セットアップとマージ モジュール プロジェクトを既存の InstallShield プロジェクトにインポートする機能 (プロジェクト コンバーターの強化)

InstallShield では、今回より、Visual Studio セットアップ プロジェクト (.vdproj) を InstallShield 基本の MSI またはマージ モジュール プロジェクト (.ism) にインポートするか、Visual Studio マージ モジュール プロジェクト (.vdproj) を InstallShield 基本の MSI またはマージ モジュール プロジェクト (.ism) にインポートすることができます。これらのタスクを行うと、Visual Studio プロジェクトに含まれている同じデータと設定を含む InstallShield イ

ンストール プロジェクトおよびマージ モジュール プロジェクトを開発できます。ウィザードを使って、プロジェクト出力、ファイル、レジストリ キー、ファイル拡張子、カスタム アクション、ターゲット システム検索、および起動条件を Visual Studio プロジェクトから既存の InstallShield プロジェクトにインポートします。

Visual Studio プロジェクトを既存の InstallShield プロジェクトにインポートするには、InstallShield の Visual Studio デプロイメント プロジェクト インポート ウィザードを使います。このウィザードでは、Visual Studio プロジェクト内の特定の設定をインポートするか、無視するかを選択することができます。

Visual Studio プロジェクトを新しい InstallShield プロジェクトに変換するための、これまでのサポートが拡張されました。Visual Studio プロジェクトに定義済み前提条件が含まれている場合、InstallShield は今回より、プロジェクトの変換中にそれらを対応する InstallShield 前提条件に変換します。これと同じ前提条件の変換機能が、Visual Studio プロジェクトを InstallShield プロジェクトにインポートするための新しいウィザードにも搭載されています。

Visual Studio プロジェクトに 1 つ以上のプロジェクト出力が含まれている場合は、変換プロセスの代わりにインポート ウィザードを使用します。InstallShield プロジェクトは、Visual Studio セットアップまたはマージ モジュール プロジェクトおよびそのプロジェクト依存関係が含まれているのと同じ Visual Studio ソリューションに含まれていなくてはなりません。インポート ウィザードを使ってプロジェクト出力を InstallShield プロジェクトにインポートするためには、InstallShield を Visual Studio 内部で開いてください。

詳しくは、次を参照してください:

- Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートする
- Visual Studio デプロイメント プロジェクト インポート ウィザード

SQL スクリプトにおける Unicode および UTF-8 サポート

InstallShield は、Unicode BOM エンコードまたは UTF-8 BOM エンコードのいずれかを使った SQL スクリプトをデザイン時および実行時にサポートします。[SQL スクリプト] ビューを使って、どちらかのエンコードを使った SQL スクリプトをプロジェクトに追加できます。これらの SQL スクリプトを、必要に応じて [SQL スクリプト] ビュー内から編集することもできます。実行時に、必要に応じてインストールおよびアンインストール中に SQL スクリプトが実行されます。

以前、InstallShield では Unicode BOM エンコードを使った SQL スクリプトが実行時にはサポートされていましたが、デザイン時にはサポートされていませんでした。したがって、このエンコードを使ったスクリプトをプロジェクトの [SQL スクリプト] ビューに追加すると、InstallShield がスクリプトを ANSI 形式に変換するかどうかを指定するためのプロンプトが表示されました。InstallShield で ANSI への変換を許可すると、それを [SQL スクリプト] ビュー内から編集できます。ただし、一部の状況において、デザイン時と実行時に文字化けが発生することがありました。InstallShield が ANSI に変換することを許可しなかった場合、そのファイルは Unicode BOM エンコードのままとなりました。このエンコードを使うと、インストールは実行時にターゲット システムのコード ページと一致しない言語の文字列を正しく使用しますが、InstallShield 内部からスクリプトを参照または編集することができませんでした。

さらに以前、InstallShield は UTF-8 BOM エンコードを使用する SQL スクリプトを適切にサポートしませんでした。このエンコード タイプを使用する SQL スクリプトをプロジェクトに追加して、スクリプトに開発システムのコード ページと一致しない言語の文字列が含まれていた場合、[SQL スクリプト] ビューは、そのファイルを ANSI エンコードとして処理するために、文字化けが発生することがありました。また、実行時に SQL スクリプトが実行されると、文字化けが発生しました。バイト オーダー マークも文字化けして表示されました。

[SQL スクリプト] ビュー内から新しい SQL スクリプトを作成すると、InstallShield はそのファイルに Unicode BOM エンコードを使用します。[SQL スクリプト] ビューで ANSI または UTF-8 BOM エンコードを使用したい場合、別のツールを使って適切なエンコードを用いて .sql ファイルを作成してから、プロジェクトの [SQL スクリプト] ビューにそのスクリプトをインポートまたは挿入することをお勧めします。

この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプで使用できます。

SQL Server 2008 Express SP1 および Adobe Reader 9 のための定義済みシステム検索

InstallShield に新しい定義済みシステム検索が追加されました:

- SQL Server 2008 Express SP1
- Adobe Reader 9

インストールでこれらの 1 つまたは両方が必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、これらのシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

この機能は、基本の MSI および InstallScript MSI プロジェクトに適用します。

64 ビット マネージコード カスタム アクションのサポート

InstallShield は、今回より、64 ビット マネージコード カスタム アクションをサポートします。プロジェクトでマネージコード カスタム アクションを含むリリースをビルドするとき、InstallShield はカスタム アクションと関連付けられたメインの .NET アセンブリのターゲット アーキテクチャ (32 ビットまたは 64 ビット) を判別しようとします。InstallShield は、実行時にマネージコードを実行するときに .NET Framework の適切なバージョン (32 ビットまたは 64 ビット) が使用されるようにリリースを構成します。

新しいデフォルトの動作をオーバーライドするには、[ダイレクト エディター] ビューを使って、次のフィールドを含む新しいレコードを ISClrWrap テーブルに追加します。

- Action_ — 変更するマネージコード カスタム アクションの名前を示します。
- Name — 次を入力します: **TargetPlatform**
- Value — 適切なアーキテクチャを指定します。.NET Framework の 32 ビット バージョンを使用するには、次を入力します:**x86**

  .NET Framework の 64 ビット バージョンを使用するには、次を入力します:**x64**

この機能は、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクト タイプで使用できます。

詳細については、「32 ビット マネージ コード カスタム アクションと 64 ビット マネージ コード カスタム アクションの違い」を参照してください。

64 ビット .NET Installer クラスと COM Interop のサポート

InstallShield は今回より、64 ビット .NET Installer クラスと COM Interop をサポートします。64 ビット システム上で InstallShield を使用している場合、InstallShield の [ツール] メニューにある [オプション] をクリックすると表示される [オプション] ボックスでは、今回より、.NET Framework に含まれている Regasm.exe および InstallUtilLib.dll ファイルの場所として、32 ビット と 64 ビットの 2 つのパスを指定できます。InstallShield は、.NET Installer クラスと COM Interop を含むリリースで、ビルド時に指定されたパスを使用します。

ISCmdBld.exe を使ってコマンドラインからビルドしている場合に、既存の -t パラメーターを使って .NET Framework の 32 ビット バージョンのパスを指定すると、ISCmdBld.exe は今回より、適切な場合に Regasm.exe および InstallUtilLib.dll の 64 ビットの場所を使用します。

MSBuild または Team Foundation Server (TFS) を使ってビルドしている場合に、InstallShield タスクで既存の DotNetUtilPath パラメーターを使って .NET Framework の 32 ビット バージョンのパスを指定すると、ビルドは今回より、適切な場合に Regasm.exe および InstallUtilLib.dll の 64 ビットの場所を使用します。

詳細は、次を参照してください:

- [.NET] タブ ([オプション] ダイアログ ボックス)
- ISCmdBld.exe
- Microsoft ビルド エンジン (MSBuild)

.NET Framework 4 をターゲットにする DLL で InstallScript 関数 DotNetCoCreateObject またはマネージ コード カスタム アクションの使用をサポート

Visual Studio 2010 で作成した DLL が .NET Framework 4 を使用する場合に、InstallScript 関数 **DotNetCoCreateObject** を使って、この DLL で関数を呼び出すことができます。以前、DLL が .NET Framework の以前のバージョンを使用したときにはインストールはクラッシュしませんでしたが、バージョン 4 を使用したときに、インストールがクラッシュしました。この機能は、次のプロジェクト タイプで使用できます: この機能は InstallScript と InstallScript MSI プロジェクトで使用できます。またこの機能は、InstallScript カスタム アクションを含む基本の MSI およびマージ モジュール プロジェクトでも使用できます。

さらに、今回より、マネージコード カスタム アクションで同じような DLL ファイルを使用できます。以前は、インストールに .NET Framework のバージョン 4 を使用する DLL のマネージコード カスタム アクションが含まれている場合、クラッシュが起こりました。この機能は、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- DotNetCoCreateObject
- 32 ビット マネージ コード カスタム アクションと 64 ビット マネージ コード カスタム アクションの違い

リリースのパス変数の値をオーバーライドする機能

InstallShield では今回より、プロジェクト内の各リリースでプロジェクトのカスタムパス変数 (標準パス変数、環境パス変数、およびレジストリ パス変数) の値をオーバーライドすることができます。この機能を使って、ビルドする特定のリリースごとに、ビルド時にプロジェクト内の特定のファイルとフォルダーを別のファイルとフォルダーに置換することができます。

たとえば、この機能を使ってカスタム アクションのバイナリ ファイルを入れ替えることができます。個別の 32 ビットと 64 ビットのターゲット システム用に個別のリリースを作成した場合、カスタム アクションに選択された DLL を参照するパス変数をオーバーライドすることができます。これによって、InstallShield が 32 ビット リリースには 32 ビット DLL を、また 64 ビット リリースには 64 ビット DLL を含みます。インストールがインストール処理を行っているファイルを入れ替えるのにパス変数のオーバーライドを行うことはお勧めできません。これは、ファイルの 32 ビット バージョンと 64 ビット バージョンに個別のコンポーネントを使用しなくてはならないためです。

プロジェクトで 1 つ以上のパス変数をオーバーライドする場合、[リリース] ビュー内のリリースの [ビルド] タブに追加された "パス変数のオーバーライド" 設定を使います。新しい "パス変数のオーバーライド" 設定と、IsCmdBld.exe または MSBuild で -I パラメーターを使ってパス変数をオーバーライドした場合、コマンドラインまたは MSBuild 値がリリース設定で指定された値よりも優先します。

オートメーション インターフェイスには、この新しい設定のサポートが含まれています。ISWiRelease オブジェクトには、リリースに含まれるプロジェクトのユーザー定義のパス変数、環境変数、およびレジストリ変数の値をオーバーライドできる、新しい PathVariableOverrides プロパティが含まれています。

この機能は、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、およびマージ モジュール プロジェクト タイプで使用できます。

詳しくは、次を参照してください:

- リリースのカスタム パス変数の値をオーバーライドする
- "パス変数のオーバーライド" 設定
- [パス変数のオーバーライド] ダイアログ ボックス
- ISWiRelease オブジェクト

IIS Web サイト、アプリケーション、および仮想ディレクトリの MIME の種類を構成する機能

[IIS 構成] ビューに新しく追加された "MIME の種類" 設定を使って、プロジェクトに含まれる Web サイト、アプリケーション、または仮想ディレクトリの MIME の種類を構成できます。この設定を使って、ターゲット システム上の Web サーバーからブラウザーまたはメール クライアントに送信できるコンテンツの種類を指定できます。

この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプに適用します。

詳細については、次を参照してください。

- Web サイトにおける MIME の種類設定
- アプリケーションまたは仮想ディレクトリにおける MIME の種類設定

既存の IIS アプリケーション プールを上書きする、またはそれが既存しない場合のみ作成する機能

[IIS 構成] ビューに、新しい "既存のアプリケーション プールを上書きする" 設定が追加されました。この設定は、[IIS 構成] ビューでアプリケーション プールを選択したときに、右側のペインに表示されます。この設定を使って、実行時にターゲット システム上に既に選択されたアプリケーション プールが存在する場合に起こる動作を指定できます。インストールによって、ターゲット システム上のアプリケーション プールの設定を上書きするか、アプリケーション プールをそのまま存続させることができます。この設定のデフォルト値は [はい] で、実行時に既存のアプリケーション プールが上書きされます。

この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプに適用します。

詳細については、アプリケーション プールの「既存のアプリケーション プールを上書きする」に記述されている説明を参照してください。

InstallScript および InstallScript MSI プロジェクトにおける新しいビルボード スタイル

InstallScript および InstallScript MSI プロジェクトには、進行状況ダイアログの進行状況バーの上に表示されるビルボードの新しいスタイルがサポートされています。このビルボード スタイルは、イメージ ファイル (.bmp, .gif, .jpg, and .jpeg) だけでなく、Adobe Flash アプリケーション ファイル (.swf) もサポートします。

プロジェクトに Flash ファイルとイメージ ファイルの両方が含まれていてるとき、ターゲット システムに Adobe Flash Player がインストールされていない場合、インストールはこれを検出して Flash ビルボードの代わりにイメージ ビルボードを表示します。

以前、InstallScript および InstallScript MSI プロジェクトで使用できる唯一のビルボード サポートでは、背景ウィンドウを使用する必要がありました。新しいビルボード スタイルで、背景ウィンドウは不要です。

新しいビルボード スタイルをプロジェクトに追加するには、[サポート ファイル/ビルボード] ビューを使って、プロジェクトにビルボード ファイルを追加します。実行時に、この新しいスタイルのビルボードを表示するには、新しい STATUSBBRD 定数を **Enable** 関数と一緒に使います。

新しいスタイルのビルボードは、スキン ダイアログを使用するプロジェクトでは使用できません。

詳しくは、次を参照してください:

- InstallScript および InstallScript MSI インストールでビルボードを表示する
- InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルおよびファイルの種類
- InstallScript または InstallScript MSI プロジェクトでコードを追加または変更してビルボードを表示する
- InstallScript または InstallScript MSI プロジェクトに Adobe Flash アプリケーション ファイル ビルボードを追加する
- InstallScript または InstallScript MSI プロジェクトにイメージ ビルボードを追加する
- InstallScript または InstallScript MSI プロジェクトにおけるビルボード ファイルの名前

基本の MSI プロジェクトにおけるイメージ ビルボードのループ サポート

基本の MSI プロジェクトの [ビルボード] ビューに新しく追加された "ビルボードのループ" 設定を使って、インストールがファイルの転送を完了するまでイメージ ビルボードを継続してループ表示してから適切な SetupComplete ダイアログを表示するかどうかを指定できます。

この設定で [はい] を選択して、ビルボードに割り当てた時間よりもファイルの転送に時間がかかった場合、インストールは最初のビルボードから再び表示します。必要な場合、ループはファイルの転送が終了するまで継続します。この設定のデフォルト値は [いいえ] です。これは、InstallShield の以前のバージョンの動作と同じです。

以前は、ビルボードに割り当てられた時間よりもファイルの転送に時間がかかった場合、インストールはファイル転送が終了するまで最後のビルボードを表示し続け、ビルボードをループしませんでした。

詳細については、次を参照してください。

- ビルボードを含む基本の MSI インストールにおける実行時の動作
- ビルボード設定

Windows Installer 5 における Windows サービスのアクセス許可の構成サポート

InstallShield は今回より、Windows サービスのアクセス許可の構成をサポートします。このサポートは Windows Installer 5 で提供されている MsiLockPermissionsEx テーブルを使用します。Windows Installer 5 は、新しいサービスのアクセス許可の設定をサポートします。以前の Windows Installer は、これらの設定を無視します。

新しいサービスのアクセス許可設定を構成するには、[セットアップのデザイン] ビュー (インストール プロジェクトの場合) または [コンポーネント] ビューで、コンポーネントの [詳細設定] 領域にある [サービス] ノードを使用します。[サービス] ノードで、プロジェクトに新しいサービスを追加します。[サービス] ノードの下にあるサブノードを選択すると、右側のペインでそのサービスの新しいアクセス許可と関連設定を構成できます。

この機能は、基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護
- サービスの設定

InstallScript プロジェクトのダイナミック ファイル リンクの構成におけるオートメーション インターフェイス サポート

InstallScript プロジェクトで、オートメーション インターフェイスの ISWiDynamicFileLinking オブジェクトおよび ISWiDynamicFileLinkings コレクションに、ダイナミック ファイル リンク サポートが追加されました。さらに今回より、InstallScript プロジェクトで ISWiComponent オブジェクトの 2 つのメソッドとプロパティを使用できます: AddDynamicFileLinking メソッドは、コンポーネントに新しいダイナミック ファイル リンクを追加し、RemoveDynamicFileLinking メソッドは、コンポーネントのダイナミック ファイル リンクを削除し、ISWiDynamicFileLinkings プロパティは、ダイナミック ファイル リンクのコンポーネントのコレクションを取得します。ISWiFile オブジェクトの DynamicFile 読み取り専用プロパティを InstallScript プロジェクトでも使用できるようになりました: このプロパティを使って、ファイルのソースがコンポーネントにダイナミックにリンクされているか、スタティックにリンクされているかを判別します。

詳細については、次を参照してください。

- ISWiDynamicFileLinking オブジェクト
- ISWiDynamicFileLinkings コレクション
- ISWiComponent オブジェクト
- AddDynamicFileLinking メソッド
- RemoveDynamicFileLinking メソッド
- ISWiFile オブジェクト

新しい FlexNet Connect 12.01 再配布可能ファイル

InstallShield はまた、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで FlexNet Connect 12.01 をサポートします。InstallShield の [アップデート通知] ビューで、2 つの FlexNet Connect 12.01 マージ モジュール (Common Software Manager が含まれているマージ モジュールと、含まれていないマージ モジュール) のどちらかを含みます。

強化機能

InstallShield には、以下のような強化機能が搭載されています。

InstallShield の Unicode ビュー

InstallShield のいくつかのビューと領域は、すべての言語からの文字を表示および入力できるように強化されています。たとえば、今回より、英語版のコンピュータ上で [システム検索] ビューにおいてシステム検索を構成するときに、ファイル名、パス、レジストリ エントリに中国語の文字を使用できます。以前、InstallShield のこれらの領域で表示される文字は疑問符として表示されました。

InstallShield で Unicode サポートが強化された領域は、[システム検索] ビュー、および [リリース] ビュー内のリリースの [複数インスタンス] タブ、および [リリース] ビュー内のリリースに表示されるタブです。InstallShield 2010 より、その他の多くのビューで Unicode がサポートされています。

[システム検索] ビューの強化は、基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトに適用します。

[リリース] ビュー内のリリースのタブにおける強化は、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、およびマージ モジュール プロジェクトに適用します。

[リリース] ビューの [複数インスタンス] タブの強化は、基本の MSI プロジェクトに適用します。

InstallShield MSI Diff を使用中にログファイルを生成するためのコマンドライン サポート

InstallShield MSI Diff には、新しく /out コマンドライン パラメーターが含まれています。InstallShield MSI Diff を実行中に、コマンドラインからこの新しいパラメーターを使って、2 つの .msi、.msm、または .pcp ファイル間の差分を確認したり、トランスフォーム (.mst) またはパッチ ファイル (.msp) によって Windows Installer データベースに適用される変更点を表示するログ ファイルを生成できます。また、このツールをコマンドラインから使用して、バイナリ形式で保存されている 2 つの InstallShield プロジェクト ファイル (.ism または .ise) 間の違いを識別するログ ファイルを生成することもできます。

詳細については、「InstallShield MSI Diff をコマンドラインから実行して、差分のログ ファイルを生成する」を参照してください。

Microsoft SQL Server のローカル インスタンスを 64 ビット システムでリストするサポート

InstallShield の [SQL スクリプト] ビューを使ってプロジェクトに SQL サポートを追加してから、そのインストールを 64 ビット ターゲット システムで実行すると、SQL 関連のビルトイン ランタイム ダイアログは、今回より、32 ビットの SQL Server のローカル インスタンス、32 ビットの SQL Server のネットワーク インスタンス、および 64 ビットの SQL Server のネットワーク インスタンスだけでなく、64 ビットの SQL Server のローカル インスタンスもリストします。以前は、インストールが 64 ビット ターゲット システムで実行された場合、SQL 関連のビルトイン ランタイム ダイアログは 64 ビットの SQL Server のローカル インスタンスをリストせず、32 ビットのローカル インスタンス、32 ビットのネットワーク インスタンス、および 64 ビットのネットワーク インスタンスのみがリストされました。

この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで利用できます。

64 ビット レジストリの変更についての InstallScript ログ記録とアンインストール

今回より、64 ビット ターゲット システム上で InstallScript または InstallScript MSI インストールが行われるときに、デフォルトで InstallScript エンジンがレジストリの 64 ビット部分に行われる変更をログ記録します。さらに、InstallScript エンジンによってログ記録された 64 ビット レジストリの変更が、今回より、アンインストール中にアンインストールされます。

以前、InstallScript エンジンがインストール中にレジストリの 64 ビット部分に変更を加えた場合、これらの変更は 64 ビット特定の変更としてログ記録されませんでした。そのためアンインストール中に削除も行われませんでした。

Microsoft SQL Server Management Studio のサポート

[SQL スクリプト] ビューでは、今回より、Microsoft SQL Server Management Studio がサポートされています。プロジェクト内の SQL スクリプト ファイルを SQL Server Management Studio で開くには、[SQL スクリプト] ビューで SQL スクリプトを右クリックしてから、[Microsoft SQL Server Management Studio でスクリプトを開く] をクリックします。この新しいコマンドは、同じコンテキスト メニューにあった [Microsoft Query Analyzer でスクリプトを開く] の代わりとなります。

この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで利用できます。

Visual Studio Web セットアップ プロジェクトからプロジェクト出力を追加するサポート

Web セットアップ プロジェクトを含む Visual Studio ソリューションと、InstallShield インストール プロジェクトを作成した場合に、Visual Studio 内部から InstallShield を使用しているとき、今回より、Web セットアップ プロジェクトのプロジェクト出力を InstallShield プロジェクトに追加することができます。

テキスト ファイルの変更を構成するときに Windows Installer プロパティを選択できる機能

[テキスト ファイルの変更] ビュー内の 2 つの設定について、テキスト ファイルのコンテンツにおける検索置換動作をより簡単に構成できるように強化されています。[テキスト ファイルの変更] ビューで置換ノードを選択すると右側のペインに表示される "検索する文字列" 設定と "置換後の文字列" 設定では、今回より、リストから Windows Installer プロパティを選択するか、文字列を入力することができます。このリストには、プロジェクトの [プロパティ マネージャー] ビューで使用可能なすべてのプロパティが含まれています。以前、これらの設定では手動で文字列またはプロパティを入力できましたが、プロパティのリストは表示されませんでした。

この強化は、基本の MSI および InstallScript MSI プロジェクトで使用できます。

IISROOTFOLDER で環境変数パスを解決する InstallScript テキスト置換の強化

InstallScript インストールには、IIS PathWWWRoot レジストリ値に環境変数が含まれている場合に、IIS データを IISROOTFOLDER にインストールするためのサポートが追加されました。以前、IIS サポートを含む InstallScript インストールで IISROOTFOLDER 変数のテキスト置換が行われた場合、Windows Server 208 以降のシステム上では、IISROOTFOLDER の環境変数部分が解決されませんでした。コンポーネントが IISROOTFOLDER にインストールされるように構成されている場合、未解決のパスが無効であるためにインストールが失敗しました。

Update.exe マニフェストに必要実行レベルを指定する機能

"必要実行レベル" 設定は、基本の MSI および InstallScript MSI プロジェクトの [パッチのデザイン] ビューにあるパッチの構成の [詳細] タブで設定できます。また、QuickPatch プロジェクトの場合、[一般情報] ビューの [ビルドの設定] 領域にある [詳細] タブで設定できます。

これらの新しい設定を使って、Windows Vista 以降のプラットフォーム上で Update.exe ファイルがアップグレードを実行するために必要とする最低実行レベルを指定します。InstallShield が、必要レベルを指定するマニフェストを追加します。デフォルトで、InstallShield は以前のセットアップランチャーのマニフェストで構成されたレベルを使用します。

以前、"必要実行レベル" 設定は Setup.exe セットアップランチャーでのみ使用できました。Update.exe パッチを作成するとき、InstallShield は以前のセットアップ ランチャーのマニフェストで構成された実行レベルを必要としました。

詳しくは、次を参照してください:

- [詳細] タブ ([パッチのデザイン] ビュー)
- [詳細] タブ (QuickPatch プロジェクト)

非圧縮 .cab ファイルの作成機能

[リリース] ビューで選択されたリリースの [ビルド] タブに、新しい "Cab の最適化タイプ" 設定が追加されました。"圧縮" 設定で [圧縮] またはカスタム オプションの 1 つを選択した場合、"Cab 最適化タイプ" 設定を使って、リリースの .cab ファイルをビルドするときに InstallShield が使用する圧縮の種類を指定します。使用可能なオプションには、[LZX 圧縮]、[MSZIP 圧縮]、または [非圧縮] があります。

"Cab の最適化タイプ" 設定は、これまでの "サイズの最適化" 設定に取って代わります。"サイズの最適化" 設定は、LZX 圧縮と MSZIP 圧縮のみをサポートし、非圧縮オプションは提供されていませんでした。

オートメーション インターフェイスには、この新しい設定のサポートが含まれています。ISWiRelease object オブジェクトには、新しく CabCompressionType プロパティが追加されました。このプロパティを使って、オートメーション インターフェイスを通してリリースをビルドするときに、3 つの圧縮オプションの中から 1 つを指定できます。

この強化は、基本の MSI および InstallScript MSI プロジェクトで使用できます。

詳細については、「リリースの [ビルド] タブ」を参照してください。

InstallShield 前提条件エディターにおける定義済みオペレーティング システム条件のリスト拡張

InstallShield 前提条件エディターを使って InstallShield 前提条件の設定を追加または変更するときに表示される [前提条件設定] ダイアログ ボックスに、Windows Server 2008 R2 用の定義済みオペレーティング システム条件が追加されました。一部の Windows 7 オプションは、ワークステーションのバージョン、サーバーのバージョン、またはいずれかのバージョンをチェックするため、どの Windows バージョンが選択されているかをより明確に示すように、これらのオプションの名前が変更されました。[前提条件設定] ダイアログ ボックスには、Windows Server 2008 R2 ではなく Windows 7 のみをチェックする新しいオプションも追加されました。これらの変更によって、任意の InstallShield 前提条件について、よりターゲットを絞ったオペレーティング システム条件を素早く定義することができます。

InstallShield 前提条件は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加できます。

[再配布可能ファイル] ビューと [前提条件] ビューの新しい [更新] ボタン

[再配布可能ファイル] ビュー (基本の MSI、InstallScript MSI、およびトランスフォーム プロジェクト) および [前提条件] ビュー (InstallScript プロジェクト) に表示される再配布可能ファイルのリストを更新するための、新しい [更新] ボタンが追加されました。以前は、InstallShield でいずれかのビューが開かれている状態でコンピュータに再配布可能ファイルを追加したとき、更新済みのリストを表示するためには、プロジェクトを一度閉じてから再び開く必要がありました。

この強化は、基本の MSI、InstallScript、InstallScript MSI、およびトランスフォーム プロジェクトで使用できます。

IUSR ユーザー アカウントのアクセス許可設定におけるサポート

InstallShield は、よく知られるユーザー アカウント IUSR のファイル、フォルダー、およびレジストリ キーの保護をサポートします。

プロジェクトでアクセス許可の構成にカスタム InstallShield 処理を使用する場合、[アクセス許可] ダイアログ ボックスの [ユーザー] リストに新しい [IUSR] オプションが追加されています。カスタム InstallShield 処理を使う場合、[一般情報] ビューの "ロックダウンの設定方法" 設定でこのオプションを選択します。この機能は、基本の MSI、InstallScript MSI、マージ モジュール、およびトランスフォーム プロジェクト タイプで使用できます。

InstallScript 関数 **SetObjectPermissions** を使ってアクセス許可を設定する場合、今回より、szUser パラメーターで IUSR を渡すことができます。この関数は、InstallScript プロジェクトおよび InstallScript MSI プロジェクトの InstallScript イベントで使用できます。基本の MSI、InstallScript MSI、およびマージ モジュールプロジェクトでは、InstallScript カスタム アクションを使って、この関数を使用することができます。

SetObjectPermissions 関数の新しい IS_PERMISSIONS_OPTION_ALLOW_ACCESS 定数

InstallScript 関数 **SetObjectPermissions** と使用できる、IS_PERMISSIONS_OPTION_ALLOW_ACCESS 定数が追加されました。**SetObjectPermissions** の nOptions パラメータで、数値 0 の代わりにこの定数を渡して、設定されている許可が特定のファイル、フォルダー、またはレジストリ キーへのアクセスを可能とすることを示します。

この関数は、InstallScript プロジェクトおよび InstallScript MSI プロジェクトの InstallScript イベントで使用できます。基本の MSI、InstallScript MSI、およびマージ モジュールプロジェクトでは、InstallScript カスタム アクションを使って、この関数を使用することができます。

詳細については、「SetObjectPermissions」を参照してください。

DialogSetInfo 関数の新しい定数

nInfoType パラメーターに DLG_INFO_ALTIMAGE を使用するときに、InstallScript 関数 **DialogSetInfo** の nParameter パラメーターで 2 つの定数を利用できます:

- DLG_INFO_ALTIMAGE_VERIFY_BMP – このていすうは、szInfoString で示されたビットマップを、次に続くダイアログに使用することを指定します。このビットマップを使用する前に、インストールはそのビットマップの存在を確認します。そのビットマップが存在しない場合、関数はビットマップが見つからなかったことを示すエラーを返します。
- DLG_INFO_ALTIMAGE_REVERT_IMAGE – この定数は、そのダイアログが、デフォルトのビットマップを表示することを指定します。この定数は、値 -1 の代替として使用できます。この定数を使用すると、インストールはビットマップの存在を確認しません。

nParameter パラメーターで TRUE を渡すと、インストールはビットマップの存在を確認しません。この状況でビットマップが存在しない場合、関数はエラーを返します。

この関数は、InstallScript プロジェクトおよび InstallScript MSI プロジェクトの InstallScript イベントで使用できます。

詳細については、「DialogSetInfo」を参照してください。

.NET Framework 4.0 サポートの強化

新しい FOLDER_DOTNET_40 InstallScript 変数が、追加されました。この変数は、.NET Framework 4.0 ファイルのパスを格納します。

Is 関数と使用するための 2 つの新しい定数が追加されました:

- REGDB_KEYPATH_DOTNET_40_CLIENT
- REGDB_KEYPATH_DOTNET_40_FULL

各パッチ構成の出力場所とキャッシュにおけるパス変数のサポート

[パッチのデザイン] ビューで選択されたパッチの構成の [共通] タブには、適切なフォルダを参照できる 2 つの設定 (″パッチ出力の場所″ および ″パッチ作成キャッシュ″) があります。InstallShield では、フォルダに絶対パスの代わりにパス変数を使用できるようになりました。このサポートによって、異なるディレクトリ構造を使用するビルドおよび開発マシンでリリースをビルドし易くなります。

このサポートは、開発システム上で InstallShield がパス変数を使用するように構成されている場合に活用できます。(つまり、[オプション] ダイアログ ボックスの [パス変数] タブで、パス変数のサポートを許可するように設定する必要があります。)InstallShield で絶対パスを使用するように構成した場合、[パッチのデザイン] ビューにはフォルダへの絶対パスが表示されます。

この強化は、基本の MSI および InstallScript MSI プロジェクトで使用できます。

InstallScript 関数 ServiceExistsService および ServiceGetServiceState は昇格された権限を必要としません

今後、InstallScript 関数 **ServiceExistsService** および **ServiceGetServiceState** は昇格された権限を必要としません。したがって、基本の MSI インストールの [インストール] – UI シーケンスなどで、エンド ユーザーが制限された権限を持つ場合でも、インストールでこれらの関数を呼び出すことができます。

重要な情報

InstallShield の複数エディションをインストールする

InstallShield 2020 のどれか 1 つのエディション (Premier、Professional、または Express) を同じシステムに同時にインストールすることができます。

InstallShield の複数バージョンをインストールする

InstallShield 2020 は、別のバージョンの InstallShield と同じマシン上にインストールすることが可能です。

InstallShield 2020 Standalone Build は、同じマシン上で別のバージョンの Standalone Build と共存することができます。ほとんどの場合、InstallShield がインストールされているマシン上に Standalone Build がインストールされることはありません。両方を同じマシン上にインストールしてオートメーション インターフェイスを使用したい場合は、登録およびアンインストール時の特別考慮について記載されている「Standalone Build と InstallShield を同一のマシン上にインストールする」を参照してください。

InstallShield と Visual Studio との統合

Microsoft Visual Studio の統合は 1 度に InstallShield の 1 バージョンとのみ可能です。システムで最後にインストールまたは修復された InstallShield のバージョンが Visual Studio の統合に使用されます。

InstallShield 2010 の新しい機能

Expansion Pack for Visual Studio 2010

InstallShield 2010 Expansion Pack for Visual Studio 2010 は、Visual Studio 2010 および .NET Framework 4 の最終版をサポートするための変更が含まれています。また、その他の変更も含まれています。

Microsoft .NET Framework 4 前提条件

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる .NET 関連の新しい 4 つの InstallShield 前提条件が含まれています:

- Microsoft .NET Framework 4.0 Full
- Microsoft .NET Framework 4.0 Full (Web ダウンロード)
- Microsoft .NET Framework 4.0 Client
- Microsoft .NET Framework 4.0 Client (Web ダウンロード)

これらの前提条件ついて以下の点にご注意ください:

- 完全版の前提条件は、.NET Framework 4 をターゲットにするアプリケーションを実行および開発するのに必要な .NET Framework ランタイムと関連ファイルをインストールします。
- クライアント前提条件は、ほとんどのクライアント アプリケーションを実行するのに必要な .NET Framework ランタイムと関連ファイルをインストールします。
- 2 つの Web ダウンロード前提条件には、インターネットへの接続が必要です。これらの前提条件のダウンロードには、適切な場合、再配布可能ファイルが必要です。その他 2 つの前提条件は、インターネットへの接続が不要なスタンドアロン インストールです。

.NET Framework 4.0 には、Windows Installer 3.1 以降、および Windows Imaging Component が必要です。したがって、.NET Framework 4.0 Full 前提条件は、以下の InstallShield 前提条件を依存関係とします:

- Windows Installer 3.1 (x86)
- Windows Imaging Component (x86)
- Windows Server 2003 SP1 (x86) 用 Windows Installer 3.1
- Windows Imaging Component (x64)
- Windows Server 2003 SP1 (IA64) 用 Windows Installer 3.1
- Windows Server 2003 SP1 (x64) 用 Windows Installer 3.1
- Windows XP (x64) 用 Microsoft Windows Installer 3.1

また、.NET Framework 4.0 Client 前提条件は、以下の InstallShield 前提条件を依存関係とします:

- Windows Installer 3.1 (x86)
- Windows Imaging Component (x86)
- Windows Server 2003 SP1 (x86) 用 Windows Installer 3.1
- Windows Imaging Component (x64)
- Windows Server 2003 SP1 (x64) 用 Windows Installer 3.1
- Windows XP (x64) 用 Microsoft Windows Installer 3.1

したがって、任意の .NET Framework 4.0 前提条件をプロジェクトに追加すると、InstallShield はデフォルトで Windows Installer と Windows Imaging Component の前提条件ををインストールに追加します。このため、インストールのサイズが大きくなることがあります。一部の状況において、InstallShield 前提条件エディタ (InstallShield Premier および Professional Edition で提供されています) を使って .NET Framework 4 前提条件から一部の依存関係を削除する必要があります。たとえば、製品が Windows Vista以降または Windows Server 2008 以降を必要とする場合は、以前のバージョンの Windows をターゲットとする Windows Installer 3.1 の依存関係を削除する必要があるかもしれません。

Visual Studio 2010 サポートのための追加前提条件

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる、以下の新しい InstallShield 前提条件が含まれています:

- Microsoft SQL CE 3.5 SP2
- Microsoft ReportViewer 2010
- Microsoft VSTO 2010 Runtime
- FSharp Redistributable Package 2.0
- Microsoft Office 2007 PIA (この前提条件は Microsoft Office 2007 Primary Interop Assemblies をインストールします。この前提条件を使用するには、マイクロソフトの Web サイトから PrimaryInteropAssembly.exe ファイルをダウンロードおよび実行してo2007pia.msi ファイルを抽出します。)システム上の .msi パッケージの場所によって、.prq ファイル内の o2007pia.msi インストールのパスを変更しなくてはならない場合があります。)

.NET Framework 4 用の定義済みシステム検索

InstallShield に 2 つの新しい定義済みシステム検索が追加されました:

- Microsoft .NET Framework 4.0 Full パッケージ
- Microsoft .NET Framework 4.0 Client パッケージ

インストールでこれらのいずれかが必要な場合、プロジェクト アシスタントの [インストール要件] ページまたは [システム検索] ビューを使って、これらのシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

この機能は、基本の MSI および InstallScript MSI プロジェクトに適用します。

InstallShield 2010 SP1 の新しい機能

InstallShield 2010 Service Pack 1 (SP1) には、Windows 7、Windows Server 2008 R2、および Windows Installer 4.5 の最終版をサポートするための変更が含まれています。また、その他の変更も含まれています。

重要 *• InstallShield 2010 SP1 で InstallShield 2010 を開くとき、プロジェクトを InstallShield 2010 SP1 にアップグレードすることを許可する必要があります。InstallShield 2010 SP1 では、InstallShield 2010 プロジェクトで使用できないテーブルをサポートするため、アップグレード中にこれらのテーブルを追加しなくてはなりません。*
InstallShield 2010 SP1 プロジェクトを以前のバージョンの InstallShield (SP1 を適用する前の InstallShield 2010 を含む) で開くことはできませんので、ご注意ください。このため、複数のユーザーが InstallShield プロジェクトを開いて編集する場合、すべてのユーザーが同時に SP1 パッチを適用するようにしてください。

InstallShield 2010 プロジェクトを InstallShield 2010 SP1 で開くと、そのプロジェクトを新しいバージョンに変換するかどうかをたずねるメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前にプロジェクトのバックアップ コピーが作成されます。

Setup.exe マニフェストには、Windows 7 および Windows Server 2008 R2 システム上で Program Compatibility Assistant のトリガーを回避するための、互換性セクションが追加されました。

InstallShield プロジェクトで、インストール用のセットアップランチャーの作成を構成した場合、InstallShield がセットアップランチャーに対して作成するマニフェストには、今回より互換性セクションが含まれます。以前は、この互換性セクションが含まれておらず、Windows 7 および Windows Server 2008 R2 システム上でインストールの終了時に Program Compatibility Assistant (PCA) ダイアログ ボックスが表示されました。この PCA ダイアログ ボックスは、プログラムが正しくインストールされなかった可能性があることを通知しました。このダイアログ ボックスは、インストールがアプリケーションのアンインストール キーを作成しなかった場合に表示されました。この状況は、エンド ユーザーがインストールをキャンセルした場合、またはインストールが正しく完了しなかった場合に発生します。

App-V パッケージのアップグレードの作成と App-V パッケージの圧縮サポート

InstallShield は、今回より App-V アップグレードの作成をサポートします。Microsoft App-V アシスタントの [パッケージ情報] ページに、新しい [アップグレードの設定] リンクが追加されました。このリンクをクリックして、アップグレードを作成するかどうかを指定します。アップグレードを作成すると指定した場合、App-V パッケージのファイル名の後にバージョン番号を付加するかどうかなど、追加の情報を指定することができます。デフォルトでは、App-V パッケージのアップグレードは作成されません。

Microsoft App-V アシスタントの [ビルド オプション] ページには、App-V パッケージ内のデータ ファイルの圧縮を使用するかどうかを指定できる新しい設定が追加されました。この設定に [はい] を選択すると、InstallShield は App-V パッケージに zlib 圧縮を使用します。

App-V のアップグレードおよび圧縮機能は、基本の MSI と MSI データベース プロジェクトタイプで使用できます。

Microsoft App-V アシスタントは、InstallShield Premier Edition で使用できます。

詳細については、次を参照してください。

- アップグレード パッケージ情報を指定する
- App-V パッケージ内のデータ ファイルを圧縮するかどうかを指定する

Windows サービス用の新しいカスタマイズ オプションの構成機能およびその他のサービス関連の強化機能における Windows Installer 5 サポート

InstallShield は今回より、Windows サービスに対する拡張カスタマイズ オプションの構成をサポートします。カスタマイズ オプションには、システム起動時のパフォーマンスを向上させる新しい遅延自動開始機能、強化されたエラー検出機能、およびシステムの信頼性を向上させる回復オプションなどがあります。Windows Installer 5 は、これらの新しいオプションをサポートしますが、以前のバージョンの Windows Installer はこれらを無視します。

新しいサービス関連の設定を構成するには、[セットアップのデザイン] ビュー (インストール プロジェクトの場合) または [コンポーネント] ビューで、コンポーネントの [詳細設定] 領域にある [サービス] ノードを使用します。[サービス] ノードで、コンポーネントのサービスはすべて、サービス名ごとにグループに分けてリストされています。また、以前から利用可能なサービス関連の設定は、今回より新しい設定と共に同じグリッドに統合されています。InstallShield の以前のバージョンでは、これらの以前から利用可能なサービス関連の設定は [NT サービスのインストール] ノードと [NT サービスのコントロール] ノードのサブビューに分けて表示されました。
統合された設定グリッドを使って、インストール中、またはアンインストール中に新しい、または既存のサービスをインストール、構成、開始、停止、または削除するコンポーネントを簡単に作成することができます。

この機能は、基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- Windows サービスのインストール、制御、および構成
- サービスの設定

Windows 7 ロゴ プログラムの検証

InstallShield では、新しい 2 つの検証スイート (InstallShield 検証スイート – Windows 7 および InstallShield マージ モジュール検証スイート - Windows 7) が提供されています。これらの検証スイートを使って、Windows Installer ベースのインストール、またはマージ モジュールが Windows 7 ロゴ プログラムに対応するインストール要件を満たすかどうかを検証することができます。Windows 7 ロゴを使用する場合、アプリケーションのインストールがプログラムの要件を満たしている必要があります。

InstallShield を使って、リリースが正しくビルドされるたびに、これらの検証スイートを実行するように構成する場合: [ツール] メニューから [オプション] を選択します。[検証] タブで、適切なチェック ボックスを選択します。

ビルド処理とは別に検証を行う場合: [ビルド] メニューから [検証] をポイントして、適切な新しいスイートを選択します。

ショートカットに Windows シェル プロパティを設定する Windows Installer 5 サポート

InstallShield の [ショートカット] ビューでは、実行時に Windows シェルによる設定が必要な 1 つ以上のショートカット プロパティを指定できます。たとえば、エンド ユーザーが製品をインストールした後に、そのショートカットの [スタート] メニュー エントリを新しくインストールされた製品として強調表示しないようにする場合、ショートカットのプロパティを設定できます。このプロパティは、インストールの一部であるツールまたは従属的な製品のショートカットで使用すると便利です。

Windows Installer 5 では、シェル ショートカット プロパティの設定がサポートされています。以前のバージョンの Windows Installer は、これらのプロパティを無視します。

詳細については、「ショートカットのシェル プロパティを設定する」を参照してください。

MsiPrint および MsiLaunchApp コントロール イベントに関する Windows Installer 5 サポート

プロジェクトに含まれるダイアログにコントロールを追加するとき、Windows Installer 5 で使用できる新しいイベントの 1 つを選択できます:

- **MsiPrint** – このイベントを、スクロール可能なテキスト コントロールを含むダイアログにあるプッシュ ボタン コントロールに追加します。エンド ユーザーがプッシュ ボタン コントロールをクリックすると、スクロール可能なテキスト コントロールの内容が印刷されます。
- **MsiLaunchApp** – このイベントをダイアログに含まれるチェック ボックス コントロールに追加してから、イベントの "引数" 設定で起動するファイルを選択できます。エンド ユーザーはチェック ボックスを使って、インストールの終わりにファイルを実行するかどうかを選択できます。通常このイベントは、SetupCompleteSuccess ダイアログ上のチェック ボックス コントロールと共に使用します。チェック ボックス コントロールには、アンインストール中にコントロールが表示されるのを防ぐ条件を含みます。

Windows Installer 5 で、これらのコントロール イベントがサポートされています。以前のバージョンの Windows Installer は、これらのイベントを無視します。したがって、Windows Installer 4.5 以前が搭載されているシステム上でインストールを実行するときに、これらの新しいイベントの 1 つまたは両方を使用する場合、ダイアログ コントロールに条件を追加することで、Windows Installer 4.5 以前が搭載されたシステム上でそれらが表示されないようにします。

この機能は、基本の MSI プロジェクトで使用できます。

プロジェクトに印刷または起動サポートを追加するときに、インストールのターゲット システムに Windows Installer 4.5 以前が搭載されている場合、InstallShield が提供するサポートの使用をお勧めします。印刷サポートの追加に関する情報については、「プロジェクト アシスタントでインストールに使用するダイアログを指定する」を参照してください。印刷サポートの追加に関する情報については、「[印刷] ボタンをダイアログに追加する」を参照してください。InstallShield で提供されているサポートに、Windows Installer 5 は不要です。

追加の変更

InstallShield 2010 SP1 で解決されている問題の一覧は、リリース ノートをご覧ください。リリース ノートは、InstallShield の [ヘルプ] メニューからご覧になることができます。

InstallShield 2010 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

カスタマイズされた仮想アプリケーションの作成をサポート

InstallShield を使って、Microsoft App-V フォーマットでカスタマイズされた仮想アプリケーションを作成できます。仮想化技術を使って、アプリケーションを独自の環境に隔離することで、既存アプリケーションとの競合を回避したり、基盤となるオペレーティング システムの変更を避けたりすることが可能です。

仮想アプリケーションは、アプリケーション層とオペレーティング システム層を別々に維持する仮想環境で実行されます。各アプリケーションには、その仮想環境に、それ自身の構成情報を持ちます。これにより、多数のアプリケーションは、競合することなく、同一のコンピュータで他のアプリケーションとサイド バイ サイドで実行が可能になります。

仮想アプリケーションを作成するには、基本の MSI および MSI データベース プロジェクトで使用できる、新しい Microsoft App-V タブを使います。

仮想化サポートは、InstallShield Premier Edition で提供されています。

また、InstallShield には Microsoft App-V 4.5 Desktop Client インストールおよび Microsoft Application Error Reporting インストール用の InstallShield 前提条件が含まれています。Application Error Reporting 前提条件は、App-V 前提条件の依存関係です。これらの InstallShield 前提条件用の再配布可能ファイルは Microsoft から入手しなくてはならないため、InstallShield 内部からダウンロードすることはできません。Microsoft からそれらのファイルを入手した後、InstallShield 前提条件エディターで前提条件を編集するときに表示される場所に配置してください。

詳細については、次を参照してください。

- カスタム仮想アプリケーションの作成
- 仮想化について
- Microsoft App-V アプリケーションの作成

Windows 7 および Windows Server 2008 R2 システムをターゲットにできる機能

InstallShield では、インストールに Windows 7 または Windows Server 2008 R2 が必要であることを指定できます。また、これらのオペレーティング システムに対する機能条件およびコンポーネント条件をビルドすることができます。

Windows 7 および Windows Server 2008 R2 でタスクバーにインストールの進行状況を表示するサポート

Windows 7 と Windows Server 2008 R2 で実行されるインストールは、今回より、ファイル転送中に Windows タスクバーに進行状況バーを表示します。この情報は、InstallScript および InstallScript MSI インストールに適用します。また、[ビルボード] ビューで構成されたビルボードを表示する基本の MSI インストールにも適用します。進行状況バーは、Windows の以前のバージョンのタスクバーには表示されません。また、セットアップの初期化中、または InstallShield 前提条件のインストール中にも表示されません。

Beta Windows Installer 5 におけるユーザーごとインストールをサポート

[一般情報] ビューには、"[ユーザーごと] オプションの表示" 設定があります。この設定を使って、特定の状況下において ReadyToInstall ダイアログにエンド ユーザーが製品をインストールする方法 (現在のユーザーまたはすべてのユーザー) を指定できるボタンを含めるかどうかを指定できます。[ユーザーごと] ボタンは、Windows Installer プロパティ MSIINSTALLPERUSER を 1 に等しく設定して、パッケージを現在のユーザーにインストールすることを示します。MSIINSTALLPERUSER プロパティは Windows Installer 5 で使用できます。

この機能は、基本の MSI プロジェクトで使用できます。

詳細については、「ユーザーごとのインストールとマシンごとのインストールの違い」を参照してください。

Windows Installer 5 Beta における大きいパッケージのインストールにかかる所要時間の短縮機能をサポート

[一般情報] ビューにある "高速インストール" 設定では、大きい Windows Installer パッケージのインストールにかかる所要時間を短縮するのに役立つ 1 つまたは複数のオプションを選択できます。たとえば、インストールについてシステム復元ポイントを保存しないことを指定できます。インストールがファイル コスティングのみを実行して、その他のコスト チェックをスキップするように指定することも可能です。

この設定は、新しい Windows Installer プロパティ MSIFASTINSTALL を構成します。このプロパティは、コマンドラインで設定できます。Windows Installer 5 で、このプロパティがサポートされています。以前のバージョンの Windows Installer はこれを無視します。

この設定は、基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、"高速インストール" 設定の説明を参照してください。

Windows Installer 5 追加機能のベータ サポート

InstallShield のダイレクト エディターには、Windows Installer 5 テーブル (MsiLockPermissionEx、MsiServiceConfig、MsiServiceConfigFailureActions、および MsiShortcutProperty) のベータ サポートが含まれています。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

仮想マシンの存在を検出できる機能

InstallShield を使って、インストールが以下の種類の仮想マシンで実行されているかどうかを検出できます:

- Microsoft Hyper-V
- VMware Player、VMware Workstation、または VMware Server などの VMware 製品
- Microsoft Virtual PC

MSI DLL カスタム アクションをプロジェクトに追加する場合に、仮想マシンを検出するための 2 つの新しい Windows Installer プロパティIS_VM_DETECTED と IS_VM_TYPE が追加されました。カスタム アクションを、InstallShield と共にインストールされる SetAllUsers.dll ファイルで ISDetectVM 関数を呼び出すように構成します。

また、InstallScript 言語が検出をサポートできるように拡張されました。構造 SYSINFO は、新しい bIsVirtualMachine メンバーを含み、InstallScript 関数 **GetSystemInfo** と使用できる、新しい VIRTUAL_MACHINE_TYPE 定数が追加されました。

この機能は、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、およびマージ モジュール プロジェクト タイプで使用できます。

詳細については、「インストールが仮想マシン上で実行されているかどうかを検出する」を参照してください。

64 ビット COM 抽出サポート

InstallShield は、今回より、64 ビット COM 抽出をサポートします。InstallShield を 64 ビット オペレーティング システムで使用している場合、InstallShield は 64 ビット COM サーバーから COM データを抽出できます。データを正しい場所にインストールするため、コンポーネントを 64 ビットとマークしなくてはなりません。

64 ビット COM 抽出を実行するためには、InstallShield が、64 ビット オペレーティング システムにインストールされていなくてはなりません。

この機能は、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクト タイプで使用できます。

詳細は、次を参照してください:

- 64 ビット オペレーティング システムをターゲットにする
- COM 情報を COM サーバーから抽出する

ファイル、フォルダー、およびレジストリ キーのアクセス許可を設定するための新しいサポート

InstallShield では、ロックダウン環境で製品を実行するエンド ユーザー向けに、ファイル、フォルダー、およびレジストリ キーを保護するための新しい 2 つの方法が提供されています:

- **カスタム InstallShield 処理** — Windows Installer ベースのプロジェクトでは、実行時にアクセス許可を設定するためのカスタム サポートの使用を選択できます。このオプションを使うと、InstallShield は .msi データベースのカスタム ISLockPermissions テーブルに製品のアクセス許可情報を格納します。InstallShield はまた、アクセス許可を設定するためのカスタム アクションをプロジェクトに追加します。このサポートは、基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。
- **InstallScript 関数、SetObjectPermissions** — InstallScript イベントおよび InstallScript カスタム アクションで新しい **SetObjectPermissions** 関数を使って、実行時にアクセス許可を設定できます。この関数は、InstallScript、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクト タイプで使用できます。

[カスタム InstallShield 処理] オプションを使う場合、アクセス許可を設定するファイル、フォルダー、またはレジストリ キーを、インストールの一部としてインストールしなくてはなりません。**SetObjectPermissions** 関数を使う場合、ファイル、フォルダー、またはレジストリ キーをインストールの一部としてインストールするか、ターゲット システム上に既存するかどうかを問いません。

以前、InstallShield でアクセス許可を設定するための唯一のオプションは、従来型の Windows Installer 処理のみでした。このオプションを使うと、.msi データベースの LockPermissions テーブルに製品のアクセス許可情報が格納されます。新しい [カスタム InstallShield 処理] オプションと **SetObjectPermissions** 関数には、[従来型の Windows Installer 処理] オプションよりも多くの利点があります。

- カスタム オプションおよび **SetObjectPermissions** 関数を使うと、[従来型の Windows Installer 処理] オプションではサポートされていない、多くのセキュリティ識別子 (SID) を使用できます。
- カスタム オプションと **SetObjectPermissions** 関数は、従来型のオプションとは異なり、サポート対象の SID の翻訳されたユーザー名をサポートします。従来型のオプションで、非英語システム上で翻訳された名前を使ってアクセス許可を設定すると、インストールが失敗する可能性があります。
- カスタム オプションおよび **SetObjectPermissions** 関数を使って、指定するアクセス許可を特定のユーザーまたはグループが所持することを拒否できます。従来型の処理で、これは不可能です。
- カスタム オプションおよび **SetObjectPermissions** 関数を使って、ターゲット システムに既存するファイル、フォルダー、またはレジストリ キーに、アクセス許可を追加できます。このとき、そのオブジェクトに既存するアクセス許可は削除されません。従来型の処理では、既存するアクセス許可が削除されました。
- カスタム オプションおよび **SetObjectPermissions** 関数を使って、フォルダー (またはレジストリ キー) のアクセス許可を構成し、そのフォルダーのサブフォルダーおよびファイル (またはレジストリ キーのサブキー) すべてに同じアクセス許可を適用するかどうかを指定できます。従来型の処理では、フォルダー内のサブフォルダーまたはファイル (レジストリ キーの下にあるサブ キー) にアクセス許可を構成する場合、ターゲット システム上で作成された親は、自動的に子のアクセス許可を継承します。
- カスタム オプションおよび **SetObjectPermissions** 関数を使って、インストール中に作成される新しいユーザーのアクセス許可を構成できます。従来型の処理では、実行時にユーザーがターゲット システム上に既存する必要があったため、これは不可能でした。

[一般情報] ビューに追加された "ロックダウンの設定方法" 設定を使って、プロジェクトに含まれるファイル、フォルダー、およびレジストリ キーに設定する新しいアクセス許可について、新しいカスタム InstallShield 処理または従来型の Windows Installer 処理のどちらを使用するかを指定できます。プロジェクトで既にいくつかのアクセス許可を構成済みの場合にこの設定の値を変更する場合、InstallShield では、既存アクセス許可に別の処理方法を使用するかどうかを指定できます。すべての新しいプロジェクトにおける、この設定のデフォルト値は [カスタム InstallShield 処理] オプションです。プロジェクトを InstallShield 2009 以前から InstallShield 2010 にアップグレードした場合、この設定のデフォルト値は [従来型の Windows Installer 処理] オプションです。この新しい設定は、基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで利用できます。

詳細は、次を参照してください:

- ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護
- プロジェクトで、ロックダウン環境でのアクセス許可タイプを選択する
- SetObjectPermissions
- SetObjectPermissions の例

InstallScript プロジェクトにおける InstallShield 前提条件のサポート

InstallShield では、InstallScript プロジェクトに InstallShield 前提条件を追加できるようになりました。以前、この種類の再配布可能ファイルは基本の MSI プロジェクトおよび InstallScript MSI プロジェクト プロジェクトでのみサポートされていました。

InstallShield 前提条件 は、通常、製品が必要とする製品またはテクノロジ フレームワークをインストールする再配布可能ファイルです。今回より、InstallShield で提供されている InstallShield 前提条件または独自で作成したカスタム InstallShield 前提条件をどれでも InstallScript プロジェクトに追加できます。異なる種類のプロジェクト タイプを組み合わせている場合、InstallShield ではすべての基本の MSI、InstallScript、および InstallScript MSI プロジェクトに含まれるこの種類の再配布可能ファイルを再利用できるため、簡単にマトリックスのテストを行うことができます。

InstallShield 前提条件を InstallScript プロジェクトに追加するには、新しい [前提条件] ビューを使います。InstallScript プロジェクトでは、InstallShield 前提条件タイプのセットアップ前提条件をサポートします。これは、メイン インストールのユーザー インターフェイスが実行される前に実行します。InstallScript プロジェクトは、機能と関連付けられた InstallShield 前提条件である、機能前提条件をサポートしません。

詳細については、次を参照してください。

- InstallShield 前提条件、マージ モジュール、およびオブジェクトを InstallScript プロジェクトに追加する
- InstallShield 前提条件を含むインストールの実行時の動作
- InstallShield 前提条件のインストール順を指定する
- 特定の InstallShield 前提条件の実行時の場所を指定する
- リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する
- InstallShield 前提条件およびその他の再配布可能ファイルをデザインする
- [前提条件] ビュー

Windows Installer、.NET Framework、Crystal Reports、およびその他の再配布可能ファイルのための新しい InstallShield 前提条件

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる多くの InstallShield 前提条件が含まれています。

- Windows Installer 4.5 (Windows Installer 4.5 用の InstallShield 前提条件は、Microsoft KB958655 の修正を含みます。)
- Windows Installer 4.5 Update (Windows Installer 4.5 用の InstallShield 前提条件は、Microsoft KB958655 の修正を含みます。このアップデートには、Windows Installer 4.5 がターゲット システム上にインストール済みでなくてはなりません。)
- Windows Installer 3.1、Windows Installer 3.0、および Windows Installer 2.0 (これらのバージョンの Windows Installer 再配布可能ファイルは、以前、[リリース] ビューを使ってプロジェクトに Windows Installer を追加すると利用することができました。InstallShield 前提条件としては提供されていませんでした。)
- .NET Framework 3.0 SP1
- .NET Framework 2.0 SP2
- Internet Explorer 8
- Microsoft SQL Server 2008 Express SP1
- Microsoft SQL Server 2005 Express SP3
- Microsoft Visual C++ 2005 SP1 Redistributable Package
- Oracle 11g Instant Client 11.1.0.7 (Oracle は、Oracle Instant Client ファイル用のインストーラーを提供していません。したがって、プロジェクトでこの InstallShield 前提条件を使用する前に、.msi パッケージを作成する必要があります。これは、InstallShield と共にインストールされる Oracle Instant Client インストール プロジェクト (*InstallShield Program Files フォルダー*¥Support¥Oracle Instant Client) を使って簡単に作成できます。
- Crystal Reports Basic for Visual Studio 2008 (Visual Studio 2008 と共にインストールされた Crystal Reports Basic と共に使用できる前提条件。システム上の .msi パッケージの場所によって、.prq ファイル内の Crystal Reports Basic のパスを変更しなくてはならない場合があります。)

Microsoft SQL Server 2008 SP1 サポート

InstallShield は、今回より、SQL Server 2008 SP1 上で SQL スクリプトを実行するためのサポートを含みます。また、InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストには、SQL Server 2008 SP1 が含まれています。

Oracle 11g サポート

InstallShield には、今回より、Oracle 11g で SQL スクリプトを実行するためのサポートが含まれています。また、InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストには、Oracle 11g が含まれています。

IIS Web サイトのスキャン、その設定の記録、およびそれらの設定を InstallShield プロジェクトにインポートする新しいツール

InstallShield には、IIS スキャナー (IISscan.exe) が搭載されています。この新しいコマンドライン ツールを使って、既存の IIS Web サイトをスキャンし、その Web サイトに関する IIS データを記録できます。IIS スキャナーは、Web サイト、その仮想ディレクトリ、アプリケーション、およびアプリケーション プールすべての設定を含む XML

ファイルを作成します。この XML ファイルを使って、InstallShield の [IIS 構成] ビューに IIS データをインポートできます。IIS データをプロジェクトにインポートしてから、必要に応じて、[IIS 構成] ビューを使って IIS の設定を変更できます。

IIS データを InstallShield プロジェクトにインポートできる機能は、InstallShield Premier Edition のみで使用できます。

詳細については、次を参照してください。

- IIS Web サイトのスキャンをして、その設定を InstallShield プロジェクトにインポートする
- Web サイトおよびその設定を InstallShield プロジェクトにインポートするときに IIS データをフィルターする
- IISscan.exe

IIS Web アプリケーションを Web サイトに追加できる機能と、その他の IIS 関連機能の強化

InstallShield では、今回より、IIS Web アプリケーションを Web サイトに追加できます。そのためには、[IIS 構成] ビューで Web サイトを右クリックしてから、[新しいアプリケーション] をクリックします。新しいアプリケーションを追加した後、その設定を右側のペインで構成できます。

また、InstallShield ではアプリケーション無しで仮想ディレクトリを作成することもできます。以前は、仮想ディレクトリを作成するたびに、アプリケーションも自動的に作成されました。

また、InstallShield には次の新しい IIS 設定が追加されました:

- **マネージ パイプライン モード** – 適切な要求処理パイプライン モード (統合またはクラシック) を指定できます。
- **32 ビット アプリケーションを有効にする** – 選択したアプリケーション プールの 32 ビット アプリケーションを 64 ビット システム上で実行可能にするかどうかを指定できます。
- **.NET Framework バージョン** – アプリケーション プールがロードする .NET Framework のバージョンを指定できます。
- **ASP.NET プラットフォーム** – .NET Framework がインストールされている Windows の 64 ビット バージョンでインストールを実行できるようにする場合、どの ASP.NET プラットフォームを使って Web サイト、アプリケーション、または仮想ディレクトリを ASP.NET バージョンにマップするのかを指定します。

この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプで使用できます。

詳細は、次を参照してください:

- Web サイトの作成とアプリケーションまたは仮想ディレクトリの追加
- [IIS 構成] ビュー

ランタイム テキスト ファイルの変更を構成できる新しいビュー

InstallShield に新しく追加された [テキスト ファイルの変更] ビューを使って、ターゲット システム上で実行時に変更を行うテキスト ファイル (たとえば、.txt、.htm、.xml、.config、.ini、および .sql) 内の検索/置換処理を構成できます。テキスト ファイルは、インストールの一部またはシステム上に既存するファイルのどちらでも構いません。

Windows Installer プロパティを使って、検索に含める、または検索から除外するテキスト ファイルの名前を指定することができます。プロパティを使って、検索文字列と置換文字列を指定することもできます。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、製品のテキスト ファ

イルがターゲット システムで変更される時に使用できるようになります。たとえば、エンド ユーザーが IP アドレスを指定しなくてはならないダイアログがプロジェクトに含まれている場合、インストールは一連のファイル内でトークンを検索して、エンド ユーザーが入力した IP アドレスで、それを置換します。

[テキスト ファイルの変更] ビューは、[XML ファイルの変更] ビューで XML ファイルを構成する代わりの方法として使用できます。[テキスト ファイルの変更] ビューには、いくつかの利点があります。たとえば、この新しいビューにはランタイム要件がありません。一方、[XML ファイルの変更] ビューを使って構成された変更の場合、ターゲット システムで MSXML が必要です。また、[テキスト ファイルの変更] ビューで変更を構成する場合、XPath クエリの入力は不要ですが、[XML ファイルの変更] ビューを使った場合は、それが必要です。

[テキスト ファイルの変更] ビューは、基本の MSI プロジェクトおよび InstallScript MSI プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- テキスト ファイルの変更
- テキスト 変更セットを作成する
- テキスト ファイルの変更を指定する
- テキスト ファイルの変更が行われる順番を変更する
- Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する
- ANSI テキスト ファイルを開く時に使用するコード ページを指定する

新しい [文字列エディター] ビュー

InstallShield に [文字列エディター] ビューが追加されました。このビューには、プロジェクト内の言語非依存文字列 ID と、それにに対応する言語固有の値がスプレッドシート形式のテーブルで表示されます。[文字列エディター] ビューでは、インストール処理で実行時に表示されるローカライズ可能なすべてのテキスト文字列を、1 ヶ所でまとめて制御することができます。このビューは、以前 [一般情報] ビュー内のノードで表示された文字列テーブルの代わりとなります。以下は、新しいビューの主な特徴を説明します:

- ビューには文字列エントリを追加、編集、削除、検索、置換、エクスポート、およびインポートするためのボタンを含むツールバーがあります。特定の文字列 ID が使用されているすべてのインスタンスを識別するために、プロジェクト内を検索するためのボタンもあります。
- ビューの上部には新しいグループ ボックス領域があり、列ヘッダーをこの領域にドラッグ アンド ドロップすることで、ビュー内の行を階層形式に整列することができます。この機能を使って、言語または更新日などのカテゴリ別に文字列エントリを簡単に並べ替えることができます。
- このビューではダイナミック検索もサポートされていて、検索ボックスに文字列を一文字づつ入力するのと同時に InstallShield がその文字を含まない文字列を隠します。

この機能は、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、およびマージ モジュール プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- [文字列エディター] ビュー
- エンドユーザー インターフェイスのローカライズ

Unicode サポート

InstallShield は次の 3 つの側面から、モダン言語の複数言語インストールを完全にサポートします: Windows Istaller データベースを Unicode 形式でビルド可能、InstallShield プロジェクトを Unicode 形式で保存可能、InstallShield インターフェイスで複数文字セットから同時に Unicode 文字を入力および表示可能

Unicode (UTF-8) データベース

[リリース] ビューの [ビルド] タブにある "UTF-8 データベースのビルド" 設定では、Windows Installer データベースを指定して、任意のインスタンスまたは言語トランスフォームと共に UTF-8 エンコードでビルドできます。UTF-8 エンコードは、すべての言語の文字を同時にサポートするため、エンド ユーザーに表示するテキストおよびファイル名とレジストリ キーの両方で、たとえば日本語とドイツ語、またはロシア語とポーランド語のように自由に言語を組み合わせて使用できます。組み合わされた言語は、ターゲット システムの現在の言語設定に関わらず正しく表示されます。Unicode サポートはまた、IIS サポートやテキストおよび XML ファイルの変更をはじめとするインストール実行時の主要な部分にも追加されています。

新しい "UTF-8 データベースのビルド" 設定のデフォルト値は [いいえ] です。

オートメーション インターフェイスには、この新しい設定のサポートが含まれています。ISWiRelease オブジェクトには、UTF-8 エンコードを使用するかどうかを指定できる、新しい BuildUTF8Database プロパティが含まれています。

この機能は、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクト タイプで使用できます。

さらに詳しい情報は、"UTF-8 データベースのビルド" 設定の説明を参照してください。

Unicode プロジェクト ファイル (.ism)*

InstallShield は、今回より、バイナリおよび XML プロジェクト ファイルを保存するときに UTF-8 エンコードを使用します。この変更により、プロジェクトで使用されるファイル、レジストリ データ、その他の文字列には、同時にすべての言語の文字を含めることが可能です。このエンコードにより、InstallShield では今後 ISString テーブルに格納される文字列に人間が判読不可能な Base64 エンコードを使用する必要はありません。代りに、プロジェクトに文字列を追加または変更するとき、InstallShield はそれらを判読可能な Unicode 文字列として保存します。これによって、プロジェクトの修正時に変更点を容易に確認することができます。このため、InstallShield 2010 で作成されるすべての新規プロジェクトでは Unicode 文字列のみが使用されます。アップグレード プロジェクトでは、新しい文字列または変更された文字列、およびエクスポート後に再インポートされた文字列には Unicode が使用されます。

ターゲットのビルドで表示できない Unicode 値を使用すると (たとえば、"UTF-8 データベースのビルド" 設定で [いいえ] が選択されている InstallScript インストールまたは基本の MSI インストール)、新しいビルド エラーが変更を必要とする項目をポイントします。一部において、InstallShield の以前のバージョンではサイレントで受け入れられていた無効な文字列エントリを検出する場合もあります。

この機能はすべてのプロジェクト タイプに適用します。

InstallShield の Unicode ビュー

InstallShield の多くのビューは、すべての言語からの文字を表示および入力できるように強化されています。たとえば、[ファイルとフォルダー] ビューでは、英語ローカル システムにある中国語のファイル名が疑問符で表示されることがなくなり、これらのファイルをプロジェクトに追加できるようになりました。同様に、[レジストリ] ビューでタイ語のレジストリ キーまたは値が疑問符に変換されることがなくなり、これらを Windows Installer ベースのプロジェクトにインストールできるようになりました。さらに、新しい独立した [文字列エディター] ビューで、すべての言語からの文字列を表示および変更できるようになりました。以前、文字列エントリは [一般情報] ビューで個別の言語ノードに表示されました。その他、[プロパティ マネージャー]、[パス変数]、[ダイレクト エディター]、[一般情報]、および [セットアップのデザイン] ビューも強化されています。

″UTF-8 データベースのビルド″ 設定を [いいえ] に設定する場合は常に、すべてのファイル名、レジストリ キー、およびその他の文字列をターゲット言語のコードページからの文字で構成しなくてはなりません。その場合、ある項目がターゲット言語のコード ページで使用不可能な文字を含む場合、InstallShield はその項目をポイントする新しいビルド エラーを報告します。つまり、″UTF-8 データベースのビルド″ 設定に [はい] が選択されていない限り、中国語ファイル名を英語インストールで使用することはできません。

この機能はすべてのプロジェクト タイプに適用します。

基本の MSI プロジェクトで新しく追加された、ビルボード サポート

基本の MSI プロジェクトには、ビルボード サポートが含まれています。ビルボードをプロジェクトに追加して、インストール処理中にエンド ユーザーに対して情報を提供できます。ビルボードは、エンド ユーザーと連絡を取ったり、広告、教育、およびエンターテインメントを提示するために使用することができます。たとえば、ビルボードを使ってインストール中の製品に含まれる新しい機能の概要や貴社の他の製品についての情報を提供できます。各ビルボードは、貴社のグラフィック担当者がファイル転送の外観を完全にカスタマイズできるファイルです。

基本の MSI プロジェクトにおけるビルボード サポートの主な特徴は次のとおりです:

- プロジェクトに Adobe Flash アプリケーション ファイル (.swf) をビルボードとして追加できます。Flash アプリケーション ファイルは、ビデオ、動画、音声、インタラクティブ インターフェイス、ゲーム、テキスト、その他の .swf ファイルがサポートするあらゆる要素で構成されます。
- InstallShield では .bmp、.gif、.jpg、および .jpeg ファイルをビルボードとして使用できます。
- InstallShield には ″ビルボードの種類″ 設定が含まれていて、インストールで使用するビルボードのスタイルを指定できます。たとえば、インストールが全画面背景を使用し、ビルボードを前景に、また小さい進行状況ボックスを画面の右下に表示するスタイルがあります。別のスタイルでは、インストールがビルボードを表示する標準サイズのダイアログを表示します。このダイアログの下の部分に、進行状況バーが表示されます。
- InstallShield では、リリースをビルドおよび実行せずに、実行時にビルボードがどのように表示されるのかをプレビューできます。ビルボードをプレビューすると、そのビルボードに現在構成されている背景色、位置、および関連設定を使ったビルボードの外観を確認できます。

InstallShield の [ビルボード] ビューでは、ビルボード ファイルの追加、その関連設定の構成、ならびにビルボードのプレビューを行います。

詳しくは、次を参照してください:

- 基本の MSI インストールでビルボードを表示する
- 基本の MSI プロジェクトにおけるビルボード ファイルの種類
- 基本の MSI プロジェクトにおけるビルボードの種類
- 基本の MSI プロジェクトに Adobe Flash アプリケーション ファイル ビルボードを追加する
- 基本の MSI プロジェクトにイメージ ビルボードを追加する
- 基本の MSI プロジェクトでビルボードの設定を構成する
- リリースをビルドまたは起動せずにビルボードをプレビューする
- 基本の MSI プロジェクトでビルボードの順序を設定する
- ビルボードを含む基本の MSI インストールにおける実行時の動作

以前は、InstallScript とInstallScript MSI プロジェクトでのみ、ビルボードがサポートされていました。これらのプロジェクト タイプに含まれるビルボード サポートは、基本の MSI プロジェクトに含まれるサポートとは異なりますので、ご注意ください。詳細については、「ビルボードを表示する」を参照してください。

InstallScript および InstallScript MSI プロジェクトで拡張されたビルボード サポート

InstallScript および InstallScript MSI プロジェクトでは、今回より .gif、.jpg、および .jpeg ファイルをビルボードとして使用できます。以前は、.bmp ファイルしかサポートされていませんでした。[サポート ファイル/ビルボード] ビューを使って、プロジェクトにビルボードを追加できます。

InstallScript および InstallScript MSI プロジェクトのビルボード ファイルは、指定された命名規則に従わなくてはなりません。各ファイル名は **bbrd** で始まり、ビルボード番号 (1 から 99) が続きます。最後に、サポート対象のファイル拡張子 (.bmp、.gif、.jpg、または .jpg) で終わります。ファイル名 番号が連続番号である必要はなくなりました。つまり、プロジェクトに bbrd1.jpg、bbrd3.jpg、および bbrd5.jpg を追加すると、実行時に各イメージが順番通りに表示されます。以前、シーケンス内にあるビルボード ファイル名の番号は連続している必要がありました。たとえば、bbrd1.bmp、bbrd2.bmp、および bbrd4.bmp をインストール プロジェクトに追加した場合、bbrd1.bmp と bbrd2.bmp は表示されますが、bbrd4.bmp はシーケンスに bbrd3.bmp がないため表示されません。

詳細については、次を参照してください。

- InstallScript および InstallScript MSI インストールでビルボードを表示する
- InstallScript または InstallScript MSI プロジェクトにおけるビルボード ファイルの名前
- InstallScript または InstallScript MSI プロジェクトにイメージ ビルボードを追加する

InstallScript 関数 PlayMMedia を使って Adobe Flash Application File (.swf) を再生できる機能

InstallScript 関数 **PlayMMedia** が、今回より、InstallScript と InstallScript MSI インストール中に背景ウィンドウで Adobe Flash application file (.swf) の再生をサポートします。Flash アプリケーション ファイルは、ビデオ、動画、音声、インタラクティブ インターフェイス、ゲーム、テキスト、その他の .swf ファイルがサポートするあらゆる要素で構成されます。

Flash ファイルを使用する場合、**SizeWindow** と **PlaceWindow** を使って、Flash ファイルを表示する背景ウィンドウのサイズと配置を制御できます。

詳細については、次を参照してください。

- PlayMMedia
- SizeWindow
- PlaceWindow
- InstallScript および InstallScript MSI インストールで背景ウィンドウを表示する

InstallScript と InstallScript MSI インストール中、ダイアログにおける HTML コントロールをサポート

InstallShield では、InstallScript と InstallScript MSI プロジェクトに含まれるダイアログで HTML コントロールをサポートします。HTML コントロール を使って、ダイアログ コントロールで HTML マークアップを使用できます。ダイアログで Web ページ、インストール済みの HTML ファイル、および HTML サポートファイルへのリンクを含めることができます。エンド ユーザーが実行時のダイアログでハイパーリンクをクリックしたときに、インターネット ブラウザーで HTML ページを開くか、InstallScript コードを使って定義した別の動作をトリガーできます。HTML コントロールを使って、外観を制御するためのスタイルを含む、有効な HTML マークアップを使用できます。

HTML コンテンツがインストール済みの HTML ファイルまたは HTML サポート ファイルである場合、HTML コントロールを使って、それを直接ダイアログに表示することも可能です。

CtrlGetUrlForLinkClicked という名前の新しい InstallScript 関数が追加されました。この関数は、エンド ユーザーがクリックしたリンクへの URL を取得します。

詳細については、次を参照してください。

- ダイアログで HTML コントロールを使用する
- CtrlGetUrlForLinkClicked
- CtrlGetUrlForLinkClicked の例

サポートされていない言語を InstallScript プロジェクトに追加できる機能

InstallShield Premier Edition には、InstallScript プロジェクト用に 33 ヶ国語のデフォルト ランタイム文字列が含まれています。InstallShield Premier Edition では、新しい言語ウィザードを使って 33 ヶ国語以外のサポートされていない言語を InstallScript プロジェクトに追加することができます。サポートされていない言語を InstallScript プロジェクトに追加した後、[文字列エディター] ビューを使ってその言語用の翻訳された文字列を提供できます。別の手段として、言語の文字列エントリをファイルにエクスポートして、そのファイルの文字列値を翻訳してから、プロジェクトにそれをインポートすることもできます。以前、新しい言語ウィザードを使ってサポートされていない言語を追加できるのは、基本の MSI プロジェクトと InstallScript MSI プロジェクトのみでした。

詳細については、次を参照してください。

- InstallShield 実行時の言語サポート
- 新しい言語ウィザード
- [文字列エディター] ビュー

InstallScript コンパイラがリンクするライブラリのパスを指定できる機能

[設定] ダイアログ ボックスにある [コンパイル/リンク] タブに新しく追加された [追加ライブラリ パス] ボックスを使って、InstallScript コンパイラがスクリプトにリンクする InstallScript ライブラリ (.obl ファイル) を検索する場所を指定できます。さらに、このタブでカスタム ライブラリを指定するとき、フル パスではなく、ファイル名のみを指定できるようになりました。これらの変更によって、ライブラリを別のディレクトリに移動させた場合でも、[コンパイル/リンク] タブでライブラリのパスを手動で変更せずにスクリプトを正常にコンパイルすることができます。

詳細については、「[コンパイル/リンク] タブ」を参照してください。

新しい FlexNet Connect 11.6 再配布可能ファイル

InstallShield はまた、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで FlexNet Connect 11.6 をサポートします。InstallShield の [アップデート通知] ビューを使って、2 つの FlexNet Connect 11.6 マージ モジュールうち、いずれかが含まれています (Common Software Manager が含まれているマージ モジュールと、含まれていないマージ モジュール)。これらのマージモジュールは、FlexNet Connect 11 マージ モジュールに取って代わります。

DIFx 2.1.1 サポート

InstallShield では、最新版の Driver Install Frameworks for Applications (DIFx 2.1.1) がサポートされています。Microsoft の最新バイナリ ファイルを含むこの新バージョンは、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプで使用できます。 基本の MSI、InstallScript、InstallScript MSI

強化機能

InstallShield には、以下のような強化機能が搭載されています。

ユーザビリティの強化点

InstallShield の多くのビューは、その生産性とユーザビリティが強化されています。たとえば、一部のビューにはオプションを簡単に見つけられるツールバーが含まれています。いくつかのビューでは、グリッド内の行を整列する方法をカスタマイズできます。検索機能があるビューでは、より迅速に検索が行われます。主要な例は以下のとおりです:

- [ダイレクト エディター] ビュー – このビューでテーブルを選択すると、新しいツールバーが表示されます。ツールバーには、テーブル内でデータを追加、切り取り、コピー、貼り付け、検索、および置換できるボタンがあります。また、ビューではダイナミック検索もサポートされていて、検索ボックスに文字列を一文字づつ入力するのと同時に InstallShield がその文字を含まないテーブル レコードを隠します。レコードを追加または編集するとき、InstallShield は各フィールドのヘルプ テキストを表示します。
- [プロパティ マネージャー] ビュー – このビューには、プロパティの追加および削除などの処理を行うためのボタンを含む新しいツールバーがあります。また、ビューではダイナミック検索もサポートされていて、検索ボックスに文字列を一文字づつ入力するのと同時に InstallShield がその文字を含まないテーブル レコードを隠します。ビューの上部には新しいグループ ボックス領域があり、列ヘッダーをこの領域にドラッグ アンド ドロップすることで、ビュー内の行を階層形式に整列することができます。このビューでは、マウスと SHIFT または CTRL ボタンを使って複数のプロパティを選択してから、すべてを一斉に削除することができます。
- [再配布可能ファイル] ビュー – このビューに含まれる新しいツールバーとグループ ボックス領域は、強力な検索および組織化機能を提供します。列ヘッダーをグループ ボックス領域にドラッグ アンド ドロップして、再配布可能ファイルのリストを階層形式で表示することができます。また、ツールバーの検索ボックスに文字列を入力すると、InstallShield はその文字を含まないすべての再配布可能ファイルを隠します。
- [IIS 構成] ビュー – このビューは IIS 7 に似せて外観デザインが一新され、設定がタブではなくグリッドで表示されるようになりました。グリッドには、カテゴリ別、またはアルファベット順にグリッド設定を並べ替えるためのボタンがあります。このビューのグリッドの中から設定の 1 つを選択すると、右下のペインにその設定のヘルプ情報が表示されます。
- [一般情報] ビュー – このビューのすべての設定は、ビュー内でノードに関連付けられた個別のグリッドの代わりに、1 つのグリッドで表示されます。設定はいくつかのカテゴリにグループ分けされていて、特定の設定を見つけやすくなっています。また、カテゴリ別またはアルファベット順にグリッド設定を並べ替えることができるボタンもあります。以前、このビューに表示された文字列テーブルは [文字列エディター] ビューに移動しました。
- [パス変数] ビュー – このビューには、パス変数の追加および削除などの処理を行うためのボタンが含まれる新しいツールバーがあります。このビューでまた、ダイナミック検索もサポートされていて、検索ボックスに文字列を一文字づつ入力するのと同時に InstallShield がその文字を含まない行を隠します。ビューの上部には新しいグループ ボックス領域があり、列ヘッダーをこの領域にドラッグ アンド ドロップすることで、ビュー内の行を階層形式に整列することができます。このビューでは、マウスと SHIFT または CTRL ボタンを使って複数のパス変数を選択してから、すべてを一斉に削除することができます。

また、リリースをビルド、検証、またはスクリプトをコンパイルするときに表示される [出力] ウィンドウが強化されました。[出力] ウィンドウ、またその個別のタブは、InstallShield のワークスペースの任意の側面に固定するか、独立した場所にドラッグすることができます。[出力] ウィンドウまたはそのタブの 1 つを InstallShield インターフェイスの端にドラッグすると、固定ウィンドウとして表示されます。[出力] ウィンドウまたはそのタブの 1 つを InstallShield インターフェイスの端から離れた場所にドラッグすると、取り外されます。

詳細は、次を参照してください:

- 様々なビューで、[グループ ボックス] 領域を使って作業する
- [出力] ウィンドウを固定する/取り外す

アクションの進行状況メッセージの指定をサポート

エンド ユーザーに進行状況を伝えるために、インストールは一般的に、実行中の処理を説明する進行状況ダイアログで説明テキストを表示します。通常、インストールのステータスを表示する進行状況バーも同時に表示されます。標準アクションおよびカスタム アクションが発生すると、そのアクションについてのメッセージが進行状況ダイアログに表示されます。この機能は、実行するのに時間がかかるアクションで、特に役立ちます。インストールのログ記録が作成される場合、同じアクション テキストが記録されます。

[カスタム アクションとシーケンス] ビューに新しく追加された [アクション テキスト] 領域では、アクションの説明と詳細を指定できます。これは、基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- アクション テキストを使う
- アクションの説明とアクション データのテンプレートを指定する
- 進行状況ダイアログにアクションの説明を表示する
- 進行状況ダイアログにアクション データを表示する

64 ビット システムが 32 ビット IIS アプリケーションの実行を許可するかどうかを検出できる機能

実行時、IIS 6 を持つシステム上でインストールが Enable32bitAppOnWin64 プロパティのチェックが必要な場合があります。そのとき、製品の要件およびチェック結果に基づいて、たとえば 32 ビット IIS 6 アプリケーションまたは 64 ビット IIS アプリケーションを含む特定のコンポーネントのインストールをスキップして、残りのファイル転送を続行できます。

InstallShield には、ターゲット システム上で Enable32bitAppOnWin64 プロパティがどのように設定されているかを検出できるサンプル Windows Installer DLL ファイルが含まれています。プロジェクトに、この DLL のカスタム アクションを追加できます。32 ビット アプリケーションが許可されている場合、Windows Installer プロパティ ISIIS6APPPOOLSUPPORTS32BIT は値 1 に設定され、許可されていない場合、このプロパティは設定されません。条件でこのプロパティをを使って、特定の動作をトリガーまたは阻止することができます。たとえば、ISIIS6APPPOOLSUPPORTS32BIT が設定されているかいないかによって、インストールが終了する起動条件を作成できます。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

プロジェクトにカスタム アクションを追加する方法についての詳細は、「64 ビット プラットフォーム上で IIS 6 をサポートする際の考慮」を参照してください。

IIS 6 メタベースの互換性機能がインストールされているかどうかを検出できる機能

実行時に、インストールによって、ターゲット システムに IIS メタベースおよび IIS 6 構成との互換性機能がインストールされているかどうか、または IIS 以前がインストールされているかを検出できます。製品の要件と検出結果に基づいて、インストールを終了およびエラー メッセージを表示できます。

たとえば、Web サービス拡張は IIS 6 を持つシステムにインストールできます。IIS 7 を持つシステムでは、Web サービス拡張は IIS メタベースおよび IIS 6 構成との互換性機能がインストールされている場合のみインストール可能です。このため、IIS 6 が存在するか、IIS 6 との互換性機能がインストールされていることをインストールが検証するように構成することが必要な場合があります。これらのどちらの条件も False の場合、インストールが終了してエラー メッセージが表示されます。

InstallShield には、IIS メタベースおよび IIS 6 構成との互換性機能がインストールされているかどうかを検出するためのサンプル Windows Installer DLL ファイルが含まれています。プロジェクトに、この DLL のカスタムアクションを追加できます。If the IIS Metabase and IIS 6 Configuration Compatibility feature is installed, the Windows Installer property ISIISMETABASECOMPATPRESENT is set to a value of 1; if it is not installed, this property is not set.条件でこのプロパティをを使って、特定の動作をトリガーまたは阻止することができます。たとえば、ISIISMETABASECOMPATPRESENT が設定されているかいないかによって、インストールが終了する起動条件を作成できます。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

プロジェクトにカスタム アクションを追加する方法についての詳細は、「ターゲット システムに IIS 6 以前が搭載されているか、または IIS 6 メタベースの互換性機能があるかどうかを判別する」を参照してください。

名前空間プレフィックスを XML ファイル要素の属性と関連付ける機能

InstallShield では、今回より、名前空間プレフィックスを [XML ファイルの変更] ビューで属性に追加できるようになりました。そのためには、属性名の前にプレフィックスを入力して、コロン (:) を続けます。以前、属性に名前空間プレフィックスが含まれている場合、インストールが失敗しました。

この強化機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプに適用します。

詳細については、「名前空間プレフィックスを属性に追加する」を参照してください。

ランタイム変更後に XML ファイルをフォーマッするかどうかを指定できる機能

[XML ファイルの変更] ビューで XML ファイルの [詳細] タブに追加された **[変更後に XML をフォーマットする]** チェック ボックスを使って、実行時にファイルが変更された後に XML ファイルをフォーマットするかどうかを指定できます。ファイルをフォーマットするとき、インストールはファイルにインデントを追加して、空白要素タグを開始タグと終了タグに置換します。これは、web.config ファイルでは問題を起こす可能性があるため、プロジェクトでこのチェック ボックスをクリアしなくてはならない場合もあります。

この強化機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプに適用します。

詳細については、「XML ファイルの [詳細] タブ」を参照してください。

.NET Framework および Internet Explorer 8 のための定義済みシステム検索

InstallShield に 2 つの新しい定義済みシステム検索が追加されました:

- Microsoft .NET Framework 3.5 SP1
- Internet Explorer 8

インストールでこれらのいずれかが必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、これらのシステム検索をプロジェクトに追加することができます。エンドユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

この強化は、基本の MSI および InstallScript MSI プロジェクトに適用します。

カスタム プロパティを条件ビルダーに一覧表示

[条件ビルダー] ダイアログ ボックスのプロパティ リストには、今回より、[プロパティ マネージャー] ビューで追加されたプロパティが含まれます。また、システム検索ウィザードで作成された検索からのプロパティも含まれています。このため、InstallShield で条件を作成または編集するとき、独自のカスタム プロパティを手入力する代りに、リストから選択することができます。

この強化機能は、基本の MSI、InstallScript MSI、マージ モジュール および トランスフォーム プロジェクト タイプに適用します。

InstallShield 前提条件の設定で最大サービス パック番号と 64 ビットの場所を指定できる機能

InstallShield 前提条件エディターを使って InstallShield 前提条件の設定を追加または変更するときに表示される [前提条件設定] ダイアログ ボックスに、最大サービス パック番号を指定できるフィールドが新しく追加されました。以前は、最小サービス パック番号を指定できるだけで、最大サービス パック 番号を指定することはできませんでした。

また、[前提条件設定] ダイアログ ボックスでは、ファイル関連の条件を指定するときに表示されるボックスに 64 ビットの場所 ([CommonFiles64Folder]、[ProgramFiles64Folder]、および [System64Folder]) が含まれています。ファイル パスに 64 ビットの場所を使用する場合、これらのプロパティから選択できます。実行時に、ターゲット システムが 64 ビットの場合、インストールはこれらの 64 ビットの場所をチェックします。32 ビット システムの場合、インストールは対応する 32 ビットの場所をチェックします。

詳細については、「[前提条件設定] ダイアログ ボックス」を参照してください。

MSBuild サポートの強化点

MSBuild の InstallShield タスクには、次の新しいパラメーターが追加されました:

- **RunMsiValidator** — 検証に使用する .cub ファイルを指定します。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldMsiValidators として露出されます。
- **PatchConfiguration** — MSBuild を使ってビルドするパッチ構成を指定します。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、プロパティ InstallShieldPatchConfiguration として露出されます。
- **PathVariables** — パス変数の値をオーバーライドします。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPathVariableOverrides として公開されます。
- **PreprocessorDefines** — InstallScript をコンパイルするためのプリプロセッサ定義を指定します。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPreprocessorDefines として露出されます。

詳細については、「Microsoft ビルド エンジン (MSBuild)」を参照してください。

オートメーション インターフェイスの強化

Windows 7、Windows Server 2008 R2、Windows Vista、Windows Server 2008、および Windows Server 2003 の OSFilter 値

オートメーション インターフェイスで、ISWiComponent と ISWiRelease オブジェクトの OSFilter メンバーと共に、以下の定数を使用できます:

- eosWin7 (33554432) — Windows 7 および Windows Server 2008 R2 用
- eosWinVista (16777216) — Windows Vista および Windows Server 2008 用
- eosWinServer2003 (8388608)

また、eosAll の値は 5308624 から 64028880 に変更されました。

OSFilter メンバーは、InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトで ISWiComponent オブジェクトに適用します。OSFilter メンバーは、InstallScript、および InstallScript オブジェクト プロジェクトで ISWiRelease オブジェクトに適用します。

詳細は、次を参照してください:

- ISWiComponent オブジェクト
- ISWiRelease オブジェクト

新しい IsPlatformSelected プロパティ値

ISWiComponent オブジェクトの IsPlatformSelected プロパティで使用できるプロパティ値のリストが拡張されました。このプロパティには、32-bit、64-bit Itanium、AMD64、および Windows Server 2003 R2 の値が含まれます。これは、InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクト タイプに適用します。

既存の定数の一部の値が変更されています。ご注意ください。新しい値についての詳細は、「ISWiComponent オブジェクト」を参照してください。

SQL スクリプトを接続に追加する新しい AddSQLScriptEx メソッド

ISWiSQLConnection オブジェクト に AddSQLScriptEx メソッドが追加されました。このメソッドを使って ISSQLScriptFile エントリを追加し、渡された文字列から有効な名前を生成します。このメソッドは、ISSQLScriptFile テーブルのエントリ名が一意であること、またその文字数が 47 文字未満であることを保証します。

ISWiSQLScript オブジェクトの新しい RunOnLogon および Condition プロパティ

ISWiSQLScript オブジェクト に読み取り RunOnLogon プロパティが追加されました。このプロパティは、[SQL スクリプト] ビューにある SQL スクリプトの [ランタイム] タブにある [ログイン中にスクリプトを実行] チェックボックスに対応します。

ISWiSQLScript オブジェクトにはまた、読み取り/書き込み Condition プロパティが追加されました。このプロパティは、SQL スクリプトをインストールまたはアンインストール中に実行するかどうかを判断するために実行時に評価される条件を指定します。この条件が True と評価された場合に、スクリプトが実行します。このプロパティは、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

ISWiUpgradeTableEntry オブジェクトの新しい DisplayName プロパティ

ISWiUpgradeTableEntry オブジェクト に読み取り DisplayName プロパティが追加されました。このプロパティは、アップグレード エントリの名前を設定または取得します。これは [アップグレード] ビューのアップグレード アイテムに表示される内部名です。このプロパティは、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

オペレーティング システムに関する InstallScript 言語の強化

次の構造メンバーと定義済み定数が InstallScript 言語に追加されました:

- **SYSINFO.WINNT.bWin7_Server2008R2** — 新しい SYSINFO 構造メンバーです。オペレーティング システムが Windows 7 または Windows Server 2008 R2 の場合、この値は TRUE です。
- **SYSINFO.bWinServer2003R2** — 新しい SYSINFO 構造メンバーです。オペレーティング システムが Windows Server 2003 R2 の場合、この値は TRUE です。(このオペレーティング システムでは、SYSINFO.WINNT.bWinServer2003 の値も TRUE です。)

- ISOSL_WIN7_SERVER2008R2 – FeatureFilterOS 関数と SYSINFO 構造変数と共に使用できる新しい定義済み定数です。これは、ターゲット システムが Windows 7 または Windows Server 2008 R2 を実行中であることを示します。
- ISOS_ST_SERVER2003_R2 – FeatureFilterOS 関数と SYSINFO 構造変数と共に使用できる新しいオペレーティング システム スイートです。これは、ターゲット システムが Windows Server 2003 R2 を実行中であることを示します。

また、一部のアイテムの名前が変更されました:

- SYSINFO.WINNT.bWinVista 構造メンバーは Windows Vista または Windows Server 2008 を区別しません。したがって、新しいメンバー SYSINFO.WINNT.bWinVista_Server2008 が追加されました。古いエイリアスも引き続き使用できますが、コードを明確にするため、新しいメンバーが推奨されます。
- ISOSL_WINVISTA 定義済み定数は Windows Vista または Windows Server 2008 を区別しません。したがって、新しい定数 ISOSL_WINVISTA_SERVER2008 が追加されました。古いエイリアスも引き続き使用できますが、コードを明確にするため、新しいメンバーが推奨されます。

詳細については、「FeatureFilterOS 関数と SYSINFO 構造変数」を参照してください。

[InstallScript] ビューで InstallScript ダイアログ ソース コードを容易にオーバーライドできる機能

[InstallScript] ビューのイベント カテゴリ ドロップダウン リストに、新しい Dialog Source オプションが追加されました。このオプションを選択すると、イベントハンドラー ドロップダウン リストには、すべての InstallScript ダイアログがリストされます。このリストから任意のダイアログを選択して、そのコードを変更できます。

この機能は、InstallScript 、InstallScript MSI、および InstallScript オブジェクト プロジェクト タイプに適用します。

Dialog Source オプションを選択すると使用可能となる ビルトイン InstallScript ダイアログ 関数についての詳細は、「ダイアログ関数」を参照してください:

InstallScript インストールのアンインストール キーにおけるメジャーおよびマイナー バージョンのレジストリ エントリの変更

InstallScript インストールは、アンインストール キーに VersionMajor および VersionMinor レジストリ値を作成します。これらの値の名前は、今回より、基本の MSI および InstallScript MSI インストール中に作成されるエントリの名前に一致します。これは、InstallScript 2010 で作成された新しいインストール、および InstallShield 2009 以前からアップグレードされたインストールに適用します。以前、InstallShield 2009 以前では、InstallScript インストールが作成する値の名前は MajorVersion および MinorVersion でしたが、今後は作成しません。

新しいレジストリ値を使用するため、次の InstallScript 定数の値が変更されました:

- REGDB_VALUENAME_UNINSTALL_MAJORVERSION は、今回より、MajorVersion ではなく VersionMajor です。
- REGDB_VALUENAME_UNINSTALL_MINORVERSION は、今回より、MinorVersion ではなく VersionMinor です。

MaintenanceStart 関数が呼び出されると、レジストリに更新された値名が作成されます。デフォルトで、古い値名が存在するとき、それが削除されます。ターゲット システムから古い値名を削除したくない場合、REGDB_OPTION_NO_DELETE_OLD_MAJMIN_VERSION という名前の新しい REGDB_OPTIONS オプションを使用します。

RegDBGetItem 関数と共に REGDB_UNINSTALL_MAJOR_VERSION または REGDB_UNINSTALL_MINOR_VERSION を使用すると、**RegDBGetItem** は、まず新しい値をチェックします。新しい値が検出されると、関数は新しい値から値データを返します。新しい値が検出されなかった場合、関数は自動的に古い値をチェックします。古い値が検出されると、関数は古い値から値データを返します。

下位互換性を目的として、次の新しい定数が提供されています:

- REGDB_UNINSTALL_MAJOR_VERSION_OLD
- REGDB_UNINSTALL_MINOR_VERSION_OLD

これらの定数を **RegDBGetItem**、**RegDBSetItem**、および **RegDBDeleteItem** 関数と共に指定して、古い値を取得、設定、および削除できます。

次の新しい文字列定数も使用できます:

- REGDB_VALUENAME_UNINSTALL_MAJORVERSION_OLD は、MajorVersion として定義されています。
- REGDB_VALUENAME_UNINSTALL_MINORVERSION_OLD は、MinorVersion として定義されています。

詳細は、次を参照してください:

- REGDB_VALUENAME_UNINSTALL_MAJORVERSION
- REGDB_VALUENAME_UNINSTALL_MAJORVERSION_OLD
- REGDB_VALUENAME_UNINSTALL_MINORVERSION
- REGDB_VALUENAME_UNINSTALL_MINORVERSION_OLD
- REGDB_UNINSTALL_MAJOR_VERSION_OLD
- REGDB_UNINSTALL_MINOR_VERSION_OLD
- REGDB_OPTIONS
- RegDBSetItem
- RegDBGetItem
- RegDBDeleteItem

カスタム InstallScript ダイアログ内のコントロールのウィンドウ ハンドルを取得するための、新しい CtrlGetDlgItem 関数

CtrlGetDlgItem という名前の新しい InstallScript 関数が追加されました。**CtrlGetDlgItem** は、カスタム ダイアログ内のコントロールのウィンドウ ハンドルを取得します。**CtrlGetDlgItem** は Windows API **GetDlgItem** と似ていますが、**CtrlGetDlgItem** の場合、ダイアログのウィンドウ ハンドルの代わりに InstallScript ダイアログ名を指定できます。詳細については、「CtrlGetDlgItem」を参照してください。

InstallScript 文字列のポインターを外部 DLL 関数に渡す、新しい InstallScript 定数

IS_NULLSTR_PTR 変数を使って、ヌル ポインターを InstallScript 文字列としてプロトタイプされているパラメーターを通して 外部 DLL 関数または Windows API に渡すことができます。この機能は byval 文字列、byref 文字列、wstring、およびバイナリ データ タイプで使用できます。詳細については、「IS_NULLSTR_PTR」を参照してください。

InstallScript サイズ単位定数を表示文字列に変換する、新しい StrConvertSizeUnit 関数

StrConvertSizeUnit という名前の新しい InstallScript 関数が追加されました。**StrConvertSizeUnit** 関数は、指定された InstallScript サイズ単位定数の適切な表示文字列を返します。詳細は、「StrConvertSizeUnit」を参照してください。

文字列から先頭と行末の空白およびタブを削除する、新しい StrTrim 関数

StrTrim という名前の新しい InstallScript 関数が追加されました。**StrTrim** 関数は、文字列から先頭と行末の空白およびタブを削除します。詳細については、「StrTrim」を参照してください。

既存の SdLicense* ダイアログ関数に優先する新しい SdLicense* ダイアログ関数

新しい 2 つの InstallScript ダイアログ 関数 (**SdLicenseEx** と **SdLicense2Ex**) が追加されました。どちらの関数も、複数行編集フィールドに使用許諾契約書が記載されたダイアログを表示します。使用許諾契約書は、テキスト ファイル (.txt) またはリッチ テキスト ファイル (.rtf) で保存できます。

- SdLicenseEx は、質問をスタティック フィールドに表示するダイアログを表示します。エンド ユーザーは、[はい] または [いいえ] ボタンをクリックして質問に答えます。

  SdLicenseEx は、**SdLicense** と **SdLicenseRtf** よりも優先します。

- SdLicense2Ex は、2 つのラジオ ボタンを持つダイアログを表示します (使用許諾契約書の条件に同意するためのボタンと、同意しないためのボタン)。エンド ユーザーが使用許諾契約書の条件に同意するための適切なボタンをクリックすると、[次へ] ボタンが有効になります。

  SdLicense2Ex は、**SdLicense2** と **SdLicense2Rtf** に優先します。.

キーと値のペアー覧を検索するための、新しい ListFindKeyValueString 関数

ListFindKeyValueString という名前の新しい InstallScript 関数が追加されました。**ListFindKeyValueString** 関数は、指定された値の文字列または数値リストを検索します。最初のリストで見つかった文字列の位置に対応する追加リストから値が返されます。これによって、キーと値のペアー覧で特定のキーを検索して、対応する値を取得できます。

詳細については、「ListFindKeyValueString」を参照してください。

新しい InstallScript コードの例

InstallShield ドキュメントには、次の InstallScript 機能のためのサンプル コードが含まれています:

- AdminAskPath
- CharReplace
- FormatMessage
- LogReadCustomNumber
- LogReadCustomString
- LogWriteCustomNumber
- LogWriteCustomString

このコードを InstallShield ドキュメントからコピーして、InstallScript コードに貼り付け、必要に応じてカスタマイズできます。

InstallShield キャビネット ファイル ビューアーの拡張機能

キャビネット ファイル ビューアーでは、InstallScript プロジェクト用の .cab ファイルについての追加情報が提供されています。

- InstallShield でビルドされた機能について、ビューアーは "パスワード保護"、"前を分割"、"後を分割"、"分割なし"、および "前を分割しない" フィールドとイメージ インデックスを表示します。

- InstallScript インストールに含まれる InstallScript オブジェクトについて、ビューアーは新しい "オブジェクト バージョン" フィールドを表示します。
- コンポーネントについて、ビューアーは新しい "暗号化"、"データ ファイル"、および ".NET アセンブリ" フィールドを表示します。
- メディアについて、ビューアーは新しい "実行可能ファイル" フィールドを表示します。

InstallShield 2009 SP2 の新しい機能

InstallShield 2009 Service Pack 2 (SP2) には次の変更が含まれています:

Microsoft Visual Studio 2008 SP1 サポート

InstallShield は、今回より以下に記述されているとおり、Microsoft .NET Framework 3.5 SP1 と Microsoft SQL Server 2008 をサポートします。

また、InstallShield と Visual Studio との統合では Visual Studio 2008 SP1 がサポートされており、Visual Studio 内でインストールおよび製品の開発を行うことができます。

新しい Microsoft .NET Framework 3.5 SP1 前提条件

InstallShield には、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加することができる .NET 関連の新しい 2 つの InstallShield 前提条件が含まれています。

- Microsoft .NET Framework 3.5 SP1 (Web ダウンロード版)
- Microsoft .NET Framework 3.5 SP1 (完全パッケージ)

詳細については、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。

Microsoft SQL Server 2008 サポート

InstallShield は、今回より、SQL Server 2008 上で SQL スクリプトを実行するためのサポートを含みます。また、InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストには、SQL Server 2008 が含まれています。

新しい Microsoft SQL Server 2008 Express 前提条件

InstallShield に Microsoft SQL Server 2008 Express Edition の InstallShield 前提条件が追加されました。これらの InstallShield 前提条件は、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加できます。

これらの新しい前提条件をサポートするため、InstallShield 前提条件エディターに新しいレジストリ条件タイプが追加されました。

InstallShield 前提条件エディターに追加された新しいレジストリ条件タイプ

InstallShield 前提条件エディターに、新しく [**レジストリ エントリが指定のバージョン値を含む**] 条件オプション タイプが追加されました。この条件タイプを使って、ターゲット システム上で特定のレジストリ エントリの値データとして格納されているバージョン番号が、InstallShield 前提条件エディターで指定されたバージョン番号よりも大きいか、小さいか、または等しいかをインストールがチェックするように設定できます。この新しい条件の詳細については、「[前提条件設定] ダイアログ ボックス」を参照してください。

新しい Microsoft SQL Server 2005 Express SP2 (x86 と x64 WOW) 前提条件

InstallShield に Microsoft SQL Server 2005 Express Edition SP2 の InstallShield 前提条件が追加されました。この前提条件は Microsoft SQL Server 2005 Express Edition SP2 を 32 ビットおよび 64 ビット (WOW) システムにインストールします。この InstallShield 前提条件は、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加できます。

新しい FlexNet Connect 11.0.1 再配布可能ファイル

InstallShield はまた、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで Connect 11.0.1 のサポートを含みます。InstallShield の [アップデート通知] ビューで、2 つの FlexNet Connect 11.0.1 マージ モジュール (Common Software Manager が含まれているマージ モジュールと、含まれていないマージ モジュール) のどちらかを含みます。これらのマージモジュールは、FlexNet Connect 11 マージ モジュールに取って代わります。更新されたマージ モジュールの変更点についての詳細は、FlexNet Connect のリリース ノートを参照してください。

追加の変更

InstallShield 2009 SP2 で解決されている問題の一覧は、リリース ノートをご覧ください。リリース ノートは、InstallShield の [ヘルプ] メニューからご覧になることができます。

InstallShield 2009 SP1 の新しい機能

InstallShield 2009 は、Windows Installer 4.5 ベータ版をサポートします。InstallShield 2009 Service Pack 1 (SP1) には、Windows Installer 4.5 の最終リリース版のサポートを提供する変更が含まれています。

Windows Installer 4.5 用 InstallShield 前提条件

InstallShield は、Windows Installer 4.5 再配布可能ファイルの最終リリース バージョン用の以下の InstallShield 前提条件を含みます。

- Windows Vista と Server 2008 (x86) 用 Windows Installer 4.5
- Windows Vista と Server 2008 (x64) 用 Windows Installer 4.5
- Windows Vista と Server 2008 (IA64) 用 Windows Installer 4.5
- Windows Server 2003 SP1 以降 (x86) 用 Windows Installer 4.5
- Windows Server 2003 と XP (x64) 用 Windows Installer 4.5
- Windows Server 2003 (IA64) 用 Microsoft Windows Installer 4.5
- Windows XP SP2 以降 (x86) 用 Windows Installer 4.5

実行時に Windows Installer 4.5 をインストールする場合、これらの InstallShield 前提条件を 基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加できます。

詳細については、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

追加の変更

InstallShield 2009 SP1 で解決されている問題の一覧は、リリース ノートをご覧ください。リリース ノートは、InstallShield の [ヘルプ] メニューからご覧になることができます。

InstallShield 2009 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

InstallShield 前提条件を機能と関連付けて、インストールを連鎖させる機能

InstallShield では、InstallShield 前提条件を 1 つまたは複数の機能と関連付けられるようになりました。この新しい種類の InstallShield 前提条件は、*機能前提条件*と呼ばれます。機能前提条件は、前提条件を含む機能がインストールされたときに、その前提条件がシステム上に既にインストールされていない場合にインストールされます。

プロジェクトに InstallShield 前提条件を含めると、複数のインストールを連鎖することができるため、1 度に 1 つの実行シーケンスのみしか実行できない Windows Installer 制限をバイパスできます。Setup.exe セットアップランチャーは、連鎖を管理するブートストラップ アプリケーションとしての役割を果たします。

[再配布可能ファイル] ビューを使って、InstallShield 前提条件をプロジェクトに追加して、その前提条件をメイン インストールの前に実行するのか、メイン インストールに含まれる 1 つまたは複数の機能と関連付けるのかを指定できます。

以前は、すべての InstallShield 前提条件インストールが、メイン インストールの実行前に実行されたため、InstallShield 前提条件を機能に関連付けることはできませんでした。この種類の前提条件は今後も使用できますが、今回より、*セットアップ前提条件*と呼ばれます。

基本の MSI プロジェクトが、この機能のサポートを含みます。

詳細は、次を参照してください:

- セットアップ前提条件と機能前提条件の違い
- 基本の MSI プロジェクトで、InstallShield 前提条件を機能に関連付ける
- InstallShield 前提条件を含むインストールの実行時の動作
- インストール プロジェクトに含まれている InstallShield 前提条件を利用する

Windows Installer 4.5 Beta におけるトランザクション処理を使った複数パッケージのインストールを可能にする機能のサポート

InstallShield を使って、基本の MSI と InstallScript MSI プロジェクトに Windows Installer パッケージを連鎖 .msi パッケージとして追加できます。基本の MSI または InstallScript MSI インストールに連鎖 .msi パッケージが含まれていて、ターゲットシステムに Windows Installer 4.5 が存在するとき、Windows Installer はトランザクション処理を使って複数のパッケージをインストールします。パッケージを連鎖させることによって、単一のトランザクションとして処理します。トランザクションに含まれる 1 つまたは複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は全てのパッケージについてロールバックを開始して、システムを以前の状態に復元します。

[リリース] ビューの [連鎖 .msi パッケージ] 領域で、メイン インストールに連鎖させる 1 つまたは複数の .msi パッケージをプロジェクトに追加できます。この領域ではまた、連鎖 .msi パッケージにリリース フラグを割り当てたり、連鎖されたパッケージに渡す必要があるプロパティなどの設定を構成したり、条件を指定したりできます。

詳細については、次を参照してください。

- トランザクション処理を使って複数パッケージ インストールを構成する
- トランザクション処理を使った複数パッケージ インストールについての概要

- 新しい連鎖 .msi パッケージをプロジェクトに追加する
- 連鎖 .msi パッケージの設定

Windows Installer 4.5 Beta における InstallScript MSI インストールで InstallScript エンジンを埋め込みユーザー インターフェイスとして使用できる機能のサポート

InstallScript MSI プロジェクトでは、InstallScript エンジンを埋め込みカスタム ユーザー インターフェイス (UI) ハンドラーとして使用するオプションが追加されました。従来、InstallScript エンジンは外部カスタム UI ハンドラーとして使用されていました。Windows Installer 4.5 は、この新しい機能をサポートします。この新しい埋め込みオプションを使用すると、エンド ユーザーがインストールを .msi パッケージから直接起動することができると共に、InstallScript で定義された高度にカスタマイズされたユーザー インターフェイスを含んだインストールを作成できます。

この新しい埋め込み InstallScript UI 機能と、従来の外部 InstallScript UI 機能のどちらを使用するかを指定するには、新しい "InstallScript ユーザー インターフェイスの種類" 設定を使います。この設定は [一般情報] ビューにあり、プロジェクト全体に反映します。デフォルトで、InstallShield はすべての InstallScript MSI プロジェクトで従来のスタイルを使用します。この従来型のオプションには、Setup.exe セットアップランチャーが必要です。

新しく INSTALLSCRIPTMSIEEUI 変数が追加されました。新しいスタイルが使用されるとき、この変数は実行時に True に設定されます。それ以外の場合は False に設定されます。

新しいスタイルには、従来のスタイルには適用されなかった制限事項がいくつかありますので、ご注意ください。たとえば、一部の InstallScript 関数およびコマンドライン パラメーターは、新しいスタイルでは、サポートされていないか、または動作が異なるものがあります。2 つのスタイルに関する詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

追加の情報については、次を参照してください:

- InstallScript MSI インストールの InstallScript ユーザー インターフェイス タイプを指定する
- InstallScript MSI インストール プロジェクト
- INSTALLSCRIPTMSIEEUI

共有コンポーネントのパッチにおける Windows Installer 4.5 ベータ サポート

InstallShield では、Windows Installer 4.5 で提供されている新しい共有コンポーネントのパッチ機能がサポートされています。[コンポーネント] ビューの新しい "複数パッケージの共有コンポーネント" 設定を使って、選択したコンポーネントについて共有コンポーネントのパッチを有効にするかどうかを指定できます。[はい] を選択すると、新しいコンポーネントの属性が設定されます。この属性は、複数のパッケージ間で共有されているコンポーネントを含むパッチがアンインストールされるときに、Windows Installer がファイルをダウングレードすることを阻止します。これによって、最新バージョンを含むパッチがアンインストールされた場合でも、ターゲット システム上に常にコンポーネントの最新バージョンが保持されます。このオプションのデフォルト値は [いいえ] です。

Windows Installer 4.5 はこの新しい機能をサポートします。Windows Installer の以前のバージョンは、この新しい設定を無視します。また、ターゲット システムで DisableSharedComponent ポリシーが 1 に設定されている場合、Windows Installer はすべてのパッケージについて、この設定を無視します。

この機能は、基本の MSI、InstallScript MSI、およびマージ モジュール、およびトランスフォーム プロジェクトに適用します。

詳細については、「コンポーネントの "共有コンポーネントのパッチ" を有効にするかどうかを指定する」を参照してください。

Windows Installer 4.5 Beta における優先コンポーネント機能のサポート

[コンポーネント] ビューに追加された新しい "置き換えられたコンポーネントのアンインストール" 設定を使って、特定の条件下で優先するパッチのインストール中に、選択されたコンポーネントを Windows Installer 4.5 が処理する方法を指定できます。現在のパッチに含まれるこのコンポーネントをアンインストール用にフラグすることで、優先するパッチが適用された後にターゲット システム上でこのコンポーネントが孤立しないようにするためには、[はい] を選択します。後続のパッチがインストールされ、またそれが最初のパッチよりも優先されることがフラグされている場合、適切な場合に Windows Installer はこのコンポーネントを登録解除およびアンインストールします。[いいえ] を選択した場合、優先するパッチはターゲット マシン上に孤立したコンポーネントを残す可能性があり、後に残された機能のコンポーネントの管理は一切行われません。この設定のデフォルト値は [いいえ] です。

Windows Installer 4.5 はこの新しい機能をサポートします。Windows Installer の以前のバージョンは、この新しい設定を無視します。

この機能は、基本の MSI、InstallScript MSI、およびマージ モジュール、およびトランスフォーム プロジェクトに適用します。

さらに詳しい情報は、「"置き換えられたコンポーネントのアンインストール" 設定」の説明を参照してください。

パッチ シーケンスに関する詳しい情報は、「パッチ シーケンス」をご覧ください。

Windows Installer 4.5 Beta におけるパッチのアンインストール時のみカスタム アクションを実行できる機能のサポート

カスタム アクションのための新しい "パッチのアンインストールのみ実行" 設定を利用して、Windows Installer が選択されたカスタム アクションをパッチのアンインストール時のみ実行するかどうかを指定できます。この設定は、基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの [カスタム アクションとシーケンス] ビューにあります。マージ モジュール プロジェクトおよび MSM データベース プロジェクトでも提供されています。

また、カスタム アクション ウィザードの [追加のオプション] パネルにある [パッチのアンインストール時に実行する] チェック ボックスを利用しても、カスタム アクションの動作を構成することができます。

この新しい機能については、「パッチのアンインストール時に実行」をご覧ください。

Windows Installer 4.5 再配布可能ファイルのベータ サポート

InstallShield には、Windows Installer 4.5 用のいくつかの InstallShield 前提条件ファイル (.prq) が含まれています。

この機能は、基本の MSI および InstallScript MSI プロジェクトに適用します。

詳細については、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

Setup.exe および Update.exe ブートストラッパの Unicode バージョンを作成する機能

InstallShield では、基本の MSI プロジェクトで Setup.exe セットアップ ランチャーを Unicode バージョンまたは ANSI バージョンのどちらで作成するかを指定できます。以前、基本の MSI プロジェクトにセットアップランチャーが含まれているとき、常に ANSI バージョンがビルドされていました。Unicode バージョンのビルドはサポートされていませんでした。

Unicode セットアップランチャーは、ターゲット システムで 2 バイト言語のための適切なコード ページが実行されているいないにかかわらず、セットアップランチャーのユーザー インターフェイスで 2 バイト文字を正しく表示することができます。ANSI セットアップランチャーは、ターゲット システムで適切なコード ページが実行されている場合のみ、セットアップランチャー ダイアログで 2 バイト文字を正しく表示します。適切なコード ページが実行されていない場合、これらのダイアログで 2 バイト文字が文字化けして表示されます。

[リリース] ビューでリリースについて表示される Setup.exe タブに新しく追加された "最短初期化時間" 設定を利用して、Unicode を使うか、または ANSI を使うかを指定することができます。すべての基本の MSI プロジェクトでは、Unicode がデフォルトとして使用されます。

パッチまたは QuickPatch パッケージについて Update.exe アップデート ランチャを作成するときも、Unicode バージョンで作成するか、または ANSI バージョンで作成するかを指定することができます。

[パッチのデザイン] ビューでパッチ構成について表示される Setup.exe タブに新しく追加された "アップデート ランチャの種類" 設定を利用して、Unicode を使うか、または ANSI を使うかを指定することができます。QuickPatch プロジェクトでは、"アップデート ランチャの種類" 設定は [一般情報] ビューの [ビルド設定] 領域の [詳細] タブにあります。すべての新しいパッチ構成と QuickPatch パッケージでは、Unicode がデフォルトとして使用されます。

Setup.exe ブートストラッパのログファイルを作成する機能

基本の MSI プロジェクトと InstallScript MSI プロジェクトで、新しい /debuglog コマンドライン パラメーターが Setup.exe セットアップランチャーに追加されました。このコマンドライン パラメーターを使用して、デバッグ用のログ ファイルを生成することができます。詳細については、「/debuglog」を参照してください。

マネージコード カスタム アクションのサポート

InstallShield では、マネージ コード カスタム アクションを 基本の MSI プロジェクト、InstallScript MSI プロジェクト、マージ モジュール プロジェクトに追加することができます。この種類のカスタム アクションは、Visual Basic .NET または C# などのマネージ コードで書かれた .NET アセンブリ内にあるパブリック メソッドを呼び出します。

詳細については、次を参照してください。

- マネージ コード ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す
- マネージコード カスタム アクションの実行時要件
- アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する
- カスタム アクションの設定を構成する

製品の複数インスタンスをインストールするためのサポート

InstallShield では今回より、.msi パッケージを使って同じマシン上に同じコンテキストで製品の複数インスタンスをインストールすることができるインストールの作成をサポートします。[リリース] ビューの製品構成に追加された新しい [複数インスタンス] タブを使って製品の異なるインスタンスを定義し、各インスタンスの関連プロパティを構成することができます。

ビルド時、InstallShield は各インスタンスに対して製品コードを変更するためのインスタンス トランスフォームを作成し、それを .msi パッケージにストリームします。実行時、セットアップランチャーは [新しいインスタンスの選択] ダイアログを表示します。このダイアログでエンド ユーザーは新しいインスタンスをインストールするか、インストール済みインスタンスのアップデートまたはメンテナンスを行うかを指定することができます。

さらに今回より、[パッチのデザイン] ビューでインストールに複数インスタンス サポートを含む製品用のパッチをビルドすると、エンド ユーザーが特定のインスタンスまたはすべてのインスタンスを更新することができるパッチが作成されます。実行時、Update.exe ファイルはパッチ バージョンの [インスタンスの選択] ダイアログを表示します。

Setup.exe と Update.exe に新しい /instance コマンドライン オプションが追加されました。このオプションを使って、インストール、更新、またはアンインストールするインスタンスを指定することができます。また、[インスタンスの選択] ダイアログを抑制することも可能です。

複数インスタンスは、基本の MSI プロジェクトでサポートされています。

詳細については、次を参照してください。

- 製品の複数のインスタンスをインストールする
- 複数インスタンス サポートの実行時要件
- InstallShield で複数インスタンスを構成する
- 複数インスタンス サポートに関する特別考慮
- 複数インスタンスのサポートを含むリリースの構成とビルド
- 製品の複数インスタンスをインストールするためのサポート
- 製品のインスタンスを複数インストールするときの実行時の動作
- セットアップランチャーの作成
- 製品構成の [複数インスタンス] タブ
- Setup.exe および Update.exe コマンドライン パラメーター

新しい Microsoft .NET 再配布可能ファイル

InstallShield には今回、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加することができる .NET 関連の新しい InstallShield 前提条件が多数含まれています。

- Microsoft .NET Framework 3.5 (Web ダウンロード版)
- Microsoft .NET Framework 3.5 (完全パッケージ)
- Microsoft .NET Framework 3.5 言語 パック (x86、x64、IA64)
- Microsoft .NET Framework 3.0 SP1 (Web ダウンロード版)
- Microsoft .NET Framework 3.0 言語パック
- Microsoft .NET Framework 2.0 SP1 (x86、x64、IA64)

また InstallShield では、InstallScript プロジェクト用のアップデートされた Microsoft .NET Framework オブジェクトも利用できます。このオブジェクトには、32 ビット、64 ビット x64、および 64 ビット Itanium バージョンを含む、バージョン 3.5、3.0 SP1、3.0、2.0 SP1、2.0、1.1 SP1、および 1.0 SP3 の .NET Framework のサポートが含まれています。オブジェクトには、Web ダウンローダ再配布可能ファイルだけでなく、利用可能なサポート対象の言語パックも含まれています。

詳細については、次を参照してください。

- .NET Framework 再配布可能ファイルをプロジェクトへ追加する
- Microsoft .NET Framework および Microsoft .NET Framework 言語パックの前提条件を含める
- Microsoft .NET Framework オブジェクト ウィザード

新しい .msi パッケージ ツール

InstallShield には、いくつかの新しいツールが含まれています:

- InstallShield MSI Diff を使って、2 つの .msi、.msm、または .pcp ファイルを素早く比較できます。これを使って、1 つの .msi ファイルに対して 1 つまたは複数の .msp と .mst ファイルを適用し、その結果となる .msi

データベースの変更を確認することができます。また、このツールを使って、バイナリ形式で保存された 2 つの InstallShield プロジェクト ファイル (.ism または .ise) を比較することができます。このツールでは、追加、変更、削除、スキーマの違いを示すためにカラー コードが使用されます。このツールは、ほとんどのソース コード管理システムに簡単に統合できます。

- InstallShield MSI Query を使って、SQL ステートメントをビルド スクリプトで実行する前に、SQL の Windows Installer バージョンを用いてステートメントをテストすることができます。SQL ステートメントが正しくフォーマットされているかを素早く確認でき、また生成される結果を参照できます。
- InstallShield MSI Sleuth は、ターゲット システムの現在のインストール済み状態を参照できる診断ツールです。InstallShield MSI Sleuth はインストール済みのすべての .msi パッケージの一覧を表示します。リストの中から任意の .msi パッケージをクリックすると、データベース内のテーブルやバイナリ ストリームだけでなく、その機能とコンポーネントの状態および既知のソースの場所を参照できます。このツールは、特定のコンポーネント コードを持つパッケージを含むインストール済みの製品 (複数可) を識別するのにも役立ちます。
- InstallShield MSI Grep は、.msi と .msm パッケージのコレクションの中から特定のテキストを検索します。特定のテーブルまたは列に関する結果のみを表示するための詳細検索も可能です。

これらすべてのツールは、Windows [スタート] メニューの [InstallShield ツール] サブフォルダーから起動できます。

これらの InstallShield MSI ツールは、InstallShield Premier Edition および Professional Edition に含まれています。Premier Edition には、別のマシンに InstallShield 以外のツールのみをインストールできる、個別のインストールおよび追加ライセンスが含まれています。特定の条件については、InstallShield エンド ユーザー使用許諾契約書を参照してください。

詳細については、「InstallShield MSI ツール」を参照してください。

Setup.exe と ISSetup.dll にストリームされるファイルを圧縮し、圧縮レベルを指定できる機能

Setup.exe セットアップランチャーまたは ISSetup.dll ファイルを使用するリリースをビルドしたとき、InstallShield によって、Setup.exe ファイルまたはISSetup.dll ファイルにストリームされるファイルが圧縮されるようになりました。InstallShield が使用するデフォルトの圧縮レベルは、ファイルのサイズと実行時に圧縮ファイルを展開するために必要な時間のバランスをとっての目安です。圧縮レベルを変更する場合、または圧縮をしない場合、マシン全体に適用する設定を利用してデフォルトのレベルをオーバーライドすることができます。

デフォルトで、ビルド時に InstallShield がファイルを Setup.exe ファイルにストリームするとき、.cab ファイル拡張子を持つファイルは圧縮されません。これは、.cab ファイルが既に圧縮されているファイルであるためです。デフォルトの圧縮除外一覧を変更して、他の種類のファイルや特定のファイルを必要に応じて選択することができます。除外一覧は、マシン全体に適用される設定です。

この機能は、基本の MSI および InstallScript MSI プロジェクトに適用します。

詳細については、「Setup.exe と ISSetup.dll にストリームされるファイルの圧縮レベルを構成する」を参照してください。

マルチパート .cab ファイルのサポート

.cab ファイルには、いくつかの制限事項があります。たとえば、単一 .cab ファイルの最大サイズは 2 GB に設定されています。また、サイズの大きい .cab ファイルを署名しようしたとき、およびサイズの大きい署名済み .cab ファイルのデジタル署名を検証しようとしたときにトラブルが生じた経験があるユーザーもいるかもしれません。今回より、これらの制限事項を回避するために、.cab ファイルのデフォルトの制限が 600 MB に設定されました。InstallShield で、リリースの .cab ファイルを作成しているとき、この制限に達すると、データが 2 つ以上の .cab ファイルに分割され、マルチパートの .cab ファイルが作成されます。

最大サイズは必要に応じて変更することができます。InstallShield でマルチパート .cab ファイルを作成しない場合、単一の .cab ファイルを作成するように構成できます。

この機能は、基本の MSI および InstallScript MSI プロジェクトに適用します。また、この機能は、すべてのファイルが単一ファイルの .msi パッケージまたは Setup.exe セットアップランチャーに埋め込まれている圧縮済みネットワーク イメージ リリースをビルドしている場合のみ適用します。この機能は、1 つまたは複数の機能に関連付けられているファイルのみが .cab ファイルに圧縮されるカスタム圧縮には適用しません。

詳細については、「.cab ファイルの最大サイズを構成する」を参照してください。

基本の MSI インストールに [ファイルを開く] ダイアログを追加して、エンド ユーザーによるファイルの参照を可能にするサポート

InstallShield は、基本の MSI インストールのダイアログの 1 つから [ファイルを開く] ダイアログを起動するサポートを含みます。エンド ユーザーがダイアログの 1 つから [参照] ボタンをクリックすると、[ファイルを開く] ダイアログが起動します。[ファイルを開く] ダイアログを使って、エンド ユーザーはファイルを参照できます。エンド ユーザーがファイルを選択して [開く] ボタンをクリックすると、[ファイルを開く] ダイアログが閉じて、インストールがダイアログの編集フィールドにその完全パスとファイル名を書き込みます。インストールはまた、IS_BROWSE_FILEBROWSED プロパティの値を、エンド ユーザーが選択したファイルのパスとファイル名に設定します。

この機能を利用するには、プロジェクトに FileBrowse という名前の Windows Installer DLL カスタム アクションを追加する必要があります。このカスタム アクションは FileBrowse.dll ファイルを呼び出します。さらに、[ファイルを開く] ダイアログを起動する編集フィールド コントロールと [参照] ボタンを追加して、関連ダイアログイベントを設定しなくてはなりません。詳しい手順については、「[ファイルを開く] ダイアログを起動する」を参照してください。

Windows Server 2008 上に IIS 7 Web サイトをインストールするためのサポート

InstallShield では、今回より Windows Server 2008 システム上で IIS 7 Web サイト、仮想ディレクトリ、Web サービス拡張、およびアプリケーション プールの作成ならびに管理ができるようになりました。この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで使用できます。

追加された Microsoft SQL Server 2005 Express SP2 前提条件

InstallShield に Microsoft SQL Server 2005 Express Edition SP2 の InstallShield 前提条件が追加されました。この InstallShield 前提条件は、基本の MSI および InstallScript MSI プロジェクトに追加することができます。

MySQL 5.0 サポート

InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストに 、MySQL の 5.0.x バージョンもリストされるようになりました。これまでは、カスタム バージョン要件を作成しなくてはなりませんでした。

Microsoft Visual Studio 2008 サポート

InstallShield が Visual Studio 2008 と統合され、インストールと製品の開発を同じ Visual Studio インターフェイス内で行えるようになりました。

Visual Studio セットアップとマージ モジュール プロジェクトを InstallShield プロジェクトに変換する機能

今回より、InstallShield で、Visual Studio 2008、Visual Studio 2005、Visual Studio .NET 2003、または Visual Studio .NET セットアップ プロジェクト (.vdproj) を基本の MSI プロジェクト (.ism) に変換することができるようになりました。また、Visual Studio 2008、Visual Studio 2005、Visual Studio .NET 2003、または Visual Studio .NET マージ モジュール プロジェクト (.vdproj) を InstallShield マージ モジュール プロジェクト (.ism) に変換することもできるようになりました。

Visual Studio プロジェクトを InstallShield プロジェクトに変換すると、ダイアログ エディターを使ってダイアログのレイアウトを視覚的に変更したり、機能やコンポーネントを管理したり、InstallShield で提供されている他の機能を利用したりすることができます。

詳細については、「Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートする」を参照してください。

アラビア語 (サウジアラビア) とヘブライ語サポート、およびダイアログ エディターにおける右から左方向に書く言語サポート

InstallShield には、右から左に記述する言語であるアラビア語 (サウジアラビア) とヘブライ語のサポートが新しく含まれています。デフォルトのエンド ユーザー ダイアログ文字列のすべてが、これらの言語で利用できます。

これらの言語は右から左方向に読まれるため、InstallShield にはアラビア語とヘブライ語ダイアログのミラー サポートが含まれます。これによって、アラブ語とヘブライ語のダイアログには、右から左方向へのレイアウトが使用されます。たとえば、英語やその他の左から右に読まれる言語のダイアログで右側にあるボタンは、右から左に読まれる言語のダイアログでは左側に移動されます。また InstallShield は、ビルトイン ダイアログ テーマで表示されるダイアログ イメージのミラー画像バージョンを使用します。

右から左方向へのレイアウトと反転イメージは、InstallShield の [ダイアログ] ビュー内の [ダイアログ エディター] ペイン、および実行時に使用されます。

アラビア語とヘブライ語のサポートは、InstallShield Premier Edition で利用できます。Premier Edition には、35 の言語サポートが含まれています。

基本の MSI とマージ モジュール プロジェクト タイプに、この機能のサポートが含まれています。

詳細については、「右から左に記述される言語のダイアログ サポート」を参照してください。

InstallScript ファイル (.rul) の文字列エントリの検証

InstallScript ファイル (.rul) を含むプロジェクトをビルドするとき、InstallScript コードに @ 演算子を使用する文字列エントリへの参照が 1 つ以上含まれている場合、InstallShield はビルド時に文字列エントリを検証します。

プロジェクトに含まれる InstallScript ファイルの文字列 ID がプロジェクトの文字列エントリの 1 つに定義されていない場合、InstallShield はビルド警告 -7174 を表示します。

これは、InstallScript カスタム アクションを含む基本の MSI プロジェクト、InstallScript、InstallScript MSI、InstallScript オブジェクト、および InstallScript カスタム アクションを含む マージ モジュール プロジェクト タイプに適用します。

詳細は、次を参照してください:

- ビルド警告 -7174 の説明
- 文字列定数演算子 (@)

新しい FlexNet Connect 11 再配布可能ファイル

InstallShield はまた、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで FlexNet Connect 11 をサポートします。InstallShield の [アップデート通知] ビューを使って、2 つの FlexNet Connect 11 マージ モジュールうち、いずれかが含まれています (Common Software Manager が含まれているマージ モジュールと、含まれていないマージ モジュール)。

強化機能

InstallShield には、以下のような強化機能が搭載されています。

ベスト プラクティス ダイナミック ファイル リンク

プロジェクトにダイナミック ファイル リンクを追加、または変更するとき、InstallShield がコンポーネントを作成する方法について、新しいベスト プラクティスを使用するか、これまでと同様にディレクトリごとに 1 つのコンポーネント作成するのかを指定できるようになりました。

コンポーネント作成のベスト プラクティスに従うと、ダイナミック リンクを持つフォルダーにある各ポータブル実行可能ファイル (PE) ファイルにコンポーネントが別々に作成されます。ダイナミック リンクがある PE ファイルの 1 つを更新するパッチをあとで作成する場合、「ディレクトリごとに 1 つのコンポーネント」方式の代わりに、この方式を利用したほうが、パッチを簡単に作成できます。

以前、ダイナミック ファイル リンクをプロジェクトに追加すると、ビルド時に、ダイナミック リンクを持つすべてのファイルについてコンポーネントが 1 つ自動的に作成されていました。ただし、ダイナミック ファイル リンクに PE ファイルが含まれているとき、コンポーネントの作成時に Windows Installer ベスト プラクティスが実行されませんでした。

デフォルトで、InstallShield は .exe、.dll、.ocx、.vxd、.chm、.hlp、.tlb、および .ax を PE ファイルとして認識します。[オプション] ダイアログ ボックスにある新しい [ファイルの拡張子] タブで、この一覧を変更することができます。

オートメーション インターフェイスに、新しいベスト プラクティス メソッドのサポートが追加されました。ISWiDynamicFileLinking オブジェクトに、ダイナミック ファイル リンクにベスト プラクティス方式、または以前提供されていた「ディレクトリごとに 1 つのコンポーネント」方式を使用するかを指定できる新しい CreateBestPracticeComponents プロパティが追加されました。AddDynamicFileLinking メソッドを使用して新しいダイナミック ファイル リンクを作成すると、ベスト プラクティス メソッドがデフォルトで使用されます。

ベスト プラクティス ダイナミック ファイル リンクは、基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトに適用します。

詳細は、次を参照してください:

- ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する
- [ダイナミック ファイル リンクの設定] ダイアログ ボックス
- [ファイルの拡張子] タブ
- ISWiDynamicFileLinking オブジェクト
- AddDynamicFileLinking メソッド

[インストールされる前提条件] 一覧からセットアップ前提条件を隠す機能

ターゲット システムに 1 つまたは複数のセットアップ前提条件のインストールが必要な場合、実行時、メイン インストールが実行する前にセットアップ前提条件ダイアログが表示されます。セットアップ前提条件エディターの [動作] タブに追加された新しいチェック ボックスを使って、前提条件ダイアログで表示される前提条件の一覧から InstallShield 前提条件を隠すことを指定できます。前提条件を隠した場合、インストールが必要な前提条件の 1 つとしてリストされていなくても、条件がそれを必要したときに前提条件がインストールされます。

この機能が利用可能になる以前に作成された新しい前提条件や既存の前提条件は、デフォルトで表示されます。この動作は、[動作] タブに追加された新しいチェック ボックスを選択して変更できます。

詳細については、「ターゲットシステムで [インストールされるセットアップ前提条件] リストに、前提条件の名前を含めるかどうかを指定する」を参照してください。

実行時に InstallShield 前提条件のインストール進行状況を表示できる機能

IInstallShield 前提条件エディターの [動作] タブに追加された新しいチェック ボックスを使って、実行時に前提条件のインストールで Windows Installer からのインストール進行状況メッセージを表示するかどうかを指定できます。これによって、進行状況バーに進行中の .msi インストールのステータスが反映されます。この機能は、前提条件が .msi ファイルを起動する場合のみ使用が可能で、Setup.exe ファイルを起動する場合は使用できません。詳細については、「実行時に、InstallShield 前提条件のインストールの進行状況を表示するかどうかを指定する」を参照してください。

この機能が利用可能になる以前に作成された新しい前提条件や、既存の前提条件では、デフォルトで進行状況は表示されません。この動作は、[動作] タブに追加された新しいチェック ボックスを選択して変更できます。

進行状況の表示を指定した場合、利用可能なコマンドライン パラメーターのうち一部のみがサポートされている点に、ご注意ください。詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。

InstallShield 前提条件に、コマンドライン ステートメントで Windows Installer プロパティを使用できる機能

基本の MSI インストールおよび InstallScript MSI インストールのセットアップ前提条件コマンドラインで、ProductLanguage、ISPREREQDIR、SETUPEXEDIR、および SETUPEXENAME プロパティを置換、および渡すことができます。これらのプロパティを使って起動するインストールを識別 (**"[SETUPEXEDIR]¥[SETUPEXENAME]"**)、前提条件に含まれるファイルへの完全パスを識別 (**"[ISPREREQDIR]myconfig.ini"**)、または InstallShield セットアップランチャーなどの複数言語前提条件に選択された言語を渡す (**/l[ProductLanguage]**) ことができます。新しい機能前提条件は、前述のプロパティを含む任意の Windows Installer プロパティのコマンドライン サポートを含みます。

InstallShield 前提条件エディターの [実行するアプリケーション] タブでコマンドラインを指定できます。

詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。

InstallShield 前提条件の再起動動作に追加された新しいオプション

InstallShield 前提条件エディターの [動作] タブでは、ターゲット マシンの再起動が必要な可能性がある場合、InstallShield 前提条件のインストールの処理方法を指定できます。**[前提条件が再起動を必要としているように見える場合]** リストで、動作を指定できます。このリストには **[記録する。マシンが再起動された場合、再開を失敗して、インストールの後に再起動する]** と呼ばれる新しいオプションが追加されました。実行時に再起動が必要であるように見える場合で、メインのインストールが終了するまで (または次の前提条件が再起動をトリガーするまで) 延期したいときは、この新しいオプションを選択します。

詳細については、「[動作] タブ」を参照してください。

最高 4 GB までの単一ファイル、圧縮 InstallScript インストールをサポート

単一ファイルの圧縮 InstallScript インストールのサイズは、最高で 4 GB まで可能です。以前、エンド ユーザーが大きな単一ファイルの圧縮 InstallScript インストールを実行した場合、インストールはセットアップの初期化中にクラッシュしました。

Windows は 4 GB を超える実行可能ファイルをロードしないため、Setup.exe ファイルは 4 GB 以下でなくてはなりません。

仮想ディレクトリを含まない IIS Web サイトをインストールできる機能と、Web サイトをコンポーネントと関連付けられる機能

InstallShield は、今回より、仮想ディレクトリを一切含まない IIS Web サイトのインストールをサポートできるようになりました。また、[IIS の構成] ビューで Web サイトを選択すると表示される [全般] タブには "コンポーネント" 設定が追加されました。この設定を使って選択された Web サイトとコンポーネントを関連付けることができます。これらの強化によって、Web サイトに関連付けられた仮想ディレクトリまたはコンポーネントがインストールされると、ターゲット マシン上にその Web サイトが作成されます。

Web サイトがコンポーネントと関連付けられている場合、Web サイトの **[アンインストール時に Web サイトを削除する]** チェック ボックスは、そのコンポーネントの "パーマネント" 設定 (基本の MSI プロジェクトまたは InstallScript MSI プロジェクトの場合) または "アンインストール" 設定 (InstallScript プロジェクトの場合) に対応します。つまり、Web サイトの **[アンインストール時に Web サイトを削除する]** チェック ボックスを選択またはクリアすると、自動的にコンポーネントの "パーマネント" 設定または "アンインストール" 設定が適切に更新されます。

以前、InstallShield では Web サイトとコンポーネントを関連付けるためのサポートが提供されていませんでした。そのため、インストールに含まれる Web サイトが仮想ディレクトリを持たない場合、実行時 Web サイトが作成されませんでした。

InstallShield 2009 の [IIS の構成] ビューを使って新しい Web サイトを追加すると、InstallShield は自動的にその Web サイトをコンポーネントに関連付けます。IIS Web サイトを含む InstallShield 2008 以前のプロジェクトをアップグレードした場合、その Web サイトは自動的にコンポーネントに関連付けられません。

基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプで、これらの IIS 強化内容をサポートします。

InstallScript プロジェクトで Web サイトがコンポーネントに関連付けられている場合、その Web サイトに追加される仮想ディレクトリも同じコンポーネントに関連付けなくてはなりません。したがって、1 つまたは複数の仮想ディレクトリを含む Web サイトのコンポーネントを変更しようとすると、InstallShield はメッセージ ボックスを表示して、Web サイトの仮想ディレクトリのすべてに対して、同じコンポーネントの変更が行われることを通知します。このメッセージ ボックスは、Web サイトに含まれる任意の仮想ディレクトリのコンポーネントを変更しようとした場合にも表示されます。どちらの場合も、メッセージ ボックスはコンポーネントの変更を続行するか、キャンセルするか選択肢を提供します。基本の MSI プロジェクトおよび InstallScript MSI プロジェクトでは、Web サイトのすべての仮想ディレクトリと同様に、Web サイトを同じコンポーネントに保持することが必須ではありませんが、推奨されます。

詳細は、次を参照してください:

- Web サイトの作成とアプリケーションまたは仮想ディレクトリの追加
- IIS サポートの機能とコンポーネントの関連付け
- Web サイト、アプリケーション、および仮想ディレクトリのアンインストール

QuickPatch パッケージの簡素化

今回より、一般的に以前の InstallShield でビルドされたパッケージに比べて新しいサブ機能とビルトイン InstallShield カスタム アクションの数が少ない、簡素化された QuickPatch パッケージのビルドが可能となりました。QuickPatch プロジェクトの [詳細] タブに追加された "QuickPatch の簡素化" 設定で、この新しいタイプの QuickPatch パッケージを作成するかどうかを指定できます。

詳細については、次を参照してください。

- QuickPatch パッケージを簡素化するかどうかを指定する
- [詳細] タブ

コマンドラインと MSBuild からパッチをビルドできる機能

–patch_config コマンドライン パラメーターが、IsCmdBld.exe を使用したコマンドライン ビルド (Standalone Build コマンドライン ビルドを含む) 用に追加されました。このパラメーターを使用して、コマンドラインからパッチをビルドすることができます。

.ini ファイルを使ってコマンドラインにパラメーターを渡す場合、.ini ファイルの [Project] セクションに追加された PatchConfigName パラメーターを使って、ビルドするパッチ構成を指示することができます。

また、MSBuild の InstallShield タスクに、MSBuild を使ってビルドするパッチ構成を指定することができる PatchConfiguration パラメーターが追加されました。

詳しくは、次を参照してください:

- ISCmdBld.exe
- .ini ファイルでコマンドライン ビルド パラメーターを渡す
- Microsoft ビルド エンジン (MSBuild)

パッチと QuickPatch パッケージをパスワードで保護する機能

今回より、パッチと QuickPatch パッケージをパスワードで保護するためのパスワード設定が追加されました。これらの設定は基本の MSI プロジェクトと InstallScript MSI プロジェクトでは [パッチのデザイン] ビューの [詳細] タブに、また QuickPatch プロジェクトでは [詳細] タブにあります。

パッチまたは QuickPatch パッケージをパスワードで保護すると、すべてのエンド ユーザーはパッケージをインストールする時に、アップデートを起動するためのパスワード (大文字と小文字を区別する) を入力しなくてはなりません。

詳しくは、次を参照してください:

- パッチ パッケージをパスワードで保護する
- QuickPatch パッケージをパスワードで保護する
- [詳細] タブ ([パッチのデザイン] ビューのパッチ構成用)
- [詳細] タブ (QuickPatch プロジェクト)

QuickPatch プロジェクトのパッチとアップグレードの検証サポート

QuickPatch プロジェクトをビルドしたとき、InstallShield はパッチとアップグレードの検証を実行するようになりました。この検証は、QuickPatch パッケージを使って製品のアップグレードを試みるときに発生する可能性のある、一般的な問題を識別するのに役立ちます。

これまでパッチとアップグレードの検証は、基本の MSI プロジェクトと InstallScript MSI プロジェクトで [パッチのデザイン] ビューで作成されたパッチでしか利用できませんでした。

詳細については、「アップグレード、パッチ、および QuickPatch パッケージを検証する」を参照してください。

ビルド時の検証に複数 .cub ファイルを選択できる機能

InstallShield の [オプション] ダイアログ ボックスにある [検証] タブでは、InstallShield 内でビルドされたインストール パッケージまたはマージ モジュールを検証する際に使用される .cub ファイルを複数指定できるようになりました。

また、ISCmdBld.exe を使ったコマンドライン ビルド (Standalone Build コマンドライン ビルドを含む) で –m コマンドライン パラメーターを使って、複数の .cub ファイルを渡すこともできるようになりました。.iniファイルを使ってコマンドラインにパラメーターを渡す場合、今回より .ini ファイルの CubFile パラメーターに 複数の .cub ファイルを指定することができるようになりました。

MSBuild 用の InstallShield タスクの RunMsiValidator パラメーターが強化され、複数の .cub ファイルを使用できるようになりました。

詳細については、次を参照してください。

- Windows ロゴ プログラムの要件
- インストール パッケージまたはマージ モジュールを検証する
- ビルド時に検証を行うかどうかを指定する
- インストール パッケージまたはマージ モジュールを検証する
- [検証] タブ ([オプション] ダイアログ ボックス)
- [ビルド] メニュー
- ISCmdBld.exe
- .ini ファイルでコマンドライン ビルド パラメーターを渡す
- Microsoft ビルド エンジン (MSBuild)

InstallShield ベストプラクティス スイートの新しい検証ツール

ISBP20 は InstallShield ベスト プラクティス スイートで利用可能な新しい検証ツールです。ISBP20 は、Registry テーブルに含まれるレジストリ エントリが、ルート レベルのレジストリ キーや、その削除によってターゲット マシン上でのトラブルの原因となるようなキーの削除が行われないことを検証します。

InstallShield ベスト プラクティス スイートは、InstallShield Premier Edition の基本の MSI、InstallScript MSI、および MSI データベース プロジェクトで利用できます。

/v コマンドライン パラメーターを複数回使用して、Setup.exe から .msi ファイルに複数のパラメーターを渡すことができる機能

Setup.exe から Msiexec.exe へ複数の引数を渡す場合、コマンドラインで /v オプションを各引数ごとに 1 回ずつ、複数回使用することができます。以前、/v オプションの使用は 1 回に限られていたため、すべてのパラメーターはこのインスタンスを通して渡されていました。

基本の MSI と InstallScript MSI プロジェクト タイプに、この強化サポートが含まれています。

詳細については、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

改善された Standalone Build と InstallShield の間の互換性

今回より、InstallShield Premier Edition で提供されている Standalone Build は、InstallShield がプログラム ファイルに使用するディレクトリ構造と同じディレクトリ構造を使用します。これにより、InstallShield があるマシンから再配布可能ファイルや他のファイルを Standalone Build があるマシンにコピーするとき、同じ相対パスを使用できるようになりました。以前、異なるディレクトリ構造が別々に使用されていました。

また、InstallShield のコマンドライン ビルドで使われる ISCmdBld.exe も Standalone Build と共にインストールされています。以前、Standalone Build は別のファイル (IsSaBld.exe) をコマンドライン ビルドに使用していました。ISCmdBld.exe は今回より、以前 IsSaBld.exe のみがサポートしていたパラメーターをサポートします。

- -o <マージ モジュール検索パス>ープロジェクトで参照されるマージ モジュール (.msm) ファイルを含むフォルダーをコンマで区切って指定します(複数指定可)。
- -t <Microsoft .NET Framework のパス>ー Microsoft .NET Framework へのパスを指定します。パスは、ビルド マシンにインストールされている .NET Framework の場所です。
- -h ー ビルドの終わりでアップグレード検証ツールをスキップすることを示します。
- -g <最小ターゲット MSI バージョン> ー ターゲット マシン上でインストールに必要な Windows Installer の最小バージョンを指定します。
- -j <最小ターゲット Microsoft .NET Framework バージョン> ー ターゲット マシン上でインストールに必要な .NET Framework の最小バージョンを指定します。

この強化の一部として、Standalone Build のインターフェイスで、InstallShield と同じ ISWiAutomation15.dll ファイルが使用されるようになりましたが、インストールされる場所は異なります。InstallShield オートメーション インターフェイスと共に動作する既存のオートメーション スクリプトがある場合、スタンドアロン オートメーション インターフェイスと共に使用するためにスクリプト全体でライブラリの名前を IswiAuto*N* から SAAuto*N* (N はバージョン番号です) に変更する作業は必要なくなりました。この変更により、ISWiProject オブジェクトに、以前 Standalone オートメーション インターフェイスでのみ提供されていたいくつかのプロパティのサポートが追加されました。

- DotNetFrameworkPath
- MergeModuleSearchPath
- MinimumTargetDotNetVersion
- MinimumTargetMSIVersion
- SelfRegistrationMethod
- SkipUpgradeValidators

互換性の改善により、別々の 2 セットのバイナリの代わりに 1 セットのバイナリのみが Standalone Build と InstallShield にビルドされるため、InstallShield ホットフィックスまたはサービス パックがリリースされたとき、それを Standalone Build と InstallShield に使用することができます。以前、別々のホットフィックスとサービス パックが必要でした。

詳しくは、次を参照してください:

- Standalone Build をビルド マシンにインストールする
- Standalone Build 用に再配布可能ファイルをビルド マシンへ追加する
- スタンドアロン コマンドライン ビルド
- スタンドアロン オートメーション インターフェイス

- Standalone Build と InstallShield を同一のマシン上にインストールする
- ISCmdBld.exe
- ISWiProject オブジェクト

SQLBrowse 実行時ダイアログで、ローカル、リモート、エイリアス SQL Server を参照が可能

SQLBrowse ダイアログのフィルター機能が強化されました。

基本の MSI と InstallScript MSI インストールの SQL Server の参照コンボ ボックスとリスト ボックス コントロールでリモート サーバーのみ表示したい場合、新しい Windows Installer プロパティ IS_SQLSERVER_REMOTE_ONLY を設定します。基本の MSI と InstallScript MSI インストールの SQL Server の参照コンボ ボックスとリスト ボックス コントロールでサーバー エイリアスのみ表示したい場合、新しい Windows Installer プロパティ IS_SQLSERVER_ALIAS_ONLY を設定します。以前は、ローカル サーバーのみを表示するフィルターだけしか利用できませんでした。これには、IS_SQLSERVER_LOCAL_ONLY プロパティを設定します。これらのプロパティを任意に組み合わせて、SQL Server 参照コンボ ボックスとリスト ボックス コントロールで、複数の種類のサーバーを表示することができます。

詳細については、「デフォルトの SQL ランタイム動作をオーバーライドする」を参照してください。

InstallScript プロジェクトに、次の 2 つの新しい InstallScript 関数が追加されました:

- SQLRTSetBrowseOption—この関数を利用して、SQL Server の参照コンボ ボックスとリスト ボックス コントロールで、ローカル サーバー、リモート サーバー、サーバーのエイリアス、またはこれらのサーバーの組み合わせを表示するかどうかを指定できます。
- SQLRTGetBrowseOption—この関数は、SQL Server の参照コンボ ボックスとリスト ボックス コントロールの参照オプションの現在の値を返します。これらでは、ローカル サーバー、リモート サーバー、サーバー エイリアス、およびこれらのサーバーの組み合わせを表示できます。

SQL データベース サーバーの最小要件が満たされない場合でもインストールを続行する機能

[SQL スクリプト] ビューで SQL 接続を選択すると表示される [要件] タブに、新しい **[最小要件が満たされていない場合でもインストールの続行を許可する]** チェック ボックスが追加されました。

このチェック ボックスを選択すると、ターゲットシステム上でデータベース サーバーの最小要件が満たされていない場合、ランタイムは SQL 接続とその SQL スクリプトのすべてをスキップしてインストールを続行します。

このチェック ボックスをクリアしていて最小要件が満たされない場合、エンド ユーザーはインストールを続行することができません。これがデフォルトの動作です。

この強化内容は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに適用します。

詳細については、「[要件] タブ」を参照してください。

SQL Server エイリアスのサポート

SQL ランタイム サポートが強化され、インストールが SQLBrowse ダイアログで SQL Server のエイリアス名をリストできるようになりました。また SQLLogin ダイアログを使って、エンド ユーザーはエイリアス名で SQL Server に接続することが可能です。

この強化内容は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに適用します。

詳細については、「新しい SQL 接続の追加」を参照してください。

.NET Framework 用の定義済みシステム検索

InstallShield に、次の新しい定義済みシステム検索が追加されました。

- Microsoft .NET Framework 3.5
- Microsoft .NET Framework 3.0 SP1
- Microsoft .NET Framework 3.0
- Microsoft .NET Framework 2.0 SP1
- Microsoft .NET Framework 2.0
- Microsoft .NET Framework 1.1
- Microsoft .NET Framework 1.0

インストールでこれらのいずれかが必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、これらのシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

この強化は、基本の MSI および InstallScript MSI プロジェクトに適用します。

エラー カスタム アクションの拡張サポート

このカスタム アクション ウィザードに、指定されたエラー メッセージを表示し、失敗を戻して、インストールを終了するタイプ 19 カスタム アクションのサポートが追加されました。以前、この種類のカスタム アクションを作成するには、[カスタム アクションとシーケンス] ビューで [カスタム アクション] エクスプローラーを右クリックしてから、[エラー] をクリックするか、またはダイレクト エディターを利用して、手動でカスタム アクションのテーブル エントリを入力する必要がありました。

基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプでエラー カスタム アクションがサポートされるようになりました。以前、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトでのみサポートされていました。

ソース ファイルにデジタル署名するかどうかを指定できる機能

[リリース] ビューの [署名] タブに、新しい **[元の場所にあるファイルに署名する]** チェック ボックスが追加されました。このチェック ボックスを利用して、InstallShield で、元のソース ファイルにも署名をするか、またはリリースに組み込まれるファイルのみ署名するかを指定できます。このチェック ボックスは、リリース ウィザードの [デジタル署名のオプション] パネルでも提供されています。このチェック ボックスは、デフォルトで、クリアになっています。

基本の MSI プロジェクトまたは InstallScript MSI プロジェクトで、このチェック ボックスを選択すると、もともと署名されていなかったファイルを含むリリースの圧縮バージョンと非圧縮バージョンの両方を更新する単一のパッチを作成する場合に便利です。

以前、このチェック ボックスがなかったため、InstallShield で元の場所にあるファイルを署名することができませんでした。

オートメーション インターフェイスには、このあたらしいデジタル署名機能のサポートが含まれています。ISWiRelease オブジェクトに、InstallShield が元のソース ファイルに署名するか、リリースに組み込まれたファイルにのみ署名するかを指定できる **SignFilesInPlace** プロパティが追加されました。

詳しくは、次を参照してください:

- リリースの [署名] タブ
- [デジタル署名のオプション] パネル
- デジタル署名とセキュリティ
- ISWiRelease オブジェクト

改良されたスタティック COM 抽出

スタティック COM 抽出を使用する場合、Registry テーブルのプライマリ キーを生成するときに InstallShield が MD5 アルゴリズムを使用するようになりました。したがって、COM データが変更されない場合、パッケージの異なるバージョン間や、抽出された COM データが更新されるときにプライマリ キーが変更されることはありません。

以前、InstallShield はスタティック COM 抽出中に作成されたプライマリ キーにはランダム値を使用しました。その結果、COM データが更新されたとき、またはパッチがビルドされたとき、COM データが変更されていないにも関わらず新しいプライマリ キーが作成される可能性がありました。パッチに関しては、プライマリ キーが変更された場合に COM データがパッチに含まれます。パッチが変更されていない COM データを持つコンポーネントを更新した場合、パッチのアンインストール中に COM データが削除される可能性があり、これは製品の以前のバージョンに問題をもたらす原因となりかねません。

この強化は、基本の MSI および InstallScript MSI プロジェクトに適用します。

InstallScript 言語の強化および新機能

InstallScript 言語が一部強化されました。

.NET Framework 3.5 および .NET Framework 3.0 SP1 の強化

新しい FOLDER_DOTNET_35 InstallScript 変数が追加されました。この変数は、.NET Framework 3.0 ファイルのパスを格納します。

Is 関数と使用するための 2 つの新しい定数が追加されました:

- REGDB_KEYPATH_DOTNET_35
- REGDB_KEYPATH_DOTNET_30_SP

.NET Framework 3.0 の SP1 またはそれ以降のサービス パックがインストールされているかどうかをクエリするとき、REGDB_KEYPATH_DOTNET_30_SP 変数を使用できます。.NET Framework 3.0 の RTM バージョンがインストールされているかどうかを検出するには、REGDB_KEYPATH_DOTNET_30 を使用します。

一時ファイルを作成する Windows API GetTempFileName を呼び出す、新しい GetTempFileNameIS 関数

GetTempFileNameIS という名前の新しい InstallScript 関数が追加されました。この関数は Windows API **GetTempFileName** を呼び出して一時ファイルを作成し、その関連アクションを実行します。

詳しくは、「GetTempFileName」を参照してください。

InstallShield 2008 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

基本の MSI プロジェクト用の新しいエンド ユーザー ダイアログ テーマ

ダイアログ テーマは、エンドユーザー ダイアログに統一感のとれた個性的な印象を与えることができる、あらかじめ定義されている 1 セットのイメージです。ボタンをクリックするだけで、プロジェクトに提供されているテーマから 1 つ選んで、Setup.exe 初期化ダイアログを含む、プロジェクトで使用されているすべての内部および外部ダイアログに適用することができます。それぞれのダイアログは [ダイアログ] ビュー内から簡単にプレビューすることができ、選択したテーマがどのように見えるかを実際に確認することができます。

一部のテーマは、InstallShield の Premier Edition と Professional Edition の両方で提供されていますが、Premier Edition のみで提供されているものもあります。詳細については、「提供されているテーマと対応するダイアログのサイズ」を参照してください。

この機能の詳細は、「ダイアログのテーマ」を参照してください。

デジタル署名の強化機能

ビルド時に、製品の実行可能ファイルを含むプロジェクト内のすべてのファイルにデジタル署名が可能になりました。また、今回より、デジタル署名に personal information exchange ファイル (.pfx) が使用できるようになりました。この機能は、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、およびマージ モジュール プロジェクト タイプでサポートされています。

[リリース] ビューで新しい [署名] タブを使って、InstallShield がファイルに署名するときに使用するデジタル署名に関する情報 (証明機関より付与されたデジタル証明書ファイルを含む) を指定します。[署名] タブでまた、デジタル署名をするインストール内のファイルを指定することもできます。リリース ウィザードを使って、すべてのデジタル署名情報を指定することもできます。

署名に .pfx ファイルを指定すると、InstallShield でファイルが署名されるとき SignTool.exe が使用されます。.spc ファイルと .pvk ファイルを指定すると、ファイルの署名に Signcode.exe が使用されます。.pfx ファイルは、より多くの異なる環境 (ロックされたビルド マシンなど) で動作するため、より頻繁に利用されています。InstallShield でデジタル署名パスワードを指定するとき、.pfx ファイルを使用している場合、パスワードのプロンプトは表示されなくなります。.spc ファイルと .pvk ファイルを使用している場合は、パスワードのプロンプトが表示されることがあります。

以前、InstallShield では、.msi ファイル、.hdr ファイル、Setup.exe ファイルのみ署名が可能でした。また、デジタル署名用に .spc ファイルと .pvk ファイルは指定できましたが、.pfx ファイルは指定できませんでした。

詳しくは、次を参照してください:

- デジタル署名とセキュリティ
- ビルド時にリリースとそのファイルにデジタル署名を行う
- リリースの [署名] タブ

新しい InstallShield ベスト プラクティス スイート (Premier Edition)

InstallShield には、InstallShield ベスト プラクティス スイートという名前の 1 セットの検証ツールが含まれています。インストールがベスト プラクティス ガイドラインに違反している場合、このスイートの InstallShield ベスト プラクティス (ISBP) 検証ツールによって警告されます。

この機能は、基本の MSI、InstallScript MSI、MSI データベース プロジェクトで提供されています。

詳細については、次を参照してください。

- InstallShield ベスト プラクティス スイート
- 検証中に実行する ICE、ISICE、ISVICE および ISBP を指定する

インターネット インフォメーション サービス (IIS) 7.0 と SSL のサポート

InstallShield には今回より IIS 7 のサポートも含まれています。

また、インストールに、Web サイトの SSL 証明書を含めることもできます。SSL サーバー証明書を含めることにより、ユーザーは Web サーバーの認証および Web コンテンツの有効性の確認を行うことができると共に、セキュリティで保護された接続を確立することができます。

詳細については、次を参照してください。

- [IIS 構成] ビュー
- Web サイトの SSL 証明書を指定する
- ターゲット システムに IIS 6 以前が搭載されているか、または IIS 6 メタベースの互換性機能があるかどうかを判別する
- InstallShield における IIS サポートのバージョン固有情報

新しく追加された Microsoft .NET 前提条件

InstallShield には今回、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加することができる .NET 関連の新しいセットアップ前提条件が多数含まれています。

- .NET Framework 2.0 (x64)
- .NET Framework 2.0 (x64) 言語パック
- .NET Framework 2.0 (IA64)
- .NET Framework 2.0 (IA64) 言語パック
- .NET Framework 3.0 (x64)

詳細については、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。

InstallScript プロジェクトに更新された Microsoft .NET オブジェクト

新たに更新された Microsoft .NET オブジェクトが、InstallShield で提供されています。このオブジェクトには、32 ビット、64 ビット x64、および 64 ビット Itanium バージョンを含む、バージョン 1.0 (SP3)、1.1 (SP1)、2.0、および 3.0 の .NET Framework のサポートが含まれています。オブジェクトにはまた、言語パックおよび、1.0 と 1.1 の最新サービス パックが含まれています。

このオブジェクトはまた、.NET オブジェクトを含む機能がインストールされたとき、.NET Framework のインストールを起動して、実行完了します。このオブジェクトを使用して .NET Framework を即座にインストールすることにより、その後にインストールされるファイルのインストールまたは構成で .NET Framework が必要になったときに、確実に使用できるようになります。

詳細については、「Microsoft .NET Framework オブジェクト ウィザード」を参照してください。

Visual C++ 8.0 マージ モジュールの追加

InstallShield に、Visual C++ 8.0 SP1 のマージ モジュール (バージョン 8.0.50727.762) が追加されました。

ダイアログ ボタンの UAC シールド アイコンのサポート (基本の MSI プロジェクト)

基本の MSI プロジェクトのダイアログ エディターで、すべてのボタン コントロールに新しい Show UAC Shield Icon プロパティが追加されました。このプロパティで True を選択すると、エンドユーザーが Windows Vista システムでインストールを実行しているとき、[ユーザー アカウント制御 (UAC)] シールド アイコンがボタンに表示さ

れます。Windows Vista システムで InstallShield を使用している場合、実行時と同じシールド アイコンがダイアログ エディターのボタンに表示されます。シールド アイコンによって、昇格された権限が必要かもしれないことがエンドユーザーに通知されます。

これから作成するすべての新しい基本の MSI プロジェクトでは、Show UAC Shield Icon プロパティは ReadyToInstall ダイアログの [インストール] ボタンについて True に設定されています。InstallShield 12 以前で作成された基本の MSI プロジェクトを InstallShield 2008 にアップグレードしたとき、[インストール] ボタンの "UAC シールド アイコンの表示" プロパティのデフォルト値は False になっています。この値は、このボタンについてのみではなく、他のすべてのボタンについても、必要に応じてオーバーライドすることができます。

LicenseAgreement ダイアログで、エンドユーザーが EULA の最後までスクロールすることを必須にする機能

InstallShield では、エンドユーザーがマウスまたはキーボードを使ってスクロール可能な EULA コントロール内にあるエンドユーザー使用許諾契約 (EULA) テキストの終わりに到達するまで、LicenseAgreement ダイアログ上の [次へ] ボタンを無効にすることができます。

エンドユーザーはまた、[次へ] ボタンが有効にされる前に **[ソフトウェア使用許諾契約に同意します]** オプションを選択する必要があります。この動作は、InstallShield の以前のリリースと同じです。

デフォルトで、スクロール要件は LicenseAgreement ダイアログで提供されていません。この機能を利用するには、プロジェクトに WatchScroll という名前の Windows Installer DLL カスタム アクションを追加する必要があります。このカスタム アクションは EulaScrollWatcher.dll ファイルを呼び出します。また、[次へ] ボタンのコントロール条件を変更し、イベントを Memo コントロールに追加する必要があります。

これは基本の MSI プロジェクトで提供されています。

詳しい手順については、「LicenseAgreement ダイアログでエンドユーザーが EULA を最初から最後までスクロールするのを必須にする」を参照してください。

追加された SQL Server 2005 Express Edition SP1 セットアップ前提条件

InstallShield に Microsoft SQL Server 2005 Express Edition SP1 のセットアップ前提条件が追加されました。このセットアップ前提条件は、基本の MSI および InstallScript MSI プロジェクトに追加することができます。

更新された DirectX 9.0c オブジェクト

InstallShield では 2 つの DirectX 9.0 オブジェクト (基本の MSI プロジェクトと InstallScript MSI プロジェクトで提供されている Windows Installer ベースのオブジェクトと、InstallScript プロジェクトで提供されている InstallScript ベースのオブジェクト) が利用できます。これらの両方のオブジェクトは、32 ビット固有および 64 ビット固有のコンポーネントを含む、最新の DirectX 9.0c コアおよびオプションのコンポーネントをすべてインストールします。

また、基本の MSIプロジェクトおよび InstallScript MSI の DirectX 9 オブジェクト ウィザードにも一部変更が加えられました。このウィザードでは今回より、再配布可能ファイルを Disk1 フォルダーに含めるか、または .msi ファイルにストリームするかを指定することができます。この変更により、圧縮インストールで DirectX 9 オブジェクトが使用できるようになりました。また、サイレント インストールでも今回より DirectX 9 オブジェクトが使用できるようになりました。

基本の MSI プロジェクトと InstallScript MSI プロジェクトで、DirectX インストールを起動するカスタム アクションは Windows Vista システムで昇格された権限を使って実行できるように、[実行] シーケンスにスケジュールされ、遅延システム コンテキストで実行されます。

詳細は、次を参照してください:

- DirectX 9.0 オブジェクトを含める

- DirectX オブジェクト ウィザード

DIFx 2.1 のサポート

InstallShield では、最新版の Driver Install Frameworks for Applications (DIFx) がサポートされています。この新しいバージョンには、マイクロソフトからの最新バイナリ ファイルが含まれているため、InstallShield で作成する基本の MSI プロジェクト、InstallScript プロジェクトまたは InstallScript MSI プロジェクトで使用することができます。この新しいバージョンは Windows Vista システムにインストールすることができます。InstallShield の以前のバージョンには、場合によって Windows Vista にインストールすることができなかった DIFx の以前のバージョンが含まれていました。

COM サーバーの 64 ビット自己登録のサポート (基本の MSI プロジェクト)

InstallShield では今回より、基本の MSI プロジェクトで、COM サーバーの 64 ビット自己登録がサポートされています。コンポーネントを 64 ビットとしてマークしてから、そのコンポーネントにファイルを追加したとき、そのファイルの [自己登録] チェック ボックスを選択して、インストール時の 64 ビット自己登録を有効にすることができます。InstallShield ではまた、動的にリンクされている COM サーバーの 64 ビット自己登録もサポートされています。詳細については、次を参照してください。

- 64 ビットのオペレーティング システムを基本の MSI および InstallScript MSI インストールを使ってターゲットする
- [ファイルのプロパティ] ダイアログ ボックス
- ダイナミック ファイル リンクをコンポーネントに追加する

セットアップ前提条件エディターの拡張されたオペレーティング システム条件の設定

セットアップ前提条件エディターで、前提条件に条件を作成するとき、より詳しいオペレーティング システム要件に関する情報を指定できるようになりました。セットアップ前提条件エディターで前提条件の条件を追加または変更するときに表示される [前提条件設定] ダイアログ ボックスでは、定義済みのオペレーティング システムを選択することもできますし、カスタム オプションを選択することもできます。新しいカスタム オプションを使用して、プラットフォーム、メジャーとマイナー バージョン、プロセッサ アーキテクチャ (64 ビットまたは 32 ビット) およびサービス パックなどのオペレーティング システム要件の設定を構成することができます。ターゲット システムがオペレーティング システムの要件を満たさない場合、前提条件はインストールされません。

詳細は、次を参照してください:

- InstallShield 前提条件のオペレーティング システム条件を追加する
- [前提条件設定] ダイアログ ボックス
- [条件] タブ

Windows Server 2008 システムをターゲットする機能

InstallShield では、Windows Server 2008 がインストールに必須であると指定することができます。また、機能およびコンポーネントに Windows Server 2008 に関連する条件をビルドすることもできます。

Windows Vista と Windows Server 2008 は、同じメジャー バージョン番号とマイナー バージョン番号を持つ点に注意してください。したがって、InstallScript を使って Windows Server 2008 と Windows Vista を区別するには、SYSINFO.nOSProductType = VER_NT_WORKSTATION が、Windows Vista の場合は TRUE、Windows Server 2008 の場合は FALSE であるかどうかを確認してください。詳細については、「SYSINFO」を参照してください。

新しい MSXML 6 SP1 セットアップ前提条件

InstallShield には今回より、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加することができる新しい MSXML セットアップ前提条件がいくつか含まれています。

- MSXML 6.0 SP1
- MSXML 6.0 SP1 (IA64)
- MSXML 6.0 SP1 (x64)

MSXML の詳しい情報については、「XML ファイルの変更の実行時の要件」を参照してください。

FlexNet Connect サポート

基本の MSI プロジェクト、および InstallScript MSI プロジェクトに FlexNet Connect 6.1 の再配布可能ファイルを追加することができます。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの [アップデート通知] ビューで、プロジェクトに含める FlexNet Connect のバージョンを選択することができます。バージョン 6.1、または [オプション] ダイアログ ボックスの [マージ モジュール] タブにある [マージ モジュールの場所] 領域で指定されている場所にインストールされている任意のバージョンを含めることができます。

[アップデート通知] ビューに、FlexNet Connect 6.1 がサポートする新しい "ベンダー データベース" 設定が追加されました。

強化機能

リリースにおけるユーザビリティの強化点

[リリース] ビューで、リリースの設定がカテゴリ別に複数のタブで再構成されました。

また、[リリース] ビュー内から "圧縮" 設定に、[圧縮] または [非圧縮] を選択できるようになりました。これまで、この設定の変更を行うには、リリース ウィザードを起動しなければなりませんでした。カスタム圧縮の設定を指定する場合 (機能ごとに 1 つの .cab ファイル、またはコンポーネントごとに 1 つの .cab ファイル)、リリース ウィザードをこれまでどおり利用する必要があります。"圧縮" 設定は、基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクト使用できます。

基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトでは、[配布] ビューにあった設定は、[リリース] ビューの新しい [ポストビルド] タブに移されました。[ポストビルド] タブには、ビルド時にリリースをフォルダーまたは FTP サイトに自動的に配布できるように構成できる設定があります。

基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトで、[リリース] ビューでリリースを右クリックしたときに新しい [配布] コマンドが表示されます。このコマンドを選択すると、リリースに関連するすべてのファイルが [ポストビルド] タブで指定された場所にコピーされます。InstallScript および InstallScript Object プロジェクトの場合、リリースは、ビルドされるたびに、自動的に [ポストビルド] タブで指定した場所 (複数可) にコピーされます。

詳しくは、次を参照してください:

- リリースの [ビルド] タブ
- リリースの [Setup.exe] タブ
- リリースの [署名] タブ
- リリースの [インターネット] タブ

- リリースの [イベント] タブ
- フォルダーまたは FTP サイトにリリースを自動的に配布する

カスタム アクションとシーケンスにおけるユーザビリティの強化

[カスタム アクション] ビューと [シーケンス] ビューが [カスタム アクションとシーケンス] ビューという名前のビューに統合され、より強力な機能を提供できるようになりました。統合されたビューでは、ドラッグアンドドロップ編集およびコピー機能がサポートされています。

- 新しいカスタム アクションをシーケンスするには、[カスタム アクション] エクスプローラーからそれを、[シーケンス] エクスプローラーの下にあるシーケンス内の適切な位置までドラッグします。
- ダイアログ、標準アクション、またはカスタム アクションを、シーケンス内の異なる位置へ(または、あるシーケンスから別のシーケンスへ) 移動するには、それを元の位置から新しい場所へドラッグします。
- カスタム アクションを別のシーケンスへコピーするには、CTRL を押しながら、カスタム アクションを別のシーケンスへドラッグします。

[カスタム アクションとシーケンス] ビューは、基本の MSI、InstallScript、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトで使用できます。

詳細については、次を参照してください。

- [カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー)
- [カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でカスタム アクションを作成する
- アクションをシーケンスに挿入する
- カスタム アクションを別のシーケンスへコピーする
- シーケンスの順序を変更する

[ファイルとフォルダー] ビュー、[レジストリ] ビュー、および [再配布可能ファイル] ビューにおけるユーザビリティの強化

[ファイルとフォルダー] ビューにおける強化内容は次のとおりです。

- **[インストール先コンピューターのファイル]** ペイン内でファイルを右クリックしてから、新しい [1 つ上のフォルダーを開く] コマンドをクリックできます。Windows エクスプローラー ウィンドウが開き、右クリックして選択したファイルを含むフォルダーが表示されます。
- **[インストール先のコンピューターのファイル]** ペインを右クリックすると、新しい [追加] ボタンを使用できます。このコマンドを使うと [開く] ダイアログ ボックスが表示され、プロジェクトに追加するファイルを参照することができます。
- このビューの右上に、新しいリンク ([ソース ペインの表示] または [ソース ペインの非表示]) が追加されました。この新しいリンクを使うと、このビューの上部に表示される **[ソース コンピューターのフォルダー]** ペインおよび **[ソース コンピューターのファイル]** ペインを表示または非表示に切り替えることができます。この 2 つのペインを非表示にして Windows エクスプローラー ウィンドウを開き、InstallShield 内に表示されている残りの 2 つのペインに Windows エクスプローラー ウィンドウから直接ファイルをドラッグ アンド ドロップすることができます。

[レジストリ] ビューの右上にも新しいリンク ([ソース ペインの表示] または [ソース ペインの非表示]) が追加されました。この新しいリンクを使うと、このビューの上部に表示される **[ソース コンピューターのフォルダー]** ペインおよび **[ソース コンピューターのファイル]** ペインを表示または非表示に切り替えることができます。

以下はまた、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの [再配布可能ファイル] ビューにおける 2 の強化点です:

- このビューの右側のペインに、左上のペインで選択されたマージ モジュール、オブジェクト、またはセットアップ前提条件に関する詳細が表示されます。このビューの右上にある [詳細の表示] または [詳細の非表示] リンクをクリックすると、この詳細ペインを表示または非表示に切り替えることができます。
- セットアップ前提条件の [詳細] ペインに、選択されたセットアップ前提条件に関する完全な情報が表示されます。この情報には、前提条件に構成されている条件、コマンドライン パラメーター、およびその他の情報が含まれます。

オートメーション インターフェイスの強化

オートメーション インターフェイスが、大幅に強化されました。

可能になった CreateProject メソッドによるマージ モジュール プロジェクトの作成

ISWiProject オブジェクトの CreateProject メソッドを使用して、マージ モジュール プロジェクトを作成することができるようになりました。以前、このメソッドは、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト プロジェクトでのみサポートされていました。

詳細については、「CreateProject メソッド」を参照してください。

ダイナミック ファイル リンクの構成をサポート

オートメーション インターフェイスに、ダイナミック ファイル リンクのための新しいオブジェクトと新しいコレクションが追加されました。また、ISWiComponent オブジェクトに、2 つの新しいメソッドと 1 つのプロパティ (コンポーネントのダイナミック ファイル リンクを追加する AddDynamicFileLinking と、削除する RemoveDynamicFileLinking、およびダイナミック ファイル リンクのコレクションを取得する ISWiDynamicFileLinkings) が追加されました。

詳細については、次を参照してください。

- ISWiDynamicFileLinking オブジェクト
- ISWiDynamicFileLinkings コレクション
- ISWiComponent オブジェクト
- AddDynamicFileLinking メソッド
- RemoveDynamicFileLinking メソッド

パス変数の構成をサポート

オートメーション インターフェイスに、プロジェクトでパス変数を構成するための新しいオブジェクトとコレクションが追加されました。また、ISWiProject オブジェクトに、パス変数のコレクションを取得 (ISWiPathVariables) でき、かつパス変数をプロジェクトに追加 (AddPathVariable) し、削除 (DeletePathVariable) することができる 2 つの新しいメソッドとプロパティが追加されました。

詳しくは、次を参照してください:

- ISWiPathVariable オブジェクト
- ISWiPathVariables コレクション
- ISWiProject オブジェクト
- AddPathVariable メソッド

- DeletePathVariable メソッド

文字列テーブル エントリの変更をサポート

オートメーション インターフェイスに、プロジェクトで言語と文字列エントリを構成するための新しいオブジェクトとコレクションが追加されました。また、ISWiLanguage オブジェクトに、文字列エントリのコレクションを取得 (ISWiStringEntries) でき、かつ文字列エントリをプロジェクトに追加 (AddStringEntry) し、削除 (DeleteStringEntry) することができる 2 つのメソッドとプロパティが追加されました。また、ISWiProject オブジェクトに、現在のプロジェクトに含まれている言語のコレクションを取得する新しいプロパティ (ISWiLanguages) が追加されました。

詳細については、次を参照してください。

- ISWiLanguage Object
- ISWiLanguages コレクション
- ISWiStringEntry オブジェクト
- ISWiStringEntries コレクション
- AddStringEntry メソッド
- DeleteStringEntry メソッド
- ISWiProject オブジェクト

環境変数の構成をサポート

オートメーション インターフェイスに、環境変数のための新しいオブジェクトと新しいコレクションが追加されました。また、ISWiComponent オブジェクトに、2 つの新しいメソッドとプロパティ (コンポーネントの環境変数を追加する AddEnvironmentVar と 削除する RemoveEnvironmentVar、および環境変数のコレクションを取得する ISWiEnvironmentVars) が追加されました。

- ISWiEnvironmentVar オブジェクト
- ISWiEnvironmentVars コレクション
- ISWiComponent オブジェクト
- AddEnvironmentVar メソッド
- RemoveEnvironmentVar メソッド

会社名、会社 URL、および会社の電話番号を設定するための ISWiProject オブジェクトのプロパティ

ISWiProject オブジェクトに新しくプロパティが追加され、[一般情報] ビューの設定の値を指定することができます。

- CompanyName
- CompanyURL
- CompanyPhone

ファイルにデジタル署名するための ISWiRelease オブジェクトのプロパティ

ISWiRelease オブジェクトに追加された新しいプロパティを使用して、ビルド時にリリース用のファイルに行うデジタル署名用の設定を構成することができます。新しいプロパティは、以下のとおりです。

- CertificatePassword
- SignFiles
- SignFilesExclude
- SignFilesInclude
- SignSignedFiles

Directory テーブルに未使用のディレクトリを保持するかどうかを指定するための ISWiRelease オブジェクトのプロパティ

ISWiRelease オブジェクトに新しく追加された KeepUnusedDirectories プロパティを使用して、このリリースをビルドしたとき InstallShield が .msi ファイルの Directory テーブルから未使用のディレクトリを削除するかどうかを指定することができます。

.NET Framework のインストール/更新のための再起動を遅延するかどうかを構成するための ISWiRelease オブジェクト プロパティ

ISWiRelease オブジェクト に追加された新しい DotNetDelayReboot プロパティ を使用して、インストールが完了するまで、ターゲット システム上での .NET Framework のインストールまたは更新に関連付けられた再起動を遅延するかどうかを指定できます。

エンドユーザーに .NET Framework をインストールするかどうかをたずねるメッセージ ボックスを表示するかどうかを指定するための ISWiRelease オブジェクト プロパティ

ISWiRelease オブジェクトに、新しい DisplayDotNetOptionDialog プロパティが追加されました。このプロパティを使用して、エンドユーザーが .NET Framework のインストールが必要がどうかを指定できるメッセージ ボックスを実行時に表示するかどうかを指定できます。

ビルド時の配布オプションを構成するための ISWiRelease オブジェクトのプロパティ

ISWiRelease オブジェクトに追加された新しいプロパティを使用して、ビルド時にリリースをフォルダーまたは FTP サイトに自動的に配布するための設定を構成することができます。新しいプロパティは、以下のとおりです。

- DistributeLoc
- DistributeToURLLoc
- DistributeToURLUserName
- DistributeToURLPassword
- DistributeAfterBuild

ISWiSequence コレクションの新しい RemoveSequenceRecord メソッド

ISWiSequence コレクションに、シーケンスからアイテムを削除することができる RemoveSequenceRecord メソッドが新しく追加されました。詳細については、「RemoveSequenceRecord メソッド」を参照してください。

基本の MSI プロジェクトにおけるサポート ファイルの追加と削除

オートメーション インターフェイスは今回より、基本の MSI プロジェクトで ISWiSetupFile オブジェクトと ISWiAdvancedFile オブジェクトをサポートします。これらのオブジェクトには、今回より基本の MSI プロジェクトで利用できるメソッド (AddSetupFile、DeleteSetupFile、AddAdvancedFile、および DeleteAdvancedFile) が含まれています。以前、これらのオブジェクトとメソッドは、InstallScript、InstallScript MSI、InstallScript オブジェクト プロジェクトでのみ提供されていました。

詳細は、次を参照してください:

- ISWiAdvancedFile オブジェクト
- ISWiAdvancedFiles コレクション
- AddAdvancedFile メソッド
- DeleteAdvancedFile メソッド
- ISWiSetupFile オブジェクト
- ISWiSetupFiles コレクション
- AddSetupFile メソッド
- DeleteSetupFile メソッド
- ISWiProject オブジェクト

[XML ファイルの変更] ビューの強化

InstallShield で、XML ファイルの変更のサポートが拡張されました。

- 今回より、インストール全部をビルドして実行する手間をかけずに、[XML ファイルの変更] ビューでプロジェクトで構成された XML ファイルの変更のみをテストすることができます。
- [XML ファイルの変更] ビューは今回より、XML ファイルで名前空間をサポートします。
- InstallShield では、XML ファイルの XML エンコードを指定することができます。

詳しくは、次を参照してください。

- XML ファイルに加えられるインストール時の変更をテストする
- XML ファイルに加えられるアンインストール時の変更をテストする
- XML ファイルで名前空間を使用する
- XML ファイルに名前空間のマッピングを宣言する
- 名前空間プレフィックスを要素に追加する
- [XML ファイルの変更] ビュー
- XML ファイルの [名前空間] タブ

XML ファイルの変更に関する詳しい情報は、InstallShield ヘルプ ライブラリの「XML ファイルの変更」セクションを参照してください。

より速くなったダイレクト エディター、文字列テーブル エディター および [ファイル] サブビュー

InstallShield のダイレクト エディターと文字列テーブル エディターで、レコードのロード時間が短縮されました。また、プロジェクトに多数のファイルが含まれているとき、InstallShield の [コンポーネント] ビュー内にある [ファイル] サブビューで、ファイルが以前のリリースに比べより一層速く表示されるようになりました。

強化された InstallScript プロジェクトにおけるユーザー アカウント制御のサポート

InstallScript プロジェクトで、[リリース] ビューの Setup.exe タブに "必要実行レベル" 設定が追加されました。この設定を使って、インストールの Setup.exe ファイルが Windows Vista プラットフォーム上でインストール (セットアップランチャー) を実行するために必要な最低権限レベルを指定できます。InstallShield が、必要レベルを指定するマニフェストを追加します。

以前は、InstallScript プロジェクトには常に最高権限マニフェストが含まれ、"必要実行レベル" 設定は基本の MSI プロジェクトまたは InstallScript MSI プロジェクトでしか利用できませんでした。

詳細については、「Windows Vista 以降のプラットフォームでのセットアップランチャーの必要実行レベルを指定する」を参照してください。

[ショートカット] ビューの強化

基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトで、[ショートカット] ビューが一部強化されました。

- ショートカットに使用されるアイコンを変更するには、そのショートカットを右クリックして、新しい **[ショートカット アイコンの変更]** コマンドをクリックします。[アイコンの変更] ダイアログ ボックスが開き、ショートカットが実行時にターゲット システムで作成されるときに使用されるアイコン ファイルと関連付けられたアイコン インデックスを選択することができます。
- [ショートカット] エクスプローラーに一覧表示されるショートカットは、ターゲット システムで使用されるアイコン イメージと共に表示されます。以前、[ショートカット] エクスプローラーでは、アイコンがショートカットに指定されていても、すべての種類のショートカットに異なるイメージが使用されていました。

強化内容についての詳細は、次を参照してください:

- ショートカットのアイコンを指定する
- [ショートカット] ビュー

One-Click Install インストールにおける Windows Vista と Internet Explorer 7 のサポート

InstallScript プロジェクトで、One-Click Install インストールを Windows Vista システムで Internet Explorer 7 と共に使用できます。One-Click Install セットアップ プレーヤーは Setup.exe ファイルに埋め込まれた形式から、外部 .ocx ファイル形式へ変更されました。セットアップ プレーヤーは適切なコマンドラインを使って、Setup.exe ファイルのダウンロードと起動を行います。これにより、限定された権限で Windows Vista 以降を使用しているエンドユーザーもインストールを実行できるようになります。インストールのマニフェストで指定された必要実行レベルにより昇格された権限が必要な場合、適切なユーザー アカウント制御 (UAC) プロンプトが Setup.exe ファイルが起動されたときに表示されます。

この新しい動作をサポートするため、新しい "One-Click Install の生成" 設定が [リリース] ビューの [インターネット] タブとリリース ウィザードの [インターネット オプション] パネルに追加されました。この設定で [はい] を選択すると、インストールに .ocx ファイルが含められます。

One-Click Install インストールについての詳細は、「標準 Web インストールと One-Click Install の違い」を参照してください。また、次も参照してください:

- リリースの [インターネット] タブ
- [インターネット オプション] パネル

SecureCustomProperties プロパティの強化されたサポート

[実行] シーケンスで昇格された権限が必要なインストールの [ユーザー インターフェイス] シーケンスにパブリック プロパティが設定されている場合、そのプロパティの値を [実行] シーケンスに渡すためには、プロパティが SecureCustomProperties プロパティの値としてリストされているか、または制限付きパブリック プロパティである必要があります。

InstallShield は今回より、場合によって [ユーザー インターフェイス] シーケンスから [実行] シーケンスに渡す必要があるプロパティを SecureCustomProperties プロパティに自動的に追加します。詳細については、「パブリック プロパティが制限付きパブリック プロパティである必要があることを指定する」を参照してください。

このサポートは、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトに適用します。

基本の MSI プロジェクトと InstallScript MSI プロジェクトにおける自動ダウングレード防止エントリ

エンドユーザーが製品の現在のバージョンで同製品の将来のメジャー バージョンを上書きインストールできないようにするためには、、[アップグレード] ビューにメジャー アップグレード アイテムがあること、製品の現在のバージョンによって将来のバージョンが上書きインストールされないようにメジャー アップグレード アイテムが適切に構成されていること、および製品に適切に構成、スケジュールされたタイプ 19 のカスタム アクションが含まれている必要があります。

新しい基本の MSI または InstallScript MSI プロジェクトを作成したとき、現在のインストールが将来のメジャー バージョンを上書きするのを防ぐためのサポートが自動的に追加されます。詳しくは、次を参照してください:

- 現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ
- ISICE19

ALLUSERS と CustomerInformation ダイアログの変更

InstallShield 2008 より、新しい基本の MSI プロジェクトでは、デフォルトで ALLUSERS プロパティに 1 が設定されます。ほとんどのインストールは、マシンごとに管理者権限を使用して実行される必要があるため、これが推奨される実装です。

InstallShield 12 以前で作成されたプロジェクトを InstallShield 2025 にアップグレードしたとき、ALLUSERS プロパティの値は自動的に変更されません。また、このプロパティが以前のプロジェクトで定義されていない場合も、自動的に追加されません。

また installShield 2008 から、デフォルトで、すべての新しい基本の MSI プロジェクトの CustomerInformation ダイアログは、エンドユーザーが製品をすべてのユーザーにインストールするか、または現在のユーザーのみにインストールするかを指定できるラジオ ボタン グループを表示しないようになっています。このダイアログについては、これが推奨される実装です。

InstallShield 12 以前で作成されたプロジェクトを InstallShield 2025 にアップグレードしたとき、CustomerInformation ダイアログは自動的に変更されません。

詳しくは、次を参照してください:

- ユーザーごとのインストールとマシンごとのインストールの違い
- ALLUSERS

コマンドラインまたは MSBuild タスク パラメーターで製品バージョンを変更する機能

-y コマンドライン パラメーターが、ISCmdBld.exe と IsSaBld.exe を使用したコマンドライン ビルド用に追加されました。このパラメーターを使用して、コマンドラインから製品のバージョンを指定することができます。

また、MSBuild の InstallShield タスクに、MSBuild で製品バージョンを指定することができる ProductVersion パラメーターが追加されました。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、プロパティ InstallShieldProductVersion として露出されます。

–y コマンドライン パラメーターまたは InstallShield タスク ProductVersion パラメーターは、製品バージョンのビルド バージョン (3 番目のフィールド) を増加するとき、特に便利です。

詳しくは、次を参照してください:

- ISCmdBld.exe
- スタンドアロン コマンドライン ビルド
- Microsoft ビルド エンジン (MSBuild)

コマンドラインまたは MSBuild タスク パラメーターで Windows Installer のプロパティ値をオーバーライドする機能

–z コマンドライン パラメーターが、ISCmdBld.exe と IsSaBld.exe を使用したコマンドライン ビルド用に追加されました。このパラメーターを使用して、Windows Installer プロパティの値をオーバーライドしたり、プロパティが存在しないとき、それを新規作成したりできます。

また、MSBuild の InstallShield タスクに新しく追加された PropertyOverrides パラメーターを利用して、Windows Installer プロパティの値をオーバーライドしたり、または、そのプロパティが存在しないとき、新しく作成したりできます。このプロパティは、InstallShield タスクの PropertyOverrides プロパティに InstallShieldPropertyOverrides ItemGroup パススルーとして露出されます。

詳しくは、次を参照してください:

- ISCmdBld.exe
- スタンドアロン コマンドライン ビルド
- Microsoft ビルド エンジン (MSBuild)

デフォルトで、IIS データに使用されている Windows Installer プロパティがレジストリに格納されます

実行時に、IIS Web サイトをインストールし、Windows Installer プロパティを使用して動的に IIS データを設定するインストールでは、プロパティとその値が次のレジストリ キーに書き込まれます:
HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Uninstall¥InstallShield Uninstall Information¥{ProductCode}

この変更は、アンインストールと修復の実行時に、この値を使用可能にするために行われました。したがって、Web サイトがエンドユーザーによって指定されたサイト番号にインストールされた場合、Web サイトと仮想ディレクトリはこのサイト番号から正常にアンインストールされるようになります。IIS データをレジストリに格納しないほうが良い場合、プロジェクトで IS_IIS_DO_NOT_USE_REG プロパティを設定します。

この変更は、基本の MSI および InstallScript MSI プロジェクトに適用します。

IIS Web サーバーで CMD コマンドが SSI #exec ディレクティブに使用されるのを許可するかどうかを指定するための新しい設定

IIS Web サーバーを構成して、#exec ディレクティブの CMD コマンドがシェル コマンドの実行に使用されるのを防いだり、CMD コマンドがこのタイプのコマンドの実行に使用されることを許可することができます。HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥W3SVC¥Parameters レジストリ キーの SSIEnableCmdDirective レジストリ値によって、CMD コマンドが許可されているかどうかが判別されます。

InstallShield の [IIS 構成] ビューに、新しい "SSIEnableCmdDirective レジストリ値" 設定が追加されました。この設定を使って、インストールがターゲット システム上で SSIEnableCmdDirective レジストリ値をどのように構成するのかを指定することができます。また、SSIEnableCmdDirective レジストリ値を実行時に変更しないように指定することもできます (デフォルト動作)。

詳細については、「Web サーバーで CMD コマンドが SSI #exec ディレクティブに使用されるのを許可するかどうかを指定する」を参照してください。

IIS Web サイト用の新しい "ホスト ヘッダー名" 設定

[IIS 構成] ビュー内の Web サイトについて、[Web サイト] タブに追加された新しい "ホスト ヘッダー名" 設定を使って、インストール中に追加された IIS Web サイトを識別するホスト ヘッダー名を指定することができます。以前は、ホスト ヘッダー名を指定するには、同じビュー内の [詳細] タブにある ServerBindings プロパティで構成しなくてはなりませんでした。

詳細については、「Web サイトの IIS ホスト ヘッダー名を指定する」を参照してください。

IIS Web サイトとその仮想ディレクトリを次に利用できる新しいサイト番号にインストールできる機能

InstallShield で、IIS Web サイトとその仮想ディレクトリを次に利用できる新しいサイト番号にインストールすることができます。実装は、作業をしているプロジェクトの種類によって若干異なります。詳細については、「TCP ポート番号とサイト番号の構成」を参照してください。

新しい SQL 接続で同じ Windows Installer プロパティを共有するかどうかを指定できる機能

[オプション] ダイアログ ボックスに新しく追加された [SQL スクリプト] タブで、新規のデータベース接続がデフォルトでどのように作成されるかを指定することができます: プロジェクトの最初の接続にデフォルトで使用された Windows Installer プロパティを使用する方法と、新しい一意の Windows Installer プロパティ セットを使用する方法があります。

これは、基本の MSI および InstallScript MSI プロジェクト タイプに適用されます。

詳細は、次を参照してください:

- 新しい SQL 接続が同じ Windows Installer のプロパティを共有するかどうかを指定する
- [SQL スクリプト] タブ

SQLLogin ダイアログの強化

SQLLogin ダイアログに、エンドユーザーがターゲット データベース カタログの名前を指定できるようにする新しいコントロールが追加されました。新しいコントロールのとなりには [参照] ボタンもあり、エンドユーザーはこれをボタンを利用して、ターゲット データベース サーバーで提供されているデータベース カタログの一覧から選択することができます。

SQL サポートを含む基本の MSI プロジェクトを InstallShield 12 以前から InstallShield 2025 にアップグレードした場合、新しいバージョンの SQLLogin ダイアログを使用するには、SQLLogin.isd と SQLBrowse.isd ダイアログを手動でプロジェクトにインポートする必要があります。.isd ファイルは、*InstallShield Program Files フォルダー*¥Support にインストールされます。この更新された SQLLogin ダイアログを InstallScript プロジェクトまたは InstallScript MSI プロジェクトで使用する場合、InstallScript コード内の **SQLServerSelectLogin** 関数の呼び出しを新しい **SQLServerSelectLogin2** の呼び出しで置き換えてください。

データベース インポート ウィザードの強化

データベース インポート ウィザードに、データベース スクリプト、データベース ユーザーとデータベース ロール スクリプト、SQL Server ログイン スクリプト、オブジェクト レベル権限スクリプトなどのセキュリティ スクリプト作成オプションを指定することができる新しい [スクリプト作成詳細オプション] パネルが追加されました。

詳細については、「[スクリプト作成詳細オプション] パネル」を参照してください。

インストールまたはアンインストールしない SQL 接続を指定するための新しい Windows Installer プロパティ

InstallShield に、新しい Windows Installer プロパティ IS_SQLSERVER_CXNS_ABSENT_FROM_INSTALL のサポートが追加されました。このプロパティを使用して、インストールまたはアンインストール中にスキップする 1 つまたは複数の SQL 接続を指定することができます。1 つ以上の SQL 接続を指定するには、各接続をセミコロン (;) で区切ります。すべての SQL 接続をスキップするには、このプロパティの値を **ALL** に設定します。このプロパティは、SQL スクリプト作成エラーが原因で製品をアンインストールできないとき便利です。

SQL 関連のプロパティの詳細については、「デフォルトの SQL ランタイム動作をオーバーライドする」を参照してください。

.msi ファイルから参照されていないディレクトリを削除する機能

[リリース] ビューの [ビルド] タブに、新しい "未使用のディレクトリを保持する" 設定が追加されました。この設定を使って、選択されたリリースをビルドするときに、InstallShield が .msi ファイルの Directory テーブルから未使用のディレクトリを削除するかどうかを指定することができます。デフォルト値は [いいえ] です。

この設定は、基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトで使用できます。

詳細については、「リリースの [ビルド] タブ」を参照してください。

[コンポーネント サービス] ビュー内から COM+ コンポーネント ファイルのインストール先を指定する機能

[コンポーネント サービス] ビューの [インストール] タブに次の 2 つの新しい "インストール先" フィールド (サーバー タイプのインストール用と、プロキシ タイプのインストール用) が追加されました。これらのフィールドを使用して、選択した COM+ コンポーネント ファイルのターゲット先を指定することができます。以前、COM+ アプリケーションに関連付けられたコンポーネントのインストール先は、[コンポーネント] ビューまたは [セットアップ デザイン] ビューでしか指定できませんでした。

InstallFinalize アクションの後 COM+ アプリケーションがインストールされるように指定するための新しい Windows Installer プロパティ

InstallShield に、新しい Windows Installer プロパティ IS_COMPLUS_INSTALL_AT_FINALIZE のサポートが追加されました。このプロパティを使用して、ISComponentServiceFinalize アクションによってインストールされる COM+ アプリケーションを指定できます(複数可)。ISComponentServiceFinalize アクションは InstallFinalize アクションの後に呼び出される。1 つ以上の COM+ アプリケーションを指定するには、それぞれをセミコロン (;) で区切ります。すべての COM+ アプリケーションが ISComponentServiceFinalize アクションによってインストールされるように指定するには、このプロパティの値を **ALL** に設定します。

Windows Installer は、InstallFinalize アクションが実行されるまで、スクリプト内のセッションで加えられた変更を GAC にコミットしないため、このプロパティは、GAC にインストールされる .NET アセンブリを含む COM+ アプリケーションをインストールするときに便利です。

基本の MSI プロジェクトと InstallScript MSI プロジェクトに追加された定義済みシステム検索

InstallShield に、次の新しい定義済みシステム検索が追加されました。

- Adobe Reader 7
- Adobe Reader 6
- Internet Explorer 7.0

インストールでこれらの製品のいずれかが必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、これらのシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

新規および強化された Setup.exe コマンドライン パラメーターと Update.exe コマンドライン パラメーター (InstallScript プロジェクト)

Setup.exe と Update.exe に、次の 2 つの新しいコマンドライン パラメーターが追加されました:

- /installfromweb
- /media_path

また、16 進と 10 進の値が両方 /L パラメーターでサポートされるようになりました。

詳細については、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

ダウンローダー タイプの Web リリースで増加された .cab ファイルにサポートされている場所のオプション

ダウンローダー タイプの Web リリースで、インストール時に必要に応じてダウンロードされる外部 .cab ファイルを InstallShield で作成するかどうかを指定することができるようになりました。以前、ダウンローダー タイプの Web リリースでは、.cab ファイルは常に .msi パッケージにストリームされていました。

新しい外部 .cab ファイルのオプションがリリース ウィザードの [ダウンローダーのオプション] パネルに追加されました。このパネルは、メディアの種類が Web で、Web タイプが [ダウンローダー] の場合、リリース ウィザードで表示されます。[ダウンローダーのオプション] パネルに今回より、新しい [外部 .cab ファイルを作成する] チェック ボックスがあります。このチェック ボックスをクリアした場合の動作は以前と同じで、.CAB ファイルが .msi パッケージにストリームされます。このチェック ボックスを選択すると、.cab ファイルの作成方法 (機能ごとに 1 つの .cab ファイル、コンポーネントごとに 1 つの .cab ファイル、または指定した特定のサイズに基づいて複数の .cab ファイルを作成) を指定することができます。

すべてのトランスフォームが埋め込まれるようになったダウンローダーと Web タイプの Web リリースからのインストール

ダウンローダー タイプの Web リリースと Web タイプの Web リリースからのインストールでは今回より、すべての .mst ファイルと .ini ファイルを Setup.exe ファイルに埋め込みます。

パッチ表示情報における強化

[識別] タブ (以前は [アンインストール] タブと呼ばれていました) では、Windows Installer 3.0 以降を実行しているシステムの [プログラムの追加と削除] でパッチについて表示される情報を指定することができます。基本の MSI または InstallScript MSI プロジェクトの [パッチのデザイン] ビューおよび QuickPatch プロジェクトの [一般情報] ビューにあるこのタブには、表示名、製造元名、サポート URL などのアイテムのための設定があります。[

パッチのデザイン] ビューまたは QuickPatch プロジェクトでパッチ構成の最新セットアップを変更するたびに、その最新セットアップからの [プログラムの追加と削除] 情報が [識別] タブにある設定用の値として使用されます。必要に応じて、[識別] タブの値をオーバーライドすることもできます。

また、**[パッチのアンインストールを許可する** (Windows Installer 3.0 が必要)] チェック ボックスも今回 [共通] タブに追加されました。この設定は以前、[アンインストール] タブで提供されていました。

詳細については、次を参照してください。

- [共通] タブ ([パッチのデザイン] ビュー)
- [識別] タブ ([パッチのデザイン] ビュー)
- [共通] タブ (QuickPatch プロジェクト)
- [識別] タブ (QuickPatch プロジェクト)

最短初期化時間の指定機能

[リリース] ビューにあるリリースについての Setup.exe タブに "最短初期化時間" 設定が新しく追加されました。この設定を利用して、エンドユーザーがこのリリースを実行した時に、インストールが初期化ダイアログ (およびスプラッシュ画面) を表示する最短時間 (秒) を指定できます。

InstallScript 言語の強化および新機能

InstallScript 言語が一部強化されました。

アプリケーションを昇格された権限で起動するとき LaunchAppAndWait に優先する新しい LaunchApplication 関数と WaitForApplication 関数

LaunchApplication 関数は Windows API 関数 **CreateProcess** または Windows API 関数 **ShellExecuteEx** を使用して、指定されたアプリケーションを起動します。アプリケーションが起動されたあと、インストールで新しい **WaitForApplication** 関数を呼び出して (オプション)、アプリケーションが終了するのを待機することができます。

WaitForApplication 関数は、結果を返す前に、実行されているアプリケーション、および実行されているアプリケーションによって起動されたすべての子アプリケーション (オプション) が終了するのを待機します。

今回より、**LaunchAppAndWait** を呼び出すと、次が呼び出されますので注意してください:

LaunchApplication(szProgram, szCmdLine, "", LAAW_STARTUPINFO.wShowWindow, LAAW_PARAMETERS.nTimeOut, nOptions | LAAW_OPTION_CHANGEDIRECTORY | LAAW_OPTION_FIXUP_PROGRAM);

新しい **LaunchApplicationInit** 関数が追加されました。**LaunchApplication** 関数のネーミング規則に合致するために追加されたこの関数は、**LaunchAppAndWaitInitStartupInfo** 関数と同じ動作をします。

詳細については、次を参照してください。

- LaunchApplication
- WaitForApplication
- LaunchAppAndWait
- LaunchApplicationInit

2 つ新しい定義済み定数が追加されました:

- LAAW_OPTION_CHANGEDIRECTORY
- LAAW_OPTION_FIXUP_PROGRAM

- LAAW_OPTION_USE_SHELLEXECUTE
- LAAW_OPTION_WAIT_INCL_CHILD

また、LAAW_OPTION_NO_CHANGEDIRECTORY は現在使用されていません。このパラメーターを渡しても、何も効果はありません。

次のスクリプト変数もまた、追加になりました:

- LAAW_SHELLEXECUTEINFO
- LAAW_SHELLEXECUTEVERB

InstallScript MSI プロジェクトで再起動マネージャー インフラストラクチャで再起動回数を最小化するための新しい SdRMFilesInUse ダイアログと OnRMFilesInUse イベント ハンドラー

新しい **SdRMFilesInUse** 関数が追加されました。この関数は、開いた状態でファイルをロックしているアプリケーションの一覧を表示するリスト ボックスを含むダイアログを表示します。ダイアログにはまた、インストールが、
1) 再起動マネージャーを使用して、ファイルをロックしているアプリケーションを閉じる試みをするか、または
2) ロックされているファイルを上書きする試みをするか (結果として、インストールの完了に再起動が必要になる可能性が高くなります) エンドユーザーが指定できる 2 つのラジオ ボタンがあります。

InstallScript MSI プロジェクトで、新しい OnRMFilesInUse イベント ハンドラーで、新しい **SdRMFilesInUse** ダイアログが表示されます。このイベント ハンドラーは、再起動マネージャーが有効になっていて、Windows Installer 4.0 が INSTALLMESSAGE_RMFILESINUSE メッセージをインストールに送ったときに呼び出されます。

詳細については、次を参照してください。

- SdRMFilesInUse
- OnRMFilesInUse

プロジェクトごとに、追加のコンパイラ インクルード パスを指定できる機能

InstallShield の [ビルド] メニューで使用できる [設定] ダイアログ ボックスの [コンパイル/リンク] タブに、[インクルード パス] ボックスが追加されました。この [インクルード パス] ボックスを使用して、#include ステートメントによってメイン インストールの InstallScript コードに含められたソース ファイルを探すときに InstallShield が検索するディレクトリを指定します。このボックスは、コマンドライン コンパイラ Compile.exe の -i オプションに対応しています。

詳細については、「[コンパイル/リンク] タブ」を参照してください。

.NET Framework 3.0 サポートの強化

新しい FOLDER_DOTNET_30 InstallScript 変数が追加されました。この変数は、.NET Framework 3.0 のパスを格納します。この変数に対応するテキスト置換は、<FOLDER_DOTNET_30>です。

.NET Framework 3.0 は、レジストリがインストールされていることを示すとき、レジストリに *InstallSuccess* という値を 1 という値データと共に書き込みます。今回より、この InstallSuccess 値を確認するために、**Is** 関数を使って、DOTNETFRAMEWORKINSTALLED 定数を渡すことができます。この定数を **Is** 関数を使って渡すことにより、.NET Framework の以前のバージョンで使用されている *Install* の値も従来どおり確認することができます。詳細については、「Is」を参照してください。

Is 関数と使用するための 2 つの新しい定数が追加されました。

- REGDB_KEYPATH_DOTNET_30
- REGDB_VALUENAME_INSTALLSUCCESS

.NET Framework の最小サービス パック番号の存在を確認するための新しい Is 定数

新しい DOTNETSERVICEPACKINSTALLED 定数を **Is** 関数と使用して、.NET Framework の特定のサービス パック（または、より新しいバージョンのサービス パック）がインストールされているかどうかを判断します。詳細については、「Is」を参照してください。

SQL サポート用の新規追加および更新された関数

SQL サポート用に、いくつかの InstallScript 関数が新しく追加および更新されました。

InstallScript プロジェクトと InstallScript MSI プロジェクト用に、次の新しい InstallScript 関数が追加されました:

- SQLRTInitialize2—InstallScript プロジェクトでは SQLRT.dll ファイルをロードし、InstallScript MSI プロジェクトでは ISSQLSRV.dllファイルをロードします。また SQLRTInitialize2 は、設定ファイルを使用して .dll ファイルを初期化します。この関数は SQLRTInitialize 関数に優先します。
- SQLServerSelectLogin2—ターゲットされたエンド ユーザーが、現在の接続に使用する SQL Server および使用するログインの資格情報を指定できるログイン ダイアログを作成します。オプションで、接続情報に関連付けられた接続名をこのダイアログに表示することもできます。またオプションで、エンドユーザーは現在の接続にどのデータベース カタログを使用するのかを指定することができます。この関数は SQLServerSelectLogin 関数に優先します。
- SQLDatabaseBrowse—エンド ユーザーが、指定されたデータベース サーバー上で使用できるすべてのデータベース カタログを一覧表示することができるダイアログを作成します。
- SQLBrowse2—エンド ユーザーが、接続に指定されたデータベース テクノロジ用にネットワーク上で提供されているすべてのデータベース サーバーのリストを表示できるようにするダイアログを作成します。この関数は SQLBrowse 関数に優先します。
- SQLRTPutConnectionInfo2—接続情報（デフォルト サーバー、デフォルト データベース カタログ、デフォルト ユーザー名、デフォルト パスワード）を設定します。この関数は SQLRTPutConnectionInfo 関数よりも優先されます。

InstallScript プロジェクト用に、次の新しい InstallScript 関数が追加されました。

- SQLRTGetLastError2—SQL ランタイムで最後に発生したエラーについて詳細な情報を返し、適切な SQL エラー メッセージをロードします。この関数は SQLRTGetLastError 関数に優先します。
- SQLRTGetErrorMessage—接続を開いているときに、SQL ランタイムで最後に発生したエラーについての説明を返します。
- SQLRTGetScriptErrorMessage—SQL スクリプトが実行しているときに、SQL ランタイムで最後に発生したエラーについての説明を返します。

InstallScript MSI プロジェクト用に、次の新しい InstallScript 関数が追加されました。

- SQLRTTestConnection2—接続を確立します。この関数は SQLRTTestConnection 関数に優先します。

次の InstallScript 関数は、以前 InstallScript プロジェクトでのみ提供されていましたが、今回より InstallScript MSI プロジェクトでも使えるようになりました:

- SQLRTGetConnections
- SQLRTGetConnectionInfo
- SQLRTPutConnectionInfo
- SQLRTGetConnectionAuthentication

- SQLRTPutConnectionAuthentication

InstallShield では、新しいバージョンに優先される関数も、従来どおりサポートされています。ただし、InstallShield 12 以前で作成されたプロジェクトを InstallShield 2025 にアップグレードしても、既存の InstallScript コードは新しい関数を使えるように自動的に更新されません。

詳細については、次を参照してください。

- InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する
- SQL 関数

ユーザーが管理者グループに属するかどうかを確認するための新しい Is 定数

新しい USER_INADMINGROUP 定数を **Is** 関数で使用すると、ユーザーがインストールを標準アクセス トークンを使って実行しているかどうかにかかわらず、ユーザーが管理者グループかどうかを判別することができます。詳細については、「Is」を参照してください。

Is 関数を使って USER_ADMINISTRATOR 定数を渡したとき、今回より、ユーザーが管理者グループに属していても、グループの SE_GROUP_USE_FOR_DENY_ONLY セキュリティ ID (SID) 属性が設定されている場合 (つまり、ユーザーが標準アクセス トークンを使ってインストールを実行している場合)、FALSE を返します。

InstallScript を使ってロードされた .NET ライブラリをインストールが完了する前にアンロードできるようにする新しい関数

新しい 2 つの InstallScript 関数 (DotNetCoCreateObject と DotNetUnloadAppDomain) を利用できます。

- **DotNetCoCreateObject** 関数は、アセンブリが COM 相互運用性のために登録されることなく .NET アセンブリの関数を呼び出します。この関数では、.NET アセンブリがロードされ実行される .NET アプリケーションのドメインを指定することができます。

  DotNetCoCreateObject 関数は、CoCreateObjectDotNet 関数に類似しています。唯一異なる点は、**DotNetCoCreateObject** と使用すると、ロードする .NET アプリケーション ドメインを指定できるという点です。このドメインで、アセンブリが実行されます。

  CoCreateObjectDotNet の場合、.NET アセンブリは、インストールが完了したあと、デフォルトのアプリケーションにロードされます。このため、.NET アセンブリ ファイルは、インストールが完了するまでロックされます。

- **DotNetUnloadAppDomain** 関数は、指定された .NET アプリケーション ドメインをアンロードし、現在ロードされているアセンブリをすべて指定されたアプリケーション ドメインにリリースします。

新しい RegDBDeleteItem 関数

RegDBDeleteItem という名前の新しい InstallScript 関数が追加されました。**RegDBDeleteItem** 関数は、nItem の値に従ってアプリケーションごとのパスキーまたはアプリケーション アンインストール キーの下にある値を削除します。詳細については、RegDBDeleteItem を参照してください。

オブジェクトのための新しい GetStatus 関数

GetStatus という名前の新しい InstallScript 関数が、InstallScript オブジェクト プロジェクトで使用できるようになりました。**GetStatus** 関数は、オブジェクトの現在のステータス (Status.Number の現在の値) を取得します。詳細については、「GetStatus」を参照してください。

ListWriteToFileEx 関数とそれに関連する定数への変更

LWTF_OPTION_APPEND_TO_FILE 定数が、InstallScript 言語に追加されました。この定数を **ListWriteToFileEx** 関数の nOptions パラメーターとして渡すことにより、既存のファイルにリストを追加することができます。

また、2 つの既存の nOptions 定数の名前が変更されました。

- LWFT_OPTION_WRITE_AS_ANSI は LWTF_OPTION_WRITE_AS_ANSI という名前に変わりました。
- LWFT_OPTION_WRITE_AS_UNICODE は LWTF_OPTION_WRITE_AS_UNICODE という名前に変わりました。

後方互換性を維持するために、上記の 2 つの LWFT_* 定数は継続して使用することができます。これらの定数の定義は、対応する新しい LWTF_* 定数と同じです。

詳細は、次を参照してください:

- ListWriteToFileEx
- LWTF_OPTION_APPEND_TO_FILE
- LWTF_OPTION_WRITE_AS_ANSI
- LWTF_OPTION_WRITE_AS_UNICODE

InstallScript テキスト置換機能の拡張

今回より、InstallScript テキスト置換の関連付けを、別のテキスト置換の関連付けに埋め込むことができるようになりました。たとえば、"<MYTEXTSUB1>" を "My Text Sub 1 Value" に、"<MYTEXTSUB2>" を "Text Sub <MYTEXTSUB1> Embedded" にというように関連付けることができます。以前、ローカルのテキスト置換の関連付けがグローバルのテキスト置換の関連付けに埋め込まれた場合、ローカルのテキスト置換の関連付けは実行されませんでした。

InstallShield 12 SP1 の新しい機能

InstallShield 12 は、Microsoft Windows Vista のベータ版をサポートしています。InstallShield 12 SP1 には、Microsoft Windows Vista の RTM 版のサポートを提供する変更が含まれています。

新しい Microsoft .NET 3.0 前提条件条件

InstallShield には今回、基本の MSI プロジェクトと InstallScript MSI プロジェクトに追加することができる .NET Framework 3.0 前提条件が含まれています。

セットアップ前提条件に関する詳しい情報は、「InstallShield 前提条件、マージ モジュール、オブジェクトを基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加する」をご覧ください。

アドバタイズを使用して、前提条件インストールで発生する UAC プロンプトの回数を最小限にする機能

[リリース] ビューの新しい "前提条件が昇格必要時のアドバタイズ" 設定では、Windows Vista マシンにおけるインストールでセットアップ前提条件が昇格された権限で正常にインストールされたあとに、.msi ファイルをアドバタイズして、実行するかどうか指定することができます。アドバタイズすることにより、エンドユーザーにユーザー アカウント制御 (UAC) のプロンプトを避けることを許可できる場合があります。アドバタイズを行わなかった場合、このプロンプトが表示され、昇格された権限を必要とする .msi パッケージを要求します。詳細については、「InstallShield 前提条件が昇格された権限で実行されるときに、製品をアドバタイズするかどうか指定する」を参照してください。

"前提条件が昇格必要時のアドバタイズ" 設定は、インストールで、昇格された権限を要求するための UAC の同意プロンプトまたは資格情報プロンプトを起動するかどうかを判断するときに関わってくる InstallShield の設定の 1 つです。Windows Vista システムでエンドユーザーがインストールを実行する際の UAC の動作を適切に定義するためには、これらの異なる設定をよく理解することが必要です。また、これらの設定を使って、場合により、インストール中に表示される UAC プロンプトの数を最小限に抑えることもできます。これらの各設定についての情報は、「インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する」を参照してください。

Certified for Windows Vista ロゴ プログラム用に強化された検証

InstallShield に新しく追加された InstallShield 内部整合性評価プログラム (ICE) を利用して、作成したインストールが Certified for Windows Vista ロゴ プログラムで定義されているベスト プラクティスに十分に準拠していることを確認することができます。新しい評価ツール (ISICE11 ～ ISICE20) は、すべての .exe ファイルに埋め込みマニフェストがあること、保護されているレジストリ キーが変更されていないこと、.dll および .exe ファイルで古い形式の API が使われていないことなどを確認します。また、ISICE02 は、今回より、.ocx、.sys、.cpl、.drv、および .scr ファイルが署名されていることを検証します。これまで、ISICE02 は、exe ファイルの署名だけ検証していました。ISICE06 は今回より、保護された Windows ファイルと同じ名前を持つインストール内のファイルのパスを確認します。インストール内のファイルのインストール先パスが、保護された Windows ファイルの場所と一致しないとき、ISICE06 は今回より、検証エラーの代わりに検証警告を表示します。

パッケージに検証を実行すると、検証メッセージが出力ウィンドウの [タスク] タブに一覧表示されます。[タスク] タブの検証メッセージをダブルクリックすると、今回より、そのメッセージに対応するダイレクト エディターの領域にすぐ移動できます。この機能は、標準 ICE 同様、ISICE でも利用できます。以前は、標準 ICE にのみ提供されていました。

詳細については、次を参照してください。

- ISICE
- 検証結果を表示する

追加の変更

InstallShield 12 SP1 で解決されている問題の一覧は、リリース ノートをご覧ください。リリース ノートは、InstallShield の [ヘルプ] メニューからご覧になることができます。

InstallShield 12 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

Windows Vista システムをターゲットする機能

InstallShield では、Windows Vista がインストールに必須であると指定することができます。また、機能およびコンポーネントに Windows Vista に関連する条件をビルドすることもできます。

Certified for Windows Vista ロゴ プログラムの検証

エンドユーザーに最高のインストール エクスペリエンスを提供するため、作成したインストールが Certified for Windows Vista ロゴ プログラムで定義されているベスト プラクティスに準拠していることが大切です。InstallShield には今回より、基本の MSI プロジェクトと InstallScript MSI プロジェクトで作成したインストールが Certified for

Windows Vista ロゴ プログラムのインストール要件を満たしているかどうかを検証する検証スイートが搭載されています。Windows Vista ロゴ を使用する場合、アプリケーションのインストールがプログラムの要件を満たしている必要があります。

また、インストール パッケージの検証およびマージ モジュールの検証に使用する内部整合性評価プログラム (ICE) の評価項目をカスタマイズすることも可能になりました。このため、ロゴ資格認定を取得する必要はないけれども、検証で作成中のインストールまたはマージ モジュールが特定のベスト プラクティスに適合していることを確認したいときに便利です。検証の設定を構成するには、[ツール] メニューで、[オプション] をクリックし、[検証] タブを選択します。

詳しくは、次を参照してください:

- ビルド時に検証を行うかどうかを指定する
- 検証中に実行する ICE、ISICE、ISVICE および ISBP を指定する
- ISICE

ユーザー アカウント制御のサポート

InstallShield では、Microsoft が Windows Vista に追加したユーザー アカウント制御機能がサポートされています。[一般情報] ビューに新しく追加された "管理者権限" 設定を使って、インストールが管理者権限を必要とするかどうかをプロジェクト全体に対して指定することができます。また、[リリース] ビューの [必要実行レベル] 設定では、Setup.exe ファイルが Windows Vista プラットフォーム上でインストール (セットアップランチャー、すべてのセットアップ前提条件、および .msi ファイル) を実行するときに必要な最小特権レベルを指定できます。

詳細については、次を参照してください:

- 概要情報ストリーム データを入力する
- Windows Vista 以降のプラットフォームでのセットアップランチャーの必要実行レベルを指定する

再起動マネージャー インフラストラクチャによる再起動回数の最小化をサポート

インストール終了後のシステム再起動は、エンドユーザーにとって不都合なものです。Certified for Windows Vista ロゴ プログラムの要件の 1 つに、エンドユーザーがインストール完了後自動的にアプリケーションを閉じて再起動を行うことができるオプションを含まなくてはならないという項目があります。

この品質ガイドラインをサポートするため、すべての基本の MSI プロジェクトで MsiRMFilesInUse ダイアログが提供されています。インストール中に更新が必要なファイルが他のアプリケーションによって使用中の場合、インストールでこのダイアログが表示されます。詳細については、「Windows Vista 以降のシステムの再起動を最小限にする」を参照してください。

デジタル署名の強化

インストールにデジタル署名情報を指定した場合、InstallShield は自動的に MsiDigitalCertificate および MsiPatchCertificate テーブルへ必要な情報を追加します。**MsiPatchCertificate** テーブルには、ユーザー アカウント制御 (UAC) のパッチを有効化するために必要な情報が含まれています。これによって、非管理者がインストールすることができるパッチの作成が可能になります。

また、プロジェクト アシスタントの [インストールのビルド] ページで、インストールにデジタル署名情報を指定できるようになりました。

詳細については、次を参照してください。

- 非管理者パッチのインストールを準備する

- デジタル署名とセキュリティ

カスタム アクション動作の文書化をサポート

各カスタム アクションの意図された動作は、Certified for Windows Vista ロゴ プログラムに基づいて文書化する必要があります。これは、システム管理者が製品をエンタープライズ環境に配布する場合など、カスタム アクションの動作を把握する必要がある場面において特に有益です。この要件を満たすための支援として、InstallShield の [カスタム アクション] ビューに新しい "ヘルプ ファイル パス" 設定が追加されました。この設定を使って、プロジェクトに作成したカスタム アクションの動作について説明するファイルへのパスを指定します。詳細については、次を参照してください:

- カスタム アクションの動作をドキュメント化する
- Windows ロゴ プログラムの要件を満たすカスタム アクションを作成するときのガイドライン

この文書は、様々な機能をサポートするために InstallShield プロジェクトに自動的に追加されるビルトイン InstallShield カスタム アクションそれぞれについて説明します。「InstallShield カスタム アクション リファレンス」を参照してください。

セットアップ前提条件の強化

セットアップ前提条件エディターが大幅に強化されました。

- ユーザー インターフェイスが改善され、簡単にアクセスできるコマンドを一覧表示するメニューが追加されました。
- 新しい [動作] タブでは、エンドユーザーが前提条件のインストールをスキップできるかどうかを指定することができます。また、ターゲット マシンの再起動の要否に応じて、前提条件のインストールをどのように進めるかを指定することもできます。
- DWORD レジストリを比較するときの前提インストール条件を作成できるようになりました。64 ビット マシン用の条件の作成も可能になりました。

詳細については、次を参照してください。

- [動作] タブ
- InstallShield 前提条件のインストールの要否をエンド ユーザーが選択できるようにする
- InstallShield 前提条件のインストールに伴う可能性のある問題の対処方法について
- 再起動を必要とする InstallShield 前提条件の動作を指定する

セットアップ前提条件の再配布は、これまで以上に柔軟になりました。インストールに含まれる個々のセットアップ前提条件について異なる提供方法を指定することができるようになりました。これによって、セットアップ前提条件ファイルを部分的に、ソース メディアに格納したり、Setup.exe に圧縮して実行時に抽出したり、ダウンロードしたりすることが可能になりました。さらに、セットアップ前提条件にリリース フラグを割り当てることができるようになりました。このため、異なるリリースのビルド時にセットアップ前提条件を含めたり除外するなど自由に組み合わせることが可能です。

詳しくは、次を参照してください:

- 特定の InstallShield 前提条件の実行時の場所を指定する
- リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する
- リリース フラグを InstallShield 前提条件に割り当てる

プロジェクト全体における Windows Installer 4.0 ログ ファイルのサポート

InstallShield では Windows Installer 4.0 を使って実行されるインストールについて、コマンドラインの使用やレジストリを使ったログ パラメーターの設定などの必要なしに、プロジェクト全体で簡単にログ記録を行うオプションが提供されています。ログ記録を有効にするには、[一般情報] ビューの新しい "MSI ログの作成" 設定で [はい] を選択します。デフォルトのログ記録パラメーターをオーバーライドするには、[プロパティ マネージャー] ビューの MsiLogging プロパティを編集します。インストールが Windows Installer 4.0 以降が搭載されているターゲット システム上で実行される場合、インストーラーはログ ファイルを作成して MsiLogFileLocation プロパティにそのパスを挿入します。さらに、SetupCompleteSuccess、SetupCompleteError および SetupInterrupted ダイアログに **[Windows Installer ログを表示]** チェック ボックスが追加されます。

詳細については、「Windows Installer インストールをログ記録するかどうかを指定する」を参照してください。

64 ビット システムにおけるレジストリ リフレクションのサポート

InstallShield では、Component テーブルにある msidbComponentAttributesDisableRegistryReflection という名前の新しい Windows Installer 4.0 属性がサポートされています。この属性を素早く構成するには、[コンポーネント] ビューで選択したコンポーネントについて "レジストリ リフレクションを無効にする" 設定を使用します。レジストリ リフレクションによって、ターゲット マシン上で 32 ビットの [レジストリ] ビューと 64 ビットの [レジストリ] ビューの同期が保たれます。この設定は、Windows Installer 4.0 がある 64 ビット システムでのみサポートされています。他のシステムでは、この設定は無視されます。詳細については、「レジストリ リフレクションの有効化と無効化」を参照してください。

複数言語ユーザー インターフェイス (MUI) のサポート

Windows Installer 4.0 によって実行される複数言語アプリケーション用のインストールを作成するとき、InstallShield を使って Windows 複数言語ユーザー インターフェイス (MUI) のサポートを含むショートカットを作成することができます。[ショートカット] ビューでは、選択されたショートカットに対して、次の 4 つの新しい設定が使用できるようになりました。

- 表示リソース DLL
- 表示リソース ID
- 説明リソース DLL
- 説明リソース ID

これらの新しい設定は、Windows Installer 4.0 の Shortcut テーブルにある新しい 4 つの列に対応しています。詳細については、「[ショートカット] ビュー」を参照してください。

Premier Edition 用の追加 InstallShield Collaboration ライセンス

InstallShield Premier Edition には、InstallShield Collaboration for Visual Studio のライセンスが 5 本含まれています。これらのライセンスを使って InstallShield はインストールされていないが、Visual Studio .NET 2002、Visual Studio .NET 2003 または Visual Studio 2005 がインストールされている開発システムに InstallShield Collaboration をインストールすることができます。

InstallScript アーキテクチャ再構築による InstallScript MSI プロジェクトおよび InstallScript カスタム アクションがある基本の MSI プロジェクトの機能強化

InstallScript MSI アーキテクチャには、いくつかのセキュリティ (COM/DCOM) とその他の箇所に関する問題があったため、様々な原因でインストールが失敗することがありました。これらの問題を解決して InstallScript MSI プロジェクトの安定性を高めるため、InstallShield 12 ではアーキテクチャが大幅に改良されました。この改良によって、InstallScript カスタム アクションを含む基本の MSI プロジェクトの安定性も高まりました。

詳しくは、次を参照してください:

- InstallScript カスタム アクションを使った、基本の MSI インストールのランタイム動作
- InstallScript MSI インストールのランタイム動作
- InstallShield 11.5 以前のプロジェクトをアップグレードする

DIFx 2.01 のサポート

InstallShield では、最新版の Driver Install Frameworks for Applications (DIFx) がサポートされています。この新しいバージョンには、マイクロソフトからの最新バイナリ ファイルが含まれているため、InstallShield で作成する基本の MSI プロジェクト、InstallScript プロジェクトまたは InstallScript MSI プロジェクトで使用することができます。

Setup.exe コマンドライン パラメーターと Update.exe コマンドライン パラメーターの強化

Setup.exe および Update.exe のコマンドライン パラメーターの一部は追加または変更されています。

- /Clone_wait (InstallScript のみ)
- /runfromtemp (基本の MSI 、InstallScript、および InstallScript MSI)
- /v"ISSCRIPTCMDLINE=¥"<Option1> <Option2>¥"" (基本の MSI のみ)
- /hide_progress (基本の MSI プロジェクト、InstallScript プロジェクト、および InstallScript MSI プロジェクト)
- /hide_splash (InstallScript および InstallScript MSI プロジェクト)

詳細については、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

COM 抽出と依存関係スキャナーにおけるレジストリおよびファイルのフィルター機能の強化

InstallShield と共にインストールされる新しい Filters.xml を編集して、COM サーバーから必要のない COM データが抽出されるのを防ぐことができます。この Filters.xml ファイルを編集することにより、COM 抽出から除外されるレジストリ キーの一覧をカスタマイズすることができます。

Filters.xml ファイルに、スタティック依存関係スキャン、ダイナミック依存関係スキャン、および Visual Basic 依存関係スキャンで含めるまたは除外するファイルがリストされます。以前は、2 つの異なるファイル (Userscan.ini と Iswiscan.ini) が除外および選択の対象になるファイルをリストするために使用されていました。

詳細については、次を参照してください。

- COM 抽出のレジストリ変更をフィルターする
- 依存関係スキャナーでファイルをフィルターする

Internet Explorer 7.0 との互換性

Internet Explorer 7.0 と互換性を持たせるために、InstallShield の一部が今回改訂されました。主な例として、トランスフォーム ウィザード、カスタム アクション ウィザード、文字列テーブル エディター、[配布] ビュー、および [一般情報] ビューが更新されました。

強化されたスタート ページ

スタート ページで表示される最近開いたプロジェクトの一覧に、プロジェクトの種類を表示する列が挿入されました。また、一覧に表示されるプロジェクトの最大数が、4 つから 8 つに増えました。

InstallScript の強化機能および新機能

InstallScript 言語が一部強化されました。新規および改訂された変数、関数および定数が今回のリリースで提供されています。

システム変数の変更およびテキスト置換の追加

DISK1SETUPEXENAME と呼ばれるシステム変数が追加されました。

2 つのシステム変数には新しいデフォルト値が割り当てられました。

- UNINST
- UNINSTALL_STRING

次の 2 つの InstallScript テキスト置換が新しく提供されました。

- DISK1SETUPEXENAME
- SELECTED_LANGUAGE

さらに詳しい情報は、「システム 変数」を参照してください。

スクリプト変数の追加

2 つのスクリプト変数が新しく提供されました。

- INSTALLSCRIPTMSI
- BASICMSI

関数および定義済み定数の変更

新しい定義済み定数 LANGUAGE_SUPPORTED は、**Is** 関数と共に使用できます。詳しくは、「Is」を参照してください。

新しい ISOSL_WINVISTA 定義済み定数は、**FeatureFilterOS** 関数と SYSINFO 構造変数と共に使用できます。詳細については、「FeatureFilterOS 関数と SYSINFO」を参照してください。

DoInstall 関数は、**LaunchAppAndWait** 関数と類似しています。詳細については、「DoInstall」を参照してください。

アップグレードの詳細

InstallShield 12 では、セキュリティ関連の問題 (COM/DCOM)、およびその他の様々な理由で起こったインストールの失敗を解決するため、InstallScript MSI アーキテクチャが大幅に改良されました。InstallShield 12 でアーキテクチャが大幅に改良された理由は、これらの問題を解決して InstallScript MSI プロジェクトの安定性を高めるためです。この改良によって、InstallScript カスタム アクションを含む基本の MSI プロジェクト、および InstallScript の安定性も高まりました。

さらにマージ モジュール プロジェクト タイプが強化され、基本の MSI プロジェクトと同じ InstallScript ビューを使用できるようになりました。この新しく強化されたマージ モジュール プロジェクト タイプが、InstallScript MSI オブジェクト プロジェクト タイプ の代わりに使用されます。InstallScript MSI オブジェクト プロジェクト タイプは InstallShield では使用できなくなりました。

詳細については、「InstallShield 11.5 以前のプロジェクトをアップグレードする」を参照してください。

ターゲット システムの要件

InstallShield を使って、Windows ベースのシステムをターゲットにするインストールを素早くビルド、テストおよび配布することができます。

デスクトップ コンピューターの要件

オペレーティング システム

ターゲット システムは、次のオペレーティング システムの最小要件を満たさなくてはなりません:

- Windows 7
- Windows Server 2008 R2
- Windows 8
- Windows Server 2012
- Windows Server 8.1
- Windows Server 2012 R2
- Windows 10
- Windows Server 2016
- Windows Server 2019
- Windows 11
- Windows Server 2022
- Windows Server 2025

ターゲット システムで、SSE2 インストラクション セットがサポートされていることが必須です。

インストーラー エンジンの要件

各インストーラー エンジンの最小ターゲット システム要件は、以下のとおりです。

テーブル 1-3 ・ デスクトップ コンピューターのターゲット システム要件

インストーラー エンジン	オペレーティング システム、その他の要件
Windows Installer 5.0 **(これは再配布可能ファイルとしては利用できません。)**	Windows 7 以降および Windows Server 2008 R2 以降
Windows Installer 4.5	Windows サーバー2003 SP1 以降
Windows Installer 4.0 **(これは再配布可能ファイルとしては利用できません。)**	Windows Server 2008

テーブル 1-3 ・ デスクトップ コンピューターのターゲット システム要件

インストーラー エンジン	オペレーティング システム、その他の要件
スイート/アドバンスト UI	Windows 7 以降および Windows Server 2008 R2 以降
アドバンスト UI	Windows 7 以降および Windows Server 2008 R2 以降
InstallScript	Windows 7 以降および Windows Server 2008 R2 以降

Windows Installer 再配布可能ファイルについての詳細は、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

64 ビット オペレーティング システムをターゲットにする

Microsoft では、既存の 32 ビットのアプリケーションがシームレスに動作する 64 バージョンの Windows がデザインされています。また、同一コードが再コンパイルされたバージョンが 64 ビット アプリケーションとしてシームレスに動作する 64 ビット バージョンの Windows もデザインされています。このサポートを提供するために、64 ビット バージョンの Windows では、主に 2 つの方法で 32 ビットと 64 ビットをアプリケーションの分離ウィザードしています。それぞれのファイルは、別々の場所 (Program Files と. Program Files (x86); System32 と. SysWow64) に格納され、レジストリ キーは、別々に分かれています (HKLM¥Software と HKLM¥Software¥Wow6432Node)。

一部のケースでは、通常有益な分離となるものも、問題になることがあります (例、インストール)。通常、インストールは、(32 ビット マシンで実行できるように) それ自身が 32 ビットのアプリケーションであるため、64 ビットの場所にアクセスして、64 ビットのアプリケーションをインストールことは、標準的なファイル コピーやレジストリの書き込みにくらべ、かなり複雑です。InstallShield で提供されている各プロジェクト タイプでは、この複雑さを異なる方法で対応しています。

Windows Installer ベースのインストールでは、ターゲットが 64 ビット システムのみのときを除き、64 ビットの場所へのアクセスを許可していません。ただし、64 ビット システムでの実行時は常に 64 ビット DLL コードの実行がサポートされています。一部のターゲット システム (たとえば、Windows Server Core システム) は、WOW64 (32-bit Windows-on-Windows) をサポートしません。InstallShield では、これらのシステム上で動作する64 ビット専用のインストールを作成できます。詳細については、「64 ビットのオペレーティング システムを基本の MSI および InstallScript MSI インストールを使ってターゲットする」を参照してください。

InstallScript のインストールは、**[リリース]** エクスプローラーの **[ビルド]** タブにある **"アーキテクチャ"** 設定に応じて、32 ビットまたは 64 ビットにできます。したがって、32 ビットまたは 64 ビット DLL を実行するように、または 32 ビットまたは 64 ビット .exe ファイルを起動するように構成できます。一部のターゲットシステムは、WOW 64 (32 – bit Windows – on – Windows) 機能をサポートしない場合があります。InstallShield では、すべてのコンポーネントを 64 ビットとして設定することで、そのようなシステムと互換性のある純粋な 64 ビット InstallScript インストールを作成できます。詳しくは、「64 ビットのオペレーティング システムを 32 ビットInstallScript セットアップ ランチャーを使ってターゲットする」を参照してください。

スイート/アドバンスト UI およびアドバンスト UI インストールでは、32 ビットおよび 64 ビットのいずれの場所も参照することができ、ターゲット システムに応じて、32 ビットおよび 64 ビットのいずれのインストールも起動することができます。ただし、スイート/アドバンスト UI またはアドバンスト UI プロジェクトのすべての拡張 DLL コードは、32 ビットである必要があります。詳細については、「64 ビット オペレーティング システムをスイート/アドバンスト UI およびアドバンスト UI インストールでターゲットにする」を参照してください。

64 ビットのオペレーティング システムを基本の MSI および InstallScript MSI インストールを使ってターゲットする



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

InstallScript MSI プロジェクトを作成してから、"テンプレートの概要" プロパティを Intel から x64 に変更すると、ビルド時に 64 ビット MSI インストールと 32 ビット InstallScript MSI インストールの組み合わせが作成されます。

InstallScript MSI プロジェクトでは、アーキテクチャの検証がサポートされていないので注意してください。

64 ビット Windows ベースのシステム用のアプリケーションを構築する場合、64 ビット ファイルおよびその他のデータをインストールする手段が必要です。32 ビットの Windows Installer ベースのインストールを 64 ビットのマシンで実行することは可能ですが、64 ビットの場所にインストールすることはできません。Windows Installer サービスは、64 ビットのインストールをサポートしています。これらのセットアップは 32 ビットマシン上で設計およびビルドできますが、64 ビットのマシン上でしか実行できません。リリース フラグを使用して、単一のプロジェクトから 2 つのインストール (32 ビットと 64 ビット) をビルドすることができます。

"テンプレート概要" プロパティの構成

"テンプレート概要" プロパティでは、.msi パッケージが 32 ビット パッケージであるか 64 ビット パッケージであるかを指定します。

インストールが 64 ビット システムをターゲットにする場合、適切な 64 ビット値 (x64 または Intel64) を使用して "テンプレート概要" プロパティを構成します。この構成により、エンドユーザーが 64 ビット システムの 64 ビットの場所に貴社の製品をインストールできるようになります。また、エンドユーザーが貴社の製品を 32 ビット システムにインストールすることを防ぐこともできます。

[一般情報] ビューにある "テンプレートの概要" 設定を使って、"テンプレートの概要" プロパティを構成できます。[リリース] ビューで、製品構成について "テンプレートの概要" 設定を構成することもできます。[リリース] ビューに入力された値は、[一般情報] ビューに設定された値を上書きします。

詳細については、「"テンプレート概要" プロパティを使用する」を参照してください。

64 ビット コンポーネントの作成

コンポーネントが 64 ビットであると指定するには、コンポーネントの "64 ビット コンポーネント" 設定で [はい] を選択します。1 以上の 64 ビット コンポーネントを含み、"テンプレートの概要" プロパティが 64 ビット値に設定されたリリースをビルドすると、その結果、64 ビットのパッケージが作成されます。

"64 ビット コンポーネント" 設定、および追加のコンポーネントの設定に関する詳細については、「コンポーネントの設定」を参照してください。

COM データを 64 ビット COM サーバーから抽出する

InstallShield を 64 ビット オペレーティング システムで使用している場合、InstallShield は 64 ビット COM サーバーから COM データを抽出できます。データを正しい場所にインストールするため、コンポーネントを 64 ビットとマークしなくてはなりません。

InstallShield を 32 ビット システム上で使用している場合は、64 ビット COM 抽出機能は使用できません。

COM データの抽出についての詳細は、「COM 情報を COM サーバーから抽出する」を参照してください。

64 ビット COM サーバーの自己登録

InstallShield では、COM サーバーの 64 ビット自己登録がサポートされています。コンポーネントを 64 ビットとしてマークしてから、そのコンポーネントにファイルを追加したとき、そのファイルの [自己登録] チェック ボックスを選択して、インストール時の 64 ビット自己登録を有効にすることができます。[自己登録] チェック ボックスは、ファイルの [プロパティ] ダイアログ ボックスにあります。

InstallShield では、動的にリンクされている COM サーバーの 64 ビット自己登録もサポートされています。これを有効にするには、ファイルを動的に 64 ビット コンポーネントに追加する際、[ダイナミック ファイル リンクの設定] ダイアログ ボックスにある **[すべてのファイルを自己登録する]** チェック ボックスを選択します。詳細については、「ダイナミック ファイル リンクをコンポーネントに追加する」を参照してください。

レジストリ リフレクションの有効化と無効化

InstallShield では、Component テーブルに Windows Installer 4.0 属性 msidbComponentAttributesDisableRegistryReflection が含まれています。この属性を使用するには、[コンポーネント] ビューで選択したコンポーネントの "レジストリ リフレクションを無効にする" 設定を [はい] に設定します。レジストリ リフレクションによって、ターゲット マシン上で 32 ビットの [レジストリ] ビューと 64 ビットの [レジストリ] ビューの同期が保たれます。この設定は、Windows Installer 4.0 以降がある 64 ビット システムでのみサポートされています。他のシステムでは、この設定は無視されます。詳細については、「レジストリ リフレクションの有効化と無効化」を参照してください。

他の 64 ビット要素を追加する

InstallShield では、追加の 64 ビット要素もサポートされています。たとえば、InstallShield には、プロジェクトに追加することができる 64 ビットの .NET Framework 再配布可能ファイルが含まれています。カスタム アクションには 64 ビットの Jscript または VBScript コードを含めることができます。64 ビット フォルダー プロパティ からは、64 ビット Program Files フォルダーなどの重要なオペレーティング システム フォルダーにアクセスすることができます。さらにセットアップ前提条件エディターを使って、ターゲット システム上の 64 ビット要件をチェックする条件を持つセットアップ前提条件を構成することができます。



***メモ・** 64 ビット サポートには、Windows Installer バージョン 2.0 以上が必要です。*

適切なアーキテクチャの検証を指定する

基本の MSI プロジェクトには、ビルド時にアーキテクチャの検証を使用するかどうかを指定できる製品構成設定が含まれています。アーキテクチャの検証を使って、インストールがターゲット システムのアーキテクチャと一致しない製品ファイルをインストールしようとしたり、ランタイム バイナリを使用したりする問題を引き起こす可能性のある状況を検出することができます。

たとえば、エンド ユーザーが WOW64 (32-bit Windows-on-Windows) サポートが搭載されていない 64 ビット Windows Server Core システムでインストールを実行する場合、アーキテクチャの検証を使って、インストールに含まれるすべての 32 ビット の製品ファイルおよび 32 ビットのカスタム アクション ファイルを識別できます。32 ビット バイナリを WOW64 サポートを搭載しない 64 ビット ターゲット システムにロードすることはできません。

アーキテクチャの検証を使って、ビルトイン InstallShield カスタム アクション DLL の 32 ビット バージョンのみを含む 32 ビット専用 .msi パッケージ、またはビルトイン InstallShield カスタム アクション DLL の 64 ビット バージョンのみを含む 64 ビット専用 .msi パッケージを生成することができます。

ビルド時に使用するアーキテクチャの検証の種類を指定し、32 ビット専用パッケージ、または 64 ビット パッケージのどちらをビルドするのかを識別するには、[リリース] ビューの製品構成で "アーキテクチャの検証" 設定を使用します。

詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。

64 ビット セットアップ ランチャーで InstallScript MSI をサポート

InstallScript MSI プロジェクトは、32 ビットと 64 ビットの両方のWindows オペレーティング システム用にビルドできるだけでなく、64 ビットの Windows オペレーティング システム専用にビルドすることができるようになりました。**[アーキテクチャ]** 設定を構成すると、**[メディア]** ビューの **[リリース]** エクスプローラーにあるリリースの **[ビルド]** タブで、アプリケーションを実行するために必要なターゲット システムに基づいて、setup.exe ファイル (セットアップランチャー) を作成できます。詳細については、「リリースの [ビルド] タブ」を参照してください。

単一プロジェクトから 2 つのインストール (32 ビットと 64ビット) をビルドするためのヒント

単一プロジェクトから 2 つのインストール (32 ビットおよび 64 ビット) をビルドできるようにするためには、リリース フラグの使用を考慮してください。必要に応じてリリース フラグを機能、InstallShield 前提条件、および連鎖 .msi パッケージに割り当てて、32 ビットのアイテムと 64 ビットのアイテムとを区別します。次に、[リリース] ビューの各リリースまたは製品構成において、ビルド時に特定のリリース フラグを持つアイテムを含めるか除外するかを構成します。リリース フラグに基づいて特定のカスタム アクションを条件付きで起動することによって、32 ビット インストール中に 32 ビット カスタム アクションを起動し、64 ビット インストール中に 64 ビット カスタム アクションを起動することもできます。

詳しくは、次を参照してください:

- リリース フラグ
- リリース フラグに基づくフィルター
- カスタム アクションをリリース フラグに基づいて条件付きで起動する

InstallShield では、プロジェクト内の各リリースでプロジェクトのパス変数の値をオーバーライドすることができます。この機能を使って、ビルドする特定のリリースごとに、ビルド時にプロジェクト内の特定のファイルとフォルダーを別のファイルとフォルダーに置換することができます。

たとえば、この機能を使ってカスタム アクションのバイナリ ファイルを入れ替えることができます。個別の 32 ビットと 64 ビットのターゲット システム用に個別のリリースを作成した場合、カスタム アクションに選択された DLL を参照するパス変数をオーバーライドすることができます。これによって、InstallShield が 32 ビット リリースには 32 ビット DLL を、また 64 ビット リリースには 64 ビット DLL を含みます。インストールがインストール処理を行っているファイルを入れ替えるのにパス変数のオーバーライドを行うことはお勧めできません。これは、ファイルの 32 ビット バージョンと 64 ビット バージョンに個別のコンポーネントを使用しなくてはならないためです。

プロジェクトで 1 つ以上のパス変数をオーバーライドする場合、[リリース] ビュー内のリリースの [ビルド] タブに追加された "パス変数のオーバーライド" 設定を使います。詳細については、「リリースの [ビルド] タブ」を参照してください。

Using 64-Bit Windows Installer Packages (Windows Installer ヘルプ ライブラリ)

64 ビット セットアップ ランチャーのビルド

64 ビットのオペレーティング システムを 32 ビットInstallScript セット アップ ランチャーを使ってターゲットする



***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

1 つの InstallScript プロジェクトから、32 ビット システムの 32 ビットの場所、および、必要に応じて、64 ビット システムの 64 ビットと32 ビットの場所にインストールされるインストールを 1 つ作成することができます。

InstallScript インストールが 32 ビットの場合、32 ビットの DLL コードのみ実行 (または、32 ビットまたは 32 ビット .exe ファイルを起動) します。ただし、すべての InstallScript インストールでは、64 ビット コンポーネントの設定およびその他の方法を通じて、64 ビットの場所へのアクセスがサポートされています。

64 ビット System32 フォルダーの読み取りと書き込み

InstallScript には、ターゲット システム上の 32 ビットの System32 フォルダーと 64 ビットの System32 フォルダーのいずれかを選択できる次のシステム変数が使われています:

- WINSYSDIR
- WINSYSDIR64

WINSYSDIR64 変数は 64 ビット Windows フォルダーに設定されていますが、64 ビット Windows には、自動的に 32 ビットアプリケーション (InstallScript エンジンなど) を 32 ビット System32 フォルダー (SysWOW64) にリダイレクトする機能が含まれています。従って、WINSYSDIR64 にファイルとフォルダーをインストールする場合、64 ビットとマークされているコンポーネント内にフォルダーまたはファイルを配置して、ファイル システムのリダイレクトを無効にできます。コンポーネントが 64 ビットであると指定するには、コンポーネントの "64 ビット コンポーネント" 設定で [はい] を選択します。"64 ビット コンポーネント" 設定、および追加のコンポーネントの設定に関する詳細については、「コンポーネントの設定」を参照してください。

代わりに、定数 WOW64FSREDIRECTION を Disable と Enable いう 2 つの関数を使って、InstallScript コードから WINSYSDIR64 フォルダーに書き込みまたは読み取りを実行している間に 64 ビット Windows のファイル システムのリダイレクトを無効にして、書き込み/読み取りが完了したら、再度有効にすることができます。インストールが利用する可能性のある Windows 機能にはファイル システム リダイレクトを有効にしておく必要があるため、Windows ドキュメンテーションではリダイレクトを無効にするのはそれが必要な場合のみにとどめることが推奨されています。

その他の 64 ビットのインストール先フォルダーの読み取りと書き込み

InstallScript には、ターゲット システム上の 64 ビットの場所をターゲットにするための、いくつかの追加システム変数が提供されています:

- COMMONFILES64
- FOLDER_APPLICATIONS64
- PROGRAMFILES64

64 ビットのレジストリの場所の読み取りと書き込み

64 ビット バージョンの Windows には、一部の 64 ビットのレジストリの場所を対応する 32 ビットの場所にマップする機能が含まれています。たとえば、64 ビット バージョンの Windows では、HKEY_LOCAL_MACHINE¥Software¥Wow6432Node は、32 ビットの HKEY_LOCAL_MACHINE¥Software に相当します。

レジストリ データを、32 ビット領域にリダイレクトせずに、64 ビット領域のレジストリにインストールする場合、レジストリ データを 64 ビットとしてマークされたコンポーネントに配置します。コンポーネントが 64 ビットであると指定するには、コンポーネントの "64 ビット コンポーネント" 設定で [はい] を選択します。"64 ビット コンポーネント" 設定、および追加のコンポーネントの設定に関する詳細については、「コンポーネントの設定」を参照してください。

代わりに、REGDB_OPTION_WOW64_64KEY オプションを REGDB_OPTIONS 変数と一緒に使って、InstallScript コードでレジストリの編集または読み取りを実行中に、レジストリのリダイレクトを回避し、REGDB_OPTION_USE_DEFAULT_OPTIONS オプションを REGDB_OPTIONS と一緒に使って、32 ビット領域のレジストリへの変更を有効にすることもできます。インストールが利用する可能性のある Windows 機能にはファイル システム リダイレクトを有効にしておく必要があるため、Windows ドキュメンテーションではリダイレクトを無効にするのはそれが必要な場合のみにとどめることが推奨されています。

自己登録型の 64 ビット COM サーバーのインストール

InstallShield では、COM サーバーの 64 ビット自己登録がサポートされています。自己登録型の DLL、.exe、.tlb、または .olb ファイルを含むコンポーネントの場合、コンポーネントの "自己登録" 設定で [はい] を選択します。

自己登録に対してファイル システムのリダイレクトは無効にできないため、32 ビットの自己登録型ファイルは 64 ビットの System32 フォルダーにはインストールできません。このため、32 ビットの System32 フォルダーにインストールする必要がある 32 ビットの自己登録型 COM サーバーがある場合、COM サーバーを含むコンポーネントの "64 ビットのコンポーネント" 設定で [いいえ] が設定されていることを確認してください。

[プログラムの追加と削除] 情報について

InstallScript インストールは、レジストリの 64 ビット部分にある [プログラムの追加と削除] 情報の読み書きをサポートしません。したがって、InstallScript インストールは、レジストリの 32 ビット部分に [プログラムの追加と削除] 情報をインストールします。また、REGDB_OPTION_WOW64_64KEY オプションを使って、**CreateInstallationInfo**、**MaintenanceStart**、**RegDBGetItem**、**RegDBSetItem**、**RegDBGetAppInfo**、**RegDBSetAppInfo**、および **RegGetUninstCmdLine** などのレジストリ関数に対して、レジストリ リフレクションが無効にすることはできません。

64 ビット オペレーティング システムをスイート/アドバンスト UI およびアドバンスト UI インストールでターゲットにする



***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



***エディション・**アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

***メモ・** デフォルトで、スイート/アドバンスト UI およびアドバンスト UI インストールは 32 ビットです。必要な場合、64 ビット インストールのリリース構成の下にある ""64 ビット セットアップ ランチャー" 設定を変更することができます。64 ビット セットアップ ランチャーについては、「64 ビット セットアップ EXE のビルド」を参照して下さい。*

スイート/アドバンスト UI およびアドバンスト UI インストールでは、32 ビットおよび 64 ビットのいずれの場所も参照することができ、ターゲット システムに応じて、32 ビットおよび 64 ビットのいずれのインストールも起動することができます。

たとえば、スイート/アドバンスト UI プロジェクトに、32 ビット システムをターゲットするインストールを 1 つと、64 ビット システムをターゲットするインストールを 1 つ含める場合、これら両方のパッケージに対して、[パッケージ] ビューで対象条件を作成できます。32 ビット パッケージの場合、対象条件には、x86 アーキテクチャの有無を確認するプロファイル条件が含まれ、64 ビット パッケージには、x64 または IA64 アーキテクチャの有無を確認するプラットフォーム条件が含まれます。詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類」を参照してください。

32 ビットのオペレーティング システムを 64 ビット InstallScript セットアップ ランチャーを使ってターゲットする

***プロジェクト・** この情報は、InstallScript プロジェクトに適用します。*

InstallShield では、64 ビット セットアップ ランチャーと 64 ビットInstallScript エンジン ファイルを使用して、64 ビット Windows オペレーティング システムを対象とした InstallScript プロジェクトを構築することができます。64 ビット システムでインストールを起動すると、64 ビットファイル システムとレジストリの場所をターゲットにして、完全に 64 ビット環境内で実行されます。

InstallScript ベースのプロジェクトの 64 ビット アーキテクチャ オプションでは、純粋な 64 ビット パッケージをビルドできますが、64 ビット プラットフォーム上の 32 ビットファイルシステムの場所と 32 ビット レジストリ ビューへのアクセスもサポートしています。必要に応じて、適切な REGDB_OPTIONS オプションとシステム変数を使用してこれを達成できます。

その他の 32 ビットのインストール先フォルダーの読み取りと書き込み

InstallScript には、64 ビット ターゲット システム上の 32 ビットの場所をターゲットにするための、いくつかのシステム変数が提供されています:

- COMMONFILES
- FOLDER_APPLICATIONS
- FOLDER_APPLICATIONS32
- PROGRAMFILES

64 ビット プラットフォーム上の 32 ビットのレジストリの場所の読み取りと書き込み

Windows の 64 ビット バージョンでは、64 ビット アプリケーション (64 ビット InstallScript エンジンなど) は、デフォルトで適切な 64 ビットレジストリビューにアクセスします。レジストリ データをレジストリの 64 ビット領域ではなく 32 ビット領域に明示的にインストールしたい場合は、32 ビットとマークされたコンポーネントにレジストリ データを配置することができます。コンポーネントが 32 ビットであると指定するには、**[コンポーネント]** ビューでコンポーネントの **"64 ビット コンポーネント"** 設定で **[いいえ]** を選択します。"64 ビット/32 ビットコンポーネント" 設定、および追加のコンポーネントの設定に関する詳細については、「コンポーネントの設定」を参照してください:

別の方法として、REGDB_OPTION_WOW 64_32 KEY オプションをREGDB_OPTIONS 変数とともに使用すると、システム アーキテクチャに関係なく、すべてのレジストリ操作をレジストリの 32 ビット部分に転送できます。

64 ビット プラットフォームに自己登録型の 32 ビット COM サーバーをインストール

InstallShieldは、64 ビット Windows オペレーティング システムに 32 ビット COM サーバーをインストールして自己登録するためのサポートを提供します。自己登録 COM サーバーは、通常 **DllRegisterServer** を呼び出すことによって、インストール中に Windows レジストリに自己登録を行うロジックを含む DLL または EXE です。32 ビット COM サーバーを 64 ビット プラットフォームにインストールする場合、環境の 32 ビットの場所に正しくインストールされ、登録されていることを確認することが重要です。

64 ビットマシンの SysWOW 64 フォルダーに 32 ビットの自己登録型 COM サーバーをインストールするには、COM サーバーを含む **[コンポーネント]** ビューでコンポーネントの **"64 ビット コンポーネント"** 設定に **[いいえ]** が選択されていることを確認します。

InstallShield を管理者権限を使って、または管理者権限を持たずに起動する違い

管理者権限を持たずに InstallShield を起動した場合、以下の機能は使用できなせん:

- **COM 抽出**—COM サーバーからの COM 情報の抽出には、管理者権限が必要です。

  InstallShield を管理者権限を持たずに実行している場合に、プロジェクトで COM サーバーから COM 情報の抽出を行うように指定してからリリースをビルドしようとすると、ビルド エラー -6017 が発生します。

- **再配布可能ファイルのダウンロード**—[再配布可能ファイル] ビュー内から再配布可能ファイルをダウンロードするには、管理者権限が必要です。これは、InstallShield がファイルをマシンごとの場所にダウンロードする際に管理者権限が必要なためです。

  [再配布可能ファイル] ビュー内から再配布可能ファイルをダウンロードしようとしたとき、管理者権限が無い場合には次のメッセージが表示されます:

  ダウンロードが失敗しました。管理者として実行していること、および使用中のマシンがインターネットに接続されていることを確認してください。再試行しますか?

- **InstallShield 前提条件に [すべてのユーザー] の場所を指定できる機能** — [オプション] ダイアログ ボックスの [前提条件] タブを使って、[再配布可能ファイル] ビューおよび [前提条件] ビューに表示する InstallShield 前提条件を含むフォルダーを指定できます。このタブで [すべてのユーザー] の場所を変更するためには管理者権限が必要です。これは、InstallShield がレジストリ内でマシンごとの場所に情報を書き込むためです。したがって、InstallShield を管理者権限を持たずに実行している場合、このタブの [すべてのユーザー] の場所は無効となっています。

- **マージ モジュールに [すべてのユーザー] の場所を指定できる機能** – [オプション] ダイアログ ボックスの [マージ モジュール] タブを使って、[再配布可能ファイル] ビューに表示するマージ モジュールを含むフォルダーを指定できます。このタブで [すべてのユーザー] の場所を変更するためには管理者権限が必要です。これは、InstallShield がレジストリ内でマシンごとの場所に情報を書き込むためです。したがって、InstallShield を管理者権限を持たずに実行している場合、このタブの [すべてのユーザー] の場所は無効となっています。
- **Regasm.exe および InstallUtilLib.dll の場所を編集できる機能** – [オプション] ダイアログ ボックスの [.NET] タブで、.NET Framework に含まれているユーティリティである Regasm.exe および InstallUtilLib.dll ファイルの場所を指定できます。これらのユーティリティは COM interop と .NET カスタム アクションで利用されます。[.NET] タブでこれらの場所を変更するためには管理者権限が必要です。これは、InstallShield がレジストリ内でマシンごとの場所に情報を書き込むためです。したがって、InstallShield を管理者権限を持たずに実行している場合、このタブの場所設定は無効となっています。

管理者コンテキストと非管理者コンテキストの切り替えを行い、プロジェクトでマップされたドライブの場所を使用している場合、問題が発生することがあります。たとえば、管理者権限を持たずに Windows Explorer を使ってドライブ名を共有ネットワーク フォルダーにマップした場合、InstallShield の非管理者インスタンスではこのドライブにアクセスできますが、管理者インスタンスではアクセスできません。同様に、管理者権限を持っている場合に Windows Explorer を使ってドライブ名を共有ネットワーク フォルダーにマップすると、InstallShield の管理者インスタンスではこの場所にアクセスできますが、非管理者インスタンスではアクセスできません。このため、プロジェクトでネットワークの場所を参照する場合は、UNC パス (たとえば、¥¥server¥share) を使用するか、管理者と非管理者の両方でドライブ名をマッピングすることをお勧めします。

Visual Studio 内から InstallShield を使用している場合、管理者権限を持たない可能性があります。デフォルトでは、Windows Vista 以降のシステム上で Visual Studio のショートカットをダブルクリックして起動した場合、管理者権限はありません。

タスク

Windows Vista 以降のシステム上で管理者権限を使って Visual Studio 内部から InstallShield を実行するには、以下の手順に従います:

1. [スタート] メニューで Visual Studio のショートカットを右クリックしてから、**[管理者として実行]** をクリックします。
2. 新しい InstallShield プロジェクトを作成するか、または既存のプロジェクトを開きます。詳細は、以下のどちらかを参照してください:
   - Microsoft Visual Studio で InstallShield プロジェクトを作成する
   - Microsoft Visual Studio で InstallShield プロジェクトを開く

32 ビットと 64 ビット システムにおけるインストールの開発およびビルドの違い

InstallShield は、32 ビットと 64 ビット システムの両方で実行できる 32 ビット アプリケーションです。場合によって、InstallShield は、32 ビット システムで使用されているか、または 64 ビット システムで使用されているかによって動作が異なります。また、使用しているオペレーティング システムによっても、違いが見られることがあります。次のセクションは、これらの違いを説明します。

InstallShield 同様、コマンドライン ビルド (ISCmdBld.exe)、Standalone Build、およびオートメーション インターフェイスも、32 ビット アプリケーションです。したがって、同じプラットフォーム特有の違いは、これらのツールでも発生します。

オートメーション インターフェイスのロード

オートメーション インターフェイスは 32 ビット インターフェイスを持つため、32 ビット プロセスからロードしなくてはなりません。オートメーション インターフェイスを 64 ビット マシンで使用する場合、これを 32 ビット実行可能ファイルを使ってロードする必要があります。そうしないと、エラーが発生する可能性があります。

詳細については、「64 ビット システム上でオートメーション インターフェイスを使用する」を参照してください。

プロジェクトへシステム ファイルを追加する

64 ビット システム上で InstallShield を使用している場合に、ソースの場所が開発マシン上の 64 ビット システム フォルダー (System32) であるシステム ファイルをプロジェクトに追加するとき、[ファイルとフォルダー] ビューの上部にあるソース コンピューターのペインからインストール先コンピューターのペインの適切な場所にドラッグすることはできません。

開発マシン上の 64 ビット System32 ファイルを InstallShield プロジェクトに追加するには、マシン上の Sysnative フォルダーを参照してから、プロジェクトに適切なファイルを選択します。詳細については、「64 ビット ソース マシンの 64 ビット System32 フォルダーからファイルを追加する」を参照してください。

64 ビット開発システム上で、ソースマシンのレジストリの 32 ビットおよび 64 ビット領域の両方を表示する

InstallShield を 64 ビット開発システム上で使用する場合、InstallShield が表示する [レジストリ] ビューは、使用中のマシンのレジストリの 32 ビットおよび 64 ビット領域の両方を表示します:

- HKEY_LOCAL_MACHINE¥Software
- HKEY_LOCAL_MACHINE¥Software¥Wow6432Node

このサポートによって、プロジェクトのレジストリ データの変更を構成する際、これらのソース領域からこのビューのインストール先ペインの適切な領域にエントリーをドラッグ アンド ドロップすることが可能となります。

インストールがレジストリ データを、32 ビット領域にリダイレクトせずに、64 ビット ターゲット システム上のレジストリの 64 ビット領域にインストールする場合、レジストリ データを 64 ビットとしてマークされたコンポーネントに配置します。[レジストリ] ビューのソースペインから 64 ビット データを、ビュー内のインストール先ペインにドラッグするだけでは、そのコンポーネントが 64 ビットであるとマークされません。詳細は、次を参照してください:

- 64 ビット オペレーティング システムをターゲットにする
- レジストリ エントリをドラッグアンドドロップしてレジストリ キーを作成する

COM データを 64 ビット COM サーバーから抽出する

デザイン タイムまたはビルド時に、64 ビット COM サーバーから COM データを抽出する場合、64 ビット オペレーティング システム上で InstallShield を使用する必要があります。InstallShield を 32 ビット システム上で使用している場合は、64 ビット COM 抽出機能は使用できません。

COM データの抽出についての詳細は、「ビルド時に COM 登録データを抽出する」を参照してください。

64 ビット ファイルをスキャンして依存関係を確認する

InstallShield で依存関係スキャナーを使って、プロジェクトに追加する必要がある可能性がある依存関係を判別する場合、64 ビット ファイルの依存関係の確認は、InstallShield を 64 ビット オペレーティング システムでを使用している場合のみ可能です。

32 ビット システムで InstallShield を使用している場合、InstallShield の依存関係スキャナーを使って、プロジェクトの 64 ビット ファイルの依存関係を確認することはできません。32 ビット システムで 64 ビットの依存関係をプロジェクトに追加する場合、必須ファイルおよびマージ モジュールをプロジェクトに手動で追加する方法があります。

ファイルの依存関係のスキャンに関する詳細は、「アプリケーションの依存関係を識別する」を参照してください。

依存関係の 64 ビット .NET アセンブリとプロパティをスキャンする

Windows Vista 以降の 64 ビット バージョン、または Windows Server 2008 以降の 64 ビット バージョンで InstallShield を使用していて、いずれかのビルトイン処理によって依存関係を検出する場合、InstallShield はプロジェクトに含まれる 64 ビット .NET アセンブリの 64 ビット依存関係をスキャンすることができます。

これらの処理は、プロジェクトに含まれる 32 ビット .NET アセンブリの 32 ビット依存関係のスキャンも行います。

Windows の 32 ビット バージョンで InstallShield を使用する場合、これらのビルトイン スキャンはプロジェクトに含まれる 32 ビット ファイルの 32 ビット依存関係のみを検出できます。プロジェクトに 64 ビット ファイルが含まれている場合、必要に応じてプロジェクトに依存関係を手動で追加することができます。

詳しくは、次を参照してください:

- .NET アセンブリのプロパティおよび依存関係を識別する
- 依存関係の 64 ビット .NET アセンブリをスキャンする

64 ビット .NET Installer クラスと COM Interop を使用する

Windows の 64 ビット バージョンで InstallShield を使用している場合、InstallShield の [ツール] メニューにある [オプション] をクリックすると表示される [オプション] ボックスでは、今回より、.NET Framework に含まれている Regasm.exe および InstallUtilLib.dll ファイルの場所として、32 ビット と 64 ビットの 2 つのパスを指定できます。InstallShield は、.NET Installer クラスと COM Interop を含むリリースで、ビルド時に指定されたパスを使用します。

64 ビット バージョンの Windows 上で ISCmdBld.exe を使ってコマンドラインからビルドしている場合に、既存の -t パラメーターを使って .NET Framework の 32 ビット バージョンのパスを指定すると、ISCmdBld.exe は、64 ビット .NET Installer クラスと COM Interop 用に Regasm.exe および InstallUtilLib.dll の 64 ビットの場所を使用します。

MSBuild または Team Foundation Server (TFS) を使ってビルドしている場合に、InstallShield タスクで既存の DotNetUtilPath パラメーターを使って .NET Framework の 32 ビット バージョンのパスを指定すると、ビルドは、64 ビット .NET Installer クラスと COM Interop 用に Regasm.exe および InstallUtilLib.dll の 64 ビットの場所を使用します。

32 ビット オペレーティング システムで InstallShield を使用している場合、64 ビットの場所の設定は無効になり、64 ビット .NET Installer クラスと COM Interop の 32 ビット サポートのみ有効になります。このサポートは、ISCmdBld.exe を使ってコマンドラインから実行するビルド、および、MSBuild または TFS を介して実行するビルドにも適用します。

詳しくは、次を参照してください:

- [.NET] タブ ([オプション] ダイアログ ボックス)
- ISCmdBld.exe
- Microsoft ビルド エンジン (MSBuild)

ヘルプの使い方

Revenera は、役に立つ情報やヘルプ リソースがいつでもすぐ取り出せることがいかに重要かを理解しています。InstallShield インターフェイスの様々なビュー内部に埋め込まれているインライン ヘルプとは別に、InstallShield には製品と共にインストールされるヘルプ ライブラリ (Web で使用できるオンライン ヘルプ システム、および PDF フォーマットで提供されるドキュメント) が含まれています。

InstallShield ヘルプ ライブラリ

製品に関してご不明な点がありましたら、まず InstallShield ヘルプ ライブラリを参照してください。これは総合的なユーザー ガイドです。

InstallShield の **[ヘルプ]** メニューから InstallShield ヘルプ ライブラリにアクセスするには、F1 を押すか、インターフェイスの [ヘルプ] ボタンをクリックします。

InstallShield ヘルプ ライブラリの利用に、インターネットの接続は必要ありません。オンライン ヘルプ ビューアーは、基本的に、個人のニーズに基づいてテクニカル情報を表示、検索、フィルターするためのツールです。

コンテキスト ヘルプを使用する

プロジェクトの作業中に、ソフトウェア オブジェクトをクリックすると、[ヘルプ] ウィンドウにヘルプ情報が表示されます。これは、「コンテキスト」ヘルプとも呼ばれます。

ヘルプ情報は、エクスプローラー ウィンドウで表示される選択されたソフトウェア オブジェクトの各プロパティでも参照することができ、プロパティの設定方法を見ることができます。

Web ベースのオンライン ヘルプ

Web ベースのオンライン ヘルプは、24 時間いつでも、http://docs.revenera.com から利用することができます。このヘルプ リソース センターより、最新の情報がほぼリアル タイムに入手可能です。

PDF 版のドキュメント

InstallShield ドキュメントの PDF 版は https://community.revenera.com/s/ からご利用いただけます。

docs.revenera.com でプロジェクトの種類ごとに検索結果を絞り込むことができる機能

InstallShield 2024 から、docs.revenera.com の InstallShield Help Library (英語版) を参照する時、プロジェクトの種類ごとに検索結果を絞り込むことができます。

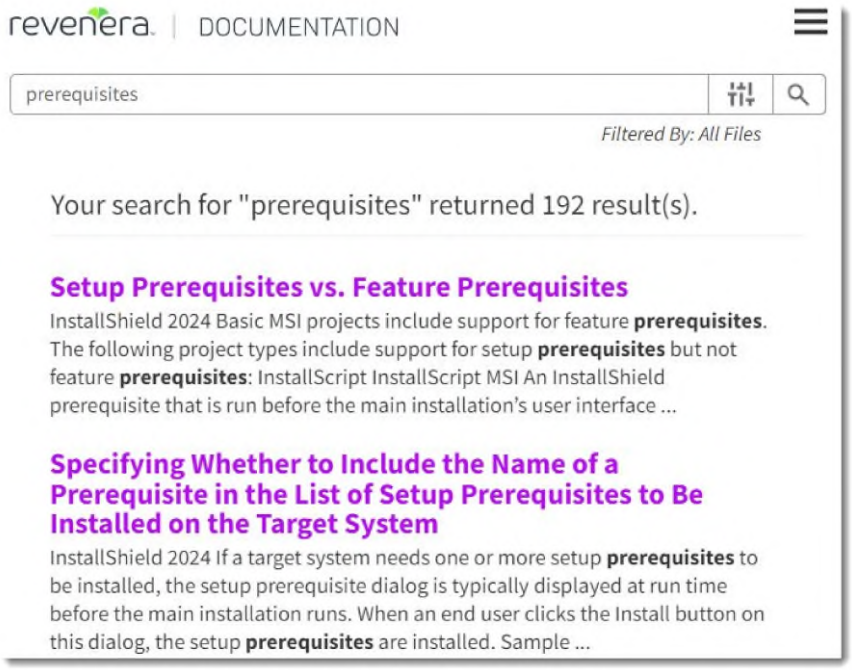
ブラウザー ウィンドウの右上にある**検索**バーに**フィルター** アイコンが表示され、選択したフィルターがアイコンの下に表示されます。



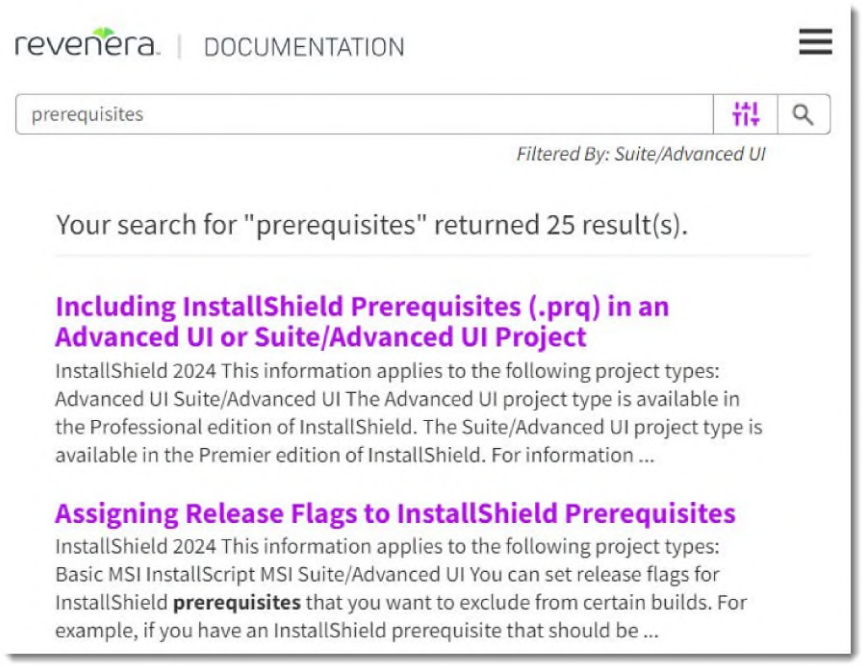
Filtered By: All Files

タスク ***検索結果を絞り込むには、次の手順に従います:***

1. **検索**ボックスに検索語を入力し、虫メガネをクリックするか、エンター キーを押します。検索結果が表示され、結果の総数が一覧表示されます(この例では**192**件):



2. **フィルター**アイコンをクリックし、**[スイート/アドバンスト UI]** など、検索結果を絞り込みたい**プロジェクトの種類**を選択します。絞り込まれた検索結果が表示され、選択した**プロジェクト タイプ** に適用するように指定されたトピックのみが一覧表示されます (この例では **25** 件):



***メモ・** InstallShield ヘルプ ライブラリのプロジェクトの種類で絞り込めるトピックには、そのトピックがどのプロジェクトの種類に適用されるかを示す**プロジェクト** メモが上部に表示されます。*



お問い合わせ先

Revenera は本社をイリノイ州イタスカに置き、世界各地に拠点を持ちます。製品についてのお問い合わせなど、連絡先は当社の Web サイトをご覧ください:

http://www.revenera.com

お問い合わせ先

2

スタート ガイド

InstallShield では、安定した Windows Installer および InstallScript インストールを簡単に作成することができる強力な機能と、作業時間を大幅に節約することができるツールが多数提供されています。InstallShield ヘルプ ライブラリは、InstallShield で提供されている機能をより効果的に活用するためのリソースです。まず最初にどのトピックから読み始めるかは、InstallShield インストール作成ソフトウェアの利用経験によって異なります。下のテーブルは、それぞれの経験レベルに応じて様々なトピックにリンクしています。

テーブル 1 - スタート ガイド ロード マップ

経験レベル	トップ ヘルプ トピック
インストールを以前に作成した経験なし。	インストールに関する一般的な情報については、次のトピックを参照してください。 • インストールの基本 • アプリケーション ライフサイクル
InstallShield を利用するのは初めて。	インストール作成経験はあるが、InstallShield を利用するのは初めての場合、次のトピックを参照してください。 • InstallShield インターフェイスを使って作業する • プロジェクトについて • 基本の MSI プロジェクト チュートリアル • グローバリゼーション チュートリアル • InstallScript プロジェクト チュートリアル • InstallShield 実行時の言語サポート • サポートされているアプリケーション プログラミング言語

テーブル 1・スタート ガイド ロード マップ (続き.)

経験レベル	トップ ヘルプ トピック
他の Revenera 製品を利用した経験がある。	Revenera の他の製品を利用したことがある場合、次のトピックを参照してください。 • 以前のバージョンの InstallShield からアップグレードする • InstallShield の他のエディションからアップグレードする • プロジェクト アシスタント • インストール プロジェクトの概要
InstallShield の中級、または上級レベルユーザー。	InstallShield を使い慣れている場合、次のトピックの参照をお勧めします: • コマンドライン ツール • エラーと警告 • InstallScript 言語リファレンス • サイレント モードでインストールを実行する • InstallShield の詳細設定を構成する

インストールの基本

インストールとは、一言で説明すると、ファイルおよびプログラムをユーザーのマシンにインストールするために使われる「パッケージ」を意味します。インストールは、インストーラー エンジンと対話するロジック、並びにアプリケーション ファイルで構成される集合体です。インストールの最も基本的な役割は、アプリケーション ファイルをソース媒体からエンド ユーザーのコンピューターへ転送することです。Windows オペレーティング システムは大変複雑なため、InstallShield のようなユーティリティのヘルプなしに、効果的で首尾一貫したインストールを簡単に作成することはできません。

インストールは、「製品」、「機能」、「コンポーネント」の 3 つの階層に分けられます。次の図はこの階層を説明します。

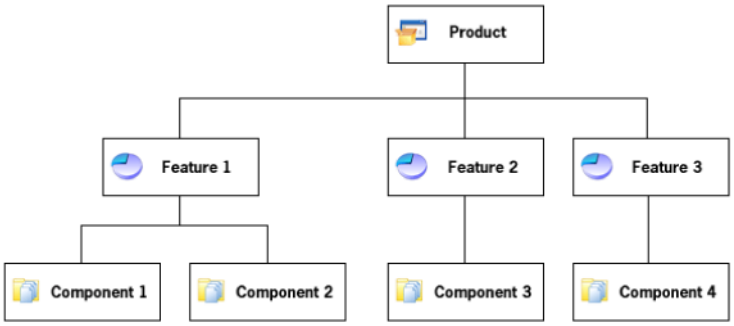


図 1: インストールの階層構造

*製品*は、インストール プロジェクトの中で最も上位にある構成区分です。製品は、通常 1 つのメイン アプリケーション (たとえば、ワードプロセッサ) と、そのアプリケーションが必要とするすべてのファイルとデータで構成されます。一連のアプリケーションが 1 つの製品となることも可能です。

*機能*は、エンドユーザーから見て、個別にインストール可能な最小の製品構成単位です。インストール プログラムの設計者は一般的に、どの機能をインストールし、どの機能をソース メディアに残すのかをユーザーが選択できるようにします。ワードプロセッサ製品では、メインの実行可能ファイルが 1 つの機能で、オプションの辞書が別の機能となっている場合があります。機能には必要な要素のすべてが含まれており、兄弟機能を必要としないのが原則です。たとえば、類語辞典機能はユーザーがインストールしないことを選択できる辞書機能を必要とはしません。ただし、機能にサブ機能を含めることもできます。サブ機能があると、エンドユーザーはインストールするファイルやデータをさらに細かく選択できるようになります。

プロジェクト内の各機能は 1 つ以上の*コンポーネント*で構成されます。コンポーネントは、インストール開発者から見て、個別にインストール可能な最小の製品構成単位です。コンポーネントはエンド ユーザーには見えないようになっています。各コンポーネントは類似のプロパティを持つファイル (およびその他のリソース) を含んでいます。たとえば、コンポーネント内のファイルはすべてエンド ユーザーのマシン上の同じディレクトリにインストールされます。また、コンポーネント内のファイルはすべて同じオペレーティング システムまたは言語に適用される必要があります。辞書機能には複数の言語別辞書コンポーネントが含まれている場合が考えられます。コンポーネントはファイルだけでなく、レジストリデータ、ショートカット、ファイル拡張子情報、およびユーザーのマシンに書き込まれるその他のシステムデータを通常含んでいます。

インストールを設計する際の総合的なタスクは、ファイルのインストール先、オペレーティング システム、言語、およびその他のプロパティに基づいて製品のファイルをコンポーネントに分割し、各コンポーネントを 1 つまたは複数の機能に関連付けることです。

アプリケーション ライフサイクル

アプリケーションのライフ サイクルは、顧客がアプリケーションをインストールした段階ですべて終了ではありません。ソフトウェア ベンダーとして、初回インストールが顧客のデスクトップで完了しただけでは、アプリケーションが成功したとは言えません。顧客は、製品のアップデートや強化内容、および重要な情報に簡単にアクセスできることを当然と考えています。この意味で顧客とのコミュニケーションやアプリケーションの状態を監視できることは、将来的な利益と発展には非常に重要です。

ソフトウェア ベンダーの中には、顧客主導のコミュニケーションを要求しがちですが、積極的に顧客との対話の機会を持たないベンダーは、多くのチャンスを逃すことになります。顧客が Web サイトまたはユーザー コミュニティを頻繁に訪れない限り、彼らはアップデート、アップグレード、パッチ、および一般的な技術記事などの情報を見逃すことになり、ベンダーは収益とサービスの機会を逃す結果となります。



図 2: FlexNet Connect のアプリケーション ライフサイクルの管理方法

上記のダイアグラムは、アプリケーション ライフサイクルの管理における FlexNet Connect の役割を図式したものです。

1. **インストールの作成**—を利用して、ソフトウェア開発者はすべてのプラットフォーム上で実行可能なインストールを容易に作成することができます。
2. **インストールの実行**—InstallShield 技術を使って作成されたインストールは、世界中 4 億以上のマシン上にインストールされています。
3. **アップデートの作成**—InstallShieldを使用すると、ソフトウェア開発者は素早くパッチおよびアップデートをビルドすることができます。
4. **ユーザーへの通知**—FlexNet Connect は、新しいアップデートのインストールが入手可能であることを各ユーザーに通知します。
5. **ダウンロードとインストール** — FlexNet Connect はアップデートのダウンロードおよびインストールをシームレスに統合し、一括処理します。
6. **レポートの表示** — FlexNet Connect はアップデートの利用率について即座にフィードバックを提供します。

InstallShield の起動

InstallShield を開始すると（製品インストール語の初回起動を除く）、すぐに InstallShield スタート ページが表示されます。スタート ページを使うと、製品情報、前回開いたプロジェクト、InstallShield リソースに簡単にアクセスすることができます。

InstallShield スタート ページ

InstallShield スタートページから、製品情報、最近開いたプロジェクト、InstallShield のリソースに簡単にアクセスすることができます。[スタート ページ] は次のセクションに分かれています。

テーブル 2・スタート ページのセクション

セクション	説明
プロジェクト タスク	プロジェクト タスクをクリックして、新規プロジェクトを簡単に作成したり、既存プロジェクトを開くことができる他、InstallShieldのインストール付属のサンプル プロジェクトを参照することもできます。
ヘルプ トピック	アクセスの多いヘルプ トピックをこのセクションに示します。[スタート] ページから InstallShield ヘルプ ライブラリ全体にアクセスするには、F1 を押すか [リソース] セクションのヘルプ ライブラリ リンクをクリックします。
(最近開いたプロジェクト)	スタート ページの中央には、前回アクセスしたプロジェクト、プロジェクト タイプ、および前回変更が加えられた日時が表示されます。
スタート ガイド	InstallShield およびインストール作成ツールに関する知識レベルに基づいて、InstallShield ヘルプ ライブラリのどのトピックを読むべきかを案内するスタート ガイドをクリックしてください。
リソース	[リソース] セクションには、役に立つ InstallShield 情報へのリンクが表示されています。 ・ **ヘルプ ライブラリ** ― InstallShield のマニュアルを表示します。 ・ **InstallShield コミュニティ**―他の InstallShield ユーザーとの情報交換、質問の投稿、または回答の検索ができる Web ベースのフォーラムです。 ・ **オンライン セミナー** ― InstallShield の評価、また Revenera 製品を最大限に利用するのに役立つ Web ベースの無料セミナーへのリンクです。 ・ **ダウンロード**―最新版の InstallShield 前提条件、InstallShield マージ モジュール、オブジェクト、ならびにサービス パック、パッチなど、ご使用中の InstallShield バージョンで利用できるダウンロードへのリンクです。 ・ **リリース ノート** ― Revenera Web サイトのナレッジ ベースに掲載されているリリース ノートを表示します。 ・ **既知の問題**―ご使用中の InstallShield バージョンに関する既知の問題を取り扱うナレッジベース記事を表示し、ワークアラウンドおよび解決策を提供します。 ・ **アップグレード アラート** ―ナレッジベース記事が表示され、InstallShield の古いバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードしたときに発生する可能性がある問題についての情報が提供されます。また、新しい InstallShield 2025 プロジェクトと以前のバージョンからアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。 ・ **RSS フィード** ― InstallShield ナレッジ ベース記事をサブスクライブできる Web ページが表示されます。

テーブル 2・スタート ページのセクション (続き.)

セクション	説明
お問い合わせ	Revenera Web サイトの [サポート] にアクセスするには、このセクションにあるリンクの 1 つをクリックしてください。

プロジェクトについて

InstallShield プロジェクトは、プロジェクトの出力を構成するファイル、フォルダーおよびオペレーションを指定します。プロジェクトの出力は、以下のいずれか 1 つです (プロジェクトの種類による):

- インストール: アプリケーションのコピーをターゲット システムで作成または更新します。
- 再配布可能ファイル (マージ モジュールまたは InstallShield オブジェクト): ンストールに含められるとき、個別の機能をインストールするために必要なロジックとファイルがすべて含まれています。
- トランスフォーム: 管理者は、インストールを配布する際、変更した設定を Windows Installer インストールに適用することができます。

プロジェクトは、必要度に応じてシンプルにも複雑にもなります。シンプルなプロジェクトは、ファイル、コンポーネント、機能、レジストリ エントリのみということもあります。より複雑なプロジェクトはこれらの他さらに、再配布可能ファイル、初期化ファイルの変更を含み、外部の DLL 関数を呼び出すこともあります。

このセクションには、さらなる詳細を含む次のサブセクションが含まれています:

- インストール プロジェクトの概要
- ニーズに合ったインストール プロジェクトを選択する
- プロジェクトの種類
- プロジェクトの使用
- サンプル プロジェクト

インストール プロジェクトの概要

InstallShield では、異なるインストール プロジェクトの種類を選択できます (InstallShield の堅牢な InstallScript プログラム言語 (InstallScript) を使ったプロジェクト、Windows Installer データベース (基本の MSI) 、またはその両方 (InstallScript MSI) を使ったプロジェクト)。

インストール プロジェクトの種類の選択は、ソフトウェアのインストールのニーズによって異なります。InstallShield は、どのプロジェクトの種類にも同じ直感的なユーザー インターフェイスを提供します。したがって、アプリケーションのニーズに応じて簡単にプロジェクトの種類が選択することができ、オーサリング エクスペリエンスはまったく変わりません。さらに、1 つの種類のプロジェクト作成で身に付けた知識は、他の種類のプロジェクトにも適用できます。

プロジェクトの種類の類似事項

すべての InstallShield プロジェクト タイプでは、ほとんどのエンドユーザーによく知られている InstallShield 独特の業界標準の外観と使用感を持ったインストールを作成することができます。すべてのプロジェクトで、エンドユーザーのニーズや期待に沿ったカスタム インストールを作成することができます。

プロジェクトタイプの機能セットは似ていますが、異なるプロジェクト間には（特に InstallScript と基本の MSI プロジェクト間）では、いくつかの重要な相違点があります。ソフトウェア インストールのニーズに最も適したプロジェクトタイプを決める方法については、「ニーズに合ったインストール プロジェクトを選択する」を参照してください。

ニーズに合ったインストール プロジェクトを選択する

InstallShield を使って様々な種類のプロジェクトを作成できますが、初心者の場合、Windows Installer テクノロジまたは InstallScript テクノロジのどちらを使ってインストールをビルドするのかという決定的な選択が 1 つだけ必要です。最も適したプロジェクトの種類の選び方は、InstallShield ソフトウエアおよびインストール開発における経験度、そしてインストールとデプロイメントにおけるニーズによって異なります。以下は、3 つの主なプロパティの種類の説明です。

基本の MSI プロジェクト

基本の MSI プロジェクトは Windows Installer サービスを使ってカスタム アクション（InstallScript、VBScript、JScript、.exe ファイル、.dll ファイル、マネージ コード）の呼び出しを含んだインストール全体を制御します。基本の MSI プロジェクト タイプは、以下のいずれかを行う場合にお勧めします:

- Windows ロゴ プログラムの要件を満たしたいとき。
- Microsoft System Center Configuration Manager などの管理ツールを使って互換性を最大限にあげたいとき、または作成中のソフトウェアが、配布される前に企業のシステム管理者によってカスタマイズされる場合。基本の MSI プロジェクト タイプでは、インストールのリパッケージする手間を省いて、インストール パッケージとその関連プロパティのトランスフォーム作成することができます。
- スクリプト コードを作成せずに、プロパティを設定してテーブル エントリを作成したいとき。
- 既存の基本の MSI プロジェクトをアップグレードしたいとき。

InstallScript プロジェクト

InstallScript プロジェクトは InstallScript を使用してインストールを操作します.このプロジェクトの種類は、以下のシナリオに推奨されます:

- エンドユーザー エクスペリエンス（エンドユーザー ダイアログ）で高度な要件がある（例、マルチメディア要素など）。
- インストールの実行時に、全画面ビルボードを使用する。
- 構造の中心で、データベース テーブルを使わずに、実行言語を使ってプロジェクトを作成したい。
- メインのインストールが実行される前後にアクションを実行したい。
- 既存の InstallScript プロジェクトをアップグレードしたい。

InstallScript MSI プロジェクト

InstallScript MSI プロジェクトは Windows Installer エンジンと InstallScript エンジンを使って、インストールの実行を進めます。インストールに、これらのエンジンが両方使用されるため、このプロジェクト タイプは最も複雑であるため、初心者ユーザーには推奨されません。このプロジェクト タイプは、次のようなシナリオで、使用できます:

- Windows ロゴ プログラムの要件を満たしたいとき。

- エンドユーザー エクスペリエンス（エンドユーザー ダイアログ）で高度な要件がある（例、マルチメディア要素など）。
- 構造の中心で、データベース テーブルを使わずに、実行言語を使ってプロジェクトを作成したい。
- メインのインストールが実行される前後にアクションを実行したい。
- 既存の InstallScript MSI プロジェクトをアップグレードしたい。



ヒント- Repackager は AdminStudio で提供されているプロジェクト変換ツールです。このツールを使って、InstallScript プロジェクトや InstallScript MSI プロジェクトを基本の MSI プロジェクトに変換できます。

プロジェクトの種類

InstallShield は様々なプロジェクトの種類を提供し、エンドユーザー向けの最適なプロジェクトの作成を支援します。

テーブル 3 - プロジェクトの種類の説明

プロジェクトの種類	アイコン	Description
アドバンスト UI		 ***エディション-** このプロジェクト タイプは、InstallShield の Professional Edition で使用できます。InstallShield Premier で、このタイプのプロジェクトを開くと、スイート/アドバンスト UI プロジェクトとして開きます。* アドバンスト UI プロジェクトでは、単一の .msi パッケージ、.msp パッケージ、および InstallScript パッケージだけでなく InstallShield 前提条件にも対して、最新のカスタマイズ可能なユーザー インターフェイスを持つブートスラップ アプリケーションを作成することができます。アドバンスト UI では、条件付きで、必要に応じて、ターゲット システムでパッケージを起動するセットアップ ランチャー（Setup.exe）を使用します。
基本の MSI		基本の MSI プロジェクトは Windows Installer を使用してインストールのユーザー インターフェイスを提供します。このプロジェクト タイプを選択した場合、機能またはコンポーネントを手動で作成し、アプリケーション ファイルおよび他の配布可能なデータをすべて指定する必要があります。

テーブル 3 - プロジェクトの種類の説明 (続き.)

プロジェクトの種類	アイコン	Description
DIM		DIM プロジェクトは、コラボレーションを促進したいエンジニア チームのために提供されています。DIM プロジェクトを使うと、組織で、インストール開発を分担して、効率化を図ることができます。デベロッパー インストール マニフェスト (DIM) は、製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、およびインストール パッケージの論理的に分かれている部分を別個に構成している要素など、関連するアイテムを集めたものです。サイズは、機能とほぼ同じです。基本の MSI に DIM を複数統合することができます。
InstallScript		InstallScript インストール プロジェクト は InstallScript 言語が持つ柔軟性を提供すると共に、機能が証明されている InstallScript エンジンを利用します。
InstallScript MSI		InstallScript MSI プロジェクト は InstallScript と Windows Installer テーブルの両方を利用します。このプロジェクトの種類では、機能またはコンポーネントを手動で作成し、アプリケーション ファイルおよび他の配布可能なデータをすべて指定する必要があります。
InstallScript オブジェクト		InstallScript オブジェクト プロジェクトは、InstallScript を利用した InstallShield マージ モジュールを作成することができます。ただし、このタイプのマージ モジュールは InstallShield でしか利用できません。
マージ モジュール		マージ モジュールを作成するには、このオプションを選択します。コンポーネントやファイルの関連付けを手動で作成する必要があります。
MSI データベース		MSI データベースを選択すると、ダイレクト編集モードでインストール プロジェクトを編集することができます。つまり、InstallShield 内で Windows Installer テーブルを直接編集してインストールを構成することができるということです。
MSM データベース		MSM データベースを選択すると、ダイレクト編集モードでマージ モジュールを編集することができます。つまり、InstallShield 内で Windows Installer テーブルを直接編集してマージ モジュールを構成することができるということです。
QuickPatch		このプロジェクトの種類は、サイズが小さい単一アップグレードをエンドユーザーに配布するインストール作成者にお勧めします。QuickPatch はカスタマイズ可能な範囲が限られてはいますが、[パッチのデザイン] ビューを使わない別のパッチ構成方法として利用できます。QuickPatch の作成は、新規 QuickPatch 作成ウィザードを使って開始してください。

テーブル 3 • プロジェクトの種類の説明 (続き.)

プロジェクトの種類	アイコン	Description
スイート/アドバンスト UI		 ***エディション•*** *このプロジェクト タイプは、InstallShield の Premier Edition で使用できます。* スイート/アドバンスト UI プロジェクトでは、複数の .msi パッケージ、.msp パッケージ、InstallScript パッケージ、.exe パッケージ、Web 配布パッケージ、サイドローディング Windows アプリ パッケージ (.appx)、および Windows Installer トランザクションだけでなく、複数の InstallShield 前提条件にも対して、最新でカスタマイズ可能なユーザー インターフェイスを持つブートスラップ アプリケーションを作成することができます。スイート/アドバンスト UI インストールでは、複数の個別のインストールを、統合されたユーザー インターフェイスを使用する単一のインストールにパッケージ化し、条件付きで、必要に応じて、ターゲット システムでパッケージを起動するセットアップ ランチャー (Setup.exe) を使用します。
MSIX		 ***エディション•*** *このプロジェクトの種類は、InstallShield Premier Edition および Professional Edition で使用できます。*  ***エディション•*** *MSIX は Windows アプリ パッケージ形式で、すべての Windows アプリにモダンなパッケージ エクスペリエンスを提供します。MSIX パッケージ形式は、既存アプリ パッケージおよび/またはファイルのインストール機能をはそのままに、Win32、WPF、および WinForm アプリに新しくモダンなパッケージ作成およびデプロイメント機能を追加します。*
MSI からMSIX への変換ウィザード		MSI からMSIX への変換ウィザードを使って、基本の MSI プロジェクトを通して既存の Windows Installer ファイル (.msi ファイル)を MSIX パッケージに変換することができます。

テーブル 3 • プロジェクトの種類の説明 (続き.)

プロジェクトの種類	アイコン	Description
MSIX 変更パッケージプロジェクト		 ***エディション:*** *このプロジェクトの種類は、InstallShield Premier Edition および Professional Edition で使用できます。* MSIX 変更パッケージは、カスタマイズを格納する新しい種類のパッケージです (Windows 10、バージョン 1809 から使用可能)。MSI 変更パッケージは、インストール時ではなくプリケーションの実行時に MSIX パッケージのオーバーレイを使って適用されます。変更パッケージは、ライセンス認証ポイントがないプラグイン/アドオンの場合もあります。IT プロフェッショナルはこの機能を使用して、企業のカスタマイズによってアプリケーションがオーバーレイされるように、MSIX コンテナを柔軟に変更できます。
トランスフォーム		トランスフォーム (.mst ファイル) は、2 つの Windows Installer データベースの差分が含まれる Windows Installer の簡易データベースです。新しいトランスフォーム プロジェクトを作成すると、トランスフォーム オープン ウィザードが開始します。
Visual Basic .NET ウィザード		このオプションを選択して、Visual Basic .NET、C# .NET、および Visual C++ .NET 用の Visual Studio .NET ウィザードを起動します。
C# .NET ウィザード		このオプションを選択して、Visual Basic .NET、C# .NET、および Visual C++ .NET 用の Visual Studio .NET ウィザードを起動します。
Visual C++ .NET ウィザード		このオプションを選択して、Visual Basic .NET、C# .NET、および Visual C++ .NET 用の Visual Studio .NET ウィザードを起動します。

アドバンスト UI およびスイート/アドバンスト UI プロジェクト



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI インストールは、インストールと InstallShield 前提条件を合わせてパッケージ化し、カスタマイズ可能な統合ユーザー インターフェイスを持つ単一インストールを作成するブートストラップ アプリケーションです。これらのインストールでは、条件付きで、必要に応じて、ターゲット システムでパッケージを起動するセットアップ ランチャー (Setup.exe) を使用します。

アドバンスト UI およびスイート/アドバンスト UI インストールには、Windows Vista 以降または Windows Server 2008 以降が必要です。

サイドローディング Windows アプリ パッケージ (.appx) を含むスイート/アドバンスト UI インストールを作成およびビルドするには、InstallShield または Standalone Build が搭載されているマシン上に Windows Vista 以降または Windows Server 2008 以降が必要です。

MSIX プロジェクト



エディション: *MSIX プロジェクト タイプは、InstallShield Premier Edition および Professional Edition で使用できます。*

MSIX は Windows アプリ パッケージ形式で、すべての Windows アプリにモダンなパッケージ エクスペリエンスを提供します。MSIX パッケージ形式は、既存アプリ パッケージおよび/またはファイルのインストール機能をはそののままに、Win32、WPF、および WinForm アプリに新しくモダンなパッケージ作成およびデプロイメント機能を追加します。

InstallShield 2019 以降では、MSIX プロジェクト タイプを使って MAIX パッケージをビルドすることができます。サイドロードまたは Windows ストア経由で配布を行う Windows アプリケーションが含まれます。

- パッケージ情報
- パッケージ ペイロード
- メディア

パッケージ情報

今回より、次を使ってプロジェクトを説明および識別することができます:

- **[一般情報]** ビューでは、プロジェクト ファイルの名前やフォーマットなどの詳細を指定します。
- **[MSIX ビジュアル資産]** エクスプローラーでは、統合的に視覚的な方法で、MSIX アプリのビジュアル的な側面を説明することができます。
- **[アプリ機能]** ビューは、特定の API または画像、音楽、またはカメラやマイクなどのデバイスといったリソースにアクセスできるよう、アプリのパッケージ マニフェスト内で有効化さている必要があります。
- **[宣言]** ビューは、アプリケーション、パッケージ レベルの宣言を作成および管理できるビジュアル ツールを提供し、それらのプロパティを構成することができます。
- **[コンテンツ URI]** ビューでは、window.external.notify を使って Script-Notify イベントをアプリに送る URI を指定します。

パッケージ ペイロード

今回より、MSIX アプリケーションをデザインするファイルを指定することができます:

- **[ファイルとフォルダー]** ビューで、InstallShield プロジェクトにファイルを追加することができます。ターゲット上にあるフォルダーへこれらのファイルを整理することができます。



重要: *MSIX プロジェクトを使って UWP アプリケーションをパッケージしている場合、同じフォルダー内にすべての資産を追加する必要があります。*

- **[アプリケーション]** ビューを使って、パッケージで提供される機能の一部またはすべてを構成する Windows アプリのプロパティを視覚的および統合的にデザインすることができます。
- **[レジストリ]** ビューでは、インストールが作成するレジストリキーおよびレジストリ値を定義することができます。

メディア

今回より、インストールを配布するのに利用するファイルとフォルダーをカスタマイズできます:

- **[パス変数]** ビューでは、変数を使って開発システム間で簡単にインストールを移動させることができます。
- **[リリース]** ビューには、プロジェクトに含まれる各製品のビジュアルツールがあります。

基本の MSI インストール プロジェクト

基本の MSI は、InstallShield における Windows Installer ベースのプロジェクト タイプです。基本の MSI プロジェクトは、Windows Installer サービスを使ってインストール全体を作成する場合にお勧めです。これを使うと、Windows Installer 機能セットだけを使ってインストールをオーサリングすることができます。MSI パッケージでは、セットアップのユーザー インターフェイス (UI) のフローだけでなく、エンドユーザー ダイアログの寸法も、直接オーサリングすることができます。Windows Installer サービスはユーザー インターフェイスのレンダリング機能を使用して、エンドユーザーに UI を表示します。このプロジェクトの種類は、インストールをオープン仕様で作成する必要がある場合に、最も適します。



メモ: *作成中のソフトウェアが、配布される前に企業のシステム管理者によってカスタマイズされる場合、基本の MSI プロジェクトが推奨されます。これはインストールのリパッケージを行なわずにインストール パッケージやその関連プロパティのトランスフォーム作成を可能にする柔軟性を提供します。*

基本の MSI プロジェクトでは、InstallScript コードをカスタム アクションという形で実行することができます。InstallScript プロジェクトと同様、基本の MSI プロジェクトは、ダイアログのオーサリング、標準 Windows .dll ファイルでのカスタム アクションの呼び出し機能、サポート ファイルの指定機能などの InstallShield により提供される他の堅牢な機能を使用します。

InstallScript カスタム アクションを使った、基本の MSI インストールのランタイム動作

InstallShield 12 以降のプロジェクト

次のステップは、InstallScript カスタム アクションを使った基本の MSI インストールのランタイム動作の概要を説明します。

1. Windows Installer サービスによってパッケージが起動されます。
2. シーケンスされた InstallScript カスタム アクションが次のように実行する。
    a. Windows Installer が Binary テーブルからロードされた ISSetup.dll の関連エントリ ポイントを呼び出す。

b. ISSetup.dll が Setup.inx のリソースを一時保管場所へ抽出する。

c. ISSetup.dll が関連スクリプト コードを実行する。

d. Windows Installer が ISSetup.dll をアンロードする。

InstallScript カスタム アクションがある InstallShield 12 以降の基本の MSI のランタイム動作は、InstallShield 11.5 以前での動作とは大きく異なります。これは強化されたアーキテクチャによるものです。強化内容についての詳細は、「InstallShield 11.5 以前のプロジェクトをアップグレードする」を参照してください。

InstallShield 11.5 および以前のプロジェクト

InstallShield 11.5 および以前のバージョンで作成されたインストールのランタイム動作は次の通りです。

1. Setup.exe が ISScript.msi をインストールする。
2. Setup.exe が MsiExec.exe を起動して、プライマリ .msi ファイルをインストールする。Setup.exe がインストールに含まれているか、またはターゲットシステムに既存する必要があります。)
3. ISMsiServerStartup (始めの カスタム アクション) が開始し、IDriver.exe が起動する。
4. シーケンスされた InstallScript カスタム アクションが次のように実行する。
   a. Windows Installer が ISScriptBridge.dll で関連エントリ ポイントを探す。
   b. ISScriptBridge.dll が ROT (実行中オブジェクト テーブル) を使って、実行中の IDriver.exe インスタンスを検出する。
   c. IDriver.exe が関連スクリプト コードを実行する。
5. ISCleanUpSuccess が IDriver.exe をシャット ダウンする。

InstallShield 11.5 以前のプロジェクトを InstallShield 2025 へアップグレードした場合、ランタイム動作は InstallShield 12 以降のプロジェクトで説明されている動作にアップデートされます。

DIM プロジェクト

InstallShield の DIM プロジェクトは、コラボレーションを促進したいエンジニア チームのために提供されています。DIM プロジェクトを使うと、組織で、インストール開発を分担して、効率化を図ることができます。デベロッパー インストール マニフェスト (DIM) は、製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、およびインストール パッケージの論理的に分かれている部分を別個に構成している要素など、関連するアイテムを集めたものです。サイズは、機能とほぼ同じです。以下は、DIM を利用した時のいくつかの利点です:

- DIM を利用することにより、複数のチーム メンバーが、インストールの開発を同時に携わることができます。各ソフトウェア開発者またはチームメンバーは、異なる DIM について個別で作業することができ、リリース エンジニアは、1 つまたは複数のインストール プロジェクトでそれらを参照することができます。
- リリース エンジニアは、DIM を複数のインストール プロジェクトで繰り返し使用できるため、非常に効率的です。
- DIM では、基本の MSI プロジェクトで提供されている機能とほぼ同じ機能がサポートされています。このため、DIM の作成者には、インストールの部分開発を行うために必要な柔軟性がすべて提供されています。

InstallScript インストール プロジェクト

InstallScript インストール プロジェクトの種類は、広く親しまれている InstallShield Professional プロジェクトと共に、InstallScript プログラム言語のパワーと柔軟性を提供します。Windows Installer は一切使用されません。インストールは、完全にスクリプトによって制御され、非常に柔軟性があります。InstallShield Professional で使用されている信頼性の高い InstallScript エンジンが使用されています。



メモ: *InstallScript インストールは一般的にシステム管理者からは好まれません。このタイプは .msi ファイルの形式ですべてがまとまっていないので Setup.exe を利用しなくてはならず、また配布に先立って完全にカスタマイズすることが不可能な場合もあるからです。作成中のソフトウェアが、配布される前に企業のシステム管理者によってカスタマイズされる場合、基本の MSI インストール プロジェクトが推奨されます。*

InstallScript エンジン

InstallScript プロジェクトでは、InstallScript によってランタイムユーザー インターフェイスの表示やフロー制御が行われ、ターゲットオペレーティング システムへの変更は Windows Installer に全く頼ることなく、信頼性の高い InstallShield の InstallScript エンジンによって行われます。

InstallScript をインストールのドライバーとして使用することには多くの利点があります。最初の利点は、InstallScript のイベントモデルによりコードを 1 行も書かずにスクリプト型のインストールを作成できる点です。カスタムの機能性を追加したい場合は、機能性を変更したいイベントを実装するだけで済みます。

基本の MSI プロジェクトのユーザー インターフェイス機能には、使用できるコントロールの種類やダイアログイベントに行うコントロールの種類に関していくらかの制限があります。InstallScript MSI プロジェクトにはそのような制限はなく、カスタム ダイアログコントロールを追加する機能とともに幅広い標準コントロールを提供します。スクリプト内でダイアログメッセージを使用することで、エンドユーザーダイアログの動作を完全にコントロールすることができます。

InstallShield Professional を使用して作成したプロジェクトのアップグレード

InstallShield Professional (バージョン 5.5 以降) を使用して作成されたインストール プロジェクトは、InstallScript プロジェクトの種類にアップグレードされます。

InstallScript プロジェクトの追加機能

InstallScript プロジェクトは、基本の MSI プロジェクトでは利用できない追加機能を提供します。

テーブル 4 - InstallScript プロジェクトの追加機能

機能	説明
[セットアップの種類] ビュー	このビューでアプリケーションの異なるインストールの構成を簡単に作成できます。また、このビューでは、[カスタム] セットアップ タイプのデフォルトを選択することもできます。
ビルボード サポート	ランタイム ビルボード サポートは、InstallScript プロジェクトでのみ利用可能です。ビルボードにより、マシンへのファイル転送中にエンドユーザーにビットマップを表示することができます。これらのビットマップを使用して、エンターテイメントや情報を提供することができます。

InstallScript MSI インストール プロジェクト

InstallScript MSI インストールは 2 つの異なるインストーラー エンジンを使用します:

- Windows Instaler エンジンは、.msi パッケージの標準 [実行] シーケンスを実行します。これは、通常ターゲット システムを変更するシーケンスです。
- InstallScript エンジンは、インストールのカスタム ユーザー インターフェイス (UI) ハンドラーとしての役割を果します。また、InstallScript コードを実行します。UI に InstallScript エンジンを使用する際の利点は、高度にカスタマイズされたランタイム ダイアログのサポートが提供されていることです。

さらに、InstallShield では、InstallScript MSI インストールのUI に、次の 2 つの異なるスタイルが追加されました:

- **従来型スタイル** – InstallScript MSI インストールで、InstallScript エンジンを外部 UI ハンドラーとして使用することができます。このスタイルを使用する場合は、インストールに Setup.exe セットアップランチャーを必ず含めてください。セットアップランチャーはブートストラップ アプリケーションとして機能し、UI を表示する InstallScript エンジンを開始して InstallScript コードを実行し、また Windows Installer を開始して .msi パッケージの [実行] シーケンスを実行します。
- **新しいスタイル** – InstallScript MSI インストールで、InstallScript エンジンを埋め込み UI ハンドラーとして使用することができます。このスタイルでは、InstallScript エンジンが .msi パッケージ内に埋め込まれます。Windows Installer は InstallScript エンジンを呼び出して UI を表示します。Windows Instaler はまた、.msi パッケージの [実行] シーケンスを実行します。

  このオプションには、ターゲット マシン上に Windows Installer 4.5 が必要です。また、いくつかの制限事項があるため、このスタイルを使用する場合は綿密な計画が必要です。

2 つのスタイルに関する詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

メモ: *このプロジェクトの種類は異なるエンジンを使用するため、単なる InstallScript や 基本の MSI インストール プロジェクトより複雑になっています.これは高度な知識をもつユーザーのみに推奨されます。*

システム管理者には、従来型スタイルの InstallScript MSI インストールよりも、基本の MSI インストールまたは新しいスタイルの InstallScript MSI インストールが好まれます。これは、従来型スタイルの InstallScript MSI インストールが、.msi パッケージのように自己完結形式ではないため、Setup.exe を使用する必要があることと、展開の前に完全なカスタマイズが不可能な場合があるためです。作成中のソフトウェアが、配布される前に企業のシステム管理者によってカスタマイズされる場合、基本の MSI プロジェクトが推奨されます。代わりの方法として、新しいスタイルの InstallScript MSI インストールを使用することもできますが、必要なすべてのシステム変更は、[インストール]–[実行] シーケンスでカスタム アクションを使用して行う必要があることに注意してください。権限が不足しないように、カスタム アクションは、システム コンテキストで "スクリプト内実行" 設定が遅延実行になっている必要があります。

InstallScript MSI プロジェクトのその他の機能

InstallScript MSI プロジェクトでは、基本の MSI プロジェクトで利用できない追加機能がいくつか提供されます:

テーブル 5 ・ InstallScript MSI プロジェクトの追加機能

機能	説明
[セットアップの種類] ビュー	このビューでは、アプリケーションに、[標準]、[最小] など、異なる定義済みのインストール構成を簡単に作成することができます。また、このビューでは、[カスタム] セットアップ タイプのデフォルトを選択することもできます。
ビルボード サポート	実行時のビルボードは、InstallScript MSI インストールでのみサポートされています。基本の MSI インストールではサポートされていません。ビルボードを利用すると、エンド ユーザーのマシンへファイルを転送中に、ビットマップを表示することができます。これらのビットマップを使用して、エンドユーザーにエンターテイメントや役に立つ情報を提供することができます。

InstallScript MSI インストールのランタイム動作

InstallShield 12 以降のプロジェクト

次のステップは、InstallScript MSI インストールのランタイム動作についての概要を説明します。

1. Setup.exe が ISSetup.dll を起動する。
2. ISSetup.dll が行う処理は次の通りです。
    a. Setup.inx をロードする。
    b. プリ インストール InstallScript イベント (例、OnBegin) を実行する。
    c. Windows Installer (外部ユーザー インターフェイスの場合、InstallScript エンジン) を起動する。
    d. 各 InstallShield カスタム アクションが次のように実行する (例、OnMoved)。(これは、InstallScript カスタム アクションを使った基本の MSI で使われるメカニズムと基本的に同じです。)
        i. Windows Installer が ISSetup.dll で適切なエントリ ポイント呼び出す。(これは、ISSetup.dll の異なるインスタンスで、Binary テーブルからロードされます。)
        ii. ISSetup.dll が Setup.inx のリソースを一時保管場所へ抽出する。
        iii. ISSetup.dll が関連スクリプト コードを実行する。
        iv. ポスト インストール InstallScript イベント (例、OnEnd) を実行する。

InstallShield 12 以降の InstallScript MSI プロジェクトのランタイム動作は、InstallShield 11.5 以前での動作とは大きく異なります。これは強化されたアーキテクチャによるものです。強化内容についての詳細は、「InstallShield 11.5 以前のプロジェクトをアップグレードする」を参照してください。

InstallShield 11.5 および以前のプロジェクト

InstallShield 11.5 および以前のバージョンで作成された InstallScript MSI インストールのランタイム動作は次の通りです。

1. Setup.exe が ISScript.msi をインストールする。
2. Setup.exe が IDriver.exe を起動する。
3. IDriver.exe が行う処理は次の通りです。
    a. Setup.inx をロードする。
    b. プリ インストール InstallScript イベント (例、OnBegin) を実行する。
    c. Windows Installer (外部ユーザー インターフェイスの場合、InstallScript エンジン) を起動する。

        各 InstallScript カスタム アクションが次のように実行する (例えば、OnMoved)。(これは、カスタム アクションを使う基本の MSI インストールで使われるメカニズムと基本的に同じです。)

        i. Windows Installer が ISScriptBridge.dll で適切なエントリ ポイントを呼び出す。
        ii. ISScriptBridge.dll が ROT (実行中オブジェクト テーブル) を使って、実行中の IDriver.exe インスタンスを検出する。
        iii. IDriver.exe が関連スクリプト コードを実行する。
    d. ポスト インストール InstallScript イベント (例、OnEnd) を実行する。

マージ モジュール プロジェクト

マージ モジュールにより、既存の各機能をインストールに追加できます。たとえば、アプリケーションで VB Runtime DLL が必要な場合、Microsoft's Visual Basic Virtual Machine モジュールをセットアップ アプリケーションに追加することができます。従来は、レジストリ エントリを作成し、ターゲットフォルダー、カスタム アクション、および任意の他の必要なステップを設定しなければならなりませんでした。マージ モジュールでは、既存のマージ モジュールをインストールに関連付けて、次の作業に移動するだけでこれらのステップを完了できます。

InstallShield を利用して、専用のマージ モジュールを作成し、インストールの配布または他の開発者への供与ができるようになります。これは、すべてのインストール プロジェクトに機能を分散する場合に役立ちます。その部分のインストールを毎回作り直すのではなく、マージ モジュールとして 1 度作成し、それをすべてのインストール プロジェクトに関連付けます。

マージ モジュール プロジェクトに含まれる InstallScript カスタム アクション

マージ モジュールに InstallScript カスタム アクションを含めることができます。ただし、カスタム アクションが適切にマージされるよう、Module**Sequence テーブルエントリを手作業で作成しなくてはなりません。

マージ モジュールの動作のしくみ

マージ モジュールをインストール プロジェクトに関連付けると、ビルド時に .msi データベースにマージされます。その結果、2 つのプロジェクト (新規インストールと既存のマージ モジュール) が MSI データベースにマージされることになります。マージ モジュールはコンポーネントのように動作します。つまり、関連付けされている機能がインストールされなければ、マージ モジュールもインストールされません。したがって、VB Runtime DLL を必要とする機能があるのに VB Runtime DLL がインストール対象になっていない場合、VB マージ モジュールはインストールされません。

MSI データベースおよび MSM データベース プロジェクト

MSI データベースおよび MSM データベース プロジェクトを使用すると、中間プロジェクト フォーマット (.ism ファイル) で作業せずに、Windows Installer インストール パッケージおよびマージ モジュール データベースを編集することができます。これらのプロジェクトタイプは、ダイレクト エディター機能を拡張して、標準インストール プロジェクトとマージ モジュール プロジェクトでサポートされている異なるビューを含めます。これらのプロジェクトの種類でサポートされているビューは、.ism プロジェクトで機能するのと同じように機能するように設計されています。



プロジェクト: *MSI データベースおよび MSM データベース プロジェクトでは、文字列エントリもパス変数もサポートされていません。*

既存の .msi ファイルまたは .msm ファイルを直接編集したり、新規の MSI データベースまたは MSM データベースを開いて新規のデータベースを直接作成することもできます。このファイルの中のデータは、ブランクの基本の MSI、またはマージ モジュール プロジェクトを作成してそのプロジェクトからデータベースをビルドしたときに取得するものと合致します。

InstallScript オブジェクト プロジェクト

InstallShield オブジェクトを使用して、Windows テクノロジを簡単にインストールできます.テクノロジをインストールするには、インストールに対応する InstallShield オブジェクトを含めるだけです。特定のコンポーネントが必要な場合は、機能に対応する InstallShield オブジェクトを関連付けます。

QuickPatch プロジェクト

QuickPatch プロジェクトは、規模の小さいシングル アップグレードをユーザーへ配布したいインストール作成者へお勧めするプロジェクトの種類です。カスタム アクションの追加、.ini データの変更などのより広範囲におよび変更には通常、標準パッチが必要です。

QuickPatch はカスタマイズ可能な範囲が限られてはいますが、[パッチのデザイン] ビューを使わないシンプルなパッチ構成方法として利用できます。基本的にどちらのパッチ作成方法も同じ配布タイプ (.msp と .exe ファイル) を作成します。

QuickPatch では、次のすべてを実行することができます。

- 元のインストールまたは以前の QuickPatch へ新しいファイルを追加する。
- 元のインストールのファイルを削除する。
- 以前の QuickPatch と共に追加されたファイルを削除する。
- 上記と同じ操作をレジストリ エントリで実行する。
- 元のインストールに含まれていたが、現在の QuickPatch プロジェクトには適用しないカスタム アクションを削除する。

QuickPatch プロジェクトの作成は、新規 QuickPatch 作成ウィザードで始めます。ウィザードを完了した後、InstallShield でプロジェクトが開いてから QuickPatch プロジェクトの設定を構成できます。

トランスフォーム プロジェクト

トランスフォーム (.mst ファイル) は、2 つの .msi データベース間の差分を含むシンプルな Windows Installer データベースです。トランスフォームを使うと、管理者は、インストール パッケージを配布する際、変更した設定をデータベースに適用することができます。

InstallShield は、.msi パッケージを、InstallShield (.ism) プロジェクトに変換することなく編集することができるトランスフォーム プロジェクト タイプを備えています。変更点はトランスフォーム (.mst) ファイルとして保存できます。

トランスフォームプロジェクトを作成、または InstallShield でトランスフォームを開くと、トランスフォーム オープン ウィザードが起動し、ベース .msi ファイルやベース .msi ファイルに適用するその他のトランスフォームファイルについての情報を収集することができます。

プロジェクトの使用

InstallShield では、インストール プロジェクトから InstallShield 内で機能性を再使用することができる InstallShield オブジェクト プロジェクトまで、様々な種類のプロジェクトを作成、編集、アップグレードおよび保存することができます。

このセクション内のページでは、特定のプロジェクトの種類およびプロジェクトテンプレートの作成の仕方、また、異なる InstallShield 製品からアップグレードするとき考慮に入れる点等、いろいろなトピックがカバーされています。

新しいプロジェクトの作成

InstallShield プロジェクトを新規に作成するとき、その方法はいくつかあります。



タスク

新しいプロジェクトを作成するには、以下のいずれかを行います。

- ツールバーまたは InstallShield スタート ページ 上で **[新規プロジェクト]** ボタンをクリックします。
- Ctrl+N を押します。
- **[ファイル]** メニューで、**[新規]** をクリックします。

これらの手順を行うと、[新規プロジェクト] ダイアログ ボックスが起動されるので、そこで作成するプロジェクトを選択できます。

テンプレートを使って新規プロジェクトを作成する

プロジェクト テンプレートは、該当するタブの [新規プロジェクト] ダイアログ ボックスに表示されます。詳細については、「テンプレートを使用した新規プロジェクトの作成」を参照してください。

Microsoft Visual Studio 内からプロジェクトを作成する

Microsoft Visual Studio .NET ワークスペース内から InstallShield プロジェクトを作成することができます。詳細については、「Microsoft Visual Studio で InstallShield プロジェクトを作成する」を参照してください。

プロジェクトを開く

タスク ***既存の InstallShield プロジェクトを開くには、次のいずれかの操作を実行します。***

- ツールバーにある **[プロジェクトを開く]** ボタンをクリックする。
- **[ファイル]** メニューで、**[開く]** をクリックします。[**開く**] ダイアログ ボックスで、プロジェクトファイルに移動します。
- Ctrl+O を押します。
- **[既存のプロジェクトを開く]** リンクをクリックするか、InstallShield スタート ページで前回開いたプロジェクトのリンクをクリックします。
- デスクトップ上または Windows エクスプローラー内でプロジェクト ファイル (.ism) をダブルクリックする。

特定のファイルまたはファイル リンクをクリックした場合を除き、上のすべてのオプションで [開く] ダイアログ ボックスが開くので、ここでプロジェクト ファイルを参照する。

ソース コード管理からプロジェクトを開く

[開く] ダイアログのオプションを使うと、ソース コントロール アプリケーションから最新バージョンのプロジェクト フォルダーを使用できるようになります。

タスク ***ソース管理からプロジェクトを開くには、以下の手順に従います:***

1. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
2. **[ソース管理]** ボタンをクリックします。ソース管理プログラムのログイン ダイアログが表示されます。
3. ソース管理プログラムにログインし、プロジェクト フォルダーまたはデータベースを参照します。

InstallShield では、プロジェクトのすべてのファイルについて最新のバージョンが作業ディレクトリに取得されます。

メモ: *InstallShield があるシステムにソース管理アプリケーションがない場合、[開く] ダイアログ ボックスは、[ソース管理] ボタンを表示しません。*

ダイレクト編集モードでパッチ作成プロパティ ファイルを開く

[開く] ダイアログ ボックスから、以前に作成したパッチ作成プロパティ ファイルを開くことができます。

タスク ***パッチ作成プロパティ (.pcp ファイル) を開くには、以下の手順に従います:***

1. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
2. **[ファイルの種類]** 一覧で、**[パッチ作成プロパティ ファイル (*.pcp)]** をクリックします。
3. 編集する .pcp ファイルを選択して、**[開く]** をクリックします。

パッチプロジェクトファイルがダイレクトエディターで開きます。

Windows Installer パッケージを開く

InstallShield には、IDE から直接、ダイレクト編集モードを使って MSI パッケージを編集するか、InstallShield プロジェクト (.ism) に変換してから IDE で基本の MSI プロジェクトとして編集するオプションがあります。



メモ: *MSI パッケージをダイレクト編集モードで編集する場合、制限される機能もあります。*



タスク ***MSI パッケージを IDE の直接編集モードで開くには、以下の手順を実行します。***

1. ツールバーにある **[プロジェクトを開く]** ボタンをクリックする。参照ダイアログが表示されます。
2. **[ファイルの種類]** リストから **[Windows Installer パッケージ (*.msi)]** を選択します。
3. 開く MSI パッケージを選択します。
4. 必ず **"形式を選択して開く"** フィールドを [自動] に設定してから **[開く]** をクリックしてください。



メモ: *デフォルトでは、IDE は直接編集モードで MSI パッケージを開きます。*

MSI/MSM オープン ウィザードを使用すると、既存の Windows Installer (.msi) パッケージを InstallShield プロジェクト (.ism) ファイルに変換し、新規プロジェクトを開いて IDE で変更できます。



メモ: *InstallShield で、パッチ作成プロジェクト ファイル (.pcp ファイル) を開いて、ダイレクトエディターで編集することもできます。*

マージ モジュールを開く

MSI/MSM オープン ウィザード を使用すると、既存の Windows Installer のマージ モジュール (.msm ファイル) を InstallShield プロジェクト (.ism) ファイルに変換し、InstallShield IDE で新規プロジェクトを開くことができます。



タスク ***MSI/MSM オープンウィザードを起動するには、次の操作を実行します。***

1. ツールバーにある **[プロジェクトを開く]** ボタンをクリックする。参照ダイアログが表示されます。
2. **[ファイルの種類]** リストから **[Windows Installer モジュール (*.msm)]** を選択します。
3. マージ モジュールまで移動して [**開く**] をクリックします。

作成するプロジェクトの詳細情報を指定するよう、MSI/MSM オープンウィザードにメッセージが表示され、新規マージ モジュールプロジェクトが IDE で開かれます。

オブジェクト プロジェクトを開く

InstallShield Professional を使って作成されたオブジェクトプロジェクト (.ipo または .ipr) を開くことができます。InstallShield でこのプロジェクトを開くと、プロジェクトはマージ モジュール (.msm) プロジェクトに移行されます。

[再配布可能ファイル] ビューにマージ モジュールを配置する

マージ モジュールを InstallShield インストール プロジェクトを関連付けるとき、セットアッププロジェクトの [マージ モジュール] ビュー内で使用することができるように、マージ モジュールプロジェクトを Modules フォルダーに自動的に配置することができます。



タスク ***マージ モジュール プロジェクトを Modules フォルダーに自動的に配置するには、以下の手順に従います:***

リリース ウィザードの [マージ モジュールのオプション] パネルで "ローカル マージ モジュール カタログに追加" オプションを選択します。

プロジェクトの保存

新しいプロジェクトを作成すると、プロジェクトは自動的に [新規プロジェクト] ダイアログ ボックス で指定した名前と場所を使って保存されます。

InstallShield では、開いたプロジェクトのコピーをテンプレートとして、または新しいプロジェクトとして別の名前をつけて別の場所に保存することができます。

新しい名前と場所でプロジェクトを保存する

新しい名前をつけて新しい場所にプロジェクト保存したとき、名前が変更されたプロジェクトファイルおよびその関連ファイル並びにフォルダーのコピーが新しい場所に保存されます。次回プロジェクトを保存する際、変更はすべてこの新しいフォルダーに保存されます。



タスク ***新しい名前を付けて、新しい場所にプロジェクトを保存するには、次の手順を実行します。***

1. **[ファイル]** メニューで、**[名前を付けて保存]** をクリックします。**[名前を付けて保存]** ダイアログ ボックスが開きます。
2. **[名前を付けて保存]** ボックスで、適切な場所を選択します。
3. 必要に応じて **[″プロジェクト名″ サブフォルダーを作成し、作成したフォルダーにプロジェクトを保存する]** チェック ボックスを選択またはクリアします。どちらの場合も、*<プロジェクト名>* サブフォルダーは、**[プロジェクトの場所]** ボックス (**[オプション]** ダイアログ ボックスの **[ファイルの場所]** 上) で作成されます。

   必要に応じて **[‘プロジェクト名’ サブフォルダーを作成してプロジェクトを保存する]** チェック ボックスを選択すると、プロジェクト ファイル (.ism) は、*<プロジェクト名>* サブフォルダーに保存されます。チェック ボックスをクリアすると、プロジェクト ファイルは **[プロジェクトの場所]**ボックスで指定した場所に保存されます。

4. 新しいプロジェクトの GUID を、オリジナルプロジェクトとは異なる GUID にするには、[**保存したプロジェクトに新しいプロジェクト GUID を作成して割り当てる**] を選択します。新しいプロジェクトの GUID をオリジナルプロジェクトと同じ GUID にするには、このチェック ボックスをクリアします。
5. ([**一般情報**] ビューにある) **"製品名"** 設定の値として、**[ファイル名]** ボックスで指定した名前を使用するには、**[新しいプロジェクト名に基づいて適切にプロジェクト設定をアップデートする]** チェック ボックスを選択します。新しいプロジェクトの **"名前"** プロパティ を元のプロジェクトと同じにするには、このチェック ボックスをクリアします。



メモ: *InstallScript ファイルなど、外部依存関係と共にプロジェクトを保存すると、新しいプロジェクトはこれらのファイルの元のコピーをポイントします。コピーは複製されません。したがって、オリジナルのプロジェクトを削除する場合は、新しいプロジェクトで使用する可能性のあるファイルを削除しないように注意してください。*

プロジェクトを別の種類のプロジェクトに変換する

InstallShield では、一部のプロジェクトを別の種類のプロジェクトに変換することができます。

テーブル 6 ・ プロジェクト コンバーター

プロジェクト コンバーター	手順
基本 MSI プロジェクトを InstallScript MSI プロジェクトに変換	「基本の MSI プロジェクトを InstallScript MSI プロジェクトへ変換する」を参照してください。
InstallScript MSI プロジェクトを InstallScript プロジェクトに変換	「InstallScript MSI プロジェクトから InstallScript プロジェクトへ変換する」を参照してください。
InstallScript プロジェクトを基本の MSI プロジェクトに変換	Repackager を使って、InstallScript プロジェクトを基本の MSI プロジェクトに変換します。  ***エディション:*** *Repackager は AdminStudio に含まれています。*
InstallScript MSI プロジェクトを基本の MSI プロジェクトに変換	Repackager を使って、InstallScript MSI プロジェクトを基本の MSI プロジェクトに変換します。  ***エディション:*** *Repackager は AdminStudio に含まれています。*

テーブル 6 • プロジェクト コンバーター

プロジェクト コンバーター	手順
InstallScript プロジェクトを InstallScript MSI プロジェクトに変換	変換は、次の 2 つのステップを実行します: 1. Repackager を使って、InstallScript プロジェクトを基本の MSI プロジェクトに変換します。 2. InstallShield を使って、基本の MSI プロジェクトを InstallScript MSI プロジェクトに変換します。「基本の MSI プロジェクトを InstallScript MSI プロジェクトへ変換する」を参照してください。  ***エディション:*** *Repackager は AdminStudio に含まれています。*
Visual Studio セットアップ プロジェクトを InstallShield の基本の MSI プロジェクトに変換	「Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートする」を参照してください。
Visual Studio マージ モジュール プロジェクトを InstallShield のマージ モジュール プロジェクトに変換	「Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートする」を参照してください。

基本の MSI プロジェクトを InstallScript MSI プロジェクトへ変換する

基本の MSI プロジェクトを InstallScript MSI プロジェクトに変換すると、InstallShield のスクリプト作成機能を最大限に活用できます。



注意 • *一度変換したプロジェクトを元に戻すことはできません。InstallScript MSI プロジェクトに変換された基本の MSI プロジェクトは、一度保存されると、容易に元の基本の MSI プロジェクトに戻すことはできなくなります。基本の MSI プロジェクトを変換する前に、プロジェクトのバックアップ コピーを作成することが一般的に推奨されます。*

プロジェクトの変換



タスク ***基本の MSI プロジェクトを InstallScript MSI プロジェクトに変換するには、以下の手順を実行します。***

1. 基本の MSI プロジェクトを開きます。
2. **[プロジェクト]** メニューで、**[プロジェクト コンバーター]** をポイントして、**[InstallScript MSI プロジェクトへ変換]** をクリックします。

ダイアログの変換

InstallScript MSI プロジェクトは、Windows Installer ダイアログ関連のテーブルではなく、リソースベースのダイアログ を使用するため、元のプロジェクトで使っていた Windows Installer ベースのダイアログは、プロジェクトの移行後は利用不可能になります。このような場合、カスタム ダイアログは手作業で追加してください。

変換後のインストール プロジェクトには、デフォルトの InstallScript MSI ダイアログが用意されています。このダイアログ ボックスを確かめるには、InstallScript ビューに移動して、OnFirstUIBefore か OnFirstUIAfter イベントで表示されるダイアログを確認してください。

InstallScript MSI プロジェクトから InstallScript プロジェクトへ変換する

InstallShield、InstallShield DevStudio、または InstallShield Developer を使用して作成された InstallScript MSI プロジェクトも、簡単に InstallScript プロジェクトに変換することができます。

タスク ***既存の InstallScript MSI プロジェクトを InstallScriptプロジェクトに変換するには、以下の手順を実行します。***

1. InstallScript MSI プロジェクトを開きます。
2. **[プロジェクト]** メニューで、**[プロジェクト コンバーター]** をポイントして、**[InstallScript プロジェクトへ変換]** をクリックします。

GUID

GUID は、グローバル固有 ID (*Globally Unique Identifier*) の略です。GUID は 128 ビット長で、各 GUID は GUID 生成に使用されるアルゴリズムによって一意になるように生成されます。GUID は固有でなければならないため、COM クラスや製品コード、およびその他のさまざまなコードを識別するのに使用することができます。

たとえば製品をインストールした後で、HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Uninstall の下にキーを作成し、インストールの製品コードに従って命名することができます。このキーは製品名に従って命名されていたことがありました。しかしこのために競合が起きる可能性がありました。2 つのインストールが同じマシンにインストールされて、両方が同じ製品名を共有する場合、同じレジストリキーが共有されることになったからです。GUID が使用されるようになったため、このような競合が起きることがなくなりました。

GUID は {5D607F6A-AF48-4003-AFA8-69E019A4496F} のような形式です。GUID の文字はすべて大文字でなければなりません。

プロジェクトの GUID

インストール プロジェクトを作成すると、プロジェクトに関連した多くの異なる GUID ができます。

テーブル 7 ・ GUID

GUID 名	説明
製品コード	製品 GUID は、アプリケーションを一意に識別します。
パッケージ コード	パッケージ コードはインストール パッケージを一意に識別します。 ***プロジェクト:*** *この GUID は、基本の MSI、InstallScript MSI、および QuickPatch プロジェクト タイプで使用できます。*

テーブル 7 • GUID (続き.)

GUID 名	説明
パッケージ GUID	パッケージ GUID は、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるインストール パッケージを一意に識別します。  ***プロジェクト•*** *この GUID は、アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用できます。*
パッチ GUID	パッチ GUID は、パッチ パッケージを一意に識別します。  ***プロジェクト•*** *この GUID は、基本の MSI、InstallScript MSI、および QuickPatch プロジェクト タイプで使用できます。*
スイート GUID	スイート GUID は、アドバンスト UI またはスイート/アドバンスト UI インストールを一意に識別します。  ***プロジェクト•*** *この GUID は、アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用できます。*
アップグレード コードとアップグレード GUID	アップグレード GUID はアップグレードの目的で製品のファミリーを識別します。これはメジャーアップグレードに重要です。  ***プロジェクト•*** *この GUID は、基本の MSI、InstallScript MSI、および QuickPatch プロジェクト タイプで使用できます。*



プロジェクト• *基本の MSI プロジェクトまたは InstallScript MSI プロジェクトで GUID を変更する必要がある場合についての情報は、「メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い」を参照してください。*

デフォルトのプロジェクトの場所を変更する

すべての新しいプロジェクトは次のデフォルトの保存場所に保存されます:

C:¥InstallShield 2022 Projects

インストール プロジェクトに対して新規デフォルトの場所を指定するには、**[オプション]** ダイアログ ボックスの **[ファイルの場所]** タブを利用します。

メモ: このフォルダーは、すべての新規インストール プロジェクトに対して使用されますが、既存のプロジェクトは以前のフォルダーにそのまま保存されます。

プロジェクト要素を再利用する

プロジェクトの要素をいくつか保存して別のプロジェクトで再利用することができます。データによっては中間ファイルに保存しなければならないものもあれば、別の .ism ファイルに直接エクスポートできるものもあります。各要素の使用方法を以下に説明します。

プロジェクト全体

プロジェクトを InstallShield プロジェクトのテンプレートとして保存すると、元のプロジェクトとほぼ同じプロジェクトを、そのテンプレートから作成することができます。

レジストリ エントリ

REG ファイルにレジストリ エントリをエクスポートすると、同じプロジェクトまたは全く別のプロジェクトの別のコンポーネントのレジストリデータに、このエクスポートしたエントリをインポートすることができます。

文字列のエントリ

プロジェクトの特定言語の文字列エントリを、タブ区切りのテキスト ファイルにエクスポートすることができます。一般的には、テキスト ファイルを翻訳に出して、その文字列エントリを別の言語用にプロジェクトにインポートしますが、テキスト ファイルを別のプロジェクトにインポートする方法で、エクスポートされた文字列エントリを別のプロジェクトに追加することもできます。

エンドユーザー ダイアログ

すべてのダイアログを .rc ファイルに保存し、それぞれを個別に InstallShield ダイアログ テンプレート ファイルにエクスポートするか、または別のプロジェクトに直接エクスポートすることができます。

コンポーネント

コンポーネントを別のプロジェクトにエクスポートするには、コンポーネントを右クリックして [コンポーネントのエクスポート ウィザード] を選択します。ウィザードの指示に従うと、データのすべてが指定のプロジェクトに追加されます。

カスタム アクション

[カスタム アクション] エクスプローラーでカスタム アクションを右クリックして、[エクスポート] をクリックすることにより、カスタム アクションを別のプロジェクトにエクスポートすることができます。[カスタム アクション] エクスプローラーは、[カスタム アクションとシーケンス] ビュー (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) および [カスタム アクション] ビュー (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) にあります。

リポジトリを使用してプロジェクトの要素を共有する

リポジトリの使用は、InstallShield でサポートされています。リポジトリは、異なるインストール プロジェクトで共有および再利用することができる共通要素のコレクションです。以下は、リポジトリに格納できる要素です。

- エンドユーザー ダイアログ
- InstallScript ファイル
- マージ モジュール
- プロジェクト テンプレート
- SQL スクリプト
- システム検索

リポジトリには、複数のプロジェクトでプロジェクト要素を再利用することができる機能が用意されており、整合性を保つのに役立ちます。また、前に行った作業を複製する手間も省くことができます。たとえば、ある組織で、多数の製品インストールが特定のカスタム ダイアログを含むとき、そのカスタム ダイアログを一度だけ作成し、それをリポジトリにパブリッシュするだけですみます。他のインストールでそのダイアログを使用したいとき、そのダイアログをリポジトリからプロジェクトに追加するだけです。

InstallShield で利用できるリポジトリには 2 つのタイプがあります。

- ネットワーク リポジトリ ー ローカル リポジトリは、複数のプロジェクトで再利用することができる、自分専用のインストール要素のコレクションです。ローカル リポジトリは、ローカルマシンに格納され、他のインストール作成者は利用することが *できません*。
- ネットワーク リポジトリ ー ネットワーク リポジトリは、複数のインストール作成者が必要に応じてアクセスでき、プロジェクトで再利用することができるインストール要素のコレクションです。ネットワーク リポジトリは、インストール作成者同士のコラボレーション作業を促進し、ネットワークに格納されます。ネットワーク リポジトリの構成するための手順と注意事項は、「ネットワーク リポジトリを設定する」を参照してください。



エディション: *ネットワーク リポジトリは、InstallShield の Premier Edition でのみ利用することができます。ローカル レポジトリは InstallShield Premier Edition および Professional Edition の両方で使用できます。*

ネットワーク リポジトリを設定する



エディション: *ネットワーク リポジトリは、InstallShield の Premier Edition でのみ利用することができます。*

ネットワークリポジトリを構成する、またはネットワークリポジトリの設定を変更するには、[オプション] ダイアログ ボックスにある [リポジトリ] タブを利用します。

プロジェクト テンプレート

プロジェクト テンプレートは、インストール プロジェクトやマージ モジュールの作成を開始するときに使用する、すべてのデフォルトの設定とデザイン要素を含みます。

InstallShield でテンプレートを開き、通常のプロジェクトを編集するのと同様に編集してください。テンプレートを作成するには、任意のプロジェクトをテンプレートとして保存します。詳細については、「プロジェクト テンプレートの作成」を参照してください。

テンプレートを基に新しいプロジェクトを作成する方法については、「テンプレートを使用した新規プロジェクトの作成」をご覧ください。

InstallShield プロジェクトテンプレートは、新規プロジェクトの開始点としてのみ利用することができます。テンプレートを使ってプロジェクトを作成しても、現在のプロジェクトと既存のテンプレートの間に何ら関連はありません。つまり、テンプレートの要素を変更しても、このテンプレートを使って作成したプロジェクトは影響を受けません。ただし、テンプレートを変更し、テンプレートの更新バージョンを使って別のプロジェクトを作成することはできます。

プロジェクト テンプレートの作成

テンプレート ファイルの作成方法は、.ism プロジェクト ファイル（基本の MSI、InstallScript、InstallScript MSI、マージ モジュール）を使用するプロジェクト タイプと、.issuite プロジェクト ファイルを使用するプロジェクト タイプのどちらのテンプレートを作成するのかによって異なります。

.ism プロジェクト ファイルを使用するプロジェクトのプロジェクト テンプレート ファイルを作成する

任意の .ism プロジェクト ファイル（インストール プロジェクトとマージ モジュール プロジェクトを含む）からプロジェクト テンプレート を作成できます。.ism プロジェクト ファイルからテンプレートを作成すると、テンプレート ファイルに拡張子 .ist が付きます。



タスク ***.ism プロジェクトからテンプレート（.ist）を作成するには、以下の手順に従います:***

1. 希望のデフォルトの設定、ダイアログ、機能、その他を含めるプロジェクトを編集します。
2. **[ファイル]** メニューで、**[名前を付けて保存]** をクリックします。**[名前を付けて保存]** ダイアログ ボックスが開きます。
3. **[名前を付けて保存]** ボックスで **[InstallShield テンプレート (*.ist)]** を選択します。
4. ファイル名（.ist 拡張子を付ける）を入力して、新しいテンプレートの場所を選択します。テンプレートの推奨場所は、プロジェクトの場所フォルダーの下にある［テンプレート］フォルダーです。
5. **[OK]** をクリックします。

InstallShield でテンプレートを開き、変更を追加することができます。最終バージョンのテンプレートを使用して、新規プロジェクトを作成 することができます。

.issuite プロジェクト ファイルを使用するプロジェクトのプロジェクト テンプレート ファイルを作成する

任意のアドバンスト UI またはスイート アドバンスト UI プロジェクト（.issuite）をテンプレートに指定できます。これらのプロジェクト タイプのプロジェクト ファイルとテンプレート ファイルは、同じ拡張子 .issuite を持ちます。



タスク ***.issuite プロジェクト ファイルをテンプレート ファイルとし、[新規プロジェクト] ダイアログ ボックスでそのテンプレートを選択できるようにするには、以下の手順に従います:***

[新規プロジェクト] ダイアログ ボックスの **[すべてのタイプ]** タブを右クリックしてから、**[新規テンプレートの追加]** をクリックします。テンプレートとして使用する .issuite ファイルを参照して、**[すべてのタイプ]** タブに新しく追加したテンプレートのアイコンが表示されます。

テンプレートを使用した新規プロジェクトの作成

時間を節約するため、デフォルトの設定と設計要素を持つ単一のプロジェクトテンプレート上に複数のプロジェクトを作成できます。テンプレートは、システム上の任意の場所に格納することが可能です。レポジトリに格納することもできます。



タスク ***テンプレートを出発点としてプロジェクトを作成するには、次の手順に従います。***

1. **[ファイル]** メニューで、**[新規]** をクリックします。**[新規プロジェクト]** ダイアログ ボックスが開きます。
2. **[すべてのタイプ]** タブをクリックします。
3. 使用するテンプレートを選択します。

   ***メモ・レポジトリに格納されているテンプレートを表示する**には、プロジェクトタイプのボックスを右クリックして、レポジトリのテンプレートを表示するが有効となっていることを確認します。タブへテンプレートを追加するには、プロジェクトタイプのボックスを右クリックして、**[新規テンプレートの追加]** をクリックします。そして必要なテンプレートファイル (.ist ファイル) を参照します。*
4. **[プロジェクト名]** ボックスで、プロジェクトの名前を入力します。
5. **[場所]** ボックスに、テンプレートを格納するディレクトリへのパスを入力するか、**[参照]** ボタンをクリックして、その場所を指定します。
6. **[OK]** をクリックします。InstallShield が選択したテンプレートを基にして、新しいプロジェクトを作成し、新しいプロジェクトは InstallShield で開かれます。

新しいプロジェクトは、事実上、プロジェクト テンプレートと同じですが、InstallShield が新しい製品コード (インストール プロジェクトの場合)、パッケージ コード、およびアップグレード コード (Windows Installer ベースのインストール プロジェクトの場合)、それにマージ モジュール プロジェクトの モジュール ID を生成する点が異なります。

プロジェクト テンプレートをリポジトリにパブリッシュする

別のプロジェクトで再利用、または他のユーザーと共有したい既存のテンプレートがある場合、リポジトリにそれをパブリッシュすることができます。

タスク ***テンプレートをリポジトリにパブリッシュするには、次の手順に従います。***

1. **[ファイル]** メニューで、**[新規]** をクリックします。**[新規プロジェクト]** ダイアログ ボックスが開きます。
2. **[すべてのタイプ]** タブをクリックします。
3. パブリッシュするテンプレートを右クリックしてから、**[テンプレートをリポジトリへパブリッシュする]** をクリックします。**パブリッシュ ウィザード**が開きます。
4. パブリッシュ ウィザードのパネルを完成します。

リポジトリにあるテンプレートに基づいてプロジェクトを作成しても、現在のプロジェクトと既存するリポジトリ テンプレートとの間にリンクはありません。テンプレートを変更してリポジトリへ再パブリッシュしても、テンプレートに基づいて作成したプロジェクトには影響しません。ただし、リポジトリにある更新済みのテンプレートに基づいて新しいプロジェクトを作成することはできます。

サンプル プロジェクト

InstallShield インストールには、いくつかのプロジェクト (.ism) ファイルの例が付属しています。これらのプロジェクトは、Samples フォルダーに格納されています。デフォルト保存先は次の場所です:

C:¥Program Files¥InstallShield¥2022¥Samples

これらのプロジェクトを見ると、インストール プロジェクトのある側面 (リリースフラグ等) をどのように導入するかが分かります。

プロジェクト アシスタント

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

InstallShield で用意されている プロジェクト アシスタントを使って、基本のインストール プロジェクトを素早く簡単に作成することができます。プロジェクト アシスタントはインストール プロジェクト作業のフレームワークを提供し、プロジェクト作成プロセスの手順を案内し、その関連情報を提供します。

プロジェクト アシスタントのしくみ

基本の MSI プロジェクトまたはトランスフォームを新しく作成すると、[プロジェクト アシスタント] ビューが自動的に開きます。

プロジェクト アシスタントで入力した情報は、基本となるプロジェクト ファイルに直接保存されます。したがって、インストールデザイナ(下記参照)に切替えて、InstallShield インストール開発環境(IDE) の機能をフル活用して情報を表示したり変更することができます.この場合、プロジェクト アシスタントを使用すると、必要に応じてインストール デザイナーを使用する高度なインストールのために基礎を作成します。

インストールデザイナとの統合

[インストール デザイナー] タブには、作成中のプロジェクトの種類に利用可能な InstallShield インターフェイスのビューがすべて表示されます。ここから、より複雑でパワフルな要素をインストール プロジェクトに追加することができます。プロジェクト アシスタントでインストール プロジェクトを作成してから、インストールデザイナー を使用してプロジェクト要素を微調整することができます。

インストール デザイナーとプロジェクト アシスタントは同時に実行されます。一方に変更を加えると、すぐにもう一方にその変更が反映されます。たとえば、[インストール デザイナー] タブを使用時にある機能を削除すると、その機能はインストール プロジェクトからなくなり、プロジェクト アシスタントでも表示されなくなります。

このセクションには、さらなる詳細を含む次のサブセクションが含まれています:

- プロジェクト アシスタントを使用する
- [アプリケーション情報] ページ
- [インストール要件] ページ
- [インストール アーキテクチャ] ページ
- [アプリケーション ファイル] ページ
- [アプリケーションのショートカット] ページ
- [アプリケーション レジストリ] ページ
- [インストール インタビュー] ページ
- [インストール ローカリゼーション] ページ
- [インストールのビルド] ページ

プロジェクト アシスタントを使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

インストール プロジェクトを新しく作成すると、プロジェクト アシスタントが自動的に開きます。ホーム ページには、インストール デザインのダイアグラムがあり、インストール作成の手順が一目で分かるようになっています。プロジェクト アシスタントを使ってプロジェクトを作成するか、[インストールデザイナ] タブをクリックして基本のインストール プロジェクトをさらに定義します。

プロジェクト アシスタントの [他のオプション]、[他の場所]、および [ヘルプ リンク] を使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト アシスタントの各ページにある左のコラムのリンクは、インストールの作成および情報の検索に使用します。

- **その他のオプション**—プロジェクト アシスタント ページにある特定部分に関する追加構成オプションを提供します。これらは、プロジェクト アシスタントの機能のための他より一般的ではないオプションです。
- **他の場所**—現在のプロジェクト アシスタントのページに対応するインストール デザイナー内のビュー。リンクをクリックすると、インストール デザイナが起動され、そのビューがアクティブになります。
- **ヘルプ リンク**—このリストは現在のプロジェクト アシスタント ページに関連するヘルプ トピックへのリンクを提供します。

プロジェクト アシスタント内を移動する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*



タスク ***別のプロジェクト アシスタントに移動するには、次のどれかを実行します。***

- 特定のページに直接移動するには、ページの下部にあるナビゲーション バーで表示されている該当するアイコンをクリックします。
- プロジェクト アシスタントの手順に従うには、以下の通りに実行します。
  - [次へ] または [戻る] 矢印ボタンをクリックして、前/後ろに移動します。
  - CTRL+TAB を押すと、次のページに移動し、CTRL+SHIFT+TAB を押すと、前のページに移動します。
- [ホーム] に戻ってインストール デザイン ダイアグラムを表示するには、ナビゲーション バーの [ホーム] ボタンをクリックします。

インストール デザイナーを開く



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[インストール デザイナー] タブには、InstallShield インストール開発環境 (IDE) のビューが表示されます。このタブでは、プロジェクト アシスタントでは実現不可能な、より複雑でパワフルな要素をインストール プロジェクトに追加することができます。インストール デザイナーでビューを開くには、[インストール デザイナー] タブをクリックします。



メモ- *インストール デザイナーとプロジェクト アシスタントは同時に実行されます。一方に変更を加えると、すぐにもう一方にその変更が反映されます。*

プロジェクト アシスタントの表示または非表示



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト アシスタントでは、簡素化されたインストール プロジェクトのタスクを見ることができます。インストール デザイナーでは、これらすべてのタスクに加え、プロジェクトをさらに高度にカスタマイズすることができます。

InstallShield プロジェクトの作成をインストール デザイナーのみを利用して行う場合、プロジェクト アシスタントを非表示にして、タブが InstallShield インターフェイスで表示されないようにすることができます。同様に、プロジェクト アシスタントが非表示になっている場合、それを選択で表示することもできます。



タスク

プロジェクト アシスタントの表示または非表示の選択は、以下の手順に従います:

[ビュー] メニューで、**[プロジェクト アシスタント]** をクリックします。

プロジェクト アシスタント コマンドの隣にチェック マークがあるとき、プロジェクト アシスタントのタブが InstallShield インターフェイスに表示されます。チェック マークが表示されていないとき、プロジェクト アシスタントは非表示になっています。

プロジェクト アシスタント コマンドの隣にチェック マークがないとき、インストール デザイナーが、プロジェクトを新規作成するときのデフォルト タブになります。

[アプリケーション情報] ページ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[アプリケーション情報] ページで、プロジェクトがインストールするアプリケーションに関する情報を指定することができます。これには、アプリケーション名とバージョン、会社名、Web サイト アドレス、およびアプリケーション アイコンがあります。

コントロール パネルの [プログラムの追加と削除]



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

コントロール パネルの [プログラムの追加と削除] には、コンピューター システムにインストールされているアプリケーションの一覧が表示されます。[プログラムの追加と削除] から特定のプログラムについての情報を表示したり、プログラムを追加、変更または削除することができます。

プロジェクト アシスタントの [アプリケーション情報] ページに提供する情報は、アプリケーションがインストールされた時にその [プログラムの追加と削除]の情報を完成する際に利用されます。

インストールの会社名と製品名



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

会社名と製品名は、インストール プロジェクトの数箇所で使用されます。

インストール プロジェクト内で会社名はどのように使われますか？

会社名は、アプリケーションのデフォルト インストール ディレクトリを設定するのに使用されます。また、エンド ユーザーのシステム上のアプリケーションのコントロール パネルにある [プログラムの追加と削除] でも使用されます。

インストール プロジェクト内で製品名はどのように使われますか？

製品名はアプリケーションの [プログラムの追加と削除] (サポート情報リンク) で使用されてます。これはデフォルト インストール ディレクトリの設定にも使用されます。

[インストール要件] ページ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[インストール要件] ページでは、ターゲット システムのインストールの要件を簡単に設定することができます。たとえば、アプリケーションに特定の OS が必要な場合、このページの最初の部分で示すことができます。

プロジェクト アシスタントでオペレーティング システムを指定する



プロジェクト: *プロジェクト アシスタントでオペレーティング の設定が可能なプロジェクトの種類は以下の通りです:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト アシスタントの [インストール要件] ページでオペレーティング を指定すると、InstallShield が起動条件を作成します。これらの条件は .msi ファイルの LaunchCondition テーブルに追加されます。このテーブルはダイレクトエディター のテーブルで表示および編集することが可能です。

InstallShield がオペレーティング システム起動条件を作成する方法

[インストール要件] ページでオペレーティングを指定すると、アプリケーションをサポートしないオペレーティング システムを除外することになります。

たとえば、最新 Windows オペレーティング システムのチェック ボックスのみを選択した場合、InstallShield は [インストール要件] ページで選択しなかったオペレーティング システムを除外する起動条件を作成します。この起動条件タイプでは、将来的にリリースされる Windows オペレーティング システムのバージョンが自動的にサポートされます。これは起動条件で除外されていないためです。

インストールによる要件確認のタイミング



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

必要なソフトウェアまたはオペレーティング システムがターゲット システムに確実に存在するよう、インストールはファイル転送を開始する前にインストールについてこれらの必要条件を確認します。

ソフトウェア要件のランタイム メッセージを変更する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

インストールにソフトウェア要件があり、ターゲット システムに選択されたソフトウェアがない場合、ランタイムのメッセージがインストール中に表示されます。表示されるメッセージは編集できます。



タスク ***ランタイム メッセージを編集するには、以下の手順を実行します。***

1. プロジェクト アシスタントで、**[インストール要件]** ページを開きます。
2. ソフトウエア要件に関する質問には、**[はい]** を選びます。
3. アプリケーションに必要なソフトウェアを選択します。デフォルトのランタイム メッセージが右側に表示されます。
4. ランタイム メッセージを編集します。

カスタム インストール要件の作成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[インストール デザイナー] で製品の "インストール条件" 設定を使って、カスタマイズされたインストール要件を作成できます。この設定に、インストールを起動する前に Windows Installer が評価しなくてはならない条件を作成することができます。詳細については、「製品の条件を設定する」を参照してください。

また、システム検索ウィザードを利用してターゲット システム上にある特定のファイル、フォルダー、レジストリキー、または .ini 値を検索することもできます。システム検索ウィザードを利用して、検索値を使った条件を作成することができます。

[インストール アーキテクチャ] ページ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[インストール アーキテクチャ] ページで、インストールでエンド ユーザーに表示する機能を指定することができます。機能とは、ユーザーから見て個別にインストール可能な最小の製品構成単位のことです。インストール中に [カスタム] セットアップ タイプを選択すると、個々の機能がエンドユーザーに対して表示されます。



メモ: *機能には、サブ機能やサブサブ機能などを含めることも可能で、セットアップ プログラムの要求に応じて必要なレベルの多層構造化が可能です。*

プロジェクト アシスタントで機能を追加する



プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*



タスク ***機能を追加するには、以下の手順を実行します。***

1. プロジェクト アシスタントで、**[インストール要件]** ページを開きます。
2. **インストール アーキテクチャをカスタマイズしますか?** の質問で、**[はい]** を選択します。
3. メイン機能を追加するには、**[インストール アーキテクチャ]** エクスプローラーをクリックします。サブ機能を追加するには、親機能となる機能をクリックしてから **[新規作成]** をクリックします。プロジェクト アシスタントは新しい機能を作成します。
4. 機能に名前を付けるか、後で **[名前の変更]** をクリックして名前を入力します。

複数機能インストールを作成するかどうかを判断する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

機能は、エンド ユーザーから見たインストールの構成ブロックです。このため、機能はインストール内部での機能上はっきりと区別できる要素を象徴しなくてはなりません。

アプリケーションに機能的に異なるブロックが複数存在する場合、複数機能インストールを作成しなくてはなりません。たとえば、インストールにアプリケーション (.exe ファイル) とヘルプ ライブラリ (.hlp ファイル) を含む場合、インストール プロジェクトは各要素につき 1 つの機能とした最低 2 つの機能を含まなくてはなりません。

プロジェクト アシスタント内での複数機能インストールの作成に関する詳しい情報は、「複数の機能を持つインストールを作成する.」をご覧下さい。

複数の機能を持つインストールを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

インストール デザイナー (IDE) のプロジェクト アシスタントを利用して、複数機能を持つインストールを作成することができます。

タスク ***プロジェクト アシスタントで複数機能インストールを作成するには、以下の手順を実行します。***

1. プロジェクト アシスタントで、**[インストール要件]** ページを開きます。
2. インストール アーキテクチャをカスタマイズするには **[はい]** を選択します。
3. **[インストール アーキテクチャ]** エクスプローラーをクリックしてから、**[新規作成]** をクリックします。プロジェクト アシスタントは新規機能を作成します。
4. F2 を押すか、機能を右クリックして、**[名前の変更]** を選択して新しい機能の名前を付けます。
5. 同じレベルに別の機能を追加するには、**[インストール アーキテクチャ]** エクスプローラーをクリックしてから **[新規作成]** をクリックします。サブ機能を作成するには、親機能となる機能をクリックしてから **[新規作成]** をクリックします。プロジェクト アシスタントは新しい機能を作成します。
6. 必要に応じて機能およびサブ機能を追加します。

インストールデザイナの機能を使った作業については、「機能の作成」を参照してください。

デフォルトの機能

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

*デフォルトの機能*の概念はプロジェクト アシスタントでのみ存在します。インストール プロジェクトに追加されたすべてのリソース (たとえば、ファイルまたはレジストリデータ) は機能に割り当てる必要があります。リソースが機能に割り当てられていない場合、それらは実行時にターゲット システムへインストールされません。

デフォルトの機能を利用すると、プロジェクト アシスタントでのオーサリング作業が簡素化されます。プロジェクトリソースが確実にインストールされるよう機能への割り当てに気を配る必要はありません。レジストリデータを追加するとき、新規ショートカットを作成するとき、またはすべてのアプリケーションデータが選択された時にファイルを追加するとき、これらすべてのリソースはデフォルト機能へ追加されます。

デフォルトの機能の設定

デフォルトの機能は、プロジェクト アシスタントの [インストール アーキテクチャ]ページで設定することができます。

機能が存在しない、またはデフォルトの機能が選択されていない場合はどうなりますか?

[インストールアーキテクチャ]ページへ移動したとき、および [アプリケーションファイル]、[アプリケーションショートカット]、または [アプリケーションレジストリ] ページにデータを追加したとき、InstallShield は 最初のルート機能をデフォルトの機能として選択します。機能が存在しない場合、InstallShield がサイレントで作成します。

機能の階層を定義する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

最上位の機能とは、機能階層の最も上にある機能です。最上位の機能にはインストールするアプリケーション、ヘルプ ライブラリ機能、そしてサンプル プロジェクト機能などを含みます。

最上位の機能の下には、サブ機能または *子*機能があります。これはインストールの都合上、別の機能に依存している機能です。*親*機能 (最上位の機能) がターゲット システムにインストールされないと、子機能もインストールされません。

[アプリケーション ファイル] ページ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[アプリケーションファイル] ページでは、機能に関連付けるファイルを指定することができます。

プロジェクト アシスタントで機能にファイルを追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*



タスク ***ファイルを機能に追加するには、以下の手順を実行します。***

1. プロジェクト アシスタントで、**[アプリケーション ファイル]** ページを開きます。
2. ページ上部にある機能リストで、ファイルが含まれる機能を選択します。
3. **[インストール先コンピューター]** エクスプローラーで、ファイルを追加するフォルダーを選択します。
4. **[ファイルの追加]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
5. 追加するファイルを参照します。
6. **[開く]** をクリックして、選択した機能にファイルを追加します。「追加したファイルには依存関係がある可能性があります」というメッセージが表示されます。
7. インストール プロジェクトに依存関係を自動的に追加するようにする場合は、**[はい]** をクリックします。

プロジェクト アシスタントで機能からファイルを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*



タスク ***ファイルを機能から削除するには、以下の手順を実行します。***

1. プロジェクト アシスタントで、**[アプリケーション ファイル]** ページを開きます。
2. 削除するファイルをクリックして、**Delete** キーを押します。

固定のフォルダーの場所へファイルを追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

ターゲット システム上のどの場所にプロジェクトファイルをインストールするのかが明白な場合、固定フォルダーの場所へのパスをハードコードで入力することができます。

タスク ***プロジェクト アシスタントでファイルを固定フォルダーの場所へ追加するには、以下の手順を実行します。***

1. プロジェクト アシスタントで、**[アプリケーション ファイル]** ページを開きます。
2. **[インストール先コンピューター]** を右クリックして、**[新しいフォルダー]** を選択します。
3. 新しいフォルダーの名前には、たとえば C: の様にインストール先を含むドライブ名を入力します。
4. ドライブ名フォルダーの下にサブフォルダーを追加して、詳しいインストール先のパスを定義します。

追加の定義済みフォルダーを表示する

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト アシスタントの [アプリケーション ファイル] ページには、一般的によく利用される定義済みフォルダーが表示されます。このページで定義済みフォルダーを表示または非表示にすることができます。

タスク ***追加定義済みフォルダーを表示するには、以下の手順に従います:***

1. プロジェクト アシスタントで、**[アプリケーション ファイル]** ページを開きます。
2. **[インストール先コンピューター]** を右クリックして、**[定義済みフォルダーを表示]** を選択します。
3. 定義済みフォルダーのリストで表示するフォルダーを選択します。

ヒント: *定義済みフォルダーを非表示にするには、定義済みフォルダーのリストで該当するものについて選択を解除します。*

[アプリケーションのショートカット] ページ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[アプリケーション ショートカット] ページでは、ターゲット システムのデスクトップ、または [スタート] メニュー上にあるアプリケーション ファイルのショートカットを指定できます。デフォルトで、このページは [プロジェクト アシスタント] を使ってプロジェクトに追加された各実行可能ファイルのショートカットを表示します。これらは削除することができ、インストール プロジェクトに入れたショートカットを他のファイルに追加することもできます。

ファイル拡張子



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

ファイル名拡張子の関連付け、つまりファイルの関連付けは、特定の種類のファイルを開くのにどのアプリケーションを使用すべきかを Windows に指示するためのレジストリ設定です。たとえば、Windows は通常 Windows のメモ帳を使用してテキストファイル (.txt の拡張子を持つファイル) を開き、Microsoft ペイントを使用してビットマップファイル (.bmp 拡張子を持つファイル) を開きます。

ファイル拡張子を使って、ファイルにアクセスせずにファイルの種類を識別することができます。ファイル名の末尾に接尾辞 (.abc) が追加されます。ファイル拡張子はまた、別のアプリケーションが拡張子に基づいてそのファイルと互換性があるかどうか (たとえば、そのファイルを開くのか変更するのか) を識別するのにも便利です。

InstallShield では、[コンポーネント] ビューまたは [セットアップのデザイン] ビューでコンンポーネントの [詳細設定] 領域にある設定を構成して、独自のファイル拡張子を登録できます。ファイル拡張子を登録すると、エンド ユーザーがファイルをクリックしたとき、ターゲット マシンのオペレーティング システムが特定のファイル拡張子を持つファイルを特定のアプリケーションで開くように指示することができます。

インストールに含まれていないファイルへのショートカットを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

既にターゲット システム上に存在するファイルを対象としたショートカットを作成するようにインストールを設定できます。このファイルはインストール プロジェクトに含まれている必要はありません。

タスク ***インストールに含まれていないファイルへのショートカットを作成するには、以下の手順を実行します。***

1. インストール デザイナーを開きます。
2. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
3. **[ショートカット]** エクスプローラで、ショートカットのインストール先を右クリックしてから、**[既存ファイルへの新しいショートカット]** をクリックします。**[ショートカット ターゲットの参照]** ダイアログが開きます。
4. ターゲット ファイルの場所を参照して **"ファイル名"** 設定にファイルの名前を入力します。
5. **[OK]** をクリックします。
6. ショートカットの設定を構成します。

プロジェクト アシスタントでデフォルトのショートカットを変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

タスク ***デフォルトのショートカットを変更するには、以下の手順に従います:***

1. **プロジェクト アシスタント**で、**[アプリケーション ショートカット]** ページを開きます。
2. 変更するショートカットを選択します。
3. 必要に応じて変更を行います。

プロジェクト アシスタントでショートカットのターゲットにファイル拡張子を関連付ける

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

ファイル拡張子とショートカットのターゲットを関連付けることができます。これを行うとき、Windowsはターゲット ファイルを使用し、指定された拡張子のファイルを開きます。たとえば、.txt と入力すると、エンドユーザーが .txtファイルを開く際、このショートカットのターゲットファイルが起動して開きます。



タスク ***ショートカットのターゲットをファイル拡張子に関連付けるには、以下の手順に従います:***

1. **プロジェクト アシスタント**で、**[アプリケーション ショートカット]** ページを開きます。
2. ショートカットをクリックして、ショートカット オプションを有効にします。
3. **[ショートカットとファイル拡張子を関連付けるオプション]** を選択します。
4. このショートカットのターゲットと関連付けるファイル拡張子を入力します (例、txt)。カンマで区切って複数の拡張子を追加することもできます。

[アプリケーション レジストリ] ページ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[アプリケーション レジストリ] ページでは、アプリケーションに必要なレジストリ データを指定できます。

レジストリの更新



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

レジストリはコンピューターの構成情報が入ったデータベースです。コンピューターのレジストリに情報にはユーザープロフィール、コンピューターにインストールされたハードウエアおよびソフトウェア、プロパティ設定が含まれます。

アプリケーションに必要なレジストリデータの調べ方

アプリケーション開発者にレジストリ情報をたずねてください。特に、インストールするアプリケーションにユーザー固有 (HKEY_CURRENT_USER) またはマシン固有 (HKEY_LOCAL_MACHINE) の設定が必要かどうかを知る必要があります。

開発者からインストールに追加する .reg ファイルを受け取ります。InstallShield では、.reg ファイルをインストール プロジェクトにインポートすることができます。

プロジェクト アシスタントでレジストリ データを構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*



タスク ***レジストリ データを構成するには、以下の手順に従います。***

1. **プロジェクト アシスタント**で、**[アプリケーションのレジストリ]** ページを開きます。
2. レジストリ データの構成についての質問に対して、**[はい]** を選択します。
3. データを追加するレジストリ アイテムを右クリックして、**[新規]** を選択してから **[キー]** をポイントします。
4. キーに名前を入力します。
5. キーを右クリックし、[**新規**]を選択してから適切なコマンドをポイントします。登録するデータのタイプによって、[デフォルト値]、[文字列値]、[バイナリ値]、[DWORD 値]、[拡張可能文字列値] または [複数行値] から選択します。

プロジェクト アシスタントでレジストリ データの値を変更する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*



タスク ***レジストリ データを変更するには、以下の手順に従います。***

1. **プロジェクト アシスタント**で、**[アプリケーションのレジストリ]** ページを開きます。
2. データをダブルクリックする。

複数行文字列をダブルクリックすると、**[複数行文字列値]** ダイアログ ボックスが開きます。別の種類のレジストリ データをダブルクリックすると、**[データの編集]** ダイアログ ボックスが開きます。

3. データを編集して、**[OK]** をクリックします。

機能にレジストリ データを関連付ける

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト アシスタントの [アプリケーション レジストリ] ページに追加したすべてのレジストリ データがプロジェクトのデフォルトの機能へ追加されます。インストール デザイナーでレジストリ データを別の機能に関連付けることができます。

タスク

インストール デザイナーを使って、レジストリ データをデフォルトの機能以外の機能に関連付けるには、以下の手順を実行します:

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. [**ビュー フィルター**] リストで、レジストリ データを関連付ける機能を選択します。
3. レジストリ データを適切なレジストリの場所で作成するか、またはドラッグアンドドロップします。

レジストリ データで変数データ型を使用する

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

InstallShield ではインストール プロジェクト用レジストリデータを作成する際に、変数データ型またはプロパティを利用することができます。

タスク

レジストリで INSTALLDIR を変数として利用するには、以下の手順を実行します(Windows Installer ベースのプロジェクトのみ)。

1. **プロジェクト アシスタント**で、**[アプリケーションのレジストリ]** ページを開きます。
2. **[はい]** を選択して、アプリケーションがインストールするレジストリ データを構成することを示します。

3. HKEY_CLASSES_ROOT を右クリックし、**[新規]** をポイントしてから、**[キー]** をクリックします。
4. キーに [インストール先] という名前を付けます。
5. [インストール先] キーを右クリックし、**[新規作成]** をポイントしてから、**[文字列値]** をクリックします。
6. 文字列値に My Installation Location と名前を付けます。
7. My Installation Location キーをダブルクリックします。**[データの編集]** ダイアログ ボックスが開きます。
8. **"値データ"** フィールドに **[INSTALLDIR]** と入力します。

実行時に、[INSTALLDIR] の値はインストール ディレクトリと置き換えられます。

アプリケーション パス



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

アプリケーション パスのレジストリ キーには、指定アプリケーションの .dll ファイル用のプライベート検索パスとして Windows が使用するデータが含まれます。アプリケーションの .dll ファイルを PATH 環境変数で指定されていないディレクトリ (および、アプリケーションのディレクトリ以外) にインストールする場合は 、インストール中に .dll ファイル ディレクトリを含む適切なアプリケーション パスを設定するようにしてください。アプリケーション パス情報は、レジストリの HKLM¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥App Paths¥AppName.exe の下に格納されます。

[インストール インタビュー] ページ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *MSI データベース*

[インストール インタビュー] ページでは、インストール プログラムの実行時にエンド ユーザーに表示するダイアログ ボックスを選択します。このページでの回答によって、プロジェクト アシスタントは対応するダイアログをインストール プロジェクトに追加します。

プロジェクト アシスタントでインストールに使用するダイアログを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *MSI データベース*

InstallScript プロジェクトでは、インストール完了時にアプリケーションを起動するオプションは使用できません。

プロジェクト アシスタントの [インストール インタビュー] ページに表示される質問に答えると、作成したインストール プログラムが実行されるときにエンド ユーザーに対して表示されるダイアログを指定することができます。

タスク ***インストールのダイアログを指定するには、次の手順に従います。***

1. **使用許諾契約ダイアログを表示しますか?**—**[はい]** を選択して、使用許諾契約書ファイルを参照します。
2. **ユーザーに対して会社名とユーザー名の入力をプロンプトしますか?** — **[はい]** を選択してこの情報を要求するダイアログを表示します。
3. **ユーザーがアプリケーションのインストール場所を変更することができるようにしますか?** — **[はい]** を選択して、エンド ユーザーがインストールの場所を変更できるようにします。詳細については、「エンドユーザーによるインストール先の変更を許可する」を参照してください。
4. **ユーザーがアプリケーションの特定部分のみ選択してインストールできるようにしますか?** — **[はい]** を選択して、エンド ユーザーがアプリケーションの特定部分のみインストールできるようにします。詳細については、「インストールが選択できるインストールを作成する」を参照してください。
5. **インストールの完了時にアプリケーションを起動するオプションをユーザーに提供しますか?**—**[はい]** を選択して、アプリケーション ファイルを参照します。このオプションが **[はい]** に設定されると、最後のダイアログにはチェック ボックスがあり、エンドユーザーが **[完了]** ボタンを押すとアプリケーションが直ちに起動します。

ヒント: *カスタム グラフィックをインストール ダイアログに追加するには、[他のオプション] セクションでリンクをクリックし、[ダイアログイメージ] ダイアログ ボックスを起動します。*

エンドユーザーによるインストール先の変更を許可する

エンドユーザーがシステム上でソフトウェアをインストール場所を選べるように、インストール場所の変更を許可することができます。

Windows Installer ベースのインストールでは、Windows Installer プロパティ INSTALLDIR がデフォルト インストール ディレクトリとなります。ユーザーがインストール場所を変更できるようにすると、インストール中に CustomSetup ダイアログが表示されます。

InstallScript プロジェクトでは、TARGETDIR がデフォルトインストールディレクトリとなります。ユーザーがインストール場所を変更できるようにすると、インストール中に SdSetupType2 ダイアログが表示されます。

使用許諾契約書

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *MSI データベース*

製品をインストールするために、エンド ユーザーが特定の法的要件に同意しなくてはならない場合があります。たとえば、ほとんどのソフトウェア ベンダはエンド ユーザーによるソフトウェアのコピーや第三者への配布を禁止しています。

インストールは実行時に、[使用許諾契約] ダイアログでエンド ユーザー使用許諾契約書 (EULA) を表示できます。EULA は製品の使用について、エンドユーザーとの間に交わされる法的契約です。

[使用許諾契約書] ダイアログは、使用許諾契約書テキストを表示するとともに、ラジオボタングループ ([はい] または [いいえ]) を含みます。エンドユーザーが EULA に同意しない場合、ソフトウェアはインストールされることなくインストール処理が終了します。



タスク

プロジェクト アシスタントで [使用許諾契約書] ダイアログをプロジェクトへ追加するには、以下の手順を実行します。

1. **プロジェクト アシスタント**で、**[インストール インタビュー]** ページを開きます。
2. **[はい]** を選択して **[使用許諾契約]** ダイアログの追加を指定します。
3. 使用許諾契約書ファイルへのパスを入力する、またはファイルを参照します。



プロジェクト: *基本の MSI と MSI データベース プロジェクトでは、このファイルはリッチ テキスト ファイル (.rtf) でなくてはなりません。*

InstallScript プロジェクトでは、どのライセンス ダイアログを使用するのか、またどのパラメーターに渡すのかによって、ファイルの種類が異なります。SdLicenseEx および SdLicense2Ex 関数は、.rtf ファイルとテキスト ファイル (.txt) をサポートします。

インストールが選択できるインストールを作成する

エンドユーザーがインストールする部分を選択してシステムにインストールするようなインストールを作成することができます。これは、インストール内で使用できる機能の一覧を表示するカスタムインストールです。エンドユーザーは実行時に表示されるダイアログでインストールする機能を選択することができます。

たとえば、インストールの中にアプリケーションの実行可能ファイル(.exe)ファイル、ドキュメント(.chm)ファイル、およびサンプルファイルが入っているとします。これらのファイルはすべて異なる機能の中に入っていて、オプション一覧からエンドユーザーに提供されます。エンドユーザーがアプリケーションのみを必要とする場合は、実行可能ファイルだけをインストールし、ドキュメント ファイルとサンプル ファイルはインストールしないように選択することができます。

[インストール ローカリゼーション] ページ



エディション: *複数言語サポートは、InstallShield の Premier Edition で使用することができます。*



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

[インストールのローカライズ] ページで、インストールが対応する言語を指定することができます。文字列値と関連識別子を指定して、エンド ユーザーのインターフェイスで簡単に他の言語にローカライズすることができます。



メモ: *ドロップダウンメニューで編集する文字列データセットを指定できます(ページ上部)。*

プロジェクト アシスタントから文字列データをローカライズする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

インストール プロジェクトのテキストの多くは文字列エントリとして保存されます。インストールのローカライズを簡単にするため、InstallShield では文字列をエクスポートして翻訳し、翻訳した文字列をプロジェクトにインポートする機能があります。



タスク ***プロジェクト内のスクリプト ファイルすべてを翻訳用にエクスポートするには、以下の手順に従います:***

1. **プロジェクト アシスタント**で、**[インストール のローカリゼーション]** ページを開きます。
2. ページ上部にあるデータ リストで、**[すべての文字列データ]** を選択します。
3. 言語一覧の中から、言語を選択します。
4. **[他のオプション]** 領域で、**[ローカライズ文字列のエクスポート]** をクリックします。**[ファイル名]** ダイアログ ボックスが開きます。
5. 任意の場所にテキスト (.txt) ファイルを保存します。
6. テキスト ファイルを翻訳会社または担当部署に渡します。



タスク ***テキストファイルの文字列の翻訳終了後、その文字列をプロジェクトに再インポートすることができます。***

1. **プロジェクト アシスタント**で、**[インストール のローカリゼーション]** ページを開きます。
2. ページ上部にあるデータ リストで、**[すべての文字列データ]** を選択します。
3. 言語一覧で、インポートする文字列と一致する言語をクリックします。

4. **[他のオプション]** 領域で、**[ローカライズ文字列のインポート]** をクリックします。**[ファイル名]** ダイアログ ボックスが開きます。
5. .txt ファイルを参照して、**[開く]** をクリックします。翻訳済み文字列がプロジェクトに再インポートされます。

インストールでローカライズされた文字列データの使用のされ方

インストール プロジェクトには、インストール専用の文字列データとアプリケーション専用の文字列データの 2 種類の文字列データがあります。

InstallShield プロジェクトでの大半の文字列データは、エンドユーザーがインストールを実行するときにらいタイムで使用されます。但し、文字列データの一部はターゲット システムインストールされます。たとえば、ローカライズされたショートカットは、エンド ユーザーのシステムにインストールされるローカライズされた文字列データです。

[インストールのビルド] ページ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*



ヒント: *以下の情報は、(Visual Studio との統合を行わず) InstallShield 内部でビルドされたリリースに適用します。Visual Studio 内部から InstallShield リリースをビルドする方法については、「Microsoft Visual Studio でリリースをビルドする」を参照してください。*

[インストールのビルド] ページでは、ビルドするタイプを指定し、再配布可能ファイルをコピーする場所も指定することができます。パッケージにデジタル署名を行うこともできます。

プロジェクト アシスタントからインストールをビルドする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*



ヒント: *以下の情報は、(Visual Studio との統合を行わず) InstallShield 内部でビルドされたリリースに適用します。Visual Studio 内部から InstallShield リリースをビルドする方法については、「Microsoft Visual Studio でリリースをビルドする」を参照してください。*



タスク ***インストールをビルドするには次の手順を実行します。***

1. **プロジェクト アシスタント**で、**[インストールのビルド]** ページを開きます。
2. インストールのイメージの種類を選択します。

   メモ・ *Windows Installer が既にインストールされているマシンには、Windows Installer の存在をチェックし、またそれが無かった場合にエンジンをインストールすることがない* ***"単一の MSI パッケージ"*** *オプションを選択します。*
3. InstallShieldで、ビルド後に自動的にインストールを別の場所にコピーする場合、各ビルド オプションの**オプション配布の設定** リンクをクリックして、場所を指定します。

   ビルド後に InstallShield がインストールを配布するように設定する場合、各ビルドオプションの **オプション配布**の設定 リンクをクリックして、**ビルド後に配布** をチェック ボックスを選択します。
4. Setup.exe ファイルにデジタル署名を行い、アプリケーションに含まれるコードが変更または破損していないことをエンド ユーザーに対して保証するには、**[セットアップにデジタル署名する] ハイパーリンク** をクリックします。[**セットアップにデジタル署名する**] ダイアログ ボックスが開きます。必要に応じて設定を指定します。
5. **[インストールのビルド]** をクリックします。

[出力] ウィンドウが開き、[ビルド] タブにビルドの進行状況についての情報が表示されます。[ビルド] タブがログファイル情報を表示すると同時にビルドが完了します。



ヒント・ *iインストール デザイナーの [署名] タブでは、ビルド時に、インストールのどの部分に対してデジタル署名を行うかを指定できます。InstallShield は、作業中のプロジェクトの種類に応じて、リリースに含まれる次の任意およびすべてのファイルに署名することができます。*

- *基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトの Windows Installer パッケージ (.msi ファイルと .msm ファイル)*
- *基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの* Setup.exe *ファイル*
- *InstallScript プロジェクトのメディア ヘッダー ファイル*
- *InstallScript プロジェクトのパッケージ (自己展開型実行可能ファイル)*
- *リリースの任意のファイル (アプリケーション ファイルを含む)*

詳細については、「ビルド時にリリースとそのファイルにデジタル署名を行う」を参照してください。



Windows ロゴ・ *インストールのすべての実行可能ファイル (.exe、.dll、.ocx、.sys、.cpl、.drv、および .scr ファイル) は、Windows ロゴ プログラムに準拠するためにデジタル署名が必要です。*

プロジェクト アシスタントの完了後: 次のステップ



プロジェクト・ *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト アシスタントの一連のページでフィールドへの入力を完了すると、インストール プロジェクトの枠組みが完成します。これは機能するインストールとして利用することができ、また必要に応じてさらにカスタマイズすることも可能です。

プロジェクトをさらにカスタマイズする

インストールデザイナから、InstallShield で利用可能なすべてのインストール作成ビューへ簡単にアクセスすることができます。[プロジェクト アシスタント] ワークスペースの上部にある [インストールデザイナ] タブをクリックしてビューを表示します。

インストール デザイナー内には、作成中のプロジェクトの種類で利用可能なビューが一覧となった [ビュー リスト] があります。ワークスペースの左側に [ビュー リスト] を表示するには、F4 を押します。

InstallShield ビューについて詳しい情報

InstallShield の各ビューに関する詳細は、InstallShield ヘルプライブラリの [ビューリスト] セクションにあるトピックを参照してください。

InstallShield インターフェイスを使って作業する

InstallShield インターフェイスは、メニュー バー、ツールバー、ダイアログ ボックスなど通常の Windows ベースの要素をもったグラフィックなユーザー インターフェイスです。このセクションでは、これらの要素を使用した基本的なタスクの実行方法、および、インターフェイスのカスタマイズの方法について説明します。さらなる詳細は、次のセクションに分けて説明されています:

- ビュー リストを表示する
- InstallShield ユーザー インターフェイスでビューを開く
- 様々なビューで、[グループ ボックス] 領域を使って作業する
- ツールバーの表示または非表示
- ツールバーにボタンおよびメニューを追加する
- ツールバーからボタンおよびメニューを削除する
- カスタム ツールバーの作成
- [出力] ウィンドウを固定する/取り外す
- 様々なビューで、[スクリプト エディター] ペインを使って作業する

ビュー リストを表示する

タスク ***InstallShield インターフェイスでビュー リストを表示するには、以下の手順に従います:***

1. InstallShield インターフェイス上部の **[インストール デザイナー]** タブをクリックします。
2. 以下のいずれかの方法でビュー リストを表示します。
   - **[ビュー]** メニューで、**[ビュー リスト]** をクリックします。InstallShield インターフェイスの左側に、**ビュー リスト**が表示されます。
   - ツールバーの **[ビュー リスト]** ボタンをクリックします。
   - F4 を押すと、**[ビュー リスト]** が表示または非表示になります。

InstallShield ユーザー インターフェイスでビューを開く

InstallShield での多くの手順では、まず最初にインストール開発環境 (IDE)で特定のビューを開きます。

タスク ***ビューを開くには、以下の手順を実行します。***

1. **[インストール デザイナー]** タブをクリックします。IDE の左側に**ビュー リスト**が表示されます。**ビュー リスト** が表示されない場合、「ビュー リストを表示する」を参照してください。
2. **ビュー リスト**で、開くビューを選択します。使用できるすべてのビューを表示するには、[ビュー リスト] フォルダーを展開します。

様々なビューで、[グループ ボックス] 領域を使って作業する

InstallShield の多くのビューには、ビュー内の行をグループ分けすることができるグループ ボックス領域があります。このグループ ボックスを含むビューでは、行ヘッダーをグループ ボックス ([フィールドごとにグループ分けするときは、ここにフィールド名をドラッグします。] と表示される領域) にドラッグするするだけで、複数階層のグループに分けることができます。ビュー内の行は、グループ ボックスの配列に従って階層構造で表示されます。グループ ボックスを含むビューには、[文字列エディター]、[プロパティ マネージャー]、[再配布可能ファイル]、[前提条件]、および [パス変数] ビューなどがあります。

グループ ボックスを使用する際、以下の点を参考にしてください:

- 列ヘッダーをグループ ボックス領域に移動するとき、列ヘッダーをドラッグしてグループ ボックスにドロップします。
- 列ヘッダーをグループ ボックス領域にコピーするとき、CTRL キーを押しながら列ヘッダーをドラッグしてグループ ボックスにドロップします。この場合、列ヘッダーはそのままの位置に残り、またグループ ボックスにも表示されます。
- グループ ボックスのヘッダーをグループ ボックス領域にドロップするとき、別の列ヘッダー上にドロップすることができます。これによって、行が階層構造で表示されます。

- 列ヘッダーをグループ ボックスから削除するには、グループ ボックス領域からそれをドラッグして列ヘッダーの行にドロップします。列ヘッダーの行をドラッグすると、それをドロップしたときに列ヘッダー行のどの位置に表示されるのかが、矢印で示されます。
- グリッド内のアイテムを特定の列ヘッダーごとに並べ替えるには、グループ ボックス内または列ヘッダー行内の列ヘッダーをクリックします。

次の例を使って、ビュー内のコンテンツをグループ ボックスを使ってグループ化する様々な方法をデモンストレーションします。

デフォルトの動作: グループ ボックス領域を空白にする

デフォルトで、グループ ボックスに列ヘッダーは表示されません。次のスクリーン ショットは、[再配布可能ファイル] ビューの一部です。項目が [名前] 列で整列しています。

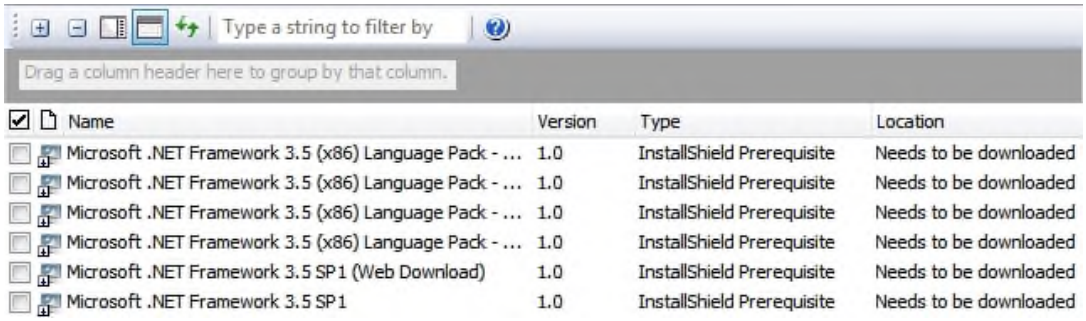


図 3: [グループ ボックス] 領域を空白にする

1 つの列ヘッダーでグループ化する

CTRL を押しながらグループ ボックスに列ヘッダーをドラッグ アンド ドロップするとき、グリッドの行が項目ごとにグループ化されます。次のスクリーン ショットは、[文字列エディター] ビューの一部です。翻訳が必要な最近追加および変更された行が識別しやすくグループ化されます。

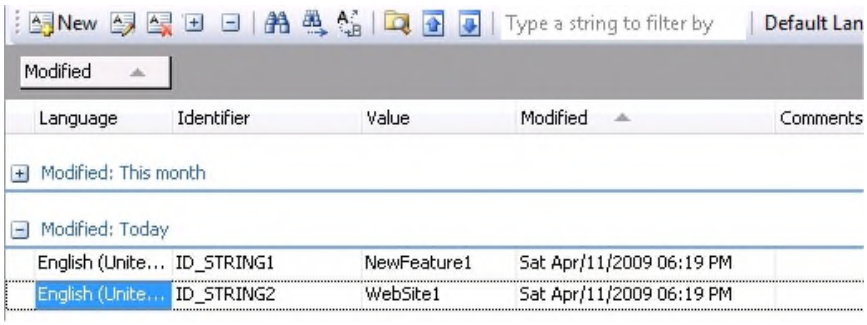


図 4: 1 つの列ヘッダーで行をグループ化する

2 つの列ヘッダーでグループ化する

CTRL を押しながらグループ ボックスに列ヘッダーをドラッグ アンド ドロップするとき、グリッドの行が複数の項目ごとにグループ化されます。次のスクリーン ショットは、[再配布可能ファイル] ビューの一部です。プロジェクトに追加された InstallShield 前提条件とマージ モジュールが、識別しやすくグループ化されます。

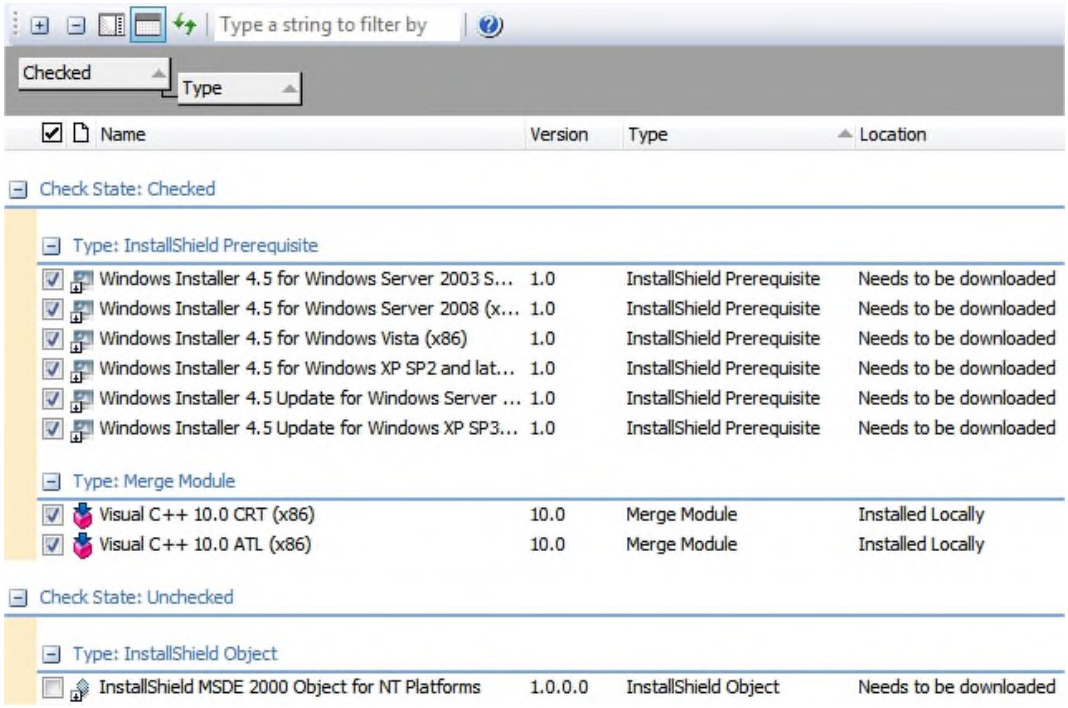


図 5: チェック ボックス列および種類列ごとに行を並べ替える

ツールバーの表示または非表示

タスク ***ツールバーを表示または非表示にするには、以下の手順を実行します。***

- ツールバーを右クリックして表示する、または非表示にするツールバーを選択します。
- **[ツール]** メニューで、**[カスタマイズ]** をクリックします。**[カスタマイズ]** ダイアログ ボックスが開きます。表示する各ツールバーのチェック ボックスを選択します。非表示にする各ツールバーのチェック ボックスを選択解除します。

ツールバーにボタンおよびメニューを追加する

タスク ***ツールバーにボタンまたはメニューを追加するには、以下の手順を実行します。***

1. 変更するツールバーが表示されていることを確認します。
2. **[ツール]** メニューで、**[カスタマイズ]** をクリックします。**[カスタマイズ]** ダイアログ ボックスが開きます。
3. **[コマンド]** タブをクリックします。
4. **[カテゴリ]** ボックスで、追加するボタンまたはメニューのカテゴリをクリックします。
5. **[コマンド]** ボックスから、ボタンまたはメニューを適切なツールバーへドラッグします。



ヒント: 独自のカスタム ツールバーを作成するには、ボタンまたはメニューをツールバーの近くにある灰色の空白部分にドラッグします。

ツールバーからボタンおよびメニューを削除する



タスク ***ツールバーからボタンまたはメニューを削除するには、以下の手順を実行します。***

1. 変更するツールバーが表示されていることを確認します。
2. **[ツール]** メニューで、**[カスタマイズ]** をクリックします。**[カスタマイズ]** ダイアログ ボックスが開きます。
3. 削除するボタンまたはメニューを右クリックして、**[削除]** をクリックします。

カスタム ツールバーの作成



タスク ***カスタム ツールバーを作成するには、以下の手順を実行します。***

1. **[ツール]** メニューで、**[カスタマイズ]** をクリックします。**[カスタマイズ]** ダイアログ ボックスが開きます。
2. **[ツールバー]** タブをクリックします。
3. **[新規]** ボタンをクリックします。**[新規ツールバー]** ダイアログ ボックスが開きます。
4. **[ツールバーの名前]** テキスト ボックスにツールバーの説明的な名前を入力し、**[OK]** をクリックします。
5. メニューまたはボタンを追加して、新規作成したツールバーをカスタマイズします。

[出力] ウィンドウを固定する/取り外す

[出力] ウィンドウ、またその個別のタブは、InstallShield のワークスペースの任意の側面に固定するか、独立した場所にドラッグすることができます。

[出力] ウィンドウまたはそのタブの 1 つを InstallShield インターフェイスの端にドラッグすると、固定ウィンドウとして表示されます。[出力] ウィンドウまたはそのタブの 1 つを InstallShield インターフェイスの端から離れた場所にドラッグすると、取り外されます。



タスク ***[出力] ウィンドウを取り外すには、以下の手順に従います:***

[出力] ウィンドウのタイトル バーを新しい場所にドラッグします。必要に応じて、[出力] ウィンドウのサイズを変更します。



タスク ***[出力] ウィンドウを固定するには、以下の手順に従います:***

[出力] ウィンドウのタイトル バーを InstallShield インターフェイスの右、左、上、下の端にドラッグします。



タスク ***[出力] ウィンドウのタブを取り外すには、以下の手順に従います:***

タブを新しい場所にドラッグします。必要に応じて、[出力] ウィンドウのサイズを変更します。



タスク ***[出力] ウィンドウのタブの 1 つを固定するには、以下の手順に従います:***

タブを InstallShield インターフェイスの右、左、上、下の端にドラッグします。

様々なビューで、[スクリプト エディター] ペインを使って作業する

InstallShield のいくつかのビューにはスクリプト エディター ペインがあり、そこにプロジェクトのコードを書き込むことができます。以下は、スクリプトエディター ペインを持つビューの例です:

- **[InstallScript] ビュー** – このビューでスクリプト ファイルを選択すると、InstallScript コードを書き込める [スクリプト エディター] ペインが表示されます。
- **[SQL スクリプト] ビュー** – このビューで SQL スクリプトを選択すると、[スクリプト] タブに SQL ファイルを編集できる [スクリプト エディター] ペインが表示されます。
- **[カスタム アクションとシーケンス] ビュー** – このビューで VBScript または JScript カスタム アクションを選択すると、[スクリプト] タブに VBScript または JScript ファイルを編集できる [スクリプト エディター] ペインが表示されます。

このセクションでは、InstallShield のスクリプト エディターの使い方について説明します。

スクリプト エディターでブックマークを使用する

ブックマークによって、スクリプト内の特定の行へ最小のキーストロークを使ってジャンプできます。ブックマークは、スクリプト エディターの左マージンに表示されます。InstallShield では、プロジェクトを閉じた時、ブックマークは保存されませんので注意してください。



タスク ***新しいブックマークを追加したり、既存のブックマークをクリアするには、以下の手順に従います:***

1. 以下のいずれかを実行します。
   - ブックマークを追加するスクリプト内の行に挿入ポイントを置きます。
   - クリアするブックマークを含むスクリプト内の行に挿入ポイントを置きます。
2. ALT+K を押します。

タスク ***挿入ポイントをブックマークが含まれる次の行に移動するには、以下を実行します:***

CTRL+K を押します。

タスク ***挿入ポイントをブックマークが含まれる前の行に移動するには、以下を実行します:***

CTRL+SHIFT+K を押します。

スクリプト エディターでオートコンプリート機能を有効または無効にする

InstallShield 様々なビュー (たとえば、[InstallScript] ビューや [SQL スクリプト] ビュー) のスクリプト エディターに入力するときに、オートコンプリート機能を使用するかどうかを指定できます。オートコンプリート機能を有効にすると、スクリプト エディター内に入力している文字で始まる関数、キーワード、定数、文字列識別子の一覧がアルファベット順でポップアップ リストに表示されます。用語全体を手入力する代りに、そのポップアップ リストから選択すると、InstallShield によってその用語がスクリプトに追加されます。

オートコンプリートを使うと、コードを手入力する手間が省けるため、作業効率が上がります。また、コードのスペル ミスを回避することができます。

ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、同じオートコンプリートの設定が、すべてのスクリプト エディターで使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*

タスク ***オートコンプリート機能をスクリプト エディターで有効にするかどうかを指定するには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"オートコンプリート"** 設定で、適切なオプションを選択します:
   - **はい** ― オートコンプリート機能を有効にすると、スクリプト エディター内に入力している文字で始まる関数、キーワード、定数、文字列識別子の一覧がアルファベット順でポップアップ リストに表示されます。デフォルトでは、これが設定されています。
   - **いいえ** ― スクリプト エディター内にコードを入力しているとき、ポップアップ リストを表示しません。
3. **"ローカル変数を含める"** 設定で、適切なオプションを選択します:
   - **はい** ― InstallScript エディター内に入力しているコードの文字で始まる使用可能な関数およびその他のスクリプト用語のポップアップ リストに InstallScript で定義されたローカル変数が追加されます。デフォルトでは、これが設定されています。
   - **いいえ** ― InstallScript エディター内にコードを入力しているとき表示されるポップアップ リストに、ローカル変数は表示されません。

[いいえ] を選択した場合、[InstallScript] ビューの中央ペインにある関数、プロパティ、およびメソッド フォルダーには、スクリプトファイルからの関数、プロパティ、またはメソッドが一覧表示されません。

この設定で [はい] を選択して、[InstallScript] ビューでコードを入力しているときにパフォーマンスに影響が出た場合は、この設定を [いいえ] に設定します。



***メモ・"ローカル変数を含める"** 設定は、[InstallScript] ビュー内のスクリプト エディターに適用します。たとえば [SQL スクリプト] ビューなど、その他のスクリプト エディターには反映しません。**"オートコンプリート"** 設定に **[いいえ]** を選択すると、この設定は無視されます。*

4. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じて InstallScript エディターでオートコンプリート機能を有効または無効にします。

オートコンプリート機能の仕組みについての詳細は、「スクリプト エディターでのコードの書き込みにオートコンプリート機能を使う」を参照してください。

スクリプト エディターでのコードの書き込みにオートコンプリート機能を使う

InstallShield のスクリプトエディタでオートコンプリート機能を有効にすると、スクリプト エディター内に入力している文字で始まる関数、キーワード、定数、文字列識別子の一覧がアルファベット順でポップアップ リストに表示されます。ポップアップ リストには、使用中のビューによって言語に適したオプションが表示されます。たとえば、[InstallScript] ビューのスクリプト エディターでは、InstallScript コードのオートコンプリート機能がサポートされています。[カスタム アクションとシーケンス] ビューの VBScript スクリプト エディターでは、VBScript のオートコンプリート機能がサポートされています。

また、ローカル変数のオートコンプリート機能が有効な場合、[InstallScript] ビューのスクリプト エディターで表示されるポップアップ リストには、ローカル変数も含まれています。

オートコンプリート機能を有効にする方法については、「スクリプト エディターでオートコンプリート機能を有効または無効にする」をご参照ください。



タスク ***関数、キーワード、定数、または文字列識別子のオートコンプリート機能を使用するには、以下の手順に従います:***

1. 適切なスクリプト エディターを含むビューを開いてから、編集するスクリプトを選択します。
2. スクリプトでコードを入力する挿入ポイントを置きます。
3. 以下のいずれかを実行します。
   - 関数、キーワード、その他のスクリプト用語の最初の文字を入力します。

     使用しているビューによって、またそのビュー内のスクリプト エディターに入力した言語によって、スクリプト エディターに入力している文字で始まる関数、キーワード、その他のスクリプト用語のアルファベット順のリストを含むポップアップ リストが表示されます。入力した文字と一致する最初の用語がリストで選択されています。入力した文字と一致する用語がない場合、リスト内の用語は選択されていません。
   - [InstallScript] ビューでスクリプト エディタ使うとき、使用可能な文字列識別子のリストにオートコンプリート機能を使用する場合、文字列定数演算子 (@) を入力すると、InstallShield は文字列識別子がアルファベット順に並んだポップアップ リストを表示します。

4. 選択された用語が希望の用語と異なるか、用語が選択されなかった場合、次のいずれかの方法で別の用語を選択します。
   - ポップアップ リストのスクロールバーを使ってリスト内を移動してから、希望の用語をクリックして選択します。
   - 上矢印と下矢印キーを使って前または次の用語を選んで変更します。
5. 選択した用語をスクリプトに貼り付けるには、その用語をダブルクリックするか、ENTER または TAB キーを押します。

用語を貼り付けずにリストを閉じる場合は、ESC を押すかポップアップ リストの外側をクリックします。

[スクリプト エディター] 内で InstallScript 関数の呼び出しについてのヒントを有効または無効にする

InstallShield では、[InstallScript] ビューを使ってスクリプトに関数呼び出しを入力するときに、InstallScript 関数の呼び出しについてのヒント (一種のツールヒント) を表示するかどうかを指定できます。関数呼び出しのヒントは、ビルトイン関数のパラメーター情報を表示します。その他、ビルトイン関数の説明や、入力しているパラメーターについての説明も表示します。

関数呼び出しのヒントを使うと、別のウィンドウに切り替えなくてもスクリプト エディター内で言語情報を参照できます。

ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、同じオートコンプリートの設定が、すべてのスクリプト エディターで使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*

タスク ***[InstallScript] ビューのスクリプト エディターで、関数呼び出しのヒントを表示するかどうかを指定するには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"関数呼び出しのヒントを表示"** 設定で、適切なオプションを選択します:
   - **はい** ― スクリプトに関数呼び出しを入力するときに、関数呼び出しのヒントが表示されます。デフォルトでは、これが設定されています。
   - **いいえ** ― スクリプトに関数呼び出しを入力するときに、関数呼び出しのヒントは表示されません。
3. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じて InstallScript エディター内で関数呼び出しのヒントを有効または無効にします。

関数呼び出しのヒントについての詳細は、「スクリプト エディター内で InstallScript 関数について関数呼び出しのヒントを表示する」を参照してください。

スクリプト エディター内で InstallScript 関数について関数呼び出しのヒントを表示する

関数呼び出しのヒントが有効な場合、[InstallScript] ビューでスクリプトに関数の呼び出しを入力しているときに、InstallScript 関数呼び出しのヒント (一種のツールヒント) が表示されます。関数呼び出しのヒントは、ビルトイン関数のパラメーター情報を表示します。その他、ビルトイン関数の説明や、入力しているパラメーターについての説明も表示します。

関数呼び出しのヒントを有効化する方法については、「[スクリプト エディター] 内で InstallScript 関数の呼び出しについてのヒントを有効または無効にする」を参照してください。



タスク

関数の説明、およびその関数のパラメーターに関する詳細を表示するには、以下の手順に従います:

1. 上記の説明のように、直接入力するまたは関数名の自動完了を利用してスクリプトに関数名を入力します。
2. 左かっこ "(" を関数名の後に入力します。すべてのパラメーターを含む、完全な関数の呼び出しを表示する関数呼び出しのヒントが表示されます。このヒントには、関数の説明と、次に入力する必要があるパラメーターの説明が含まれています。最初のパラメーターは、デフォルトで青色で表示されます。
3. 呼び出しのヒントに表示された通りにパラメーターを入力します。カンマを入力する度に、デフォルトで関数構文の次のパラメーターが青色で表示されます。
4. 呼び出しヒントを閉じるには、右かっこ ")" を入力します。または、ESC を押すか、呼び出しヒントの外にあるスクリプト エディター ペインをクリックします。



ヒント: *特定のビルトイン関数、定数、その他の InstallScript 用語に関するさらなる詳細を表示するには、挿入ポイントをその用語内に配置してから、F1 を押します。InstallShield ヘルプ ライブラリが開いて、その特定の用語に関する説明が表示されます。*

スクリプト エディターで構文ハイライト機能を有効または無効にする

InstallShield の様々なビューに含まれているスクリプト エディターでスクリプト ファイルを表示するときに、色属性を使ってキーワード、コメント、文字列、その他のスクリプト要素を識別するうかどうかを指定できます。各カテゴリは、識別しやすくするために異なる色で表示されます。

この機能は「*構文ハイライト*」と呼ばれ、コードを読み取りやすくして、コンテキストを改良します。さらに、たとえば予約語をユーザー定義識別子として使用したときなどに発生するエラーを避けるのに役立ちます。また、構文の色付けは、キーワードのスペルミス、文字列末尾の引用符の付け忘れ、コメント文字の付け忘れなど、スクリプト内の他のエラーの特定にも役立ちます。



ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、同じ構文ハイライトと色設定が、すべてのスクリプト エディターで使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*



タスク ***構文ハイライト機能をスクリプト エディターで有効にするには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"構文ハイライト"** 設定で、適切なオプションを選択します:
   - **はい** ー InstallShield は、**[スクリプト エディターのプロパティ]** ダイアログ ボックスにある **[色]** 領域で定義された構文ハイライトをすべてのスクリプト エディターで使用します。デフォルトでは、これが設定されています。
   - **いいえ** ー InstallShield は、いずれのスクリプト エディターでも構文ハイライトを使用しません。
3. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じてスクリプト エディターで構文ハイライト機能を有効または無効にします。

InstallShield では、スクリプト用語のカテゴリごとに使用される前景と背景の色を定義できます。詳しくは、「スクリプト エディターで構文ハイライトの色を変更する」をご覧ください。

スクリプト エディターで構文ハイライトの色を変更する

InstallShield では、たとえば [InstallScript] ビューや [SQL スクリプト] ビューといった、様々なスクリプト ビューで、異なるスクリプト要素を区別できるように色を変更することができます。コメント、関数、および文字列などのスクリプト要素について、前景の色 (テキストに使用される色) または背景色を変更できます。

この機能は「*構文ハイライト*」と呼ばれ、コードを読み取りやすくして、コンテキストを改良します。さらに、たとえば予約語をユーザー定義識別子として使用したときなどに発生するエラーを避けるのに役立ちます。また、構文の色付けは、キーワードのスペルミス、文字列末尾の引用符の付け忘れ、コメント文字の付け忘れなど、スクリプト内の他のエラーの特定にも役立ちます。



ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、同じ構文色の設定が、すべてのスクリプト エディターで使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*



タスク ***スクリプト エディターで使用される構文ハイライトを変更するには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **[色]** 領域で、構成するスクリプト要素の設定をクリックします。その設定で、[前景] と [背景] ボタンが表示されます。
3. 以下のいずれかを実行します。
   - [前景] ボタンまたは [背景] ボタンをクリックします。[**色**] ダイアログ ボックスが開いて、使用する色を選択できます。

- 前景または背景に使用する色の RGB 値を入力します。たとえば、0; 0; 0 / 255; 255; 255 は、前景に黒 (RGB 値、0; 0; 0) を使って、背景には 白 (255; 255; 255) を使うことを示します。特定の領域に色を設定したくない場合は、ダッシュ (-;-;-) を使います。

4. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

スクリプトのハイライト機能が有効な場合、必要に応じて、スクリプト エディターでテキストを表示するのに使用される色が変更されます。詳細については、「スクリプト エディターで構文ハイライト機能を有効または無効にする」を参照してください。

この機能を最大限に活用するために、以下の点にご注意ください:

- 予約語をスペルミスして入力した場合、その予約語はエディターで認識されないため色付けされません。スクリプト内に色属性なしで表示された "予約語" があれば、それはおそらくスペルミスです。
- 文字列リテラルの色付けには、前後の引用符も含まれます。後ろの引用符を忘れた場合、文字列の色は行の終わりまで拡張されます。この場合、引用符の後に続くはずだったテキストは、文字列リテラルの一部であるかのように表示されます。
- 色による構文のハイライトにより、/* で始まって */ で正しく閉じられなかったコメントを、容易に識別することができます。この場合、コメントに続くテキストは、すべてコメントの色属性で表示されます。
- 2 つのスラッシュ (//) で始まるコメントを含む行では、コメント文字 以降のすべてのテキストが行末までコメントとして認識されます。
- [InstallScript] ビューで表示されるスクリプト エディターでは、ログ ファイルやレポート ファイルなどの .rul または .h 以外の拡張子を持つファイルに、構文色は使用されません。

スクリプト エディターでテキストを表示するためのフォントを変更する

InstallShield では、たとえば [InstallScript] ビューや [SQL スクリプト] ビューのスクリプト エディターで使用する色を変更することができます。

ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、すべてのスクリプト エディターで同じフォントが使用されます。[SQL スクリプト] ビューでフォントを変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューのフォントも変更されています。*

タスク ***スクリプト エディターでテキストを表示するためのフォント、フォント スタイル、またはフォント サイズを変更するには、以下に従います。***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **[全般]** 領域で、**"フォント"** 設定をクリックしてから、この設定の省略記号ボタン (...) をクリックします。**[フォント]** ダイアログ ボックスが開きます。
3. 必要に応じてフォントの変更を指定します。
4. **[OK]** をクリックします。[**フォント**] ダイアログ ボックスが閉じます。
5. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じてスクリプト エディターでテキストを表示するためのフォントを変更します。

スクリプト エディターで行番号を表示または非表示にする

InstallShield のスクリプト エディターの左マージンに、行番号を表示するかどうかを指定できます。



ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、同じ行番号を付ける設定が、すべてのスクリプト エディターで使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*



タスク ***行番号をスクリプト エディターで有効にするには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"行番号を付ける"** 設定で、適切なオプションを選択します:
   - **はい** ー InstallShield が、すべてのスクリプト エディターの左マージンに行番号を追加します。
   - **いいえ** ー InstallShield は、いずれのスクリプト エディターの左マージンにも行番号を表示しません。デフォルトでは、これが設定されています。
3. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じてスクリプト エディターで行番号を付ける設定を表示または非表示にします。

スクリプト エディター内で表示されているスクリプトの行番号へジャンプする



タスク ***スクリプト中の特定の行番号へ挿入ポイントを移動するには、以下の手順を実行します:***

1. **[編集]** メニューで **[行へジャンプ]** をクリックします。[**行へジャンプ**] ダイアログ ボックスが開きます。
2. **[行]** ボックスで、挿入ポイントを動かす行の番号を入力します。
3. **[OK]** をクリックします。

スクリプト エディターで自動インデント機能を有効または無効にする

InstallShield では、コードを書き込み中にスクリプト エディター内で自動インデント機能を使用するかどうかを指定できます。自動インデントを使うと、読みやすくなります。

自動インデント機能が有効な場合、InstallShield は前の行で使われたインデントに基づいて新しい行をインデントさせます。[スクリプト エディター] ペインで ENTER を押すと、挿入ポイントは新しい行で、前の行の最初の文字の真下に置かれます。新しい行のインデントを無効にするには、BACKSPACE を押して、インデントを削除します。

自動インデントが無効な場合、改行後の行はインデントされません。つまり挿入ポイントは新しい行の第 1 列に置かれます。



ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、同じ自動インデント設定が、すべてのスクリプト エディターで使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*



タスク ***自動インデント機能をスクリプト エディターで有効にするには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"自動インデント"** 設定で、適切なオプションを選択します:
   - **はい** ー InstallShield は、すべてのスクリプト エディターで自動インデントをサポートします。デフォルトでは、これが設定されています。
   - **いいえ** ー InstallShield は、いずれのスクリプト エディターでも自動インデントを行いません。
3. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じてスクリプト エディターで自動インデント機能を有効または無効にします。

スクリプト エディターでタブ幅を設定する

InstallShield では、たとえば [InstallScript] ビューや [SQL スクリプト] ビューのスクリプト エディターで使用するタブ幅を指定することができます。



ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、すべてのスクリプト エディターで同じタブ設定が使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*



タスク ***スクリプト エディターで入力されるタブの幅を指定するには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"タブ幅"** 設定、スクリプトで使用する幅 (文字のスペース サイズの数の倍数) を入力します。デフォルト値は **4**です。
3. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じてスクリプトのタブ幅を調整します。

スクリプト エディターで構文の折りたたみ機能を有効または無効にする

InstallShield の様々なビューで、スクリプト エディターの構文の折りたたみ機能をサポートするかどうかを指定できます。構文の折りたたみ機能が有効な場合、スクリプトで展開可能または折りたたみ可能なブロックで始まるコードの各行の横のマージンに、プラス (+) またはマイナス (–) 記号が追加されます。プラス記号をクリックして非表示となっているコードを展開したり、マイナス記号をクリックしてコードを非表示にしたりできます。

構文の折りたたみ機能を使って、長いスクリプトを縮小することで、現在行っている作業に関連のあるコードに焦点を当てることができます。また、スクリプトの全体的な構造を確認するのに便利です。



ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、すべてのスクリプト エディターで構文の折りたたみ設定が使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*



タスク ***構文の折りたたみ機能をスクリプト エディターで有効にするには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"構文の折りたたみ"** 設定で、適切なオプションを選択します:
   - **はい** ― InstallShield は、すべてのスクリプト エディターで構文の折りたたみを行います。
   - **いいえ** ― InstallShield は、いずれのスクリプト エディターでも構文の折りたたみを行いません。デフォルトでは、これが設定されています。
3. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じてスクリプト エディターで構文の折りたたみ機能を有効または無効にします。

スクリプト エディターで空白スペースを表示または非表示にする

InstallShield では、様々なビューのスクリプト エディターで、スクリプト内の空白ページが用いられている場所に、記号その他の印を表示するかどうかを指定することができます。たとえば、[はい] を選択すると、スクリプト内の各スペースは中黒 (·)、各タブは矢印記号で表示されます。また、空白スペースが有効な場合は、改行も表示されます。また、空白スペースが有効な場合は、改行も表示されます。



ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、すべてのスクリプト エディターで空白スペース設定が使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*



タスク ***スクリプト エディターで空白スペースを表示するか非表示にするかを指定するには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"空白スペースを表示"** 設定で、適切なオプションを選択します:
   - **はい** ー InstallShield は、すべてのスクリプト エディターで空白スペース文字と記号を表示します。
   - **いいえ** ー InstallShield は、いずれのスクリプト エディターでも空白スペース文字と記号を表示しません。デフォルトでは、これが設定されています。
3. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じてスクリプト エディター内の空白スペース文字と記号を表示または非表示にします。

スクリプト エディター内でテキストをドラッグ アンド ドロップする

InstallShield のスクリプト エディターでは、テキスト編集時にドラッグ アンド ドロップをサポートするため、選択したテキストをコード内のある場所から別の場所に移動またはコピーすることができます。

テキストの移動



タスク ***テキストを移動するには、以下の手順を実行します。***

1. 移動するテキストを選択します。
2. 選択したテキスト上にマウスポインターを置きます。
3. 左マウスボタンを押したままにします。
4. マウスポインターの下に小さな四角が表示されたら、スクリプト内での選択テキストの移動先にポインターを動かします。ポインターを動かしている間は、左マウスボタンを押したままにしてください。
5. ポインターをテキストの移動先まで動かしたら、マウスボタンを離します。

テキストのコピー



タスク ***ドラッグアンド ドロップを使って選択したテキストをコピーするには、次の手順を実行します。***

1. 移動するテキストを選択します。
2. 選択したテキスト上にマウスポインターを置きます。
3. CTRL を押して、左マウスボタンを押したままにします。
4. マウスポインターの下に小さな四角とプラス記号が表示されたら、スクリプト内の選択テキストのコピー先にポインターを動かします。ポインターを動かしている間は、CTRL と左マウスボタンを押したままにしてください。
5. ポインターをテキストの移動先まで動かしたら、マウスボタンを離します。

[スクリプト エディタ] 内からスクリプト ファイルを印刷する

InstallShield では、たとえば [InstallScript] ビューや [SQL スクリプト] ビューなどのスクリプト エディターで表示されるスクリプトを印刷することができます。



タスク

アクティブな スクリプト エディター ペインに表示されているスクリプトを印刷するには、以下の手順に従います:

1. **[ファイル]** メニューで、**[印刷]** をクリックします。このコマンドは、挿入ポイントがスクリプト エディター ペインにある場合にみ有効です。**[印刷]** ダイアログ ボックスが開きます。
2. どちらかひとつのオプションを設定するには、以下の手順を実行します。
3. **[OK]** をクリックします。

スクリプト エディターのキーボード ショートカット

InstallShield のスクリプト エディターは、以下のキーボード ショートカットをサポートします。

テーブル 8 ・ スクリプト エディターのキーボード ショートカット

キーボード ショートカット	説明
CTRL+C CTRL+INSERT	選択されたテキストをクリップボードにコピーします。
CTRL+X SHIFT+DELETE	選択した文字列を削除して、クリップボードに置きます。
Ctrl+F	指定した文字列を検索します。
CTRL+H	テキストまたはその他の文字を検索/置換できる [置換] ダイアログ ボックスを表示します。
CTRL+G	指定した行に移動します。
TAB	選択された文字列をタブ 1 つ分右インデントします。
CTRL+U	選択した文字列をすべて小文字に変更します。
CTRL+V SHIFT+INSERT	クリップボードの内容をカーソル位置に挿入します。
CTRL+Y	取り消した操作をやり直します。
CTRL+A	文書の全文字列を選択します。

テーブル 8 • スクリプト エディターのキーボード ショートカット (続き.)

キーボード ショートカット	説明
CTRL+Z ALT+BACKSPACE	直前の操作を元に戻します。
SHIFT+TAB	選択した文字列のインデントを戻します。
CTRL+SHIFT+U	選択した文字列をすべて大文字に変更します。
CTRL+M	スクリプト エディターの幅を最大化、または以前の幅に戻します。
CTRL+I	[InstallScript] ビューに表示されているスクリプトの挿入ポイントで、関数の呼び出しを追加できる関数ウィザードを開きます。
ALT+K	現在挿入ポイントが置かれているスクリプト行で、ブックマークの追加と削除操作が切り替わります。
CTRL+K	次のブックマークを含むスクリプト行に移動します。
CTRL+SHIFT+K	前のブックマークを含むスクリプト行に移動します。

InstallShield の詳細設定を構成する

InstallShield プログラム ファイルの中に Settings.xml というファイルがあります。このファイルで、InstallShield のためのマシン全体に関する詳細設定が一部含まれています。InstallShield をインストールしたとき、使用している InstallShield の言語に応じて Settings.xml が次の場所の 1 つにインストールされます。

- **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
- **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411

Standalone Build の Settings.xml ファイルは、次の場所のいずれかにインストールされています:

- **英語** — *Standalone Build Program Files フォルダー*¥Support¥0409
- **日本語** — *Standalone Build Program Files フォルダー*¥Support¥0411

通常、Settings.xml ファイルの変更は推奨されません。ただし、あるケースでは、このファイルでの変更が必要になる場合があります。このセクションでは、Settings.xml の変更が必要にある場合のシナリオがいくつか説明されています:

- デジタル署名のタイムスタンプ サーバーを変更する
- Setup.exe と ISSetup.dll にストリームされるファイルの圧縮レベルを構成する
- .cab ファイルの最大サイズを構成する
- 仮想メディア メモリの最大サイズを構成する
- Standalone Build のポータブル実行可能ファイルの一覧を変更する
- XML エンコード オプションのサポートを追加する

- すべての仮想パッケージをビルドする場所を指定する
- セットアップのデフォルト Clone_wait パラメーターを変更する
- デジタル署名のプラットフォーム アーキテクチャを構成する
- ランタイム中の MSI の検証を構成する
- イベント設定コマンドの実行失敗に対するエラー表示の管理



注意 - *Settings.xml ファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*

デジタル署名のタイムスタンプ サーバーを変更する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*

リリースのデジタル署名情報を指定すると、InstallShield はビルド中に、DigiCert のサーバー (**http://timestamp.digicert.com**) をデフォルトのタイムスタンプ サーバーとして使用します。InstallShield には、そのデフォルト サーバーを異なるタイム スタンプ サーバーに変更できるマシン全体の設定があります。この設定を使って、タイムスタンプを無効にすることもできます。

InstallShield 2019 R2 以降では、連続するデジタル署名の間に遅延を設定することができます。この機能は、タイムスタンプ サーバーが連続する署名要求の処理に失敗した場合のみ必要になります。

Settings.xml の <DevStudio/Build> ノードの下にある <DelayBetweenSigning default="1500" /> ノードで、ミリ秒単位で指定する必要があります。



注意 - *次の手順では、InstallShield と共にインストールされている Settings.xml ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*



タスク ***デジタル署名のタイムスタンプ サーバーを構成するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409

- **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411

3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. <DigitalSignature> 要素を検索します。要素は次のように表示されています:

   <DigitalSignature Timestamp="http://timestamp.verisign.com/scripts/timstamp.dll"/>
6. 別のタイムスタンプ サーバーでオーバーライドするには、Timestamp 属性の値を適切な URL に設定します。

   タイムスタンプを無効にするには、Timestamp 属性の値を空白にします:

   <DigitalSignature Timestamp=""/>

   ***メモ・** タイムスタンプを無効にすると、デジタル署名の有効期間に影響します。*
7. Settings.xml ファイルを保存します。
8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。ファイル内にある主な要素は、縮小、展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。

InstallShield 2019 以降では、タイムスタンプ サーバーが SHA-2 サーバーに設定されています:

- <DigitalSignature Timestamp="http://sha256timestamp.ws.symantec.com/sha256/timestamp"/>

デジタル署名情報を含むリリースをビルドするたびに、構成された設定に基づいて InstallShield によってタイムスタンプが設定されます。

***ヒント・** Standalone Build を使ってリリースをビルドする場合、Standalone Build と共にインストールされている Settings.xml ファイルを更新します。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:*

- **英語** — *Standalone Build Program Files フォルダー¥Support¥0409*
- **日本語** — *Standalone Build Program Files フォルダー¥Support¥0411*

Setup.exe と ISSetup.dll にストリームされるファイルの圧縮レベルを構成する

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

この情報は、1 つまたは複数の機能に関連付けられているファイルのみが .cab ファイルに圧縮されるカスタム圧縮には適用しません。

InstallShield には、ビルド時に Setup.exe ファイルと ISSetup.dll ファイルにストリームされるファイルに使用する圧縮レベルを指定することができる設定が含まれています (マシン全体に適用されます)。以下は、Setup.exe ファイルと ISSetup.dll ファイルにストリームすることができるファイルの例です:

- すべての製品ファイル (すべてのファイルが Setup.exe セットアップランチャーに圧縮されるリリースの場合)
- "Setup.exe から抽出" の場所がある InstallShield 前提条件インストール
- "Setup.exe から抽出" の場所がある .NET Framework インストール
- "Setup.exe から抽出" の場所がある Windows Installer インストール

InstallShield では、ファイルが Setup.exe ファイルまたは ISSetup.dll ファイルにストリームされるとき、圧縮から除外するファイルを指定することができます (ワイルドカード文字の使用可)。

***注意** - 次の手順では、InstallShield と共にインストールされている Settings.xml ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*

タスク ***ストリームされたファイルの圧縮設定を構成するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. <StreamCompression> 要素を検索します。要素は次のように表示されています:

   <StreamCompression exclude="*.CAB" compressionlevel="-1" />
6. ファイルが Setup.exe ファイルまたは ISSetup.dll ファイルにストリームされるとき、特定のファイルまたはファイル種類を圧縮から除外する場合、除外属性の値をこれらのファイルの名前に設定します。次の事項に注意してください。
   - 複数のファイルを指定する場合、各ファイル名をカンマで区切ります。
   - ワイルドカード文字の指定には、アスタリスク (*) を使用します。

   たとえば、.cab ファイル、.exe ファイル、および test.txt という名前のファイルを圧縮から除外するように指定する場合、除外属性の値を次のように設定します:

   <StreamCompression exclude="*.CAB,*EXE,test.txt" compressionlevel="-1" />

   .cab ファイルは圧縮ファイルであるため、デフォルトの値は ***.CAB** に設定されています。
7. 以下のいずれかを実行します。

- 圧縮ファイルのサイズと実行時に圧縮ファイルを展開するために必要な時間のバランスを考慮に入れた圧縮レベルを使用する場合、compressionlevel 属性の値を **-1** に設定します。これがデフォルトの値です。
- 特定の圧縮レベルを指定する場合、compressionlevel 属性の値に 0 から 9 の間の数値を指定します。0 は圧縮なしを意味し、9 が最大圧縮を意味します。

  一般的に、0 から 9 の値を指定した場合、指定した値が大きいほど、圧縮されたファイルのサイズは小さくなり、実行時にファイルを展開するときの時間が長くなります。

8. Settings.xml ファイルを保存します。
9. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。ファイル内にある主な要素は、縮小、展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。

アプリケーション プロジェクト タイプの 1 つに圧縮されたリリースをビルドしたとき、構成した設定に従って、ファイルが Setup.exe ファイルと ISSetup.dll ファイルにストリームされます。

ヒント: *Standalone Build を使ってリリースをビルドする場合、Standalone Build と共にインストールされている* Settings.xml *ファイルを更新します。*Settings.xml *は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:*

- **英語** — *Standalone Build Program Files フォルダー¥Support¥0409*
- **日本語** — *Standalone Build Program Files フォルダー¥Support¥0411*

.cab ファイルの最大サイズを構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

また、この機能は、すべてのファイルが単一ファイルの .msi パッケージまたは Setup.exe *セットアップランチャーに埋め込まれている圧縮済みネットワーク イメージ リリースをビルドしている場合のみ適用します。*

この情報は、1 つまたは複数の機能に関連付けられているファイルのみが .cab ファイルに圧縮されるカスタム圧縮には適用しません。

.cab ファイルには、いくつかの制限事項があります。たとえば、単一 .cab ファイルの最大サイズは 2 GB に設定されています。また、サイズの大きい .cab ファイルを署名しようしたとき、およびサイズの大きい署名済み .cab ファイルのデジタル署名を検証しようとしたときにトラブルが生じた経験があるユーザーもいるかもしれません。

これらの制限事項を回避するため、InstallShield では、圧縮ネットワーク イメージ リリースに対してビルドされる各 .cab ファイルの最大サイズを指定することができます (設定はマシン全体に適用されます)。InstallShield で、リリースの .cab ファイルを作成しているとき、構成した .cab ファイルのしきい値に達すると、データが 2 つ以上の.cab ファイルに分割され、マルチパートの .cab ファイルが作成されます。InstallShield でマルチパート .cab ファイルを作成しない場合、単一の .cab ファイルにデータを格納するように構成できます。



***注意・**次の手順では、InstallShield と共にインストールされている* Settings.xml *ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*



タスク

InstallShield でマルチパート .cab ファイルを作成するかどうか、また .cab ファイルの最大サイズを指定する場合、以下の手順に従います:

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. <CompressedNetworkCABSize> 要素を検索します。要素は次のように表示されています:

   <CompressedNetworkCABSize default="600"/>
6. 以下のいずれかを実行します。
   - .cab ファイルの最大サイズを設定する場合、デフォルト属性の値としてサイズを MB で入力します。上記の例では、最大サイズは 600 に設定されています。.cab ファイルの最大サイズは 2 GB (2048 MB) に設定されているため、値は 2048 以下でなければなりません。

     デフォルト値は **600**です。
   - InstallShield でマルチパート .cab ファイルを作成しない場合、デフォルト属性の値を **-1** に設定します。
7. Settings.xml ファイルを保存します。
8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。ファイル内にある主な要素は、縮小、展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。

アプリケーション プロジェクト タイプの 1 つに圧縮されたネットワーク イメージ リリースをビルドしたとき、Settings.xml ファイルで構成した要件に従って、.cab ファイルが作成されます。Settings.xml ファイルで指定した値に応じて、.cab ファイルが .msi パッケージのMedia テーブルに表示されます。



***メモ-** .cab ファイル サイズを設定する前にメディア サイズをチェックする。異なる配布メディアの種類についての情報は、「[メディアの種類] パネル」を参照してください。*



ヒント: *Standalone Build を使ってリリースをビルドする場合、Standalone Build と共にインストールされている* Settings.xml *ファイルを更新します。*Settings.xml *は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:*

- **英語** — *Standalone Build Program Files フォルダー¥Support¥0409*
- **日本語** — *Standalone Build Program Files フォルダー¥Support¥0411*

仮想メディア メモリの最大サイズを構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

仮想メディア メモリ タイプには制限事項があります。たとえば、仮想メディア メモリの最大サイズは 512MB に設定されています。



注意: *次の手順では、InstallShield と共にインストールされている* Settings.xml *ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*



タスク ***InstallShield が仮想メディア メモリの最大サイズを指定するかどうかを指定するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. <CompressedNetworkCABSize> 要素を検索します。要素は次のように表示されています:

   <CompressedNetworkCABSize default="600"/>
6. 以下のいずれかを実行します。
   - .cab ファイルの最大サイズを設定する場合、デフォルト属性の値としてサイズを MB で入力します。上記の例では、最大サイズは 600 に設定されています。.cab ファイルの最大サイズは 2 GB (2048 MB) に設定されているため、値は 2048 以下でなければなりません。

     デフォルト値は **600**です。

- InstallShield でマルチパート .cab ファイルを作成しない場合、デフォルト属性の値を -1 に設定します。

7. Settings.xml ファイルを保存します。
8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。ファイル内にある主な要素は、縮小、展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。

アプリケーション プロジェクト タイプの 1 つに圧縮されたネットワーク イメージ リリースをビルドしたとき、Settings.xml ファイルで構成した要件に従って、.cab ファイルが作成されます。Settings.xml ファイルで指定した値に応じて、.cab ファイルが .msi パッケージのMedia テーブルに表示されます。



ヒント: *Standalone Build を使ってリリースをビルドする場合、Standalone Build と共にインストールされている* Settings.xml *ファイルを更新します。*Settings.xml *は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:*

- **英語** — *Standalone Build Program Files フォルダー¥Support¥0409*
- **日本語** — *Standalone Build Program Files フォルダー¥Support¥0411*

Standalone Build のポータブル実行可能ファイルの一覧を変更する

InstallShield にある [オプション] ダイアログ ボックスの [ファイルの拡張子] タブで、InstallShield でポータブル実行可能 (PE) ファイルと見なすファイル拡張子を変更することができます。InstallShield は、この一覧を使って、ダイナミック リンクがあるファイルのコンポーネントを、コンポーネント作成時のベスト プラクティスに沿ってどう作成するかを判別します。

Standalone Build を使ってリリースをビルドする場合、[オプション] ダイアログ ボックスは使用できません。ただし、Standalone Build が PE ファイルと見なすファイル拡張子の一覧を変更する場合、InstallShield Standalone Build Program Files フォルダーのサブフォルダーにインストールされているファイルの 1 つを変更することはできます。以下は、その手順の説明です。



注意: *次の手順では、InstallShield と共にインストールされている* Settings.xml *ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*



タスク

Standalone Build が PE ファイルと見なすファイル拡張子の一覧を変更するには、以下の手順に従います:

1. Standalone Build を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語** — *Standalone Build Program Files フォルダー*¥Support¥0409
   - **日本語** — *Standalone Build Program Files フォルダー*¥Support¥0411

3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. PEFileExtensions 要素を検索します。要素は次のように表示されています:

   <PEFileExtensions default="EXE|DLL|OCX|VXD|CHM|HLB|TLB|AX">
6. 必要に応じて、デフォルト属性のファイル拡張子の一覧に変更を加えます。各ファイル拡張子は垂直バー (|) で区切られています。
7. Settings.xml ファイルを保存します。
8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。ファイル内にある主な要素は、縮小、展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。

次回 Standalone Build を使用するとき、ファイル拡張子の更新された一覧が使用されます。

XML エンコード オプションのサポートを追加する

InstallShield では、XML ファイルに使用するエンコードの種類を指定することができます。この設定は、[XML ファイルの変更] ビューで XML ファイルを選択したとき [詳細] タブで表示されます。使用するエンコードの種類が提供されているエンコード オプションの一覧にない場合、IInstallShield Program Files フォルダーのサブフォルダーにインストールされているファイルの 1 つを変更して、追加のオプションに追加することができます。MSXML でサポートされているエンコードはすべて追加することができます。以下は、その手順の説明です。



***注意 ・** 次の手順では、InstallShield と共にインストールされている* Settings.xml *ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*



タスク ***[XML ファイルの変更] ビューの [詳細] タブで追加のエンコード オプションを [エンコード] 一覧に追加するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語** — *InstallShield Program Files フォルダー*¥System¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥System¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. ISXML 要素とその子要素を検索します。要素は次のように表示されています:

   <ISXML>

```
<Encodings>
  <Encoding>WINDOWS-1250</Encoding>
  <Encoding>WINDOWS-1251</Encoding>
  <Encoding>WINDOWS-1252</Encoding>
  <Encoding>WINDOWS-1253</Encoding>
  <Encoding>WINDOWS-1254</Encoding>
  <Encoding>WINDOWS-1255</Encoding>
  <Encoding>WINDOWS-1256</Encoding>
  <Encoding>WINDOWS-1257</Encoding>
  <Encoding>WINDOWS-1258</Encoding>
  <Encoding>ISO-8859-1</Encoding>
  <Encoding>ISO-8859-2</Encoding>
  <Encoding>ISO-8859-3</Encoding>
  <Encoding>ISO-8859-4</Encoding>
  <Encoding>ISO-8859-5</Encoding>
  <Encoding>ISO-8859-6</Encoding>
  <Encoding>ISO-8859-7</Encoding>
  <Encoding>ISO-8859-8</Encoding>
  <Encoding>ISO-8859-9</Encoding>
  <Encoding>US-ASCII</Encoding>
  <Encoding>UNICODE-1-1-UTF-8</Encoding>
  <Encoding>UNICODE-2-0-UTF-16</Encoding>
  <Encoding>UNICODE-2-0-UTF-8</Encoding>
  <Encoding>ISO-10646-UCS-2</Encoding>
  <Encoding>UCS-4</Encoding>
  <Encoding>UCS-2</Encoding>
  <Encoding>UTF-16</Encoding>
  <Encoding Default="true">UTF-8</Encoding>
 </Encodings>
</ISXML>
```

6. 始めと終わりの Encodings タグの間に、次のような新しい行を追加します:

   <Encoding>Type_of_Encoding</Encoding>

   Type_of_Encoding がある箇所は、使用可能にするエンコードを示します。ここには、InstallShield で XML 文書のエンコード属性に使用する値が入ります。

7. Settings.xml ファイルを保存します。

8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。<ISXML> 要素、および <Encodings> 要素は縮小、展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。

次回 InstallShield で [XML ファイルの変更] ビューを開いたとき、使用可能なオプションの 1 つとして追加したエンコードの種類が [XML ファイルの変更] ビューにある XML ファイルの [詳細] タブで **[新しいファイルに使用するエンコード]** 一覧に表示されます。

すべての仮想パッケージをビルドする場所を指定する



エディション: *InstallShield Premier Edition には、仮想化サポートが含まれています。*

InstallShield が .msi パッケージを変換するとき、デフォルトで、その .msi ファイルが含まれるフォルダーのサブフォルダーに仮想パッケージを保存します。場合によって、すべての仮想パッケージを生成する別の場所を指定したい場合が想定されます。たとえば、変換する .msi パッケージが読み取り専用の場所にある場合、デフォルトの仮想パッケージの場所を既存の書き込み可能な場所でオーバーライドする必要があります。

InstallShield には、すべての仮想パッケージをビルドする既存の書き込み可能な場所を指定できる、マシン全体に適用される設定が含まれています。このグローバル設定を構成する方法は、以下の通りです。



注意・次の手順では、InstallShield と共にインストールされている Settings.xml ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。



タスク　すべての仮想パッケージをビルドする場所を指定するには、以下の手順に従います:

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語** – *InstallShield Program Files フォルダー*¥System¥0409
   - **日本語**–*InstallShield Program Files フォルダー*¥System¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. <Virtualization> 要素とその子要素を検索します。要素は次のように表示されています:

   ```
   <Virtualization>
     <!— Instructions on how to specify a global path —>
     <GlobalBuildRedirectFolder></GlobalBuildRedirectFolder>
   </Virtualization>
   ```

6. <GlobalBuildRedirectFolder>要素のテキスト コンテンツとしてグローバル パスを入力します。たとえば、¥¥NetworkFolder¥VirtualPackages のサブフォルダーに仮想パッケージを作成するには、次を使用します:

   <GlobalBuildRedirectFolder>¥¥NetworkFolder¥VirtualPackages</GlobalBuildRedirectFolder>

   パスは既存のパスでなくてはなりません。パスを引用符で囲まないでください。
7. Settings.xml ファイルを保存します。
8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。<Virtualization> 要素は縮小、展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。

InstallShield で .msi パッケージを仮想パッケージに変換するときは常に、指定されたパスが使用されます。

カスタム仮想アプリケーションの作成

セットアップのデフォルト Clone_wait パラメーターを変更する



注意 *・次の手順では、InstallShield と共にインストールされている Settings.xml ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*



プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

InstallShield 2018 では、settings.xml に Clone_wait パラメーターの新しい設定が追加されています。デフォルトでは、元のセットアップがインストール処理の完了を待機するように設定されています。埋め込み Clone_wait オプションなしでセットアップ ランチャーを生成したい場合、この設定を変更できます。

エンド ユーザーは、コマンドライン パラメーターを使ってこの設定をオーバーライドすることができます。



タスク ***セットアップのデフォルト Clone_wait パラメーターを変更するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. <Clone_wait > 要素とその子要素を検索します。デフォルトでは次のように表示されています:

   <Clone_wait default="Yes"/>
6. デフォルト Clone_wait パラメーターを [いいえ] に変更します。

   <Clone_wait default="No"/>
7. Settings.xml ファイルを保存します。
8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。コードのエラーを確認します。

デジタル署名のプラットフォーム アーキテクチャを構成する

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*

InstallShield 2025 では、ファイルのデジタル署名に使用するプラットフォーム アーキテクチャを選択できます。

タスク ***デジタル署名のプラットフォーム アーキテクチャを構成するには、以下の手順に従います:***

1. **InstallShield** を閉じます。
2. InstallShield と共にインストールされている **Settings.xml** ファイルを検出します。**Settings.xml** は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
3. 後で元のバージョンに戻す必要があるときのために、**Settings.xml** ファイルのバックアップ コピーを作成しておきます。
4. テキスト エディターまたは XML ファイル エディターを使って、**Settings.xml** ファイルを開きます。

   メモ - *ファイルが編集のための管理者モードで開きます。*
5. **<DigitalSignature>**. タグを検索します。実際のタグは **<DigitalSignature Platform="X86"/>** です。
6. 異なるプラットフォーム アーキテクチャを使用する場合、"プラットフォーム" を X86 または X64 のどちらかに設定します。
   たとえば、
   - 32ビット署名の場合: <DigitalSignature Platform="X86"/>
   - 64ビット署名の場合: <DigitalSignature Platform="X64"/>

   メモ - *無効なエントリの場合、自動的に 32 ビット 署名が使用されます。*
7. **Settings.xml** ファイルを保存します。

メモ- 32 ビットのオペレーティング システム上で 64 ビット署名を使用することはできません。

ランタイム中の MSI の検証を構成する

プロジェクト- この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*

InstallShield 2025 では、ランタイム中に行なわれるデジタル署名の検証を有効にして、インストーラーの整合性を保つことができます。

タスク ***デジタル署名の MSI の検証を有効化するには、次の手順に従います:***

1. **InstallShield** を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを検出します。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
3. 後で元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成しておきます。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。

   メモ - ファイルが編集のための管理者モードで開きます。
5. **<DigitalSignature ValidateSetupAndMSISignatures>** タグを検索します。実際のタグは **<DigitalSignature ValidateSetupAndMSISignatures="no"/>です。**
6. MSI の検証を有効化するには、この値を [はい] に設定します。
7. Settings.xml ファイルを保存します。

イベント設定コマンドの実行失敗に対するエラー表示の管理

プロジェクト- この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

InstallShield を使用して、**[イベント]** タブの **[ビルド前のイベント]**、**[ファイルの準備イベント]**、**[圧縮前のイベント]**、または **[ビルド後のイベント]** 設定で指定されたコマンドがビルド時間中に実行されなかった場合、インストーラーが正しくビルドされた後にエラーメッセージを表示するかどうかを管理または制御できます。

デフォルトでは、これらの失敗は、ビルドが成功した後にエラーメッセージをトリガーしません。Settings.xml ファイルを変更することで、このような場合にエラーメッセージを表示するように InstallShield を構成できます。このような失敗に対するエラーメッセージ表示を有効にするには、以下の手順に従ってください。



***注意・** Settings.xml ファイルには、重要な構成データが含まれています。変更が正しくないと、InstallShield が動作しなくなる可能性があります。編集する前に、必ずこのファイルをバックアップしてください。*



タスク ***コマンド（[イベント] タブの必要な設定で指定します）の実行エラー メッセージの表示を有効化にするには、次の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを検出します。Settings.xml は、使用している InstallShield の言語バージョンに応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. <TreatBuildEventFailuresAsErrors> 要素とその子要素を検索します。デフォルトでは次のように表示されています:

   <TreatBuildEventFailuresAsErrors default="no"/>
6. デフォルト TreatBuildEventFailuresAsErrors パラメーターを [はい] に変更します。

   <TreatBuildEventFailuresAsErrors default="yes"/>
7. Settings.xml ファイルを保存します。
8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。コードのエラーを確認します。

リリースの配布用の仮想マシン設定を共有する



***エディション・** 仮想マシンへのリリース配布機能は、InstallShield Premier Edition でご利用いただけます。*



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

インストールのビルドが成功するたびに InstallShield が仮想マシン (VM) を指定のスナップショットに戻し、VM の電源をオンにしてインストールを VM にコピーし、テストが可能な状態となるようにプロジェクト内のリリースを構成できます。また、これらのテスト準備を必要なときにオンデマンドで行うこともできます。

[リリース] ビューで選択されているリリースの [イベント] タブにある "構成" 設定を使用して VM の詳細を構成すると、InstallShield によって 指定されたデータが VMConfigurations.xml という名前のファイルに書き込まれます。この VMConfigurations.xml ファイルは、マシン全体に適用するファイルで、一度構成を行うと他のInstallShield プロジェクトでも使用できます。また、他のチームメンバーと共有することも可能です。また、このファイルを Standalone Build で使用することも可能です。

このグローバル ファイルを共有可能にする方法は、以下の通りです:



タスク

カスタマイズ済みの VM 構成設定ファイルを、InstallShield または スタントアロン ビルドがインストールされている別のマシンと共有するには、以下の手順に従います:

1. 開発マシン上で、プロジェクト内の 1 つのリリースの VM 設定を構成します。詳しくは、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」をご覧ください。
2. InstallShield を閉じます。
3. InstallShield と共にインストールされている VMConfigurations.xml ファイルを見つけて、別のマシンにこれをコピーします。VMConfigurations.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
4. 元のバージョンに復元する必要がある場合を想定して、別のマシン上の VMConfigurations.xml のバックアップ コピーを作成しておき、使用中の開発マシンからのバージョンを使って、インストール済みのバージョンを上書きします。

   Standalone Build を使ってリリースをビルドする場合、Standalone Build と共にインストールされている VMConfigurations.xml ファイルを更新します。VMConfigurations.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語** — *Standalone Build Program Files フォルダー*¥Support¥0409
   - **日本語** — *Standalone Build Program Files フォルダー*¥Support¥0411



ヒント: *テキスト エディターを使って XML ファイルを手動で変更する場合、XML コードが正しく作成されていることを確認してください。そうでない場合、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで* VMConfigurations.xml *ファイルを開いて、不適切に書かれた XML コードを確認*

することができます。<VMConfigurations> 要素は折りたたみ/展開が可能です。それが不可能な場合、コードにエラーが無いか確認してください。

以前のバージョンの InstallShield からアップグレードする

InstallShield 2025 を使い始めるとき、以前のバージョンまたは別の InstallShield 製品を使って作成したインストール プロジェクトをアップグレードすることができます。このセクションでは、これらのアップグレードについて説明します。

- InstallShield 2014 以前のプロジェクトをアップグレードする
- InstallShield 2013 以前のプロジェクトをアップグレードする
- InstallShield 2012 Spring 以前のプロジェクトをアップグレードする
- InstallShield 2012 以前のプロジェクトをアップグレードする
- InstallShield 2011 以前のプロジェクトをアップグレードする
- InstallShield 2010 以前のプロジェクトをアップグレードする
- InstallShield 2009 以前のプロジェクトをアップグレードする
- InstallShield 2008 以前のプロジェクトをアップグレードする
- プロジェクトを InstallShield 12 以前からアップグレードする
- InstallShield 11.5 以前のプロジェクトをアップグレードする
- InstallShield Express で作成したプロジェクトをアップグレードする
- InstallShield Professional で作成したプロジェクトをアップグレードする
- InstallShield—Windows Installer Edition で作成したプロジェクトをアップグレードする

InstallShield 2014 以前のプロジェクトをアップグレードする

以下は、InstallShield 2014 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2014 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば 0.775 (.ism プロジェクトの場合) または .2014 (.issuite プロジェクトの場合) というファイル拡張

子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から 0.775 または .2014 を取り除いてください。InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので、ご注意ください。

InstallShield 2014 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

ターゲット システムとしてサポートされている Windows のバージョン リストに関する変更

今回より、Windows XP SP3 および Windows Server 2003 SP2 が、InstallShield で作成されたインストールを実行するターゲット システムで必要な Windows の最低バージョンです。これはすべてのプロジェクト タイプに適用します。

.spc および .pvk ファイルを使ったデジタル署名サポートの削除

InstallShield で、ビルド時に .spc および .pvk ファイルにデジタル署名を行うサポートは、今後使用できません。

InstallShield 2014 以前で、リリースまたはパッチに .spc および .pvk ファイルを使って実行時にデジタル署名を行うように構成済みで、そのプロジェクトを InstallShield 2025 で開こうとすると、アップグレード警告 –6048 (リリース)、–6049 (パッチ)、または –6050 (QuickPatch プロジェクト) が表示されます。この警告は、アップグレード中に InstallShield によって .pvk ファイルおよび関連パスワードがプロジェクトから削除されることを説明します。

InstallShield 2025 でリリースまたはパッチを正しくビルドするためには、そのリリースまたはパッチ構成から .spc の参照を削除する必要があります。これを、.pfx 証明書または証明書ストアにある証明書への参照と入れ換えることができます。詳しくは、次を参照してください:

- デジタル署名とセキュリティ
- ビルド時にリリースとそのファイルにデジタル署名を行う
- パッチ パッケージに署名する
- QuickPatch パッケージに署名する

.spc の参照を削除せずにリリースまたはパッチのビルドを試みると、.spc ファイルの削除が必要であることを通知するビルド エラー –7347 が表示されます。

.spc ファイルおよび .pvk ファイルを .pfx ファイルに変換する方法については、「デジタル署名とセキュリティ」を参照してください。

デジタル署名サポートにおけるオートメーション インターフェイスの変更

ISWiRelease オブジェクトは、今回より、次の読み書きプロパティをサポートしません:

- CorrespondingPrivateKey
- SoftwarePublishingCredentials

これらのプロパティを呼び出すと、エラーが発生します。これらのプロパティは、読み書き文字列プロパティ DigitalCertificateInfo に置き換えられています。

詳細については、「ISWiRelease オブジェクト」を参照してください。

InstallShield インストールからの SignTool.exe および Signcode.exe の削除

InstallShield をインストールしたとき、今回より SignTool.exe および Signcode.exe は開発マシンにインストールされません。手動でファイルにデジタル署名を行う場合、SignTool.exe の使用を考慮してください。SignTool.exe は、Microsoft Windows Software Development Kit (SDK) に含まれていて、Visual Studio と共にインストールされます。

基本の MSI インストールおよび InstallScript MSI インストールにおける PowerShell サポートの変更

基本の MSI インストールおよび InstallScript MSI インストールの PowerShell カスタム アクション サポートが改訂されました。サポートは Windows Installer プロパティ IS_PS_EXECUTIONPOLICY を使って、ターゲット システム上で PowerShell カスタム アクションを実行するのに使用する PowerShell 実行ポリシーの名前を示すことがなくなりました。実行時にこのプロパティを設定しても、インストールに何ら影響はありません。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける .msi パッケージのデフォルトの対象条件の変更点

アドバンスト UI およびスイート/アドバンスト UI プロジェクト内の共有パッケージをサポートできるように、InstallShield では以前必要だった .msi パッケージに対する MSI パッケージの対象条件が不要となりました。今回より、プラットフォームなどの実行時環境の要件を確認する .msi パッケージの対象条件の使用を制限できます。

InstallShield 2025 でアドバンスト UI またはスイート/アドバンスト UI プロジェクトに .msi パッケージを追加したとき、デフォルトでそのパッケージに MSI パッケージの対象条件が作成されることがなくなりました。この動作は新しい InstallShield 2025 プロジェクト、および InstallShield 2014 以前で作成してから InstallShield 2025 にアップグレードしたプロジェクトに適用します。

さらに、.msi パッケージに古いデフォルトの MSI パッケージの対象条件を含む InstallShield 2014 以前のプロジェクトを InstallShield 2025 にアップグレードした場合、InstallShield はパッケージの "対象条件" 設定から条件の不要な部分を削除します。

一部の状況では、InstallShield 2014 以前のデフォルト MSI パッケージの対象条件をカスタマイズした場合、InstallShield 2025 はアップグレード中にそれを削除することがありますが、そのまま残す場合もあります。実際のアップグレード動作はカスタマイズによって異なります。たとえば、単純にプラットフォーム条件を追加した場合、InstallShield は条件の元の MSI パッケージ部分を削除し、プラットフォーム条件のみを残します。ただし、カスタマイズが複雑な場合、InstallShield は条件をそのまま残します。

InstallShield 2025 にアップグレードした後、.msi パッケージの対象条件が予定通りに構成されていることを確認することが推奨されます。

カスタマイズされた、または編集済みのデフォルト MSI パッケージの対象条件がある場合、InstallShield はアップグレード時にカスタム条件をそのまま残します。.msi パッケージの対象条件が予定通りに構成されていることを確認することが推奨されます。

以前、.msi パッケージの "対象条件" 設定のデフォルトの MSI パッケージ条件は、パッケージの独自の製品コードと製品バージョンのプレースホルダーとして、条件の "製品コード" および "製品バージョン" 設定にアステリスク (*) を使用しました。

アドバンスト UI およびスイート/アドバンスト UI インストールのウィザード インターフェイスで使用されているアンパサンドの解釈に関する変更

アドバンスト UI およびスイート/アドバンスト UI プロジェクトのウィザード インターフェイスの特定の領域で、アンパサンドの解釈が更新されました。

新しい InstallShield 2025 プロジェクトの場合、ならびに InstallShield 2014 以前で作成したプロジェクトを InstallShield 2025 にアップグレードした場合、インストールでウィザード インターフェイスの次の任意の領域でアンパサンド (&) を使用すると、今回より、インストールはアンパサンドをリテラル文字として表示します。以前はアンパサンドがキーボード ショートカットの先頭の文字として解釈されました。この変更はまた、これらの文字列にアンパサンドを含む値に解決するプロパティが含まれている場合にも適用します。

- ウィザード ページまたは 2 番目のウィンドウのヘッダー領域にある文字列 – これは [ウィザード インターフェイス] ビューでウィザード ページまたはウィンドウの "タイトル" 設定で構成されます。
- ウィザード ページのキャプション バー – これは [ウィザード インターフェイス] ビューで [ウィザード ページ] ノードの "ウィザード キャプション" 設定で構成されます。
- コマンド リンク コントロールの [補足説明] 領域 – これはウィザード ページまたはウィンドウの "メモ" 設定で構成されます。
- イメージ コントロールの代替テキスト – これはウィザード ページまたはウィンドウにあるイメージ コントロールの "代替テキスト" 設定で構成されます。

さらに、ほとんどのラベル コントロールでは、今回より、"スタイル" 設定の下にある SS_NOPREFIX サブ設定のデフォルトの選択が True となりました。以前はデフォルトで False が選択されていました。このサブ設定が True 値に設定されていることで、これらのコントロールの文字列エントリに含まれるアンパサンドが誤ってキーボード ショートカットとして解釈されないようにし、コントロールの文字列に含まれるアンパサンドをウィザード インターフェイスのアンパサンドとして正確に表示します。

これらの SS_NOPREFIX の変更は、新しい InstallShield 2025 プロジェクトで使用可能なビルトイン デフォルトおよび定義済みのウィザード ページとウィンドウに適用します。InstallShield 2014 以前のプロジェクトを InstallShield 2025 にアップグレードした場合、SS_NOPREFIX 設定は標準のビルトイン デフォルト ウィザード ページとウィンドウへ自動的に変更されます。ただし、これらのプロジェクトを InstallShield 2025 にアップグレードするとき、定義済みウィザード ページを含むカスタムウィザード ページとウィンドウは自動的に変更されません。これらのアンパサンドの解釈を変更する場合、手動で変更を行なってください。

以前は、単一のアンパサンドを表示する代わりに、ウィザード インターフェイスの前述の領域でアンパサンドが誤ってキーボード ショートカットを指定した問題を回避するために、単一のアンパサンドの場所にエスケープ アンパサンド (&&) を使用することができました。しかし、これらのワークアラウンドを 1 つ以上含むプロジェクトを InstallShield 2025 にアップグレードした場合、エスケープ アンパサンドが 2 つのアンパサンドを表示することになります。このため、プロジェクト内のウィザード インターフェイスにおけるアンパサンドの使用を確認して、必要に応じて変更を行なう必要があります。

詳細については、「ウィザード インターフェイスにおけるキーボード ショートカットの指定とアンパサンド (&) の使用」を参照してください。

Trialware サポート

InstallShield は今後、Try and Buy/プロダクト アクティベーション タイプの Trialware を作成するためのサポートを含みません。今後、Trialware ビューは InstallShield に含まれていません。

InstallShield 2013 以前のプロジェクトをアップグレードする

以下は、InstallShield 2013 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2013 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば 0.774 (.ism プロジェクトの場合) または .2013 (.issuite プロジェクトの場合) というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から 0.774 または .2013 を取り除いてください。
InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので、ご注意ください。

InstallShield 2013 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。
InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

InstallShield 2012 Spring 以前のプロジェクトをアップグレードする

以下は、InstallShield 2012 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2012 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば 0.773 (.ism プロジェクトの場合) または .2012.4 (.issuite プロジェクトの場合) というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から 0.773 または .2012.4 を取り除いてください。
InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので、ご注意ください。

InstallShield 2012 Spring 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

すべてのプロジェクトに影響する変更 (新規プロジェクトおよびアップグレードされたプロジェクト)

このセクションでは、新規プロジェクトおよび InstallShield の以前のバージョンからアップグレードされたプロジェクトに影響する変更について説明されています。

ターゲット システムの要件における変更

InstallShield は、今後 Windows 2000 システム用のインストール作成をサポートしません。Windows 2000 が搭載されているマシン上で、エンド ユーザーが InstallShield 2013 でビルドされたインストールを実行しようとすると、インストールが正常に実行する場合がありますが、プロジェクトにエンド ユーザーがレガシー オペレーティング システム上でインストールを実行しないように防ぐ起動条件を含まない限り、予期しない結果が起こる可能性があります。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクトでは、エンド ユーザーが Windows 2000 ステム上でインストールを実行するとメッセージを表示する起動条件を含めることができます。InstallScript プロジェクトでは、Windows 2000 システムを確認して、適切な場合にメッセージを表示する SYSINFO 構造変数を使用する InstallScript コードを追加することができます。

アドバンスト UI およびスイート/アドバンスト UI インストールには、今回より Windows Vista および Windows Server 2008 が必要です。

InstallShield は、今後 モバイル デバイス用のインストール作成をサポートしません。したがって、[モバイル デバイス] ビュー、スマート デバイス プロジェクト タイプ、Palm OS オブジェクト、および Windows Mobile オブジェクトは今後 InstallShield に含まれません。スマート デバイス プロジェクトを InstallShield 2012 Spring Express Edition 以前から InstallShield 2025 にアップグレードしようとすると、がエラー メッセージを表示して、プロジェクトを開くことができません。InstallShield 2012 Spring Express Edition 以前から InstallShield 2025 にプロジェクトをアップグレードして、プロジェクトがデスクトップ プラットフォームをターゲットとし、モバイル デバイス サポートを含む場合、InstallShield はアップグレード中にモバイル デバイス サポートを削除して警告をログ記録します。

InstallShield を実行するシステムの要件に関する変更

InstallShield、Standalone Build、および InstallShield Developer Installation Manifest (DIM) Editor を実行するオペレーティング システムの最小要件は、今回より、Windows XP SP3 または Windows Server 2003 SP2 です。以前、オペレーティング システムの最小要件は、これらのオペレーティング システムのどちらかの RTM バージョンでした。

仮想化パックの変更

仮想化パックは InstallShield Premier のスタンドアロン バージョンの一部として提供されています。InstallShield Professional Edition では、今後 アドオンとして提供されません。

Repackager プロジェクト変換ツールが含まれなくなりました

Repackager は今後、InstallShield Premier Edition には含まれません。Repackager は、今回より AdminStudio でのみ使用することができます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクト ファイル (.issuite) における SML スキーマの変更

アドバンスト UI およびスイート/アドバンスト UI プロジェクト ファイル (.issuite) は XML ベースのファイルです。InstallShield 2025 では、ファイルのユーザー インターフェイス部分の基になる XML スキーマが大幅に変更されています。多くの属性が子要素に移動しています。条件、アクション、および検証が根本的に異なります。属性文字列の代わりに要素コレクションが使用されています。新しいスキーマは、インストールのユーザー インターフェイスに構成されている内容が、より明示的です。

InstallShield 2025 で新しいアドバンスト UI またはスイート/アドバンスト UI プロジェクトを作成すると、InstallShield は自動的に .issuite ファイルに新しい XML スキーマを使用します。InstallShield 2012 Spring 以前から InstallShield 2025 にプロジェクトをアップグレードすると、InstallShield が自動的に XML スキーマを更新します。

ビルトイン DLL カスタム アクションの新しい定義済みパス変数

InstallShield は、プロジェクトに 2 つの新しい定義済みパス変数 (ISRedistPlatformDependentFolder および ISRedistPlatformDependentExpressFolder) を含みます。これらのフォルダーのデフォルト値は、ビルトイン InstallShield カスタム アクション DLL の32 ビット バージョンが格納されている、*InstallShield Program Files フォルダー*¥Redist フォルダー内のサブフォルダーを参照します。新しい InstallShield 2025 プロジェクトを作成するか、InstallShield 2012 Spring 以降のプロジェクトを にアップグレードすると、InstallShield は自動的にこれらの 2 つの定義済みパス変数をプロジェクトに含みます。また、InstallShield はビルトイン InstallShield DLL カスタム アクションのソース場所といったパスにこれらのパス変数を使います。これはアップグレードされたプロジェクトだけでなく、新しいプロジェクトにも適用します。以前、InstallShield は DLL ファイルの保存場所として、<ISProductFolder>¥Redist¥Language Independent¥i386 または <ISProductFolder>¥Redist¥Language Independent¥i386 Express のどちらかのフォルダーを使用しました。この変更は、情報提供を目的として報告されています。

オートメーション インターフェイスの変更

InstallShield または Standalone Build のオートメーション インターフェイスまたは Standalone Build を使用する場合、既存のコードを更新して、新しい ProgID (IswiAutoAutomation Interface Version.ISWiProject) を反映させてください。スタンドアロン オートメーション インターフェイスで、InstallShield と同じ ISWiAutomationAutomation Interface Version.dll ファイルが使用されるようになりましたが、インストールされる場所は異なります。

Standalone Build を InstallShield と同じマシンにインストールする場合、最後に登録された ISWiAutomationAutomation Interface Version.dll ファイルが使用されますので注意してください。

新規プロジェクトに影響し、アップグレードされたプロジェクトに影響しない変更

このセクションでは、潜在的に新規プロジェクトに影響し、以前のバージョンからアップグレードされたプロジェクトには影響しない InstallShield の変更について説明されています。アップグレードされたプロジェクトの場合、手動による変更が必要になる場合があります。

スイート/アドバンスト UI およびアドバンスト UI ウィザード インターフェイスの デフォルト ウィザード フォーマットの変更

InstallShield 2025 で新しいスイート/アドバンスト UI またはアドバンスト UI プロジェクトを作成した場合、"ウィザード フォーマット" 設定のデフォルト値は新しいオプションの [Glass] です。スイート/アドバンスト UI またはアドバンスト UI プロジェクトを以前のバージョンの InstallShield から InstallShield 2025 にアップグレードした場合、"ウィザード フォーマット" 設定の値は変更されません。その場合、以前のバージョンの InstallShield で選択された値が保持されます。

プロジェクトで、適切な場合に異なるウィザード フォーマットを使用するように構成することができます。その場合、[ウィザード ページ] ノードを選択したときに [ウィザード インターフェイス] ビューに表示される "ウィザード フォーマット" 設定の値を変更します。

詳細については、「ウィザード インターフェイスのフォーマットを選択する」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI インストールのナビゲーション ボタン テキストのフォント色の変更

一部の状況下で、アドバンスト UI またはスイート/アドバンスト UI プロジェクトを InstallShield 2012 Spring 以前から InstallShield 2025 にアップグレードしたとき、テキストとボタンの色のコントラストが不十分なために、ナビゲーション ボタンのテキストが判読不可能な場合があります。これは、ナビゲーション ボタン テキストに使用するテキスト スタイルに薄いフォント色が指定されていて、ターゲット システムではウィンドウのボタンに薄い色の使用が構成されている場合に発生します。

InstallShield 2025 より、アドバンスト UI またはスイート/アドバンスト UI インストールでは、ナビゲーション ボタンに使用されているテキストのテキスト スタイルに指定されているフォント色が考慮されます。したがって、アドバンスト UI またはスイート/アドバンスト UI プロジェクトを InstallShield 2025 にアップグレードしたとき、ナビゲーション ボタン テキストに使用するテキスト スタイルの色を調整する必要があるかもしれません。その場合、[ウィザード インターフェイス] ビューで [ウィザード ページ] ノードをクリックしてから、"ナビゲーション テキスト スタイル" 設定に選択されているテキスト スタイルを確認します。次に、同じビューの [スタイル] ノードの下にあるテキスト スタイルで、必要に応じてその設定を調整します。

InstallShield 2012 Spring 以前の場合、アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、ナビゲーション ボタン テキストのテキスト スタイルの色が無視されました。実行時に、インストールは Windows がナビゲーション ボタンに使用した色 (通常は黒) をフォント色に使用しました。

新しいショートカットのシェル プロパティのサポート

InstallShield の [ショートカット] ビューでは、ショートカットの Windows シェル プロパティを設定するためのビルトイン サポートが提供されています:

- "Windows 8 スタート画面にピン留めする" 設定は、次の GUID およびプロパティ ID の組み合わせを使って、System.AppUserModel.StartPinOption プロパティを設定します。

  9F4C2855-9F79-4B39-A8D0-E1D42DE1D5F3, 12

- "シェル プロパティ" 設定の [ピン留めをしない] オプションは、次の GUID およびプロパティ ID の組み合わせを使って、System.AppUserModel.PreventPinning プロパティを設定します。

  9F4C2855-9F79-4B39-A8D0-E1D42DE1D5F3, 9

- "シェル プロパティ" 設定の [新規として強調表示しない] オプションは、次の GUID およびプロパティ ID の組み合わせを使って、System.AppUserModel.ExcludeFromShowInNewInstall プロパティを設定します。

  9F4C2855-9F79-4B39-A8D0-E1D42DE1D5F3, 8

これらのプロパティをサポートしないバージョンの Windows 上で実行されるパッケージに、(GUID とプロパティ ID の代わりに) これらのプロパティ名が 1 つ以上あったとき、Windows Insataller はエラーを生成することがあります。

InstallShield 2012 Spring 以前で "シェル プロパティ" 設定を使ってショートカットのシェル プロパティを構成し、プロジェクトをInstallShield 2025 にアップグレードした場合、プロパティ名は適切な GUID とプロパティ ID の組み合わせに自動的に置換されません。GUID とプロパティ IDの組み合わせに素早く切り替えるには、プロジェクトをアップグレードした後に [シェル プロパティ] ビューで古い構成を削除してから、新しいサポートを使ってこれらのプロパティの 1 つ以上を構成することを考慮してください。その他、適切な GUID とプロパティ ID を使ってエントリを手動でオーバーライドすることもできます。InstallShield のビルトイン サポートについては、「ショートカットのシェル プロパティを設定する」を参照してください。

このサポートは、基本の MSI、DIM、 InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

InstallShield 2012 以前のプロジェクトをアップグレードする

以下は、InstallShield 2012 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2012 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば 0.772 (.ism プロジェクトの場合) または .2012 (.issuite プロジェクトの場合) というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から 0.772 または .2012 を取り除いてください。
InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので、ご注意ください。

InstallShield 2012 以前、InstallShield 12 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

すべてのプロジェクトに影響する変更 (新規プロジェクトおよびアップグレードされたプロジェクト)

このセクションでは、新規プロジェクトおよび InstallShield の以前のバージョンからアップグレードされたプロジェクトに影響する変更について説明されています。

ターゲット システムで以前とバージョンまたは同一バージョンのスイート/アドバンスト UI インストールを起動したときの新しいデフォルト動作

デフォルトで、アドバンスト UI またはスイート/アドバンスト UI インストールには、それぞれ、2 つの "インストール済みスイート" 条件が含まれています。

- 新しいインストール済みスイートの終了条件は、エンドユーザーが、アドバンスト UI またはスイート/アドバンスト UI インストールの現在のバージョンによって、同じアドバンスト UI またはスイート/アドバンスト UI インストールの将来のバージョンが上書きインストールされるのを防ぎます。
- 新しい "インストール済みスイート" の終了条件によって、エンドユーザーが、新しいバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールを、以前のバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールの上から上書きインストールしたとき、アドバンスト UI またはスイート/アドバンスト UI インストールは初回インストール モードで実行されます。

これらの新しいデフォルト条件は、スイート/アドバンスト UI プロジェクトで使用できます。InstallShield 2012 スイート プロジェクトを InstallShield 2025 にアップグレードした場合、これらのデフォルト条件は自動的にプロジェクトに追加されます。これらの条件に関する詳しい情報は、「特定のバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールが既にインストールされているかどうかを判別する」をご覧ください。

以前、エンドユーザーが特定のバージョンのスイート インストールを、新しいバージョンのスイートが既にインストールされているターゲット システムでインストールしたとき、古いバージョンのスイートによって新しいバージョンが上書きインストールされていました。また、エンドユーザーが特定のバージョンのスイート インストールを、同一バージョンのスイートが既にインストールされているターゲット システムでインストールしたときも、古いバージョンのスイートが初回インストール モードで実行されていました。さらに、エンドユーザーが新しいバージョンのスイート インストールを、古いバージョンのスイートが既にインストールされているターゲット システムでインストールしたとき、インストールはメンテナンス モードで実行されていました。

オートメーション インターフェイスの変更

InstallShield または Standalone Build のオートメーション インターフェイスまたは Standalone Build を使用する場合、既存のコードを更新して、新しい ProgID (IswiAutoAutomation Interface Version.ISWiProject) を反映させてください。スタンドアロン オートメーション インターフェイスで、InstallShield と同じ ISWiAutomationAutomation Interface Version.dll ファイルが使用されるようになりましたが、インストールされる場所は異なります。

Standalone Build を InstallShield と同じマシンにインストールする場合、最後に登録された ISWiAutomationAutomation Interface Version.dll ファイルが使用されますので注意してください。

新規プロジェクトに影響し、アップグレードされたプロジェクトに影響しない変更

このセクションでは、潜在的に新規プロジェクトに影響し、以前のバージョンからアップグレードされたプロジェクトには影響しない InstallShield の変更について説明されています。アップグレードされたプロジェクトの場合、手動による変更が必要になる場合があります。

スイート/アドバンスト UI プロジェクトのウィザード インターフェイスに追加された新しいコンボ ボックスのデフォルト動作への変更事項

InstallShield 2025 のスイート/アドバンスト UI プロジェクトのウィザード インターフェイスまたは 2 番目のウィンドウで新しいコンボ ボックスを作成する場合、そのコントロールは、予め定義された値を持つドロップダウン リストを含むボックスになっています。ボックスには、エンドユーザーがカスタム値が入力できるテキスト ボックスです。以前 InstallShield 2012 で、新しいコンボボックス コントロールを追加したとき、コントロールにドロップダウン リストが含まれていましたが、テキストボックスは提供されていなかったため、エンドユーザーはカスタム値を入力することができませんでした。

プロジェクトを InstallShield 2012 から InstallShield 2025 にアップグレードして、スイート/アドバンスト UI ウィザード インターフェイスにコンボ ボックスが使用されている場合、そのコンボ ボックスは、予め定義された値を持つドロップダウン リストとして残りますが、テキスト ボックスではありません。このコントロールをテキスト ボックスを持つドロップダウン リストに変更するには、コントロールの CBS_DROPDOWNLIST スタイルを False に設定します。

InstallShield 2011 以前のプロジェクトをアップグレードする

以下は、InstallShield 2011 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2011 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば .771 というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から .771 を取り除いてください。InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので注意してください。

InstallShield 2011 以前、InstallShield 12 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

すべてのプロジェクトに影響する変更 (新規プロジェクトおよびアップグレードされたプロジェクト)

このセクションでは、新規プロジェクトおよび InstallShield の以前のバージョンからアップグレードされたプロジェクトに影響する変更について説明されています。

InstallScript カスタム アクションで呼び出される一部の MSI API の動作を変更する

InstallScript エンジンが **MsiGetProperty** などの Windows Installer API を呼び出す方法が変更されました。エンジンは今回より、Windows Installer API を直接呼び出します。以前は順番に Windows Installer API を呼び出すラッパー API を使用しました。この変更は、バッファー処理とバッファー サイズに関する問題を解決するために行われました。これは、InstallShield 2025 で作成された新しいプロジェクト、および InstallShield の以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクトに適用します。

この変更によって、インストール ハンドルを必要とするすべての Windows Installer API を使用するカスタム アクション関数に渡される MSI インストール ハンドルを使用することができます。以前、ハンドルは InstallScript エンジンによって管理されていたため、Windows Installer API に直接渡すことはできませんでした。

また、この変更によって、バッファー サイズの指定が必要な任意の Windows Installer API は、正しくそのバッファー サイズが指定されない限り失敗します。0 は無効なサイズです。コード内でバッファーのサイズに 0 を渡さないようにしてください。

Windows Installer API を呼び出す既存の InstallScript カスタム アクションがある場合、十分なサイズのバッファーが指定されていることを確認してください。そうでない場合、予期しないエラーが発生する可能性があります。

次のサンプル コードでは、InstallScript で Windows Installer プロパティ値文字列の値を取得する方法、および必要に応じてバッファーのサイズを増やす方法をデモンストレーションします。

```
prototype STRING MyGetProperty (HWND, STRING);
////////////////////////////////////////////////////////////////////////////
//  MyGetProperty
//
//  MSI プロパティ文字列値を戻す
//  パラメーターを入力:
//      hMSIHandle:     現在実行中の MSI データベースへのハンドル
//      szPropertyName:  :取得するプロパティの名前
//   出力:
//  プロパティ値を含む文字列
////////////////////////////////////////////////////////////////////////////
```

```
function STRING MyGetProperty (hMSIHandle, szPropertyName)
    NUMBER  nvBuf, nResult;
    STRING  szReturn;
begin

//プロパティ値のサイズを取得
    szReturn = "";

    nResult = MsiGetProperty (hMSIHandle, szPropertyName, szReturn, nvBuf);

    if (ERROR_MORE_DATA = nResult) then
        nvBuf = nvBuf + 1;   //最後のヌル文字のバッファー サイズを増加

        //プロパティ値を取得
        nResult = MsiGetProperty (hMSIHandle, szPropertyName, szReturn, nvBuf);

        if (nResult != ERROR_SUCCESS) then
            szReturn = "";
        endif;
    endif;
    return szReturn;
end;
```

スクリプトでif (ERROR_MORE_DATA == nResult) then の代わりに if (ERROR_SUCCESS == nResult) then を使用した場合、予期しないエラーが発生する可能性があります。

基本の MSI プロジェクトにおけるビルド警告 -7235

デフォルトで、ソフトウェア識別タグ機能は、すべての基本の MSI プロジェクトで有効になっています。これは、InstallShield 2025 で作成された新しいプロジェクト、および InstallShield の以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクトに適用します。

基本の MSI プロジェクトで、必須識別タグ設定 ([一般情報] ビューの "一意な ID"、"タグ作成者"、"タグ作成者 ID" 設定) にデータを入力せずにリリースをビルドすると、そのプロジェクトでタグ機能を有効のままにしておいた場合、ビルド警告 -7235 が発生します。このビルド警告では、特定の必須タグが空白であるために、ソフトウェア識別タグが作成されず、インストールに含まれなかったことが通知されます。この警告を解決するには、[一般情報] ビューで、各設定に適切な値を入力するか、または "ソフトウェア識別タグの使用" 設定で [いいえ] を選択します。

COM 抽出の変更

InstallShield では、COM 抽出時に新しい監視方式をサポートします。Windows Vista 以降のシステムまたは Windows Server 2008 以降のシステム上で、InstallShield を使用している場合、この新しい方式がデフォルトとなります。この方法は、カーネル ドライバーを使って、ビルド時のダイナミック COM 抽出中、およびデザイン時のスタティック COM 抽出中に変更されたレジストリ領域を監視します。この新しい方式は、DLL が既存のレジストリ エントリを読み込んでビルド マシンへの変更を妨げる以前の方法の利点を組み合わせたものです。

必要な場合、UseAPIRegistryHooks レジストリ値 (32 ビット マシンの場合は HKEY_LOCAL_MACHINE¥SOFTWARE¥InstallShield¥RegSpy レジストリ キーに含まれる、64 ビット マシンの場合は HKEY_LOCAL_MACHINE¥SOFTWARE¥Wow6432Node¥InstallShield¥RegSpy レジストリ キーに含まれる) の値データを設定して 3 つの異なる COM 抽出方式を切り替えることができます。使用可能な REG_DWORD 値データ:

- 0—API フックを使って、既存 DLLのレジストリ エントリを読み取ります。
- 1—レジストリのリダイレクトを使って、ビルド マシン上の登録済み DLL への変更を防止します。値を設定しなかった場合、これが Windows XP および Windows Server 2003 システム上でのデフォルト動作となります。

- 2—新しいカーネル モードの監視を使って、2 つのメソッドの両方の利点を組み合わせます。値が設定されていない場合、これが Windows Vista 以降および Windows Server 2008 以降のシステム上でのデフォルト動作となります。

この機能は、基本の MSI、DIM、InstallScript MSI、および マージ モジュール プロジェクト タイプに適用します。

マージ モジュール プロジェクトの文字列エントリ

マージ プロジェクトで作成される各文字列 ID には、今回よりマージ モジュールのモジュール ID GUID が含まれます。これは、すべての新しいマージ モジュール プロジェクトで作成される新しい文字列 ID、および InstallShield 2011 以前から InstallShield 2025 にアップグレードされた既存のマージ モジュール プロジェクトで作成された新しい文字列 ID に適用します。マージ モジュール プロジェクトを InstallShield 2011 以前から InstallShield 2025 にアップグレードしたとき、既存の文字列エントリにはモジュール ID GUID が追加されません。

マージ モジュール プロジェクトの文字列エントリの文字列 ID に GUID を使用することで、マージ モジュール プロジェクトとマージ モジュールを含むインストール プロジェクトで同じ文字列 ID が使用されたとき、これらの 2 つの文字列 ID に異なる値が割り当てられた場合に起こる競合を避けることができます。

InstallShield 2010 以前のプロジェクトをアップグレードする

以下は、InstallShield 2010 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2010 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば .770 というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から .770 を取り除いてください。InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので注意してください。

InstallShield 2010 以前、InstallShield 12 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

すべてのプロジェクトに影響する変更 (新規プロジェクトおよびアップグレードされたプロジェクト)

このセクションでは、新規プロジェクトおよび InstallShield の以前のバージョンからアップグレードされたプロジェクトに影響する変更について説明されています。

Visual Studio 2003 以前の統合サポートの終了

Visual Studio 内部から直接 InstallShield プロジェクトを作成、編集、およびビルドするには、Visual Studio 2005 以降が必要です。InstallShield を Visual Studio 2003 以前と統合することはできなくなりました。

InstallScript オブジェクトが非推奨となりました

InstallScript オブジェクトに代わって、InstallShield 前提条件が推奨されます。将来的なリリースで、InstallShield では InstallScript オブジェクトを作成または使用することはできません。また、定義済み InstallScript オブジェクトも提供されません。さらに、マージ モジュール ホルダー オブジェクトも使用できなくなります。InstallScript オブジェクトの代わりに、InstallShield 前提条件が推奨されます。InstallShield 前提条件エディターを使って独自の InstallShield 前提条件を作成して、InstallScript オブジェクト テクノロジが使用できなくなるときに備えてください。これらの InstallShield 前提条件は、InstallScript、InstallScript MSI、および基本の MSI プロジェクトで共有することができます。

InstallShield は Setup.exe および Update.exe の Unicode バージョンのみをビルド (今回より、ANSI バージョンは作成できません)

今回より、ではすべての Setup.exe と Update.exe ファイルが Unicode でビルドされます。これは、InstallShield 2025 で作成されるすべての基本の MSI、InstallScript、InstallScript MSI、および QuickPatch プロジェクトに適用します。また、InstallShield の以前のバージョンから InstallShield 2025 にアップグレードされたすべてのプロジェクトにも適用します。このため、セットアップランチャーを Unicode バージョンまたは ANSI バージョンのどちらでビルドするかを指定するためのこれまでの設定は削除されました:

- 基本の MSI プロジェクトの [リリース] ビューにあるリリースの [Setup.exe] タブにあった "セットアップ ランチャの種類 " 設定が削除されました。
- 基本の MSI、InstallScript MSI、および QuickPatch プロジェクトから "アップデート ランチャの種類" 設定が削除されました。

  この設定は、基本の MSI および InstallScript MSI プロジェクトの [パッチのデザイン] ビュー内にあるパッチの構成の [詳細] タブで使用できました。

  QuickPatch プロジェクトの場合、この設定は [一般情報] ビューの [ビルド設定] 領域にある [詳細] タブで使用できました。

Win32 API の定義変更

InstallScript エンジンに Unicode サポートが追加されたため、InstallScript ヘッダー ファイル ISRTWindows.h でプロトタイプ化される Win32 API 関数が更新されました。適切な場合、既存の ANSI (A) 定義のほかに、API プロトタイプのワイド (W) バージョンが追加されています。一部のプロトタイプでは、A または W バージョンが指定されていません。その場合、エンジンは W バージョンを使用しようとします。以前は A バージョンが使用されました。

InstallShield 2010 以前のプロジェクトを InstallShield 2025 にアップグレードする場合、これらの新しいプロトタイプが、同じ API のユーザー定義のプロトタイプと競合する可能性があります。可能な限り、InstallScript で提供されているプロトタイプを使用することが推奨されます。ただし、これらの Windows API に InstallScript 用に新しく追加されたプロトタイプではなく、独自のプロトタイプを使用する場合、プリプロセッサ定義のリストに ISINCLUDE_NO_WINAPI_H を追加します。その場合、[ビルド] メニューから [設定] を選択します。[コンパイル/リンク] タブにある [プリプロセッサ定義] ボックスに ISINCLUDE_NO_WINAPI_H と入力します。そうしないと、コンパイル エラーが発生する可能性があります。

InstallScript コードにおける Unicode サポートを確認

InstallScript コードでユーザー定義の Win32 API その他の外部 DLL プロトタイプを使用する場合で、API が Unicode 文字列入力をサポートするバージョンを持つとき、プロトタイプが BYVAL/BYREF WSTRING または WPOINTER を使用するように更新します。また、構造体を入力として受け取るコード内の API 関数を確認して、文字列または文字列ポインター メンバーを含む API 関数が必要な場合に Unicode として適切に宣言されていることを確認します。

スクリプトで、現在 Win32 の A バージョンを呼び出す場合、W バージョンに更新します。

InstallScript エンジンが Unicode をサポートするようにアップデートされたため、InstallScript 関数 StrLength の戻り値で、この変更が確認できます。その結果、この関数への InstallScript コード呼び出しに変更を加える必要があるかもしれません。

StrLength は、**StrLengthChars** と同様に動作します。これらの関数はどちらも、指定の文字列変数 (つまり、UTF-16 エンコード文字列のコード ユニット数) で最初のヌル文字までの文字数を返します。これは、InstallShield 2025 で作成された InstallScript プロジェクト、および InstallShield の以前のバージョンから InstallShield 2025 にアップグレードされた InstallScript プロジェクトすべてに適用します。以前、**StrLength** は指定の文字列変数で ANSI コード ページのヌル文字までの文字数を返しました。**StrLengthChars** の動作に変更はありません。

InstallScript プロジェクトにおける差分リリースのサポート

InstallShield 2011 以降の InstallScript プロジェクトで作成された差分リリースは、その InstallScript インストールも InstallShield 2011 以降で作成されている場合にのみ製品をアップデートできます。InstallShield 2011 以降を使って、以前の InstallScript インストールが InstallShield 2010 以前を使って作成されている製品のアップデートを作成する場合、差分リリースではなく、完全リリースを作成してください。

これは、Unicode サポートが InstallShield 2011 の InstallScript エンジンに追加された新しい機能であるためです。差分リリースを使って、ターゲット システム上で製品の以前のバージョンをアップデートした後、以前のバージョンのインストールおよび以前のランタイム エンジンを使って、メンテナンス処理を行います。InstallScript エンジンの InstallShield 2010 以前のバージョンは、Unicode 格納形式を読み取ることができないため、インストールは失敗します。

InstallScript エンジンによってインストールされた 64 ビット レジストリ エントリのアンインストール

今回より、64 ビット ターゲット システム上で InstallScript または InstallScript MSI インストールが行われるときに、デフォルトで InstallScript エンジンがレジストリの 64 ビット部分に行われる変更をログ記録します。さらに、InstallScript エンジンによってログ記録された 64 ビット レジストリの変更が、今回より、アンインストール中にアンインストールされます。

InstallShield 2010 以前を使って、64 ビット レジストリ データを作成するインストールを作成した場合で、InstallShield 2025 を使って製品のアップグレードを作成するとき、ベース インストールによってログ記録された既存の 64 ビット レジストリ エントリは、製品がターゲット システムからアンインストールされるときに削除されません。この制限を回避する唯一の方法は、アンインストール中にレジストリ データを手動で削除する方法のみです。

スクリプト エディターの変更

[InstallScript スクリプト] ビューのスクリプト エディター ペイン内で、InstallShield 2010 以前のバージョンでは CTRL+SPACEBAR キーを押すとオートコンプリート機能としてビルトイン InstallScript 関数のリストが表示されましたが、今後は表示されません。今回より、オートコンプリート機能によるビルトイン InstallScript 関数を参照するには、その関数の最初のいくつかの文字を入力するだけです。ポップアップ リストには、その他の InstallScript キーワードも含まれています。詳細については、「スクリプト エディターでのコードの書き込みにオートコンプリート機能を使う」を参照してください。

スクリプト エディター ペイン内で右クリックすると表示されるコンテキスト メニューは、InstallShield 2010 以前のバージョンとは異なります。

- コンテキスト メニューから [空白スペースを表示] コマンドが削除されました。スクリプト エディターで空白スペース文字や記号を表示または非表示にするには、[スクリプト エディターのプロパティ] ダイアログ ボックスを使用します。このダイアログを使って、フォント、構文の色、行番号などのスクリプト エディターのその他の設定も変更できます。このダイアログ ボックスにアクセスするには、スクリプト エディター内を右クリックしてから、[プロパティ] を選択します。

- コンテキスト メニューから、[大文字に変換する] コマンド、または [小文字に変換する] コマンドが削除されました。スクリプト エディター内のテキストを大文字に変換するには、そのテキストを選択して CTRL+SHIFT+U を押します。小文字に変換するには CTRL+U を押します。
- コンテキスト メニューから [検索] コマンドまたは [置換] コマンドが削除されました。スクリプト内で検索を行うには、CTRL+F を押してから、検索したいテキストを指定します。文字列を検索してから、それを別の文字に置換するには、CTRL+H を押してから、適切な情報を指定します。別の方法として、[編集] メニューにある [検索/置換] コマンドを使用することもできます。

キーボード ショートカットの詳細については、「スクリプト エディターのキーボード ショートカット」を参照してください。

改訂されたスクリプト エディターを含むビューは、[InstallScript] ビュー、[SQL スクリプト] ビュー、および [カスタム アクションとシーケンス] ビュー (このビューで VBScript または JScript ファイルを参照する場合) です。

InstallScript デバッガーの変更

InstallScript コードをデバッグするために InstallScript デバッガー (ISDbg.exe) をインストール開発マシンからデバッグを行うマシンにコピーする場合、必ずコピー先マシン上の同じフォルダに SciLexer.dll と呼ばれるファイルもコピーしてください。SciLexer.dll は、インストール開発マシン上の ISDbg.exe ファイルと同じフォルダにあります (*InstallShield Program Files フォルダー*¥System)。

詳細については、「任意のコンピューターでのインストールのデバッグ」を参照してください。

新規プロジェクトに影響し、アップグレードされたプロジェクトに影響しない変更

このセクションでは、潜在的に新規プロジェクトに影響し、以前のバージョンからアップグレードされたプロジェクトには影響しない InstallShield の変更について説明されています。アップグレードされたプロジェクトの場合、手動による変更が必要になる場合があります。

DPI の変更と、その InstallScript ダイアログにおける影響

1 つ以上の編集済み InstallScript ダイアログを含む、InstallShield 2010 以前で作成した InstallScript または InstallScript MSI プロジェクトを InstallShield 2025 以降にアップグレードする場合、使用中のコンピュータが InstallScript ダイアログを編集したときに選択されていたのと同じ DPI 値を使用することを確認してください。そうでない場合、実行時にダイアログのサイズに問題が生じることがあります。

プロジェクトをアップグレードした後、必要に応じて使用中のコンピュータの DPI 値を変更することが出来ます。そうすると、実行時にダイアログのサイズが正しく表示されます。1 つ以上の編集済み InstallScript ダイアログを含む InstallShield 2025 以降のプロジェクトを InstallShield の将来のバージョンにアップグレードするとき、アップグレード中に同じ DPI 値を使用する必要はありません。

既存のプロジェクトにおける既存の InstallShield 前提条件のデザイン時およびビルド時の場所変更

今回より、InstallShield が InstallShield 前提条件ファイル (.prq ファイル)、その関連データ ファイル、および依存関係を検索するフォルダーを指定できるようになりました。以前は、InstallShield が .prq ファイルを検索するフォルダーは、*InstallShield Program Files フォルダー*¥SetupPrerequisites のみでした。

InstallShield 前提条件を *InstallShield Program Files フォルダー*¥SetupPrerequisites フォルダの場所から、[オプション] ダイアログ ボックスの [プロパティ] タブで定義した新しいカスタム場所に移動させた場合、InstallShield 2010 以前のプロジェクトを InstallShield 2025 にアップグレードするときに以下の手順が必要な場合があります:

1. [再配布可能ファイル] ビューまたは [前提条件] ビューで、プロジェクトに含まれているが、カスタム場所に存在する各 InstallShield 前提条件のチェック ボックスをクリアします。そのデータ ファイルまたは依存関

係がデフォルトの場所からカスタム場所に移動されている各 InstallShield 前提条件のチェック ボックスもクリアします。

2. 新しい [更新] ボタンをクリックします。

3. 手順 1 で、プロジェクトから削除した各 InstallShield 前提条件のチェック ボックスを選択します。

InstallShield がプロジェクトの ISSetupPrerequisites テーブルから前提条件のパスを削除します。InstallShield 2010 以前のプロジェクトでは、このテーブルに完全パスが格納されました。前提条件のチェック ボックスをクリアしてから、[更新] ボタンをクリックしないで再び選択した場合、InstallShield は ISSetupPrerequisites テーブルにファイル名のみではなく、引き続き完全パスを使用します。

InstallShield 2010 以前のプロジェクトを InstallShield 2025 にアップグレードした場合で、InstallShield 前提条件の場所を変更してから、プロジェクトにその前提条件を追加した場合、この [更新] 手順を行う必要はありません。また、InstallShield 2025 を使って新しいプロジェクトを作成した場合も、[更新] 手順は不要です。両方の状況において、InstallShield はプロジェクトの ISSetupPrerequisites テーブルにパスを含みません。このため、デフォルト パスではなく、カスタム検索パスを利用することができます。

InstallScript MSI インストールによる同製品の将来のメジャー バージョンの上書きを防ぐ

新しい InstallScript MSI プロジェクトの [アップグレード] ビューには、ISPreventDowngrade というメジャー アップグレード項目が含まれています。この項目は、エンド ユーザーが製品の現在のバージョンで同じ製品の将来のメジャー バージョンを上書きできないように防ぎます。InstallScript MSI プロジェクトを InstallShield 2010 以前から InstallShield 2025 にアップグレードした場合、ISPreventDowngrade 項目は自動的に追加されません。必要に応じて、手動で ISPreventDowngrade 項目を追加することができます。詳しくは、「現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ」をご覧ください。

InstallShield 2009 以前のプロジェクトをアップグレードする

以下は、InstallShield 2009 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2009 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば .768 というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から .768 を取り除いてください。InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので注意してください。

InstallShield 2009 以前、InstallShield 12 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

InstallShield の複数エディションをインストールする

InstallShield 2025 のどれか 1 つのエディション (Premier、Professional、または Express) を同じシステムに同時にインストールすることができます。以前は、InstallShield バージョンの Premier または Professional Edition がインストールされた同じシステム上に Express Edition をインストールすることが可能でした。

InstallShield を実行できるサポート対象オペレーティング システムのリストにおける変更

今回より、InstallShield (オーサリング環境) を実行するシステムの最小オペレーティング システム要件は、Windows XP または Windows Server 2003 です。以前、最小オペレーティング システム要件は Windows 2000 XP3 でした。

すべてのプロジェクトに影響する変更 (新規プロジェクトおよびアップグレードされたプロジェクト)

このセクションでは、新規プロジェクトおよび InstallShield の以前のバージョンからアップグレードされたプロジェクトに影響する変更について説明されています。

Setup.exe が Windows 9x、Windows NT4、または Windows システムで実行不可能となりました

InstallShield で作成された Setup.exe インストールは、Windows 9x、Windows NT4、または Windows Me 上で実行不可能となります。エンド ユーザーが Setup.exe を Windows 9x または Windows Me システム上で実行しようとすると、「*FullSetup.exePathAndFileName* ファイルには、Windows の新しいバージョンが必要です。 Windows のバージョンをアップグレードしてください。」というエラー メッセージが表示されます。Windows のバージョンをアップグレードしてください。」というエラー メッセージが表示されます。Windows NT4 システム上では、Windows はメッセージボックスを表示して、「*FullSetup.exePathAndFileName* は有効な Windows NT アプリケーションではありません。」とエラーを通知します。

InstallShield では、ターゲット オペレーティング システムを選択できる領域で、これらのレガシー オペレーティング システムをリストしません。たとえば、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトのプロジェクト アシスタントにある [インストール要件] タブで、これらのオペレーティング システムはリストされません。InstallScript プロジェクトでは、[プロジェクトの設定] ダイアログ ボックスの [プラットフォーム] タブでこれらのオペレーティング システムはリストされません。

InstallShield 2009 以前で作成された InstallScript プロジェクトを InstallShield 2025 にアップグレードしたときに、以前のプロジェクトのオペレーティング システムの設定にこれらのレガシー オペレーティング システムのみへの参照が含まれている場合、InstallShield によってレガシー オペレーティング システムが、すべてのサポート対象プラットフォームをターゲットにするオプションで置換されます。

Windows Installer 1.x 再配布可能ファイルが使用できなくなりました

Windows Installer 1.x 再配布可能ファイルは、サポートされていない Windows のレガシー バージョンのみをターゲットとするため、今後 InstallShield では使用できなくなりました。以前は、[リリース] ビューまたはリリース ウィザードを使って、プロジェクトに Windows Installer 1.x 再配布可能ファイルを追加することが可能でした。

VBScript ランタイム ファイルの再配布可能ファイルが使用できなくなりました

InstallShield では、今回より、VBScript ランタイム ファイル用の InstallShield オブジェクトが提供されていません。この再配布可能ファイルは、今ではサポートされていない Windows の古いバージョンをターゲットとします。

InstallScript ダイアログ ソース コードの変更

ビルトイン InstallScript ダイアログ用の InstallScript コードは、個別の InstallScript スクリプト ファイル (.rul) から単一の ISRTScriptDialogs.rul ファイルに移動しました。また、InstallScript ビューのイベント クラス ドロップダウン リストに、新しい Dialog Source オプションが追加されました。このオプションを選択すると、イベントハンドラー ドロップダウン リストには、すべての InstallScript ダイアログがリストされます。このリストから任意のダイアログを選択して、そのコードをカスタマイズできます。

InstallShield は、下位互換性をサポートします。ダイアログ ソース コードを InstallShield 2009 以前のプロジェクトにインポートしてから、プロジェクトを InstallShield 2025 にアップグレードしたとき、そのダイアログ コードを引き続き使用できます。ただし、イベント ハンドラー ドロップダウン リストでダイアログを選択すると使用可能となるダイアログ ソースを使う場合、先にアップグレード済みプロジェクトにいくつかの変更を行わなくてはなりません。そうしないと、コンパイル エラーが発生する可能性があります。ダイアログ コードを使用するには、[ビルド] メニューから [設定] を選択します。[コンパイル/リンク] タブにある [プリプロセッサ定義] ボックスに次を入力します:

_ISSCRIPT_NEW_STYLE_DLG_DEFS

この定義を追加すると、それ以前に InstallShield の以前のバージョンでプロジェクトにダイアログがインポートされている場合、これらのインポートされた .rul ファイル内のコードがコンパイル エラーを引き起こす場合があります。これらのエラーは解決する必要があります。

_ISSCRIPT_NEW_STYLE_DLG_DEFS が定義されていない場合、InstallScript ビューのイベント クラス ドロップダウン リストで Dialog Source オプションを選択すると、警告メッセージが表示されます。

_ISSCRIPT_NEW_STYLE_DLG_DEFS 定義は、InstallShield 2025 で作成されたすべての新規プロジェクトに自動的に追加されます。

この機能は、InstallScript 、InstallScript MSI、および InstallScript オブジェクト プロジェクト タイプに適用します。

InstallScript ヘッダー ファイルの変更

InstallScript ヘッダー ファイル (.h) が再編成されました。その結果、いくつかの .h ファイルが使用できなくなりました。プロジェクト内の InstallScript ファイル (.rul) がこれらの古い .h ファイルを 1 つ以上参照する場合、スクリプトをコンパイルするとき、またはリリースをビルドするときにコンパイル警告が表示されます。この警告を解決するには、古い .h ファイルを参照する #include ステートメントをすべて削除してから、リリースをビルドします。また、Setup.rul ファイルまたは Setup.rul が参照するその他のスクリプト ファイルにおいて、ifx.h が #include ステートメントで参照されていることを確認します。

Windows API プロトタイプに関する InstallScript の変更

サービス関連の Windows API プロトタイプ (CreateServiceA、StartServiceA、および ControlService など) が、すべての InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトにおいて ISRTWindows.h でプロトタイプ化されました。また、これらは InstallScript カスタム アクションを含む基本の MSI プロジェクトとマージ モジュール プロジェクトでも、ISRTWindows.h でプロトタイプ化されました。

これらの Windows API に InstallScript 用に今回よりプロトタイプ化された定義ではなく、独自の定義を使用する場合、プリプロセッサ定義のリストに **ISINCLUDE_NO_SERVICEAPI** を追加します。これには、[ビルド] メニューから [設定] を選択します。[コンパイル/リンク] タブにある [プリプロセッサ定義] ボックスに **ISINCLUDE_NO_SERVICEAPI** と入力します。そうしないと、コンパイル エラーが発生する可能性があります。

定数 ISINCLUDE_NO_WINAPI_H は、今回より Windows 定数または Windows 構造定義ではなく、Windows API プロトタイプのみを抑制します。

パッチの作成 (基本の MSI、InstallScript MSI)

InstallShield は今回より、パッチの作成時に Windows Installer 4.5 テクノロジを使用します。この変更は、情報提供を目的として報告されています。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで、新しく追加された IIS サポート用のカスタム アクションとデータベース テーブル

InstallShield の基本の MSI または InstallScript MSI プロジェクトで [IIS 構成] ビューを使って IIS サポートを追加した場合、IIS 機能をサポートするためのいくつかの DLL カスタム アクションがプロジェクトに自動的に追加されます。InstallShield 2025 では、これらのカスタム アクションが強化され、アプリケーションを Web サイトに追加できるようになりました。また、これらのアクションの名前は、標準の命名規則に従って変更されています:

- **ISIISCosting** ー caExtractIISSuppFiles アクションに代わるカスタム アクション
- **ISIISRollback** ー caRlbackVRoots アクションに代わるカスタム アクション
- **ISIISRollback** ー caRemoveVRootsアクションに代わるカスタム アクション
- **ISIISInstall** ー caCreateVRoots アクションに代わるカスタム アクション
- **ISIISCleanup** ー calISCleanup アクションに代わるカスタム アクション

各 DLL カスタム アクションのエントリ ポイントは、対応するカスタム アクションの名前と一致するように名前が変更されています。

IIS 機能には、管理者権限が必要です。そのため、これらの各 DLL カスタム アクションは Privileged プロパティの値をチェックします。値は 1 でなくてはならず、それ以外の場合は、実行時にエラーが表示されます。InstallShield 2009 以前では、各 IIS カスタム アクションは Privileged = 1 条件を含んでいました。InstallShield 2025 では、カスタム アクションが Privileged プロパティ値をチェックするため、IIS Web サイトまたはその他の IIS データをプロジェクトに追加するときに、この条件は設定されません。

IIS サポートを含む基本の MSI プロジェクトまたは InstallScript MSI プロジェクトを InstallShield 2009 以前のバージョンから InstallShield 2025 にアップグレードすると、自動的にカスタム アクションが正しく更新され、名前が変更されます。さらに、古いカスタム アクションの条件がデフォルト条件から変更されていない場合、Privileged = 1 条件も削除されます。条件が変更されている場合、InstallShield は既存の条件をそのまま残します。必要に応じて、任意の条件を手動で変更することができます。

InstallShield 2025 では、すべての IIS データは ISIISItem および ISIISProperty テーブルに格納されます。InstallShield 2009 以前では、IIS データは ISIISAppPool、ISIISCommon、ISIISMetaData、ISIISWebServiceExtension、ISVRoot、ISVRootAppMaps および ISWebSite テーブルに格納されました。IIS サポートを含む基本の MSI プロジェクトまたは InstallScript MSI プロジェクトを InstallShield 2009 以前のバージョンから InstallShield 2025 にアップグレードすると、IIS データが自動的に新しいテーブルに移動し、また古いテーブルはプロジェクトから削除されます。

詳細については、次を参照してください。

- インターネット インフォメーション サービス
- InstallShield カスタム アクション リファレンス

[再配布可能ファイル] ビューの変更

[再配布可能ファイル] ビューには、強力な検索機能と組織化機能を提供する新しいツールバーとグループ ボックス領域が追加されました。このビューの新しい [詳細の表示] ボタンを使って、選択した再配布可能ファイルについての詳細ペインの表示/非表示を切り替えることができます。詳細ペインで、再配布可能ファイルがインストールするファイルなどの情報を確認することができます。[詳細の表示] ボタンは、以前、このビューの右上にあった [詳細の表示] および [詳細の非表示] リンクの代わりとなります。

新しいグループ ボックス領域は、[再配布可能ファイル] ビューの新しいツールバーの下にあります。列ヘッダーを、このグループ ボックス領域にドラッグ アンド ドロップして、再配布可能ファイルのリストを階層形式で表示することができます。すべての再配布可能ファイルをチェック ボックスが選択されているアイテムと選択されていないアイテムごとに 2 つのグループに分けて表示するには、チェック ボックス列をグループ ボックス領域にドラッグします。これで、プロジェクトに含まれているすべての再配布可能ファイルを認識しやすくなります。これは、以前、任意の再配布可能ファイルを右クリックしてから [選択したアイテムのみを表示] をクリックしたときの動作と似ています。[再配布可能ファイル] ビューで [選択したアイテムのみを表示] コマンドは、今後使用できません。

詳細については、「様々なビューで、[グループ ボックス] 領域を使って作業する」を参照してください。

連鎖 .msi パッケージの UI レベルをメインの .msi パッケージの UI レベルに限定する機能

連鎖 .msi パッケージの UI レベルは、今回より、親パッケージの現在の UI レベル以内に制限されました。たとえば、[リリース] ビューを使ってプロジェクトに連鎖 .msi パッケージを追加して、その "UI レベル" 設定に 完全 UI (/qf) を選択するが、メイン インストールがサイレント (/qn) で起動される場合、連鎖 .msi パッケージはサイレントで起動します。以前、連鎖パッケージはメイン インストールの [リリース] ビューでオーサされた UI レベルと同じレベルを表示しました。この動作を復元するには、ISChainExceedUILevel プロパティを 1 と等しく設定します。

XML ファイルの変更におけるエンコードと関連する相違点

[XML ファイルの変更] ビューを使ってターゲット マシンに既存するファイルまたはインストールの一部としてインストールされるファイルの変更を構成する場合、インストールは、その XML ファイルで指定されているエンコードを使用するようになりました。以前は、[XML ファイルの変更] ビューで指定されたエンコードが使用されました。これは、InstallShield 2025 で作成された新しいプロジェクト、および InstallShield 2009 以前からアップグレードされたプロジェクトに適用します。

また、ターゲット ファイルに存在しない要素の [この要素が存在しない場合、常に作成する] チェック ボックスがクリアされている場合、今回より、その子要素は作成されません。従って、たとえば //A/B/C という XML ファイルで、システム B が存在しない、または作成されるように設定されていない場合、C がターゲット システムで作成されることはありません。

InstallScript インストールのアンインストール キーにおけるメジャーおよびマイナー バージョンのレジストリ エントリの変更

InstallScript インストールは、アンインストール キーに VersionMajor および VersionMinor レジストリ値を常に作成します。これらの値の名前は、基本の MSI および InstallScript MSI インストール中に作成されるエントリの名前に一致します。これは、InstallShield 2025 で作成された新しいインストール、および InstallShield 2009 以前からアップグレードされたインストールに適用します。以前、InstallShield 2009 以前では、InstallScript インストールが作成する値の名前は MajorVersion および MinorVersion でしたが、今後は作成しません。

新しいレジストリ値を使用するため、次の InstallScript 定数の値が変更されました:

- REGDB_VALUENAME_UNINSTALL_MAJORVERSION は、今回より、MajorVersion ではなく VersionMajor です。
- REGDB_VALUENAME_UNINSTALL_MINORVERSION は、今回より、MinorVersion ではなく VersionMinor です。

MaintenanceStart 関数が呼び出されると、レジストリに更新された値名が作成されます。デフォルトで、古い値名が存在するとき、それが削除されます。ターゲット システムから古い値名を削除したくない場合、REGDB_OPTION_NO_DELETE_OLD_MAJMIN_VERSION という名前の新しい REGDB_OPTIONS オプションを使用します。古い値名のみを使用し続けたい場合は、**MaintenanceStart** が戻った後に新しいバージョンを削除しなくてはなりません。

RegDBGetItem 関数と共に REGDB_UNINSTALL_MAJOR_VERSION または REGDB_UNINSTALL_MINOR_VERSION を使用すると、**RegDBGetItem** は、まず新しい値をチェックします。新しい値が検出されると、関数は新しい値から値データを返します。新しい値が検出されなかった場合、関数は自動的に古い値をチェックします。古い値が検出されると、関数は古い値から値データを返します。

下位互換性を目的として、次の新しい定数が提供されています:

- REGDB_UNINSTALL_MAJOR_VERSION_OLD
- REGDB_UNINSTALL_MINOR_VERSION_OLD

これらの定数を **RegDBGetItem**、**RegDBSetItem**、および **RegDBDeleteItem** 関数と共に指定して、古い値を取得、設定、および削除できます。

次の新しい文字列定数も使用できます:

- REGDB_VALUENAME_UNINSTALL_MAJORVERSION_OLD は、MajorVersion として定義されています。
- REGDB_VALUENAME_UNINSTALL_MINORVERSION_OLD は、MinorVersion として定義されています。

詳細については、InstallScript 言語リファレンスの次のセクションを参照してください:

- REGDB_VALUENAME_UNINSTALL_MAJORVERSION
- REGDB_VALUENAME_UNINSTALL_MAJORVERSION_OLD
- REGDB_VALUENAME_UNINSTALL_MINORVERSION
- REGDB_VALUENAME_UNINSTALL_MINORVERSION_OLD
- REGDB_UNINSTALL_MAJOR_VERSION_OLD
- REGDB_UNINSTALL_MINOR_VERSION_OLD
- REGDB_OPTIONS
- RegDBSetItem
- RegDBGetItem
- RegDBDeleteItem

編集可能なダイアログの一覧からの SdShowMsg ダイアログの削除

ダイアログ エディターは現在、**SdShowMsg** などのタイトル バーを含まないダイアログをサポートしません。**SdShowMsg** ダイアログをカスタマイズすると、ダイアログが破損する可能性があります。このダイアログは今後、編集可能なダイアログとして [ダイアログ] ビューには表示されません。このダイアログをカスタマイズするには、ダイアログ エディターではなく、**SdShowMsg** 呼び出しを使います。

オートメーション インターフェイスの変更

OSFilter の eosAll 定数の値 (オートメーション インターフェイスの ISWiComponent および ISWiRelease のメンバー) が変更されました。新しい値は 64028880 です。以前は 5308624 でした。この定数の値を使用して、オートメーション インターフェイスを通してコンポーネントまたはリリースのオペレーティング システム一覧を構成する場合、スクリプトを新しい値にアップデートしなくてはなりません。

詳細は、次を参照してください:

- ISWiComponent オブジェクト
- ISWiRelease オブジェクト

InstallScript ランタイム スクリプト、ライブラリ、および ヘッダー ファイルの場所変更

InstallShield と共にインストールされる InstallScript ランタイム ライブラリ ファイルは、複数の個別のサブディレクトリではなく、1 ヶ所にまとめられました。スクリプト、ライブラリ、およびヘッダー ファイルは、以下のディレクトリ内の Src、Lib、および Include フォルダーにインストールされます:

InstallShield Program Files フォルダー¥Script¥Isrt

次のフォルダーは、そこに含まれていたファイルが前述の場所に統合されたため、今後はインストールされません。

InstallShield Program Files フォルダー¥Script¥IISRuntime

InstallShield Program Files フォルダー¥Script¥SQLRuntime

InstallShield Program Files フォルダー¥Script¥XMLRuntime

名前の競合を避けるために、次の場所にある Assert.h ファイルは、ISAssert.h に名前が変更されました:

InstallShield Program Files フォルダー¥Script¥Isrt¥Include

InstallShield 2025 で新しいプロジェクトを作成すると、新しいファイルの場所が使用されます。InstallShield 2009 以前から InstallShield 2025 にプロジェクトをアップグレードすると、新しい場所を使ってプロジェクトが更新されます。

リンク ライブラリとその場所の指定方法に関する変更点

[設定] ダイアログ ボックスにある [コンパイル/リンク] タブに新しく追加された [追加ライブラリ パス] ボックスを使って、InstallScript コンパイラが標準の InstallShield の場所には存在しない InstallScript ライブラリ (.obl ファイル) を検索する場所を指定できます。InstallScript と InstallScript オブジェクト プロジェクトでの標準の場所は次のとおりです:

- <ISProductFolder>¥Script¥Ifx¥Lib
- <ISProductFolder>¥Script¥Isrt¥Lib

基本の MSI および InstallScript MSI プロジェクトでの標準の場所は次のとおりです:

- <ISProductFolder>¥Script¥Iswi¥Lib
- <ISProductFolder>¥Script¥Isrt¥Lib

InstallShield 2025 で新しいプロジェクトを作成すると、[コンパイル/リンク] タブの [ライブラリ (.obl)] ボックスに ISRT.obl などの標準 InstallShield スクリプト ライブラリが自動的に一覧表示されます。ただし、このボックスでフル パスは表示されなくなりました。独自のカスタム ライブラリを追加するには、[ライブラリ (.obl)] ボックスでライブラリ ファイル名を、また [追加ライブラリ パス] ボックスにパスを指定します。[ライブラリ (.obl)] ボックスでフル パスとファイル名を指定する必要はありません。

InstallShield 2009 以前で作成したプロジェクトを InstallShield 2025 にアップグレードすると、[ライブラリ (.obl)] ボックスにリストされている標準スクリプト ライブラリのパスが自動的に削除されます。詳細については、「[コンパイル/リンク] タブ」を参照してください。

ISOSVERSIONINFO の InstallScript 構造定義の削除

ISOSVERSIONINFO 構造の定義と、それに対応する使用されていない、この構造のグローバル インスタンスが削除されました。代りに、OSVERSIONINFO 構造が使用できます。この定義の削除による機能の変更はありませんが、ISOSVERSIONINFO 構造定義またはグローバル構造インスタンスを使用すると、コンパイラ エラーが発生します。

コンパイラ エラーを回避するには、以下のどちらかを行います:

- スクリプトを対応する OSVERSIONINFO 構造を使って更新し、必要な場合は、この構造のローカル インスタンスを宣言します。スクリプトを適切な構造メンバー名で更新します。(OSVERSIONINFO メンバー名は、ISOSVERSIONINFO メンバー名とは異なりますので、ご注意ください。)OSVERSIONINFO の定義は以下のとおりです:

```
typedef OSVERSIONINFO
    begin
        NUMBER nOSVersionInfoSize;
        NUMBER nMajorVersion;
        NUMBER nMinorVersion;
        NUMBER nBuildNumber;
        NUMBER nPlatformId;
        STRING szCSDVersion[128];
    end;
```

- 以下のように、構造と構造インスタンスをローカルで宣言します:

```
// オペレーティング システムのバージョン情報を含むデータ構造。
// ISCompareServicePack が使用します。
typedef ISOSVERSIONINFO // 構造の定義
begin
    LONG ISIOSVersionInfoSize;    // このデータ構造のバイト サイズ
    data structureLONG ISIMajorVersion;        // OS のメジャー バージョン番号
    LONG ISIMinorVersion;        // OS のマイナー バージョン 番号
    LONG ISIBuildNumber;         // OS のビルド番号
    LONG ISIPlatformId;          // OS プラットフォーム
    STRING szISCSDVersion [128];  // OS に関する追加情報
end;

// OS バージョン 情報データ構造の変数
// ISCompareServicePack が使用します。
ISOSVERSIONINFO ISVersion;

// OS バージョン情報変数をポイントするポインター
// ISCompareServicePack が使用します。
ISOSVERSIONINFO POINTER pISVersion;
```

InstallScript プロジェクトにおける Setup.ini の旧形式キーの書き込みについて

InstallScript プロジェクトで、旧形式のキー (Resource、EngineVersion、および EngineBinding) が Setup.ini ファイルに書き込まれることがなくなりました。

ISCab.exe が使用できなくなりました

今回より ISCab.exe はサポートされていません。そのため、ISCab.exe は InstallShield に含まれていません。

新規プロジェクトに影響し、アップグレードされたプロジェクトに影響しない変更

このセクションでは、潜在的に新規プロジェクトに影響し、以前のバージョンからアップグレードされたプロジェクトには影響しない InstallShield の変更について説明されています。アップグレードされたプロジェクトの場合、手動による変更が必要になる場合があります。

ファイル、フォルダー、およびレジストリ キーのアクセス許可を保護するためのサポート変更点

[一般情報] ビューに追加された "ロックダウンの設定方法" 設定を使って、プロジェクトに含まれるファイル、フォルダー、およびレジストリ キーに設定する新しいアクセス許可について、新しいカスタム InstallShield 処理または従来型の Windows Installer 処理のどちらを使用するかを指定できます。新しい [カスタム InstallShield 処理] オプションには、[従来型の Windows Installer 処理] オプションよりも多くの利点があります。

すべての新しいプロジェクトにおける、この設定のデフォルト値は [カスタム InstallShield 処理] オプションです。プロジェクトを InstallShield 2009 以前から InstallShield 2025 にアップグレードした場合、この設定のデフォルト値は [従来型の Windows Installer 処理] オプションです。

この新しい設定は、基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで利用できます。

詳細は、次を参照してください:

- ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護
- プロジェクトで、ロックダウン環境でのアクセス許可タイプを選択する

Beta Windows Installer 5 のユーザーごとインストール サポートにおける ReadyToInstall ダイアログの変更

[一般情報] ビューには、"[ユーザーごと] オプションの表示" 設定があります。この設定を使って、特定の状況下において ReadyToInstall ダイアログにエンド ユーザーが製品をインストールする方法 (現在のユーザーまたはすべてのユーザー) を指定できるボタンを含めるかどうかを指定できます。[ユーザーごと] ボタンは、Windows Installer プロパティ MSIINSTALLPERUSER を 1 に等しく設定して、パッケージを現在のユーザーにインストールすることを示します。MSIINSTALLPERUSER プロパティは Windows Installer 5 で使用できます。

InstallShield 2025 で新しい基本の MSI プロジェクトを作成すると、ReadyToInstall ダイアログはユーザーごとおよびマシンごとボタンのサポートを含みます。これらのボタンは、適切な場合、実行時に表示または非表示されます。基本の MSI プロジェクトを InstallShield 2009 以前から InstallShield 2025 にアップグレードした場合、このサポートは ReadyToInstall ダイアログに自動的には追加されません。必要な場合、これらのボタンと関連条件を ReadyToInstall ダイアログに手動で追加することができます。新規 InstallShield 2025 プロジェクトで ReadyToInstall ダイアログを使用することがガイドラインとされています。

SecureCustomProperties に追加された機能のインストール先のパブリック ディレクトリ プロパティ

機能の "インストール先" 設定に場所を指定するとき、その場所にパブリック ディレクトリ プロパティが含まれる場合、今回より、InstallShield はそのプロパティを SecureCustomProperties プロパティに追加して、エンド ユーザーが製品のアドバタイズ後にインストール先を変更できるようにします。これは、InstallShield 2025 で作成された新しいプロジェクトで発生します。プロジェクトを InstallShield 2009 以前から InstallShield 2025 にアップグレードした場合、すべての機能のインストール先にも同様の変更が行われます。

この変更は、基本の MSI および InstallScript MSI プロジェクト タイプに適用します。

InstallWelcome ダイアログと ResolveSource アクションにおける条件の変更

InstallWelcome ダイアログと ResolveSource アクションの条件は、InstallShield 2025 で作成されたすべての新しい基本の MSI プロジェクトにおいて、**Not Installed** に変更されました。この条件は、InstallWelcome ダイアログと ResolveSource アクションを、パッチを伴う初回インストールで使用できるように変更されました。基本の MSI プロジェクトを InstallShield 2009 以前から InstallShield 2025 にアップグレードした場合、この条件は自動的には変更されません。パッチを含む初回インストールでダイアログとアクションを使用する場合、この条件をアップグレードされたプロジェクトで **"Not Installed"** に変更できます。

SdLicenseRtf と SdLicense2Rtf.rtf 関数の .rtf ファイルサイズの制限についての強化

InstallScript ダイアログ関数 **SdLicenseRtf** および **SdLicense2Rtf** で使用される .rtf ファイルのサイズ制限は、64 KB に代って 16 MB となりました。以前、ファイル サイズが 64 KB を超えると、実行時に EULA テキストの一部がライセンス ダイアログに表示されませんでした。

InstallShield 2009 以前でスクリプトの **SdLicenseRtf** または **SdLicense2Rtf** 関数をオーバーライドしたあと、そのプロジェクトを InstallShield 2025 にアップグレードした場合、**SendMessageTimeout** 呼び出しを DLG_INIT の EM_EXLIMITTEXT メッセージで更新して、サイズ制限を手動で変更する必要があります。**SendMessageTimeout** 呼び出しの iParam パラメーター (4 番目のパラメーター) を変更します。**SendMessageTimeout** 呼び出しは、次のように変更します:

```
SendMessageTimeout( hEdit, EM_EXLIMITTEXT, 0, 0xffffff );
```

以前は、コードに以下が含まれていました:

```
SendMessageTimeout( hEdit, EM_EXLIMITTEXT, 0, 0 );
```

InstallScript MSI インストールからの OnResolveSource イベント ハンドラーの削除

InstallScript イベント ハンドラー OnResolveSource が、InstallScript MSI プロジェクトから削除されました。今回より、Windows Installer がソース解決をすべて処理します。InstallShield 2009 以前で InstallScript MSI プロジェクトに OnResolveSource イベントを追加して、そのプロジェクトを InstallShield 2025 にアップグレードした場合、そのイベントは呼び出されません。

基本の MSI インストールで SetupCompleteSuccess ダイアログからのログ ファイルの表示方法に関する変更

ShowMsiLog カスタム アクションは、今回より、WindowsFolder ディレクトリではなく、SystemFolder ディレクトリから Notepad.exe を起動します。このため、インストールが Windows Vista 以降で実行され、エンド ユーザーが SetupCompleteSuccess ダイアログでログ ファイルの表示を選択した場合、インストールは SystemFolder ディレクトリから Notepad.exe を起動します。これは、Windows Server 2008 Standard Edition では、Notepad.exe が Windows ディレクトリではなく、System32 ディレクトリに配置されているために変更されました。

この動作は、InstallShield で作成されるすべての新しい基本の MSI プロジェクトではデフォルトで使用できます。InstallShield 2009 以前のプロジェクトを InstallShield 2025 にアップグレードした場合、InstallShield が自動的にこの動作を変更することはありません。必要に応じて動作を手動で変更することができます。その場合、[カスタム アクションとシーケンス] ビューで ShowMsiLog アクションをクリックします。(このアクションが表示されない場合、[カスタム アクション] ノードを右クリックしてから、[すべてのカスタム アクションを表示] をクリックします。)"ファイル名とコマンドライン" 設定を次のように構成します:

[SystemFolder]notepad.exe "[MsiLogFileLocation]"

つまり、値には [WindowsFolder] の代わりに [SystemFolder] を使用します。

InstallScript MSI インストールにおける ALLUSERS プロパティの変更点

InstallShield 2010 より、新しい InstallScript MSI プロジェクトでは、デフォルトで ALLUSERS プロパティに 1 が設定されます。ほとんどのインストールは、マシンごとに管理者権限を使用して実行される必要があるため、これが推奨される実装です。この値は、InstallScript MSI がサイレントで実行される際の ALLUSERS 関連の問題を回避するためにも推奨されます。

InstallShield 2009 以前で作成されたプロジェクトを InstallShield 2025 にアップグレードしたとき、ALLUSERS プロパティの値は自動的に変更されません。また、このプロパティが以前のプロジェクトで定義されていない場合も、自動的に追加されません。

InstallScript インストールは今後、_Setup.dll を含みません

InstallScript インストールは、今回より _Setup.dll を含みません。以前の一部のバージョン (DevStudio 9 および InstallShield X) は、アンインストールで _Setup.dll をログ記録しませんでした。その結果、アンインストール後にこのファイルが Disk1 フォルダーの場所 (DISK1TARGET) に残りました。InstallShield のより新しいバージョン (InstallShield 10.5 から InstallShield 2009) でアップデートを作成して、そのアップデートが DevStudio 9 または InstallShield X で作成された元のインストール向けの場合、_Setup.dll がアップデートによってログ記録されたために、それがアンインストール中に削除されました。_Setup.dll は InstallShield 2025 で作成された InstallScript インストールに含まれていないため、_Setup.dll ファイルが残されたままになる可能性があります。このため、DevStudio 9 または InstallShield X で作成されたインストールからのアップデートを行う場合、アンインストールを確実に完了するために、アンインストール中に _Setup.dll ファイル (DISK1TARGET ^ "_Setup.dll") を手動で削除しなくてはならない場合があります。

プロジェクトを以前のバージョンで保存する

InstallShield は、今後プロジェクトのダウングレードをサポートしません。つまり、InstallShield 2025 プロジェクトを InstallShield 2009 以前のプロジェクトとして保存することはできません。

Trialware サポート

Trialware ビューを含むのは InstallShield Premier Edition のみです。このエディションでは、Try and Die タイプの Trialware を作成できます。InstallShield は今後、Try and Buy/プロダクト アクティベーション タイプの Trialware を作成するためのサポートを含みません。

Web プロジェクト

Web プロジェクト タイプは、今後、InstallShield で作成可能な新しいプロジェクトの種類の 1 つとしてリストされません。InstallShield 2009 以前の Web プロジェクトで提供されていた同じ機能が必要な場合は、基本の MSI プロジェクトを作成してから、[IIS 構成] ビューで Web サイトを追加します。

Web プロジェクトと基本の MSI プロジェクトとの唯一の違いは、新しい Web プロジェクトには [ファイルとフォルダー] ビューの IISROOTFOLDER ディレクトリに定義済みのフォルダーが自動的に含まれていた点です。IISROOTFOLDER ディレクトリに追加されたファイルはすべて、ターゲット システムの Web サーバーのルート ディレクトリにインストールされます。プロジェクトに Web サイトを追加すると、InstallShield が基本の MSI プロジェクトに IISROOTFOLDER ディレクトリの定義済みフォルダーを追加します。このため、[IIS 構成] ビューで 1 つ以上の Web サイトが構成されている基本の MSI プロジェクトは、InstallShield 2009 以前で作成された Web プロジェクトに相応します。

Compact プロジェクト

InstallShield は、今後 Compact プロジェクトをサポートしません。

Visual Studio の統合

Microsoft Visual Studio の統合は 1 回につき、InstallShield Premier Edition または InstallShield Professional Edition のいずれかの 1 バージョンとのみ可能です。システムで最後にインストールまたは修復された InstallShield のバージョンが Visual Studio の統合に使用されます。

InstallShield 2008 以前のプロジェクトをアップグレードする

以下は、InstallShield 2008 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2008 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば .766 というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から .766 を取り除いてください。InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので注意してください。

InstallShield 2008、InstallShield 12 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

すべてのプロジェクトに影響する変更 (新規プロジェクトおよびアップグレードされたプロジェクト)

このセクションでは、新規プロジェクトおよび InstallShield の以前のバージョンからアップグレードされたプロジェクトに影響する変更について説明されています。

追加された新しいリリース用のデフォルト セットアップランチャー値: Windows Installer Is Not Included

基本の MSI プロジェクトまたは InstallScript MSI プロジェクトで新しいリリースを作成するとき、Windows Installer エンジンの再配布可能ファイルがデフォルトで含まれなくなりました:

- リリース ウィザードを使って新しいリリースを作成する場合、[セットアップランチャー] パネルで Windows Installer を含めるかどうかを指定します。"**これらのオペレーティング システムをサポートする**" 設定のデフォルト値は **[Windows Installer をインストールしない]** です。以前のデフォルト値は **[Windows 9X と NT の両方]** でした。
- 今回より、[リリース] ビューの [リリース] エクスプローラーで新しいリリースを右クリックすると、[Setup.exe] タブにある [セットアップランチャー] 設定のデフォルト値は **[はい (MSI エンジンを含まない)]** です。この設定の以前のデフォルト値は **[はい (Windows NT および Windows 9x MSI エンジンを含める)]** でした。

この変更は、新しい InstallShield 2025 プロジェクトで作成されたすべての新しいリリースに適用します。

プロジェクトを InstallShield 2008 以前から InstallShield 2025 にアップグレードする場合、この変更はすべての新しいリリースに適用します。InstallShield の以前のバージョンで作成されたリリースの値が自動的に変更されることはありません。

アップグレードとパッチの検証

QuickPatch プロジェクトをビルドしたとき、InstallShield はパッチとアップグレードの検証を実行するようになりました。したがって、InstallShield 2025 で QuickPatch パッケージをビルドしたとき、検証エラーおよび警告が表示される場合があります。InstallShield 2008 以前で QuickPatch プロジェクトをビルドしたとき、InstallShield はアップグレードとパッチの検証を実行しないため、ビルド時にパッチまたはアップグレードの検証エラーまたは警告が表示されることはありませんでした。

さらに、QuickPatch プロジェクトおよび基本の MSI と InstallShield MSI プロジェクトの [パッチのデザイン] ビューを使って作成されたパッチで、Val0015 警告が ISSelfReg テーブルをチェックするようになりました。QuickPatch またはパッチ プロジェクトで ISSelfReg テーブルが追加されたれた場合、Val0015 はパッチがアンインストール可能であることを警告します。以前、Val0015 はこのテーブルのエントリをチェックしませんでした。

ビルド時に Setup.exe と ISSetup.dll にストリームされるファイルの圧縮

Setup.exe セットアップランチャーまたは ISSetup.dll ファイルを使用するリリースをビルドしたとき、InstallShield によって、ビルド時に Setup.exe ファイルまたはISSetup.dll ファイルにストリームされるファイルが圧縮されるようになりました。InstallShield が使用するデフォルトの圧縮レベルは、ファイルのサイズと実行時に圧縮ファイルを展開するために必要な時間のバランスをとっての目安です。これは、InstallShield 2008 以前から InstallShield 2025 にアップグレードされた既存の基本の MSI プロジェクトおよび InstallScript MSI プロジェクト、および新規の基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに適用します。

圧縮レベルを変更する場合、または圧縮をしない場合、マシン全体に適用する設定を利用してデフォルトのレベルをオーバーライドすることができます。詳細については、「Setup.exe と ISSetup.dll にストリームされるファイルの圧縮レベルを構成する」を参照してください。

以前、InstallShield には、ビルド時に Setup.exe ファイルまたは ISSetup.dll ファイルにストリームされたファイルを圧縮することができませんでした。このため、InstallShield 2008 以前でビルドされたリリースと InstallShield 2025 のデフォルト圧縮レベルでビルドされた同じリリースを比較したとき、Setup.exe または ISSetup.dll のファイル サイズに若干違いがある場合があります。また、ファイルの展開にかかる時間にも差が生じる場合があります。

マルチパート .cab ファイル

InstallShield には、今回より、ビルド時にネットワーク イメージ リリースについて作成した各 .cab ファイル (標準圧縮タイプ(カスタム圧縮ではない圧縮) で、かつすべてのファイルが単一ファイルの .msi パッケージまたは Setup.exe セットアップランチャーに埋め込まれている .cab ファイル) に対して 600 MB がデフォルト制限として設定されています。InstallShield で、この種類のリリースについて .cab ファイルを作成しているとき、この制限に達すると、データが 2 つ以上の.cab ファイルに分割され、マルチパートの .cab ファイルが作成されます。これは、InstallShield 2008 以前から InstallShield 2025 にアップグレードされた既存の基本の MSI プロジェクトおよび InstallScript MSI プロジェクト、および新規の基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに適用します。

.cab サイズの制限は必要に応じて変更することができます。InstallShield でマルチパート .cab ファイルを作成しない場合、単一の .cab ファイルを作成するように構成できます。詳細については、「.cab ファイルの最大サイズを構成する」を参照してください。

以前、InstallShield では、マルチパート .cab ファイルを作成できなかったため、.cab ファイルのサイズのビルトイン制限もありませんでした。

オートメーション インターフェイスと Standalone Build

InstallShield オートメーション インターフェイスと共に動作する既存のオートメーション スクリプトがある場合、スタンドアロン オートメーション インターフェイスと共に使用するためにスクリプト全体でライブラリの名前を IswiAuto*N* から SAAuto*N* (*N* はバージョン番号です) に変更する作業は必要なくなりました。

InstallShield Premier Edition で提供されている Standalone Build は、InstallShield と同じディレクトリ構造を使用するようになりました。

InstallShield のコマンドライン ビルドで使われる ISCmdBld.exe も Standalone Build と共にインストールされています。以前、Standalone Build は別のファイル (IsSaBld.exe) をコマンドライン ビルドに使用していました。

InstallScript プロジェクトにおける FlexNet Connect のサポート

FlexNet Connect のサポートをインストール プロジェクトに追加するときに使用する [アップデート通知] ビューが InstallScript プロジェクトで使用できなくなりました。

InstallScript プロジェクトにおける FlexNet Connect のサポートに関する詳細は次の参照してください:

- InstallShield テーブルで ISEnableUpdateService が 1 に変更されたとき、ISUS.obl はライブラリー覧に追加されません (つまり、他の FlexNet Connect 機能が有効にされません)。ISUS.obl ライブラリは現在サポートされていません。
- InstallScript プロジェクトを InstallShield 2008 以前から InstallShield 2025 にアップグレードすると、FlexNet Connect のサポートが自動的に無効化され、ISUS.obl がリンクされたスクリプト ライブラリの一覧から削除されます。
- FlexNet Connect の定数と関数はこれまで同様コンパイルされます。ただし、すべての関数は今回より ISERR_NOT_IMPLEMENTED を返します (**UpdateServiceGetAgentTarget** は除きます)。**UpdateServiceGetAgentTarget** はヌル ("") を返します。
- FlexNet Connect 用の InstallScript 関数の呼び出しは、すべてデフォルトの InstallScript コードから削除されました。OnUpdateUIBefore イベント ハンドラーでは、**UpdateServiceOnEnabledStateChange** のコードが削除されました。OnMoveData イベント ハンドラーでは、**UpdateServiceRegisterProduct** と **UpdateServiceCreateShortcut** のコード ブロックが削除されました。

  InstallScript MSI プロジェクトを InstallShield 2008 以前から InstallShield 2025 へアップグレードしたとき、初期のバージョンのプロジェクトでこれらのイベントがオーバーライドされている場合、上記の関数が呼び出されます。これにより、問題が起きることはありません。関数は、現在、ISERR_NOT_IMPLEMENTED を返します。

  また、OnFirstUIAfter イベント ハンドラーが更新され、**SdFinishUpdate** の呼び出すオプションが除かれました。このオプションは以前デフォルトで無効にされています。
- すべての FlexNet Connect に関する再起動の後に実行される機能が削除されました。このため、製品は、再起動の後、FlexNet Connect に登録されていません。
- 今回より、InstallScript プロジェクトをビルドしたとき、FlexNet Connect に関するファイルはメディアに追加されません。
- ISUS.obl は、サポートされなくなるため、製品で提供されなくなります。

InstallScript MSI プロジェクトにおける FlexNet Connect のサポート

次の FlexNet Connect 関連の関数は、今後 InstallScript MSI プロジェクトでは推奨されません:

- **GetUpdateStatus** ー この関数は常に FALSE を返します。
- **SetUpdateStatus** ーこの関数は ISERR_NOT_IMPL を返します。
- **GetUpdateStatusReboot** ー この関数は常に FALSE を返します。
- **SetUpdateStatusReboot** ー この関数は ISERR_NOT_IMPL を返します。

SetUpdateStatus と **SetUpdateStatusReboot** 関数へのすべての呼び出しは、デフォルトの InstallScript コードから削除されました。

基本の MSI プロジェクトと InstallScript MSI プロジェクトにおける複数言語サポート

インストールが複数言語をサポートし、(製品が既にインストールされている状態で) エンド ユーザーが Setup.exe を再び起動してインストールをメンテナンス モードで実行した場合、言語選択ダイアログが再度表示されることはありません。メンテナンス ダイアログは、製品をインストールしたときに使用された言語で表示されます。

同様に、インストール済みの製品の以前のバージョンをアップデートする複数言語のマイナー アップグレードまたはスモール アップデートが実行されたとき、言語選択ダイアログは表示されません。アップグレード ダイアログは、製品をインストールしたときに使用された言語で表示されます。

これにより、初回でインストールされず、メンテナンス モード、またはアップグレードでインストールされるその他の機能を追加するときだけでなく、元のインストールが実行され、その機能がインストールされるときに使用された言語が、元のインストールの修復でも使用されるようになります。

InstallScript における複数インスタンスのサポート

言語の選択ダイアログを表示する複数言語、複数インスタンス InstallScript インストールで、インスタンスの選択ダイアログが言語の選択ダイアログの前に表示されるようになりました。ダイアログの順番が変更されたことにより、エンドユーザーが製品の新しいインスタンスに異なる言語を選択することができるようになりました。インスタンスの選択ダイアログで、エンドユーザーが既存のインスタンスを保持することを指定した場合、言語の選択ダイアログは表示されません。インスタンスの選択ダイアログが表示されるとき、言語の選択ダイアログが表示されるときと同じ言語で表示されます。

非複数インスタンス インストールの複数のインスタンスが、Setup.exe /ig コマンドライン パラメーターを通してインストールされた場合、言語の選択ダイアログは初回インストールでのみ表示され、メンテナンス モードでは表示されません。これにより、元のインストールが実行され、その機能がインストールされたときに使用された言語が、元のインストールの修復でも使用されるようになります。また、初回でインストールされず、メンテナンス モードでインストールされた機能を追加するときにも、この言語が使用されます。

サイレント モードまたは Setup.exe -hide_usd コマンドライン パラメーターと共に実行されるインストールなど、インスタンスの選択ダイアログを表示しない InstallScript の完全リリース インストールが実行されるとき、新しいインスタンスが自動的にインストールされます。この動作は、InstallShield 2025 におけるすべてのプロジェクト、および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクトで発生します。InstallShield 2008 では、/hide_usd が使用されたとき、初回でインストールされたインスタンスが保持され、サイレント モードが実行されたとき、新しいインスタンスがインストールされていました。InstallShield 12 以前では、初回でインストールされたインスタンスが、/hide_usd が使用されたときも、また、サイレント モードが実行されたときも保持されていました。

差分メディアでは、読み込まれた初回のインスタンスが更新されます。

Setup.exe /ig コマンドライン パラメーターを使用してインストールされたインスタンスが 2 つ以上あるシステムで非複数インスタンスが実行されたとき、インスタンスの選択ダイアログが表示されます。非複数インスタンスのインストールでは、新しいインスタンスに対して新しい GUID がランダムに生成されないため、インスタンスの選択ダイアログでは新しいインスタンスをインストールできません。インスタンスの選択ダイアログには、GUID が製品コードに一致するインスタンスがインストールされていて、かつ、それがメンテナンス可能なインスタンスの 1 つである場合、それが「デフォルト」のインスタンスとして表示されます。この動作は、InstallShield 2025 におけるすべての新規プロジェクト、および以前のバージョンからアップグレードされたプロジェクトに適用します。

インスタンスが 1 つのみインストールされている場合、インストール済みインスタンスが自動的にメンテナンスされる試みが行われ、インスタンスの選択ダイアログは表示されません。これは、インストールされているインスタンスがデフォルトのインスタンスではない場合も当てはまります。この動作の結果、デフォルトのインスタンスがインストールされていなく、非デフォルトのインスタンスが (/ig パラメーターが使用されて) インストールされている場合、デフォルトのインスタンスは、(a) 最初に非デフォルトのインスタンスをアンインストールするか、または、(b) /ig パラメーターを使ってデフォルト インスタンスの GUID を指定しないかぎり、インストールまたはメンテナンスできません。したがって、/ig パラメーターを使った複数インスタンスのインストールを可能にする場合、常にこのパラメーターを使用することをお勧めします。この動作は、InstallShield 2025 におけるすべての新規プロジェクト、および以前のバージョンからアップグレードされたプロジェクトに適用します。

InstallShield 2008 では、非複数インスタンスのインストールが、複数のインスタンスがインストールされているターゲット システムで実行された場合、インスタンスの選択ダイアログは表示されませんでした。このようなケースでは、デフォルトのインスタンスが常に使用されていました。InstallShield 12 以前では、非複数インスタンスのインストールが、複数のインスタンスがインストールされているターゲット システムで実行された場合、インスタンスの選択ダイアログは表示されましたが、非複数インスタンスのインストールでは新しいインスタンスに対して新しい GUID がランダムに生成されないため、インスタンスの選択ダイアログでは新しいインスタンスを表示できませんでした。しかしながら、ダイアログのタイトル バーで、エンドユーザーが更新するインスタンスを選択する必要があるというメッセージが表示されていました。エンドユーザーが更新またはメンテナンスを行うインスタンスを選択する必要があるというメッセージが本来表示されるべきでした。

InstallScript MSI インストールにおける SdRegisterUser、SdRegisterUserEx、SdCustomerInformation、およびSdCustomerInformationEx の変更

InstallScript MSI インストールで、次の InstallScript テキスト置換が Windows Installer プロパティに直接マップされるようになりました。

- IFX_PRODUCT_REGISTEREDOWNER は Windows Installer プロパティ USERNAME にマップされます。
- IFX_PRODUCT_REGISTEREDCOMPANY は Windows Installer プロパティ COMPANYNAME にマップされます。

このため、これらの InstallScript 変数が設定および取得されるとき、対応する Windows Installer プロパティが設定および取得されます。これらの変数のデフォルト値はレジストリから読み込まれず、Windows Installer プロパティがデフォルトとして使用されます。以前の機能を使用したい場合は、次のサンプルコードを使って変数を手動で設定します。

```
// 登録された所有者 (ログ ファイルからロード済みの値が "" の場合のみ、レジストリからロードする)
if( !StrLengthChars( IFX_PRODUCT_REGISTEREDOWNER ) ) then
    szValue = "";
    RegDBGetItem( REGDB_WINCURRVER_REGOWNER, szValue );
    if( StrLengthChars( szValue ) ) then
        IFX_PRODUCT_REGISTEREDOWNER = szValue;
    endif;
endif;

// 登録された会社 (ログ ファイルからロード済みの値が "" の場合のみ、レジストリからロードする)
if( !StrLengthChars( IFX_PRODUCT_REGISTEREDCOMPANY ) ) then
        szValue = "";
        RegDBGetItem( REGDB_WINCURRVER_REGORGANIZATION, szValue );
        if( StrLengthChars( szValue ) ) then
            IFX_PRODUCT_REGISTEREDCOMPANY = szValue;
        endif;
endif;
```

InstallScript MSI インストールで、様々な登録関連ダイアログ (**SdRegisterUser**、**SdRegisterUserEx**、**SdCustomerInformation**、および **SdCustomerInformationEx**) は、エンド ユーザーの選択に従って IFX_PRODUCT_REGISTEREDOWNER、IFX_PRODUCT_REGISTEREDCOMPANY、および IFX_PRODUCT_REGISTEREDCOMPANY を適切に設定するようになりました。これによって、対応する Windows Installer プロパティが自動的に更新されます。

登録関連ダイアログ関数 (**SdRegisterUser**、**SdRegisterUserEx**、**SdCustomerInformation**、または **SdCustomerInformationEx**) のいずれかで svName または svCompany のどちらかにヌル文字列 ("") が指定された場合、関数は (Windows Installer プロパティから判別した) 適切な変数をデフォルトとして使用します。以前、変数は svName と svCompany の両方がヌルの場合のみ使用され、**SdCustomerInformation** と **SdCustomerInformationEx** では、スクリプト変数がバイパスされて、Windows Installer プロパティから直接値が読み込まれました。

svSerial パラメーターにヌル文字列 ("") が指定されると、"シリアル番号" フィールドのデフォルト値が IFX_PRODUCT_REGISTEREDSERIALNUM に設定されるようになりました。以前、この場合はダイアログに値が表示されませんでした。

次の InstallScript 変数が新しく追加または変更されました:

- DISABLE_PERUSERBTN – この既存の変数は、通常は有効であるユーザーごとのラジオ ボタンが無効 (または非表示) であることを示します。この変数は、常に FALSE に初期化されます。以前、この変数は Windows 9.x システム上では TRUE に、その他のシステム上では FALSE に初期化されました。
- DISABLE_ALLUSERBTN – この既存の変数は、通常は有効であるすべてのユーザーのラジオ ボタンが無効 (または非表示) であることを示します。この変数は、常に FALSE に初期化されます。以前、オペレーティング システムが Windows 9x 以外で、エンド ユーザーが管理者またはパワー ユーザーではない場合、この値は TRUE に初期化されました。
- HIDE_DISABLED_BTNS – この新しい変数は、すべてのユーザーおよびユーザーごとのラジオ ボタンを無効ではなく非表示にすることを示します。デフォルト値は TRUE です。この変数が TRUE に設定されると、ラジオ ボタンのどちらかが無効である場合、両方のラジオ ボタンが非表示となります。

登録ダイアログは、今回より DISABLE_ALLUSERBTN の値に関わらず Windows 9x 上でユーザーごとのラジオ ボタンを自動的に無効 (または非表示) とします。以前、ユーザーごとのラジオ ボタンは DISABLE_PERUSERBTN が FALSE の場合のみ無効となりました。

登録ダイアログはまた、DISABLE_ALLUSERBTN の値に関わらず、エンド ユーザーが管理者またはパワー ユーザーでない場合に、すべてのユーザー ラジオ ボタンを自動的に無効 (または非表示) にします。この動作は、以前と変更はありません。

これらの変更は、InstallShield 2025 で作成されるすべての新しい InstallScript MSI プロジェクトだけでなく、以前のバージョンの InstallShield で作成され、InstallShield 2025 にアップグレードされた InstallScript MSI プロジェクトに適用します。

プロキシ サーバーのサポート

特定のファイルがターゲット システム上で必要な場合のみ、インストールがそのファイルをダウンロードするように構成できます。たとえば、Windows Installer エンジン、.NET Framework、および一部の InstallShield 前提条件が、一部またはほとんどのターゲット システム上に既存する可能性があります。これらのファイルをインストールに埋め込む代わりに、必要なファイルだけを実行時にダウンロードするようにプロジェクトを構成することができます。こうすることで、インストール全体のサイズを抑えることができます。

エンド ユーザーがプロキシ サーバーを使ってインターネットにアクセスする場合で、インストールがファイルをダウンロードするように構成されていると、今回より、インストールはダウンロード中に、Internet Explorer で手動で構成されたシステム プロキシ設定を使用するようになりました。これは、ターゲット システム上で別のブラウザーがデフォルトとして設定されている場合でも同じです。

InstallShield は、Internet Explorer の "設定を自動的に検出する" 設定をサポートしませんので、ご注意ください。(エンドユーザーが使用している Internet Explorer で、LAN 接続に対して [設定を自動的に検出する] チェック ボックスが選択されているときに、インストールでファイルのダウンロードが必要な場合、ファイルのダウンロードができないため、インストールは失敗します。エンドユーザーが使用している Internet Explorer で、LAN 接続に対して [設定を自動的に検出する] チェック ボックスが選択されている可能性があるとき、ダウンロードされるように構成する代わりに、すべてのファイルをインストールに埋めこんだ方が良い場合があります。ファイルが埋めこまれている場合、失敗は避けられます。)ただし、InstallShield は Internet Explorer の LAN 接続用にセットアップされた自動構成スクリプト機能をサポートします。

この動作は、InstallShield 2025 におけるすべての新規プロジェクト、および以前のバージョンで作成され、InstallShield 2025 にアップグレードされたプロジェクトでも同様です。

InstallShield 2008 以前では、デフォルト ブラウザーとして指定されているブラウザーで構成されているプロキシ サーバーの設定が使用されていましたが、場合によって失敗し、次のような問題が発生していました:

- Netscape 6 または 7 がデフォルト ブラウザーの場合、Netscape 4 の設定が使用される。Netscape 8 または 9 がデフォルト ブラウザーの場合、システム (Internet Explorer) の設定が使用される。
- Netscape 4 の設定が使用された場合、プロキシ サーバーの一覧のみが正しく読み込まれ、インポートされる。プロキシ バイパス一覧は読み込まれるが、正しくインポートされない。
- Internet Explorer 4 と互換性がない設定 (例、オートプロキシ スクリプト設定) はインポートされない。
- インストールでデフォルト ブラウザーの判別に使用する方法が Windows Vista と互換性がない。このため、Windows Vista で、デフォルト ブラウザーが正しく判別されない可能性があった。

1.x .msi パッケージを .cab ファイルにラップするための Web ダウンローダ オプションの削除

InstallShield では、.msi パッケージを .cab ファイルにラップできるオプションが使用できなくなりました。以前このオプションは、Windows Installer 1.1 または 1.2 を含む基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの Web ダウンローダ タイプのメディアで使用できました。Windows Installer 2.0 以降を使って、パッケージにデジタル署名を施すことが推奨されます。

InstallScript MSI プロジェクトにおける状態テキスト

InstallScript MSI プロジェクトの OnFirstUIBefore、OnMaintUIBefore、OnAdminInstallUIBefore、OnAdminPatchUIBefore、OnPatchUIBefore、OnUninstall、および OnResumeUIBefore イベント ハンドラーには、デフォルトで、すべての新しい InstallScript MSI プロジェクトに **SetStatusExStaticText** の呼び出しを含めます。この新しいデフォルトのコードは STATUSEX ダイアログの状態テキストを設定します。今回より、表示される状態テキストは、実行中の操作の種類 (初回インストール、メンテナンス、アンインストール、修復など) に適しています。以前、**SetStatusExStaticText** の呼び出しを手動で追加する必要があり、追加しなかったとき、表示された状態テキストは、実行中の操作に対応していませんでした。

InstallScript MSI プロジェクトを InstallShield 2008 以前から InstallShield 2025 にアップグレードしたとき、初期のバージョンのプロジェクトで 1 つ以上の更新されたイベントがオーバーライドされている場合、**SetStatusExStaticText** コードを手動でこれらのイベントに追加する必要があります。

InstallScript 変数 SHELL_OBJECT_FOLDER の変更点

[ショートカット] ビューにあるフォルダーの "表示名" 設定で **SHELL_OBJECT_FOLDER** (InstallScript または InstallScript MSI プロジェクトの場合) または **<SHELL_OBJECT_FOLDER>** (InstallScript プロジェクトの場合) を指定できます。その後、ショートカットが作成される前にスクリプトで SHELL_OBJECT_FOLDER 変数を設定することにより、このフォルダーの表示名を実行時に定義することができます。通常、ショートカットはファイルの転送時に作成されます。

InstallScript MSI インストールでこの機能を使用する場合、[ショートカット] ビューに表示されるフォルダーの "キー名" 設定で指定される文字は、すべて大文字でなくてはなりません (例、NEWFOLDER1)。

SHELL_OBJECT_FOLDER は、初期化中に IFX_PRODUCT_NAME と同じ値に初期化されるようになりました。これらの変数は、一旦初期化が完了すると同期されません。したがって、片方の変数を変更してから、もう片方の変数も変更する必要がある場合、両方とも手動で変更しなければなりません。

また、この変更の一部として、InstallScript MSI プロジェクトのデフォルトの OnFirstUIBefore イベント ハンドラー コードから次のコード行が削除されました:

```
SHELL_OBJECT_FOLDER = @PRODUCT_NAME;
```

InstallShield 2008 以前で作成された InstallScript MSI プロジェクトで、デフォルトの OnFirstUIBefore コードが変更されている場合、このプロジェクトを InstallShield 2025 にアップグレードしたとき、この行は既存のコードから削除されません。

詳しくは、「SHELL_OBJECT_FOLDER」を参照してください。

InstallScript 関数 FeatureFileEnum における動作の変更

InstallScript 関数 FeatureFileEnum の szQuery パラメーターで疑問符 (?) をワイルドカード文字として使用すると、その関数で疑問符が正確に 1 文字分の代替として使用されるようになりました。以前のバージョンの InstallShield で作成したプロジェクトで、**FeatureFileEnum** で疑問符をワイルドカード文字として使用している場合、InstallShield 2025 にアップグレードする際、InstallScript コードを修正する必要がある場合があります。以前のバージョンの InstallShield では、関数は疑問符を 1 文字または 0 文字の代替として使用していました。

たとえば、新しい InstallScript プロジェクトをデフォルトの設定で作成して、次の InstallScript コードをプロジェクトに追加すると、実行時に表示されるダイアログでは、何も表示されません。

```
#include "ifx.h"

  LIST listFiles;

program

  listFiles = ListCreate( STRINGLIST );
  FeatureFileEnum( MEDIA, "DefaultFeature", "Default?Component", listFiles, NO_SUBDIR );
  SdShowInfoList( "", "", listFiles );

endprogram
```

以前のバージョンの InstallShield で作成されたインストールでは、DefaultComponent がダイアログで表示されていました。

すべての InstallScript インストールで更新 UI の表示が可能

すべての InstallScript プロジェクトで、更新ユーザー インターフェイス (UI) の表示が可能になりました。以前、更新 UI を無効にすることができる紛らわしいオプションが [一般情報] ビューと [プロジェクトの設定] ダイアログ ボックスで提供されていました。更新 UI が無効にされた場合、MEDIA_FLAG_UPDATEMODE_SUPPORTED フラグが MEDIA_FIELD_MEDIA_FLAGS で設定されず、OnSetUpdateMode イベント ハンドラーによって UPDATEMODE 変数が適切に設定されませんでした。この結果、更新 UI が表示されませんでした。

今回より、InstallShield テーブルで ISShowUpdateUI の値 (以前のオプションによって更新されていました) を変更しても、影響は出ません。

以前と同じ機能は、以下の方法で複製することができます:

- UPDATEMODE 変数を設定する前に、OnSetUpdateMode イベントをオーバーライドし、結果を返す関数を更新する。インストールで、更新 UI は表示されません。
- OnMaintUIBefore イベントをオーバーライドし、FeatureRemoveAllInMediaAndLog の呼び出しを FeatureRemoveAllInMedia に変更する。

Certified for Windows Vista 検証スイートの変更

インストールまたはマージ モジュールが Windows Vista ロゴ プログラムの要件を満たしているかどうかを判別するときに使用する 2 つの Certified for Windows Vista 検証スイートが改訂されました。これらのスイートは今回より InstallShield ICE (ISICE) のみで構成されています。マイクロソフトが作成した ICE (ICE01 ～ ICE99) は、インストール パッケージの完全 MSI 検証ツールおよびマージ モジュールのマージ モジュール検証スイートで提供されているため、取り除かれました。

また、両スイートの名前も変更されました。

- Certified for Windows Vista 検証スイート (および InstallShield ICE) は *InstallShield Certified for Windows Vista 検証スイート*に変わりました。
- Certified for Windows Vista マージモジュール検証スイート (および InstallShield ICE) は *InstallShield Certified for Windows Vista マージ モジュール検証スイート*に変わりました。

検証と Certified for Windows Vista ロゴ プログラムの詳しい情報については、次を参照してください:

- Windows ロゴ プログラムの要件
- プロジェクトの検証

アップデートされた InstallScript オブジェクトとオブジェクト テンプレート

InstallShield 2025 InstallScript プロジェクトで使用できる InstallScript オブジェクトと InstallScript オブジェクト テンプレートが大幅に改良されました。

インターネット接続が利用できる場合、InstallShield の [アップデートの確認] 機能を使って、使用中の InstallShield バージョン用の InstallScript オブジェクトとオブジェクト テンプレートのアップデートを取得できます。アップデートを確認するには、[ツール] メニューで [アップデートの確認] をクリックします。InstallShield がウィンドウまたはダイアログを起動して、アップデートを確認します。

新規プロジェクトに影響し、アップグレードされたプロジェクトに影響しない変更

このセクションでは、潜在的に新規プロジェクトに影響し、以前のバージョンからアップグレードされたプロジェクトには影響しない InstallShield の変更について説明されています。アップグレードされたプロジェクトの場合、手動による変更が必要になる場合があります。

Setup.exe および Update.exe ブートストラッパの Unicode バージョンと ANSI バージョン

InstallShield では、基本の MSI プロジェクトで Setup.exe セットアップ ランチャーを Unicode バージョンまたは ANSI バージョンのどちらで作成するかを指定できます。以前、基本の MSI プロジェクトにセットアップランチャーが含まれているとき、常に ANSI バージョンがビルドされていました。Unicode バージョンのビルドはサポートされていませんでした。

Unicode セットアップランチャーは、ターゲット システムで 2 バイト言語のための適切なコード ページが実行されているいないにかかわらず、セットアップランチャーのユーザー インターフェイスで 2 バイト文字を正しく表示することができます。ANSI セットアップランチャーは、ターゲット システムで適切なコード ページが実行されている場合のみ、セットアップランチャー ダイアログで 2 バイト文字を正しく表示します。適切なコード ページが実行されていない場合、これらのダイアログで 2 バイト文字が文字化けして表示されます。

InstallShield 2025 で、新しい基本の MSI プロジェクトのリリースを作成したとき、デフォルトのセットアップランチャーの種類は Unicode になります。また、InstallShield 2025 で新しいパッチ構成または新しい QuickPatch プロジェクトを作成したときも、デフォルトのアップデート ランチャの種類は Unicode になります。

基本の MSI プロジェクトを InstallShield 2008 以前から InstallShield 2025 にアップグレードした場合、既存のリリースのセットアップランチャーの種類はすべて ANSI です。種類は、必要に応じてオーバーライドが可能です。

同様に、基本の MSI プロジェクトまたは QuickPatch プロジェクトを InstallShield 2008 以前から InstallShield 2025 にアップグレードした場合、既存のパッチのアップデート ランチャの種類はすべて ANSI です。種類は、必要に応じてオーバーライドが可能です。

ダイナミック ファイル リンク

プロジェクトにダイナミック ファイル リンクを追加、または変更するとき、InstallShield がコンポーネントを作成する方法について、新しいベスト プラクティスを使用するか、これまでと同様にディレクトリごとに 1 つのコンポーネント作成するのかを指定できます。これらの方式は、基本の MSI プロジェクト、InstallScript MSI プロジェクトおよびマージ モジュール プロジェクトに適用されます。

で新しいダイナミック ファイル リンクを作成すると、InstallShield はデフォルトでベスト プラクティス メソッドを使用します。

InstallShield 2008 以前で作成されたすべてのダイナミック ファイル リンクは、[1 つのディレクトリごとに 1 つのコンポーネント] メソッドを使用します。ダイナミック ファイル リンクがあるプロジェクトを InstallShield 2008 以前から InstallShield 2025 にアップグレードした場合、InstallShield は、既存するダイナミック ファイル リンクのコンポーネントを作成するとき、継続して [ディレクトリごとに 1 つのコンポーネント] メソッドを使用します。アップグレードされたプロジェクトで作成した新しいダイナミック ファイル リンクには、デフォルトでベスト プラクティス メソッドが使われます。2 つのコンポーネント作成メソッドおよび適切なメソッドの選択方法については、「ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する」を参照してください。

IIS Web サイトと関連コンポーネント

InstallShield は、今回より、仮想ディレクトリを一切含まない IIS Web サイトのインストールをサポートできるようになりました。また、Web サイトをコンポーネントに関連付けることも可能です。これらの強化によって、Web サイトに関連付けられた仮想ディレクトリまたはコンポーネントがインストールされると、ターゲット マシン上にその Web サイトが作成されます。

InstallShield 2025 の [IIS の構成] ビューを使って新しい Web サイトを追加すると、InstallShield は自動的にその Web サイトをコンポーネントに関連付けます。IIS Web サイトを含む InstallShield 2008 以前のプロジェクトをアップグレードした場合、その Web サイトは自動的にコンポーネントに関連付けられません。Web サイトとコンポーネントを関連付けるには、[IIS の構成] ビューでその Web サイトを選択したときに表示される [全般] タブを使います。

今回より、Web サイトがコンポーネントと関連付けられている場合、Web サイトの **[アンインストール時に Web サイトを削除する]** チェックボックスは、そのコンポーネントの "パーマネント" 設定 (基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合) または "アンインストール" 設定 (InstallScript プロジェクトの場合) に対応します。つまり、Web サイトの **[アンインストール時に Web サイトを削除する]** チェック ボックスを選択またはクリアすると、自動的にコンポーネントの "パーマネント" 設定または "アンインストール" 設定が適切に更新されます。

QuickPatch パッケージの簡素化

QuickPatch プロジェクトの [詳細] タブにある新しい "QuickPatch の簡素化" 設定は、InstallShield で QuickPatch パッケージのビルド方法を決定します。簡素化された QuickPatch には、簡素化されていない QuickPatch パッケージに比べ、より少ない数の新しいサブ機能とカスタム アクションが含まれています。

場合によって、QuickPatch パッケージの簡素化ができない場合があります。たとえば、インストール済みのファイルを削除するように QuickPatch パッケージを構成した場合、簡素化は行えません。

新しい QuickPatch プロジェクトを作成したとき、"QuickPatch の簡素化" 設定のデフォルト値は [はい] に設定されています。ただし、QuickPatch プロジェクトを InstallShield 2008 以前から InstallShield 2025 にアップグレードした場合、この設定は [いいえ] となっています。この値は、必要に応じて変更が可能です。詳細については、「QuickPatch パッケージを簡素化するかどうかを指定する」を参照してください。

SetARPINSTALLLOCATION カスタム アクションのデフォルト条件

デフォルトで、すべての新しい基本の MSI プロジェクトおよび InstallScript MSI プロジェクトには、ビルトイン InstallShield カスタム アクション SetARPINSTALLLOCATION が含まれています。このカスタム アクション (ARPINSTALLLOCATION プロパティの値を製品のプライマリ フォルダーの完全修飾パスに設定します) は、[インストール]–[実行] シーケンスにシーケンスされ、条件を持ちません。InstallShield 2008 以前では、このカスタム アクションのデフォルト条件は Not Installed でした。デフォルトの [Not Installed] 条件を使用すると、カスタム アクションはメンテナンス モード中に実行されず、その結果 ARPINSTALLLOCATION プロパティの値が空白になります。InstallShield 2008 以前から InstallShield 2025 にアップグレードしたプロジェクトでこの動作を回避するには、[カスタム アクションとシーケンス] ビューを開いて、このカスタム アクションの "インストール実行条件" 設定から Not Installed 値を削除します。

MigrateFeatureStates アクションの新しいシーケンスの場所

デフォルトで、すべての新しい基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで、MigrateFeatureStates アクションは CostFinalize アクションの直後ににシーケンスされるようになりました。[SQL スクリプト] ビューを使って、SQL サポートを新しいプロジェクトに追加すると、InstallShield ビルトイン カスタム アクション ISSQLServerFilteredList は、今回より、MigrateFeatureStates アクションのすぐあとにスケジュールされます。InstallScript 2008 以前では、ISSQLServerFilteredList は MigrateFeatureStates アクションの前にシーケンスされていたため、ランタイム エラー 2601 が発生していました。InstallShield 2008 以前から InstallShield 2025 にアップグレードされたプロジェクトでこのエラーを回避するには、[カスタム アクションとシーケンス] ビューを開いて、[インストール] – [ユーザー インターフェイス] シーケンスで ISSQLServerFilteredList アクションを MigrateFeatureStates の後に移動します。

"テンプレートの概要" プロパティの変更

[一般情報] ビュー、または [リリース] ビューの製品構成で "テンプレートの概要" プロパティの 値を指定できます。たとえば、複数のプラットフォームまたは複数のセミコロンを使用するなど、無効な値を入力しようとすると、InstallShield は警告メッセージ ボックスを表示して、プロパティ値の変更を阻止します。

InstallShield の以前のバージョンでは、無効な値を入力することが可能でした。InstallShield 2008 以前のプロジェクトを InstallShield 2025 にアップグレードした場合に、無効な値が自動的に変更されることはありませんが、必要に応じて、値を手動で更新することができます。詳細については、「"テンプレート概要" プロパティを使用する」を参照してください。

削除されたウェルカム アシスタント

InstallShield を初めて開いたときに表示されていたウェルカム アシスタントは今回より含まれていません。

InstallShield MSIPackageDiff から InstallShield MSI Diff への変更

InstallShield MSIPackageDiff と呼ばれるツールは、今回より InstallShield には含まれていません。InstallShield MSI Diff と呼ばれる、よりパワフルなツールに変更されています。このツールは、[ツール] メニューで [差分] をポイントして InstallShield MSI Diff をクリックして、InstallShield 内からも起動できます。

詳細については、「InstallShield MSI Diff」を参照してください。

プロジェクトを InstallShield 12 以前からアップグレードする

以下の情報は、InstallShield 12 以前から InstallShield 2025 へアップグレードされたプロジェクトに影響が出る可能性がある変更についての説明です。

以前のバージョンの InstallShield で作成されたプロジェクトをアップグレードする

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば .765 というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から .765 を取り除いてください。InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので注意してください。

InstallShield 12 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2008 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2008 にアップグレードすることはできませんので、ご注意ください。

ターゲット システムにおける Windows 9x、Windows NT 4、および Windows Me のサポートの終了

InstallShield では、今回のリリースより、Windows 9x、Windows NT4、および Windows Me システムのためのインストールを作成することができなくなりました。これらのオペレーティング システムを使用しているエンドユーザーが InstallShield 2025 でビルドされたインストールを実行しようとしたときに、エンドユーザーがレガシー オペレーティング システム上でインストールを実行するのを防ぐ起動条件がプロジェクトに含まれていない場合、予期しない結果が発生する可能性があります。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクトでは、エンド ユーザーがレガシー システム上でインストールを実行するとメッセージを表示する起動条件を含めることができます。InstallScript プロジェクトでは、これらのレガシー システムを確認して、それが存在する場合にメッセージを表示する SYSINFO 構造変数を使用する InstallScript コードを追加することができます。

詳しくは、次を参照してください:

- プロジェクト アシスタントでオペレーティング システムを指定する
- [プラットフォーム] タブ
- [プラットフォーム] ダイアログ ボックス
- [プロパティの変更] ダイアログ ボックス / リリース ウィザード — [プラットフォーム] パネル

COM 抽出

今回より、InstallShield を使用して COM サーバーから COM 情報を抽出すると、データは TypeLib テーブルではなく Registry テーブルに書き込まれます。マイクロソフト社は TypeLib テーブルを使用しないことを強く推奨しています。

デフォルトで、ビルド時に未使用のディレクトリを .msi ファイルから自動的に削除する

InstallShield 12 以前のバージョンを使って作成した基本の MSI プロジェクト、InstallScript MSI プロジェクト、またはマージ モジュール プロジェクトを InstallShield 2025 にアップグレードすると、[リリース] ビューの [ビルド] タブに新しく追加された "未使用のディレクトリを保持する" 設定は、デフォルトで [いいえ] に設定されています。したがって、Directory テーブルの Directory 列に一覧表示されているディレクトリが .msi ファイル内の既知の場所で参照されない場合、ビルド時に InstallShield が作成する .msi ファイルの Directory テーブルからそのディレクトリは削除されます。基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで、これはマージ モジュールがマージされてから削除されますが、.msi ファイルに存在するディレクトリのみが削除の対象となります。したがって、マージ モジュールの Directory テーブルに新しい未使用のディレクトリが含まれている場合、そのディレクトリはインストールに追加されます。

ALLUSERS と CustomerInformation ダイアログの変更

InstallShield 2008 より、新しい基本の MSI プロジェクトでは、デフォルトで ALLUSERS プロパティに 1 が設定されます。ほとんどのインストールは、マシンごとに管理者権限を使用して実行される必要があるため、これが推奨される実装です。

InstallShield 12 以前で作成されたプロジェクトを InstallShield 2025 にアップグレードしたとき、ALLUSERS プロパティの値は自動的に変更されません。また、このプロパティが以前のプロジェクトで定義されていない場合も、自動的に追加されません。

また installShield 2008 から、デフォルトで、すべての新しい基本の MSI プロジェクトの CustomerInformation ダイアログは、エンドユーザーが製品をすべてのユーザーにインストールするか、または現在のユーザーのみにインストールするかを指定できるラジオ ボタン グループを表示しないようになっています。このダイアログについては、これが推奨される実装です。

InstallShield 12 以前で作成されたプロジェクトを InstallShield 2025 にアップグレードしたとき、CustomerInformation ダイアログは自動的に変更されません。

詳しくは、次を参照してください:

- ユーザーごとのインストールとマシンごとのインストールの違い
- ALLUSERS

セットアップ前提条件の Windows Server 2003 条件および 64 ビット Windows XP 条件

Windows Server 2003 および 64 ビット Windows XP のオペレーティング システムのバージョン番号は、どちらも 5.2 です。そのため、InstallShield 12 で作成され、かつ Windows Server 2003 を必要とする前提条件を 64 ビット Windows XP システムにインストールすることはできますが、InstallShield 12 で作成され、かつ Windows XP を必要とする前提条件は、64 ビット Windows XP システムにインストールすることはできません。

この問題を解決するため、強化された InstallShield 2008 セットアップ前提条件エディターでは、ターゲット システムがワークステーション、サーバー、またはドメイン コントローラーのいずれかである必要があるかどうかを指定できるようになりました。

Windows Server 2003 要件または 64 ビット Windows XP 要件を含む既存の前提条件については、これを InstallShield 2008 のセットアップ前提条件エディターで開き、[条件] タブで必要に応じて条件を更新することで、問題を回避することができます。[**前提条件を実行するプラットフォームを選択します**] ボックスで、適切なオペレーティング システム要件を選択します。この処理を適切に行なうことで、新しい "製品 (OS) タイプ" 設定がワークステーション、サーバー、またはドメイン コントローラー値に正しく設定されます。

オートメーション インターフェイス

"[保存オプション] ダイアログの表示" 設定が [リリース] ビューから削除されました。したがって、この設定に対応する WebSaveOptionsDlg プロパティが、オートメーション インターフェイスの ISWiRelease オブジェクトで使用できなくなりました。

リリースの "キャッシュ パス"設定における新しいデフォルト値

今回より、[リリース] ビューの "キャッシュ パス" 設定における圧縮リリースのためのデフォルト値が [LocalAppDataFolder]Downloaded Installations に設定されています。以前のデフォルト値 [WindowsFolder]Downloaded Installations は、ロックされたシステムで使用できないことがあります。プロジェクトを InstallShield 12 以前から InstallShield 2025 に移行したとき、"キャッシュ パス" 設定は自動的に変更されないため、必要な場合、この値を手動で変更します。

InstallScript One-Click Install インストール

Setup.exe が InstallScript One-Click Install インストールのセットアップ プレーヤーとして使用されなくなりました。代わりに Setup.ocx が今回より使用されます。Setup.ocx ファイルをインストールに含めるには、[リリース] ビューに新しく追加された "One-Click Install の生成" 設定が [はい] に設定されている必要があります。InstallScript プロジェクトを InstallShield 12 以前から InstallShield 2025 にアップグレードするとき、[リリース] ビューの "デフォルト Web ページの作成" 設定が [はい] に設定されていると、アップグレード中 "ClickOnce Install の生成" 設定は自動的に [はい] に設定されます。反対に、"デフォルト Web ページの作成" 設定が [いいえ] に設定されていて、インストールをインターネット上で配布する予定の場合は、プロジェクトをアップグレードしたあと、"ClickOnce Install の生成" 設定を手動で [はい] に設定する必要があります。

[セットアップを保存/実行] ダイアログ ボックス関連の機能はすべて、InstallShield から削除されました。エンド ユーザーがインストールをダウンロードおよび保存 (またはデスクトップにアイコンを作成) できるオプションを提供するには、これをインストールの Web ページ内に実装して、Web ページを通してダウンロードおよび保存操作を処理する必要があります。

起動された One-Click Install インストールにデータを渡すには、is::CmdLine parameter と CMDLINE スクリプト変数を使用します。以前、Web ページからインストールにデータを渡すことが可能でしたが、このサポートは削除されました。SetProperty の最初のパラメーターに有効な唯一の値は、is::CmdLine です。

詳しくは、「InstallScript プロジェクトの One-Click Install インストール」を参照してください。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクト用のパッチの作成

今回のリリースより、InstallShield はパッチの作成に Patchwiz.dll のバージョン 3.1 を使用します。

DemoShield のサポート

DemoShield は現在販売されていません。サポートも終了しました。このため、今後 InstallShield に DemoShield は統合されません。

Windows Vista 検証スイートにおける変更

インストールが Windows Vista ロゴ プログラムのインストール要件を満たしているかどうかを判別するときに使用する検証スイートの名前が変更されました。

- Windows Vista Quality 品質検証スイート (および InstallShield ICE) の新名称は *Certified for Windows Vista 検証スイート (および InstallShield ICE)* です。
- マージ モジュール品質検証スイート (および InstallShield ICE) の新名称は *Certified for Windows Vista マージ モジュール検証スイート (および InstallShield ICE)* です。

また、3 つの InstallShield ICE が InstallShield ベスト プラクティス スイートに移されました。

検証と Certified for Windows Vista ロゴ プログラムの詳しい情報については、次を参照してください:

- Windows ロゴ プログラムの要件
- プロジェクトの検証

InstallShield ベスト プラクティスとして提供されている検証ツールについての詳細は、次を参照してください:

- ISBP17 (以前は ISICE13 という名前でした)
- ISBP18 (以前は ISICE14 という名前でした)
- ISBP19 (以前は ISICE15 という名前でした)

InstallShield 11.5 以前のプロジェクトをアップグレードする

InstallShield 12 では、セキュリティ関連の問題 (COM/DCOM)、およびその他の様々な理由で起こったインストールの失敗を解決するため、InstallScript MSI アーキテクチャが大幅に改良されました。InstallShield 12 でアーキテクチャが大幅に改良された理由は、これらの問題を解決して InstallScript MSI プロジェクトの安定性を高めるためです。この改良によって、InstallScript カスタム アクションを含む基本の MSI プロジェクト、および InstallScript の安定性も高まりました。

次の表には、新旧のデザインの違いが比較されています。

テーブル 9 ・ アーキテクチャの変更

InstallShield 11.5 以前	InstallShield 12 以降
各 InstallScript カスタム アクションは ISScriptBridge.dll (C++ MSI DLL) によって処理されます。この処理により、実際のスクリプトの実行が実行中の IDriver.exe プロセスに渡されます。	各 InstallScript カスタム アクションは、ISSetup.dll (単一の C++ MSI DLL) によって処理されます。これには完全な InstallScript エンジンが含まれます。
エンジン ファイルのセットが必要。これらは、プライマリ インストールが開始する前に ISScript.msi によってインストールされなくてはなりません。	エンジン ファイルまたは ISScript.msi ファイルは不要。ISSetup.dll には、必要な情報がすべてが含まれています。

テーブル 9 • アーキテクチャの変更 (続き)

InstallShield 11.5 以前	InstallShield 12 以降
すべての InstallScript カスタム アクションは、共有 IDriver.exe プロセスで実行します。IDriver.exe プロセスは最終のシャットダウン カスタム アクションが必要になるまで存在し続けます。	各 InstallScript カスタム アクションは非依存型で、独立したライフサイクルを持ちます。カスタム アクション エントリ ポイントから、ISSetup.dll がスクリプト エンジンを開始し、関連スクリプトを実行してから、エンジンをシャットダウンします。

InstallShield 12 で導入されたアーキテクチャの変更における利点

InstallShield 12 以降のアーキテクチャには、以前のアーキテクチャと比較して次のような利点があります。

- 実行中の IDriver.exe の検出に関連するエラー 1607、1608、またはその他の COM/DCOM 実行時エラーをエンドユーザーが経験することはありません。以前のモデルでこれらのエラーを解決するためには、場合によって DCOM 設定の変更が必要なため、容易ではありませんでした。また実行中のオブジェクト テーブルに依存していたことで、"ユーザーの簡易切り替え" および Windows Terminal Services などのすべての利用シナリオにおいて、モデルの脆弱性をもたらしていました。
- エンジン バインディングはスタティックです。Program Files¥Common Files¥InstallShield に、共有エンジンファイルはインストールされません。11.5 以前のバージョンの InstallShield では共有エンジン ファイルに依存するため、モデルに脆弱性をもたらしました。これは個別のインストールが隔離されていないためです。1 つのエンジン バージョンを変更すると、既存のインストールを破損する可能性があります。特にインストーラーの隔離は安定性の重要な鍵となります。
- ISScript.msi (InstallScript エンジン インストーラー) ファイルが必要なくなりました。以前のモデルでは、プライマリ .msi ファイルの前に ISScript.msi を実行する必要がありました。完全アプリケーションが単一の .msi ファイルに含まれていないため、InstallScript カスタム アクションがない基本の MSI インストールでは、インストール時にブートストラップが必要でした。Active Directory を使って環境管理を行っている企業にとって、これは柔軟性に欠けていました。また、メインの .msi ファイルに必要な情報がすべてが含まれていることを期待している上級ユーザーを困惑させる原因にもなりました。InstallScript MSI インストールでは、要件が以前のアーキテクチャと変わらないようにするため、これまで同様ブートストラップが必要です。ただし、InstallScript カスタム アクションがある基本の MSI インストールでは、ブートストラップは必要ありません。
- インストールの実行に必要な InstallShield カスタム アクションの数が少なくなりました。以前のモデルは、エラーの起きやすいスタートアップおよびシャットダウン カスタム アクションの密接な結合に依存していました。必要なカスタム アクションのうちの 1 つが削除または移動された場合、インストールが適切に動作しませんでした。

InstallShield 12 で導入されたアーキテクチャの変更におけるマイナス点

InstallShield 12 以降のアーキテクチャには、次のようなマイナス点があります。

- InstallScript イベントは基本の MSI プロジェクトおよびマージ モジュール プロジェクトでは利用できないため、これらのプロジェクトの InstallScript コードは、InstallScript カスタム アクションを使って実行しなくてはなりません。グローバル変数は、これらのカスタム アクションが起動されたときの状態を共有しません。したがって、InstallScript カスタム アクションでグローバル変数を宣言してから、スクリプトでグローバル変数を使用します。ただし、基本の MSI インストールに 2 番目の InstallScript カスタム アクションが含まれている場合、最初のスクリプトで宣言されたグローバル変数を 2 番目のスクリプトで使用することはできません。

以前のモデルでは、InstallScript 開発者は基本の MSI プロジェクトですべての InstallScript イベントのスクリプト コードおよび InstallScript グローバル変数の共有状態を使用することができました。このため、セットアップ作成者は、インストール全体にわたって総体的な視点および自然に流れる感覚でカスタム アクションを作成することができました。しかしながら、プログラミングでは一般的にグローバル変数を使用しないほうがいいため、この新しい制限によって、実際にはより良い InstallScript コードが書かれることになります。

- メモリ内のオブジェクトはカスタム アクションの呼び出し間で共有できるよう、シリアル化しなくてはなりません。以前のモデルでは、InstallScript 開発者はグローバル変数に複雑なオブジェクト ベースのデータを格納することができました。前述の利点の通り、これによってより良い InstallScript コードとなります。

InstallShield 11.5 以前のプロジェクトのアップグレード

InstallShield 12 で導入された広範囲にわたる再アーキテクチャは、InstallShield 11.5 以前で作成されたプロジェクトを InstallShield 2025 にアップグレードするとき、場合によって、手動で変更を行う必要があります。



タスク ***プロジェクトをアップグレードするには、以下の手順に従います:***

1. プロジェクトを変換する前に、プロジェクト ファイルと InstallScript ファイルのバックアップ バージョンを作成してください。
2. InstallShield 2025を開きます。
3. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. [参照] をクリックして、アップグレードする InstallShield 11.5 以前のプロジェクトのプロジェクト ファイル (.ism) を選択してください。プロジェクトをアップグレードするかどうかを指定するように求めるダイアログ ボックスが開きます。
5. [はい] をクリックします。

この後、プロジェクトがアップグレードされ、プロジェクト ファイル (.ism ファイル) のバックアップ コピーが保存されます。

場合によって、アップグレードされたプロジェクトは、手動による変更が必要です。詳しくは、次を参照してください。

- InstallScript カスタム アクションを含む InstallShield 11.5 または以前の基本の MSI プロジェクトをアップグレードする
- InstallShield 11.5 以前の InstallScript MSI プロジェクトをアップグレードする
- InstallShield 11.5 以前の InstallScript プロジェクトをアップグレードする
- InstallScript カスタム アクションを含む InstallShield 11.5 または以前の QuickPatch プロジェクトをアップグレードする
- InstallShield 11.5 以前の InstallScript MSI プロジェクトに標準のパッチを作成する
- InstallShield 11.5 以前の InstallScript MSI オブジェクト プロジェクトまたはこの種類のオブジェクトを含むプロジェクトをアップグレードする

InstallScript カスタム アクションを含む InstallShield 11.5 または以前の基本の MSI プロジェクトをアップグレードする

InstallShield 11.5 以前で作成された基本の MSI プロジェクトを InstallShield 2025 にアップグレードするときの詳しい説明については、以下のセクションを参照してください。

グローバル変数、グローバル ポインター、および SUPPORTDIR

InstallShield 12 以降では、基本の MSI インストールで InstallScript カスタム アクションが実行されたとき、コンパイルされた InstallScript が、アクションが呼び出される前にロードされ、アクションが完了された後、アンロードされます。このため、各 InstallScript カスタム アクションは、それぞれのセッションで実行され、実行ごとに全 InstallScript エンジンがロードおよびアンロードされます。この動作は、InstallShield 11.5 以前での動作とは異なります。InstallShield 11.5 以前では、コンパイルされた InstallScript は、InstallScript で使用された最初の InstallScript カスタム アクションが実行される前に一度だけロードされ、すべての InstallScript カスタム アクションが完了したあと、インストールの最後でアンロードされます。

個々の InstallScript カスタム アクションの呼び出し間で、グローバル変数とポインターが保持されなくなったことが、この動作の変更による最も大きな影響です。

- 複数のカスタム アクションの呼び出し間で値を保存する必要がある場合、レジストリ、Windows Installer プロパティ、または外部データ ファイルなどの外部メカニズムを使用して、呼び出し間の情報を保存しなければなりません。遅延/コミット/ロールバック InstallScript カスタム アクションで Windows Installer プロパティを使用する選択をした場合、「Windows Installer プロパティと遅延/コミット/ロールバック InstallScript カスタム アクション」にあるガイドラインを参照してください。
- カスタム アクションの呼び出し間で COM オブジェクトまたは他のグローバル オブジェクトを使用する必要がある場合、個々のカスタム アクションの呼び出しのオブジェクトを初期化して、それを有効にする必要があります。

また、この変更により、個々のカスタム アクションでは、それぞれ固有の SUPPORTDIR が初期化および使用されます。したがって、各カスタム アクションの起動には、それぞれ一意の SUPPORTDIR が用意されるため、SUPPORTDIR のファイルを使用して、個々の呼び出し間で情報を共有することはできません。情報は、FOLDER_TEMP または他のファイルの場所を使用して共有することができます。

FOLDER_TEMP のパスは、すべての InstallScript カスタム アクションに対して同一とは限りませんので注意してください。システム コンテキストで実行される InstallScript カスタム アクションと実行されない InstallScript カスタム アクションがある場合、パッケージが昇格された状態で実行された場合、それぞれ異なる一時パスが与えられます。InstallScript カスタム アクションは、異なるユーザーのコンテキストで実行されるため、一時ディレクトリにファイルを保存して、それをあとで取り出す操作は、特定のシナリオで適切に動作しないことがあります。

Windows Installer プロパティと遅延/コミット/ロールバック InstallScript カスタム アクション

基本の MSI インストールの遅延/コミット/ロールバックの InstallScript カスタム アクションは、一部のビルトイン Windows Installer プロパティ (CustomActionData、ProductCode、および UserSID) にのみアクセスすることができます。遅延/コミット/ロールバックの実行中に、InstallScript カスタム アクションで他のプロパティ (例、SUPPORTDIR) にアクセスする場合は、それらを CustomActionData に渡す必要があります。これを行うには、すぐにプロパティを設定するタイプのカスタム アクション (例、**MsiSetProperty** 関数をもつ即時 InstallScript カスタム アクション) をスケジュールして、カスタム アクションの名前に一致するプロパティを設定します。このプロパティの値が、遅延のカスタム アクション内の CustomActionData プロパティから取得できるようになります。

たとえば、SUPPORTDIR のようなプロパティにアクセスする場合、MyCustomActionName という名前で、MyCustomActionName プロパティを [SUPPORTDIR] に設定する即時カスタム アクションを作成して、MsiGetProperty 呼び出しで "SUPPORTDIR" を "CutomActionData" で置き換えることもできます。

MsiGetProperty(hMSI, "CustomActionData", szSupportDir, nLen)

詳細については、次を参照してください。

- 遅延/コミット/ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う
- 遅延実行カスタム アクションのコンテキスト情報を取得する

提供されていない Windows Installer プロパティを補うことを忘れると、インストール時に予期しない結果が起きることがありますので注意してください。たとえば、遅延アクションでは、MsiGetProperty(hMSI, "INSTALLDIR", szInstallDir, nLen) が szInstallDir を空の文字列に設定するため、szInstallDir ^ szFile は、[INSTALLDIR] のファイルの代わりに、現在のディレクトリにあるファイルを参照する場合があります。遅延カスタム アクションの現在のディレクトリは、通常 [SystemFolder] です。

遅延/コミット/ロールバック InstallScript カスタム アクションは、ProductLanguage プロパティへのアクセスがありません。インストールに複数言語のサポートが含まれていて、かつエンドユーザーがインストールを実行している言語へのアクセスが必要な遅延/コミット/ロールバック InstallScript カスタム アクションもあるとき、インストールは、この言語をインストールのデフォルト言語であると仮定します。これは、エンドユーザーがインストールをデフォルト言語以外の言語で実行するとき、問題となります。ただし、遅延/コミット/ロールバック カスタム アクションは通常、ユーザー インターフェイスを表示しないため、これが問題になることは、ごく稀にしかありません。

定義済み InstallScript イベント ハンドラーの関数

定義済みの InstallScript イベント ハンドラーの関数は今回より、InstallScript カスタム アクションを含む基本の MSI プロジェクトで使用できなくなりました。InstallShield 11.5 以前では、次の InstallScript イベント ハンドラーの関数が基本の MSI プロジェクトで提供されていました。

- OnBegin
- OnMoving
- OnMoved
- OnEnd

これらのイベント ハンドラーの関数は、InstallScript カスタム アクションがある基本の MSI プロジェクトではサポートされていません。一旦プロジェクトが InstallShield 12 以降で再ビルドされると、それ以後呼び出されなくなります。これらのイベント ハンドラー関数は、InstallShield 12 以降でも続けてコンパイル可能ですが、呼び出されることはありません。

これらのイベント ハンドラー関数をプロジェクトで使用する場合、これらの関数を呼び出すカスタム アクションを手動でスケジュールする必要があります。詳しくは、「基本の MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール」をご覧ください。

InstallShield InstallScript スクリプト作成エンジン マージ モジュール

InstallShield 12 より、InstallShield スクリプト作成エンジン マージ モジュールを、基本の MSI プロジェクトに含めることができなくなりました。InstallShield 11.5 以前では、このマージ モジュールを利用することにより、Setup.exe を含まないインストールで InstallScript エンジンを配布することができました。InstallShield 12 以降では、

Setup.exe が使用されるされないにかかわらず、InstallScript エンジンが、InstallScript カスタム アクションをもつ基本の MSI インストールに ISSetup.dll の一部として自動的に含められます。したがって、InstallShield 2025 プロジェクトで、このマージ モジュールを使う必要はありません。

InstallShield 2025 を使用して、このマージ モジュールがあるプロジェクトのリリースをビルドしようとすると、次のようなビルド エラーが発生することがあります。

ISDEV : エラー -4075: ファイルが見つかりません。機能 'NewFeature1' にモジュール 'InstallShieldScriptingEngine.4F635B62_07BF_4779_B74E_D80C29D508E3:0' をマージ中に、エラーが発生しました。

このビルド エラーを解決するには、このマージ モジュールをプロジェクトから削除します。マージ モジュールは、[再配布可能ファイル] ビューにあるチェック ボックスをクリアすると削除されます。

InstallScript エンジン ファイルへの変更

InstallShield 12 以降では、InstallScript カスタム アクションを含む基本の MSI プロジェクトは、InstallScript エンジン ファイルを次のディレクトリにインストールしません。

<COMMONFILES>¥InstallShield¥Driver¥<バージョン>¥Intel 32

InstallShield 11.5 以前の InstallScript カスタム アクションがある基本の MSI プロジェクトでは、いくつかのファイルは Binary テーブルに格納されていました。

- Setup.inx
- Isconfig.ini
- Isrt.dll
- ISScriptBridge.dll
- _isresXXXX.dll (ここで、XXXX は言語です。.dll は、インストールに含まれている各言語に対して 1 つ含まれました。)
- StringxXXXX.txt (ここで、XXXX は言語です。.txt は、インストールの各言語に対して 1 つ含まれました。)

InstallShield 12 以降では、これらのファイル (使われなくなった ISScriptBridge.dll を除く) はすべて、ISSetup.dll ファイル内に格納されます。ISSetup.dll は、Binary テーブルに格納される唯一のファイルです。

InstallScript エンジン ファイルの変更が原因によるアップグレードの問題は、現在、報告されていません。これらの変更は、情報提供のみを目的として報告されています。

基本の MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール

InstallShield 12 以降で、定義済みの InstallScript イベント ハンドラーの関数は今回より、InstallScript カスタム アクションを含む基本の MSI プロジェクトで使用できません。InstallShield 11.5 以前では、次の InstallScript イベント ハンドラーの関数が、基本の MSI プロジェクトと InstallScript オブジェクト プロジェクトで提供されていました。

- OnBegin
- OnMoving
- OnMoved
- OnEnd

これらのイベント ハンドラーの関数は、InstallScript カスタム アクションがある基本の MSI プロジェクトではサポートされていません。一旦プロジェクトが InstallShield 2025 で再ビルドされると、それ以後呼び出されなくなります。これらのイベント ハンドラー関数は、InstallShield 2025 でも続けてコンパイル可能ですが、呼び出されることはありません。

同様に、これらの定義済み InstallScript イベント ハンドラー関数は、InstallShield 2025 で開かれたときに自動的にマージ モジュール プロジェクトに変換される InstallScript MSI オブジェクト プロジェクトではサポートされていません。これらの関数は、一旦変換されたマージ モジュール プロジェクトが InstallShield 2025 でビルドされると、呼び出されません。

基本の MSI または InstallScript MSI オブジェクト プロジェクトにこれらのイベントがあるときに、プロジェクトを InstallShield 2025 にアップグレードする場合、イベント ハンドラー関数を呼び出すカスタム アクションを手動でスケジュールする必要があります。以下は、その手順です。



タスク

定義済み InstallScript イベント ハンドラー関数を呼び出す InstallScript カスタム アクションを手動でスケジュールするには、以下の手順に従います:

1. InstallShield 2025 で、アップグレード済みのプロジェクトを開きます。
2. InstallScript ファイルへ適切な変更を行います。
   a. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
   b. スクリプトで OnBegin、OnMoving、OnMoved、および OnEnd イベント ハンドラー関数を検索します。ビューの中央にあるペインで関数名をクリックすると、簡単に関数を検索することができます。
   c. 関数の名前を代替名に変更して、ifx.h に自動的に含まれる既存の関数プロトタイプと競合するのを避けるようにします。例:
      - MyOnBegin
      - MyOnMoving
      - MyOnMoved
      - MyOnEnd
   d. 既存の関数を、単一 HWND パラメーターを取得するようにアップグレードします。例:

      ```
      function MyOnBegin(hMSI) begin end;
      ```

   e. これらの新しい関数の適切なプロトタイプを追加します。

      ```
      export prototype MyOnBegin(HWND);
      export prototype MyOnEnd(HWND);
      export prototype MyOnMoved(HWND);
      export prototype MyOnMoving(HWND);
      ```

3. 名前が変更された InstallScript イベント ハンドラー関数を呼び出す InstallScript カスタム アクションを追加します:
   a. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI プロジェクトの場合) または **[カスタム アクション]** (マージ モジュール プロジェクトの場合) をクリックします。
   b. 中央のペインで、**[カスタム アクション]** エクスプローラーを右クリックして、**[新しい InstallScript]** をクリックします。InstallScript カスタム アクションが追加されます。

c. カスタム アクションの名前を入力します。この名前には、InstallScript 関数の名前を変更したときに使用した名前を使用します。例:

- MyOnBegin
- MyOnMoving
- MyOnMoved
- MyOnEnd

d. 作成した新しい InstallScript カスタム アクションを選択します。

e. **"関数名"** 設定を、InstallScript 関数の名前に設定します。

f. **"スクリプト内実行"** 設定で、下のテーブルで示されている値を指定します。

4. InstallScript カスタム アクションをインストールの適切な箇所にスケジュールします。

a. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。

b. InstallScript カスタム アクションの前にくるアクションまたはダイアログを右クリックして、**[挿入]** をクリックします。**[アクションの挿入] ダイアログ ボックス**が開き、プロジェクトに現在関連付けられているアクションおよびダイアログの一覧が表示されます。

c. 作成した InstallScript カスタム アクションを選択します。必要に応じて、InstallScript イベントの起動についての条件を **[条件]** ボックスに入力します。

d. **[OK]** をクリックします。

以前の機能にできるだけ近づけるため、ステップ 3f のスクリプト内実行の設定と、ステップ 4b の InstallScript カスタム アクションのスケジュールを次のように行います。

テーブル 10 ・ カスタム アクションのスケジュール

カスタム アクション	スクリプト内実行	シーケンス中の場所
MyOnBegin	即時 (デフォルト)	[インストール]–[ユーザー インターフェイス] シーケンスの最初のアクションの 1 つです。
MyOnMoving	システム コンテキストの遅延	[インストール] の [実行] シーケンス、InstallInitialize アクションと AllocateRegistrySpace アクションの間。
MyOnMoved	システム コンテキストの遅延	[インストール] の [実行] シーケンス、ScheduleReboot アクションと InstallFinalize アクションの間
MyOnEnd	即時 (デフォルト)	[インストール] の [実行] シーケンスの最後のイベント、InstallFinalize アクションの後(以前のリリースで、OnEnd は、インストールが完了した結果として他のすべてのイベントの後で呼び出されていました。)

次の事項に注意してください。

- グローバル変数とポインターは、個々の InstallScript カスタム アクションの呼び出し間で保持されなくなりました。また、個々の InstallScript カスタム アクションでは、それぞれ固有の SUPPORTDIR が初期化され、使用されます。したがって、SUPPORTDIR のファイルでは、個々の呼び出し間で情報を共有することはできませ

ん。詳細については、「グローバル変数、グローバル ポインター、および SUPPORTDIR」を参照してください。

- ほとんどの Windows Installer プロパティは、遅延/コミット/ロールバック InstallScript カスタム アクションで使用することはできません。詳細については、「Windows Installer プロパティと遅延/コミット/ロールバック InstallScript カスタム アクション」を参照してください。

InstallShield 11.5 以前の InstallScript MSI プロジェクトをアップグレードする

InstallShield 11.5 以前で作成された InstallScript MSI プロジェクトを InstallShield 2025 にアップグレードするときの詳しい説明については、以下のセクションを参照してください。

グローバル変数、グローバル ポインター、および SUPPORTDIR

InstallShield 2025 およびそれ以前のバージョンでは、InstallScript MSI インストールが初期化されたとき、コンパイルされた InstallScript ファイルがロードされ、必要に応じてインストール中に、この "メイン" のロードされた InstallScript ファイルの InstallScript イベントが呼び出されます。

ただし、InstallShield 12 以降では、基本の MSI インストールで InstallScript 内の関数が Windows Installer エンジンによって InstallScript カスタム アクションとして呼び出されると、カスタム アクションは、InstallScript カスタム アクションと同じ動作をします。つまり、関数は、完全に別々にロードされたスクリプト インスタンスで実行されます。スクリプトは、アクションが呼び出される前にロードされ、アクションが完了した後で、アンロードされます。このため、Windows Installer エンジンによって、InstallScript カスタム アクションとして呼び出された各 InstallScript 関数は、それ自身のセッションで実行され、実行ごとに全 InstallScript エンジンがロードおよびアンロードされます。この動作は、InstallShield 11.5 以前での動作とは異なります。InstallShield 11.5 以前では、カスタム アクションは、初期化中にロードされ、インストールの最後でアンロードされたロードされた単一のメイン スクリプト ファイル インストールで呼び出されていました。

この動作の変更による主な影響は、InstallScript カスタム アクションとして呼び出された InstallScript 関数は、メインの実行スクリプト ファイルで保持されるグローバル変数およびポインターへアクセスしなくなったいうことです。

- メインの実行 InstallScript ファイルとインタラクトして、グローバル変数の使用や変更、またはメインの実行スクリプト ファイル によって保持されているオブジェクトの変更などのタスクを行う必要がある場合 (アンインストールのステータス バージョンの更新や、スクリプト操作のログ記録を含みます)、関数は、InstallScript カスタム アクションとしてではなく、InstallScript イベントから呼び出されなければなりません。InstallScript イベントから呼び出された関数は、メインの InstallScript インスタンスで実行されます。(例外: いくつかの OnFilesInUse などの InstallScript エラー イベントは、Windows Installer エンジンからのエラー メッセージの結果として呼び出されます。Windows Installer エンジンは、これらの InstallScript エラー イベントを InstallScript カスタム アクションとして起動します。このため、InstallScript カスタム アクションの制限は、これらの InstallScript イベントにも適用します。)
- 複数の InstallScript カスタム アクションの呼び出し間で値を保存する必要がある場合、レジストリ、Windows Installer プロパティ、または外部データ ファイルなどの外部メカニズムを使用して、呼び出し間の情報を保存しなければなりません。遅延/コミット/ロールバック InstallScript カスタム アクションで Windows Installer プロパティを使用する選択をした場合、「Windows Installer プロパティと遅延/コミット/ロールバック InstallScript カスタム アクション」にあるガイドラインを参照してください。
- カスタム アクションの呼び出し間で COM オブジェクトまたは他のグローバル オブジェクトを使用する必要がある場合、個々のカスタム アクションの呼び出しのオブジェクトを初期化して、それを有効にする必要があります。

また、この変更により、個々のカスタム アクションでは、それぞれ固有の SUPPORTDIR が初期化および使用されます。したがって、各カスタム アクションの起動には、それぞれ一意の SUPPORTDIR が用意されるため、SUPPORTDIR のファイルを使用して、個々の呼び出し間で情報を共有することはできません。情報は、FOLDER_TEMP または他のファイルの場所を使用して共有することができます。

FOLDER_TEMP のパスは、すべての InstallScript カスタム アクションに対して同一とは限りませんので注意してください。システム コンテキストで実行される InstallScript カスタム アクションと実行されない InstallScript カスタム アクションがある場合、パッケージが昇格された状態で実行された場合、それぞれ異なる一時パスが与えられます。InstallScript カスタム アクションは、異なるユーザーのコンテキストで実行されるため、一時ディレクトリにファイルを保存して、それをあとで取り出す操作は、特定のシナリオで適切に動作しないことがあります。

Windows Installer プロパティと遅延/コミット/ロールバック InstallScript カスタム アクション

InstallScript MSI インストールの遅延/コミット/ロールバックの InstallScript カスタム アクションは、一部のビルトイン Windows Installer プロパティ (CustomActionData、ProductCode、および UserSID) にのみアクセスすることができます。遅延/コミット/ロールバックの実行中に、InstallScript カスタム アクションで他のプロパティ (例、SUPPORTDIR) にアクセスする場合は、それらを CustomActionData に渡す必要があります。これを行うには、すぐにプロパティを設定するタイプのカスタム アクション (例、MsiSetProperty 関数をもつ即時 InstallScript カスタム アクション) をスケジュールして、カスタム アクションの名前に一致するプロパティを設定します。このプロパティの値が、遅延のカスタム アクション内の CustomActionData プロパティから取得できるようになります。

たとえば、SUPPORTDIR のようなプロパティにアクセスする場合、MyCustomActionName という名前で、MyCustomActionName プロパティを [SUPPORTDIR] に設定する即時カスタム アクションを作成して、MsiGetProperty 呼び出しで "SUPPORTDIR" を "CutomActionData" で置き換えることもできます。

```
MsiGetProperty(hMSI, "CustomActionData", szSupportDir, nLen)
```

詳細については、次を参照してください。

- 遅延/コミット/ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う
- 遅延実行カスタム アクションのコンテキスト情報を取得する

提供されていない Windows Installer プロパティを補うことを忘れると、インストール時に予期しない結果が起きることがありますので注意してください。たとえば、遅延アクションでは、MsiGetProperty(hMSI, "INSTALLDIR", szInstallDir, nLen) が szInstallDir を空の文字列に設定するため、szInstallDir ^ szFile は、[INSTALLDIR] のファイルの代わりに、現在のディレクトリにあるファイルを参照する場合があります。遅延カスタム アクションの現在のディレクトリは、通常 [SystemFolder] です。

遅延/コミット/ロールバック InstallScript カスタム アクションは、ProductLanguage プロパティへのアクセスがありません。インストールに複数言語のサポートが含まれていて、かつエンドユーザーがインストールを実行している言語へのアクセスが必要な遅延/コミット/ロールバック InstallScript カスタム アクションもあるとき、インストールは、この言語をインストールのデフォルト言語であると仮定します。これは、エンドユーザーがインストールをデフォルト言語以外の言語で実行するとき、問題となります。ただし、遅延/コミット/ロールバック カスタム アクションは通常、ユーザー インターフェイスを表示しないため、これが問題になることは、ごく稀にしかありません。

DoInstall 関数への変更

DoInstall 関数は今回より、既に実行されているインストール エンジンを使用する代わりに、子インストールのセットアップ実行可能ファイルを起動して、子インストールを実行します。この変更は自動的に行われますので、既存の InstallScript コードへの変更は必要ありません。インストール実行可能ファイルは初期化が必要なため、子インストールの初期化は、以前のバージョンよりも若干時間がかかる場合があります。詳細については、「DoInstall」を参照してください。

今回より、**DoInstall** の呼び出しは、**LaunchAppAndWait** の呼び出しに似ています。インストールが、CD-ROM、フロッピー ディスクなどのリムーバブル メディアから呼び出されたとき、Disk1 の Setup.exe ファイルは、インストール中、常に使用できる状態にあるとはかぎりません。(実行中に、Setup.exe が使用できなくなった場合、オペレーティング システムでプロンプトが表示され、エンドユーザーに正しいディスクを挿入するように求めてくることがあります。これにより、インストールが失敗することがあります。)したがって、この問題を回避するために、Setup.exe ファイルが Temp フォルダーにコピーされ、インストールがそこから再起動されます。元の Setup.exe は、ここで終了します。ただし、この処理が発生すると、**DoInstall** (または、呼び出された場合、**LaunchAppAndWait**) は、インストールが完了したと仮定して動作し、待機しません。

この問題のワークアラウンドは、いくつかあります。その 1 つはまず、子インストールを起動するときに、/Clone_wait パラメーターを使用することです。このワークアラウンドの実行すると、起動されたインストールで、起動された元のプロセスが実行中のまま保たれ、親インストールは待機します。ただし、このワークアラウンドでは、Setup.exe を含む元の CD がインストールの途中で使用できない状態になる場合、問題が発生する場合があります。この例として、インストールの途中で一枚目の CD が使用できなくなる複数 CD のインストールがあげられます。

この問題を回避する唯一の方法は、起動されたプロセスの子プロセスの ID を判別するコードを追加して、子プロセスが完了するのを待機することです。

Disk1 ファイルへの変更

InstallShield は以前、次のファイルをビルドされたインストールの Disk1 イメージに配置していましたが、この処理は必要なくなったため、今後実行されません。

ISScript<バージョン>.msi

InstallShield 12 以降では、次のファイルが、ビルドされたインストールの Disk1 イメージに配置されます。

ISSetup.dll

Disk1 ファイルの変更が原因によるアップグレードの問題は、現在報告されていません。これらの変更は、情報提供のみを目的として報告されています。

InstallScript エンジン ファイルへの変更

InstallShield 12 以降では、InstallScript MSI インストールは、InstallScript エンジン ファイルを次のディレクトリにインストールしません。

<COMMONFILES>¥InstallShield¥Professional¥Runtime¥<MajorVersion>¥<MinorVersion>¥Intel32

InstallShield 11.5 以前の InstallScript MSI プロジェクトでは、いくつかのファイルは Binary テーブルに格納されていました。

- Setup.inx
- Isconfig.ini
- Isrt.dll

- ISScriptBridge.dll
- _isresXXXX.dll（ここで、XXXX は言語です。.dll は、インストールに含まれている各言語に対して 1 つ含まれました。）
- StringxXXXX.txt（ここで、XXXX は言語です。.txt は、インストールの各言語に対して 1 つ含まれました。）

InstallShield 12 以降では、これらのファイル（使われなくなった ISScriptBridge.dll を除く）はすべて、ISSetup.dll ファイル内に格納されます。ISSetup.dll は、Binary テーブルに格納される唯一のファイルです。

InstallScript エンジン ファイルの変更が原因によるアップグレードの問題は、現在、報告されていません。これらの変更は、情報提供のみを目的として報告されています。

InstallScript 関連のカスタム アクション

InstallShield 12 では、いくつかの InstallScript 関連のビルトイン カスタム アクションが、InstallScript MSI プロジェクトで提供されなくなりました。InstallScript MSI プロジェクトを InstallShield 11.5 以前から InstallShield 2025 へアップグレードした場合、InstallShield によって、これらのビルトイン カスタム アクションはプロジェクトから削除されます。この削除のため、以前これらのカスタム アクションによって呼び出されていた既存の InstallScript イベント ハンドラー関数は、今回より、InstallScript から直接呼び出されます。詳細については、「InstallScript MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール」を参照してください。

Setup.exe ファイルのない InstallScript MSI インストールを配布するための変更

パッケージを Active Directory などのテクノロジを通して配布できるようにするために、InstallScript MSI インストールが Setup.exe ファイルなしに実行されるのを許可するには、「ナレッジベース記事 Q108166 – HOWTO: Deploying an MSI Wrapped with an InstallShield Script–Based Setup.exe (MSI を InstallShield スクリプト ベースの Setup.exe でラップして配布する)」で説明されている手順を使用できる場合があります。ただし、この記事で説明されている手順を適用して、ユーザーがインストール時に .msi ファイルを直接起動すると、一部の InstallScript イベント（OnMoved など）が起動されなくなります。これらのイベントが呼び出されない理由は、InstallScript 関連のカスタム アクションが削除されたため、以前 InstallScript 関連のビルトイン カスタム アクションのよって呼び出されていた一部の InstallScript イベント ハンドラー関数が呼び出されないためです。

詳細については、「InstallScript MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール」を参照してください。

InstallScript MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール

InstallShield 12 では、いくつかの InstallScript 関連のビルトイン カスタム アクションが、InstallScript MSI プロジェクトで提供されなくなりました。InstallScript MSI プロジェクトを InstallShield 11.5 以前から InstallShield 2025 へアップグレードした場合、によって、これらのビルトイン カスタム アクションはプロジェクトから削除されます。この削除のため、以前これらのカスタム アクションによって呼び出されていた既存の InstallScript イベントは、今回より、InstallScript から直接呼び出されます。この記事では、削除されたカスタム アクションが確認でき、スケジュールが InstallShield 11.5 以前で使用されたスケジュールとほぼ同一になるように、これらの InstallScript イベントを呼び出すカスタム アクションをどのように追加するかが説明されています。

パッケージを Active Directory などのテクノロジを通して配布できるようにするために、InstallScript MSI インストールがSetup.exe ファイルなしに実行できるようにした場合、削除されたカスタム アクションが原因で InstallScript MSI インストールでも影響が出ることがあります。これを実装するための手順は、ナレッジベース記事「Q108166 – HOWTO: Deploying an MSI Wrapped with an InstallShield Script–Based Setup.exe (MSI を InstallShield ス

クリプト ベースの Setup.exe でラップして配布する)」を参照してください。この記事で説明されている手順を適用して、ユーザーがインストール時に .msi ファイルを直接起動すると、一部の InstallScript イベント (OnMoved など) が起動されなくなりますますので注意してください。これらのイベントが呼び出されない理由は、InstallScript 関連のカスタム アクションが削除されたため、以前 InstallScript 関連のビルトイン カスタム アクションのよって呼び出されていた一部の InstallScript イベント ハンドラー関数が呼び出されないためです。また、インストールを Q108166 記事にしたがって構成した場合、以下の手順にしたがって、ビルトイン InstallScript イベントを呼び出すカスタム アクションを追加する必要があります。

次の InstallScript 関連のビルトイン カスタム アクションが、InstallShield 12 で提供されなくなりました。これらのカスタム アクションは、InstallShield 11.5 以前から InstallShield 2025 にアップグレードされたとき、InstallScript MSI プロジェクトから削除されます:

- ISCleanupSuccess
- ISCleanUpSuspend
- ISCleanUpUserTerminate
- ISCleanupFatalExit
- ISMsiServerStartup
- ISRebootPatchHandler
- ISRollbackCleanup
- ISStartup
- OnCheckSilentInstall (このカスタム アクションの変更に関する詳細は、「OnCheckSilentInstall」を参照してください。)
- OnFeaturesInstalled
- OnFeaturesInstalling
- OnInstallFilesActionAfter
- OnInstallFilesActionBefore
- OnMoved
- OnMoving

この削除のため、以前これらのカスタム アクションによって呼び出されていた既存の InstallScript イベント ハンドラーは、今回より、InstallScript から直接呼び出されます。これには、ファイル転送の直前で呼び出される InstallScript イベント ハンドラーが含まれます。

- Installing および Uninstalling の機能関数
- OnGeneratedMSIScript
- OnGeneratingMSIScript
- OnInstallFilesActionBefore
- OnMoving

これにはまた、ファイル転送の直後に呼び出される InstallScript イベント ハンドラーも含まれます。

- Installed および Uninstalled の機能関数

- OnMoved
- OnInstallFilesActionAfter

これらの変更は、グローバル変数およびグローバル ポインターが、これらの InstallScript イベント ハンドラー関数で使用できるように加えられました。ただし、イベントは以前、カスタム アクションから呼び出されていたため、これらのイベントのシーケンスは、Windows Installer によって直接呼び出される他のカスタム アクションに関連して変更されています。したがって、場合によって、これらの変更が適切に反映されるように、InstallScript コードを変更する必要があります。

ただし、機能のイベント ハンドラーの場合のみ、スケジュールが InstallShield 11.5 以前で使用されたスケジュールとほぼ同一になるように、これらの InstallScript イベントを呼び出すカスタム アクションを追加することも可能です。ただし、グローバル変数とポインターは、「グローバル変数、グローバル ポインター、および SUPPORTDIR」で議論されているように、個々の InstallScript カスタム アクションの呼び出し間で保持されません。

ファイル転送中にカスタム アクションから OnFeatureInstalling、OnFeatureUninstalling、OnFeatureInstalled、または OnFeatureUninstalled を呼び出しても、インストールへの影響はありません。これは、これらの関数がファイル転送の直前に InstallScript から呼び出されるためで、これらが複数回呼び出されても、影響はありません。

タスク

定義済み InstallScript イベント ハンドラー関数を呼び出す InstallScript カスタム アクションを手動でスケジュールするには、以下の手順に従います:

1. InstallShield 2025 で、アップグレード済みのプロジェクトを開きます。
2. InstallScript ファイルへ適切な変更を行います。
   a. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
   b. スクリプトでイベント ハンドラー関数を検索します。ビューの中央にあるペインで関数名をクリックすると、簡単に関数を検索することができます。
   c. 関数の名前を代替名に変更して、ifx.h に自動的に含まれる既存の関数プロトタイプと競合するのを避けるようにします。例:
      - MyOnGeneratingMSIScript
      - MyOnMoving
      - MyOnMoved
   d. 既存の関数を、単一 HWND パラメーターを取得するようにアップグレードします。例:

      ```
      function MyOnGeneratingMSIScript(hMSI) begin end;
      ```

   e. この新しい関数の適切なプロトタイプを追加します。

      ```
      export prototype MyOnGeneratingMSIScript(HWND);
      ```

3. 名前が変更された InstallScript イベント ハンドラー関数を呼び出す InstallScript カスタム アクションを追加します:
   a. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
   b. 中央のペインで、**[カスタム アクション]** エクスプローラーを右クリックして、**[新しい InstallScript]** をクリックします。InstallScript カスタム アクションが追加されます。
   c. カスタム アクションの名前を入力します。この名前には、InstallScript 関数の名前を変更したときに使用した名前を使用します。例:

- MyOnGeneratingMSIScript
- MyOnMoving
- MyOnMoved

d. 作成した新しい InstallScript カスタム アクションを選択します。

e. **"関数名"** 設定を、InstallScript 関数の名前に設定します。

f. **"スクリプト内実行"** 設定で、下のテーブルで示されている値を指定します。

4. InstallScript カスタム アクションをインストールの適切な箇所にスケジュールします。

a. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。

b. InstallScript カスタム アクションの前にくるアクションまたはダイアログを右クリックして、**[挿入]** をクリックします。**[アクションの挿入] ダイアログ ボックス**が開き、プロジェクトに現在関連付けられているアクションおよびダイアログの一覧が表示されます。

c. 作成した InstallScript カスタム アクションを選択します。必要に応じて、InstallScript イベントの起動についての条件を **[条件]** ボックスに入力します。

d. **[OK]** をクリックします。

以前の機能にできるだけ近づけるため、ステップ 3f のスクリプト内実行の設定と、ステップ 4b の InstallScript カスタム アクションのスケジュールを次のように行います。

テーブル 11 ・ カスタム アクションのスケジュール

カスタム アクション	スクリプト内実行	シーケンス中の場所
OnGeneratingMSIScript	即時 (デフォルト)	[インストール] の [実行] シーケンス、RemoveExistingProducts アクションと InstallInitialize アクションの間
OnMoving	システム コンテキストの遅延	[インストール] の [実行] シーケンス、InstallInitialize アクションと AllocateRegistrySpace アクションの間
OnFeaturesInstalling。定義されたすべての機能の Installing イベントと Uninstalling イベントを呼び出します。メモ: 上記で説明されているように、これによる影響はありません。	システム コンテキストの遅延	[インストール] の [実行] シーケンス、InstallInitialize アクションと AllocateRegistrySpace アクションの間 (OnMoving の後)
OnInstallFilesActionBefore	システム コンテキストの遅延	[インストール] の [実行] シーケンス、MoveFiles アクションと InstallFiles アクションの間

テーブル 11・カスタム アクションのスケジュール (続き.)

カスタム アクション	スクリプト内実行	シーケンス中の場所
OnFeaturesInstalled。定義されたすべての機能の Installed イベントと Uninstalled イベントを呼び出します。メモ: 上記で説明されているように、これによる影響はありません。	システム コンテキストの遅延	[インストール] の [実行] シーケンス、ScheduleReboot アクションと InstallFinalize アクションの間 (OnMoved の前)
OnMoved	システム コンテキストの遅延	[インストール] の [実行] シーケンス、ScheduleReboot アクションと InstallFinalize アクションの間 (OnGeneratedMSIScript の前)
OnInstallFilesActionAfter	システム コンテキストの遅延	[インストール] の [実行] シーケンス、InstallFiles アクションと PatchFiles アクションの間
OnGeneratedMSIScript	即時 (デフォルト)	[インストール] の [実行] シーケンス、ScheduleReboot アクションと InstallFinalize アクションの間 (OnMoved の後)

次の事項に注意してください。

- グローバル変数とポインターは、個々の InstallScript カスタム アクションの呼び出し間で保持されなくなりました。また、個々の InstallScript カスタム アクションでは、それぞれ固有の SUPPORTDIR が初期化され、使用されます。したがって、SUPPORTDIR のファイルでは、個々の呼び出し間で情報を共有することはできません。詳細については、「グローバル変数、グローバル ポインター、および SUPPORTDIR」を参照してください。
- ほとんどの Windows Installer プロパティは、遅延/コミット/ロールバック InstallScript カスタム アクションで使用することはできません。詳細については、「Windows Installer プロパティと遅延/コミット/ロールバック InstallScript カスタム アクション」を参照してください。

OnCheckSilentInstall

InstallShield 11.5 以前では、InstallScript MSI インストールが Setup.exe を使わずにサイレントでインストールされているとき、OnCheckSilentInstall カスタム アクションが自動的に OnMsiSilentInstall InstallScript イベントを呼び出していました (たとえば、次もコマンドラインが使用された場合:Msi.exe /i<Package> /qn)。InstallShield 12 以降では、

このイベントは InstallScript MSI インストールで呼び出されません。ただし、OnMsiSilentInstall イベント ハンドラー関数を呼び出すカスタム アクションを追加することができます。これを行うには、次のステップ 3f と 4b で説明されているスケジュール時における要件に留意して、上述の手順を実行します。

テーブル 12 - OnCheckSilentInstall カスタム アクションのスケジュール

カスタム アクション	スクリプト内実行	シーケンス中の場所
OnCheckSilentInstall	即時 (デフォルト)	[インストール] の [実行] シーケンスの最初のアクションとして

また、InstallScript イベントが予期されているシナリオでのみ実行されるようにするには、次のコードをイベントに追加します。

```
cchValueBuf = MAX_PATH;
// Setup.exe が使用されているか確認し、使用されている場合、戻ります。
MsiGetProperty(nHandle, "ISSETUPDRIVEN", szValueBuf, cchValueBuf);
if(StrLengthChars(szValueBuf)) then
   return ISERR_SUCCESS;
endif;
```

InstallShield 11.5 以前の InstallScript プロジェクトをアップグレードする

InstallShield 11.5 以前で作成された InstallScript プロジェクトを InstallShield 2025 にアップグレードするときの詳しい説明については、以下のセクションを参照してください。

InstallScript オブジェクトを InstallScript プロジェクトに含める

InstallShield 2025 で作成された InstallScript インストールは、以前のバージョンの InstallShield 11.5 で作成されたオブジェクトを使用できません。

InstallScript プロジェクトを InstallShield 2025 へアップグレードすると、すべての InstallShield オブジェクトへの参照は、InstallShield 2025 バージョンのオブジェクトをポイントするように更新されます。新しいバージョンのオブジェクトがシステムに存在しない場合、「オブジェクトが見つかりませんでした」というエラー メッセージがビルド時に表示されます。

InstallShield 2025 オブジェクトを取得する方法については、「InstallShield のアップデートを取得する」を参照してください。

DoInstall 関数への変更

DoInstall 関数は今回より、既に実行されているインストール エンジンを使用する代わりに、子インストールのセットアップ実行可能ファイルを起動して、子インストールを実行します。この変更は自動的に行われますので、既存の InstallScript コードへの変更は必要ありません。インストール実行可能ファイルは初期化が必要なため、子インストールの初期化は、以前のバージョンよりも若干時間がかかる場合があります。詳細については、「DoInstall」を参照してください。

今回より、**DoInstall** の呼び出しは、**LaunchAppAndWait** の呼び出しに似ています。インストールが、CD-ROM または DVD などのリムーバブル メディアから実行されたとき、Disk1 の Setup.exe ファイルは、インストール中、常に使用できる状態にあるとはかぎりません。(実行中に、Setup.exe が使用できなくなった場合、オペレーティング システムでプロンプトが表示され、エンドユーザーに正しいディスクを挿入するように求めてくることがあります。

これにより、インストールが失敗することがあります。)したがって、この問題を回避するために、Setup.exe ファイルが Temp フォルダーにコピーされ、インストールがそこから再起動されます。元の Setup.exe は、ここで終了します。ただし、この処理が発生すると、**DoInstall** (または、呼び出された場合、**LaunchAppAndWait**) は、インストールが完了したと仮定して動作し、待機しません。

この問題のワークアラウンドは、いくつかあります。その 1 つはまず、子インストールを起動するときに、/Clone_wait パラメーターを使用することです。このワークアラウンドの実行すると、起動されたインストールで、起動された元のプロセスが実行中のまま保たれ、親インストールは待機します。ただし、このワークアラウンドでは、Setup.exe を含む元の CD がインストールの途中で使用できない状態になる場合、問題が発生する場合があります。この例として、インストールの途中で一枚目の CD が使用できなくなる複数 CD のインストールがあげられます。

この問題を回避する唯一の方法は、起動されたプロセスの子プロセスの ID を判別するコードを追加して、子プロセスが完了するのを待機することです。

Setup.exe の変更

Setup.exe の /deleter コマンドライン パラメーターが今回より、必要なくなりました。このパラメーターを指定すると、インストールは実行されません。InstallScript インストールは、インストールの起動後すぐにインストールのクローンを作成しなくなったため、/deleter が必要なくなりました (ただし、/runfromtemp パラメーターが指定されたために、インストールが一時フォルダーから実行されている場合などは除きます)。

Setup.exe の /clone_nowait コマンドライン パラメーターが今回より、必要なくなりました。このパラメーターを指定すると、Setup.exe はそれを無視します。InstallScript インストールは今回より、デフォルトでインストールを完了するためにクローンされたインストールを待機しません (/Clone_wait パラメーターが指定された場合を除きます)。/Clone_wait についての詳しい情報は、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

InstallScript MSI と基本の MSI インストール同様、InstallScript の Setup.exe は、有効なリターン コードを返します。したがって、Setup.exe の戻り値を確認するとき、InstallShield 11.5 ではいつも 0 でしたが、InstallShield 12 以降では、Setup.exe が適切な戻り値を返します。返される戻り値についての詳細については、「Setup.exe 戻り値および実行時のエラー (InstallScript プロジェクト)」を参照してください。

Disk1 ファイルへの変更

InstallShield は以前、次のファイルをビルドされたインストールの Disk1 イメージに配置していましたが、この処理は必要なくなったため、今後実行されません。

- Engine32.cab
- Setup.ibt

InstallShield 12 以降で、次のファイルが、ビルドされたインストールの Disk1 イメージに配置されます。

- ISSetup.dll
- _Setup.dll

Disk1 ファイルの変更が原因によるアップグレードの問題は、現在報告されていません。これらの変更は、情報提供のみを目的として報告されています。

InstallScript エンジン ファイルへの変更

InstallShield 12 以降では、InstallScript インストールは、InstallScript エンジン ファイルを次のディレクトリにインストールしません。

<COMMONFILES>¥InstallShield¥Professional¥Runtime¥<MajorVersion>¥<MinorVersion>¥Intel32

この変更が原因によるアップグレードの問題は、現在報告されていません。この変更は、情報提供を目的として報告されています。

InstallScript カスタム アクションを含む InstallShield 11.5 または以前の QuickPatch プロジェクトをアップグレードする

InstallShield 12 以降で InstallScript MSI プロジェクトに作成された QuickPatch プロジェクトは、InstallShield 12 以降で作成されたパッチ メディアのみパッチすることができます。InstallShield 2025 を使って 以前の InstallScript MSI インストールが InstallShield 11.5 以前で作成されたアプリケーションのアップデートを作成する場合、QuickPatch の代わりにマイナーアップグレード パッケージを作成して、[パッチのデザイン] ビューを使って、標準パッチを作成します。このバージョン以後のパッチは、QuickPatch プロジェクトとして作成できます。

(InstallScript カスタム アクションがあるないにかかわらず) 基本の MSI プロジェクトに作成された QuickPatch に関する 問題は、現在報告されていません。

InstallShield 11.5 以前の InstallScript MSI プロジェクトに標準のパッチを作成する

InstallShield 12 以降は、最新および以前のセットアップが InstallShield 11.5 以前で作成された InstallScript MSI パッチの作成 ([パッチのデザイン] ビューを使用して) をサポートしていません。最新のセットアップが InstallShield 12 以降で作成されたパッチ、または以前のセットアップが InstallShield 11.5 以前で作成されたパッチの作成に関する問題は、現在報告されていません。

InstallShield 11.5 以前の InstallScript MSI オブジェクト プロジェクトまたはこの種類のオブジェクトを含むプロジェクトをアップグレードする

InstallShield 11.5 以前には、プロジェクトの InstallScript MSI オブジェクト タイプが含まれました。このプロジェクトの種類は InstallShield では利用できません。

InstallScript MSI オブジェクト プロジェクトをアップグレードする

InstallScript MSI オブジェクト プロジェクトを InstallShield 2025 で開くと、InstallShield はこれをマージ モジュール プロジェクトに変換します。

定義済み InstallScript イベント ハンドラーは現在、InstallScript カスタム アクションを含むマージ モジュール プロジェクトで使用できません。したがって、InstallScript イベント ハンドラー関数を使用する InstallShield 11.5 以前の InstallScript MSI オブジェクト プロジェクトがある場合、InstallShield 2025 でプロジェクトがマージ モジュールの変換されるときに、これらの関数を呼び出すカスタム アクションを手動でスケジュールする必要があります。詳しくは、「基本の MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール」をご覧ください。

InstallShield 2025 は、InstallShield の以前および最新のバージョン同様、InstallScript カスタム アクションがある InstallShield 2025 マージ モジュールを使用することができます。

InstallScript MSI オブジェクトを含むプロジェクトのアップグレード

ビルド済みの InstallScript MSI オブジェクト (.imm フィル) を含むアップグレード プロジェクトは、ビルドをしても失敗します。これは、これらのオブジェクトが InstallShield 2025 では使用できないためです。これを試みると、次のようなビルド エラーが発生します。

ISDEV : エラー -4075: ファイルが見つかりません。機能 'NewFeature1' にモジュール 'InstallShieldMSIObjectName.4F635B62_07BF_4779_B74E_D80C29D508E3:0' をマージ中に、エラーが発生しました。

"InstallShieldMSIObjectName.4F635B62_07BF_4779_B74E_D80C29D508E3:0" は、見つからない InstallScript MSI オブジェクトに固有の情報です。このビルド エラーを解決するには、この InstallScript MSI オブジェクトをプロジェクトから削除します。マージ モジュールは、[再配布可能ファイル] ビューにあるチェック ボックスをクリアすると削除されます。

InstallShield Express で作成したプロジェクトをアップグレードする

InstallShield Express のバージョン 2.1 から 5.x で作成した InstallShield プロジェクトがある場合、そのプロジェクトは InstallShield 2025 で開いたとき自動的に基本の MSI プロジェクトにアップグレードされます。InstallShield 2025 は InstallShield Express プロジェクトを、それと互換性のあるプロジェクトの種類である基本の MSI プロジェクトへ変換します。

アップグレードしたプロジェクトを InstallScript MSI プロジェクトへ変換する方法については、基本の MSI プロジェクトを InstallScript MSI プロジェクトへ変換する を参照してください。

タスク

Express プロジェクトをアップグレードするには、以下の手順の従います。

InstallShield でプロジェクトを開きます。プロジェクトをアップグレードするかどうかたずねるダイアログ ボックスが開きます。

- **[はい]** をクリックしてプロジェクトをアップグレードします。プロジェクトが InstallShield で開きます。
- プロジェクトを開かないでおく場合は **[いいえ]** をクリックします。

元のプロジェクトのバックアップコピーは、このダイアログ ボックスで指定した場所に作成および保管されます。

InstallShield 2025 は、InstallShield 2025 で Express 2.x プロジェクト (.iwz) を基本の MSI プロジェクトに移行したときに、[セットアップのタイプ] ビューのロジックとダイアログを保存します。Express の以前のバージョンは、セットアップのタイプはコンポーネントに基づいていました。Express 2.x プロジェクト (.iwz) では、コンポーネントは機能に置換され、ファイルグループはコンポーネントではなくインストール先フォルダーにマップされています。

InstallShield 2025 で、Express 2.x プロジェクト (.iwz) を 基本の MSI プロジェクトにアップグレードすると、機能の "インストールレベル" プロパティとダイアログ コントロール イベントを変更することによって、セットアップの種類を編集できます。

メモ: *Express プロジェクトの [ビルボード] プロパティは、基本の MSI プロジェクトでサポートされていないため、移行されません。ビルボード プロパティが以前の Express プロジェクトで指定されている場合、警告 701 が出されます。*

現在のバージョンの InstallShield にインストールされていない言語を持つ Express プロジェクトを移行すると、その言語のサポートが失われます。言語サポートは、InstallShield の Premier Edition でのみ使用することができます。

InstallShield Professional で作成したプロジェクトをアップグレードする

InstallShield Professional を使って作成されたプロジェクトをアップグレードすることができます。InstallShield 2025 を使用して InstallShield Professional で作成されたインストール プロジェクトを開くと、プロジェクトは自動的にアップグレードされます。これにより、InstallShield 2025 ですぐ作業を開始することができます。

InstallShield Professional を使用して作成されたプロジェクトは、.ipr ファイルにより参照される、いくつかの .ini ファイルから構成されていました。InstallShield 2025 では、インストール プロジェクトは .ism ファイル 1 つだけです。

最初のステップ: インストール プロジェクトを開く



タスク

InstallShield Professional で作成されたインストール プロジェクトを開くには、以下の手順を実行します。

1. InstallShieldを開きます。
2. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. 開きたい InstallShield Professional プロジェクト ファイル(.ipr) を選択します。
4. **[開く]** をクリックします。

プロジェクトが InstallShield で開き、拡張子が .ipr から .ism に変わります。元の .ipr プロジェクトのバックアップは、同じフォルダーに *ProjectName*.ipr.bak として保存されます。

次のステップ: インストール プロジェクトの変更

インストール プロジェクトをアップグレードした後、 InstallShield ユーザー インターフェイス のビュー、メニュー、ウィザードその他を使用して、プロジェクト設定を変更できます。

アップグレードしたプロジェクトに変更を加えなくてならない場合もあります。詳しくは、「InstallShield Professional 6.x からの移行」および「InstallShield Professional 5.x からの移行」を参照してください。



メモ- *InstallShield 2025 では、サポート ファイルがプロジェクト フォルダーのサブフォルダー—に リンクされています。InstallShield Proefessional の場合、サポート ファイルはプロジェクト フォルダーのサブフォルダー—にコピーされました。*

InstallShield Professional 6.x からの移行

InstallShield Professional 6.x インストールを InstallShield 2025 にアップグレードする際は、以下の事項に注意してください:

- program…endprogram ブロックを使用するスクリプトを、ユーザー インターフェイスおよびファイル転送関数を除くすべての適切なイベント ハンドラー関数を呼び出すイベントベースのスクリプトに簡単に変換することができます。詳細については、「OnShowUI」を参照してください。

- インターネット インストール用の 新しい簡易化モデル は、数多くの Ether オブジェクトのメソッドとそのプロパティとサブオブジェクトをすべて省略します（この時同時に、インターネット固有の必要条件に対応できるように InstallScript の強化点も提供します）。インターネット インストールにこれらのプロパティやメソッドを使用している場合、「古いプロパティとメソッドの置換」をご覧ください。
- version 6.x で作成されたインストールは自動作成された文字エントリを数多く含み、これらの文字列エントリを情報取得のために使用していました。これらの文字列エントリは、同じ情報が [プロジェクトの設定] プロパティ シートにも存在していたため、不要でした。InstallShield 2025 は、[プロジェクト設定] で指定された情報を、実行時にも使用できるようにして、作業を単純化しました。

  この情報を使用することは必須ではありません。文字列エントリは、プロジェクト内の既存のものをそのままに残すことで、これまで同様に使い続けることができます。

  移行したインストールがプロジェクトの [プロジェクトの設定] プロパティシートから情報を自動的に使用できるようにしたい場合、以下の文字列エントリをプロジェクトから削除します。

  COMPANY_NAME

  PRODUCT_NAME

  PRODUCT_KEY

  PRODUCT_VERSION

  TITLE_CAPTIONBAR

  FOLDER_NAME

  TITLE_MAIN

  また、次の表で示されたような、これらの文字列エントリのひとつを使って、新規の対応するシステム変数を代わりに使用している可能性があるスクリプト コードを更新します。

テーブル 13 ・ 文字列エントリと対応する新しいシステム変数

文字列のエントリ	システム変数
COMPANY_NAME	IFX_COMPANY _NAME
PRODUCT_NAME	IFX_PRODUCT_NAME
PRODUCT_KEY	IFX_PRODUCT_KEY
PRODUCT_VERSION	IFX_PRODUCT_VERSION
TITLE_CAPTIONBAR	IFX_SETUP_TITLE
FOLDER_NAME	IFX_PRODUCT_NAME
TITLE_MAIN	IFX_SETUP_TITLE

ある値を文字列テーブルから読み込み、別の値をプロジェクトの設定から読み込む場合は、これらの文字列値のサブセットのみを削除してください。保持する文字列値がある場合、未使用のプロジェクトの設定を変更せずに、文字列テーブルの値を必要に応じて修正するようにしてください。特に、アップデート インストールを作成する場合は、PRODUCT_VERSION 文字列エントリの値が存在する場合、それを必ず更新してください。

- インストールでイベントベースのスクリプトを使用して、デフォルト OnSetUpdateMode イベント ハンドラー コードがオーバーライドされている場合、実行時に「インストールされているアプリケーションのバージョンを判別できません。セットアップを終了します。」といったメッセージが表示される場合があります。このエラーが発生する場合、OnSetUpdateMode イベント中に手動で IFX_INSTALLED_VERSION を設定するコードを追加します。サンプル コードは、新しい InstallShield 2025 InstallScript プロジェクトの OnSetUpdateMode イベントに含まれているデフォルト コードを参照してください。

- インストールがイベントベースのスクリプトを使用し、デフォルトの OnFirstUIBefore イベント ハンドラー コードがオーバーライドされている場合、ファイルを適切なデフォルトの保存場所にインストールするために、(OnFirstUIBefore の 6.x バージョンからの) 次のコードを置換する必要があります。

  ```
  TARGETDIR = PROGRAMFILES ^ @COMPANY_NAME ^ @PRODUCT_NAME;
  ```

  (新規のデフォルトの OnFirstUIBefore コードからの)次のコードを TARGETDIR に設定します。

  ```
  /* 管理者権限があるエンドユーザー、またはそのような権限を持たないパワーユーザーおよび
  エンドユーザーにも対処します。*/
  if ( ALLUSERS ) then
     TARGETDIR = PROGRAMFILES ^ IFX_COMPANY_NAME ^ IFX_PRODUCT_NAME;
  else
     TARGETDIR = FOLDER_APPDATA ^ IFX_COMPANY_NAME ^ IFX_PRODUCT_NAME;
  endif;

  /* 標準と 複数インスタンス インストールの両方を対処します*/
  if( MAINT_OPTION = MAINT_OPTION_MULTI_INSTANCE && MULTI_INSTANCE_COUNT > 0) then
     nLoop = 1;
     svDir = TARGETDIR;
     while(ExistsDir(TARGETDIR) = EXISTS)
        NumToStr(szTargetDirAppendix,nLoop);
        TARGETDIR = svDir + szTargetDirAppendix;
        nLoop = nLoop + 1;
     endwhile;
  endif;
  ```

  手続き型のスクリプト (programÉendprogram ブロックがあるスクリプト) を使用している場合、管理者権限がないユーザー用に、デフォルトのターゲット ディレクトリを適切に設定する TARGETDIR を設定するコードを更新しなければなりません。

- インストールがイベントベースのスクリプトを使用し、デフォルトの OnMaintUIBefore イベント ハンドラー コードがオーバーライドされていて、アンインストール実行時に、メディアにはリストされていなくログファイル(FeatureRemoveAllInMediaAndLog を参照) にのみリストされているものをはじめとするすべての機能を削除する場合、(OnMaintUIBefore の 6.x バージョンからの) 次にコードを置換する必要があります。

  ```
  case REMOVEALL: ComponentRemoveAll();
  ```

  (新規のデフォルトの OnMaintUIBefore コードからの)次のコードを使用します。

  ```
  MediaGetData( MEDIA, MEDIA_FIELD_MEDIA_FLAGS, nMediaFlags, szIgnore );

  case REMOVEALL:
     /* プロパティは 更新を行います。*/
  ```

```
if( nMediaFlags & MEDIA_FLAG_UPDATEMODE_SUPPORTED ) then
    FeatureRemoveAllInMediaAndLog();
else
    FeatureRemoveAllInMedia();
endif;
```

スクリプトベースのアンインストールをサポートする手続き型のスクリプトを使用している場合、ComponentRemoveAll の呼び出しを適切に更新する必要があります。

- スクリプトが RegDBSetItem を呼んで MaintenanceStart (または、DeinstallStart) によって作成されたレジストリ エントリを変更し、RegDBSetItem への呼び出しを OnMoved ハンドラー (例, OnMoving または <機能名>_OnInstalled) の前で呼び出されたイベント ハンドラーで行う場合、RegDBSetItem への呼び出しを移動する必要があります。MaintenanceStart は現在 OnMoveData イベント ハンドラーのデフォルトのコードで呼び出されています。MaintenanceStart は、以前、メンテナンス/アンインストール機能 がインストールされたすぐ後、他の機能がすべてインストールされる前に自動的に呼び出されていました。

- InstallShield Professional 6.x のデフォルトのイベント ハンドラーのコードは、以下のラインを OnFirstUIBefore と OnMaintUIBefore の両方に含んでいました。

  ```
  SetStatusWindow (0, "");
  Enable(STATUSEX);
  StatusUpdate (ON, 100);
  ```

  InstallShield 2025 では、このコードはデフォルトの OnMoveData イベント ハンドラーコードに移動したので、以下のオプションができました。

  - このコードをカスタマイズしていない場合、余分な呼び出しは問題の原因なるので、特に何もする必要はありません。コードの重複を避けたい場合、このコードを OnFirstUIBefore と OnMaintUIBefore から支障なく削除することができます。
  - このコードが既にカスタマイズされていて、このカスタマイズを、呼び出す UI (ユーザー インターフェイス) イベント関わらず適用したい場合、OnFirstUIBefore と OnMaintUIBefore からコードを削除して、OnMoveData イベントをオーバーライドし、デフォルトのコードの位置にカスタマイズしたコードを起きます。
  - コードが既にカスタマイズされていて、呼び出される UI イベントによって特定のコードを指定したい場合、OnMoveData をオーバーライドして、デフォルトのコードをコメントとして除き、既存のコードを継続して使います。このとき、インストールがアップデートをサポートしていない場合、OnUpdateUIBefore のカスタマイズが必要になる場合もあります。

- バージョン 6.x のデフォルトのイベント ハンドラーコードは、OnFirstUIBefore および OnMaintUIBefore イベントに以下のコメントアウトされたコードを含んでいました。

  ```
  //処理内容: 背景、ウィンドウタイトル、およびキャプションバーのタイトルを有効にする場合
  // SetTitle( @TITLE_MAIN, 24, WHITE );
  // SetTitle( @TITLE_CAPTIONBAR, 0, BACKGROUNDCAPTION );
  // Enable( FULLWINDOWMODE );
  // Enable( BACKGROUND );
  // SetColor( BACKGROUND, RGB( 0, 128, 128 ) );
  ```

  このコードをアクティブにして、呼び出された UI イベントに関わらず安定した UI エクスペリエンスを提供したい場合は、OnFirstUIBefore と OnMaintUIBefore からこのコードを削除し、OnShowUI をオーバーライドしてコードを適切にカスタマイズします。

- InstallShield 2025 では、多くの Windows API 関数はインクルードされたヘッダー ファイル (.h ファイル) 内で宣言されます。これらの関数が明示的にスクリプト コードで宣言されている場合、以下を実行します。

- 関数宣言を削除し、InstallShield Professional が提供する宣言を使用します。
- InstallShield 2025 と以前のバージョン両方でコンパイル可能なスクリプトを作成するとき、宣言を以下のようなコードで囲みます。

```
#if _ISCRIPT_VER < 0x700
prototype ...
#endif
```

InstallShield Professional が定義した指定が異なる時、Windows API 関数を呼び出すコードを更新する必要がある場合があります。

- エンドユーザーのインターフェイスがスムーズに行われるようにするため、インストール初期化ダイアログ ボックスは、スクリプトによってダイアログ ボックスが表示されるまでデフォルトで表示されたままになります。インストール初期化ダイアログ ボックスを閉じるには、Disable(DIALOGCACHE) またはダイアログ関数を呼び出します。

InstallShield Professional 5.x からの移行



プロジェクト: *この情報は、InstallScript Object プロジェクトに適用します。*

スクリプトの変更

InstallShield Professional 5.x スクリプトの基本構造は、InstallShield 2025 の InstallScript プロジェクト タイプでサポートされています。InstallShield Professional 5.x インストールを InstallShield でコンパイルする場合、まず次のような変更を行う必要があります。

プリプロセッサ命令

- スクリプトの始めに、次のステートメントを含みます。

```
#include "ifx.h"
```

- SD_SINGLE_DIALOGS の #define ステートメント、およびそれに関連した SD_<ダイアログ名> 定数を、すべて削除します。
- Windows の定数を定義する #define ステートメントは、コンパイラのエラーの原因になっている場合、削除が必要です。多くの Windows 定数は Ifx.h (*InstallShield Program Files フォルダー*¥Script¥Isrt¥Include サブフォルダー内)、またはそこに含まれるファイルで定義されています。

ビルトイン関数

- 次の関数を削除します。これらの関数は、Professional 5.x ではサポートされていましたが、InstallShield 2025 ではサポートされていません。これらの関数のうちいくつかは、コンパイル可能ですが、期待される結果は得られません。可能な場合、サポートされている関数を代わりに使用します。

テーブル 14 ・ 現在不使用の Professional 5.x 関数

Professional 5.x 関数	代替となる関数
AppCommand	なし。(現在はサポートされていないプログラム マネージャー シェル使用のプラットフォームでのみ適用。)

テーブル 14 • 現在不使用の Professional 5.x 関数 (続き.)

Professional 5.x 関数	代替となる関数
AddProgItemEx	AddFolderIcon
CmdGetMsg	Windows API 関数の GetMessage (CmdGetHwndDlg を呼び出してダイアログ ハンドルを呼び出す)
CmdGetParam1	Windows API 関数の GetMessage (CmdGetHwndDlg を呼び出してダイアログ ハンドルを呼び出す)
CmdGetParam2	Windows API 関数の GetMessage (CmdGetHwndDlg を呼び出してダイアログ ハンドルを呼び出す)
CommitSharedFiles	この関数を使用して Professional 5.x で実行したアクションは、InstallShield 2025 では自動的に実行されます。
CreateProgGroupEx	AddFolderIcon
DeleteGroup	DeleteProgramFolder
DeleteProgItem	DeleteFolderIcon
ExitProgMan	なし。(現在はサポートされていないプログラム マネージャー シェル使用のプラットフォームでのみ適用。)
GetGroupNameList	GetFolderNameList
GetItemNameList	なし。(現在はサポートされていないプログラム マネージャー シェル使用のプラットフォームでのみ適用。)
QueryProgGroup	なし。(現在はサポートされていないプログラム マネージャー シェル使用のプラットフォームでのみ適用。)
RegDBCreateKey	RegDBCreateKeyEx
RegDBCreateKeyValue	RegDBCreateKeyValueEx
RegDBGetKeyValue	RegDBGetKeyValueEx
RegDBSetKeyValue	RegDBSetKeyValueEx
ReloadProgGroup	なし。(現在はサポートされていないプログラム マネージャー シェル使用のプラットフォームでのみ適用。)
ReplaceProgItem	なし。(現在はサポートされていないプログラム マネージャー シェル使用のプラットフォームでのみ適用。)
ShowGroup	なし。(現在はサポートされていないプログラム マネージャー シェル使用のプラットフォームでのみ適用。)

- アンインストールのためにログされてほしくないターゲット システムを変更するコードの前に Disable(LOGGING) を呼び出し、そのコードの後で Enable(LOGGING) を呼び出します。これは、スクリプト コードの実行前にアンインストールの変更のログが自動的に有効になるので、必要です。

  これは、DeInstallStart が呼び出されるまでログが開始しなかった、Professional 5.x のインストールとは異なります。DeInstallStart の前に起こったために以前はログされなかったインストールアクションがある場合、これらのインストールアクションの前にさらに Disable(LOGGING) 呼び出しを追加して、ログを手動で無効にする必要があります。

  インストールで背景を表示する場合はEnable(BACKGROUND)を呼び出します。InstallShield 2025 では、背景はデフォルトで無効になっていますが、Professional 5.x ではデフォルトで有効になっています。(イベント ベースのスクリプトでは、背景のカスタマイズおよび表示用コードは、OnFirstUIBefore に対するデフォルト コードの始めのコメント行部分にあります。このコードをアクティブにするには、希望の行の始めのスラッシュ文字を削除します。)

  PlaceBitmap などの一部のユーザー インターフェイス関数は、インストール背景が有効になっていないと正しく動作しません (また失敗を示す負の値を返すこともありません)。

- アンインストールキーの [DisplayName] の値を設定するために、引数である REGDB_UNINSTALL_NAME を使用して RegDBSetItem を呼び出す必要はありません。このバージョンでは、MaintenanceStart または DeinstallStart を呼び出した際に行います。

- 引数の STATUSBAR を使用して SetColor を呼び出しても、InstallShield 2025 では何も起こりません。ステータス バーは、システム カラーを使用して作成されます。

- StrGetTokens を呼び出す場合、szString の最初 (または最後) の文字が szDelimiterSet の文字と一致すると、ヌル文字列 ("") が最初 (または最後) の要素としてリストに挿入されません。代わりに、最初と 2 番目(または最後と最後から2番目) の区切り文字の間の文字が最初 (または最後) の要素としてリストに挿入されます。

- StrGetTokens を呼び出すと、szSrcFile で指定されたファイル名にワイルドカード文字が含まれる場合、szTargetFile のファイル名部分は以前のバージョンの InstallShield Professional のように無視されません。代わりに szTargetFile はすべて各ソースファイルを既存の名前を使ってコピーするターゲットパスとして処理されます。たとえば、CopyFile("C:¥¥*.*","D:¥¥File.txt") はファイルを D ドライブの File.txt フォルダーにコピーします。

- WaitOnDialog では、エンドユーザーがシステムメニューの [閉じる] コマンドを選択するか、タイトルバーの [閉じる] ボタンをカスタム ダイアログからクリックすると、IDCANCEL を返します。以前のバージョンでは、このような場合に DLG_CLOSE が返されていました。

- GetSystemInfo を DRIVE の最初の引数と UNC パスの 3 番目の引数に渡すと、関数は正しく負の値を返します。これまでは成功を示すゼロが間違って返されていました。オペレーティング システム制限のため、UNC パスはこの場合サポートされていませんでした。

- 3 番目の引数を PATH に設定し、2 番目の引数を行末の円記号(例 "¥¥¥TheServer") がないサーバー名だけを含む UNC パスを設定して、ParsePath を呼び出すと、最初の引数が二重の円記号 ("¥¥¥") に正しく設定されます。Professional 5.x ではこの関数は、最初の引数を単一の円記号("¥¥") にしていました。

- Professional 5.x ヘルプライブラリには、成功を表す 0 (ゼロ)以外の特定の数値の戻り値がリストされています。これらの数値の中には、InstallShield 2025 で変更されたものもあります。InstallShield 2025 で使用されているスクリプトは、その関数のヘルプに現在記載されている定数とだけその戻り値を比較します。

- エンドユーザーのインターフェイスがスムーズに行われるようにするため、インストール初期化ダイアログ ボックスは、スクリプトによってダイアログ ボックスが表示されるまでデフォルトで表示されたままになり

ます。インストール初期化ダイアログを閉じるには、Disable(DIALOGCACHE) またはダイアログ関数を呼び出します。

- ComponentMoveData の 2 番目の引数は有用なデータを返すことはなくなりました。
- SendMessageTimeout の4番目の引数としてポインターを文字列に渡しても、期待される結果は今後得られません。文字列データを SendMessageTimeout を渡すには、SendMessageTimeout トピックの追加情報セクションを参照してください。

定義済み定数

- 次の定数を削除します。これらの関数は、Professional 5.x ではサポートされていましたが、InstallShield 2025 ではサポートされていません。

  COMPONENT_VALUE_ALWAYSOVERWRITE

  COMPONENT_VALUE_NEVEROVERWRITE

  COMPONENT_VALUE_NEWERDATE

  COMPONENT_VALUE_NEWERVERSION

  COMPONENT_VALUE_OLDERDATE

  COMPONENT_VALUE_OLDERVERSION

  COMPONENT_VALUE_SAMEORNEWDATE

  COMPONENT_VALUE_SAMEORNEWERVERSION

  COMPONENT_FIELD_DESTINATION

  COMPONENT_FIELD_OVERWRITE

- あらかじめ定義されている以下の変数 (TARGETDISK=″C:¥¥″; など) に値を割り当てるようなステートメントは削除します。このようなステートメントは、InstallShield 2025 ではコンパイルされません。これらの定数は読み取り専用です。

  CMDLINE

  COMMONFILES

  ERRORFILENAME

  FOLDER_DESKTOP

  FOLDER_PROGRAMS

  FOLDER_STARTMENU

  FOLDER_STARTUP

  ISRES

  ISUSER

  ISVERSION

  MODE

  PROGRAMFILES

SUPPORTDIR

TARGETDISK (TARGETDIR が変更されると、この変数の値も自動的に更新されます。)

UNINST

WINDIR

WINDISK

WINSYSDIR

WINSYSDISK

DLL

- Professional 5.x で DLL 関数を呼び出すと、すべての文字列引数は参照として渡されます。InstallShield 2025 では文字列引数を値で渡すことができます。引数を渡すメソッドを指定するには、関数プロトタイプで、MyDLL.MyFunc(BYREF NUMBER, BYVAL STRING) のように、引数のデータ型の前に BYREF または BYVAL キーワードを付けます。InstallShield 2025 コンパイラには適切なキーワードの使用を推奨するための次のようなコンパイラ警告が含まれています。
  - BYREF も BAVAL も指定されていない場所でスクリプトが DLL 関数呼び出しを含む場合、コンパイラ警告 W7 507 が発生します。この警告を削除するには、関数プロトタイプに適切なキーワードを追加します。
  - 参照として渡された文字列引数のリテラル文字列を指定する場合、コンパイラ警告 W7 511 が発生します。この警告を削除するには、関数プロトタイプに BYVAL キーワードを追加します。

  外部 DLL で関数を呼び出す場合、スクリプトと DLL で同じ呼び出し規則が使われていることを確認してください。Professional 5.x ではインストールエンジンで常に stdcall 規則を使用していましたが、一貫性のない DLL 規則を無視することがありました。ただし InstallShield 2025 エンジンでは、間違った呼び出し規則が使用されるとエラー (例外 0x80040704) が発生します。

  DLL 関数を宣言するとき、スクリプトで呼び出し規則の cdecl または stdcall を宣言できます。例:

  prototype cdecl MyDLL.MyFunction (INT, INT);

  呼び出し規則を明示的に宣言しなかった場合、InstallShield は stdcall を使用します。
- InstallShield 2025 では、InstallScript 配列は、内部的には C または C++ 配列ではなく OLE オートメーション セーフ配列としてフォーマットされます。InstallScript 配列を C/C++ 配列が必要な DLL 関数に渡すためには、GetCArrayFromISArray を呼び出すことによって配列データを要求フォーマットに配置しなければなりません。
- InstallShield 2025 では、アドレスがポインター変数に保存されている文字列変数に関して、ポインターを DLL 関数に渡して変更することはできません。DLL 関数で文字列変数値を変更できるようにするには、BYREF 演算子を指定して引数のデータタイプを宣言してから、変数そのものを引数として関数に渡します。
- Windows の DLL 検索パスにある DLL を呼び出す前に、UseDLL 関数を呼び出す必要はありません。DLL 検索パスは、次のとおりです。
  1. ロードされたアプリケーションからのフォルダー
  2. 現在のフォルダー
  3. 32 ビットの Windows システム フォルダー
  4. 16 ビットの Windows システム フォルダー
  5. Windows フォルダー

6. PATH 環境変数に一覧表示されているフォルダー

現在のフォルダーとエンド ユーザーのマシンにある PATH 環境変数の値は予期できないため、この機能の使用には注意してください。

その他

- 明記されていない配列構文 ArrayName[nArrayIndex] はサポートされていません。InstallShield 2025 では、すべての配列の構文は ArrayName(nArrayIndex) で表されます。
- スクリプトで定義された関数の戻り値の種類を明白に指定する場合、関数定義でキーワードの "function" の後に同じ戻り値の種類を指定する必要があります。これを行わないと、スクリプトを完了できません。(戻り値のタイプを明白に指定しない場合は、戻り値タイプは NUMBER であると想定されます。)以前のバージョンの InstallShield Professional では、戻り値の型は実際に指定された値に関係なく NUMBER と想定されたため、これは問題になりませんでした。
- HKEY_CURRENT_USER の下にアンインストール情報を配置するには、ALLUSERS システム変数を FALSE に設定します。
- リテラル文字を引数としてビルトイン関数またはユーザー定義関数に渡す場合、ASCII 数値を渡す必要があります。
- システム変数である CMDLINE を処理する際、コマンドラインスイッチ (-SMS、-z、-c、-e、-q、-t、および -x) が Setup.exe では現在使用されていないため、CMDLINE に含まれていることに注意してください。-SWS スイッチは、インストールが完了するまで Setup.exe がメモリに残るようになったため、使用されていません。また、256 メガバイト以上のメモリを搭載したシステムでも Setup.exe は正しく初期化できるため、-z スイッチは、現在は使用されていません。
- InstallShield 2025 ではコンパイル不可能な Professional 5.x の ODBC テンプレート コードを削除します。これは、次のような関数呼び出しがコード内でネストされるからです。

```
ComponentGetData ( szMedia, szCompName,
   COMPONENT_FIELD_DESTINATION, nvData, svDest );
```

InstallShield 2025 では、Destination プロパティがコンポーネント プロパティではなく、ファイル グループ プロパティであるため、この関数の呼び出しはコンパイルされません。

他の変更

- インストール終了に F3 を使うことはできません。
- Setup.exe で -f スイッチを使用して名前が変わった Setup.inx を使うことはできません。
- InstallShield 2025 インストールは低解像度システムの特別ビルボードなどをサポートしていません。
- ソース ファイルも既存のターゲット ファイルにもバージョン番号がなく、ソース ファイルのファイル グループの "上書き" プロパティが [新しいバージョン]、[同じか新しいバージョン]、[古いバージョン] のいずれかだった場合、ターゲット システムのファイルは上書きされません。Professional 5.x インストールではターゲットファイルが上書きされます。
- InstallShield 2025 は、Windows 95 形式のダイアログの表示はサポートしていません。
- サイレントインストールの応答ファイル (.iss) の [Application] セクションにある Lang キーは何も効果を持たなくなりました。
- In Professional 5.x インストールでは、インストーラーによってログされたレジストリキーは、アンインストール中、すべて無条件で削除されます。InstallShield 2025 では、RegDBCreateKeyEx で作成されたキーおよ

びレジストリセットの中の非共有キーは、インストーラーがそのキーを作成した場合のみ、つまり、インストールが実行されたときキーが存在しなかったときのみ削除されます。

スクリプトの変更: 語彙変換

用語の変更

InstallShield Professional では、エンド ユーザーの視点から見たインストールの機能的な構成要素をコンポーネントと呼んでいました。InstallShield 2025 では、これらのビルドブロックを機能と呼びます。

このように用語が変更されたため、多くの InstallScript 関数名も変更されました。InstallShield Professional を使用して作成したプロジェクトをアップグレードすると、スクリプトの項目の一部は用語変更され、一部はエイリアスが作成されます。

- 関数名
- ComponentFileInfo (FeatureFileInfo) フラグ
- ComponentSetData (FeatureSetData) および ComponentGetData (FeatureGetData) フラグ

関数名

コンポーネントを参照していたすべての InstallScript 関数名は、機能を参照するようになります。たとえば、ComponentDialog は FeatureDialog になります。これらの関数のパラメーターには変更はありません。機能関数をクリックすると、対応するヘルプトピックを表示できます。

テーブル 15 ・ 関数名の変更

コンポーネント関数	機能関数
ComponentAddItem	FeatureAddItem
ComponentCompareSizeRequired	FeatureCompareSizeRequired
ComponentDialog	FeatureDialog
ComponentError	FeatureError
ComponentErrorInfo	FeatureErrorInfo
ComponentFileEnum	FeatureFileEnum
ComponentFileInfo	FeatureFileInfo
ComponentFilterLanguage	FeatureFilterLanguage
ComponentFilterOS	FeatureFilterOS
ComponentGetData	FeatureGetData
ComponentGetItemSize	FeatureGetItemSize

テーブル 15 • 関数名の変更 (続き.)

コンポーネント関数	機能関数
ComponentGetTotalCost	FeatureGetTotalCost
ComponentInitialize	FeatureInitialize
ComponentIsItemSelected	FeatureIsItemSelected
ComponentListItems	FeatureListItems
ComponentLoadTarget	FeatureLoadTarget
ComponentMoveData	FeatureMoveData
ComponentPatch	FeaturePatch
ComponentReinstall	FeatureReinstall
ComponentRemoveAll	FeatureRemoveAll
ComponentRemoveAllInLogOnly	FeatureRemoveAllInLogOnly
ComponentRemoveAllInMedia	FeatureRemoveAllInMedia
ComponentRemoveAllInMediaAndLog	FeatureRemoveAllInMediaAndLog
ComponentSaveTarget	FeatureSaveTarget
ComponentSelectItem	FeatureSelectItem
ComponentSelectNew	FeatureSelectNew
ComponentSetData	FeatureSetData
ComponentSetTarget	FeatureSetTarget
ComponentSetupTypeEnum	FeatureSetupTypeEnum
ComponentSetupTypeGetData	FeatureSetupTypeGetData
ComponentSetupTypeSet	FeatureSetupTypeSet
ComponentTotalSize	FeatureTotalSize
ComponentTransferData	FeatureTransferData
ComponentUpdate	FeatureUpdate
ComponentValidate	FeatureValidate

テーブル 15・関数名の変更 (続き.)

コンポーネント関数	機能関数
SdComponentDialog	SdFeatureDialog
SdComponentDialogAdv	SdFeatureDialogAdv
SdComponentMult	SdFeatureMult
SdComponentTree	SdFeatureTree

ComponentFileInfo (FeatureFileInfo) フラグ

InstallShield Professional の ComponentFileInfo (現在は FeatureFileInfo) 関数で使用していたフラグは、InstallShield 2025 の機能で使用できるように変換されます。

テーブル 16・ComponentFileInfo フラグ

使用不可のフラグ	新しいフラグ
COMPONENT_INFO_ATTRIBUTE	FEATURE_INFO_ATTRIBUTE
COMPONENT_INFO_LANGUAGE	FEATURE_INFO_LANGUAGE
COMPONENT_INFO_OS	FEATURE_INFO_OS
COMPONENT_INFO_ORIGSIZE	FEATURE_INFO_ORIGSIZE
COMPONENT_INFO_COMPSIZE **COMPONENT_INFO_DATE** **COMPONENT_INFO_DATE_EX** **COMPONENT_INFO_TIME**	 ***メモ・*** *利用不可能なフラグは、オプションが機能していないことを示す -137 を戻します。FeatureError 関数を利用して、戻り値についての詳しい情報を提供することができます。*
COMPONENT_INFO_VERSIONLS	FEATURE_INFO_VERSIONLS
COMPONENT_INFO_VERSIONMS	FEATURE_INFO_VERSIONMS
COMPONENT_INFO_VERSIONSTR	FEATURE_INFO_VERSIONSTR

ComponentSetData (FeatureSetData) および ComponentGetData (FeatureGetData) フラグ

InstallShield Professional の ComponentSetData および ComponentGetData（現在 FeatureSetData および FeatureGetData）関数で使用していたフラグは、InstallShield 2025 の機能で使用できるように変換されます。

テーブル 17 ・ 使用できない ComponentSetData と ComponentGetData フラグ

使用不可のフラグ	新しいフラグ
COMPONENT_FIELD_CDROM_FOLDER	FEATURE_FIELD_CDROM_FOLDER
COMPONENT_FIELD_DESCRIPTION	FEATURE_FIELD_DESCRIPTION
COMPONENT_FIELD_DISPLAYNAME	FEATURE_FIELD_DISPLAYNAME
COMPONENT_FIELD_FILENEED	FEATURE_FIELD_FILENEED
COMPONENT_FIELD_FTPLOCATION	FEATURE_FIELD_FTPLOCATION
COMPONENT_FIELD_HTTPLOCATION	FEATURE_FIELD_HTTPLOCATION
COMPONENT_FIELD_IMAGE	FEATURE_FIELD_IMAGE
COMPONENT_FIELD_MISC	FEATURE_FIELD_MISC
COMPONENT_FIELD_PASSWORD	FEATURE_FIELD_PASSWORD
COMPONENT_FIELD_SELECTED	FEATURE_FIELD_SELECTED
COMPONENT_FIELD_SIZE	FEATURE_FIELD_SIZE
COMPONENT_FIELD_STATUS	FEATURE_FIELD_STATUS
COMPONENT_FIELD_VISIBLE	FEATURE_FIELD_VISIBLE
COMPONENT_VALUE_CRITICAL	FEATURE_VALUE_CRITICAL
COMPONENT_VALUE_HIGHLYRECOMMENDED	FEATURE_VALUE_HIGHLYRECOMMENDED
COMPONENT_VALUE_STANDARD	FEATURE_VALUE_STANDARD

プロジェクト アシスタント ダイアログ サポートを InstallShield Professional プロジェクトからアップグレードされたプロジェクトに追加する

インストール プロジェクトを InstallShield Professional から InstallShield にアップグレードした場合で、そのプロジェクトに InstallShield Professional で作成した Setup.rul ファイルが含まれるとき、プロジェクト アシスタントの [インストール インタビュー] ページの質問で利用できないものもあります。これは、Setup.rul ファイルから InstallShield がスクリプトタグでキーとして利用する OnFirstUIBefore イベントコードの一部が欠落しているためです。

新規 InstallScript プロジェクトを作成して OnFirstUIBefore イベントを確認すると、ダイアログ関数をサポートするために [プロジェクト アシスタント] が利用する次のコードが見つかります。

```
//{{IS_SCRIPT_TAG(FunctionName)
   InstallScript コード...
//}}IS_SCRIPT_TAG(FunctionName)
```

タスク ***[プロジェクト アシスタント] ダイアログ サポートをアップグレードしたプロジェクトへ追加するには、以下の手順を実行します。***

1. **[InstallScript]** ビューで、Setup.rul ファイルを開きます。
2. OnFirstUIBefore イベントコードをコメントアウトします。
3. InstallScript ペインの上部にあるイベント カテゴリ リストで OnFirstUIBefore イベントを選択します。InstallShield がスクリプトへこのイベントを再び追加します。
4. 適切なダイアログコードを OnFirstUIBefore イベントのコメントアウトコードにコピーします。
5. ステップ 3 で追加した OnFirstUIBefore イベントコードを削除します。
6. 残りの OnFirstUIBefore イベントコードからコメントを削除します。

InstallShield—Windows Installer Edition で作成したプロジェクトをアップグレードする

InstallShield–Windows Installer Edition で作成した InstallShield プロジェクト (.ism ファイル) がある場合、そのプロジェクトは InstallShield で開くと自動的に基本の MSI プロジェクトにアップグレードされます。アップグレードしたプロジェクトを InstallScript MSI プロジェクトに変換する方法については、「基本の MSI プロジェクトを InstallScript MSI プロジェクトへ変換する」を参照してください。

タスク ***プロジェクトを開くと、プロジェクトをアップグレードするかどうかたずねるダイアログ ボックスが表示されます。***

- ***[はい]*** *をクリックしてプロジェクトをアップグレードします。プロジェクトが IDE で開きます。*
- *プロジェクトを開かないでおく場合は* ***[いいえ]*** *をクリックします。*

元のプロジェクトのバックアップ コピーは、このダイアログで指定した場所に作成および保管されます。

コマンドラインからのセットアップのアップグレード

セットアップ プロジェクトをコマンドラインからビルドしたときもプロジェクトをアップグレードできます。セットアップ プロジェクトをビルドせずにコマンドラインからアップグレードする場合は、-u パラメーターを使用します。

アップグレードエラーのトラブルシューティング

アップグレードが原因で発生したエラーがあれば、IDE 下部の出力パネルに表示されます。これらのエラーについての詳細は、「アップグレード エラーと警告 (InstallShield—Windows Installer Edition からのアップグレード)」を参照してください。

プロパティ マネージャーでのアップグレード後の変更

プロパティマネージャーで空の値 (値なし) のあるプロパティは MSI には無効です。そのため、それらのプロパティの空の値に代わって ***DO_NOT_BUILD*** が表示されます。このようにすることにより、アップグレードの後、空の値のあるプロパティはプレースホルダーとして保存されますが、msi パッケージには組み込まれません。

InstallShield の他のエディションからアップグレードする

InstallShield には、Premier、Professional、および Express の 3 つのエディションが用意されています。

Premier Edition のみで提供されている機能

次の一覧は、Professional Edition または Epress Edition ではなく、Premier Edition のみで使用可能な機能です:

- **スイート/アドバンスト UI インストールの作成およびビルド機能**—複数の .msi パッケージ、.msp パッケージ、InstallScript パッケージ、.exe パッケージ、サイドローディング Windows アプリ パッケージ (.appx) および Windows Installer トランザクションだけでなく、複数の InstallShield 前提条件にも対して、最新のカスタマイズ可能なユーザー インターフェイスを持つブートスラップ アプリケーションを作成することができます。スイート/アドバンスト UI インストールでは、複数の個別のインストールを、統合されたユーザー インターフェイスを使用する単一のインストールにパッケージ化し、条件付きで、必要に応じて、ターゲット システムでパッケージを起動するセットアップ ランチャー (Setup.exe) を使用します。
- **スイート/アドバンスト UI インストールで InstallScript アクション、およびその他の種類のアクションをサポート** — スイート/アドバンスト UI インストールでは、今回より、インストールに含めるパッケージの範囲を超えた様々な実行時タスク行うための InstallScript アクションを起動できるビルトイン サポートが提供されています。アクションを使って、実行可能ファイルを実行、DLL 関数を呼び出し、PowerShell スクリプトを実行、スイート/アドバンスト UI プロパティを設定、InstallScript コードを実行、またはマネージ アセンブリでパブリック メソッドを呼び出すことができます。
- **仮想化サポート** — Microsoft App-V アシスタントは、InstallShield Premier Edition に含まれています。このアシスタントを使って、Microsoft App-V 形式で、カスタム仮想アプリケーションを作成することができます。仮想化技術を使って、アプリケーションを独自の環境に隔離することで、既存アプリケーションとの競合を回避したり、基盤となるオペレーティング システムの変更を避けたりすることが可能です。
- **アプリケーション仮想化適合性スイート**—InstallShield では、製品を仮想化するための準備が整っているかどうかを判断するのに役立つ、新しい検証スイートが提供されています。これらのスイートに含まれている InstallShield 仮想化内部整合性検証ツール (ISVICE) を使って、Microsoft App-V 4.x、Microsoft App-V 5、

Microsoft Server App-V、VMware ThinApp、および Citrix XenApp との適合性をチェックできます。顧客に仮想バージョンの提供を考慮する場合に、検証スイートを使用して、製品をどのようにビルドするか豊富な情報に基づいた意思決定を行うことができます。

- **DIM ファイルのサポート** – DIM プロジェクトの作成機能はInstallShield Premier Edition で使用できます。このサポートは、コラボレーション アドオン、InstallShield Developer Installation Manifest Editor でも提供されています。DIM ファイルを基本の MSI プロジェクトに追加する機能は、InstallShield Premier Edition で提供されています。

  DIM プロジェクトは、機能サイズのプロジェクトで、インストール パッケージの別個に分かれている部分を構成する製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、および要素など、関連するアイテムを集めたものです。DIM を利用することにより、複数のチーム メンバーが、インストールの開発を同時に携わることができます。各ソフトウェア開発者またはチームメンバーは、異なる DIM について個別で作業することができ、リリース エンジニアは、1 つまたは複数の基本の MSI プロジェクトでそれらを参照することができます。

- **複数言語インストール** – 複数の言語でエンドユーザー テキストを表示する単一のインストール プロジェクトを作成し、言語固有のファイルの条件付きインストールを処理できます。翻訳済みの文字列を利用して、ダイアログおよびメッセージを 34 の追加言語の 1 つに変更することができます。

  プロジェクト・*アラビア語 (サウジアラビア) とヘブライ語の 2 つの言語サポートは、基本の MSI とマージ モジュール プロジェクトでのみ利用できます。*

- **アラビア語 (サウジアラビア) とヘブライ語サポート** – InstallShield Premier Edition は、右から左方向へ読み書きするアラビア語 (サウジアラビア) とヘブライ語のサポートを含みます。デフォルトのエンド ユーザー ダイアログ文字列のすべてが、これらの言語で利用できます。

  これらの言語は右から左方向に読まれるため、Premier Edition では、アラビア語とヘブライ語ダイアログのミラーリングが サポートされています。このサポートにより、アラブ語とヘブライ語のダイアログでは、右から左方向へのレイアウトが使用されます。たとえば、英語やその他の左から右に読まれる言語のダイアログで右側にあるボタンは、右から左に読まれる言語のダイアログでは左側に移動されます。

- **ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する機能**– InstallShield Premier Edition には、ビルド プロセスの様々な段階で実行するコマンドを指定できるリリース設定が含まれています。次のビルド イベントで実行するコマンドをスケジュールすることができます: (a) InstallShield がリリースのビルドを開始する前、(b) InstallShield が .msi パッケージおよび (製品のデータ ファイルが .cab ファイルに格納される場合) .cab ファイルをビルドした後、.msi パッケージに署名が行われて、Setup.exe ファイルにストリームされる前、(c) InstallShield がリリースをビルドならびに署名を行った後。

- **ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにインストールを配布できる機能** – インストールのビルドが成功するたびに InstallShield が仮想マシン (VM) を指定のスナップショットに戻し、VM の電源をオンにしてインストールを VM にコピーし、テストが可能な状態になるようプロジェクトを構成できます。また、これらのテスト準備を必要なときにオンデマンドで行うこともできます。このテスト準備機能により、手作業で行う処理を減らし、テストにかかる時間を短縮できます。VM は Microsoft Hyper-V Server、VMware ESX または ESXi Server、または VMware Workstation を指定できます。

- **InstallShield MSI ツール用の追加のライセンス** – InstallShield には、InstallShield MSI Diff、InstallShield MSI Query、InstallShield MSI Sleuth、および InstallShield MSI Grep 用のライセンスが追加で含まれています。これらのツールを利用して、Windows Installer パッケージに関連する問題を解決することができます。InstallShield Premier Edition には、別のマシンに InstallShield 以外のツールのみをインストールできる、個別のインストールおよび追加ライセンスが含まれています。詳しい使用条件については、InstallShield MSI ツールの使用許諾契約書を参照してください。

- **既存の IIS Web サイトからプロジェクトへの IIS データのインポート機能** – InstallShield では、IIS スキャナー (IIScan.exe) が提供されています。このコマンドライン ツールを使って、既存の IIS Web サイトをスキャンして、Web サイトに関する IIS データを記録できます。IIS スキャナーは、Web サイト、その仮想ディレクトリ、アプリケーション、およびアプリケーション プールすべての設定を含む XML ファイルを作成します。この XML ファイルを使って、InstallShield Premier Edition の [IIS 構成] ビューに IIS データをインポートできます。IIS データをプロジェクトにインポートしてから、必要に応じて、[IIS 構成] ビューを使って IIS の設定を変更できます。
- **Web 配置パッケージの配置をサポート** – InstallShield Premier Edition で作成可能なスイート/アドバンスト UI インストールでは、Web 配置パッケージを IIS Web サーバーまたはクラウドに配置できるビルトイン サポートを利用できます。
- **InstallShield Collaboration** – InstallShield Premier Edition には、InstallShield Collaboration for Visual Studio のライセンスが含まれています。
- **ネットワーク リポジトリ** – ネットワーク リポジトリは、複数のインストール作成者が必要に応じアクセスでき、プロジェクトで再利用することができるインストール要素のコレクションです。ネットワーク リポジトリは、インストール作成者同士のコラボレーション作業を促進し、ネットワークに格納されます。
- **InstallShield ベスト プラクティス スイート**– InstallShield には、InstallShield ベスト プラクティス スイートという名前の 1 セットの検証ツールが含まれています。インストールがベスト プラクティス ガイドラインに違反している場合、このスイートの InstallShield ベスト プラクティス (ISBP) 検証ツールによって警告されます。
- **追加のダイアログ テーマ** – 一部のダイアログ テーマは InstallShield Premier Edition のみで提供されています。

Premier Edition および Professional Edition のみで使用できる機能

以下は、Premier Edition および Professional Edition で提供されていて、Express Edition では提供されていない機能の一部です:

- **アドバンスト UI インストールの作成およびビルド機能**—単一の .msi パッケージ、.msp パッケージ、および InstallScript パッケージだけでなく InstallShield 前提条件にも対して、最新のカスタマイズ可能なユーザー インターフェイスを持つブートストラップ アプリケーションを作成することができます。アドバンスト UI では、条件付きで、必要に応じて、ターゲット システムでパッケージを起動するセットアップ ランチャー (Setup.exe) を使用します。

  (InstallShield Premier Edition の スイート/アドバンスト UI 機能には、この機能に対して、さらに拡張サポートが追加されています。スイート/アドバンスト UI 機能を利用すると、複数の .msi パッケージ、.msp パッケージ、InstallScript パッケージ、.exe パッケージ、サイドローディング Windows アプリ パッケージ (.appx)、および Windows Installer トランザクションだけでなく、複数の InstallShield 前提条件も、モダンでカスタマイズ可能なユーザー インターフェイスを持つ単一のアプリケーションにパッケージできます。)
- **Standalone Build**– InstallShield Premier Edition および Professional Edition で使用可能なこのツールは、InstallShield のインストールをビルドする機能のみ、および選択した任意の再配布可能ファイルをビルドマシン上にインストールすることができます。Standalone Build 用の追加ライセンスを購入することもできます。
- **ダイアログ エディター** – ダイアログ エディターを利用して、既存のエンド ユーザー ダイアログのレイアウトを変更したり、新規カスタム ダイアログを作成することができます。プロジェクト全体でダイアログをインポートおよびエクスポートして、共有することができます。プロジェクトでサポートしている各言語用に、別のダイアログを作成します。
- **InstallShield MSI ツール**– InstallShield Premier Edition および Professional Edition には、InstallShield MSI Diff、InstallShield MSI Query、InstallShield MSI Sleuth、および InstallShield MSI Grep が含まれています。これらのツールを利用して、Windows Installer パッケージに関連する問題を解決することができます。

- **オートメーション インターフェイス** – スクリプトを指定して、新しいファイルの追加、機能の追加または削除、製品名とアップグレード コードの変更、リリースの設定の変更、概要情報ストリーム項目の変更、リリース フラグの変更、プロパティの変更、ビルド処理の開始などを行うことができます。
- **リリースのカスタマイズ** – プロジェクトのどの部分を圧縮し、どの機能をどのディスクに配置して、どの言語を含めるか定義します。ローカライズ作業をサポートするために、言語に基づいてアプリケーションデータをフィルタリングしてください。
- **ソース コード管理の統合** – プロジェクトをソース コード管理システムにチェックインおよびチェックアウトするプロセスを単純化し、プロジェクトの差分を検出するときに容量を節約します。InstallShield Premier Edition および Professional Edition で SCC の統合機能は、様々なソース コード コントロール システムとの統合をサポートします。
- **柔軟なローカリゼーション サポート**– InstallShield Premier Edition および Professional Edition には、[文字列エディター] ビューが搭載されていて、インストール中、実行時に表示される翻訳可能なテキスト文字列すべてを一箇所にまとめて管理できます。このビューを使って、ボタン テキストから機能の説明まで全ての文字列を編集できます。このビューを使って、文字列エントリを翻訳可能な形式にエクスポートしてから、翻訳済み文字列をプロジェクトにインポートすることもできます。
- **プロジェクト検証** – 標準 .cub ファイルを使って、インストールとマージ モジュールを検証します。アップグレードとパッチの検証を利用して、潜在的に存在するアップグレードに関する問題を検出し、リリースされる前にそれらを解決します。
- **パッチの作成** – QuickPatch プロジェクトの作成のほか、Premier Edition と Professional Edition では、製品の以前のバージョンに対するアップデートを含む標準パッチを作成することができます。
- **複数製品バージョンの管理** – 1 つのプロジェクトから、評価版、デバッグ版、標準版、拡張版などのバージョンをビルドします。ユーザー定義フラグを使って、リリースに特定の機能、InstallShield 前提条件、およびその他の要素を選択 (または除外) することができます。
- **InstallScript のサポート**– InstallShield Premier Edition と Professional Edition では、シンプルかつパワフルなプログラム言語 InstallScript がサポートされています。InstallScript カスタム アクションを Windows Installer ベースのインストールに追加したり、Windows Installer エンジンの代わりに InstallScript エンジンを使用してインストール全体を制御する InstallScript プロジェクトを作成したりできます。
- **柔軟なカスタム アクション サポート**– InstallShield Premier Edition および Professional Edition は、Express Edition では提供されていない種類のいくつかのカスタム アクションをサポートします。これらの追加カスタム アクションを利用して、プロパティの設定、ディレクトリの設定、マネージ アセンブリ内でのパブリック メソッドの呼び出し、または特定の条件下でエラーメッセージを表示してインストールを中止することが可能です。
- **柔軟なショートカット サポート**– InstallShield Premier Edition および Professional Edition は、Express Edition では構成することができない、ショートカットの様々な詳細設定を構成するためのサポートが含まれています。たとえば、Premier Edition と Professional Edition では、プロジェクト内のショートカットをタスクバーまたは [スタート] メニューにピン留めしないように防ぐことができます。また、エンド ユーザーが製品をインストールした後に、[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐことができます。これらのショートカット オプションは、しばしばインストールの一部となるツールまたは付属的な製品に使用されます。
- **マージ モジュールの作成と編集** – 他のアプリケーションのインストール パッケージで再利用できるように、プロジェクトの小さい単位をパッケージすることができます。独自に作成したもの、または、製品に含まれるすべてのモジュールを再利用できます。さらに詳細なカスタマイズを行うために、モジュールを開いて編集することができます。

- **プロジェクトのテンプレート** – インストール プロジェクトやマージ モジュールプロジェクトの作成するとき、その出発点として利用できるすべてのデフォルト設定とデザイン要素を含むプロジェクト テンプレートを作成できます。
- **複数の IIS Web サイト** – InstallShield Express および Limited Edition では、1 つのインストールによって 1 つの Web サイトのみインストールできます。Premier Edition および Professional Edition では、1 つのインストールによって複数の Web サイトをインストールできます。
- **IIS アプリケーション プールと Web サービス拡張のサポート** – IIS アプリケーション プールと Web サービス拡張を管理します。
- **Windows サービスの詳細サポート**– InstallShield Express Edition では、サービスに関するいくつかの作業がサポートされていますが、 Premier Edition および Professional Edition では、サービスに関する追加の柔軟性が提供されています。たとえば、Premier Edition および Professional Edition ではインストールおよびアンインストール中にサービスを開始、停止、または削除することができます。サービスはインストールの一部である場合と、ターゲット システムに既存する場合があります。これらのエディションではまた、Windows Installer 5 で利用可能な拡張サービス カスタマイズ オプションを構成することもできます。
- **SQL サポート** – SQL サーバーへの接続、データベース スキーマおよびデータのインポート、SQL スクリプトと機能の関連付けなど多数サポートされています。
- **テキスト ファイルまたは XML ファイルを変更できる機能** – [テキスト ファイルの変更] ビューまたは [XML ファイルの変更] ビューを使って、実行時にターゲット システム上で変更するファイルを構成できます。
- **64 ビット専用サポート** – Windows Server Core は、WOW64 (32-bit Windows-on-Windows) サポートの無効化をサポートします。この構成がより一般化するのに伴い、64 ビットのアプリケーションが 32 ビットの機能に全く依存しないでインストールできるようにすることが必要となります。これを実現するため、InstallShield Premier Edition および InstallShield Professional Edition では 64 ビット専用 .msi パッケージのビルドが可能です。これらは、WOW 64 機能を搭載していない 64 ビット版の Windows ベースのシステム上で実行します。
- **InstallShield 前提条件エディター** – このツールを使用して、新しい InstallShield 前提条件を作成したり、既存の前提条件を変更したりできます。
- **Setup.exe および Update.exe のカスタム アイコン–ビルド時に作成する** – Setup.exe および Update.exe ファイルに使用するカスタム アイコン (.exe、.dll、または .ico ファイル) を指定します。アイコンは Setup.exe の [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Setup.exe ファイルを右クリックして、[プロパティ] をクリックしたときに表示されます。エンド ユーザーが Windows Explorer で Setup.exe ファイルを参照したときにも、このアイコンが表示されます。
- **Setup.exe の有効期限日** – Setup.exe の有効期限日と有効期限メッセージを設定します。エンド ユーザーが Setup.exe をプロジェクトで指定された日付以降に実行すると、有効期限切れメッセージが表示されて、インストールが終了します。
- **トランザクション処理を使った複数パッケージのインストールをサポート** – Windows Installer 4.5 以降では、トランザクション処理を使った複数パッケージのインストールをサポートします。InstallShield Premier Edition および Professional Edition では、インストール プロジェクトに連鎖 .msi パッケージを追加できます。パッケージ並びに追加 .msi パッケージは連鎖されて、単一トランザクションとして処理されます。トランザクションに含まれる 1 つまたは複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は全てのパッケージについてロールバックを開始して、システムを以前の状態に復元します。
- **様々なプロジェクト要素をエクスポートおよび再利用できる機能** – 既存のプロジェクトの一部 (ダイアログ、カスタム アクション、または機能) をマージ モジュールまたは別のインストール プロジェクトに移動して、効率を高めます。

- **複数インスタンスのサポート** – エンド ユーザーが製品の複数のインスタンスを同じマシン上に同じユーザー コンテキストでインストールできるインストールを作成します。
- **デバイス ドライバー サポート** – Premier Edition および Professional Edition に搭載されているデバイス ドライバー サポートは、Microsoft の DIFxApp (Driver Installation Frameworks for Applications) を使って、インストールからデバイス ドライバーをインストールする処理を簡素化します。
- **追加のダイアログ テーマ**– いくつかのダイアログ テーマは InstallShield Premier Edition および Professional Edition のみで提供されています。Limited および Express Edition で提供されてるテーマは 2 種類だけです。
- **Visual Studio マージ モジュール プロジェクトの変換**– InstallShield Premier および Professional Edition では、Visual Studio マージ モジュール プロジェクトを InstallShield マージ モジュール プロジェクトに変換することができます。この処理は、他のプロジェクトで消費するマージ モジュールをビルドするのに必要です。
- **COM+ アプリケーション プロキシ サポート** – インストール中に COM+ アプリケーション プロキシを管理します。COM+ アプリケーション プロキシはサーバー アプリケーション属性のサブセットで構成され、これはクライアン ト コンピューターからアプリケーションが存在するマシンへのリモート アクセスを可能にします。

各エディションで提供されている機能についての詳細情報は、InstallShield セールスに問い合わせるか、http://www.installshield.com にアクセスしてください。

エディションからエディションへのアップグレードについて詳しく知りたいときは、該当するトピックを参照してください。

- Express Edition から Professional Edition または Premier Edition にアップグレードする
- Professional Edition から Premier Edition にアップグレードする

Express Edition から Professional Edition または Premier Edition にアップグレードする

InstallShield Express Edition を使って Express プロジェクトが作成された場合、同じプロジェクトを InstallShield Premier Edition または Professional Edition で開くと、基本の MSI プロジェクトにアップグレードされます。このアップグレードは、基本の MSI が Express プロジェクトの種類に対して同様のプロジェクトの種類なため、実行されます。Express Edition で作成された QuickPatch プロジェクトは、InstallShield Premier Edition まてゃ Professional Edition で QuickPatch エディションのままです。

InstallShield Premier または InstallShield の Express Edition で作成されたプロジェクトを開くと、そのファイルは .ise ファイルから .ism ファイルへ変換されます。.ise ファイルのバックアップは、変換が行われる前に保存されます。

Express Edition プロジェクトからのビルボードのプロパティは、基本の MSI プロジェクトでサポートされていないので、移行されません。ビルボード プロパティは Express Edition プロジェクトで指定されている場合、警告 701が出されます。



タスク ***Express Edition プロジェクトをアップグレードするには、以下の手順の従います。***

InstallShield Premier Edition または Professional Edition でプロジェクトを開いて下さい。プロジェクトをアップグレードするかどうかたずねるダイアログ ボックスが開きます。

- **[はい]** をクリックしてプロジェクトをアップグレードします。プロジェクトが InstallShield で開きます。

- プロジェクトを開かないでおく場合は **[いいえ]** をクリックします。

元のプロジェクトのバックアップコピーは、このダイアログ ボックスで指定した場所に作成および保管されます。

Professional Edition から Premier Edition にアップグレードする

Premier Edition で Professional Edition プロジェクトを開くとき、プロジェクトの変換は必要ありません。プロジェクトは、Premier Edition でシームレスに開き、Premier Edition 限定の機能を利用することができます。特定の機能に関する詳しい情報は、「InstallShield の他のエディションからアップグレードする」を参照してください。

各エディションで提供されている機能についての詳細情報は、http://www.installshield.com にアクセスしてください。

Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートする

Visual Studio は、制限付きでセットアップおよびマージ モジュール プロジェクトの作成をサポートします。InstallShield では、これらの種類の Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートすることで、InstallShield に搭載されている拡張機能と使ってインストールとマージ モジュールを作成することができます。

InstallShield では、次を行うことができます:

- Visual Studio セットアップ プロジェクト (.vdproj) を InstallShield 基本の MSI またはマージ モジュール プロジェクト (.ism) にインポートするか、Visual Studio マージ モジュール プロジェクト (.vdproj) を InstallShield 基本の MSI またはマージ モジュール プロジェクト (.ism) にインポートする。これらのタスクを行うと、Visual Studio プロジェクトに含まれている同じデータと設定を含む InstallShield インストール プロジェクトおよびマージ モジュール プロジェクトを作成できます。インポート処理中に、Visual Studio プロジェクト内の特定の設定について、それらをインポートするか無視するかを選択できます。
- Visual Studio セットアップ プロジェクトを InstallShield 基本の MSI プロジェクトに変換するか、Visual Studio マージ モジュール プロジェクトを InstallShield マージ モジュール プロジェクトに変換する。その他のプロジェクトで使用するマージ モジュールをビルドするためには、Visual Studio マージ モジュール プロジェクトを InstallShield マージ モジュール プロジェクトに変換する必要があります。



メモ: *Visual Studio プロジェクトに 1 つ以上のプロジェクト出力が含まれている場合、InstallShield を使ってその Visual Studio プロジェクトを InstallShield プロジェクトにインポートできますが、InstallShield がその Visual Studio プロジェクトを InstallShield プロジェクトに変換することはできません。*

サポート対象の Visual Studio バージョン:

- Visual Studio 2010
- Visual Studio 2008
- Visual Studio 2005
- Visual Studio .NET 2003

- Visual Studio .NET

Visual Studio プロジェクトの代わりに InstallShield プロジェクトを使用することの利点

Visual Studio セットアップ プロジェクトを InstallShield プロジェクトに変換またはインポートすると、InstallShield の機能を使ってプロジェクトをカスタマイズすることができます。

以下は、InstallShield プロジェクトで実行可能で Visual Studio プロジェクトでは実行不可能ないくつかのタスクの一覧です:

- 視覚的なダイアログ エディターを使用して、ダイアログのレイアウトを修正する。
- コンポーネントと機能を管理する。
- カスタム アクションを .msi または .msm データベースの Binary テーブルに格納する。(Visual Studio では、すべてのカスタム アクションは製品と共にインストールされる必要があります。)
- SQL サーバーの接続や SQL スクリプトの実行など、SQL 関係のタスクを管理する。
- IIS Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張を管理する。
- ファイル転送中に表示するビルボードの追加。
- 実行時にターゲット システムで XML ファイルまたはその他のテキスト ファイルを修正する。
- ターゲット システムに既に存在するファイルのショートカットを作成する。
- COM+ アプリケーションを管理する。

インポート処理

InstallShield を使って、Visual Studio セットアップまたはマージ モジュール プロジェクトを InstallShield 基本の MSI またはマージ モジュール プロジェクトにインポートできます。

メモ *InstallShield プロジェクトにインポートする Visual Studio セットアップ またはマージ モジュール プロジェクトに 1 つ以上のプロジェクト出力が含まれている場合、その InstallShield は、Visual Studio セットアップまたはマージ モジュール プロジェクトおよびそのプロジェクトのすべての依存関係を含む、同じ Visual Studio ソリューションに含まれていなくてはなりません。*

プロジェクト出力を含む Visual Studio プロジェクトをインポートするには、Visual Studio 内部から InstallShield を使用しなくてはなりません。InstallShield プロジェクトが InstallShield で開かれていて、Visual Studio 内部から開かれていない場合に、プロジェクト出力を含む Visual Studio プロジェクトを InstallShield プロジェクトにインポートしようとすると、エラーが発生します。

タスク ***Visual Studio プロジェクト (.vdproj) を InstallShield プロジェクト (.ism) にインポートするには、以下の手順に従います:***

1. InstallShield で、基本の MSI またはマージ モジュール プロジェクト開く、または作成します。
2. **[プロジェクト]** メニューで、**[Visual Studio デプロイメント プロジェクト ウィザード]** ボタンをクリックします。

3. Visual Studio デプロイメント プロジェクト ウィザードのパネルを完成させる。

InstallShield は、ウィザードで構成された設定に基づいて Visual Studio プロジェクトを現在開かれている InstallShieldプロジェクトにインポートします。InstallShield がプロジェクトをインポートするとき、[出力] ウィンドウにプロジェクトのインポート ステータスが表示されます。出力ウィンドウでは、変換プロセスの各ステップが表示され、すべての変換エラーと警告が一覧で表示されます。

変換プロセス

InstallShield を使って、Visual Studio セットアップ プロジェクトを変換すると、InstallShield では InstallShield 基本の MSI プロジェクト (.ism) が作成されます。

InstallShield を使って、Visual Studio マージ モジュール プロジェクトを変換すると、InstallShield では InstallShield マージ モジュール プロジェクト (.ism) を作成します。

タスク ***Visual Studio プロジェクト (.vdproj) を InstallShield プロジェクト (.ism) に変換するには、以下の手順に従います:***

1. InstallShieldを開きます。
2. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. **[ファイルの種類]** ボックスで、**Visual Basic セットアップ プロジェクト(*.vdproj)** を選択します。
4. 開く Visual Studio プロジェクトの場所を参照し、プロジェクト ファイルを選択します。
5. **[開く]** ボタンをクリックします。

InstallShield は、Visual Studio プロジェクトの設定に基づいて InstallShield プロジェクトを作成します。InstallShield は、.ism ファイルを .vdproj ファイルと同じフォルダーに格納します。.ism ファイルが作成されると、出力ウィンドウにプロジェクト変換のステータスが表示されます。出力ウィンドウでは、変換プロセスの各ステップが表示され、すべての変換エラーと警告が一覧で表示されます。

変換処理が成功すると、その新しい InstallShield プロジェクトが InstallShield で表示されます。

インポート後および変換後のタスク

前提条件タスク

Visual Studio では、1 つ以上の定義済み前提条件を Visual Studio セットアップ プロジェクト内の 1 つ以上の構成に追加することができます。InstallShield のインポートおよび変換プロセスは、すべての構成に含まれるすべての前提条件を、対応する InstallShield 前提条件に変換しようとします。InstallShield に対応する InstallShield 前提条件が含まれていない場合、警告 -9071 が発生して、その前提条件が変換されなかったことを通知します。この警告を解決するためには、必要な再配布可能ファイルをインストールする InstallShield 前提条件を作成して、それをプロジェクトに追加する方法を考慮してください。詳細については、「InstallShield 前提条件を定義する」および「インストール プロジェクトに含まれている InstallShield 前提条件を利用する」を参照してください。

ユーザー インターフェイス タスク

インポートおよび変換処理では、Visual Studio プロジェクトからのダイアログを InstallShield プロジェクトに組み込みません。プロジェクトをインポートまたは変換した後、InstallShield の [ダイアログ] ビューでプロジェクトに含まれるダイアログの設定を構成することができます。

言語のタスク

Visual Studio プロジェクトを InstallShield プロジェクトにインポートしたときに、以下の条件がある場合、InstallShield はプロジェクトに含まれる既存の文字列エントリ値を Visual Studio プロジェクトの言語のデフォルト文字列エントリ値で置換します:

- Visual Studio デプロイメント プロジェクト ウィザードで、Visual Studio プロジェクトの言語をインポートすることを指定した。
- Visual Studio プロジェクトの言語が InstallShield プロジェクトのデフォルト言語と一致しない。(Visual Studio で、Localization プロパティがプロジェクトの言語を示します。InstallShield では、[文字列エディター] ビューの "デフォルト言語" 設定が、プロジェクトのデフォルト言語を示します。)
- InstallShield は、Visual Studio プロジェクトで使用されている言語をサポートします。(InstallShield Profesioanl Edition を使用している場合、Visual Studio プロジェクトで使用した言語がサポートされていない場合があります。)

たとえば、Visual Studio デプロイメント プロジェクト ウィザードで Visual Studio プロジェクトの言語をインポートする場合に、InstallShield プロジェクトの言語がスペイン語で、Visual Studio プロジェクトの言語がドイツ語のとき、InstallShield Premier Edition はスペイン語のランタイム文字列をデフォルトのドイツ語翻訳と置換します。このため、[一般情報] ビュー内の "発行者" 設定を更新している場合など、文字列エントリの値を編集した後にウィザードで Visual Studio プロジェクトの言語をインポートすると、InstallShield は "発行者" 設定の値およびその他の文字列の値を、デフォルトのドイツ語の文字列エントリの値に置換します。

したがって、Visual Studio プロジェクトのインポート中にプロジェクト言語を変更する場合は、[一般情報] ビューと [文字列エディター] ビューで設定を確認の上、適切な場合は文字列エントリを変更してください。

.NET インストーラー クラスのタスク

Visual Studio プロジェクトに .NET インストーラー クラス カスタム アクションが含まれる場合、InstallShield が変換処理中に、.NET インストーラー クラス カスタム アクションを作成する代わりに、.NET アセンブリのコンポーネントの .NET インストーラー クラス情報を構成します。InstallShield は、Visutal Studio における .NET インストーラー クラス カスタム アクションの Condition プロパティをサポートしません。このため、Visual Studio プロジェクトに条件がある .NET インストーラー クラス カスタム アクションが含まれている場合、プロジェクトを変換したあと、InstallShield の [コンポーネント] ビューを使って、.NET アセンブリを含むコンポーネントの条件を作成したほうが良い場合があります。

追加のタスク

InstallShield の他のビューを利用すると、プロジェクトに追加の変更も行うことができます。



メモ: *Visual Studio では、アプリケーション フォルダーに複数のフォーマットされたプロパティが含まれたディレクトリ パス (例、[ProgramFilesFolder][Manufacturer]¥[ProductName]) を指定できます。Visual Studio プロジェクトは、実行時にディレクトリ カスタム アクションを使用して、パスを解決します。ただし、InstallShield では、この種類のディレクトリ パスはサポートされていません。したがって、InstallShield はパスを変換処理中に解決し、パスの* INSTALLDIR *プロパティを使用します。*

InstallShield のアップデートを取得する

インターネットに接続されている場合、ご使用中の InstallShield バージョンで利用可能な最新の InstallShield 前提条件、マージ モジュール、オブジェクト、およびサービス パック、パッチ、またはその他のアップデートを確認および取得することができます。



タスク ***アップデートを確認するには、以下の手順に従います:***

アップデートが使用可能な時に、InstallShield でアップデート ダイアログが表示されます。アップデートを取得するには、次のステップを実行します:

- システムに対するアップデートを確認する。
- アップデートに関する説明およびリリース ノートを確認する。
- アップデートをダウンロードおよびインストールします。

InstallShield 実行時の言語サポート



重要: *言語サポートは、ご使用中の InstallShield のエディション、プロジェクト タイプ、および InstallShield のインターフェイス言語 (英語、日本語) によって異なります。*

アラビア語 (サウジアラビア) とヘブライ語は、Premier Edition の基本の MSI、マージ モジュール、およびスイート/アドバンスト プロジェクトでのみ使用できます。

InstallShield Premier Edition には、35 ヶ国語のデフォルト ランタイム文字列がサポートされています。サポート対象の言語をプロジェクトに追加すると、その言語は InstallShield 内の様々な言語関連設定で使用可能となります。さらに、InstallShield はプロジェクトにその言語に翻訳された文字列エントリを追加します。それらは、デフォルト ダイアログ、メッセージ、およびその他のエンド ユーザー インターフェイス要素の文字列エントリです。サポート対象言語の一覧は、「言語識別子」を参照してください。

InstallShield Premier Edition ではまた、基本の MSI、InstallScript、InstallScript MSI、および スイート/アドバンスト UI プロジェクトで 新しい言語ウィザード を使って、サポート対象の 35 ヶ国語以外の言語を追加することができます。サポートされていない言語とは、デフォルトのランタイム文字列が全く翻訳されていない言語です。サポートされていない言語をプロジェクトに追加すると、その言語は InstallShield 内の様々な言語関連設定で使用可能となります。さらに、InstallShield は新しく追加されたサポートされていない言語の文字列用のプレースホルダーとして、プロジェクトのデフォルト言語の文字列を使用します。サポートされていない言語に翻訳済みの文字列を提供するには、[文字列エディター] ビューを使用します。

InstallShield Professional Edition の英語版では、33 ヶ国語 (アラビア語とヘブライ語を除くすべてのサポート対象言語) のうち 1 つの言語を、作成するすべてのプロジェクトのランタイム言語として使用できます。InstallShield Professional Edition の英語は、選択した 1 つの言語用のデフォルトのラインタイム文字列を含みます。

InstallShield Professional Edition の日本語版では、英語に加えてもう 1 つ追加サポート対象言語を選択できます。つまり、InstallShield Professional Edition の日本語版を使用している場合、英語のランタイム文字列を使ったプロジェクト、およびもう 1 つのサポート対象言語のランタイム文字列を使ったプロジェクトを作成できます。
InstallShield Professional Edition の日本語版には、英語のデフォルト ランタイム 文字列と、選択可能な追加のサポート対象言語が 1 つ含まれています。

デフォルトのランタイム文字列を変更したい場合、またはカスタム ダイアログ、メッセージ、および他のエンド ユーザー インターフェイス要素をプロジェクトに追加する場合、カスタム ランタイム文字列を InstallShield プロジェクトに直接入力することができます。別の手段として、言語の文字列エントリをファイルにエクスポートして、そのファイルの文字列値を翻訳してから、プロジェクトにそれをインポートすることもできます。この機能は、すべての InstallShield エディションでサポートされています。

InstallShield の言語サポートについての詳細は、ドキュメントの次のセクションを参照してください:

- エンドユーザー インターフェイスのローカライズ
- 複数言語インストールの作成
- 言語 サポートのコード ページ要件
- 言語の設定

言語 サポートのコード ページ要件

2 バイト文字を含む言語 (日本語、ギリシャ語、韓国語など) を使ったランタイム文字列を含むリリースの作成を予定している場合は、ビルド マシンに特定のコード ページをインストールする必要があるかどうかを判断してください。

InstallShield ではランタイム文字列の表示に Unicode エンコーディングが使用されます。Unicode エンコーディングを使うと、プロジェクトのターゲット言語がインストールされていないシステム上で InstallShield を使用している場合でも、InstallShield 内では 2 バイト文字が正しく表示されます。さらに、リリースのターゲット言語がインストールされていないシステム上でリリースをビルドすることもできます。

ただし、ANSI エンコードをランタイム文字列に使用する場合 (たとえば、プロジェクトから文字列エントリを ANSI テキスト ファイルとしてエクスポートしてから、翻訳済みの ANSI テキスト ファイルをプロジェクトにインポートする場合) は、ビルド マシンにこれらの言語用のコード ページをインストールする必要があります。コードページによって、システムがそれらの言語の文字を正確に表すことができるようになります。ビルド マシンにコード ページがインストールされていない場合、リリースをビルドすると InstallShield がビルド エラーを報告し、インストールは実行時に 2 バイト文字を正しく表示しません。

ヒント: *翻訳のために文字列エントリをエクスポートするとき、ANSI エンコードの代わりに Unicode エンコードを使用することが推奨されます。*

ビルド マシン上に補足言語サポートをインストールする

以下は、Windows XP が搭載されたビルド マシン上にコード ページをインストールする方法を説明します。これは、中国語、日本語、および韓国語を含むリリースをビルドする場合に必要になります。新しいオペレーティング システムでは、一般的にコード ページがデフォルトでインストールされます。

タスク ***Windows XP が搭載されたビルド マシン上にコード ページをインストールするには、以下の手順に従います:***

1. **[コントロール パネル]** で **[地域と言語のオプション]** を選択します。
2. **[言語]** タブをクリックします。
3. **[詳細]** ボタンをクリックします。

4. **[追加]** ボタンをクリックします。
5. コードページを追加する言語を選択します。
6. **[OK]** をクリックします。

次の手順は、アラビア語やヘブライ語など、右から左に読み書きする言語を Windows XP が搭載されたビルド マシン上にインストールする方法を説明します。新しいオペレーティング システムでは、一般的にこれらのファイルがデフォルトでインストールされます。これらのファイルがインストールされていない場合、[文字列エディター] ビューで文字列が右から左ではなく、左から右に表示されます。



タスク ***右から左に読み書きする言語のサポートを Windows XP が搭載されたビルド マシンにインストールするには、以下の手順に従います:***

1. **[コントロール パネル]** で **[地域と言語のオプション]** を選選択します。
2. **[言語]** タブをクリックします。
3. **[補足言語サポート]** 領域で **[複雑なスクリプトおよび右から左に読み書きする言語用のファイルをインストールする]** チェック ボックスを選択します。
4. **[追加]** ボタンをクリックします。
5. コードページを追加する言語を選択します。
6. **[OK]** をクリックします。
7.

サポートされているアプリケーション プログラミング言語

InstallShield は、Java, Pascal, C++, Visual Basic, Delphi, C# .NET, Visual Basic .NET, ASP .NET, および Cobol など、*任意*のプログラム言語でアプリケーションのインストールを作成できます。

上記の言語以外でアプリケーションを作成した場合でも、InstallShield 使ってそのアプリケーションのためのインストールを作成することができます。また、アプリケーションを配布はしないけれども、ファイルやデータベースをパッケージしたい場合、InstallShield を作業のために使用できます。

3

チュートリアル

このセクションでは、InstallShield を使ったインストールの作成手順をステップ バイ ステップで説明します。

テーブル 1・チュートリアルの種類

チュートリアル	説明
InstallScript プロジェクト チュートリアル	InstallScript インストール プロジェクトの作成手順を、ステップ バイ ステップでご案内します。また、このチュートリアルでは、インストール プロジェクトに関するさまざまなタスクの情報と、これらのタスクを行うために使用する InstallShield インターフェイス内の各ビューについて紹介します。
基本の MSI プロジェクト チュートリアル	基本の MSI インストール プロジェクトについて説明します。また、このチュートリアルでは、インストール プロジェクトに関するさまざまなタスクの情報と、これらのタスクを行うために使用する InstallShield インターフェイス内の各ビューについて紹介します。
グローバリゼーション チュートリアル	言語および言語固有のコンポーネントをプロジェクトに追加する方法を説明します。さらに、ターゲット システムの言語に基づいてコンポーネントを条件付きでインストールする方法も紹介します。 ***メモ:*** *このチュートリアルを完了するには、開発システム上で InstallShield Premier Edition のインストールが必要です。*

InstallScript プロジェクト チュートリアル

このチュートリアルでは、InstallShield を使用した InstallScript インストール プログラムの作成、ビルド、実行、およびそ強化方法について順を追って説明します。

チュートリアルの構成は複数のステップに分かれています。最初のステップ (ステップ 1: プロジェクトの作成、ビルドおよびテスト) を除き、残りのステップはすべて独立した内容になっているので、これらは特に順番にこだわることなく現在の作業に必要な情報のみを調べることができます。

本チュートリアルでは、インストールが取り扱うタスクの多くの処理方法について解説します。

- ファイルのインストール
- ショートカットおよびレジストリ データの設定
- データの条件付きインストール
- インストールのユーザー インターフェイスの変更
- リリース イメージのビルド
- インストールのテスト

チュートリアルを通して、ヘルプライブラリの該当箇所へのリンクが設けてあります。

ステップ 1: プロジェクトの作成、ビルドおよびテスト

このステップでは、インストール プロジェクトの作成、リリース イメージのビルド、およびインストールのテストについて説明します。本ステップを一通り完了すると、以下の項目がマスターできます。

- プロジェクト アシスタントを使った新規プロジェクトの作成法。
- インストール プロジェクトのグローバル プロパティの指定方法。
- セットアップタイプ、機能、コンポーネント、およびファイルリンクの指定法。
- 複製および配布用のリリース イメージのビルド方法。
- InstallShield ユーザー インターフェイスからのインストールの実行方法。

インストールは 3 つのレベルで構成されます。

テーブル 1・インストール内のレベル

レベル	説明
コンポーネント	コンポーネントとは、開発者から見て、個別にインストール可能な最小の製品構成単位のことです。コンポーネントは、ファイル、ショートカット、レジストリデータなど、ターゲット システムにインストールされるデータを指定します。ユーザーがコンポーネントを直接処理することは決してありません。 1 つのコンポーネントを複数の機能に配置することもでき、ユーザーがこのコンポーネントと関連付けられた機能の 1 つを選ぶと、このコンポーネントのデータがインストールされます。

テーブル 1・インストール内のレベル (続き.)

レベル	説明
機能	機能とは、ユーザーから見て、個別にインストール可能な最小の製品構成単位のことです。ユーザーが [カスタム] セットアップ タイプを選択すると、どの機能をインストールするかをユーザーが選択することができるダイアログが表示されます。 各機能はコンポーネントを含みます。
セットアップの種類	セットアップの種類とは、事前に定義された機能のコレクションです。デフォルトで、インストールは [完全] および [カスタム] のセットアップの種類を提示し、ユーザーはどのセットアップの種類でインストールするかを SetupType ダイアログで選択します。

本チュートリアルで作成するインストールは、*Tutorial App* という名前のアプリケーションについて、そのインストールと設定を行うものです。Tutorial App のソース ファイルは InstallShield Program Files フォルダーの Samples サブフォルダーにあります。デフォルト インストール先は次の場所です:

C:¥Program Files¥InstallShield¥2020¥Samples¥WindowsInstaller¥Tutorial Project

プロジェクト アシスタントを使ったプロジェクトの作成

InstallShield を起動した後、次の手順に従って新規 InstallScript プロジェクトを作成します。

1. 次のひとつを行って [**新規プロジェクト**] ダイアログ ボックスを開きます:
   - **[スタートページ] の [プロジェクト タスク]** セクション (ページの左側) にある **[新規プロジェクトの作成]** をクリックします。
   - **[ファイル]** メニューで、**[新規]** をクリックします。
   - ツールバーの **[新規プロジェクト]** ボタンをクリックします。
2. **[新規プロジェクト]** ダイアログ ボックスで、**InstallScript** タブをクリックします。
3. **InstallScript タブ**のリスト ビューで、**[InstallScript プロジェクト]** アイコンを選択します。
4. ここでは [**プロジェクト名**] 編集ボックスに、プロジェクト名として *Tutorial* と入力します。
5. **[OK]** をクリックします。

プロジェクトの作成については InstallShield Professional で作成したプロジェクトを更新するなど、他にも多くの方法があります。詳細については、「新しいプロジェクトの作成」を参照してください。

InstallShield は、*ProjectName*.ism という名前のプロジェクト ファイルを作成します。この場合、Tutorial.ism となります。InstallShield ユーザー インターフェイスで行ったすべての設定は、プロジェクトファイルに保管されます。プロジェクトを他のマシンに移動する場合は、移動先のシステムに .ism ファイル (およびインストールソースファイル) をコピーします。



ヒント: *新規プロジェクトファイルが作成されるディレクトリを変更するには、[ツール] メニューから [オプション] を選択し、[ファイルの場所] タブをクリックして表示される "プロジェクトの場所" フィールドに、新しいディレクトリを指定します。*

新規プロジェクトは、その [プロジェクト アシスタント] タブに開きます。プロジェクト アシスタントを利用するには、右下角にある [次へ] ボタンをクリックします。



ヒント: *プロジェクト アシスタントのステップの順序は問いません。また（適切なタブをクリックすることで）プロジェクト アシスタントとインストール デザイナーとを自由に切り替えることができ、インストール プロジェクトへさらに複雑でパワフルな要素を追加することが可能です。*

アプリケーション情報を入力する

[アプリケーション情報] ページでは、インストールするアプリケーションについての一般情報を設定することができます。



タスク ***このチュートリアルでは次を行います:***

1. **[会社名を入力してください]** 編集ボックスに名前「*Tutorial Co*」を入力します。
2. [**アプリケーション名を入力してください**] 編集ボックスに名前 *Tutorial App* を入力します。
3. その他のフィールドはそのままにします。

"アプリケーション名" フィールドへの入力値は、エンドユーザーに対して表示されるダイアログで、コントロール パネルの [プログラムの追加と削除] でのアプリケーションの表示名として表示されます。入力するアプリケーション名と会社名は Windows スタートメニュー上のアプリケーションのショートカットのデフォルトの場所、並びにコンポーネントファイルのデフォルトのインストール先を指定する *TARGETDIR* システム変数のデフォルト値を決定します。

インストール アーキテクチャのカスタマイズ

[インストール アーキテクチャ] ページでは、インストールに表示させる機能を指定することができます。機能とは、ユーザーから見て個別にインストール可能な最小の製品構成単位のことです。カスタム セットアップの種類を選択すると、個々の機能がエンドユーザーに対して表示されます。



メモ: *機能には、サブ機能やサブサブ機能などを含めることも可能で、インストールの要求に応じて必要なレベルの多層構造化が可能です。*



タスク ***このチュートリアルでは次を行います:***

1. **[インストールアーキテクチャをカスタマイズしますか？]** という質問には **[はい]** を選択します。
2. 既存 **DefaultFeature** 機能を選択して、その名前を **ProgramFiles** に変更します。
3. **HelpFiles** という新しい機能を作成します。そのためには **[インストール アーキテクチャ]** をクリックしてから **[新規]** ボタンをクリックします。

プロジェクトへファイルを追加する

[アプリケーション ファイル] ページでは、各機能にリンクさせるファイルを指定することができます。

まず最初に、機能の一覧の中から、ファイルを挿入する機能を選択します。ファイルリンクを追加するには、[ファイルの追加] ボタンをクリックし、選択されている機能に含めるファイルを選びます (複数指定可能)。



タスク ***このチュートリアルでは、次の手順に従って、ProgramFiles 機能に Tutorial.exe というファイルを追加します:***

1. 機能の一覧表で、**ProgramFiles** をクリックします。
2. ツリー コントロール (トップノードは[インストール先コンピューター]) 内にある **[アプリケーション ターゲット フォルダー]** を選択します。
3. **[ファイルの追加]** をクリックします。
4. ソース ディレクトリに置かれた Tutorial.exe を参照します。
5. 「追加したファイル...は依存関係を持つ可能性があります」メッセージが表示されたとき、**[いいえ]** をクリックします。Tutorial.exe には依存関係は含まれません。

ショートカットの作成

[アプリケーション ショートカット] ページでは、ターゲット システムのデスクトップ又は [スタート] メニュー上にあるアプリケーションファイルのショートカットを指定することができます。デフォルトで、このページはインストール プロジェクトに含んだ各実行可能ファイルのショートカットを表示します。これらを削除して、インストール プロジェクトに含めた別のファイルへのショートカットを追加することができます。

このチュートリアルでは、このページのデフォルトの仕様 ([スタート] メニューの Tutorial.exe へのショートカット) をそのまま利用します。

レジストリ データを構成する

[アプリケーション レジストリ] ページでは、アプリケーションに必要なレジストリ エントリを指定することができます。



メモ: *InstallScript プロジェクトは、デフォルトでアプリケーション アンインストールキー (必要に応じて HKEY_LOCAL_MACHINE の下の Software¥Microsoft¥Windows¥CurrentVersion¥Uninstall¥<GUID> または HKEY_CURRENT_USER ルートキー) を作成するスクリプト コード、その値、およびデータを含みます。これらのレジストリ エントリを指定する必要はありません。*

このチュートリアルでは、このページでレジストリ エントリの指定を行いません。レジストリのエントリについては、

チュートリアルの ステップ 2 "ショートカットとレジストリ データ" で説明されています。

インストール インタビューを使ってダイアログを選択する

[インストール インタビュー] ページでは、インストールの実行時にエンドユーザーに対して表示するダイアログ ボックスを選択します。このページの質問に対する回答に基づいて、プロジェクト アシスタントは対応するダイアログ関数をインストールスクリプトへ追加します。ダイアログに関連するスクリプトの変更については、チュートリアルのステップ 6で解説します。



タスク ***このチュートリアルでは次を行います:***

1. "使用許諾契約書ダイアログを表示しますか?" というテキストの下にあるオプションボタンで **[いいえ]** を選択します。
2. その他のオプションボタンを **[はい]** のままにします。

インストール用の言語を選択する

[インストール ローカリゼーション] ページでは、インストールがサポートする言語を指定することができます。別の言語に簡単にローカライズすることができるようにエンドユーザー インターフェイスで利用可能な文字列値および関連する識別子を指定することもできます。



エディション: *複数言語サポートは、InstallShield の Premier Edition で使用することができます。*



タスク ***このチュートリアルでは、次の要領で HelpFiles 機能の表示名を変更します:***

1. リストボックスで、**"機能文字列データ"** を選択します。右の表は、すべての機能文字列エントリを表示します。
2. **[値]** 列で、**HelpFiles** (識別子 IDS_FEATURE_DISPLAY_NAME2 に関連付けられている値) をクリックして、**Help Files** (空白スペースを挿入) に変更します。

インストールをビルドする

[インストールのビルド] ページでは、どの配布タイプをビルドするのかを指定できます (オプションで、配布ファイルをコピーする場所も指定できます)。



タスク ***このチュートリアルでは次を行います:***

1. **CD-ROM** オプションを選択します。
2. **[インストールのビルド]** をクリックします。

上部に [ビルド] タブを持つ出力ウィンドウが開き、ビルドの進行状況についての情報を表示します。[ビルド] タブに「ビルドが終了しました。 *日付と時刻*」が表示されたとき、ビルドは終了します。

インストールを実行する

インストールをInstallShield 内から実行するには、[実行] ツールバーボタンをクリックするか、 CTRL+F5 を押します。

インストールは、プロジェクト アシスタントの [インストール インタビュー] ページで選択したダイアログを表示します。プロジェクト アシスタントで入力した値は、適切なダイアログでエンドユーザーに対して表示されます。たとえば、プロジェクト アシスタントで指定した INSTALLDIR のデフォルト値は、実行時に [インストール先の選択] ダイアログで表示されます。このときエンドユーザーが異なるインストール先ディレクトリを選ぶと、TARGETDIR にはこの新しい値が格納されます。

インストールの終了後、インストール先のディレクトリを開いて、セットアッププログラムによってインストールされたファイルを確認してください。インストールが成功していれば、チュートリアルファイルがインストールされているはずです。

メンテナンス モード

既に製品がインストールされているシステム上で、ユーザーが 2 回目 (およびそれ以降) にインストールを実行をすると、インストールは*メンテナンス モード*で行われます。このメンテナンスモードを使って、ユーザーは初回インストールからの機能選択の変更、インストール済みの機能の修復、およびプログラム全体の削除が行えます。

プログラムのアンインストール

プログラムをアンインストールするには、[実行] ボタンをクリックして (または CTRL+F5 を押し)、[セットアップ メンテナンス] ダイアログから [削除] を選択します。これで、[プログラムの追加と削除] でアプリケーションを選択した場合と同じ結果になります。

基本のインストール プロジェクトを作成したあと、[インストール デザイナー] タブ (ウィンドウの上部) をクリックして、チュートリアルの 次のステップ で説明されているようにインストールを拡張、または微調整します。

InstallShield インターフェイスを使って作業する

基本のインストール プロジェクトを作成したあと、[インストール デザイナー] タブ (ウィンドウの上部) をクリックして、InstallShield のユーザー インターフェイスでプロジェクトを拡張、または微調整します。InstallShield のユーザー インターフェイスは機能カテゴリー別に並べられているので、プロジェクトに情報を簡単に追加または編集できます。ここから先のチュートリアルでは、異なる InstallShield ビューのいくつかを紹介します。

本ステップを一通り完了すると、以下の項目がマスターできます。

- プログラムの機能の表示プロパティの設定法。
- プログラムのセットアップの種類の定義。
- コンポーネントおよびファイル リンクの作成法。

機能のプロパティを設定する

まず最初に行うのは、プロジェクト アシスタントで作成した機能に対する追加プロパティの設定で、これには機能の表示名や説明などがあります。機能プロパティを編集するには、IDE の [機能] ビューへ移動します。



タスク ***機能のプロパティを設定するには、以下の手順に従います:***

1. **[移動]** メニューで、**[編成]** をポイントして **[機能]** をクリックします。
2. **[機能]** エクスプローラーで、DefaultFeature 機能を選択して **"説明"** プロパティを **[この機能は Tutorial App のプログラムファイルを含んでいます]** に設定します。
3. [新規機能] 機能を選択して **"説明"** プロパティを **[この機能は Tutorial App のヘルプ ファイルを含んでいます]** に設定します。各説明を入力すると、InstallShield はそれぞれの値を表すための文字列テーブルエントリ ({ID_STRING*n*} と表示) を作成します。
4. [**機能**] ビューにある機能の名前を、それぞれの表示名と同じ名前に変更します。機能の名前を変更するには、機能を 2 回クリックして名前をハイライト表示し、新しい名前を入力します。

実行時、エンド ユーザーが [カスタム] セットアップ タイプを選択した場合、インストールはユーザーがどの機能をインストールするかを選択できるダイアログを表示します。このダイアログには、ここで設定した機能の表示名および説明が表示されます。

"セットアップの種類" プロパティの設定

セットアップの種類はインストールされる機能のコレクションです。一般的なインストールは [完全] と [カスタム] セットアップ タイプを提供します。[完全] セットアップ タイプはすべての機能をインストールするのに対して、[カスタム] セットアップ タイプはエンド ユーザーがインストールする機能を選択することができるダイアログを表示します。

IDE の [セットアップの種類] ビュー ([ビュー リスト] の [編成] ノードの下) 内にあるセットアップの種類プロパティを変更します。



タスク ***各セットアップの種類について、機能名の前にあるボックスを選択することでインストールする機能を選択します:***

1. [完全] セットアップの場合、両方の機能を選択します。
2. [カスタム] セットアップの場合、両方の機能を選択します。

コンポーネントおよびファイルリンクを作成する

[ファイルとフォルダー] ビューの追加ファイルにリンクを作成することができます。このステップでは、HelpFiles 機能にファイルを 1 つ追加します。[ファイルとフォルダー] ビューでファイルを追加すると、セットアップベストプラクティス規則に従ってIDE がコンポーネントを作成します。



タスク ***ヘルプファイル機能にある新規コンポーネントにソースファイル Tutorial.html を追加するには、以下の手順を実行します。***

1. **[ファイルとフォルダー]** ビュー ([ビュー リスト] の **[アプリケーション データ]** ノードの下) を開きます。
2. ビューの最上部にある機能のリストから、**ヘルプファイル** を選択します。

3. [インストール先コンピューターのフォルダー] 領域で、[**インストール先コンピューター**] アイコンを右クリックし、[コンポーネントの表示] が選ばれていることを確認します。
4. **[アプリケーション ターゲット フォルダー]** アイコンを右クリックし、**[新規コンポーネント]** を選択します。
5. 新しいコンポーネントの名前を HelpComponent に変更します。
6. [ソース コンピューターのフォルダー] 領域で、TutorialHelp.html が含まれるソースフォルダーを選びます。
7. [ソース コンピューターのファイル] 領域にある TutorialHelp.html アイコンを、**HelpComponent** アイコンまでドラッグします。

コンポーネントにリンクされるファイルのリストに変更が無い、この種のファイルリンクを「*スタティック ファイル リンク*」と呼びます。ビルド毎に内容に違いがあるディレクトリ(及びそのサブディレクトリ)にリンクするには、「ダイナミック ファイル リンク」を参照してください。



ヒント: *InstallShield の依存関係スキャナーを使うと、アプリケーションについて現段階でプロジェクトに含められていない必要な追加ファイルを確認できます。たとえば、Tutorial App は MFC を使用するので、MFC のランタイムライブラリがインストールされていないシステムをターゲットにする場合は、[再配布可能ファイル] ビューでプロジェクトに MFC ランタイムオブジェクトを加える必要があります。*

チュートリアルの次のステップでは、インストール プロジェクトのリリースイメージのビルド方法について説明します。

リリースのビルド

インストールをテストする前に、リリースをビルドする必要があります。リリースイメージには、CD-ROM、フロッピーディスク、またはネットワーク経由でエンドユーザーに配布するすべてのファイルが含まれます。

リリース ウィザードを利用すると、最も簡単に新しいリリースを作成することができます。リリース ウィザードでは、使用するメディアの種類 (たとえば CD-ROM) や、メディア上のファイルを圧縮するかなど、リリースのプロパティを指定します。リリース ウィザードの起動は、[リリース ウィザード] ツールバーボタンをクリックするか、[ビルド] メニューから [リリース ウィザード] を選択して行います。

[ようこそ] パネルで [次へ] をクリックして、必要なリリース設定を行います。いずれのパネルでも [ヘルプ] をクリックすると、現在のステップに関する説明が表示されます。

リリースに名前を付ける

[リリースの指定] パネルでは、リリース名を指定します。リリース名は、ビルドされたリリースを配置するフォルダーの名前として使用されます。このサンプルでは、*cdrom* という名前で新規にリリースを作成してください。

メディアタイプと一般オプションの選択

[メディアの種類] パネル

[メディア タイプ] パネルでは、リリースをビルドするメディアの種類を指定します。ここで指定するメディアの種類は、リリース ウィザードが作成するディスク イメージ フォルダーのサイズを示します。たとえば CD-ROM 用にリリースをビルドすると、リリース ウィザードがディスクイメージを複数のフォルダーに分割し、それぞれのフォルダーサイズを 650MB 以下とします。

ここでのチュートリアルでは、CD-ROM を選択してください。

[次へ] をクリックして、作成するリリースの一般オプションを指定してください。

[一般オプション] パネル

[一般オプション] パネルでは次の処理が可能です。

- インストールを配布するための自己展開型実行可能ファイルを作成する。
- コマンドライン オプションを Setup.exe へ渡す
- プリプロセッサ変数定義をコンパイラへ渡す。
- コンパイルしたスクリプト ファイル (.inx ファイル) をキャビネット ファイルに配置するかどうかを選択する。

このチュートリアルでは、パネルのデフォルト設定をそのまま利用します。

パスワードとサポートするプラットフォームを指定する

[パスワード] パネル

[パスワード] パネルではインストールのパスワードを指定することができます。またその場合 OnCheckMediaPassword イベント ハンドラー関数のデフォルトコード内でパスワード確認コードを実行するかどうかも指定することができます。

このチュートリアルでは、パスワードの指定は行いません。

[次へ] をクリックして、現在のリリースでサポートするオペレーティング システムを指定します。

[プラットフォーム] パネル

[プラットフォーム]パネルでは、現在のリリースでサポートするオペレーティング システムを指定できます。

このチュートリアルでは、デフォルトの選択 (**"プラットフォーム" プロジェクト プロパティで指定されたプラットフォームを使用"**) をそのまま利用します。

セットアップ言語と含める機能を指定する

[セットアップ言語] パネル

[セットアップ言語] パネルでは、インストールを実行することが可能な言語の指定や、エンドユーザーによるセットアップが表示する言語の選択を可能にするダイアログを表示するかどうかを指定することができます。

ウィザードはインストールに、このパネルで選択した言語固有の要素（文字列エントリやダイアログなど）だけを組み込みます。製品プロパティやビルトインアクションなど、すべての言語非依存のリソースは当然含まれます。

このチュートリアルでは、デフォルトの選択のままにしてください。

[次へ] をクリックして、現在のリリースに含む機能を指定します。

[機能] パネル

[機能] パネルでは、ビルドしたリリースに含める機能を指定することができます。

このチュートリアルでは、デフォルトの選択（**選択するかどうかの判別に "ビルドに含める" 機能プロパティを使用**）をそのまま利用します。

メディア レイアウトとダイアログの外観を定義する

[メディア レイアウト] パネル

[メディア レイアウト] パネルでは、各機能またはすべての機能に対して、機能のファイルをキャビネット フォルダーに保存するか、ディスク イメージ内に圧縮せずに配置するかを指定します。

このチュートリアルでは、デフォルトの選択（キャビネット ファイル）をそのまま利用します。

[ユーザー インターフェイス] パネル

[ユーザー インターフェイス] パネルでは、インストールのエンド ユーザーダイアログの外観を指定します。

このチュートリアルでは、デフォルトの選択のままにしてください。

インターネット オプションの指定およびアプリケーションのデジタル署名

[インターネット オプション] パネル

[インターネットオプション] パネルでは、様々なインターネット関連オプションを指定できます。すべてのリリースは、メディアの種類を問わずインターネット上で実行できます。

このチュートリアルでは、[セットアップのデフォルトの Web ページを作成する] ボックスをチェックして、その他のデフォルトの選択はそのままにします。

[デジタル署名] パネル

[デジタル署名] パネルでは、アプリケーションにデジタル署名を行うことができます。アプリケーションにデジタル署名をすることで、アプリケーション内のコードが発表以来変更または破損していないことをエンドユーザーに対して保証することができます。

このチュートリアルでは、デフォルトの選択のままにしてください。

アップデートとポスト ビルド情報を指定する

[アップデート] パネル

[アップデート] パネルでは、リリースの形式と現在のリリースをアップデートとして実行できる既存のリリースを指定できます。

このチュートリアルでは、デフォルトの選択のままにしてください。

[ランチャ オプション] パネル

[ポストビルド] オプション パネルでは、リリースのビルドが完了した後に、ディスク イメージ フォルダーをフォルダーまたは FTP サイトにコピーするか、またはバッチ ファイルを実行します。

このチュートリアルでは、デフォルトの選択のままにしてください。

設定を確認する

[概要] パネルは、現在のリリースに関する リリース ウィザードの設定を表示します。設定が正しい場合、[リリースの ビルド] チェック ボックスを選択して [完了] をクリックし、リリースをビルドします。そうでない場合は、[戻る] をクリックして設定を修正してください。

進行中のビルドに関するステータスメッセージは、アウトプットウインドウに表示されます。

ビルドが完了すると、CD にコピーするファイルは、次のディレクトリに置かれます:

<ProjectFolder>¥Tutorial¥cdrom¥DiskImages¥DISK1.

チュートリアル後半のステップで行うプロジェクトに対する変更が終了したら、最新リリースの再ビルドを行います。この処理は、[ビルド] ツールバーボタンをクリックし、[ビルド] メニューから [ビルド] を選択するか、あるいは F7 を押して実行します。

前にこのチュートリアルで行った要領でインストールを実行することができます。インターネット インストールとしてインストールを実行するには、ツールバーの [リリース フォルダーを開く] ボタンをクリックして Setup.htm を起動します。

チュートリアルの次のステップでは、インストールのショートカットとレジストリデータの作成方法について説明します。

インストールのトラブルシューティング

インストールを実行してもファイルがインストールされなかった場合は、プロジェクトについて下記の点をチェックしてください。

- TARGETDIR が適切な値に設定されていることを確認します。これは、[一般情報] ビューで設定されます。

  次は、本チュートリアルの推奨値です。

  [ProgramFilesFolder]TutorialCo¥TutorialApp

- セットアップの種類に、関連付けられた機能が指定されていることを確認します。
- 機能にコンポーネントとファイルが関連付けられていることを確認します。
- インストールに対して何らかの変更を行った場合は、[ビルド] ボタンをクリックするか F7 キーを押して、プロジェクトの再ビルドが必要です。

ステップ 2: ショートカットとレジストリ データ

このステップでは、次の処理を行う場合の IDE 利用法を説明します。

- プログラムショートカットを作成する
- レジストリ情報の作成

ショートカットの作成

[ショートカット] ビューでは、ショートカットの作成と変更を行うことができます。ショートカットのプロパティには、表示名、ターゲット実行可能ファイルおよび引数、表示用アイコンなどがあります。プロジェクト アシスタントを利用すると、エンド ユーザーの [スタート] メニューにある Programs フォルダーに、Tutorial App へのショートカットが自動的に作成されます。



タスク

このセットアップでは、エンド ユーザーのデスクトップ上にショートカットを作成します。

1. [**ショートカット**] ビューに移動します。
2. **[デスクトップ]** アイコンを右クリックし、**[新規ショートカット]** を選択します。**[ショートカット ターゲットを参照する]** ダイアログ ボックスが開きます。
3. ダイアログ ボックスでは、**探す場所** ドロップダウンメニューから **[アプリケーション ターゲットフォルダー]** を選択し、ファイルリストから Tutorial.exe を選びます。
4. **[開く]** をクリックして **[ショートカットターゲットを参照する]** ダイアログ ボックスを閉じます。
5. ショートカットアイコンの内部名を、*tutorial* などに変更します。

次に、下記のショートカット用プロパティを設定します。

テーブル 2 ・ ショートカットのプロパティ

プロパティ	値	コメント
表示名	Tutorial App	名前を別の言語へ容易に翻訳できるように、InstallShield はプロジェクトの文字列に表示名を追加します (また、中括弧にかこまれた文字列識別子が、[表示名] フィールドに表示されます)。
ターゲット	<TARGETDIR>¥Tutorial.exe	
アイコン ファイル	<TARGETDIR>¥Tutorial.exe	
アイコン インデックス	0	
作業ディレクトリ	<TARGETDIR>	



ヒント: ユーザーのマシン上に既に存在するファイルへのショートカットを作成するには、可能なときにファイルへのパスを示すシステム変数を利用してパスを入力します。たとえば、ユーザーマシンの Windows または WinNT フォルダーに置かれた Windows Notepad のコピーを起動するには、ショートカットターゲットに <WINDIR>¥Notepad.exe と入力します。

レジストリ データを作成する

インストールに求められる機能の 1 つとして、ターゲット システムのレジストリへの情報の書き込みがあります。コンポーネントにレジストリを追加するには、[レジストリ] ビューを利用することができます。

たとえば、HKEY_LOCAL_MACHINE¥Software¥Tutorial Co¥Tutorial¥1.00.0000 の下に *TutorialData* というレジストリ値を作成するには、以下の手順を実行します。



タスク ***レジストリデータを作成するには、以下の手順を実行します。***

1. [**レジストリ**] ビューを開きます。
2. **[インストール先コンピューターのレジストリ]** ビュー 領域で **[インストール先コンピューター]** を右クリックして、**[新しいレジストリセット]** を選択します。
3. レジストリセットの名前を *tutorial* に変更します。
4. *tutorial* レジストリセットの下にある HKEY_LOCAL_MACHINE を右クリックして **[新規]** サブメニューから **[キー]** を選択します。
5. キーの名前を *Software* に変更します。
6. サブキー *Tutorial Co*、*Tutorial*、と *1.00.0000* に関しても同じ操作を繰り返します。
7. **[インストール先コンピューターのレジストリデータ]** 領域で、右クリックして、**[新しい文字列値]** を選択します。
8. 値を *TutorialData* に変更します。
9. TutorialData 値をダブルクリックして、**[値]** データ フィールドに *<TARGETDIR>* と入力します。
10. *tutorial* レジストリセットをクリックして、[**レジストリセットインストール条件**] パネルで [DefaultComponent] をチェックします。

インストール実行時に Tutorial.exe コンポーネントを含むセットアップの種類や機能のコレクションをユーザーが選択すると、ターゲット システム上にレジストリデータが作成されます。

ショートカットが作成されたことを確認する



タスク ***インストールがショートカットを作成したことを検証するには、以下の手順を実行します。***

1. **[ビルド]** ツールバーボタンをクリックするか、**F7** を押してプロジェクトを再ビルドします。
2. **[実行]** ボタンをクリックするか、CTRL+F5 を押してプロジェクトを実行します (最初にシステムからプログラムの既存バージョンをすべて削除します)。これで Tutorial App のショートカットが、**スタート** メニューの **Programs** フォルダーに作成されているはずです。

レジストリデータが作成されたことを確認する

タスク ***インストールがレジストリデータを作成したことを検証するには、以下の手順を実行します。***

1. ショートカットから Tutorial App を起動します。
2. **Tutorial** メニューから **Verify Registry Data** を選択します。レジストリデータが作成されていれば、*<TARGETDIR>* テキストを含むメッセージボックスが表示されます。

チュートリアルの次のステップでは、COM サーバー（自己登録ファイル）の登録方法について説明します。

ステップ 3: COM サーバーの登録

ほとんどのファイルに関して、インストールが果たすべき唯一の役割は、ソース メディアからターゲット システムにファイルをコピーすることです。これに対し一部のファイルについては、ターゲット システムへのファイル登録作業をインストーラーが実行する必要があります。別途処理が必要なファイルのカテゴリは *自己登録型ファイル* です。

本ステップでは、Tutorial.ocx 本体およびこれが使用する HTML ファイルのインストールと登録を行うコンポーネントを作成します。

COM サーバーは通常 DLL や .ocx の形を取りますが、こうした自己登録ファイルは、これらを使用するアプリケーションや Web ページが認識できるようあらかじめターゲット システムのレジストリへ書き込んでおくため、追加情報を必要とします。

タスク ***自己登録ファイルをインストールするには、以下の手順に従います:***

1. **[ファイルとフォルダー] ビュー**に移動します。
2. **[ファイルとフォルダー]** ビューの上部にある **[ビュー フィルター]** リストから **ProgramFiles** 機能を選択します。
3. **[ソース コンピューターのフォルダー] ペイン**で、ソース ディレクトリ 内の Tutorial.ocx を参照します。
4. **[ソースコンピューターのフォルダー]** ペインから **[インストール先コンピューターのフォルダー] ペインにある [アプリケーションのターゲットフォルダー]** に Tutorial.ocx をドラッグ アンド ドロップします。コンポーネント SelfRegFiles が［アプリケーションターゲット フォルダー］の下に作成されます。このコンポーネントには Tutorial.ocx が含まれ、その自己登録プロパティは **[はい]** に設定さています。これはコンポーネントを右クリックして **[プロパティ]** を開いて確認することができます。

次に、ProgramFiles 機能と関連付けられている新しいコンポーネントに HTML ファイルを追加します:

タスク ***ProgramFiles 機能と関連付けられている新しいコンポーネントに HTML ファイルを追加します:***

1. **[ファイルとフォルダー]** ビューの **[インストール先コンピューターのフォルダー] ペイン**で、**[アプリケーション ターゲットフォルダー]** を右クリックして **[新規コンポーネント]** を選択します。
2. コンポーネントの名前を *OcxHTML* に変更します。
3. TutorialCtrl.html を **[ソースコンピューターのファイル] ビュー**から *OcxHTML* コンポーネントにドラッグします。



タスク ***リリースを再ビルドし (F7 キーを押す) インストールを実行 (CTRL+F5 を押す) をした後、ファイルが正常に登録されたことを次の手順で確認できます。***

1. [**プログラム**] メニューのショートカットか、アイコンをダブルクリックして、Tutorial App を起動します。
2. **Tutorial** メニューから **COM Server Test** を選択します。
3. COM サーバーが正常に登録されていれば、「登録成功」のメッセージを示す HTML ページが表示されます。

チュートリアルの次のステップでは、ファイルの条件付きインストールについて説明します。

ステップ 4: 条件とプロパティ

このステップでは、ターゲット システムにデータを条件付きでインストールする方法について解説します。

インストール通常求められる機能は、特定の条件が満たされた場合にのみ必要なファイルをシステムにインストールすることです。たとえばある種のファイルは、特定のオペレーティング システムや言語にのみ必要であったり、ユーザー必要な権限を持つ場合に限りインストールできる、ということがあります。

コンポーネント (およびそのファイルとその他のデータ) を特定のオペレーティング システムにのみインストールさせるには、コンポーネントの "オペレーティング システム" プロパティを使います。コンポーネントのプロパティの変更は、[セットアップのデザイン] ビューを開いて該当する機能のアイコンを展開し、必要なコンポーネントを選択して行います。



タスク ***Windows 8 が搭載されたシステムだけにインストールされるコンポーネントを作成するには、次の手順を実行します。***

1. [**セットアップのデザイン**] ビューを開きます。
2. **HelpFiles** 機能を右クリックして、**[新規コンポーネント]** を選択します。
3. 作成されたコンポーネントの名前を *windows_8_files* に変更します。
4. コンポーネントの**ファイル**アイコンをクリックし、**ファイル**ペイン内で右クリックしてファイルを参照してソースフォルダーから ReadmeNT.txt ファイルを追加します。
5. windows_8_files コンポーネントをクリックして、コンポーネントのプロパティ グリッドを表示します。
6. コンポーネントの "オペレーティング システム" 設定を選択してから、[参照] ボタンをクリックします。[**プラットフォーム**] ダイアログが開きます。
7. [Windows 8] チェック ボックスを選択します。
8. **[OK]** をクリックします。

(F7 を押して) 再ビルドの後、(CTRL+F5 を押して) インストールを実行すると、ターゲット システムが Windows 8 を実行している場合に限りコンポーネントに含まれるファイルおよびその他のデータがインストールされるようになります。

チュートリアルの次のステップでは、プロジェクトのスクリプトの修正方法について説明します。

ステップ 5: スクリプトでの作業

InstallShield は、InstallScript 言語を使って、インストールを制御します。InstallShield ビューでプロジェクトのスクリプトを修正することができます。



ヒント: *CTRL+M を押して [スクリプト エディター] を最大化し、また CTRL+M をもう一度押して元に戻します。*

InstallScript プロジェクトのスクリプトはイベントドリブン型モデルと呼ばれるもので、定義済みの一連の関数を特定の順序で呼び出しながら処理を進めてゆきます。ビルトイン イベント ハンドラー、およびイベント ハンドラー関数の呼び出し順序については、「イベント ハンドラー」を参照してください。

スクリプト中の定義済み関数は [関数] ツリーに表示されます。既存の関数の表示および編集をするには、[関数] ツリーの該当する関数名をクリックします。



メモ: *イベント ハンドラー関数はスクリプト内にはっきりと記述されていなくても呼び出されます。イベント ハンドラー関数がスクリプト中に記述されていない場合は、そのデフォルトのコードが使用されます。*

イベント ハンドラー関数の追加および編集をするには、スクリプト エディターの左側にあるドロップダウン リストから適切なイベント カテゴリ (Before Move Data など) を選択してから、右側のドロップダウン リストから必要なイベント名 (Begin など) を選択します。スクリプト中で明示的に定義されたイベント ハンドラー関数は、太字で表示されます。

たとえば OnBegin イベント ハンドラーは、初回インストール時およびメンテナンス モード時の両方で、インストール スクリプトの最初に呼び出される関数です。



タスク

この OnBegin イベント ハンドラーに独自のコードを追加するには、最初にこのための関数を作成します。

1. 左側のイベントカテゴリリストから **Before Move Data** を選択します。
2. 右側のイベント ハンドラーリストから **Begin** を選択します。

これで、次のコードがスクリプトに追加されます。

```
////////////////////////////////////////////////////////////////////////////
//
//  関数: OnBegin
//
// イベント: Begin イベントはインストールの過程で常に最初のイベントとして
// 送られます。
//
////////////////////////////////////////////////////////////////////////////
function OnBegin()
begin
    // 作業: たとえば、デフォルトの非 UI 設定を変更します。
    //
    // またカスタムの初期化処理、要件のチェックなどを実行することもできます。
end;
```

この **OnBegin** 関数には、インストールの最初に実行させるコードを置きます。

関数ウィザードの使用

ここでは、MessageBox 関数の呼び出しを追加する関数ウィザードの利用法について説明します。これはユーザーがインストールを始めた時にメッセージを表示します。



タスク ***OnBegin 関数への MessageBox 関数の追加は、次の手順で行います。***

1. begin と end の間にあるコメント行 (// で始まる行) を削除し、テキストの挿入ポイントをこれらの行の間に置きます。
2. CTRL+I を押して、**関数ウィザード**を起動します。
3. **[関数カテゴリ]** リストから、**[ビルトイン ダイアログ]** を選択します。
4. **[関数名]** リストから、**MessageBox** を選択します。
5. **[次へ]** をクリックします。
6. 表示するメッセージを含む **szMsg** フィールドには、*"チュートリアル インストールへようこそ"* と入力します。(引用符が必要です)。
7. 表示するメッセージボックスのタイプを指定する **nType** ドロップダウンリストでは、**INFORMATION** を選択します。
8. [**完了**] をクリックすると、ここで指定した関数呼び出しがスクリプトに挿入されます。

この結果 **OnBegin** 関数は、次のように表示されます。

```
function OnBegin()
begin
MessageBox (“チュートリアル インストールにようこそ”, INFORMATION );
end;
```

スクリプトのコンパイル

スクリプトを修正した場合、加えた変更を有効にするには、スクリプトをコンパイルする必要があります。スクリプトをコンパイルするには、[コンパイル] ツールバー ボタンをクリックし [ビルド] メニューから [コンパイル] を選択するか、CTRL+F7 を押します。コンパイル中のステータスメッセージは、IDE の出力ウインドウに表示されます。



ヒント: *スクリプトのコンパイル中にエラーや警告が発せられた場合は、エラーメッセージをダブルクリックすると、エラーの発生したスクリプト行がハイライト表示されます。*

([実行] ツールバー ボタンをクリックするか、CTRL+F5 を押して) インストールを実行すると、[ようこそ] ダイアログの前に、メッセージ ボックスが表示されます。

初回インストールの時だけメッセージを表示する

OnBegin 関数は、初回インストール時およびメンテナンスモードインストール時の両方で呼び出されるので、メッセージボックスはユーザーがプログラムを再インストール、又は削除する時にも表示されます。初回インストール時のみにメッセージボックスを表示するには、ユーザーがインストールを実行するのは初めてかどうかを判断する if ステートメント の中に MessageBox 呼び出しを配置します。

InstallScript は *MAINTENANCE* というシステム変数を定義しており、メンテナンス モードでインストールが実行している場合は True、そうでない場合は False を取ります。メッセージボックスを初回インストール時のみに表示するには、**OnBegin** 関数を次のよう変更してください。

```
function OnBegin()
begin
if (!MAINTENANCE) then
    MessageBox("チュートリアル インストールにようこそ!", INFORMATION);
endif;
end;
```

このインストールをコンパイルして実行すると、メッセージボックスは初回インストール時にのみ表示され、メンテナンス モードインストール時には現れなくなります。

InstallScript

InstallScript 言語には、レジストリおよび INI ファイルの処理、ターゲット オペレーティング システムの特性テスト、またはダイアログの表示など、インストール関連タスクを実行する数百以上の関数が用意されています。これらの詳細と使用例については、「InstallScript 言語リファレンス」を参照してください。

チュートリアルの次のステップでは、インストールが表示するダイアログを変更する方法について説明します。

ステップ 6: ユーザー インターフェイスの変更

このステップでは、インストールのユーザー インターフェイスを変更する 3 種類の方法について説明します。

1. ユーザーの入力に応じて、異なるアクションを行う。
2. ユーザー インターフェイス関連のスクリプト関数を修正して、表示するダイアログを変更する。
3. **ダイアログ エディター**を使用して、ダイアログのレイアウトとプロパティを変更する。

ユーザー入力の処理

InstallScript コード中で、実行時に表示されるダイアログ関数を含むのは次のイベント ハンドラー関数です:

- **OnFirstUIBefore**、初回インストール時のデータ転送を行う前に表示されるダイアログを含む。
- **OnFirstUIAfter**、初回インストール時のデータ転送を行った後に表示されるダイアログを含む。
- **OnMaintUIBefore**、メンテナンスモードインストール時のデータ転送を行う前に表示されるダイアログを含む。
- **OnMaintUIAfter**、メンテナンス モード インストール時のデータ転送を行った後に表示されるダイアログを含む。

メモ: *プロジェクトの "メンテナンスの有効化" プロパティが [アンインストールまたはメンテナンスをしない] に設定されている場合は、OnMaintUIBefore および OnMaintUIAfter イベント ハンドラー関数は呼び出されません。*

プロジェクト アシスタントを使って作成されたデフォルト InstallScript プロジェクトは、初回インストールのユーザー インターフェイスを定義する **OnFirstUIBefore** イベント ハンドラー関数を定義します。**OnFirstUIBefore** は、ダイアログ関数を呼び出して、プロジェクト アシスタントの [インストール インタビュー] ページで指定したダイアログを表示します。たとえば、次のコードはエンド ユーザーに対してユーザー名と会社名の入力を要求するダイアログを表示します:

```
szMsg = "";
szTitle = "";
nResult = SdRegisterUser( szTitle, szMsg, szName, szCompany );
```

エンド ユーザーのユーザー名と会社名は最後の 2 つの変数で戻されます。これらを自由に利用してレジストリ キーの作成や、ファイルに格納した情報との照合が可能です。

表示ダイアログを変更する

InstallScript 言語には、インストールのユーザー インターフェイス用に、あらかじめ定義された多数のダイアログが用意されています。



タスク

デフォルトのユーザー情報ダイアログを、ユーザーにシリアル番号の入力も要求するダイアログに置き換えるには、SdRegisterUser 関数を SdRegisterUserEx の呼び出しと置き換えます:

1. **InstallScript** ビューの **[関数]** ノードで **OnFirstUIBefore** を選択します。
2. 文字列スクリプト変数 szSerial を宣言するには、**OnFirstUIBefore()** 関数の begin の前に次のスクリプトを追加します。

   ```
   string szSerial;
   ```

3. SdCustomerInformation の呼び出しを含む行を削除します :

   ```
   Dlg_SdRegisterUser:
       szMsg = "";
       szTitle = "";
       nResult = SdRegisterUser( szTitle, szMsg, szName, szCompany );
       if (nResult = BACK) goto Dlg_SdLicense2;
   ```

4. 削除した行の代わりに、次の行を入力します:

   ```
   Dlg_SdRegisterUserEx:
       szMsg = "";
       szTitle = "";
       szSerial = "";
       nResult = SdRegisterUserEx(szTitle, szMsg, szName, szCompany, szSerial );
       if (nResult = BACK) goto Dlg_SdLicense2;
   ```

5. Dlg_SdRegisterUser にポイントしていた goto ステートメントを Dlg_SdRegisterUserEx へポイントするように変更します。
6. CTRL+F7 を押して、スクリプトをコンパイルします。

同様の手順で、インストールのユーザー インターフェイスに新しいダイアログを追加したり、既存のダイアログを別のものと置き換えることができます。

ダイアログ エディターを使用する

ダイアログ エディターを使って、インストールで表示されるダイアログの外観を修正することができます。

以前のステップで説明したように、エンドユーザーが入力するシリアル番号は SdRegisterUserEx ダイアログ ボックス上にプレーン テキストで表示されます。



タスク

ダイアログを修正して、エンド ユーザーが入力したパスワードが隠れるように設定するには、次の手順に従います。

1. **[ダイアログ]** ビュー（[リスト] ビューの [ユーザー インターフェイス] ノードの下）で、**SdRegisterUserEx** アイコンを右クリックして **[編集]** を選択します。
2. **英語 (U.S.)** アイコンを選択します。
3. **ダイアログ エディター**で、**シリアル番号** ラベルの編集フィールドを選択します。
4. **編集**コントロールの**" パスワード"** プロパティを False から True に変更します。



ヒント: *CTRL+M を押すと、[ダイアログ エディター] ビューが最大化します。*

プロジェクトを再ビルド（F7 を押す）してから実行する（CTRL+F5 を押す）と、ユーザーが入力するシリアル番号は隠されて表示されます。



ヒント: *ダイアログをデフォルトの状態に戻すには、ダイアログのアイコンを右クリックして、[ダイアログをデフォルトに戻す] を選択します。*

基本の MSI プロジェクト チュートリアル

このチュートリアルでは、InstallShield を使用した基本の MSI インストール プロジェクトの作成、ビルド、実行、およびその他各種の機能について必要な操作方法を順を追って説明します。

チュートリアルの構成は複数のステップに分かれています。最初のステップ (ステップ 1: プロジェクトの作成、ビルドおよびテスト) を除き、残りのステップはすべて独立した内容になっているので、これらは特に順番にこだわることなく現在の作業に必要な情報のみを調べることができます。

本チュートリアルでは、インストールが取り扱うタスクの多くの処理方法について解説します。

- ファイルのインストール
- ショートカットおよびレジストリ データの設定
- データの条件付きインストール
- COM サーバーの登録
- インストールのユーザー インターフェイスの変更
- リリースのビルド
- インストールのテスト

チュートリアルを通して、InstallShield ヘルプ ライブラリの該当箇所へのリンクが設けてあります。

ステップ 1: プロジェクトの作成、ビルドおよびテスト

このステップでは、インストール プロジェクトの作成、リリース イメージのビルド、およびインストールのテストについて説明します。本ステップを一通り完了すると、以下の項目がマスターできます。

- プロジェクト アシスタントを使った新規プロジェクトの作成法。
- インストール プロジェクト内の多くの設定を構成。
- 機能、コンポーネント、およびファイル リンクの指定方法。
- 複製および配布用リリースのビルド方法。
- インストールの実行。

基本の MSI インストール構成ブロック

基本の MSI インストールは 2 段階で構成されます:

テーブル 1・基本の MSI インストール内のレベル

レベル	説明
コンポーネント	コンポーネントとは、開発者から見て、個別にインストール可能な最小の製品構成単位のことです。コンポーネントには、ファイル、ショートカット、レジストリ データなど、ターゲット システムにインストールされるデータが含まれます。ユーザーがコンポーネントを直接処理することは決してありません。 1 つのコンポーネントを複数の機能に配置することもでき、ユーザーがこのコンポーネントと関連付けられた機能の 1 つを選ぶと、このコンポーネントのデータがインストールされます。
機能	機能とは、エンドユーザーから見て、個別にインストール可能な最小の製品構成単位のことです。ユーザーがインストール中に [カスタム] セットアップ タイプを選択すると、どの機能をインストールするかをユーザーが選択することができるダイアログが表示されます。 各機能はコンポーネントを含みます。

チュートリアルファイル

本チュートリアルで作成するインストールは、*Tutorial App* という名前のアプリケーションについて、そのインストールと設定を行うものです。Tutorial App のソース ファイルは InstallShield Program Files フォルダーの Samples サブフォルダーにあります。デフォルト インストール先は次の場所です:

C:¥Program Files¥InstallShield¥2022¥Samples¥WindowsInstaller¥Tutorial Project

新規基本の MSI プロジェクトの作成

チュートリアルの最初のステップは、新しい基本の MSI プロジェクトの作成です。

タスク ***新規で基本の MSI プロジェクトを作成するには、以下の手順を実行します。***

1. **[ファイル]** メニューから **[新規作成]** を選択するか、ツールバーの **[新規プロジェクト]** ボタンをクリックします。**[新規プロジェクト]** ダイアログ ボックスが開きます。
2. **Windows Installer** タブをクリックして、**基本の MSI** プロジェクトを選択します。
3. **"プロジェクト名"** フィールドに **Tutorial** と入力します。
4. **[場所]** ボックスでは、デフォルト値をそのまま残します。
5. **[[プロジェクト名] サブフォルダーにプロジェクト ファイルを作成]** を選択します。
6. **[OK]** をクリックします。

InstallShield がプロジェクトを作成して開きます。

InstallShield は、*ProjectName*.ism という名前のプロジェクト ファイルを作成します。この場合、Tutorial.ism となります。プロジェクト ファイルには、InstallShield で構成されたすべての設定が格納されます。プロジェクトを他のマシンに移動する場合は、移動先のシステムに .ism ファイル (およびインストールソースファイル) をコピーします。



ヒント: *新しいプロジェクト ファイルが作成されるデフォルト ディレクトリを変更するには、[オプション] ダイアログ ボックスを開きます ([ツール] メニューから [オプション] をクリック)。[ファイルの場所] タブにある "場所" 設定に新しいパスを入力します。*

アプリケーション情報を入力する

新規プロジェクトを作成すると、プロジェクト アシスタントが起動し、プロジェクトやアプリケーション情報の指定手順を案内します。プロジェクト アシスタントの最初のページでは、インストール作成プロセスの概要が図式化されます。プロジェクト アシスタントを利用するには、ビューの下にある [アプリケーション情報] アイコンをクリックします。

[アプリケーション情報] ページでは、プロジェクトがインストールするアプリケーションについての一般情報を指定します。



タスク ***チュートリアル用のアプリケーション情報を指定するには、以下の手順を実行します。***

1. **"会社名を指定"** 設定に **TutorialCo** を入力します。これにより、**"会社の Web アドレスを指定"** 設定の情報が自動的に更新されます。
2. **"アプリケーション名を指定"** 設定に **TutorialApp** を入力します。入力された値は、エンド ユーザーに表示されるダイアログに使用されます。また、[プログラムの追加と削除] に表示されるアプリケーションの名前として使用されます。
3. **"アプリケーション バージョンの指定"** および **"会社 Web アドレスの指定"** 設定は、デフォルト値のままにします。
4. **[プログラムの追加と削除] でアプリケーションと共に表示するアイコンを選択します]** の下にある **[参照]** ボタン をクリックして Tutorial.exe の場所を参照します。デフォルトの場所は、C:¥Program Files¥InstallShield¥2022¥Samples¥WindowsInstaller¥Tutorial Project です。.exe ファイルを開いて **アイコン インデックス 0** を選択します。

完了すると、[アプリケーション情報] パネルは次のように表示されます。

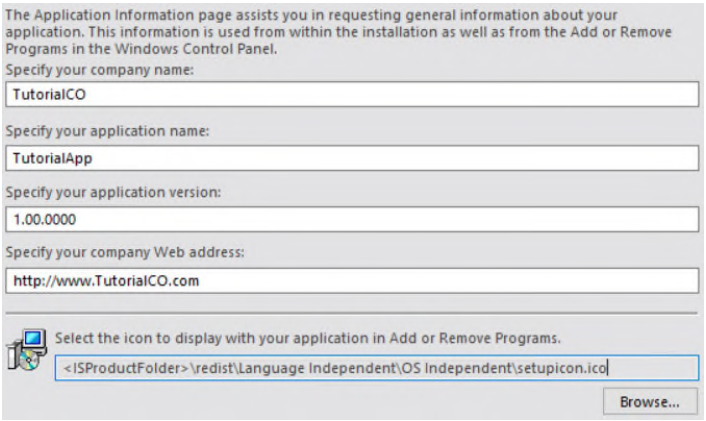


図 1: [アプリケーション情報] パネル

入力するアプリケーション名と会社名は *Windows スタート* メニュー上のアプリケーションのショートカットのデフォルトの場所、並びにプログラム ファイルのデフォルトのインストール先を指定する *INSTALLDIR* プロパティのデフォルト値を決定します。

メモ- *[INSTALLDIR] のデフォルト値は [ProgramFilesFolder] 会社名¥製品名 です。このうち [ProgramFilesFolder] の部分は、実行時にユーザーの Program Files フォルダーの位置に置き換えられます。Windows Installer が定義する他のディレクトリ関連のプロパティについては、Windows Installer プロパティ リファレンスの「インストーラーが設定するシステム フォルダー」セクションを参照してください。*

インストール要件の設定

[インストール要件] ページでは、ターゲット システムのインストールの要件を簡単に設定することができます。たとえば、アプリケーションを適切に実行するために特定の OS が必要な場合、このパネルの最初の部分で示すことができます。

オペレーティング システム

アプリケーションがターゲット システムに 特定の Windows バージョン以降を必要とする場合、[はい] を選択してからアプリケーションに必要な OS を選択します。

チュートリアルでは *[いいえ]* のままにします

ソフトウェア要件

アプリケーションのインストールに、ターゲット システムへ特定のソフトウェアがインストールされていることが必要な場合、*[はい]* を選択して該当するソフトウェアを選択します。必要なソフトウェアがターゲット システム上で見つからない場合に表示されるランタイム メッセージをカスタマイズするには、そのメッセージをクリックして編集します。

メモ- *このセクションのランタイムメッセージはソフトウェア要件が選択されるまでは表示されません。*

チュートリアルでは *[いいえ]* のままにします。

インストール アーキテクチャのカスタマイズ

[インストール アーキテクチャ] ページで、インストールがエンド ユーザーに対して表示する機能を指定することができます。機能とは、ユーザーから見て個別にインストール可能な最小の製品構成単位のことです。インストール中に [カスタム] セットアップ タイプを選択すると、個々の機能がエンドユーザーに対して表示されます。

メモ: *機能には、サブ機能やサブサブ機能などを含めることも可能で、インストールの要求に応じて必要なレベルの多層構造化が可能です。*

インストールアーキテクチャは現在デフォルト機能、*Tutorial_Files* を含みます。デフォルトの機能は、エンドユーザーがインストールを実行するときに常にインストールされます。このステップでは、インストールアーキテクチャへ別の機能を追加します。

タスク ***チュートリアルでは、新しい Help_Files 機能を追加します。***

1. **インストール アーキテクチャをカスタマイズしますか？**には **[はい]** を選択します。
2. **[インストール アーキテクチャ]** ノードを右クリックして、**[新規作成]** を選択します。
3. 新規機能に *Help_Files* と名前をつけます。

このステップを完了すると、インストールアーキテクチャはこの様になります。

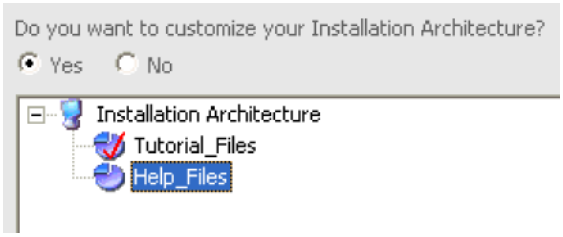


図 2: インストールアーキテクチャ

プロジェクトへファイルを追加する

次のステップでは、アプリケーションのファイルをインストール プロジェクトへ追加します。[アプリケーションファイル] ページでは、機能に関連付けるファイルを指定することができます。

このステップでは、Tutorial 実行可能ファイルを Tutorial_Files 機能へ追加します。

タスク ***Tutorial 実行可能ファイルを Tutorial_Files 機能へ追加するには、以下の手順を実行します。***

1. ページ上部の機能ドロップダウンリストから、Tutorial_Files 機能を選択します。
2. ツリー コントロール（トップノードは [インストール先コンピューター]）内にある **INSTALLDIR** ノードを選択します。
3. **[ファイルの追加]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. チュートリアル ファイル ソース ディレクトリに置かれた Tutorial.exe を参照します。

5. **[開く]** をクリックして、Tutorial_Files 機能にファイルを追加します。
6. **「追加したファイル...は依存関係を持つ可能性があります」**メッセージが表示されたとき、**[いいえ]** をクリックします。Tutorial.exe には依存関係は含まれません。ファイルが機能へ追加され、右側のファイルリストパネル内に表示されます。



メモ- *ファイルの隣にあるキーアイコンは、ファイルが機能のコンポーネントのキーファイルであることを示します。Windows Installer ではほとんどの場合、コンポーネントにキーファイルが 1 つ含まれている必要があります。Windows Installer サービスは各種の目的でコンポーネントのキーファイルを使用します。たとえばキーファイルの存在を調べてコンポーネントを修復する必要があるかどうかを判断する場合や、ショートカットのデフォルトのターゲットとしてキーファイルを使うなどの用途があります。実行可能ファイルを基本の MSI プロジェクトの機能に追加するとき、プロジェクト アシスタントは自動的にそれをファイルの為に内部で作成されるコンポーネントのキーファイルに設定します。詳細については、「コンポーネントのキーファイルを設定する」を参照してください。*

ショートカットの作成

[アプリケーション ショートカット] ページでは、ターゲット システムのデスクトップ又は [スタート] メニュー上にあるアプリケーションファイルのショートカットを指定することができます。デフォルトで、このページはインストール プロジェクトに入れた各実行可能ファイルのショートカットを表示します。これらは削除することができ、インストール プロジェクトに入れたショートカットを他のファイルに追加することもできます。



タスク ***Tutorial.exe へのショートカットをアクティブにするには、以下の手順を実行します。***

[起動 Tutorial.exe] アイコンをクリックします。**[スタート] メニューにショートカットを作成する**を選択したデフォルト設定をそのままにします。InstallShield は、インストールが実行されたときに、ユーザーの **[スタート]** メニュー上に Tutorial.exe へのショートカットを作成します。

レジストリ データを構成する

[アプリケーション レジストリ] ページでは、アプリケーションに必要なレジストリ エントリを指定することができます。

チュートリアルでは、このページでレジストリ エントリの指定は行いません。レジストリ エントリについては、チュートリアルのステップ 2: ショートカットとレジストリ データ" で説明します。

インストール インタビューを使ってダイアログを選択する

[インストール インタビュー] ページでは、インストールの実行時にエンドユーザーに対して表示するダイアログボックスを選択します。このページでの回答によって、プロジェクト アシスタントは対応するダイアログをインストール プロジェクトに追加します。



タスク ***チュートリアル用のダイアログを指定するには、以下の手順を実行します。***

1. **使用許諾契約ダイアログを表示しますか?** – **[いいえ]** を選択します。このオプションに **[はい]** を選択した場合は、使用許諾契約書ファイルを参照します。
2. **[会社名] と [ユーザー名] を入力するプロンプトを表示しますか ?** – **[はい]** を選択します。インストールは、この情報を求めるダイアログを表示します。
3. **ユーザーがアプリケーションのインストール場所を変更することができるようにしますか?** – **[はい]** を選択します。詳細については、「エンドユーザーによるインストール先の変更を許可する」を参照してください。
4. **ユーザーがアプリケーションの一部のみを選択してインストールできるようにしますか?** – **[はい]** を選択します。詳細については、「インストールが選択できるインストールを作成する」を参照してください。
5. **インストールの完了時にアプリケーションを起動するオプションをユーザーに提供しますか?** – **[はい]** を選択して、Tutorial.exe ファイル ([ProgramFilesFolder]TutorialCo¥Tutorial にある) を参照します。このオプションが **[はい]** に設定されると、最後のダイアログにはチェック ボックスがあり、エンドユーザーが **[完了]** ボタンを押すとアプリケーションが直ちに起動します。

インストール用の言語を選択する

[インストールのローカライズ] ページで、インストールが対応する言語を指定することができます。文字列値と関連識別子を指定して、エンド ユーザーのインターフェイスで簡単に他の言語にローカライズすることができます。



エディション: *InstallShield をインストールした際に選択した言語に追加して他の言語を選択するには、InstallShield Premier Edition が必要です。*



タスク ***このチュートリアルでは英語(U.S.) を選択した状態にし、次の手順に従ってインストールの機能の表示名を変更します。***

1. リストボックスで、**"機能文字列データ"** を選択します。右の表は、すべての機能文字列エントリを表示します。
2. **[値]** 列で **Tutorial_Files** (識別子 IDS_FEATURE_DISPLAY_NAME2 と関連付けられた値) をクリックして、その値を **「チュートリアル ファイル」** に変更します。
3. **Help_Files** (識別子 IDS_FEATURE_DISPLAY_NAME3 と関連付けられた値) をクリックして、その値を**「ヘルプ ファイル」**に変更します。



メモ: *詳細については、「複数言語インストールの作成」を参照してください。*

インストールをビルドする

インストール プロジェクトのアーキテクチャを定義し、アプリケーションファイルを追加し、ショートカットを作成し、ダイアログを選択したあと、インストールのビルドの準備が整います。

[インストールのビルド] ページでは、ビルドするタイプを指定し、再配布可能ファイルをコピーする場所も指定することができます。

タスク ***作成したばかりのインストールをビルドするには、以下の手順を実行します。***

1. CD-ROM オプションを選択します。
2. **[インストールのビルド]** をクリックします。

[出力] ウィンドウが開き、ビルドの進行状況についての情報が表示されます。

メモ *ビルドが警告 -7235 を生成しました。これは予期する警告です。この警告を解決せずに次のステップに進むか、この警告を解決する場合は、必要に応じて [一般情報] ビューの [ソフトウェア識別タグ] 領域で設定を構成できます。"ソフトウェア識別タグの使用" 設定で [はい] が選択されている時に、4 つの必須識別設定 ([一般情報] ビューの "一意な ID"、"一意な登録 ID"、"タグ作成者"、および "タグ作成者 ID" 設定) のうち最低 1 つの値が設定されていなかった場合、空白の設定ごとに、ビルド警告 -7235 が一回発生します。詳細については、「製品のソフトウェア識別タグを含める」を参照してください。*

次のステップでは、インストールを実行します。

インストールを実行する

このチュートリアルでプロジェクト アシスタントのステップを完了すると、Tutorial 実行可能ファイルをインストールする完全なインストール が作成されます。

タスク ***インストールを実行するには、以下の手順を実行します。***

ツールバー上の **[実行]** ボタンをクリックするか、または、CTRL+F5 を押します。

インストールは、プロジェクト アシスタントの [インストール インタビュー] ページで選択したダイアログを表示します。プロジェクト アシスタントで入力した値は、適切なダイアログでエンドユーザーに対して表示されます。たとえば、プロジェクト アシスタントで指定した *INSTALLDIR* のデフォルト値は、インストーラーの実行時に [インストール先の選択] ダイアログ ボックス上に現れます。このときエンドユーザーが異なるインストール先ディレクトリを選ぶと、*INSTALLDIR* にはこの新しい値が格納されます。

インストールが完了した時点でインストール先のディレクトリを開き、インストールされたファイルを確認してください。インストールが成功していれば、チュートリアルファイルがインストールされているはずです。

メンテナンス モード

システム上にインストール済みのアプリケーションについて、ユーザーが 2 回目 (およびそれ以降) にインストールを実行すると、インストールは *メンテナンス モード* で実行されます。このメンテナンスモードでユーザーは、最初のインストールで選択した機能の変更、インストール済みの機能の修復、およびアプリケーション全体を削除することが可能です。

アプリケーションのアンインストール

タスク ***チュートリアルアプリケーションをアンインストールするには、以下の手順を実行します。***

[アンインストール] をクリックします。

基本のインストール プロジェクトを作成したあと、[インストール デザイナー] タブ (InstallShield ウィンドウの上部) をクリックして、チュートリアルの 次のステップ で説明されているようにインストールを拡張、または微調整します。

IDE で作業する

プロジェクトの作成後は、*IDE* とも呼ばれる InstallShield インストール開発環境でプロジェクトのプロパティを設定します。IDE は機能カテゴリー別に並べられているので、プロジェクトに情報を簡単に追加または編集できます。ここから先のチュートリアルでは IDE ビューのいくつかを紹介します。

メモ: *IDE で表示されるビューは、作成するプロジェクトの種類によって異なります。*

本ステップを一通り完了すると、以下の項目がマスターできます。

- プログラムの機能のプロパティの設定法。
- コンポーネントおよびファイル リンクの作成法。

機能のプロパティを設定する

まず最初に行うのは、プロジェクト アシスタントで作成した機能に対する追加プロパティの設定で、これには機能の表示名や説明などがあります。機能プロパティを編集するには、IDE の [機能] ビューへ移動します。

タスク ***このチュートリアルの機能のプロパティを設定するには、以下の手順を実行します。***

1. **[機能]** ビューを開きます。**[機能]** ビューは、**[ビューリスト]** の **[編成]** セクションにあります 。
2. [**機能**] ビューで、Tutorial_Files 機能を選択して機能のプロパティグリッドを右側に表示します。
3. **"説明"** フィールドに次のテキストを入力します:
4. **New_Feature** 機能を選択してそのプロパティ グリッドを表示します。
5. **"説明"** フィールドに次のテキストを入力します:

表示名と説明を入力すると、IDE はそれぞれの値を表すための文字列テーブルエントリ ([ID_STRING*n*] と表示) を作成します。

実行時、エンド ユーザーが [カスタム] セットアップ タイプを選択した場合、インストールはユーザーがどの機能をインストールするかを選択できるダイアログを表示します。このダイアログには、ここで設定した機能の表示名および説明が表示されます。

コンポーネントおよびファイルリンクを作成する

[ファイルとフォルダー] ビューの追加ファイルにリンクを作成することができます。このステップでは、Help_Files 機能にファイルを 1 つ追加します。[ファイルとフォルダー] ビューでファイルを追加すると、セットアップベストプラクティス規則に従ってIDE がコンポーネントを作成します。



タスク ***Tutorial.html を Help_Files 機能にある新規コンポーネントへ追加するには、以下の手順を実行します。***

1. **[ファイルとフォルダー]** ビューを開きます。
2. **[インストール先コンピューターのフォルダー] ペイン**で、**[インストール先コンピューター]** アイコンを右クリックし、**[コンポーネントの表示]** が選ばれていることを確認します。
3. ビューの最上部にある機能のリストから、**Help_Files** を選択します。
4. [**インストール先コンピューターのフォルダー**] ペインにあるツリーを広げて [INSTALLDIR] フォルダーを探します。
5. [INSTALLDIR] フォルダーを右クリックして **[新規コンポーネント]** を選択します。コンポーネントに *Help_Component* と名前をつけます。
6. [**ソース コンピューターのフォルダー**] ペインで、TutorialHelp.html が置かれたチュートリアル ファイル ソース フォルダーを選びます。
7. **TutorialHelp.html** アイコンを **[ソース コンピューターのファイル] ペイン**から **インストール先コンピューターのフォルダー**にある **Help_Component** コンポーネントへドラッグします。InstallShield は、このファイルを Help_Files 機能の Help_Component コンポーネントへ追加します。
8. **Help_Component** アイコンをクリックして、**[インストール先コンピューターのファイル] ペイン**にコンポーネントのファイルを表示します。
9. 各コンポーネントにはキーファイルを含まなくてはならないので、TutorialHelp.html ファイルを右クリックして **[キーファイルの設定]** を選択します。

コンポーネントにリンクされるファイルのリストに変更が無い、この種のファイルリンクを「*スタティック ファイル リンク*」と呼びます。ビルド毎に内容に違いがあるディレクトリ(及びそのサブディレクトリ)にリンクするには、「ダイナミック ファイル リンク」を参照してください。

チュートリアルの次のステップでは、インストール プロジェクトのリリースイメージのビルド方法について説明します。

リリースのビルド

インストールをテストする前に、リリースをビルドしてプロジェクト設定の更新を行なう必要があります。リリースには、CD-ROM やネットワーク経由でエンド ユーザーに配布するすべてのファイルが含まれます。

リリース ウィザードを使って設定を構成すると、リリースを最も簡単に作成することができます。リリース ウィザードでは、使用するメディアの種類 (たとえば CD-ROM) や、メディア上のファイルを圧縮するかなど、リリースのプロパティを指定します。



タスク ***リリース ウィザードを開始するには、以下の手順を実行します。***

1. ツールバーにある **[リリース ウィザード]** ボタンをクリックするか、**[ビルド]** メニューから **[リリース ウィザード]** を選択します。
2. **[ようこそ]** パネルで [次へ] をクリックしてリリース設定の定義を開始します。

製品構成とリリースに名前を付ける

[製品構成] パネル

[製品構成] パネルでは、現在の製品構成に対する名前を指定します。製品形状の名前とは、ビルドしたリリースを置く (プロジェクトフォルダー中の) フォルダーの名前です。



タスク ***チュートリアルでは、***

1. *Tutorial* と名づけた新規プロダクト構成を作成します。
2. **[次へ]** をクリックして、リリース名の指定の説明に進んでください。

[リリースの指定] パネル

[**リリースの指定**] パネルでは、リリース名を指定します。リリース名は、ビルドされたリリースを置く (製品構成フォルダー中の) フォルダーの名前として使用されます。



タスク ***チュートリアルでは、***

1. *CDROM* と名づけた新しい製品リリースを作成します。
2. **[次へ]** をクリックします。

フィルター設定と言語の指定

[フィルターの設定] パネル

[フィルターの設定] パネルでは、現在のリリースから除外しておく機能やコンポーネントを指定します。



タスク ***チュートリアルでは、***

デフォルト設定 (フィルタリング無し) のままとし、**[次へ]** をクリックしてください。

セットアップ言語

[セットアップ言語] パネルでは、インストールのユーザー インターフェイスで表示する言語を (プロジェクト言語の中から) 指定し、またユーザー用インストール言語の選択ダイアログ表示の有無を指定します。



タスク *チュートリアルでは、*

デフォルト設定 (ユーザー インターフェイスに日本語を含める) のままとし、**[次へ]**をクリックしてください。

メディアの種類とディスク分割のオプションを選択する

[メディアの種類] パネル

[メディア タイプ] パネルでは、リリースをビルドするメディアの種類を指定します。ここで指定するメディアの種類は、[リリース ウィザード] が作成するディスクイメージフォルダーのサイズをデフォルトします。たとえば CD-ROM 用にリリースをビルドすると、リリース ウィザードがディスクイメージを複数のフォルダーに分割します。それぞれのフォルダーサイズは 650MB 以下です。



タスク *チュートリアルでは、*

[メディアタイプ] メニューから **CD-ROM** を選択します。

[ディスク分割オプション] パネル

[ディスクス分割オプション] パネルでは、複数のディスクイメージが必要な場合に、プログラム ファイルをどのように配置するかを指定します。[カスタム] 分割タイプでは、特定の機能のファイルをどのディスク イメージ上に置くかを指定できます。



タスク *チュートリアルでは、*

[自動] を選択し、各機能のファイルを置くディスクイメージを**リリース ウィザード**が決定します。

詳細については、「インストールを複数のディスクまたは CD に分割する」を参照してください。

圧縮の設定とセットアップランチャーのオプションを指定する

[リリース構成] パネル

[リリース構成] パネルでは、インストール プロジェクト内のファイルの圧縮について、すべて圧縮する、全く圧縮しない、一部のファイルのみを圧縮する、のどれかを指定することができます。



タスク *チュートリアルでは、*

[すべてのファイルを圧縮] を選択します。

[セットアップランチャ] パネル

[セットアップランチャー] パネルでは、Setup.exe セットアップ起動ファイルを作成するか、インストールに Windows Installer 3.1 エンジンを含めるかどうかを指定します。Windows Installer エンジンのインストールは、古いバージョンの Windows Installer が搭載されたターゲット システムで必要です。



タスク ***チュートリアルでは、***

デフォルトの設定をそのままにします。

このステップでは、すべてデフォルトの設定にしてください。

デジタル署名とパスワード保護を追加する

[デジタル署名] パネル

[デジタル署名] パネルで、アプリケーションにデジタル署名を付加することができます。アプリケーションにデジタル署名をすることで、アプリケーション内のコードが発表以来変更または破損していないことをエンドユーザーに対して保証することができます。



タスク ***チュートリアルでは、***

デフォルトの設定をそのままにします(デジタル署名を行わない)。

[パスワードと著作権] パネル

[パスワードと著作権] パネルでは、インストール プロジェクトのパスワード保護をアクティブにして、アプリケーションの著作権の特定の情報を指定することができます。



タスク ***チュートリアルでは、***

デフォルトの設定をそのままにします(パスワード保護または著作権情報を含みません)。

.NET Framework のサポートを含める/詳細設定の選択

.NET Framework パネル

[.NET Framework] パネルで、リリースに .NET Framework サポートを含むかどうかを指定します。



タスク ***チュートリアルでは、***

デフォルトの設定をそのままにします(.NET Framework を含みません)。

[詳細設定] パネル

[詳細設定] パネルでは、圧縮のレベルや、SMS 配布用 PDF ファイルの作成など、現在のリリースに対する追加設定を行います。

タスク ***このチュートリアルでは、次のチェックボックスを選択します。***

- 長いファイル名を使用する
- サイズを最適化する
- Autorun.inf を作成する ー これは CD-ROM イメージ用の AutoPlay を可能にするファイルを生成します。

その他の設定については、[詳細設定] パネルの [ヘルプ] ボタンをクリックしてください。

設定を確認する

概要

[概要] パネルでは、現在のリリースに関するリリース ウィザードの設定をすべて表示します。

タスク ***設定が正しい場合***

1. **[リリースをビルドする]** チェックボックスを選択します。
2. **[完了]** をクリックしてリリースをビルドします。

進行中のビルドに関するステータスメッセージは、アウトプットウインドウに表示されます。ビルドが完了すると、CD にコピーするファイルは、下記のディレクトリに置かれます。

<ProjectFolder>¥Tutorial¥cdrom¥DiskImages¥DISK1.

[リリース] ビューの [イベント] タブを使って、InstallShield で、ビルドされたディスク イメージを別のディレクトリにコピーすることができます。

プロジェクトの再ビルド

チュートリアル後半のステップで行うプロジェクトに対する変更が終了したら、最新リリースの再ビルドを行います。この処理は、[ビルド] ツールバーボタン をクリックするか、[ビルド] メニューから [ビルド] を選択するか、または F7 を押して実行します。

チュートリアルの次のステップでは、インストールのショートカットとレジストリデータの作成方法について説明します。

インストールのトラブルシューティング

インストールを実行してもファイルがインストールされなかった場合は、プロジェクトについて下記の点をチェックしてください。

- INSTALLDIR が適切な値に設定されていることを確認します。これは、[一般情報] ビューで設定されます。

  本チュートリアルの推奨値は、[ProgramFilesFolder]TutorialCo¥TutorialApp です。
- 機能にコンポーネントとファイルが関連付けられていることを確認します。
- インストールに対して何らかの変更を行った場合は、[ビルド] ボタンをクリックするか F7 キーを押して、プロジェクトの再ビルドが必要です。

ステップ 2: ショートカットとレジストリ データ

このステップでは、次の処理を行う場合の IDE 利用法を説明します。

- プログラムショートカットを作成する
- レジストリ情報の作成

その他のシステムデータの作成も、本ステップでの手順と同様に進めることができます。詳細については、下記にリストアップしたトピックを参照してください。

ショートカットの作成

[ショートカット] ビューでは、ショートカットの作成と変更を行うことができます。ショートカットのプロパティには、表示名、ターゲット実行可能ファイルおよび引数、表示用アイコンなどがあります。

ショートカットの作成



タスク

このステップでは、ユーザーのスタートメニューにあるプログラムフォルダーに Tutorial App のショートカットを作成します。

1. **[ショートカット]** ビューを開きます。**[ショートカット]** ビューは、**[ビュー リスト]** の **[システム構成]** セクションにあります 。
2. **[プログラム メニュー]** フォルダー アイコンを右クリックし、**[アドバタイズ ショートカットの新規作成]** を選択します。[**コンポーネントの参照**] ダイアログが表示されます。
3. ダイアログで、**[機能]** ドロップダウンメニューから **Tutorial_Files** を選択し、ファイルリストから Tutorial.exe を選択して **[開く]** をクリックしてダイアログを閉じます。
4. ショートカットアイコンの内部名を、*Tutorial* などに変更します。
5. 次のショートカットプロパティを設定します。

テーブル 2・ショートカットのプロパティ

プロパティ	値	コメント
表示名	チュートリアルアプリケーション	長いファイル名をサポートしないターゲット システムに対応するため、IDE は “TUTORI~1|TutorialApp” のような短いファイル名を含む表示を作成します。必要であれば、“TUTORIAL|TutorialApp” のように短いファイル名表示の一部を変更することができます。
説明	チュートリアル アプリケーションを起動する	ツールヒントとして表示されます。

テーブル 2・ショートカットのプロパティ（続き）

プロパティ	値	コメント
アドバタイズ	はい	実行時に、エンドユーザーがこのショートカットを含む製品または機能をアドバタイズすると、ショートカットが作成されますが、コンポーネントのファイルは、エンドユーザーがショートカットを起動するまでインストールされません。
ターゲット	[INSTALLDIR]Tutorial.exe へのアドバタイズ ショートカット	アドバタイズショートカットはコンポーネントのキーファイルへと自動的に設定されます。
アイコン ファイル	<TutorialSource>¥Tutorial.exe	ソースロケーションから Tutorial.exe を探し出し、そこにあるアイコンを選択します。
アイコン インデックス	0	実行可能ファイルに複数のアイコンリソースがある場合、アイコン インデックスは特定のアイコンを識別します。
作業ディレクトリ	[INSTALLDIR]	作業ディレクトリは [名前を付けて保存] および [開く] ダイアログに対するデフォルトのディレクトリに設定します。



ヒント: *エンドユーザーマシンの既存ファイルにショートカットを作成するには、[アドバタイズ] プロパティを [いいえ] に設定して、そのファイルへのパスを入力しますが、可能ならば Windows Installer のディレクトリプロパティを使用してこのファイルへのパスを指定をします。たとえば、ユーザーマシンの Windows または WinNT フォルダーに置かれた Windows Notepad のコピーを起動するには、ショートカット ターゲットに [WindowsFolder]Notepad.exe と入力します。*

レジストリ データを作成する

インストールに求められる機能の 1 つとして、ターゲット システムのレジストリへの情報の書き込みがあります。コンポーネントにレジストリを追加するには、[レジストリ] ビューを利用することができます。



タスク

HKEY_LOCAL_MACHINE¥SOFTWARE¥Tutorial Co¥Tutorial¥1.00.0000 の下に TutorialData というレジストリ値を作成するには、以下の手順を実行します。

1. **[レジストリ]** ビューを開きます。

   [レジストリ] ビューは、**[ビュー リスト]** の **[システム構成]** セクションにあります。

2. ビューの最上部にある **[ビューフィルター]** から、**Tutorial.exe** を選択します。

3. **[インストール先コンピューターのレジストリビュー]** ペインで、HKEY_LOCAL_MACHINE を右クリックして、**[新規作成]** を選択し、**[キー]** をポイントします。

4. キーの名前を *SOFTWARE* に変更します。

5. サブキー *Tutorial Co*、*Tutorial*、と *1.00.0000* に関しても同じ操作を繰り返します。

6. **[インストール先コンピューターのレジストリデータ]** ペインで、右クリックして、**[新しい文字列値]** を選択します。
7. 値を *TutorialData* に変更します。
8. TutorialData 値をダブルクリックして、"値のデータ" フィールドに *[INSTALLDIR]* と入力します。

これで [**レジストリ**] ビューは、次のように表示されます。

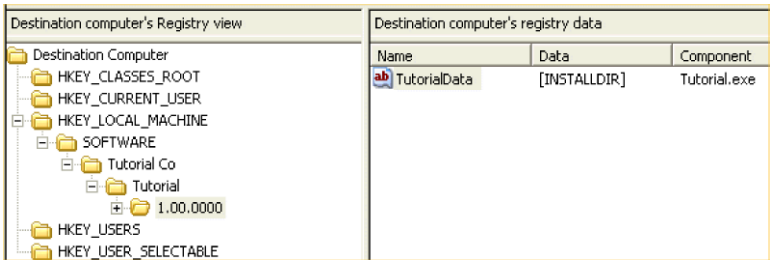


図 3: [レジストリ] ビュー

ヒント: *レジストリに Windows Installer プロパティの値を書き込む場合、その表記は [PropertyName] となります。この例の場合、[INSTALLDIR] というレジストリ値を作成すると、レジストリには INSTALLDIR という値が書き込まれます。*

インストール実行時に Tutorial.exe コンポーネントを含むセットアップの種類や機能のコレクションをユーザーが選択すると、ターゲット システム上にレジストリデータが作成されます。

ショートカットが作成されたことを確認する

タスク ***インストールがショートカットを作成したことを検証するには、以下の手順を実行します。***

1. **[ビルド]** ツールバーボタンをクリックするか、**F7** を押してプロジェクトを再ビルドします。
2. **[実行]** ボタンをクリックするか、CTRL+F5 を押してプロジェクトを実行します（最初にシステムからプログラムの既存バージョンをすべて削除します）。これでチュートリアルアプリケーションのショートカットが、**[スタート]** メニューの **[プログラム]** フォルダーに作成されているはずです。

レジストリデータが作成されたことを確認する

タスク ***インストールがレジストリデータを作成したことを検証するには、以下の手順を実行します。***

1. ショートカットから Tutorial App を起動します。
2. **Tutorial** メニューから **Verify Registry Data** を選択します。レジストリデータが作成されている場合、*INSTALLDIR* の値を表示するメッセージ ボックスが開きます。

チュートリアルの次のステップでは、COM サーバー（自己登録ファイル）の登録方法について説明します。

ステップ 3: COM サーバーの登録

ほとんどのファイルに関して、インストールが果たすべき唯一の役割は、ソース メディアからターゲット システムにファイルをコピーすることです。これに対し一部のファイルについては、ターゲット システムへのファイル登録作業をインストーラーが実行する必要があります。こうした例外的な処理が必要なファイルの 1 つが *COM サーバー*で、これは*自己登録ファイル*や *ActiveX コントロール*として知られれています。COM サーバーは通常 DLL や OCX の形を取りますが、こうした自動登録ファイルは、これらを使用するアプリケーションや Web ページが認識できるよう、あらかじめターゲット システムのレジストリへ書き込んでおくため、追加情報を必要とします。

COM サーバーコンポーネントの作成

これらのファイルのインストールと登録には、コンポーネントウィザードを利用することができます。コンポーネントウィザードでは、ファイルコピーや登録用の追加ステップを実行するコンポーネントを作成します。本ステップでは、Tutorial.ocx 本体およびこれが使用する HTML ファイルのインストールと登録を行うコンポーネントを作成します。



タスク ***COM サーバーコンポーネントの作成は、次の手順を実行します。***

1. **[ファイルとフォルダー]** ビューを開きます。**[ファイルとフォルダー]** ビューは、**[ビューリスト]** の **[アプリケーション データ]** セクションにあります 。
2. **[ファイルとフォルダー]** ビューの上部にある **[ビュー フィルター]** リストから **Tutorial_Files** 機能を選択します。
3. **[インストール先コンピューターのフォルダー]** ペインで、**[INSTALLDIR]** フォルダーを右クリックし、**[コンポーネントウイザードの起動]** を選択します。
4. **コンポーネント ウイザード**の **[ようこそ]** パネルで、[タイプを選択してコンポーネントを定義する] を選択し、**[次へ]** をクリックします。
5. **[コンポーネントのタイプ]** パネルで **COM サーバー** アイコンを選択し、**"コンポーネント名"** フィールドに *Tutorial.ocx* を入力してから **[次へ]** をクリックします。
6. **[COM サーバー – インストール先]** パネルで、インストール先が [INSTALLDIR] となっていることを確認します。
7. **[COM サーバー ファイル]** パネルで、**"COM サーバー ファイル"** フィールドの隣にある参照ボタンをクリックし、チュートリアル ファイル ソース ディレクトリの Tutorial.ocx を選択します。**[次へ]** をクリックします。
8. コンポーネントウイザード による COM 情報の抽出が完了したら、COM 情報を確認し、**[完了]** をクリックしてコンポーネントを作成します。

次のステップでは、作成したコンポーネントに HTML ファイルを追加します。

タスク ***コンポーネントに HTML ファイルを追加するには、以下の手順を実行します。***

1. **[ファイルとフォルダー]** ビューの **[インストール先コンピューターのフォルダー]** ペインで、新規に作成した Tutorial.ocx コンポーネントを選択します。
2. **[ソース コンピューターのファイル]** ペインから TutorialCtrl.html というファイルを、**[インストール先コンピューターのファイル] ペイン**へドラッグします。
3. Tutorial.ocx がコンポーネントのキーファイルとしてマークされていることを確認します。

メモ- *リリースの再ビルドごとにインターフェイスを変更した場合の COM 情報の抽出や、ファイルの自己登録関数の呼び出しなど、自己登録ファイルの登録に関するその他のオプションについては、「COM サーバーの登録」を参照してください。*

COM サーバーが登録されていることを確認する

タスク ***リリースを再ビルドし (F7 キーを押す) インストールを実行 (CTRL+F5 を押す) をした後、COM サーバーが正常に登録されたかを次の手順で確認できます。***

1. [**プログラム**] メニューのショートカットか、アイコンをダブルクリックして、Tutorial App を起動します。
2. **Tutorial** メニューから **COM Server Test** を選択します。
3. COM サーバーが正常に登録されていれば、「登録成功」のメッセージを示す HTML ページが表示されます。

その他にもコンポーネント ウイザードでは、フォントや Windows NT サービスのインストールおよび設定を行うコンポーネントも作成できます。

チュートリアルの次のステップでは、ファイルの条件付きインストールについて説明します。

ステップ 4: 条件とプロパティ

このステップでは、ターゲット システムにデータを条件付きでインストールする方法について解説します。

オペレーティング システム条件

インストール通常求められる機能は、特定の条件が満たされた場合にのみ必要なファイルをシステムにインストールすることです。たとえばある種のファイルは、特定のオペレーティング システムや言語にのみ必要であったり、ユーザー必要な権限を持つ場合に限りインストールできる、ということがあります。

コンポーネント (およびそのファイルとその他のデータ) を特定のオペレーティング システムにのみインストールさせるには、コンポーネントの "オペレーティング システム" プロパティを使います。コンポーネントのプロパティの変更は、[セットアップのデザイン] ビューを開いて該当する機能のアイコンを展開し、必要なコンポーネントを選択して行います。

タスク

Windows 7 またはそれ以降を実行するシステム上でのみインストールが可能なコンポーネントを作成するには、以下の手順を実行します。

1. **[セットアップのデザイン]** ビューを開きます。**[セットアップのデザイン]** ビューは、**[ビューリスト]** の **[編成]** セクションにあります 。
2. **Help_Files** 機能を右クリックして、**[新しいコンポーネント]** を選択します。
3. コンポーネントの名前を *Windows_7_Files* に変更します。
4. Windows_7_Files コンポーネントを開き、コンポーネントの [**ファイル**] アイコンをクリックし、[ファイル] ペイン内で右クリックしてファイルを参照してチュートリアル ファイル ソース フォルダー から ReadmeNT.txt ファイルを追加します。
5. .txt ファイルを右クリックして **[キーファイルの設定]** を選択します。
6. Windows_7_Files コンポーネントをクリックして、コンポーネントのプロパティ グリッドを表示します。
7. コンポーネントの [条件] プロパティを選択して参照ボタンをクリックし、**[条件ビルダー]** ダイアログ を起動します。
8. 次の条件を作成します: *VersionNT>=601*条件の作成に関する詳細については、「条件ステートメントのビルド」を参照してください。
9. **[OK]** をクリックして **[条件ビルダー]** ダイアログを閉じ、条件を追加します。

(F7 を押して) 再ビルドの後、(CTRL+F5 を押して) インストールを実行する と、ターゲット システムが Windows 7 以降を実行している場合に限りコンポーネントに含まれるファイルおよびその他のデータがインストールされるようになります。

Windows Installer 条件

Windows Installer サービスは、作成したインストールおよびユーザーのオペレーティング システムに関して、一部のグローバル情報をプロパティとして記録します。こうしたプロパティには、MSI データベースの Property テーブルに用意されているものもあれば、ユーザーによるインストールの起動時に、Windows Installer エンジンが作成して設定するものもあります。

条件の判定でよく使用されるのは、次のプロパティです:

- AdminUser、これはインストールを実行するユーザーが管理者権限を持つ場合に設定されます。
- VersionNT、これはユーザーが使用しているオペレーティング システム バージョンを数値で示します。
- PhysicalMemory、これはユーザーが使用しているシステムの RAM をメガバイト単位で示します。

Windows Installer 条件は論理ステートメントで、プロパティ値を定数値と比較したり、プロパティがついているかの判定を行います。たとえば、Windows Installer は ScreenX および ScreenY というプロパティを定義しますが、これらはユーザーが使用するモニターの解像度を、ピクセル単位で示します。解像度が 800 × 600 以上であるかを調べる Windows Installer 条件の場合は、“(ScreenX>=800) And (ScreenY>=600)” と記述します。

定義済みのプロパティは、それだけで判定条件として使用できます。たとえば、AdminUser プロパティはユーザーが管理者権限を持つ場合に限り設定されるので、ユーザーが管理者権限を持っているかどうかを判断する条件は、単に ″AdminUser″ です。



タスク ***ユーザーが管理者権限を持つ場合にのみインストールされるコンポーネントを作成するには、次の手順を実行します。***

1. **Help_Files** 機能を右クリックして、**[新しいコンポーネント]** を選択します。
2. コンポーネントの名前を *Admin_Component* に変更します。
3. **Admin_Component** コンポーネントを開いて **[ファイル]** アイコンをクリックします。
4. チュートリアル ファイル ソース フォルダーから AdminOnly.txt ファイルを追加し、これをコンポーネントのキーファイルとして設定します。
5. コンポーネントの **"条件"** プロパティで参照ボタンをクリックして、**[条件ビルダー]** ダイアログ を表示します。
6. **[条件ビルダー]** ダイアログ ボックスで、**"条件"** フィールドに *AdminUser* と入力します。
7. **[OK]** をクリックします。

完成後のインストーラーを実行すると、このコンポーネントのデータは、ユーザーが管理者権限を持つ場合にのみインストールされます。

チュートリアルの次のステップでは、インストールのユーザー インターフェイスの変更方法について説明します。

ステップ 5: エンドユーザー インターフェイスを変更する

このステップでは、インストールのユーザー インターフェイスを変更する 3 種類の方法について説明します。

- インストール中に表示するダイアログを指定する。
- ダイアログ エディターを使用して、ダイアログのレイアウトとプロパティを変更する。

新しいダイアログの追加

基本の MSI プロジェクトは、インストールのユーザー インターフェイスで表示可能な多数のダイアログ ボックスが用意されています。このチュートリアルの「インストールを実行する」トピックでは、プロジェクト アシスタントの [インストール アーキテクチャ] ページで行った選択に基づいてインストールが表示するダイアログについて説明します。



タスク ***新しいダイアログを作成するには、以下の手順を実行します。***

1. **[ダイアログ]** ビューを開きます。**[ダイアログ]** ビューは、**[ビューリスト]** の **[ユーザー インターフェイス]** セクションにあります 。
2. **[すべてのダイアログ]** エクスプローラーを右クリックして、**[ダイアログの新規作成]** を選択します。**ダイアログ ウィザード**が開きます。**[次へ]** をクリックして、**[ようこそ]** パネルをとばします。
3. **[ダイアログ テンプレート]** パネルで、**[内部ウィザード パネル]** をクリックして、**[このダイアログを [ユーザー インターフェイス] シーケンスに挿入する]** チェック ボックスを選択します。

4. **[ユーザー インターフェイス]** パネルで、**[ユーザー シーケンス]** リストから **[インストール]** を選択します。ダイアログのリストから、**InstallWelcome** を選択します。これらの選択に基づき、InstallShield は新しいダイアログをシーケンスの **InstallWelcome** ダイアログの直後に挿入します。
5. **[ダイアログの位置と条件]** パネルでは、デフォルト設定をそのままにして **[完了]** をクリックします。新しいダイアログが **[ダイアログ]** リストに表示されます。
6. ダイアログを右クリックして **[名前の変更]** を選択します。**WelcomeBitmap** ダイアログの名前を変更します。

同様の手順で、インストール プログラムのユーザー インターフェイスにダイアログを追加することができます。

ダイアログ エディターでダイアログのレイアウトを変更する

ダイアログ エディターを使って、インストールで表示されるダイアログの外観を修正することができます。

タスク ***このステップでは、作成したばかりの WelcomeBitmap ダイアログの修正を行います。***

1. まず、(Microsoft Paint のようなプログラムを使用して) 300 x 150 サイズのビットマップを作成します。
2. **[ダイアログ]** ビューを開きます。
3. WelcomeBitmap ダイアログのノードを開きます。**[日本語]** をクリックして、**ダイアログ エディター**を開きます。
4. ダイアログ上部にある [**ダイアログ太字タイトル**] テキスト ボックスをクリックします。**"テキスト"** フィールドに*「ようこそビットマップ」*と入力します。これでダイアログのメインタイトルが変更されます。
5. ダイアログ上部にある [**ダイアログ標準説明**] テキストボックスをクリックします。**"テキスト"** フィールドに*「ようこそビットマップを表示する」*と入力します。これでダイアログの説明が変更されます。
6. **"ダイアログ コントロール"** ツールバーにある **[ビットマップ]** ボタンをクリックして、カーソルを利用してダイアログ上にボックスをドラッグします。高さ 150、幅 300 に設定します。
7. **"ファイル"** フィールドで、ステップ 1 で作成したビットマップ ファイルを参照します。

プロジェクトを再ビルド (F7 キーを押す) してから実行すると (CTRL+F5 キーを押す)、Welcome Bitmap ダイアログは InstallWelcome ダイアログの後に表示されます。

ヒント: *CTRL+M を押すと、[ダイアログ エディター] ビューが最大化します。*

グローバリゼーション チュートリアル

グローバリゼーションチュートリアルでは、InstallShield が提供するグローバルインストール パッケージ作成のためのツールとオプションを紹介します。グローバルインストールは、多くの異なる言語で実行できるセットアップです。インストールのビルド方法に応じて、1 つのパッケージにすべての言語を含めてユーザーに言語の選択オプションを提供する、またはターゲットとするそれぞれの言語ごとに個別のインストールパッケージをビルドすることもできます。このチュートリアルでは、すべての言語を含むインストールの作成過程を説明します。



エディション: *このチュートリアルを完了するには、開発システム上で InstallShield Premier Edition のインストールが必要です。*

InstallShield チュートリアルをまだご利用になったことがない方は、インストールパッケージの作成方法を理解するために 基本の MSI チュートリアルを是非一度お試しいただくことをお勧めいたします。基本の MSI チュートリアルを完了すると、別の言語を追加するのに最適な MSI インストール プロジェクトを作成することができます。基本の MSI チュートリアルの完全プロジェクト ファイルは、InstallShield Program Files フォルダーの Samples サブフォルダーに製品と共にインストールされます。デフォルト保存先は次の場所です:

C:¥Program Files¥InstallShield¥2022¥Samples¥WindowsInstaller¥Tutorial Project¥Tutorial



プロジェクト: *チュートリアル内の情報はほとんどすべて InstallScript インストール プロジェクトにも適用されます。異なる点についてはチュートリアル テキストに記述してあります。*

プロジェクトファイルを開く

チュートリアルでは、InstallShield を使ってインストールするサンプル プロジェクトファイルとして Othello.ism を利用します。



タスク ***チュートリアルを開始するには、以下の手順に従います:***

Othello.ism (InstallShield Program Files フォルダーの Samples サブフォルダーにあります) を開きます。デフォルト保存先は次の場所です:

C:¥Program Files¥InstallShield¥2022¥Samples¥WindowsInstaller¥Basic Installation Project



プロジェクト: *Othello.ism は基本の MSI インストール プロジェクトです。このチュートリアルに記述されている情報のほとんどは、InstallScript インストール プロジェクトにも適用されます。異なる点についてはチュートリアルテキストに記述してあります。*

ターゲット言語の選択

グローバルインストールを作成するには、まずターゲットとする言語を選択します。このチュートリアルでは、インストール プロジェクトに 2 つの言語 (ドイツ語とポーランド語) を追加します。



タスク ***プロジェクトに言語を追加するには、以下の手順を実行します。***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"セットアップ言語"** 設定で、省略記号ボタン (...) をクリックします。**[セットアップ言語]** ダイアログ ボックスが開きます。
3. **[英語(米国)]**、**[ドイツ語]**、および **[ポーランド語]** チェック ボックスを選択します。

プロジェクトに、英語、ドイツ語、およびポーランド語の文字列エントリが追加されます。各文字列エントリは、言語非依存の識別子と対応する言語固有の値で構成されます。文字列エントリには、翻訳済みのビルトイン ユーザー インターフェイス文字列リソースが含まれます。すべての文字列エントリを参照するには、[文字列エディター] ビューを使用します。([ユーザー インターフェイス] の下のビュー リストにある [文字列エディター] をクリックします。)

デフォルト言語に新しい文字列エントリを追加するたびに、プロジェクトに含まれているすべての別の言語用に、対応する文字列エントリが作成されます。

言語固有の文字列エントリを編集する

次のステップは、インストール プロジェクトの文字列エントリの編集です。文字列エントリを、3 つの異なる設定 (機能の表示名、ショートカットの表示名、およびショートカットの説明) で編集します。(ショートカットの説明は、チュートリアルの次のステップで作成します。)

機能の表示名



タスク ***機能の表示名を提供するには、以下の手順を実行します。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. **[セットアップのデザイン]** エクスプローラーで、**ProgramFiles** をクリックします。
3. 右側のペインに表示される **"表示名"** 設定に「**プログラム ファイル**」が既に存在しない場合は、それを入力します。

[セットアップのデザイン] ビュー内の別の場所をクリックすると、入力したテキストの始まりに中括弧 ({}) に囲まれた文字列 ID が追加されていることが確認できます。"表示名" 設定をクリックしたときに表示される省略記号ボタン (...) をクリックすると、プロジェクトに含まれるすべての文字列エントリを参照できます。

ショートカットの表示名



タスク ***ショートカットの表示名を追加するには、以下の手順を実行します。***

1. **[セットアップのデザイン]** エクスプローラーで、**ProgramFiles** 機能を展開して、**Program_Executables** コンポーネントを展開します。

   Program_Executables コンポーネントには、Othello.exe ファイルへショートカットが含まれています。
2. **Program_Executables** コンポーネントの下にある **[ショートカット]** アイコンをクリックします。

3. **[ショートカット]** エクスプローラーで、**Othello** ショートカットをクリックします。右側のペインにショートカットの設定が表示されます。

   現在、**"表示名"** 設定は、**Othello** に設定されています。この値の始まりには、中括弧 ({}) で囲まれた文字列 ID が追加されています。

4. 表示名を変更するには、新しい名前を **"表示名"** 設定に入力します。

文字列エントリの作成

テキスト文字列の多くはプロジェクトの複数の場所で使用される可能性があるため、これらの文字列の各インスタンスをプロジェクトに保存することは非効率的です。その代わり、文字列を一度作成して必要な任意の場所でその文字列を使用することができます。プロジェクトのローカライズ処理を簡素化するために、インストール処理中、実行時に表示されるすべてのテキスト文字列は、統合された 1 つのビュー、[文字列エディター] ビューに表示されます。このビューを使って新しい文字列エントリを作成できます。

"ショートカットの説明" 設定には、[文字列エディター] ビューを使って新しい文字列エントリを入力します。次に、この文字列をショートカットの説明に関連付けます。



タスク

新しい文字列エントリを作成して、それを "ショートカットの説明" 設定で使用するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. 以下のいずれかを実行します。
   - **[新規]** ボタンをクリックします。
   - INSERT キーを押す。

   [文字列エントリ] ダイアログ ボックスが開きます。
3. **[文字列 ID]** ボックスで、次のように入力します:

   MYSTRING
4. **[値]** ボックスで、次の文字列を入力します:

   これは Othello ショートカットの説明です。
5. **[コメント]** ボックスに、オプションとして、文字列エントリについて内部で使用するメモを指定します。コメントは実行時には表示されません。
6. **[OK]** をクリックします。[文字列エディター] ビューで各言語ごと (英語、ドイツ語、ポーランド語) に新しい行が追加されます。
7. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
8. **[ショートカット]** エクスプローラーで、**Othello** ショートカットをクリックします。右側のペインにショートカットの設定が表示されます。
9. **"説明"** 設定で、省略記号 (...) をクリックします。**[文字列の選択]** ダイアログ ボックスが開きます。
10. **MYSTRING** 行を選択してから、**[OK]** をクリックします。

新しい文字列が**"説明"** 設定の値として入力されます。

言語別ファイルとコンポーネントを含める

次のステップでは、インストールの中に言語固有のファイルおよびコンポーネントを含むグローバルインストールを作成します。たとえば、作成したプログラムファイルの多くが特定言語に依存していない場合でも、ヘルプファイルおよびランタイム文字列は共に言語固有の情報を含みます。この実習の目的は、インストールに 3 つの新規コンポーネントを追加することです。各コンポーネントには、サポートするすべての言語にローカライズされた Readame ファイルが含まれます。



タスク ***プロジェクトにコンポーネントを追加するには、以下の手順を実行します。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. **[セットアップのデザイン]** エクスプローラーで、**ProgramFiles** という名前の機能を右クリックしてから、**[新しいコンポーネント]** をクリックします。
3. コンポーネントの名前として *English_Readme* と入力します。
4. この手順を 2 回繰り返します。それらのコンポーネントに *Polish_Readme* および *German_Readme* と名前を付けます。

InstallShield には、翻訳済みのシンプルな Readme ファイルが添付されています。これらのファイルは、InstallShield Program Files フォルダーの Samples サブフォルダーにあります。

InstallShield Program Files フォルダー¥Samples¥WindowsInstaller¥Basic Installation Project¥Data Files¥Readme



タスク ***English_Readme コンポーネントにファイルを追加するには、以下の手順を実行します:***

1. 新しい **English_Readme** コンポーネントを展開して、**English_Readme** の下にある **[ファイル]** アイコンをクリックします。
2. **[ファイル]** リストを右クリックして、**[追加]** をクリックします。
3. English.txt ファイルを参照して、**[開く]** をクリックします。

上記のステップをドイツ語およびポーランド語のコンポーネントに対して繰り返し、それぞれ Deutsch.txt および Polski.txt を追加します。

コンポーネント インストール条件を指定する

言語固有のコンポーネントを作成した後は、論理式を指定することによって作成したどのコンポーネントをインストールするかをインストーラーに通知する必要があります。コンポーネント条件を指定して、ターゲット システムのデフォルト言語を判別し、該当するファイルをインストールできます。作成した 3 つの言語固有のコンポーネントのそれぞれに対して条件が必要です。この条件が True と評価された場合に、コンポーネントがインストールされます。



タスク ***コンポーネント条件を作成するには、以下の手順を実行します。***

1. German_Readme コンポーネントをクリックします。
2. 右側のペインで、**"条件"** 設定をクリックしてから、この設定の省略記号ボタン (...) をクリックします。**[条件ビルダー]** ダイアログ ボックスが開きます。
3. **[プロパティ]** リストで、**SystemLanguageID** を選択してから、**[追加]** ボタンをクリックします。
4. **[演算子]** リストで、等号 (=) を選択してから、**[追加]** ボタンをクリックします。

   [条件] ボックスには、これまでに選択した内容を反映する **SystemLanguageID =** が含まれています。
5. 次に、インストールの実行時に調べる値を与える必要があります。現在ドイツ語コンポーネントを編集中のため、等号の後に **1031** と入力します。1031 はドイツ語の言語 ID です。コンポーネントは、この式が真の場合 (つまり、ターゲットシステムの言語がドイツ語の場合) にのみインストールされるため、ドイツ語で実行されていないマシンにはコンポーネントはインストールされません。

上記のステップにしたがって、同様に Polish_Readme コンポーネントに条件を追加します。言語値として、1031 ではなく、ポーランド語の言語 ID である 1045 を使用します。

デフォルトの言語として、1 つの言語を選択する必要があります。この例では、英語がデフォルトです。したがって、English_Readme コンポーネントに使用する条件は他の 2 つとは異なります。English_Readme コンポーネントの条件は、以下のとおりです:

SystemLanguageID◇1045 AND SystemLanguageID◇1031

この論理により、ターゲットマシンの言語がドイツ語またはポーランド語のどちらでもない場合に、English_Readme コンポーネントがインストールされます。



プロジェクト: *InstallScript と InstallScript オブジェクト プロジェクトで実行時にインストールされる言語依存コンポーネントを指定する方法については、「言語基づいてコンポーネントをインストールする」を参照してください。*

文字列の翻訳

インストール プロジェクトをビルドする前に、機能の表示名、ショートカットの表示名、およびショートカットの説明に入力した英語の文字列を翻訳する必要があります。このチュートリアルでは、翻訳済みの文字列を用意しました。ここでは、ドイツ語とポーランド語文字列エントリに正しいテキストを入力するだけです。[文字列エディター] ビューを使用できます。

ドイツ語文字列値

[文字列エディター] ビューを開きます。次の各ドイツ語の文字列 ID を見つけます。[検索グリッド] ボックスに文字列 ID を入力して、各文字列を簡単に見つけることができます。代りに、ID 列ヘッダーをクリックして、すべての文字列エントリを ID 順に並べ替えることもできます。3 つの文字列エントリのドイツ語の値を、以下のように更新します:

テーブル 1 ・ ドイツ語文字列エントリ

Identifier	値
FEAT_DISPLAYNAME	Programmdateien
SHORTCUT_DISPLAYNAME	Othello-Verknuepfung
MYSTRING	Dies ist die Beschreibung der Verknuepfung fuer Othello.

ポーランド語文字列値

3 つの文字列エントリのポーランド語の値を更新します:

テーブル 2 ・ ポーランド語文字列エントリ

Identifier	値
FEAT_DISPLAYNAME	Pliki Programu
SHORTCUT_DISPLAYNAME	Skrt do Othello
MYSTRING	Opis skrtu Othello.

通常は、翻訳用に文字列エントリをエクスポートします。詳細については、「文字列エントリの翻訳」を参照してください。このチュートリアルでは [文字列エディター] ビューで文字列エントリを編集した方が簡単です。

インストールのビルド

インストール プロジェクトは完全にグローバル化されており、直ぐにテストできる状態です。ただしインストールをテストする前にビルドしなくてはなりません。



タスク ***インストールをビルドするには、次の手順に従います:***

1. ツールバーの **[リリース ウィザード]** ボタンをクリックします。
2. 最初の 2 つのパネルでは、製品構成とリリース名を指定します。
3. **[セットアップ言語] パネル**を除き、他のすべてのパネルはデフォルト設定のままにします。
4. [**セットアップ言語**] パネルでは、インストールに含む言語を選択することができます。[一般情報] ビューで指定した言語 (英語、ポーランド語、およびドイツ語) のみが、使用可能言語のリストに表示されます。各言語のとなりにあるチェック ボックスを選択します。

5. また、必ず **[セットアップ言語ダイアログを表示する]** チェック ボックスも選択してください。このダイアログは、インストールを実行する言語をユーザーが選択できるようにします。
6. 残りのウィザードパネルはデフォルトの設定のままにします。
7. **[概要]** パネルに到達したら、**[リリースをビルドする]** チェック ボックスが選択されていることを確認して、**[完了]** ボタンをクリックします。
8. InstallShield がリリースをビルドします。

インストールの実行

タスク ***インストールを実行するには、以下の手順を実行します。***

1. ツールバーの **[セットアップの実行]** ボタンをクリックします。[**セットアップ言語の選択**] ダイアログが表示されます。
2. インストールを実行するには、**[ドイツ語 (標準)]** を選択して **[OK]** をクリックします。これ以降は、すべてのダイアログがドイツ語で表示されます。

メモ- *インストールが特定の言語で実行された後、Windows Installer はこの情報をキャッシュして、インストールを常に同じ言語で実行します。*

インストール ウィザードの処理が進むにつれ、いくつかのボタン サイズが適切でないことが確認できます。この問題の解決策として、ダイアログ エディター を開いてテキストが正しく収まるようにコントロールのサイズを変更して簡単に修正することができます。

[カスタム セットアップ] パネル (ドイツ語では Angepasstes Setup と呼びます) では、機能名が *Programmdateien* となります。ローカライズした文字列エントリが、インストールの一部となっています。

インストールのテスト

最後のステップは、作成したグローバライズされたインストールのテストです。

タスク ***インストールをテストするには、以下の手順を実行します。***

1. **[スタート]** メニューを開き、**[プログラム]** を選択します。Othello Verknpfung として Othello ショートカットが表示されます。ショートカットの説明もドイツ語で表示されていることを確認できます。
2. Othello のインストールディレクトリに移動します。ディレクトリは <Program Files フォルダー>¥Shakespeare Inc¥Othello になります。インストールされた readme ファイルの名前は Deutsch.txt です。

4

インストールの作成

コンピューターにアプリケーションをインストールしたことがある方は、実行中のインストールを既にエンドユーザーの視点でご覧になっています。インストールの主要な役割は、ソース メディアからローカル ドライブへファイルを転送することです。インストールはまた、ユーザー インターフェイスを表示してエンドユーザーの選択を取得し、ターゲット システムを構成し (たとえば、必要なレジストリ エントリやショートカットの作成)、またインストール済みアプリケーションの変更またはアンインストールを行います。インストールの作成は次のタスクの一部またはすべての処理を含みます。

インストール情報の指定

[一般情報] ビューで入力した基本情報は、インストールの様々な場所で利用されます。たとえば、製品名はアプリケーション情報 レジストリ キーの作成に利用されます。

ファイルの編成と転送

ファイル転送は CD または DVD のようなソース メディアからファイルをエンドユーザーのマシンにあるローカル ドライブへコピーします。エンドユーザーがセットアップの種類 (InstallScript または InstallScript MSI インストールの場合) または機能を選択して希望した構成に基づき、ファイルの一部またはすべてがローカルディスクに転送されます。

インストールするファイルをセットアップの種類および機能に編成して、エンドユーザーが最も適切なファイルを選択できるようにします。各機能について、たとえば同じターゲットフォルダーにインストールするファイルなど、ファイルをその種類と目的に応じてコンポーネントに編成します。

ターゲット システムの構成

ファイルをインストールするだけでなく、ショートカットやプログラムフォルダーの作成、レジストリの変更、初期化ファイル (.ini ファイル) データの変更、ODBC リソースの構成、環境変数の変更、XML ファイルの変更、テキスト ファイルの変更、タスクのスケジュール、および Windows サービスのインストールと制御が可能です。

インストール動作のカスタマイズ

InstallShield では、幅広いカスタマイズ オプションが提供されています。InstallScript インストールは InstallShield のシンプルかつパワフルな InstallScript プログラム言語によって実行が進められます。InstallScript では、ビルトイン関数の他に DLL および Windows API 関数の呼び出しも可能で、子インストールやその他のアプリケーションを

インストールから起動することも可能です。Windows Installer ベースのインストールでは、カスタム アクションを使った InstallScript、VBScript、または JavaScript コードの実行、DLL 関数の呼び出し、実行可能ファイルの実行、マネージ アセンブリ内にあるマネージ メソッドの呼び出し、プロパティやディレクトリの設定、エラーの起動とインストールの中止、他のインストール パッケージの実行が可能です。

エンドユーザー インターフェイスの定義

インストールのエンドユーザー インターフェイスは、エンドユーザーに対して情報およびインストール構成オプションを提供します。エンド ーザーはユーザー インターフェイスを通して、製品の一部をインストールするか、一部のファイルをソース メディアに残すか、使用許諾契約を表示するか、またはインストールを正しく構成するための情報をインストーラーに提供するかを選択できます。

ユーザー インターフェイスはインストールのニーズに合わせたカスタマイズが可能です。たとえば、ソフトウェアを不正な使用から保護するために、インストールの前にシリアル番号の入力をユーザーに求めるように指定できます。ファイル転送中、インストールは新機能や便利なヒントなどの製品情報を提供するビルボードを表示します。ファイル転送処理の進行状況がわかるステータス バーも表示されます。

サーバーの構成

サーバー側インストールには、新しい IIS Web サイトの作成および管理、COM+ アプリケーションおよびコンポーネントの管理、またはサーバー接続および設定による SQL スクリプトの管理および編成が必要な場合があります。

インストールにメンテナンスとアンインストール機能を追加する

アプリケーションのアンインストール、変更、または修復を行うためには、アプリケーションの存在がオペレーティング システムで示されなくてはなりません。これに対応するには、アプリケーションを簡単にメンテナンスまたはアンインストールできるように、インストールでアプリケーションをオペレーティング システムに登録します。

この処理で登録されるほとんどの情報は、コントロール パネルの [プログラムの追加と削除] でエンドユーザーに提供されます。たとえば、テクニカル サポートの連絡先情報、製品更新の情報、製品のバージョン、および製品発行元の情報はこのプロセスで登録されます。

インストールのビルド、テスト、および配布

インストール プロジェクトを作成した後は、インストールのビルド、テストおよび配布が必要です。つまりユーザーへリリースするファイルを作成し、インストールのエラーをテストし、オプションとしてローカル、ネットワークまたは FTP サイトへファイルをコピーします。

まず始めに

InstallShield ヘルプ ライブラリの「まず始めに」は、インストール作成者が InstallShield で新規のインストール プロジェクトを作成するときに役に立つ情報が盛り込まれています。各トピックでは、Windows Installer、Windows ロゴ プログラム、INSTALLDIR、TARGETDIR、およびインストール開発のその他の分野についてのバックグランド情報を読むことができます。

このセクションには、次のトピックが含まれています:

- Windows ロゴ プログラムの要件
- Windows Installer 入門

- インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する
- アプリケーションのディスクの使用量を設定する
- NSTALLDIR と TARGETDIR の違い
- 現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ
- 非管理者パッチのインストールを準備する
- プラットフォームとプラットフォーム スイートの設定

Windows ロゴ プログラムの要件

WWindows 8 デスクトップ アプリの認定を受けるために製品とそのインストールが満たさなくてはならない要件の一覧が、マイクロソフトによって確立されています。要件では、製品が Windows システムで実行されたとき、より高い互換性と安定性、および安全性を持つことができる基準が規定されています。Windows 8 デスクトップ アプリの認定要件を満たす製品は、Compatible with Windows 8 ロゴを表示できます。

Windows ロゴの取得について

Windows ロゴの取得に関する詳細は、「MSDN」を参照してください。この Web サイトには、Windows ロゴ プログラムに関する情報が掲載されています。

InstallShield の Certified for Windows 検証スイート

製品が Windows ロゴ プログラムの要件を満たしているかどうかを判断するために、インストール パッケージまたはマージ モジュールの検証が役に立つ場合があります。パッケージまたはマージ モジュールが 1 つ以上の検証規則に違反した場合、InstallShield は違反の対象となった特定の規則をレポートし、問題を解決するための追加情報を提供します。

このため、Windows ロゴの取得に関心があるユーザーは、次の両方のスイートを使ってインストール パッケージの検証を行うことをお勧めします:

- **InstallShield 検証スイート – Windows 8**—このスイートは、インストール時に潜在的に Windows システムで予期しない動作が発生する原因になる問題を識別する多数の InstallShield 内部整合性評価プログラム (ISICE) から構成されています。このスイートは、完全 MSI 検証スイートでは必ずしも発見できるとは限らない問題を確認します。
- **完全 MSI 検証スイート**—このスイートは、マイクロソフトが作成した ICE から構成されています。

InstallShield でマージ モジュールを作成した場合、次のスイートを利用して、マージ モジュールを検証することができます:

- **InstallShield マージ モジュール検証スイート – Windows 8** このスイートは、Windows システム上でマージ モジュールの予期しない動作が発生する原因になる問題を識別する多数の InstallShield (ISICE) から構成されています。このスイートは、マージ モジュール検証スイートでは必ずしも発見できるとは限らない問題を確認します。
- **マージ モジュール検証スイート**—このスイートは、マイクロソフトが作成した ICE から構成されています。

その他の検証スイートもあります。

詳細については、「プロジェクトの検証」を参照してください。



Windows ロゴ *• InstallShield ヘルプ ライブラリ全体を通して、情報が Windows ロゴ プログラム ガイドラインのコンプライアンスに関連する箇所では Windows Logo ガイドライン アラートが表示されています。*

Windows Installer and Logo Requirements (Windows Installer ヘルプ ライブラリ)

Windows Installer 入門

Windows Installer ベースのインストールは .msi パッケージとして配布され、パッケージは Windows Installer データベース (.msi データベース) と関連データ ファイル (.cab ファイル、非圧縮データ ファイルなど) から構成されます。.msi データベースは COM 構造のストレージとして実装され、ターゲット システム上で処理される変更について記述する何十ものテーブルで構成されます。次に、.msi テーブルのいくつかの例を挙げます:

- File – インストールされるファイルを記述する
- Registry – 書き込まれるレジストリ データを記述する
- Shortcut – ショートカットの設定を記述する

その他の .msi データベース テーブルは、インストールのユーザー インターフェイスの外観と動作を記述したり、Windows サービスおよび ODBC 情報を構成したり、ターゲット システムの特徴を確定したり、インストール中に使用されるアイコンやその他のバイナリ データを保管したりします。

開発者の視点から見て、Windows Installer インストール プログラムの最も大きな変更点は、スクリプトを明示的に書き込む必要がないという点です。その代わり、Windows Installer ベースのインストールは標準カスタムアクションを実行して、ダイアログを表示したり、ターゲット システムをクエリしたり、ターゲット システムに変更を加えたりします。これらのアクションはシーケンスに配置されていて、アクションの集合が順番に並べられています。

Windows Installer は製品のインストール管理専用のアプリケーション プログラム インターフェイス関数の集合 (または API) を含みます。Windows Installer で提供されている機能を活用するには、アプリケーションが Windows Installer API を呼び出す必要があります。

Windows オペレーティング システムに統合されている Windows Installer は、アプリケーションやシステム ツールを管理するためのインターフェイスだけでなく、コンポーネントの管理を行うための標準フォーマットも提供します。様々なバージョンの Windows Installer が Windows オペレーティング システムの再配布可能ファイルとして利用できます。

.msi データベース テーブルを直接編集して Windows Installer パッケージを作成することは可能ですが、多数のテーブルとその関連性を編集するには非常な困難を伴います。InstallShield は、Windows Installer 用のインストール開発過程を様々なビューに分け、.msi データベースにまつわる実装についての詳細の多くのを開発者の目から遮るグラフィカル エディターやウィザードを提供します。

Windows Installer テクノロジについての詳細は、Windows Installer ヘルプライブラリを参照してください。

インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する



プロジェクト *• この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

特定の詳細の一部は、これらのプロジェクト タイプの一部にのみ適用します。これらの違いについては、必要に応じて記述されています。

Windows Vista 以降およびユーザー アカウント制御 (UAC) の目的は、ユーザーが常に標準ユーザーとして実行できるようにすることです。昇格が必要になることは、ほとんどありませんが、必要となる場合、できるだけ短い時間に限られる必要があります。

InstallShield のいくつかの異なる領域では、昇格された権限を要求するために、インストールで UAC の同意または資格情報プロンプトを起動するかどうかが問題になります。Windows Vista 以降のシステムでエンド ユーザーがインストールを実行する際の UAC の動作を適切に定義するためには、これらの異なる設定をよく理解することが必要です。また、これはインストール中に表示される UAC プロンプトの数を最小限に抑えるための手助けとなります。

構成方法によって、InstallShield 前提条件を含むインストールがインストール中のいくつかの時点で、Windows Vista 以降のシステム上で昇格された権限のプロンプトを表示することができます:

1. エンド ユーザーが Setup.exe ファイルを起動するとき
2. Setup.exe ファイルが、昇格された権限を必要とするセットアップ前提条件を起動するとき
3. インストールされる 1 つまたは複数の機能に関連付けられた機能前提条件が含まれるため、ISInstallPrerequisites カスタム アクションが Setup.exe ファイルを機能前提条件インストール モードで再起動するとき

   ISInstallPrerequisites カスタム アクションは、昇格された権限のプロンプトが表示される前に、機能前提条件が昇格された権限を必要とするかどうかを検証しませんので、ご注意ください。また、ISInstallPrerequisites カスタム アクションは、機能前提条件のインストールが必要かどうかを決定するために機能前提条件の条件を確認することもありません。昇格された権限のプロンプトは常に表示されます。

   ***プロジェクト** - 基本の MSI プロジェクトが、機能前提条件のサポートを含みます。*
4. Windows Installer が .msi パッケージの [実行] シーケンスを開始するとき

   ***プロジェクト** - この最後のインストール ポイントは、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに適用しますが、InstallScript プロジェクトには適用しません。*

InstallShield における UAC 関連の設定

以下は、Windows Vista 以降のシステム上で、インストール中に UAC プロンプトを表示するかどうかを判断するための InstallShield の設定です。

- **必要実行レベル**—[リリース] ビューにあるこの設定を使用して、インストールの Setup.exe ファイルが必要とする最小実行レベルを指定します。InstallShield は、Setup.exe ランチャに埋め込まれるアプリケーション マニフェストで選択した値 (管理者、最高権限、または起動者) を使用します。詳細については、「Windows Vista 以降のプラットフォームでのセットアップランチャーの必要実行レベルを指定する」を参照してください。
- **前提条件は管理者権限を必要とする**—インストールの InstallShield 前提条件を作成または変更している場合、このチェック ボックスを使用して、前提条件をインストールするために管理者権限が必要かどうかを指定し

ます。このチェック ボックスは、InstallShield 前提条件エディターの [動作] タブにあります。詳細については、「InstallShield 前提条件に管理者権限が必要であることを指定する」を参照してください。

- **管理者権限が必要**—[一般情報] ビューの [概要情報ストリーム] 領域にあるこの設定を使用して、インストールの .msi パッケージの [実行] シーケンスで管理者権限が必要かどうかを指定します。[いいえ] を設定した場合、InstallShield は Word Count Summary プロパティ の 3 ビット目を設定して、製品のインストールに昇格された権限が必要であることを示します。詳細については、「概要情報ストリーム データを入力する」を参照してください。



***プロジェクト・**"管理者権限が必要" 設定は、InstallScript プロジェクトには適用しません。*

- **前提条件が昇格必要時のアドバタイズ**—[リリース] ビューのこの設定を使用して、Windows Vista 以降のマシンにおけるインストールで InstallShield 前提条件が昇格された権限で正常にインストールされたあと、.msi パッケージをアドバタイズするかどうか指定します。アドバタイズすると指定した場合、それをサイレントまたは完全ユーザー インターフェイス (UI) で実行するかどうかも指定します。アドバタイズすることにより、エンドユーザーに UAC のプロンプトを避けることを許可できる場合があります。アドバタイズを行わない場合、昇格された権限を必要とする .msi パッケージに対して、UAC のプロンプトが表示されます。詳細については、「InstallShield 前提条件が昇格された権限で実行されるときに、製品をアドバタイズするかどうか指定する」を参照してください。



***プロジェクト・**"前提条件が昇格必要時のアドバタイズ" 設定は InstallScript プロジェクトには適用しません。*

また、InstallShield 前提条件の種類 (セットアップ前提条件または機能前提条件)—は、UAC プロンプトが Windows Vista 以降のシステム上でのインストール中に表示されるかどうかに影響を及ぼす可能性があります。これら 2 種類の InstallShield 前提条件についての詳細は、「セットアップ前提条件と機能前提条件の違い」を参照してください。

以下は、Windows Vista 以降における UAC 関連の動作についての注意点です:

- "必要実行レベル" が [起動者] に設定されていて、インストールが管理者権限を必要とする InstallShield 前提条件を含み、"管理者権限" が [いいえ] に設定されている場合、インストール中、プロンプトはエンドユーザーに表示されません。
- "必要実行レベル" が [起動者] に設定されていて、インストールが管理者権限を必要とするセットアップ前提条件を含み、"管理者権限" が [いいえ] に設定されている場合、インストール中、プロンプトはエンドユーザーに 1 回のみ表示されます。再起動がある場合、再起動ごとに、1 回のみ UAC プロンプトが表示されます。
- セットアップランチャーの完全ユーザー インターフェイスが表示される場合で、インストールする必要があるセットアップ前提条件がインストールに含まれているとき、セットアップランチャーは通常、メイン インストールが開始する前にセットアップ前提条件ダイアログを表示します。インストールが必要な 1 つまたは複数のセットアップ前提条件が管理者権限を必要とするとき、メッセージ ボックスの [インストール] ボタンが、エンドユーザーに昇格された権限が必要であることを通知するシールド アイコンと共に表示されます。
- 再起動後、インストールが続行し、権限の昇格が必要な場合、続行メッセージ ボックスの [OK] ボタンがシールド アイコンと共に表示されます。権限の昇格が必要がない場合、シールド ボタンは表示されません。
- インストールに、ターゲット マシンにインストールしなければならないセットアップ前提条件が複数あり、このうち 1 つまたは複数のセットアップ前提条件が管理者権限を必要とする場合、UAC プロンプトが、一番最初のセットアップ前提条件がインストールされる前に表示されます。これにより、それぞれの前提条件のインストールごとに別々の UAC プロンプトを要求することなく、昇格された権限をすべての前提条件に対し

て使用できるようになる場合があります。ただし、セットアップ前提条件のインストールで再起動がある場合、管理者権限は失われることになり、これにより、残りの前提条件が管理者権限を必要とする場合、UAC プロンプトが表示されることがありますので注意してください。

機能前提条件では動作が若干異なります。インストールが、前提条件に関連付けられている機能をインストールする場合、ISInstallPrerequisites カスタム アクションが機能前提条件インストール モードで Setup.exe を再起動するときに UAC プロンプトが表示されます。これは、機能前提条件のいずれかが昇格された権限を必要とするかどうかに関わらず発生します。また、これは機能前提条件のインストールが必要かどうかを決定するために機能前提条件のいずれかが評価される前に発生します。機能前提条件インストールによって再起動が行われる場合、管理者権限は失われますのでご注意ください。再起動のあと、再び ReadyToInstall ダイアログが表示され、エンド ユーザーは残りのインストールを続行するために [インストール] ボタンをクリックする必要があります。この場合、ISInstallPrerequisites カスタム アクションが機能前提条件インストール モードで Setup.exe を再起動するとき、UAC プロンプトが再び表示されます。

- "管理者権限が必要" が [いいえ] に設定されている状態で、適切な権限をもたずに .msi パッケージがタスクを実行しようとすると、Windows Installer によって実行時エラーが表示される場合があります。
- 権限がインストールの終わりで昇格され、SetupCompleteSuccess ダイアログが製品を起動すると、昇格された権限が製品に引き継がれます。通常、昇格された権限を使用したアプリケーションの実行は推奨されていません。

サンプル シナリオ

次のセクションは、InstallShield で上記の設定の値を異なる組み合わせで使用した場合の例です。これらの図では、Windows Vista 以降が標準ユーザーまたは限られた権限を持つ管理者ユーザーに昇格された権限を要求しています。これらの例は、Windows Vista 以降のシステム上の UAC のデフォルト設定に基づいています。

例 1: UAC プロンプトが、管理者権限が必要な前提条件に表示される – .msi ファイルがアドバタイズされる

例 1 の図表は、セットアップ前提条件、機能前提条件、および .msi パッケージの [実行] シーケンスで昇格が必要なインストールを説明します。Windows Vista 以降では、セットアップ前提条件に UAC プロンプトを、また機能前提条件に別のもう 1 つの UAC プロンプトを表示します。

機能前提条件が含まれなかった場合、または機能前提条件ではなくセットアップ前提条件であった場合、2 番目の UAC プロンプト (図表では *UAC prompt #2* と名付けられています) は表示されません。これらの場合、セットアップ前提条件が昇格された権限を使ってインストールされた後、.msi パッケージが正しくアドバタイズされるため、UAC prompt #2 は必要ありません。

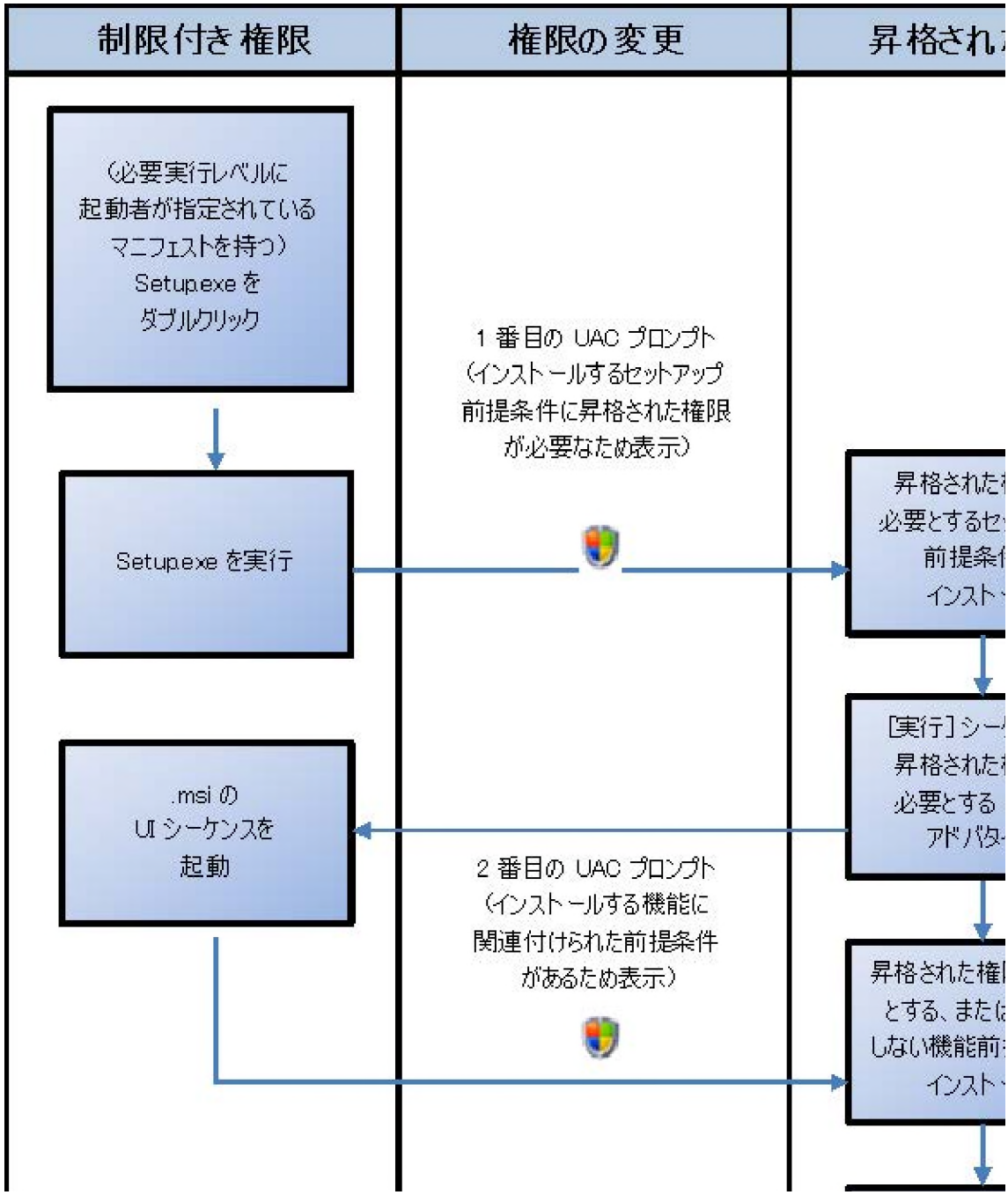


図 1: 例 1: "必要実行レベル" が [起動者] で、かつ .msi ファイルをアドバタイズするインストールのダイアグラム

例 2: UAC プロンプトが Setup.exe と、再起動後に管理者権限が必要な前提条件に表示される

例 2 の図表は、Setup.exe、2 つのセットアップ前提条件、機能前提条件、および .msi パッケージの [実行] シーケンスで昇格が必要なインストールを説明します。Setup.exe ファイルに、管理者を必要実行レベルとして指定するマニフェストがあるため、昇格された権限はインストールの各部分で使用されます。昇格された権限は再起動中に失われるため、2 つ目の UAC プロンプトが表示されます。

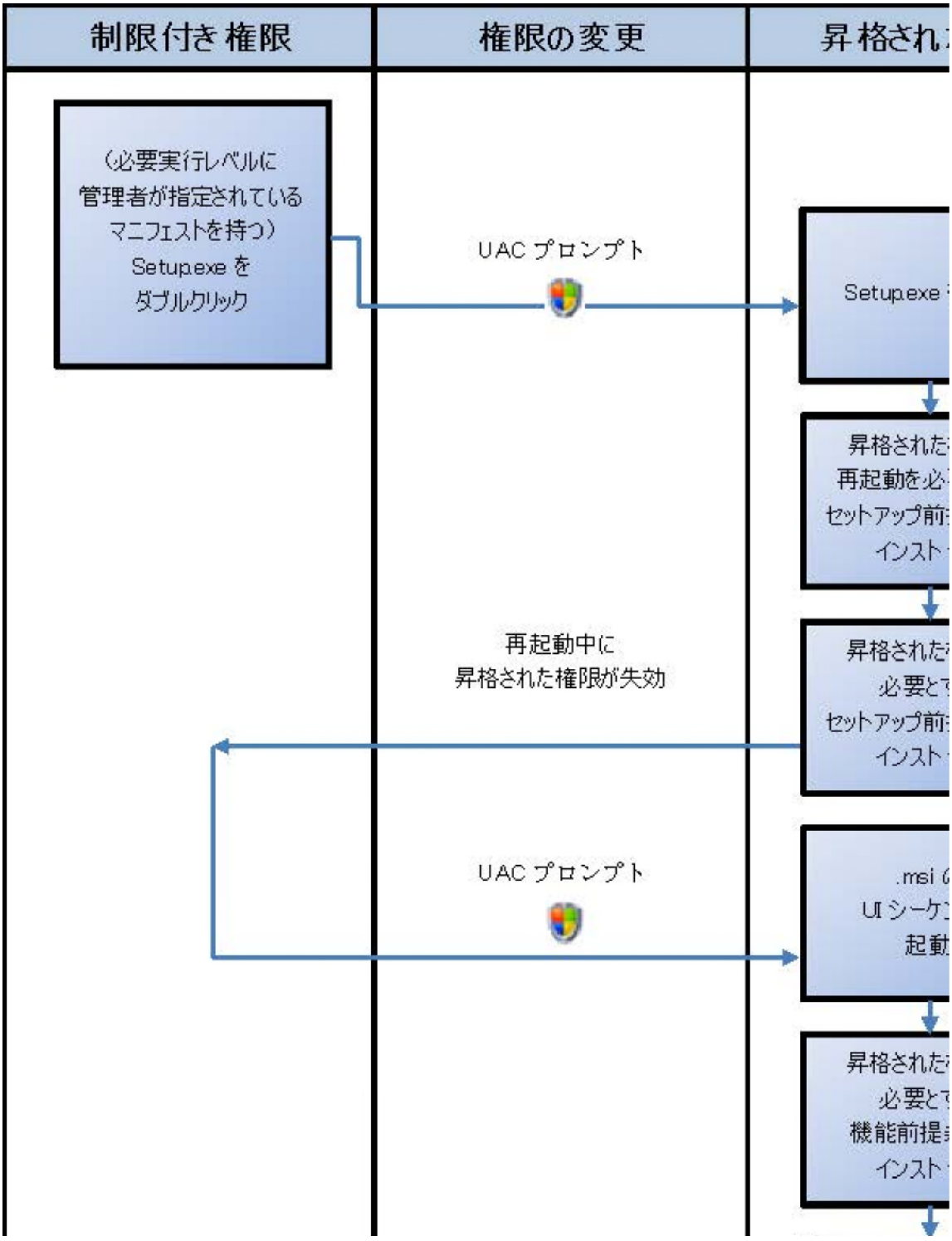


図 2: 例 2: "必要実行レベル" が [管理者] に設定されているインストールのダイアグラム

Using Windows Installer with UAC(Windows Installer ヘルプ ライブラリ)
Guidelines for Packages (Windows Installer ヘルプ ライブラリ)
Authoring Packages without the UAC Dialog Box (Windows Installer ヘルプ ライブラリ)

Installing a Package with Elevated Privileges for a Non-Admin (Windows Installer ヘルプ ライブラリ)

アプリケーションのディスクの使用量を設定する

Windows Installer はファイル コスティングと呼ばれるプロセスを使用して現在のインストールに必要な総ディスク容量を決定します。これには、インストールまたは削除されるファイル、レジストリ エントリ、ショートカット、およびインストールのその他のコンポーネントのコストが含まれます。

ファイル コスティング

ファイル コスティングは、いくつかのファイルが新規バージョンによって上書きされ、それによってファイル コストが削減されるという事実を考慮しています。これらの値は各ファイルがインストールまたは削除されるボリュームに依存し、コンポーネントのディレクトリ関連付けが変更される場合には再計算されます。

ファイルコストは、各コンポーネントについて、ローカルにインストールされるか、または CD などのソース メディアからインストールして実行するか、または削除されるかに応じて決定されます。

InstallShield では、アプリケーションのディスク使用量を設定することができます。これによって、ソース メディアから、またはローカルマシンからアプリケーションを実行するか、あるいは要求されたときにインストールするかを選択することができ、ファイルコストをコントロールすることが可能となります。ソースからアプリケーションを実行すると、アプリケーションの弾力性が拡張され、エンドユーザーのシステムのスペースが節約されます。

アプリケーションの復元性

Windows Installer がコンポーネントを提供できない状況で、該当するファイルが破損しているか、現在のファイルが使用可能なバージョンよりも古い場合には、Windows Installer 技術にを使ったアプリケーションによるコンポーネントの修復と再インストールの試行が可能です。

ソースの復元性

アプリケーションの復元性に加え、Windows Installer ではソースリストによってソースの復元性もサポートします。ソースリストには、必要時にアプリケーションをインストールするネットワークの場所、URL、CD などを含めることができます。管理者はグループ ポリシーエディターを使用してこの機能を無効にすることができます。

NSTALLDIR と TARGETDIR の違い

INSTALLDIR は、エンドユーザーによって起動される Setup.exe や .msi データベースなど、基本の MSI および InstallScript MSI のインストールで使用するメインの製品インストール ディレクトリを表します。

TARGETDIR は、InstallScript インストールまたは管理 Windows Installer ベースのインストール (/a コマンドラインスイッチを使って Setup.exe や MsiExec.exe をユーザーが実行した場合) のインストール ディレクトリを示します。

InstallScript MSI プロジェクトでは、InstallScript 変数の MSI_TARGETDIR が管理インストール時のインストール先を記録します。

現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

エンドユーザーが製品の現在のバージョンで同製品の将来のバージョンを上書きインストールするのを防止するためには、[アップグレード] ビューにメジャー アップグレード アイテムがあること、製品の現在のバージョンによって将来のバージョンが上書きインストールされないようにメジャー アップグレード アイテムが適切に構成されていること、および、製品に適切に構成、スケジュールされたタイプ 19 のカスタム アクションが含まれている必要があります。

新しい基本の MSI または InstallScript MSI プロジェクトを作成したとき、現在のインストールが将来のメジャー バージョンを上書きするのを防ぐためのサポートが自動的に追加されます:

- [アップグレード] ビューに、ISPreventDowngrade という名前のメジャー アップグレード アイテムが含まれています。

  [アップグレード コードを共有している製品] オプションは、メジャー アップグレード アイテムの [共通] タブで選択されています。[詳細] タブの "アップグレード コード" 設定の値は {000000000000-0000-0000-0000-00000000} です。リリースをビルドすると、InstallShield は、ビルド時に生成する .msi パッケージのプレースホルダー値の代わりに、適切なアップグレード コード値を使用します。アップグレード コードに関する詳細は、「アップグレード コードを設定する」を参照してください。

  InstallShield は、このメジャー アップグレード アイテムの "検出プロパティ" 設定を ISFOUNDNEWERPRODUCTVERSION に設定し、他の設定を適切に構成します。

- [カスタム アクションとシーケンス] ビューには ISPreventDowngrade という名前のカスタム アクションが含まれています。ISPreventDowngrade は、エンドユーザーが現在のバージョンの製品で将来のメジャー バージョンを上書きインストールしようとしたとき Windows Installer が起動するタイプ 19 カスタム アクションです。

  InstallShield は、使用されているユーザー インターフェイスのレベルにかかわらず、適宜 Windows Installer が実行するように、ISPreventDowngrade カスタム アクションを [インストール] シーケンスの [ユーザー インターフェイス] シーケンスと [実行] シーケンスにスケジュールします。また、InstallShield は ISFOUNDNEWERPRODUCTVERSION をこのカスタム アクションの条件として使用します。

以下は、InstallShield 12 以前で作成し、InstallShield 2025 にアップグレードしたプロジェクトにこのサポートを手動で追加する方法です。



タスク

エンドユーザーが、製品の現在のバージョンで将来のメジャー バージョンを上書きインストールするためのサポートを手動で追加するには、以下の手順に従います:

1. **[アップグレード]** ビューを使用して、メジャー アップグレード アイテムをプロジェクトに追加します。
2. **[共通]** タブで、**[自分のアップグレード コードを使用する製品]** オプションを選択する。

3. メジャーアップグレードアイテムの **[詳細]** タブで設定を次のように構成します。

   a. **"最小バージョン"** 設定で、現在のプロジェクトに使用している製品バージョンを指定します。

      その代わりに、この設定に次の文字列を入力することもできます:

      *****ALL_VERSION*****

      その値はプレースホルダー値です。**"最小バージョン"** 設定で抽出された文字列を使うと、InstallShield は、ビルド時に生成する .msi パッケージのプレースホルダー値の代わりに、現在開いているプロジェクトの製品バージョンを使用します。

   b. **"最大バージョン"** 設定は空白のままにします。この設定に値が表示されている場合、それを削除します。

   c. プロジェクトに複数の言語が含まれる場合、**"言語"** 設定で言語識別子を指定します。

   d. **"検出のみ"** 設定で、**[はい]** を選択します。他の [はい]/[いいえ] で答える設定は、すべて **[いいえ]** を選択します。

   e. **"検出プロパティ"** 設定で、次のような分かりやすい名前を入力します。

      FOUNDNEWERVERSION

4. プロジェクトで、エンド ユーザーが製品の現在のバージョンで将来のメジャー バージョンを上書きインストールしようと試みた場合に対応するために、タイプ 19 カスタム アクションを追加し、スケジュールします。

   a. **[カスタム アクションとシーケンス]** ビューで、**[カスタム アクション]** エクスプローラーを右クリックし、**[新しいエラー]** をクリックします。

   b. 新しいカスタム アクションを選択します。

   c. **"エラー メッセージ"** 設定で、エンド ユーザーが製品の現在のバージョンで将来のメジャー バージョンを上書きインストールしようを試みた場合に表示される、エラー メッセージ テキストを入力します。

   d. **"インストール UI シーケンス"** 設定と **"インストール実行シーケンス"** 設定で、**[次の後: FindRelatedProducts]** を選択します。

   e. **"インストール UI 条件"** と **"インストール実行条件"** 設定で、ステップ 3e で指定した値を入力します。

Preventing an Old Package from Installing Over a Newer Version (Windows Installer ヘルプ ライブラリ)

非管理者パッチのインストールを準備する

Windows Installer 3.0 以上では、管理者以外によるインストールが可能なパッチを作成することができます。非管理者によるパッチは厳しい条件が満たされたときのみ利用することができます。たとえば、パッチが更新するベース インストールは、パッチ パッケージの署名に使用される証明書を含まなくてはなりません。満たされなければならない他の基準については、「非管理者パッチ」を参照してください。



タスク

後で非管理者パッチによってアップデートが可能なベースインストールを作成するには、以下の手順を実行します。

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、デジタル署名の情報を構成するリリースをクリックします。

3. **[署名]** タブをクリックします。
4. デジタル署名情報を指定します。

必要な情報が自動的に、MsiDigitalCertificate テーブルおよび MsiPatchCertificate テーブルに追加されます。これによって、非管理者がインストールすることができるパッチの作成が可能になります。



ヒント: *[リリース] ビューでは、デジタル署名設定も利用できます。*

Windows Installer のデザインにより、パッケージの署名にある証明書を使用し、それとは異なる証明書で署名されたパッチを許可することができます。以下は、ベース インストールにパッチのための証明書を追加する方法です。



タスク ***追加のデジタル証明書を追加するには、以下の手順に従います:***

1. サンプル ファイルの署名には SignTool.exe 等のツールを使います。SignTool.exe は、Windows SDK に含まれています。
2. 元のインストール プロジェクトで、**ダイレクト エディター**を開きます。
3. **[テーブル]** エクスプローラーで、**MsiDigitalCertificate** をクリックします。
4. テーブルの最後の行をクリックして新規エントリを追加します。
5. **DigitalCertificate** フィールドに、証明書の一意の名前を入力します。
6. **CertData** フィールドをクリックします。**[バイナリ ストリームの編集]** ダイアログ ボックスが開きます。
7. **[ファイル名]** ボックスには、ステップ 1 で署名したファイルのパスと名前を入力します。参照ボタンをクリックしてファイルを指定することもできます。
8. **[OK]** をクリックします。
9. **[テーブル]** エクスプローラーで、**MsiPatchCertificate** をクリックします。
10. テーブルの最後の行をクリックして新規エントリを追加します。
11. **PatchCertificate** フィールドに、パッチ証明書の一意の名前を入力します。
12. **DigitalCertificate** フィールドで、ステップ 4 の MsiDigitalCertificate テーブルに作成したエントリを選択します。



メモ: *ベース インストールが期限切れのデジタル証明書で署名してあり、MsiDigitalCertificate および MsiPatchCertificate テーブルに以前の証明書と新しい証明書を含む中間パッチが適用されなかった場合、非管理者パッチは、ユーザー アカウント管理 (UAC) を表示します。*

User Account Control (UAC) Patching (Windows Installer ヘルプ ライブラリ)
MsiDigitalCertificate Table (Windows Installer ヘルプ ライブラリ)
MsiPatchCertificate Table (Windows Installer ヘルプ ライブラリ)

プラットフォームとプラットフォーム スイートの設定



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

一部の設定は両方のプロジェクトの種類に適用しますが、他の一部はどちらか一方のプロジェクトの種類にのみ適用します。

InstallShield には、プラットフォーム、プラットフォーム スイートおよび言語の設定がいくつか含まれています。

プロジェクト レベルでプラットフォームを指定する

1 つのプロジェクト レベルの設定を使って、プロジェクトをサポートするプラットフォームを指定できます:

- **プラットフォーム フィルター**—この設定で、プロジェクトでコンポーネントまたはリリースにオペレーティング システム要件を選択するときの選択肢に加えるプラットフォームを指定します。一般的に、プロジェクト レベルのこの設定で表示されないプラットフォームがあるとき、プロジェクト内にある特定のコンポーネントまたはリリースをこのプラットフォームにターゲットするという指定ができなくなります。

  "プラットフォーム" 設定は InstallScript プロジェクトで提供されています。この設定にアクセスするには、[一般情報] ビューを開きます。[プロジェクトの設定] ダイアログ ボックスでもこの設定を構成することができます。その場合、[プロジェクト] メニューで [設定] をクリックして、[プラットフォーム] タブをクリックします。



メモ: *プロジェクト レベルでプラットフォームを指定しても、インストールを実行するときのターゲット システム要件は作成されません。InstallScript プロジェクトでターゲット システム要件を作成する場合、SYSINFO 構造を使用して、ターゲット システムのオペレーティング システムを識別します。*

InstallScript MSI プロジェクトにおけるターゲット システム要件の方法については、「プロジェクト アシスタントでオペレーティング システムを指定する」を参照してください。

コンポーネント レベルで、オペレーティング システムとプラットフォーム スイートを指定する

2 つのコンポーネント レベルの設定を利用して、コンポーネントのプラットフォーム情報を指定できます:

- **オペレーティング システム**—コンポーネントが 1 つまたは複数のオペレーティング システムに固有の場合、この設定を使用して、それらのオペレーティング システムを指定します。ターゲット マシンのオペレーティング システムがこの設定に指定されたオペレーティング システムの中にない場合、コンポーネントはインストールされません。

  デフォルトでは、コンポーネントはオペレーティング システムに依存しません。つまり、コンポーネントに特定のオペレーティング システム固有のデータがないことを意味します。

  "オペレーティング システム" 設定は、InstallScript プロジェクトと InstallScript MSI プロジェクトで使用できます。この設定にアクセスするには、[コンポーネント] ビューを開いて、コンポーネントを選択します。

- **プラットフォーム スイート**—コンポーネントが 1 つまたは複数のプラットフォーム スイートに固有の場合、この設定を使用して、それらのスイートを指定します。1 つ以上のスイートを指定する場合、コンポーネント

がインストールされるために、指定されたスイートすべて、または 1 つ以上のスイートがターゲット マシンに存在している必要があるかどうかを示すことができます。

デフォルトでは、コンポーネントはプラットフォーム スイートに依存しません。つまり、コンポーネントのデータ (ファイル、レジストリ エントリなど) はどれも特定のプラットフォーム スイートに固有ではありません。

この設定により、"オペレーティング システム" 設定よりもさらに細かくフィルターすることができます。"プラットフォームスイート" 設定は必要な場合のみ設定し、またアプリケーションが適切に機能するために必要なプラットフォーム スイートのみを確実に選択してください。たとえば、コンポーネントが WindowsXP の Home と Professional バージョンの両方にインストールされる必要がある場合、この設定の値を [スイート非依存] のままにしておくことができます。"オペレーティング システム" 設定に Windows XP を選択すると、両方のエディションが含まれます。

"プラットフォーム スイート" 設定は InstallScript プロジェクトで提供されています。この設定にアクセスするには、[コンポーネント] ビューを開いて、コンポーネントを選択します。

リリース レベルでプラットフォームを指定する

1 つのリリース レベルの設定を利用して、リリースのプラットフォーム情報を指定できます:

- **プラットフォーム**—リリースが 1 つまたは複数のプラットフォームに固有の場合、この設定を使用して、それらのプラットフォームを指定します。コンポーネントに指定されたプラットフォームが、この設定で選択されたプラットフォームのどれにも一致しない場合、そのコンポーネントはリリースに含まれません。

  この設定のデフォルト値は、[プロジェクト設定を使用] です。この値は、リリースがプロジェクト レベルで指定されたプラットフォームをサポートすることを示します。

  "プラットフォーム" 設定は InstallScript プロジェクトで提供されています。この設定にアクセスするには、[リリース] ビューを開いて、リリースを選択します。この設定は、リリース ウィザードの [プラットフォーム] パネルでも構成できます。

実行時にプラットフォームおよびプラットフォーム スイートのサポートを制御する (InstallScript プロジェクト)

InstallScript インストールの実行中に、**FeatureFilterOS** 関数を呼び出して、インストールがサポートするプラットフォームとプラットフォーム スイートを制御することができます。

OnFilterComponents イベント ハンドラーで、フレームワークは通常、適切なコンポーネントだけがインストールされるように、この関数をターゲット システムに一致するプラットフォームおよびプラットフォーム スイートで呼び出します。**FeatureFilterOS** を呼び出して、デフォルトの動作をオーバーライドし、指定したプラットフォームまたはプラットフォーム スイート基準に基づいてコンポーネントをインストールしたり、またはコンポーネントのインストールを防いだりすることができます。

詳細については、「FeatureFilterOS」を参照してください。

インストール情報を指定する

インストール プロジェクトを作成を開始するとき、まず最初にインストールの重要情報を指定する必要があります。これには、製品およびプロジェクト設定の指定と "プログラムの追加と削除" 設定の構成が含まれます。

このセクションには、次のトピックが含まれています:

- 一般的なプロジェクト設定を構成する
- 概要情報ストリーム データを入力する
- [プログラムの追加と削除] 情報を構成する
- 製品のソフトウェア識別タグを含める
- 製品の Microsoft System Center Configuration Manager アプリケーション モデル データを含める

一般的なプロジェクト設定を構成する

InstallShield は、プロジェクト設定を単一インストール プロジェクトファイル (.ism ファイル) に保管します。このファイルには、プロジェクトについてのすべての情報が格納されています。[一般情報] ビューを使うと、作成者名、プロジェクトがサポートする言語、および記入するコメントなど、インストール プロジェクトについての基本情報を編集することができます。

[一般情報] ビューではまた、製品名、製品コード (GUID)、およびバージョン番号といった一般的な製品情報も構成できます。製品は、インストール プロジェクトの構成中で最上位のものです。インストールは機能とコンポーネントに分割でき、これらが製品のサブセットになります。インストールに複数の機能とコンポーネントがある場合でも、製品は 1 つしかありません。

各プロジェクトとプロジェクト設定についての詳細は、「[一般情報] ビュー」を参照してください。

このセクションには、次のトピックが含まれています:

- InstallShield プロジェクト ファイル (.ism) を XML またはバイナリ形式で保存する
- 製品名の指定
- 製品バージョンを指定する
- InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトにおける製品バージョン番号
- Windows Installer ベースのプロジェクトで製品コードを設定する
- InstallScript ベースのプロジェクトで製品コードを設定する
- アップグレード コードを設定する
- 製品の条件を設定する
- デフォルトの製品インストール先フォルダー (INSTALLDIR) の設定
- ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護
- Windows Installer インストールをログ記録するかどうかを指定する
- InstallScript インストールを複数回実行する
- "メンテナンスの有効化" 設定を構成する

- InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い

InstallShield プロジェクト ファイル (.ism) を XML またはバイナリ形式で保存する

InstallShield では、.ism プロジェクト ファイルを XML またはバイナリ形式で保存することができます。



タスク ***プロジェクト ファイルの形式を指定するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"プロジェクト ファイルの形式"** 設定で、適切なオプションを選択します: 選択可能なオプションは以下のとおりです:
   - **バイナリ** – プロジェクト ファイルをデータベース ファイルとして保存するには、このオプションを選択します。プロジェクトを開いたり保存したりする際のスピードは、この形式が最も早いです。
   - **XML** – プロジェクト ファイルを階層構造を持つテキスト ベースの形式で保存するには、このオプションを選択します。このプロジェクト ファイル形式は、ソース コード管理システムでの使用に最適です。



メモ: *InstallShield プロジェクト ファイル (.ism) を XML またはバイナリ形式で保存した時、その拡張子は保持されます。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクト (.issuite) は、常に XML ファイルとして保存されます。

製品名の指定

[一般情報] ビューの "製品名" 設定に製品の名前を入力します。ここに入力する名前は、インストールを作成する製品の名前です。この値は、プロジェクト全体を通して、次のような名前に使用されます。

- プロジェクトの場所の下にあるソースファイルフォルダーの名前
- 基本の MSI および InstallScript MSI プロジェクトの場合: InstallShield がデフォルトでビルドする Windows Installer パッケージ (.msi file) の名前
- 実行時ダイアログ
- Windows ロゴ要件に従って登録される情報レジストリキー情報値は以下の場所にあり、エンド ユーザーが製品を変更または削除するのに使用できる [プログラムの追加と削除] で使用されます。

  HKEY_LOCAL_MACHINE¥Software¥Microsoft¥Windows¥CurrentVersion¥Uninstall¥*ProductCode*

入力値はソース ファイルのパスに組み込まれるため、製品名には次のいずれの文字も含めることはできません: ¥ / : * ? " < > | –



ヒント: *InstallScript または InstallScript MSI プロジェクトでは、UNINSTALL_DISPLAYNAME 変数を使用して、[プログラムの追加と削除] に異なる製品表示名を指定することができます。*

基本の MSI および InstallScript MSI プロジェクトの場合、.msi パッケージ ファイルの名前に製品名以外を使用するとき、[リリース] ビューにある製品構成で [全般] タブの "MSI パッケージ ファイル名" 設定を使って、異なる .msi ファイル名を指定できます。製品のアップデートを行うためのマイナー アップグレードまたはスモール アップデートをリリースできるようにしたい場合は、インストールの以前のバージョンと最新バージョンの .msi パッケージ名が同じでなくてはなりません。.msi ファイル名が異なる場合にマイナー アップグレードまたはスモール アップデートを実行しようとすると、Windows Installer ランタイム エラー 1316 の原因となる可能性があります。



***注意** - 製品名にアンパサンド (&) を含める場合、エンドユーザーダイアログで名前を正しく表示するには、2 つのアンパサンド (&&) を使用する必要があります。たとえば New & Improved Product と表示する場合、製品名を New && Improved Product と入力します。*

製品バージョンを指定する

製品のバージョン番号を指定するとき、必ず有効な製品バージョンを入力してください。バージョンには、数値のみを使用できます。一般的なフォーマットは *aaa.bbb.ccccc* または *aaa.bbb.ccccc.ddddd* で、*aaa* はメジャー バージョン番号、*bbb* はマイナー バージョン番号、*ccccc* はビルド番号、および *ddddd* はバージョン番号を示します。*aaa* と *bbb* の最大値は 255 です。*ccccc* と *ddddd* の最大値は、65,535 です。

実行時に、インストールが、インストールされる製品のバージョン番号を登録します。完全なバージョン文字列が [プログラムの追加と削除] に表示されます。製品バージョン番号は、その一部をインストール エンジンがアップグレードの適用を判断するために使用するため、重要です。

製品バージョンを構成するには、[一般情報] ビューを使用します。この製品バージョンは、たとえば基本の MSI プロジェクトの [リリース] ビューにある製品構成など、InstallShield の別の領域でオーバーライドすることができます。また、基本の MSI または InstallScript MSI プロジェクトの [アップグレード] ビューなど InstallShield の別の領域で、アップグレードがターゲットとするバージョン番号を指定することもできます。

製品のバージョンを指定するときに 4 番目のフィールド (*ddddd*) を含めることもできますが、インストールは異なる製品バージョンを区別するときに製品バージョンのこの部分を無視します。



***プロジェクト-** 基本の MSI、InstallScript、および InstallScript MSI プロジェクトの場合―リリースに* Setup.exe *が含まれる場合、指定した製品バージョンが* Setup.exe *の [プロパティ] ダイアログ ボックスに表示されます。詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。*

InstallScript および InstallScript オブジェクト プロジェクトの場合、値をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。(パス変数を使用するには、[プロジェクト] メニューから [設定] をクリックします。次に、[アプリケーション] タブで適切なパス変数を選択します。)

ProductVersion Property (Windows Installer ヘルプ ライブラリ)

InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトにおける製品バージョン番号



***プロジェクト-** この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*

- *InstallScript オブジェクト*

デフォルトで、InstallScript プロジェクトではパックされた DWORD 形式のバージョン番号を使用します。つまり 4 バイトの値で、最初のバイトはメジャー バージョン、2 番目のバイトはマイナー バージョン、最後の 2 つのバイトはビルド番号です。パックされた DWORDS が入力され、メジャー、マイナー、ビルド形式 (例、1.2.3 または 255.255.65535) で表示されます。インストール中の製品のバージョン番号を登録するデフォルトのスクリプト コードではパックされた DWORD バージョン番号が想定され、アップデートのインストール中にインストール済み製品のバージョン番号との比較に使用されます。

バージョン番号の中に パックされた DWORD 形式ではないものが存在する場合、次のセクションの内容に従ってスクリプト コードを変更する必要があります。

既にインストールされている製品のバージョン番号

デフォルトの **OnSetUpdateMode** イベント ハンドラー関数コードは、インストール中の製品のバージョン番号および、([リリース] プロパティ シートで指定されている要領で) アップデートを適用できるバージョン番号を、システム変数 *IFX_INSTALLED_VERSION* と比較します。インストールは、自動的に、このシステム変数の値を、アプリケーションアンインストールレジストリキーのVersion 値にあるデータに等しい文字列に初期化することを試みます。その値が存在しない場合、または、そのデータがパックされた DWORD 形式ではない場合、*IFX_INSTALLED_VERSION* の値はヌル文字列 ("") になり、その場合、エラー メッセージがデフォルトの **OnSetUpdateMode** コードによって表示され、インストールは中止 (abort) されます。これを解決する方法の 1 つとして、システムでバージョン情報をチェックし、*IFX_INSTALLED_VERSION* に適切なパックされた DWORD 値を設定するコードを挿入するという方法があります。たとえば、以前のインストールで HKEY_LOCAL_MACHINE¥Software¥MyCompany¥MyProduct の下に MyVersion の値にバージョン文字列が格納されていると、次のようなコードを挿入できます。

```
if (IFX_INSTALLED_VERSION="") then
    /* 登録されているバージョン情報を取得します。*/
    RegDBSetDefaultRoot ( HKEY_LOCAL_MACHINE );
    RegDBGetKeyValueEx( "Software¥¥MyCompany¥¥MyProduct",
        "MyVersion", REGDB_STRING, szVersionString, nSize );

    /*  IFX_INSTALLED_VERSION に値を割り当てます。*/
    switch (szVersionString)
        /* 登録されているバージョン文字列の "A" は、既存のバージョン番号の 1.0.0 に対応します。*/
        case "A":
            IFX_INSTALLED_VERSION = "1.0.0";
        /* 登録されているバージョン文字列の "B" は、既存のバージョン番号の 1.1.0 に対応します。*/
        case "B":
            IFX_INSTALLED_VERSION = "1.1.0";
        /* 不在のバージョン文字列は、既存のバージョン番号の 0.0.0 に対応します。*/
        デフォルト:
            IFX_INSTALLED_VERSION = "0.0.0";
    endswitch;
endif;
```

インストール中の製品のバージョン番号

デフォルトの **OnMoveData** イベント ハンドラー関数コードは、**RegDBSetVersion** を呼び出して、[一般情報] ビューにに入力された製品バージョン番号を使い、それをターゲット システムのレジストリに入力します。**RegDBSetVersion** は、製品バージョンをはパックされた DWORD 形式だと仮定します。パックされた DWORD 形式ではない製品バージョンを登録する場合、OnMoveData の **RegDBSetVersion** の呼び出しを次のようなコードで置き換えます:

```
/* セットアップは自動的にシステム変数
IFX_PRODUCT_VERSION の値を
[一般情報] ビューの "製品バージョン" 設定に入力された製品バージョンに初期化します。*/
RegDBSetItem(REGDB_UNINSTALL_VERSION, IFX_PRODUCT_VERSION );
RegDBSetItem(REGDB_UNINSTALL_DISPLAY_VERSION, IFX_PRODUCT_VERSION );
```

RegDBSetItem は、**MaintenanceStart** の後に呼び出される必要があります。

Windows Installer ベースのプロジェクトで製品コードを設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

製品コードは製品を識別する固有の文字列です。インストールは実行時に製品コードを使用して、その製品が既にインストール済みであるかどうかを判断します。

製品を一意に識別する GUID を入力するか、[一般情報] ビューの "製品コード" 設定にある **[新しい GUID の生成]** ボタン ([...]) をクリックして、自動的に新しい GUID を生成します。インストールが、この GUID を実行時に登録します。



注意 - *この製品コードは製品を一意に識別するため、製品のリリースを配布してからはこのコードを変更しないことをお勧めします。*

InstallScript ベースのプロジェクトで製品コードを設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

製品コードは製品を一意に識別する GUID 文字列です。インストールは実行時に製品コードを使用して、その製品が既にインストール済みであるかどうかを判断します。

製品を一意に識別する GUID を入力するか、[一般情報] ビューの "製品コード" 設定にある **[新しい GUID の生成]** ボタン ([...]) をクリックして、自動的に新しい GUID を生成します。インストールが、この GUID を実行時に登録します。



注意 - *プロジェクト GUID は、アンインストールまたはメンテナンスを元のインストールと関連付けるのに使用されます。新しい GUID は、(既存のプロジェクトのコピーを含む) 作成した各新規プロジェクトに対して自動的に生成されます。プロジェクトの GUID を変更すると、以前の GUID を回復することはできません。このため、プロジェクトの GUID の変更は、通常必要なく、変更する場合は注意が必要です。*

アップグレード コードを設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *QuickPatch*
- *トランスフォーム*

アップグレード コードとは、製品が所属する製品ファミリを識別する一意な GUID です。ほとんどの場合、アップグレード コードは、関連製品ファミリの異なるバージョンおよび言語にわたって統一されている必要があり、これを使って、Windows Installer がインストール済みの製品の関連バージョンを検索します。

新しい製品の GUID を構成するには、[一般情報] ビューの "アップグレード コード" 設定を使用します。このアップグレード コードの値は、InstallShield の別の領域でオーバーライドすることが可能です。たとえば、[リリース] ビューの製品構成、または [リリース] ビュー内の製品構成にある [複数インスタンス] タブで定義されている各インスタンス。

製品の新しいバージョンのアップグレードを作成中の場合、[一般情報] ビューで構成されたアップグレード コード (または、前述の InstallShield のその他の領域でオーバーライドされたアップグレード コード) をアップグレード プロジェクトまたは QuickPatch プロジェクトの [アップグレード] ビューで指定する GUID として使用します。これらのアップグレード コードが一致するとき、アップグレードまたはパッチがベース バージョンをターゲットすることができるため、Windows Installer が必要に応じてアップデートすることが可能となります。

製品の条件を設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト アシスタントの [インストール要件] ページでは、一般的に用いられるターゲット システムのインストール要件を指定できます。たとえば、アプリケーションを正しく実行するために特定の OS が必要な場合、[インストール要件] ページで指定することができます。

InstallShield ではまた、製品がインストールされる前に Windows Installer が評価しなくてはならない、独自のカスタム条件を定義できます。[一般情報] ビューと、プロジェクト アシスタントの [インストール要件] ページで作成する条件は、製品全体に適用します。ターゲット システム上で 1 つ以上の条件が False 評価された場合、インストールが終了してエラー メッセージが表示されます。

たとえば、製品を正しく実行するために RAM が 64 MB 必要な場合、[製品条件ビルだ] ダイアログ ボックスを使って、条件を作成できます。実行時、Windows Installer がターゲット システムをチェックして、インストールされている RAM のサイズを判断します。64 MB 未満の場合、またはその他の製品条件が False の場合、Windows Installer がエラーを表示して、インストールが終了します。



ヒント: *Windows Installer が製品の起動条件を評価する順番を保証することはできません。条件が評価される順番を制御する必要がある場合、[カスタム アクションとシーケンス] ビューで各条件に対してエラー カスタム アクションを作成して、それらを適切な順序にスケジュールすることができます。*



タスク ***製品のインストールにカスタム起動条件を作成するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"インストール条件"** 設定をクリックしてから、省略記号ボタン (...) をクリックします。**[製品条件ビルダー]** ダイアログ ボックスが開きます。
3. **[新しい条件]** ボタンをクリックします。InstallShield によって、**"条件"** ボックスの下に新しい条件が追加されます。
4. 以下のいずれかを実行します。
   - **[条件]** 列で、起動条件を入力します。
   - **[プロパティ]** リスト、**[演算子]** リスト、および **[追加]** ボタンを使って、条件ステートメントをビルドします:
     a. **[プロパティ]** リストで、プロパティを選択してから、**[追加]** ボタンをクリックします。InstallShield は、プロパティを **[条件]** 列に追加します。
     b. 条件ステートメントに演算子を含める場合、**[演算子]** リストから演算子を選択してから、**[追加]** ボタンをクリックします。InstallShield によって、条件ステートメントに演算子が追加されます。
     c. 条件ステートメントが値を含む場合、"条件" フィールドをダブルクリックしてから END を押し、挿入ポイントを条件ステートメントの終わりに移動して、値を入力します。
5. **[メッセージ]** 列に、条件が False 評価された場合に表示するエラー メッセージを入力します。

   この列に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定をダブルクリックしてから省略記号ボタン (...) をクリックして、既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

   メモ: *[メッセージ] 列で指定したテキストを表示する Windows Installer ダイアログは、¥r (改行)、¥n (新しい行) または ¥t (タブ) 等のエスケープ シーケンスを認識しません。*
6. **[OK]** をクリックします。



重要: *InstallShield は、基本の条件検証を行いますが、条件ステートメントが予定通りの結果を評価することを確認してください。詳しい情報と条件のサンプルについては、「条件ステートメントのビルド」を参照してください。*

Conditional Statement Syntax (Windows Installer ヘルプ ライブラリ)

デフォルトの製品インストール先フォルダー (INSTALLDIR) の設定



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクトの INSTALLDIR プロパティは、製品に含まれるすべてのファイル用のデフォルト フォルダーとして機能します。その値の割り当て先は、Windows Installer フォルダーのプロパティである INSTALLDIR となり、これはデフォルト機能およびコンポーネントのインストール先フォルダーの役割を果たします。



Windows ロゴ: *Windows ロゴ要件によると、ターゲット システムの言語にかかわらず、製品のファイルのデフォルトのインストール先を Program Files のサブフォルダー、またはエンド ユーザー のアプリケーション データ フォルダーにする必要があります。ProgramFilesFolder を製品のインストール先フォルダー設定の親フォルダーとして使用すると、ファイルは正しい場所にインストールされます。INSTALLDIR 設定のデフォルト値は次の通りです:*

[ProgramFilesFolder]*会社名¥製品名*



タスク ***製品の INSTALLDIR プロパティを設定するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. ビルトイン Windows Installer ディレクトリをパスの一部として使用するには、**INSTALLDIR** 設定で、省略記号ボタン (...) をクリックします。**.[INSTALLDIR の設定]** ダイアログ ボックスが開きます。**[インストール先ディレクトリ]** ボックスで、インストール先のフォルダーを選択します。

   その代わりに、**INSTALLDIR** 設定に手動でパスを入力することもできます。



メモ: *[INSTALLDIR の設定] ダイアログ ボックスで新しいフォルダーを選択すると、INSTALLDIR 設定の値がオーバーライドされます。たとえば [ProgramFilesFolder]My Company¥Program のように、円記号でサブフォルダーを分割することによって、フォルダー プロパティのサブフォルダーを指定できます。*

その他のインストール先フォルダーについての考慮事項 (Windows Installer ベースのプロジェクトのみ)

機能の "リモート インストール" 設定が [ソースを優先] (または、サブ機能の親機能が [ソースを優先] に設定されている場合は [親を優先]) に設定すると、製品の INSTALLDIR プロパティに関わらず、機能のファイルはターゲット システムにインストールされません。

各機能およびコンポーネントには、"インストール先" 設定が含まれています。機能の "インストール先" 設定によって、製品の "インストール先フォルダー" 設定がオーバーライドされ、コンポーネントの "インストール先" 設定 によって、機能の "インストール先フォルダー" 設定がオーバーライドされます。したがって、製品のすべてのファイルをデフォルトで製品のインストール先フォルダーにインストールする場合は、すべての機能およびコンポーネントの "インストール先" 設定に **[INSTALLDIR]** を入力します。

INSTALLDIR などのインストールフォルダープロパティを使用している場合、*デフォルト*値を指定していることになります。エンド ユーザーは、Msiexec.exe 起動時に、コマンドラインでプロパティを設定したり、CustomSetup ダイアログで機能に対して新しいインストール先フォルダーを選択することにより、この値を変更できます。

ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護

InstallShield では、ロックダウン環境において製品を実行するエンド ユーザー向けに、ファイル、フォルダー、レジストリ キー、および Windows サービスを保護するためのいくつかの方法が提供されています:

- **従来型の Windows Installer 処理** — Windows Installer ベースのプロジェクトでは、実行時にファイル、フォルダー、およびレジストリ キーのアクセス許可を設定できるビルトイン Windows Installer サポートを使用できます。このオプションを使うと、InstallShield は .msi データベースの LockPermissions テーブルに製品のアクセス許可情報を格納します。

  この種類のアクセス許可処理を新しい Windows Installer 処理と組み合わせることはできません。MsiLockPermissionsEx テーブルおよび LockPermissions テーブルを含むリリースをビルドしようとすると、ビルド エラー −7207 が発生します。

- **新しい Windows Installer 処理** — Windows Installer ベースのプロジェクトでは、実行時にファイル、フォルダー、レジストリ キー、および Windows サービスのアクセス許可を設定できる最新の Windows Installer サポートを使用できます。このオプションを使うと、InstallShield は .msi データベースの MsiLockPermissionsEx テーブルに製品のアクセス許可情報を格納します。

  このオプションを使用するには、ターゲット システム上に Windows Installer 5 以降が必要です。以前のバージョンの Windows Installer は、この種類の処理に関する設定を無視します。

  この種類のアクセス許可処理を従来型の Windows Installer 処理と組み合わせることはできません。MsiLockPermissionsEx テーブルおよび LockPermissions テーブルを含むリリースをビルドしようとすると、ビルド エラー −7207 が発生します。

- **カスタム InstallShield 処理** — Windows Installer ベースのプロジェクトでは、実行時にアクセス許可を設定するためのカスタムサポートを選択できます。このオプションを使うと、InstallShield は .msi データベースのカスタム ISLockPermissions テーブルに製品のアクセス許可情報を格納します。InstallShield はまた、プロジェクトにカスタム アクションを追加します。

- **InstallScript 関数、SetObjectPermissions** — InstallScript イベントおよび InstallScript カスタム アクションで **SetObjectPermissions** 関数を使って、実行時にアクセス許可を設定できます。

これらの方法を使って、特定グループおよびユーザーにファイル、フォルダー、またはレジストリ キーのアクセス許可を割り当てることができます。たとえば、管理者グループに特定のファイルについての [読み取り]、[書き込み]、および [削除] アクセス許可を割り当てることができますが、別のグループのすべてにユーザーについては [読み取り] 許可のみ割り当てることができます。新しい [Windows Installer 処理] オプションを使うと、Windows サービスのアクセス許可を割り当てることもできます。

使用するオプションを決定する

次のテーブルでは、アクセス許可を設定するため様々な方法を比較します。

テーブル 1 • ロックダウン環境においてオブジェクトをセキュリティ保護する異なる方法の比較

比較カテゴリ	利用できるサポートについて
プロジェクトの種類	• **従来型の Windows Installer 処理、新しい Windows Instller 処理、およびカスタム InstallShield 処理** — 基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで利用できます。 • **SetObjectPermissions 関数** — InstallScript プロジェクトおよび InstallScript MSI プロジェクト タイプの InstallScript イベントで利用できます。 基本の MSI、DIM、InstallScript MSI、および マージ モジュール プロジェクトでは、InstallScript カスタム アクションを使って利用することもできます。
広く知られているセキュリティ識別子 (SID)	• **従来型の Windows Installer 処理** — 限られた数の SID (Administrators、Everyone) をサポートします。 • **新しい Windows Installer 処理、カスタム InstallShield 処理、および SetObjectPermissions 関数** — 多くの SID (Administrators、Authenticated Users、Creator Owner、Everyone、Guests、Interactive、Local Service、Local System、Network Service、Power Users、Remote Desktop Users、および Users) をサポートします。
翻訳された SID の名前	• **従来型の Windows Installer 処理** — 翻訳された SID の名前をサポートしません。翻訳された名前を使用すると、インストールは失敗します。 • **新しい Windows Installer 処理、カスタム InstallShield 処理、および SetObjectPermissions 関数** — サポートされている、よく知られる SID (Administrators、Authenticated Users、Creator Owner、Everyone、Guests、Interactive、Local Service、Local System、Network Service、Power Users、Remote Desktop Users、および Users) の翻訳されたすべての SID の名前をサポートします。
特定のアクセス許可を拒否できる機能	• **従来型の Windows Installer 処理** — サポートされていません。この処理を使って、特定のアクセス許可を設定できます。アクセス許可を拒否することはできません。したがって、ユーザーにファイルの読み取り専用アクセスを付与することができます。ただし、ユーザーが読み取り専用アクセスを所有することを防ぐことはできません。 • **新しい Windows Installer 処理、カスタム InstallShield 処理、および SetObjectPermissions 関数** — サポートします。これらのオプションを使って、指定するアクセス許可を特定のユーザーまたはグループが所持することを拒否するかどうかを指定できます。

テーブル 1・ロックダウン環境においてオブジェクトをセキュリティ保護する異なる方法の比較 (続き.)

比較カテゴリ	利用できるサポートについて
既存するアクセス許可に対する影響	• **従来型の Windows Installer 処理** — 既存のアクセス許可を削除することができます。たとえば、ターゲット システム上のフォルダーに Everyone ユーザーのアクセス許可が既に設定されている場合で、インストールが Administrators ユーザーのアクセス許可を設定するとき、このオプションを使って Administrators ユーザー用のアクセス許可を設定できます。ただし、既存する Everyone のアクセス許可は削除されます。 • **新しい Windows Installer 処理、カスタム InstallShield 処理、および SetObjectPermissions 関数** — これらのオプションを使って、ターゲット システムに既存するファイル、フォルダー、またはレジストリ キーに、アクセス許可を追加できます。このとき、そのオブジェクトに既存するアクセス許可は削除されません。たとえば、ターゲット システム上のフォルダーに Everyone ユーザーのアクセス許可が既に設定されている場合で、インストールが Administrators ユーザーのアクセス許可を設定するとき、これらのオプションを使って既存する Everyone ユーザーのアクセス許可を削除することなく、Administrators ユーザー用のアクセス許可を設定できます。
子オブジェクト (サブフォルダー、ファイル、およびサブキー) にアクセス許可を反映させる機能	• **従来型の Windows Installer 処理** — サポートされていません。フォルダー内のサブフォルダーまたはファイル (またはレジストリ キーの下にあるサブ キー) にアクセス許可を構成する場合、ターゲット システム上で作成された親は、自動的に子のアクセス許可を継承します。 • **新しい Windows Installer 処理、カスタム InstallShield 処理、および SetObjectPermissions 関数** — サポートします。これらのオプションを使って、フォルダー (またはレジストリ キー) のアクセス許可を構成し、そのフォルダーのサブフォルダーおよびファイル (またはレジストリ キーのサブキー) すべてに同じアクセス許可を適用するかどうかを指定できます。
インストールの一部としてインストールされないオブジェクトのアクセス許可を設定できる機能	• **従来型の Windows Installer 処理、新しい Windows Installer 処理、およびカスタム InstallShield 処理** — サポートしません。 • **SetObjectPermissions 関数** — サポートされています。インストールの一部としてインストールされた、またはターゲット システムに既存するファイル、フォルダー、またはレジストリ キーを、アクセス許可でセキュリティ保護することができます。
インストール中に作成された新しいユーザーに対するアクセス許可を設定できる機能	• **従来型の Windows Installer 処理** — サポートされていません。 • **新しい Windows Installer 処理、カスタム InstallShield 処理、および SetObjectPermissions 関数** — サポートします。インストール中に新しいユーザーが作成される場合、そのユーザーに対するアクセス許可を構成できます。

[カスタム InstallShield 処理] オプションまたは [従来型の Windows Installer 処理] オプションについての詳細

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトでは、カスタム InstallShield 処理または Windows Installer 処理のどちらを使用するかを指定しなくてはなりません。詳しくは、「プロジェクトで、ロックダウン環境でのアクセス許可タイプを選択する」をご覧ください。

いずれかのオプションを使ってファイルまたはフォルダーのアクセス許可を設定する方法については、「ファイルとフォルダーのアクセス許可を構成する」を参照してください。これらのオプションの 1 つを使ってレジストリ キーのアクセス許可を設定する方法については、「レジストリ キーのアクセス許可を構成する」を参照してください。

新しい [Windows Installer 処理] オプションについての詳細

サービスに新しい Windows Installer 処理オプションを使用するには、サービスをプロジェクトに追加してから、その設定を構成します。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。

ファイル、フォルダー、またはレジストリ キーに新しい Windows Installer 処理オプションを使用するには、[ダイレクト エディター] ビューの MsiLockPermissionsEx テーブルを使います。

InstallScript 関数、SetObjectPermissions についての詳細

SetObjectPermissions 関数についての詳細は、「SetObjectPermissions」を参照してください。

LockPermissions (Windows Installer ヘルプ ライブラリ)
MsiLockPermissionsEx (Windows Installer ヘルプ ライブラリ)

プロジェクトで、ロックダウン環境でのアクセス許可タイプを選択する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield には、ロックダウン環境にあるエンド ユーザー向けにファイル、フォルダー、およびレジストリ キーのアクセス許可をインストールがどのように構成するかを指定できる、プロジェクト全体に反映される設定があります。



タスク ***プロジェクトに対して、ロックダウン環境でのアクセス許可タイプを選択するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"ロックダウンの設定方法"** 設定で、適切なオプションを選択します:
   - **カスタム InstallShield 処理**—InstallShield は、プロジェクトにカスタム テーブルとカスタム アクションを追加して、ターゲット システム上のアクセス許可を設定します。これがデフォルトの値です。
   - **従来型の Windows Installer 処理** — InstallShield は、.msi データベースの LockPermissions テーブルを使って、製品のアクセス許可情報を格納します。

これらの 2 つのオプションについての詳細な比較情報は、「ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。

プロジェクトで次にファイル、フォルダー、またはレジストリ キーのアクセス許可を構成するとき、InstallShield は選択済みのロックダウン環境でのアクセス許可を使用します。

- [従来型の Windows Installer 処理] オプションを選択した場合、InstallShield はプロジェクトで LockPermissions テーブルを使用します。
- [カスタム InstallShield 処理] オプションを選択した場合、InstallShield はプロジェクトで ISLockPermissions テーブルを使用し、さらに ISLockPermissionsCost と ISLockPermissionsInstall カスタム アクションをプロジェクトに追加します。

"ロックダウンの設定方法" 設定の値を変更する時に、プロジェクトにファイル、フォルダー、またはレジストリ キーのアクセス許可が既に含まれている場合、InstallShield は適切なテーブルにアクセス許可データを移行するかどうかを問い合わせるメッセージ ボックスを表示します。データの移行を選択すると、選択済みのオプションに対応するテーブルにデータが移動されます。[カスタム InstallShield 処理] オプションから [従来型の Windows Installer 処理] オプションに切り替えると、プロジェクトから ISLockPermissionsCost と ISLockPermissionsInstall カスタム アクションも削除されます。

LockPermissions (Windows Installer ヘルプ ライブラリ)

Windows Installer インストールをログ記録するかどうかを指定する

InstallShield では、Windows Installer 4.0 以降がインストールのログ記録を行うかどうかを、プロジェクト全体にわたって指定することができます。また、ログ記録されるメッセージの種類をカスタマイズすることもできます。



タスク ***Windows Installer 4.0 以降で、プロジェクト全体のログ情報を指定するには、次の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **[MSI ログの作成]** 設定をクリックして、省略記号ボタン (...) ボタンをクリックします。**[Windows Installer 4.0 以降のログ記録オプション]** ダイアログ ボックスが開きます。
3. 適切なオプションを選択します。カスタム オプションを選択する場合、MsiLogging 値を入力します。

   MsiLogging 値の有効なパラメーターのリストは、Windows Installer ヘルプ ライブラリの MsiLogging Property を参照してください。
4. **[OK]** をクリックします。

結果は、[Windows Installer 4.0 以降のログ記録オプション] ダイアログ ボックスで行った選択によって異なります。

- [いいえ] を選択すると、ログ記録は行なわれません。これがデフォルトの値です。
- [はい] を選択すると、InstallShield は MsiLogging プロパティにデフォルト値である *voicewarmup* を挿入します。Windows Installer 4.0 が搭載されたターゲット システム上でインストールが実行されると、次の処理が行なわれます。
  - *voicewarmup* のデフォルト ログ モードに従って、インストーラーがログ ファイルを作成します。
  - インストーラーが、MsiLogFileLocation プロパティに、ログ ファイルのパスを挿入します。
  - SetupCompleteSuccess、SetupCompleteError および SetupInterrupted ダイアログに **[Windows Installer ログを表示]** チェック ボックスが追加されます。エンドユーザーがこのチェック ボックスを選択してから [終了] をクリックすると、テキスト ファイル ビューアーまたはエディターでログ ファイルが開きます。
- [カスタム] を選択すると、InstallShield は、このボックスに指定された値を MsiLogging プロパティに挿入します。Windows Installer 4.0 が搭載されたターゲット システム上でインストールが実行されると、次の処理が行なわれます。
  - このボックスに指定されたカスタム値に基づいて、インストーラーがログ ファイルを作成します。
  - インストーラーが、MsiLogFileLocation プロパティに、ログ ファイルのパスを挿入します。
  - SetupCompleteSuccess、SetupCompleteError および SetupInterrupted ダイアログに **[Windows Installer ログを表示]** チェック ボックスが追加されます。エンドユーザーがこのチェック ボックスを選択してから [終了] をクリックすると、テキスト ファイル ビューアーまたはエディターでログ ファイルが開きます。

以前のバージョンの Windows Installer は MsiLogging 設定を無視します。以前のバージョンの Windows Installer を実行中のシステム上で表示される実行時ダイアログでは、**[Windows Installer ログを表示]** チェック ボックスは表示されません。



重要: *MsiLogFileLocation プロパティは、読み取り専用のため、ログ ファイルの場所の設定および変更には使用できません。*

InstallScript インストールを複数回実行する

InstallScript プロジェクトの種類には、ターゲットマシンに製品がインストールされている場合、およびエンドユーザーがインストールを再試行した場合のインストール動作に関するいくつかのオプションがあります。

テーブル 2 ・ メンテナンス エクスペリエンスの設定

メンテナンス エクスペリエンス	メンテナンス UI 表示のタイミング	[プログラムの追加と削除] で作成されるエントリの数
標準	インストールを再試行した時	1
複数インスタンス	インストールが [プログラムの追加と削除] を通して実行された時	複数—各インストールごとに個別のエントリ
アンインストールまたはメンテナンスはなし	チェックしない	なし

これらのオプションは、[一般情報] ビューの "メンテナンス エクスペリエンス" 設定から選択できます。

デフォルトの動作 (標準オプション) では、エンドユーザーがマシン上でインストールを再実行すると、メンテナンス ユーザー インターフェイスが表示されます。

複数インスタンスのオプションを使うと、エンドユーザーはインストールをメンテナンスインストールではなく初回インストールとして複数回にわたって再試行することが可能となりす。このオプションを選択すると、デフォルトで、エンドユーザーがインストール実行ごとに異なる場所に製品をインストールすることができるようになります。インストールを初回インストールとして実行するたびに、直接選択またはセットアップの種類の選択に関わらず、選択されたコンポーネントが毎回インストールされます。

マシン上に製品の 1 つ以上のインスタンスが存在するとき、エンドユーザーがユーザー インターフェイスを使って複数インスタンス インストールを再実行した場合、インストールは [正規の製品が検出されました] ダイアログ ボックスを表示します。このダイアログでは、エンドユーザーはアップデートを適用するインスタンスを選択することができます。ただし、エンドユーザーがインストールをサイレントで再実行した場合、インストールは [正規の製品が検出されました] ダイアログ ボックスを抑制して、マシン上に新しいインスタンスを作成します。



プロジェクト: *基本の MSI プロジェクトにおける複数インスタンスのサポートに関する詳細は、「製品の複数のインスタンスをインストールする」を参照してください。*

"メンテナンスの有効化" 設定を構成する



プロジェクト: *この情報は、InstallScript MSI プロジェクトに適用します。*

[一般情報] ビューにある "ンテナンスを有効にする" 設定を使って、製品が既にインストールされているシステム上で、エンド ユーザーがインストールを再度実行したときに、完全なメンテナンス ユーザー インターフェイス (UI) またはアンインストール UI を表示するかどうかを指定します。

次のテーブルは、"メンテナンスの有効化" 設定で選択できる設定です。

テーブル 3 ・ "メンテナンスの有効化" の設定のオプション

オプション	説明
はい	/removeonly コマンドライン パラメーターが Setup.exe に渡されない限り、エンド ユーザーがインストールを再度実行したとき、システム変数 REMOVEONLY は FALSE に設定され、標準メンテナンス UI が表示されます。

テーブル 3 - "メンテナンスの有効化" の設定のオプション (続き.)

オプション	説明
いいえ	エンド ユーザーがインストールを再度実行したとき、システム変数 REMOVEONLY は TRUE に設定され、アンインストール UI が表示されます。  ***メモ-*** *"ンテナンスを有効にする" 設定に [いいえ] を選択すると、OnUninstall イベント ハンドラの呼び出しが行われません。OnUninstall イベント ハンドラの呼び出しを行う場合は、*Setup.exe *に /*uninst *コマンドライン パラメーターを使用しなくてはなりません。これは、InstallScript UI スタイルが InstallScript エンジンを外部 UI ハンドラとして使用する従来型のスタイルの場合にのみ適用します。InstallScript UI スタイル が InstallScript エンジンを埋め込み UI ハンドラとして使用する新しいスタイルの場合、/*uninst *コマンドライン パラメーターは、サポートされていません。詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。*



メモ- *InstallScript MSI インストールでは、デフォルトで、コントロール パネルの [プログラムの追加と削除] で [変更] ボタンと [削除] ボタンが別々に表示されます。[変更] ボタンをクリックするとメンテナンス UI が、また [削除] ボタンをクリックするとアンインストール UI が常に表示されます。*

"メンテナンスの有効化" 設定は、エンド ユーザーが [プログラムの追加と削除] で製品のエントリに表示されている [変更] ボタンをクリックして製品のメンテナンスを開始した場合の動作には、効果を持ちません。

エンド ユーザーが [プログラムの追加と削除] 内からメンテナンスを実行できないようにするには、[一般情報] ビューの "変更ボタンを無効にする" 設定で [はい] を選択します。

InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い



プロジェクト- *この情報は、InstallScript MSI プロジェクトに適用します。*

InstallScript MSI インストールは 2 つの異なるインストーラー エンジンを使用します:

- Windows Instaler エンジンは、.msi パッケージの標準 [実行] シーケンスを実行します。これは、通常ターゲット システムを変更するシーケンスです。
- InstallScript エンジンは、インストールのカスタム ユーザー インターフェイス (UI) ハンドラーとしての役割を果します。また、InstallScript コードを実行します。UI に InstallScript エンジンを使用する際の利点は、高度にカスタマイズされたランタイム ダイアログのサポートが提供されていることです。

従来、Windows Installer では、外部カスタム UI ハンドラーのみがサポートされていたため、InstallScript MSI インストールでは、常に Setup.exe セットアップランチャーを使用する必要がありました。セットアップランチャーはブートストラップ アプリケーションとして機能し、UI を表示する InstallScript エンジンを開始して InstallScript コードを実行し、また Windows Installer を開始して .msi パッケージの [実行] シーケンスを実行します。

Windows Installer 4.5 では、埋め込みカスタム UI ハンドラーのサポートが追加されました。InstallScript エンジンが .msi パッケージ内に埋め込まれている場合、InstallScript MSI インストールでは、Setup.exe セットアップランチャーが必要なくなり、エンドユーザーは .msi パッケージを起動してインストールを開始することができます。このケースでは、.msi パッケージに、インストールの UI で InstallScript エンジンを起動するにあたって Windows Installer が知る必要がある情報が含まれています。

したがって、InstallShield では、InstallScript MSI インストールのUI に、次の 2 つの異なるスタイルが提供されています:

- 従来型スタイル (Setup.exe セットアップランチャーが必要) – InstallScript MSI インストールで、InstallScript エンジンを外部 UI ハンドラーとして使用することができます。これがデフォルトのスタイルになります。
- 新しいスタイル (ターゲット マシンで Windows Installer 4.5 が必要) – InstallScript MSI インストールで、InstallScript エンジンを埋め込み UI ハンドラーとして使用することができます。このスタイルでは、InstallShield が InstallScript エンジンを .msi パッケージ内に埋め込みます。このオプションには、いくつかの制限事項があります。

次のセクションは、これらの 2 つのスタイルに関する詳しい情報です。これらの情報を参考にして、ニースに最も適したスタイルを判別してください。

従来型スタイル (外部 UI ハンドラーとしての InstallScript エンジン)

初回インストールにおける従来型スタイルの InstallScript MSI パッケージの一般的な流れは、次の通りです:

1. InstallScript MSI が、.msi パッケージを開きます。
2. ExecuteAction を除く、[インストール]– UI シーケンスにあるすべてのアクションが実行されます。アクションは昇順に実行されます。
3. InstallScript エンジンは、自分で内部コンポーネントのコスト計算を行って、インストール内にあるすべての機能とコンポーネントに必要な空き容量を判別します。
4. InstallScript エンジンが、ISSetup.dll に含まれているコンパイル済みのスクリプトを起動します。.msi パッケージのインストールの前に実行されるイベントのイベント シーケンスは、次の通りです:
   a. OnBegin
   b. O*nXxx*UIBefore (Xxx は、First、Maint、Admin、Patch、Resume などが入ります)
   c. OnGeneratingMsiScript
   d. OnMoving
   e. OnFeaturesInstalling (機能の installing と uninstalling イベントは、すべてこの時点で実行されます。)
   f. OnInstallFilesActionBefore
   g. OnGeneratedMsiScript
5. InstallScript エンジンは、**MsiInstallProduct** を通して .msi パッケージ インストールを開きます。次の要因によって、InstallScript エンジンが **MsiInstallProduct** を通して渡すコマンドラインが判別されます: インストール モード (例、初回インストール、メンテナンス モード、マイナー アップグレード、パッチなど)、内部機能の選択、および現在のプロパティの値。
6. .msi パッケージ インストールが成功した場合、InstallScript エンジンは、セカンダリ アンインストール キー (InstallShield_{ProductCode}) をマシンに書き込みし、次のイベントを起動します:
   a. OnInstallFilesActionAfter

b. OnFeaturesInstalled（機能の installed と uninstalled イベントはすべてこの時点で実行されます。）

c. OnMoved

d. On*Xxx*UIAfter *(Xxx* は、First、Maint、Admin、Patch、Resume などが入ります）

e. OnEnd

7. スクリプトが完了すると、InstallScript エンジンは InstallScript ログをマシンに書き込みします。

8. InstallScript エンジンがシャットダウンします。

新しいスタイル（埋め込み UI ハンドラーとしての InstallScript エンジン）

InstallScript MSI の埋め込み UI のサポートは、従来型の InstallScript MSI インストールと同じ一般的なフローを従おうとします。ただし、一部のフローは、発生しないか、または、必要ありません。初回インストールにおける新しいスタイルの InstallScript MSI パッケージの一般的な流れは、次の通りです:

1. .msi パッケージは、直接、または Setup.exe（Setup.exe は、このあと、基本の MSI インストールで同じ動作をする Msiexec.exe を起動します）によって起動されます。

2. Windows Installer エンジンによって、MsiEmbeddedUI テーブルにあるすべてのファイルが抽出され、DLL 内の埋め込み UI ハンドラーの属性を含む "埋め込み UI の初期化" 関数を呼び出します。（InstallScript MSI ケースでは、これは ISSetup.dll です。）

3. InstallScript エンジンは、ISSetupFilesExtract アクションが存在し、InstallUISequence で条件が True に評価された場合、そのアクションを初期化し、手動で起動します。

4. InstallScript エンジンは、Directory テーブル内にあるすべてのエントリの解決を手動で解決します。

5. InstallScript エンジンは、自分で内部コンポーネントのコスト計算を行って、インストール内にあるすべての機能とコンポーネントに必要な空き容量を判別します。

6. InstallScript エンジンが、ISSetup.dll に含まれているコンパイル済みのスクリプトを起動します。.msi パッケージのインストールの前に実行されるイベントのイベント シーケンスは、次の通りです:

   a. OnBegin

   b. O*nXxx*UIBefore *(Xxx* は、First、Maint、Admin、Patch、Resume などが入ります）

   c. OnGeneratingMsiScript

   d. OnMoving

   e. OnFeaturesInstalling（機能の installing と uninstalling イベントは、すべてこの時点で実行されます。）

   f. OnInstallFilesActionBefore

   g. OnGeneratedMsiScript

7. InstallScript エンジンは、コントロールを Windows Installer エンジンに返します。そのあと、Windows Installer エンジンは、インストールの［実行］シーケンスの実行を開始します。

8. .msi パッケージ インストールが成功した場合、Windows Installer エンジンは ISSetup.dll に呼び戻します。そのあと、ISSetup.dll によって、次のイベントが起動されます。

   a. OnInstallFilesActionAfter

   b. OnFeaturesInstalled（機能の installed と uninstalled イベントはすべてこの時点で実行されます。）

   c. OnMoved

d. On*Xxx*UIAfter *(Xxx* は、First、Maint、Admin、Patch、Resume などが入ります)

e. OnEnd

9. InstallScript エンジンがシャットダウンし、コントロールを Windows Installer に返します。

新しいスタイルの InstallScript MSI インストールでは、[インストール]- UI シーケンスは実行されません。[インストール]- UI シーケンスにシーケンスされているカスタム アクションがある場合、これらのアクションは、/qb または /qn と共に起動された基本の MSI と同じ理由で、実行されません。

InstallScript エンジンは、InstallScript イベントからシステムに行われた変更をログ記録しません。

従来型の InstallScript MSI インストールでサポートされているほとんどのコマンドライン パラメーターは、新しいスタイルの InstallScript MSI インストールでもサポートされています。エンドユーザーが .msi パッケージを直接実行した場合、これらのパラメーターを ISSCRIPTCMDLINE プロパティに渡すことができません。

従来型スタイル (外部 UI ハンドラーとしての InstallScript エンジン) の制限

アンインストール可能なパッチ サポート

従来型のスタイルは、アンインストール可能なパッチを作成するためのサポートを含みません。

新しいスタイル (埋め込み UI ハンドラーとしての InstallScript エンジン) の制限事項と既知の問題

新しいスタイルを検討している場合、次の点に注意してください。

Windows Installer 4.5 の要件

新しいスタイルには、ターゲット マシン上に Windows Installer 4.5 が必要です。存在していないとき、Windows Installer 4.5 がインストールされるように、Windows Installer 4.5 の InstallShield 前提条件をプロジェクトに追加することができます。InstallShield 前提条件をインストールに含めると、Setup.exe セットアップランチャーを使用する必要があります。詳細については、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。Windows Installer 4.5 は、Windows の初期のバージョンの一部にはインストールできませんので注意してください。

実行時に Windows Installer 4.5 がターゲット システムにインストールされていなく、かつ、インストールが Setup.exe で起動された場合、エラー 1713 が表示され、インストールの実行に Windows Installer 4.5 が必要であることが通知されます。インストールは、エンドユーザーがこのエラー メッセージを閉じたあと中止されます。.msi パッケージが直接起動された場合、パッケージが Setup.exe ファイルから起動する必要があることを通知するエラーが表示されます。

アップグレードとパッチのサポート

InstallShield には、アップグレードが従来型のスタイルでインストールされた製品を更新する場合、新しいスタイルを使用するアップグレードを作成するためのビルトイン サポートは含まれていません。したがって、従来型スタイルを使って、製品のリリースをビルドしてから、後でアップグレードを作成する場合、そのアップグレードに対して従来型のスタイルを使用することを検討してください。これは、全ての種類のアップグレード (メジャー アップグレード、マイナー アップグレード、スモール アップデート) に適用します。

パッチがターゲットとする以前のセットアップが従来型スタイルを使用している場合、パッチも従来型スタイルを使用しなくてはなりません。また、QuickPatch プロジェクトは、元のインストールの UI スタイルを保持します。つまり、元のインストールが従来型スタイルを使用している場合、QuickPatch も従来型スタイルを使用します。元のインストールが新しいスタイルを使用している場合、QuickPatch も新しいスタイルを使用します。

メジャー アップグレード シナリオの回避策として、新しいスタイルのメジャー アップグレード インストールに、レジストリから以前の製品のアンインストール文字列を読み込ませ、新しいバージョンをインストールする前に、以前のバージョンの製品をアンインストールする InstallScript 関数 LaunchApplication を使ってそのインストールを起動することもできます。場合により、スクリプトで、**LaunchApplication** を使用する前に、LAAW_SHELLEXECUTEVERB を **runas** に設定する必要があります。

MsiDoAction

.msi パッケージのインストールの前に実行されるイベントの 1 つから **MsiDoAction** 関数を呼び出そうとした場合、エラーが返されます。このエラーは、Windows Installer が InstallScript エンジンに渡すハンドラーが原因で発生します。ハンドルは、.msi パッケージ内にある標準およびカスタム アクションを実行できません。この関数は、必要な場合、InstallScript カスタム アクションから関数を呼び出すことができます。

MsiGetTargetPath と MsiSetTargetPath

Windows Installer の **MsiGetTargetPath** と **MsiSetTargetPath** 関数は、これらがイベント ドリブン スクリプトにある場合、新しいスタイルで適切に動作しません。これらの関数は、Windows Installer のコスト計算アクションによって初期化されるディレクトリ マネージャーに依存します。新しいスタイルの InstallScript MSI インストールでは、Windows Installer のコスト計算は実行されません。したがって、**MsiGetTargetPath** はパス情報を返すことができず、**MsiSetTargetPath** も要求されたターゲット パスを設定することができません。両方のケースで、これらの関数が呼び出された場合、Windows Installer はメッセージを詳細ログにログ記録します。

インストール パスを更新するには、FeatureSetTarget を使用します。

MsiGetTargetPath と **MsiSetTargetPath** は、[インストール]–[実行] シーケンスで CostFinalize の後にシーケンスされた InstallScript カスタム アクションを通して呼び出すことができます。

FeatureTransferData と ComponentTransferData

InstallScript 関数 **FeatureTransferData** と **ComponentTransferData** は、InstallScript MSI インストールの新しいスタイルでは定義されていないため、使用できません。

programÉendprogram

新しいスタイルの InstallScript MSI インストールでは、手続きのスクリプト (programÉendprogram ブロックがあるスクリプト) を使用できません。このケースでは、プログラム関数は呼び出されません。イベント ドリブン スクリプトは、インストールの動作をカスタマイズするために使用します。

/uninst

新しいスタイルの InstallScript MSI インストールでは、/uninst コマンドライン パラメーターはサポートされていません。

BATCH_INSTALL

BATCH_INSTALL 変数を設定して再起動を試みても、正しく動作しません。再起動を実行するには、ScheduleReboot アクション または REBOOT プロパティを利用します。

Reboot のサポート

新しいスタイルの InstallScript MSI インストールでの再起動は、InstallScript ベースのインストールではなく、基本の MSI インストールと同じ要領で処理されます (InstallScript ベースのインストールでは、再起動のあと、インストールが続行し、**OnRebooted** イベント ハンドラーが呼び出されます)。

新しいスタイルの InstallScript MSI インストールは、基本の MSI インストールと同じ要領で再起動を処理するように作成されます。つまり、インストールは、スケジュール済みの再起動が発生した後実行されず、**OnRebooted** イベント ハンドラーが呼び出されません。

Msiexec.exe /x (for Uninstallation)

エンドユーザーが、Msiexec.exe /x {ProductCode} ステートメントを使ってコマンドラインから製品をアンインストールしようと試みたとき、アンインストールが成功することがあります。ただし、アンインストールの終わりのほうで、Windows Installer によってエラー ダイアログが表示されます。エラー ダイアログでは、Windows Installer サービスにアクセスできなかったというメッセージが表示されます。

コマンドラインからアンインストールを実行する場合、次のいずれかのメソッドを実行することが現在推奨されています:

msiexec.exe /i {ProductCode} REMOVE=ALL

msiexec.exe /x {ProductCode} /qn

これらの例では、Windows Installer エラーはトリガーされません。

推奨事項

新しいスタイルの InstallScript MSI インストールでは、UI 機能を管理者権限なしに実行することができます。したがって、次の点が推奨されます (および、一般的に、すべての Windows Installer ベースのインストールに適用します):

- 管理者権限が、InstallScript イベントから提供されると仮定することはできません。
- システムに行う必要がある変更は、[インストール]-[実行] シーケンスでカスタム アクションで行う必要があります。InstallScript カスタムアクションを使用することができます。権限が不足しないように、カスタム アクションは、システム コンテキストで "スクリプト内実行" 設定が遅延実行になっている必要があります。
- システムを変更するカスタム アクションには、インストールが失敗した場合、システムを以前の状態に復元する対応するロールバック アクションが必要です。
- システムを変更するカスタム アクションにも、システムからインストールされているアイテムを削除するためにアンインストールで実行する適切なアクションが必要です。
- InstallScript イベントから変更がシステムに行われた場合、変更は InstallScript エンジンによってログされないため、アンインストールでこれらのリソースをクリーンアップする追加のスクリプト コードが必要になります。

InstallScript を使って、実行時に 2 つのスタイルを識別する

INSTALLSCRIPTMSIEEUI 変数を利用して、異なる UI スタイルに異なる実行時の動作を生成する単一スクリプトを作成することができます。詳しくは、「INSTALLSCRIPTMSIEEUI」を参照してください。

MsiDoAction (Windows Installer ヘルプ ライブラリ)
MsiGetTargetPath (Windows Installer ヘルプ ライブラリ)
MsiSetTargetPath (Windows Installer ヘルプ ライブラリ)

InstallScript MSI インストールの InstallScript ユーザー インターフェイス タイプを指定する



プロジェクト: *この情報は、InstallScript MSI プロジェクトに適用します。*

InstallScript プロジェクトには、2 つの異なる種類の InstallScript ユーザー インターフェイス (UI) があります。制限事項を含む、これら 2 つのスタイルに関する詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。



タスク

InstallScript MSI プロジェクトの InstallScript ユーザー インターフェイス タイプを指定するには、以下の手順に従います:

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"InstallScript ユーザー インターフェイスのタイプ"** 設定で、適切なオプションを選択します。
   - **従来のスタイル (Setup.exe が必要)** – InstallScript エンジンを InstallScript MSI インストールの 外部 UI ハンドラーとして使用する場合は、このオプションを選択します。このスタイルを使用する場合は、インストールに Setup.exe セットアップランチャーを必ず含めてください。セットアップランチャーはブートストラップ アプリケーションとして機能し、UI を表示する InstallScript エンジンを開始して InstallScript コードを実行し、また Windows Installer を開始して .msi パッケージの [実行] シーケンスを実行します。

     この設定では、このオプションがデフォルトで設定されています。
   - **新しいスタイル (Windows Installer 4.5 が必要)** – InstallScript エンジンを InstallScript MSI インストールの埋め込み UI ハンドラーとして使用する場合は、このオプションを選択します。このスタイルでは、InstallShield が InstallScript エンジンを .msi パッケージ内に埋め込みます。Windows Installer は InstallScript エンジンを呼び出して UI を表示します。Windows Instaler はまた、.msi パッケージの [実行] シーケンスを実行します。

     このオプションには、ターゲット マシン上に Windows Installer 4.5 が必要です。また、いくつかの制限事項があるため、このスタイルを使用する場合は綿密な計画が必要です。

概要情報ストリーム データを入力する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

概要情報ストリームの設定の一部は、DIM プロジェクトで使用できます。このプロジェクト タイプでは、これら設定の値は実行時に使用または表示されません。このプロジェクト タイプにおけるこれらの設定は、プロジェクトを内部的に識別するためのリファレンスとして使用できます。

Windows Installer データベース (.msi ファイルと .msm ファイル) は COM 構造化ストレージとして実装されており、通常 COM 構造化ストレージ ファイルには [概要情報ストリーム] が入っています。概要情報ストリームには、会社や、インストール中のソフトウェアに関する情報が含まれます。ファイルの概要情報の表示方法についての詳細は、「[概要情報ストリーム パネル] へのアクセス」を参照してください。

概要情報ストリームの各設定の説明については、「[一般情報] ビュー」を参照してください。

[概要情報ストリーム パネル] へのアクセス



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*



タスク ***.msi ファイルまたは .msm ファイルの概要情報にアクセスするには、以下の手順をに従います:***

1. .msi ファイルまたは .msm ファイルを右クリックしてから、**[プロパティ]** を選選択します。**[プロパティ]** ダイアログ ボックスが開きます。
2. [**概要**] タブをクリックして、概要情報を表示します。



ヒント: *[一般情報] ビューで、インストールの [プロパティ] ダイアログ ボックスに表示する情報を入力します。*

概要情報ストリームのプロパティを設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

- *トランスフォーム*



タスク ***プロジェクトの概要情報を指定するには、次の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. ビュー内で **[概要情報ストリーム]** 領域に移動します。
3. 必要に応じて設定の値を変更します。

ビルド時の概要情報ストリームのコメントを設定する

ビルド時にコメントを入力できるオプションもあります。MSBuild.exe タスクの SummaryInfoComments パラメーターを使って、次の例のようにビルド番号を含めるなど、ビルド時に **[概要情報ストリーム コメント]** を設定することができます。

MSBuild.exe c:¥installers¥Setup.sln /Property:SummaryInfoComments="ここにコメントを入力"

"テンプレート概要" プロパティを使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[一般情報] ビューにある "テンプレートの概要" 設定を使って、プロセッサの種類と言語の基本的な条件を設定できます。ターゲット システムがこの要件を満たさない場合、インストーラーがエラー メッセージを表示してインストールを終了します。



ヒント: *[リリース] ビューで、製品構成について "テンプレートの概要" 設定を構成することもできます。[リリース] ビューに入力された値は、[一般情報] ビューに設定された値を上書きします。*

構文

プロセッサの種類を最初にリストしてから、セットアップのデフォルト言語に続きます。言語に 4 桁の言語 ID か、言語に中立にするにはゼロ (0) を入力します。

- 2 つのカテゴリは、間にセミコロンを入れて区切ります。
- インストールが複数の製品をサポートする場合で、言語カテゴリーに複数のエントリ (たとえば英語は 1033、ドイツ語は 1031) を含めるには、それらをコンマで区切ります。



注意・*インストールが 64 ビットマシンを対象にしている場合、Intel と Intel64 の両方をこのフィールドに入力しないでください。両方を入力するとインストール エラーが発生します。さらに、Intel64 を入力した場合、インストールは 32 ビットオペレーティング システムでは作動しません。*

有効なプロセッサ値は、次のとおりです:

- Intel
- Intel64
- x64

プロセッサタイプと言語に基づいた条件の設定

インストールが Intel プロセッサを持つ英語システムでのみ実行する場合、"テンプレートの概要" プロパティで次の構文を使用します。

Intel;1033

プロセッサの種類の条件が一致しないと、インストーラーはエラーメッセージを表示して終了します。

製品が x64 プロセッサを搭載した英語およびドイツ語システム上で実行する場合は、次のように入力します:

x64;1033,1031

パッケージを言語非依存に指定する場合、**Intel;** または **Intel;0** という構文を使用します。

言語のみに基づいた条件の設定

言語のみに基づいて条件を設定するには、セミコロンを入力して、セットアップのデフォルトの言語を指定します。

;1036

最初の部分が空の場合、ターゲット プロセッサが Intel であることを意味します。言語が指定されていない場合、InstallShield は [概要情報ストリーム] の リリース言語を提供します。

Template Summary Property (Windows Installer ヘルプ ライブラリ)

[プログラムの追加と削除] 情報を構成する

[一般情報] ビューの [プログラムの追加と削除] 領域にある設定は、[プログラムの追加と削除] ツールに表示される情報です。

特定の "プログラムの追加と削除" 設定に値を入力しなかった場合、サンプルの値が灰色テキストで表示されます。この値は例として表示されているため、最終的に Windows Installer データベースには含まれません。

[一般情報] ビューを使うと、[プログラムの追加と削除] を通じてエンド ユーザーに表示される情報の多くを設定することができます。



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*



プロジェクト: *基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合 – 指定したほとんどの値は、ARP で始まる名前 (例、ARPPRODUCTICON または ARPHELPLINK など) を持つ Windows Installer のプロパティに格納されます。唯一の例外は "発行元" 設定です。この値は会社名と共に設定され、[製造業者] プロパティに格納されます。*

基本の MSI プロジェクトの場合 – 製品が [プログラムの追加と削除] に表示されないようにするには、プロパティ マネージャーで Windows Installer プロパティ ARPSYSTEMCOMPONENT *を 1 に設定します。このプロパティを設定すると、[プログラムの追加と削除] に製品を表示しないように抑制するだけです。エンド ユーザーは、コマンド ラインを使ってインストールをメンテナンス モードで実行することで、製品を削除することができます。*

InstallScript MSI プロジェクトの場合ー製品が [プログラムの追加と削除] に表示されないようにするには、[リリース] ビューの "[追加/削除] パネルエントリを隠す" 設定で [はい] を選択します。

Readme ファイルを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

[一般情報] ビューの README 設定では、製品の Readme ファイルの名前を入力します。Windows の一部のバージョンでは、この情報は、[プログラムの追加と削除] で製品のエントリ用の [サポート情報] ダイアログ ボックスで表示されます。また、有効な URL を指定して、インターネット上の Readme ファイルのリンクを表示することもできます。



プロジェクト: *Windows Installer ベースのプロジェクト (基本の MSI または InstallScript MSI プロジェクト) では、Readme ファイルをインストールの一部としてインストールする時に、パスを適切に表示するための特別なフォーマットが必要です。次のセクションを参照してください。InstallScript プロジェクトまたは InstallScript オブジェクト プロジェクトでは、ハードコード化されたパスまたは URLを入力するか、山かっこで括ったシステム変数の相対パス (たとえば、<TARGETDIR>¥Readme.txt) を指定します。このデータはデフォルトの OnMoveData イベント ハンドラー関数によってターゲット システムのレジストリに書き込まれます。*

基本の MSI および InstallScript MSI プロジェクトで Readme ファイル パスを表示する

Readme ファイルのパスを直接入力するか、または URL を入力します。ただし、[MyDirectory] のようなフォルダー名、または [$MyComponent] のようなコンポーネント名は解決されません。つまり、[サポート情報] ダイアログ ボックスの Readme 値が [INSTALLDIR]Readme.txt や [$MyComp]Readme.htm のように表示されます。このプロパティは Windows Installer プロパティ ARPREADME の初期値を単に設定するだけなので、パスは解決されません。したがって、書式設定された文字列が Windows Installer プロパティで解決されることはありません。

InstallShield の解決法はカスタム アクションを使用してARPREADME を README 設定 に入力した値で設定することです。SetARPReadme カスタム アクションで値を設定することにより、パスはインストール時に解決されます。たとえば、Readme.doc がコンポーネント Help_Files のファイルと仮定します。この場合、README 設定に **[$Help_Files]Readme.doc** を入力します。Help_Files が C:¥Program Files¥MyCompany¥MyProduct¥Help にインストールされると仮定した場合、ターゲット システムの [プログラムの追加と削除] で表示されるパスは、C:¥Program Files¥MyCompany¥MyProduct¥Help¥Readme.doc となります。

コンポーネント名の代わりに、ファイルキーを使用することもできます。実行時に解決できないため、Directory テーブルからフォルダーを使用しないでください。ファイルキーはコンポーネント名ほど信頼できません。これは、ファイルが 動的にリンクされている、またはリリースでパッチの最適化を使用している場合、ファイル名が変更される可能性があるからです。上記の例を続けると、Readme.doc が F501_Readme.doc のファイルキーを持っていれば、README 設定に **[#F501_Readme.doc]** と入力します。.同じフォルダーにインストールされた場合、ターゲット システム上で [プログラムの追加と削除] は次のパスを表示します:

C:¥Program Files¥MyCompany¥MyProduct¥Help¥Readme.doc

カスタム アクション SetARPReadme は、[一般情報] ビューの README 設定に文字列があり、[インストール] – [実行] シーケンスに CostFinalize アクションが含まれている場合に、毎回追加されます。SetARPReadme は、[インストール] の [実行] シーケンスと [ユーザー インターフェイス] シーケンスで CostFinalize のすぐ後に挿入されます。

製品のソフトウェア識別タグを含める



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

ISO/IEC 19770-2 は、ソフトウェア識別タグを作成するための国際規格です。ソフトウェア識別タグは、製品名、製品エディション、パブリッシャーなど、ソフトウェアに関する説明的な情報を含むサイズが小さい XML ベースのファイルです。ソフトウェア資産管理ツールは、企業でインストールされているソフトウェアについての正確なアプリケーション ID を提供する目的で、タグ内のデータを収集するツールです。

ソフトウェア識別タグ機能は、業界標準として現在進化しつつあるもので、この機能により、独立系ソフトウェア ベンダーは、顧客に対して、ソフトウェア資産管理およびライセンス最適化イニシアチブに有用な、より適切な情報を提供することができる、より洗練されたアプリケーションを作ることができるようになります。製品のインストール パッケージに識別タグを持たせることで、顧客は、インストールした製品の内部的使用状況を監視できるツールが使えるようになります。これにより、顧客が入手済み製品のライセンスの数を把握、管理および最適化することができるようになります。

適切にタグを作成するためには、[一般情報] ビューにある ″製品名″ や ″製品バージョン″ 設定といった基本設定を構成する必要があります。また、[一般情報] ビューにある識別タグ関連の設定も構成しなくてはなりません。



タスク ***インストールにソフトウェア識別タグを含めるには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. このビュー内の **[ソフトウェア識別タグ]** 領域で、必要に応じて設定の値を変更します。

   "ソフトウェア識別タグの使用" 設定を使って、インストールにタグを含めるかどうかを指定できます。デフォルト値の **[はい]** を選択してから、必要に応じて **[ソフトウェア識別タグ]** 領域にあるその他の設定を構成します。

プロジェクトでタグ機能を使用する場合、InstallShield が作成する 2 つの新しいコンポーネントにタグが追加され、これらのコンポーネントが製品機能の 1 つに関連付けられます。コンポーネントは、以下のとおりです:

- INSTALLDIR をインストール先とする ISO19770_LocalTag
- CommonAppDataFolder をインストール先とする ISO19770_SystemTag

これらのコンポーネントをプロジェクト内の別の機能に関連付ける場合は、[セットアップのデザイン] ビューを使います。詳細については、「コンポーネントと機能の関連付け」を参照してください。

ビルド時に以下の条件が True 評価された場合、InstallShield がビルドするインストールにソフトウェア識別タグが追加されます:

- [一般情報] ビューの "ソフトウェア識別タグの使用" 設定に、デフォルト値の [はい] が選択されている。
- [一般情報] ビューの "一意な ID""タグ作成者"、および "タグ作成者 ID" 設定に値が設定されている。

タグ機能が有効である場合に、前述の 3 つのタグ識別設定のうち 1 つでも値が不足している場合、InstallShield はビルド警告を生成して、リリースにタグが含まれなかったことを通知します。この警告を解決するためには、必要に応じて [一般情報] ビューの [ソフトウェア識別タグ] 領域にある設定を構成してください。

プロジェクトにソフトウェア ID タグを含めて [リリース] ビューで .pfx ファイルを使ってリリースに署名を行うことを設定すると、InstallShield がビルド時にタグをデジタル署名します。タグ ファイルに署名するためには、.NET Framework 3.5 以降をビルドマシンにインストールする必要があります。

製品の Microsoft System Center Configuration Manager アプリケーション モデル データを含める



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

Microsoft System Center 2012 Configuration Manager は、IT 管理者が、エンタープライズ全体において、効率良くアプリケーションを管理および配信できるようにするためのソリューションです。ユーザー側に重点を置いたアプローチがとられているこのソリューションでは、それぞれのアプリケーションを、必要な依存アプリケーションと共に、最も適切なフォーマットで様々なデバイス上に配信できるように定義することが可能です。

配置メタデータを正確に識別することは、アプリケーションを Configuration Manager アプリケーション モデルに移行する上で必須です。InstallShield では、ソフトウェア識別タグを使って、アプリケーションのアプリケーション モデル メタデータの一部を指定できる機能が提供されています。AdminStudio のユーザーが AdminStudio のアプリケーション カタログにパッケージをインポートしたとき、パッケージの要素が解析されて、検出方法、依存

関係、要件などの配置データ、およびソフトウェア識別タグ内の情報が確認されます。AdminStudio では、この情報を Microsoft System Center 2012 Configuration Manager に公開する前に、確認およびテスト用として、ユーザーに提供しています。



タスク

System Center Configuration Manager のアプリケーション モデル データをインストールに含めるには、以下の手順に従います:

1. 製品に必ずソフトウェア識別タグを含めてください。詳細については、「製品のソフトウェア識別タグを含める」を参照してください。
2. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
3. **"SCCM アプリ モデル データの追加"**設定で **[はい]** を選択します。
4. **"SCCM アプリ モデル データの追加"** 設定の下のサブ設定を適宜構成します。詳細については、「[一般情報] ビューの設定」を参照してください。

プロジェクトにタグ機能およびアプリケーション モデル データを含めると、InstallShield が作成する 2 つの新しいコンポーネントにタグがアプリケーション モデル データと共に追加され、これらのコンポーネントが製品機能の 1 つに関連付けられます。コンポーネントは、以下のとおりです:

- INSTALLDIR をインストール先とする ISO19770_LocalTag
- CommonAppDataFolder をインストール先とする ISO19770_SystemTag

これらのコンポーネントをプロジェクト内の別の機能に関連付ける場合は、[セットアップのデザイン] ビューを使います。詳細については、「コンポーネントと機能の関連付け」を参照してください。

ビルド時に以下の条件が True 評価された場合、InstallShield がビルドするインストールにソフトウェア識別タグが追加されます。また、タグにアプリケーション モデル データも含められます。

- [一般情報] ビューの "ソフトウェア識別タグの使用" 設定に、デフォルト値の [はい] が選択されている。
- [一般情報] ビューの "一意な ID""タグ作成者"、および "タグ作成者 ID" 設定に値が設定されている。
- [一般情報] ビューの "SCCM アプリ モデル データの追加" 設定で、[はい] が選択されている。

プロジェクトにソフトウェア ID タグを含めて [リリース] ビューで .pfx ファイルを使ってリリースに署名を行うことを設定すると、InstallShield がビルド時にタグをデジタル署名します。タグ ファイルに署名するためには、.NET Framework 2.0 以降をビルドマシンにインストールする必要があります。

インストールのファイルを編成する

インストールの一番重要なタスクは、ファイルを配布メディアからエンドユーザーのハード ディスクへ転送することです。InstallShield インストールでは、ファイルは階層構造で編成されていて、ファイルはコンポーネントに含まれています。コンポーネントは機能 (およびオプションでサブ機能) に関連付けられています。InstallScript と InstallScript MSI インストールでは、機能はセットアップの種類に関連付けられています。

実行時に、エンドユーザーは、単純にセットアップの種類、または、可能な場合、インストールする機能およびサブ機能を選択するだけです。エンドユーザーにはコンポーネントは見えません。コンポーネントは、ファイルをその種類および目的によってグループ分けするときに使われます。たとえば、同一のターゲットフォルダーにインストールされるファイルは、自己登録ファイルと同様、同一のコンポーネントに配置することができます。

あるファイルが、他のファイルの関数に依存してタスクを実行することがよくあります。ただし、インストール プロジェクトにアプリケーションファイルを含める際に、依存関係と呼ばれる他のファイルに気が付かない場合があります。依存関係ファイルが識別しやすくなるように、InstallShield には自動的に依存関係をプロジェクトに追加する 3 種類の依存関係スキャナーがあります。

プロジェクトへの個別ファイルの追加に加え、再配布可能ファイル (InstallShield 前提条件、マージ モジュールおよびオブジェクト) も含めることができます。再配布可能ファイルには、特定の機能性をインストールするのに必要なロジックとファイルが含まれています。たとえば、Java Runtime Environment (JRE) ファイルをインストールに含める場合、JRE の InstallShield 前提条件をインストール プロジェクトに追加します。

このセクションには、次のトピックが含まれています:

- インストールをデザインする
- パス変数を使用する
- ファイルとフォルダーを含める
- コンポーネントを使用する
- 機能の定義
- セットアップの種類について
- インストールに再配布可能ファイルを含める
- アプリケーションの依存関係を識別する
- COM サーバーの登録
- プロジェクトに DIM を追加する

インストールをデザインする

インストール プロジェクトには、開発者用とエンドユーザー用の 2 つの主要な観点があります。InstallShield では、それぞれ [コンポーネント] ビューと [機能] ビューで詳しくその観点を説明しています。

コンポーネント

コンポーネント は、製品の開発者用の観点を表します。コンポーネントは、ファイル、レジストリ エントリ、およびショートカットなど、開発者が類似したアプリケーション データを論理的にグループ分けするのに役立つインストール オーサリング ツールです。

エンドユーザーが、インストールする機能を選択できるようにするには、アプリケーションの機能に対応するコンポーネントにアプリケーションを分割する必要があります。詳細については、「アプリケーションをコンポーネントに分ける」を参照してください。

InstallShield を使用すると、プロジェクトのコンポーネントを公開することもできます。



メモ *コンポーネント、製品、またはアクションのパブリッシュに指定した条件は、設計時には検証されません。必ず 正しい構文を使用して、条件によって出力が正常に生成されることを確認してください。*

機能

機能は、エンドユーザーから見たアプリケーションの構成要素です。各機能は、たとえばヘルプファイルのような製品の特定の機能を表します。エンドユーザーは、製品の別個の機能をインストールまたはアンインストールできます。

たとえば、ハードディスクの容量に制限のあるエンドユーザーは、製品チュートリアルをインストールしないよう選択できます。そのユーザーは、後で別のコンピューターを購入するか、または既存のコンピューター上のリソースに空きを作り、前にアンインストールした製品チュートリアルをインストールすることもできます。

アプリケーションのコンポーネントに対応して、アプリケーションを機能に分割する必要があります。詳細については、「アプリケーションを機能に分ける」を参照してください。

アプリケーションをコンポーネントに分ける

次のガイドラインを使用して、このアプリケーションをコンポーネントに分けます。

- グループとして機能するリソースを特定します。各グループをコンポーネントとすることができます。
- Windows Installer では、コンポーネント内のあらゆるファイルが同じディレクトリにインストールされる必要があります。ディレクトリ構造をインストールする必要がある場合、各サブディレクトリが個別のコンポーネントに対応している必要があります。
- コンポーネントのリソースは、同じ条件を必要とします。
- Windows Installer の修復機能を使用するためには、各 .exe、.dll および .ocx ファイルを、個別のコンポーネントに配置し、そのコンポーネントのキーファイルにする必要があります。(コンポーネントウィザードのベストプラクティスオプションでは、この規則を使ってコンポーネントを作成できます。)
- 同様に、各 .chm ファイルまたは .hlp ファイルも、対応する .chi または .cnt ファイルと一緒に、独自のコンポーネントに配置する必要があります。
- いくつかの製品全体を通して、2 つ以上のコンポーネントにリソースを含めることはできません。コンポーネントのリソースが 2 つ以上の製品で必要とされる場合は、共有リソースのマージ モジュール作成をご検討ください。
- コンポーネントを1つまたは複数の機能にマップする際は、かならずコンポーネントのすべての部分が機能にマップされるようにします。
- インストールをテストする際は、アプリケーションをいくつかのコンポーネントに分割して、そのパフォーマンスを徹底的に測定できます。

アプリケーションを機能に分ける

次のガイドラインを使用して、このアプリケーションを機能に分けます。

- アプリケーションを、ヘルプ ファイル、クリップ アート ファイル、プログラム ファイルなどの非依存な部分に分けます。これにより、エンドユーザーはアプリケーションをインストールする際オプションを組み合わせることができます。たとえば、アプリケーションにサイズが大きく画像の多いクリップ アート ファイルが含まれている場合、クリップ アート ファイルを機能にすることができます。これにより、エンドユーザーはファイルをインストールするかしないかを選択することができます。これは、使用できるリソースが限られている場合に極めて重要な機能です。
- アプリケーションを機能に分ける際、かならずエンドユーザーが特定のニーズを実現するため、分けた部分をいくつかの方法で再度組み合わせることができるようにします。これを行う際、システム管理者から顧客サービス担当者、開発者などすべてのユーザーのニーズについて考慮します。ユーザーのすべてのグループに対応することによって、アプリケーションの配布と使用の増加を促すことになります。
- 各機能は、ヘルプ ファイルなどの 1 つの機能性を持ち、この機能性にしたがって明確に定義を行うことで認識および理解されやすくします。機能には、ユーザーが選択で自ら機能をインストールして使用できる独立した機能性が必要です。
- ある機能が別の機能を必要とする場合、依存する機能をもう一方の子にします。
- 混乱を避けるため、システムやアプリケーションの管理に関するあらゆる情報をユーザーに透過にします。

パス変数を使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

従来、ソースファイルをインストール プロジェクトにリンクするには、ハードコード化されたパスを使用して、そのファイルへのリファレンスを作成する必要がありました。たとえば、ソースファイルをインストールに含める場合は、C:¥Work¥Files にある Program.exe のように指定していました。但し、ハードコード化したパスを使用すると、ディレクトリからソースファイルを関連付けるときに毎回完全パスを入力する必要がありました。ファイルを別のディレクトリに移動すると、インストール プロジェクトで表示されるようにハードコード化したパスを変更する必要がありました。インストールに含まれているソースファイルの数が少ない場合は、このことは問題にならなかったかもしれません。残念ながら、インストールによっては、フォルダー構造を変えたり、別のマシンにプロジェクトを移行すると、これらすべてのファイルを再度マッピングし直す必要があり何千ものファイルを含んでいることがあります。

プロジェクトを移動したり、ディレクトリ構造を変更するたびに各ソースファイルのパスを変更する必要がないように、パス変数を使って共通して使用するパスを中央ディレクトリに定義することができます。

すべてのパス変数は、[パス変数] ビューで表示、修正することができます。ダイアログ エディター、ダイナミックファイルリンク、リリースフォルダーなど、ソースファイルにリンクされている InstallShield のほとんどすべてのフォルダーのパス変数を使用できます。パス変数を自分で入力するかわりに、パスを参照するたびに InstallShield の推奨値を使用することができます。InstallShield がパス変数を自動的に推奨するように設定するには、[オプション] ダイアログ ボックスの [パス変数] タブで、[常に [パス変数の推奨] ダイアログを表示] オプションを選択します。このセクションには、次のトピックが含まれています:

- パス変数の種類
- カスタム パス変数の作成と定義
- リリースのカスタム パス変数の値をオーバーライドする
- プロジェクト内の絶対パスの参照をパス変数に変換する
- パス変数の削除



メモ *パス変数は、インストール プロジェクトの開発中に使用されます。これらのパスは、アプリケーションがインストールされるターゲットマシンには適用されません。代わりに、インストール プロジェクトのソースファイルにリンクするために使用されます。プロジェクトをビルドする際、これらのリンクが評価され、ポイント先のファイルがインストールに組み込まれます。*

パス変数の種類



プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

InstallShield では、次の種類のパス変数がサポートされています:

- 定義済みパス変数
- 標準パス変数
- レジストリ パス変数
- 環境パス変数

定義済みパス変数は、標準 Windows ディレクトリを定義します。この種類の変数は、名前を変更することができません。標準、レジストリ、および環境パス変数は、独自に作成および定義が可能なカスタム変数です。

定義済みパス変数



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

定義済みパス変数は特定の標準的の Windows ディレクトリに設定された変数です。これらの変数は修正したり名前を変更することはできませんが、あらかじめ決められたディレクトリをポイントするためにインストール プロジェクトで使用することができます。定義済みパス変数、Windows XP システムでのそれらの通常の値、および対応する InstallScript パス変数を以下のリストに示します。(InstallScript プロジェクトでは、InstallScript パス変数は、たとえば、コンポーネントの "インストール先" プロパティのリストのオプションとして InstallShield で使用されます。InstallScript パス変数はまた、インストールスクリプトでも使用できるシステム変数に対応しています。)

テーブル 1 ・ 定義済みパス変数

定義済みパス変数	値	InstallScript パス変数
<CommonFilesFolder>	C:¥Program Files¥Common Files¥	<COMMONFILES>
<ISProductFolder>	C:¥Program Files¥InstallShield¥2022	
<ISProjectDataFolder>	<ISProjectFolder>¥ProjectName	<ISPROJECTDIR>
<ISProjectFolder>	C:¥InstallShield 2022 Projects¥	
<ISRedistPlatformDependentExpressFolder>	C:¥Program Files¥InstallShield¥2022¥Redist¥Language Independent¥i386 Express	
<ISRedistPlatformDependentFolder>	C:¥Program Files¥InstallShield¥2022¥Redist¥Language Independent¥i386	
<ProgramFilesFolder>	C:¥Program Files¥	<PROGRAMFILES>
<SystemFolder>	C:¥Windows¥System32¥	<WINSYSDIR>

テーブル 1・定義済みパス変数 (続き.)

定義済みパス変数	値	InstallScript パス変数
<VSSolutionFolder>	状況によって異なる。この変数は、ハイレベルの基本ディレクトリを参照します。このサポートによって、InstallShield プロジェクトで Visual Studio ソリューション フォルダー内にある姉妹プロジェクトのファイルへのスタティック リンクを含むことができます。詳細については、「Visual Studio ソリューションで VSSolutionFolder パス 変数を使用する」を参照してください。	
<WindowsFolder>	C:¥Windows¥	<WINDIR>

デフォルトのパス変数の使用



プロジェクト: *Windows Installer ベースのプロジェクトのみ: [パス変数の推奨] ダイアログ ボックスで、定義済みパス変数を使用するかどうか指定することができます。このダイアログ ボックスは、ソース ファイルをインストールに挿入するときにハードコードされたパスを入力し、[オプション] ダイアログ ボックスにある [パス変数] タブで、[常に [パス変数の推奨] ダイアログを表示する] を選択した場合に起動します。定義済みのパス変数に含まれるパスを入力した場合、InstallShield はハードコード化されたパスではなく、パス変数を使用するように指示します。*

定義済みのパス変数は、[標準パス変数] の値を定義する場合にも利用することができます。定義済み変数のサブフォルダーに標準パス変数を定義する必要がある場合があります。たとえば、標準パス変数の値として <ProgramFilesFolder>¥InstallShield を使用することができます。

定義済みパス変数は スマート 変数です。つまり、Windows ディレクトリが C ドライブではなく D ドライブ上にある場合でも正しいディレクトリに設定されていることを意味します。[オプション] ダイアログ ボックスの [ファイルの場所] タブでデフォルトのプロジェクトの場所を変更すると、変数が更新され、新しいディレクトリに反映します。

標準パス変数



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*

- *スイート/アドバンスト UI*

標準パス変数は、ユーザー定義変数と呼ばれることもあります。これらの変数タイプは InstallShield に組み込まれています。つまり、これらはレジストリまたは環境変数のような外部リソースに依存しません。

標準パス変数の使用

標準パス変数を作成すると、ソースファイルをプロジェクトに追加するたびに、その変数を使用できます。たとえば、*<MyFiles>* という変数を作成し、その変数が C:¥Work¥Files をポイントしているとします。インストール先にその変数のパスが含まれているプロジェクトにソースファイルを追加する場合、パスをハードコード化するのではなく、変数を使用するようお勧めします。この推奨メッセージは、[パス変数の推奨] ダイアログ ボックスに表示されます。

InstallShield がパス変数を自動的に推奨するように設定するには、[オプション] ダイアログ ボックスの [パス変数] タブで、[常に [パス変数の推奨] ダイアログを表示] オプションを選択します。



ヒント: *ビルド時に特定のリリースにおけるユーザー定義のパス変数の値、環境変数、またはレジストリ値をオーバーライドするには、そのリリースの [ビルド] タブにある "パス変数のオーバーライド" 設定を使います。詳細については、「リリースのカスタム パス変数の値をオーバーライドする」を参照してください。*

レジストリ パス変数



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

レジストリ パス変数を使用すると、指定したレジストリキーのデフォルト値に基づいて独自の変数を定義することができます。

レジストリパス変数を作成すると、ソースファイルをプロジェクトに追加するたびに、その変数を使用できます。ファイルを *MyRegVar* の値を含むフォルダーからコンポーネントに追加する場合、ハードコード化されたパスよりもこのレジストリパス変数を使用することをお勧めします。この推奨メッセージは、[パス変数の推奨] ダイアログ ボックスに表示されます。

InstallShield がパス変数を自動的に推奨するように設定するには、[オプション] ダイアログ ボックスの [パス変数] タブで、[常に [パス変数の推奨] ダイアログを表示] オプションを選択します。

ヒント: *ビルド時に特定のリリースにおけるユーザー定義のパス変数の値、環境変数、またはレジストリ値をオーバーライドするには、そのリリースの [ビルド] タブにある "パス変数のオーバーライド" 設定を使います。詳細については、「リリースのカスタム パス変数の値をオーバーライドする」を参照してください。*

環境パス変数

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

環境パス変数を使用すると、[環境] ダイアログ ボックスの特定の値に基づいて、独自のパス変数を定義できます。

タスク

[環境] ダイアログ ボックスを表示するには、以下の手順を実行します。

1. **[マイ コンピューター]** を右クリックして、**[プロパティ]** を選択します。**[システム プロパティ]** ダイアログ ボックスが開きます。
2. **[詳細]** タブで、**[環境変数]** をクリックします。

環境パス変数が役に立つのは、通常、コマンドラインからビルドを実行する場合です。専用のマシンで夜間ビルドを実行するときは、コマンドラインからパス変数を変更する必要があります。環境パス変数を使用している場合は、バッチファイルまたはコマンドラインからプロジェクトファイルのパスを定義できます。これらのパスは、インストール プロジェクトをビルドする際に評価され、ファイルへの正しいパスが使用されます。

環境パス変数の使用

環境パス変数を作成すると、ソースファイルをプロジェクトに追加するときはいつでもその変数を使用できます。たとえば、*<MyFiles>* という変数を作成し、その変数が C:¥Work¥Files をポイントしているとします。インストール先にその変数のパスが含まれているプロジェクトにソースファイルを追加する場合、パスをハードコード化するのではなく、変数を使用するようお勧めします。この推奨メッセージは、[パス変数の推奨] ダイアログ ボックスに表示されます。

InstallShield がパス変数を自動的に推奨するように設定するには、[オプション] ダイアログ ボックスの [パス変数] タブで、**[常に [パス変数の推奨] ダイアログを表示]** オプションを選択します。

ヒント: *ビルド時に特定のリリースにおけるユーザー定義のパス変数の値、環境変数、またはレジストリ値をオーバーライドするには、そのリリースの [ビルド] タブにある "パス変数のオーバーライド" 設定を使います。詳細については、「リリースのカスタム パス変数の値をオーバーライドする」を参照してください。*

カスタム パス変数の作成と定義

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

[パス変数] ビューを使って、標準パス変数、環境パス変数、およびレジストリ パス変数を作成したり定義したりできます。

タスク ***パス変数を作成および定義するには、以下の手順に従います:***

1. 以下のいずれかを実行します。
   - QuickPatch プロジェクトの場合: **[パッチの設定]** の下にあるビュー リストで、**[パス変数]** をクリックします。
   - その他のプロジェクトの場合: **[メディア]** の下にあるビュー リストで、**[パス変数]** をクリックします。
2. 以下のいずれかを実行します。
   - 標準パス変数を作成するには、**[新しいパス変数]** ボタンをクリックします。
   - 環境変数を作成するには、**[新しいパス変数]** ボタンの隣りにある矢印をクリックしてから、**[環境]** をクリックします。
   - レジストリ変数を作成するには、**[新しいパス変数]** ボタンの隣りにある矢印をクリックしてから、**[レジストリ]** をクリックします。

   ビューの下に新しい行が追加されます。
3. 必要に応じて、新しい行の各設定を入力します。

各列の詳細については、「[パス変数] ビュー」を参照してください。



ヒント: *ビルド時に特定のリリースにおけるユーザー定義のパス変数の値、環境変数、またはレジストリ値をオーバーライドするには、そのリリースの [ビルド] タブにある "パス変数のオーバーライド" 設定を使います。詳細については、「リリースの [ビルド] タブ」を参照してください。*

リリースのカスタム パス変数の値をオーバーライドする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallShield では、今回より、プロジェクト内の各リリースでプロジェクトの標準パス変数、環境パス変数、およびレジストリ パス変数の値をオーバーライドすることができます。この機能を使って、ビルドする特定のリリースごとに、ビルド時にプロジェクト内の特定のファイルとフォルダーを別のファイルとフォルダーに置換することができます。

たとえば、基本の MSI プロジェクトの場合、この機能を使ってカスタム アクション DLL ファイル (プロジェクトの 32 ビット リリース用の 32 ビット バージョン ファイルおよびプロジェクトの 64 ビット リリース用の 64 ビット バージョン ファイル) を置換することができます。あるいは、スイート/アドバンスト UI プロジェクトの場合、この機能を使って、プロジェクト内の異なるリリースでイメージや EULA などの UI 要素を置換することができます。これによって、プロジェクトの異なるエディションまたは異なるブランド バージョンのインストールを簡単に生成することができます。



タスク ***リリースのカスタム パス変数の値をオーバーライドするには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、構成するリリースを選択します。
3. **[ビルド]** タブにある **"パス変数のオーバーライド"** 設定で、省略記号設定 (...) をクリックします。 **[パス変数のオーバーライド]** ダイアログ ボックスが開きます。

   [パス変数のオーバーライド] ダイアログ ボックスには、[パス変数] ビューで構成されたパス変数、環境パス変数、およびレジストリ パス変数のチェック ボックスが表示されます。定義済みのパス変数、または [リリース] ビュー内の同じリリースで既にオーバーライドされているパス変数のチェック ボックスは含まれていません。
4. 選択されたリリースで、オーバーライドする各パス変数のチェック ボックスを選択します。
5. **[OK]** をクリックします。ダイアログ ボックスを閉じます。**"パス変数のオーバーライド"** 設定の下に、選択された各パス変数に新しいサブ設定が追加されます。
6. 各パス変数のサブ設定で、適切な新しい値を指定します。

ビルド時、InstallShield は [リリース] ビューで指定された新しい値を使って、[パス変数] ビューで設定されたデフォルト値をオーバーライドします。

パス変数のオーバーライドを行うその他の方法

[リリース] ビューのパス変数をオーバーライドする方法以外に、次の方法でパス変数をオーバーライドすることができます:

- IsCmdBld.exe を使って、コマンドラインからビルドする場合は、-I パラメーターを使って、パス変数と新しい値を指定します。
- MSBuild または Team Foundation Server (TFS) を使ってビルドする場合、InstallShield タスクで PathVariables パラメーターを使います。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPathVariableOverrides として公開されます。

これらの方法は、[リリース] ビューの "パス変数のオーバーライド" 設定で設定された値よりも優先されます。

プロジェクト内の絶対パスの参照をパス変数に変換する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

インストール プロジェクトを始めた段階では、パス変数を使用する機会は多くはないかもしれませんが、作業を進めるにしたがって、その有用性が明らかになるでしょう。各ファイルリンクを確認してパス変数に変更する代わりに、ソースパス変換ウィザードを利用することができます。

このウィザードは、プロジェクトのスタティックリンクをスキャンして、すべてをパス変数に変換します。スタティックリンクどれかが既存のパス変数に置換できる場合は、ウィザードで自動的に置換されます。その他のリンクについては、ウィザードが指定可能な変数のリストを表示します。利用する変数を選択した後、ウィザードはリンクのすべてを変換します。



タスク ***ソースパス変換ウィザードを起動するには、以下の手順に従います:***

[プロジェクト] メニューで **[ソース パスの変換]** をクリックします。

パス変数の削除



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*



タスク ***パス変数をプロジェクトから削除するには、以下の手順に従います:***

1. 以下のいずれかを実行します。
   - QuickPatch プロジェクトの場合: **[パッチの設定]** の下にあるビュー リストで、**[パス変数]** をクリックします。
   - その他のプロジェクトの場合: **[メディア]** の下にあるビュー リストで、**[パス変数]** をクリックします。
2. 削除するパス変数を選択します(複数選択可)。

   連続する複数のパス変数を選択するには、最初のパス変数を選択してから SHIFT キーを押しながら最後のパス変数を選択します。連続しない複数のパス変数を選択するには、最初のパス変数を選択してから CTRL キーを押しながらその他のパス変数を選択します。
3. **[選択したパス変数の削除]** ボタンをクリックします。

ファイルとフォルダーを含める

ファイルは、製品の核であるとともにインストールの核でもあります。ファイルは、コンポーネントレベルでのみプロジェクトに追加されます。インストール プロジェクトでは、コンポーネントは機能と関連付けられています。エンドユーザーがインストールを実行したときに表示されるのは機能です。したがって、インストールでコンポーネントの機能が選択された場合、そのコンポーネントのファイルがターゲット システムにインストールされます。

Windows Installer ベースのプロジェクトでコンポーネントにファイルを追加する場合には、セットアップベストプラクティスに留意する必要があります。デフォルトでは、セットアップベストプラクティスウィザードがセットアップのデザインを監視し、ベストプラクティスに反した場合には警告を表示します。

ファイルを [セットアップのデザイン] ビュー(インストール プロジェクトの場合)または [コンポーネント] ビュー、および [ファイルとフォルダー] ビューでプロジェクトに追加できます。

このセクションには、次のトピックが含まれています:

- プロジェクト ファイルに追加してターゲット システムにインストールする

- コンテキスト メニューを使ってファイルをドラッグ アンド ドロップする
- 64 ビット ソース マシンの 64 ビット System32 フォルダーからファイルを追加する
- 新しいインストール先フォルダーの作成
- 空フォルダーの作成
- ハードコード化されたインストール先ディレクトリを指定する
- ターゲット システムからファイルとフォルダーを削除する
- プロジェクトでファイルとフォルダーを管理するときのヒント
- プロジェクトでファイルとフォルダーを検索する
- [ファイルとフォルダー] ビューにコンポーネントを表示する
- [ファイルとフォルダー] ビューを使ってコンポーネントを変更する
- ファイルの設定を構成する
- ファイルの関連付け
- ダイナミック ファイル リンク
- ターゲット システム上でファイルとコンポーネントを上書きする
- [ファイル] ビューで定義済みフォルダーを表示する
- コンパニオン ファイル
- ファイルとフォルダーのアクセス許可を構成する
- ファイルの追加時に COM データを抽出する
- 自己登録ファイルのインストール
- .NET アセンブリのプロパティおよび依存関係を識別する
- InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトでフォントをインストールする

プロジェクト ファイルに追加してターゲット システムにインストールする

[ファイルとフォルダー] ビューを使用して、プロジェクトにファイルを追加することができます。



タスク ***プロジェクトにファイルを追加するには、以下の手順を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[ビュー フィルター]** リストで、追加するファイルを含む機能を選択します。
3. Windows Installer ベースのプロジェクトのみ: **[インストール先コンピューターのフォルダー]** ペインで、**[インストール先コンピューター]** を右クリックして、**[定義済みフォルダーの表示]** をポイントし、使用する定義済みフォルダーをクリックします。

4. **[インストール先コンピューターのフォルダー]** ペインで、ファイルを配置するフォルダーをクリックします。
5. **[ソース コンピューターのフォルダー]** ペインで、追加するファイルが入ったフォルダーに移動します。

   INSTALLDIR (Windows Installer ベースのプロジェクト) または Application Target Folder (InstallScript プロジェクト) は、アプリケーションのファイルのデフォルトのルートディレクトリであるため最もよく使用されるターゲット先となります。
6. 追加するファイルを **[ソース コンピューターのファイル]** ペインから選択して、**[インストール先コンピューターのファイル]** ペインにドラッグします。



ヒント: *INSTALLDIR の [ファイルとフォルダー] ビューでの [インストール先コンピューターのフォルダー] ペインでの表示方法を指定するには、[オプション] ダイアログ ボックスの [ディレクトリ] タブで設定を変更します。*

インストール先ディレクトリをハードコード化して指定することもできます。

[インストール先コンピューターのフォルダー] ペインでは、定義済みのインストール先 リストを使用できます。定義済みのインストール先または独自に作成したインストール先のどちらかにファイルを追加することもできます。

ファイル追加時に自動的に新しいコンポーネントを作成する

プロジェクトに新しいファイルを追加すると、InstallShield が新しいコンポーネントを作成して、これを適切に処理することができます。これを行うには、このビューの [ビュー フィルター] を使って、新しく作成したコンポーネントを関連付ける機能を選択します。

機能が存在しない場合、このビューにファイルを追加してから、機能を作成することもできます。

自動コンポーネントの作成におけるプロジェクト固有の違い

Windows Installer ベースのプロジェクトの場合、InstallShield はセットアップ ベスト プラクティスに基づいてコンポーネントを作成します。たとえば、すべてのポータブル実行可能ファイル (.exe、.dll、.ocx ファイル) に、独自のコンポーネントが与えられます。その他のすべてのファイルは、各インストール先ディレクトリのデフォルトのコンポーネントに追加されます。しかし、インストール先フォルダーの下にリスト表示されているコンポーネントの 1 つにファイルを直接ドラッグする場合、セットアップベストプラクティスの規則は無視されます。

InstallScript プロジェクトでは、InstallShield がファイルの種類に従ってコンポーネントを作成します。たとえば、すべての自己登録ファイルは 1 つのコンポーネントに含まれ、自己登録ファイル以外は別のコンポーネントに含まれ、そして .NET ファイルはすべて別のコンポーネントに含まれます。コンポーネントはサブフォルダーとして [ファイルとフォルダー] ビューに表示されます。

コンポーネントと機能の関係の変更

[インストール先コンピューターのフォルダー] ペインにあるコンポーネントを右クリックしてから [プロパティ] を選択することにより、コンポーネントと機能の関係を変更することができます。[プロパティ] ダイアログ ボックスにある [機能] タブを使って、必要に応じて情報を変更します。



ヒント: *[ファイルとフォルダー] ビューにコンポーネントを表示するには、インストール先フォルダー (ビューの左下ペイン) を右クリックして [コンポーネントの表示] を選択します。*

コンテキスト メニューを使ってファイルをドラッグ アンド ドロップする

[ファイルとフォルダー] ビューにある [ファイル] エクスプローラーを使うと、ソース コンピューターからターゲット システムのインストール先までフォルダーをドラッグ アンド ドロップすることができます。**[ソースコンピューターのフォルダー]** ペインから **[インストール先コンピューターのフォルダー]** ペインにファイルをドラッグする場合、多くのオプションがあります。



ヒント: *64 ビット システム上で InstallShield を使用している場合に、ソースの場所が開発マシン上の 64 ビット システム フォルダー (System32) であるシステム ファイルをプロジェクトに追加するとき、[ファイルとフォルダー] ビューの上部にあるソース コンピューターのペインからインストール先コンピューターのペインの適切な場所にドラッグすることはできません。詳細については、「64 ビット ソース マシンの 64 ビット System32 フォルダーからファイルを追加する」を参照してください。*



タスク ***コンテキスト メニューを表示するには、以下の手順を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[ソース コンピューターのフォルダー]** ペイン、または **[ソース コンピューターのファイル]** ペインでフォルダーまたはファイルを右クリックしてから、**[インストール先コンピューターのフォルダー]** ペインまたは **[インストール先コンピューターのファイル]** ペインにドラッグします。そしてマウス ボタンを離します。

   フォルダーまたはファイルを追加する定義済みフォルダーが **[インストール先コンピューターのフォルダー]** ペインに表示されていない場合、それを追加することができます。詳しくは、「[ファイル] ビューで定義済みフォルダーを表示する」をご覧ください。

InstallShield は、いくつかのコマンドを含むコンテキスト メニューを表示します。

テーブル 2 - [ファイルとフォルダー] ビューのコンテキスト メニューから使用できるコマンド

コマンド	説明
追加	選択されたフォルダー、サブフォルダーおよび/またはファイルを追加します。これはデフォルトのドラッグ アンド ドロップの動作と同じです。
ソース構造を保持したまま追加	ソース コンピューターのファイル/フォルダー構造を保持しながら、選択されたフォルダー、サブフォルダーおよび/またはファイルを追加します。 このコマンドは、ソース フォルダーが定義済みの Windows インストール先フォルダーと一致する場合にのみ使用できます。さらに、ファイルまたはフォルダーをドロップするインストール先は、インストール先コンピューターにある必要があります。
フォルダーのみを追加	選択されたフォルダーと、選択されたフォルダーに含まれるすべてのサブフォルダーのみを追加します。選択されたフォルダーまたはサブフォルダーに含まれているファイルは追加されません。

テーブル 2 - [ファイルとフォルダー] ビューのコンテキスト メニューから使用できるコマンド (続き.)

コマンド	説明
ソース構造を保持したままフォルダーのみを追加	選択されたフォルダーと、選択されたフォルダーに含まれるすべてのサブフォルダーのみを追加します。選択されたフォルダーまたはサブフォルダーに含まれているファイルは追加されません。このオプションは、また、ソース コンピューターで見つかったフォルダー構造を保持します。 このオプションは、ソース フォルダーが定義済みの Windows インストール先フォルダーと一致する場合にのみ使用できます。さらに、ファイルまたはフォルダーをドロップするインストール先は、インストール先コンピューターにある必要があります。
Cancel	変更を加えずにドラッグアンドドロップ操作を終了します。

64 ビット ソース マシンの 64 ビット System32 フォルダーからファイルを追加する

64 ビット システムでは、System32 フォルダーは、64 ビット アプリケーションに予約されています。InstallShield の [ファイルとフォルダー] ビューで、開発マシンの 64 ビット システム フォルダーを表示しようとした時、32 ビット バージョンのフォルダーである SysWOW64 フォルダーが代わりに表示されます。したがって、64 ビット システム上で InstallShield を使用している場合に、ソースの場所が開発マシン上の 64 ビット システム フォルダー (System32) であるシステム ファイルをプロジェクトに追加するとき、[ファイルとフォルダー] ビューの上部にあるソース コンピューターのペインからインストール先コンピューターのペインの適切な場所にドラッグすることはできません。

リダイレクトを回避して、開発マシン上の 64 ビット System32 ファイルを InstallShield プロジェクトに追加するには、マシン上の Sysnative フォルダーを参照してから、プロジェクトに適切なファイルを選択します。以下は、その手順の説明です。



注意 - *システム フォルダーは Windows によって保護されているため、通常、インストールにシステム ファイルを含めることは推奨されません。これらの場合、システム ファイルの配置/更新には、使用可能な場合、マイクロソフトの再配布可能ファイルを使用すること、または、Windows アップデートを使ってエンドユーザーに更新プログラムを取得してもらう方法を優先してください。*



タスク

64 ビット Windows が搭載された開発システム上の 64 ビット System32 フォルダーのファイルをプロジェクトに追加するには、以下の手順に従います:

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、ファイルを配置するフォルダーをクリックします。
3. **[インストール先コンピューターのフォルダー]** ペインを右クリックしてから **[ファイルの追加]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. 次のパスを指定します (ドライブ文字は適切なドライブ文字で適宜置き換えてください):

C:¥Windows¥Sysnative

5. プロジェクトに追加したい適切なファイルを選択してから、**[開く]** ボタンをクリックします。

ファイルがプロジェクトへ追加されます。Sysnative フォルダーが、追加したソース ファイルのパスの一部として使用されるようになります。WOW64 は Sysnative フォルダーを特殊なエイリアスとして認識するため、ファイル システムはこのフォルダーからアクセスのリダイレクトを行いません。



ヒント: *また、[コンポーネント] ビューまたは [セットアップのデザイン] ビューを使って 64 ビット System32 ファイルを追加することもできます。これらのビューで、64 ビット System32 ファイルを含むコンポーネントのノードを展開します。そのコンポーネントの下にある [ファイル] ノードを選択します。次に、右側の [ファイル] ペインを右クリックして、[ファイルの追加] をクリックします。前述の手順の通り、C:¥Windows¥Sysnative パスを指定します。*

Windows ベースのシステムにおける Sysnative フォルダー サポートについて

Sysnative フォルダーの使用は、32 ビット マシンでサポートされていません。64 ビット システム上の InstallShield プロジェクトで Sysnative フォルダーを使用していて、32 ビット システムで、その InstallShield プロジェクトのリリースをビルドしようとした時、ソース ファイルが見つからなかったことを通知する 1 つ以上のビルド エラーまたは警告が生成されます。

Sysnative のサポートがある 64 ビット マシン環境にある InstallShield または Standalone Build のオートメーション インターフェイスを使用して、インストールを作成、編集、またはビルドする場合、64 ビット システム フォルダー内にあるシステム ファイルのソース フォルダーを指定する時、Sysnative フォルダーをパスの一部として使用できます。

新しいインストール先フォルダーの作成



タスク ***定義済みフォルダーのサブフォルダーを作成するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのレジストリ]** ペインで、フォルダーを右クリックし、**[新規フォルダー]** をクリックします。
3. 新しいフォルダーを選択して F2 キーを押した後、新しい名前を入力して名前を変更します。

または、全体のフォルダーをインストール先のフォルダーの 1 つにドラッグして、新しいサブフォルダーをソースフォルダーファイルを含むターゲットフォルダーに追加します。この新しいサブフォルダーの名前は、ソースフォルダーの名前と同じです。

空フォルダーの作成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript MSI*

基本の MSI プロジェクト

ダイレクト エディター で表示される .msi データベースの CreateFolder テーブル使用して、空のディレクトリを作成できます。

例については、ナレッジベース記事 Q103218を参照してください。

InstallScript MSI プロジェクト

CreateDir 関数で、空のディレクトリを作成できます。

ハードコード化されたインストール先ディレクトリを指定する

[ファイルとフォルダー] ビューで、ハードコード化したインストール先ディレクトリを指定することができます。

インストール先ドライブの指定



タスク

特定のドライブを指定するには、以下の手順を実行します。

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、**[インストール先コンピューター]** を右クリックし、**[新規フォルダー]** をクリックするか、**[インストール先コンピューター]** をクリックして、INSERT キーを押します。新しいフォルダーが追加されます。
3. フォルダーの名前は、ドライブ文字を入力し、続けてコロンを入力します (例 **C:**)。
4. ENTER を押します。

フォルダーおよびサブフォルダーを指定してインストール先パスを作成する

ドライブ文字フォルダーの下のフォルダーおよびサブフォルダーを指定してハードコード化されたインストール先パスを作成することができます。

1. フォルダーを追加するドライブ フォルダー (たとえば **C:**) またはサブフォルダーを追加するフォルダーを右クリックします。また、フォルダーをクリックして、INSERT キーを押しても同じことができます。
2. フォルダーの名前は、新しい名前を入力して ENTER キーを押します。

インストール先パスをすばやく作成する



タスク

ネストされたインストール先パスを時間を掛けずに作成するには、以下の手順に従います:

1. フォルダーを右クリックして **[新規フォルダー]** をクリックするか、フォルダーをクリックして INSERT キーを押します。
2. フォルダーの名前は、インストール先パスを入力します (例、**a¥b¥c**)。これにより、**a** を最上位に、**a** の下に **b**、**b** の下に **c** が来るフォルダー階層構造が作成されます。

ターゲット システムからファイルとフォルダーを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield では、実行時にターゲット システムから削除するファイルとフォルダーを簡単に指定することができるビルトイン サポートが提供されています。このファイルとフォルダーの削除機能は、アプリケーションによって作成されるファイルの削除など、インストールが追跡を行わない処理に使用すると便利です。

ファイルまたはフォルダーの削除は、次のイベントの 1 つにスケジュールできます:

- ファイルまたはフォルダーのコンポーネントがインストールされるとき
- ファイルまたはフォルダーのコンポーネントがアンインストールされるとき
- ファイルまたはフォルダーのコンポーネントがインストールまたはアンインストールされるとき

削除されるアイテムがフォルダーの場合、そのフォルダーが空の場合のみ削除されます。

InstallShield では、プロジェクトからのファイルまたはフォルダーのを削除を構成するさまざまな方法が提供されています:

[ファイルとフォルダー] ビューを使って、削除するファイルとフォルダーを指定します。



タスク ***[ファイルとフォルダー] ビューを使って、ファイルとフォルダーの削除を構成するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、削除するファイルまたはフォルダーを含むフォルダーを選択します。
3. **[インストール先コンピューターのファイル]** ペインを右クリックしてから **[ファイルの削除を追加]** をクリックします。**[プロパティ]** ダイアログ ボックスが開きます。
4. 必要に応じて設定を指定します。詳細については、「ファイル 削除の [プロパティ] ダイアログ ボックス」を参照してください。

InstallShield によって、**[インストール先コンピューターのファイル]** ペインにファイルまたはフォルダー アイコンが追加されます。アイコンに赤い X 印がついて、削除される項目を参照していることを示します。

[セットアップのデザイン] ビューまたは [コンポーネント] ビューを使って、削除するファイルとフォルダーを指定する



タスク

[セットアップのデザイン] ビューまたは [コンポーネント] ビューを使って、ファイルとフォルダーの削除を構成するには、次の手順に従います:

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトの場合) または **[コンポーネント]** をクリックします。
2. **[セットアップのデザイン]** または **[コンポーネント]** ツリーで、削除するファイルまたはフォルダーが含まれているコンポーネントのノードを拡張してから、**[ファイル]** サブノードをクリックします。
3. **[ファイル]** ペインを右クリックしてから **[ファイルの削除を追加]** をクリックします。**[プロパティ]** ダイアログ ボックスが開きます。
4. 必要に応じて設定を指定します。詳細については、「ファイル 削除の [プロパティ] ダイアログ ボックス」を参照してください。

InstallShield によって、[ファイル] ペインにファイルまたはフォルダー アイコンが追加されます。アイコンに赤い X 印がついて、削除される項目を参照していることを示します。

ファイルまたはフォルダーの削除の設定を編集する

ファイルまたはフォルダーの削除の設定を変更するには、削除するアイテムを右クリックしてから、[プロパティ] をクリックします。[プロパティ] ダイアログ ボックス が開き、ここで必要に応じて設定を編集できます。

プロジェクトでファイルとフォルダーを管理するときのヒント

ドラッグ アンド ドロップ処理、CTRL+C や CTRL+P といった一般的なキーボード ショートカット、およびアイテムを右クリックすると表示されるコンテキスト メニューを使って、ファイルとフォルダーを異なるインストール先フォルダーや機能へ簡単に移動させることができます。

プロジェクト内のファイルとフォルダーのインストール先を管理する



タスク

プロジェクト内のファイルとフォルダーのインストール先を管理するには、次の手順に従います:

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. 以下のいずれかを実行します:
   - ファイルまたはフォルダーの設定を変更するには、アイテムを右クリックしてから、**[プロパティ]** をクリックします。**[プロパティ]** ダイアログ ボックス が開き、ここで必要に応じて設定を編集できます。
   - あるインストール先フォルダーから別のインストール先フォルダーにファイルを移動させるには、**[インストール先コンピューターのファイル]** ペインの 1 つの場所から **[インストール先コンピューターのフォルダー]** ペイン内の適切なフォルダーにドラッグします。
   - あるインストール先フォルダーから別のインストール先フォルダーにフォルダーを移動させるには、**[インストール先コンピューターのフォルダー]** ペインの 1 つの場所から同じペイン内の適切なフォルダーにドラッグします。

- 1 つの場所から別の場所にファイルまたはフォルダーをコピーするには、CTRL を押しながら、アイテムを 1 つの場所から別の場所へドラッグします。

  別の方法として、アイテムをクリップボードにコピーしてから、それを適切な場所に貼り付けることもできます。アイテムを右クリックしてから **[コピー]** をクリックするか、それをクリックして CTRL+C を押します。 次にファイルを含める別のフォルダーを選択します。そのフォルダーを右クリックして **[貼り付け]** をクリックするか、フォルダーをクリックしてから CTRL+P を押します。

- プロジェクトからファイルまたはフォルダーを削除するには、それを右クリックしてから **[削除]** をクリックします。

プロジェクトに含まれるファイルと機能の関連付けを管理する



タスク ***プロジェクトに含まれるファイルと機能の関連付けを管理するには、次の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトの場合) または **[コンポーネント]** をクリックします。
2. 機能の関連付けを管理するファイルを含むコンポーネントのノードを展開してから、**[ファイル]** サブノードをクリックします。右側の **[ファイル]** ペインにファイルが表示されます。
3. 以下のいずれかを実行します:

   - ファイルを 1 つの機能から別の機能にコピーするには、ファイルをクリップボードにコピーしてから、適切な機能にペーストします。ファイルを右クリックしてから **[コピー]** をクリックするか、それをクリックして CTRL+C を押します。 次にファイルを含める別の機能を選択します。その [ファイル] ペインを右クリックして **[貼り付け]** をクリックするか、フォルダーをクリックしてから CTRL+P を押します。
   - ファイルを機能から削除するには、そのファイルを右クリックしてから **[削除]** をクリックします。

プロジェクトでファイルとフォルダーを検索する

多数のフォルダーとファイルをプロジェクトに追加したとき、特定のフォルダーまたはファイルの検索に手間が掛かることがあります。[ファイル] ビューでフォルダーおよびファイルの検索を行うことができます。InstallShield は、マッチしたものをすべて見つけ、最初に検索されたものをハイライトで表示します。検索条件にマッチしたものがすべて見つかるまで検索を続けることができます。



タスク ***プロジェクトでファイルとフォルダーを検索するには、以下の手順を従います。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、**[インストール先コンピューター]** を選択します。
3. **[編集]** メニューで、**[検索]** をクリックします。**[検索]** ダイアログ ボックスが開きます。

   また、CTRL+F を押しても同様の結果が得られます。
4. **[検索する文字列]** ボックスで、検索するテキストを入力します。***.exe** などのワイルドカードを利用することができます。
5. **[検索対象]** でファイル、フォルダー、またはその両方を検索するかどうかを指定します。
6. その他に必要な基準を指定します。

7. **[次を検索]** をクリックします。検索基準に一致したものがあったとき、最初のアイテムは、**[インストール先コンピューターのフォルダー]** ペインまたは **[インストール先コンピューターのファイル]** ペインのどちらかで選択されます。
8. 基準に適合する次のアイテムがある場合にそれを検索するには、F3 キーを押します。このステップを必要に応じて繰り返します。

[ファイルとフォルダー] ビューにコンポーネントを表示する

タスク ***インストールのコンポーネントとその関連ファイルを表示するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、[インストール先コンピューター] を右クリックし、**[コンポーネントの表示]** (Windows Installer ベースのプロジェクトの場合) または、**[コンポーネントとサブフォルダーを表示]** (InstallScript ベースのプロジェクトの場合) をクリックします。

機能に関連付けられていないインストール プロジェクト内のコンポーネントは、親のないコンポーネントのアイコン () で表示されます。

[ファイルとフォルダー] ビューを使ってコンポーネントを変更する

[ファイルとフォルダー] ビューの [フォルダー] エクスプローラーで、コンポーネントを右クリックして [プロパティ] をクリックし、[コンポーネントのプロパティ] ダイアログ ボックスを表示します。このダイアログ ボックスでは、コンポーネントのプロパティの変更、ダイナミックファイルリンクの追加、およびコンポーネントを含む機能を変更することができます。

メモ: *(Windows Installer ベースのプロジェクトのみ) フォルダー アイコンを右クリックして、[コンポーネント ウィザードの起動] を選択することにより、コンポーネント ウィザードを起動することができます。*

ファイルの設定を構成する

ターゲット システムにインストールする各ファイルに多くのプロパティを設定することができます。

タスク ***ファイルのプロパティを指定するには、次の手順に従います。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、プロパティを構成するファイルを含むフォルダーをクリックします。
3. **[インストール先コンピューターのファイル]** ペインで、フォルダーを右クリックし、**[プロパティ]** をクリックします。**[プロパティ]** ダイアログ ボックスが開きます。

設定可能な各プロパティについての説明は、「[ファイルのプロパティ] ダイアログ ボックス」を参照してください。

ファイルの関連付け



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ファイルの関連付けは、特定のタイプのファイルを開くのにどのアプリケーションを使用すべきかを Windows に指示するためのレジストリ設定です。たとえば、Windows では、テキスト(.txt)ファイルを開くと通常「メモ帳」が起動します。

システムで登録済みファイルを表示および変更するには、Windows エクスプローラーを開き、[ツール] メニューで、[フォルダー オプション] をクリックします。ファイルの関連付けの構成方法を見るには、[フォルダー オプション] ダイアログ ボックス の [ファイルの種類] タブを使用します。

似たような方法で、Windows エクスプローラーでファイルを右クリックしてから [プロパティ] をクリックして該当のファイルに関連付けられているアプリケーションを識別することもできます。

ファイルの関連付けは HKLM¥SOFTWARE¥Classes と HKCU¥SOFTWARE¥Classes の両方に格納され、マージされたデータは HKEY_CLASSES_ROOT の下に表示されます。

インストール プロジェクトのファイルの関連付けを作成する

ベストプラクティス ガイドラインは、プロジェクトで作成または使用されるすべての表示タイプのファイルに対してファイルの関連付けを作成することを推奨しています。[ファイル拡張子] ビューを使用すると、インストール プロジェクトにファイルの関連付けを素早く簡単に作成することができます。[ファイル拡張子] ビューは、[コンポーネント] ビューおよび [セットアップのデザイン] ビュー (インストール プロジェクトでのみ)内の詳細設定から使用できます。エンドユーザーがファイルの関連付けを含む機能をインストールと、ファイルの関連付けはターゲット マシンに登録されます。エントリが、レジストリの適切な場所に作成され、ProgID を通してファイルの種類が アプリケーションにリンク付けされます。ProgID は、時折、ファイル タイプのアプリケーション ID またはタグ名と呼ばれ、アプリケーションを一意に識別し、オペレーティング システムがファイルの関連付けを認識しやすくします。

ファイル拡張子のプロパティ構成の他、ファイル拡張子の追加と削除についての詳細は、次のセクションを参照して下さい:

- 新しいファイル拡張子の追加
- ファイル拡張子の削除
- ファイル拡張子のプロパティを設定する

新しいファイル拡張子の追加



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

アプリケーションがファイルの操作に一意のファイル拡張子を使用する場合、そのファイルの種類を登録することができます。たとえば、アプリケーションが .xyz という拡張子を持つファイルを操作する場合、このファイルの種類を登録しておくと、ユーザーがそのアイコンをダブルクリックしたときにファイルをそのアプリケーションで開くようにオペレーティング システムに指示することができます。



タスク ***新しいファイル拡張子を追加するには、次の操作を実行します。***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。
5. **[ファイルの種類]** エクスプローラーで、**[拡張子]** を右クリックして、**[新しい拡張子]** をクリックします。InstallShield は **New ExtensionN** (ここで *N* は一意の番号です) というデフォルトの名前を持つ新しい拡張子を作成します。
6. 拡張子をピリオドを付けずに入力します (たとえば、**.ext** ではなく、**ext** と入力します)。

ファイル拡張子の削除



プロジェクト: *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*



タスク ***拡張子の削除方法:***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ)、または **[コンポーネント]** ビューを開きます。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。

5. **[ファイルの種類]** エクスプローラーで、拡張子を右クリックし **[削除]** をクリックします。

ファイル拡張子のプロパティを設定する



プロジェクト: *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*

拡張子のプロパティによって、ファイル拡張子のレジストリ エントリ、およびそれに関連付けられる ProgID が決まります。詳しいヘルプは、各プロパティをクリックすると、InstallShield のヘルプ ペインから見ることができます。

ファイル拡張子のプロパティ構成についての詳細は、次のセクションを参照してください:

- コマンド動詞の指定
- ファイル拡張子から動詞を削除する
- 新しい MIME タイプを追加する
- MIME タイプを削除する
- ProgID の作成
- ProgID の削除

コマンド動詞の指定



プロジェクト: *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*

デフォルトで、[開く] という動詞がファイル拡張子に追加されています。



タスク ***新しい動詞を追加してそのプロパティを指定するには、以下の手順を実行します。***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ)、または **[コンポーネント]** ビューを開きます。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。
5. **[ファイルの種類]** エクスプローラーで、拡張子を右クリックし **[新しい動詞]** をクリックします。InstallShield は **New VerbN** (ここで *N* は一意の番号です) というデフォルトの名前を持つ新しい動詞を作成します。
6. 新しい動詞の名前を入力します。**開く、印刷、検索、プロパティ** などのあらゆる 標準的な動詞を使用することができます。
7. プロパティを編集する新しい動詞を選択します。プロパティ シートが右側に開きます。

ファイル拡張子から動詞を削除する



プロジェクト- *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*



タスク ***動詞を削除するには、以下の手順に従います:***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ)、または **[コンポーネント]** ビューを開きます。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。
5. 動詞を右クリックして **[削除]** をクリックします。

新しい MIME タイプを追加する



プロジェクト- *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*



タスク ***新しい MIME タイプを追加するには、次の操作を実行します。***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ)、または **[コンポーネント]** ビューを開きます。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。
5. **[ファイルの種類]** エクスプローラーで、拡張子を右クリックし **[新しい MIME タイプ]** をクリックします。InstallShield は **MIME TypeN** (ここで *N* は一意の番号です) というデフォルトの名前を持つ新しい MIME タイプを作成します。
6. MIME タイプを入力します (たとえば, **image/jpeg**)。

MIME タイプをクリックしてその [クラス ID] プロパティを表示します。[クラス ID] プロパティをクリックして、下のペインで編集します。CTRL+V キー を押し、クラス ID をプロパティ フィールドに貼り付けます。

MIME タイプを削除する



プロジェクト- *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*

タスク ***MIMEの種類を削除するには、以下の手順に従います:***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ)、または **[コンポーネント]** ビューを開きます。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。
5. MIME タイプを右クリックして **[削除]** をクリックします。

ProgID の作成

プロジェクト: *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*

[ファイルの種類] エクスプローラーの [ProgID] 項目には、ファイル拡張子と関連付けるすべてのプログラム識別子が入ります (ProgID は、ファイルの種類のサポート時にはアプリケーション識別子やタグ名とも呼ばれます)。ファイルの種類の ProgID は任意の文字列ですが、ターゲット システム上で一意である必要があります。ProgID の命名規則には、拡張子にドットを付けず、**file** という語を追加する、というものがあります。従って、.ext 拡張子は *extfile* という ProgID ともなりえます。別の命名規則には、SampleApp.Documentのように、特定の種類のファイルを開くのに使用されるアプリケーションの名前をとってファイルの種類 ProgID を命名する、というものがあります。

たとえば、*.xyz* ファイル拡張子は xyzfile progID をポイントすることができ、*.xyz* ファイルの種類情報はすべて xyzfile の下に登録されます。

タスク ***ProgID を作成するには、以下の手順に従ってください。***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ)、または **[コンポーネント]** ビューを開きます。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。
5. **[ファイルの種類]** エクスプローラーで、拡張子をクリックします。拡張子のプロパティが、右のペインで表示されます。
6. **ProgID** プロパティで、ProgID の名前を入力します。ProgID が、**[ファイルの種類]** エクスプローラーの **ProgID** の下に追加されます。
7. **[ファイルの種類]** エクスプローラーで、新しい ProgID をクリックし、そのプロパティを設定します。

ProgID の削除



プロジェクト- *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*



タスク ***ProgID を削除するには、以下の手順を実行します。***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ)、または **[コンポーネント]** ビューを開きます。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。
5. **[ファイルの種類]** エクスプローラーで、拡張子をクリックします。拡張子のプロパティが、右のペインで表示されます。
6. **ProgID** プロパティで、ProgID を削除します。

ダイナミック ファイル リンク

ディレクトリのコンテンツをすべてプロジェクトに追加する場合、ダイナミック ファイル リンクを使用できます。ダイナミック リンクにソース フォルダーを選択すると、ビルド時にそのフォルダー内のファイルがリリースに追加されます。ソース フォルダーは常にビルドの前にスキャンされ、すべての新規または変更ファイルが自動的にリリースへ組み込まれます。ダイナミック ファイル リンクは、フォルダー内のファイル一覧 (および該当する場合、サブフォルダー内のファイル一覧) がビルドとビルドの間で変わる可能性があるとき便利です。



重要- *ダイナミック ファイル リンクの使用は注意が必要です。ダイナミック リンクが参照するソース ファイルからダイナミック リンクがあるファイルを誤って削除してしまった場合、そのファイルは、次回リリースをビルドしたとき、リリースに含まれません。このとき、ビルドの警告やエラーも表示されません。製品は問題なくインストールされる場合がありますが、誤って削除されたダイナミック リンクがあるファイルがインストールされないため、適切に動作しない可能性があります。このため、重要な実行可能ファイル (例、.exe,、dll、.ocx ファイル) にダイナミック ファイル リンクを使用しないことが推奨されます (特に、製品でこれらのファイルが正常に実行される必要がある場合)。*

ダイナミック リンクがあるファイルのコンポーネントを作成するとき、可能なかぎり、ディレクトリごとメソッドではなく、ベスト プラクティス メソッドを使用することをお勧めします。ただし、どちらのメソッドでも、ターゲット イメージにあるファイルがアップグレードまたはパッチのダイナミック リンクから削除されると、マイナー アップグレード、スモール アップデート、またはパッチが正しくインストールされない場合があります。

ダイナミック リンクがあるファイルをフィルターする

ダイナミック ファイル リンクを構成するとき、ダイナミック リンクがあるフォルダーのサブフォルダーを含めるかどうかを指定することができます。ダイナミック リンクがあるファイルをさらに細かくフィルターする場合、ダイナミック リンクに含める、または除外するファイルの特定の名前を指定することができます。また、ワイルド カードを使用して、追加または除外する特定のファイルやファイルの種類のみを指定することができます。

たとえば、すべての画像ファイルがサウンド ファイルと共に 1 つのフォルダーの中にあるとき、画像ファイルのみダイナミック リンクを付加する場合、ダイナミック リンクがあるフォルダーに .bmp ファイルと .ico ファイルのみを含めるように指定することができます。これを行うには、以下の例のように、選択パターンにアスタリスク (*) を使用します:

***.bmp、*.ico**

特定のファイルを選択または除外する場合、選択また除外のパターン ボックスに完全なファイル名を入力します。詳細については、「ダイナミック リンクの作成」を参照してください。

InstallShield インターフェイスで、ダイナミック リンクがあるファイル/フォルダーを静的ファイル/フォルダーから識別する

InstallShield インターフェイスでダイナミック ファイルが表示されるとき、ファイルのアイコンに左隅に、ファイルにダイナミック リンクがあることを示すイメージが表示されます:



同じダイナミック ファイルのイメージが、ダイナミック ファイル リンクも含まれているサブフォルダーのアイコンにも含まれています。



また、ダイナミック ファイル リンクに含まれているサブフォルダーのコンポーネント アイコンにあるフォルダーのイメージには、ダイナミック ファイルのイメージのほかに、矢印も追加されています。



InstallShield インターフェイスが静的ファイルとフォルダーに表示するアイコンは、このダイナミック リンクのイメージを含みません。

このセクションには、次のトピックが含まれています:

- ダイナミック ファイル リンクの制限事項
- ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する
- ダイナミック リンクの作成
- ダイナミック ファイル リンクをコンポーネントに追加する
- ダイナミック ファイル リンクにキー ファイルを設定する

ダイナミック ファイル リンクの制限事項



重要: *ダイナミック ファイル リンクの使用は注意が必要です。ダイナミック リンクが参照するソース ファイルからダイナミック リンクがあるファイルを誤って削除してしまった場合、そのファイルは、次回リリースをビルドしたとき、リリースに含まれません。このとき、ビルドの警告やエラーも表示されません。製品は問題なくインストールされる場合がありますが、誤って削除されたダイナミック リンクがあるファイルがインストールされないため、適切に動作しない可能性があります。このため、重要な実行可能ファイル (例、.exe,、dll、.ocx ファイル) にダイナミック ファイル リンクを使用しないことが推奨されます (特に、製品でこれらのファイルが正常に実行される必要がある場合)。*

ダイナミック リンクがあるファイルのコンポーネントを作成するとき、可能なかぎり、ディレクトリごとメソッドではなく、ベスト プラクティス メソッドを使用することをお勧めします。ただし、どちらのメソッドでも、ターゲット イメージにあるファイルがアップグレードまたはパッチのダイナミック リンクから削除されると、マイナー アップグレード、スモール アップデート、またはパッチが正しくインストールされない場合があります。

Windows Installer ベースのプロジェクトの制限事項

基本の MSI プロジェクト、DIM プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトでダイナミック ファイル リンの使用を検討する際、次の制限事項に留意してください:

- ダイナミック リンク ファイル (DLF) へカスタム アクションを作成することはできません。
- ファイル拡張子を動的にリンクされたファイルに作成することはできません。
- 動的にリンクされたファイルから COM 情報を抽出することはできません。
- "共有"、"パーマネント"、"上書き禁止" といったプロパティを動的にリンクされたファイルに設定することはできません。
- .NET Installer クラス機能を、動的にリンクされたファイルに設定することはできません。
- COM Interop を動的にリンクされたファイルで有効にするように指定することはできません。
- デフォルトのファイル設定 ([読み取り専用]、[非表示] など) を変更することはできません。
- 動的にリンクされたファイルにファイルのアクセス許可を設定することはできません。
- 動的にリンクされたファイルへのショートカットを作成することはできません。
- 動的にリンクされたファイルについて、スタティック スキャンまたはダイナミック スキャンを実行することはできません。
- 動的にリンクされたフォルダーからサブフォルダーを除外することはできません。
- 基本の MSI プロジェクトの場合: SetupCompleteSuccess エンド ユーザー ダイアログからダイナミックにリンクされたファイルを起動することはできません。

プロジェクトに (ダイナミック リンクを使わずに) 直接追加したファイルは、すべて内部名 (FileKey) を持ちます。カスタム アクション、ファイル拡張子、ショートカット、または他の種類のアイテムを作成したとき、この内部名を実際にポイントします。

ダイナミック リンクを使ってファイルをプロジェクトを追加すると、そのファイルは物理的にプロジェクトに追加されていません。つまり、これらのファイルが、カスタム アクション、ファイル拡張子などに関連付けることができる FileKeys を含んでいないことを意味します。

InstallScript ベース プロジェクトの制限事項

InstallScript プロジェクトと InstallScript オブジェクト プロジェクトでは、単一のコンポーネントに、静的ファイルもダイナミック リンクがあるファイルも含めることはできません。

ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield では、ダイナミックにリンクされたファイルのコンポーネントを作成するとき、ベストプラクティスを使用する方法と 1 つのディレクトリに対して 1 つのコンポーネントを配置する 2 つの作成方法が提供されています。

ベスト プラクティス方式を使用する

コンポーネント作成のベスト プラクティスに従うと、ダイナミック リンクの選択と除外のフィルター基準を満たすすべてのファイルに対して、次のタスクがビルド時に実行されます。

- ダイナミック リンクがあるフォルダーにある各ポータブル実行可能 (PE) ファイルについてコンポーネントが別々に作成されます。各 PE ファイルは、そのコンポーネントのキー ファイルです。
- ダイナミック リンクのルート レベルにあるすべての非 PE ファイルがリンクを含むコンポーネントに追加されます。
- ダイナミック リンクにサブフォルダーが含まれている場合、サブフォルダー内にあるすべての非 PE ファイルに新しいコンポーネントが作成されます。ダイナミック リンクに複数のサブフォルダーが含まれている場合、各サブフォルダー内のすべての非 PE ファイルにコンポーネントが 1 つずつ別々に作成されます。

これは、すべての新しいダイナミック リンクについてのデフォルト機能です。



ヒント: *[オプション] ダイアログ ボックスの [ファイルの拡張子] タブでは、PE ファイルとして指定するファイルの種類を選択できます。*

ディレクトリごと方式を使用する

コンポーネントの作成にディレクトリごとメソッドが使用されると、ダイナミック リンクの選択と除外のフィルター基準を満たすすべてのファイルに対して、次のタスクがビルド時に実行されます。

- ファイルの種類に関わらず、ダイナミック リンクがあるソース フォルダーのルート レベルにあるすべてのファイルにコンポーネントが 1 つ作成されます。
- ダイナミック リンクに 1 つまたは複数のサブフォルダーが含まれている場合、ファイルの種類に関わらず、各サブフォルダーのすべてのファイルにコンポーネントが 1 つずつ作成されます。サブフォルダーのコンポーネント内にある最初のダイナミック リンクが付いたファイルが、そのコンポーネントのキー ファイルです。

コンポーネント作成のこのメソッドは、ベスト プラクティス メソッドが導入される前に InstallShield で提供されていた従来型のメソッドです。

利用するコンポーネントの作成方式を判別する

ほとんどのダイナミック リンクの場合、コンポーネントの作成に、ベスト プラクティス メソッドが好まれます。この メソッドを利用すると、ディレクトリごとメソッドを利用したときに比べ、パッチをより簡単に作成することができます。マイナー アップグレードとスモール アップデートの場合、コンポーネント、キー ファイル、および機能–コンポーネントの編成は、以前およびその後の .msi データベース間で保持されている必要があります。パッチの場合、File テーブル キーも保持される必要があります。各コンポーネント名とコンポーネント コード

（および、場合により、キー ファイル）はビルド時にダイナミック ファイル リンクのディレクトリごとメソッドによって変更されるため、問題が発生する場合があります。ベスト プラクティス メソッドの利点は、ディレクトリごとメソッドに比べ、予測可能性がより高くなることです。



ヒント: *アップグレードを構成するとき、ビルドの設定にあるパッチの最適化機能を利用します。パッチの最適化を行うと、コンポーネントの名前、コンポーネント コード、File テーブルのキーおよび Directory テーブルのキーを以前のリリースと統一しやすくなります。詳細については、「アップグレード関する考慮事項」を参照してください。*

初期のバージョンの製品にパッチを作成するとき、初期のインストールにディレクトリごとメソッドを使ったダイナミック リンクを含む場合、同ダイナミック リンクに対してディレクトリごとメソッドを続けて使用する必要があります。ただし、アップグレード プロジェクトに新しいダイナミック リンクを追加した場合、これらの新しいダイナミック リンクに対してベスト プラクティス メソッドを使用することができます。つまり、同じプロジェクトで両方の種類のダイナミック リンクを混合して使用し、アップグレードを配布するパッチを作成できるということです。



重要: *ダイナミック リンクがあるファイルのコンポーネントを作成するとき、可能なかぎり、ディレクトリごとメソッドではなく、ベスト プラクティス メソッドを使用することをお勧めします。ただし、どちらのメソッドでも、ターゲット イメージにあるファイルがアップグレードまたはパッチのダイナミック リンクから削除されると、マイナー アップグレード、スモール アップデート、またはパッチが正しくインストールされない場合があります。*



メモ: *パッケージに含まれているファイルがターゲット システムに既に存在するファイルを上書きするとき Windows Installer が使用する規則については、「ターゲット システム上でファイルとコンポーネントを上書きする」をご覧ください。*

利用するコンポーネントの作成方式を指定する

[コンポーネントのプロパティ] ダイアログ ボックスの [ファイル リンク] タブでは、利用するコンポーネントの作成メソッドを指定することができます。

ダイナミック リンクの作成



プロジェクト: *ダイナミック ファイル リンクを作成する手順は、使用するプロジェクトの種類によって異なります。一部の手順は、次の Windows Installer ベースのプロジェクトに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

一部の手順は、次の InstallScript ベースのプロジェクトに適用します:

- *InstallScript*
- *InstallScript オブジェクト*

[ファイルとフォルダー] ビューを利用して、ダイナミック リンクを作成することができます。



タスク ***ダイナミック リンクを作成するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. [ビュー フィルター] リストで、ダイナミック リンクを持つファイルがあるコンポーネントを含める機能を選択します。
3. コンポーネントが、**[インストール先コンピューターのフォルダー]** ペインに表示されていることを確認してください。

   コンポーネントが表示されない場合: [**インストール先コンピューターのフォルダー**] ペインで、**インストール先コンピューター**を右クリックし、[**コンポーネントの表示**] (Windows Installer ベースのプロジェクトの場合) または、[**コンポーネントとサブフォルダーを表示**] (InstallScript ベースのプロジェクトの場合) をクリックします。
4. Windows Installer ベースのプロジェクトの場合: コンポーネントを右クリックしてから **[ダイナミック ファイル リンク]** を選択します。**[コンポーネントのプロパティ]** ダイアログ ボックスが開き、[ファイルのリンク] タブが表示されます。

   InstallScript ベースのプロジェクトの場合: コンポーネントを右クリックしてから、**[ファイルのリンク]** を選択します。[リンク タイプ] ダイアログ ボックスが開きます。
5. ダイナミック リンクを定義して、**[OK]** をクリックしてください。



ヒント: *[コンポーネント] ビューを利用して、ダイナミック ファイル リンクを追加することもできます。詳細については、「ダイナミック ファイル リンクをコンポーネントに追加する」を参照してください。*

ダイナミック ファイル リンクをコンポーネントに追加する

[コンポーネント] ビューを利用して、ダイナミック ファイル リンクを Windows Installer ベースのプロジェクトと InstallScript ベースのプロジェクトの両方でコンポーネントに追加することができます。ダイナミック ファイル リンクを作成する手順は、使用するプロジェクトの種類によって異なります。

Windows Installer ベースのプロジェクト

次の手順は、基本の MSI プロジェクト、DIM プロジェクト、InstallScript MSI プロジェクトおよびマージ モジュール プロジェクトに適用します。



タスク ***ダイナミック リンクをコンポーネントに追加するには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[コンポーネント]** をクリックします。
2. **[コンポーネント]** エクスプローラーで、ダイナミック ファイル リンクを追加するコンポーネントを展開して、その下にある **[ファイル]** をクリックします。
3. **[ファイル]** ペインで右クリックして、**[ダイナミック ファイル リンク]** をクリックします。**[ダイナミック ファイル リンクの変更]** ダイアログ ボックスが開きます。

4. **[新しいリンク]** をクリックします。[ダイナミック ファイル リンクの設定] ダイアログ ボックスが開きます。このダイアログ ボックスで、リンクのソース フォルダーの設定、サブフォルダーを含めるかどうか、およびファイルが自己登録かどうかの指定を行うことができます。また、選択または除外するファイルの種類も指定できます。
5. **[OK]** をクリックします。ファイルの競合は、既存のファイルを新バージョンで上書きして解決します。上書きするときは **[はい]** を、既存のファイルを残す時は **[いいえ]** を、ダイナミック ファイル リンクの設定を保存しないでダイアログを終了する時は **[キャンセル]** をクリックします。フォルダーにファイルが入っていない場合や、他のダイナミックリンクですでに使用されている場合には、警告が表示されます。

新しいファイル リンクの詳細が [ファイル] ペインに表示されます。

InstallScript ベースのプロジェクト

次の手順は、InstallScript および InstallScript オブジェクト プロジェクトに適用します。

タスク ***ダイナミック リンクをコンポーネントに追加するには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[コンポーネント]** をクリックします。
2. **[コンポーネント]** エクスプローラーで、ダイナミック リンクを含めるコンポーネントをクリックします。
3. **"リンクの種類"** プロパティの値を選択し、省略記号ボタン(...)をクリックします。[リンク タイプ] ダイアログ ボックスが開きます。
4. 含めるファイルを持つフォルダーを指定し、任意のオプションを指定します。
5. **[OK]** をクリックします。

ヒント: *[ファイルとフォルダー] ビューを利用して、ダイナミック ファイル リンクを追加することもできます。詳細については、「ダイナミック リンクの作成」を参照してください。*

ダイナミック ファイル リンクにキー ファイルを設定する

プロジェクト: *この情報は、次の Windows Installer ベースのプロジェクトで、コンポーネントの作成にディレクトリごとに 1 つのコンポーネント メソッド (ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する に説明があります) を使用したダイナミック リンクに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

この情報は、コンポーネントの作成にベスト プラクティス メソッドを使用するダイナミック リンクには適用しません。

コンポーネントの作成にディレクトリごとに 1 つのコンポーネント メソッドが使用されるダイナミック リンクあるファイルを、コンポーネントのキー ファイルに指定できません。

ダイナミックリンクされたコンポーネントにキーファイルを設定するには、目的のファイルにスタティックリンクを張り、次にこのスタティックリンクをコンポーネントのキーファイルに設定します。ダイナミック リンクの設定で、**"次の拡張子のファイルを除外する"** フィールドにキー ファイルの完全名を入力します。

たとえば、ソースディレクトリに複数の .txt ファイルがあり、Key.txt というファイルをキーファイルにする場合を例とします。まず最初に、Key.txt にコンポーネントへのスタティックリンクを張り、Key.txt をキーファイルに設定します。次に、**"次の拡張子のファイルを含める"** の設定を ***.txt** とし、**"次の拡張子のファイルを除外する"** の設定を **Key.txt** として、先のソース フォルダーへのダイナミック リンクを作成します。

サブディレクトリを含むダイナミックリンクの場合は、ビルドのプロセスで、サブディレクトリ内のコンポーネントの最初のファイルが、動的に生成されたコンポーネントのキーファイルに設定されます。

ターゲット システム上でファイルとコンポーネントを上書きする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

実行時、Windows Installer は次の質問事項を考慮して、ターゲット システム上でファイルを上書きするかどうかを決定します:

- コンポーネントをインストールするべきかどうか?
- コンポーネントをインストールする必要がある場合、そのファイルをインストールするのか、またはアップデートするのか?

次のセクションでは、Windows Installer がこれらの質問事項を評価する方法について説明します。

コンポーネントのインストールが必要かどうかを決定する

コスト分析中、Windows Installer はコンポーネントのキー パスを確認して、コンポーネントのインストールが必要かどうかを決定します。コンポーネントがディレクトリまたは ODBC キー パスを持つ場合、Windows Installer はそのコンポーネントをインストールします。さらに、コンポーネントにレジストリ キー パスが使用されていて、以下の条件の両方が True である場合、Windows Installer はコンポーネントをインストールします:

- コンポーネントの "上書きしない" 設定で [いいえ] が選択されている。
- キー パスがターゲット システムに存在しない。

レジストリ キー パスを持つコンポーネントでこれらの条件の 1 つまたは両方が False の場合、Windows Installer は実行時にそのコンポーネントをインストールしません。

コンポーネントがキー パスの代りにキー ファイルを持つ場合、Windows Installer はターゲット システムでそのファイルの存在を確認します。Windows Installer は、以下のすべての状況下でコンポーネントをインストールします:

- ターゲット システムにそのファイルが存在しない。
- ターゲット システムにそのファイルが存在し、コンポーネントのキー ファイルのバージョン番号が、ターゲット システム上のファイルのバージョン番号よりも新しい、またはそれに等しい。
- ターゲット システム上にそのファイルが存在し、コンポーネントのキー ファイルとターゲット システム上のファイルがバージョン指定されていない。

コンポーネントのキーファイルのバージョン番号がターゲット システム上のファイルよりも小さい場合、実行時に Windows Installer はそのコンポーネントをインストールしません。さらに、Windows Installer はコンポーネントのキーファイルにバージョンが指定されていないが、ターゲット システム上のファイルにはバージョンが指定されている場合、実行時にそのコンポーネントをインストールしません。つまり、Windows Installer は実行時にキーファイルがダウングレードされるような場合、コンポーネントをインストールしません。

ファイルのインストールが必要かどうかを決定する

Windows Installer がコンポーネントのインストールが必要であると判断したとき、そのコンポーネントのファイル確認して、それらをインストールする必要があるかどうかを決定します。キー ファイルの名前とインストール先がターゲット システムにあるファイルの名前と場所に一致すると、Windows Installer は、デフォルトで、次の規則を使って、インストール内のファイルでターゲット システムの対応するファイルを上書きするかどうかを判別します:

- **常に上書き** – ファイルに [常に上書き] チェック ボックスが選択されているとき、ターゲット マシンのファイルはバージョン番号に関わらず上書きされます。
- **バージョン付きファイル** – ターゲット システム上のファイルの方がインストールされるバージョンより新しい場合でも、常に最新バージョンのファイルが保持されます。さらに、バージョン指定されていないファイルではなく、バージョン指定されたファイルが保持されます。
- **ファイル言語** – 他の要素が同じ場合、インストールと同じ言語を持つファイルが他言語のバージョンより優先的に保持されます。この規則に対する例外はファイルが複数言語だった場合にのみ適用されます。単一言語のバージョンのファイルに対して複数言語のファイルが保持されます。
- **日付** – ターゲット システム上に既存するファイルの変更日がそのファイルの作成日以降である場合、ファイルは上書きされません。この規則によってアップグレードまたは再インストール中にユーザー環境設定ファイルが上書きされるのを防ぐことができます。

ファイルがインストールされるコンポーネントに含まれている場合で、それがターゲット システムに存在しないとき、Windows Installer はそのファイルを常にインストールします。



メモ: REINSTALLMODE *プロパティの設定により、デフォルトのファイル バージョン規則を変更することができます。詳細については、「ファイルの上書き規則について」を参照してください。*

File Versioning Rules (Windows Installer ヘルプ ライブラリ)

[ファイル] ビューで定義済みフォルダーを表示する



タスク ***定義済みのインストール先フォルダーを表示するには、次の手順に従います:***

1. **[アプリケーション データの指定]** の下にあるビュー リストで、**[ファイル]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、**[インストール先コンピューター]** を右クリックして、**[定義済みフォルダーの表示]** をポイントし、使用する定義済みフォルダーをクリックします。

コンパニオン ファイル

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

コンパニオンファイルを使用することで、複数ファイルのインストールアクションを 1 つにバインドすることができます。たとえば、インストール プロジェクトに **FileA.exe** と **FileB.dat** という 2 つのファイルを含む場合、コンパニオン ファイルを使って **FileB.dat** を **FileA.exe** にバインドすることができ、**FileA.exe** をインストールまたは再インストールするときは、**FileB.dat** も同様に処理されるようになります。また、**FileA.exe** をアンインストールするときは、**FileB.dat** も同様にアンインストールされます。

コンパニオンファイルの使用

この機能は、Windows Installer がデフォルトで用いるファイルのバージョン規則をオーバーライドしたいときに便利です。たとえば、バージョン番号のないファイルに対するバージョン規則では、ターゲットマシン上にあるファイルで、その作成日以降に更新されたものはすべて、ユーザーデータのファイルと見なされ、上書き禁止にされます。しかし、こうした判定は必ずしも妥当でない場合があるため、コンパニオンファイルを使用してバージョン番号のないファイルをバージョン番号のあるファイルにバインドできるようになっています。

コンパニオンファイルの関連付け

タスク ***コンパニオンファイルの関連付けは、次の手順で行います。***

1. **[ファイルとフォルダー]** ビューで、プロジェクトにバージョン付きファイルを追加します。
2. **[コンポーネント]** ビューを開きます。
3. **[コンポーネント]** エクスプローラーで、追加したファイルを含むコンポーネントを展開してから **[ファイル]** をクリックします。そのファイルの**キー**列にある値に注目してください。
4. ファイルを右クリックして、**[追加]** をクリックします。
5. 選択したコンポーネントに追加するバージョンが付いていないファイルを選択します。
6. 2 番目のファイルを右クリックして、**[プロパティ]** をクリックします。[プロパティ] ダイアログ ボックスが開きます。
7. **"システム バージョンをオーバーライドする"** チェック ボックスを選択します。
8. **[バージョン]** ボックスには、ステップ 3 で見た **[キー]** 列の値の名前を入力します。

メモ- *ファイルを追加するたびに InstallShield は一意のファイル キーを生成します。ですから、バインド先に設定したファイルを削除すると、ファイルキーが存続し続けない場合があります。バージョンが付いていないファイルのバージョンをアップデートする必要があります。*

ファイルとフォルダーのアクセス許可を構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield で、ロックダウンされた環境で製品を実行するエンドユーザーのために、ファイルとフォルダーを保護するための設定を構成することができます。ファイルまたはフォルダーのアクセス許可を特定のグループとユーザーに割り当てることができます。たとえば、管理者グループに特定のファイルについての [読み取り]、[書き込み]、および [削除] アクセス許可を割り当てることができますが、別のグループのすべてにユーザーについては [読み取り] 許可のみ割り当てることができます。



タスク

ファイルまたはフォルダーのアクセス許可を構成するには、以下の手順に従います:

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. ファイルの場合: **[インストール先コンピューターのファイル]** ペインで、ファイルを右クリックして**[プロパティ]** を選択します。**[プロパティ]** ダイアログ ボックスが開きます。

   フォルダーの場合: **[インストール先コンピューターのフォルダー]** ペインで、フォルダーを右クリックして**[プロパティ]** を選択します。**[プロパティ]** ダイアログ ボックスが開きます。
3. **[アクセス許可]** ボタンをクリックします。**[アクセス許可]** ダイアログ ボックスが開きます。
4. 必要に応じて、アクセス許可を追加/変更/削除します。詳細については、「ファイルとディレクトリの [アクセス許可] ダイアログ ボックス」を参照してください。

プロジェクトの [一般情報] ビューにある "ロックダウンの設定方法" 設定の選択に従って、InstallShield は ISLockPermissions テーブルまたは LockPermissions テーブルのどちらかにアクセス許可データを追加します。詳細については、「ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。



ヒント: *コンポーネントのインストール先フォルダーのアクセス許可を構成することもできます。構成を行うには、まず [コンポーネント] ビューでコンポーネントをクリックします。次に "保存先のアクセス許可" 設定の値をクリックして、省略記号 (...) ボタンをクリックします。[アクセス許可] ダイアログ ボックスが開きます。このダイアログ ボックスで、必要なアクセス許可を構成できます。*

ファイルの追加時に COM データを抽出する

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

[ファイルとフォルダー] ビュー、またはコンポーネントの下の [ファイル] サブビューに新しいファイルをドラッグアンドドロップすると、自己登録ファイルの COM データを静的に抽出できます。これは、ビルド時に COM 登録データを抽出しなかった場合の代替方法です。

このオプションは、[コンポーネント] ビューの [詳細設定] の [COM 登録] ノードからも使用できます。

ファイルの COM データを抽出する

タスク ***COM データをビルド時ではなく、オン デマンドで抽出するには、以下の手順に従います:***

ファイルを右クリック(または **COM 登録**ノードで **[COM 登録]** を右クリック)して、**[キーファイルの COM データを抽出]** をクリックします。

メモ- *このオプションは、ファイルが .exe ファイルか自己登録で、ファイルがコンポーネントのキーファイルである場合のみ有効になります。*

ヒント- *InstallShield を 64 ビット システムで使用している場合、InstallShield は 64 ビット COM サーバーから COM データを抽出できます。データを正しい場所にインストールするため、コンポーネントを 64 ビットとマークしなくてはなりません。64 ビット サポートに関する詳細は、「64 ビット オペレーティング システムをターゲットにする」を参照してください。*

ファイルの COM データを更新する

ファイルの COM データが既に静的に抽出されている場合、[キーファイルの COM データを更新する] を選択できます。COM データを更新すると、古い COM データはプロジェクトから削除されて、ファイルの現在の COM データがプロジェクトに追加されます。

自己登録ファイルのインストール

インストーラーは、ファイルが自己登録に指定されていて (Windows Installer ベースのプロジェクトの場合)、"ビルド時に COM 抽出" プロパティが [いいえ] に設定されていると、「従来の」自己登録方法を使用して自己登録ファイルを登録します。ファイルはアンインストール時に登録解除されます。サポートされている自己登録関数は **DllRegisterServer** および **DllUnregisterServer** です。

ファイルの "登録" プロパティを設定する前に、ファイルが自己登録に指定されていることを確認してください。セットアップ ベスト プラクティスに従い、.dll、.ocx、または .exe は 1 つのコンポーネントにつき 1 つだけにする必要があります。



ヒント: *Windows Installer を使って自己登録ファイルをインストールするには、.msi データベース テーブル (クラス、ProgID およびその他) に登録情報を書き込む方法をお勧めします。ファイルを自己登録にしない場合、代わりに、コンポーネントの詳細設定を使用するか、またはCOM 情報をビルド時に抽出するか、あるいは[ファイルとフォルダー] ビューにファイルを追加したときに COM 情報を抽出して、progIDs やタイプライブラリなどを登録できます。詳細設定は、インストール中にファイルをインストールする場合でも、「ジャスト イン タイム インストール」としてファイルをアドバタイズする場合でも有効です。その他に有利な点は、インストールが失敗した場合にファイルの登録を解除できることです。*

コンポーネントの [ビルド時に COM 抽出] プロパティを いいえ に設定しても、COM 登録の詳細設定に格納されている情報はインストール時に登録されます。COM 登録の詳細設定とコンポーネントのレジストリデータをチェックすることにより、予定通りのエントリが使用されており自己登録機能で作成されたエントリと競合していないことを確認することができます。

.NET アセンブリのプロパティおよび依存関係を識別する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield は、.NET アセンブリのプロパティおよび依存関係の識別をサポートします。使用するプロジェクトの種類によって、その機能は異なります。

Windows Installer ベースのプロジェクト

基本の MSI、DIM、InstallScript MSI、およびマージ モジュール プロジェクトでは、プロジェクトに .NET アセンブリを追加する際に、それをコンポーネントのキー ファイルとして設定することが推奨されます。その他の .NET アセンブリは、そのコンポーネントに含まないでください。

.NET アセンブリがそのコンポーネントのキーファイルである場合、以下の方法の 1 つを使ってその依存関係を識別できます:

- .NET アセンブリで必要なファイルとマージ モジュールが必要が判明している場合、プロジェクトの .NET アセンブリ コンポーネントを含む機能にそれらを手動で追加することができます。プロジェクトにどのファイルを含めるのかを大幅に制御できるこの方法が推奨されます。
- .NET アセンブリに必要なファイルおよびマージ モジュールが不明な場合には、以下のビルトイン処理の 1 つを行ってください:
  - .NET アセンブリの依存関係の可能性があるファイルをオンデマンドで識別するには、スタティック スキャン ウィザードを使います。このウィザードで検出された依存関係のリストが表示され、それぞれを

プロジェクトに含めるかどうかを指定することができます。この方法は基本の MSI および InstallScript MSI プロジェクトで使用できますが、DIM またはマージ モジュール プロジェクトでは使用できません。

- プロジェクトをビルドする度に .NET アセンブリの依存関係を識別するには、コンポーネントの "ビルド時に .NET をスキャン" 設定で [依存関係およびプロパティ] を選択します。InstallShield がビルド時に不足している可能性のある依存関係を検出した場合、InstallShield が生成するリリースにはそれが組み込まれます。

.NET アセンブリのプロパティを構成することで、Windows Installer が必要に応じてアセンブリをアップデートおよびアンインストールできるようになります。.NET アセンブリがそのプロパティのキーファイルである場合、以下の方法のいずれかを使ってその依存関係を識別できます:

- 手動でプロパティおよびその値を入力する: [コンポーネント] ビューで、.NET アセンブリを含むコンポーネントの [詳細設定] 領域にある [アセンブリ] ノードの下の [.NET アセンブリ] サブノードを選択してから、必要に応じてプロパティを構成します。入力するプロパティと値は、アセンブリのマニフェスト ファイル内の情報と一致しなくてはなりません。一致しなかった場合、.NET コンポーネント含まれる機能がアンインストールされる時に、アセンブリがターゲット システムに残る可能性があります。
- プロジェクトをビルドする度に .NET アセンブリのプロパティを識別するには、コンポーネントの "ビルド時に .NET をスキャン" 設定で [プロパティ] のみまたは [依存関係およびプロパティ] を選択します。InstallShield がビルド時に不足している可能性のあるプロパティを検出した場合、InstallShield が生成するリリースにはそれが組み込まれます。

InstallScript プロジェクト

InstallScript プロジェクトの場合、プロジェクトのコンポーネントに .NET アセンブリを追加するとき、コンポーネントの ".NET アセンブリ" 設定を使って、アセンブリがローカル .NET アセンブリであることを識別することができます。また、以下の方法の 1 つを使って、.NET アセンブリの依存関係を識別できます:

- .NET アセンブリで必要なファイルとマージ モジュールが必要が判明している場合、プロジェクトの .NET アセンブリ コンポーネントを含む機能にそれらを手動で追加することができます。プロジェクトにどのファイルを含めるのかを大幅に制御できるこの方法が推奨されます。
- .NET アセンブリに必要なファイルおよびマージ モジュールが不明な場合には、以下のビルトイン処理の 1 つを行ってください:
  - .NET アセンブリの依存関係の可能性があるファイルをオンデマンドで識別するには、スタティック スキャン ウィザードを使います。このウィザードで検出された依存関係のリストが表示され、それぞれをプロジェクトに含めるかどうかを指定することができます。
  - プロジェクトをビルドする度に .NET アセンブリの依存関係を識別するには、コンポーネントの "ビルド時に .NET をスキャン" 設定で [依存関係] を選択します。InstallShield がビルド時に不足している可能性のある依存関係を検出した場合、InstallShield が生成するリリースにはそれが組み込まれます。

コンポーネントの ".NET アセンブリ" 設定で [アセンブリではない] が選択されている場合、InstallShield はコンポーネントのアセンブリの依存関係をスキャンしません。

コンポーネントの ".NET アセンブリ" 設定で [ローカル アセンブリ] が選択されている場合、インストールは実行時にコンポーネントの COM の相互運用機能への登録および .NET Installer クラス情報の構成を行います。

依存関係の 64 ビット .NET アセンブリをスキャンする



プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

Windows Vista 以降の 64 ビット バージョン、または Windows Server 2008 以降の 64 ビット バージョンで InstallShield を使用していて、以下のどちらかのビルトイン処理によって依存関係を検出する場合、InstallShield はプロジェクトに含まれる 64 ビット .NET アセンブリの 64 ビット依存関係をスキャンすることができます。

- 64 ビット .NET アセンブリの依存関係の可能性があるファイルをオンデマンドで識別するには、スタティック スキャン ウィザードを使います。このウィザードで検出された依存関係のリストが表示され、それぞれをプロジェクトに含めるかどうかを指定することができます。
- プロジェクトをビルドする度に 64 ビット .NET アセンブリの依存関係を識別するには、コンポーネントの "ビルド時に .NET をスキャン" 設定で [依存関係] オプションの 1 つを選択します。InstallScript プロジェクトの場合、コンポーネントの ".NET アセンブリ" 設定を [ローカル アセンブリ] に設定しなくてはなりません。InstallShield がビルド時に不足している可能性のある依存関係を検出した場合、InstallShield が生成するリリースにはそれが組み込まれます。

これらの処理は、プロジェクトに含まれる 32 ビット .NET アセンブリの 32 ビット依存関係のスキャンも行います。

Windows の 32 ビット バージョンで InstallShield を使用する場合、これらのビルトイン スキャンはプロジェクトに含まれる 32 ビット ファイルの 32 ビット依存関係のみを検出できます。プロジェクトに 64 ビット ファイルが含まれている場合、必要に応じてプロジェクトに依存関係を手動で追加することができます。

Windows Installer ベースのプロジェクトで、ビルトイン スキャン処理がプラットフォーム依存またはプラットフォーム非依存 .NET 依存関係を検出する方法

基本の MSI、DIM、InstallScript MSI およびマージ モジュール プロジェクトで、前述のビルトイン処理を使って .NET 依存関係を検出する場合、InstallShield は次の表に示される特定の順序に従って検出を行います。

テーブル 3 ・ .NET アセンブリの依存関係を検出する順序

スキャンが行われるファイルの種類	32 ビット Windows システム上のスキャン結果	64 ビット Windows システム上のスキャン結果
GAC 内のプラットフォーム非依存 .NET アセンブリ	InstallShield が依存関係をスキャンする順序: 1. 32 ビット固有 .NET 依存関係 2. プラットフォーム非依存 .NET 依存関係	InstallShield が依存関係をスキャンする順序: 1. 64 ビット固有 .NET 依存関係 2. プラットフォーム非依存 .NET 依存関係

テーブル 3 ・ .NET アセンブリの依存関係を検出する順序 (続き.)

スキャンが行われるファイルの種類	32 ビット Windows システム上のスキャン結果	64 ビット Windows システム上のスキャン結果
GAC 内の 64 ビット .NET アセンブリ	InstallShield は、依存関係を検出しません。	InstallShield が依存関係をスキャンする順序: 1. 64 ビット固有 .NET 依存関係 2. プラットフォーム非依存 .NET 依存関係
GAC 内の 32 ビット .NET アセンブリ	InstallShield が依存関係をスキャンする順序: 1. 32 ビット固有 .NET 依存関係 2. プラットフォーム非依存 .NET 依存関係	InstallShield が依存関係をスキャンする順序: 1. 32 ビット固有 .NET 依存関係 2. プラットフォーム非依存 .NET 依存関係

.NET アセンブリが GAC に存在しない場合、InstallShield は依存関係をスキャンするときにアセンブリが含まれているフォルダーを検索してから、サブフォルダーを検索します。

.NET 依存関係スキャナー結果の確認



プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield では、様々な .NET アセンブリの依存関係を識別するためのビルトイン処理 が提供されています。これらのビルトイン処理を使用すると、スキャンによって製品が必要としないファイルが依存関係またはマージ モジュールとして識別される可能性があります。スタティック スキャン ウィザードを使用する際にこの状況が発生した場合、プロジェクトに特定の依存関係を追加しないことをウィザードで指定することができます。ビルド時に .NET 依存関係の可能性があるすべてのファイルをプロジェクトに自動的に含める、[ビルド時に .NET をスキャン] 機能を使用する場合、.NET アセンブリが自動的にスキャンされないように、コンポーネントの設定値を変更することができます。

- 基本の MSI、InstallScript MSI およびマージ モジュール プロジェクトの場合、InstallShield はデフォルトでビルド時にコンポーネントの .NET アセンブリの依存関係を自動的にスキャンします。必要に応じて、コンポーネントの "ビルド時に .NET をスキャン" 設定の値を変更して、このデフォルト動作を上書きすることができます。
- InstallScript プロジェクトの場合、InstallShield はデフォルトでビルド時に各コンポーネントの .NET アセンブリの依存関係をスキャンしません。必要に応じて、コンポーネントの ".NET アセンブリ" 設定およびその "ビルド時に .NET をスキャン" 設定の値を変更して、このデフォルト動作を上書きすることができます。

さらに、InstallShield を使って依存関係スキャンを行うときは常に自動的に含めたり除外したりするファイルをマシン全体で指定することも可能です。詳細については、「依存関係スキャナーでファイルをフィルターする」を参照してください。

依存関係を識別するときに最も良い結果を得るために、クリーン マシン上で製品とそのインストールを十分にテストすることが推奨されます。製品が予定通りに動作しなかった場合、マシン上で足りない依存関係がないか、またそれをインストールに含むべきかどうかを判断してください。

InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトでフォントをインストールする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

([コンポーネント] ビューまたは[ファイルとフォルダー] ビューを使用して) インストール プロジェクトにフォント ファイル(.ttf、 .ttc または .fon ファイル) 含めるとき、グローバルフォント登録を有効にしておくと、ファイルは自動的にターゲットマシンに登録されるよう内部的にマークされます。実行時に、**OnInstalledFontFile** イベント ハンドラー関数が各フォントファイルのインストールされた直後に呼び出されます。フォントが登録されるよう内部的にマークされている場合、このイベント ハンドラー関数のデフォルトコードは、**RegisterFontResource** 関数を呼び出してフォントを登録します。

フォントタイトル

InstallShield は、インストールが登録するフォントタイトルを以下のように判断します。

- .fon ファイルの場合、InstallShield は、デフォルトで、フォント タイトルをソース システムのレジストリから読み込みます。タイトルがレジストリに見つからなかった場合、InstallShield は、ファイル名をフォント タイトルとして格納します。"リンク イブ" プロパティに [スタティック] が設定されているコンポーネントにある .fon ファイルの場合、InstallShield はそのファイルの [ファイルのプロパティ] ダイアログ ボックスの [フォントのタイトル] ボックスにフォントのタイトルを表示します。
- TrueType ファイル (.ttf または .ttc ファイル)の場合、デフォルトでは、インストールは実行時にそのファイルからフォント タイトルを読み込みますが、デザイン時には、[フォント タイトル] ボックスにはテキストは表示されません。
- 3つすべてのタイプのファイルにおいて、ファイルが リンクタイプ プロパティに スタティックが設定されているコンポーネントにあるとき、[フォントタイトル] ボックスに非デフォルトテキストを入力した場合、インストールはそのテキストをフォントタイトルとして登録します。

Windows Fonts Folder にフォントをインストールする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*

- *InstallScript オブジェクト*



タスク ***次のいずれかを実行して、Windows Fonts フォルダーにフォントファイルをインストールします:***

- **[ファイルとフォルダー]** ビューで、**[インストール先コンピューターのフォルダー]** ペインの **Fonts Folder** フォルダーにフォント ファイルを置きます。Fonts Folder フォルダーは、**Windows** フォルダーのサブフォルダーです。
- **[コンポーネント]** ビューで、**"インストール先"** プロパティに **<FOLDER_FONTS>** が設定されているコンポーネントにフォント ファイルを置きます。

グローバル フォント登録を有効または無効にする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

フォントファイルの自己登録をグローバルに有効または無効にすることができます。



タスク ***グローバルフォント登録を有効または無効にするには、以下の手順を実行します。***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"フォントの登録"** 設定で、適切なオプションを指定します:
   - グローバル フォント登録を有効にするには、**[有効]** を選択します。
   - グローバル フォント登録を無効にするには、**[無効]** を選択します。

フォント ファイルの登録を有効または無効にする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

個々のファイルに対して、フォント ファイルの自己登録を有効または無効にすることができます。



タスク ***個々のフォントファイルの自己登録を有効または無効にするには、以下の手順を実行してください。***

1. **[コンポーネント]** ビュー、**[セットアップのデザイン]** ビューまたは **[ファイルとフォルダー]** ビューを開きます。
2. フォント ファイルを右クリックして、**プロパティ**をクリックします。**[プロパティ]** ダイアログ ボックスが開きます。

3. [**フォントファイルを登録する**] チェック ボックスを選択して、フォントファイルの自己登録を有効にします。

   自己登録を無効にするには、[**フォント ファイルの登録**] チェック ボックスをクリアしてください。

4. [OK] をクリックします。

コンポーネントを使用する

インストール開発者の視点からみると、コンポーネントはアプリケーションの要素です。コンポーネントはエンドユーザーには表示されません。ユーザーがインストール用の機能を選択すると、インストーラーによってその機能に関連したコンポーネントが判断され、それらのコンポーネントがインストールされます。アプリケーションのコンポーネントには実行可能バイナリファイル、データファイル、ショートカット、ヘルプ システムファイル、およびレジストリ エントリが含まれます。

コンポーネントは、[セットアップのデザイン] ビュー(インストール プロジェクトの場合)または [コンポーネント] ビューで修正できます。

コンポーネントと機能の関係



プロジェクト: *機能は DIM プロジェクトまたはマージ モジュール プロジェクトでは使用されません。*

コンポーネントは、[セットアップのデザイン] ビューで機能に関連付けられます。詳細については、「新しいコンポーネントを機能に関連付ける」を参照してください。

ファイル

[セットアップのデザイン] ビューまたは [コンポーネント] ビューでコンポーネントのツリーを展開して、ファイル をクリックし、コンポーネントに関連付けられた全ファイルの一覧を表示します。ファイルの追加方法に関する情報は、「ファイルをコンポーネントに追加する」をご覧ください。

セットアップ ベスト プラクティス



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield は、[セットアップのデザイン] ビューおよびコンポーネント ウィザードでコンポーネントを作成中に発生したベスト プラクティス違反に対して警告を行い、セットアップ ベスト プラクティスの厳守を容易にします。これらのガイドラインに従うことにより、再使用可能なコンポーネントを有効に配布して、ファイルの非互換性問題を伴わないクリーンなインストールを作成することができます。

[セットアップのデザイン] ビューでコンポーネントを作成する際、セットアップベストプラクティスウィザードは、コンポーネントに追加するファイルを監視し、ベストプラクティスとの競合がある場合に警告し、ウィザードでアクションを修正できるようにします。ウィザードのセットアップのデザインの自動スキャン機能をオフにするには、[ツール] メニューの [オプション] を選択します。

InstallShield は、次のセットアップ ベストプラクティスに準拠しているかどうかを確認します。

テーブル 4 ・ InstallShield がモニターするセットアップベストプラクティス

ベスト プラクティス	説明
コンポーネントに複数の .exe、.dll、.ocx、.chm、または .hlp ファイルを含めない	各コンポーネントに含めるのは、ポータブル実行可能ファイル (.exe、.dll、または .ocx ファイル) または WinHelp ファイル (.hlp ファイル) の 1 つだけにする必要があります。Windows Installer のコンポーネントは、GUID コード など、すべての詳細設定とコンポーネントの設定が、理想的には単一のポータブル実行可能ファイルまたはヘルプ ファイルを参照するように設計されています。他の .exe、.dll、.ocx、または .hlp ファイルを新しいコンポーネントに配置します。 この理由は、インストール実行時にユーザーがインストール済み製品に対する修復を選択した場合、Windows Installer は、インストール済みの各コンポーネントのキー ファイルの存在をチェックするためです。そして欠落しているキーファイルがあれば、Windows Installer はコンポーネントを再インストールします。そのため、コンポーネントのキーファイルではないファイルが欠落してる場合、修復モードでは復旧されない可能性があります。
マージ モジュールの使用	マージ モジュールには、個別の機能をインストールするために必要な、すべてのファイル、レジストリ エントリ、および論理が含まれています。マージ モジュールが可能なファイルを配布しないでください。また、マージ モジュールを使用すると、1 つのファイルを複数のコンポーネントに関連付けないというベストプラクティスと、いかなるコア コンポーネントも外に出さないという Windows ロゴ ガイドラインの 2 つの関連要件に準拠することができます。 ベストプラクティスをモニターするオプションを有効にすると、InstallShield が提供するマージ モジュールの一部となっているファイルをコンポーネントに追加したとき、常にセットアップベストプラクティスウィザードが警告を発します。

InstallShield では自動的に警告が出ませんが、以下のコンポーネント作成におけるセットアップベストプラクティスについても注意してください。

テーブル 5 ・ InstallShield が自動的に警告を発しないセットアップ ベスト プラクティス

ベスト プラクティス	説明
ショートカット ターゲットをそれ自身のコンポーネントへ配置する	ショートカットのターゲットになるファイルは、それ自身のコンポーネントが必要です。このファイルは、コンポーネントのキーファイルである必要があります。

テーブル 5 • InstallShield が自動的に警告を発しないセットアップ ベスト プラクティス (続き.)

ベスト プラクティス	説明
他のファイルをコンポーネントへグループ化する	共通のインストール先フォルダーやバージョン確認などファイルの要件に従って、上記のどのコンポーネントのカテゴリにも該当しないすべてのファイルを整理してください。
ファイルを自己登録しない	自己登録機能を呼び出して COM サーバー情報の登録および登録解除を行う代わりに、インストール時にコンポーネント用のこの情報を登録するようにしてください (インストーラーはアンインストール時にこの登録を解除します)。 自己登録はファイルの登録や登録解除には信頼性に欠ける方法です。インストールが情報を登録することの 1 つの利点は、ファイルがアドバタイズされるかすぐにインストールされない場合に、後でインストーラーによりファイルが要求されたときでも、登録情報をすぐに取得できるということです。このため、InstallShield は COM サーバーの登録についてだけ、詳細設定をサポートします。

コンポーネントの作成

InstallShield でのコンポーネントの作成には、いくつかの方法があります。

- [セットアップのデザイン] ビュー (インストール プロジェクトのみ) または [コンポーネント] ビューを使って、新しいコンポーネントを作成してから手動で構成する。詳細については、「[セットアップのデザイン] ビューまたは [コンポーネント] ビューを使ってコンポーネントを作成する」を参照してください。
- コンポーネント ウィザードを使用して、InstallShield が自動的にセットアップ ベスト プラクティスに従ってコンポーネントを定義付ける。このウィザードは、基本の MSI、DIM、InstallScript、InstallScript MSI、およびマージ モジュール プロジェクト タイプで利用できます。詳細については、「コンポーネント ウィザードのベスト プラクティス オプションを使用する」を参照してください。
- COM サーバーまたはフォント用のコンポーネントを作成する場合、コンポーネント ウィザードを使用することが推奨されます。このウィザードは、基本の MSI、DIM、InstallScript、InstallScript MSI、およびマージ モジュール プロジェクト タイプで利用できます。詳細については、「コンポーネント ウィザードの [コンポーネント タイプ] オプションを使用する」を参照してください。

コンポーネントをプロジェクトにエクスポートする手順は、「他のプロジェクトへコンポーネントをエクスポートする」を参照してください。

プロジェクト内のコンポーネントを削除するには、「プロジェクトのコンポーネントを削除する」を参照してください。

[セットアップのデザイン] ビューまたは [コンポーネント] ビューを使ってコンポーネントを作成する

次に説明するように、[コンポーネント] ビューで新しいコンポーネントを作成して機能と関連付けたり、[セットアップのデザイン] ビューで特定の機能のコンポーネントを作成することができます。[セットアップのデザイン] ビューは、DIM プロジェクトまたはマージ モジュール プロジェクトには使用できません。

[セットアップのデザイン] ビューまたは [機能] ビューの InstallShield オブジェクト インストーラー機能の上に配置する機能に必要なファイルを含めてください。

インストール プロジェクトのみ、以下の手順に従って [セットアップのデザイン] ビューでコンポーネントを作成します。



タスク ***[セットアップのデザイン] ビューで、コンポーネントを作成するには、次の手順に従います。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. **[セットアップのデザイン]** エクスプローラーで、機能またはサブ機能を右クリックして、**[新しいコンポーネント]** をクリックするか、CRTL+INSERT を押します。InstallShield が、**New Component n** (ここで *n* は連続番号です) というデフォルトの名前で新しいコンポーネントを追加します。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。

選択した機能と新規コンポーネントが自動的に関連付けられます。

[コンポーネント] ビューを使用して、コンポーネントを作成する

DIM プロジェクトとマージ モジュール プロジェクトの場合、以下の手順に従って [コンポーネント] ビューでコンポーネントを作成します。



タスク ***[コンポーネント] ビューでコンポーネントを作成するには、以下の手順を実行します。***

1. **[編成]** の下のビュー リストにある **[コンポーネント]** をクリックします。
2. **[コンポーネント]** エクスプローラーを右クリックして、**[新しいコンポーネント]** をクリックするか、INSERT を押します。InstallShield が、**New Component n** (ここで *n* は連続番号です) というデフォルトの名前で新しいコンポーネントを追加します。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。
4. インストール プロジェクトの場合、1 つまたは複数の機能とコンポーネントの関連付けを行う必要があります。



メモ *インストール プロジェクトでは、機能に関連付けられていないコンポーネントは、親のないコンポーネントのアイコン () で表示されます。*

InstallScript プロジェクトでは次の文字をコンポーネント名に使用できません:

*¥ / : * ? " ' < > |*

コンポーネントを作成すると、そのプロパティ シートが右側に表示されます。必須のコンポーネント コード (Windows Installer ベースのプロジェクトのみ) やインストール先プロパティなど、コンポーネントのプロパティを直ちに指定します。

コンポーネントを作成しプロパティを指定した後、これに、アプリケーションのファイルやレジストリ エントリ、ショートカットをコンポーネントに関連付けることができます。

コンポーネント ウィザードのベスト プラクティス オプションを使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントを作成する最も速い方法は、コンポーネントウィザードを起動して InstallShield によってファイルをコンポーネントに構成する方法です。**"ベストプラクティスを使用してコンポーネントを作成する"** オプションを選択すると、ウィザードがセットアップ ベスト プラクティス を指定されたファイルに適用して、コンポーネントを作成します。



タスク ***コンポーネント ウィザードを起動するには、以下のいずれかを実行します。***

- インストール プロジェクトのみ: [**セットアップのデザイン**] ビューの機能を右クリックして、**コンポーネントウィザード**を選択します。
- インストール プロジェクト、DIM プロジェクト、およびマージ モジュール プロジェクトの場合: **[コンポーネント]** ビューのエクスプローラーで **[コンポーネント]** を右クリックしてから、**[コンポーネント ウィザード]** を選択します。

コンポーネント作成におけるベスト プラクティス規則



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネント ウィザードを使ってコンポーネントを作成するときに [ベストプラクティス] オプションを選択すると、ウィザードがセットアップ ベスト プラクティスに基づいて、自動的にファイルをコンポーネントを編成します。その場合、次のルールにしたがってコンポーネントを作成します。

- ウィザードは、追加されるすべてのファイルが既にプロジェクトにインクルードされていないか調べます。複製ファイルの場合、ウィザードはこのファイルのコンポーネントを作成しません。

- 新しいコンポーネントは、各 ポータブル実行可能ファイル (.exe、.dll、または .ocx ファイル)に対して作成する必要があります。コンポーネントには、ポータブル実行可能形式の名前が付きます。ウィザードは GUID を [コンポーネントコード] プロパティに作成し、ポータブル実行可能ファイルを、コンポーネントのキーファイルとして設定します。
- コンポーネントウィザードは、各々のポータブル実行可能形式の登録情報を抽出しようとします。正常に抽出されると、ウィザードは COM サーバーコンポーネントを作成して、データをコンポーネントの COM サーバーの詳細設定に書込みます。
- 指定されるすべてのヘルプ (.hlp) ファイルはそのコンポーネントの中に、関連のコンテンツ (.cnt) ファイルと共に存在します。コンポーネントには、そのヘルプファイルの名前がつけられます。ウィザードは GUID を [コンポーネントコード] プロパティに作成して、ヘルプファイルをコンポーネントのキーファイルとします。コンポーネントを伴う HTML ヘルプ(.chm) ファイルと .chi ファイルにも同じ規則が適用されます。
- コンポーネントウィザードは、すべてのフォント (.ttf と .ttc) ファイルを、フォントファイルと呼ばれる1つのコンポーネントに挿入します。ウィザードは GUID を [コンポーネントコード] プロパティに作成し、リストの最初のファイルをキーファイルに設定します。指定する .fon ファイルは、作成される AllOtherFiles コンポーネントに自動的に含まれます。ただし .fon ファイルにはタイトルが付かないので、これらは .msi パッケージのフォントテーブルに追加されません。これらのファイルをフォントとして追加するには、コンポーネント ウィザードの"フォント オプション"を使います。
- 上記のカテゴリに含まれないファイルは、「その他のファイル」という名の1つのコンポーネントに分類されます。ウィザードは GUID を [コンポーネントコード] プロパティに作成し、リストの最初のファイルをキーファイルに設定します。

InstallShield による COM サーバー コンポーネントの作成



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネント ウィザードの [ようこそ] パネル内の **[ベストプラクティスを使用してコンポーネントを作成する]** オプションを選択した場合、InstallShield は指定した各ポータブル実行可能ファイルに対して個々にコンポーネントを自動作成します。ウィザードはまた、各々のポータブル実行可能形式ファイルの登録情報を抽出します。

正しく抽出されると、COM 登録詳細設定にマップする登録情報のすべてがそこに書き込まれます。ウィザードは、InProcServer32 ThreadingModel 値など、すべての追加エントリをコンポーネントのレジストリ データに作成します。



メモ- *COM サーバーが実行可能な場合、値 1 の OLESelfRegister 属性を、コンポーネントのリソースファイルのバージョンセクションに置く必要があります。*

InstallShield によるフォント コンポーネントの作成



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネント ウィザードの [ようこそ] パネル内の **[ベスト プラクティスを使用してコンポーネントを作成する]** オプションを選択した場合、InstallShield は指定したすべてのフォント ファイル (.ttf および .ttc) に対して個々にコンポーネントを自動作成します。すべての .fon ファイルは、埋め込みフォントタイトルを持っていないために作成される AllOtherFiles コンポーネントに追加されます。.fon ファイルをセットアップにフォントとして追加するには、コンポーネントウィザードの "フォント オプション" を使用します。

フォントコンポーネント用のインストール先フォルダーは、コンポーネントウィザードで選択したインストール先フォルダーに設定されます。コンポーネントを作成した後、そのインストール先フォルダーをフォント用の通常のデフォルトの場所である Windows Installer フォルダー プロパティ [FontsFolder](つまり、[FontsFolder] をインストール先として使う)に設定できます。FontsFolder の両側に [INSTALLDIR] を指定するときと同様に角かっこをつけます。



タスク ***システムにフォントが登録されていない場合は、以下の手順でフォントタイトルを指定する必要があります。***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、フォント ファイルを含むコンポーネントを展開し、**ファイル**をクリックします。**[ファイル]** ビューが開きます。
3. フォント ファイルを右クリックして、**プロパティ**をクリックします。**[プロパティ]** ダイアログ ボックスが開きます。
4. **[フォント タイトル]** ボックスで、**FontName (FontType)** の形式で、フォントタイトルを入力します (例、**Roman (All res)**)。
5. **[OK]** をクリックします。

コンポーネント ウィザードの [コンポーネント タイプ] オプションを使用する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

特殊なインストール要件を持つファイルがある場合、コンポーネントウィザードの **[タイプを選択してコンポーネントを定義する]** オプションを使用して、これらのファイルをコンポーネントにまとめることをお勧めします。



タスク ***コンポーネント ウィザードを起動するには、以下のいずれかを実行します。***

- インストール プロジェクトのみ: [**セットアップのデザイン**] ビューで機能を右クリックして、**コンポーネントウィザード**を選択します。
- インストール プロジェクト、DIM プロジェクト、およびマージ モジュール プロジェクトの場合: **[コンポーネント]** ビューで **[コンポーネント]** エクスプローラーを右クリックしてから、**[コンポーネント ウィザード]** を選択します。

コンポーネント ウィザードの**タイプを選択してコンポーネントを自分で定義する** オプションは、次のコンポーネント 他オプションをサポートします。

- COM サーバー
- フォントファイル
- サービスのインストール
- サービスのコントロール



メモ: *[サービス] ビューを使って、インストールまたはアンインストール中にサービスのインストール、開始、停止、または削除を行うコンポーネントを作成できます。また、このビューを使って、コンポーネント ウィザードでは構成不可能な、拡張サービス カスタマイズ オプションを構成できます。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。*

COM サーバーをインストールする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントを作成する方法はいくつもありますが、COM サーバ (.exe、.dll または .ocx) ファイルをセットアップにインストールするには、コンポーネント ウィザードで COM サーバーのコンポーネントを作成することをお勧めします。

表示されるウィザードのパネルは、前のパネルで選択した内容によって異なります。たとえば、ウィザードが [COM サーバー実行可能ファイル] パネルの登録情報を正常に抽出できた場合は、[クラス] パネルは表示されません。



ヒント: *InstallShield では、64 ビット COM 抽出をサポートします。詳細については、「64 ビット オペレーティング システムをターゲットにする」を参照してください。*

フォントのインストール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

さまざまな方法でコンポーネントを作成できますが、インストールにフォント (.ttf、.fon、または.ttc ファイル) をインストールする最も簡単な方法は、コンポーネントウィザードでフォントコンポーネントを作成することです。

コンポーネントウィザードでは、各パネルで行う設定によって、次に表示されるパネルが異なります。たとえば、[インストールされているフォントの追加] パネルで **"このシステムにインストールされていないフォントを追加する"** を選択しなかった場合は、[新規フォントの追加] パネルは表示されません。

他のプロジェクトへコンポーネントをエクスポートする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*



タスク ***コンポーネントを別のプロジェクトへ保存するには、以下の手順を実行します。***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. コンポーネントを右クリックして、**コンポーネントのエクスポートウィザード**を選択します。



ヒント: *コンポーネントのエクスポート ウィザードは、InstallShield のプロジェクト メニューからアクセスできます。*

コンポーネントを他のプロジェクトにエクスポートする場合、このコンポーネントのコピーが、ファイル、ショートカット、レジストリ エントリ、詳細設定など、すべてのコンポーネントのデータと共に、指定した ism ファイルに追加されます。ダイアログで使用されるすべての文字列テーブル、プロパティ、およびパス変数も新規プロジェクトにコピーされます。

対象の .ism ファイルが既に異なるプロパティで同じ名前のコンポーネントを持つ場合は、[競合の解決] ダイアログ ボックスが表示され、競合を解決するオプションを選択できます。

プロジェクトのコンポーネントを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[コンポーネント] ビューおよび[セットアップのデザイン] ビューの両方で、コンポーネントを永久に削除することできます。



タスク ***コンポーネントを削除するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. コンポーネントを右クリックして、[**プロジェクトから削除**] をクリックします。

コンポーネントはこれでプロジェクトから永久に削除されます。

コンポーネントと機能の関連付け



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*

コンポーネントを必要に応じて任意の数の機能またはサブ機能に関連付けることができます。たとえば、エディターとスペルチェックの 2 つの機能を持つテキストエディターを例とします。エディターとスペルチェックは両者ともシステム .dll コンポーネントに依存しています。このインストールを設計する場合には、システム .dll コンポーネントをエディター機能とスペルチェック機能の両方に関連付ける必要があります。

[セットアップのデザイン] ビューで、コンポーネントを機能に関連付けることができます。[セットアップのデザイン] ビューでは、コンポーネントが既に存在するかしないによって、コンポーネントを機能に関連付ける方法が 2 つあります。詳細は、次を参照してください:

- 新しいコンポーネントを機能に関連付ける.
- 既存コンポーネントを機能に関連付ける.

コンポーネントを関連付けられた機能から削除する手順は、「コンポーネントと機能の関連付けを解除する」を参照してください:



メモ: *機能に関連付けられていないコンポーネントは、次の親のないコンポーネントのアイコンで表示されます:*



新しいコンポーネントを機能に関連付ける



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*



タスク ***新しいコンポーネントを機能に関連付けるには、以下の手順に従ってください。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. 機能またはサブ機能を右クリックして、**[新しいコンポーネント]** または **[コンポーネント ウィザード]** をクリックします。InstallShield により新しいコンポーネントが作成され、それが選択された機能に関連付けられます。

既存コンポーネントを機能に関連付ける



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*



タスク ***既存のコンポーネントを機能に関連付けるには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. 機能、またはサブ機能を右クリックして **[コンポーネントを挿入]** をクリックします。
3. この機能に関連付けたいコンポーネントをリストから選択して **[OK]** をクリックします。

ある機能から別の機能へドラッグアンドドロップを使って既存のコンポーネントを移動またはコピーすることができます。

- 既存のコンポーネントを移動するには、コンポーネントをある機能から別の機能へドラッグします。
- 既存のコンポーネントをコピーするには、CTRL を押しながら、ある機能から別の機能へコンポーネントをドラッグします。

コンポーネントと機能の関連付けを解除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*



タスク ***コンポーネント－機能の関連付けを削除するには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. 機能との関連を解除するコンポーネントを右クリックしてから、**[機能から削除]** を選択します。

コンポーネントとこの機能の関連が削除されても、コンポーネントはプロジェクト中に存在しています。

データをコンポーネントに追加する

[コンポーネント] ビューを使って、ファイル、レジストリ データ、およびショートカットをコンポーネントに追加できます。

このセクションには、次のトピックが含まれています:

- ファイルをコンポーネントに追加する
- コンポーネントのファイルの [リンク] 列で値を変更する
- コンポーネントからファイルを削除する
- レジストリ データをコンポーネントに追加する
- [コンポーネント] ビューでショートカットを作成する
- 静的にリンクされたコンポーネントにサブフォルダーを追加する

ファイルをコンポーネントに追加する

コンポーネントの中にあるすべてのファイルは、同じコンポーネント設定を共有する必要があります。

コンポーネントへファイルを追加する際の条件

以下の条件を満たしている場合のみ、コンポーネントにファイルを追加することができます。

- すべてのファイルのインストール先フォルダーが同じであること。
- すべてのファイルを同じ条件 (オペレーティング システム、言語を含む) でインストールすること。

この条件を満たしていない場合は、ファイルのインストールのニーズやコンポーネントプロパティに合わせて新しいコンポーネントを作成します。また、ファイルをコンポーネントに追加する際は、セットアップ ベスト プラクティスにも注意する必要があります。セットアップベストプラクティスを使うと、新しくインストールを作成したり、再使用可能コンポーネントを効果的に配布しやすくなります。

コンポーネントへのファイルの追加方法

コンポーネントへファイルを追加する方法は、コンポーネントの作成の仕方によって 多少の違いがあります。ファイルをコンポーネントに関連付ける方法を、コンポーネントの作成方法別に、以下に説明します。

InstallShield で新しいコンポーネントを作成する

[セットアップのデザイン] ビュー(インストール プロジェクトの場合)または [コンポーネント] ビューで右クリックしてコンポーネントを作成する場合は、まずコンポーネントの [ファイル] の項目を選択してファイルリストを表示してください。



タスク ***次に、以下のいずれかの方法で、ファイルをコンポーネントに関連付けます。***

- Windows エクスプローラーからファイル リストにファイルをドラッグアンドドロップする。
- ファイルリスト内を右クリックして **[追加]** をクリックする。表示されるダイアログ ボックスで参照し、そのフォルダーから追加するファイルを必要なだけ選択してください。**[開く]** をクリックします。

コンポーネントウィザードを使用したコンポーネントの新規作成

コンポーネント ウィザードを使用してコンポーネントを作成する際、[ファイル] パネルでコンポーネントにファイルを追加することができます。



タスク ***次に、以下のいずれかの方法で、ファイルをコンポーネントに関連付けます。***

- Windows エクスプローラーからファイル リストにファイルをドラッグアンドドロップする。
- ファイルリスト内を右クリックして **[追加]** をクリックする。表示されるダイアログ ボックスで参照し、そのフォルダーから追加するファイルを必要なだけ選択してください。**[開く]** をクリックします。

InstallShield はアプリケーションのファイルにリンクを作成します。リンクは、インストールのリリースをビルドする際、ファイルを探すのに使用されます。リンクが無効になると、"ファイルが見つかりません" がファイルの横に表示されます。

ファイルを追加する上記の方法はすべて、ファイルを スタティック (静的)にリンクします。これは、新しいファイルを追加したり、ソースフォルダーから除去しても、ファイルのリンクは変わらないということを意味します。スタティック リンクに代わる方法は、「ダイナミック ファイル リンクをコンポーネントに追加する」を参照してください。

コンポーネントのファイルの [リンク] 列で値を変更する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[コンポーネント] ビューで、コンポーネントを拡張して[ファイル] ノードを表示することができます。[ファイル] ノードを表示して、コンポーネントに関連付けられているファイルを見ることができます。[ファイル] エクスプローラーの、[リンク] 列はディレクトリまたはファイルを追加したときに使用したパス変数を提供します。[ダイレクトエディター] を利用してこの列に表示される内容を変更することができます。

タスク ***ディレクトリを編集するには、以下の手順を実行します。***

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーで、**[ファイル]** をクリックします。
3. ファイルを参照して **ISBuildSourcePath** 列の値を編集します。

コンポーネントからファイルを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

タスク ***コンポーネントからファイルを削除するには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[コンポーネント]** をクリックします。
2. **[コンポーネント]** エクスプローラーで、削除するファイルを含むコンポーネントを展開してから、その下の **ファイル**アイテムをクリックします。コンポーネントに関連付けられているファイルは、右のペインに表示されます。
3. 削除するファイルを右クリックして、[**削除**] をクリックします。

レジストリ データをコンポーネントに追加する

[レジストリ] ビューを使用して、レジストリキーとレジストリ値をコンポーネントに追加することができます。この情報は、コンポーネントがインストールされると、ターゲット システムのレジストリに書き込まれます。レジストリ キーと値の追加方法については、次を参照してください。

- レジストリ キーの作成
- レジストリ エントリをドラッグアンドドロップしてレジストリ キーを作成する
- レジストリ値の作成
- レジストリ ファイルをインポートする
- レジストリ ファイルのエクスポート

[コンポーネント] ビューでショートカットを作成する

プロジェクト: *インストール プロジェクトでは、[ショートカット] ビューでショートカットを作成できます。*

ショートカットを作成する前に、まずショートカットをリンクさせるファイルが入ったコンポーネントを作成する必要があります。

タスク ***新しいショートカットを作成するには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[コンポーネント]** をクリックします。
2. **[コンポーネント]** エクスプローラーで、作成するショートカットに関連付ける必要のあるコンポーネントを展開したあと、**[ショートカット]** をクリックします。**[ショートカット]** エクスプローラーが新しいペインに開きます。
3. **[ショートカット]** エクスプローラーで、インストール先フォルダーを右クリックしてから、**[新しいフォルダー]** または **[新しいショートカット]** をクリックします。たとえば、会社名の下にショートカットを表示したい場合は、プログラム フォルダーを作成することができます。

   ショートカット用のフォルダーを作成した後、そのフォルダーを右クリックし **[新しいショートカット]** をクリックして、ショートカットを作成します。
4. ショートカットのプロパティ を定義します。

静的にリンクされたコンポーネントにサブフォルダーを追加する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

タスク ***静的にリンクされたコンポーネントにサブフォルダーを追加するには以下の手順に従います:***

[コンポーネント] または **[セットアップのデザイン]** ビューでコンポーネントの **[スタティック ファイル リンク]** サブノードを右クリックして、**[新しいフォルダー]** を選択します。

コンポーネントのキーファイル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

- *トランスフォーム*

コンポーネントのファイルの 1 つはキーファイルとなります。キーファイルは、ターゲットマシン上にコンポーネントが存在するかどうかを検出し、またそのアップデートが必要かどうかを決定するために Windows Installer が使用するファイルです。コンポーネントの詳細設定やショートカットを作成するには、キーファイルを指定する必要があります。

このセクションには、次のトピックが含まれています:

- コンポーネントのキーファイルを設定する
- コンポーネントからキー ファイルをクリアする



注意 - *プロジェクトにダイナミック リンクがあるファイルが含まれている場合、それらのファイルに一部がコンポーネントのキー ファイルとして自動的に設定される場合があります。詳細については、「ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する」を参照してください。*

コンポーネントのキーファイルを設定する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*



タスク ***コンポーネントのファイルの 1 つをキーファイルに指定するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを展開し、**ファイル**をクリックします。
3. ファイル リストを右クリックして、**[キー ファイルの設定]** をクリックします。そのファイルのファイルアイコン () が、キーアイコン () に置換されます。

コンポーネントは、1 つのキー パス、または、1 つのキーフ ァイルを持つことができます。キーパスまたはキーファイルがコンポーネントに既に割り当てられている場合、他のキーファイルを設定しようとすると、警告メッセージ ボックスが表示されます。既存のキーファイルまたはキーパスを新規キーファイルで置き換えるには、このメッセージ ボックスで [はい] をクリックします。



メモ- *[ファイルとフォルダー] ビューのキーファイルは指定できません。キーファイルは、[セットアップのデザイン] ビューまたは [コンポーネント] ビューのいずれかで設定できます。*

コンポーネントからキー ファイルをクリアする

プロジェクト- この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

このファイルをコンポーネントのキーファイルとして使用しない場合、コンポーネントからキー ファイルをクリアすることができます。

タスク ***コンポーネントからファイルをクリアするには、以下の手順を実行します。***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを展開し、**ファイル**をクリックします。
3. キー ファイルを右クリックして [**キー パスのクリア**] をクリックします。

コンポーネントの設定

プロジェクト- この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントを作成すると、コンポーネントをビューで作成したかウィザードで作成したかによって、デフォルトの値が設定されます。



タスク ***コンポーネントの設定を編集するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. 構成を行うコンポーネントをクリックしてから、その設定を必要に応じて変更します。

各設定についての詳細は、「コンポーネントの設定」を参照してください。

このセクションには、次のトピックが含まれています:

- コンポーネントのインストール先と機能のインストール先の違い
- アンインストール時にコンポーネントのファイルと他の関連データをアンインストールするかどうかを指定する
- コンポーネント条件を構成する
- 再インストール中にコンポーネント条件を再評価する
- 共有ファイルの参照カウントを管理する
- インストール前にファイルのバージョンを確認する
- 同じ名前のファイルをインストールする
- コンポーネントの "リモート インストール" 設定を構成する
- ビルド時に COM 登録データを抽出する
- .NET 依存関係とプロパティを確認するためのスキャン
- .NET アプリケーション ファイルを指定する
- .NET Installer クラスへ渡されたプロパティを読み込む
- レジストリ リフレクションの有効化と無効化
- コンポーネントの "共有コンポーネントのパッチ" を有効にするかどうかを指定する

コンポーネントのインストール先と機能のインストール先の違い



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

DIM、マージ モジュール、および MSM データベース プロジェクトは機能を含みません。これらのプロジェクトの種類は、基本の MSI プロジェクトの機能に追加されます。したがって、DIM、マージ モジュール、および MSM

データベース プロジェクトのコンポーネントは、それらを使用するインストール プロジェクトに含まれる機能に関連付けられます。

コンポーネントおよび機能 のどちらにも "インストール先" 設定がありますが、エンド ユーザーがインストールを実行した時に利用される方法が異なります。

エンドユーザーによるインストール先の変更を許可する

機能のインストール先設定を使って、ユーザーによるターゲット システムでの機能のインストール先変更を可能にできます。機能のインストール先はエンド ユーザーによって変更されるため、機能のインストール先にはパブリック プロパティを利用しなくてはなりません。パブリック プロパティの名前には、大文字しか使用できません (例、INSTALLDIR)。



メモ: *機能のすべてのコンポーネントを機能のインストール先にインストールするには、コンポーネントのすべてのインストール先が機能のインストール先と一致するように設定する必要があります。*

特定のインストール先を設定する

コンポーネントをエンドユーザーによる変更が不可能な特定の場所にインストールする場合、コンポーネントのインストール先を機能のインストール先として利用されないディレクトリに設定しなくてはなりません。

コンポーネントのファイルのインストール先フォルダーを設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

各コンポーネントのファイルに対し、異なるインストール先を指定できます。"インストール先" 設定のデフォルト値は、次にようになります:

- 基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合は INSTALLDIR で、製品の INSTALLDIR プロパティでの初期値は [ProgramFilesFolder]Company Name¥Product Name です。
- InstallScript および InstallScript オブジェクトプロジェクトの場合は TARGETDIR で、**OnFirstUIBefore** イベント ハンドラー関数のデフォルトのスクリプト コードでのデフォルトの初期値は PROGRAMFILES ^ IFX_COMPANY_NAME ^ IFX_PRODUCT_NAME です。



***Windows ロゴ・** Windows ロゴ要件によると、アプリケーションのファイルのデフォルトのインストール先は、エンド ユーザーの Program Files フォルダーのサブフォルダーである必要があります。*INSTALLDIR または ProgramFilesFolder *を機能の "インストール先" 設定の親フォルダーとして使用すると、ファイルは正しい場所にインストールされます。*



タスク ***コンポーネントのインストール先フォルダーを変更するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. インストール先を変更するコンポーネントを選択します。
3. **"インストール先"** 設定で、一覧からオプションの 1 つを選択するか、省略記号ボタン (...) をクリックして、ディレクトリを選択または作成します。

インストール先フォルダーに関するその他の考慮事項



***プロジェクト・** この情報は、次のプロジェクトの種類に適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントの "リモート インストール" 設定

コンポーネントの "リモート インストール" 設定を [ソースを優先] (または、コンポーネントの機能が [ソースを優先] に設定されている場合は [オプション]) に設定すると、コンポーネントの "インストール先" 設定に関わらず、コンポーネントのファイルはターゲット システムにインストールされません。

機能の "リモート インストール" 設定

各機能にも "インストール先" 設定があります。異なる場合、コンポーネントの "インストール先" 設定は、機能のインストール先をオーバーライドします。機能の "インストール先" 設定はオプションですが、コンポーネントの "インストール先" 設定は必須です。

デフォルトのインストール先としての INSTALLDIR の使用

すべての機能およびコンポーネントに対し [インストール先] 設定のデフォルトが INSTALLDIR に設定されているのは、すべてのアプリケーションのファイルを同じルートフォルダーにインストールするためです。この結果、エンドユーザーがある機能のインストール先フォルダーを Custom Setup ダイアログで変更すると、INSTALLDIR に含まれるパスにインストールするよう設定されたすべての機能のインストール先フォルダーも変更されます。

メモ- コンポーネントのインストール先が INSTALLDIR 以外に設定されていると、コンポーネントは各コンポーネントに指定されたインストール先にインストールされるため、INSTALLDIR を変更してもコンポーネントのインストール先には影響がありません。

INSTALLDIR などのインストールフォルダープロパティは、デフォルト値として指定されます。エンドユーザーは、Msiexec.exe 起動時に、コマンドラインでプロパティを設定したり、CustomSetup ダイアログで機能に対して新しいインストール先フォルダーを選択することにより、この値を変更できます。

.NET アセンブリを含むコンポーネント用の [GlobalAssemblyCache]

コンポーネントに .NET アセンブリが含まれる場合、コンポーネントのインストール先を [GlobalAssemblyCache] に設定できます。.NET アセンブリがターゲット システムのグローバルアセンブリキャッシュ (GAC) にインストールされている場合、アセンブリにシステムの他のアプリケーションからアクセスできます。

一般に、アセンブリはローカルアプリケーション ディレクトリにインストールすることをお勧めします。こうするとアプリケーション分離が向上します。

メモ- アセンブリをグローバル アセンブリ キャッシュにインストールするには、アセンブリ コンポーネントの "ビルド時に .NET をスキャン" 設定を [プロパティのみ] または [依存関係とプロパティ] に設定しておく必要があります。

スクリプトからコンポーネントのインストール先を指定する

***プロジェクト-** この情報は、InstallScript プロジェクトに適用します。*

コンポーネントのファイルのターゲットのインストール先をスクリプトから指定すると、エンド ユーザーの入力またはその他の条件に従ってランタイム時にそのインストール先を変更できます。

タスク** **コンポーネントのファイルのターゲット インストール先をスクリプトから指定するには、以下の手順に従います:

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、**[スクリプト定義のフォルダー]** を右クリックし、**[新規フォルダー]** をクリックします。サブフォルダーが、**<NEW VARIABLE N>** というデフォルト名で追加されます (ここで *N* は連続番号です)。このサブフォルダーの名前を **<スクリプト定義フォルダー>** などのように山かっこで囲って任意の文字列に変更できます。
3. 既存のコンポーネントのファイルのターゲットインストール先を指定するには、以下の手順を実行します。
   a. **[編成]** のビュー リストにある **[コンポーネント]** または **[セットアップのデザイン]** をクリックします。
   b. 構成するコンポーネントを選択します。
   c. コンポーネントのプロパティグリッドで "**インストール先**" プロパティの値をクリックし、リストから手順 1 で作成したサブフォルダーの名前を選択します。

   新しいコンポーネントを作成してそのファイルのターゲットのインストール先を指定するには、以下の手順を実行します。

a. **[ファイルとフォルダー]** ビューで、希望のファイルを **[ソース コンピューターのフォルダー]** ペインからステップ 1 で作成したサブフォルダーにドラッグアンドドロップします。デフォルト名に **FilesN** という名前を持つ新しいコンポーネントが作成されます。

b. スクリプトで **FeatureSetTarget** を呼び出して手順 1 で作成したサブフォルダーにターゲット先を割り当てます。下に例を示します。

```
AskDestPath( "" , "XYZ ファイルの場所を選択してください。", svEndUserSelectedPath, 0 );
FeatureSetTarget( MEDIA , "<スクリプト定義フォルダー>" , svEndUserSelectedPath);
```

大文字のディレクトリ識別子とコンポーネントのインストール先



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ディレクトリ識別子を大文字で指定すると、その値がユーザー インターフェイスから設定できる パブリックプロパティになります。エンドユーザーまたは管理者がユーザー インターフェイスまたはコマンドラインからインストール先を変更するには、コンポーネントのインストール先のディレクトリ識別子をパブリック プロパティにする必要があります。

大文字と小文字を混ぜたり、小文字でディレクトリ識別子を指定すると、ディレクトリエントリが プライベート プロパティ として定義されます。プライベートプロパティは、ユーザー インターフェイスから変更できません。

アンインストール時にコンポーネントのファイルと他の関連データをアンインストールするかどうかを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

エンド ユーザーが製品をアンインストールするとき、場合によって、コンポーネントのファイル、レジストリ エントリ、ショートカットおよび他のデータがアンインストールされないようにする必要があります。以下は、コンポーネントのデータがアンインストールされないようにする必要がある場合のシナリオ例です。

- 他の製品に使用される可能性があるファイルのアンインストールを防ぎたい。
- コンポーネントにフォントが含まれている。
- コンポーネントが [SystemFolder] にインストールされる。検証規則によると、この場所にインストールされるコンポーネントは、アンインストール時にアンインストールされるべきではないとあります。

コンポーネントのデータをアンインストールするかどうかを制御する設定は、使用しているプロジェクトの種類によって異なります。



タスク ***コンポーネントのファイル、レジストリ エントリ および他のデータをアンインストールするかどうかを指定するには、以下の手順に従います:***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ) または **[コンポーネント]** ビューで、構成するコンポーネントを選択します。InstallShield の右側に設定が表示されます。
2. 基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合ー **"パーマネント"** 設定を必要に応じて次のように構成します:
   - エンドユーザーが製品をアンインストールするとき、コンポーネント データがアンインストールされないようにするには、**[はい]** を設定します。
   - コンポーネントデータがアンインストールされるようにする場合、**[いいえ]** を選択します。

   InstallScript および InstallScript オブジェクト プロジェクトの場合 ー **"アンインストール"** 設定を必要に応じて次のように構成します:
   - エンドユーザーが製品をアンインストールするとき、コンポーネント データがアンインストールされないようにするには、**[いいえ]** を設定します。
   - コンポーネントデータがアンインストールされるようにする場合、**[はい]** を選択します。

"パーマネント" 設定と "アンインストール" 設定は、コンポーネントに関連付けられているすべてのデータ型に適用します。これには、ファイル、レジストリ エントリ、ショートカット、XML ファイルの変更および SQL スクリプトが含まれます。

コンポーネント条件を構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

試用版と完全版など、同じプログラムの異なるバージョンを作成する場合、コンポーネントの条件付きインストールが役立ちます。トライアル版では、一部の機能を提供しないようにするため、一部のコンポーネントをインストールしません。また、コンポーネントの条件付きインストールによって、ディスク容量を節約することもできます。ターゲット システムにすべてのコンポーネントをインストールするのに十分なディスク容量がない場合、必ずしも必要でないコンポーネントを条件付きでインストールするよう設定できます。

"コンポーネントの条件" 設定を使用すると、ターゲット システムにコンポーネントのデータをセットアップする前にインストールが評価するステートメントを入力できます。このコンポーネントは、条件が False に評価された場合にはインストールされません。条件が True と評価された場合には、インストールする機能が選択されているとして、コンポーネントがインストールまたはアドバタイズされます。



タスク ***プロジェクトのコンポーネントに条件を設定するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. 構成するコンポーネントを選択します。
3. **"条件"** 設定をクリックして、省略記号ボタン (...) をクリックします。**[条件ビルダー]** ダイアログ ボックスが開きます。
4. 以下のいずれかを実行します。
   - **[条件]** ボックスで、コンポーネントの条件を入力します。
   - **[プロパティ]** リスト、**[演算子]** リスト、および **[追加]** ボタンを使って、条件をビルドします:
     a. **[プロパティ]** リストで、プロパティを選択してから、**[追加]** ボタンをクリックします。InstallShield は、プロパティを **[条件]** 列に追加します。
     b. 条件ステートメントに演算子を含める場合、**[演算子]** リストから演算子を選択してから、**[追加]** ボタンをクリックします。InstallShield によって、条件ステートメントに演算子が追加されます。
     c. 条件ステートメントに値を含める場合は、その値を入力します。
5. **[OK]** をクリックします。



重要 *InstallShield は、基本の条件検証を行いますが、条件ステートメントが予定通りの結果を評価することを確認してください。詳しい情報と条件のサンプルについては、「条件ステートメントのビルド」を参照してください。*

再インストール中にコンポーネント条件を再評価する



プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

- *MSM データベース*
- *トランスフォーム*

コンポーネントの "条件の再評価" 設定で [はい] を選択すると、Windows Installer は、コンポーネントの条件を再評価します。*再インストールについての条件が True と評価された場合、コンポーネントは再インストールされます。また、条件が最初に False と評価されていた場合、あるいはコンポーネントがインストール用に選択されていなかった場合は、インストールされます。

移行コンポーネント

インストール済みのコンポーネントで、再インストール時に条件が False と評価されたコンポーネントは削除されます。この特別機能によって、再インストール時にコンポーネントを交換することができます。"再評価条件" 設定で [はい] が選択されたコンポーネントは*移行コンポーネント*と考えられます。

Windows XP または Windows Vista のどちらにインストールされるかによって、異なる .dll ファイルを必要とするアプリケーションを例とします。各ファイルにコンポーネントを作成して、各コンポーネントにオペレーティング システムのバージョンをチェックする条件を追加します。Windows Vista 用のコンポーネントの条件の例を以下に示します:

VersionNT=600

WindowsXP 専用コンポーネントの条件の例を以下に示します:

VersionNT<600

製品を Windows XP にインストールする場合、適切なバージョンの .dll インストールされ、Windows Vista バージョンはインストールされません。エンドユーザーが Windows Vista にアップグレードした場合はどうなるのでしょうか。両方のコンポーネントが移行可能であれば、製品の再インストール時に Windows XP 専用コンポーネントはアンインストールされ、Windows Vista 専用のコンポーネントがインストールされます。

Using Transitive Components (Windows Installer ヘルプ ライブラリ)

共有ファイルの参照カウントを管理する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ファイルの参照カウント (*refcount* とも呼ばれます) は、特定のファイルを使用するターゲット システム上にある製品の数です。参照カウントを使って、あるファイルが複数の製品で共有されている場合に、それを共有する製品すべてが削除されるまで、ターゲット システムに保持することができます。

共有ファイルの参照カウントは、次のレジストリ キーに格納されます:

HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥SharedDLLs

Windows Installer ベースのプロジェクトと InstallScript ベースのプロジェクトの両方で、共有ファイルの参照カウントを管理することができます。プロジェクトの種類によって、その機能は多少異なります。

Windows Installer ベースのインストールでの動作

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトでは、キーファイルのコンポーネントの "共有" 設定で [はい] を選択して、キー ファイルが共有されていることを指定できます。インストールがキー ファイルをインストールする場合で、コンポーネントの "共有" 設定で [はい] が選択されているとき、レジストリに参照カウントが既存しない場合はそれが作成され、既存する場合はそのカウントが増加されます。キーファイルのアンインストール中、参照カウントが減少されます。

Windows ロゴ・ *Windows ロゴガイドラインでは、共有ファイルをインストールするときに参照カウントを増加し、アンインストールするときに減少することが必要です(インストールしないように推奨されている) コア コンポーネントファイルは参照カウントされません。*

InstallScript ベースのインストールでの動作

InstallScript および InstallScript オブジェクト プロジェクトでは、コンポーネントを共有としてマークできます。インストールがコンポーネントのファイルをインストールする場合で、コンポーネントの "共有" 設定で [はい] が選択されているとき、レジストリに各ファイルの参照カウントが既存しない場合はそれが作成され、既存する場合はそのカウントが増加されます。コンポーネントのアンインストール中、参照カウントが減少されます。

InstallScript 関数 **GetFileInfo** を FILE_SHARED_COUNT 定数を使って呼び出して、既存ファイルの参照カウントを判断できます。

コンポーネントを共有としてマークする方法

タスク

コンポーネントのキー ファイルを共有として指定する (Windows Installer ベースのプロジェクトの場合) 、または コンポーネントを共有として指定する (InstallScript ベースのプロジェクトの場合) には、以下の手順に従います:

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. 構成するコンポーネントを選択します。
3. **"共有"** 設定で **[はい]** を選択します。

ヒント・ *コンポーネントを常に共有としてマークしなくてはならない一例として、そのコンポーネントのファイルが System フォルダーまたは Common Files フォルダーといった共有ディレクトリにインストールされる場合が挙げられます。*

メモ・ *コンポーネントを共有としてマークしているいないにかかわらず、インストーラーによってコンポーネントのすべてのファイルの参照カウントが増分されますので、ご注意ください。ただし、参照カウントが存在しない場合、この "共有" 設定が [はい] に設定されていない限り、インストールはこれを作成しません。*

インストール前にファイルのバージョンを確認する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallScript プロジェクトの場合、この機能はコンポーネントの "上書き" 設定によって処理されます。この設定で省略記号ボタン (...) をクリックすると開く [上書き] ダイアログ ボックス で、コンポーネントがターゲット システム上の既存ファイルを上書きするかどうかを指定できます。

コンポーネントの **"上書きしない"** 設定では、ファイルが既にターゲット システムに存在する場合、インストールでファイルを上書きするかどうか指定できます。

- [はい] を選択すると、(ターゲット システムに存在する場合) ファイルは、ファイルのバージョンに関係なく、決して上書きされません。この設定で [はい] を選択すると、ファイルのバージョン規則がオーバーライドされます。
- [いいえ] を選択しても、ターゲット システムのファイルバージョンがインストールされるバージョンより新しい場合、ターゲット システムのファイルは上書きされません。インストールされるバージョンの方が新しい場合、ターゲット システム上のファイルは 上書きされます 。

Windows Installer は、コンポーネントをインストールするかどうか決定する際、コンポーネントの キーファイルの存在を確認します。詳細については、「ターゲット システム上でファイルとコンポーネントを上書きする」を参照してください。

同じ名前のファイルをインストールする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

コンポーネントの "ソースの場所" 設定は、コンポーネントのファイルが圧縮されない場合に、コンポーネントファイルがソース ディスク イメージに格納されるサブフォルダーを識別します。コンポーネントのファイルはリリースイメージでこのフォルダーにコピーされます。

"ソースの場所" 設定の値は必須ではなく、たいていの場合は、空白のままで構いません。ただし値を指定する場合は、有効な Windows フォルダー名を入力する必要があります。

"ソースの場所"設定が使用される例の 1 つは、複数の言語を持つインストールを作成する場合です。こうしたケースでは、複数のファイルに共通の名前をつけておきたい状況も考えられます。各言語用のコンポーネントを作成して、それぞれのファイルに対して必要に応じて "ソースの場所" 設定を構成できます。この "ソースの場所" 設定を使用することで、同じ名前を持つファイルを上書きしてしまう危険を冒すことなく、ディスク上の異なる場所にコピーすることができます。

たとえば、*German* および *English* という 2 つのコンポーネントを作成したとします。最初のコンポーネントの "ソースの場所" 設定には、**GermanVersion** を入力します。2 番目のコンポーネントの "ソースの場所" 設定には、**GermanVersion** を入力します。そして *Test.txt* という名前で、内容が少しだけ異なるファイルを、2 つ用意します。各ファイルをコンポーネントに割り当てます。

非圧縮ファイルを使用してインストールをビルドすると、ディスクイメージ上に *GermanVersion* と *EnglishVersion* という名前の異なるフォルダーが作成されます。上記フォルダーに別々のバージョンの Test.txt が、どちらも上書きされることなしにコピーされます。

メモ: *"ソースの場所" 設定と「インストール先」は異なります。2 つのバージョンのファイルを両方ともエンドユーザーのマシンにコピーするような場合もありますが、通常はファイルに言語でフィルターをかける場合の方が多くなります。*

コンポーネントの "リモート インストール" 設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

"リモート インストール" 設定によって、このコンポーネントのファイルをターゲット システムにインストールするか、または CD-ROM やネットワークサーバーのようなソース メディアから実行するかが決定されます。新規コンポーネントのデフォルト値は [ローカルを優先] です。これは、コンポーネントのファイルがターゲット システムにインストールされることを意味します。

タスク ***この値を変更してコンポーネントのファイルがソース メディアからのみ実行されるようにするには、以下のようにします。***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. 構成するコンポーネントを選択します。
3. "リモート インストール" 設定で、**[ソースを優先]**を選択します。**[オプション]** を選択すると、このコンポーネントにその機能の **"リモート インストール"** 設定を与えることになります。

コンポーネントの "リモート インストール" 設定は、機能のそれをオーバーライドします。詳細については、「コンポーネントの "リモート インストール" 設定と機能の "リモートインストール" 設定の違い」を参照してください。



注意 *・コンポーネントに Windows サービスが含まれる場合、[ローカルを優先する] オプションを選択します。エンドユーザーは CustomSetup ダイアログを使用して機能のインストール状態を変更することができますが、Windows Installer はサービスをリモート インストールすることはできません。*

コンポーネントの "リモート インストール" 設定と機能の "リモートインストール" 設定の違い



プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *トランスフォーム*

コンポーネントの "リモート インストール" 設定は、常に機能の "リモート インストール" 設定をオーバーライドします。たとえば、コンポーネントの "リモート インストール" 設定が [ローカルを優先する] に設定されていると、機能の "リモート インストール" 設定に関係なくターゲット システムにそのファイルがインストールされます。

また、コンポーネントの "リモート インストール" 設定が [ソースを優先] に設定されている場合には、ファイルは常にソース メディアから実行されます。コンポーネントの機能が、ファイルをソース メディアから実行するかどうかを決定する場合、コンポーネントの "リモート インストール" 設定で [オプション] を選択します。

コンポーネントが複数の機能に関連付けられていて、コンポーネントの "リモート インストール" 設定が [オプション] に設定されているとき、その機能で [ローカルを優先] が設定されてる場合、ファイルはローカルにインストールされます。コンポーネントが [ローカルを優先] または [ソースを優先] に設定されていれば、その設定に従ってファイルがインストールされます。

ビルド時に COM 登録データを抽出する



プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

コンポーネントの "ビルド時に COM 抽出" 設定で [はい] を選択すると、InstallShield はそのコンポーネントを含むリリースをビルドするたびに、コンポーネントのキー ファイルをスキャンして COM 登録データを抽出します。抽出された情報はリリースに配置され、インストールまたはアドバタイズされるときに Windows Installer が COM サーバーを登録できるようにします。コンポーネントによって作成されたすべての必要なレジストリ設定は、この設定で [はい] を選択すると抽出されます。

コンポーネント ウィザードと異なり、ビルドプロセスではプロジェクト (.ism) ファイルに抽出された COM 情報が書き込まれません。代わりに、ビルドし直すたびにアップデートされるというように、動的になります。これはまた、プロジェクトファイルへの書き込みアクセス許可を持たなくても、InstallShield やコマンドラインから既存のリリースを再ビルドできることを意味します。



メモ: *ビルドされたマシンの PATH システム変数は、COM サーバーのリンク先となるすべての .dll ファイルのディレクトリを含むように設定する必要があります。この設定を怠った場合、ファイルの登録は失敗し、COM 情報の抽出が行なわれません。*

ビルドのフィードバック ([出力] ウィンドウおよびビルドログファイルで表示される) では、抽出された登録情報が詳細に表示されます。



ヒント: *InstallShield を 64 ビット システムで使用している場合、InstallShield は 64 ビット COM サーバーから COM データを抽出できます。データを正しい場所にインストールするため、コンポーネントを 64 ビットとマークしなくてはなりません。64 ビット サポートに関する詳細は、「64 ビット オペレーティング システムをターゲットにする」を参照してください。*

競合の解決

"ビルド時に COM 抽出" 設定に [はい] が選択されていても、コンポーネントの [COM 登録] 詳細設定とコンポーネントの [レジストリ] エクスプローラーで、COM 情報を指定することができます。既存の情報は、コンポーネントがインストールされるときに常に登録されます。

エントリが COM 登録の下で検出されると、InstallShield は "ビルド時に COM 抽出" 設定に [はい] が設定されたときにそれらを削除するかどうかを問い合わせます。ビルド時に競合が見つかった場合、動的に取得したデータで上書きされた詳細設定の項目に関して警告メッセージが表示されます。

これらの安全策の他にも、COM 登録の詳細設定とコンポーネントのレジストリ データをチェックすることによって、予定通りのエントリが使用されており、ビルド時に抽出されたエントリと競合していないことを確認することができます。

.NET 依存関係とプロパティを確認するためのスキャン



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

InstallShield では、ビルド時にコンポーネントの .NET 依存関係とプロパティをスキャンするかどうかを指定できます。使用するプロジェクトの種類によって、その機能は異なります。

Windows Installer ベースのプロジェクト

基本の MSI、DIM、InstallScript MSI、およびマージ モジュール プロジェクトでは、"ビルド時に .NET をスキャン" 設定を使って、ビルド時にコンポーネントのキー ファイルについて .NET 依存関係とプロパティをスキャンするかどうかを指定できます。



メモ: *キーファイルが .NET アセンブリ ファイルではない場合、ビルド時のスキャンではコンポーネントのキー ファイルはチェックされません。*

コンポーネントの "ビルド時に .NET をスキャン" 設定で利用可能なオプションは、次のとおりです:

テーブル 6 ・ Windows Installer ベースのプロジェクトにおける "ビルド時に .NET をスキャン" 設定のオプション

オプション	説明
なし	InstallShield、このコンポーネントのキーファイルをスキャンして .NET 依存関係またはプロパティのチェックを行いません。
プロパティのみ	ビルド時に、このコンポーネントのキー ファイルをスキャンして、.NET 属性をチェックします。必要に応じて、InstallShield は MsiAssembly と MsiAssemblyName テーブルにアセンブリ プロパティを挿入します。
依存関係およびプロパティ	ビルド時に、このコンポーネントのキー ファイルをスキャンして、.NET 依存関係とプロパティをチェックします。必要に応じて、InstallShield は MsiAssembly と MsiAssemblyName テーブルにアセンブリ プロパティを挿入します。さらに InstallShield は、.NET アセンブリが必要とする、不足しているファイル、コンポーネント、およびマージ モジュールをリリースに追加します。



メモ: *アセンブリを*グローバル アセンブリ キャッシュ*にインストールするには、**"ビルド時に .NET をスキャン"** 設定を [プロパティのみ] または [依存とプロパティ] に設定しておく必要があります。*

*この設定は、プロジェクトで*スタティック スキャニング ウィザードがどのようにファイルをスキャンするかに影響します*。*

InstallScript ベースのプロジェクト

InstallScript および InstallScript オブジェクト プロジェクトでは、"ビルド時に .NET をスキャン" 設定を使って、ビルド時にコンポーネントのキー ファイルについて .NET 依存関係をスキャンするかどうかを指定できます。

コンポーネントの "ビルド時に .NET をスキャン" 設定で利用可能なオプションは、次のとおりです:

テーブル 7 • InstallScript ベースのプロジェクトにおける "ビルド時に .NET をスキャン" 設定のオプション

オプション	説明
なし	InstallShield は、このコンポーネントのファイルをスキャンして .NET 依存関係のチェックを行いません。
依存関係	InstallShield は、このコンポーネントのファイルをスキャンして .NET 依存関係をチェックします。InstallShieldは、不足している依存関係をリリースに追加します。

MsiAssembly テーブル (Windows Installer ヘルプ ライブラリ)
MsiAssemblyName テーブル (Windows Installer ヘルプ ライブラリ)

.NET アプリケーション ファイルを指定する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

".NET アプリケーション ファイル" 設定は、このコンポーネントがビルド時にスキャン ([ビルド時に.NETをスキャン] 設定による) されるか、スタティック スキャン ウィザード でスキャンされたときに使用されます。スキャナーはコンポーネントのインストール先と一緒にこの設定を使用して、アセンブリの "ファイル アプリケーション" 設定の値を決定します。

.NET アセンブリの "ファイル アプリケーション" 設定を構成するためのスキャンのアルゴリズムは次のとおりです:

1. アルゴリズムをスキャンすると、コンポーネントの "インストール先" 設定をチェックします。この値が [GlobalAssemblyCache] の場合、.NET アセンブリの "ファイル アプリケーション" 設定はヌルに設定されます。
2. コンポーネントのインストール先が [GlobalAssemblyCache] 以外の場合、スキャンのアルゴリズムはコンポーネントの ".NET アプリケーションファイル" 設定をチェックします。この値がヌル値でない場合、この設定の値を使ってアセンブリの "ファイル アプリケーション" 設定が構成されます。
3. コンポーネントのインストール先が [GlobalAssemblyCache] 以外で、".NET アプリケーション ファイル" 設定がヌルの場合、コンポーネントのキーファイルを使用してアセンブリの "ファイル アプリケーション" 設定が構成されます。

.NET Installer クラスへ渡されたプロパティを読み込む

これは Installer クラスの実装し、インストールからInstaller クラスへ渡されるプロパティの読み出し方をデモンストレーションする .NET クラスの例です。

```
using System;
using System.Configuration.Install;
using System.Windows.Forms;
using System.Collections;

namespace MyInstall
{
    ///
    /// Class1 の概要説明。
    ///
    [System.ComponentModel.RunInstallerAttribute(true)]
    public class MyInstallClass : Installer
    {
        public MyInstallClass()
        {
        }
        public override void Install(IDictionary stateSaver)
        {

            base.Install(stateSaver);
            foreach(string strKey in Context.Parameters.Keys)
            {
                MessageBox.Show(strKey + " は " + Context.Parameters[strKey] + "です");
            }
        }

        public override void Commit(IDictionary stateSaver)
        {
            base.Commit(stateSaver);
            foreach(string strKey in Context.Parameters.Keys)
            {
                MessageBox.Show(strKey + " は " + Context.Parameters[strKey] + "です");
            }
        }

        public override void Uninstall(IDictionary stateSaver)
        {
            base.Uninstall(stateSaver);
            foreach(string strKey in Context.Parameters.Keys)
            {
                MessageBox.Show(strKey + " は " + Context.Parameters[strKey] + "です");
            }
        }

        public override void Rollback(IDictionary stateSaver)
        {
            base.Rollback(stateSaver);
            foreach(string strKey in Context.Parameters.Keys)
            {
                MessageBox.Show(strKey + " は " + Context.Parameters[strKey] + "です");
            }
        }
    };
}
```

レジストリ リフレクションの有効化と無効化



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *トランスフォーム*

レジストリ リフレクションを使用して、ターゲット マシン上で 32 ビットの [レジストリ] ビューと 64 ビットの [レジストリ] ビューの同期を保つことができます。



メモ: *Windows Installer 4 以降がある 64 ビット システムでのみレジストリ リフレクションがサポートされています。また、Windows Vista 以降と、Windows Server 2008 以降でのみサポートされています。*

エンド ユーザーがレジストリ リフレクションが有効にされたコンポーネントを持つ 64 ビットのアプリケーションをインストールすると、まず Windows Installer によりレジストリの 64 ビット ビューで (関連付けられた) レジストリの変更が行われ、その後リフレクタによりそのレジストリの変更が 32 ビットのレジストリ ビューにコピーされます。同様に、エンド ユーザーが同レジストリのキーまたは値を変更する 32 ビットのアプリケーションをインストールすると、まず Windows Installer によりレジストリの 32 ビット ビューで (関連付けられた) レジストリの変更が行われ、その後リフレクタによりそのレジストリの変更が 62 ビットの [レジストリ] ビューにコピーされます。

コンポーネントが各キーにアクセスする際、Windows Installer が **RegDisableReflectionKey** 関数を呼び出します。この関数は、指定されたキーのレジストリ リフレクションを無効にします。キーのリフレクションを無効にしても、サブキーのリフレクションには影響ありません。



タスク

コンポーネントによって影響が及ぼされるすべての新しいまたは既存のレジストリキーに対してレジストリ リフレクションを有効化するには、次の手順に従います。

1. **[編成]** の下のビュー リストにある **[コンポーネント]** をクリックします。
2. **[コンポーネント]** エクスプローラーで、レジストリ リフレクションの設定を構成するコンポーネントを選択します。
3. レジストリ リフレクションを有効にするには、**[レジストリ リフレクションを無効にする]** 設定を **[いいえ]** にします。これがデフォルト値です。

   レジストリ リフレクションを無効にするには、**[レジストリ リフレクションを無効にする]** 設定を **[はい]** にします。

レジストリ リフレクションの詳しい情報については、MSDN ライブラリの「Registry Reflection」を参照してください。

コンポーネントの "共有コンポーネントのパッチ" を有効にするかどうかを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield では、コンポーネントについて共有コンポーネントのパッチを有効にするかどうかを指定できます。デフォルトでは無効です。

ターゲット システムにインストールされている少なくとも 1 つのパッケージについて、この複数パッケージの共有機能が有効な場合、Windows Installer 4.5 はこれらすべてのパッケージ間でコンポーネントが共有されているものとして処理します。このコンポーネントを共有するパッチがアンインストールされると、Windows Installer はシステム上で、コンポーネントのファイルの最も上位バージョンを共有し続けることができます。

複数パッケージ コンポーネントの共有を行う目的は、1 つまたは複数の他のインストール パッケージによって共有されているコンポーネントを含むパッチのアンインストール中に、ファイルがダウングレードされないように防ぐことです。これによって、そのパッチがアンインストールされた後でも、コンポーネントのファイルの最も上位バージョンがマシン上で保持されます。



メモ: *ターゲット システムで DisableSharedComponent ポリシーが 1 に設定されている場合、Windows Installer はすべてのパッケージについて、この設定を無視します。*

Windows Installer 4.0 以前では、この設定は無視されます。

次の図表は、file.dll と呼ばれるファイルを含むコンポーネントを共有する ABC と XYZ という 2 つの製品の例を示します。エンド ユーザーは、最初に製品 ABC をインストールし、共有コンポーネントの file.dll バージョン 1.0.0.0 がインストールされます。次に、エンド ユーザーは file.dll バージョン 1.1.0.0 を含む製品 XYZ をインストールします。このバージョンは製品 ABC 用にインストールされたファイルよりも上位バージョンであるため、Windows Installer は現在のバージョンを 1.1.0.0 で上書きします。その後、エンド ユーザーが製品 ABC 用のアンインストール可能なパッチをインストールします。このパッチは、file.dll の 1.2.0.0 バージョンを含みます。これは、ターゲット システムに既存するファイルよりも上位バージョンであるため、Windows Installer は現在のバージョンを 1.2.0.0 で上書きします。エンド ユーザーがパッチをアンインストールすると、次の結果のどちらかが発生します:

- 製品 ABC または 製品 XYZ のどちらかについて、"複数パッケージの共有コンポーネント" 設定が [はい] に設定されている場合で、Windows Installer 4.5 が存在するときは、パッチを製品 ABC からアンインストールすると、ターゲット システムでバージョン 1.1.0.0 が復元されます。
- 製品 ABC と製品 XYZ でこの設定の値が [いいえ] の場合、製品 XYZ が 1.1.0.0 を使用するのにも関わらず、ファイルのバージョンは 1.0.0.0 にダウングレードされます。このため、製品 XYZ がバージョン 1.1.0.0 を必要とするときに、XYZ が適切に動作しない可能性があります。

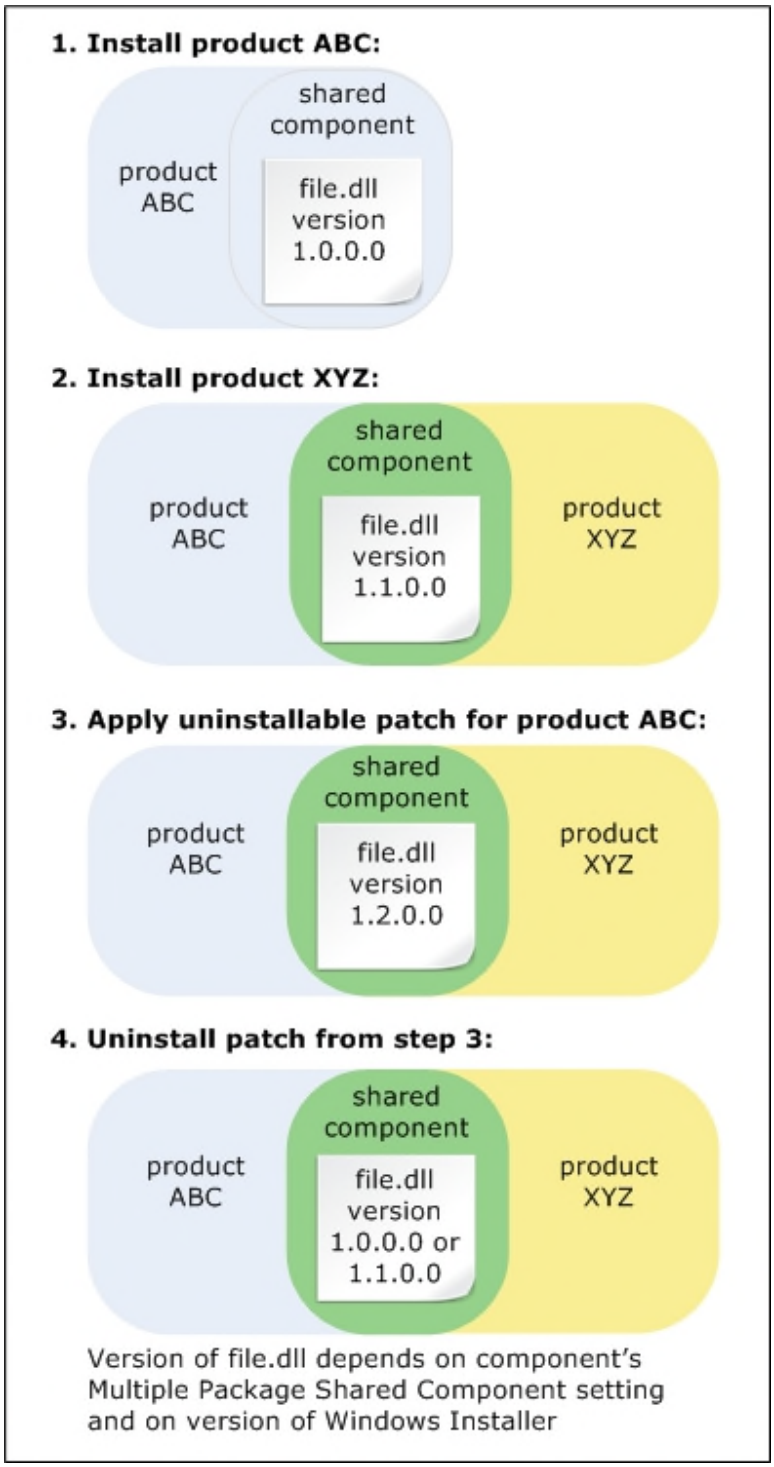


図 1: どちらか 1 つのパッケージで共有コンポーネントの "複数パッケージの共有コンポーネント" 設定が [はい] に設定されている場合で、Windows Installer 4.5 が存在するとき、ファイルのバージョン 1.1.0.0 の復元が可能な場合があります。そうでない場合、ファイルが意図せずにバージョン 1.0.0.0 にダウングレードされる可能性があります。



タスク ***コンポーネントについて、共有コンポーネントのパッチを有効にするかどうかを指定するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. 構成するコンポーネントを選択します。
3. ***"複数パッケージの共有コンポーネント"*** に適切な値を選択します:
   - 共有コンポーネントのパッチを有効にするには、**[はい]** を選択します。InstallShield が選択されたコンポーネントの msidbComponentAttributesShared 属性を設定して、このコンポーネントが共有されることを示します。
   - 共有コンポーネントのパッチを無効にするには、**[いいえ]** を選択します。InstallShield は選択されたコンポーネントの msidbComponentAttributesShared 属性を設定しません。ただし、コンポーネントが別のパッケージと共有される場合、そのパッケージに含まれる同じコンポーネントに msidbComponentAttributesShared 属性が設定されている可能性があります。その場合、Windows Installer 4.5 は、複数パッケージのコンポーネントが共有されていると見なします。

コンポーネントの詳細設定



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[コンポーネント] ビュー (および [セットアップのデザイン] ビュー) 内にあるコンポーネントの [詳細設定] 領域では、特定のコンポーネント タイプのインストール条件を満たすことができます。たとえば .ocx ファイルをターゲット システムにコピーする場合、ファイルのメソッドが適切にアクセスされるようにクラス、ProgID、およびタイプ ライブラリを登録する必要があります。詳細設定は Windows Installer のビルトイン機能を利用して COM サーバーの登録、ODBC ドライバー、データソース、およびトランスレーターの設定、Windows サービスのインストール、制御、および構成、並びにファイル関連付けの登録を行います。

詳細設定を指定してコンポーネントのパブリッシュ、COM サーバー、ファイル拡張サーバー、および MIME の種類の登録を行なうことができます。コンポーネントが選択されている場合、コンポーネントがインストールまたはアドバタイズされるときにターゲット システム上で詳細設定が行なわれます。したがって、ファイルはインストールされた直後に実行可能となります。アドバタイズの一種であるコンポーネントのパブリッシュは、パブリッシュの詳細設定を通して行います。

このセクションには、次のトピックが含まれています:

- COM 登録の設定を手動で構成する
- ファイル拡張子を登録する

- アセンブリのインストール
- コンポーネントのアプリケーションパスを指定する
- デバイスドライバー設定を構成する
- コンポーネントをパブリッシュする

COM 登録の設定を手動で構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*



重要: *COM サーバーをインストールする場合、Windows Installer でそのクラス、ProgID などを登録する方法をお勧めします。自己登録機能を呼び出すことは、セットアップベストプラクティスの違反になります。*

TypeLib テーブルの使用は避けることが推奨されます。さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「TypeLib Table」を参照してください。

COM 登録詳細設定を組み込む最も簡単な方法は、コンポーネントウィザード に必要な情報を抽出させるか、キーファイルの COM データを抽出する方法です。(ビルド時にダイナミック (動的) に抽出することもできます。その場合には、この詳細設定を使用する必要はありません。)コンポーネントのレジストリエクスプローラーで COM 登録詳細設定を変更または COM エントリを作成する作業は、ファイル登録に関する技術詳細に十分精通している開発者向けです。



タスク

COM サーバーを、コンポーネントの詳細設定の編集のみで、インストールするには、以下の手順に従います:

1. COM サーバーが 1 つコンポーネントに追加されていることを確認します。セットアップベストプラクティスによると、ファイルは単一のポータブル実行可能ファイル (.exe、.dll、または.ocx ファイル) である必要です。
2. COM サーバーをコンポーネントのキーファイル にします。
3. コンポーネントの"ビルド時に COM 抽出" 設定を **[いいえ]** に設定します。
4. コンポーネントの**詳細設定**項目を展開して、すべての詳細設定を表示します。
5. **COM の登録** をクリックして、COM サーバー情報を構成します。

[COM 登録] エクスプローラーにあるアイテムの 1 つを右クリックし、COM サーバーの登録情報を変更または作成します。アイテムを右クリックして[名前の変更] をクリックし、新しいアイテムの名前を変更します。



メモ- *ビルドされたマシンの PATH システム変数は、COM サーバーのリンク先となるすべての .dll ファイルのディレクトリを含むように設定する必要があります。この設定を怠った場合、ファイルの登録は失敗し、COM 情報の抽出が行なわれません。*

作成した各登録アイテムまたはサブアイテムの設定を構成します。この他のヘルプは、各登録設定をクリックすると、InstallShield のヘルプ ペインから見ることができます。

COM クラス、プログラム識別子、および **COM 登録** の構成手順については、次を参照してください:

- COM 登録の COM クラスを手動で構成する
- COM 登録の ProgID を手動で構成する
- COM 登録のタイプ ライブラリを手動で構成する

COM 登録の COM クラスを手動で構成する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*



タスク ***クラス ID を登録するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを開き、**[詳細設定]** の下で、**[COM 登録]** を選択します。**[COM 登録]** エクスプローラーが別のウィンドウで表示されます。
3. **[COM 登録]** エクスプローラーで、**[COM クラス]** を右クリックしてから、**[新しい COM クラス]** をクリックします。
4. 必要に応じて、COM クラスの新しい名前を入力します。COM クラスに指定する名前は、デフォルト値として HKEY_CLASSES_ROOT¥CLSID¥<GUID> に登録されます。クラスに新しい名前を入力するには、それを右クリックしてから [**名前の変更**] を選択してください。
5. 新しい COM クラスをクリックして、設定の構成を行います。
6. このクラスのコンテキストの種類を指定します。以下のリストに、COM サーバーの種類に適したサーバー コンテキストを示します。
   - **LocalServer32** — 32 ビット .exe ファイル
   - **LocalServer** — 16 ビット .exe ファイル

- **InprocServer32** — 32 ビット .dll または .ocx ファイル
- **InprocServer** — 16 ビット .dll または .ocx ファイル

サーバー コンテキストが LocalServer または LocalServer32 の場合、コンテキストをクリックして "デフォルトの Inproc ハンドラー" 設定と "引数" 設定を構成します。

COM 登録の ProgID を手動で構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

progID は、**Component.Class.N** の形式で示される、クラス識別用の文字列です。



タスク ***新しいプログラム上の識別子を指定するには、次の操作を実行します。***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを開き、**[詳細設定]** の下で、**[COM 登録]** を選択します。**[COM 登録]** エクスプローラーが別のウィンドウで表示されます。
3. **[COM 登録]** エクスプローラーで、**[ProgID]** を右クリックしてから、**[新しい ProgID]** をクリックします。
4. ProgID の新しい名前を入力します。**Component.Class.N** 形式を使用します。クラスに新しい名前を入力するには、それを右クリックしてから [**名前の変更**] を選択してください。
5. 新しい ProgID をクリックして、その設定を構成します。

バージョン非依存のProgID

バージョン非依存の progID はクラスを **Component.Class**の形式で識別する文字列で、クラスのすべてのバージョンに対して一定です。



タスク ***バージョン非依存の新しいプログラム上の識別子を指定するには、次の操作を実行します。***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを開き、**[詳細設定]** の下で、**[COM 登録]** を選択します。**[COM 登録]** エクスプローラーが別のウィンドウで表示されます。

3. **[COM 登録]** エクスプローラーで、**[バージョン非依存 ProgID]** を右クリックしてから、**[新しい ProgID]** をクリックします。
4. ProgID の新しい名前を入力します。**Component.Class** 形式を使用します。クラスに新しい名前を入力するには、それを右クリックしてから [**名前の変更**] を選択してください。

COM 登録のタイプ ライブラリを手動で構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*



重要: *TypeLib テーブルの使用は避けることが推奨されます。さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「TypeLib Table」を参照してください。*



タスク ***タイプ ライブラリまたは libID を指定するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを開き、**[詳細設定]** の下で、**[COM 登録]** を選択します。**[COM 登録]** エクスプローラーが別のウィンドウで表示されます。
3. **[COM 登録]** エクスプローラーで、**[タイプ ライブラリ]** を右クリックしてから、**[新しい タイプ ライブラリ]** をクリックします。
4. この COM サーバーによって参照されるタイプライブラリ または libID の新しい名前を入力します。タイプライブラリのアイテムに指定する名前は、デフォルト値として HKEY_CLASSES_ROOT¥TYPELIB¥<libID>¥<version> の下に登録されます。**Component.Class** 形式を使用します。クラスに新しい名前を入力するには、それを右クリックしてから [**名前の変更**] を選択してください。
5. 設定を構成するタイプ ライブラリをクリックします。

ファイル拡張子を登録する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

アプリケーションがファイルの操作に一意のファイル拡張子を使用する場合、[ファイルの種類] 領域で、そのファイル タイプを登録することができます。[ファイルの種類] 領域は、[コンポーネント] ビューおよび [セットアップのデザイン] ビュー (インストール プロジェクトでのみ)内の詳細設定からアクセスできます。たとえば、アプリケーションが .xyz という拡張子を持つファイルを操作する場合、このファイルの種類を登録しておくと、ユーザーがそのアイコンをダブルクリックしたときにファイルをそのアプリケーションで開くようにオペレーティング システムに指示することができます。

この詳細設定は、コンポーネントがインストールまたはアドバタイズされている時に、ターゲット システム上でファイルの種類に関する次の情報を登録します。

テーブル 8 ・ ファイルの種類情報

登録されている情報	説明
ファイル拡張子	ファイル拡張子 (.doc や .txt など) をコンポーネントのキーファイル に関連付けることができます。
ProgID	ProgID プロパティを拡張子に設定すると、たとえばファイルタイプの登録を含む extfile などに ProgID を指定することができます。
動詞	エンドユーザーが現在の拡張子を持つファイルを右クリックしたときに Windows Explorer が表示するコンテキストメニューの中に表示されるコマンド動詞 (Open や Print など) を登録できます。
MIME タイプ	多目的で、メディアの種類やコンテンツの種類として知られるインターネットメール拡張子 (MIME) タイプもコンポーネントのキーファイルに登録できます。MIMEの種類をクラス ID に関連付けることもできます。

メモ・ *ファイルの関連付けは HKLM¥SOFTWARE¥Classes と HKCU¥SOFTWARE¥Classes の両方に格納され、マージされたデータは HKEY_CLASSES_ROOT の下に表示されます。Windows Installer による機能のアドバタイズをサポートする場合、レジストリに直接書き込むのではなく、ファイルの種類エディターを使用されることをお勧めします。*

また、Windows Installer はファイル拡張子アドバタイズ情報を、一見ランダムな文字列としてレジストリに書き込みます。しかし、これは正常な動作です。

アセンブリのインストール

プロジェクト・ *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*



メモ- *プロジェクトに .NET アセンブリを追加 する場合、アセンブリをコンポーネントのキー ファイルとして追加して、コンポーネントの "ビルド時に .NET をスキャン" 設定を選択する方法が推奨されます。*

コンポーネントの詳細設定の [アセンブリ] セクションでは、現在のコンポーネントをインストールしたときに、プライベートまたはグローバルの Win32 アセンブリや .NET アセンブリを追加することができます。アセンブリは、システム上の他のアプリケーションに影響を与えない、自己完結型の製品をインストールするときに便利です。

1 つのコンポーネントに複数のアセンブリを含めることはできません。

このセクションには、次のトピックが含まれています:

- アセンブリを追加する
- アセンブリを削除する
- ターゲット システムで .NET アセンブリ サポートをテストする

アセンブリを追加する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

.NET アセンブリをコンポーネントのキーファイルとして追加すると、コンポーネントがビルド時にスキャン されたとき、または スタティック スキャニング ウィザードを実行したときに InstallShield は自動的に .NET アセンブリの設定に値を追加します。



タスク ***コンポーネントに .NET または Win32 アセンブリを追加するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを展開します。

3. **[詳細設定]** 項目をクリックして展開します。
4. **アセンブリ** を右クリックして、**[新しい Win32 アセンブリ]** または **[新しい .NET アセンブリ]** をクリックします。
5. アセンブリをクリックして、**マニフェスト**、**ファイル** **アプリケーション**および関連する設定を構成します。

InstallShield が、アセンブリに入力された情報を Windows Installer データベースの MsiAssembly および MsiAssemblyName テーブルに追加します。これらのテーブルは、ダイレクトエディターで表示、編集することができます。

アセンブリを削除する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*



タスク ***コンポーネントからアセンブリを削除するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. アセンブリを右クリックして [**削除**] をクリックします。

ターゲット システムで .NET アセンブリ サポートをテストする



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ターゲット システムが .NET アセンブリをサポートするかどうかを確認するには、MsiNetAssemblySupport プロパティをテストします。ターゲット システムが Win 32 アセンブリをサポートするかどうかを確認するには、MsiWin32AssemblySupport プロパティをテストします。(Win32 のアセンブリは、Windows XP 以降のみでサポートされています。)アセンブリテーブル、アクションおよびプロパティを使用するには、Windows Installer バージョン 2.0 以降が必要です。

コンポーネントのアプリケーションパスを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[App Paths] レジストリキーは、実行可能ファイルが .dll ファイルを見つけるときに使用するインストール関連レジストリキーで、これがあると PATH 環境変数を変更する必要がありません。実行可能ファイルの App Paths キーは次のような形をしています:

HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥App Paths¥*ファイル名*.exe

プログラムの App Paths キーには、通常、"Path" という名前の値が含まれます。ここにはセミコロンで区切られたディレクトリの一覧が含まれており、プログラムの .dll ファイルはこの中から見つけることができます。アプリケーションの場所がシステムのパスに含まれていない場合、Windows ではこのキーを使用してアプリケーションとその .dll ファイルを探します。さらにエンドユーザーがエクスプローラー シェルによってアプリケーションの実行可能ファイルを移動したり名前を変更したりすると、Windows は自動的にファイルの App Paths キーを更新します。



タスク ***コンポーネントのアプリケーションパスを指定するには、次の操作を実行します。***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. 構成するコンポーネントを選択して、その **[詳細設定]** 項目を展開します。
3. **[詳細設定]** の下の **[アプリケーションのパス]** 項目をクリックします。
4. キーを作成するファイルのチェック ボックスを選択します。
5. **[アプリケーション パス]** 列で、依存関係へのパスを入力するか、パスをハードコーディングしない場合は、Windows Installer フォルダーのプロパティをリストから選択します。複数のパスはセミコロン (;) で区切ります。
6. **[OK]** をクリックします。

デバイスドライバー設定を構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

InstallScript プロジェクトにデバイス ドライバーをインストールするための詳しい情報は、「デバイス ドライバーのインストール」を参照してください。

デバイス ドライバー ウィザード...を実行して、インストールでデバイスドライバーを含めるのに必要なテーブルやエントリ、カスタム アクション、機能、およびコンポーネントを追加した後、デバイスドライバーを含むコンポーネントと関連付けられたプロパティを設定することができます。次の情報は InstallShield 内で設定可能な様々なオプションについて説明します。

[コンポーネント] ビューにあるデバイスドライバー詳細設定の [共通] タブを利用して、現在のコンポーネントにデバイス ドライバーを含めるかどうか、また含める場合、希望のランタイム インストール オプションを選択することができます。[シーケンス] タブでは、プロジェクトのデバイス ドライバー (現在のコンポーネントのデバイス ドライバーだけでなくすべてのデバイス ドライバー) をインストールする順番を指定することができます。

コンポーネントをパブリッシュする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントの [パブリッシュ] 詳細設定では、コンポーネントのパブリッシュ情報を指定できます。パブリッシュ (公開) は、アドバタイズ (ジャストインタイム インストール) の一種です。インストール中にコンポーネントのユーザー インターフェイス要素は作成されませんが、コンポーネントは [プログラムの追加と削除] アプレットからインストールすることができます。また、インストールされたコンポーネントがインストーラーからパブリッシュされたコンポーネントを要求した際にもインストールできます。

各アドバタイズ コンポーネントに対し、1 つ以上のコンポーネント ID を作成しなくてはなりません。



重要: *コンポーネント ID と コンポーネントの "コンポーネント コード" で入力する GUID とは異なりますので、ご注意ください。これらは両方とも一意の値でなくてはなりません。"パブリッシュ" 詳細設定で使用するコン*

ポーネント ID は、修飾コンポーネントとしてグループ化された複数のコンポーネントのカテゴリを表すカテゴリ IDです。

各コンポーネント ID には、少なくとも 1 つの修飾子が必要です。修飾子とは、たとえば言語を指定する場合に、この言語またはコンポーネントのバージョンを他から区別するために使用する文字列です。修飾子は、コンポーネントに対して一意である必要があります。

実行時にインストーラーはコンポーネント ID と修飾子を登録して、これらの固有の値を使って、パブリッシュされたコンポーネントを管理します。Windows Installer 関数を呼び出すことにより、インストールされたコンポーネントまたは別のインストールされたコンポーネント (「クロスプロダクト」アドバタイズという名前で知られています) はアドバタイズされたコンポーネントに関する情報を要求し、それをインストールすることができます。

このセクションには、次のトピックが含まれています:

- パブリッシュ情報を指定する
- パブリッシュされるコンポーネントのコンポーネント ID を追加する
- ComponentID を削除する
- componentID へ修飾子を追加する
- componentID から修飾子を削除する
- 修飾子の設定を構成する

パブリッシュ情報を指定する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントの [パブリッシュ] 詳細設定では、コンポーネントのパブリッシュ情報を指定できます。パブリッシュ (公開) は、アドバタイズ (ジャストインタイム インストール) の一種です。インストール中にコンポーネントのユーザー インターフェイス要素は作成されませんが、コンポーネントは [プログラムの追加と削除] アプレットからインストールすることができます。また、インストールされたコンポーネントがインストーラーからパブリッシュされたコンポーネントを要求した際にもインストールできます。



タスク ***作成したコンポーネントをパブリッシュするには、以下の手順を実行します。***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、パブリッシュするコンポーネントを開き、**[詳細設定]** の下で、**[パブリッシュ]** を選択します。**[パブリッシュ]** エクスプローラーが別のウィンドウで表示されます。

3. **[パブリッシュ]** エクスプローラーを右クリックし、**[新しい ComponentID]** をクリックして新しいコンポーネントの ID を生成します。一意のコンポーネント ID および対応する修飾子が追加されます。
4. [**パブリッシュ**] エクスプローラーで、新しい修飾子をクリックして、その値を構成します。

インストール プロジェクトの [カスタム アクションとシーケンス] ビューで、ターゲット システムで製品をアドバタイズするために必要な条件を設定できます。

タスク ***インストール プロジェクトでアドバタイズ条件を設定するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[シーケンス]** エクスプローラーで、**[アドバタイズ]** アイテムを展開して、**[実行]** アイテムを展開します。
3. 適切な **[実行]** アクションをクリックして、必要に応じて **"条件"** 設定を構成します。

パブリッシュされるコンポーネントのコンポーネント ID を追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントをパブリッシュする場合、少なくとも 1 つのコンポーネント ID を [パブリッシュ] 詳細設定に追加する必要があります。

タスク ***componentID を追加するには、以下の操作を実行します。***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、新しいコンポーネント ID が必要なコンポーネントを開き、**[詳細設定]** の下で、**[パブリッシュ]** を選択します。**[パブリッシュ]** エクスプローラーが別のウィンドウで表示されます。
3. **[パブリッシュ]** エクスプローラーを右クリックしてから、**[新しい componentID]** をクリックします。一意のコンポーネント ID および対応する修飾子が追加されます。

ComponentID を削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*



タスク ***componentID を削除するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーでコンポーネントを開き、**[詳細設定]** の下で、**[パブリッシュ]** を選択します。**[パブリッシュ]** エクスプローラーが別のウィンドウで表示されます。
3. **[パブリッシュ]** エクスプローラーで、componentID を右クリックし **[削除]** をクリックします。

componentID へ修飾子を追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

各 ComponentID には少なくとも 1 つの修飾子が必要です。コンポーネント ID を作成すると、デフォルト修飾子が指定されます。この修飾子の名前を変更するには、右クリックしてから、[名前の変更] をクリックします。



タスク ***新しい修飾子を追加するには、以下の手順を実行します。***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーでコンポーネントを開き、**[詳細設定]** の下で、**[パブリッシュ]** を選択します。**[パブリッシュ]** エクスプローラーが別のウィンドウで表示されます。
3. **[パブリッシュ]** エクスプローラーで、componentID を右クリックし **[新しい修飾子]** をクリックします。InstallShield は、デフォルト名を持つ新しい修飾子を追加します。
4. 修飾子の名前を入力します。

componentID から修飾子を削除する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*



タスク ***修飾子を削除するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーでコンポーネントを開き、**[詳細設定]** の下で、**[パブリッシュ]** を選択します。**[パブリッシュ]** エクスプローラーが別のウィンドウで表示されます。
3. **[パブリッシュ]** エクスプローラーで、修飾子を右クリックし **[削除]** をクリックします。

修飾子の設定を構成する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

各修飾子について、*アプリケーション データ*と呼ばれる情報文字列を指定するオプションがあります。



タスク ***修飾子のプロパティを設定するには、次の操作を実行します。***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーでコンポーネントを開き、**[詳細設定]** の下で、**[パブリッシュ]** を選択します。**[パブリッシュ]** エクスプローラーが別のウィンドウで表示されます。
3. **[パブリッシュ]** エクスプローラーで、修飾子をクリックします。

4. **"アプリケーション データ"** 設定に適切な値を入力します。

   この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

機能の定義

機能は、エンド ユーザーから見て、個別にインストール可能な最小の製品構成単位です。これは、ヘルプ ファイルや製品スイートの一部などの製品の特定機能を表し、エンド ユーザーがこれらをインストールまたはアンインストールするかどうかを決定できます。インストール全体は特定の目的を実行する各機能ごとに分ける必要があります。

サブ機能は機能をさらに分割したものです。機能は、ユーザーが選択してインストールできる製品または製品スイートに含まれた自己完結型の要素である必要があります。そのため、インストールの各部分をある "親" 機能のサブ機能として構成するのが、合理的な方法と言えます。



ヒント: *多くのサブ機能を作成できますが、編成のためにデザインを可能な限り単純にしておくようお勧めします。*

このセクションには、次のトピックが含まれています:

- 機能の作成
- 機能の設定を構成する
- 機能のインストール先を設定する
- 機能の条件を設定する
- 機能をエンドユーザーへ表示する
- 機能の条件付き選択
- 機能の条件付き非表示
- 条件付きで実行時にスイート機能の脆弱性を設定する
- 機能のインストールを必須にする
- 機能のアドバタイズ
- 機能の "インストール レベル" 設定を構成する
- 機能の "リモート インストール" 設定を設定する
- 機能にリリース フラグを使用する
- 機能の並べ替え
- "必要な機能" 設定を使用する

機能の作成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*

[セットアップのデザイン] ビューまたは [機能] ビューを使って、プロジェクトの機能およびサブ機能を作成できます。



タスク

機能を作成するには、以下の手順に従います:

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. エクスプローラーで最上部のアイテムを右クリックして、[**新しい機能**]をクリックします。InstallShield が、**New Feature_n** (ここで *n* は連続番号です) というデフォルトの名前で新しい機能を追加します。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。
4. 右側のペインで機能の設定を構成します。



プロジェクト: *基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合、機能名には必ずアルファベット、数字、アンダースコア (_)、およびピリオド (.) のみを使用し、名前の先頭にはアルファベット、またはアンダースコアを使用しなくてはなりません。*

InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトでは、次の文字を機能名に使用できません。

*¥ / : * ? " ' < > |*



ヒント: *サブ機能を追加するには、その親機能を右クリックして [新しい機能] をクリックします。*

*新しい機能を追加し、機能の名前に K***Feature 1¥Feature 2¥Feature 3** *と入力することで、ネストされた複数の機能を一度に作成することができます。InstallShield は、***Feature 3** *が* **Feature 2** *のサブ機能であり、Feature 2 が* **Feature 1** *のサブ機能である、ネストされた機能構造を作成します。*

プロジェクトの機能やサブ機能のすべてを作成した後、コンポーネントを作成して、機能と関連付ける必要があります。

機能の設定を構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*



タスク ***機能の設定を構成するには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. 右側ペインに表示されているグリッド内の設定を構成します。

機能のインストール先を設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

各機能およびサブ機能のファイルに対し、別のインストール先を指定できます。新しい機能の "インストール先" 設定のデフォルト値は INSTALLDIR で、これは [一般情報] ビューで設定できます。



Windows ロゴ: *Windows ロゴ要件によると、ターゲット システムの言語にかかわらず、製品のファイルのデフォルトのインストール先を Program Files のサブフォルダー、またはエンド ユーザー のアプリケーション データ フォルダーにする必要があります。*



タスク ***機能のインストール先フォルダーを変更するには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. インストール先を変更する機能を選択します。
3. ***"インストール先"*** 設定で、一覧からオプションの 1 つを選択するか、省略記号ボタン (...) をクリックして、ディレクトリを選択または作成します。

インストール先を実行時に構成できるようにするには、選択するインストール先フォルダーに、必ずパブリック プロパティ (すべて大文字で表記) を選択してください。



メモ- *機能の "インストール先" 設定は空白にしておくことができます。*

インストール先フォルダーに関するその他の考慮事項

実行時における機能のインストール先の変更

機能のインストール先ディレクトリを、実行時にエンド ユーザーの機能選択に基づいて変更する場合、Windows Installer カスタム アクション タイプ 35 を利用することができます。

ディレクトリのターゲット先を変更する方法に関する詳細は、Windows Installer ヘルプ ライブラリの「Changing the Target Location for a Directory」を参照してください。

機能の "リモート インストール" 設定

機能の "リモートインストール" 設定を [ソースを優先] (または、サブ機能の親機能が [ソースを優先] に設定されている場合は [親を優先]) に設定すると、機能の "インストール先" 設定に関わらず、機能のファイルはターゲット システムにインストールされません。

"コンポーネントのインストール先" 設定

各コンポーネントにも "インストール先" 設定があります。機能の "インストール先" 設定とコンポーネントの "インストール先" 設定の値が異なる場合、コンポーネントの "インストール先" 設定が機能のそれをオーバーライドします。機能の "インストール先" 設定はオプションですが、コンポーネントの "インストール先" 設定は必須です。

デフォルトのインストール先としての INSTALLDIR の使用

すべての機能とコンポーネントのデフォルトのインストール先プロパティとして INSTALLDIR を設定しているのは、すべてのアプリケーションのファイルを同じルートフォルダーにインストールするためです。この結果、エンド ユーザーがある機能のインストール先フォルダーを CustomSetup ダイアログで変更すると、INSTALLDIR に含まれるパスにインストールするよう設定されたすべての機能のインストール先フォルダーも変更されます。



メモ- *コンポーネントのインストール先が* INSTALLDIR *以外に設定されていると、コンポーネントは各コンポーネントに指定されたインストール先にインストールされるため、*INSTALLDIR *を変更してもコンポーネントのインストール先には影響がありません。*

INSTALLDIR *などのディレクトリ プロパティは、デフォルト値を指定します。エンドユーザーは、*Msiexec.exe *起動時に、コマンドラインでプロパティを設定したり、CustomSetup ダイアログで機能に対して新しいインストール先フォルダーを選択することにより、この値を変更できます。*

機能の条件を設定する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

試用版と完全版など、同じプログラムの異なるバージョンを作成する場合、機能の条件付きインストールが便利です。トライアル版では、一部の機能を提供しないようにするため、一部の機能をインストールしません。また、機能の条件付きインストールによって、ディスク容量を節約することもできます。ターゲット システムにすべての機能をインストールするのに十分なディスク容量がない場合、不要な機能を条件付きでインストールするよう設定できます。



タスク ***プロジェクトの機能に条件を設定するには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. **"条件"** 設定をクリックして、省略記号ボタン (...) をクリックします。**[機能条件ビルダー]** ダイアログ ボックスが開きます。
4. **[新しい条件]** ボタンをクリックします。InstallShield によって、**"条件"** ボックスの下に新しい条件が追加されます。
5. **[レベル]** 列に、条件が満たされている場合に機能に使用するインストール レベルを入力します。

   条件が満たされている場合、この値は機能の "インストール レベル" 設定で指定された値を上書きします。
6. 以下のいずれかを実行します。
   - **[条件]** 列で、機能の条件を入力します。
   - **[プロパティ]** リスト、**[演算子]** リスト、および **[追加]** ボタンを使って、条件をビルドします:
     a. **[プロパティ]** リストで、プロパティを選択してから、**[追加]** ボタンをクリックします。InstallShield は、プロパティを **[条件]** 列に追加します。
     b. 条件ステートメントに演算子を含める場合、**[演算子]** リストから演算子を選択してから、**[追加]** ボタンをクリックします。InstallShield によって、条件ステートメントに演算子が追加されます。
     c. 条件ステートメントが値を含む場合、"条件" フィールドをダブルクリックしてから END を押し、挿入ポイントを条件ステートメントの終わりに移動して、値を入力します。
7. **[OK]** をクリックします。



重要: *InstallShield は、基本の条件検証を行いますが、条件ステートメントが予定通りの結果を評価することを確認してください。詳しい情報と条件のサンプルについては、「条件ステートメントのビルド」を参照してください。*

機能条件の実行時の動作

機能のデフォルトのインストール レベルは、[機能] ビューまたは [セットアップのデザイン] ビューの "インストール レベル" 設定で構成された値です。この値は、機能の条件が True 評価された場合に上書きされます。その場合、機能のインストール レベルは、True 条件ステートメントに関連付けられている "レベル" 値に設定されます。

各機能のインストール レベル値は、グローバル パブリック プロパティ INSTALLLEVEL の値と比較され、インストール レベル値が INSTALLLEVEL と等しいかそれ以下の機能だけがインストール用に選択されます。

たとえば、ユーザーが昇格された権限を持つ場合のみデフォルトで選択されるようにする機能がある場合、この機能に対して、条件 **Not Privileged** およびインストール レベル **200** を設定できます。エンドユーザーが昇格された権限を持たない場合、この条件が True 評価され、機能にインストール レベル 200 が与えられます。200 は、デフォルトの製品インストール レベル (100) より大きいため、機能は選択されません。

ヒント: *インストール レベルを数値 0 に設定する条件を機能に与えることによって、機能を条件付きで非表示にすることができます。条件が True 評価されると、機能は選択解除されて、CustomSetup ダイアログにも表示されません。*

機能をエンドユーザーへ表示する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

InstallShield では、プロジェクトの種類に対応するカスタム セットアップ ダイアログで、エンド ユーザー向けに機能が表示される方法を指定することができます。

- 基本の MSI、MSI データベース、およびトランスフォーム プロジェクトの場合 – CustomSetup ダイアログ
- InstallScript MSI プロジェクトの場合 – **SdFeatureDialog2**、**SdFeatureMult** または **SdFeatureTree**

[機能] ビューまたは [セットアップのデザイン] ビューにある機能の "表示" 設定を使って、機能を表示するかどうか、およびどのように表示するかを指定できます。選択可能なオプションは以下のとおりです:

テーブル 9 · "表示" 設定で使用できるオプション

オプション	説明
閉じて表示する	デフォルトで、実行時ダイアログにサブ機能が閉じた状態で、機能が表示されます。
展開して表示する	デフォルトで、実行時ダイアログにサブ機能が展開された状態で、機能が表示されます。
非表示	実行時ダイアログに、機能およびそのサブ機能は表示させません。

メモ: *この設定で [表示しない] を選択しても、機能がインストールされるかどうかには影響しません。非表示にした機能は、自動的にすべてインストールされるというわけではなく、インストールが必要な機能の場合は選択解除できず、インストールするべきではない機能の場合は選択できないようになります。*

機能の条件付き選択

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

実行時に機能を条件付きで選択する手順は、使用するプロジェクトの種類によって異なります。

基本の MSI、MSI データベース、およびトランスフォーム プロジェクト

機能の [条件] 設定を使って、指定した条件が満たされた場合のデフォルト以外のインストールレベル値を、指定できます。機能のインストールレベル値が、プロジェクトの INSTALLLEVEL プロパティ以下の場合、その機能は選択されてインストールされます。

タスク

たとえば、管理者権限を持たないエンドユーザーに対しては非選択にしたい機能があれば、次の手順で設定をします。

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. **"条件"** 設定をクリックして、省略記号ボタン (...) をクリックします。**[機能条件ビルダー]** ダイアログ ボックスが開きます。
4. **[新しい条件]** ボタンをクリックします。InstallShield によって、**"条件"** ボックスの下に新しい条件が追加されます。
5. **[レベル]** 列で、**200** と入力します。
6. **[条件]** 列に、**Not AdminUser** と入力します。
7. **[OK]** をクリックします。

実行時、エンドユーザーが管理権限を持たない場合(つまり条件が成立すると)、機能の "インストール レベル" プロパティは 200 に設定されます。プロジェクトのデフォルト INSTALLLEVEL プロパティは 100 なので、機能は選択解除されます。

タスク

INSTALLLEVEL プロパティをプロパティマネージャー で変更することができます。

InstallScript および InstallScript MSI プロジェクト

FeatureSelectItem 関数を使うと、**SdFeatureTree** などの機能選択ダイアログ ボックスに表示される機能の選択/非選択を設定できます。

機能の条件付き非表示



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

実行時に機能を条件付きで隠す手順は、使用するプロジェクトの種類によって異なります。

基本の MSI、MSI データベース、およびトランスフォーム プロジェクト

インストールレベルをゼロとした機能は、すべて非表示 (かつ非選択) にされます。



タスク ***たとえば、インストールの実行エンドユーザーが管理者権限を持たない場合に、特定の機能を非表示にしたければ、次の手順で設定をします。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. **"条件"** 設定をクリックして、省略記号ボタン (...) をクリックします。**[機能条件ビルダー]** ダイアログ ボックスが開きます。
4. **[新しい条件]** ボタンをクリックします。InstallShield によって、**"条件"** ボックスの下に新しい条件が追加されます。
5. **[レベル]** 列で、**0** と入力します。
6. **[条件]** 列に、**Not AdminUser** と入力します。
7. **[OK]** をクリックします。

このプロジェクトを再ビルドしてインストールを実行すると、管理者権限を持たないエンドユーザーに対しては、ここで設定した機能は非表示にされインストールもされません。

InstallScript および InstallScript MSI プロジェクト

FeatureSetData 関数は FEATURE_FIELD_VISIBLE 定数を受け取り、これを使って特定の機能を表示するかどうかをコントロールすることができます。たとえば、HiddenFeature という名前の機能を非表示にする場合は、スクリプトに次の関数呼び出しを置きます。

```
FeatureSetData(MEDIA,
   "HiddenFeature",
   FEATURE_FIELD_VISIBLE, FALSE,
   "");
```



メモ- *機能を非表示にしても、その機能の選択は解除されません。機能のデータがインストールされないように選択を解除するには、以下を呼び出します:*

FeatureSelectItem(MEDIA, "FeatureName", FALSE);

条件付きで実行時にスイート機能の脆弱性を設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では、インストール デザイナーの **[機能]** ビューで **"可視"** プロパティの下にある **[条件]** オプションを使って、実行時プロパティに基づいて条件付きで機能を表示または非表示することができます。

"条件" 設定を使って、**InstallationFeatures** ウィザード ページで機能をインストール可能としてデフォルトで表示するかどうかを評価するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する 1 つ以上の条件を指定できます。

たとえば、Windows の特定バージョンが搭載されたターゲット システム上では特定の機能をデフォルトで表示する場合、その Windows バージョンを指定する条件を作成します。



タスク

アドバンスト UI またはスイート/アドバンスト UI インストールで条件付きで機能を表示する場合、以下の手順に従います:

1. **[機能]** ビューで **"表示"** プロパティの下にある **[条件]** 行をクリックします。緑色の + 記号 (**[新しい条件]** ボタン) が行の終わりに表示されます。
2. **[新しい条件]** ボタンをクリックします。**[条件]** 行の下に新しい行が追加されます。
3. **[新しい条件]** ボタンの横にある下矢印をクリックして、一覧から適切なオプション (**[All]**、**[Any]**、または **[None]**) を選択します。
4. 次に同じ行で、**[新しい条件]** ボタンをクリックし、適切なオプションを選択してから条件ステートメントのビルドを続行します。

   1 つ以上の条件ステートメントが構成されると、**"条件"** プロパティには **(条件)** と表示されます。何も構成されていない場合、**"条件"** プロパティには **(空白)** と表示されます。

詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。

条件付きの表示をサポートするための、オートメーション インターフェイスのメソッド

オートメーション インターフェイスで次のメソッドを使って、実行時に条件付きでスイート プロジェクト内の機能の表示/非表示を設定することもできます:

テーブル 10・スイート プロジェクトで条件付き表示をサポートするオートメーション インターフェイス メソッド

メソッド	構文
AddVisibleCondition	AddVisibleCondition() As ISWiSuiteCondition

テーブル 10 • スイート プロジェクトで条件付き表示をサポートするオートメーション インターフェイス メソッド

メソッド	構文
DeleteVisibleCondition	DeleteVisibleCondition()
VisibleCondition	読み取り専用オブジェクト プロパティ

機能のインストールを必須にする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

機能の "必須" 設定に [はい] を選択すると、エンド ユーザーは CustomSetup ダイアログ (基本の MSI 、MSI データベース、およびトランスフォーム プロジェクトの場合)、あるいは **SdFeatureDialog2**、**SdFeatureMult**、または **SdFeatureTree** ダイアログ (InstallScript MSI プロジェクトの場合) でそれを選択解除できません。その機能はターゲット システムにインストールされます。

"必須" 設定が [いいえ] に設定されていると、機能はデフォルトでインストールされますが、エンドユーザーは選択を解除できます。

InstallScript MSI プロジェクトでは、"必須" 設定は初回インストール中にルート レベルの機能に適用します。アップグレードまたはパッチに含まれるサブ機能にも適用できます。この設定は、InstallScript MSI インストールの初回インストール中、サブ機能には適用しません。

機能のアドバタイズ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*
- *トランスフォーム*

InstallShield では、機能のアドバタイズを選択的に有効または無効にすることができます。アドバタイズされた機能は、インストール過程ですぐにはインストールされません。要求されたときにインストールされます。機能を割り当てると、その機能はすでにインストールされているように見えますが、エンドユーザーが要求するまではインストールされません。(機能を割り当てるとショートカットがインストールされ、コントロール パネルの [プログラムの追加と削除] アプレットからインストールできます。ただし割り当てられた機能は、ユーザーが要求するまではアドバタイズされるだけです。)パブリッシュされた機能は、インストーラーから要求されるまでターゲット システムに表示されません。(パブリッシュされた機能には、エンド ユーザー インターフェイス要素がありません。これらのインストールは、プログラム上で行うか、MIMEの種類を関連付けて行います。)

[機能] ビューの "アドバタイズ" 設定を使って、アドバタイズを許可するかどうかを指定します。この設定で選択できるオプションは、次のとおりです:

テーブル 11 · "アドバタイズ" 設定で使用できるオプション

オプション	説明
アドバタイズを許可する	エンド ユーザーは、CustomSetp ダイアログで、この機能のアドバタイズ オプションを選択できます。アドバタイズは許可されますが、インストール実行時のデフォルトのオプションではありません。
アドバタイズを優先する	機能は、デフォルトでアドバタイズされます。エンド ユーザーは、CustomSetup ダイアログの機能のアドバタイズ オプションを変更できます。
アドバタイズを許可しない	この機能には、アドバタイズは許可されません。エンドユーザーは、CustomSetup ダイアログで機能のアドバタイズを選択できません。
サポートされていない場合にアドバタイズを無効にする	アドバタイズメントは、Internet Explorer 4.01 以上のシステムでのみ機能します。この条件を満たさないターゲット システムでは、アドバタイズは許可されません。ターゲット システムがアドバタイズをサポートする場合、アドバタイズは許可されます。

機能のアドバタイズを有効にすると、アドバタイズを阻止するその他の要素がない限り、インストールが実行されているモードにかかわらず機能はアドバタイズされます。CustomSetup ダイアログで、エン ドユーザーは直ぐにインストールする機能と、後で使用する機能を制御できます。

アドバタイズする場合には、通常、アプリケーションでこの機能をサポートしている必要があります。たとえば、ある製品でスペルチェック機能をアドバタイズするとします。アプリケーションインターフェイスは、メニュー コマンドまたはツールバーボタンから、スペルチェックを提供します。さらに、機能のインストール状態を確認し、カスタマーが [スペルチェック] コマンドまたはボタンをクリックするとそれがインストールされるようにアプリケーションに書き込む必要があります。

機能の "インストール レベル" 設定を構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

*InstallScript MSI プロジェクトでは、この情報は*セットアップの種類*が定義されていない場合にのみ適用します。*

"インストールレベル" 設定は、実行時に INSTALLLEVEL プロパティと比較され、どの機能をインストールできるかを決定します。この設定を使って、特定の機能設定を作成することもできます。

エンド ユーザーが CustomSetup ダイアログで機能を選択解除しない限り、パッケージの INSTALLLEVEL プロパティの値と同等またはそれ以下のインストールレベルの機能がすべてターゲット システムにインストールされます。



***メモ・** パッケージの* INSTALLLEVEL *プロパティをプロパティマネージャーで変更することができます。*



タスク ***機能のインストールレベルプロパティを設定するには、次の操作を実行します。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. **"インストール レベル"** 設定で、この機能のインストール レベルに整数を入力します。推奨値は **100** です。

機能の "リモート インストール" 設定を設定する



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

機能の "リモート インストール" 設定によって、この機能のファイルをターゲット システムにインストールするか、または CD-ROM やネットワークサーバーのようなソース メディアから実行するかが決定されます。新しい機能のデフォルト値は "ローカルを優先" です。これは、選択された機能のファイルがターゲット システムにインストールされることを意味します。



タスク ***"リモート インストール" 設定を変更して、機能のファイルがソース メディアからのみ実行されるようにするには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. **"リモート インストール"** 設定で、**[ソースを優先]**を選択します。



***ヒント・** [**親を優先**] を選択すると、サブ機能にその親機能と同じ値が指定されます。*



***注意・** 機能のコンポーネントに Windows サービスが含まれる場合は、"リモート インストール' 設定で [ローカルを優先する] を選択します。エンド ユーザーは CustomSetup ダイアログを使用してインストール状態を変更することができますが、Windows Installer はサービスをリモート インストールすることはできません。*

機能にリリース フラグを使用する



プロジェクト: *次のプロジェクト タイプは、リリース フラグのサポートを含みます:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

リリースフラグを使用すると、複数のプロジェクトを作成せずに、異なるバージョンの製品を作成できます。機能のフィルタリングには、リリースフラグに応じて 2 つのステップ (リリース フラグの割り当てと、リリースに含めるフラグの指定) があります。

リリース フラグを機能に割り当てる



プロジェクト: *次のプロジェクト タイプは、リリース フラグのサポートを含みます:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

リリースフラグは、特定のビルドから除外する機能に設定する必要があります。たとえば、製品の特別版だけに含みたいアドオンという機能がある場合、この機能にフラグを付けて、必要なときだけ含めることができます。



タスク

リリースフラグを機能に追加するには、次の手順で行ってください。

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. **"リリース フラグ"** 設定で、文字列を入力します。文字列には、文字または数字を使用できます。複数のフラグをこの機能に付ける場合は、カンマを使用してフラグを区切ります。

リリース フラグに従って機能をフィルターにかける方法については、「リリース フラグ」を参照してください。

リリース フラグを削除する



プロジェクト: *次のプロジェクト タイプは、リリース フラグのサポートを含みます:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

タスク ***機能からリリースフラグを削除するには、以下の手順を実行します。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. **[リリース フラグ]** 設定で、値を削除します。

機能の並べ替え

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

エンド ユーザーが製品をインストールするとき、CustomSetup ダイアログ (基本の MSI、MSI データベース、およびトランスフォーム インストールの場合) または機能選択ダイアログの 1 つ (InstallScript または InstallScript MSI インストールの場合) では、InstallShield の [機能] ビューで表示されるのと同じ順序で機能が表示されます。機能が表示される順番を変更することができます。

タスク ***機能の順序を変更するには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 移動する機能を右クリックして、**[上に移動]** または **[下に移動]** をクリックします。機能を右または左に移動したり、それによって別の機能のサブ機能やトップレベルの機能に変更することもできます。

ヒント: *ドラッグアンドドロップ操作で、機能の順序を変更できます。機能またはサブ機能は、すべてこの方法で移動できます。*

"必要な機能" 設定を使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

"必要な機能" 設定では、選択された機能に必要な機能を指定できます。たとえば、**ProgramFiles** と **HelpFiles** という 2 つの機能を持つインストールがあると想定します。**HelpFiles** 機能が選択されたときにエンドユーザーが **ProgramFiles** をインストールしなければならない場合は、"必要な機能" プロパティを使用する必要があります。



タスク ***必要な機能を設定するには、次の操作を実行します。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. **HelpFiles** 機能を選択します。
3. **必要な機能** プロパティをクリックして、省略記号 (...) ボタンをクリックします。**[必要な機能]** ダイアログ ボックスが開きます。
4. **ProgramFiles** チェック ボックスを選択します。
5. **[OK]** をクリックします。

実行時に、ユーザーがカスタム セットアップタイプを選択すると、**SdFeatureTree** などの機能選択ダイアログは、**HelpFiles** 機能が選択されているときにユーザーが **ProgramFiles** 機能の選択を解除できないようにします。

セットアップの種類について



プロジェクト: *セットアップ タイプは、次のプロジェクト タイプで使用できます:*

- *InstallScript*
- *InstallScript MSI*

基本の MSI プロジェクト用のセットアップ タイプを作成するには、機能の "インストール レベル" 設定を使用します。

セットアップ タイプを使って、エンド ユーザーに様々なバージョンの製品を提供できます。たとえば、デフォルトのセットアップの種類は「完全」と「カスタム」です。「完全」セットアップは、インストールに含まれるすべてのファイルをインストールします。カスタム セットアップの種類は、どの機能をインストールするかエンド ユーザーが選択することができます。

セットアップの種類は機能に基づいています。各セットアップ タイプに関連付ける機能を選択します。次に、エンドユーザーが特定のセットアップの種類を選択すると、そのセットアップの種類に関連付けた機能だけがインストールされます。

デフォルトでは、作成する各プロジェクトは定義済みセットアップの種類 を含みます。[セットアップの種類] ビューで、セットアップの種類を追加/削除したり、既存のセットアップの種類名を変更できるほか、各セットアップの種類に関連付けられている機能を変更することもできます。詳細は、次のセクションを参照してください。

- セットアップの種類を追加する
- セットアップの種類を編集する
- セットアップの種類の名前を変更する
- セットアップの種類を削除する

セットアップの種類を追加する



プロジェクト: *セットアップ タイプは、次のプロジェクト タイプで使用できます:*

- *InstallScript*
- *InstallScript MSI*

基本の MSI プロジェクト用のセットアップ タイプを作成するには、機能の "インストール レベル" 設定を使用します。



タスク ***セットアップの種類を追加するには、次の手順を実行します。***

1. **[編成]** のビュー リストにある **[セットアップの種類]** をクリックします。
2. **[セットアップの種類]** エクスプローラーを右クリックし、**[追加]** をクリックします。InstallShieldが、以前のセットアップを新規追加します。
3. セットアップのタイプの新しい名前を入力します。

セットアップの種類を編集する



プロジェクト: *セットアップ タイプは、次のプロジェクト タイプで使用できます:*

- *InstallScript*
- *InstallScript MSI*

基本の MSI プロジェクト用のセットアップ タイプを作成するには、機能の "インストール レベル" 設定を使用します。



タスク ***セットアップの種類に関連付けられている機能を変更するには、次の操作を実行します。***

1. **[編成]** のビュー リストにある **[セットアップの種類]** をクリックします。
2. **[セットアップの種類]** エクスプローラーで、編集するセットアップの種類をクリックします。インストール プロジェクトに含まれるすべての機能はペインの左下に表示されます。
3. 選択したセットアップの種類に含めない機能の横にあるチェック ボックスを選択解除します。含める機能の横にあるチェック ボックスを選択します。

[セットアップの種類] ペインにリストされているすべてのセットアップのタイプは、自動的にインストール プロジェクトに追加されます。



メモ: ***SetupType2*** *関数は、それぞれに規定の説明テキストがある標準のセットアップの種類 (完全およびカスタム) のみを表示します。より幅広い柔軟性が必要な場合は、スクリプトで* **SetupType2** *の代わりに* **SdSetupTypeEx** *を呼び出します。(***OnFirstUIBefore*** *イベント ハンドラーのデフォルト コードには、***SetupType2*** *関数への呼び出しが含まれています。)*



ヒント: *セットアップの種類にアクセラレータ キーを提供する場合、名前の文字の前にアンパサンド (&) を入力してください。たとえば、**Cu&stom** という名前は、Custom というラベルになります。エンドユーザーがセットアップ時に S キーを押すことでそのオプションボタンを選択することができるようになります。*

セットアップの種類の名前を変更する



プロジェクト: *セットアップ タイプは、次のプロジェクト タイプで使用できます:*

- *InstallScript*
- *InstallScript MSI*

基本の MSI プロジェクト用のセットアップ タイプを作成するには、機能の "インストール レベル" 設定を使用します。



タスク ***セットアップの種類の名前を変更するには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップの種類]** をクリックします。
2. **[セットアップの種類]** エクスプローラーで、編集するセットアップの種類をクリックし、**[名前の変更]** をクリックします。
3. セットアップの種類の新しい名前を入力します。

これで、選択したセットアップ タイプの "表示名" 設定が更新されます。この名前は、実行時に [セットアップの種類] ダイアログに表示されます。

セットアップの種類を削除する



プロジェクト: *セットアップ タイプは、次のプロジェクト タイプで使用できます:*

- *InstallScript*
- *InstallScript MSI*

基本の MSI プロジェクト用のセットアップ タイプを作成するには、機能の "インストール レベル" 設定を使用します。



タスク ***セットアップの種類を削除するには、次の手順を実行します。***

1. **[編成]** のビュー リストにある **[セットアップの種類]** をクリックします。
2. **[セットアップの種類]** エクスプローラーで、削除するセットアップの種類をクリックし、**[削除]** をクリックします。

インストールに再配布可能ファイルを含める

InstallShield は、一般によく利用されるサードパーティ再配布可能ファイルを含み、.NET Framework のような技術に対するサポートをインストールに簡単に組み込むことができます。プロジェクトへ再配布可能ファイルを追加すると、再配布可能ファイルおよびすべての関連ファイルがインストールへ追加されます。これによって再配布可能ファイルのパッケージ処理が簡素化され、内部または外部での利用で一貫性を保ちやすくなります。

[再配布可能ファイル] ビュー(InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトでは [オブジェクト] ビュー) には、すべての InstallShield オブジェクト並びに InstallShield に含まれているサードパーティ マージ モジュールが含まれています。基本の MSI プロジェクトおよび InstallScript MSI プロジェクトでは、このビューにはインストールに追加可能な InstallShield 前提条件も含まれます。InstallScript プロジェクトでは、[前提条件] ビューを使ってインストールに InstallShield 前提条件を追加できます。

InstallShield 前提条件

InstallShield 前提条件 は、製品が必要とする製品、またはテクノロジ フレームワークのためのインストールです。任意の既存の InstallShield 前提条件をインストール プロジェクトに追加して、その設定の多くを構成することができます。また、独自の InstallShield 前提条件を作成して、それをプロジェクトに追加することもできます。

InstallShield は InstallShield 前提条件のベース セットを含みます。また、では InstallShieldInstallShield 前提条件エディターを利用してカスタム InstallShield 前提条件を定義したり、既存の InstallShield 前提条件の設定を編集したりできます。

InstallShield では、次の 2 つのタイプの InstallShield 前提条件がサポートされています:

- **セットアップ前提条件** – この種類の前提条件のインストールは、インストールの実行の前に実行されます。
- **機能前提条件** – この種類の前提条件は、1 つまたは複数の機能に関連付けられています。機能前提条件は、前提条件を含む機能がインストールされたときに、その前提条件がシステム上に既にインストールされていない場合にインストールされます。したがって、機能の条件がターゲット システム上で満たされていない場合、またはエンド ユーザーが機能のインストールを行わないことを選択した場合、その機能はインストールされません。その結果、インストールされる別の機能にも機能前提条件が関連付けられていない限り、関連付けられた機能前提条件はいずれもインストールされません。



プロジェクト: *基本の MSI プロジェクトが、機能前提条件のサポートを含みます。*

InstallShield には、InstallShield 前提条件を、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに、パッケージとして含めることができるサポートも含まれています。詳細については、「InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める」を参照してください。

マージ モジュール

マージ モジュール (.msm ファイル) には、個別機能をインストールするために必要なロジックとファイルのすべてが含まれています。たとえば、一部のアプリケーションには、Visual Basic ランタイム ライブラリが必要です。機能にファイルを含めてインストール要件を調べる必要はなく、プロジェクトに含まれる機能の 1 つに Microsoft C++ ランタイム ライブラリを添付するだけでこれを実行できます。



メモ: *[再配布可能ファイル] ビューに含まれているマージ モジュールの多くは Microsoft またはその他のサードパーティによるものです。InstallShieldでは、これらのモジュールを無料配布することによって、インストール プロジェクトの作成を支援します。ただし、サードパーティが作成したモジュールに存在する問題を InstallShield が*

修正したり直すことはできません。サード パーティが作成したモジュールに関する問題は、ベンダーへお問い合わせください。

オブジェクト

オブジェクトには、マージ モジュール同様、個別の機能をインストールするために必要なロジックとファイルがすべて含まれています。InstallShield に含まれる DirectX オブジェクトなどのオブジェクトは、ウィザードを使ってカスタマイズする必要があります。オブジェクトをインストールに追加すると、すぐに該当するカスタマイズ ウィザードが開きます。オブジェクトを追加時点でカスタマイズすることも、ウィザードをいったんキャンセルし、後でオブジェクトを右クリックして [オブジェクトの設定変更] を選択することによってカスタマイズすることもできます。

ライブ再配布可能ファイルギャラリー

多くの再配布可能ファイルはサイズが大きいため、プロジェクトで利用可能なものでも InstallShield のインストールと同時にコンピューターへ追加されない場合があります。その場合も、これらの再配布可能ファイルはインターネットからコンピューターへダウンロードすることができます。

構成可能マージ モジュール

構成可能な再配布可能ファイルは、ModuleSubstitution テーブル内の少なくとも 1 行によって参照される ModuleConfiguration テーブル内の少なくとも 1 行を含むマージ モジュールまたはオブジェクトです。これによって再配布可能ファイルの値を変更することができます。[再配布可能ファイル] ビューで構成可能モジュールを選択した場合、表示される [マージ モジュール構成可能値] ダイアログ ボックスでモジュール追加時にそれを構成することができます。マージ モジュールを後でカスタマイズするには、それを右クリックして [マージ モジュールの構成]を選択します。

このセクションには、次のトピックが含まれています:

- 再配布可能ファイルの出荷
- 再配布可能ファイルギャラリーを管理する
- InstallShield 前提条件、マージ モジュール、およびオブジェクトをプロジェクトに組み込む
- インストール プロジェクトに含まれている InstallShield 前提条件を利用する
- インストール プロジェクトに含まれているマージ モジュールを使用する
- Windows Installer 再配布可能ファイルをプロジェクトに追加する
- .NET Framework 再配布可能ファイルをプロジェクトへ追加する
- MySQL Connector ODBC 前提条件を含める
- Oracle 11g Instant Client の InstallShield 前提条件を含める
- DirectX 9.0 オブジェクトを含める

ModuleConfiguration テーブル (Windows Installer ヘルプ ライブラリ)
ModuleSubstitution テーブル (Windows Installer ヘルプ ライブラリ)

再配布可能ファイルの出荷

InstallShield は、インストール プロジェクトに組み込むことが可能なサードパーティ再配布可能ファイルを提供します。プロジェクトに、たとえば Crystal Reports といった再配布可能技術を含む場合、その再配布可能ファイルについてベンダーからライセンスが付与されていなくてはなりません。適切なライセンス無しにこれらの技術を再配布することは法的に許可されていません。詳細は、ベンダーのマニュアルを参照してください。

InstallShield プロジェクトでリリースをビルドする時、InstallShield によってビルド出力に様々な InstallShield 再配布可能ファイルが含まれます。InstallShield の使用許諾契約に基づいて、ビルド出力に含まれるこれらの InstallShield 再配布可能ファイルを使用することができます。これらのファイルのほとんどは *InstallShield Program Files フォルダー*¥Redist フォルダーにインストールされ、必要に応じてビルドに含まれます。以下はInstallShield 再配布可能ファイルのリストです。

- _isres_LanguageID.dll
- AppxStub.exe
- ClrSuitePSHelper.dll
- ClrWrap.dll
- CommonHelper.dll
- corecomp.ini
- default.pal
- DLLWrap.dll
- dotnetfx.exe
- DotNetInstaller.exe
- EulaScrollWatcher.dll
- FileBrowse.dll
- IISHelper.dll
- IISRT.dll
- InstallShield.ClrHelper.dll
- InstallShield.Interop.Msi.dll
- ISBEW.exe
- ISChain.exe
- ISChainPackages.dll
- ISComSrv.dll
- ISExpHlp.dll
- isexternalui.dll
- IsLockPermissions.dll
- ISNetAPI.dll
- ISNetApiRT.dll

- ISNetworkShares.dll
- ISRegSvr.dll
- Isrt.dll
- ISScheduledTasks.dll
- IsSchRpl.dll
- ISSetup.dll
- ISSQLSrv.dll
- IsWebDeploy.dll
- ISWindowsFeaturesAction.dll
- ISWindowsFeaturesAction64.dll
- ISXmlCfg.dll
- Layout.bin
- PowerShellWrap.dll
- PrqLaunch.dll
- QuickPatchHelper.dll
- SerialNumCAHelper.dll
- SetAllUsers.dll
- Setup.exe
- Setup.ini
- setup.inx
- setup.isn
- setup.ocx
- setup.skin
- Setup_UI.dll
- setupPreReq.exe
- SetupSuite.exe
- SetupSuite64.exe
- SFHelper.dll
- SQLRT.dll
- SuiteAppxHelper.exe
- SuiteSQLRT.dll
- XMLRT.dll
- 次のフォルダーのサブフォルダーにインストールされるイメージ ファイル

InstallShield Program Files Folder¥Support¥Themes

- 次のフォルダー、およびこのフォルダーの scale-150 および scale-200 サブフォルダーにインストールされるイメージとアイコン ファイル

  InstallShield Program Files フォルダー¥Redist¥Language Independent¥OS Independent

再配布可能ファイルギャラリーを管理する

多くの再配布可能ファイルはサイズが大きいため、プロジェクトで利用可能なものでも InstallShield のインストールと同時にコンピューターへ追加されない場合があります。その場合も、これらの再配布可能ファイルはインターネットからコンピューターへダウンロードすることができます。

アイコンの種類から再配布可能ファイルのステータスを判断することができます。以下は、再配布可能ファイルビュー内 (または、InstallScript プロジェクトの場合、[オブジェクト] ビューまたは [前提条件] ビュー内) の利用可能なアイコンのリストとそれぞれの説明です。

テーブル 12 ・ 再配布可能ファイルのアイコン

アイコン	プロジェクトの種類	説明
	基本の MSI、InstallScript、InstallScript MSI	この InstallShield 前提条件はコンピューターにインストールされます。
	基本の MSI、InstallScript、InstallScript MSI	この InstallShield 前提条件ははコンピューターにインストールされず、ダウンロードして利用することができます。
	基本の MSI、InstallScript、InstallScript MSI	この InstallShield 前提条件はプロジェクトに含まれていますが、その場所は [オプション] ダイアログ ボックスの [前提条件] タブ で指定されたディレクトリの 1 つとしてリストされません。
	基本の MSI、InstallScript、InstallScript MSI	このマージ モジュールはコンピューターにインストールされます。
	基本の MSI、InstallScript MSI	このマージンジュールはリポジトリに格納され、プロジェクトに含めることが可能です。リポジトリに関する詳細については、「リポジトリを使用してプロジェクトの要素を共有する」を参照してください。
	基本の MSI、InstallScript、InstallScript MSI	このマージ モジュールはコンピューターにインストールされず、ダウンロードして利用することができます。
	基本の MSI、InstallScript MSI	ご利用のコンピューターにはこのマージ モジュールの古いバージョンがインストールされています。最新版のダウンロードが可能です。
	基本の MSI、InstallScript、InstallScript MSI	このオブジェクトはコンピューターにインストールされます。

テーブル 12 - 再配布可能ファイルのアイコン (続き.)

アイコン	プロジェクトの種類	説明
	基本の MSI、InstallScript MSI	このオブジェクトはコンピューターにインストールされず、ダウンロードして利用することができます。
	基本の MSI、InstallScript MSI	ご利用のコンピューターにはこのオブジェクトの古いバージョンがインストールされています。最新版のダウンロードが可能です。



プロジェクト: *利用中のコンピューターにインストールされていない再配布可能ファイルをプロジェクトに追加する場合、リリースをビルドしたときに 1 つまたは複数のビルドエラーが発生します。ビルドエラーを避けるには、プロジェクトから再配布可能ファイルを削除するか、リリースを再ビルドする前にダウンロードします。再配布可能ファイルがコンピューターにインストールされていない場合、その再配布可能ファイルの [場所] 列で* ***[ダウンロードが必要]*** *が指定されます。*

InstallScript プロジェクトに追加する [オブジェクト] ビューの再配布可能ファイルがコンピューターにインストールされていない場合、InstallShield は追加を許可しません。

ライブ再配布可能ファイル ギャラリーについて

基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの [再配布可能ファイル] ビューで表示される再配布可能ファイル ギャラリーには、インストールに含むことができるマージ モジュール、オブジェクト、および InstallShield 前提条件が含まれています。また、InstallScript プロジェクトの [前提条件] ビューには、プロジェクトに追加可能な InstallShield 前提条件のギャラリーが表示されます。

InstallShield 前提条件

InstallShield では、多数の InstallShield 前提条件が提供されています。さらに InstallShield のInstallShield 前提条件エディターでは、これらの前提条件を変更したり、独自の前提条件を作成したりできます。すべての InstallShield 前提条件ファイル (.prq) は、次の場所に格納されています:

InstallShield Program Files フォルダー¥SetupPrerequisites

マージ モジュール

マージ モジュールは、さまざまなソースから利用することができます。InstallShield には多くの再配布可能モジュールが含まれていますが、新しいバージョンがある場合や、他のソフトウェア開発会社から必要なモジュールがリリースされている場合があります。さらに InstallShield では、独自のマージ モジュールを作成して再配布可能ファイルギャラリーに追加することも可能です。

[再配布可能ファイル] ビューに一覧表示されているマージ モジュール ファイルのソースは、[オプション] ダイアログ ボックスの [マージ モジュール] タブで指定されたフォルダーです。[オプション] ダイアログ ボックスにアクセスするには、[ツール] メニューから [オプション] を選択します。

次のディレクトリは InstallShield に付属されているモジュールのデフォルトの場所です。

InstallShield Program Files フォルダー¥Modules¥i386

オブジェクト

InstallShield　は多くの再配布可能オブジェクトを提供すると同時に、インストールに含める再配布可能ファイルを独自に作成することも可能です。さらに、他の開発者が InstallShield を使って作成したオブジェクトをプロジェクトに追加することも考えられます。

InstallShield に付属されているオブジェクトのデフォルトの場所は以下の通りです。

InstallShield Program Files フォルダー¥Objects

上記の場所に含まれるオブジェクトは 再配布可能ファイルビューに一覧表示されています。

再配布可能ファイル ギャラリーの管理についての詳細は、次セクションを参照してください:

- 再配布可能ファイルをコンピューターにダウンロードする
- 再配布可能ファイルギャラリーに InstallShield 前提条件を追加する
- 再配布可能ファイルギャラリーから InstallShield 前提条件を削除する
- マージ モジュールを参照する
- マージ モジュールを参照した場合に起きること
- 再配布可能ファイル ギャラリーにマージ モジュールを追加する
- 再配布可能ファイルギャラリーからマージ モジュールを削除する
- InstallScript プロジェクトでオブジェクトを登録する

再配布可能ファイルをコンピューターにダウンロードする

再配布可能ファイルをコンピューターへダウンロードする手順は、使用するプロジェクトの種類によって異なります。

基本の MSI プロジェクトと InstallScript MSI プロジェクトの場合

[再配布可能ファイル] ビューを使用して、最新の InstallShield 前提条件、マージ モジュールとオブジェクトを Revenera Web サイトからコンピューターへダウンロードすることができます。再配布可能ファイルがコンピューターにインストールされていない場合、その再配布可能ファイルの [場所] 列で **[ダウンロードが必要]** が指定されます。



タスク

特定の InstallShield 前提条件、マージ モジュール、オブジェクトをダウンロードするには、以下の手順を実行します:

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. ダウンロードする InstallShield 前提条件、マージ モジュール、または、オブジェクトを右クリックして、**[選択したアイテムをダウンロード]** をクリックします。

タスク ***インストール プロジェクトに必要なすべての InstallShield 前提条件、マージ モジュール、およびオブジェクトをダウンロードするには、以下の手順を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. InstallShield 前提条件、マージ モジュール、、またはオブジェクトのどれかを右クリックして **[必要なアイテムをすべてダウンロード]** をクリックします。

InstallScript プロジェクトの場合

[再配布可能ファイル] ビューを使って、最新の InstallShield 前提条件を Revenera Web サイトからコンピューターへダウンロードすることができます。再配布可能ファイルがコンピューターにインストールされていない場合、その再配布可能ファイルの [場所] 列で **[ダウンロードが必要]** が指定されます。

タスク ***特定の InstallShield 前提条件をダウンロードするには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. ダウンロードする InstallShield 前提条件を右クリックしてから、**[選択したアイテムをダウンロード]** をクリックします。

タスク ***インストール プロジェクトに必要なすべての InstallShield 前提条件をダウンロードするには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. InstallShield 前提条件の 1 つを右クリックしてから、**[必要なアイテムをすべてダウンロード]** をクリックします。

[オブジェクト] ビューを利用して、最新のマージ モジュールとオブジェクトを Revenera Web サイトからコンピューターへダウンロードすることができます。マージ モジュールがコンピューターにインストールされていない場合、インストールされていないことがそのアイコン（  ）によって表示されます。InstallScript プロジェクトに追加するマージ モジュールがコンピューターにインストールされていない場合、InstallShield は追加を許可しません。オブジェクトがコンピューターにインストールされていない場合、[オブジェクト] ビューには表示されません。

タスク ***特定のマージ モジュールをダウンロードするには、以下の手順を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[オブジェクト]** をクリックします。
2. **[InstallShield オブジェクト/マージ モジュール]** ペインで、ダウンロードするマージ モジュールを右クリックして **[選択したアイテムのダウンロード]** をクリックします。

タスク ***オブジェクトをダウンロードするには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[オブジェクト]** をクリックします。
2. **[InstallShield オブジェクト/マージ モジュール]** ペインで、オブジェクトを右クリックし **[Web をチェック]** をクリックします。Revenera Web サイトの [ダウンロード] ページがインターネット ブラウザーで開きます。
3. ダウンロード/インストールするオブジェクトを選択します。インストール中に、InstallShield を閉じてオブジェクトのインストールを完了するようプロンプトが表示されます。

再配布可能ファイルギャラリーに InstallShield 前提条件を追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

タスク ***再配布可能ファイル ギャラリーへ InstallShield 前提条件を追加するには、以下の手順を実行します。***

1. 新規または更新された InstallShield 前提条件 (.prq) ファイルを取得します。
2. Windows Explorer を利用して、新しい前提条件を次の場所にコピーします。

   InstallShield Program Files フォルダー¥SetupPrerequisites
3. 現在 InstallShield が開いている場合は閉じます。
4. InstallShield を起動します。

変更内容が [再配布可能ファイル] ビュー (基本の MSI および InstallScript MSI プロジェクトの場合) および [前提条件] ビュー (InstallScript プロジェクトの場合) に反映されます。

再配布可能ファイルギャラリーから InstallShield 前提条件を削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

タスク ***再配布可能ファイル ギャラリーから InstallShield 前提条件を削除するには、以下の手順を実行します。***

1. InstallShield を閉じます。
2. Windows Explorer を利用してギャラリーから削除する InstallShield 前提条件を探して削除します。InstallShield セットアップ前提条件ファイルは、次のディレクトリに格納されています:

   InstallShield Program Files フォルダー¥SetupPrerequisites
3. InstallShield を起動します。

変更内容が [再配布可能ファイル] ビュー (基本の MSI および InstallScript MSI プロジェクトの場合) または [前提条件] ビュー (InstallScript プロジェクトの場合) に反映されます。

マージ モジュールを参照する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

プロジェクトに追加するマージ モジュールが [再配布可能ファイル] ビューに表示されていない場合、それを検索してプロジェクトおよび [再配布可能ファイル] ビューに追加することができます。

タスク ***マージ モジュールを参照するには、次の操作を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. アイテムを右クリックして、**マージ モジュールの参照**をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. マージ モジュールファイルを参照します。
4. **[OK]** をクリックします。

マージ モジュールを参照した場合に起きること

InstallShield は、マージ モジュールへの参照を明示パスとして維持しません。そのかわり、マージ モジュール GUID とマージ モジュールロケールに基づいてマージ モジュールのキーを生成します。InstallShield がマージ モジュールにアクセスする時、そのキーと一致するファイルを [マージ モジュールの場所] ボックスで指定したフォルダー内で探します。[マージ モジュールの場所] ボックスは、[オプション] ダイアログ ボックスの [マージ モジュール] タブにあります。

マージ モジュールを参照する場合、マージ モジュールを含むフォルダーへのパスがマージ モジュールの場所ボックス内のパスのリストへ追加されます。さらに GUID:ロケールキーが選択されたファイルに基づいてインストール プロジェクトに追加されます。

インストールへの影響

マージ モジュールの場所ボックスにある 2 つのマージ モジュールが GUID:ロケール キーを持つ場合、ファイル名が異なる場合でも 1 つだけがインストールに含まれます。InstallShield による [マージ モジュールの場所] ボックスの検索方法のため、どのマージ モジュールが含まれるかを予期するのは不可能です。

マージ モジュールの場所ボックス内のディレクトリ数の限定

共有マージ モジュールギャラリを使用する場合、ターゲットマシンに存在するバージョンより古いか新しいバージョンのマージ モジュールが存在する可能性があります。このため、マージ モジュールの場所ボックスのディレクトリ数を制限した方が賢明な場合があります。

タスク ***ディレクトリ数を制限するには、次のいずれかを実行します。***

- Windows エクスプローラーを使用して、**[マージ モジュールの場所]** ボックスに既にリストされているフォルダーの 1 つにマージ モジュールをコピーします。
- 検索パスからデフォルトのフォルダーを削除して、共有の場所だけが参照されるようにします。

再配布可能ファイル ギャラリーにマージ モジュールを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

タスク ***再配布可能ファイルギャラリーへマージ モジュールを追加するには、以下の手順に従います:***

1. 新規、またはアップデートされたマージ モジュールを取得します。
2. Windows Explorer を利用して、**[オプション]** ダイアログ ボックスの **[マージ モジュール]** タブで指定されたフォルダーの 1 つに新しいモジュールをコピーします。

   InstallShield に付属されているモジュールのデフォルトの場所は以下の通りです:

   InstallShield Program Files フォルダー¥Modules¥i386
3. 現在 InstallShield が開いている場合は閉じます。
4. InstallShield を起動します。

変更内容が [再配布可能ファイル] ビュー (基本の MSI および InstallScript MSI プロジェクトの場合) および [オブジェクト] ビュー (InstallScript プロジェクトの場合) に反映されます。

再配布可能ファイルギャラリーからマージ モジュールを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

タスク ***再配布可能ファイル ギャラリーからモジュールを削除するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. Windows Explorer を利用してギャラリーから削除するマージ モジュールを探して削除します。**[オプション]** ダイアログ ボックスの[マージ モジュール] タブで指定されたディレクトリをすべて検索して下さい。
3. InstallShield を起動します。

変更内容が [再配布可能ファイル] ビュー (基本の MSI および InstallScript MSI プロジェクトの場合) および [オブジェクト] ビュー (InstallScript プロジェクトの場合) に反映されます。

メモ: *インストールに現在関連付けられているマージ モジュールを削除すると* ***[マージ モジュールが見つかりません]*** *というメッセージが表示され、モジュールをインストールに追加できないことを通知します。*

InstallScript プロジェクトでオブジェクトを登録する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript プロジェクト内からオブジェクトを登録すると、そのオブジェクトが再配布可能ファイル ギャラリに追加され、他のプロジェクトに含めることができるようになります。

タスク ***オブジェクトを登録するには、次の手順を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[オブジェクト]** をクリックします。
2. 項目を右クリックして **[詳細]** をポイントしてから、**[新しいオブジェクトの登録]** をクリックします。**[プロジェクトの設定]** ダイアログ ボックスが開きます。
3. [言語非依存オブジェクトのプロパティ] タブで、メディア ファイル (Data1.hdr ファイル) を指定します。
4. その他希望の設定を指定して **[OK]** をクリックします。

情報は、次のセクションを参照してください:

- InstallShield 前提条件、マージ モジュール、オブジェクトを基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加する
- InstallShield 前提条件、マージ モジュール、およびオブジェクトを InstallScript プロジェクトに追加する

- InstallShield 前提条件をプロジェクトから削除する
- プロジェクトからマージ モジュールとオブジェクトを削除する
- InstallShield 前提条件、マージ モジュールおよびオブジェクトのファイルを決定する

InstallShield 前提条件、マージ モジュール、およびオブジェクトをプロジェクトに組み込む

InstallShield では、InstallShield 前提条件、マージ モジュール、および InstallScript オブジェクトとしてパッケージ化されている多数のサード パーティ再配布可能ファイルが提供されています。これらのビルトイン再配布可能ファイルをインストール プロジェクトに追加することができます。詳しくは、ドキュメントの該当セクションをご覧ください。



ヒント: *InstallShield 前提条件、マージ モジュール、およびオブジェクトを自分自身で作成するときの情報については、「InstallShield 前提条件およびその他の再配布可能ファイルをデザインする」をご覧ください。*

InstallShield 前提条件、マージ モジュール、オブジェクトを基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加する



プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

InstallScript プロジェクトに再配布可能ファイルに追加するときの情報は、「InstallShield 前提条件、マージ モジュール、およびオブジェクトを InstallScript プロジェクトに追加する」を参照してください。

2 つの種類の再配布可能ファイル (マージ モジュールおよびオブジェクト) は、インストールされるために機能への関連付けが必要です。マージ モジュールまたはオブジェクトの関連付けを行うとき、関連付けることができる機能やサブ機能の数に制限はありませんマージ モジュールまたはオブジェクトを追加するときにインストール プロジェクトに機能が存在しない場合は、機能を作成できる [新しい機能の作成] ダイアログ ボックスが開きます。機能を作成しない場合、これら 2 種類の再配布可能ファイルはインストール プロジェクトには追加されません。

InstallShield 前提条件を InstallScript MSI プロジェクトに追加した場合、それらを 1 つまたは複数の機能と関連付けることはできません。

InstallShield 前提条件を基本の MSI プロジェクトに追加した場合、その前提条件はメインのインストールが開始される前に実行されるため、デフォルトでどの機能にも関連付けられていません。これらの前提条件は、*セットアップ前提条件*と呼ばれます。必要に応じて、InstallShield 前提条件を現在プロジェクトに存在する 1 つ以上の機能に関連付けることができます。

タスク

InstallShield 前提条件、マージ モジュールまたはオブジェクトを基本の MSI プロジェクトまたは InstallScript MSI プロジェクトに追加するには、以下の手順に従います:

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. リポジトリへパブリッシュされたマージ モジュールを表示するには、再配布可能ファイルを右クリックして **"リポジトリのマージ モジュールを表示する"** が有効となっていることを確認してください。
3. 追加する再配布可能ファイルの前にあるチェック ボックスを選択します。オブジェクトを選択した場合、関連するウィザードが開いてカスタマイズ処理の手順を案内します。
4. マージ モジュールまたはオブジェクトの場合: **[条件付きインストール]** ペインで、この再配布可能ファイルを含める必要がある各機能のチェック ボックスを選択します。

   基本の MSI プロジェクトで、前提条件を機能と関連付ける場合: **[条件付きインストール]** ペインで、この前提条件を含める必要がある各機能のチェック ボックスを選択します。機能に前提条件を関連付けない場合、**[機能の選択の前にインストールする]** チェック ボックスを選択状態のままにしておきます。このチェック ボックスは、InstallShield 前提条件を基本の MSI プロジェクトに追加したとき、デフォルトで選択されています。

ヒント: *[再配布可能ファイル] ビューの右側のペインに、提供されている再配布可能ファイルの一覧から選択されたマージ モジュール、オブジェクト、または InstallShield 前提条件の詳細が表示されます。この詳細ペインで、再配布可能ファイルがインストールするファイルなどの情報を確認することができます。このビューで [詳細を表示] ボタンをクリックして、詳細の表示/非表示を切り替えることができます。*

メモ: *プロジェクトに追加した InstallShield 前提条件の [場所] 列で **[ダウンロードが必要]**を指定した場合、その InstallShield 前提条件はコンピューターにインストールされません。InstallShield 前提条件をプロジェクトに含める必要がある場合、コンピューターにインターネットからその前提条件をダウンロードすることができます。1 つまたは複数の InstallShield 前提条件をダウンロードしないでリリースをビルドした場合で、さらに InstallShield 前提条件を* Setup.exe *から抽出する、または (Web からエンド ユーザーのコンピューターにダウンロードするのではなく) ソース メディアからコピーするを指定した場合、1 つまたは複数のビルドエラーが生成される可能性があります。ビルドエラーを除去するには、プロジェクトから InstallShield 前提条件を削除してコンピューターにダウンロードするか、またはリリースの InstallShield 前提条件の場所をダウンロード オプションに変更してから、リリースを再ビルドします。*

InstallShield 前提条件とオブジェクトの取得

一部の InstallShield 前提条件とオブジェクトは、InstallShield と共にインストールされませんので注意してください。これらについては、場合により、ダウンロードする必要があります。詳細については、「InstallShield のアップデートを取得する」を参照してください。

また、インストール時に起動するインストール (例、Setup.exe ファイルまたは .msi パッケージ) がある場合、カスタム InstallShield 前提条件を自分で作成して、それを必要に応じてプロジェクトに追加することができます。InstallShield 前提条件を自分で作成する方法については、「InstallShield 前提条件を定義する」をご覧ください。

InstallShield 前提条件、マージ モジュール、およびオブジェクトを InstallScript プロジェクトに追加する



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに再配布可能ファイルを追加する方法については、「InstallShield 前提条件、マージ モジュール、オブジェクトを基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加する」を参照してください。

機能に関連付けられていないマージ モジュールとオブジェクトは、InstallScript インストールの実行時にインストールされません。マージ モジュールまたはオブジェクトの関連付けを行うとき、関連付けることができる機能やサブ機能の数に制限はありませんマージ モジュールまたはオブジェクトを追加するときにインストール プロジェクトに機能が存在しない場合は、機能を作成できる [新しい機能の作成] ダイアログ ボックスが開きます。機能を作成しない場合、これら 2 種類の再配布可能ファイルはインストール プロジェクトには追加されません。

InstallShield 前提条件を InstallScript プロジェクトに追加する場合、それを 1 つまたは複数の機能と関連付けることはできません。



タスク

InstallShield 前提条件を InstallScript プロジェクトに追加するには、以下の手順を実行します:

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. 追加する InstallShield 前提条件の前にあるチェック ボックスを選択します。



ヒント: *[前提条件] ビューの右側のペインに、提供されている再配布可能ファイルの一覧から選択された InstallShield 前提条件の詳細が表示されます。この詳細ペインで、再配布可能ファイルがインストールするファイルなどの情報を確認することができます。このビューで [詳細を表示] ボタンをクリックして、詳細の表示/非表示を切り替えることができます。*



メモ: *プロジェクトに追加した InstallShield 前提条件の [場所] 列で **[ダウンロードが必要]**を指定した場合、その InstallShield 前提条件はコンピューターにインストールされません。InstallShield 前提条件をプロジェクトに含める必要がある場合、コンピューターにインターネットからその前提条件をダウンロードすることができます。1 つまたは複数の InstallShield 前提条件をダウンロードしないでリリースをビルドした場合で、さらに InstallShield 前提条件を (Web からエンド ユーザーのコンピューターにダウンロードするのではなく) メディアに含めることを指定した場合、1 つまたは複数のビルドエラーが生成される可能性があります。ビルドエラーを除去するには、プロジェクトから InstallShield 前提条件を削除してコンピューターにダウンロードするか、またはリリースの InstallShield 前提条件の場所をダウンロード オプションに変更してから、リリースを再ビルドします。*



タスク

マージ モジュールまたはオブジェクトを InstallScript プロジェクトへ追加するには、以下の手順に従います:

1. **[アプリケーション データ]** の下にあるビュー リストで、**[オブジェクト]** をクリックします。
2. **[機能]** ペインで、オブジェクトまたはマージ モジュールを追加する機能を選択します。

3. 追加するオブジェクトまたはマージ モジュールを右クリックして、**[選択した機能に追加する]** を選択します。(オブジェクトまたはマージ モジュールを機能にドラッグアンドドロップすることもできます。)一部のオブジェクトについては、関連したウィザードが表示されるので、これに従ってカスタマイズします。



***メモ・** InstallScript プロジェクトにある機能に追加されたマージ モジュールは、マージ モジュール ホルダー オブジェクトのサブアイテムとして機能ペインに表示されます。このオブジェクトには、Windows Installer エンジンが必要です。詳細については、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。*



***ヒント・** インストールされるファイルや実行されるその他の操作など、オブジェクトについての詳細を表示するにはオブジェクト名またはマージ モジュール名を選択します。右側のペインに情報が表示されます。*



***注意・** 他の企業のマージ モジュールは変更しないでください。*

オブジェクトを取得する

一部のオブジェクトは InstallShield と共にインストールされませんので注意してください。これらについては、場合により、ダウンロードする必要があります。詳細については、「InstallShield のアップデートを取得する」を参照してください。

InstallShield 前提条件をプロジェクトから削除する



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*



タスク** **InstallShield 前提条件をプロジェクトから削除するには、以下の手順に従います:

1. **[アプリケーション データ]** の下のビュー リストで **[再配布可能ファイル]** (基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合)、または **[前提条件]** (InstallScript プロジェクトの場合) をクリックします。
2. 削除する InstallShield 前提条件の前にあるチェック ボックスをクリアします。



***ヒント・** InstallShield 前提条件が別の InstallShield 前提条件の依存関係としてインストールに含まれているが、インストールから依存前提条件を削除したい場合、対応する依存関係を InstallShield 前提条件から削除する必要があります。詳細については、「InstallShield 前提条件の依存関係を削除する」を参照してください。*

プロジェクトからマージ モジュールとオブジェクトを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*



タスク ***プロジェクトからマージ モジュールやオブジェクトを削除するには、以下の手順に従います:***

- 基本の MSI および InstallScript MSI プロジェクトの場合: **[再配布可能ファイル]** ビューで、再配布可能ファイルの前にあるチェック ボックスをクリアします。
- InstallScript プロジェクトの場合: **[オブジェクト]** ビューで、**[機能]** ウィンドウの再配布可能ファイルを右クリックして、**[プロジェクトから削除]** をクリックします。

プロジェクトからマージ モジュールやオブジェクトが移動されます。また、再配布可能ファイルに関連付けられている依存関係もすべて自動的に削除されます。

InstallShield 前提条件、マージ モジュールおよびオブジェクトのファイルを決定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield 前提条件、マージ モジュール、オブジェクト内にあるファイルを確認するには、基本の MSI および InstallScript MSI プロジェクトの [再配布可能ファイル] ビューでそれらを表示できます。このビューの右側のペインに、提供されている再配布可能ファイルの一覧から選択された InstallShield 前提条件、マージ モジュール、またはオブジェクトの詳細が表示されます。この詳細ペインで、再配布可能ファイルがインストールするファイルなどの情報を確認することができます。このビューで [詳細を表示] ボタンをクリックして、詳細の表示/非表示を切り替えることができます。InstallScript プロジェクトの [前提条件] ビューで [詳細の表示] ボタンをクリックすると、選択した InstallShield 前提条件に含まれるファイルが表示されます。

InstallScript プロジェクトで、[オブジェクト] ビューに表示されているマージ モジュールまたはオブジェクトの詳細を表示する場合、そのオブジェクトを選択すると、右のペインに再配布可能ファイル内のファイルの一覧が追加の情報と共に表示されます。



ヒント: *マージ モジュールまたはオブジェクトに含まれるファイルを参照する他の方法は、ナレッジベース記事 Q106474 を参照してください。この記事はダウンロードが可能な [マージ モジュール依存関係] ビューアーへのリンクを含んでいます。*

インストール プロジェクトに含まれている InstallShield 前提条件を利用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield 前提条件 は、製品が必要とする製品、またはテクノロジ フレームワークのためのインストールです。InstallShield で提供されている InstallShield 前提条件の一例として、Java Runtime Environment (JRE) および SQL Server Express Edition があります。任意の既存の InstallShield 前提条件をインストール プロジェクトに追加して、その設定の多くを構成することができます。また、独自の InstallShield 前提条件を作成して、それをプロジェクトに追加することもできます。



プロジェクト: *Windows Installer ベースのプロジェクトに InstallShield 前提条件を含めると、複数のインストールを連鎖させることができるため、1 度に 1 つの実行シーケンスのみしか実行できない Windows Installer 制限をバイパスできます。*Setup.exe *セットアップランチャーは、連鎖を管理するブートストラップ アプリケーションとしての役割を果たします。*



メモ: *Windows Installer 4.5 のチェーンとは異なり、InstallShield 前提条件のインストールは単一のトランザクションとして処理されません。つまり、正常に完了したインストールは、後続の前提条件のインストールが失敗した場合でもロールバックされることはありません。Windows Installer 4.5 のチェーン サポートの詳細については、「トランザクション処理を使って複数パッケージ インストールを構成する」を参照してください。*

[再配布可能ファイル] ビューで、InstallShield 前提条件を基本の MSI および InstallScript MSI プロジェクトに追加することができます。[前提条件] ビューでは、InstallShield 前提条件を InstallScript プロジェクトに追加します。

このセクションには、次のトピックが含まれています:

- セットアップ前提条件と機能前提条件の違い
- 基本の MSI プロジェクトで、InstallShield 前提条件を機能に関連付ける
- 基本の MSI プロジェクトで、機能から InstallShield 前提条件の関連付けを解除する
- InstallShield 前提条件のインストール順を指定する
- 前提条件を含むリリースを構成する
- InstallShield 前提条件を含むディレクトリを指定する
- 特定の InstallShield 前提条件の実行時の場所を指定する
- リリース フラグを InstallShield 前提条件に割り当てる
- InstallShield 前提条件を含むリリースをビルドする
- InstallShield 前提条件を含むインストールの実行時の動作
- インストールに InstallShield 前提条件が含まれていたアプリケーションをアンインストールする

セットアップ前提条件と機能前提条件の違い



プロジェクト: *基本の MSI プロジェクトが、機能前提条件のサポートを含みます。*

次のプロジェクト タイプでは、セットアップ前提条件のさポートが含まれていますが、機能前提条件は含まれていません:

- *InstallScript*
- *InstallScript MSI*

メインのインストールの [ユーザー インターフェイス] シーケンスが開始する前に実行される InstallShield 前提条件は、*セットアップ前提条件*と呼ばれます。セットアップ前提条件は、インストールされている製品のすべての構成に対してインストールする必要がある、または、それ自身のインストールで使用する機能を提供するベースアプリケーションおよびテクノロジ フレームワークに適しています。プロジェクトに InstallShield 前提条件を追加すると、デフォルトでセットアップ前提条件タイプの InstallShield 前提条件になります。

基本の MSI プロジェクトでは、InstallShield 前提条件をメイン インストールに含まれる機能に関連付けることができます。InstallShield 前提条件が 1 つまたは複数の機能と関連付けられている場合、それは *機能前提条件*と呼ばれます。機能の前提条件は、エンドユーザーがインストールする機能を選択したときインストールされます。つまり、マージ モジュール同様、機能前提条件は、それを含む 1 つまたは複数の機能がインストールされたときのみインストールされます。このため、機能前提条件は、インストールされている製品の一部の構成に対してのみインストールする必要がある、および、それ自身のインストールでは使用されないアプリケーションまたはコンポーネントに適しています。

ニーズに最も適した InstallShield 前提条件の種類を判別するときに役に立つ情報は、次のセクションで読むことができます:

セットアップ前提条件の特別考慮

次は、1 つまたは複数のセットアップ前提条件をプロジェクトに含めるとき、次のヒントを参考にしてください:

.NET Framework の要件

製品の実行に、.NET Framework がターゲット システムにインストールされている必要がある場合、.NET Framework 再配布可能ファイルをプロジェクトに含めることができます。.NET Framework がターゲット システムにないとき、インストール中にインストールされます。詳細については、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。

インストールに .NET Framework 再配布可能ファイルと、ターゲット マシンで .NET Framework が必要なセットアップ前提条件 (ファイルを GAC にインストール場合など) が含まれている場合、セットアップ前提条件がインストールされる前に NET Framework がインストールされる必要があることを指定できます。詳細については、「InstallShield 前提条件をインストールするためのパラメーターを指定する」を参照してください。

エンドユーザーがセットアップランチャーの代わりに .msi パッケージを起動したとき、エラーを表示する

セットアップ前提条件が含まれている基本の MSI または InstallScript MSI インストールで、エンドユーザーが、製品の Setup.exe セットアップランチャーを起動する代わりに、.msi パッケージを直接起動した場合、セットアップ前提条件のインストールは実行されません。前提条件がターゲット システムに存在しないとき、製品が適切に動作しないことがあります。これは、.msi パッケージが Setup.exe ファイルにストリームされない非圧縮のリリースをビルドしたとき発生することがあります。

この問題が発生するのを防ぐために、タイプ 19 カスタム アクションを基本の MSI または InstallScript MSI プロジェクトに追加することもできます。このカスタム アクションは、InstallShield 前提条件エディターの [条件] タブで前提条件について構成された条件を使用して評価を行います。カスタム アクションによって、セットアップ前提条件が引き続き必要かどうかが検証され、必要な場合、タイプ 19 エラー カスタム アクションによって、エラー メッセージが表示され、インストールが終了します。

機能前提条件の特別考慮

1 つまたは複数の機能前提条件を基本の MSI プロジェクトに含めるとき、次のヒントを参考にしてください。

Windows Installer の要件

プロジェクトに Windows Installer をインストールする前提条件が含まれている場合、前提条件は機能前提条件ではなくセットアップ前提条件になります。したがって、この前提条件を機能に関連付けることはできません。

.NET Framework の要件

Windows Installer をインストールする前提条件が含まれているプロジェクトで、インストールで .NET Framework が存在している必要がある場合 (ファイルを GAC にインストール場合など)、.NET Framework の前提条件は機能前提条件ではなくセットアップ前提条件である必要があります。したがって、この前提条件を機能に関連付けることはできません。

機能前提条件の再起動に関する潜在的な問題

プロジェクトに InstallShield 前提条件が含まれていて、潜在的に再起動の必要がある場合、この前提条件を機能に関連付けないことをお勧めします。機能前提条件によって再起動がトリガーされた場合、再起動のあと再び ReadyToInstall ダイアログが表示され、エンド ユーザーは [インストール] ボタンをクリックして残りのインストールを続行する必要があります。

必要ディスク容量の計算

Windows Installer がファイルのコスト計算関連のアクションを実行するとき、機能の前提条件で必要なディスク容量は自動的には含まれません。したがって、CustomSetup ダイアログが実行時に表示された場合、異なる機能に表示された空きディスク領域の容量は、機能前提条件に必要なディスク容量が考慮されないため正確ではない場合があります。また、可能性として、メイン インストールに機能前提条件を付加したとき、ターゲット システムのディスク空き容量が足りなくなるということも発生します。このような場合、インストールの途中でターゲット システムの空きディスク容量が足りなくなることがあります。

プロジェクトの機能の前提条件でファイルのコスト計算関係の問題を防ぐために、InstallShield 前提条件エディターの [条件] タブで前提条件について構成された条件を使用して評価を行うカスタム アクションを追加することができます。このカスタム アクションは、関連付けられている機能のインストールが選択されたとき、機能前提条件を実行する必要があるかどうかを判別します。

機能前提条件の実行が必要な場合、カスタム アクションによって、一時的な行が ReserveCost テーブル に追加されます。

基本の MSI プロジェクトで、InstallShield 前提条件を機能に関連付ける



プロジェクト: *基本の MSI プロジェクトは、前提条件と機能の関連付けをサポートします。*

セットアップ前提条件と機能前提条件 (2 つの異なる種類の InstallShield 前提条件) の違いについては、「セットアップ前提条件と機能前提条件の違い」を参照してください。

プロジェクトで、InstallShield 前提条件が機能と関連付けられている場合、それは *機能前提条件*と見なされます。機能に関連付けられていない場合、それは *セットアップ前提条件*と見なされます。

インストール プロジェクトに InstallShield 前提条件を追加したとき、デフォルトで、その前提条件がセットアップ前提条件として追加されます。セットアップ前提条件は、それをプロジェクトに既に存在する 1 つまたは複数の機能に関連付けることで機能前提条件にすることができます。



タスク ***InstallShield 前提条件を機能に関連付けるには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. 再配布可能ファイルの一覧で、機能に関連付ける InstallShield 前提条件を選択します。

   ***メモ・** InstallShield 前提条件のチェック ボックスは、既に選択されている必要があります。選択されたチェック ボックスは、前提条件がプロジェクトに含められることを示します。詳細については、「InstallShield 前提条件、マージ モジュール、オブジェクトを基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加する」を参照してください。*
3. **[条件付インストール]** ペイン内で、この InstallShield 前提条件を追加するすべての機能のチェック ボックスを選択します。

前提条件を新しい機能に関連付ける場合、まずその機能を作成する必要があります。新しい機能の作成方法については、「機能の作成」をご覧ください。

前提条件をプロジェクト内のすべての機能に関連付けたあとで新しい機能を追加した場合、その機能前提条件は新しい機能に自動的に関連付けられません。



***メモ・** 機能前提条件には、セットアップ前提条件には適用されない制限事項があります。詳細については、「セットアップ前提条件と機能前提条件の違い」を参照してください。*

基本の MSI プロジェクトで、機能から InstallShield 前提条件の関連付けを解除する



***プロジェクト・** 基本の MSI プロジェクトは、前提条件と機能の関連付けをサポートします。*

セットアップ前提条件と機能前提条件 (2 つの異なる種類の InstallShield 前提条件) の違いについては、「セットアップ前提条件と機能前提条件の違い」を参照してください。

プロジェクトで、InstallShield 前提条件が機能と関連付けられている場合、それは *機能前提条件*と見なされます。機能に関連付けられていない場合、それは *セットアップ前提条件*と見なされます。



タスク ***機能から InstallShield 前提条件を削除するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. 再配布可能ファイルの一覧で、機能との関連付けを解除する InstallShield 前提条件を選択します。
3. **[条件付きインストール]** ペインで、**[機能の選択の前にインストールする]** チェック ボックスを選択します。このチェック ボックスは、InstallShield 前提条件を基本の MSI プロジェクトに追加したとき、デフォルトで選択されています。



メモ: *セットアップ前提条件には、機能前提条件にはない利点がいくつかあります。詳細については、「セットアップ前提条件と機能前提条件の違い」を参照してください。*

InstallShield 前提条件のインストール順を指定する



プロジェクト: *基本の MSI プロジェクトが、機能前提条件のサポートを含みます。*

次のプロジェクト タイプでは、セットアップ前提条件のさポートが含まれていますが、機能前提条件は含まれていません:

- *InstallScript*
- *InstallScript MSI*

セットアップ前提条件と機能前提条件 (2 つの異なる種類の InstallShield 前提条件) の違いについては、「セットアップ前提条件と機能前提条件の違い」を参照してください。

[再配布可能ファイル] ビューでは、基本の MSI プロジェクトまたは InstallScript MSI プロジェクトに複数の InstallShield 前提条件を含める場合、それらをインストールする順番を指定することができます。[前提条件] ビューでは、InstallScript プロジェクトに複数の InstallShield 前提条件を含める場合、それらをインストールする順番を指定することができます。



タスク ***InstallShield 前提条件がターゲット マシン上にインストールされる順番を指定するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下のビュー リストで **[再配布可能ファイル]** (基本の MSI プロジェクトまたは InstallScript MSI プロジェクトの場合)、または **[前提条件]** (InstallScript プロジェクトの場合) をクリックします。
2. 必要な InstallShield 前提条件をプロジェクトに追加していない場合、それを行います。
3. 任意の再配布可能ファイルを右クリックして、**InstallShield 前提条件の順番を設定**をクリックします。**InstallShield[前提条件のインストール順]** ダイアログ ボックスが開きます。
4. リストから前提条件を選択してから上下矢印をクリックしてインストールの順に並べます。



プロジェクト: *基本の MSI プロジェクトで順番を指定するとき、InstallShield では、セットアップ前提条件と機能前提条件の違いは識別されません。したがって、プロジェクトにセットアップ前提条件と機能前提条件が混在す*

る場合、[InstallShield 前提条件のインストール順] ダイアログ ボックスで、それらがすべて同じ一覧内に表示されます。実行時に、メインのインストールが起動される前、Setup.exe セットアップランチャーはセットアップ前提条件のみ評価し (必要時のみ)、[InstallShield 前提条件のインストール順] ダイアログ ボックスで指定した順番でそれらをインストールします。このあと、インストールの後半で、Windows Installer エンジンによって機能前提条件のみが評価され (必要時のみ)、それらが指定された順番でインストールされます。



ヒント: *InstallShield 前提条件がインストールされる前に Windows Installer エンジンまたは .NET Framework、または両方 をインストールしなくてはならない場合、InstallShield 前提条件エディターで InstallShield前提条件を開いて、この要件を指定することができます。詳細については、「InstallShield 前提条件をインストールするためのパラメーターを指定する」を参照してください。*

前提条件を含むリリースを構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield 前提条件を含む基本の MSI インストールまたは InstallScript MSI インストールをパッケージする場合、以下のいずれか 1 つの方法を使ってエンド ユーザーに対して InstallShield 前提条件ファイルを提供することができます。

- InstallShield 前提条件ファイルを、ソース メディアに格納する。
- InstallShield 前提条件ファイルを Setup.exe に圧縮し、実行時に必要に応じて抽出されるように設定する。
- 必要な場合、インストールはプロジェクトに含まれた InstallShield 前提条件ファイルを、必要に応じて各前提条件の InstallShield 前提条件ファイル (.prq) で指定した URL からダウンロードすることができます。

InstallShield 前提条件を含む InstallScript インストールの場合、利用できる方法は多少異なります:

- リリースの構成に従って、InstallShield 前提条件ファイルを、ソース メディアまたは Setup.exe に格納します。
- インストールはプロジェクトに含まれた InstallShield 前提条件ファイルを、必要に応じて各前提条件の InstallShield 前提条件ファイル (.prq) で指定した URL からダウンロードすることができます。

インストールに含まれる各 InstallShield 前提条件の提供方法を指定することができます。詳細については、「特定の InstallShield 前提条件の実行時の場所を指定する」を参照してください。

リリースに含まれるすべての InstallShield 前提条件を同じ方法で利用可能にする場合、リリース レベルで個別の方法をオーバーライドすることも可能です。詳細については、「リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する」を参照してください。

Windows Installer エンジンおよび .NET Framework のインストールの前または後でインストールされるように InstallShield 前提条件を構成することができます。詳細については、「InstallShield 前提条件をインストールするためのパラメーターを指定する」を参照してください。

InstallShield 前提条件を含むディレクトリを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield前提条件ファイル (.prq) のデフォルトの場所は、以下のとおりです:

InstallShield Program Files フォルダー¥SetupPrerequisites

InstallShield では、ローカル マシンまたはネットワーク上の追加の場所または代替の場所を指定することができます。この柔軟な機能を使って、InstallShield 前提条件をソース コード管理システムに格納し、InstallShield 前提条件の共通のセットを他のチーム メンバーと共有することができます。

InstallShield では、InstallShield 前提条件ファイル (.prq) の検索パスを指定するいくつかの方法があります:

- InstallShield 内部から編集またはビルドを行う場合、[ツール] メニューで [オプション] をクリックすると表示される [オプション] ボックスにある [前提条件] タブを使って、マシン共通および現在のユーザーのフォルダをコンマ区切りのリストで指定できます。

  InstallShield は、[前提条件] タブで指定したパスをマシン上のレジストリに保存します。現在のユーザーに指定されたパスは、以下の場所に保存されます:

  HKEY_CURRENT_USER¥Software¥InstallShield¥*Version*¥Professional¥Project Settings¥PrerequisiteSearchPath

  すべてのユーザーに指定されたパスは、以下の場所に保存されます:

  HKEY_LOCAL_MACHINE¥Software¥InstallShield¥*Version*¥Professional¥PrerequisiteSearchPath

  Standalone Build を使ってリリースをビルドする場合、[オプション] ダイアログ ボックスは使用できません。ただし、必要であれば Standalone Build が搭載されているマシン上のレジストリへのパスを手動で追加することができます。

- ISCmdBld.exe を使って、コマンドラインからビルドする場合は、–prqpath パラメーターを使ってフォルダのコンマ区切りのリストを指定します。

  .ini ファイルを使って ISCmdBld.exe パラメーターを指定する場合、.ini ファイルの [Mode] セクションで新しい PrerequisitePath パラメーターを使用して、フォルダのコンマ区切りのリストを指定できます。

- MSBuild または Team Foundation Server (TFS) を使ってビルドする場合、InstallShield タスクで PrerequisitePath パラメーターを使います。このパラメーターは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPrerequisitePath として露出されます。複数のパスを指定するには、順序指定されたパスの配列を使用します。

ハードコード化されたパスの代わりに、以下の例のようにパスにパス変数を使用することができます:

<ISProductFolder>¥SetupPrerequisites,<ISProjectFolder>¥MyCustomPrerequisites

[再配布可能ファイル] ビューと [前提条件] ビューには、[オプション] ダイアログ ボックスの [前提条件] タブで指定されている様々な検索パスに含まれている .prq ファイルに対応する InstallShield 前提条件の名前がリストされます。同じ .prq ファイルが複数の検索パスに含まれている場合、InstallShield は最初に検出されたインスタンスのみを表示します。InstallShield は、最初に [前提条件] タブでユーザーごとの設定にリストされた各パスをチェックする。次に、マシン共通の設定にリストされている各パスをチェックします。

ビルド時、プロジェクトに 1 つ以上の InstallShield 前提条件が含まれている場合、InstallShield (または Standalone Build) が指定された場所を検索して、必要に応じて適切な InstallShield 前提条件をリリースに含みます。同じ .prq ファイルが複数の検索パスに含まれている場合、InstallShield は最初に検出されたインスタンスのみをビルドに含みます。.prq ファイルの検索は以下の順で行われます:

1. InstallShield (または Standalone Build) が –prqpath コマンドライン パラメーター、PrerequisitePath .ini ファイル パラメーター、または InstallShield タスクの PrerequisitePath パラメーターを通して指定されたパスをチェックする。
2. InstallShield が [前提条件] タブでユーザーごとの設定にリストされた各パスをチェックする。パスは、レジストリの次の場所に保存されます。

   HKEY_CURRENT_USER¥Software¥InstallShield¥*Version*¥Professional¥Project Settings¥PrerequisiteSearchPath

   Standalone Build で [前提条件] タブを使用することはできませんが、レジストリ パスは使用できます。
3. InstallShield が [前提条件] タブのマシン共通の設定にリストされた各パスをチェックする。パスは、レジストリの次の場所に保存されます。

   HKEY_LOCAL_MACHINE¥Software¥InstallShield¥*Version*¥Professional¥PrerequisiteSearchPath

   Standalone Build で [前提条件] タブを使用することはできませんが、レジストリ パスは使用できます。
4. 前述の場所のいずれにもパスが指定されていない場合、InstallShield はデフォルトの場所 (*InstallShield (または Standalone Build) Program Files フォルダー*¥SetupPrerequisites) をチェックします。



ヒント: *Standalone Build を使って InstallShield 前提条件を含むが、1 つ以上の検索パスが指定されていないリリースをビルドすると、Standalone Build はデフォルト パス (<ISProductFolder>¥SetupPrerequisites) で InstallShield 前提条件ファイルを検索します。ただし、1 つ以上の検索パスを指定して、デフォルト パスを明記しなかった場合、Standalone Build はデフォルト パスを検索しません。*

特定の InstallShield 前提条件の実行時の場所を指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield では、プロジェクトの各 InstallShield 前提条件に異なる実行時の場所を指定することができます。



タスク ***インストールの各 InstallShield 前提条件に異なる実行時の場所を指定するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下のビュー リストで **[再配布可能ファイル]** (基本の MSI プロジェクトまたは InstallScript MSI プロジェクトの場合)、または **[前提条件]** (InstallScript プロジェクトの場合) をクリックします。
2. インストールに含める InstallShield 前提条件の 1 つのチェック ボックスを選択します。

3. InstallShield 前提条件を右クリックして、**[プロパティ]** をクリックします。InstallShield**[前提条件のプロパティ]** ダイアログ ボックスが開きます。

4. **[ビルドの場所]** リストで、適切なオプションをクリックします。

指定した場所は、リリース レベルでオーバーライドすることができます。個々の InstallShield 前提条件に選択した値のオーバーライドを禁止するには、リリース レベルにある "InstallShield 前提条件の場所" 設定を [個々の選択に従う] に設定します。詳細については、「リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する」を参照してください。

ヒント: *InstallShield 前提条件が別の前提条件の依存関係としてプロジェクトに追加される場合、前提条件の依存関係の場所は、それを必要とする InstallShield 前提条件の場所設定に従います。*

リリース フラグを InstallShield 前提条件に割り当てる

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

特定のビルドから除外する InstallShield 前提条件にリリース フラグを設定することができます。たとえば、InstallShield 前提条件を必要とする特別なアドオンを含んだ製品の特別なエディションにのみ含める InstallShield 前提条件がある場合、その InstallShield 前提条件にフラグをつけることで、必要な場合にのみそれがビルドに含まれるようにします。

タスク ***インストール プロジェクトに追加した InstallShield 前提条件にリリース フラグを追加するには、次の手順に従います。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。

2. InstallShield 前提条件を右クリックして、**[プロパティ]** をクリックします。InstallShield**[前提条件のプロパティ]** ダイアログ ボックスが開きます。

3. **[リリース フラグ]** ボックスで、文字列を入力します。文字列には、文字または数字を使用できます。複数のフラグを前提条件追加する場合は、コンマを使ってフラグを区切ります。

リリース フラグに基づいて InstallShield 前提条件をフィルターする方法について、さらに詳しい情報は「リリース フラグ」を参照してください。

InstallShield 前提条件を含むリリースをビルドする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

InstallShield は、前提条件を含まないプロジェクトの Setup.exe ファイルをビルドするとき、以下の場所に格納されたベース Setup.exe ファイルからビルドを始めます。

InstallShield Program Files フォルダー¥redist¥Language Independent¥i386

ただし、InstallShield が前提条件を含むプロジェクトの Setup.exe ファイルをビルドするとき、前述のファイルをベースとして利用することはできません。これは、Setup.exe ファイルに前提条件を含むことができないためです。代わりに SetupPrereq.exe と呼ばれる、多少サイズが大きいファイルを利用します。ベース SetupPrereq.exe ファイルはベース Setup.exe ファイルと同じディレクトリに配置されています。異なるベースファイル (Setup.exe および SetupPrereq.exe) が使用されるため、エンドユーザーに配布される最終ビルド Setup.exe ファイルの追加サイズ オーバーヘッドは、実際プロジェクトに前提条件を含む作業を行うインストール作成者にのみ負担が掛かることになります。

InstallShield 前提条件を含むインストールの実行時の動作



プロジェクト: *次のプロジェクト タイプでは、セットアップ前提条件と機能前提条件が両方含まれています:*

- *基本の MSI*

次のプロジェクト タイプでは、セットアップ前提条件のさポートが含まれていますが、機能前提条件は含まれていません:

- *InstallScript*
- *InstallScript MSI*

セットアップ前提条件と機能前提条件 (2 つの異なる種類の InstallShield 前提条件) の違いについては、「セットアップ前提条件と機能前提条件の違い」を参照してください。

InstallShield 前提条件を含むインストールの概要

次のプロシージャは、エンドユーザーがセットアップ前提条件と機能前提条件を含むインストールを起動したときに、一般的に実行時に発生する事柄の説明です。一部の手順は、特定の種類のプロジェクトにのみ適用します。

1. セットアップランチャー (通常、Setup.exe と呼ばれます) は、必要に応じて言語の選択ダイアログを表示します。
2. セットアップランチャーは、必要に応じてセットアップの前提条件ダイアログを表示し、セットアップ前提条件インストールを起動します。
3. インストールは、エンド ユーザーが機能を選択したり、項目を構成したりできるインストール UI を表示します。インストール UI は、進行状況ダイアログを表示します。
4. 基本の MSI インストールでは、適切な場合、セットアップランチャーが機能前提条件のインストールを起動します:
    a. ビルトイン InstallShield カスタム アクション ISInstallPrerequisites (SetupProgress ダイアログと ExecuteAction アクションの間にスケジュールされる) は、インストールすることが選択された機能を Windows Installer プロパティ IsPrerequisiteFeatures のリストと比較します。全く一致しなかった場合、機能前提条件はインストールされません。
    b. ISInstallPrerequisites アクションは、セットアップランチャーを検索および起動し、またインストール中の機能のリストを提供します。セットアップランチャーのパスは、Windows Installer プロパティ SETUPEXEDIR と SETUPEXENAME によって識別されます:

[SETUPEXEDIR]¥[SETUPEXENAME]

ISInstallPrerequisites は、その場所でセットアップランチャーが見つからなかった場合、別の場所を検索します。初回のインストールでは、SourceDir が確認されます。メンテナンス モードでは、インストール ソース パスに関連するパスが確認されます。

ISInstallPrerequisites がセットアップランチャーが見つけられなかった場合、または、複数の .exe ファイルが見つかった場合、エンドユーザーにプロンプトが表示され、セットアップランチャー ファイルを参照するように要求します。エンドユーザーがファイルを識別すると、インストールが続行します。そうでない場合、インストールは終了します。

c. セットアップランチャーは、機能のリストを評価してインストールする機能前提条件を選択し、また適切なインストールを起動します。

5. インストールはエンド ユーザーの選択にしたがって、ターゲット システム上で変更を完了します。

6. インストールは進行状況ダイアログから SetupCompleteSuccess ダイアログに切り替わります。

InstallShield 前提条件を含むインストールのユーザー インターフェイス

ターゲット システムにインストールする必要があるセットアップ前提条件が 1 つ以上ある場合、通常、メイン インストールの実行が開始される前に、セットアップ前提条件ダイアログが表示されます。このセットアップ前提条件によって、ターゲット システムに不足している非表示のセットアップ前提条件がすべて表示されます。エンド ユーザーがこのダイアログで [インストール] ボタンをクリックすると、必要なセットアップ前提条件のインストールが起動されます管理者権限が必要とマークされているセットアップ前提条件が 1 つ以上あり、かつ、インストールがユーザー アカウント制御 (UAC) が有効にされたシステムで実行される場合、このダイアログの [インストール] ボタンが、昇格された権限が必要であることをエンドユーザーに通知するシールド アイコンと共に表示されます。

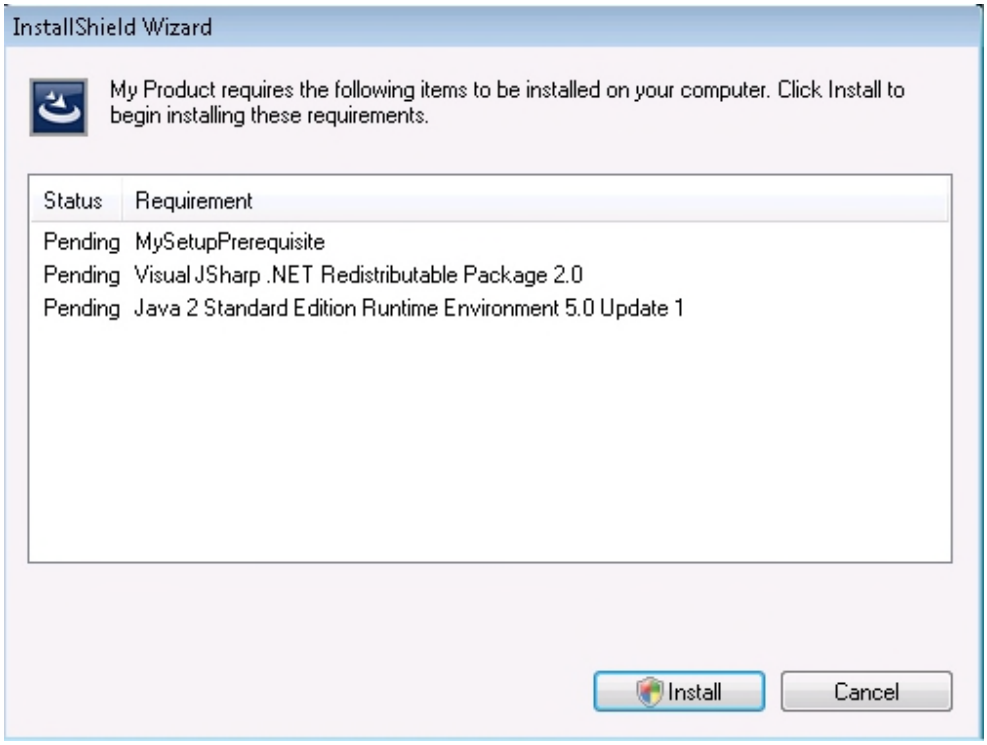


図 2: [インストールするセットアップ前提条件] リストを表示するサンプル [セットアップ前提条件] ダイアログ

セットアップ前提条件が非表示と構成された場合、この前提条件はセットアップ前提条件ダイアログに表示されませんが、インストールはされます。インストールに含まれるすべてのセットアップ前提条件が隠されている場合、セットアップ前提条件ダイアログの代わりに、セットアップランチャーの標準初期ダイアログがインストールで表示されます。

ヒント: セットアップ前提条件ダイアログでセットアップ前提条件の表示と非表示を切り替える方法については、「ターゲットシステムで [インストールされるセットアップ前提条件] リストに、前提条件の名前を含めるかどうかを指定する」をご覧ください。

セットアップ前提条件のインストールが起動するファイルが .msi パッケージで、前提条件が進行状況を表示するようにマークされている場合、前提条件のインストール中、ユーザー インターフェイスで、Windows Installer からのインストールの進行状況メッセージおよびステータス バージョンが表示されます。これは、メイン インストールが基本の MSI インストール、または InstallScript MSI インストールの場合のみ適用します。詳細については、「実行時に、InstallShield 前提条件のインストールの進行状況を表示するかどうかを指定する」を参照してください。

セットアップ前提条件がエンドユーザーによってオプションでインストールされると構成されている場合、エンド ユーザーが前提条件をインストールするかどうかを選択することができるメッセージ ボックスが表示されます。詳細については、「InstallShield 前提条件のインストールの要否をエンド ユーザーが選択できるようにする」を参照してください。

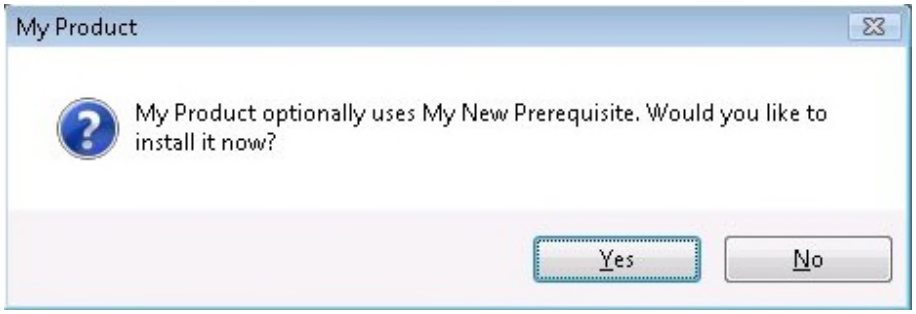


図 3: オプション前提条件のメッセージ ボックス

インストールに機能前提条件が含まれている場合、セットアップランチャーが表示するすべての前提条件のダイアログでそれらが表示されません。ただし、ユーザー インターフェイスでは、必要に応じて進行状況メッセージが表示されます。また、機能前提条件がオプションとマークされている場合、オプションの前提条件メッセージ ボックスが表示されます。

サイレント シナリオ – 基本の MSI インストールで抑制されたユーザー インターフェイスの場合

セットアップ前提条件と機能前提条件は、インストールがサイレントで実行されるときもインストールされます。InstallShield 前提条件は、次のすべてのシナリオでサポートされています:

- **サイレント セットアップランチャーと可視の .msi パッケージ** – セットアップランチャーのユーザー インターフェイスは抑制されますが、.msi パッケージのユーザー インターフェイスは表示されます。たとえば、エンドユーザーは次のコマンドライン ステートメントを使用できます:

  Setup.exe /s

  このシナリオでは、言語の選択ダイアログとセットアップ前提条件ダイアログは表示されません。

- **可視のセットアップランチャーとサイレント .msi パッケージ** – セットアップランチャーのユーザー インターフェイスは表示されますが、.msi パッケージのユーザー インターフェイスは抑制されます。たとえば、エンドユーザーは次のコマンドライン ステートメントを使用できます:

  Setup.exe /v"/qn"

  このシナリオでは、メイン インストールの言語の選択ダイアログとセットアップ前提条件ダイアログは表示されません。ただし、エンドユーザーは、インストールする機能を指定するとき、コマンドラインから ADDLOCAL、ADDSOURCE、ADDDEFAULT、ADVERTISE などの Windows Installer プロパティを設定することができます。

- **サイレント セットアップランチャーとサイレント .msi パッケージ** – セットアップランチャーと .msi パッケージのユーザー インターフェイスは抑制されます。たとえば、エンドユーザーは次のコマンドライン ステートメントを使用できます:

  Setup.exe /s /v"/qn"

  このシナリオでは、すべてのセットアップランチャーと .msi パッケージのダイアログが抑制されます。

メイン インストール内にある .msi パッケージの UI シーケンスがスキップされると、セットアップランチャーは ADDLOCAL、ADDSOURCE、ADDDEFAULT、ADVERTISE などの Windows Installer プロパティを評価して、インストールが必要な機能前提条件があるかどうかを判別し、その結果にしたがって、機能前提条件をインストールします。

サイレント シナリオ – InstallScript および InstallScript MSI インストールで抑制されたユーザー インターフェイスの場合

InstallScript または InstallScript MSI インストールでは、インストールがサイレントで実行される場合でもセットアップ前提条件をインストールできます。つまり、エンド ユーザーが以下のコマンドライン ステートメントを入力してインストールを起動した場合に、InstallShield 前提条件がサポートされます。

Setup.exe /s

これには、応答ファイル (Setup.iss) が必要です。詳細については、「応答ファイルを作成する」を参照してください。

言語の選択ダイアログとセットアップ前提条件ダイアログは表示されません。

UAC のプロンプト

構成方法によって、InstallShield 前提条件を含むインストールがインストール中のいくつかの時点で、Windows Vista 以降のシステム上で昇格された権限のプロンプトを表示することができます:

1. エンド ユーザーが Setup.exe ファイルを起動するとき
2. Setup.exe ファイルが、昇格された権限を必要とするセットアップ前提条件を起動するとき
3. Setup.exe ファイルが、昇格された権限を必要とする機能前提条件を起動するとき
4. Windows Installer が .msi パッケージの [実行] シーケンスを開始するとき

詳細については、「インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する」を参照してください。

InstallShield 前提条件の動作を変更する

インストールの動作を変更したい場合、InstallShield 前提条件エディターを使ってInstallShield 前提条件を変更することができます。たとえば、ユーザーが前提条件のインストールをスキップするかどうか、前提条件には管理者権限が必要かどうか、またターゲットマシンの再起動の要否に応じて前提条件インストールをどのように処理するかを指定することができます。詳細については、「InstallShield 前提条件のインストール動作を指定する」を参照してください。

Windows Installer エンジンおよび .NET Framework のインストールの前または後でインストールされるようにInstallShield 前提条件を構成することができます。詳細については、「InstallShield 前提条件をインストールするためのパラメーターを指定する」を参照してください。

インストールに InstallShield 前提条件が含まれていたアプリケーションをアンインストールする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

インストールがアプリケーションおよび 1 つまたは複数の InstallShield 前提条件から構成される場合があります。エンドユーザーがコントロール パネルのプログラムの追加と削除を使ってアプリケーションをアンインストールした場合、InstallShield 前提条件はマシン上にインストールされたままの状態です。InstallShield 前提条件インストールによって、[プログラムの追加と削除] にエントリが追加された場合、エンドユーザーは [プログラムの追加と削除] を通して InstallShield 前提条件を削除することができます。

インストール プロジェクトに含まれているマージ モジュールを使用する

ドキュメントのこのセクションでは、インストール プロジェクト内からマージ モジュールを使用する方法について説明します。

自分自身のマージ モジュールを作成または変更するときの情報については、「マージ モジュールのデザイン」をご覧ください。

このセクションには、次のトピックが含まれています:

- マージ モジュールを含むディレクトリを指定する
- マージ モジュールのインストール先をオーバーライドする
- マージ モジュールに関するトラブルシューティング
- マージ モジュールをインストール プロジェクト内からリポジトリにパブリッシュする

マージ モジュールを含むディレクトリを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield では、ローカル マシンまたはネットワーク上でマージ モジュール (.msm ファイル) を格納する場所を指定できます。この柔軟な機能を使って、マージ モジュールをソース コード管理システムに格納し、マージ モジュールの共通のセットを他のチーム メンバーと共有することができます。

InstallShield では、マージ モジュールの検索パスを指定するいくつかの方法があります:

- InstallShield 内部から編集またはビルドを行う場合、[ツール] メニューで [オプション] をクリックすると表示される [オプション] ボックスにある [マージ モジュール] タブを使って、マシン共通および現在のユーザーのフォルダをコンマ区切りのリストで指定できます。

  InstallShield は、[マージ モジュール] タブで指定したパスをマシン上のレジストリに保存します。現在のユーザーに指定されたパスは、以下の場所に保存されます:

  HKEY_CURRENT_USER¥Software¥InstallShield¥*Version*¥Professional¥Project Settings¥MMSearchPath

  すべてのユーザーに指定されたパスは、以下の場所に保存されます:

  HKEY_LOCAL_MACHINE¥Software¥InstallShield¥*Version*¥Professional¥MMSearchPath

  Standalone Build を使ってリリースをビルドする場合、[オプション] ダイアログ ボックスは使用できません。ただし、必要であれば Standalone Build が搭載されているマシン上のレジストリへのパスを手動で追加することができます。

- ISCmdBld.exe を使って、コマンドラインからビルドする場合は、–o パラメーターを使ってフォルダのコンマ区切りのリストを指定します。

  .ini ファイルを使って ISCmdBld.exe パラメーターを指定する場合、.ini ファイルの [Mode] セクションで新しい MergeModulePath パラメーターを使用して、フォルダのコンマ区切りのリストを指定できます。

- MSBuild または Team Foundation Server (TFS) を使ってビルドする場合、InstallShield タスクで MergeModulePath パラメーター を使います。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldMergeModulePath として露出されます。複数のパスを指定するには、順序指定されたパスの配列を使用します。

ハードコード化されたパスの代わりに、以下の例のようにパスにパス変数を使用することができます:

<ISProductFolder>¥MergeModules,<ISProjectFolder>¥MyCustomMergeModules

[再配布可能ファイル] ビューと [オブジェクト] ビューには、[オプション] ダイアログ ボックスの [マージ モジュール] タブで指定されている様々な検索パスに含まれているマージ モジュールファイルに対応するマージ モジュールの名前がリストされます。同じマージ モジュールが複数の検索パスに含まれている場合、InstallShield は最初に検出されたインスタンスのみを表示します。InstallShield は、最初に [マージ モジュール] タブでユーザーごとの設定にリストされた各パスをチェックします。次に、マシン共通の設定にリストされている各パスをチェックします。

ビルド時、プロジェクトに 1 つ以上のマージ モジュールが含まれている場合、InstallShield (または Standalone Build) が指定された場所を検索して、必要に応じて適切な マージ モジュールをリリースに含みます。同じマージ モジュールが複数の検索パスに含まれている場合、InstallShield は最初に検出されたインスタンスのみをビルドに含みます。マージ モジュールの検索は以下の順で行われます:

1. InstallShield (または Standalone Build)が –o コマンドライン パラメーター、MergeModulePath .ini ファイル パラメーター、または InstallShield タスクの MergeModulePath パラメーターを通して指定されたパスをチェックする。
2. InstallShield が [マージ モジュール] タブでユーザーごとの設定にリストされた各パスをチェックする。パスは、レジストリの次の場所に保存されます。

   HKEY_CURRENT_USER¥Software¥InstallShield¥*Version*¥Professional¥Project Settings¥MMSearchPath

   Standalone Build で [マージ モジュール] タブを使用することはできませんが、レジストリ パスは使用できます。
3. InstallShield が [マージ モジュール] タブのマシン共通の設定にリストされた各パスをチェックする。パスは、レジストリの次の場所に保存されます。

   HKEY_LOCAL_MACHINE¥Software¥InstallShield¥*Version*¥Professional¥MMSearchPath

   Standalone Build で [マージ モジュール] タブを使用することはできませんが、レジストリ パスは使用できます。
4. 前述の場所のどこにもパスが指定されていない場合、InstallShield は次の順序でデフォルト ディレクトリをチェックします:
   a. *InstallShield (または Standalone Build) Program Files フォルダー*¥System
   b. *InstallShield (または Standalone Build) Program Files フォルダー*¥Modules¥i386
   c. *InstallShield (または Standalone Build) Program Files フォルダー*¥Objects
   d. *InstallShield (または Standalone Build) Program Files フォルダー*¥Modules¥i386¥Japanese
   e. *InstallShield (または Standalone Build) Program Files フォルダー*¥Modules¥i386¥German
   f. *Program Files フォルダー*¥Common Files¥Merge Modules

ヒント: *Standalone Build を使ってマージ モジュールを含むリリースをビルドするときに、ユーザー定義による方法で (つまり、コマンドラインまたは .ini ファイルを通して、あるいは InstallShield タスクを通してレジストリで) 1 つ以上の検索パスを指定しなかった場合 、Standalone Build 前述のデフォルト ディレクトリでマージ モジュールを検索します。ただし、1 つ以上の検索パスを指定して、デフォルト ディレクトリを明記しなかった場合、Standalone Build はデフォルト ディレクトリを検索しません。*

マージ モジュールのインストール先をオーバーライドする

サードパーティ マージ モジュールは変更すべきではありませんが、いくつかのサードパーティ マージ モジュールおよび InstallShield 作成のマージ モジュールのインストール先を上書きすることは可能です。



メモ: *この手順は、マージ モジュールの* TARGETDIR *ディレクトリ、または* TARGETDIR *から直接派生したディレクトリのみを転送します。マージ モジュールが定義済みフォルダー (たとえば、SystemFolder) へファイルを送るように構成されている場合 、モジュールのインストール先をオーバーライドすることはできません。*



タスク ***マージ モジュールのインストール先をオーバーライドするには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. マージ モジュールの横にあるチェック ボックスを選択して、インストールに追加します。
3. モジュールを右クリックして、**プロパティ**をクリックします。[マージ モジュールプロパティ] ダイアログ ボックス が表示されます。
4. **[インストール先]** ボックスで、インストール先を入力するか、定義済みのインストール先から 1 つ選択します。
5. **[OK]** をクリックします。
6. **[条件付きインストール]** ペインで、マージ モジュールを含める機能を選択します (複数可)。

マージ モジュールに関するトラブルシューティング

関連付けられているマージ モジュールの削除

インストールに現在関連付けられているマージ モジュールを削除すると、[マージ モジュールが見つかりません] というメッセージが表示されます。このことにより、そのモジュールをインストールに追加できないことがわかります。

マージ モジュールの言語プロパティの変更

ユーザー定義または別の会社のマージ モジュールが再配布可能ファイルギャラリーに追加された後、その言語プロパティを変更すると、変更後のマージ モジュールをプロジェクトに関連付けようとしたときにビルドエラーが発生します。このエラーが発生するのは、InstallShield がモジュールファイルを識別する際にモジュール識別子の他に言語識別子を使用するためです。

マージ モジュールファイルが再配布可能ファイルギャラリーに追加された後、そのファイルの言語識別子を変更すると、InstallShield はファイルを見つけられなくなります。

正しいマージ モジュール ファイルの場所を特定するには、[再配布可能ファイル] ビューから参照することができます。詳しくは、「マージ モジュールを参照する」をご覧ください。

マージ モジュールをインストール プロジェクト内からリポジトリにパブリッシュする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Windows Installer ベースのインストール プロジェクトで再利用可能な、または他のユーザーと共有可能な既存のマージ モジュールがある場合、それをリポジトリへパブリッシュすることができます。

タスク ***Windows Installer ベースのインストール プロジェクトで作業中に、マージ モジュールをリポジトリへパブリッシュするには、次の手順を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. マージ モジュールを右クリックして、**[パブリッシュ ウィザード]** をクリックします。**パブリッシュ ウィザード**が開きます。
3. パブリッシュ ウィザードのパネルを完成します。

ビルド時にマージ モジュールがインストールへ組み込まれます。リポジトリ マージ モジュールに変更を加えてからリポジトリへ再パブリッシュすると、プロジェクトのインストールが次回再ビルドされたときに、古いバージョンのリポジトリマージ モジュールを持つプロジェクトはすべて更新されます。

Windows Installer 再配布可能ファイルをプロジェクトに追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

Windows Installer は Windows のほとんどのバージョンに組み込まれていますが、Windows Inslaller ベースのインストールは Windows Installer の最新バージョンでのみ動作する特定の機能に依存している可能性があります。また、一部の InstallScript インストールでは、特定のバージョンの Windows Installer が必要な場合もあります。

InstallShield では、Windows Installer の再配布可能ファイルをプロジェクトに含めることができます。Windows Installer 実行可能ファイルの追加方法は、インストールに必要な Windows Installer のバージョンだけでなく、使用中のプロジェクトの種類によって異なります。

Windows Installer の各バージョンごとの最小オペレーティング システムの一覧は、「ターゲット システムの要件」を参照してください。

Windows の各バージョンで一緒にリリースされた Windows Installer のバージョン一覧については、Windows Installer ヘルプ ライブラリの「Released Versions of Windows Installer」をご覧ください。

Windows Installer 5 と Windows Installer 4 は、再配布可能ファイルとして提供されていません。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクト

InstallShield には、Windows Installer 再配布可能ファイルを自己展開型実行可能ファイル (Setup.exe) としてインストールに含めるためのオプションがあります。

Windows Installer の配布

デフォルトで、InstallShield は Setup.exe をセットアップ パッケージと同時に作成します。セットアップランチャーは、インストールで Windows Installer エンジンをインストールする場合、必ず必要です。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに Windows Installer 再配布可能ファイルを含める場合、次のいずれかを実行します:

- **Windows Installer 4.5 の場合**—プロジェクトに 1 つまたは複数の Microsoft Windows Installer 前提条件を追加します。InstallShield は、異なるバージョンの Windows をターゲットとするいくつかのバージョンを含みます。詳細については、「Microsoft Windows Installer の前提条件を含める」を参照してください。
- **Windows Installer 3.1、3.0、または 2.0 の場合**—[リリース] ビューの Setup.exe タブでは、Setup.exe ランチャーを使用するかどうか、これらの Windows Installer 再配布可能ファイルの 1 つのバージョンを含めるかどうか、および含める Windows Installer のバージョンなどの情報を指定することができます。詳細については、「リリースの [Setup.exe] タブ」を参照してください。

  代りに、プロジェクトに 1 つまたは複数の Microsoft Windows Installer 前提条件を追加することともできます。詳細については、「Microsoft Windows Installer の前提条件を含める」を参照してください。



***ヒント・** セットアップランチャーの要件は、リリース ウィザードの [セットアップランチャー] パネルでも指定できます。*

インストール プロセスの概要

実行時、Setup.exe は Windows Installer が既にターゲット システムにインストールされているかどうかを判別します。ターゲット システムに Windows Installer が見つかり、かつ、最小バージョンの要件を満たした場合、インストール パッケージが起動されます。Windows Installer がインストールされていない場合や、または新しいバージョンをインストールする必要がある場合、 Setup.exe は Windows Installer をインストールしてからインストール パッケージを起動します。Windows Installer を更新するためにシステムを再起動する必要があることに注意してください。

InstallScript プロジェクト

場合によって、InstallScript プロジェクトに、Windows Installer 再配布可能ファイルを追加したほうが良い場合があります。たとえば、InstallScript プロジェクトを使って、Windows Installer の特定の最小バージョンが必要な複数の Windows Installer ベースのインストールを連鎖させることがあります。また、マージ モジュールを InstallScript プロジェクトに追加した場合、そのマージ モジュールを [オブジェクト] ビューに マージ モジュール ホルダー– オブジェクトのサブアイテムとして追加する必要があります。このマージ モジュール ホルダー– オブジェクトは Windows Installer エンジンを必要とするため、ターゲット システム上に Windows Installer が存在しない可能性がある場合、Windows Installer 再配布可能ファイルを InstallScript プロジェクトに追加する必要があります。

Windows Installer 再配布可能ファイルを InstallScript プロジェクトに含めるには、1 つ以上の Microsoft Windows Installer 前提条件をプロジェクトに追加します。InstallShield は、異なるバージョンの Windows をターゲットとするいくつかのバージョンを含みます。詳細については、「Microsoft Windows Installer の前提条件を含める」を参照してください。

Microsoft Windows Installer の前提条件を含める



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield には、複数の Windows Installer バージョン用の InstallShield 前提条件が含まれています。[再配布可能ファイル] ビュー (基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合) または [前提条件] ビュー (InstallScript プロジェクトの場合) を使って、プロジェクトにこれらの InstallShield 前提条件を追加できます。

基本の MSI インストールまたは InstallScript MSI インストールの実行中に Windows Installer 4.5 の存在をチェックする

Windows Installer 4.5 がインストールされているかどうかを判別するために、インストールがターゲット システムをチェックする必要があるかもしれません。Windows Installer 4.5 がインストールされていない場合、インストールはエラー メッセージ ボックスを表示して、処理の続行を阻止します。この動作は、以下の状況下で役立ちます:

- インストールが Windows Installer 4.5 の機能に依存している場合。
- 製品のインストール時に、必要に応じて Windows Installer 4.5 をインストールする Setup.exe セットアップランチャーではなく、.msi ファイルが使用される可能性がある場合。

  例えば、インストールを非圧縮リリースとして配布した場合に、エンド ユーザーが Setup.exe ファイルではなく .msi ファイルを直接起動できるとき。さらに、システム管理者が製品を企業環境に配布するためにインストールをカスタマイズする必要がある場合、管理インストール バージョンを作成および配布する可能性があります。

Windows Installer 4.5 がインストールされているかどうかを判別して、インストールされていない場合にエラーメッセージを表示するには、エラー カスタム アクション (タイプ 19 カスタム アクション) を作成します。以下は、その手順の説明です。

タスク

Windows Installer 4.5 がインストールされているかどうかを判別して、インストールされていない場合にエラーメッセージを表示するには、以下の手順に従います:

1. **[カスタム アクションとシーケンス]** ビューで、**[カスタム アクション]** エクスプローラーを右クリックし、**[新しいエラー]** をクリックします。
2. 新しいカスタム アクションを選択します。
3. **"エラー メッセージ"** 設定に、Windows Insatller 4.5 がインストールされていない状態でエンド ユーザーがインストールを起動した場合に表示するエラー メッセージ テキストを入力します。
4. **"インストール UI シーケンス"** と **"インストール 実行シーケンス"** 設定で、エラー メッセージを LaunchConditions アクションの前にスケジュールするオプションを選択します。たとえば、このエラー アクションの一般的なシーケンスは **<最初のアクション>** です。
5. **"インストール UI 条件"** と **"インストール実行条件"** 設定で、次の条件を入力します:

   VersionMsi < "4.05"

ロックされたファイル/再起動の問題

InstallShield 前提条件が Windows Installer 4.5 エンジンをインストールするとき、既存のエンジン ファイルがロックされている場合がよくあります。ターゲット システムを再起動する前に、インストールが更新済みエンジンを必要とする場合は、以下の動作に注意してください。

Windows XP と Windows Server 2003 システムでは、Windows Installer 4.5 エンジン ファイルが直ぐにアップデートされ、再起動の要求が記録されて、メインのインストールが終了するまで (またはその前に起こる再起動が発生するまで) 遅延されます。メイン インストールはアップデート済みの Windows Installer 4.5 エンジンを使用し、インストールの最後で再起動を要求します。

Windows Vista と Windows Server 2008 システムでは、Windows Installer 4.5 エンジンは再起動後までアップデートされないため、即時、再起動が要求されます。エンド ユーザーがマシンの再起動を受け入れた場合、インストールは再起動の後に続行します。ただしエンド ユーザーがマシンの再起動を許可しなかった場合、インストールが終了します。再起動の保留中は、Windows Installer 4.5 エンジンは未インストールの状態となり、続けて前提条件を実行しても失敗して、インストールの続行が不可能となります。マシンの再起動を行うと、Windows Installer 4.5 のインストールが完了し、インストールを予定通りに続行できます。

.NET Framework 再配布可能ファイルをプロジェクトへ追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

製品の実行に、.NET Framework がターゲット システムにインストールされている必要がある場合、.NET Framework 再配布可能ファイルをプロジェクトに追加することができます。.NET Framework がターゲット システムにないとき、インストール中にインストールされます。

プロジェクトに、.NET Framework の言語パックの再配布可能ファイルを含めることもできます。言語パックには、英語以外の言語のための翻訳済みテキスト (エラー メッセージなど) が含まれています。

.NET Framework および .NET Framework 言語パックをプロジェクトに追加する方法は、使用しているプロジェクト タイプ、および製品が必要とする .NET Framework のバージョンによって異なります。



メモ: *一部の .NET Framework バージョンは、以前の .NET Framework バージョンを含みます:*

- *.NET Framework 3.5 は、.NET Framework 3.0 SP1 と .NET Framework 2.0 SP1 を含みます*
- *.NET 3.0 Framework SP1 は .NET Framework 2.0 SP1 を含みます。*
- *.NET 3.0 Framework RTM は .NET Framework 2.0 RTM を含みます。*

基本の MSI プロジェクトおよび InstallScript MSI プロジェクト

基本の MSI プロジェクトまたは InstallScript MSI プロジェクトに .NET サポートを含める場合、適切な Microsoft .NET Framework 前提条件を追加します: 詳細については、「Microsoft .NET Framework および Microsoft .NET Framework 言語パックの前提条件を含める」を参照してください。



メモ: *.NET Framework 4.5 Full、4.5 Web、4.0 Full、4.0 Client、3.5 SP1、3.5、3.0 SP1、3.0、2.0 SP1、または 2.0 (x64、IA64 のみ) 再配布可能ファイルが含まれます。*

ターゲット システムに .NET Framework が既にインストールされているかどうかをテストするには、ビルトインの MsiNetAssemblySupport プロパティを使用することができます。.NET Framework がインストールされている場合、プロパティには、特定の .NET DLL (fusion.dll) のバージョンに設定され、インストールされていない場合、何も設定されません。

InstallScript プロジェクト

InstallScript プロジェクトで .NET Framework 再配布可能ファイルをプロジェクトに追加する方法は 2 種類あります:

- [前提条件] ビューを使って、プロジェクトに 1 つまたは複数の .NET Framework 前提条件を追加します。
- [オブジェクト] ビューを使って、インストールに Microsoft .NET Framework オブジェクトを追加します。このオブジェクトを追加すると、InstallScript プロジェクトに 1 つ以上の言語パックを追加することもできます。詳細については、「Microsoft .NET Framework オブジェクト ウィザード」を参照してください。

特定のバージョンの .NET Framework または言語パックがインストールされているかどうかを判別するには、Is 関数 を使用して、DOTNETFRAMEWORKINSTALLED 定義済み定数を渡します。

InstallShield 前提条件とオブジェクトの取得

一部の InstallShield 前提条件とオブジェクトは、InstallShield と共にインストールされませんので注意してください。これらについては、場合により、ダウンロードする必要があります。詳細については、「InstallShield のアップデートを取得する」を参照してください。

Microsoft .NET Framework および Microsoft .NET Framework 言語パックの前提条件を含める



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield には、一部のバージョンの .NET Framework と .NET Framework 言語パックの InstallShield 前提条件が含まれています。これらのバージョンの .NET Framework と言語パックを再配布する場合、基本の MSI、InstallScript および InstallScript MSI プロジェクトに、これらの InstallShield 前提条件を含めることができます。

以下は、InstallShield 前提条件として提供されている .NET Framework 再配布可能ファイルの一覧です。関連する言語パック 前提条件も、配布されている場合、含まれています。

- Microsoft .NET Framework 4.5(完全パッケージに 1 つ、Web パッケージに 1 つの InstallShield 前提条件が含まれています。サイズは Web パッケージのほうが小さいですが、実行時にターゲット システムでインターネット接続が必要になります。)
- Microsoft .NET Framework 4 Full。これは、.NET Framework 4 をターゲットにするアプリケーションを実行および開発するのに必要な .NET Framework ランタイムと関連ファイルをインストールします。(完全パッケージに 1 つ、Web ダウンロード パッケージに 1 つの InstallShield 前提条件が含まれています。サイズは Web ダウンロード パッケージのほうが小さいですが、実行時にターゲット システムでインターネット接続が必要になります。)
- Microsoft .NET Framework 4 Client。ほとんどのクライアントアプリケーションを実行するために必要な .NET Framework ランタイムおよび関連ファイルをインストールします。(完全パッケージに 1 つ、Web ダウンロー

ド パッケージに 1 つの InstallShield 前提条件が含まれています。サイズは Web ダウンロード パッケージのほうが小さいですが、実行時にターゲット システムでインターネット接続が必要になります。）

- Microsoft .NET Framework 3.5 SP1（完全パッケージに 1 つ、Web ダウンロード パッケージに 1 つの InstallShield 前提条件が含まれています。サイズは Web ダウンロード パッケージのほうが小さいですが、実行時にターゲット システムでインターネット接続が必要になります。）
- Microsoft .NET Framework 3.5（完全パッケージに 1 つ、Web ダウンロード パッケージに 1 つの InstallShield 前提条件が含まれています。サイズは Web ダウンロード パッケージのほうが小さいですが、実行時にターゲット システムでインターネット接続が必要になります。）
- Microsoft .NET Framework 3.0 SP1（これは、実行時にターゲット システムでインターネット接続が必要になる Web ダウンロード パッケージです。）
- Microsoft .NET Framework 3.0
- Microsoft .NET Framework 3.0（x64）
- Microsoft .NET Framework 2.0 SP2
- Microsoft .NET Framework 2.0 SP2（x64）
- Microsoft .NET Framework 2.0 SP2（IA64）
- Microsoft .NET Framework 2.0 SP1（x86）
- Microsoft .NET Framework 2.0 SP1（x64）
- Microsoft .NET Framework 2.0 SP1（IA64）
- Microsoft .NET Framework 2.0（x64）
- Microsoft .NET Framework 2.0（IA64）



ヒント: *.NET Framework 再配布可能ファイルの他のバージョンに関する詳細は、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。*

これらの InstallShield 前提条件のインストールは、サイレント モードで実行されます。したがって、.NET Framework インストールが実行されるときの言語は、問題になりません。

現在、Windows XP を実行している 64 ビット Itanium に Windows Installer 3.x をインストールできないことと、.NET Framework 2.0 以降に Windows Installer 3.x 以降が必要であるということのために、64 ビット .NET Framework 2.0 前提条件は、Windows XP を実行している 64 ビット Itanium にインストールすることはできません。

InstallShield 前提条件のインストールは、特定の HKEY_LOCAL_MACHINE キーの Install または InstallSuccess の値データを確認して、.NET Framework の対応するバージョンが既にターゲット マシンにインストールされているかどうかを確認します。詳細については、InstallShield 前提条件エディターで任意の InstallShield 前提条件を開いて、設定されている条件をご覧ください。

.NET Framework 再配布可能ファイルをプロジェクトへ追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

プロジェクトが、ターゲット システム上に .NET Core のインストールが必要な場合、そのプロジェクトに .NET Core 再配布可能ファイルを追加することができます。ターゲット システムに .NET Core が搭載されていない場合、インストール中に .NET Core もインストールされます。

.NET Core をプロジェクトに追加する方法は、そのプロジェクトの種類およびプロジェクトで必要な .NET Core ランタイムのバージョンによって異なります。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクト

基本の MSI または InstallScript MSI プロジェクトに .NET サポートを含みたい場合、適切な Microsoft .NET Core 前提条件を追加してください。詳細については、「Microsoft .NET Core 前提条件を含める」を参照してください。

InstallShield 前提条件とオブジェクトの取得

一部の InstallShield 前提条件とオブジェクトは、InstallShield と共にインストールされませんので注意してください。これらについては、場合により、ダウンロードする必要があります。詳細については、「InstallShield のアップデートを取得する」を参照してください。

Microsoft .NET Core 前提条件を含める



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

InstallShield には、.NET Core の一部のバージョンの InstallShield 前提条件が含まれています。これらのバージョンの MSI .NET Core を再配布したい場合、基本の MSI MSI、InstallScript MSI プロジェクトにこれらの InstallShield 前提条件を含めることができます。

InstallShield 前提条件として、次の .NET Core 再配布可能ファイルを使用できます:

- Microsoft .NET 8 Desktop Runtime 8.0.6 (x64)
- Microsoft .NET 8 Desktop Runtime 8.0.6 (x86)



ヒント: *.NET Core 再配布可能ファイルのその他のバージョンについての情報は、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。*

InstallShield 前提条件インストールは、特定の HKEY_LOCAL_MACHINE キーの Install または InstallSuccess 値データを確認して、対応するバージョンの .NET Core がターゲット マシン上に既にインストールされているかどうかを判別します。詳細は、InstallShield エディターで InstallShield 前提条件を開いて、設定されている条件を確認することができます。

MySQL Connector ODBC 前提条件を含める



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

MySQL Connector/ODBC 3.51 をインストールする InstallShield 前提条件をインストールに含めることができます。これにより、エンドユーザーよる ODBC を使用する MySQL データベース サーバーへの接続が可能になります。この InstallShield 前提条件をプロジェクトに追加する前に、MySQL Connector/ODBC ドライバーをダウンロードして、システムで InstallShield 前提条件を構成する必要があります。



タスク

MySQL Connector ODBC 3.51 前提条件をシステムに追加して、プロジェクトに追加できるようにするには、以下の手順に従います:

1. Windows エクスプローラーを開いて、InstallShield 前提条件テンプレート フォルダーを参照します。デフォルト保存先は次の場所です:

   C:¥Program Files¥InstallShield¥2022¥SetupPrerequisites¥Templates

2. Templates フォルダーにある MySQL Connector ODBC 3.51.prq ファイルをコピーし、InstallShield 前提条件フォルダーに貼り付けます。デフォルト保存先は次の場所です:

   C:¥Program Files¥InstallShield¥2022¥SetupPrerequisites

3. http://dev.mysql.com/downloads/connector/odbc/3.51.html にアクセスして、MySQL Connector/ODBC 3.51 driver for Windows 用の MSI インストーラーをダウンロードしてください。

4. ファイルを次の場所に保存します。

   InstallShield Program Files フォルダー¥Objects¥MySQL¥Redist

次回 InstallShield を起動したとき、MySQL Connector ODBC 前提条件が、[再配布可能ファイル] ビューに表示されます。

マシン上で MySQL Connector/ODBC 3.51 ドライバーのインストーラーを保存する場所を変更する場合、InstallShield 前提条件エディターで MySQL Connector ODBC 3.51.prq ファイルを開いて、設定を変更します。InstallShield 前提条件エディターを開くには、[ツール] メニューで、[前提条件エディター] をクリックします。詳細については、「InstallShield 前提条件のファイルを指定する」を参照してください。

Oracle 11g Instant Client の InstallShield 前提条件を含める



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

インストールが実行される前に、Oracle 11g Instant Client をインストールする場合、Oracle 11g Instant Client 前提条件をプロジェクトに含めます。この InstallShield 前提条件をプロジェクトに追加する前に、Oracle Instant Client をダウンロードして、システムで .msi パッケージを作成する必要があります。詳細については、「Oracle のインスタンスに接続して SQL スクリプトを実行する」を参照してください。



メモ: *Oracle 11g InstaInt Client がターゲット システム上で必要な時にインストールがこれをダウンロードする場合、Oracle に問い合わせて、それが可能かどうかを確認してください。可能な場合、独自の Web サイトでダウンロードをホストして、InstallShield 前提条件にそのダウンロード パスを追加しなくてはなりません。Revenera では、このインストールのダウンロードを提供していません。*

DirectX 9.0 オブジェクトを含める

DirectX は、最新のグラフィック カードを含む、マルチメディア アプリケーションおよびハードウェアの API ライブラリがサポートとしています。製品の実行に DirectX がターゲット システムにインストールされている必要がある場合、DirectX オブジェクト (基本の MSI と InstallScript MSI プロジェクトの場合、Windows Installer ベースのオブジェクト、InstallScript プロジェクトの場合、InstallScript オブジェクト) をプロジェクトに追加することができます。ターゲット システムに DirectX がない場合、DirectX がインストール時にインストールされます。

インストール後、DirectX ランタイムはアンインストールできません。DirectX はシステム コンポーネントであるため、エンドユーザーが DirectX をアンインストールするには、オペレーティング システムの再インストールが必要になります。



ヒント: *DirectX オブジェクトは InstallShield と共にインストールされないため、ダウンロードする必要があります。基本の MSI プロジェクト または InstallScript MSI プロジェクトに DirectX オブジェクトを含めるには、MSI オブジェクトのダウンロードを入手します。基本の MSI プロジェクト または InstallScript MSI プロジェクトに DirectX オブジェクトを含めるには、MSI オブジェクトのダウンロードを入手します。詳細については、「InstallShield のアップデートを取得する」を参照してください。*

再配布可能ファイル

DirectX オブジェクトは、32 ビットおよび 64 ビット固有のコンポーネントを含む、最新の DirectX 9.0c コアおよびオプションのコンポーネントをすべてインストールします。

基本の MSI プロジェクトと InstallScript MSI プロジェクトに DirectX オブジェクトを含める

DirectX オブジェクトを基本の MSI または InstallScript MSI プロジェクトに含めると、DirectX オブジェクト ウィザードが起動されます。

DirectX オブジェクトは、圧縮、非圧縮の両方のインストールで使用することができます。DirectX オブジェクト ウィザードを利用して、DirectX ファイルを Disk1 フォルダーに含めるか、または .msi ファイルにストリームするかを指定することができます。

- ファイルを Disk1 フォルダーにあるフォルダーに含めるように指定すると、ビルド時に作成中のインストールに対して DirectX フォルダーが作成され、リリースの Disk1 フォルダーに配置されます。DirectX フォルダーは [サポート ファイル] ビュー (基本の MSI プロジェクトの場合)、または [サポート ファイル/ビルボード] ビューの Disk1 領域に表示されます。
- ファイルを Disk1 フォルダーに含めないように指定すると、ファイルはインストールの .msi ファイルに埋め込まれます。これらのファイルは、[サポート ファイル] ビューの [言語非依存] 領域 (基本の MSI プロジェクトの場合) または [サポート ファイル/ビルボード] ビュー (InstallScript MSI プロジェクトの場合) で一覧表示されます。

メモ: *DirectX インストールを起動するカスタム アクションは Windows Vista 以降のシステムで昇格された権限を使って実行できるように、[実行] シーケンスにスケジュールされ、遅延システム コンテキストで実行されます。*

基本の MSI プロジェクトと InstallScript MSI プロジェクトで DirectX オブジェクト ファイルを更新する

ある DirectX ファイルのアップデートを入手して、DirectX オブジェクトに含める場合、それらを適切な InstallShield Program Files サブフォルダーに他の DirectX ファイルと共に配置します。保存先:

InstallShield Program Files フォルダー¥Objects¥DirectX9c¥Redist

一部のファイルが製品で必要なく、インストールに含める必要がない場合、それらをフォルダーから削除することもできます。InstallShield の現在のバージョンがリリースされてからマイクロソフトがリリースしている可能性があるアップデートに関する詳細など、DirectX 再配布可能ファイルの関する詳しい情報は、最新の DirectX SDK または マイクロソフトの MSDNWeb サイト をご覧ください。

ビルド時に、InstallShield がリリースのビルドをするとき、DirectX フォルダー内の再配布可能ファイルはすべて使用されます。

アプリケーションの依存関係を識別する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

あるファイルが、他のファイルの関数に依存してタスクを実行することがよくあります。ただし、インストールプロジェクトにアプリケーションファイルを含める際に、*「依存関係」*と呼ばれる他のファイルに気が付かない場合があります。InstallShieldでは、これらのファイルを見つけて作業を行うスキャン ウィザードが用意されています。スキャナーは、依存関係スキャナー ビューでアクセスできます。

テーブル 13・利用可能なスキャン ウィザード

スキャナー	説明
スタティック スキャン ウィザード	プロジェクトのポータブル実行可能ファイル（例、.exe、.ocx、.com、.tlb、.hlp、および .chm）を確認して、必要な依存関係を検出します。
ダイナミック スキャン ウィザード	実行可能ファイルが実行中にシステムを監視して、実行可能ファイルで必要となる可能性がある .dll または .ocx ファイルを確認します。

これらのスキャナーのどれか 1 つを使って基本の MSI または InstallScript MSI プロジェクトに追加されるファイルは、セットアップベストプラクティス に従って追加されます。

スタティックおよびダイナミック スキャン ウィザードを使用する場合、InstallShield を使ってスタティックまたはダイナミック スキャンを実行するとき必ず自動的に選択または除外されるファイルを指定することができます。詳細については、「依存関係スキャナーでファイルをフィルターする」を参照してください。

スタティック スキャン



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

スタティック スキャン ウィザードを利用して、プロジェクトに既に追加されているファイルをスキャンして必要な依存関係があるかどうかをチェックすることができます。このウィザードはプロジェクトに含まれるすべてのポータブル実行可能ファイル（.exe、.dll、.ocx、.sys、.com、.drv、.scr、および .cpl ファイル）をスキャンして、必要な可能性がある依存関係を検出します。ウィザードで検出された依存関係のリストが表示され、それぞれをプロジェクトに含めるかどうかを指定することができます。

プロジェクトに追加された新規ファイルは、ファイルが依存している同じ機能に追加されるので、インストールが必要なときに確実に実行されます。

[ビルド時に .NET スキャン] 設定によるスタティック スキャンへの影響

基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトで、コンポーネントの "ビルド時に .NET をスキャン" 設定で選択された値は、スタティック スキャン ウィザードがそのコンポーネントに含まれるファイルをスキャンする方法に影響します。

テーブル 14 ・ スタティック スキャン ウィザードがコンポーネントに含まれるファイルをスキャンする方法を判断する

コンポーネントの "ビルド時に .NET をスキャン" 設定の値	説明
なし	スタティック スキャン ウィザードは、コンポーネントの .NET 部分の依存関係もプロパティもスキャンしません。たとえば、コンポーネントに Notepad.exe と .NET アセンブリファイルが含まれている場合、スタティック スキャン ウィザードは Notepad.exe の依存関係を調べますが、.NET アセンブリ ファイルの依存関係とプロパティの*いずれもスキャンしません*。
プロパティのみ	スタティック スキャン ウィザードは、.NET アセンブリを含むコンポーネントのすべてのファイルをスキャンしてプロパティを調べますが、依存関係を調べてコンポーネントの .NET 部分の特定はしません。たとえば、コンポーネントに Notepad.exe と .NET アセンブリファイルが含まれている場合、スタティック スキャン ウィザードは両方のファイルの依存関係を調べますが、依存関係の .NET アセンブリファイルはスキャンしません。
依存関係およびプロパティ	スタティック スキャン ウィザードは、.NET アセンブリを含むコンポーネントのすべてのファイルをスキャンして、依存関係とプロパティを調べます。スキャン中に見つかった依存関係は、[ファイル選択] パネルに表示されます。.NET ファイルにプロパティが検出された場合、InstallShield によってそのプロパティがコンポーネントの [詳細設定] 領域にある [アセンブリ] ノードの下の [.NET アセンブリ] サブノードに追加されます。

ダイナミック スキャン



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

ダイナミック スキャン ウィザードは、実行可能ファイルの実行中にシステムを監視する使い易いツールです。ウィザードでは実行可能ファイルが必要とする可能性のある .dll および .ocx ファイルのリストが表示され、それぞれをプロジェクトに含めるかどうかを指定できます。

スキャンする実行可能ファイルは、既にプロジェクトに含められているものでも、後からウィザードで追加するものでも構いません。

このウィザードに関する詳しい情報は、「ダイナミック スキャン ウィザード」をご覧ください。

64 ビット依存関係のスキャン



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

Windows Vista 以降の 64 ビット バージョン、または Windows Server 2008 以降の 64 ビット バージョンで InstallShield を使用する場合、スタティック スキャン ウィザードおよびダイナミック スキャン ウィザードは、プロジェクトに含まれる 64 ビット ファイルの 64 ビット依存関係をスキャンすることができます。これらのウィザードは、プロジェクトに含まれる 32 ビット ファイルの 32 ビット依存関係のスキャンも行います。

Windows の 32 ビット バージョンで InstallShield を使用する場合、これらのウィザードはプロジェクトに含まれる 32 ビット ファイルの 32 ビット依存関係のみをスキャンできます。プロジェクトに 64 ビット ファイルが含まれている場合、必要に応じてプロジェクトに依存関係を手動で追加することができます。

依存関係スキャナー結果の確認



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

スタティック スキャン ウィザード およびダイナミック スキャン ウィザードを使ってプロジェクトをスキャンして、製品が必要とする可能性のあるその他のファイルを識別することができます。それぞれのウィザード パネルには、プロジェクトに追加する必要がある可能性の高いファイルおよびマージ モジュールのリストが表示されます。スキャン結果をよく確認して、指定された各ファイルやマージ モジュールをプロジェクトに追加する必要があるかどうかを判断してください。

両方のスキャナーは、異なる方法で依存関係を識別します。1 つのキャナーでは識別されなかった依存関係が、もう 1 つのスキャナーによって識別される場合もあります。そのため、スタティック スキャナーおよびダイナミック スキャナーの両方を使って、依存関係の可能性があるファイルのより完全なリストを作成することをお勧めします。一部の状況において、依存関係スキャナーが製品で必要のないファイルまたはマージ モジュールを依存関係として識別する場合もあります。その場合、ウィザードでは識別された各依存関係を含めたり除外したりすることが可能なので、不要なファイルまたはマージ モジュールを除外することができます。さらに、InstallShield を使ってスタティックまたはダイナミック スキャンを行うときは常に自動的に含めたり除外したりするファイルをマシン全体で指定することも可能です。詳細については、「依存関係スキャナーでファイルをフィルターする」を参照してください。

依存関係を識別するときに最も良い結果を得るために、クリーン マシン上で製品とそのインストールを十分にテストすることが推奨されます。製品が予定通りに動作しなかった場合、マシン上で足りない依存関係がないか、またそれをインストールに含むべきかどうかを判断してください。

依存関係スキャナーでファイルをフィルターする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

スタティックおよびダイナミック スキャン ウィザードを実行すると、インストールに追加したくない依存関係ファイルが一覧表示されることがあります。スキャナーを実行するたびにこれらのファイルが追加されないようにするために、Filters.xml を編集できます。このファイルを使うと、スキャナーが無視する、または含めるファイルを指定できます。

Filters.xml は、以下の場所にあります:

InstallShield Program Files フォルダー¥Support

ファイルは編集後もこの場所に残る必要があります。そうでない場合、スキャナーが正しく動作しません。



ヒント: Filters.xml *ファイルを使って、COM 抽出中にどのレジストリ項目を除外するのかを制御することもできます。詳細については、「COM 抽出のレジストリ変更をフィルターする」を参照してください。*

Filters.xml *ファイルを使って、IIS Web サイトをインポートするときに除外する IIS 設定を制御できます。詳細については、「Web サイトおよびその設定を InstallShield プロジェクトにインポートするときに IIS データをフィルターする」を参照してください。*

ファイルを除外する

Filters.xml ファイルの <Exclude> 要素には、スキャナーが除外する各ファイルのサブ要素を追加します。ここにリストされたファイルは、スキャナーによってインストール プロジェクトに追加されることはありません。

デフォルトで <Exclude> 要素には、すべての Windows ベースのマシン上に存在する一般的なシステム ファイルのサブ要素が含まれています。

ファイルを含める

Filters.xml ファイルの <Include> 要素を使うと、Exclude 要素のサブ要素である個別のファイルをオーバーライドすることができます。スキャナーは、<Include> 要素のサブ要素にリストされているファイルをすべてインストール プロジェクトに追加します。これは <Exclude> 要素のサブ要素にリストされているファイルにも適用されます。



メモ: *次の重要なオペレーティング システム ファイルは、<Include> 要素のサブ要素に追加してスキャナーでは認識されません。*

- *kernel32.dll*
- *ntdll.dll*
- *user32.dll*
- *gdi32.dll*
- *advapi32.dll*

- *shell32.dll*
- *ole32.dll*

<Exclude> および <Include> 要素にファイルを指定する

<Exclude> または <Include> 要素にファイルをリストする場合、そのファイルをサブ要素として追加しなくてはなりません。適切にフォーマットされたサブ要素のサンプルは次の通りです。

```
<File name="myfile.dll" path="[SystemFolder]" We="needthis"/>
```

テーブル 15 ・ <File> サブ要素で利用可能な属性

属性	説明
名前	この属性は小文字のみ使用可能です。この属性の値 (例えば、前述の **myfile.dll**) は、含めるまたは除外するファイルの名前を示します。
パス	この属性はオプションです。この属性の値 (例えば、前述の **[SystemFolder]**) は、ファイルのパスを示します。

その他の属性はすべてオプションで、スキャナーが認識することはありません。例えば前述の **We** 属性と、それに対応する **"needthis"** 値のように、特定の項目が含まれるまたは除外される理由を説明する属性を追加したい場合があります。



重要・ *その場合、XML コードが適切に書かれていることを再確認してください。不適切な場合には、すべてのフィルターが失敗します。多くの場合、インターネット エクスプローラーで* Filters.xml *ファイルを開いて、不適切に書かれた XML コードを確認することができます。<Filters>、<Include>、および <Exclude> 要素は縮小および展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。*

<Exclude> または <Include> 要素にサブ要素を追加した場合、それらが誤ってコメントアウトされたセクションに配置されていないことを確認してください。InstallShield は Filters.xml *ファイルのコメントアウト部分を無視します。*

次のサンプル XML コードで、Filters.xml ファイルの形式を説明します。

```
<Filters>
 <Include>
  <!—この要素にファイルを追加する方法
  —>
  <File name="mfc42.dll" We="needthis"/>
 <Include>
 <Exclude>
  <!—この要素にファイルを追加する方法
  —>
  <Registry key="HKEY_CLASSES_ROOT¥Interface¥{00020404-0000-0000-C000-000000000046}"/>
  <File name="12520437.cpx" path="[SystemFolder]" wrp="4.0-10.0" />
  <File name="12520850.cpx" path="[SystemFolder]" wrp="4.0-10.0" />
 </Exclude>
</Filters>
```

COM サーバーの登録



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

ほとんどのアプリケーションでは、正常に機能するために特定の COM サーバーが必要です。COM サーバーがオペレーティング システムで認識されるようにするには、登録が必要です。

従来型の COM を使用すると、COM でーたは、レジストリに直接書き込まれます。レジストリ フリー COM (Reg-Free COM)、従来型の方法に代わる簡単な方法を提供します。Reg-Free COM を使用すると、COM データは、アプリケーション マニフェスト ファイルに書き込まれます。

- 従来型の COM 登録
- レジストリ フリー COM 登録

従来型の COM 登録



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

InstallShield では、ターゲット マシンへ COM サーバー登録するため方法が複数サポートされています。以下の方法は、従来型の方法です。

- ビルド時 (またはデザインタイム) に COM 情報をファイルから抽出し、.msi データベースの COM 関連データベースに書き込む方法。
- COM サーバーの自己登録関数をインストール時に呼び出す方法。

リストされている最初の方法 (COM 情報を抽出して、.msi データベースに書き込む) は、自己登録登録方法よりも好まれています。最初の方法は、.msi データベースの Class、ProgID、および **Registry** テーブルに COM クラス情報を書き込みます。

このセクションには、次のトピックが含まれています:

- COM サーバーが自己登録をサポートするかどうかを判断する
- COM 情報を COM サーバーから抽出する

- COM 抽出のレジストリ変更をフィルターする
- COM サーバーの自己登録

COM サーバーが自己登録をサポートするかどうかを判断する

すべての COM サーバーが自己登録をサポートするとは限りません。通常、インストール内でどのファイルが自己登録かどうかは明らかな場合が多いですが、そうではない場合もあります。自己登録ファイルの扱いは、非自己登録ファイルと異なるため、特定のファイルが自己登録かどうか判断できる必要があります。

通常、InstallShield プロジェクトにファイルを追加する際、ファイルが自己登録かどうか判別できます。InstallShield は、COM サーバー ファイルがプロジェクトに追加されると、それが自己登録ファイルがどうか判断するためにテストを行います。InstallShield が、プロジェクトに追加された COM サーバーが自己登録であると判断した場合、次のような操作が行われます。

- Windows Installer ベースのプロジェクトの場合: ファイルのコンポーネントの "ビルド時に COM 抽出" プロパティは [はい] に設定されます ([オプション] ダイアログ ボックスの [プリファレンス] タブで、[ビルド時に COM 抽出が行われます] が指定されているとき)。
- InstallScript ベースのプロジェクトの場合: ファイルは、自己登録としてマークされます。

COM 情報を COM サーバーから抽出する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

MSI データベースと MSM データベース プロジェクトの場合、COM 情報をビルド時に抽出することは不可能なため、デザイン時に抽出します。

COM サーバーの登録には、.msi データベース内の Windows Installer テーブルを使って必要なレジストリ エントリを作成する方法が推奨されます。この方法を利用すると、Windows Installer はファイルをアドバタイズされたときに登録することが可能で、またインストールが失敗した場合に登録情報を確実にロールバックすることができます。

各コンポーネントはポータブル実行可能ファイルを 1 つしか持たないため、すべての登録情報はコンポーネントのキーファイルとしてマークされた単一のファイルに属すると想定されます。

InstallShield は、COM サーバーから COM 情報を抽出するいくつかの方法をサポートします。

- コンポーネント ウィザードを使用します。このウィザードを利用して、新規 COM サーバーコンポーネントを作成します。この方法を使うと、ウィザードはコンポーネントのキーファイルから COM 情報のスキャンを 1 回だけ行います。

- コンポーネントの"ビルド時に COM 抽出" 設定を [はい] に設定します。リリースをビルドする度に COM データが抽出されるため、この方法は COM サーバーのインターフェイスが頻繁に変更される場合に有効です。この場合、COM サーバーがそのコンポーネントのキーファイルでなくてはなりません。
- [ファイルとフォルダー] ビューにあるファイルを右クリックしてから、[キーファイルの COM データを抽出] をクリックし、情報を抽出します。COM サーバーファイルをコンポーネントへ追加済みの場合は、この方法を使います。

InstallShield が COM データを抽出する方法の代わりに、手動で コンポーネントの詳細設定を使って COM 情報をコンポーネントに追加する こともできます。

リリースをビルドする度に自動的に COM 情報を抽出したくない場合は、コンポーネントの "ビルド時に COM 抽出" 設定で {いいえ] を選択します。InstallShield に、コンポーネントの COM 登録詳細設定に静的に含まれる情報に従って自動的にファイルを登録させるには、この方法を使います。この詳細設定には、コンポーネントウィザードで抽出されたか、または詳細設定に手動で入力された、ファイルの COM クラス、PprogID などの情報が保存されます。

InstallShield が COM サーバーから COM 情報を抽出する時、COM クラス情報は .msi データベースの Class、ProgId および Registry テーブルに書き込まれます。

InstallShield を 64 ビット オペレーティング システムで使用している場合、InstallShield は 64 ビット COM サーバーから COM データを抽出できます。データを正しい場所にインストールするため、コンポーネントを 64 ビットとマークしなくてはなりません。64 ビット サポートに関する詳細は、「64 ビット オペレーティング システムをターゲットにする」を参照してください。

InstallShield では、登録せずに COM サーバー上のすべての COM 情報を見つける特別な技術を採用しています。したがって COM 抽出をしてもシステムレジストリは影響を受けません。ただし一部の COM サーバーの登録プロセスは既存のレジストリ エントリ値によって変わってきます。InstallShield はこのようなシナリオを回避するアルゴリズムを作成しましたが、極端な場合は絶対安全とは言えません。



注意 *・WinRunner など一部のアプリケーションは、COM 抽出エンジンにフック .dll ファイルを挿入します。その結果、COM 抽出が失敗してメッセージ「ISRegSpy は、次のモジュール %1 がこのプロセスにフックされ、そのために ISRegSpy の誤動作が発生していることを検出しました。アプリケーションをシャットダウンして COM 抽出を再起動してください。」が表示されます。このメッセージが表示された場合、ダイアログ ボックスの指示通りアプリケーションをシャットダウンして COM 抽出を再開します。*

自己登録ではない .exe ファイルには、"自己登録" プロパティを選択しないでください。.exe ファイルを自己登録するには、/regserver コマンドを使って .exe ファイルを起動する必要があります。しかし .exe ファイルがコマンドラインスイッチをサポートしない場合、ビルド時の抽出中に .exe が起動されます。



メモ *・COM 関連の Windows Installer テーブルの定義は、COM サーバーが 2 つ以上の機能に配置されるのを防ぎます。1 つのCOM サーバーを 2 つ以上の機能に配置する必要がある場合は、次のオプションのどれかを使用してください。*

- *その COM サーバーが最初の機能の子となるような 2 つ目の機能を作成し、親機能にファイルを配置する。*
- *COM 関連の Windows Installer データベーステーブルを使用せずに、COM サーバーの自己登録を使用する。*

COM 抽出のレジストリ変更をフィルターする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

InstallShield による COM サーバーからの不必要な COM データの抽出 (ビルド時またはデザイン時) を阻止するために、Filters.xml ファイルを編集して特定のレジストリ キーを除外することができます。Filters.xml は、以下の場所にあります:

InstallShield Program Files フォルダー¥Support

ファイルは編集後もこの場所に残る必要があります。そうでない場合、COM 抽出が正しく動作しません。



ヒント: Filters.xml *ファイルを使って、依存関係のスキャン中にどのファイルを除外するまたは含めるのかを制御することもできます。詳細については、「依存関係スキャナーでファイルをフィルターする」を参照してください。*

Filters.xml *ファイルを使って、IIS Web サイトをインポートするときに除外する IIS 設定を制御できます。詳細については、「Web サイトおよびその設定を InstallShield プロジェクトにインポートするときに IIS データをフィルターする」を参照してください。*

COM 抽出からレジストリ キーを除外する

Filters.xml ファイルの <Exclude> 要素には、COM 抽出プロセスが除外する各レジストリ キーのサブ要素を追加します。ここにリストされているすべてのキーは、製品がアンインストールされるのと同時にアンインストールされることはありません。

デフォルトで、<Exclude> 要素は必要とされる一般的なシステム レジストリ キーのサブ要素を含みます。

<Exclude> 要素にレジストリ キーを指定する

<Exclude> 要素にキーをリストする場合、そのキーをレジストリ サブ要素として追加しなくてはなりません。

InprocServer32 レジストリ キー、そのすべての値、およびそのサブキーのすべてに対する変更を阻止する、適切にフォーマットされたレジストリ サブ要素のサンプルは次の通りです。

```
<Registry key="HKEY_LOCAL_MACHINE¥SOFTWARE¥Classes¥CLSID¥{00000231-0000-0010-8000-00AA006D2EA4}¥InprocServer32" />
```

InprocServer32 レジストリ キーのデフォルト値のみに対する変更を阻止する、適切にフォーマットされたレジストリ サブ要素は次の通りです。

```
<Registry key="HKEY_LOCAL_MACHINE¥SOFTWARE¥Classes¥CLSID¥{00000231-0000-0010-8000-00AA006D2EA4}¥InprocServer32" value="" />
```

InprocServer32 レジストリ キーのThreadingModel 値名のみに対する変更を阻止する、適切にフォーマットされたレジストリ サブ要素は次の通りです。

<Registry key="HKEY_LOCAL_MACHINE¥SOFTWARE¥Classes¥CLSID¥{00000231-0000-0010-8000-00AA006D2EA4}¥InprocServer32" value="ThreadingModel"/>

テーブル 16 ・ <Registry>サブ要素で利用可能な属性

属性	説明
キー	この属性は小文字のみ使用可能です。この属性の値 (例えば、前述の HKEY_LOCAL_MACHINE¥SOFTWARE¥Classes¥CLSID¥{00000231-0000-0010-8000-00AA006D2EA4}¥InprocServer32) は、フィルターを行うレジストリ キーの名前を示します。
値	この属性はオプションです。 ・ レジストリ キー全体に対する変更を阻止するには、値属性を含まないでください。 ・ 特定のキーのデフォルト値に対する変更を阻止するには、この属性の値をヌルに設定します。 ・ 特定のキーの値名に対する変更を阻止するには、この属性の値をそのレジストリ値の名前に設定します。

<Registry> サブ要素のその他の属性はオプションで、COM 抽出プロセスでは認識されません。特定の項目が除外される理由を説明するために、属性を追加することもできます。



重要・ *その場合、XML コードが適切に書かれていることを再確認してください。不適切な場合には、すべてのフィルターが失敗します。多くの場合、インターネット エクスプローラーで* Filters.xml *ファイルを開いて、不適切に書かれた XML コードを確認することができます。<Filters>、<Include>、および <Exclude> 要素は縮小および展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。*

<Exclude> または <Include> 要素にサブ要素を追加した場合、それらが誤ってコメントアウトされたセクションに配置されていないことを確認してください。InstallShield は Filters.xml *ファイルのコメントアウト部分を無視します。*

次のサンプル XML コードで、Filters.xml ファイルの形式を説明します。

```
<Filters>
 <Include>
  <!—この要素にファイルを追加する方法
  —>
 <Include>
 <Exclude>
  <!—この要素にファイルを追加する方法
  —>
  <Registry key="HKEY_CLASSES_ROOT¥Interface¥{00020404-0000-0000-C000-000000000046}"/>
  <File name="12520437.cpx" path="[SystemFolder]" wrp="4.0-10.0" />
  <File name="12520850.cpx" path="[SystemFolder]" wrp="4.0-10.0" />
 </Exclude>
</Filters>
```

COM サーバーの自己登録



プロジェクト- この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

自己登録の COM サーバーをお持ちの場合、インストール時に COM サーバーの自己登録関数を呼び出して、ターゲットマシンに COM サーバーを登録することができます。

InstallShield では、COM 関連ファイル (.dll、.ocx、.exe、.tlb、.olb) が自己登録であることを示すための異なるメソッドがサポートされています。

- 特定のファイルが自己登録であることを指定できます。詳細については、「[ファイルのプロパティ] ダイアログ ボックス」を参照してください。
- ダイナミック リンクが自己登録であることを指定できます。詳細については、「[ダイナミック ファイル リンクの設定] ダイアログ ボックス」を参照してください。



ヒント- 自己登録ファイルが 64 ビット コンポーネントの一部の場合、64 ビット自己登録がターゲット マシンで実行されます。詳細については、「64 ビット オペレーティング システムをターゲットにする」を参照してください。

COM サーバーの .dll ファイルまたは .ocx ファイルを自己登録としてマークすると、ファイルの独自の登録関数 (**DllRegisterServer**) がインストール中に呼び出されてターゲット システムに登録され、アンインストール中には登録解除関数 (**DllUnregisterServer**) が呼び出されてファイルが登録解除されます。

DllRegisterServer を指定してファイルを登録すると、いくつかの制限があります。

- **DllRegisterServer** に登録した COM 情報はアドバタイズできない。
- **DllRegisterServer** は .dll コードであるため、インストールに失敗した場合にそれによる変更を正しくロールバックできるという保証がない。
- **DllRegisterServer** は、ユーザーごとまたはマシンごとの COM 情報の違いを判別できません。
- 自己登録時、COM サーバーは、場合によって、特定の順番で登録される必要があります。InstallShield 自己登録 (ISSelfReg) を使用すると、この制限を回避できます。
- 通常、**DllRegisterServer** は相対パスを使用してファイルを登録しない。相対パスはサイドバイサイド共有をサポートしているシステムに必要なものです。

また、パッチで任意の種類の自己登録を使用すると、パッチはアンインストール不可能となります。

1 つまたは複数のファイルを自己登録としてマークすると、InstallShield はデータを .msi データベースのテーブルにデータを追加し、自己登録に関連したカスタム アクションを [インストール]–[実行] シーケンスに追加します。詳細については、「自己登録メソッド」と「InstallShield 自己登録 (ISSelfReg)」を参照してください。

マシンごとの自己登録情報はレジストリキー HKLM¥Software¥Classes¥CLSID に格納され、ユーザーごとのデータは HKCU¥Software¥Classes¥CLSID に格納されます。マージされたデータのビューは、HKCR¥CLSID の下に表示されます。

自己登録メソッド



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

[オプション] ダイアログの [プリファレンス] タブで、InstallShield 自己登録メソッド (ISSelfReg) または Windows Installer 自己登録メソッド (SelfReg) のどちらを使用するかを選択できます。



メモ: *パッチで任意の種類の自己登録を使用すると、パッチはアンインストール不可能となります。*

Windows Installer 自己登録

Windows Installer 自己登録は次の特徴があります。

- .dll ファイルと .ocx ファイルのみを登録する。
- ファイルの登録順は任意である。
- 自己登録ファイルが 64 ビット コンポーネントの一部である場合、64 ビット ファイルを登録できる。

InstallShield 自己登録

InstallShield 自己登録は次の特徴があります。

- .exe ファイル、.tlb ファイル、.dll ファイルおよび .ocx ファイルを登録できる。
- バッチモード登録を使用してランタイムのすべての登録依存を処理する。
- オーバーヘッドで70 KB の自己登録エンジンを含める必要がある。
- 自己登録ファイルが 64 ビット コンポーネントの一部である場合、64 ビット ファイルを登録できる。



メモ: *InstallShield 自己登録の場合、実行時に登録するファイルの順番を指定できます。順番は、ダイレクト エディター の ISSelfReg テーブル(Order 列)で設定できます。*

InstallShield 自己登録 (ISSelfReg)



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

COM サーバーの自己登録方法として ISSelfReg を選択した場合で、プロジェクトに自己登録としてマークされているファイル (またはダイナミックリンク) が含まれている場合、InstallShield はこれらのファイルについての情報を自動的に .msi データベースの ISSelfReg テーブルへ追加します。ISSelfReg テーブルは、ダイレクト エディターで参照および編集できます。このテーブルのフィールドについての詳細は、「ISSelfReg テーブル」をご覧ください。

さらに、プロジェクトに自己登録指定のファイル (またはダイナミックリンク) が含まれている場合、InstallShield は次のカスタム アクションを [インストール]–[実行] シーケンスに追加します。

テーブル 17 ・ 自己登録 COM サーバーがあるプロジェクトに追加されたカスタム アクション

アクション	説明
ISSelfRegisterCosting	**ISSelfReg** テーブルを読み取って、登録または登録解除するファイルを決定する、[即時実行] アクション。ファイルはそのコンポーネントがインストールされる予定に合わせて登録され、アンインストールされる予定に合わせて登録解除されます。
ISUnSelfRegisterFiles	コンポーネントの削除が予定されている各ファイルを登録解除する [遅延実行] カスタム アクション。
ISSelfRegisterFiles	コンポーネントのインストールが予定されている各ファイルを登録する [遅延実行] カスタム アクション。
ISSelfRegisterFinalize	自己登録に失敗したファイルのエラー情報を表示します。

レジストリ フリー COM 登録



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

Reg-Free COM を使用すると、COM データは、アプリケーション フォルダーに格納されているアプリケーション マニフェスト ファイルに書き込まれます。マニフェスト ファイルは、アプリケーションおよびそれに関連付けられているライブラリに関する情報を含む XML ファイルです。Reg-Free COM マニフェスト ファイル、実行可能ファイル、COM ライブラリはすべて、ターゲット マシン上の同一のフォルダーにインストールする必要があるので注意してください。

Reg-Free COM の利点

Reg-Free COM には、従来型 COM にはなかったいくつかの利点があります。たとえば、Reg-Free COM を使用すると、コンポーネントは、アプリケーション自体のスコープ内で定義されます。同一の COM コンポーネント (異なるバージョンも含む) を使用する他のアプリケーションがそれの登録を要求してきた場合も、このアプリケーションには干渉しません。

複数のバージョンの共有ライブラリがターゲット システムに存在する場合、従来型の COM 登録に関して問題が発生する可能性があります。たとえば、インストールが共有ライブラリの新しいバージョンを古いバージョン、または、古いバージョンと後方互換性を持たない新しいバージョンで上書きしたりするということが起こりえます。このようなことが発生した場合、特定のバージョンの機能が必要なアプリケーションがクラッシュする可能性があります。このような状況は、一般に *DLL Hell (DELL 地獄)* と呼ばれます。Reg-Free COM を使用すると、他のアプリケーションは作成したアプリケーションの COM コンポーネントにアクセスできないため、これらの問題を回避することができます。

さらに、Reg-Free COM はアップグレードおよびアンインストール処理を簡素化します。アップグレードの場合、単純にアプリケーション フォルダーを置き換えるだけです。アンインストールの場合、単純に同フォルダーを削除します。

Reg-Free COM の制限

Reg-Free COM は、一部のソリューションには適しません。以下のようないくつかの制限があります。

- Reg-Free COM は、Windows XP 以降でのみ動作します。
- コンポーネントは、それがシステム コンポーネントまたはオペレーティング システムの一部であるとき、Reg-Free COM には適しません。また、MDAC (Microsoft Data Access Components) などのデータ アクセス コンポーネントである場合も、不適当です。これらのタイプのコンポーネントは、分離することができません。
- COM コンポーネントは、1 つのアプリケーションにつき一回のみ分離することができます。このような制限の対処策として、単一クラス ライブラリにある複数の COM コンポーネントをグループ化することも考えられます。

このセクションには、次のトピックが含まれています:

- インストールに Reg-Free COM ファイルを作成および変更する
- Ref-Free COM ファイルのサンプル マニフェスト ファイル

インストールに Reg-Free COM ファイルを作成および変更する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

- *マージ モジュール*

Reg-Free COM ウィザードを使用して、インストールに含める Reg-Free COM マニフェスト ファイルを作成および変更することができます。Reg-Free COM ウィザードを使用する前に、COM ライブラリ (.dll および .ocx ファイル) と、それを使用する実行可能ファイルを InstallShield プロジェクトに追加する必要があります。

ウィザードは、COM 情報をアプリケーション マニフェスト ファイルに指定および追加したライブラリから抽出します。また、ウィザードは、新規のコンポーネントを、マニフェスト ファイルがキーファイルとして設定された状態で作成します。マニフェスト コンポーネントは、関連付けられた実行可能ファイル コンポーネントと同じインストール先および条件を持ちます。

Ref-Free COM ファイルのサンプル マニフェスト ファイル



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

マニフェスト ファイルのコンテンツには、実行可能ファイル名、実行可能ファイルに関連付けられたライブラリの名前、および、COM 情報が含まれます。次のマニフェスト ファイルのサンプルでは、実行可能ファイル名は **Sample.exe**、ライブラリ名は **SampleCircle.dll** になっています。

```
<?xml version="1.0" encoding="utf-8"?>
<assembly xsi:schemaLocation="urn:schemas-microsoft-com:asm.v1 assembly.adaptive.xsd" manifestVersion="1.0"
xmlns:asmv1="urn:schemas-microsoft-com:asm.v1" xmlns:asmv2="urn:schemas-microsoft-com:asm.v2" xmlns:dsig="http://www.w3.org/
2000/09/xmldsig#" xmlns="urn:schemas-microsoft-com:asm.v1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <assemblyIdentity type="win32" name="Sample.exe" version="1.0.0.10" />
    <file asmv2:size="24576" name="SampleCircle.dll">
        <hash xmlns="urn:schemas-microsoft-com:asm.v2"> ...</hash>
        <typelib flags="HASDISKIMAGE" helpdir="" resourceid="0" tlbid="{6A2304BC-F200-49B8-955E-0ACDE272B6E0}" version="1.0" /
>
        <comClass clsid="{C621F4D3-8463-4B0C-9BEC-84D979C41385}" description="SampleCircle.Server"
progid="SampleCircle.Server" threadingModel="Apartment" tlbid="{6A2304BC-F200-49B8-955E-0ACDE272B6E0}" />
    </file>
</assembly>
```

Windows XP システム上にマニフェスト ファイルがあると、アプリケーションは、ライブラリがターゲット システムのレジストリに登録されていなくても COM ライブラリからメソッドを使用することができます。Windows XP は、ターゲット システムに登録されている可能性がある他のバージョンのライブラリを使用する前に現在のディレクトリにある COM ライブラリを使用します。

プロジェクトに DIM を追加する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

機能と同じサイズのデベロッパー インストール マニフェスト (DIM) は、インストール パッケージの別個の部分を構成している製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、および要素など、関連するアイテムを集めたものです。DIM プロジェクトによって、リリース エンジニアは、機能別に分かれたインストールの異なる部分を効率的に再利用することができます。

また、DIM を利用することにより、複数の開発者が、インストールの開発を同時に携わることができます。各ソフトウェア開発者は、異なる DIM について個別で作業することができ、リリース エンジニアは、1 つまたは複数のインストール プロジェクトでそれらを参照することができます。

このセクションでは、基本の MSI インストール プロジェクトへの DIM プロジェクトの追加の仕方について説明します。また、1 つ以上の DIM を含む基本の MSI プロジェクトの使用方法についても説明されています。

このセクションには、次のトピックが含まれています:

- インストール プロジェクトに DIM を統合するための適切な方法を判別する
- プロジェクトで DIM ファイルを参照する
- DIM をプロジェクトにインポートする
- インストール プロジェクトで DIM の要素を識別する
- インストール プロジェクトで使用する DIM プロジェクト内のパス変数をオーバーライドする



ヒント: *DIM プロジェクトを作成方法については、「開発プロセスを再利用/分担するためのインストール プロジェクトのモジュール化」をご覧ください。*

インストール プロジェクトに DIM を統合するための適切な方法を判別する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

機能と同じサイズのデベロッパー インストール マニフェスト (DIM) は、インストール パッケージの別個の部分を構成している製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、および要素など、関連するアイテムを集めたものです。DIM を作成したあと、2 つある方法のいずれかで基本の MSI プロジェクトに追加することができます:

- **参照** – DIM プロジェクトの参照を基本の MSI プロジェクトに追加することができます。この方法を使用すると、DIM の要素は、ビルド時に基本の MSI プロジェクトにマージされます。基本の MSI インストールがビルドされるたびに、DIM プロジェクトの最新バージョンが参照され、生成されたインストールに含められます。

  基本の MSI プロジェクトで DIM リファレンスを使用すると、複数のソフトウェア開発者が、インストール開発作業に同時に携わることができます。各ソフトウェア開発者は、異なる DIM について個別で作業することができ、リリース エンジニアは、1 つまたは複数のインストール プロジェクトでそれらを参照することができます。

  DIM プロジェクトのある箇所を変更する場合、まず InstallShield で DIM プロジェクトを開き、必要な変更を行った後、その DIM を参照する任意の基本の MSI プロジェクトを再ビルドします。

  この方法は、最も頻繁に使われる方法です。

  詳細については、「プロジェクトで DIM ファイルを参照する」を参照してください。

- **インポート** – DIM プロジェクトの参照を基本の MSI プロジェクトにインポートすることができます。この方法は、デザイン時に、DIM データを基本の MSI プロジェクトにマージするときに行う目的で一回切り行われ、一度行われると元に戻すことはできません。

  DIM データにさらに変更を行う必要がある場合は、基本の MSI プロジェクトで行います。元の DIM プロジェクトを変更しても、基本の MSI プロジェクト内の対応するデータは変更されます。

  DIM プロジェクトをインポートして、製品またはそのインストール パッケージの問題をトラブルシュートすることを考えているとします。DIM プロジェクトをインポートする時、インポートする前に、基本の MSI プロジェクト (.ism) のバックアップを作成するオプションがあります。このオプションを使うことにより、DIM プロジェクトがインポートされていない元の基本の MSI プロジェクトに復元することができます。

  詳細については、「DIM をプロジェクトにインポートする」を参照してください。

DIM プロジェクトによって、開発者とリリース エンジニアは、機能別に分かれたインストールの異なる部分を効率的に再利用することができます。

プロジェクトで DIM ファイルを参照する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

基本の MSI プロジェクトに、DIM プロジェクトのリファレンスを追加すると、DIM の要素が、ビルド時に、基本の MSI プロジェクトにマージされます。基本の MSI インストールがビルドされるたびに、DIM プロジェクトの最新バージョンが参照され、生成されたインストールに含められます。



タスク

DIM のリファレンスを、基本の MSI プロジェクト内の特定機能に追加するには、以下の手順に従います:

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. **[セットアップのデザイン]** エクスプローラーで、DIM を含める機能を右クリックして、**[DIM リファレンスの追加]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. 参照する DIM プロジェクト ファイル (.dim) を参照し、見つかったファイルを選択します。
4. **[開く]** ボタンをクリックします。

適切な機能に、DIM プロジェクト内の各コンポーネントに対するコンポーネント ノードが追加されます。このセクションには、次のトピックが含まれています:

- DIM リファレンスを機能に関連付ける
- ビルド時に発生する基本の MSI プロジェクトと DIM リファレンス間の競合を解決する
- DIM リファレンスのビルドの手順を表示する
- DIM リファレンスのインストール先のオーバーライド
- DIM リファレンスからのカスタム アクションとダイアログをスケジュールする
- インストール プロジェクト内から参照された DIM プロジェクトを開く



ヒント: *[DIM リファレンス] ビューを使って、基本の MSI プロジェクトにDIM のリファレンスを追加することもできます。このビューでは、[DIM] エクスプローラーを右クリックして、[DIM リファレンスの追加] をクリックします。この方法を通してリファレンスを追加すると、DIM は基本の MSI プロジェクト内にあるどの機能にも関連付けられません。関連付けの設定方法については、「DIM リファレンスを機能に関連付ける」をご覧ください。参照された DIMはそれぞれ、基本の MSI プロジェクト内の機能に関連付けられる必要があります。関連付けられなかった場合、ビルド時にそれを生成する基本の MSI インストールに含められません。*

DIM リファレンスを機能に関連付ける



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM リファレンスを、基本の MSI プロジェクトに含める場合、その DIM リファレンスは最低 1 つの機能に関連付けられている必要があります。関連付けられていなかった場合、ビルド時にそれを生成する基本の MSI インストールに含められません。

DIM リファレンスは、基本の MSI プロジェクトの [DIM リファレンス] ビルドまたは [セットアップのデザイン] ビューで、1 つまたは複数の機能に関連付けることができます。



タスク ***基本の MSI プロジェクトの [DIM リファレンス] ビューを使って、DIM リファレンスを機能に関連付けるには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[DIM リファレンス]** をクリックします。
2. **[DIM]** エクスプローラーで、構成する DIM を選択します。
3. **[機能]** タブをクリックします。
4. 選択した DIM を含める機能のチェック ボックスをすべて選択します。選択した DIM を含めない機能のチェック ボックスをすべてクリアします。



タスク ***基本の MSI プロジェクトの [セットアップのデザイン] ビューを使って、DIM リファレンスを機能に関連付けるには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. **[セットアップのデザイン]** エクスプローラーで、DIM を含める機能を右クリックして、**[DIM リファレンスを関連付ける]** をクリックします。**[DIM リファレンスを関連付ける]** ダイアログ ボックスが開き、基本の MSI プロジェクト内の機能に関連付けられていない DIM リファレンスがすべて表示されます。
3. 選択した機能に関連付ける DIM ファイルをそれぞれ選択して、**[OK]** をクリックします。

ビルド時に発生する基本の MSI プロジェクトと DIM リファレンス間の競合を解決する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM リファレンスを含む基本の MSI リリースをビルドした時、DIM データが生成された基本の MSI リリースに組み込まれます。データの競合が存在する場合、DIM プロジェクトの値か、または基本の MSI プロジェクトの値か、優先されるべきデータが判別される必要があります。

InstallShield では、DIM プロジェクト内の設定と、その DIM が参照されている基本の MSI プロジェクト内の設定間の競合を解決する方法を指定できます。次は、発生する可能性があるいくつかの競合の例です:

- 同一の名前のプロパティが両方のプロジェクトで使用されていて、異なる値が設定されている場合。
- 同一の文字列識別子がある文字列が両方のプロジェクトでエントリされていて、特定の言語に対して異なる値が設定されている場合。
- 同一の名前が指定されているが、異なる値が設定されているパス変数が両方のプロジェクトで使用されている場合。
- 同一名のコンポーネントが両方のプロジェクトで使用されていて、1 つまたは複数の設定で異なる値が設定されている場合。

ほとんど場合、プロパティ、文字列エントリ、パス変数、コンポーネントなどのアイテムを DIM プロジェクトで作成して、基本の MSI プロジェクトでは作成しなかった場合、DIM プロジェクト固有の GUID がアイテム名の一部として自動的に使用されるため、競合は発生しません。基本の MSI プロジェクトで MYPROPERTY という名前のプロパティを作成した場合、このプロパティは MYPROPERTY という名前になります。DIM プロジェクトで MYPROPERTY という名前のプロパティを作成した場合、このプロパティは内部的に MYPROPERTY.DIM_GUID という名前で使用されます。*DIM_GUID* は、DIM プロジェクトの [一般情報] ビューにある "DIM GUID" 設定で定義されている識別子を表しています。例えば、DIM でプロパティ名は、**MYPROPERTY.8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC** のようになります。この特定の例では、ダイレクト エディターを使って、DIM プロジェクトの Property テーブルで、プロパティの内部名を確認することができます。



タスク

基本の MSI プロジェクトと使われている DIM リファレンスの 1 つ間で発生した競合を自動解決するように設定するには、以下の手順に従います:

1. **[編成]** のビュー リストにある **[DIM リファレンス]** をクリックします。
2. **[DIM]** エクスプローラーで、構成する DIM を選択します。
3. **[ビルド オプション]** タブをクリックします。
4. **"競合の解決"** 設定で、適切な値を選択します。選択可能なオプションは以下のとおりです:
   - **ベース プロジェクト値を使用する** – ビルド時に DIM データを基本の MSI インストールに結合した時に競合が発生した場合、DIM プロジェクト内の値の代わりに、基本の MSI プロジェクト内の値が使用されます。
   - **DIM プロジェクト値を使用する** – ビルド時に DIM データを基本の MSI インストールに結合した時に競合が発生した場合、基本の MSI プロジェクト内の値の代わりに、DIM プロジェクト内の値が使用されます。

可能性は低いけれども競合が発生する可能性があるケースとして、DIM を基本の MSI プロジェクトにインポートして、アイテムの 1 つを DIM プロジェクトで変更し、その DIM のリファレンスを基本の MSI プロジェクトに追加した場合が考えられます。このシナリオでは、競合を避けるために "競合の解決" 設定がどのように構成されているかによって、DIM の設定の値または基本の MSI の設定の値が上書きされます。

DIM リファレンスのビルドの手順を表示する



プロジェクト- *この情報は、基本の MSI プロジェクトに適用します。*

DIM リファレンスを基本の MSI プロジェクトに追加する場合、DIM の作成者によって、ビルドの手順が提供されているかどうかを確認することをお勧めします。この手順で提示されているガイダンス、ヒント、コメントなどによって、追加の手続きが必要かどうか判別できることがあります。



タスク ***基本の MSI プロジェクトで DIM リファレンスのビルドの手順を表示するには、以下の手順に従います:***

1. DIM リファレンスを含む基本の MSI プロジェクトを開きます。
2. **[編成]** のビュー リストにある **[DIM リファレンス]** をクリックします。
3. **[DIM]** エクスプローラーで、構成する DIM を選択します。
4. **[手順]** タブをクリックします。

[手順] タブで、作成または編集時に DIM プロジェクトの作成者が入力したコメントが表示されます。

DIM リファレンスのインストール先のオーバーライド



プロジェクト- *この情報は、基本の MSI プロジェクトに適用します。*

DIM プロジェクトを作成した場合、DIM プロジェクト内のファイル用に、ターゲット システム上で、デフォルトのインストール先を指定することができます。この場所は通常、INSTALLDIR プロパティの値か、このフォルダーのサブフォルダーです。DIM リファレンスを基本の MSI プロジェクトに追加した時、DIM プロジェクトの作成者が構成したインストール先を使うか、または、既存の場所を上書きして、別の場所を指定するかを選択できます。



タスク ***基本の MSI プロジェクトで、DIM リファレンスのファイル用に指定されたデフォルトのインストール先を表示して、それを必要に応じて上書きするには、以下の手順に従います:***

1. DIM リファレンスを含む基本の MSI プロジェクトを開きます。
2. **[編成]** のビュー リストにある **[DIM リファレンス]** をクリックします。
3. **[DIM]** エクスプローラーで、構成する DIM を選択します。
4. **[ビルド オプション]** タブをクリックします。**"インストール先"** 設定で、DIM プロジェクトの作成者が構成した値が表示されます。
5. デフォルトの値を上書きするには、適切な値を指定します。

   パスをハードコード化する代わりに、パスの一部としてディレクトリ プロパティを入力することができます。ディレクトリ プロパティを選択するには、この設定で省略記号ボタン (...) をクリックします。ここで、適切なディレクトリをリストから選択するか、定義済みディレクトリ内に新しいディレクトリを作成できます。円記号を使用して、サブディレクトリの下位レベルを **[ProgramFilesFolder]MyApp¥Bin** のように区切ります。

機能およびコンポーネント レベルで、インストール先を上書きすることもできます。詳細については、「デフォルトの製品インストール先フォルダー (INSTALLDIR) の設定」を参照してください。

DIM リファレンスからのカスタム アクションとダイアログをスケジュールする



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM プロジェクトを作成した時、カスタム アクションとダイアログを作成することができます。DIM リファレンスを基本の MSI プロジェクトに追加した時、基本の MSI インストールの実行中、どのタイミングで、その DIM 内のカスタム アクションを実行するかをスケジュールできます。また、基本の MSI インストールの実行中、DIM のダイアログをいつ実行するかも指定できます。



タスク

DIM リファレンスからのカスタム アクションとダイアログを基本の MSI プロジェクトに挿入するには、以下の手順に従います:

1. DIM リファレンスを含む基本の MSI プロジェクトを開きます。
2. **[編成]** のビュー リストにある **[DIM リファレンス]** をクリックします。
3. **[DIM]** エクスプローラーで、構成する DIM を選択します。
4. **[シーケンス]** タブをクリックします。
5. **[シーケンス]** エクスプローラーで、新しいアクションの前にくるアクションまたはダイアログを右クリックして、**[挿入]** をクリックします。**[アクションの挿入]** ダイアログ ボックスが開き、シーケンスに追加することができるアクションおよびダイアログが一覧表示されます。
6. ダイアログ ボックスの上にあるリストから、挿入するアクションの種類を選択します。
7. アクションの一覧があるボックスで、挿入するアクションの種類を選択します。
8. **[OK]** をクリックします。

インストール プロジェクト内から参照された DIM プロジェクトを開く



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

1 つまたは複数の DIM リファレンスを含む基本の MSI プロジェクトを変更中、DIM プロジェクトを編集、またはその詳しい情報を表示する場合、リンクをクリックすることで、そのプロジェクトを開くことができます。DIM プロジェクトは、基本の MSI プロジェクトを変更しているマシンからアクセスできる必要があるので注意してください。



タスク

基本の MSI プロジェクトから参照された DIM プロジェクトを開くには、以下の手順に従います:

1. **[編成]** のビュー リストにある **[DIM リファレンス]** をクリックします。
2. **[DIM]** エクスプローラーで、開く DIM を選択します。

3. **[全般]** タブで、**[DIM プロジェクトを起動]** リンクをクリックします。

InstallShield の新しいインスタンスが起動し、DIM プロジェクトが InstallShield 内で開きます。

DIM をプロジェクトにインポートする



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM プロジェクトを基本の MSI プロジェクトにインポートすると、DIM データが基本の MSI プロジェクトに結合されます。インポートは 1 回きりの手続きで、復元不可能な操作です。

DIM プロジェクトをインポートして、製品またはそのインストール パッケージの問題をトラブルシュートすることを考えているとします。DIM プロジェクトをインポートする時、インポートする前に、基本の MSI プロジェクト (.ism) のバックアップを作成するオプションがあります。このオプションを使うことにより、DIM プロジェクトがインポートされていない元の基本の MSI プロジェクトに復元することができます。



タスク ***基本の MSI プロジェクトで、DIM を特定機能にインポートするには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. **[セットアップのデザイン]** エクスプローラーで、DIM を含める機能を右クリックして、**[DIM のインポート ウィザード]** をクリックします。**[DIM のインポート ウィザード]** ダイアログ ボックスが開きます。
3. ウィザードの各パネルを完成します。

DIM を基本の MSI プロジェクトにインポート中、デザイン時の競合を解決する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM プロジェクトをインストール プロジェクトにインポートすると、DIM データがインストール プロジェクトに結合されます。データの競合が存在する場合、DIM プロジェクトの値か、または基本の MSI プロジェクトの値か、優先されるべきデータが判別される必要があります。

InstallShield では、インポート中、2 つのプロジェクトの設定間の競合を解決することができます。次は、発生する可能性があるいくつかの競合の例です:

- 同一の名前のプロパティが両方のプロジェクトで使用されていて、異なる値が設定されている場合。
- 同一の文字列識別子がある文字列が両方のプロジェクトでエントリされていて、特定の言語に対して異なる値が設定されている場合。
- 同一の名前が指定されているが、異なる値が設定されているパス変数が両方のプロジェクトで使用されている場合。
- 同一名のコンポーネントが両方のプロジェクトで使用されていて、1 つまたは複数の設定で異なる値が設定されている場合。

DIM プロジェクトのインポート中に競合が発生した時、[競合の解決] ダイアログ ボックスが開きます。このダイアログ ボックスで、任意の対応方法を指定できます。詳細については、「[競合の解決] ダイアログ ボックス」を参照してください。

ほとんど場合、プロパティ、文字列エントリ、パス変数、コンポーネントなどのアイテムを DIM プロジェクトで作成して、基本の MSI プロジェクトでは作成しなかった場合、DIM プロジェクト固有の GUID がアイテム名の一部として自動的に使用されるため、競合は発生しません。基本の MSI プロジェクトで MYPROPERTY という名前のプロパティを作成した場合、このプロパティは MYPROPERTY という名前になります。DIM プロジェクトで MYPROPERTY という名前のプロパティを作成した場合、このプロパティは内部的に MYPROPERTY.DIM_GUID という名前で使用されます。*DIM_GUID* は、DIM プロジェクトの [一般情報] ビューにある "DIM GUID" 設定で定義されている識別子を表しています。例えば、DIM でプロパティ名は、**MYPROPERTY.8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC** のようになります。この特定の例では、ダイレクト エディターを使って、DIM プロジェクトの Property テーブルで、プロパティの内部名を確認することができます。

インストール プロジェクトで DIM の要素を識別する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield で DIM プロジェクトを作成すると、プロジェクトに GUID が割り当てられます。この GUID によって、DIM プロジェクトを個別に識別できます。InstallShield では、DIM プロジェクトおよびこの DIM を含むすべての基本の MSI プロジェクトの様々箇所で、DIM に属するデータを識別するために、この GUID が使われます。

たとえば、**MyFile.exe** という名前のファイルを **8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC** という GUID を持つ DIM プロジェクトに含めると、このファイルの値が次のように、DIM プロジェクトの File テーブルに書き込まれます:

MyFile.exe.8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC

異なる DIM プロジェクトに、同一の MyFile.exe ソース ファイルを含めた場合、ファイルの File テーブル エントリの終わりに、別の GUID が付加されます。DIM プロジェクトを基本の MSI プロジェクトにインポートすると、このファイルの GUID エントリも基本の MSI プロジェクトの File テーブルにインポートされます。**MyFile.exe** ディレクトリを基本の MSI プロジェクトに追加した場合、GUID は、ファイル名の終わりに付加されません。**MyFile.exe** を含む DIM を基本の MSI プロジェクトにインポートした場合、

ファイル、コンポーネント、パス変数、および他のインストール データに対して、DIM の GUID が使用されます。これにより、基本の MSI プロジェクトのどの部分が、基本の MSI プロジェクトから直接来ているかが簡単に判別できます。また、基本の MSI プロジェクトで参照されている、または、インポートされた様々な DIM プロジェクトに元々属している箇所も簡単に判別できます。

GUID の使用は、競合の予防にもなります。たとえば、複数の DIM プロジェクトで特定のパス変数に対して異なる値を使用していて、それらの DIM を、その同じパス変数に対してさらに別の異なる値を使用している基本の MSI プロジェクトにインポートした時も、競合は発生しません。基本の MSI インストールがビルドされた時、それぞれのパス変数のインスタンスに対して適切な値が使用されます。

インストール プロジェクトで使用する DIM プロジェクト内のパス変数をオーバーライドする

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM プロジェクトの作成者によって、プロジェクトのパス変数がどう定義されたかにより、場合によって、その DIM を含む基本の MSI のリリースをビルドする時、それらの値をオーバーライドする必要があります。この手順を省くと、インストールをビルドした時、ファイルが不足しているためにビルド エラーが発生したり、間違ったファイルがビルドに取り込まれたりすることがあります。

InstallShield では、DIM プロジェクトのパス変数をオーバーライドする方法がいくつか提供されています:

- 基本の MSI リリースをビルドする時、そのプロジェクトにインポートされた DIM のパス変数をオーバーライドする場合、[リリース] ビルドの [ビルド] タブにある "パス変数のオーバーライド" 設定を使うことができます。
- ISCmdBld.exe を使って、コマンドラインからビルドする場合は、新しい -I パラメーターを使って、パス変数の名前と新しい値を指定します。1 つまたは複数の DIM を含む基本の MSI プロジェクトのインストールをビルドする場合、この方法を使用します。
- MSBuild または Team Foundation Server (TFS) を使ってビルドする場合、InstallShield タスクで PathVariables パラメーターを使います。このパラメーターは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPathVariableOverrides として露出されます。
- DIM プロジェクトを開き、必要に応じて、パス変数の値を更新します。DIM プロジェクトを開いている基本の MSI プロジェクト内から開く方法については、「インストール プロジェクト内から参照された DIM プロジェクトを開く」を参照してください。

ビルド時に製品のソース ファイルを取得して、リリースをビルドする時、設定されたパス変数のオーバーライドが使用されます (これは Standalone Build にも適用します)。

DIM プロジェクトのパス変数に関する情報およびヒントに関しては、「DIM でパス変数を使用する」を参照してください。

ターゲット システムの構成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

また記述の通り、このトピックの一部は特定のプロジェクトの種類には適用しません。

すべてのインストールは、何らかの方法でターゲット システムを変更します。簡単なインストールでは、ファイルをコピーするだけのものもあります。より複雑なインストールでは、レジストリの変更、.ini ファイルの編集、ショートカットの作成、ODBC リソースの構成、環境変数の使用、XML ファイルの変更、およびテキスト ファイルの変更、タスクのスケジュール、および Windows サービスの制御が行われます。ターゲット システムの構成方法に関する詳しい情報は、ドキュメントのこのセクションを参照してください。

このセクションには、次のトピックが含まれています:

- ショートカットおよびプログラム フォルダーの作成
- レジストリの編集
- .ini ファイル データの変更
- ODBC リソースの構成
- 環境変数を使用する
- XML ファイルの変更
- テキスト ファイルの変更
- スケジュール タスク
- Windows サービスのインストール、制御、および構成
- 実行時に既定のユーザー アカウントを作成する
- ユーザーごとのインストールとマシンごとのインストールの違い
- 管理者権限なしに InstallScript インストールを実行する

ショートカットおよびプログラム フォルダーの作成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ショートカット] ビューの [ショートカット] エクスプローラーを使って、Windows デスクトップと [スタート] メニュー上に、ショートカットおよびプログラム フォルダーを作成できます。ファイルと同様に、ショートカットはコンポーネントに関連しています。インストール プロジェクトでは、コンポーネントが所属する機能がインストールに選択された場合にのみ、ターゲット システムに作成されます。ショートカットおよびプログラム フォルダーについての詳細は、次を参照してください:

- ショートカットの種類
- ショートカットの作成
- ショートカットの設定を構成する
- ショートカットのアイコンを指定する
- インターネット ショートカットの作成
- フォルダーにショートカットを作成する
- ショートカットにアクセスできるキーボード ショートカットを指定する
- ショートカット名の変更
- ショートカットのシェル プロパティを設定する
- 基本の MSI プロジェクトのアンインストール ショートカットを作成する
- InstallScript と InstallScript MSI プロジェクトのアンインストール ショートカットを作成する
- スタート画面上のデスクトップ アプリのタイルの外観を構成する

ショートカットの種類



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*

- *MSM データベース*
- *トランスフォーム*

InstallShield では、いくつかのタイプのショートカットが提供されています。

テーブル 1・ショートカットの種類

ショートカットの種類	プロジェクトの種類	説明
新しいショートカット	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	プロジェクトに存在するファイルへの新しいショートカットを作成します。  ***メモ:*** *このオプションは、コンポーネントのファイルが指定されていない場合は無効です。*
新しいアドバタイズ ショートカット	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	アドバタイズ ショートカットの作成コンポーネントのファイルはエンドユーザーがショートカットを起動するまでターゲット システムにインストールされません。
新しいインターネット ショートカット	InstallScript	"インターネット ショートカット" 設定が [はい] に設定されているインターネット ショートカットを作成します。"ターゲット" 設定は、デフォルト値の **http://www.YourCompanyName.com** になっていますが、この値を希望の URL に変更します。
既存ファイルへの新しいショートカット	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	ターゲット システムに既に存在するファイルへのショートカットを作成します。
新しいフォルダー	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	プログラム フォルダーを作成します。たとえば、会社名の下にショートカットを表示したい場合は、プログラム フォルダーを作成することができます。

ショートカットの作成



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ショートカットを作成する前に、最低でも 1 つの機能とコンポーネントを作成する必要があります。最初のショートカットを作成するときに、コンポーネントが作成されていない場合は、InstallShield によってコンポーネントが作成されます。ショートカットが属するコンポーネントは、ショートカットの "コンポーネント" 設定を構成することによって変更できます。



タスク

ショートカットを作成するには、以下の手順に従ってください。

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、インストール先ディレクトリの 1 つを右クリックして、適切なコマンドをクリックします。使用可能なコマンドの一覧は、「ショートカットの種類」をご覧ください。

   新しいショートカットが、**新しいショートカットN** というデフォルト名で追加されます (ここで *N* は連続番号です)。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。
4. ショートカットの設定を構成します。

ショートカットとフォルダーについて構成可能な各設定についての詳細は、以下を参照してください:

- ショートカットの設定
- フォルダーの設定



メモ- *たとえば、会社名の下にショートカットを表示したい場合は、プログラム フォルダーを作成することができます。ショートカットのフォルダーを作成した後、そのフォルダーを右クリックし、[新しいショートカット] を選択して、ショートカットを作成できます。*

動的にリンクされたファイルへのショートカットを作成することはできません。詳細については、「ダイナミック ファイル リンクの制限事項」を参照してください。



ヒント: *[コンポーネント] ビューまたは、インストール プロジェクトの場合は、[セットアップのデザイン] ビューを使ってショートカットを作成することもできます。[コンポーネント] ビューまたは [セットアップのデザイン] ビューを使用する場合、エクスプローラーで [ショートカット] 項目をクリックします。[ショートカット] エクスプローラーが新しいペインで開きます。*

ショートカットの設定を構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ショートカットを作成後、ショートカットの設定を構成します。ショートカットの設定を利用して、ショートカットをファイルにリンクさせたり、ショートカットの説明を提供したり、または引数を渡したりすることができます。



タスク

ショートカットの設定を構成するには、以下の手順に従います。

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、ショートカットをクリックします。ショートカットの設定が、右のペインに表示されます。
3. 必要に応じて設定を指定します。

ショートカットとフォルダーについて構成可能な各設定についての詳細は、以下を参照してください:

- ショートカットの設定
- フォルダーの設定

ショートカットのアイコンを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield では、実行時にターゲット システムで作成されるショートカットに使用されるアイコンを指定することができます。



タスク ***ショートカットのアイコンを指定するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、アイコンを指定するショートカットをクリックします。ショートカットの設定が、右のペインに表示されます。
3. **"アイコン ファイル"** 設定で、作成するショートカットのアイコンが含まれているファイルを指定します。アイコン リソースを含む .ico ファイル、または実行可能ファイル (.dll または .exe) を指定する必要があります。

   基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトの場合: アイコンを含むファイルの完全修飾パスを入力するか、または省略記号 (...) ボタンをクリックして参照します。

   InstallScript プロジェクトの場合: アイコンを含むターゲット システム上にあるファイルへの完全修飾パスを入力します。
4. 指定したアイコン ファイルに 1 つ以上のアイコン リソースがある場合、**"アイコン インデックス"** 設定にインデックスを入力します。

   負の数以外の整数を指定すると、実行可能ファイルのアイコン リソースの順番が参照されます。たとえば、0 はファイル内の最初のアイコン、1 は 2 番目のアイコン、2 は 3 番目のアイコンを参照します。

次のいずれかの条件が True でない限り、InstallShield は [ショートカット] エクスプローラーのショートカットに表示されたアイコンを指定したアイコンにアイコンに変更します。

- プロジェクトの種類が InstallScript である。
- ショートカットが、ターゲット システム上に既に存在するファイルである。

上記の両条件の場合、アイコン ファイルは実行時まで認識されません。したがって、[ショートカット] エクスプローラーでは、ターゲット システム上で実行時に使用されるアイコンの代わりに、各ショートカットに次のアイコンが表示されます。



"アイコン ファイル" 設定で選択されたファイルがアイコンを含まない場合、このアイコンが [ショートカット] エクスプローラーのショートカットに使用されます。



プロジェクト: *基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトの場合: Windows Installer ではコンポーネントがアドバタイズされるときに別のアイコンが必要なため、InstallShield は指定した実行ファイルからアイコンを取り出します。*



ヒント: *基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトの場合: [ショートカット] ビューのアイコンを右クリックしてから、[ショートカットの変更] をクリックして、異なるショートカット アイコンを指定することもできます。この方法で指定された値でアイコンの設定内の値が更新されます。*

インターネット ショートカットの作成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

インターネット ショートカットとは、拡張子 .url を持つテキスト ファイルのことです。



タスク ***Web サイトを開くインターネット ショートカットを作成するには、次の手順を実行します。***

1. 次の内容を含む **MySite.url** という名前のテキスト ファイルを作成します:

   [InternetShortcut]

   URL=http://www.MySite.com

2. このファイルをコンポーネントに追加します。

単純に右クリックして [追加] をクリックした場合、ショートカットそのものではなくショートカットがポイントしているものを追加することになります。

ユーザーがこのショートカットを起動すると、指定された Web サイト (**www.MySite.com**) がユーザーのデフォルトのブラウザーで開きます。

フォルダーにショートカットを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield では、フォルダーへのショートカットを作成できます。フォルダーへのショートカット作成方法は、使用するプロジェクトの種類によって異なります。

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト

ショートカットの "ターゲット" 設定には、角括弧の中にディレクトリ識別子を含めることができます。たとえば、INSTALLDIR にショートカットを作成するには、次の設定でショートカットを作成します。

- 表示名: INSTALLDIR へのショートカット
- アドバタイズ: なし
- ターゲット: [INSTALLDIR]

InstallScript および InstallScript オブジェクト プロジェクト

AddFolderIcon 関数は 3 番目の (szCommandLine) パラメーターでフォルダーへのパスを渡すと、そのフォルダーへのショートカットを作成します。

たとえば、このコードはエンド ユーザーの [プログラム] フォルダーに Common Files フォルダーへのショートカットを作成します。

```
ProgDefGroupType(COMMON)

szFolder = TARGETDIR;
LongPathToQuote(szFolder, TRUE);

AddFolderIcon(
    ProgramMenuFolder,        // ショートカットの作成場所
    "TARGETDIR へのショートカット",// ショートカット表示名
    szFolder,        // ショートカットによって起動されるもの
    ""// 作業ディレクトリ
    TARGETDIR ^ "Sample.exe", 0,       // アイコンファイル、インデックス
    "",    // ショートカット キー
    NULL);              // 特殊設定
```

InstallScript MSI プロジェクト

前の InstallScript セクションでの作業と同様に基本の処理を行い、サンプルコード内のすべての TARGETDIR を INSTALLDIR へ変更します。

ショートカットにアクセスできるキーボード ショートカットを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

キーボード ショートカット (ホット キーとも呼ばれます) を使って、マウスを使う代わりに CTRL+ALT+A などのキーの組み合わせを押さえることで、素早く処理を行うことができます。キーボード ショートカットを製品のショートカットに割り当てると、エンド ユーザーは適切なホット キーを押してショートカットを起動できます。



注意 - *ターゲット システム上の既存のキーボード ショートカットと競合する可能性があるため、ショートカットのキーボード ショートカットを構成することは避けることをお推めします。*



タスク ***キーボード ショートカットをプロジェクト内のショートカットに割り当てるには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、ホットキーを指定するショートカットを選択します。
3. **"ホット キー"** 設定で、省略記号ボタン (...) をクリックします。**[ホットキー]** ダイアログ ボックスが開きます。
4. このショートカットに使用するキーボード ショートカットを押します。
5. **[OK]** をクリックします。

"ホット キー" 設定に、押されたキーの組み合わせを示す適切な 10 進数の値が表示されます。

たとえば、キーの組み合わせが CTRL+ALT+A の場合、この設定には 1601 と表示されます。この数値は、CTRL の 16 進数の値 (200) と ALT の 16 真数の値 (400)、および論理 Or 演算子を組み合わせて取得されます。次に、この数値 (600) に A キーの 16 進数値 (41) を追加し、最後に 10 進数値に変換します。この例では、10 進数に変換される数値は 641 で、変換後は 1601 となります。

ショートカット名の変更



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

新しいショートカットを作成する、デフォルトの内部名で表示されます。この名前はエンドユーザーに表示されませんが、プロジェクトに関連した名前に変更することができます。



タスク ***ショートカットの名前を変更するには、次の操作を実行します。***

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、名前を変更するショートカットを右クリックして、**[名前の変更]** をクリックします。
3. 新しい名前を入力します。

ショートカットのシェル プロパティを設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield では、インストール実行時に Windows シェルによる設定が必要な 1 つ以上のショートカット プロパティを指定できます。たとえば、InstallShield には次の動作を制御するシェル プロパティを設定するためのビルトイン サポートが搭載されています:

- Windows 8 スタート画面への初回のショートカットのピン留めを抑制する
- エンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めできるかどうかを指定する
- [スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐ

また、Windows シェルがサポートする追加プロパティを設定することもできます。詳細については、「カスタム アクション シェルのプロパティを設定する」を参照してください。

Windows 8 スタート画面への初回のショートカットのピン留めを抑制する

プロジェクト- この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

プロジェクト固有の違いについては、必要に応じて記述されています。

InstallShield では、Windows 8 ターゲット システム上で、スタート画面への初回のショートカットのピン留めを抑制するかどうかを指定できます。

***メモ-** Windows 8.1 では、新しくインストールされたアプリケーション/ショートカットの自動的なピン留めが無効化されました。*

ショートカットのピン留めを抑制する場合、インストールは Windows 8 で使用可能になった Windows シェル プロパティを設定します。インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを無効化したい場合があります。

以前のバージョンの Windows はこのショートカット プロパティを無視します。

タスク

Windows 8 スタート画面へのショートカットのピン止めをデフォルトで抑制するかどうかを指定するには、次の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、構成するショートカットを選択します。ショートカットの設定が、右のペインに表示されます。
3. **"シェル プロパティ"** 設定で、**[新しいシェル ショートカット プロパティ の追加]** ボタンをクリックしてから、**[Windows 8 スタート画面への初回のピン止めを抑制する]** を選択します。InstallShield は、新しい **"キー名"** 設定と、その下に関連する設定の追加行を追加して、必要に応じてそれらを構成します。

Windows Installer 5 で、ショートカット プロパティがサポートされています。以前のバージョンの Windows Installer は、これらの設定を無視します。

Windows 8 は、アプリケーションのアンインストールによってショートカットが削除された後でも、ショートカットのスタート画面へのピン留めに関する情報を保持します。そのため、ショートカットがインストール済みの場合、ターゲット システム上でこの設定は効果を持ちません。この機能をテストする際、ショートカットとそのターゲットが既にインストールされていない、クリーン マシン上でテストするようにして下さい。

Windows Installer ベースのプロジェクト (基本の MSI、DIM,InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム) で、Windows Installer がシェル プロパティを構成するショートカットをインストールするとき、次のようなメッセージと共に表示されるランタイム警告またはエラーが、一部のエンド ユーザーから報告されました。

ショートカット '[2].lnk' のプロパティ [1] を設定できませんでした。

このメッセージで、[1] は Windows Installer が設定しようとしているシェル プロパティの名前で、[2].lnk はショートカット ファイルの名前です。その場合、.lnk ファイルが別のプロセスによってロックされているかのように見られます。

InstallScript 言語のサポート

次の InstallScript 関数を使って、ショートカットをスタート画面にピン留めしないように防ぐことができます:

- CreateShortcut
- SetShortcutProperty

エンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めできるかどうかを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

プロジェクト固有の違いについては、必要に応じて記述されています。

InstallShield では、エンド ユーザーが製品をインストールした後に、ショートカットをタスクバーおよび [スタート] メニューにピン留めするためのコンテキスト メニュー コマンドを表示するかどうかを指定できます。インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを無効化したい場合があります。このピン留めを行わないようにショートカットを構成した場合、[スタート] メニューの最もよく使われている製品のリストに、ショートカットのターゲットを含められなくなります。

このピン留め機能を行わないようにショートカットを構成した場合、インストールは Windows シェル プロパティを設定します。デフォルトで、新しいショートカットにプロパティは設定されないため、エンド ユーザーはショートカットをタスクバー、および [スタート] メニューにピン留めすることができます。

特定の文字列を含むショートカットは、タスクバーまたは [スタート] メニューにピン留めすることができません。また、それらを最もよく使う製品リストに表示することもできません。例:

- Documentation
- Help

- Install
- Remove
- Setup
- Support

使用するプロジェクトの種類によって、コンテキスト メニュー コマンドを隠す方法は異なります。

Windows Installer ベースのプロジェクトの手順

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合、"シェル プロパティ" 設定を使ってコンテキスト メニュー コマンドを隠します。



タスク

ショートカットをタスクバーまたは [スタート] メニューにピン留めするためのコンテキスト メニューコマンドを隠すには、次の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、構成するショートカットを選択します。ショートカットの設定が、右のペインに表示されます。
3. **"シェル プロパティ"** 設定で、**[新しいシェル ショートカット プロパティ の追加]** ボタンをクリックしてから、**[ピン留めをしない]** を選択します。InstallShield は、新しい **"キー名"** 設定と、その下に関連する設定の追加行を追加して、必要に応じてそれらを構成します。

コンテキスト メニュー コマンドの表示を許可するには、次のいずれかを含む Property サブ設定を持つ "キー名" 設定を探してから、"キー名" 設定で [**このショートカット シェル プロパティを削除する**] ボタンをクリックします。

- System.AppUserModel.PreventPinning
- {9F4C2855-9F79-4B39-A8D0-E1D42DE1D5F3}, 9

Windows Installer 5 で、このショートカット プロパティがサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。

Windows Installer ベースのインストールでは、一部のエンド ユーザーから、Windows Installer がシェル プロパティを構成するショートカットをインストールしているときに発生する、以下のようなメッセージを含むランタイム警告またはエラーが報告されています:

ショートカット '[2].lnk' のプロパティ [1] を設定できませんでした。

このメッセージで、[1] は Windows Installer が設定しようとしているシェル プロパティの名前で、[2].lnk はショートカット ファイルの名前です。その場合、.lnk ファイルが別のプロセスによってロックされているかのように見られます。

InstallScript プロジェクトの手順

InstallScript プロジェクトの場合、"ピン留めを禁止する" 設定を使います。

タスク

ショートカットをタスクバーまたは [スタート] メニューにピン留めするためのコンテキスト メニューコマンドを隠すには、次の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、構成するショートカットを選択します。ショートカットの設定が、右のペインに表示されます。
3. **"ピン留めを禁止する"** 設定で **[はい]** を選択します。

Windows Installer 7 から、この設定がサポートされています。以前のバージョンの Windows はこの設定を無視します。

InstallScript 言語のサポート

次の InstallScript 関数を使って、ショートカットをタスクバーまたは [スタート] メニューにピン留めするためのコンテキスト メニューコマンドを隠すことができます:

- CreateShortcut
- SetShortcutProperty

[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

プロジェクト固有の違いについては、必要に応じて記述されています。

オプションで InstallShield では、エンド ユーザーが製品をインストールした後に、[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐことができます。これは、ターゲット システム上で個別のアイテムに対して [[スタート] メニューのカスタマイズ] ダイアログ ボックスで **[新しくインストールされたプログラムを強調表示する]** チェック ボックスをクリアした場合と同じ効果を持ちます。インストールの一部であるツールまたは従属的な製品のショートカットのプロパティを設定したい場合があります。

ショートカットの強調表示機能の抑制を指定した場合、インストールは Windows シェル プロパティを設定します。デフォルトで、新しいショートカットのプロパティは設定されていません。

使用するプロジェクトの種類によって、強調表示をしないように防ぐ方法が異なります。

Windows Installer ベースのプロジェクトの手順

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合、"シェル プロパティ" 設定を使って強調表示をしないように防ぐ動作を設定します。



タスク

ショートカットの [スタート] メニューのエントリを、新しくインストールされたアイテムとして強調表示しない場合は、以下の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、構成するショートカットを選択します。ショートカットの設定が、右のペインに表示されます。
3. **"シェル プロパティ"** 設定で、**[新しいシェル ショートカット プロパティ の追加]** ボタンをクリックしてから、**[新規として強調表示しない]** を選択します。InstallShield は、新しい **"キー名"** 設定と、その下に関連する設定の追加行を追加して、必要に応じてそれらを構成します。

[スタート] メニュー エントリの強調表示を有効化するには、次のいずれかを含む "プロパティ" サブ設定を持つ "キー名" 設定を探してから、"キー名" 設定で [**このショートカット シェル プロパティを削除する**] ボタンをクリックします。

- System.AppUserModel.ExcludeFromShowInNewInstall
- {8F4C2855-9F79-4B39-A8D0-E1D42DE1D5F3}, 9

Windows Installer 5 で、このショートカット プロパティがサポートされています。以前のバージョンの Windows Installer はこのプロパティを無視します。

Windows Installer ベースのインストールでは、一部のエンド ユーザーから、Windows Installer がシェル プロパティを構成するショートカットをインストールしているときに発生する、以下のようなメッセージを含むランタイム警告またはエラーが報告されています:

ショートカット '[2].lnk' のプロパティ [1] を設定できませんでした。

このメッセージで、[1] は Windows Installer が設定しようとしているシェル プロパティの名前で、[2].lnk はショートカット ファイルの名前です。その場合、.lnk ファイルが別のプロセスによってロックされているかのように見られます。

InstallScript プロジェクトの手順

InstallScript プロジェクトの場合、"新規として強調表示しない" 設定を使います。



タスク

ショートカットの [スタート] メニューのエントリを、新しくインストールされたアイテムとして強調表示しない場合は、以下の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、構成するショートカットを選択します。ショートカットの設定が、右のペインに表示されます。
3. **"新規として強調表示しない"** 設定で **[はい]** を選択します。

Windows Installer 7 から、この設定がサポートされています。以前のバージョンの Windows はこの設定を無視します。

InstallScript 言語のサポート

次の InstallScript 関数を使って、強調表示しないように防ぐことができます:

- CreateShortcut
- SetShortcutProperty

カスタム アクション シェルのプロパティを設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

プロジェクト固有の違いについては、必要に応じて記述されています。

InstallShield では、実行時に Windows シェルによる設定が必要な 1 つ以上のショートカット プロパティを指定できます。シェルが設定できるプロパティは、Windows SDK に含まれている propkey.h で定義されています。

使用するプロジェクトの種類によって、強調表示をしないように防ぐ方法が異なります。

Windows Installer ベースのプロジェクトの手順

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合、"シェル プロパティ" 設定を使って 1 つ以上のシェル プロパティを設定します。



タスク ***プロジェクトに含まれるショートカットの "シェル" プロパティを設定するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、構成するショートカットを選択します。ショートカットの設定が、右のペインに表示されます。
3. **"シェル プロパティ"** 設定で、**[新しいシェル ショートカット プロパティ の追加]** ボタンをクリックしてから、**[カスタム プロパティ]** を選択します。InstallShield は、新しい **"キー名"** 設定と、その下に関連する設定とプレースホルダー— テキストを含む追加行を追加します。
4. 必要に応じて、各設定を構成します。

必要に応じて、ショートカットに複数のプロパティを追加できます。

Windows Installer 5 では、シェル ショートカット プロパティがサポートされています。以前のバージョンの Windows Installer は、これらのプロパティを無視します。

ショートカットのプロパティの構成に関する詳細は Windows Installer ヘルプ ライブラリの「MsiShortcutProperty Table」を参照してください。

Windows Installer ベースのインストールでは、一部のエンド ユーザーから、Windows Installer がシェル プロパティを構成するショートカットをインストールしているときに発生する、以下のようなメッセージを含むランタイム警告またはエラーが報告されています:

ショートカット '[2].lnk' のプロパティ [1] を設定できませんでした。

このメッセージで、[1] は Windows Installer が設定しようとしているシェル プロパティの名前で、[2].lnk はショートカット ファイルの名前です。その場合、.lnk ファイルが別のプロセスによってロックされているかのように見られます。

InstallScript 言語のサポート

次の InstallScript 関数を使って、InstallScript プロジェクトおよびその他のプロジェクト タイプでシェル プロパティを構成できます:

- CreateShortcut
- SetShortcutProperty

基本の MSI プロジェクトのアンインストール ショートカットを作成する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*



Windows ロゴ: *現在の Windows ロゴ ガイドラインによると、ベスト プラクティスは [スタート] メニューでアプリケーションを削除するショートカットを配置しないこととなっています。アンインストール ショートカットは、アプリケーションのアンインストーラが [プログラムの追加と削除] にあるので不要です。*

エンドユーザーが簡単にシステムから製品をアンインストールできるように、アンインストールのショートカットを作成することができます。このショートカットは、起動すると自動的にアンインストール プロセスを開始します。



タスク ***アンインストール ショートカットを作成するには、以下の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラで、ショートカットのインストール先を右クリックしてから、**[既存ファイルへの新しいショートカット]** をクリックします。新しいショートカットが、**新しいショートカットN** というデフォルト名で追加されます (ここで *N* は連続番号です)。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。
4. [**引数**] フィールドに、**/x [ProductCode]** とタイプします。2 つの引数は、間にスペースを入れて区切ります。

Msiexec.exe は、Windows Installer サービスのコマンドライン エンジンです。/x 引数は、Windows Installer サービスに、製品コードによって参照された製品をアンインストールするよう指示します。

5. **"アドバタイズ"** フィールドを **[いいえ]** に設定します。
6. **"ターゲット"** フィールドで、リストから **[SystemFolder]** を選択し、この場所に **msiexec.exe** を付加します。
7. **"コンポーネント"** フィールドで、このショートカットを関連付けるコンポーネントを選択します。ショートカットが常にインストールされるようにするには、アプリケーションのメイン実行可能ファイルが含まれるコンポーネントにショートカットを関連付けます。

InstallScript と InstallScript MSI プロジェクトのアンインストール ショートカットを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*



メモ: *アンインストール ショートカットの作成は必要ないと思うユーザーの方もいるでしょう。アンインストール ショートカットは、アプリケーションのアンインストーラが [プログラムの追加と削除] にあるので不要です。*

エンドユーザーが簡単にシステムから製品をアンインストールできるように、アンインストールのショートカットを作成することができます。エンドユーザーがショートカットを起動すると、自動的にアンインストール プロセスが開始します。

次の InstallScript コードは、ファイルの転送後、デスクトップに **[Uninstall Application (アプリケーションのアンインストール)]** という名前のショートカットを作成します。このショートカットを起動すると、インストールはメンテナンス モードで実行されます。このコードは、InstallScript プロジェクトと InstallScript MSI プロジェクトで使用できます。

```
function OnMoved()
begin
  if( !REMOVEALLMODE ) then
    AddFolderIcon( FOLDER_DESKTOP, "Uninstall Application", UNINSTALL_STRING + ADDREMOVE_STRING_REMOVEONLY, "", "", 0,
"", REPLACE );
  endif;
end;
```



ヒント: *ADDREMOVE_STRING_REMOVEONLY を含めると、–removeonly がコマンドラインに追加されます。この追加によって、REMOVEONLY が設定され、OnMaintUIBefore イベント ハンドラーがアンインストール オプションのみを表示するようになります。標準のメンテナンス ユーザー インターフェイスを表示する場合、ADDREMOVE_STRING_REMOVEONLY を含めないでください。*

スタート画面上のデスクトップ アプリのタイルの外観を構成する

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

Windows 8 からアプリケーション タイルのグリッドをスタート画面に表示できるようになりました。これは、今までのショートカットの一覧に取って代わるもので、ショートカットの代わりにタイルを配置します。InstallShield は、スタート画面上のデスクトップ アプリのタイルの外観をカスタマイズすることができます。次のタイル構成設定が使用できます:

- アプリケーション名を中サイズ (150x150) のタイルに含めるとき、明色または暗色のテキストを切り替える
- タイル背景色を選択
- カスタム タイル イメージ (小: 70x70、中:150x150) の使用オプション
- アプリケーション名を中サイズ タイルに表示または非表示を選択

[タイルの構成] ノードは、メインの **[ショートカット]** ビューおよび各コンポーネントの **[ショートカット]** サブビューに表示されます。すべての該当するタイル構成が一覧表示されます。コンポーネントの場合、**[タイルの構成]** は、そのコンポーネントに含まれる .exe ファイルの構成にのみ 適用します。メイン ビューの場合、フィルターに構成済みの .exe ファイルをターゲットとするコンポーネントが含まれる場合にのみ適用します。

メモ- *ショートカットおよびタイルは関連していますが、これらは異なります。ショートカットを個別に構成して、異なるアイコンを使用する同じ .exe ファイルに対して複数のショートカットを作成することもできますが、1 つの .exe ファイルに対して 1 つのタイル構成のみ作成することが可能です。*

タスク ***スタート画面上のデスクトップ アプリのタイルの外観を構成するには、以下の手順に従います:***

1. **[ショートカット]** ビュー、または**[セットアップのデザイン]** あるいは **[コンポーネント]** ビューの **[ショートカット]** ノードで、**[ショートカット]** をクリックします。
2. **[タイルの構成]** を右クリックしてから、**[タイル構成の追加]** コンテキスト メニュー オプションを選択します。.

   メモ・ *この手順は、.exe ファイルがプロジェクトに追加されていることを前提とします。.Exe ファイルがプロジェクトに含まれていない場合、**[構成の追加]** コンテキスト メニュー オプションは無効になります。*

   ヒント・ *別の方法として、スタート画面タイルを構成する .exe ファイルをターゲットとするショートカットを右クリックしてから、**[タイルの構成]** コンテキスト メニュー オプションを選択します。新しいタイル 構*

*成が追加および選択されます。ショートカットに構成が既に含まれている場合、**[タイルの構成]** オプションは無効化されます。*

3. [タイル ターゲットの参照] ダイアログ ボックスで、タイルの外観を構成するアプリケーションの .exe ファイルを参照して選択します。新しいタイル 構成が追加されます。
4. **[タイルの構成]** で使用可能な任意の設定を使って、スタート画面上のデスクトップ アプリのタイルをカスタマイズします。使用可能な設定についての詳細は、「[タイル構成] の設定」を参照してください。

レジストリの編集



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

Windows レジストリは、アプリケーションとオペレーション システムで使用される構成情報を含むシステム全体のデータベースです。このレジストリは、次を含むすべての情報を格納します。

- 会社名、製品名、バージョン番号等のアプリケーション情報
- アプリケーションの実行を可能にするパス情報
- エンドユーザーが、システム上にある他のアプリケーションを妨げることなく簡単にアプリケーションをアンインストールができるアンインストール情報
- アプリケーションによって作成される文書のためのシステム全体のファイル関連付け
- ライセンス情報
- ウィンドウの位置等のアプリケーション オプションのデフォルトの設定

キー、値名、および値

レジストリは、My Computer エクスプローラーの下に階層的に編成された 1 セットのキーで構成されます。My Computer のすぐ下には、いくつかのルートキーがあります。インストールは、レジストリのどのキーにもキーと値を追加することができます。以下は、通常インストールによって影響を受けるルートキーです。

- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_USER
- HKEY_CLASSES_ROOT

キーは、レジストリ内の名前が付けられた場所です。キーは、サブキー、値名、値のペア、およびデフォルト（名前が付いていない）値を含むことができます。値名と値のペアは、キーの下の2 つの部分からなるデータ構造です。値名はキーの下のストレージの値を見分けます。また、その値は値名に関連付けられた実際のデータです。値名が値に特定されていない場合、その値はそのキーのデフォルト値になります。各キーは、デフォルト（名前の付いていない）値を 1 つのみ持つことができます。

terms キーとサブキーはお互い関連していることに注意してください。レジストリでは、別のキーの下のキーを、レジストリ階層の別のキーに関連してそれをどう参照するかによって、サブキーまたはキーとして参照することもできます。

InstallShield プロジェクトとレジストリ

InstallShield の [レジストリ] ビューを利用して、エンド ユーザーのレジストリを変更する作業を簡単に行うことができます。Windows のレジストリ エディターとほぼ同じようにこのビューを使用して、キーおよび値を作成します。

すべてのレジストリ データ（InstallScript プロジェクトの <Default> レジストリ セットを除く）をコンポーネントに関連付ける必要があります。インストール プロジェクトでは、コンポーネントの機能がインストールに選択されると、そのコンポーネントのレジストリ データがターゲット システムに設定されます。



注意 - *レジストリは Windows オペレーティング システムの重要な部分なので、レジストリをむやみに変更または削除することは避けてください。不可欠なレジストリ キーが変更されると、システムが機能しなくなることがあります。*



メモ - *また、インストーラは、[一般情報] ビューに指定した値に基づいて、特定のレジストリ エントリも自動的に作成します。*



Windows ロゴ - *これらの「情報キー」は、Windows ロゴ プログラムの要件を満たすために必要です。*

また、コンポーネントの詳細設定はすべて、ターゲット システム上でファイルの登録に使用されます。

コンポーネントまたは機能によるレジストリ エントリのフィルタリング



プロジェクト - *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

いくつかのプロジェクト タイプ (DIM、マージ モジュール、MSM データベース) では、機能はサポートされていません。これらのプロジェクト タイプでは、コンポーネント別にフィルターすることはできますが、機能別にフィルターすることはできません。

[レジストリ] ビューには、ビュー フィルターが含まれています。このフィルターで、レジストリ データをビューに表示するコンポーネントまたは機能を選択することができます。

ビューフィルターは、プロジェクトの機能、サブ機能、およびコンポーネントの階層をリスト表示します。機能を選択すると、機能に含まれるすべてのコンポーネントのレジストリ データが表示されますが、既存エントリの変更と削除のみ可能です。新しいレジストリキーを追加するにはコンポーネントを選択しなくてはなりません。

機能の階層にサブ機能が含まれる場合、親機能を選択するとその機能のレジストリ エントリのみが表示されます。ここにサブ機能のレジストリ エントリは表示されません。

コンポーネントを参照する

[レジストリ] エクスプローラー内からコンポーネントを参照することができます。ビューフィルターの右側の省略ボタンをクリックして、[コンポーネント] ダイアログ ボックスの参照を起動しますダイアログ ボックスから、[機能] リストにある機能を選択してから、既存コンポーネントを選択するか、新規コンポーネント ボタンをクリックして新しいコンポーネントを作成します。

ビュー フィルターが [すべてのアプリケーション データ] に設定されている場合、インストール先コンピューターの [レジストリ] ビュー ペインでレジストリ ハイブを右クリックして、[コンポーネントを参照] を選択してダイアログ ボックスを表示することができます。

プロジェクトでのすべてのレジストリ エントリの表示

インストールのすべてのレジストリ エントリを表示するには、ビュー フィルターで [すべてのアプリケーション データ] オプションを選択します。レジストリ キーを右クリックした後、コンテキスト メニューから [すべてエクスポート]、または、[選択したブランチのエクスポート] を選択して、レジストリ データをエクスポートできます。

メモ: *[すべてのアプリケーションデータ] でビューをフィルターして、レジストリキーと値を変更、名前の変更、または削除することができます。*

[レジストリ] ビューの [インストール先コンピューターのレジストリ] ビュー ペインでレジストリ キーをクリックすると、InstallShield は、そのキーのすべてのレジストリ データを [レジストリ] ビューの右下のペインに表示します。[コンポーネント] 列には、レジストリ データが関連付けられているコンポーネントが表示されます。同じ値が複数のコンポーネントに存在する場合は、レジストリデータに関連付けられている最後のコンポーネントが表示されます。

キーに値を設定していない場合、ビュー フィルターで [すべてのアプリケーション データ] を選択しても、そのキーのレジストリ データは一切表示されません。

[レジストリ] ビューのリフレッシュ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

タスク ***[レジストリ] ビューをリフレッシュするには、以下の手順を実行します。***

F12 キーを押します。

レジストリ キーの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

タスク ***コンポーネントのインストール時に、ターゲット システムで作成されるレジストリ キーを指定するには、以下の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. *基本の MSI*、*DIM*、*InstallScript MSI*、*マージ モジュール*、*MSI データベース*、*MSM データベース*、*トランスフォーム*、*および QuickPatch プロジェクト*—**[ビュー フィルター]** リストで、レジストリ データを関連付けるコンポーネントをクリックします。

   メモ: ***[すべてのアプリケーションデータ]*** *が選択されている場合、****[レジストリ]*** *ビューは読取り専用になります。*

   InstallScript および InstallScript Object プロジェクトの場合 – **[インストール先コンピューターのレジストリ] ビュー** ペインで既存のレジストリ セットを開くか、**[インストール先コンピューター]** フォルダーを右クリックしてレジストリ セットを作成します。レジストリ セットをクリックして **[レジストリ セットのインストール条件]** ペインで適切なコンポーネントを選択し、レジストリ セットを 1 つまたは複数のコンポーネントに関連付けます。

3. **[インストール先コンピューターのレジストリ] ビュー** ペインで、レジストリ キーを右クリックして、**[新規作成]** をポイントして、**[キー]** をクリックします。InstallShield が、**新しいキー –n***n* (ここで n は連続番号です) という名前で新しいキーを追加します。
4. ここでキー名をわかりやすい名前に変更するか、または、後でキーを右クリックして **[名前の変更]** をクリックし、新しい名前を付けます。

InstallShield が、空白のデフォルトの文字列値を持つ新しいキーを追加します。

デフォルトでは、作成したすべてのキーは、自動インストールおよびアンインストールに設定されています。つまり、所属するコンポーネントがインストールされるとキーもインストールされ、そのコンポーネントがアンインストールされるとキーもアンインストールされます。レジストリ キー フラグの詳細については、「レジストリ フラグ」を参照してください。

ヒント: *新しいキーを作成し、キーの名前に* **KKey 1¥Key 2¥Key 3** *などと入力することによって、ネストされた複数のキーを 1 度に作成することができます。InstallShield は、Key 3 が Key 2 のサブキーであるところにネストされたキー構造を作成します。Key 2 は Key 1 のサブ キーになります。*

レジストリ エントリをドラッグアンドドロップしてレジストリ キーを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

インストール プロジェクトにレジストリ エントリを追加する最も手軽な方法は、[レジストリ] ビューの [ソース] ペインからレジストリ エントリのひとつをドラッグし、[インストール先] ペインにドロップする方法です。すべてのキーを **[インストール先コンピューターのレジストリ] ビュー** ペインへドロップした場合、そのキーのすべてのサブキーおよび値が選択したコンポーネントに追加されます。また InstallScript プロジェクトでは、エントリをドロップしたレジストリセットに追加されます。

コンテキスト メニューを使ってキーをドラッグ & ドロップする

コンテキスト メニューを使って、複数のキーや値を一度に移動することができます。レジストリ エントリを右クリックし、それをインストール先にドラッグし、コンテキスト メニューから [オプション] をクリックします。

テーブル 2 ・ レジストリ エントリのコンテキスト メニューから使用できるコマンド

オプション	説明
すべてのキーと値	すべての選択されたキー、サブキーおよび値を追加します。
キーとその値のみ	選択されたキーとそのキーの値のみを追加します。サブキーは追加されません。
このキーのみ	選択されたキーのみ追加し、そのサブキーまたは値は追加されません。
キャンセル	変更を加えずにドラッグアンドドロップ操作を終了します。

64 ビット開発システム上で、ソースマシンのレジストリの 32 ビットおよび 64 ビット領域の両方を表示する

InstallShield を 64 ビット開発システム上で使用する場合、InstallShield が表示する [レジストリ] ビューは、使用中のマシンのレジストリの 32 ビットおよび 64 ビット領域の両方を表示します:

- HKEY_LOCAL_MACHINE¥Software
- HKEY_LOCAL_MACHINE¥Software¥Wow6432Node

このサポートによって、プロジェクトのレジストリ データの変更を構成する際、これらのソース領域からこのビューのインストール先ペインの適切な領域にエントリーをドラッグ アンド ドロップすることが可能となります。

インストールがレジストリ データを、32 ビット領域にリダイレクトせずに、64 ビット ターゲット システム上のレジストリの 64 ビット領域にインストールする場合、レジストリ データを 64 ビットとしてマークされたコンポーネントに配置します。[レジストリ] ビューのソースペインから 64 ビット データを、ビュー内のインストール先ペインにドラッグするだけでは、そのコンポーネントが 64 ビットであるとマークされません。詳細については、「64 ビット オペレーティング システムをターゲットにする」を参照してください。

データを他のマシンからインポートする

ただし、ドラッグ アンド ドロップ操作は、レジストリ エントリがインストール開発システムに存在する場合にのみ有効です。別のマシンからのレジストリ データがある場合、REG ファイルのインポート ウィザードを使って、データをインポートできます。

レジストリ キーの削除



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*

- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*



タスク ***レジストリ キーを削除するには、以下の手順を実行します:***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. *基本の MSI*、*DIM*、*InstallScript MSI*、*マージ モジュール*、*MSI データベース*、*MSM データベース*、*トランスフォーム*、*および QuickPatch プロジェクトの場合* – **[ビュー フィルター]** リストでレジストリ キーを含むコンポーネントをクリックするか、**[すべてのアプリケーション データ]** を選択して、製品のレジストリ キーをすべて表示します。

   InstallScript と InstallScript Object プロジェクトの場合–**[インストール先コンピューターのレジストリ]** **ビュー** ペインで、レジストリ キーを含む既存のレジストリ セットを開きます。
3. **[インストール先コンピューターのレジストリ]** ビュー ペインで、削除するレジストリ キーを右クリックして、**[削除]** をクリックします。

レジストリ値の作成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*



タスク ***ターゲット システムで作成されるレジストリ キーを指定するには、以下の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[インストール先コンピューターのレジストリ]** **ビュー** ペインで、値を追加するキーをクリックします。そのキーのすべての既存のレジストリ値が **[インストール先コンピューターのレジストリ データ]** ペインにリストで表示されます。

3. 値リスト内で右クリックしてから、登録するデータ型をクリックします。使用可能なオプションは次の通りです。

テーブル 3 • レジストリ値の種類

オプション	説明
デフォルト値	キーのデフォルト値。
文字列値	規定の長さのテキスト文字列。
バイナリ値	この値は 16 進数値として解析され、格納されます。
DWORD 値	4 バイト (32 bit) の長さの数値で表示されるデータ。
複数文字列値	ヌル文字で終わる文字列配列、および 2 つの ヌル文字で終わる文字列配列としてフォーマットされた複数テキスト文字列。このコマンドを選択すると、[複数行文字列値] ダイアログ ボックスが起動します。
展開可能な文字列値	この値は展開可能な文字列として解析され、格納されます。MSDN (Microsoft Developer's Network) によると、展開可能な文字列レジストリ値は、環境変数への非展開参照を含むヌル文字で終わる文字列です (例、**%PATH%**)。

InstallShield は、New Value (新しい値) (ここで *n* は連続番号です) という名前で新しい値を追加します。ここで値名をわかりやすい名前に変更するか、または後で値を右クリックして [名前の変更] を選択し、新しい名前を付けます。



プロジェクト: *基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム および QuickPatch プロジェクトでは、すべてのレジストリの値を、コンピューターのキー パスとして使用できます。*

レジストリ値データの変更



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

タスク ***ターゲット システムで作成されるレジストリ値のデータを変更するには、以下の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[インストール先コンピューターのレジストリ] ビュー** ペインで、変更する値を含むレジストリ キーをクリックします。すべてのレジストリ値が、**[インストール先コンピューターのレジストリ データ]** ペインに一覧表示されます。
3. 変更する値をダブルクリックします。[レジストリ データの編集] ダイアログ ボックス または [複数文字列値] ダイアログ ボックス が開きます。
4. ダイアログ ボックスで情報をすべて入力し、**[OK]** をクリックします。

プロジェクト: *基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム、および QuickPatch プロジェクトの場合　角かっこ ([]) を含む値データを追加するには、各かっこの前に円記号 (¥) を入れて、それをかっこで括ります。これを省略すると、Windows Installer はこの値をプロパティと認識します。たとえば、レジストリに* **[stuff]** *と書き込む場合、値データに* **[¥[]stuff[¥]]** *を使用します。*

レジストリ値の削除

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

タスク ***レジストリ値をプロジェクトから削除するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[インストール先コンピューターのレジストリ] ビュー** ペインで、削除する値を含むレジストリ キーをクリックします。すべてのレジストリ値が、**[インストール先コンピューターのレジストリ データ]** ペインに一覧表示されます。
3. **[インストール先コンピューターのレジストリ データ]** ペインで、削除するレジストリ値を右クリックしてから **[削除]** をクリックします。

レジストリ フラグ



プロジェクト: *レジストリ フラグのサポートは、使用しているプロジェクトの種類によって異なります。*

レジストリ フラグを使用すると、レジストリ エントリのインストールを制御できます。

Windows Installer ベースのプロジェクトにおけるレジストリ フラグ



プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

レジストリ エントリは、所属先のコンポーネントがインストールされるとデフォルトでターゲットシステム上に作成されます。ターゲット システムからそのコンポーネントが削除されたときは、これらのレジストリ エントリも削除されます。製品をアンインストールした後もレジストリ エントリをターゲット システムに残しておく場合や、既に存在しない場合にのみレジストリ エントリを作成する場合には、そのキーのインストールフラグを設定する必要があります。

InstallShield では、インストール動作はサブキー レベルで設定されます。あるキー以下のすべての値には同じインストール動作およびアンインストール動作が設定されなければなりません。

キーのレジストリ フラグを変更するには、[レジストリ] ビューでプロジェクトのキーの 1 つを右クリックしてから、以下のテーブルにリストされている任意のコマンドをクリックします。

テーブル 4 · Windows Installer ベースのプロジェクトにおけるレジストリ フラグの種類

アイコン	コマンド	説明
	自動	これはすべてのレジストリキーのデフォルトのオプションです。キーが既に存在しない場合、インストールがそれを作成します。アンインストール中、キーが空白の場合は、それが削除されます。
	インストールのみ (+)	キーが存在しない場合は、作成されます。このキーが所属するコンポーネントがアンインストールされるとき、このキーはターゲット システムに残ります。 このオプションは、サブキーまたは値を含まないキーにのみ使用できます。
	すべてのキーをアンインストール (–)	このフラグが空のキーに割り当てられた場合、キーはインストール時に作成されません。このフラグが値を含むキーに割り当てられていて、そのキーがターゲット システムに存在しない場合、インストール時にそのキーと値が作成されます。どちらの場合においても、キー、すべてのサブキー、および値はアンインストール時に削除されます。これはサブキーや値がインストール後に追加された場合も同様です。

テーブル 4 - Windows Installer ベースのプロジェクトにおけるレジストリ フラグの種類 (続き.)

アイコン	コマンド	説明
	存在しない場合はインストール、存在する場合はアンインストール (*)	このオプションは、1 つの例外 (自動レジストリ フラグ) を除いて、デフォルトの動作に似ています。自動レジストリ フラグの場合、アンインストール中にそのキーが空白でないとき、それは削除されません。**[存在しない場合はインストール、存在する場合はアンインストール (*)]** フラグの場合、アンインストール中にレジストリ キーが存在するときは、そのサブキーまたは値が残されているかどうかに関わらず、キーが削除されます。

InstallScript ベースのプロジェクトにおけるレジストリ フラグ

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

インストールがターゲット システムにレジストリ セットをインストールしてレジストリ キーを作成した後、これらのキーの下に他のインストールまたはアプリケーションによって (あるいは直接エンド ユーザーによって) サブキーまたは値が作成される場合があります。インストールされたレジストリ キーに、そのインストール以外によって作成されたサブキーまたは値がある場合、そのレジストリ キーがアンインストールされることを防ぐことができます。そのためには、該当するキーのレジストリ フラグを変更を行います。まず、[レジストリ] ビューにあるプロジェクトのキーの 1 つを右クリックしてから、次のコマンドをクリックします。

テーブル 5 - InstallScript ベースのプロジェクトにおけるレジストリ フラグ

アイコン	コマンド	説明
	複数のアプリケーション間で共有	このキーが属するコンポーネントがアンインストールされるとき、キーにサブキーまたは値がある場合、そのキーはターゲット システムから削除されません。(インストールで作成された値は、**[アンインストール中に削除]** フラグの設定を解除していなければ、既にアンインストールされているはずです。)キーにサブキーまたは値がない場合、キーがインストールによって作成されたかどうか、つまり、インストール時に既に存在していたかどうかに関わらず、ターゲット システムから削除されます。

たとえば、オペレーティング システムが使用するレジストリ値にインストールが変更を行っている場合、アンインストール時にエンド ユーザーのシステム上に製品のレジストリ値の一部をそのまま残したい場合があります。その場合、アンインストールしないレジストリ値を含むキーを右クリックします。そのキーの **[複数のアプリケーション間で共有]** フラグを選択します。キーの値を右クリックしてから、**[アンインストール中に削除]** コマンドが選択されていないことを確認します。

プロジェクト内のレジストリ エントリの検索

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*



タスク ***(Windows Installer プロジェクト内の)特定のコンポーネントまたは(InstallScript プロジェクト内の)レジストリ キー内でレジストリ エントリを検索するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. *基本の MSI*、*DIM*、*InstallScript MSI*、*マージ モジュール*、*MSI データベース*、*MSM データベース*、*トランスフォーム*、*および QuickPatch プロジェクト*—**[ビュー フィルター]** リストで、検索するコンポーネントを選択します。**[すべてのエントリを表示]** が選択されている場合、**[レジストリ]** ビューは読取り専用になります。

   InstallScript と InstallScript Object プロジェクトの場合—検索するレジストリ セットを開きます。
3. レジストリ エントリを右クリックして、**[検索]** を選択します。**[検索]** ダイアログ ボックスが開きます。
4. 検索する文字列を入力します。
5. **[検索]** をクリックして、検索を開始します。

Windows Installer プロジェクトの場合、InstallShieldでは、プロジェクトのすべてのレジストリ データではなく、現在のコンポーネントのレジストリ エントリのみが検索されます。

検索順序は、選択されている項目に関係なく、一番上の項目から開始して下へと移動します。一致するとその項目で停止し、キーがハイライト表示されます。検索を続行するには、右クリックして [次を検索] を選択するか F3 を押します。[次を検索] を選択すると、エクスプローラーを下方向に移動しながら、次に一致する項目を検出します。

レジストリ キーのアンインストール動作を設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

- *QuickPatch*

InstallShield では、レジストリ エントリのアンインストール動作はキー レベルで設定されます。



タスク ***レジストリ キーのアンインストール動作を設定するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[インストール先コンピューターのレジストリ] ビュー** ペインで、キーを右クリックして、適切なアンインストールの動作をクリックします。

アンインストール動作のオプション

InstallShield では、インストール動作はサブキー レベルで設定されます。あるキー以下のすべての値には同じインストール動作およびアンインストール動作が設定されなければなりません。使用可能なオプションの一覧は、「レジストリ フラグ」をご覧ください。

レジストリ値で環境変数を使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

REG_EXPAND_SZ 文字列値を使用すると、レジストリに格納されているパスに環境変数を使用できます。これらのエントリは、オペレーティング システムで環境変数として認識されるために、特別な形式を必要とします。レジストリに表示される REG_EXPAND_SZ 値の形式は、%TEMP% です。TEMP は、TEMP ディレクトリの標準環境変数です。

構文

このタイプのレジストリ エントリを作成するときは、エントリをシャープ記号で開始し、その次にパーセント記号を付ける必要があります (#%)。次に、環境変数を入力できます。環境変数の前後にはパーセント記号を付けます。したがって、の [レジストリ] ビューで **#%%TEMP%** と入力すると、レジストリに書き込まれたときに **%TEMP%** と表示されます。

このエントリの "種類" フィールドには REG_EXPAND_SZ、"データ" フィールドには %TEMP% と表示されます。

レジストリにプロパティ値を書き込む



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

Windows Installer ベースのプロジェクトでは、Windows Installer のプロパティをレジストリ値に使って、製品が後で使用する情報を保存することができます。実行時に、Windows Installer は 自動的にレジストリ データ中の **[PropertyName]** のような式は、インストーラーの実行時に Windows Installer によって、**PropertyName** というプロパティの値に自動的に置き換えられます。[プロパティ マネージャー] ビューで定義したすべてのプロパティを同じように使用できます。

たとえば、ソフトウェアのインストール先を保存する場合は、**[INSTALLDIR]** というデータをもつレジストリ値を作成します。実行時に、インストールがレジストリ値を作成するとき、そのレジストリ値のデータは、アプリケーションがインストールされる場所に設定されます。

同じフォーマットを、レジストリキーの名前および値名にも使用できます。たとえば、キー名に会社名を付ける場合、キーの名前に **[Manufacturer]** と入力します。レジストリ キーがターゲット システムで作成されると、このキーの名前が、[一般情報] ビューの で入力されたとおりに "パブリッシャー" 設定の値になります。

レジストリ ファイルをインポートする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield では、別のインストール プロジェクトのレジストリ (.reg) ファイル、または InstallShield の外で作成した既存のレジストリ ファイルをインポートすることができます。

タスク ***.reg ファイルをインポートするには、以下の手順を実行します。***

1. REG ファイルのインポート ウィザードを起動します。
2. ウィザードパネルの指示に従って .reg ファイルをインポートします。

コンポーネントまたはレジストリ セットに .reg ファイルをインポートすると、そのレジストリ データがコンポーネントまたはレジストリ セットのレジストリ データに追加され、コンポーネントまたはレジストリ セットがエンド ユーザーのシステムにインストールされている場合に、そのシステムに書き込まれます。

レジストリ ファイルのエクスポート

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

レジストリ ファイル (.reg) をエクスポートすると、プログラムのレジストリ データの一部がレジストリ ファイルにコピーされます。

タスク ***レジストリファイルをエクスポートするには、次の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[インストール先コンピューターのレジストリ] ビュー** ペインで、エクスポートするレジストリ エントリを右クリックして、**[REG ファイルのエクスポート]** または **[選択したブランチのエクスポート]** のどちらかを選択します。
   - **REG ファイルのエクスポート** では、コンポーネントのレジストリ データ全体がエクスポートされます。
   - **[選択したブランチのエクスポート]** は、現在のレジストリ キーとサブキーだけをエクスポートします。

コンポーネントのキーパスを設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

キー パスは、コンポーネントの有無を検出するために Windows Installer が使用する各コンポーネントの一意のファイルです。この値には、ファイルまたはフォルダーが含まれていなければなりません。



タスク ***コンポーネントのキーパスを設定するには、以下の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[インストール先コンピューターのレジストリ] ビュー** ペインで、キーパスに使用するレジストリ データを含むレジストリ エントリをクリックします。
3. **[インストール先コンピューターのレジストリ データー]** ペインで、レジストリ キーの値を右クリックして **[キー パスの設定]** を選択します。

値の文字列、バイナリ、または DWORD アイコンは、キー アイコンに置き換えられます。



タスク ***コンポーネントのキーパスを削除するには、以下の手順に従います。***

キー パスを右クリックして [**キー パスのクリア**] をクリックします。



メモ *コンポーネントは、1 つのキー パス、または、1 つのキーフ ァイルを持つことができます。キーパスまたはキーファイルがコンポーネントに既に割り当てられている場合、他のキーパスを設定しようとすると、警告メッセージが表示されます。ダイアログ ボックスで [はい] をクリックして、既存のキー ファイルまたはキー パスを新しいキー パスに置き換えます。*

レジストリ キーのアクセス許可を構成する



プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield で、ロックダウンされた環境で製品を実行するエンドユーザーのために、レジストリ キーを保護するための設定を構成することができます。レジストリ キーのアクセス許可を特定のグループとユーザーに割り当てることができます。たとえば、管理者グループに特定のレジストリ キーについての [読み取り]、[書き込み]、および [削除] アクセス許可を割り当てることができますが、別のグループのすべてにユーザーについては [読み取り] 許可のみ割り当てることができます。



タスク ***レジストリ キーのアクセス許可を構成するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[インストール先コンピューターのレジストリ] ビュー** ペインでレジストリ キーを右クリックし、**[アクセス許可]** ボタンをクリックします。**[アクセス許可]** ダイアログ ボックスが開きます。
3. 必要に応じて、アクセス許可を追加/変更/削除します。詳細については、「レジストリ キーの [アクセス許可] ダイアログ ボックス」を参照してください。

プロジェクトの [一般情報] ビューにある "ロックダウンの設定方法" 設定の選択に従って、InstallShield は ISLockPermissions テーブルまたは LockPermissions テーブルのどちらかにアクセス許可データを追加します。詳細については、「ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。

レジストリ テーブルのプライマリ キー



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

Windows Installer では、Registry テーブルに追加するレジストリ キーと値それぞれについて、一意のプラマリ キーが必要です。レジストリ エントリの作成状況がすべて見えるように、InstallShield では、データベースの [レジストリ] テーブルのエントリをビルドする時に各エントリに一意の名前が割り当てられます。

カスタム アクションを作成するときに、エントリのプライマリ キーが必要になるときがあります。InstallShield では、[セットアップのデザイン] ビューまたは [コンポーネント] ビューの [レジストリ データ] エクスプローラーで、レジストリ キーまたは値のプライマリ キーを指定することができます。[レジストリ] ビューを使って、キー、値、またはプライマリ キーを割り当てることはできません。

Registry のプライマリ キーを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*



タスク ***レジストリ キーまたは値のプライマリ キーを指定するには、以下の手順に従います:***

1. キーまたは値を右クリックして、**[MSI 値]** を選択します。[MSI 値] ダイアログ ボックスが開きます。
2. キーの名前を入力します。プライマリ キーは Windows Installer 識別子でなければならないので、名前には文字、数字、下線 (_)、およびピリオド (.) だけを使用し、名前の最初には文字または下線を使用します。

プライマリ キーを表示または変更するには、レジストリ キーまたは値を右クリックし、もう一度 [MSI 値] をクリックします。

値を指定しない場合、InstallShield によって、Registry テーブルのこのエントリに一意のプライマリ キーが生成されます。

レジストリ キーに複数行の文字列値を入力する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

特定のレジストリ キーに値を追加するとき、[レジストリ] ビューでは複数行の文字列を指定することができます。



タスク ***複数行の文字列値を追加するには、以下の手順に従います。***

1. 複数行の値を追加するレジストリ キーを右クリックして、**[新規]** をポイントし、**[複数行の値]** を選択します。[複数行文字列値] ダイアログ ボックスが開きます。
2. レジストリ値の変更方法を選択してから、各ヌル区切り文字列の行を入力します。

3. [OK] をクリックします。



メモ: *文字列にはスペースだけを含めることはできますが、空白にしたり、文字列の区切り文字である ["] を使うことはできません。*

ユーザーごとのインストールで HKCU の下にエントリを書き込む



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

現在のユーザーが HKEY_LOCAL_MACHINE の下にあるキーを変更するための十分な権限を持っているとはかぎりません。そのような場合、HKEY_CURRENT_USER の下にエントリを書きこむ必要があります。

[レジストリ] ビューで HKEY_USER_SELECTABLE を選択すると、インストールのタイプとエンドユーザーのアクセス権にしたがって、適切なレジストリ ハイブにエントリが作成されます。ユーザーごとのインストール (ALLUSERS プロパティが 2 に設定されているか、またはインストールがユーザー レベルのアクセス権限を持つユーザーによって実行されている場合)では、これらのエントリは HKEY_CURRENT_USER に作成されます。マシンごとのインストールでは、つまり、ALLUSERS がヌル値ではなく、エンドユーザーが管理者である場合には、エントリは HKEY_LOCAL_MACHINE に書き込まれます。



メモ: *Windows では、HKEY_LOCAL_MACHINE の下に直接新しいキーを作成することはできません。このため、HKEY_USER_SELECTABLE の下に作成した情報は、HKEY_LOCAL_MACHINE および HKEY_CURRENT_USER に共通した唯一のサブキーである、SOFTWARE サブキーに配置する必要があります。*

たとえば、HKEY_USER_SELECTABLE¥SOFTWARE¥MyCompany というキーは有効ですが、HKEY_USER_SELECTABLE¥MyCompany というキーは有効ではありません。

レジストリで複数行文字列の設定または取得を行う



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

複数行文字列の設定

複数行文字列を設定するには、**RegDBSetKeyValueEx** 関数を呼び出します。szValue パラメーターには、ヌル文字で区切られた、追加されるサブ文字列が必要です。メインの文字列の終わりには、その終わりを示すのに 2 つのヌル文字を追加する必要があります。今述べた文字列を作成するには、以下の手順を実行します。

1. 文字列を初期化して各サブ文字列をスペース文字で区切って含めます。
2. 各行のあとに、ヌル文字列の ASCII 値を文字列部分に割り当てます。
3. 文字列の終わりにヌル文字 (ASCII 値 0) を追加で割り当てます。

例:

```
svValue = "これは第 1 行目です。これは第 2 行目です。これは第 3 行目です。";
svValue[17] = 0;
svValue[35] = 0;
svValue[55] = 0;
svValue[56] = 0;
```

複数行文字列の取得

複数行文字列を取得するには、**RegDBGetKeyValueEx** 関数を呼び出します。サブ文字列は、メインの文字列の中でヌル文字で区切られて戻されます。サブ文字列を抽出するには、次のコードを使用して文字列を文字列リストに読み込んで **SdShowInfoList** ダイアログに表示させます。

```
// svValue は、RegDBGetKeyValueEx で読み込まれた文字列です。
listID = ListCreate (STRINGLIST);
if (listID != LIST_NULL) then
    StrGetTokens( listID, svValue, "" );
    SdShowInfoList (szTitle, szMsg, listID);
endif;
```

レジストリ関数の使い方



プロジェクト: *InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

InstallScript 言語には、次のようなレジストリへアクセスするためにインストールを簡単に構成できる多数のビルトイン関数があります : レジストリ キーの読み取り、作成、および削除、アンインストール用のレジストリ関連パラメーターの設定。

たとえば、アプリケーションで使われているレジストリキーを検出することでターゲット システムにあるアプリケーションが存在するかどうかを判断できることもあります。多数のアプリケーションは、データをレジストリ キー HKLM¥SOFTWARE¥CompanyName¥ProductName¥ProductVersion に格納します。レジストリ キーが存在するかどうかを確認するには、次の InstallScript スクリプトで **RegDBKeyExist** 関数を呼び出します:

```
// 他の RegDB 関数を呼び出す前に常にルートキーを設定する
RegDBSetDefaultRoot(HKEY_LOCAL_MACHINE);

if (RegDBKeyExist("Software¥¥ThisCo¥¥ThisApp") = 1) then
   MessageBox("ThisApp はシステムに存在します。", INFORMATION);
```

```
endif;
```

InstallScript と共に利用できるビルトインレジストリ関数について詳しく学ぶには、次をご覧ください。

- レジストリ関数
- レジストリ関連の特殊関数

レジストリからデータを読み込む



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

基本の MSI プロジェクト

AppSearch および RegLocator テーブルには、レジストリの値を読み込むことができます。上記の例で使用した RegisteredOwner の値を読み込むには、ダイレクト エディターを使用して AppSearch テーブルに次のレコードを追加します。

テーブル 6 - AppSearch メソッドの値

フィールド	値	コメント
プロパティ	REGISTERED_OWNER	パブリック プロパティにする必要があります。
Signature_	registry_sig	

次に、RegLocator テーブルに次のレコードを追加します :

テーブル 7 - RegLocator テーブルの値

フィールド	値	コメント
Signature_	registry_sig	上記の Signature_ と同じ
ルート	2	HKEY_LOCAL_MACHINE
キー	Software¥Microsoft¥Windows NT¥CurrentVersion	
名前	RegisteredOwner	
種類	2	レジストリ データ

AppSearch アクションの実行後、REGISTERED_OWNER プロパティには、レジストリから読み込んだデータが含まれます。値が見つからない場合、プロパティが未定義 (空) です。

AppSearch および RegLocator テーブルに関する詳細については、Windows Installer ヘルプ ライブラリを参照してください。

Windows Installer ベースのプロジェクトでは、データの検索に システム検索ウィザードを使用します。

InstallScript MSI プロジェクト

InstallScript では、**RegDBGetKeyValueEx** 関数を使用できます。たとえば、HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows NT¥CurrentVersion キーから RegisteredOwner の値を読み込むには、次のようなコードを記述します。

```
function readRegisteredOwner( )
    STRING svRegisteredOwner;
    NUMBER nvType, nvSize;
begin
RegDBSetDefaultRoot(HKEY_LOCAL_MACHINE);

RegDBGetKeyValueEx(
    "Software¥¥Microsoft¥¥Windows NT¥¥CurrentVersion",
    "RegisteredOwner",
    nvType,
    svRegisteredOwner,
    nvSize);

MessageBox(
    "登録された所有者は: " + svRegisteredOwner,
    INFORMATION);
end;
```

Windows Installer プロパティは、**MsiSetProperty** 関数を使って読み込んだ値と等しく設定することができます。

.ini ファイル データの変更



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、.ini ファイルを変更するための .ini ファイル関数 が含まれています。InstallScript プロジェクトで、これらの関数を使用できます。

初期化 (.ini) ファイルは、アプリケーションの次回の使用までの間において、情報の保存と取得を行うデータベースの役割を果たします。Boot.ini や Wininit.ini などの .ini ファイルは、オペレーティング システムで使用されます。[INI ファイル変更] ビューでは、ターゲット システム上で .ini ファイルに行う変更を指定することができます。ターゲット システムにあるすべての .ini ファイルを編集することができますが、システム .ini ファイルの変更は推奨しません。



タスク ***.ini ファイルを編集するには、以下の手順を実行します。***

1. .ini ファイル リファレンスを作成します。
2. .ini ファイルにセクションを追加します。
3. .ini ファイルにキーワードを追加します。

.ini ファイル リファレンスを作成するには、少なくとも 1 つのコンポーネントを作成している必要があります。.ini ファイル リファレンスを作成したときにコンポーネントがなかった場合は、[新しいコンポーネントの作成] ダイアログ が表示され、ここでコンポーネントを作成することができます。

.ini ファイルのリファレンスを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、.ini ファイルを変更するための .ini ファイル関数 が含まれています。InstallScript プロジェクトで、これらの関数を使用できます。

.ini ファイル変更作成の最初の手順は、編集するファイルへのリファレンスを作成することです。そのためには、編集するターゲット システム上のファイルの場所と名前が必要です。指定した場所にファイルがない場合、インストールがこのファイルを変更することはできません。



タスク ***.ini ファイルにリファレンスを作成するには、次の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[INI ファイルの変更]** をクリックします。
2. **[INI ファイル]** エクスプローラーを右クリックして、**[INI ファイルの追加]** をクリックします。

[INI ファイル] エクスプローラーに .ini ファイルの新しいリファレンスが追加されます。.ini ファイルの設定を右側のペインで構成します。各設定の詳細については、「.ini ファイルの設定」を参照してください。

.ini ファイルへのリファレンスを作成したら、セクションを .ini ファイルに追加する次のステップに進みます。

既存の .ini ファイルをインポートする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、.ini ファイルを変更するための .ini ファイル関数 が含まれています。InstallScript プロジェクトで、これらの関数を使用できます。

InstallShield では、プロジェクトに既存する .ini ファイルをインポートできます。.ini ファイルをインポートした後、必要に応じて [INI ファイルの変更] ビューを使って、ターゲット システムに加える変更を構成することができます。



タスク

既存する .ini ファイルをインポートするには、以下の手順に従います。

1. **[システム構成]** の下のビュー リストにある **[INI ファイルの変更]** をクリックします。
2. **[INI ファイル]** エクスプローラーを右クリックしてから、**[INI ファイルのインポート]** を選択します。**[開く]** ダイアログ ボックスが開きます。
3. インポートする .ini ファイルを選択してから、**[開く]** をクリックします。

InstallShield が .ini ファイルとそのセクション、キーワード、および値をプロジェクトにインポートします。

.ini ファイルのセクションを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、.ini ファイルを変更するための .ini ファイル関数 が含まれています。InstallScript プロジェクトで、これらの関数を使用できます。

編集する .ini ファイルを指定した後の次のステップは、そのファイル内で変更するセクションの指定です。.ini ファイルは、セクションに分割され、それぞれのセクションにはキーワードがあります。セクションは、**[SectionName]** のように角かっこ [ ] で囲まれて識別されます。



タスク ***.ini ファイルのセクションを指定するには、次の操作を実行します。***

1. **[システム構成]** の下のビュー リストにある **[INI ファイルの変更]** をクリックします。
2. 存在しない場合は、.ini ファイルへの参照を作成します。
3. **[INI ファイル]** エクスプローラーで、セクションを含める .ini ファイルを右クリックして、**[セクションの追加]** をクリックします。

[INI ファイル] エクスプローラーにセクションが追加されます。右側のペインでセクションの設定を構成します。各設定の詳細については、「.ini ファイルのセクションの設定」を参照してください。

.ini ファイル内のセクションを指定した後、キーワードを追加できます。

.ini ファイルのキーワードを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、.ini ファイルを変更するための .ini ファイル関数 が含まれています。InstallScript プロジェクトで、これらの関数を使用できます。

.ini ファイルのキーワードは、.ini ファイルの構造の最下位レベルです。キーワードには、アプリケーションの終了から次回の起動までの間に保持しておく必要のあるデータを格納します。

プロジェクトに .ini ファイルに追加してから 1 つ以上のセクションを設定した後、セクションにキーワードを追加して、キーワードのプロパティを構成できます。キーワードのプロパティには、キーワードの値のほかに実行するアクション (データ値の置換、既存のデータ値への追加など) が含まれます。



タスク ***.ini ファイルにキーワードを指定するには、次の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[INI ファイルの変更]** をクリックします。
2. **[INI ファイル]** エクスプローラーで、セクションを右クリックして、**[キーワードの追加]** をクリックします。

[INI ファイル] エクスプローラーにキーワードが追加されます。右側のペインでキーワードの設定を構成します。各設定の詳細については、「.ini ファイルのキーワードの設定」を参照してください。

.ini ファイルからデータを読み込む



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

キー名などのデータを .ini ファイルから読み取るには、.ini ファイルがどこにあるか(例、ネットワーク上)にかかわらず、**GetProfString** 関数を利用することができます。次のスクリプト例を参考にしてください:

```
/*

.ini ファイルの名前が Test.ini で、以下を含むと想定します:

[ProductSettings]
Key1=One
Key2=2
Key3=III
Key4=...

[OtherSettings]
Key1=Value1
Key2=Value2

*/

function OnBegin()
    STRING svKey1Value; //  GetProfString が入力
begin

// 単一値を読み込む
GetProfString(
    "¥¥¥¥Server¥¥Config¥¥Test.ini", // ファイル名。2 つの円記号を
実際の円記号の前につけます
    "ProductSettings", // 角括弧なしのセクション名
    "Key1", // キー名
    svKey1Value); // キー値をキャプチャする STRING 変数

MessageBox("Key1 の値は: " + svKey1Value, INFORMATION);

end;
```

.ini ファイルからすべてのキー名を取得する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

.ini ファイルのセクションからキー名を読み取るには、次のスクリプト使用します:

```
function OnBegin()
    STRING svAllKeyNames; //  GetProfString が入力
    LIST   listKeyNames;  // キー名を含む文字列リスト
begin

// 単一値を読み込む
GetProfString(
    "¥¥¥¥Server¥¥Config¥¥Test.ini", // ファイル名。2 つの円記号を
実際の円記号の前につけます
    "ProductSettings", // 角括弧なしのセクション名
    "", // キー名; ヌル文字列 "" はすべてのキー名を入手するの意
    svAllKeyNames); // キー名に保存する STRING 変数

listKeyNames = ListCreate(STRINGLIST);

// キー名を含む文字列を分離する
StrGetTokens(listKeyNames, svAllKeyNames, "");

SdShowInfoList("キー名", "こちらです:", listKeyNames);

// 必要であれば、リストの中のキー名をループして各キー値を
キー値

ListDestroy(listKeyNames);

end;
```

ODBC リソースの構成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

システム構成のより複雑な領域の 1 つに、ODBC ドライバー、データソース名 (DSN)、トランスレーターの設定があります。ODBC リソースは、すべての必須属性とともにシステムに正しく登録されている必要があります。また、ドライバーおよびトランスレーターの場合、インストール .dll ファイルなどの必要なファイルをインストールする必要があります。このプロセスは、ODBC リソース ビューを使って単純化できます。このビューでは、開発システムにインストールされているドライバー、データ ソース、およびトランスレーターを選択できます。

ODBC リソースのインストール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ODBC リソース] ビューには、すべての ODBC ドライバー、データソース名 (DSN)、およびシステムに登録されているトランスレーターが表示されます。DSN は、関連ドライバーの「子」として表示されます。エクスプローラーを展開すると、既存の ODBC リソースがすべて表示されます。

インストールに ODBC リソースを組み込むには、ビューの左上にある [ODBC リソース] ペインで名前の隣にあるチェック ボックスをクリックします。インストール プロジェクトでは、リソースを少なくとも 1 つの機能に割り当てる必要があります。そのあと、選択したリソースについて属性を設定することができます。

追加の ODBC リソースを含める



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

エクスプローラーのリストにない ODBC リソースをインストールすることもできます。これらを追加する場所は、リソースのタイプにより異なります。



タスク ***[ODBC リソース] ビューの [ODBC リソース] エクスプローラーにリストされていないドライバーを追加するには、以下の手順を実行します。***

1. **[ODBC リソース]** エクスプローラーで、**[ドライバーと DSN]** を右クリックして、**[新しいドライバー]** をクリックします。
2. ドライバーの新しい名前を入力するか、後で F2 キーを押して名前を変更します。
3. 新しいドライバーの隣りにあるチェック ボックスを選択して、インストールに含めます。



タスク ***[ODBC リソース] ビューの [ODBC リソース] エクスプローラーにリストされていないデータ ソースを追加するには、以下の手順を実行します。***

1. **[ODBC リソース]** エクスプローラーで、ドライバーを右クリックして、**[新しいDSN]** をクリックします。
2. DSN の新しい名前を入力するか、後で F2 キーを押して名前を変更します。
3. 新しい DSN の隣りにあるチェック ボックスを選択して、それをインストールに含めます。



メモ: *トランスレーターをリストに追加することはできません。トランスレーターは開発システムにインストールしてから選択する必要があります。*

ODBC リソースを機能に関連付ける



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

プロジェクトのほとんどのデータ同様に、ODBC リソースも機能に関連付ける必要があります。機能がターゲット システムにインストールされると、ODBC リソースは機能の一部としてインストールされます。

インストール用のドライバー、DSN、トランスレーターをインストールするように選択した後、[機能] リストが [ODBC リソース] エクスプローラーの右側で使用できます。ODBC リソースが属する機能のすべてを選択します。InstallShield により新しいコンポーネントが作成され、それが選択された各機能に関連付けられます。リソースは、複数の機能に関連付けられている場合でも一度だけインストールされます。しかし、関連付けられている機能がインストールされていない場合、リソースはインストールされません。

InstallScript MSI プロジェクトでは、その他の機能が存在しない場合は DefaultFeature にリソースが追加されます。基本の MSI プロジェクトでは、インストールに ODBC リソースを追加するときに機能が存在しない場合、[新しい機能の作成] ダイアログ ボックスが表示されます。このダイアログ ボックスは、機能の作成をプロンプトします。1 つの ODBC リソースに関連付けられている機能が 1 つだけの場合、別の機能を少なくとも 1 つ選択しない限り、その機能の選択を解除することはできません。

ODBC リソース属性の設定



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ODBC リソースが開発システムに登録されている場合、その現在の属性は、[ODBC リソース] ビューのプロパティリストに表示されます。ドライバー、データソース名 (DSN)、またはトランスレーターを選択すると、[プロパティ] ペインで属性を編集できます。

新しい属性エントリの追加と削除

トランスレーターに属性を追加することは、サポートされていません。



タスク ***新しい属性をドライバーまたは DSN に追加するには、以下の手順に従います:***

1. **[ODBC リソース]** ビューを開きます。
2. **[ODBC リソース]** エクスプローラーで、新しい属性を含めるドライバー、または DSN をクリックします。
3. プロパティ シートの最後の行をクリックして、**[プロパティ]** または **[値]** 列に適切な情報を追加します。



タスク ***ドライバーまたは DSN から属性を削除するには、以下の手順を実行します。***

1. **[ODBC リソース]** ビューを開きます。
2. **[ODBC リソース]** エクスプローラーで、削除する属性を含む ODBC リソースをクリックします。
3. プロパティ シートで、削除する行を右クリックしてから、[**削除**] をクリックします。

環境変数を使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallScript 言語には、環境変数の現在の値を読み出すための GetEnvVar 関数が含まれており、環境変数の作成が可能です。

環境変数は、インストールと共にターゲット システム上に設定できる名前と値の組み合わせで、アプリケーションおよび実行されているその他のプログラムによってアクセスできます。環境変数はレジストリに格納されます。

[環境変数] ビュー を使うと、インストールを使ってターゲット システム上に環境変数を作成、設定 (または変更) したり、システム上の環境変数を削除することができます。このビューで環境変数プロパティを指定することもできます。

環境変数の設定



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallScript 言語には、環境変数の現在の値を読み出すための GetEnvVar 関数が含まれており、環境変数の作成が可能です。



タスク

新しい環境変数を作成するか、または既存の環境変数を変更するには、以下の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[環境変数]** をクリックします。
2. **[環境変数]** を右クリックして、**[環境変数の追加]** をクリックします。
3. InstallShield は、NewEnvironmentVariable*X* (*X* は数字) を持つ新しい環境変数を追加します。変更、削除または作成する変数名を入力します。
4. 右側のペインで、環境変数の設定を構成します。各設定の詳細については、「環境変数の設定」を参照してください。

環境変数の例



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

```
/*****************************************************************¥
* 次のコードは、Windows の
*  全システムで環境変数を作成します。OnEnd イベント ハンドラー関数ブロック
* (またはその他の関数ブロック) を変更して、このサンプルコードを含めることができます。
*
*
* メモ: また、現在のユーザーがこのコードを使用するには、
* 管理者特権が必要です。
¥*****************************************************************/

 #define WM_WININICHANGE 0x001A
 #define HWND_BROADCAST 0xffff
 NUMBER nResult;
 STRING szKey, szEnv;
 WPOINTER pEnv;

 begin

    szKey = "SYSTEM¥¥CurrentControlSet¥¥Control¥¥Session Manager¥¥Environment";
    RegDBSetDefaultRoot(HKEY_LOCAL_MACHINE);
    nResult = RegDBSetKeyValueEx(szKey, "Fame", REGDB_STRING, "C:¥¥Test", -1);
    if (nResult < 0) then
       MessageBox("環境変数の設定に失敗しました", WARNING);
    else
       MessageBox("環境変数は正常に設定されました", INFORMATION);
       // レジストリをすべてのアプリケーションにフラッシュ。
       szEnv = "環境";
       pEnv = &szEnv;
       SendMessage (HWND_BROADCAST, WM_WININICHANGE, 0, pEnv );
    endif;
    // RebootDialog("", "", SYS_BOOTMACHINE);
 end;


/*****************************************************************¥
* 次のコードは、Windows の
*  現在のユーザーの環境変数を作成します。OnEnd イベント ハンドラー関数ブロック
* (またはその他の関数ブロック) を変更して、このサンプルコードを含めることができます。
*
```

```
*
* メモ: また、現在のユーザーがこのコードを使用するには、
* 管理者特権が必要です。
¥*********************************************************************/

  #define WM_WININICHANGE 0x001A
  #define HWND_BROADCAST 0xffff

  NUMBER nResult;
  STRING szKey, szEnv;
  WPOINTER pEnv;

  begin
     szKey="環境";
     RegDBSetDefaultRoot(HKEY_CURRENT_USER);
     nResult=RegDBSetKeyValueEx(szKey,"Fame",REGDB_STRING,"C:¥¥test",-1);
     if (nResult < 0) then
        MessageBox("環境変数の設定に失敗しました",WARNING);
     else
        MessageBox("環境変数は正常に設定されました", INFORMATION);
        // レジストリをすべてのアプリケーションにフラッシュ。
        szEnv = "環境";
        pEnv = &szEnv;
        SendMessage (HWND_BROADCAST, WM_WININICHANGE, 0, pEnv );
     endif;
     //RebootDialog("","",SYS_BOOTMACHINE);
  end;
```

XML ファイルの変更



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

製品に関連する設定を保存する .xml ファイルの変更が必要な場合、または web.config および machine.config などの標準構成ファイルの変更が必要な場合があります。InstallShield には、実行時にターゲット システムで XML ファイルを変更する機能が搭載されています。XML ファイルは、インストールの一部であっても、またターゲット システム上に既に存在するファイルであっても構いません。

インストール中に XML ファイルを変更する方法については、ドキュメントのこのセクションを参照してください。

XML に関する詳細は、次の Web サイトをご覧ください。

- World Wide Web Consortium (http://www.w3.org)

- W3 Schools (http://w3schools.com)



ヒント: *XML ファイルは、InstallShield 製品の機能の他の場所でもサポートされています。InstallShield の [システム検索] 機能では、指定した XML ファイルで、要素の属性値、内容、または有無を検索することができます。詳細については、「XML データの検索」を参照してください。*

XML ファイルの変更の概要



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

デザイン時のタスク

[XML ファイル変更] ビューでは、実行時に XML ファイルに加えられる変更を定義することができます。変更を定義するときの一般的な手順は、以下の通りです。

1. 変更する XML ファイルがインストールの一部の場合、[ファイルとフォルダー] ビューでプロジェクト内にあるコンポーネントにベースの XML ファイルを追加します。
2. [XML ファイルの変更] ビューで、変更するファイルの XML ファイルの参照を作成します。
3. ターゲット マシン上の XML ファイルの保存場所を指定します。
4. XML ファイルの変更を含める機能を指定します (複数可)。ここで、ステップ 1 でコンポーネントを追加した機能と同じ機能を指定することもできますし、別の機能を指定することもできます。
5. 実行時に発生する変更を構成します。

場合によって、MSXML 再配布可能ファイルをプロジェクトに追加する必要があります。詳細については、「XML ファイルの変更の実行時の要件」を参照してください。

実行時の動作

[XML ファイル変更] ビューでは、実行時の動作を次のように構成することができます。

- 名前空間のマッピングを XML ファイルに追加し、名前空間のプレフィックスを要素と属性に追加する。
- 指定された要素が存在しない場合、それを作成する。
- 存在する場合、要素を変更するが、存在しない場合は、作成しない。
- 指定された XPath 式に一致する XML ファイルの最初の要素のみを変更するか、すべての一致するインスタンスを変更する。

- アンインストール時に要素を削除する。
- 要素にコンテンツを設定する。
- インストールまたはアンインストール時 (または両方) に、新しい属性を作成する。
- インストールまたはアンインストール時 (または両方) に、既存の属性を削除する。
- インストールまたはアンインストール時 (または両方) に、文字列を既存の属性値に付加する。
- 基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、およびトランスフォーム プロジェクトの場合: 実行時に、要素、属性、属性値、および要素のコンテンツの Windows Installer のプロパティを適切な値で置換する。
- InstallScript プロジェクトの場合: 実行時に、要素、属性、属性値、および要素のコンテンツの InstallScript の文字列変数を適切な値で置換する。

XML ファイルの変更を含むインストールがターゲット システムで実行されたとき、MSXML が XML ファイルを解析し、構成した変更に関連付けられている XPath 式を実行します。[XML ファイルの変更] ビューにある [詳細] タブで、XML ファイルのターゲット システムで実行される XPath 式を表示することができます。

MSXML が XPath 式に一致する XML ファイルの領域を検出したとき、[XML ファイルの変更] ビューで構成された変更が加えられます。

基本的な XPath 式の作成方法については、「XPath 式を使って、XML ファイル内の XML データを検索する」を参照してください。

メモ: *実行時に XML ファイルが存在しなく、XML ファイルの [詳細] タブで* ***[XML ファイルが存在しない場合、常に作成]*** *チェック ボックスが選択されているとき、XML ファイルが [XML ファイルの変更] ビューで構成された XML データで作成される。*

XML ファイルの変更の実行時の要件

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

XML ファイルの変更の実行時機能には、ターゲット マシンに Microsoft XML Core Services 3.0 (MSXML) 以降が搭載されいることが必要です。

XML ファイルの変更を含むインストールがターゲット システムで実行されたとき、MSXML が XML ファイルを解析し、構成した変更に関連付けられている XPath 式を実行します。MSXML が XPath 式に一致する XML ファイルの領域を検出したとき、[XML ファイルの変更] ビューで構成された変更が加えられます。

InstallShield には、MSXML の異なるバージョンについてそれぞれ再配布可能ファイルが用意されています。ターゲット システムに MSXML がない、または適切なバージョンの MSXML ことが考えられる場合、[再配布可能ファイル] ビュー (基本 MSI プロジェクトと InstallScript MSI プロジェクト) または [前提条件] ビュー (InstallScript プロジェクト) のいずれかで、プロジェクトに適切な MSXML 再配布可能ファイルを追加することができます。

MSXML についての詳細は、「MSDN ライブラリ」を参照してください。

XML ファイル リファレンスの作成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

実行時に変更する XML ファイルを追加する場合、プロジェクトで [ファイルとフォルダー] ビューを使用して、ベースの XML ファイルをコンポーネントに追加することをお勧めします。次いで、その XML ファイル リファレンスを [XML ファイル変更] ビューに追加し、実行時にファイルに加える変更を指定します。XML ファイル リファレンスを追加するときの最も簡単な方法は、ファイルを [XML ファイルの変更] ビューにインポートする方法です。



重要: *[XML ファイルの変更] ビューは、XML ファイルにあるすべてのノードについてノードを表示するようにはデザインされていません。パフォーマンス速度を向上させるために、[XML ファイルの変更] ビューでは、ベースの XML ファイルとは異なる設定のみが表示されます。*

- *変更する XML ファイルがインストールの一部である場合、[XML ファイル変更] ビューには、XML ファイルが実行時にインストールされた後に追加、変更、または削除されるノード、またはノード セットのみが表示されます。*
- *変更する XML ファイルがターゲット システムに既に存在するファイルの場合、[XML ファイル変更] ビューには、実行時に追加、変更、または削除されるノードおよびノード セットのみが表示されます。*

したがって、ファイルをインポートするとき、実行時に変更する XML ファイル内のノードのみをインポートします。

[XML ファイルの変更] ビューでは、新しい要素が XML ファイルに表示されるときの順番を指定することはできませんので注意してください。従って、要素が特定の順番で表示されている必要がある製品の場合、実行時に変更するノードのみをインポートすると、潜在的な問題を回避することができます。

タスク ***XML ファイルを [XML ファイルの変更] ビューへインポートするには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーを右クリックし、**[インポート]** をクリックします。**XML の設定のインポート ウィザード**が開きます。
3. 変更する XML ファイルのノードのみを選択して、ウィザードのパネルを完成します。
4. **[インポート]** をクリックします。

インポートした XML ファイルの新しいノードが追加されます。XML ファイル ノードには、ウィザードで選択したノードがそれぞれ含まれています。各ノードは、実行時に発生する XPath クエリを示します。[XML ファイルの変更] ビューでインポートした XML ファイルの新しいコンポーネントも追加されます。

XML ファイルの設定を構成して、要素、属性、コンテンツを任意に変更することができます。

ヒント: *XML ファイルは、[XML ファイル] エクスプローラーを右クリックして、[新規ファイル] をクリックしても、追加することができます。ほとんどの場合、サードパーティの XML エディターまたはテキスト エディターを利用して XML ファイルを作成する方法が一般的です。次に、このファイルを [ファイルとフォルダー] ビューからプロジェクトに追加します。次いで、ファイルを上記の手順にしたがってインポートし、実行時にファイルに加える変更を構成します。*

XML ファイルの場所を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイルの変更] ビューに XML ファイル リファレンスを追加すると、その XML ファイルの場所を指定する必要があります。XML ファイルがインストールの一部の場合、その場所は、[ファイルとフォルダー] ビューでプロジェクトにベースの XML ファイルを追加したときに指定したインストール先に一致していなければなりません。

タスク ***ターゲットマシン上の XML ファイルの場所を指定するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、場所を指定する XML ファイルをクリックします。
3. **[全般]** タブをクリックします。

4. **[XML ファイルのインストール先]** 領域で、**[参照]** ボタンをクリックします。**[ディレクトリの参照]** ダイアログ ボックスが開きます。
5. **[インストール先ディレクトリ]** ボックスで、場所を選択します。
6. **[OK]** をクリックします。

[ターゲットマシン上の XML ファイルの保存場所を指定します] ボックスに指定したインストール先が追加されます。

XML ファイルの変更のリファレンスを機能に関連付ける



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイルの変更] ビューに XML ファイル リファレンスを追加すると、そのリファレンスの新しいコンポーネントが自動的に作成されます。コンポーネントは、またデフォルトで、機能一覧の最上位にある機能にも関連付けられます。この関連付けを変更する場合は、[XML ファイルの変更] ビューを使用します。



メモ: *XML ファイルのリファレンスを追加したとき、プロジェクトに機能が存在しない場合、新しい機能を作成することができる [新しい機能の作成] ダイアログ ボックスが表示されます。*



タスク ***プロジェクトで XML ファイルの変更のリファレンスを機能に関連付けるには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、コンポーネントを機能に関連付ける XML ファイルをクリックします。
3. **[全般]** タブをクリックします。
4. **[XML ファイルが属する機能を選択します]** ボックスで、選択した XML ファイルの変更のリファレンスを含める機能のチェック ボックスを選択します(複数可)。選択する機能は、[ファイルとフォルダー] ビューでプロジェクトに追加したベースの XML ファイルを含む機能を選択することもできますし、別の機能を選択することもできます。

エンドユーザーが XML ファイル変更を含む機能のインストールを選択した場合、その XML ファイルの変更が実行時に実行されます。

ルート要素の追加



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイルの変更] ビューで新しい XML ファイルのリファレンスを追加するとき、ルート要素を追加することができます。XML ファイルに含めることができるルート要素は、1 つだけですので注意してください。



タスク ***ルート要素を XML ファイルに追加するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、新しいルート要素を追加する XML ファイルを右クリックして、**[新しいルート要素]** をクリックします。

新しいルート要素が追加されます。右のペインにあるタブを利用して、設定を構成します。



ヒント: *ルート要素および 1 つ以上のレベルのサブ要素を一度に追加するには、ルート要素の名前とサブ要素をそれぞれスラッシュで区切って入力します。たとえば、***Root** *というルート要素と、***Root** *要素の下にある* **Sub** *要素と、***Sub** *要素の下の* **Sub2** *要素を追加するには、次のように入力します:*

Root/Sub/Sub2

エクスプローラーで自動的に展開されます。

新しい要素の追加



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

新しい要素は、XML ファイルのルート要素または任意の要素に追加することができます。



タスク ***新しい要素を追加するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、新しい要素を追加する XML ファイルを右クリックして、**[新しい要素]** をクリックします。

新しい要素が追加されます。要素の名前を必要に応じて変更し、右のペインに表示されているタブを使って、設定を構成します。



ヒント: *要素とサブ要素を [XML ファイルの変更] ビューのルート要素の下に追加するとき、要素名の完全 XPath 式を入力することができます。完全 XPath は、エクスプローラーで自動的に拡張します。*

*ルート要素および 1 つ以上のレベルのサブ要素を一度に追加するには、ルート要素の名前とサブ要素をそれぞれスラッシュで区切って入力します。たとえば、**Root** というルート要素と、**Root** 要素の下にある **Sub** 要素と、**Sub** 要素の下の **Sub2** 要素を追加するには、次のように入力します:*

Root/Sub/Sub2

各要素がそれぞれのノードと共にエクスプローラーで自動的に展開されます。

要素を .NET 構成ファイルに追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

web.config ファイルで製品について .NET 構成の設定を定義する場合、これらの設定を [XML ファイルの変更] ビューで指定することができます。



タスク ***要素を .NET 構成ファイルに追加するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、構成要素を追加する XML ファイルを右クリックし、**[定義済み要素の追加]** をクリックします。次いで、**[.NET 構成ファイル]**、**[Web 構成ファイル]** と順番にポイントし、表示された一連の設定をポイントして、適切なコマンドをクリックします。

エクスプローラーに、新しい要素が追加されます。

.NET 構成ファイルの異なる要素および属性に関する詳しい参照情報は、MSDN Web サイトの「Configuration File Schema (構成ファイル スキーマ)」(http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpgenref/html/gngrfnetframeworkconfigurationfileschema.asp) をご覧ください。

属性を要素に追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

XML ファイルで、属性を要素に追加して、属性を実行時に作成するかどうか、もしくは、削除するかどうか、または、実行時に属性値を既存の値に追加するかどうかを指定することができます。インストールまたはアンインストール、もしくはその両方で実行するタスクをスケジュールすることもできます。

追加した属性がターゲット マシン上にある XML ファイル内に既に存在するが、属性の値が [XML ファイルの変更] ビューで指定した値と異なる場合、その値はインストールの実行時に更新されます。



タスク

属性を XML 要素に追加するには、以下の手順に従います。

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、属性を追加する XML ファイルをクリックします。
3. **[全般]** タブをクリックします。
4. **[追加]** ボタンをクリックします。

新しい行が、属性テーブルに [属性]、[値]、[操作]、および [スケジュール] 列にデフォルト値が入った状態で追加されます。必要に応じて設定を変更します。それぞれの列についての詳しい情報は、「XML 要素の [全般] タブ」を参照してください。



ヒント: *コンポーネントがアンインストールされたとき XML ファイルがそのままターゲット システムに残るようにインストールを構成する場合、属性テーブルでインストール/アンインストールの属性の組み合わせを作成したほうが良い場合があります。*

たとえば、インストール中に **key** *属性を追加して、アンインストール中にその同じ* **key** *属性を削除する場合、* **key** *属性を持つ 2 つの行を属性テーブルに追加します。つまり、片方の行をインストールにスケジュールして、もう 1 つの行をアンインストールにスケジュールします。*

属性の値を編集する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

InstallShield では、属性が実行時にターゲット マシンに存在する場合、その属性の値を編集することができます。属性を編集するには、まず属性を [XML ファイルの変更] ビュー内の要素に追加してから、その属性の値を適宜構成します。属性値の編集方法は、属性値を追加する方法と同じです。詳細については、「属性を要素に追加する」を参照してください。

実行時に、属性に異なる値が存在する場合、その値は [XML ファイルの変更] ビューで構成した値に置き換えられます。

コンテンツを要素に追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

XML 要素に、テキスト コンテンツを含めることができます。たとえば、次の XML コードでは、**feature** は、要素 **product** のコンテンツです:

```
<product>feature</product>
```



タスク

コンテンツを [XML ファイルの変更] ビューで一覧表示されている要素の 1 つに追加するには、以下の手順に従います。

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、コンテンツに追加する要素を選択します。
3. **[詳細]** タブを選択します。
4. **[要素のコンテンツを設定する]** チェック ボックスを選択します。

5. **[コンテンツ]** ボックスで、テキスト コンテンツを入力します。

   この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定の隣にある省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

XPath 式を使って、XML ファイル内の XML データを検索する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

実行時に、インストールが新しい要素または属性を XML ファイルに追加したり、または XML ファイルに他の変更を実行したりするために、MSXML は [XML ファイルの変更] ビューで定義された XPath 式を使って、XML ファイルの中を移動し、変更が必要な箇所を見つけます。

XPath 式の書き方については、次の Web サイトをご覧ください:

- World Wide Web Consortium (http://www.w3.org)
- W3 Schools (http://w3schools.com)

以下のセクションは、基本的な XPath 式を作成するときの例です。

例 1: 2 つの要素を追加する

例 1 では、インストール時に、次の XML 文書に太線が追加されます:

```
<?xml version="1.0" encoding="UTF-8"?>
<Books>
  <Biographies>
    <Book Author="Bill Smith" Copyright="2007">Bill's Great Biography</Book>
    <Book Author="John Smith" Copyright="2006">John's Great Biography</Book>
  </Biographies>
</Books>
```

これらの要素を追加するために、次の XPath 式が [XML ファイルの変更] ビューの Biographies ノードの下に追加されます:

```
Book[@Author="John Smith"]
Book[@Author="Bill Smith"]
```

次のスクリーン ショットは、[XML ファイルの変更] ビューにある設定です。

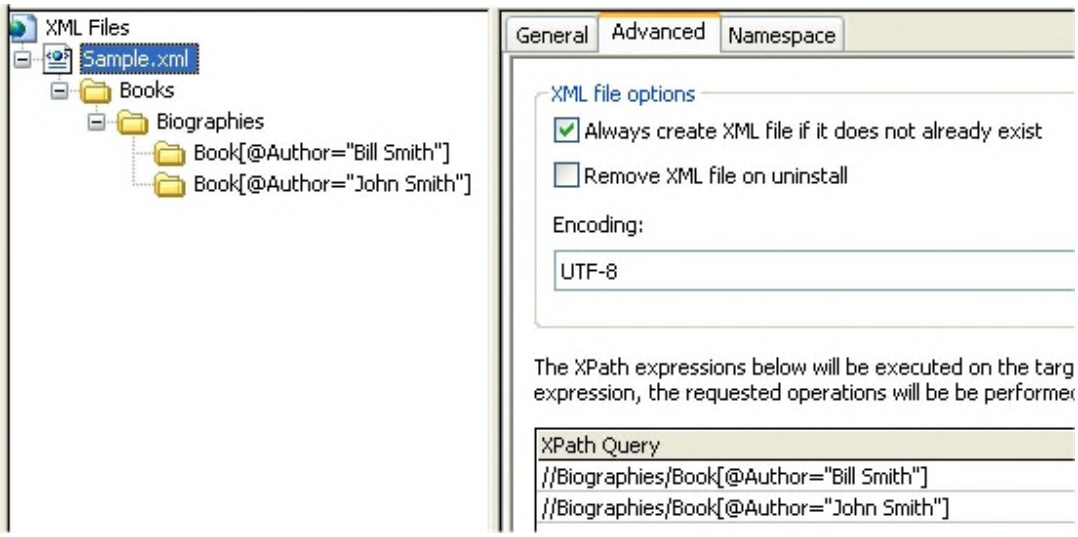


図 1: Sample.xml ファイルの設定

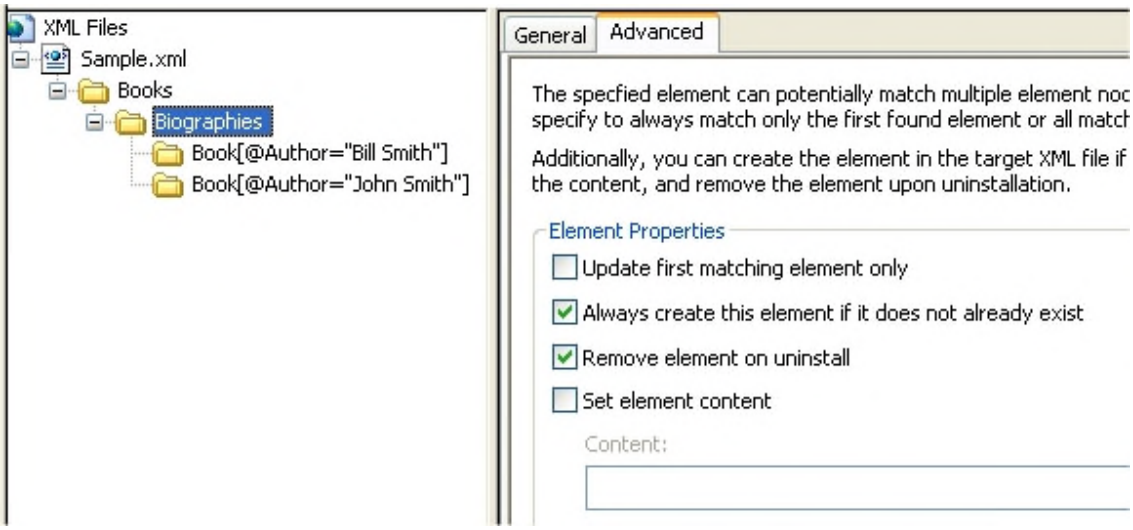


図 2: 追加される子ノードが含まれている Biographies ノードの設定

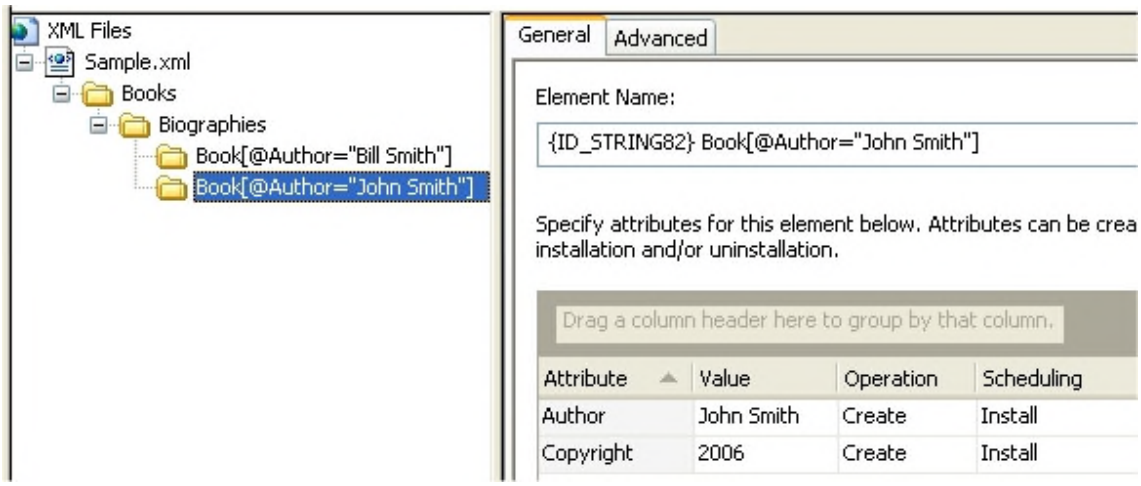


図 3: 追加される子ノードの 1 つの全般設定

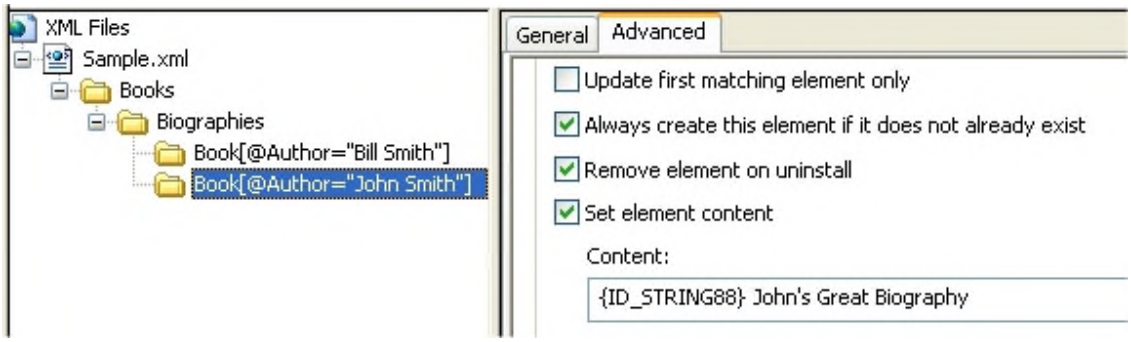


図 4: 追加される子ノードの 1 つの詳細設定

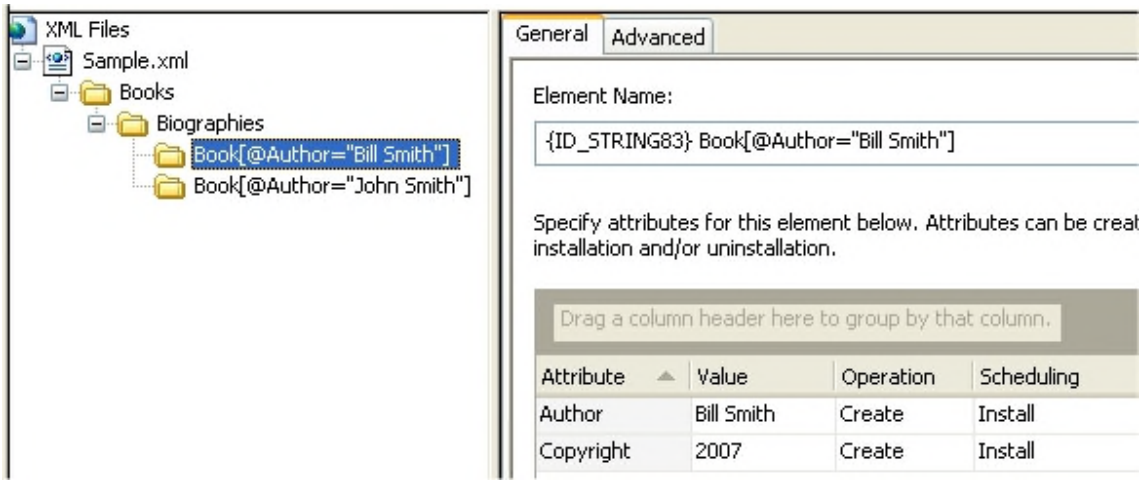


図 5: 追加される子ノードの 1 つの全般設定

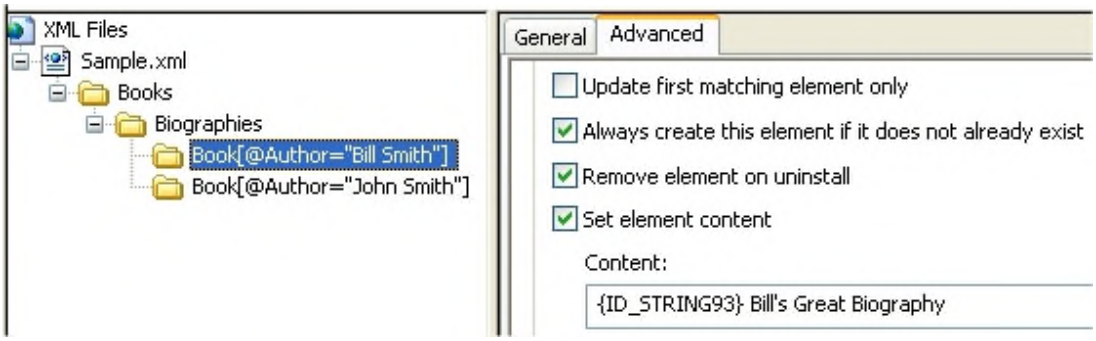


図 6: 追加される子ノードの 1 つの詳細設定

例 2: 要素に属性を追加する

例 2 では、インストール時に、太字属性が次の XML 文書に追加されます:

```
<?xml version="1.0" encoding="UTF-8"?>
<Books>
  <Biographies>
    <Book Author="Bill Smith" Copyright="2007" Publisher="Bill & John's Publish Co.">Bill's Great Biography</Book>
    <Book Author="John Smith" Copyright="2006" Publisher="Bill & John's Publish Co.">John's Great Biography</Book>
  </Biographies>
</Books>
```

次のスクリーン ショットは、[XML ファイルの変更] ビューにある設定です。

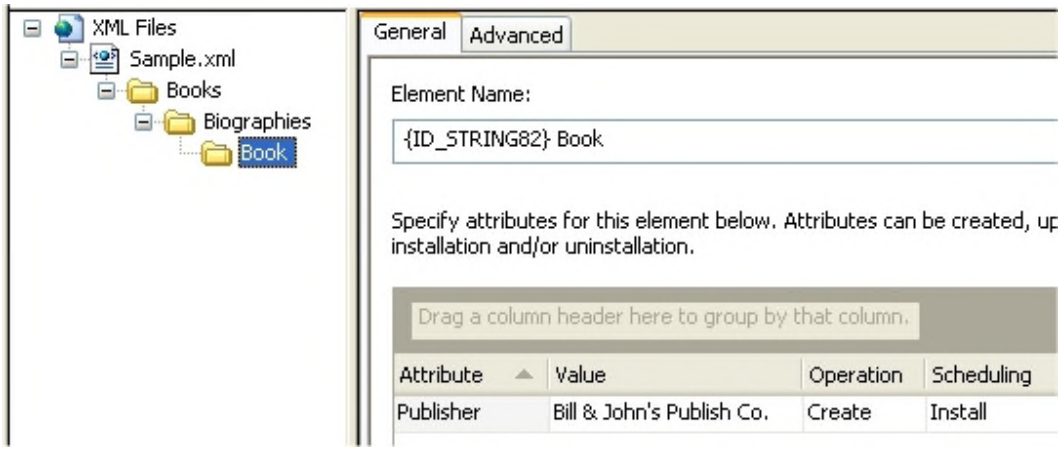


図 7: Book ノードの設定

例 3: 属性の値を更新する

例 3 では、インストール時に、2006 Copyright (次の XML 文書の中で太字で表示されています) という値がある属性が検索され、2007 Copyright という値で置き換えられます:

```
<?xml version="1.0" encoding="UTF-8"?>
<Books>
  <Biographies>
    <Book Author="Bill Smith" Copyright="2007" Publisher="Bill & John's Publish Co.">Bill's Great Biography</Book>
    <Book Author="John Smith" Copyright="2006" Publisher="Bill & John's Publish Co.">John's Great Biography</Book>
  </Biographies>
```

</Books>

これらの値を更新するために、次の XPath 式が [XML ファイルの変更] ビューの Biographies ノードの下に追加されます:

Book[@Copyright="2006"]

次のスクリーン ショットは、[XML ファイルの変更] ビューにある設定です。

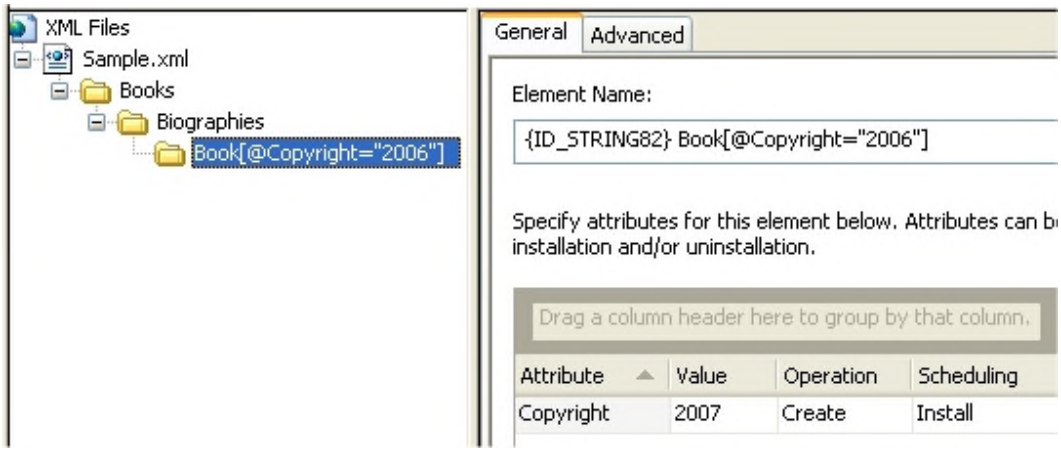


図 8: 更新される属性の全般設定

Windows Installer のプロパティを使用して、XML ファイルを動的に変更する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、およびトランスフォーム プロジェクトの場合、Windows Installer プロパティを使って、XML 要素、属性、属性値、および要素のコンテンツを指定することができます。実行時に、Windows Installer は **MsiFormatRecord** を使ってプロパティの値を解決し、それを XML ファイルのデータとして使用します。これにより、エンドユーザーがダイアログで入力したデータや、インストール時に判別された他の構成情報を製品の XML ファイルで使用できるようになります。

たとえば、インストールにエンドユーザーが SQL Server を選択することができる [SQL ログイン] ダイアログが含まれていると想定します。エンドユーザーが選択したサーバーの名前は、通常 IS_SQLSERVER_SERVER プロパティに格納されます。XML ファイルにエンドユーザーが選択した SQL サーバーの名前を含める場合、[XML ファイル変更] ビューで XML ファイルの変更を構成する際に、このプロパティをかっこで囲んで使用します (例、[IS_SQLSERVER_SERVER])。実行時に、このプロパティは Windows Installer によって自動的に関連付けられている値に置換されます。



ヒント: *プロパティの値が、整形式の XML 文書になる場合のみ、要素、属性、または要素のコンテンツに Windows Installer プロパティを使用してください。プロパティの値が、XML 文書で構文エラーになる場合、製品またはインストールが期待通りに動作しない場合があります。*

たとえば、要素の名前を [INSTALLDIR] *(値がターゲット システム上のパスの Windows Installer) に設定した場合、実行時の結果は、円記号を含む要素名です。要素名に、円記号は使用できません。円記号がある要素名を含む XML 文書は無効です。*



メモ: *[XML ファイル変更] ビュー内から XML ファイルをテストする場合、XML データに使用している Windows Installer プロパティは適切な値で置き換えられません。テストに関する詳細については、「XML ファイルに加えられるインストール時の変更をテストする」または「XML ファイルに加えられるアンインストール時の変更をテストする」を参照してください。*

例

以下は、エンドユーザーが [SQL ログイン] ダイアログで選択した SQL Server の名前を XML ファイル内のある要素のコンテンツとして使用するための手順です。[XML ファイル変更] ビューの要素、属性、属性値、要素のコンテンツのプロパティの設定でハードコード化されている値を代入することができます。このビューでプロパティを指定するとき、プロパティは角かっこで囲み、プロパティ名はすべて大文字にする必要があります (例、**[MYPROPERTY]**)。



タスク

エンドユーザーが選択した SQL Server の名前を XML 要素のコンテンツとして使用するには、以下の手順に従います:

1. SQL 接続と関連付けられている SQL スクリプトが未構成の場合、それらをまず構成し、サーバー プロパティ名の名前を決定します:
   a. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
   b. SQL 接続と SQL スクリプトを適宜追加します。手順については、「SQL サポートの構成」を参照してください。
   c. **[SQL スクリプト]** エクスプローラーで、SQL 接続をクリックします。
   d. **[詳細]** タブをクリックします。
   e. **"ターゲット サーバー プロパティ名"** 設定で選択された名前を確認してください。通常、デフォルト値は **IS_SQLSERVER_SERVER** です。
2. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
3. **[XML ファイル]** エクスプローラーで、コンテンツに追加する要素を選択します。
4. **[詳細]** タブを選択します。
5. **[要素のコンテンツを設定する]** チェック ボックスを選択します。
6. **[コンテンツ]** ボックスに、1e で選択したファイルを入力し、それをかっこで囲みます。例:

   [IS_SQLSERVER_SERVER]
7. リリースをビルドします。

InstallScript テキスト置換を使用して、XML ファイルを動的に変更する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript プロジェクトの場合、XML 要素、属性、属性値、および要素のコンテンツにテキスト置換文字列変数を利用することができます。実行時に XML ファイルの変更が発生すると、InstallScript ランタイム コードは **TextSubSubstitute** 関数を使って、文字列変数を適切な値で置き換えます。これにより、エンドユーザーがダイアログで入力したデータや、インストール時に判別された他の構成情報を製品の XML ファイルで使用できるようになります。

ヒント: *プロパティの値が、整形式の XML 文書になる場合のみ、要素、属性、または要素のコンテンツにテキスト置換を使用してください。プロパティの値が、XML 文書で構文エラーになる場合、製品またはインストールが期待通りに動作しない場合があります。*

たとえば、要素の名前を 1 つまたは複数のスペースを含むテキストで置換されるテキスト置換文字列変数に設定すると、エンドユーザーが製品またはインストールを使用するときに問題が発生する場合があります。要素名に、スペースは使用できません。したがって、スペースがある要素名を含む XML 文書は無効になります。

メモ: *[XML ファイル変更] ビュー内から XML ファイルをテストする場合、XML データに使用しているテキスト置換文字列変数は適切な値で置き換えられません。テストに関する詳細については、「XML ファイルに加えられるインストール時の変更をテストする」または「XML ファイルに加えられるアンインストール時の変更をテストする」を参照してください。*

例

以下は、エンドユーザーが [SQL ログイン] ダイアログで選択した SQL Server の名前を XML ファイル内のある要素のコンテンツとして使用するための手順です。[XML ファイル変更] ビューの要素、属性、属性値、要素のコンテンツのテキスト置換文字列変数の設定でハードコード化されている値を代入することができます。このビューで指定する文字列変数は山かっこで囲む必要があります; 例、**<MYPROPERTY>**。指定する文字列変数は大文字と小文字を区別します。

タスク

エンドユーザーが選択した SQL Server の名前を XML 要素のコンテンツとして使用するには、以下の手順に従います:

1. SQL 接続と関連付けられている SQL スクリプトが未構成の場合、それらを構成します:
   a. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
   b. SQL 接続と SQL スクリプトを適宜追加します。手順については、「SQL サポートの構成」を参照してください。
2. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
3. **[XML ファイル]** エクスプローラーで、コンテンツに追加する要素を選択します。
4. **[詳細]** タブを選択します。
5. **[要素のコンテンツを設定する]** チェック ボックスを選択します。

6. **[コンテンツ]** ボックスで、次のように入力します:

   <MYPROPERTY>

7. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。

8. SQL Server コントロールを含む [SQL ログイン] ダイアログの OnSQLLogin イベントでダイアログ コードを見つけ、InstallScript 関数 **TextSubSetValue** の呼び出しを追加します。例:

```
// ダイアログ ボックスを表示する (接続名なし)
// ダイアログを入れ替える場合、コメントアウト
nResult = SQLServerSelectLogin2( szConnection, szServer, szUser, szPassword, bWinLogin, szDB, FALSE, TRUE );
TextSubSetValue("<MYPROPERTY>", szServer, TRUE);
```

9. リリースをビルドします。

要素内で予約されている文字 (<、>、&、'、") を使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

XML 要素内で、"小なり" 記号 (<) などの予約されている文字を使用すると、MSXML パーサーが、実行時にターゲット システムでそれを定義済みエンティティ (<) に変換します。このエンティティの代わりに "小なり" 記号を使用すると、XML パーサーが新しい要素の始まりとして解釈し、それが無効な XML コードと見なされるため、エラーが発生する原因となります。結果の XML ファイルを Internet Explorer のようなブラウザーで開くと、エンティティではなく、この文字が、XML 要素内で表示されます 。

次のテーブルは、予約されている XML 文字と、それらに対応するエンティティの一覧です。テーブルにはまた、それぞれの予約されている文字について、実行時にエンティティによって置換されるかどうかが示されています。

テーブル 8 ・ 予約されている文字と定義済みエンティティ

文字	エンティティ	説明	注
<	<	より小さい	この文字は、XML 要素の始まりを示すために予約されているため、XML 要素のコンテンツとして使用することはできません。 この文字は、実行時に自動的にエンティティによって置換されます。
>	>	より大きい	この文字は、実行時に自動的にエンティティによって置換されます。

テーブル 8 ・ 予約されている文字と定義済みエンティティ (続き.)

文字	エンティティ	説明	注
&	&	And	この文字は、エンティティの始まりを示すとき以外は、XML 要素のコンテンツとしては使用できません。 この文字が、XML 要素のコンテンツとして使用されているときに、エンティティの始まりを示すものではない場合、実行時にそのエンティティによって自動的に置換されます。
'	'	アポストロフィ	この文字は、実行時に自動的にエンティティによって置換されません。
"	"	引用符	この文字は、実行時に自動的にエンティティによって置換されません。

XML ファイルで名前空間を使用する



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

XML 名前空間を利用して、要素名の競合を避けることができます。InstallShield で XML ファイルの変更を指定するとき、XML ファイルで宣言する名前空間のマッピングを指定して、任意のファイルの要素について、名前空間プレフィックスを指定することができます。



***ヒント・** XML 設定のインポート ウィザードを使用して、XML ファイルを [XML ファイルの変更] ビューへインポートすると、ファイルに宣言された名前空間がインポートされます。*

XML ファイルに名前空間のマッピングを宣言する



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*

- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイルの変更] ビューで、XML ファイルに名前空間のマッピングを宣言することができます。



タスク ***XML ファイルに名前空間のマッピングを宣言するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、名前空間を含める XML ファイルをクリックします。
3. **[名前空間]** タブをクリックします。
4. テーブル内の行をクリックして、新しい名前空間を追加します。
5. **[プレフィックス]** 列で、対応する名前空間に関連付けられているすべての要素に使用されるプレフィックスを入力します。
6. **URL**1 列で、URL、またはインターネット リソースを識別する文字列を入力します。

名前空間プレフィックスを要素に追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

XML ファイルに名前空間のマッピングを宣言すると、そのファイル内の要素を、その要素に対応するプレフィックスを追加することで、名前空間に関連付けることができます。



タスク ***名前空間プレフィックスを要素に追加するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、プレフィックスを追加する要素を右クリックして、以下のいずれかを実行します。
   - プレフィックスを選択した要素にのみ追加する場合、**[名前空間プレフィックス]** をポイントしてから、適切な名前空間マッピングをクリックします。
   - プレフィックスを要素とそのすべてのサブ要素に追加する場合、**[名前空間プレフィックス (すべてのサブ要素を含む)]** をポイントしてから、適切な名前空間マッピングをクリックします。

プレフィックスが、[XML ファイル] エクスプローラー内の要素 (および、該当する場合、そのすべてのサブ要素) に追加されます。



ヒント: *また、要素を右クリックして、[名前の変更] をクリックし、要素名に前にプレフィックスとコロン (:) を追加しても、名前空間プレフィックスを追加することができます。*

名前空間プレフィックスを属性に追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

XML ファイルに名前空間のマッピングを宣言すると、そのファイル内の属性を、その属性に対応するプレフィックスを追加することで、名前空間に関連付けることができます。



タスク ***名前空間プレフィックスを属性に追加するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、属性を含む要素をクリックします。
3. 右側のペインで **[全般]** タブをクリックします。
4. グリッドで、プレフィックスを追加する属性をダブルクリックしてから、属性名の始まりにカーソルを配置します。
5. 属性名の前にプレフィックスとコロン (:) を追加します。

名前空間プレフィックスを属性から削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

- *トランスフォーム*



タスク ***名前空間プレフィックスを属性から削除するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、属性を含む要素をクリックします。
3. 右側のペインで **[全般]** タブをクリックします。
4. グリッドで、プレフィックスを削除する属性をダブルクリックしてから、属性名の始まりにカーソルを配置します。
5. 属性名の前からプレフィックスとコロン (:) を削除します。

名前空間プレフィックスを要素から削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*



タスク ***名前空間プレフィックスを要素から削除するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、プレフィックスを削除する要素を右クリックして、以下のいずれかを実行します。
   - プレフィックスを選択した要素からのみ削除する場合、**[名前空間プレフィックス]** をポイントして、**<なし>** をクリックします。
   - プレフィックスを要素とそのすべてのサブ要素から削除する場合、**[名前空間プレフィックス (すべてのサブ要素を含む)]** をポイントして、**<なし>** をクリックします。

プレフィックスが、[XML ファイル] エクスプローラー内の要素 (および、該当する場合、そのすべてのサブ要素) から削除されます。

XML ファイルから名前空間のマッピングを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*



タスク ***名前空間マッピングを XML ファイルから削除するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、削除する名前空間属性を含む XML ファイルをクリックします。
3. **[名前空間]** タブをクリックします。
4. 削除する名前空間マッピングを含む行をクリックして、**[削除]** ボタンをクリックします。

テーブルから名前空間が削除されます。

XML ファイルに加えられるインストール時の変更をテストする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

InstallShield では、インストール全部をビルドして実行する手間をかけずに、[XML ファイルの変更] ビューでプロジェクトで構成された XML ファイルの変更のみをテストすることができます。インストールの変更をテストするとき、InstallShield はマシン上にある最新バージョンの MSXML を使用して、XML ファイルの解析と構成した XPath 式の実行を行います。MSXML が XPath 式に一致する XML ファイルの領域を検出したとき、[XML ファイルの変更] ビューで構成された変更が加えられます。



メモ: *XML ファイルの変更に、Windows Installer のプロパティまたは InstallScript テキスト置換が含まれている場合、それらはテスト中、適切な値で置換されません。*



タスク ***XML ファイルに加えられるインストール時の変更をテストするには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、テストする XML ファイルを右クリックして、**[XML ファイルのインストール時の変更のテスト]** をクリックします。**[テスト XML ファイルの選択]** ダイアログ ボックスが開きます。
3. **[ファイル名]** ボックスで、インストール時の変更を適用するターゲット ファイルを選択するか、または、新しいターゲット ファイル名と場所を指定します。
   - インストールの一部としてインストールされる XML ファイルを変更する場合、そのファイルのコピーを選択します。(XML のインストール時の変更をテストしたとき、XML ファイルに変更が加えられるため、インストール内のオリジナルのファイルは、選択しないでください。)
   - ターゲット システムに既に存在する XML ファイルを変更する場合、そのファイルのコピーを選択します。
   - XML ファイルが実行時に存在しなく、且つ、それがインストールの一部としてインストールされない場合に何が起こるかをテストする場合、新しいファイル名と場所を指定します。

   デフォルトのファイル名は、ステップ 2 で右クリックしたテスト ファイルの名前です。
4. **[開く]** をクリックします。

ターゲットファイルが既に存在する場合、テスト ファイルからの変更がターゲット ファイルに適用されます。指定したターゲット ファイルが存在しないときに、**[XML ファイルが存在しない場合、常に作成]** チェック ボックスが、その XML ファイルについて [詳細] タブで選択されている場合、ファイルが作成され、テスト ファイルからの変更が適用されます。

出力ウィンドウの [結果] タブで、インストール テストについての詳細が表示されます。詳細には、テスト ファイルへのハイパーリンクが含まれています。



ヒント: *テストの実行中に、ターゲット ファイルがブラウザーで開かれている場合、ブラウザーをリフレッシュして、場合によって、テスト変更を確認する必要があります。*

XML ファイルに加えられるアンインストール時の変更をテストする



プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイルの変更] ビューで構成した XML ファイルのインストール時の変更をテストしたあと、アンインストール時に発生する XML ファイルの変更をテストすることもお勧めします。このテストを行うことにより、構成した変更が、アンインストール時に期待通りに動作するかどうかを確認できます。

アンインストールの変更をテストするとき、InstallShield はマシン上にある最新バージョンの MSXML を使用して、XML ファイルの解析と構成した XPath 式の実行を行います。MSXML が XPath 式に一致する XML ファイルの領域を検出したとき、[XML ファイルの変更] ビューで構成された変更が加えられます。



メモ: *XML ファイルの変更に、Windows Installer のプロパティまたは InstallScript テキスト置換が含まれている場合、それらはテスト中、適切な値で置換されません。*



タスク ***XML ファイルに加えられるアンインストール時の変更をテストするには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、テストする XML ファイルを右クリックして、**[XML ファイルのアンインストール時の変更のテスト]** をクリックします。**[テスト XML ファイルの選択]** ダイアログ ボックスが開きます。
3. **[ファイル名]** ボックスで、アンインストール時の変更を適用するターゲット ファイルを選択します。

   デフォルトの値は、インストール時のテストを最後に行ったファイルの名前です。
4. **[開く]** をクリックします。

[XML ファイルの変更] ビューで構成されたアンインストールの変更がテスト ファイルに適用されます。

出力ウィンドウの [結果] タブで、アンインストール テストについての詳細が表示されます。詳細には、テスト ファイルへのハイパーリンクが含まれています。アンインストール時に削除されるように XML ファイルを構成すると、そのファイルが存在しなくなる可能性があるため、ハイパーリンクが動作しない場合があります。



ヒント: *テストの実行中に、ターゲット ファイルがブラウザーで開かれている場合、ブラウザーをリフレッシュして、場合によって、テスト変更を確認する必要があります。*

[XML ファイルの変更] ビューから要素または XML ファイルを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*



タスク ***[XML ファイルの変更] ビューから要素または XML ファイルを削除するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、削除する XML ファイルまたは XML 要素を右クリックし、**[削除]** をクリックします。

XML ファイルを削除すると、XML ファイルに関連付けられたコンポーネントもファイルと共に削除されますというメッセージが表示されます。

テキスト ファイルの変更



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための 文字列関数 が含まれます。InstallScript プロジェクトで、これらの関数を使用できます。

InstallShield では、ターゲット システム上で実行時に変更を行うテキスト ファイル (たとえば、.txt、.htm、.xml、.config、.ini、および .sql) 内の検索/置換処理を構成できます。テキスト ファイルは、インストールの一部またはシステム上に既存するファイルのどちらでも構いません。

[テキスト ファイルの変更] ビューでは、テキスト ファイルに行う変更を定義できます。このビューでは以下の処理が可能です:

- プロジェクトに 1 つ以上のテキスト変更セットを追加する。テキスト変更セットは、実行時に検索する 1 つ以上のテキスト ファイルへのリファレンスです。
- テキスト変更セットに 1 つ以上のテキスト変更アイテムを追加する。テキスト変更アイテムは、検索/置換の基準を識別します。



タスク ***テキスト ファイルの変更を構成するには、以下の手順に従います。***

1. テキスト 変更セットを作成する
2. テキスト ファイルの変更を指定する。

実行時にテキスト ファイルを変更する方法については、ドキュメントのこのセクションを参照してください。

テキスト 変更セットを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための 文字列関数 が含まれます。InstallScript プロジェクトで、これらの関数を使用できます。

テキスト ファイルの変更を構成するときの最初の手順は、編集するファイル (複数可) への参照を作成することです。この参照は、*[テキスト変更セット]* と呼ばれます。ファイルのフォーマットとしては、非バイナリファイル (たとえば .txt、.htm、.xml、.config、.ini、または .sql) のどれでも可能です。ファイルはインストールの一部 (つまり、[ファイルとフォルダー] ビューでプロジェクトに追加したファイル)、またはターゲット システム上に既存するファイルでも構いません。



メモ: *各テキスト変更セットは、必ずプロジェクト内のコンポーネントに関連付ける必要があります。このため、テキスト変更セットを作成する前に、プロジェクトに 1 つ以上のコンポーネントが必要です。テキスト変更リファレンスを作成するときにコンポーネントがなかった場合は、[新しいコンポーネントの作成] ダイアログ が表示され、ここでコンポーネントを作成することができます。*



タスク

実行時に変更する 1 つ以上のテキスト ファイルを参照するテキスト変更セットを作成するには、以下の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[テキスト ファイルの変更]** をクリックします。
2. **[テキスト ファイルの変更]** エクスプローラーを右クリックして、**[変更セットの追加]** をクリックします。

   InstallShield は、デフォルト名を持つ新しい変更セットを追加します。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。

   名前は実行時には表示されません。これはプロジェクトに含まれる様々な変更セットを区別するのに内部で使用される名前です。
4. 右側のペインで、変更セットの設定を構成します。各設定の詳細については、「変更セットの設定」を参照してください。

テキスト ファイルへのリファレンスを作成してから、その設定を構成した後、次の検索/置換基準を指定するステップに進みます。



ヒント: *Windows Installer パブリック プロパティを使って、検索に含める/除外するテキスト ファイルの名前を指定することができます。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別され*

た他の構成情報を、実行時に製品のテキスト ファイルが変更される時に使用できるようになります。詳細については、「Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する」を参照してください。

テキスト ファイルの変更を指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための 文字列関数 が含まれます。InstallScript プロジェクトで、これらの関数を使用できます。

編集するテキスト ファイルを指定した後、テキスト ファイルの変更を構成するための次の「検索/置換基準を指定する」ステップに進みます。各基準セットは、*変更アイテム* と呼ばれます。



タスク ***テキスト ファイルに行う変更を指定するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[テキスト ファイルの変更]** をクリックします。
2. **[テキスト ファイルの変更]** エクスプローラーで、定義を行うてテキストの変更を含む変更セット アイテムを右クリックしてから、**[変更の追加]** をクリックします。

   InstallShield は、デフォルト名を持つ新しい変更アイテムを追加します。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。

   名前は実行時には表示されません。これはプロジェクトに含まれる様々な変更アイテムを区別するのに内部で使用される名前です。
4. 右側のペインで、変更の設定を構成します。各設定の詳細については、「変更セットの設定」を参照してください。

テキスト ファイルで追加文字列を変更するには、このビューで変更セットに追加の変更アイテムを 1 文字列につき 1 アイテム追加します。



ヒント: *Windows Installer パブリック プロパティを使って、検索文字列と置換文字列を指定できます。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、実行時に製品のテキスト ファイルが変更される時に使用できるようになります。詳細については、「Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する」を参照してください。*

テキスト ファイルの変更が行われる順番を変更する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための 文字列関数 が含まれます。InstallScript プロジェクトで、これらの関数を使用できます。

[テキスト ファイルの変更] ビューで一覧されている順番で、ターゲット システム上のテキスト ファイルに変更が行われます。



タスク

テキスト ファイルが実行時に変更される順番を変更するには、以下の手順に従います。

1. **[システム構成]** の下のビュー リストにある **[テキスト ファイルの変更]** をクリックします。
2. **[テキスト ファイルの変更]** エクスプローラーで、変更セット アイテムまたは変更アイテムの 1 つ右クリックして、**[上に移動]** または **[下に移動]** をクリックします。

すべてのテキスト ファイルの変更が正しく並べ替えられるまで最後のステップを繰り返します。

Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための 文字列関数 が含まれます。InstallScript プロジェクトで、これらの関数を使用できます。

Windows Installer プロパティを使って、検索/変更を行うテキスト 文字列を指定します。プロパティを使って、検索に含める、または検索から除外するテキスト ファイルを指定することもできます。

実行時に、Windows Installer は **MsiFormatRecord** を使ってプロパティの値を解決し、その値を使ってテキスト ファイルを変更します。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、実行時に製品のテキスト ファイルが変更される時に使用できるようになります。

例

以下は、エンド ユーザーがインストール中に、実行時に XML ベースの web.config ファイルに書き込む IP アドレスを指定できるようにするための手順を説明します。web.config ファイルが製品と共に INSTALLDIR にインストールされていて、以下のような XML を含む:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <appSettings>
      <add key="IP アドレス" value="デフォルト" />
   </appSettings>
</configuration>
```

太字表示のデフォルト値は、エンド ユーザーが入力する IP アドレスで置換します。

[テキスト ファイルの変更] ビューにある次の変更セットの設定では、ハードコード化された値をプロパティで置換できます:

- 含めるファイル
- 除外するファイル

[テキスト ファイルの変更] ビューで、次の変更アイテムの設定にはプロパティも使用できます:

- 検索する文字列
- 置換後の文字列
- テキストの挿入

これらの設定のいずれかでプロパティを指定するとき、プロパティは角かっこで囲み、プロパティ名はすべて大文字にする必要があります (例、**[MYPROPERTY]**)。

手順のステップ 4 はプロジェクトの種類によって若干異なります。これは、基本の MSI インストールのユーザー インターフェイスは Windows Installer が制御し、InstallScript MSI インストールのユーザー インターフェイスは InstallScript エンジンが制御するためです。



タスク ***エンドユーザーが IP アドレスを指定できるようにするには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[テキスト ファイルの変更]** をクリックします。
2. インストールが検索を行うファイルを識別する、変更セット アイテムを追加および構成します:
   a. **[テキスト ファイルの変更]** エクスプローラーを右クリックして、**[変更セットの追加]** をクリックします。

      InstallShield が、新しい変更セット アイテムを追加します。ステップ 2b から 2d までは、右側のペインに表示される設定を構成する方法を説明します。

   b. **"ターゲット フォルダー"** 設定で、**[INSTALLDIR]** ディレクトリ プロパティを選択します。
   c. **"含めるファイル"** 設定に、次のように入力します:

      web.config

   d. 残りの設定のデフォルト値はそのままにします。
3. 検索/置換の基準を識別する変更アイテムを追加および構成します:

a. **[テキスト ファイルの変更]** エクスプローラーで、ステップ 2 で作成した変更セット アイテムを右クリックしてから、**[変更の追加]** をクリックします。

   InstallShield が、新しい変更アイテムを追加します。ステップ 3b から 3e までは、右側のペインに表示される設定を構成する方法を説明します。

b. **"アクション"** 設定で **[置換]** を選択します。

c. **"検索する文字列"** 設定に、次のように入力します:

   <add key="IP アドレス" value="デフォルト"

d. **"置換後の文字列"** 設定に、次のように入力します:

   <add key="IP アドレス" value="[MYPROPERTY]"

e. 残りの設定のデフォルト値はそのままにします。

4. ダイアログでプロパティを使用します。この部分の手続きは、使用しているプロジェクトの種類によって異なります。

   - 基本の MSI プロジェクトの場合:

     a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。

     b. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、User Name コントロールを含めるダイアログの下にある言語をクリックします。代わりに、新しいダイアログを追加することもできます。

     c. **Edit Field** コントロールをダイアログに追加し、**Property** プロパティを次にように設定します:

        MYPROPERTY

   - InstallScript MSI プロジェクトの場合:

     a. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。

     b. User Name コントロールを含めるダイアログの OnFirstUIBefore イベントでダイアログ コードを見つけ、Windows Installer API 関数 **MsiSetProperty** の呼び出しを追加します。たとえば、エンド ユーザーがプロジェクトに追加されている SdShowDlgEdit1 ダイアログにある編集ボックスに IP アドレスを入力できるようにするには、次のコード行で示されているように、**MsiSetProperty** の呼び出しを追加します。

        ```
        Dlg_SdShowDlgEdit1:
            nResult = SdShowDlgEdit1 (szTitle, szMsg, szField1, svEdit1);
            MsiSetProperty (ISMSI_HANDLE, "MYPROPERTY", svEdit1);
            if (nResult = BACK) goto Dlg_SdWelcome;
        ```

5. リリースをビルドします。



ヒント: 複数言語プロジェクトを作成する場合で、インストールが使用する言語に基づいて異なる値を持つプロパティを使用するには、ローカライズ可能なプロパティを使います。詳細については、「ローカライズ可能なプロパティを作成する」を参照してください。

MsiFormatRecord (Windows Installer ヘルプ ライブラリ)

ANSI テキスト ファイルを開く時に使用するコード ページを指定する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための 文字列関数 が含まれます。InstallScript プロジェクトで、これらの関数を使用できます。

インストールが ANSI テキスト ファイルを開いて、[テキスト ファイルの変更] ビューで構成された変更を行うとき、インストールはプロジェクトの ISSearchReplaceSet テーブルにある CodePage 列で指定されたコードページを使用します。この列のデフォルト値は数値 0 です。これは、CP_ACP コード ページで、ターゲット システム上で、システム Windows ANSI コード ページとして現在構成されているものです。[ダイレクト エディター] ビューを使って、この値を特定のコード ページでオーバーライドできます。

CodePage 列は 2 バイト (署名付き) です。32,767 よりもい大きい CodePage 値を提供する場合、2 の補数で指定します。

例:

名前	識別子	CodePage
UTF-8	65001	-535
UTF-7	65000	-536
Windows-1252	1252	1252



タスク

ANSI テキスト ファイルを開く時に使用するデフォルトのコード ページをオーバーライドするには、以下の手順に従います:

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]**エクスプローラーで、ISSQLDBMetaData テーブルをクリックします。InstallShield の右側に設定が表示されます。
3. グリッドで、コード ページを構成する変更セット アイテムに対応する行を見つけます。

   このグリッドの ISSearchReplaceSet 列には、**[テキスト ファイルの変更]** ビューの **[テキスト ファイルの変更]** エクスプローラーで利用可能な変更セットの名前がすべて表示されます。
4. 適切な行で **CodePage** フィールドの値を変更します。

[テキスト ファイルの変更] ビューから変更アイテムまたは変更セットを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための 文字列関数 が含まれます。InstallScript プロジェクトで、これらの関数を使用できます。

あるテキスト ファイルに関して、特定の変更を今後行わないようにするには、[テキスト ファイルの変更] ビューからその変更アイテムを削除します。また、ファイルまたはテキスト ファイルのグループを今後検索しないようにするには、[テキスト ファイルの変更] ビューから対応する変更セットを削除します。



タスク ***[テキスト ファイルの変更] ビューから変更アイテムまたは変更セットを削除するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[テキスト ファイルの変更]** をクリックします。
2. **[テキスト ファイルの変更]** エクスプローラーで、削除する変更セット アイテムまたは変更アイテムを右クリックしてから、**[削除]** をクリックします。

[テキスト ファイルの変更] ビューからアイテムが削除されます。

スケジュール タスク



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

ターゲット システムでインストールが実行された時、Windows タスク スケジューラーを通して、自動タスクを作成および構成することができます。スケジュールされたタスクは、インストールの一部であるファイルや、ターゲット システムに既に存在するファイルを起動することができます。

スケジュール タスクの追加



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*



メモ: *それぞれのスケジュール タスクは、必ずプロジェクト内のコンポーネントに関連付ける必要があります。したがって、スケジュール タスクを追加する前に、プロジェクトにコンポーネントが最低 1 つ存在している必要があります。スケジュール タスクを追加するときにコンポーネントが存在しない場合は、[新しいコンポーネントの作成] ダイアログ ボックスが表示され、ここでコンポーネントを作成することができます。*



タスク

プロジェクトにスケジュール タスクを追加するには、以下の手順を実行します。

1. **[システム構成]** の下のビュー リストにある **[スケジュール タスク]** をクリックします。
2. **[スケジュール タスク]** エクスプローラーを右クリックして、**[スケジュール タスクの追加]** をクリックします。InstallShield は、デフォルト名を持つ新しいタスクを追加します。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。

   名前は実行時には表示されません。これはプロジェクトに含まれる様々なスケジュール タスクを区別するのに内部で使用される名前です。
4. 右側のペインで、タスクの設定を構成します。各設定の詳細については、「スケジュール タスクの設定」を参照してください。

Windows Installer プロパティを使って、スケジュール タスクをダイナミックに構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

Windows Installer プロパティを使って、スケジュール タスクによって起動されるファイルを実行するために必要なアカウント情報などの情報を指定することができます。

実行時に、Windows Installer は **MsiFormatRecord** を使って、プロパティの値を解決し、その値を使ってスケジュール タスクを構成します。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、実行時に製品のスケジュール タスクが作成される時に使用できるようになります。

例

以下の手順では、インストール時に、エンドユーザーに、どのようにスケジュール タスクの実行に必要なアカウントとパスワードを指定させるかが示されています。プロパティを指定するとき、プロパティは角かっこで囲み、プロパティ名はすべて大文字にする必要があります (例、**[MYPROPERTY]**)。

手順のステップ 4 はプロジェクトの種類によって若干異なります。これは、基本の MSI インストールのユーザー インターフェイスは Windows Installer が制御し、InstallScript MSI インストールのユーザー インターフェイスは InstallScript エンジンが制御するためです。

タスク ***エンドユーザーに、スケジュール タスクのアカウントとパスワード情報を指定させるには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[スケジュール タスク]** をクリックします。
2. **[スケジュール タスク]** エクスプローラーで、構成するタスクを選択します。
3. **"別のユーザーとして実行"** 設定に、次のように入力します:

   [DOMAINNAME]¥[USERNAME]
4. **"パスワード"** 設定に、次のように入力します:

   [PASSWORD]
5. ダイアログでプロパティを使用します。この部分の手続きは、使用しているプロジェクトの種類によって異なります。

   - 基本の MSI プロジェクトの場合:

     a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。

     b. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、Domain Name、User Name、および Password コントロールを含めるダイアログの下にある言語をクリックします。代わりに、新しいダイアログを追加することもできます。

     c. **Edit Field** コントロールをダイアログに追加し、**Property** プロパティを次にように設定します:

        DOMAINNAME

     d. **USERNAME** と **PASSWORD** プロパティに対し、ステップ 5c を繰り返します。

   - InstallScript MSI プロジェクトの場合:

     a. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。

     b. Domain Name、User Name、および Password コントロールを含めるダイアログの OnFirstUIBefore イベントでダイアログ コードを見つけ、Windows Installer API 関数 **MsiSetProperty** の呼び出しを追加します。たとえば、エンド ユーザーがプロジェクトに追加されている **SdShowDlgEdit3** ダイアログにあ

る編集ボックスに情報を入力する場合、次のコード行で示されているように、**MsiSetProperty** の呼び出しを追加します。

```
Dlg_SdShowDlgEdit3:
    nResult = SdShowDlgEdit3
        (szTitle, szMsg, szField1, szField2, szField3, svEdit1, svEdit2, svEdit3);
    MsiSetProperty (ISMSI_HANDLE, "DOMAINNAME", svEdit1);
    MsiSetProperty (ISMSI_HANDLE, "USERNAME", svEdit2);
    MsiSetProperty (ISMSI_HANDLE, "PASSWORD", svEdit3);
    if (nResult = BACK) goto Dlg_SdWelcome;
```

6. リリースをビルドします。

スケジュール タスクの削除



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*



タスク ***スケジュール タスクをプロジェクトから削除するには、以下の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[スケジュール タスク]** をクリックします。
2. **[スケジュール タスク]** エクスプローラーで、削除するインスタンスを右クリックして **[削除]** をクリックします。

タスクがプロジェクトから削除されます。

Windows サービスのインストール、制御、および構成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

Windows サービスは、ログイン済みのユーザーが居ない場合でも、Windows ベースのシステム上で様々なシステム タスクを管理するためにバックグラウンドで実行される実行可能ファイルです。サービスは実行可能ファイルですが、これはサービスとして設計されたものであり、任意の実行可能ファイルをサービスとして使用することはできません。Windows サービスは、システム起動時に毎回実行することも、必要に応じてオンデマンドで実行することも可能です。InstallShield を使って、新しい Windows サービスをインストール、または既存のサービスを構成することができます。Windows で提供されているサービス管理ツールを使って、システムにインストール済みのサービスを参照および構成できます。

[サービス] ビューを使って、インストールまたはアンインストール中にサービスのインストール、開始、停止、または削除を行うコンポーネントを構成できます。また、このビューを使って、Windows Installer 5 で提供されている拡張サービスのカスタマイズ オプションを構成することもできます。[サービス] ビューの他にも、[コンポーネント] ビューまたは [セットアップのデザイン] ビューの [詳細設定] ノードの下にある [サービス] 領域を使うこともできます。これらの領域のいずれかを使ってサービス情報を構成した場合、対応する領域も自動的に更新されます。

サービスに関する作業について以下の点についてご注意ください:

- 開始、停止、削除、または構成を行うサービスは、インストールまたはアンインストール中にターゲット システムに既存するものか、インストールの一部としてインストールされるものかを問いません。
- サービス実行可能ファイルは、そのコンポーネントのキー ファイルでなくてはなりません。詳細については、「コンポーネントのキーファイル」を参照してください。
- サービスのコンポーネントの "リモート インストール" 設定は [ローカルを優先] でなくてはなりません。詳細については、「コンポーネントの "リモート インストール" 設定を構成する」を参照してください。



ヒント: *[サービス] ビューの上部にある [ビュー フィルター] を使って、ビューで表示するサービス データを含むコンポーネントまたは機能を選択し、ビューで非表示にするサービス データを含むコンポーネントまたは機能を隠すことができます。ビューフィルターは、プロジェクトの機能、サブ機能、およびコンポーネントの階層をリスト表示します。*



タスク ***サービスをインストール、開始、停止、削除、アンインストール、または構成するには、以下の手順に従います:***

1. インストールがサービスをインストールする場合、プロジェクトに含まれるコンポーネントにサービス実行可能ファイルを追加して、それをコンポーネントのキーファイルとします。詳細については、「ファイルをコンポーネントに追加する」を参照してください。

   サービスがターゲット システム上に既存する場合、この手順は省略します。

2. **[システム構成]** の下のビュー リストにある **[サービス]** をクリックします。
3. **[サービス]** ノードを右クリックして、**[サービスの追加]** をクリックします。新しいサービスが追加されます。
4. サービスの新しい名前を入力するか、後で F2 キーを押して名前を変更します。

   ここで入力する名前は、サービスの [プロパティ] ダイアログ ボックスに表示される名前と一致しなくてはなりません。(インストール済みサービスのプロパティにアクセスするには: [サービス] 管理ツールで、サービスを右クリックしてから [プロパティ] を選択します。)

5. 追加したサービスを選択してから、右側のペインに表示される設定を必要に応じて構成します。各設定についての詳細は、「[サービス] ビュー」を参照してください。

このコンポーネントのキーファイルの各サービスについて、この手順を繰り返します。



メモ: *サービスの設定を構成するにあたっては、サービスの技術的な詳細についてよく知っておく必要があります。*



ヒント: *実行時にサービスの構成が失敗すると、インストールが失敗してエラーメッセージを表示することがあります。ICE 102 は、サービス関連の設定の一部を検証して、構成エラーの回避に役立ちます。したがって、リリースに対する検証を行うことが推奨されます。*

実行時に既定のユーザー アカウントを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

InstallShield では、ログオン ダイアログを使わずに、1 つまたは複数の Windows ユーザー アカウントをグループとして作成することができます。このサポートには、実行時に設定されるアカウントごとに、ユーザー アカウント、グループ、およびパスワードの 3 つの Windows Installer プロパティが必要です。

ユーザー アカウントを作成するためのプロパティ

実行にユーザー アカウントを作成するには、次のプロパティを使用します:

- ISNetApiLogonUsername—このプロパティの値を、インストール時に作成するユーザー アカウントに設定します。次のいずれかのフォーマットを使用します:
  - *マシン名¥ユーザー名*
  - *ドメイン名¥ユーザー名*
- ISNetApiLogonGroup—このプロパティの値を、ユーザー アカウントが属するグループに設定します。
- ISNetApiLogonPassword—このプロパティの値を、ユーザー アカウントに使用されるパスワードに設定します。

必須のプロパティとその値を構成するには、[プロパティ] ビューを使用します。その手順は、「Windows Installer ベースのプロジェクトにおけるプロパティの作成」を参照してください。

ユーザー アカウントを作成するためのガイドライン

複数のユーザー アカウントを作成するには、**ISNetApiLogonUsername** プロパティ値のそれぞれのエントリを角かっこで囲んだチルダ ([~]) を使って区切ります。たとえば、**Domain1** という名前のドメインに **User1** という名前のユーザーを作成し、**Machine2** という名前のマシン上で **User2** という名前のユーザーを作成する場合、次のフォーマットを使用します:

Domain1¥User1[~]Machine2¥User2

ISNetApiLogonUsername プロパティで指定したユーザー アカウントにはそれぞれ、**ISNetApiLogonGroup** プロパティで対応するグループと、ISNetApiLogonPassword プロパティで対応するパスワードが指定されている必要があります。これらの 3 つのプロパティで、同数のエントリが指定されていなかった場合、ランタイム エラーが発生します。

以下は、3 つの異なるユーザー アカウントを作成するときのサンプル プロパティとそれらの対応する値です:

テーブル 9 ・ ユーザー アカウントとグループのプロパティ

プロパティ名	プロパティ値
ISNetApiLogonUsername	Domain1¥User1[~]Domain1¥User2[~]Domain2¥User3
ISNetApiLogonGroup	Users[~]Users[~]Administrators
ISNetApiLogonPassword	Password1![~]Password1![~]Password1!

この例では、**User1** は、**Users** グループの一部で、**Domain1** ドメイン用に **Password1!** というパスワードが設定されています。**User2** は、**Users** グループの一部で、**Domain1** ドメイン用に **Password1!** というパスワードが設定されています。**User3** は、**Administrators** グループの一部で、**Domain2** ドメイン用に **Password1!** というパスワードが設定されています。

ISNetApiLogonUsername、**ISNetApiLogonGroup**、および ISNetApiLogonPassword プロパティは、フォーマット済みのプロパティです。このため、これらのプロパティの値には、異なるプロパティ、環境変数、ファイル パス、またはコンポーネント ディレクトリ パスが指定可能です。例:

- **[PROPERTYNAME]**—実行時に、指定されたプロパティの値を解決します。
- **[%EnvironmentVariable]**—実行時、指定された環境変数の値に解決されます。
- **[#FileKey]**—実行時に、File テーブルで指定されたファイル キーがあるファイルのファイルの完全パスに解決されます。
- **[$ComponentName]**—実行時に、指定されたコンポーネントのインストール場所に解決されます。

ユーザーごとのインストールとマシンごとのインストールの違い



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

2 つの Windows Installer プロパティおよび現在のユーザーの権限によって、製品のショートカットやレジストリ エントリなどの構成情報がターゲット マシン上で All Users プロファイルまたは現在のユーザーのプロファイルのどちらに格納されるべきかが決定されます。

- ALLUSERS によって、構成が格納される場所が判別されます。
- MSIINSTALLPERUSER は、Windows Installer によって、パッケージが現在のユーザーに対してのみインストールされることを示します。

  MSIINSTALLPERUSER プロパティは、Windows Installer 5 および Windows 7、または Windows Server 2008 R2 で使用できます。以前のバージョンの Windows Installer と Windows は、このプロパティを無視します。

プロパティ マネージャーを使って、プロジェクトに ALLUSERS と MSIINSTALLPERUSER プロパティを設定できます。また、つぎの方法を使ってこれらのプロパティを設定することもできます。

- コマンドラインを使用する
- カスタム アクションを使用する
- CustomerInformation および ReadyToInstall ダイアログを使用する

ALLUSERS、MSIINSTALLPERUSER、および Windows 7 または Windows Server 2008 R2

ALLUSERS プロパティが 1 に設定されていて、**MSIINSTALLPERUSER** に 1 が設定されている場合、Windows Installer は、ユーザーごとにインストールを実行します。

マシンごとのインストール時、Windows Installer は昇格された権限を必要とし、ファイルとレジストリ エントリがマシンごとの場所に配置します。ユーザー アカウント制御 (UAC) がターゲット システムで提供されている場合、マシンごとのインストールでは、通常、ユーザーの権限レベルに応じて、同意または認証情報を求めるプロンプトが表示されます。ユーザーごとのインストールでは、Windows Installer によって認証情報を求めるプロンプトは表示されず、ファイルとレジストリ エントリはユーザーごとの場所に配置します。

詳しい情報は、MSDN Web サイトの「Single Package Authoring」を参照してください。

Windows Vista 以降における ALLUSERS の効果

スクリプト内実行設定がシステム コンテキストで遅延実行になっているカスタム アクションは、Windows で LocalSystem アカウントに与えられた権限を使ってアクションを実行するときに使用されます。これは、Windows Installer がシステム コンテキストで実行されるためです。システム コンテキストで遅延としてマークされていないアクションは、ユーザーを偽装して実行され、インストールを起動するユーザーが所有する権限を持ちます。

ユーザーごとのインストール (ALLUSERS が設定されていないインストール) が実行されると、"システム コンテキストで遅延" のアクションは、通常の遅延または即時カスタム アクションと同じコンテキストで (ユーザーを偽装して) 実行されます。これにより、次のような状況で、カスタム アクションの実行時に問題が起きる可能性があります。

- Windows Installer インストールを起動するユーザーが管理者ではないとき、またはユーザーがインストールを Windows Vista 以降で実行していて、ユーザーが管理者グループに属し、ユーザーがデフォルトで管理者権限を持たないとき。
- カスタム アクションは、マシンのマシンごとの場所でリソース (Program Files フォルダーにあるファイル、HKEY_LOCAL_MACHINE のレジストリ キーまたは値) を変更しようと試みるとき。

これは、Windows XP または Windows の初期のバージョンでは問題にはならない可能性がありますが、デフォルトで Windows Vista 以降では、ユーザーは完全管理者権限は与えられません。したがって、ALLUSERS が設定されていない場合、"システム コンテキストで遅延" のアクションがユーザー偽装で実行されると、カスタム アクションは失敗する可能性があります。

この動作を防ぐには、[プロパティ マネージャー]で ALLUSERS を 1 に指定して、常にマシンごとのインストールを実行することをお勧めします。マシンごとのインストールは通常、ユーザーごとのインストールよりも管理が簡単です。

ALLUSERS のデフォルト値

すべての基本の MSI プロジェクトでは、デフォルトで、ALLUSERS プロパティは 1 に設定されています。管理者権限を使わないでユーザーごとにインストールできるようにインストールを構成する場合、ALLUSERS プロパティの値を変更するか、このプロパティをプロジェクトから削除したほうが良い場合があります。

ReadyToInstall ダイアログのデフォルトのコントロール

[一般情報] ビューの "[ユーザーごと] オプションの表示" 設定を使って、エンド ユーザーが製品をすべてのユーザー、または現在のユーザーのみにインストールするかを選択できるオプションを提供するかどうかを指定します。以下は、"All Users オプションの表示" 設定に選択できるオプションです:

- **いいえ**—InstallShield は関連プロパティを設定しません。実行時に、ReadyToInstall はエンド ユーザーが製品をインストールする方法を指定できるボタンを表示しません。これがデフォルトの値です。
- **はい** InstallShield は **ISSupportPerUser** プロパティを 1 に設定します。

実行時に以下の条件が True 評価されると、ReadyToInstall にはユーザーごとまたはマシンごとのボタンが表示されます:

- **ISSupportPerUser** プロパティが 1 である。
- ターゲット システムに Windows 7 以降、または Windows Server 2008 R2 以降が搭載されている。
- 製品がターゲット システムに既にインストールされていない。

ReadyToInstall ダイアログのボタンを使って、エンド ユーザーはプロジェクトをインストールする方法を指定できます。昇格された権限が必要な場合、[すべてのユーザー] ボタンにシールド アイコンが含まれます。エンド ユーザーが [ユーザーごと] ボタンを選択した場合、ALLUSERS プロパティが 2 に、MSIINSTALLPERUSER プロパティが 1 に設定されます。エンド ユーザーが [すべてのユーザー] ボタンを選択した場合、ALLUSERS プロパティが 1 に設定され、MSIINSTALLPERUSER プロパティは設定されません。

メモ: *[一般情報] ビューの "[ユーザーごと] オプションの表示" 設定に [いいえ] を選択しても、エンド ユーザーがインストールを実行するときに、コマンドラインから* MSIINSTALLPERUSER *を設定することを阻止することはできません。インストールがこれをサポートしない場合、起動条件を追加するか、その他の実行時のチェックを行って、これを回避することができます。*

CustomerInformationダイアログのデフォルトのコントロール

デフォルトで、すべての基本の MSI プロジェクトの CustomerInformation ダイアログは、エンドユーザーが製品をすべてのユーザーにインストールするか、または現在のユーザーのみにインストールするかを指定できるラジオ ボタン グループを表示しないようになっています。このダイアログについては、これが推奨される実装です。

タスク

エンド ユーザーが CustomerInformation ダイアログから ALLUSERS を設定できるようにするラジオ ボタン グループを表示するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** の下にある **CustomerInformation** ダイアログを展開して、**[動作]** をクリックします。CustomerInformation ダイアログのコントロール リストが表示されます。
3. 右下のペインの下部にある、**[条件]** タブをクリックします。右上のペインに、選択されたコントロールの条件が表示されます。
4. コントロール リストから、**DlgRadioGroupText** を選択します。右側のペインにコントロールの条件が表示されます。
5. 右上のペインで、値が 1 で、かつアクションが Hide になっている条件を含む行を右クリックし、**[削除]** をクリックします。

6. **RadioGroup** コントロールについて、ステップ 4 と 5 を繰り返します。

ALLUSERS Property (Windows Installer ヘルプ ライブラリ)

管理者権限なしに InstallScript インストールを実行する



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallShield では、管理者権限を持たないエンドユーザーでも実行可能な InstallScript インストールを作成することができます。レジストリ エントリ、フォルダー、[スタート メニュー] のアイテムの適切な場所は、実行に動的に判別されます。InstallScript システム変数 ALLUSERS は、このようなインストールのキーとなります。つまり、インストールが管理者権限を持たないエンドユーザーによって実行された場合、ALLUSERS は FALSE の値で初期化され、スクリプトで新しい値を割り当てても効果はありません。

次の事項に注意してください。

- ファイルの自己登録には管理者権限が必要です (ファイルの自己登録は、"自己登録" 設定で [はい] が選択されているコンポーネントに含まれています)。
- 通常 InstallScript オブジェクトでは、ファイルを管理者権限が必要な場所にファイルをインストールまたは登録するため、権利者権限が必要です。
- DIFx ドライバーのインストールには、管理者権限が必要です。
- One-Click Install インストールのセットアップ プレーヤーのインストールには、管理者権限が必要です。
- Windows サービスのインストールには、管理者権限が必要です。
- リモート レジストリに接続するとき、管理者権限が必要です。
- 管理者またはパワーユーザー以外のエンドユーザーが、ALLUSERS が TRUE に設定された状態でインストールされた InstallScript インストールを保持しようと試みた場合、初期化中にエラー メッセージが表示されます。エラー メッセージでは、製品は管理者権限を持つユーザーによってインストールされており、製品の変更およびアンインストールには同様の権限が必要であることが示されます。

インストール動作のカスタマイズ

インストール作成の重要な要素は、それをエンドユーザーのニーズに合わせてカスタマイズすることです。ドキュメントの「インストール動作のカスタマイズ」セクションは、インストールの機能を拡張するのに役立つ InstallShield の様々な機能について説明します。たとえば、Windows Installer が直接サポートしていない機能を追加するカスタム アクションを作成するのに役立ちます。さらに、InstallScript ビューのスクリプト エディターで書いたスクリプトを呼び出すカスタム アクションを設定することができます。このセクションには、次のトピックが含まれています:

- InstallScript を使用する
- カスタム アクションを使用
- アクション テキストを使う
- シーケンスを定義する
- サポート ファイルを使用する

InstallScript を使用する

InstallScript の強力で使いやすい機能を活用することにより、インストールパッケージの機能を拡張できます。InstallScript ビューでは、スクリプト エディター ペインを使用して InstallScript コードを作成できます。



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*



エディション: *InstallScript を使用する場合は* Setup.rul *というファイルがプロジェクトに含まれていなければなりません。InstallScript と InstallScript MSI プロジェクトにはデフォルトで* Setup.rul *ファイルが含まれています。基本の MSI プロジェクト、DIM、およびマージ モジュール プロジェクトで InstallScript カスタム アクションを使用する場合は、プロジェクトに* Setup.rul *ファイルを追加する必要があります。*

InstallScript とInstallScript MSI プロジェクトで、イベント ドリブン型 InstallScript を使用する

InstallScript と InstallScript MSI プロジェクトには、イベント ドリブン型スクリプトが含まれています。これらのプロジェクト タイプで InstallScript を使用すると、多くの関数がアンインストール用にログに記録されます。

カスタム アクションでの InstallScript の使用

基本の MSI、DIM、およびマージ モジュール プロジェクト

基本の MSI、DIM、またはマージ モジュール プロジェクトでは、カスタム アクションを使って、実行時に InstallScript を実行できます。これらのプロジェクト タイプでは、イベントドリブン型スクリプトはサポートされていません。

InstallScript MSI プロジェクト

InstallScript MSI プロジェクトでは、[実行] シーケンスで、デフォルトのイベント ハンドラーがインストールのニーズに応じて適切にスケジュールされていない場合に、InstallScript カスタム アクションを使って機能を拡張することができます。

カスタム アクションの作成と使用



タスク ***InstallScript カスタム アクションを作成して、インストールで実行するには、以下の手順を実行します。***

1. 既存の Setup.rul がない場合、InstallScript ビューでプロジェクトに Setup.rul を追加します。
2. エントリ ポイント関数を記述します。エントリ ポイント関数は、InstallScript MSI インストール プロジェクトの [ユーザー インターフェイス] シーケンスでは呼び出すことができないことに注意してください。
3. スクリプトをコンパイルします。
4. InstallScript 関数を呼び出すカスタム アクション を作成します。
5. InstallScript カスタム アクションを、シーケンス内に含める (InstallScript MSI、基本の MSI、およびマージ モジュール プロジェクトの場合) か、コントロール イベントとして実行 (基本の MSI、およびマージ モジュール プロジェクトの場合のみ) することにより呼び出します。
6. 必要に応じてデバッグします。



メモ: *他のカスタム アクションと同様、InstallScript カスタム アクションによってシステムに加えた変更は、パッケージをアンインストールするときに自動的に復元されません。アンインストーラーは、InstallScript カスタム アクションをログに記録したり、削除することがないため、カスタム アクションで加えたすべての変更をアンインストールするには、アンインストール用のカスタム アクションを記述する必要があります。*

ISSetup.dll の概要



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

ISSetup.dll は、すべての InstallScript スクリプト ランタイム エンジンを含む C++ MSI DLL です。InstallScript MSI、基本の MSI、DIM、およびマージ モジュール プロジェクトでは、ISSetup.dll が InstallScript カスタム アクションを実行します。InstallScript プロジェクトでは、ISSetup.dll を必ず Disk1 フォルダーに配置します。

別のインストールによってターゲット マシン上に最新版の InstallScript スクリプト ランタイム エンジン がある場合でも、インストールは常にビルドに使われたエンジンと同じバージョンの InstallScript スクリプト ランタイム エンジンを利用します。



重要 *・次の事項に注意してください。*

- *ISSetup.dll を使用する場合、InstallScript MSI、基本の MSI、DIM、またはマージ モジュール プロジェクトに追加することができる InstallScript カスタム アクションの数は 1,000 までです。プロジェクトに InstallScript カスタム アクションを 1,000 以上含めると、ビルドエラーが発生します。*
- *InstallScript プロジェクトに使われる* ISSetup.dll *のバージョンは、InstallScript MSI、基本の MSI、DIM、およびマージ モジュール プロジェクトで使われるバージョンとは異なります。これらの 2 つのバージョンを置き換えることはできません。*

スクリプト ファイル



プロジェクト *- この情報は、InstallScript プロジェクトに適用します。*

インストール プロジェクトを作成した場合、InstallShield では 2 つのスクリプト ファイルが作成され、[プロジェクト] フォルダーの [スクリプト ファイル] フォルダーに格納されます。

- Setup.rul はグローバル イベント ハンドラーと例外ハンドラーのために作成されます。
- FeatureEvents.rul は機能イベント ハンドラーのために作成されます。

元々これら 2 つのファイルは空白で、InstallShield の InstallScript ビューから選択しない限り、InstallShield が定義したデフォルトのイベント ハンドラーはこれらのファイルには含まれません。選択すると、適切なスクリプト ファイルに挿入されて InstallScript ビューのスクリプト ペインに表示されるので、そこで編集が可能です。

FeatureEvents.rul でデフォルトの機能のイベント ハンドラーコードを変更した場合、Setup.rulに次のステートメントを入れてインストールに変更を含める必要があります。

```
#include "FeatureEvents.rul"
```

InstallScript ファイルの作成



プロジェクト *- この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

InstallScript を使用する場合は Setup.rul *というファイルがプロジェクトに含まれていなければなりません。InstallScript と InstallScript MSI プロジェクトにはデフォルトで* Setup.rul *ファイルが含まれています。基本の MSI プロジェクト、DIM、およびマージ モジュール プロジェクトで InstallScript カスタム アクションを使用する場合は、プロジェクトに* Setup.rul *ファイルを追加する必要があります。*



タスク ***プロジェクトに新規 InstallScript ファイルを追加するには、以下の手順を実行します。***

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **InstallScript** エクスプローラーで、**[ファイル]** を右クリックして、**[新しいスクリプト ファイル]** を選択します。
3. ファイルに名前をつけます。

デフォルトでは、新規スクリプト ファイルに Setup.rull という名前が付けられます。Setup.rul が既にある場合は、新規ファイルが Setup*n* (*n* は連続番号) という名前で追加されます。ファイル名を右クリックして [名前の変更] をクリックすると、ファイル名を変更できます。

新しい スクリプト ファイルは Link To (リンク先) フォルダーに配置されます。プロジェクトを移動した場合、InstallShield はパス変数の使用を試みます。Link To 値を編集することはできません。

追加のヘッダー ファイル (.h files) やスクリプト ファイルを含めることもできます。上の手順を繰り返して新規スクリプト ファイルをプロジェクトに追加します。

InstallScript ファイルを開く



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*



タスク ***編集するプロジェクト内のスクリプト ファイルを開くには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **InstallScript** エクスプローラーで、開くファイルをクリックします。

スクリプト ファイルの挿入とインポート



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

InstallShieldでは、複数のプロジェクトで InstallScript ファイル (.rul) および InstallScript ヘッダーファイル (.h) の再利用が可能です。InstallScript ビューで、プロジェクトにスクリプト ファイルを挿入したりインポートしたりすることができます。

- スクリプト ファイルを挿入すると、ファイルの現在の保存場所へのリンクが作成されます。
- スクリプト ファイルをインポートすると、プロジェクトのスクリプト ファイルを格納しているフォルダーにそのファイルがコピーされます。インポートしたスクリプト ファイルは、システム上またはリポジトリに保存することができます。

InstallShield は、これらのスクリプト ファイルの場所用に [パス変数] ビューで定義されたすべてのパス変数タイプをサポートします。ただし、ソース コード管理 ソフトウェアがパスを解決できるよう、対応するフォルダーの構成がソース コード データベースに確実に存在していることが重要です。

スクリプト ファイルの挿入

タスク ***スクリプト ファイルを挿入するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **InstallScript** エクスプローラーで、**[ファイル]** を右クリックして、**[スクリプト ファイルの挿入]** を選択します。**[開く]** ダイアログ ボックスが開きます。
3. 挿入する **InstallScript** ファイル (.rul) または InstallScript ヘッダーファイル (.h) を選択します。
4. **[開く]** をクリックします。

スクリプト ファイルのインポート

タスク ***スクリプト ファイルをインポートするには、以下の手順を実行します。***

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **InstallScript** エクスプローラーで、**[ファイル]** を右クリックして、**[スクリプト ファイルのインポート]** を選択します。[InstallScript ファイルのインポート] ダイアログ ボックスが開きます。
3. 以下のいずれかを実行します。
   - **[リポジトリ アイテム]** ボックスで、プロジェクトに追加する InstallScript ファイル (.rul) または InstallScript ヘッダーファイル (.h) をクリックします。
   - インポートしたいスクリプト ファイルがリポジトリに格納されていない場合、**参照**ボタンをクリックしてファイルを選択します。
4. **[OK]** をクリックします。

スクリプトのコンパイル



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

インストール中に InstallScript コードを呼び出すには、事前にスクリプトをコンパイルする必要があります。

スクリプトのコンパイル時に InstallShield が検索するのは、Setup.rul という名前のファイルだけです。異なる名前のファイルを含めることは可能ですが、そのファイルを Setup.rul に含めるか、または #include プリプロセッサ ステートメントを使用したインクルード ファイルに含める必要があります。



タスク ***スクリプトをコンパイルするには、以下のいずれかを行います。***

- **[ビルド]** メニューで **[コンパイル]** をクリックします。
- ツールバー上の [**コンパイル**] ボタンをクリックします。
- CTRL +F7 を押す。

コンパイルの前に、InstallShield は、スクリプト ファイルに加えた変更をすべて保存します。エラーメッセージまたは警告メッセージなどのコンパイラの状態が、出力ウィンドウに表示されます。スクリプト中のエラーが見つかった行に移動するには、コンパイラ メッセージをダブルクリックします。

変更を加えてからスクリプト ファイルをコンパイルする場合、リリースをビルドし直す必要はありません。InstallShield では、リリースのビルド時には常にスクリプトが自動的にコンパイルされます。

スクリプトが正常にコンパイルされた場合は、Setup.inx (セットアップ エンジンが実行するオブジェクト コード) が作成され、リリースのビルド時に Windows Installer パッケージにストリーム入力されます。

スクリプトの変更を確かめる前に、以前実行し InstallScript カスタム アクションをテストしたリリースをアンインストールしなければならない場合もあります。

[設定] ダイアログ ボックス の [コンパイル/リンク] タブでコンパイラ オプションを設定することができます。

スクリプトのデバッグ



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript コードを順番に確認にしながら、コードの進行をチェックする場合は、InstallShield デバッガーを使用すると便利です。

スクリプトのデバッグを行う前に、まずスクリプトをコンパイルして、カスタム アクションとして実行し（該当する場合）、リリースをビルドしておく必要があります。

タスク ***InstallShield から直接スクリプトをデバッグするには、以下のいずれかを実行します。***

- **[ビルド]** メニューで **[デバッグ]** をクリックします。
- F5 キーを押します。
- ツールバー上の [**デバッグ**] ボタンをクリックします。

InstallShield はインストールを実行し、カスタム アクションの実行時に InstallScript デバッガーを開きます。

InstallScript デバッガの使用中に、任意の場所で F1 を押すと InstallScript デバッガー ヘルプが表示されます。

テスト システムのデバッグについては、「任意のコンピューターでのインストールのデバッグ」を参照してください。

プリプロセッサ ステートメントを使用してスクリプトをデバッグする

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

スクリプトの内部デバッガー作成には、#define ステートメントと #ifdef ステートメントを使用します。

タスク ***プリプロセッサ ステートメントを使用してスクリプトをデバッグすには、以下の手順を実行します。***

1. スクリプトにデバッグステートメントを挿入するときは常に、次の #ifdef 命令で始めます:

   ```
   #ifdef DEBUG
   ```

2. その後に続く行で、デバッグ ステートメントを入力します。
3. デバッグステートメントの後の別の行に、次を入力します:

   ```
   #endif
   ```

4. デバッグには、次のコンパイラ設定をコンパイルします。

   ```
   DDEBUG=1
   ```

これは、#ifdef ステートメントを利用したデバッグセクションの例です:

```
#ifdef DEBUG
   if nResult < 0 then
      WriteLine (LogFileHandle, "PlaceBitmap が失敗しました");
   endif;
#endif
```

InstallScript デバッガーを使用して、インストールの実行と同じ状態で、プロジェクトの実行を追跡して、変数を検証することができます。

InstallScript ファイルの名前を変更する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*



タスク ***InstallScript ファイルの名前を変更するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **InstallScript** エクスプローラーで、ファイルを右クリックし **[名前の変更]** をクリックします。
3. ファイルの新しい名前を入力します。



プロジェクト: *InstallScript プロジェクトには、*Setup.rul *という名前のソースファイルが必要です。このファイルはインストール スクリプトのメイン コンパイル ユニットです。InstallScript プロジェクトに、この名前のスクリプト ファイルが含まれていない場合、スクリプトのコンパイル時、またはリリースをビルドした時に、エラー C8503 が発生します。*

スクリプト中で文字列エントリを使用する

文字列識別子を、スクリプトで文字列リテラルを受け入れる任意の値の代わり使用できます。カスタム アクションが実行される時、文字列識別子はインストールを実行中の言語に対応する文字列値で置換されます。

スクリプト中の文字列 ID の先頭には @ マークを置く必要があります。[文字列の選択] ダイアログ ボックスを使って、プロジェクト内の文字列 エントリの一覧を参照できます。ここではまた、選択した文字列識別子をスクリプトに挿入する前に、デフォルト言語の文字列エントリを変更することもできます。

たとえば、プロジェクトに MSG_ACTION_SUCCEEDED という文字列識別子が含まれているとき、その値を次のようにメッセージ ボックスに表示することができます:

```
szMsg = @MSG_ACTION_SUCCEEDED;
nType = INFORMATION;
MessageBox (szMsg, nType);
```

プロジェクトから InstallScript ファイルを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

インストールからスクリプト ファイルを削除する際には、ファイルそのものではなくそのファイルへの参照をプロジェクトから削除します。これにより、後でインストールにそのスクリプト ファイルを挿入することになった場合、スクリプトを書き直す必要がありません。

ただし、スクリプト ファイルを別のファイル中に含めると、コンパイルが可能な場合もあります。



タスク ***プロジェクトから InstallScript ファイルを削除するには、以下の手順を実行します。***

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **InstallScript** エクスプローラーで、ファイルを右クリックし **[削除]** をクリックします。



注意・スクリプト ファイルが上記手続きで削除された場合でも、ファイルの名前を、プロジェクトの場所に残っているそのファイルと同じ名前に変更することはできません。たとえば、プロジェクトに Setup.rul という名前のスクリプト ファイルがあるとします。次に、そのファイルを削除して、デフォルトで Script1.rul という名前の新しいスクリプト ファイルを追加します。このとき、Script1.rul を Setup.rul という名前に変更することはできません。それによって最初の Setup.rul が上書きされてしまうからです。これを回避するには、元の Setup.rul をプロジェクト フォルダーの外へ移動してから名前を変更するか、Setup.rul を Windows のエクスプローラーで削除してください。

スクリプト ライブラリ (.obl ファイル) を作成する

InstallScript のビルトイン関数はライブラリ ファイル (.obl ファイル) で、スクリプトがコンパイル時にリンクされるように定義されています。



タスク ***定義した関数のライブラリ ファイルを作成するには、以下の手順を実行します。***

1. 関数の定義を含む 1 つまたは複数の .rul ファイルを作成します。
2. コマンドラインで、各 .rul ファイルに対して、-c スイッチで Compile.exe を実行し、既存のライブラリ ファイルとリンクさせずに .rul ファイルをコンパイルします。これにより .obs ファイルを作成します (-c スイッチなしのコンパイルで .inx ファイルを作成する方法と異なります)。たとえば、次のコマンドラインは現在のフォルダーに MyFunc.obs ファイルを作成します。

   Compile MyFunc.rul -c

   .obs ファイルを別の名前や場所で作成するには、-o スイッチを使用します。
3. -l スイッチを使用して Compile.exe と 1 つまたは複数の .obs ファイルをパラメーターとして実行して、ライブラリ ファイルを作成します。たとえば、次のコマンドラインは現在のフォルダーに MyFunc.obl ファイルを作成します。

   Compile MyFunc.obs -l

   次のコマンドラインは現在のフォルダーに MyFunc1.obl ファイルを作成します。

Compile MyFunc1.obs MyFunc2.obs –I

.obs ファイルを別の名前や場所で作成するには、–o スイッチを使用します。



ヒント: *.obs ファイルを多数保持している場合、次の例のようにコマンドファイルを使用してコマンドラインを短縮できます :*

Compile @MyObsFiles.txt –I

コマンドファイルをすばやく作成するために、MS-DOS コマンド DIR をその /b *(ベアフォーマット) スイッチと共に使って、出力をファイルにリダイレクトすることができます。例:*

DIR *.obs /b > MyObsFiles.txt

ライブラリ ファイルを使用してインストール スクリプトをコンパイルするには、コマンドライン (または InstallShield 内でコンパイルしている場合は [設定] ダイアログ ボックスの [コンパイル/リンク] タブ) でライブラリ ファイルを指定します。

InstallScript ファイル (.rul および .h) をリポジトリにパブリッシュする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

別のプロジェクトで再利用、または他のユーザーと共有したい既存 InstallScript ファイル (.rul) または InstallScript ヘッダー ファイル (.h) がある場合、リポジトリにそれをパブリッシュすることができます。



タスク ***スクリプト ファイルをリポジトリにパブリッシュするには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **[InstallScript]** エクスプローラーでパブリッシュするスクリプト ファイルを右クリックしてから、**[パブリッシュ ウィザード]** をクリックします。**パブリッシュ ウィザード**が開きます。
3. パブリッシュ ウィザードのパネルを完成します。

リポジトリからプロジェクトへスクリプト ファイルをインポートした後、現在のスクリプト ファイルと既存のリポジトリ スクリプト ファイルとは関連性を持ちません。スクリプト ファイルを変更してリポジトリへ再パブリッシュしても、プロジェクト内のインポート済みスクリプト ファイルには影響しません。ただし、リポジトリからプロジェクトへスクリプト ファイルを再インポートすることができます。

InstallScript リスト

リストは、文字列や番号といった関連情報を格納するのに利用されます。InstallScript リストは C 言語の単方向リストと非常によく似ています。InstallScript リスト関数は柔軟性に富み、情報をそれらが格納されている順番とは異なる順序で戻すことが可能であり、さらにそれらにアクセスして様々な方法で利用することができます。

リスト関数

InstallShield ではリストを作成または操作するための関数がいくつか提供されています。InstallScript リスト関数には 3 つのタイプがあります:

- 文字列リストとのみ動作する **–String** で終わる関数
- 番号リストとのみ動作する **–Item** で終わる関数
- 文字列リストと番号リストのどちらとも共に動作する関数

InstallShield はまた、リストを作成または利用する従属的なリスト関連関数を提供します。

リスト構造

リストは、文字列や番号のいずれかに関連する情報を格納するのに利用されます。リスト内のすべての情報は同じデータ型であり、要素の数は利用可能なメモリ容量によってのみ制限されます。

InstallScript リストは 2 つの部分で構成されます。最初の部分はヘッドで、InstallShield が内部で利用します。リストのヘッドにはリストについて、文字列または番号を含んでいるか否かといった一般情報が含まれています。ヘッドにはまたリストの開始と終了へのポインターが含まれます。

リストの 2 番目の部分はリスト本体です。リスト本体には実際の文字列や番号が含まれます。システムのメモリが有効な限り、リストには文字列や番号を含むことができます。

リストは文字列と番号を同時に含むことができない点に注意してください。リストは文字列のみ、または番号のみを含まなくてはなりません。

リストを代表する変数は、type LIST または type LONG として宣言できます。リストはメモリにのみ格納されます。つまり、インストールが完了したときに、リストは破棄されます。リストが関数のローカルであった場合、関数が呼び出しコードへコントロールを戻したときにリストは破棄されます。

リストの作成と破棄

リスト作成前に、ビルドするリストが文字列リストと数値（項目）リストのどちらかを決めます。リストを作成するには、**ListCreate** 関数を呼び出します。

```
// これは文字列リスト用のリスト ヘッドをビルドします。
listID1 = ListCreate (STRINGLIST);
```

または

```
// これは数値（項目）リスト用のリスト ヘッドをビルドします。
listID2 = ListCreate (NUMBERLIST);
```

ListCreate は自動的にリストのヘッドをビルドし、その ID 番号を戻します。ID は、リストで動作する後に続く関数すべてに利用されます。したがって、別のリスト関数を利用する前に常に **ListCreate** を使ってリストを作成しなくてはなりません。**ListCreate** からの戻り値を LIST タイプまたは LONG タイプの値で格納しなくてはなりません。

この部分が数値リストと、そのあと文字列リストを作成します。また、リストが正しく作成されたことを確認するためにそれぞれをテストします。

```
// 文字列の空リストを作成します。
listID1 = ListCreate (STRINGLIST);

if (listID1 = LIST_NULL) then
    MessageBox ("文字列リストを作成することができませんでした", SEVERE);
endif;

// 数値の空リストを作成します。
listID2 = ListCreate (NUMBERLIST);

if (listID2 = LIST_NULL) then
   MessageBox ("数値リストを作成することができませんでした", SEVERE);
endif;
```

リストを利用し終わったとき、その他の用途のためにメモリーを解除するため、通常はリストを破棄します。**ListDestroy** がリストとその内容を破棄します。この例は listID が参照するリストを作成し、リストへ文字列を追加してリスト全体を破棄します。

```
listID = ListCreate (STRINGLIST);

if (listID = LIST_NULL) then
   MessageBox ("リストを作成できませんでした。", SEVERE);
   abort;
endif;

ListAddString (listID, "これはリストの文字列です", AFTER);
ListDestroy(listID);
```

ListDestroy を使ってリストを破棄しなかったなかった場合、インストールが完了したときにリストは破棄されます。リストが関数のローカルであった場合、関数が呼び出しコードへコントロールを戻したときにリストは破棄されます。

リストへ要素を追加する

InstallShieldではリストへ要素を追加するための関数がいくつか提供されています：

テーブル 1・リストへ要素を追加するときに使用する関数

関数	説明
ListAddItem	項目をリストに追加します。
ListAddString	リストへ文字列を追加します。
ListReadFromFile	テキストファイルをリストへ読み込みます。

ListAddString および **ListAddItem** は、単一の要素を指定されたリストへ追加します。リスト内での新規文字列の配置場所にかかわらず、それが現在の文字列となることに注意してください。BEFORE と AFTER オプションを使って、リスト内で現在の要素に関連して新規要素を配置する場所を指定します。新しく作成されたリストを使って作業中の場合、BEFORE または AFTER のいづれかを使用するとリストの最初の要素位置に文字列が配置されます。

リストへの要素の追加と、それがリスト順や現在の位置にある要素へ与える影響は、例をみると簡単に理解することができます。以下の例では文字列リストと **ListAddString** を使用しますが、**ListAddItem** や数値リストを使用する場合にも同じ原理と手順が当てはまります。次のシナリオを参考にしてください:

- 空白リストへ要素を追加する
- 現在の要素の前に要素を追加する
- 現在の要素の後に要素を追加する
- 現在の要素の前と後に要素を追加する

空白リストへ要素を追加する

リストへ追加した最初の文字列はリストのヘッド部分の直後に配置されます。この文字列 (サンプル コードの *String 1*) がリストで現在の文字列になります。下に表示された文字列部分は、その後に表示されるリストの様な結果となります。

```
// 空白文字列リストを作成する。
listID = ListCreate (STRINGLIST);

// 有効リストのテスト
if (listID = LIST_NULL) then
    MessageBox ("リストは作成されません", SEVERE);
else
    // リストへ文字列を追加する。
    szString = "String 1";
    ListAddString (listID, szString, AFTER);
endif;
```

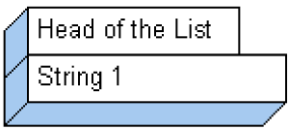


図 1: String 1 がリストへ追加されます。

現在の要素の前に要素を追加する

現在の文字列がリストのトップにある場合で、新規文字列をその前に追加するとき、新しい文字列がリストのトップになります。その結果、元々最初の要素位置にあった文字列は二番目の要素位置へと移ります。最初の要素位置にある新規文字列が現在の文字列となります:

```
// 空白文字列リストを作成する。
listID = ListCreate (STRINGLIST);

// 有効リストのテスト
if (listID = LIST_NULL) then
    MessageBox ("リストは作成されません", SEVERE);
else
    // リストへ文字列を追加する。
    szString = "String 1";
    ListAddString (listID, szString, AFTER);

    szString = "String 2";
    ListAddString (listID, szString, BEFORE);
```

```
endif;
```

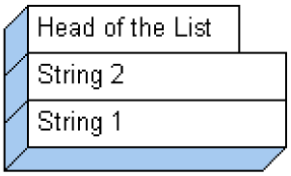


図 2: String 2 が String 1 の前へ追加されます。

現在の要素の後に要素を追加する

現在の文字列がリストのトップにある場合で、現在の文字列の後に新規文字列を追加した場合、リストで 2 番目の文字列となり新しい現在の文字列となります。次の文字列と、後に続く解説を参考にして下さい:

```
// 空白文字列リストを作成する。
listID = ListCreate (STRINGLIST);

// 有効リストのテスト
if (listID = LIST_NULL) then
    MessageBox ("リストは作成されません", SEVERE);
else
    // リストへ文字列を追加する。
    szString = "String 1";
    ListAddString (listID, szString, AFTER);

    szString = "String 2";
    ListAddString (listID, szString, AFTER);
endif;
```

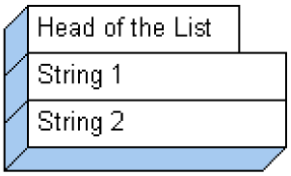


図 3: String 2 が String 1 の後へ追加されます。

現在の要素の前後に要素を追加する

別の例として、以下に示されたコード部分は新規リストを作成し String 1 を先頭位置に配置します。そして String 2 が String 1 の前に追加され、String 2 は先頭位置に現在の文字列として残ります。次に、String 3 が現在の文字列の後に追加され、以下に表示された結果となります。

```
// 空白文字列リストを作成する。
listID = ListCreate (STRINGLIST);

// 有効リストのテスト
if (listID = LIST_NULL) then
    MessageBox ("リストは作成されません", SEVERE);
else
    // リストへ文字列を追加する。
    szString = "String 1";
    ListAddString (listID, szString, AFTER);
```

```
    szString = "String 2";
    ListAddString (listID, szString, BEFORE);

    szString = "String 3";
    ListAddString (listID, szString, AFTER);
endif;
```

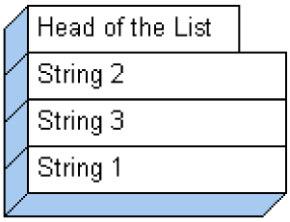


図 4: String 1 が追加されます。その後、String 2 が String 1 の前へ追加されます。最後に、String 3 が String 2 の後へ追加されます。

上の例では、String 3 が現在の文字列の前に追加された場合、下のような結果となり String 3 が現在の文字列となります。コード部分は次の通りです。

```
// 空白文字列リストを作成する。
listID = ListCreate (STRINGLIST);

// 有効リストのテスト
if (listID = LIST_NULL) then
    MessageBox ("リストは作成されません", SEVERE);
else
    // リストへ文字列を追加する。
    szString = "String 1";
    ListAddString (listID, szString, AFTER);

    szString = "String 2";
    ListAddString (listID, szString, BEFORE);

    szString = "String 3";
    ListAddString (listID, szString, BEFORE);
endif;
```

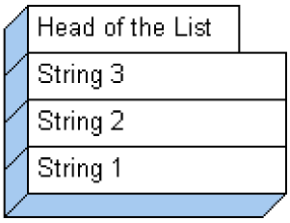


図 5: String 1 が追加されます。その後、String 2 が String 1 の前へ追加されます。最後に、String 3 が String 2 の前へ追加されます。

リスト内にある既存の要素を変更する

ListSetCurrentString 関数を呼び出して、文字列リスト内にある要素の値を変更します。現在の要素のみ変更が可能で、アップデートしたい文字列がリスト内で現在の文字列であることを確認して下さい。

ListSetCurrentItem 関数を呼び出して数値リスト内にある要素の値を変更します。上記と同じく、アップデートする項目はリスト内で現在の要素でなくてはなりません。

下の例では、**ListSetCurrentItem** を呼び出して数値リスト内の現在の項目の値を変更する方法をデモンストレーションします。**ListSetCurrentString** 関数も同じ要領で処理を行いますが、文字列リストと文字列変数を扱います。

```
// リストを作成し、その結果を確認します。
listID = ListCreate(NUMBERLIST);
if (listID = LIST_NULL) then
    MessageBox ("リストを作成できませんでした。", SEVERE);
    abort;
endif;


// 項目 (1078 と 304) をリストに追加します。
nItem = 1078;
ListAddItem (listID, nItem, AFTER);
nItem = 304;
ListAddItem (listID, nItem, AFTER);

// 現在の項目は 2 番目の項目 (304) です。
// ここで現在の項目を新しい値 (304) に設定します。
nItem = 305;
ListSetCurrentItem (listID, nItem);
ListDestroy(listID);
```

リストから要素を削除する

ListDeleteString 関数を呼び出して、リストから現在の文字列を削除します。または **ListDeleteItem** 関数を呼び出して、リストから現在の番号を削除します。他に削除する要素がない場合、これらの関数は END_OF_LIST 定数を戻します。

ListDeleteString と **ListDeleteItem** は現在の要素を削除するため、現在の要素を削除する要素へリセットしなくてはなりません。削除のあと、リストの次の要素が現在の要素となります。「リストのスキャン」で説明されているいずれかの方法で要素をリセットします。

以下の例は **ListDeleteString** を含むいくつかのリスト関数を具体的に説明します。**ListDeleteItem** 関数も同じ要領で利用されますが、異なる点はリストが数値リストであり、変数が数値変数であることです。

```
// 空白文字列リストを作成する。
listID = ListCreate (STRINGLIST);

if (listID = LIST_NULL) then  // 有効リストのテスト。
    MessageBox ("リストは作成されません", SEVERE);
endif;

// リストへ文字列を追加する。
szString = "String 1";
ListAddString (listID, szString, AFTER);
szString = "String 2";
ListAddString (listID, szString, AFTER);
szString = "String 3";
ListAddString (listID, szString, AFTER);

// 現在の文字列を削除します。
ListDeleteString (listID);
```

```
// リストにある現在の文字列を再設定します。
lResult = ListCurrentString (listID, svString);

// svString contains "文字列 2."
```

リストの特定要素を検索する

リストを降順にスキャンすることができます。リスト内の特定文字列あるいは番号要素を検索するには、**ListFindString** 関数または **ListFindItem** 関数を呼び出します。これら 2 つの関数は現在の要素から検索を開始し、その位置から続けて検索を行います。

リストの最初から検索を開始するには、**ListFindString** 関数または **ListFindItem** 関数を呼び出す前に **ListGetFirstString** 関数または **ListGetFirstItem** 関数を呼び出します。

ListFindString 関数または **ListFindItem** 関数が指定した文字列または番号を検出したとき、それがリストの現在の要素となります。

下の部分スクリプトでは、数値リストが作成され、(nItem 内の) 番号 1 が最初の要素として追加されます。そして、**ListFindItem** が番号 1 をリスト内で検索し、検出した場合はそれを削除します。最後に、リストが破棄されます。

```
listID = ListCreate(NUMBERLIST);

if (listID = LIST_NULL) then
    MessageBox ("リストを作成できませんでした。", SEVERE);
    abort;
endif;

nItem = 1;
ListAddItem (listID, nItem, AFTER);

if (ListFindItem (listID, nItem) = 0) then
     ListDeleteItem (listID);
endif;

ListDestroy(listID);
```



メモ: ***ListFindString*** *関数と* ***ListFindItem*** *関数は、現在の要素位置またはその後にある、指定した文字列または番号の最初のインスタンスのみを検索します。*

リストの 1 番目と 2 番目の要素を取得する

ListGetFirstString 関数または **ListGetFirstItem** 関数を呼び出して、リストから最初の文字列要素、または数値要素を戻します。読み出した要素が、リストの現在の要素になります。

ListGetNextString 関数または **ListGetNextItem** 関数を呼び出してリストの現在の要素の後に最初の文字列要素、または数値要素を戻します。読み出した要素が、リストの現在の要素になります。

```
// 空白文字列リストを作成する。
listID = ListCreate (STRINGLIST);

// 有効リストのテスト。
if (listID = LIST_NULL) then
```

```
    MessageBox ("リストは作成されません", SEVERE);
endif;

// リストへ文字列を追加する。
ListAddString (listID, "String 1", AFTER);
ListAddString (listID, "String 2", AFTER);
ListAddString (listID, "String 3", AFTER);

// リストをスキャンし、文字列をメッセージボックスに表示します。
lResult = ListGetFirstString(listID, szDriveName);

while (lResult !=  END_OF_LIST)
    MessageBox (szDriveName, INFORMATION);
    lResult = ListGetNextString (listID, szDriveName);
endwhile;
```

リストへファイルを読み込む

ファイル全体を文字列リストへ読み込むには、**ListReadFromFile** 関数を呼び出します。ファイルの各行はリストの要素となります。**ListReadFromFile** 関数を利用して、アイテムをひとつずつビルドする方法よりも簡単にリストをロードすることができます。

リストのインデックスを設定する

InstallShield が提供する **ListSetIndex** 関数を利用すると、インデックス番号を使ってある要素を現在の要素にすることができます。リストの特定の要素がどこにあるかが分かっている場合、**ListSetIndex** 関数を呼び出してその要素に直接アクセスすることが可能です。特定の要素を現在の要素へ設定するために、インデックスを利用してリストをどちらかの方向へトラバースすることができます。リストのインデックスは 0 (ゼロ) で始まります。

ListSetIndex 関数は、文字列リストでも数値リストでも使用できます。インデックスを付けた要素を現在の要素に設定した後で、**ListCurrentItem** 関数または **ListCurrentString** 関数を呼び出して、インデックスが付いた項目の値を返します。

この例は **ListSetIndex** を使ってリストを降順にスキャンする方法を説明します。

```
listID = ListCreate (STRINGLIST);
GetGroupNameList (listID);
nCheck = ListSetIndex (listID, LISTFIRST);

while (nCheck != END_OF_LIST)
    ListCurrentString (listID, svString);
     MessageBox (svString, INFORMATION);
     nCheck = ListSetIndex (listID, LISTNEXT);
 endwhile;

ListDestroy(listID);
```

ListCount 関数はリスト内の要素の数を報告します。**ListCount** 関数は **ListSetIndex** と共に上位インデックス値の設定に利用することもできますが、主に一般情報の目的の為に利用されます。たとえば、**ListCount** を呼び出してリストの数々の要素を取得し、その値を **ListSetIndex** と共に利用してリストをスキャンことができます。while ループを利用する前述の例は、リストの要素番号に基づいたループ用 InstallScript を使って下に説明されています。

```
listID = ListCreate (STRINGLIST);
GetGroupNameList (listID);
```

```
// リストの要素の数を取得します。
nItems = ListCount (listID);

// リストの要素の数を表示します。
SprintfBox (INFORMATION, "", "i = %d", nItems);

// ゼロで始まるループを nItems 回行い、
// メッセージボックスに順番に各リスト要素を表示します。
for i = 0 to (nItems - 1)
    ListSetIndex (listID, i);
    ListCurrentString (listID, svString);
    MessageBox (svString, INFORMATION);
    nCheck = ListSetIndex (listID, LISTNEXT);
endfor;

ListDestroy(listID);
```

リストのスキャン

InstallShield は増加順にまたは減少順に検討するためにこれらの関数を提供します:

テーブル 2 ・リストをスキャンするときに使用する関数

関数	説明
ListCount	指定したリストに含まれている文字列または数値要素の数を確認します。
ListCurrentItem	現在の項目をリストに戻します。
ListCurrentString	現在の文字列をリストに戻します。
ListFindItem	リストで数値要素を検出します。要素が見つかった場合、リストの現在の要素になります。
ListFindString	リストで文字列要素を検出します。要素が見つかった場合、リストの現在の要素になります。
ListGetFirstItem	番号リストから 1 番目のアイテムを読み出します。
ListGetFirstString	文字列リストから 1 番目の文字列を読み出します。
ListGetNextItem	番号リストから現在のアイテムの後のアイテム取得します。
ListGetNextString	文字列リストから現在のアイテムの後のアイテムを取得します。
ListSetIndex	リストの現在の要素をインデックスとします。

InstallShield は単一リンクリストを利用します。つまりインデックスを設定またはリスト内の特定の要素を検索する関数を利用しない限り、一番目の要素から最後の要素への一定方向のみにリストを増加してスキャンすることができます。

InstallShield では、索引と、リストの特定の要素を検索することによって、非増分的にリストを検討することができます。詳細については、それぞれの関数の説明を参照してください。

ほとんどのリストのトラバースとリストへのアクセス処理は、現在のリスト要素と関連付けて実行されます。さらに、リストのトラバースとリストへのアクセスに利用されるほとんどの関数は、現在の要素をそのアクションの結果として確立します。したがって、要素をリストの現在の要素とするのは、独立したアクションではなく、別のアクションの副産物といえます。

リストが空の場合にリストへ要素を追加すると、現在の要素を確立することになります。リストが空白でない場合にひとつの要素を現在の要素にするには、リストの検討またはリスト内の特定の要素の検索を行うのが最も有効です。



ヒント: *リストはしばしば while ループで処理され、通常 END_OF_LIST を確認します。無限ループはリストが無効な場合に起こります。リストを while ループで処理する場合、***ListCreate** *関数を利用してリストを作成したこと、そして* **ListDestroy** *関数を利用してリストを破棄していないことを確認してください。*

ファイルへリストを書き込む

文字列リストの内容をファイルへ書き込むには、**ListWriteToFile** 関数を呼び出します。リストの各要素はファイルの行となります。

下のスクリプト例は、 Autoexec.bat ファイルを listFile 文字列リストへ読み込み、その文字列リストを Autoexec.bak と名づけられたファイルへ書き込みます。

```
listFile   = ListCreate (STRINGLIST);
szPath     = "C:¥¥";
szFileOld  = "C:¥¥Autoexec.bat";
szFileNew  = "C:¥¥Autoexec.bak";

if (ListReadFromFile (listFile, szFileOld) < 0) then
   MessageBox ("ListReadFromFile が失敗しました。.", SEVERE);
endif;

ListWriteToFile (listFile, szFileNew);
```

InstallScript ファイルの保存



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallShield の [ファイル] メニューから [保存] をクリックすると、すべての開いているスクリプト ファイルが自動的に保存されます。

システムの復元



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

システムの復元は、エンドユーザーがソフトウェアインストール中に破損した PC を回復できる Windows の機能です。システムの復元機能は、エンド ユーザーの PCの 重要なシステム変更を自動的に監視し、記録します。システムの復元は、システムに危険をもたらす変更を簡単に取り消したり、システムが理想的に実行されていたときに戻すことを可能にすることにより、サポートコストを削減し、カスタマーの満足を向上させます。

InstallScript インストールではセットアップはファイル転送が始まる前に「*復元ポイント*」を設定してシステムの復元をサポートします。エンドユーザーは [システムの復元] を使用して、ファイル転送前の状態にシステムを復元できます。



メモ: *セットアップアクション (たとえばレジストリ変更やファイル変更) でファイル転送前に起きるものはシステムの復元でやり直しできません。*

ご使用のインストールはデフォルトでシステムの復元と互換性があります。(スクリプトの OnBegin イベント ハンドラーに以下のコードを挿入することにより、セットアップをシステムの復元機能に対応しないようにすることができます。

```
Disable( PCRESTORE );
```

Wininit.ini というファイルがターゲット マシンの Windows フォルダーに存在する場合、インストールは復元ポイントを設定できません。Wininit.ini を処理するには、**OnFirstUIBefore** と **OnMaintUIBefore** イベント ハンドラー関数に次のようなコードを挿入します。

```
/* Windows フォルダーで Wininit.ini を検索します。場所は...*/
if FindFile( WINDIR, "Wininit.ini", svResult )=0 then
    bRebootForSystemRestore = TRUE;
    /* ... サイズを取得します。*/
    GetFileInfo( WINDIR ^ "Wininit.ini", FILE_SIZE, nvSize, svResult );
    /* サイズがゼロバイトの場合 ...*/
    if nvSize=0 then
        /* ... Wininit.ini を削除します。*/
        if DeleteFile( WINDIR ^ "Wininit.ini" )=0 then
            bRebootForSystemRestore = FALSE;
        endif;
    endif;
    /* Wininit.ini がゼロ以外のサイズの場合、または削除不可能な場合はエンドユーザーに対して通知して再起動を可能にします。*/
    if bRebootForSystemRestore then
        szQuestion = "[Windows システム回復] を利用して 、コンピューターの変更を" +
            "元に戻すことができます。[システム回復] を利用して" +
            "このインストールを下に戻すようにできるようにしたい場合、 " +
            "コンピューターを今すぐ再起動してください。¥n¥n" +
            "コンピューターを今すぐ再起動しますか?"
        if AskYesNo( szQuestion, YES )=YES then
            System( SYS_BOOTMACHINE );
        endif;
    endif;
endif;
```

プロパティの取得と設定



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

Windows Installer プロパティには、製品、セットアップ、オペレーティング システム、ユーザーに関する有用な情報が保持できます。**MsiGetProperty** および **MsiSetProperty** を呼び出して、InstallScript カスタム アクション中でこれらのプロパティと直接対話することができます。



メモ: *MsiGetProperty プロパティおよび* MsiSetProperty *プロパティは、遅延 InstallScript カスタム アクションに使用することはできません。これは、アクティブな .msi データベースへのアクセスが不可能であり、Windows Installer プロパティを認識しないためです。実行スクリプトに書かれた情報のみにアクセスすることができます。*

次の例では、Windows Installer セットアップパッケージからユーザー名を取得して、値を確認し、ユーザーにそれを変更する機会を与え、データベースに新しい値を書き込んでいます。



メモ: *プロパティ値が 1024 文字を超える場合、適切なバッファ サイズを指定する必要があります。この状況におけるコードの記述例については、「プロジェクトを InstallShield 2011 以前からアップグレードする」トピック内の「InstallScript カスタム アクションで呼び出される一部の MSI API の動作を変更する」を参照してください。*

```
//ビルトイン関数のヘッダー ファイルを含める
#include "isrt.h"
//MSI API 関数と定数のヘッダー ファイルを含める
#include "iswi.h"

export prototype Func1(HWND);

function Func1(hMSI)
   STRING svName;
   NUMBER nvSize, nResponse;
begin
   //MSI データベースからユーザーの名前を読み出す
   nvSize = 256;
   MsiGetProperty (hMSI, "USERNAME", svName, nvSize);

   nResponse = AskYesNo ("登録名は、" +
                  svName + "です。間違いないですか?", YES);

   if nResponse = NO then
      AskText ("この製品の登録名を" +
            "入力してください。", svName, svName);
      MsiSetProperty(hMSI, "USERNAME", svName);
   endif;
end;
```

MsiGetProperty (Windows Installer ヘルプ ライブラリ)
MsiSetProperty (Windows Installer ヘルプ ライブラリ)

ビット フラグの使い方



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

ビットフラグは、単一数値変数に格納されているひとつまたは複数 (最大 32) の Boolean 値です。

一般的に各ビット フラグには、関連付けられた対応の定義済み定数があります。この定数はフラグ用のビットが 1 に設定されており、その他のビットは 0 です。たとえば、次の定数は、0 ビット用のビット フラグを認識します。つまり、番号の一番右側にあるビットです。

```
#define BITFLAG_EXAMPLE_1 0x00000001
```

その他のビット フラグは次の通りです:

```
#define BITFLAG_EXAMPLE_2 0x00000002
#define BITFLAG_EXAMPLE_3 0x00000004
```

0x00000003 は有効なビット フラグでない点に注意してください。なぜならこの値は 1 に設定された番号にある 2 つのビットに対応するからです。

変数でビットフラグを設定する

変数でビット フラグを設定するには、OR 演算子 (|) を利用します。たとえば、数値変数 *nFlags* の 値 BITFLAG_EXAMPLE_1 および BITFLAG_EXAMPLE_2 を割り当てる場合、次のようなコードを使用します。

```
nFlags = nFlags | BITFLAG_TEST_1 | BITFLAG_TEST_2;
```

変数からビット フラグをクリアする

変数からビット フラグをクリアするには、ビット単位の AND 演算子 (&) とビット単位 NOT 演算子 (~) を組み合わせます。たとえば、BITFLAG_TEST_1 フラグを nFlags から削除するには、次のようなコードを使用します。

```
nFlags = nFlags & ~BITFLAG_TEST_1;
```

ビットフラグ用の変数をテストする

ビットフラグ用の変数をテストするには、ビット単位 AND 演算子 (&) を利用して、テストするビット以外のビットフラグ定数をすべてゼロにします。たとえば、*nFlags* が現在 **BITFLAG_TEST_1 | BITFLAG_TEST_2** に設定されている場合、後に続く式は true 評価 (つまり、0 以外の結果) を行います。

```
nFlags & BITFLAG_TEST_1
```

& 演算子によって、一番右にあるバイト以外はすべて 0 に設定されています。定数 BITFLAG_TEST_1 と *nFlags* の両方が 1 なので、一番右のビットは 1 となります。

次の式は、nFlags の 3 番目のビットが 0、そしてその他のビットが 演算子によって 0 に設定されているので、FALSE 評価します。

nFlags & BITFLAG_TEST_3

文字列比較



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallShield が 2 つの文字列を比較するとき、まず最初の文字列の最初の文字と 2 番目の文字列の最初の文字とを比較します。これらの文字が等しい場合、InstallShield は各文字列の次の位置にある文字を比較します。これらの文字も等しい場合は、その次の位置にある文字に移動し、下の条件のひとつに合致するまでシーケンスを続けます:

- 2 つの文字列の対応する位置にある 2 つの文字が一致しません。この場合、InstallShield はこれら 2 つの文字の比較に基づいて解決します。文字列 1 の文字が文字列 2 の対応する位置にある文字よりも大きな値を持つとき、文字列 1 が大きくなります。それ以外の場合は文字列 2 が大きくなります。
- 対応する位置で異なる文字を検出せずに文字列 1 の終わりに到達しました。この場合、文字列の長さは異なり、つまり等しくないことになります。
- 対応する位置で異なる文字を検出せずに両方の文字列の終わりに到達しました。この場合、文字列の長さは同じで、文字のすべてが一致します。つまり、文字列が等しいことになります。

次の例を参考にして下さい:

```
svString1 = "trusting";
   svString2 = "TRUTHFUL";

   if svString1 = svString2 then
      MessageBox("等しい", INFORMATION);
   else
      MessageBox("等しいくない", INFORMATION);
   endif;
```

文字列比較で大文字と小文字は区別されません。大文字が対応する小文字と等しいため、InstallShield は **trusting** の最初の 3 文字と **TRUTHFUL** の最初の 3 文字が同等であると判断します。比較は各文字列の 4 番目の文字のテストで終了します。ASCII 文字テーブルで **s** は、**t** よりも下位にあるため、svString1 と svString2 は等しくありません。残りの文字は比較されず、else ブランチが実行されます。



メモ: *各文字の値は、その ASCII 値に基づきます。特定の文字や記号の ASCII 値に関する情報については、基本のプログラミングマニュアルを参照してください。*

ヌル区切り文字列の使用



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

*「複数ヌル区切り文字列」*または*「複数ヌル文字で終わる文字列」*とも呼ばれる、ヌル区切り文字列、2 つのヌルで終わる文字列 (例、**abc¥0def¥0ghi¥0¥0**) を利用するときは次の点に注意してください。

- ヌル区切り文字列値へ設定される変数を宣言する際、変数のサイズを明示します。例:

  STRING szString[1024];

  オートサイズは使用しないで下さい。インストールエンジンはオートサイズされた文字列はヌル文字を文字列内に含まないものとみなします。

- ヌル区切り文字列の配列を作成しないで下さい。配列要素は常にオートサイズされなくてはならないため、文字列配列は現在この種類の文字列をサポートしません。
- ヌル区切り文字列の指定されたサイズは保存される文字数以上と、さらにヌル文字 2 つ分を含まなくてはなりません。
- 文字列は自動的にヌル文字に初期化されるため、以前に最後から 2 番目の文字をヌル以外に設定していない限り、これをヌルと明示する必要はありません (明示したところで悪影響がもたらされる訳ではありません)。
- ヌル区切り文字列の長さを判断する最も有効な手段は、**CharReplace** 関数を呼び出してヌル文字を置換してから、**StrLengthChars** 関数を呼び出して結果文字列のサイズを判断する方法です。
- **StrGetTokens** 関数と **StrPutTokens** 関数は、ヌル区切り文字列を使った作業に便利です。
- 前述の関数以外、ほとんどの InstallScript 関数は複数ヌル区切り文字列と共に利用することはできません。ビルトイン関数と共に複数のヌル区切り文字列を利用する場合、まず **CharReplace** 関数を利用して、たとえば、アンダースコア (_) のような別の文字を使ってヌル文字を置換します。

相対パス

相対パスは、現在のドライブ上の現在のフォルダーを起点にして、ファイル検索に必要な情報をすべて含みます (例、Support¥Validation)。フォルダーは、現在のディレクトリに存在する場合のみ、相対パスで検索することができます。

長いファイル名

32 ビット Windows オペレーティング システムは、長いファイル名をサポートします。長いファイル名を使うと、エンドユーザーはディレクトリやファイルにより分かりやすい意味のある名前をつけることができます。用語 "*長いファイル名*" は、長いファイル名と長いパスの両方を意味します。

InstallShield が提供する長いファイル名関数は、長いファイル名を認識しない 16-bit アプリケーションや 32-bit アプリケーションのインストールを可能にします。アプリケーションの要件を判断するのは開発者の判断に委ねられています。InstallShield は、あらゆる種類のアプリケーションのインストールに役立つツールを装備しています。

長いファイル名と 16 ビット アプリケーション

長いファイル名を利用して 16 ビット アプリケーションを起動することができますが、長いファイル名を 16 ビット アプリケーションへ引数として渡すことはできません。16 ビット アプリケーションが適切に機能するには、長いファイル名の短いファイル名バージョンが必要です。

インストールがファイル名を 16-bit アプリケーションへ渡す場合、アプリケーションへの有効な短いファイル名を用意しなくてはなりません。長いファイル名では動作しません。

たとえば .ini ファイルや 16-bit アプリケーションが利用するファイルへインストールがファイル名を書き込む場合、16-bit アプリケーションが利用する短いファイル名を書き込まなくてはなりません。

長いファイル名と二重引用符

32 ビット オペレーティング システムで、コマンドラインに 1 つ以上の空白文字を含む長いファイル名を渡した場合 (例: DOS シェル内や、アイコンの "コマンドライン" 設定内)、長いファイル名を二重引用符で囲まなくてはなりません。これは、コマンドラインが空白文字を他の式からコマンドを区別する区切り文字として認識するために必要となります。二重引用符は長いファイル名を文字列リテラルへ変換して、コマンドラインがそれを単独の引数として受け取る事を可能にします。

32 ビット オペレーティング システムで、1 つまたは複数の空白文字を含む長いファイル名を渡す場合、それを二重引用符で囲まなくてはなりません。しかしながら、Windows NT によるアイコン ファイルへのアクセス形態のため、二重引用符で囲んだ長いファイル名がアイコンの "コマンドライン" 設定で利用されている場合、アプリケーションのアイコンの代わりにデフォルトの Windows アイコンが表示される場合もあります。

製品のアイコンを表示するには、プログラム フォルダーへアイコンを追加するときに、

関数 AddFolderIcon のパラメーター szIconPath でアイコンのパスを指定することができます。

InstallShield で短いファイル名へ変換する前に、長いファイル名から二重引用符を削除しなくてはなりません。LongPathToQuote 関数と LongPathToShortPath 関数を参照してください。

長いファイル名の形式

長いファイル名は従来の 8.3 (8 文字と 3 桁拡張子) の短いファイル名制限よりも長い名前を持ちます。長いファイル名では短いファイル名で利用される文字すべてを利用することができます。さらに、長いファイル名にはプラス記号 (+)、コンマ (,)、セミコロン (;)、等号 (=)、左右角括弧 ([ ])、そして空白を含むことができます。

先頭と行末の空白は無視されます。完全修飾の (最後のヌル文字を含む) 長いファイル名は、NTFS ファイルシステム上では 256 文字まで、そして VFAT ファイルシステム上では 260 文字まで含むことができます。

デフォルトでは、オペレーティング システムが長いファイル名それぞれに短いファイル名を作成します。短いファイル名は長いファイル名の最初の 6 文字と、チルダ (~)、そして数字で構成されます。

コンパイラで定数を定義する

InstallScript 定数をスクリプト内ではなく、[設定] ダイアログ ボックスの [コンパイル/リンク] タブで定義することができます。

このタブで定数を定義する際、次のような制限が適用します :

- 数値定数のみを定義できます。
- スクリプト内で #define ステートメントを使って定義した定数を定義すると、エラーが発生します。
- スクリプト内で変数として定義した定数を定義すると、定数の値が実行時に失われます。

スクリプトで Windows 定数を使用する

一部の Windows 定数は、*InstallShield Program Files フォルダー*¥Script¥Isrt¥Include フォルダーにある ISRTWindows.h ファイルで既に定義されています。このファイルは、スクリプトに Ifx.h を含めると、自動的にインストールに含まれます。ISRTWindows.h に定義されている定数を再定義する必要はありません。再定義すると、コンパイラ警告が発生します。どの定数が定義済みかを判別するには、ISRTWindows.h ファイルを参照します。

ISRTWindows.h に定義されていない定数を使用する場合は、インストールスクリプトの宣言ブロックに必ずそれらを定義しなければなりません (#define を使用)。通常 C++ プログラムの一部である Windows.h ファイルを、単純挿入することはできません。未定義定数に割り当てる必要がある値は、通常は、該当する Windows SDK や開発ツール付属のインクルードファイルに含まれています。(Microsoft Visual C++ の場合、ほとんどの定数は *InstallShield Program Files フォルダー*¥Script¥Resource フォルダーにある Winuser.h ファイルに含まれています。)

長い文字列リテラルのコーディング

スクリプトで長い文字列リテラルを読み易くするため、文字列を 2、3 行に分割して順に並べ、連結させます。InstallScript では、連結にはプラス記号 (+) が使われます。次の例では、長い文字列リテラルを前述の方法でいくつかのコードラインに分割する方法を示します:

```
szMonths = "1 月、2 月、3 月、4 月、" +
        "5 月、6 月、7 月、8 月、9 月、" +
        "10 月、11 月、12 月";
```

絶対パス



***プロジェクト・** この情報は、InstallScript プロジェクトに適用します。*

指定されたドライブのルート ディレクトリを起点にしてファイル検索をするのに必要な情報をすべて含む絶対パス。たとえば、C:¥Program Files¥InstallShield¥2022 は、C ドライブにインストールされたときの InstallShield フォルダーへの絶対パスです。

関数のビルド

一般的に、関数はファイルの最初で、また本文はファイルの最後で宣言されます。関数プロトタイプを宣言した後で、関数ブロックで関数そのものを定義する必要があります。各関数ブロックには 1 つの関数だけが含まれています。



タスク ***関数をビルドするには、以下の手順を実行します。***

1. 関数本体をキーワード **function** で始めます。
2. 関数プロトタイプに利用されている戻り値データタイプを入力する、または関数プロトタイプが **void** の戻りタイプを指定する場合、void を入力します (これは関数が値を戻さないことを示します。)。関数プロトタイプが戻り値の型を指定しない場合、空白のままにします。
3. 関数名を入力します。
4. 関数名の右にかっこの中で使用する引数名を入力します。引数は宣言ブロックに宣言するパラメーターのデータタイプに対応している必要があります。

   ***メモ・** InstallScript 関数では割り当てステートメントをパラメータとして渡すことができません。さらに && や || 演算子を関数への引数の中で使用することはできません。*

   ステップ 1 と 2 は関数ヘッダー を作成します。関数ヘッダーは InstallScript ではセミコロンで終わりません。

5. 関数で使用するローカル変数を指定します。次に **begin** というキーワードを区切り記号を入れずに入力します。

6. begin の行の次に、特定のタスクの実行に必要なステートメントを追加します。

   関数から特定の値を返す場合は特に、return ステートメントを使用することもできます。関数から値を戻す方法については次を参照して下さい。

7. 関数はキーワード **end** で終了します。

サンプル関数ブロックを以下に示します。

```
function GetPathParts(szFullPath, svDrv, svPath, svName)
    LONG lResult;
begin
    lResult = ParsePath(svDrv, szFullPath, DISK);
    if (lResult = 0) then
        lResult = ParsePath(svPath, szFullPath, DIRECTORY);
    endif;

    if (lResult = 0) then
        lResult = ParsePath(svName, szFullPath, FILENAME);
    endif;

    return lResult;
end;
```



メモ: *ユーザー定義関数は return ステートメントで値を戻すことが可能です。その他のデータ型は BYREF 演算子を使って宣言したパラメーターで戻すことが可能です。*

関数の呼び出し

すべての関数 (ユーザー定義関数、ビルトイン関数、Sd 関数、外部関数)は、同じ方法で呼び出されます。関数名を入力し、その次に関数に渡すパラメーターを入力します。例:

```
MyFunction (MyAge, MyHeight, MyWeight);
```



メモ: *InstallScript 関数では割り当てステートメントをパラメータとして渡すことができません。さらに && や || 演算子を関数への引数の中で使用することはできません。*

リファレンスによって関数に渡されるオートサイズ文字列変数は呼び出された関数の中では自動サイズ調整されません。関数が現在のパラメーターのサイズより大きい長さの値を割り当てようとすると、ランタイム エラー エラー401 が発生します。このエラーを回避するためには、リファレンスが関数に値を渡すときの特定の文字列サイズを宣言します。

DLL ファイル関数の呼び出し

InstallScript インストール スクリプトから DLL ファイル関数を呼び出す際に、注意する 3 つの規則があります。

- DLL ファイル名の長さは、最大 33 文字で、関数名は 63 文字です。

- .dll ファイルを呼び出す際、インストールは幅が 4 バイトを超える合成パラメーターを処理することができません。ただし、パラメーターを合成構造を指すポインターにすることは可能です。
- アドレスがポインター変数に保存されている文字列変数を、ポインターを DLL ファイル関数に渡して変更することはできません。DLL ファイル関数で文字列変数値を変更できるようにするには、BYREF 演算子を指定して引数のデータ型を宣言してから、変数そのものを引数として関数に渡します。



タスク　スクリプトから DLL ファイル関数をスクリプトから呼び出すには、以下の手順に従います:

1. DLL ファイルをサポート ファイルとしてプロジェクトに追加していない場合、まずこれを行います。詳細については、「サポート ファイルを追加する」を参照してください。
1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **InstallScript** エクスプローラーで、DLL 関数を呼び出す InstallScript ファイル (.rul) をクリックします。
3. スクリプトの先頭で、次の構文を使用して関数のプロトタイプを宣言します。

   prototype [CallingConvention] [ReturnType] DLLName.FunctionName(ParamType1, ParamType2, ...);

   例:

   prototype BOOL MyDLL.MyFunction(INT, INT, INT);

   呼び出し規則は cdecl または stdcall に指定できます。呼び出し規則を特に指定しなかった場合、InstallShield は stdcall を使用します。

   DLL ファイル名は大文字と小文字を区別することに注意してください。たとえば、スクリプト コンパイラは *MyDLL.MyFunction* および *mydll.MyFunction* を異なる関数として処理します。User32.dll、Gdi32.dll、または Kernel32.dll から関数のプロトタイプを宣言する際、USER、GDI、または KERNEL のキーワードを DLL 関数名の代わりに使用できます。

   戻り値には、LIST 以外のすべての InstallScript データ タイプを指定できます。ワイド文字の文字列引数または Unicode 文字列引数が予想される DLL ファイル関数呼び出しを宣言するとき、wstring データ タイプを利用してください。戻り値の型を指定しなかった場合、インストールは DLL ファイル関数が 4 バイト値を返すものと想定します。

4. **UseDLL** 関数を呼び出して DLL ファイルをロードします。例:

   UseDLL(SUPPORTDIR ^ "MyDLL.dll");

   メモ・ *_isuser.dll や _isres.dll、または User32.dll、Gdi32.dll、および Kernel32.dll などの Windows API DLL ファイルをロードする必要はありません。これらの DLL ファイルのロードまたはアンロードのために、* **UseDLL** *および* **UnUseDLL** *を呼び出さないで下さい。*

5. 他の関数と同様に、関数を呼び出します。例:

   bResult = MyDLL.MyFunction(nInt1, nInt2, nInt3);

   上記の例のように、関数呼び出しに DLL ファイル名を含めることをお勧めします。スクリプトが別の DLL から同じ名前の関数を宣言した場合、DLL 名を含まずに関数を呼び出した時にコンパイル エラーの原因となります。

   呼び出された関数を DLLファイルで検出できなかった場合、インストールは例外を発生します。これは try...catch...endcatch block を使って処理することができます。

6. DLL へのスクリプトの呼び出しがすべて完了すると、**UnUseDLL** を呼び出して DLL をアンロードします。例:

```
UnUseDLL( SUPPORTDIR ^ "MyDLL.dll" );
```



メモ: *IS_NULLSTR_PTR 変数を使って、ヌル ポインターを InstallScript 文字列としてプロトタイプされているパラメーターを通して 外部 DLL 関数または Windows API に渡すことができます。詳細については、「IS_NULLSTR_PTR」を参照してください。*

DLL ファイル関数に配列を渡す

InstallScript 配列は、内部的にはビルトイン C または C++ 配列ではなく OLE オートメーション セーフ配列としてフォーマットされます。InstallScript 配列を C/C++ 配列が必要な DLL 関数に渡すためには、GetCArrayFromISArray を呼び出すことによって配列データを要求フォーマットに配置しなければなりません。

関数の宣言

ユーザー定義関数を作成する最初のステップは、関数を宣言することです。キーワード *prototype* は、その行に関数定義があることを InstallShield のスクリプトコンパイラに通知します。



タスク ***関数を宣言するには、以下の手順を実行します。***

1. キーワード **prototype** を入力します。
2. 関数の戻り値タイプを入力します。(この手順はオプションです。戻り値タイプを入力しなかった場合、関数が NUMBER 値を戻す、あるいは値を戻さないものと判断します。)関数が値を戻さないようにするには、**void** を入力します。
3. 同じ行で、関数名を入力します。
4. 関数名の後、かっこ内にパラメーターのデータ タイプをコンマで区切って入力します。
5. パラメーターがない場合は、関数名の右側に空の括弧を挿入します。
6. セミコロン (;) で行を終了します。

次の例で、**FunctionName** は、3 つのパラメーターを含む関数です。**FunctionName** を呼び出す際に渡される引数は、順に INT、STRING、SHORT です。**CopyBitmapExample** にはパラメーターがありません。**FileTransfer** には 5 つのパラメーター (3 つの LONG 変数と 2 つの STRING 変数) があり、NUMBER 値を戻します。

```
prototype FunctionName (INT, STRING, SHORT);
prototype CopyBitmapExample( );
prototype NUMBER FileTransfer (LONG, LONG, LONG, STRING, STRING);
```

DLL 関数を宣言する際、DLL ファイル関数の名前に <DLL ファイル名>.<関数名> という形式を使用します。例:

```
prototype MyDLL.MyFunction (INT, INT);
```

上記の宣言は、プログラムが 2 つの INT パラメーターを使用して **Mydll.dll** というファイルにある **MyFunction** という関数を呼び出すよう、InstallScript コンパイラに通知します。

オプションとして、DLL ファイル関数を宣言するとき、呼び出し規則の cdecl または stdcall を宣言することもできます。例:

```
prototype cdecl MyDLL.MyFunction(INT, INT);
```

呼び出し規則を明示的に宣言しなかった場合、InstallShield は stdcall を使用します。呼び出し規則とリ戻り値の型の両方を宣言する場合、戻り値の型の前に呼び出し規則を配置します。



メモ: *Windows API 関数のほとんどは stdcall 呼び出し規則を利用しますが、C または C++ 開発環境の中には、C または C++ 関数を _stdcall 修飾子と共にプロトタイプ化しない限り、cdecl 呼び出し規則を使って DLL ファイル関数をビルドします。さらに詳しい情報は、コンパイラについての資料をご覧下さい。*

多くの Windows API 関数はヘッダーファイル ISRTWindows.h *で宣言され、*Ifx.h *スクリプトに含んだときに自動的に含まれます。(*[設定] *ダイアログ ボックスの* [コンパイル/リンク] *タブにある* [プリプロセッサ定義] *ボックス内にプリプロセッサ定数 ISINCLUDE_NO_WINAPI_H を配置することで Windows API が自動的に定義されないようにすることが可能です。)*

関数から値を戻す

InstallScript に組み込まれた関数と同様に、ユーザー定義関数も呼び出し側に値を戻すよう設計することができます。関数から値を戻すには、関数の end ステートメントの前に return ステートメントとそれに続く値を挿入します。return ステートメントを含まない場合、またはキーワード return の後に値を指定しない場合、関数によって戻される値は予測できません。(関数プロトタイプが void 戻り値タイプを指定する場合、関数は値を戻すことができません。)

多くのプログラマは、return ステートメントを使用して、関数呼び出しが成功したか失敗したかを示すエラーコードを戻します。InstallScript のビルトイン関数の多くは、この目的のためだけに return ステートメントを使用します。また、return ステートメントは、関数に渡されたパラメーターに行われる処理の結果を戻す関数を作成するためにも使用されます。以下に例として、長方形の面積を戻す場合を示します。

```
function RectangleArea (nLength, nWidth)
    INT nVal;
begin
    nVal = (nLength * nWidth);
    return nVal;
end;
```

キーワード return の後には、定数、変数、数値、文字列式、または関数呼び出しを使用できます。以下の例では、**RectangleArea** を変更し、代入ステートメントを省いてキーワード return の後に算術式を使用しています。

```
function RectangleArea (nLength, nWidth)
begin
    return (nLength * nWidth);
end;
```

関数から戻された値は、呼び出し側のプログラムや関数で無視することもできますが、条件式でテストしたり変数に代入することもできます。以下の例では、**RectangleArea** からの戻り値を変数 *nArea* に割り当てます。

```
nArea = RectangleArea (nLong, nWide);
```

次の例では、**RectangleArea** の結果が条件式でテストされます。

```
if (RectangleArea(nLong, nWide) > nMaxArea) then
        MessageBox("範囲が制限を越えています。", INFORMATION);
    endif;
```

複数の値、または数値以外の値を戻すには、BYREF 演算子を使用して、参照から渡されるパラメーターを定義します。

サポートされていない関数

InstallShield Professional および InstallShield–Windows Installer Edition で利用可能な関数の一部は、InstallShield ではサポートされていません。

コンポーネント関数

InstallShield では、機能はエンド ユーザーの視点から見たプロジェクトで最も上位にあたる構成区分です。このため、従来のコンポーネント関数は、機能関数となります。たとえば、**ComponentDialog** は現在、**FeatureDialog** になっています。では、従来のコンポーネント関数のほぼすべてを機能関数として使用できます。



プロジェクト: *機能関数が使用できるのは、InstallScript および InstallScript MSIプロジェクトのみです。基本の MSI プロジェクトについては、インストールに機能関数を使用する場合、InstallScript MSI プロジェクト タイプにプロジェクトを変換する必要があります。*

PreShowComponentDlg および PostShowComponentDlg

InstallShield – Windows Installer Edition 2.x の場合、コンポーネント関連の関数 (現行の機能関連の関数) を使用するには、**PreShowComponentDlg** および **PostShowComponentDlg** 関数を呼び出す必要がありました。InstallShield の場合、機能関連の関数が使用できるのは、InstallScript および InstallScript MSI プロジェクトのみです。基本の MSI プロジェクトについては、スクリプトで機能関数を使用する場合、InstallScript MSI プロジェクトにプロジェクトを変換する必要があります。

機能関数は基本の MSI プロジェクトでの使用をサポートしていないため、**PreShowComponentDlg** および **PostShowComponentDlg** 関数は不要となり、サポートされなくなりました。

エントリポイント関数の書き込み



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript カスタム アクションを含む基本の MSI*
- *InstallScript カスタム アクションを含む InstallScript MSI*
- *InstallScript カスタム アクションを含むマージ モジュール*

Windows Installer が InstallScript カスタム アクションを実行する際に、InstallShield はカスタム アクションを作成したときに指定した関数を呼び出します。すべての InstallScript カスタム アクションは、スクリプトへのエントリポイントとして、エクスポートされたユーザー定義関数が必要です。

関数のプロトタイプ宣言と定義

エントリ ポイント関数は、他の関数と同じようにプロトタイプ宣言され、定義されます。ただし、この関数には以下の要件があります。

- プロトタイプには、エクスポートされた関数であることを宣言するキーワード export が含まれます。

- 関数の引数は 1 つだけで、引数は .msi データベースへのハンドルでなければなりません。
- カスタム アクションが戻り値を待つように設計されている場合は、Windows Installer に解釈できる値を戻します。上記のカスタム アクションの戻り値は IsMsiQuery.h に定義されており、スクリプトに IsMsiQuery.h または Iswi.h をインクルードすれば使用できます。

以下のスクリプト例では、正常終了した場合に成功値を戻すエントリポイント関数が宣言されています。

```
//ビルトイン InstallScript 関数プロトタイプの Isrt.h のインクルード
#include "isrt.h"

キャッシュには次の 2 種類あります:
#include "iswi.h"

export prototype MyFunction(HWND);

function MyFunction(hMSI)
    STRING szKey, svValue, svPath;
    NUMBER nvType, nvSize, nReturn;
begin
    RegDBSetDefaultRoot (HKEY_LOCAL_MACHINE);

    szKey = "SOFTWARE¥¥Microsoft¥¥Windows¥¥CurrentVersion¥¥App Paths¥¥isdev.exe";
    nReturn = RegDBGetKeyValueEx (szKey, "Path", nvType, svValue, nvSize);

    //App Paths キーは、InstallShield がインストールされたフォルダーと、
    //その後に続くセミコロンで構成されています。
    StrSub (svPath, svValue, 0, StrFind(svValue, ";"));

    if nReturn = 0 then
        MessageBox ("InstallShield のインストールされている場所 " + svPath, INFORMATION);
        return ERROR_SUCCESS;
    else
        MessageBox("InstallShield をインストールした場所が不明です。", SEVERE);
        return ERROR_INSTALL_FAILURE;
    endif;
end;
```

複数のエントリポイント

複数のカスタム アクションを実行するために、スクリプトに複数のエントリポイント関数を持たせることができます。ただし、InstallShield は各 InstallScript カスタム アクションに対して指定された 1 つのエントリポイント関数のみ呼び出します。

関数ウィザードで関数の呼び出しを追加する

関数ウィザードを使用して関数の選択、引数の指定、スクリプトへの関数の挿入を行うことができます。

宣言済みの Windows API 関数



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

Windows API 関数はヘッダーファイル ISRT.h で宣言され、Ifx.h スクリプトに含むと自動的にインストールに含まれます。スクリプト コードでこの関数が明示的に宣言された場合、次のどれかを行ってください。

- 関数宣言を削除し、InstallShield が提供する宣言を使用します。
- InstallShield 2025 および InstallShield Professional 7 より前のバージョンの両方と互換性のあるスクリプトが必要な場合、次のようなコードを使って宣言を囲みます:

  #if _ISCRIPT_VER < 0x700prototype ...#endif

また、InstallShield が定義した宣言と利用する宣言とが異なる場合、Windows API 関数を呼び出すコードもアップデートする必要があります。

[設定] ダイアログ ボックスの [コンパイル/リンク] タブにある [プリプロセッサ定義] ボックス内にプリプロセッサ定数 ISINCLUDE_NO_WINAPI_H を配置することで Windows API が自動的に定義されないようにすることが可能です。

デフォルト以外の機能イベント ハンドラー関数を指定する

デフォルトで、Setup.rul が InstallScript ビューに存在し、<機能名>_<イベント名> 関数を定義する場合 (例 **NewFeature1_Installing**)、その関数はその機能のイベントのハンドラーです。

機能イベントのハンドラーを別の関数を選択することも可能です。これには、InstallScript ビューの任意のスクリプト ファイルで定義された任意の関数を利用できます。ただし、引数を必要とせず次のようなエキスポートキーワードを利用して宣言されていない場合に限ります:

```
export prototype MyFeatureHandler();
```



タスク ***デフォルト以外の機能のイベント ハンドラー関数を指定するには、次の操作を行います:***

1. スクリプト ファイル (.rul ファイル) を **InstallScript** ビューで開きます。
2. 次のいずれかを スクリプト エディター ペインで実行します。
   - 機能イベント ハンドラーとして指定する関数を宣言および定義します。
   - スクリプト エディター ペインの上部にある [イベント カテゴリ] と [イベント] リストで、それぞれ機能とイベント ハンドラーを選択します。InstallShield は自動的にイベント ハンドラーを作成します (たとえば Feature1Installing)。



メモ: *機能名に括弧を使うと、デフォルト形式 <機能名>_<イベント名>の名前をもつイベント ハンドラー関数は正常にコンパイルしません。*

グローバル変数へのアクセス



プロジェクト: *この情報は、InstallScript MSI プロジェクトに適用します。*

グローバル変数は、関数の外で宣言され、スクリプト中の任意のモジュールから使用できるデータです。グローバル変数を使用することにより、イベント ハンドラー、エントリ ポイント関数、ユーザー定義関数による持続データを共有できます。

たとえば、OnBegin イベント ハンドラーでオペレーティング システムのバージョン値を得た場合、エントリポイント関数でそのグローバル値にアクセスできます。次に引用したスクリプトは、この設定について説明しています。

```
//ビルトイン関数のヘッダー ファイルを含める
#include <isrt.h>
//イベント ハンドラーと MSI API のヘッダーファイルをインクルード
#include <iswi.h>

    //グローバル変数の宣言
    NUMBER nvResult;

    //エントリポイント関数のプロトタイプ宣言
    export prototype MyFn(HWND);

function OnBegin()
    //ローカル変数の宣言
    STRING svResult;
begin
    GetSystemInfo (WINMAJOR, nvResult, svResult);
end;

function MyFn(hMSI)
begin
    if nvResult > 4 then
        //Windows 4.0 より新しいバージョンのコード
    else
        //Windows 4.0 のコード
    endif;
end;
```



重要 *・グローバル変数の状態は、すべてのイベント ハンドラーで共有されます。ただし、プロジェクト用の InstallScript カスタム アクションを作成した場合、グローバル変数は InstallScript カスタム アクションからイベントドリブン型 InstallScript コードへ、またはその逆に状態が共有されることはありません。したがって、イベントドリブン型 InstallScript コードでグローバル変数を宣言した場合、InstallScript カスタム アクションでその変数を利用することはできません。同様に、InstallScript カスタム アクションで変数を宣言した場合、イベントドリブン型 InstallScript コードでその変数を利用することはできません。*

初期化 (.ini) ファイル エントリのアンインストール

アンインストールは、ログ記録が有効な状態で InstallScript 初期化ファイル関数を使って加えられた変更を削除します。さらに詳しい情報は、説明書のこのセクションをご覧ください。

初期化 (.ini) ファイル エントリのアンインストールについての一般制限事項

.ini ファイルエントリおよび変更点を自動的にアンインストールする際、適用される以下の制限事項が適用されます:

- .ini ファイル関数が呼び出されたときにログ記録が有効でなくてはなりません。
- .ini ファイルが .ini ファイル関数を使って作成された場合、それは削除されません。

- セクション名が .ini ファイル関数を使って作成された場合、キーが存在しなくてもそれは削除されません。
- InstallScript 関数によってキー値が (前後に追加されるのではなく) 完全に置き換えられている場合、インストールの前に存在したキー値は 修復されません。この場合、InstallShield は置き換えられたキーを新規に作成されたキーとみなし、インストールが作成した新しいキーと同様にアンインストールします。
- アンインストールは **WriteProfString** を使って削除したキーおよび値を修復しません。
- 既存するキーに値を追加するには (たとえば、[386Enh] の下にあるネットワーク)、新しいエントリは既存する文字列の終わり又は一番最初に追加しなくてはならず、文字列に挿入することはできません。
- コンマ (,) およびセミコロン (;) のみが有効な区切り文字です。アンインストールする文字列値が長い文字列の一部として検出された場合、区切り文字がコンマまたはセミコロンの場合のみ文字列の位置に基づいてその値および前後の区切り文字も適切に削除されます。たとえば、アンインストールが文字列 **Key=pqr,rst,uvw** から **pqr** を削除する場合、**pqr** の後に続くコンマも削除されます。コンマおよびセミコロン以外の文字は削除されません。基本的に、区切り文字の前後に空白を使わないようにしてください。

AddProfString の初期化 (.ini) ファイル エントリをアンインストールする

次の条件が満たされていると、アンインストールによってキー名と値のペアが完全に削除されます。

- **AddProfString** によってキーが正しく作成されている。
- アンインストール実行時に存在したキー名と値のペアがインストールされたキー名と値のペアと完全に一致する。インストールとアンインストールの間に、別のインストールまたはプログラムによってインストールされたキーが変更されると、キー名と値のペアは完全に削除されません。元のインストールによって作成された値だけが削除され、キー自体とその他の値はアンインストーラーによって削除されません。

元の System.ini ファイルが次のような場合 :

[386Enh]

device=votc.386

device=*vmpcd

AddProfString を呼び出した後は次の様になります :

[386Enh]

device=*vmp32

device=votc.386

device=*vmpcd

アンインストールによって device=*vmp32 が削除されます。

別のインストールが既存の行に値を追加した場合、アンインストール中のインストールからの値のみが削除されます。オリジナルキー名および新しい値はファイルに残ります。たとえば、[Test] セクション内の Test.ini に **Test Values=votc.386** 行を追加した場合は、以下のとおりです。

[Test]

Test Values=votc.386

Continuous Test=No

また、インストールの実行後に別のインストールが Test Values に新しい値を追加している場合は、以下のとおりです。

[Test]

Test Values=votc.386,pctcp.386

Continuous Test=No

アプリケーションがインストールされるとき、votc.386 が削除されて TestValues=pctcp.386 がファイル内に残ります。

[Test]

Test Values=pctcp.386

Continuous Test=No

ReplaceProfString の初期化 (.ini) ファイル エントリをアンインストールする

次の条件が満たされていると、アンインストールによってキー名と値のペアが完全に削除されます。

- **ReplaceProfString** によってキーが正しく作成されている。
- 以前、キーが存在していなかった。
- アンインストール実行時に存在したキー名と値のペアが、インストールされたキー名と値のペアと完全に一致する。インストールとアンインストールの間に、別のインストールまたはプログラムによってインストールされたキーが変更されると、キー名と値のペアは完全に削除されません。元のインストールによって作成された値だけが削除され、キー自体とその他の値はアンインストーラーによって削除されません。

元の System.ini ファイルがこのような場合 ;

[386Enh]

device=votc.386

device=*vmpcd

ReplaceProfString を呼び出した後は、以下のようになります。

[386Enh]

device=*vmp32

device=votc.386

device=*vmpcd

アンインストールによって device=*vmp32 が削除されます。

WriteProfString の初期化 (.ini) ファイル エントリをアンインストールする

次の条件が満たされていると、アンインストールによってキー名と値のペアが完全に削除されます。

- **WriteProfString** によってキーが正しく作成されている。
- 以前、キーが存在していなかった。
- アンインストール実行時に存在したキー名と値のペアが、インストールされたキー名と値のペアと完全に一致する。インストールとアンインストールの間に、別のインストールまたはプログラムによってインストールされたキーが変更されると、キー名と値のペアは完全に削除されません。元のインストールによって作成された値だけが削除され、キー自体とその他の値はアンインストーラーによって削除されません。

COM オブジェクトを使用してインストールを拡張する



重要 *・ COM オブジェクトと InstallShield オブジェクトは混在しやすいので注意してください。2 つのオブジェクトはまったく別の機能です。*

DLL または他の実行可能ファイルの代わりに、COM オブジェクトを使用してインストールを拡張することができます。COM オブジェクトは、比較的簡単にインストールに統合することができます。次の手順に従って COM オブジェクトをスクリプトに割り当てます。



タスク ***COM オブジェクトを割り当てるには、以下の手順に従います:***

1. オブジェクト変数を宣言します。例:

   ```
   OBJECT oMyCOMObject
   ```

2. COM への参照を取得して、SET キーワードを使用して、それを変数に割り当てます。例:

   ```
   set oMyCOMObject = CreateObject ( szMyProgID );
   ```

szMyProgID の値が、COM オブジェクトの有効なプログラム ID になります。(SET キーワードを省略すると、スクリプト エンジンは oMyCOMObject のデフォルト プロパティを szProgID に設定します。デフォルトとなるプロパティがないため、oMyCOMObject はCOM オブジェクトへの参照を含まないので、スクリプトは失敗します。)



重要 *・デフォルトでは、インストールが* **CoCreateObject**、**CoCreateObjectDotNet**、**CoGetObject**、*または* **DotNetCoCreateObject** *を使って COM オブジェクトをロードした後、インストールが終了するまでロードされた状態が保たれます。COM オブジェクトは、COM オブジェクト変数がスコープ外に出るか、NOTHING に設定されているかに関わらず、ロードされた状態に保たれます。*

そのため、COM オブジェクトが参照するライブラリもインストールが終了するまでリリースされません。ただし、Windows API の ***CoFreeLibrary*** *を呼び出して COM オブジェクトが参照するライブラリをアンロードすることは可能です。* ***CoFreeLibrary*** *は、ライブラリがオブジェクト変数によって参照されている状態かどうかに関わらず、これをアンロードします.したがって、ライブラリを参照するすべてのオブジェクトがスコープ外に出た後、または これらが NOTHING に設定されている場合に* ***CoFreeLibrary*** *を呼び出します。次のサンプル コードを参照してください。*

```
prototype void ole32.CoFreeLibrary( byval int );
int     hModule;

program

   //ライブラリを開放します。
   hModule = GetModuleHandle( "CSharpClassLibrary.dll" );

   if( !hModule ) then
      MessageBox( "モジュール ハンドルを取得できませんでした", INFORMATION );

   else

      CoFreeLibrary( hModule ); // ライブラリをリリースします

   endif;
```

endprogram

***CoFreeUnusedLibraries** を使ってライブラリをアンロードすることもできます。ただし、デフォルトではライブラリのアンロードを行う前にシステムが 10 分間待機するように設定されています。ほとんどの場合、インストールはライブラリをすぐにアンロードする必要があります。インストールが Windows XP 以降でのみ実行される場合、**CoFreeUnusedLibrariesEx** を呼び出すことができます。これにはライブラリをすぐにアンロードするオプションが含まれます。*

以下は、シリアル番号を検証する COM オブジェクトの例です。Validate というメソッドと、IsEval というプロパティがあります。

```
function OnFirstUIBefore()
    OBJECT oObj;
    STRING szProgID, szMsg, szTitle, svName, svCompany, svSerial;
    BOOL bValidSerialNumber, bEval, bValidate;
    NUMBER nResult;
begin
    szProgID = "MySerialValidation";
    set oObj = CreateObject( szProgID );
    if ( !IsObject( oObj ) ) then
        MessageBox( "オブジェクト " + szProgID + " は無効です。", SEVERE );
        return ISERR_GEN_FAILURE;
    endif;

    bValidSerialNumber = FALSE;
    while (!bValidSerialNumber)
        szMsg = "名前、会社名、および製品のシリアル番号を入力してください。";
        szTitle = "情報の入力";
        nResult = SdRegisterUserEx( szTitle, szMsg, svName, svCompany, svSerial );
        if ( nResult < 0 ) then
            MessageBox( "ダイアログ ボックスを表示できませんでした", INFORMATION );
            return ISERR_GEN_FAILURE;
        endif;

        /* オブジェクトの IsEval プロパティの値を確認します。*/
        bEval = oObj.IsEval;
        /* オブジェクトの Validate メソッドを実行して
        メソッドからの戻り値を取得します。*/
        bValidate = oObj.Validate( svSerial );
        if ( bEval || bValidate ) then
            bValidSerialNumber = TRUE;
        else
            MessageBox( "シリアル番号が無効です。もう一度入力してください。", INFORMATION );
        endif;
    endwhile;

    return ISERR_SUCCESS;
end;
```

***メモ:** COM オブジェクトが期待するメンバーがあれば、データ構造を COM オブジェクトに渡すことができます。次は、データ構造を渡す例です。*

```
#define PERSON_NAME_SIZE 1024
```

```
typedef PERSON
    begin
        STRING Name[PERSON_NAME_SIZE];
        NUMBER Age;
    end;

function OnBegin()
    PERSON pPerson;
    NUMBER nPhoneNumber;
    OBJECT oMyCOMObject;
    STRING szMyProgID;
begin
    /* この行で szMyProgID に値を割り当てます。*/
    set oMyCOMObject = CreateObject ( szMyProgID );
    if ( !IsObject( oMyCOMObject ) ) then
        MessageBox( "オブジェクト " + szMyProgID + " は無効です。", SEVERE );
        return ISERR_GEN_FAILURE;
    endif;
    nPhoneNumber = oMyCOMObject.GetPhoneNumber( pPerson );

    return ISERR_SUCCESS;
end;
```



メモ: *標準の COM 戻り値 HRESULT は、InstallScript などのオートメーション クライアントには表示されません。COM オブジェクト メソッドから値を戻すには、次のサンプル コードで説明されているように、メソッドのパラメーター リストに [out,retval] パラメーターが必要です。*

IDL:

```
interface IMyInterface : IDispatch
{
    [id(1)] HRESULT DoSomeWork([in] long nInputValue, [out,retval] long *pReturnValue);
}
```

C++:

```
STDMETHODIMP MyInterfaceImpl::DoSomeWork(long nInputValue, long *pReturnValue)
{
    // 実装コード
    *pReturnValue = 1; //クライアントに値を戻します
    return S_OK;
}
```

InstallScript:

```
function OnBegin()
    NUMBER nObjReturn;
begin
    set oMyCOMObject = CreateObject("MyProgID");
    if ( !IsObject( oMyCOMObject ) ) then
        nObjReturn = oMyCOMObject.DoSomething(10);
        /*nObjReturn には、実装から戻された値 1 が含まれています*/
    endif;
end;
```

関数呼び出しで機能やサブ機能を指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

*機能*とは、InstallShield で一連のコンポーネントまたはサブ機能を参照する一般的な名称です。*サブ機能*とは別の機能の下にある機能で、フォルダーとサブフォルダーの関係に似ています。

トップレベルの機能とは、階層の最も上にある機能です。トップレベルの機能がサブ機能と呼ばれることはありません。

InstallScript コード内の機能およびサブ機能の参照方法

一部の機能関数およびセットアップの種類 ダイアログ関数では、単一の機能を参照し、他では複数の機能を参照します。

単一機能の参照

1 つの機能を参照するには、機能名を使用します。サブ機能を参照するには、パスと同様に階層順に各機能を 2 つのバックスラッシュで区切って記述します。たとえば、トップレベルの機能 **Help Files** 内にあるサブ機能 **Tutorials** を指定するには、インストール スクリプトで以下の式を使用します。

```
szFeature = "Help Files¥¥Tutorials";
```

Tutorials 内にあるサブ機能 **CBT** を参照するには、以下を使用します。

```
szFeature = "Help Files¥¥Tutorials¥¥CBT";
```

機能の名前に円記号は使用できません。

複数機能の参照

InstallScript MSI インストールでは、**SdFeatureMult** などの機能およびセットアップの種類ダイアログ関数で、複数の機能およびそれらのサブ機能が表示されることもあります。その場合、階層のすぐ上の機能を指定して、複数の機能を参照します。複数のトップレベルの機能を参照するには、ヌル文字列 (“”) を使用します。

たとえば、**SdFeatureMult** 関数にヌル文字列を渡すと、MEDIA システム変数の値により、**SdFeatureMult** ダイアログの左のウィンドウに設定されているスクリプト作成コンポーネントのトップレベル機能がすべて、対応するダイアログに表示されます。すべてのサブ機能は、このダイアログの右側のウィンドウに表示されます。

Windows API 関数を呼び出す



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

Windows API 関数は、InstallScript コード内から呼び出すことができます。



タスク ***Windows API 関数を呼び出すには、以下の手順を実行します。***

1. スクリプトの先頭で、次の構文を使用して関数のプロトタイプを宣言します。

   prototype ReturnType DLLName.FunctionName(ParamType1, ParamType2, ...)

   例:

   prototype INT User.LoadString(INT, SHORT, BYREF WSTRING, INT);

   Dumpbin.exe を /EXPORTS オプションと共に使用して、Windows API DLL Kernel32.dll、User32.dll、および GDI32.dll のいずれかにある関数を判別します。関数の戻り値の型およびパラメーターのタイプを判別するには、MSDL (Microsoft Development Library) などの Windows プログラミング リファレンスを参照してください。

   メモ・ *戻り値の型を、InstallScript データ型 BOOL、CHAR、HWND、INT、LONG、 LPSTR、NUMBER、POINTER、または SHORT に指定することができます。戻り値の型が指定されなかった場合、InstallShield は DLL 関数が 4 ビットの値を戻したと仮定します。*

2. 他の関数と同様に、関数を呼び出します。例:

   nResult = LoadString(iInstance, nStringID, szMyString, MAX_SIZE);

3. 関数が失敗した際、より詳しいエラー情報が必要な場合、Err オブジェクトの LastDllError プロパティを確認してください。(セットアップは制御をスクリプトに戻す前にこの値をそれ自身で変更するため、Windows API 関数 **GetLastError** を呼び出して詳しいエラー情報を取得することはできません。)

カスタム ファイルの転送操作を埋め込む

InstallShield では、インストールの標準ファイル転送中に追加のファイル転送操作を埋め込むことができます。進行状況バーも、これらの操作中にスムースかつ適切に更新されます。この機能の使用に関して、いくつかの異なる一般的なシナリオがあります。

以下は、各シナリオで呼ばれる機能の概要を説明したテーブルです。

テーブル 3・共通カスタム ファイルの転送操作に適切な関数を判別する

シナリオ	呼び出す関数
FeatureMoveData	なし
XCopyFile または CopyFile	FeatureAddCost
LaunchApplication(外部アプリケーション)	FeatureAddCost、FeatureSpendCost、FeatureAddUninstallCost、または FeatureSpendUninstallCost

これらの手続きには、各シナリオのカスタムファイル転送操作の利用に関する手順を追っての説明が含まれています。



タスク ***標準ファイル転送操作中に、XCopyFile または CopyFile を使用して追加のファイルをインストールする場合、以下の手順を実行します。***

1. ファイル転送前に、ファイルの転送先フォルダーのクラスタサイズを考慮して、インストールされるファイルのサイズを判別します。ファイルのサイズは、**GetAndAddFileCost**、**CalculateAndAddFileCost**、および/または **GetAndAddAllFilesCost** 関数を使用することで判別できます。
2. **FeatureAddCost** 関数を利用して、手順 1 で判別したコストをこれらのファイルが関連付けられている機能のコストへ加えます。すべてのコストは特定の機能に関連付けられています。したがって、インストールされるファイルが既存の機能に関連付けられていない場合、サンプル機能を作成するか、既存の機能を使用する必要があります。
3. ファイル転送中、**XCopyFile** 関数を呼び出して、ファイルをインストールします。通常、これは追加のファイルに関連付けられている機能のイベントのインストール中、または OnMoving または OnMoved イベントで実行します。進行状況バーも、呼び出し中にスムースかつ適切に更新されます。XCopyFile 操作中、ステータス ダイアログの [キャンセル] ボタンを無効にする必要があります。詳細については、XCopyFile 関数の説明を参照してください。



メモ: *アンインストールに必要な追加のアクションはありません。ファイルは、アンインストール進行状況が適切に更新されている間にアンインストールされます。*



タスク ***標準ファイル転送操作中に、追加のファイルをインストールする外部インストールを呼び出す場合、以下の手順を実行します。***

1. ファイル転送前に、外部インストールを検査し、発生する操作を調べて、外部インストールによってインストールされるファイルのサイズおよび数 (または他の操作) を判別します。
2. **FeatureAddCost** 関数を利用して、手順 1 で判別したコストをこのインストールが関連付けられている機能のコストへ加えます。すべてのコストは特定の機能に関連付けられています。したがって、インストールされるファイルが既存の機能に関連付けられていない場合、サンプル機能を作成するか、既存の機能を使用する必要があります。以下の 2 つのシナリオを確認してください。
3. 外部インストールがアンインストール中に呼び出された場合、**FeatureAddUninstallCost** を利用して操作の数および操作のサイズをアンインストールのコストに加えます。
4. ファイル転送中、**LaunchApplication** イベントを使用してインストールを起動 (通常、サイレントモード) します。通常、これは追加のファイルに関連付けられている機能のイベントのインストール中、または OnMoving または OnMoved イベントで実行します。スクリプトが、インストールおよびアンインストール中に、定期的に制御を再取得するように LAAW_USE_CALLBACK オプションを利用します。
5. **OnLaunchAppAndWaitCallback** イベントをオーバーライドし、インストールの実行をポールするコードを追加して進行状況を判別します。それから、**FeatureSpendCost** (インストール) 関数または **FeatureSpendUninstallCost** (アンインストール) 関数を呼び出して、インストールが完了した適切なコストを使用します。この処理を、外部インストールが実行されている間繰り返します。進行状況バーも、呼び出し中にスムースかつ適切に更新されます。**OnLaunchAppAndWaitCallback** イベントが適切な回数呼ばれて進行状況バーがスムースに更新されるように、LAAW_PARAMETERS.nCallbackInterval を調整しなければならない場合がありますので注意してください。

InstallScript で大きい数値を表現する

ある数値が InstallScript 整数で格納される際、31 ビットデータだけが値の格納に使用することができます。したがって、表現可能な最大値は 2^31 (2GB) となります。InstallScript 整数は、常に符号付きと見なされるため、32 ビット目は符号ビットとして解釈され、数値データの格納に使用することはできません。この結果、符号なしデータ型で格納が可能な 2^32 (4GB) の代わりに 2^31 (2GB) に表現することができる値が制限されます。

場合においては、より大きい値を指定できることが必要もしくは望ましいことがあります。これを InstallShield で実現するため、いくつかの関数では、大きい数値の指定を各数値の符号ビット (32 ビット) を持つ 1 組の 32 ビットの数値として指定できます。符号ビットは常に正の数値を示すように 0 が設定されます。このフォーマットを使用することにより、数値を 62 ビットまでのデータまたは 2^62 までのサイズと共に指定または取得することができます。

ただし、この言語は数値を単一変数として表現できるデータ型を持たないため、このデータに実行可能な操作は限られています。このデータは通常、これらの値をサポートする関数の間でやり取りされます。データはまた、**ConvertSizeToUnits** 関数を利用してより大きいサイズ単位 (KB または MB など) で表現された単一値に変換することも可能ですし、より大きいデータ型を処理できる外部 DLL 関数に渡すことも可能です。

ある数値が上位/下位の 1 組で指定された場合、下位 32 ビットの値は、完全値の下位 31 ビットデータを指定します (符号ビットは常に 0 に設定され無視されます) (符号ビットは常に 0 に設定され無視されます)。高位 32 ビット値は、高位 31 ビット データ (同様に符号ビットは常に 0 に設定され無視されます) を指定します。たとえば、4 の上位値は実際、4 * 2^31 または 8589934592 (8 GB) サイズを示します。これが、下位値 100 と組み合わさって、合計サイズ 8589934692 単位を示します。

数値が 2^31 未満の場合、高値は必ず 0 となり、値のサイズが 2^31 を超えないと明らかな場合は、これを無視することができます。ただし、このデータを計算に使用する必要がある場合、**ConvertSizeToUnits** 関数を呼び出して、このデータを単一値で表現するとき四捨五入したときの誤差が最小になるように最も適した単位の単一値に変換することを推奨します。



メモ: *下位値 (low-value) が下位 32 ビットデータを格納し、上位値 (high value) がデータの上位 32 ビットを格納する Windows によって使用される符号なし上位/下位値の組み合わせとは規則が異なりますので注意してください。* **ConvertWinHighLowSizeToISHighLowSize** *関数は、Windows API 関数によって戻されるデータの簡易変化用に提供されています。*

次の関数は、上位/下位値の組み合わせをサポートしています。

- GetDiskInfo
- GetFileInfo
- GetAndAddFileCost
- CalculateAndAddFileCost
- GetAndAddAllFilesCost
- FeatureFileInfo
- FeatureGetCostEx
- FeatureAddCost
- FeatureSpendCost
- ConvertWinHighLowSizeToISHighLowSize

- ConvertSizeToUnits

デバイス ドライバーのインストール

InstallShield では、DIFxAPI コンポーネントを含む Microsoft Windows Driver Install Framework (DIFx) を使って InstallScript インストール内のデバイス ドライバーをインストールことがサポートされています。このコンポーネントがあると、Windows Installer を使わずにドライバーをインストールまたはアンインストールすることができます。



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。デバイス ドライバーのインストールは、DIFx Windows Installer 機能を使用した InstallScript MSI および基本の MSI プロジェクトでサポートされています。これらのプロパティの使用方法については、「デバイスドライバー設定を構成する」を参照してください。*



タスク ***InstallScript プロジェクトに 32 ビット DIFx ドライバーをインストールするには、以下の手順を実行します。***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"DIFx (32 ビット プラットフォーム用)"** 設定で **[有効]** を選択します。
3. DIFx ドライバー ファイルをプロジェクトに追加します。
4. DIFx ドライバー ファイルを含む機能の機能イベントをオーバーライドし、プラグ アンド プレイ (PnP) ではないドライバー用の **DIFxDriverPackageInstall** 関数、または、PnP 用の **DIFxDriverPackagePreinstall** 関数を呼び出します。
5. インストールのビルドと実行を行います。



タスク ***InstallScript プロジェクトに 64 ビット DIFx ドライバーをインストールするには、以下の手順を実行します。***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"DIFx サポート (64 ビット Itanium プラットフォーム用)"** 設定または **"DIFx サポート (64 ビット AMD プラットフォーム用)"** 設定で **[有効]** を選択します。
3. DIFx ドライバー ファイルをプロジェクトに追加します。
4. DIFx ドライバー ファイルを含む機能の機能イベントをオーバーライドし、プラグ アンド プレイ (PnP) ではないドライバー用の **DIFxDriverPackageInstall** 関数、または、PnP 用の **DIFxDriverPackagePreinstall** 関数を呼び出します。
5. インストールのビルドと実行を行います。

コンパイラ バージョンの確認

InstallShield では、_ISCRIPT_VER は 0x*VVM*、製品のメジャー バージョンは *VV*、メンテナンス パック リリース番号は *M* と定義付けられています。たとえば、0x900 は InstallShield DevStudio バージョン 9.0 のメンテナンスパック 0 (最初のリリース) と評価します。このプリプロセッサ定数は、InstallShield Professional 7 では 0x700、InstallShield Professional 6 では 0x600 と定義されていますが、InstallShield Professional 5.x では定義されておらず、ゼロと評価されます。

この定数は、スクリプトに以下のようなコードを含むことによって、InstallShield コンパイラが使用されているかどうかをテストするのに利用することができます。このコードは InstallShield DevStudio (バージョン 9.0、メンテナンスパック番号 0) またはそれ以降の最初のリリースが存在するかどうかをテストします。

```
#if _ISCRIPT_VER >= 0x900
    MessageBox("DevStudio 9.x とそれ以降",INFORMATION );
#elif _ISCRIPT_VER >= 0x700
    MessageBox ( "これは IS 7.x コンパイラです。", INFORMATION );
#else
    MessageBox ( "これは IS 6.x 以前のコンパイラです。", INFORMATION );
#endif
```

下位互換性のために保持されているプリプロセッサ定数

プロプロセッサ定数 _ISCRIPT_ISPRO と _ISCRIPT_ISDEV は下位互換性のために保持されています。_ISCRIPT_ISPRO は InstallScript プロジェクト (および InstallShield Professional プロジェクト) 用に定義されています。_ISCRIPT_ISDEV は InstallScript MSI プロジェクトと基本の MSI プロジェクト用に定義されています。

_ISCRIPT_ISDEV と _ISCRIPT_ISPRO を使用してオーサリング環境を確認する



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

プリプロセッサ定数 _ISCRIPT_ISPRO は InstallScript プロジェクト (そして InstallShield Professional プロジェクト) では定義されていますが、InstallScript MSI プロジェクトや基本の MSI プロジェクトでは未定義となり、ゼロ評価します。プリプロセッサ定数 _ISCRIPT_ISDEV は InstallScript MSI や基本の MSI プロジェクトでは定義されていますが、InstallScript プロジェクト (そして InstallShield Professional プロジェクト) では未定義となり、ゼロ評価します。

_ISCRIPT_ISDEV と _ISCRIPT_ISPRO を使って、異なるプロジェクトの種類で別々の動作を行う単一のスクリプトを書くことができます。そのためには次のようなコードをスクリプトに含みます。

```
#ifdef _ISCRIPT_ISPRO
    //  InstallShield Professional と InstallScript プロジェクトに特有のコード
#else
    #ifdef _ISCRIPT_ISDEV
        // 基本の MSI プロジェクトと InstallScript MSI プロジェクトに特有のコード
    #endif
#endif
```

別の InstallScript インストールからインストールを起動する



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

LaunchApplication 関数を呼び出して完全インストールを起動することができます。たとえば、子インストールのファイルを CD-ROM に配置した場合、次のように **LaunchApplication** を呼び出すことができます。

LaunchApplication (SRCDISK ^ "Launched Setup Folder¥¥Setup.exe", "", "", SW_HIDE,"", LAAW_OPTION_WAIT);

その他、プロジェクト内の子インストールをサポート ファイルとして含めることもできます。この方法を使うと、メイン インストールが子インストールをターゲット システムにコピーし、インストールを実行し、またその他のサポート ファイルと一緒にインストールを削除します。子インストールのパスで SUPPORTDIR 変数を使用します:

LaunchApplication (SUPPORTDIR ^ Setup.exe, "", "", SW_HIDE,"", LAAW_OPTION_WAIT);

また、子インストールへのリンクをファイル グループに挿入して、子インストールをターゲット システムにインストールしてから、ターゲット先からインストールを起動することもできます。(これらの方法を使った場合、親インストールがアンインストールされるまで子インストールがターゲット システムに残ります。)

InstallScript の言語サポート

次のテーブルは、InstallShield の Premier edition によってサポートされている言語です。以下は、各列の説明です。

- **InstallShield 言語** – この言語を参照するために InstallShield インターフェイスで使用される名前。
- **InstallScript 定数** – 言語固有のコンポーネントをフィルターするために InstallShield で提供されている言語定数。
- **Windows 日本語版での名前** – Windows (日本語版) でこの言語に使用される名前。

テーブル 4 ・ サポートされている言語

InstallShield 言語	InstallScript 定数	日本語版での名前
バスク語	ISLANG_BASQUE	バスク語
ブルガリア語	ISLANG_BULGARIAN	ブルガリア語
カタロニア語	ISLANG_CATALAN	カタロニア語
中国語 (簡体字)	ISLANG_CHINESE_SIMPLIFIED	中国語 (簡体字)
中国語 (繁体)	ISLANG_CHINESE_TRADITIONAL	中国語 (繁体)
クロアチア語	ISLANG_CROATIAN 下位互換性の目的で、この定数は (セルビア語との関連性から) より論理に適った 0x041a ではなく 0x001a のままです。ISLANG_CROATIAN_STANDARD ではなくこの定数を利用し続けてください。	クロアチア語
チェコ語	ISLANG_CZECH	チェコ語
デンマーク語	ISLANG_DANISH	デンマーク語
オランダ語	ISLANG_DUTCH	オランダ語 (標準)

テーブル 4 • サポートされている言語 (続き.)

InstallShield 言語	InstallScript 定数	日本語版での名前
英語	ISLANG_ENGLISH_UNITEDSTATES	英語 (U.S.)
フィンランド語	ISLANG_FINNISH	フィンランド語
フランス語(カナダ)	ISLANG_FRENCH_CANADIAN	フランス語(カナダ)
フランス語 (標準)	ISLANG_FRENCH_STANDARD	フランス語 (標準)
ドイツ語	ISLANG_GERMAN	ドイツ語 (標準)
ギリシャ語	ISLANG_GREEK	ギリシャ語
ハンガリー語	ISLANG_HUNGARIAN	ハンガリー語
インドネシア語	ISLANG_INDONESIAN	インドネシア語
イタリア語	ISLANG_ITALIAN	イタリア語 (標準)
日本語	ISLANG_JAPANESE	日本語
韓国語	ISLANG_KOREAN	韓国語
ノルウェー語	ISLANG_NORWEGIAN	ノルウェー語 (ブークモール)
ポーランド語	ISLANG_POLISH	ポーランド語
ポルトガル語(ブラジル)	ISLANG_PORTUGUESE_ BRAZILIAN	ポルトガル語(ブラジル)
ポルトガル語(標準)	ISLANG_PORTUGUESE_STANDARD	ポルトガル語(標準)
ルーマニア語	ISLANG_ROMANIAN	ルーマニア語
ロシア語	ISLANG_RUSSIAN	ロシア語
セルビア語	ISLANG_SERBIAN_CYRILLIC	セルビア語 (キリル)
スロバキア語	ISLANG_SLOVAK	スロバキア語
スロヴェニア語	ISLANG_SLOVENIAN	スロベニア語
スペイン語	ISLANG_SPANISH	スペイン語 (トラディショナル　ソート)
スウェーデン語	ISLANG_SWEDISH	スウェーデン語
タイ語	ISLANG_THAI	タイ語

テーブル 4 - サポートされている言語 (続き.)

InstallShield 言語	InstallScript 定数	日本語版での名前
トルコ語	ISLANG_TURKISH	トルコ語

色化けの防止

InstallScript カスタム アクションで色化けの問題が生じている場合は、カラーパレットの転換によるものがほとんどです。このセクションでは、Windows システムに色が割り当てられる方法、およびインストールのエンドユーザー インターフェイスのひずみを最小限にするためのトラブルシューティングのヒントについて説明します。

ビデオドライバーによってサポートされている色の数にかかわらず、256 色モードで動作するシステムでは 256 色のカラーパレットで、16 色モードで動作するシステムでは 16 色のカラーパレットで、可能な色がすべて表示されます。High Color モードまたは True Color モードで動作するシステムでは、カラーパレットは使用されません。色は直接表示されるので、これらのシステムで色化けが起きる心配はありません。カラーパレットを使用するシステムでは、ビットマップなどのオブジェクトが表示される場合など必要に応じてエントリが割り当てられます。

現在のカラーパレットが割り当てられた後では、新しい色を表示するために Windows は既に割り当てられた色と類似した色を使おうとします。そのため、256 色システムでは、複数の異なる色からなる複数のオブジェクトが同時に表示されると色化けが生じる場合があります。

また、Windows はスタートアップ時に 16 の静的カラー (VGA カラー) に 4 つのグレーのシェードを加えて割り当てます。システムはこれらの色を使用して 16 色イメージを表示し、これらの色の割り当てが Windows によって解除されることはありません。カラーパレット操作の詳細については、Windows プラットフォーム SDK を参照してください。

以下のヒントを利用すると、カラーパレット転換に関する色化けを最小限に抑えることができます。カラーパレットのヒントは、インストール時に表示する背景色、.avi、メタファイル、ビットマップなどのすべてのイメージに適用できます。

メインインストールウィンドウの背景

- 1 つのパレットエントリしか必要としない単一色の背景を使用してみます。単一色背景定数を使用して **SetColor** 関数を呼び出し、背景色を作成します。
- 16 のパレットエントリのみを必要とする 16 色のグラデーション背景を使用してみます。**SetColor** 関数をグラデーション色の定数の 1 つと共に呼び出し、このタイプの背景を作成します。
- ビットマップ背景を並べて表示、または中央に表示して使用してみます。**PlaceBitmap** 関数を呼び出して、これらの背景を 1 つ作成します。
- 約 80 のパレットエントリを必要とする 256 色のグラデーションの背景は使用しないでください。

ビットマップとメタファイル

- **PlaceBitmap** 関数を REMOVE オプションと共に呼び出してビットマップを削除すると、そのビットマップが使用していたカラー パレット エントリのうち、現在表示されている他のイメージが使用していないすべてのエントリが、別のオブジェクトを表示するために他の色が必要になった時に、カラーパレットから解放されます。色化けが生じている場合は、ビットマップが表示されなくなったら、(別のビットマップで隠すのではなく) そのビットマップを必ず削除し、カラーパレットエントリが解放されるようにしてください。

- インストール中に表示されるビットマップやビルボードに対して、類似色（常に割り当てられている 16 の静的カラーを含む）を使用して、必要なカラーパレットエントリの数を減らします。
- COLORS オプションを使用して **GetSystemInfo** 関数を呼び出し、ターゲット システムで使用できる最大の色数を決めることができます。戻り値に基づいて、異なる解像度のビットマップを表示することができます。
- 24 ビットのビットマップには、ビットマップファイルのカスタムカラーパレットが含まれません。256 色のシステムで 24 ビットのビットマップを表示している場合、使用できる追加のカラーパレットエントリがある場合でも、現在使用可能なパレットエントリだけが使用されます。インストールで 24 ビットのビットマップを表示し、256 色のシステムで実行する場合、256 色バージョンのビットマップも含めることをお勧めします。
- メタファイルは配置されるのではなく作成されるため、メタファイルが作成される際に、追加のカラーパレットエントリは割り当てられません。つまり、メタファイルの表示には、既存のカラーパレットエントリだけが使用されます。インストールを 256 色のシステムで実行する場合、標準の 16 の静的カラーは現在のカラーパレットのエントリにかかわらず使用できるため、これらの色だけをメタファイルで使用することをお勧めします。
- システムのビデオ ドライバーが 256 色以上の表示をサポートしていることを確認します。ビデオカードとモニターが 256 色表示をサポートしている場合でも、ビデオドライバーが 256 色表示をサポートしていない場合は、ビットマップは適切に表示されません。
- モニターで 256 色のビットマップが表示されない場合、16 の静的カラーだけが使用されます。このため、大幅な色化けが生じる可能性があります。インストールで 256 色のイメージを使用している場合は、256 色をサポートしているシステムでセットアップを実行するようエンドユーザーに要求することをお勧めします。

カスタム アクションを使用



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShieldでは、カスタム アクションを使った InstallScript、VBScript、または JavaScript コードの実行、DLL 関数の呼び出し、実行可能ファイルの実行、マネージ アセンブリ内にあるマネージ メソッドの呼び出し、プロパティやディレクトリの設定、エラーのトリガとインストールの中止、PowerShell スクリプトの実行、プロセスの終了、他のインストール パッケージの実行がサポートされています。



タスク

InstallShield におけるカスタム アクションの作成および実行は、以下のステップで進めます。

1. [カスタム アクションとシーケンス] ビュー（または、[カスタム アクション] ビュー）、またはカスタム アクション ウィザードを使用して、カスタム アクションを作成します。
2. カスタム アクションを実行するタイミングを決めます。

3. カスタム アクションを、シーケンスに挿入するかダイアログのコントロール イベントの結果として配置して実行します(基本の MSI プロジェクト)。

様々な機能をサポートするために InstallShield プロジェクトに自動的に追加される各 InstallShield カスタム アクションについての詳細は、「InstallShield カスタム アクション リファレンス」を参照してください。



プロジェクト: *マージ モジュール プロジェクトのみ、ダイレクト エディターで* ModuleInstallExecuteSequence *テーブルを変更して、マージ モジュール内のカスタム アクションの起動を制御することができます。インストール プロジェクトにマージ モジュールを追加する際、マージ モジュールの中に含められたすべてのカスタム アクションおよびダイアログは [カスタム アクションとシーケンス] ビューを通してインストールのシーケンスに挿入することができます。*

カスタム アクションウィザードを使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

カスタム アクションウィザードを使用すると、カスタム アクションを作成したり既存のカスタム アクションを変更できます。



タスク ***カスタム アクションウィザードを起動するには、以下の操作を行います。***

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、**[カスタム アクション ウィザード]** をクリックします。カスタム アクション ウィザードが起動します。
3. ウィザードパネルに従い、新しいカスタム アクションを作成します。

また、カスタム アクション ウィザードを使用せずに [カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でカスタム アクションを作成することもできます。詳細については、「[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でカスタム アクションを作成する」を参照してください。

進行状況ダイアログとインストールのログ ファイルにアクション情報を表示する方法については、「アクション テキストを使う」を参照してください。

[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でカスタム アクションを作成する



プロジェクト: *[カスタム アクションとシーケンス] ビューは、次のプロジェクトの種類で使用できます。*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

このビューは、次のプロジェクトの種類では、[カスタム アクション] ビューという名前で呼ばれます。

- *DIM*
- *マージ モジュール*
- *MSM データベース*

カスタム アクションの作成は、カスタム アクション ウィザードの使用をお勧めします。ウィザードを使うと、必要な手順に従ってカスタム アクションを作成し、情報を入力できます。また、このウィザードを使用する代わりに、[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) で設定を直接構成して、カスタム アクションを作成することもできます。



タスク

カスタム アクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、追加するカスタム アクションの種類をクリックします。
3. 新しいアクションの名前を変更するには、アクションを選択し、F2 キーを押して新しい名前を入力します。
4. カスタム アクションの設定を構成します。

進行状況ダイアログとインストールのログ ファイルにアクション情報を表示する方法については、「アクション テキストを使う」を参照してください。

カスタム アクションのクローン



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

- *トランスフォーム*

InstallShieldには、カスタム アクションをコピーまたは*クローン*する機能があります。クローンは、オリジナルカスタム アクションとすべて同じプロパティおよび値持つ、同じ種類の新しいカスタム アクションを作成します。クローンを利用して、各カスタム アクション属性を手動で設定する必要なく、類似する属性を持つ複数のカスタム アクションを作成することができます。

クローン処理は、カスタム アクションのクローンのみが行われます。[インストール] シーケンスへカスタム アクションを挿入することはありません。



タスク ***カスタム アクションをクローンするには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
2. **[カスタム アクション]** エクスプローラーで、クローンするカスタム アクションを右クリックして、**[クローン]** をクリックします。

クローンされたカスタム アクションのコピーが、[カスタム アクション] エクスプローラーにコピーされます。カスタム アクションの名前は、クローンされたカスタム アクションと同じですが、コピーは、名前の終わりに 1 が付加されています。

カスタム アクションを別のプロジェクトにエクスポートする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*



タスク ***カスタム アクションを別のプロジェクトへ保存するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI の場合) または **[カスタム アクション]** (DIM、マージ モジュール プロジェクトの場合) をクリックします。
2. **[カスタム アクション]** エクスプローラーで、カスタム アクションを右クリックして、**[エクスポート]** をクリックします。**[エクスポート情報]** ダイアログ ボックスが開きます。
3. 既存の InstallShield プロジェクト ファイル (.ism ファイル) の場所を参照します。保存場所は、インストール プロジェクトまたはマージ モジュール プロジェクトのどちらかを使用できますが、ファイルは InstallShield の別のインスタンスでは開きません。
4. **[開く]** をクリックします。

このカスタム アクションのコピーが特定の .ism ファイルに追加されます。対象の .ism ファイルが既に異なるプロパティで同じ名前のカスタム アクションを持つ場合は、[競合の解決] ダイアログ ボックスが表示され、競合を解決するオプションを選択できます。

カスタム アクションの設定を構成する



プロジェクト: *[カスタム アクションとシーケンス] ビューは、次のプロジェクトの種類で使用できます。*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

このビューは、次のプロジェクトの種類では、[カスタム アクション] ビューという名前で呼ばれます。

- *DIM*
- *マージ モジュール*
- *MSM データベース*

カスタム アクションを作成後、その設定を構成する必要があります。また、その設定を後で変更しなくてはならない場合もあります。たとえば、カスタム アクションに割り当てたられた条件を編集する場合があります。

作成するカスタム アクションの種類によって、そのアクションの設定が異なる意味を持つ場合があります。たとえば、アクションが .msi パッケージ内に格納されている実行可能ファイルを呼び出す場合、呼び出す実行可能ファイルをポイントするテーブル内のエントリの名前が "ソース" 設定に存在する必要があります。"ターゲット" 設定は、この場合コマンドライン引数になります。



タスク

カスタム アクションの設定を構成するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
2. [カスタム アクション] エクスプローラーで、その設定を構成するカスタム アクションをクリックします。
3. 右側のペインで、適切に設定を構成します。

各設定については、「カスタム アクションの設定」を参照してください。

進行状況ダイアログとインストールのログ ファイルにアクション情報を表示する方法については、「アクション テキストを使う」を参照してください。

Summary List of All Custom Action Types (Windows Installer ヘルプ ライブラリ)

カスタム アクションの種類についての概要



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

カスタム アクションの "MSI タイプ番号" 設定に表示される値は、カスタム アクション (標準 DLL、実行可能ファイル、または InstallScript など)、場所 (Binary テーブルに格納、製品と共にインストールなど)、および、いくつかのカスタム アクションの設定 ("戻り値の処理"、"スクリプト内処理"、"実行スケジュール" など) が組み合わされたものです。これは、Windows Installer CustomAction テーブルの Type 列で利用できるいくつかのビットを組み合わせるために OR 演算子を使って計算された十進数値です。また、InstallShield では、InstallScript カスタム アクションおよび標準 DLL ファイル関数などの拡張カスタム アクション タイプもサポートされています。

Windows Installer ヘルプライブラリでは、カスタム アクションは数値タイプで参照されています。[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でプロジェクトにカスタム アクションを追加すると、カスタム アクション タイプ番号は自動的に計算され、"MSI タイプ番号" 設定にこの値が設定されます。たとえば、完全修飾パスが Property テーブルに格納されている実行可能ファイルの起動は、カスタム アクション タイプ 50 として計算されます。基本のカスタム アクション タイプに使われる値ついての詳細は、Windows Installer ヘルプ ライブラリの「Summary List of All Custom Action Types」を参照してください。

InstallShield で使用できるカスタム アクションの各種類についての詳細は、「カスタム アクションの種類」を参照してください。

Custom Action Type 50 (Windows Installer ヘルプ ライブラリ)

カスタム アクションの戻り値



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

カスタム アクション ウィザードの [追加のオプション] パネルでの選択に応じて、Windows Installer はインストールを継続する前に正常な戻り値を待機します。カスタム アクションは、その状況に応じて以下の値のうちの 1 つを戻さなければなりません。

テーブル 5 ・ カスタム アクションの戻り値

戻り値	説明
ERROR_FUNCTION_NOT_CALLED	カスタム アクションは実行されませんでした。
ERROR_INSTALL_FAILURE	カスタム アクションが失敗しました。
ERROR_INSTALL_SUSPEND	カスタム アクションは保留されていますが、後で再実行されます。
ERROR_INSTALL_USEREXIT	エンドユーザーが実行を中止したため、カスタム アクションは正常に完了しませんでした。
ERROR_NO_MORE_ITEMS	カスタム アクションは正常に完了しましたが、残りのアクションをスキップする必要があります。
ERROR_SUCCESS	カスタム アクションは正常に終了しました。

InstallScript カスタム アクションを書いている場合、エントリポイント関数の戻り値として ERROR_SUCCESS または ERROR_INSTALL_FAILURE のいずれかを戻す可能性があります。これらの定数は、スクリプト中に Iswi.h を含めると自動的に定義されます。InstallScript エンジンはその他のすべての例外を Windows Installer に通知します。

スクリプト内実行



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

カスタム アクション作成時に構成されている必要がある設定は、カスタム アクションウィザードの [応答オプション] パネル にある "スクリプト内実行" 設定です。アクションは、シーケンスに表示される順に実行されます。ただし、シーケンスは通常のインストール中に何度も実行されます。"スクリプト内実行" 設定を使うと、アクションをトリガーするシーケンス反復を選択することができます。

(ターミナル サーバーで使用可能)

[ターミナル サーバーで使用可能] オプション (たとえば、即時実行 (ターミナル サーバーで使用可能) を選択した場合、ターミナルサーバー マシン上でマシンごとのインストールが実行されいる間、カスタム アクションはエンドユーザーに偽装します。

即時実行

名前のとおり、内部 Windows Installer インストール スクリプトがコンパイルされるとただちに起動します。.msi ファイルが起動すると、Windows Installer サービスはインストールデータベースのすべてのテーブルを内部スクリプトに変換します。(この内部スクリプトは、InstallScript コードとは違います。)このスクリプトは、インストールのすべてのアクションを表示順に循環することによってビルドされます。このスクリプトのビルドはすぐに実行されます。"スクリプト内実行" 設定を [すぐに実行] に設定しているアクションが実行されると、そのアクションが起動します。したがってこのアクションはファイル転送が起きる前に起動します。インストールのエンドユーザー インターフェイスが完全にロードする前に起動する可能性もあります。

一般に、「即時実行」にスケジューリングされているカスタム アクションはターゲット システムを変更せず、プロパティを設定してターゲット システムを照会 (たとえば、ターゲット システムが製品のに合っているか確認するなど) します。Windows Installer のプロパティを設定するカスタム アクションと、[ユーザー インターフェイス] シーケンスで発生するカスタム アクションはすぐに実行されるようにスケジューリングする必要があります。

このタイプのアクションはシステム変更が行われる前に起動するため、インストールにインストールされているファイルを使用できません。

遅延実行

[遅延実行] は、Windows Installer サービスが、[即時実行] の実行中に作成された内部スクリプトを使用する場合に実行されます。このスクリプトが完全に生成されると、Windows Installer サービスは新しくコンパイルされたスクリプトを実行します。スクリプトはシーケンス内のすべてのアクションを順番に起動します。ただし、アクションをすぐに実行するようスケジューリングした場合は、そのアクションは遅延実行時に再度起動しません。

遅延実行中に起動したアクションは、システムの一部としてインストールされているファイルにアクセスできます。したがって、インストールのこの部分で製品とともにインストールされた DLL ファイルから関数を呼び出すカスタム アクションを呼び出すことができます。ただし、遅延実行カスタム アクションを正しく実行するには、InstallInitialize と InstallFinalize の間で行う必要があります。

ターゲット システムにインストールされているファイルを使用するカスタム アクションや、既に実行されたその他のシステム変更を使用するカスタム アクションは、遅延実行を行うようスケジューリングする必要があります。

ロールバック実行

ロールバックは、インストールにエラーが発生した時や、エンドユーザーがインストール完了前にキャンセルした時に実行されます。ロールバック実行オプションを使用すると、ロールバック中にのみアクションが実行されるよう設定できます。したがって、ロールバック実行できるアクションは、遅延実行アクションのようにインストールスクリプトに書き込まれます。遅延実行アクションと異なり、ロールバック実行アクションはロールバック中にしか実行されません。(ロールバック カスタム アクションは、インストールが遅延実行中に失敗した場合にのみ実行されます。)

インストール中にターゲット システムに変更を加えるカスタム アクションは、ロールバック中にロールバック実行カスタム アクションでやり直されます。たとえば、ファイルを作成するカスタム アクションがある場合、ファイルを削除する 2 つ目のカスタム アクションを作成して、2 番目のアクションをロールバック実行にスケジューリングします。(ロールバックカスタム アクションは、元に戻すカスタム アクションの前にスケジューリングしてください。)

コミット実行

コミット実行は、InstallFinalize アクションが正常に完了するまで実行されません。つまり、インストールがファイル転送、COM サーバー登録、およびショートカットとレジストリ エントリの作成を完了するまで行われません。それから、コミット実行に設定されているアクションはシーケンスの表示順に実行されます。

たとえば、一時ファイルを作成するカスタム アクションがある場合、ファイルを削除する 2 つ目のカスタム アクションを作成して、そのアクションをコミット実行にスケジューリングします。

システム コンテキストの遅延実行

遅延実行アクション同様、これらのアクションも Windows Installer サービスによって生成されたスクリプトが実行されなければ実行されません。ただし、このタイプのアクションはユーザー偽装を実行しません。

カスタム アクションの動作をドキュメント化する



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*



***Windows ロゴ・** 各カスタム アクションの意図された動作は、Windows ロゴ プログラムに基づいて文書化する必要があります。これは、システム管理者が製品を企業環境に配布する場合など、カスタム アクションの動作を把握する必要がある場面で特に有益です。インストール パッケージを検証した時、ヘルプ ファイル パス 設定の値が指定されていない場合、InstallShield は検証エラー ISICE10 を出力します。詳細については、「ISICE10」を参照してください。*



タスク** **プロジェクトに含まれるカスタム アクションの動作をドキュメント化するには、次の手順に従います。

1. カスタム アクションの正常な動作を説明するファイルを作成します。.txt、.htm、.rtf ファイルなど、テキストベースのファイルを指定してください。カスタムアクション 1 個につき、1 つのファイルが必要です。
2. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
3. **[カスタム アクション]** エクスプローラーで、ドキュメントを作成するアクションをクリックします。
4. **[ヘルプ ファイル パス]** 設定については、省略記号ボタン (...) をクリックして、カスタム アクションの動作について説明するファイルを参照します。



ヒント: *ビルド時に InstallShield が、各カスタム アクション ヘルプ ファイルのコンテンツを .msi ファイルにストリームするかどうかを指定することができます。詳細については、[リリース] ビュー内にある製品構成の "カスタム アクションのヘルプを含める" 設定の説明を参照してください。*

Windows ロゴ プログラムの要件を満たすカスタム アクションを作成するときのガイドライン



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

Windows ロゴを適用する場合、アプリケーションは Microsoft Windows ロゴ プログラムの要件を満たさなくてはいけません。検証スイートを使って、作成中のインストール パッケージがカスタム アクション関連のロゴ要件のほとんどを満たしているかどうかを検証できます。ただし、一部の要件については検証スイートを使って検証することができません。

検証スイートまたは関連 ISICE で検証されないけれども、Windows ロゴ プログラムのロゴ コンプライアンスを達成するために満たす必要がある要件の一覧は次のとおりです。

- グローバル アセンブリ キャッシュ ツール (Gacutil.exe) をカスタム アクションから呼び出さない。このツールは、インストール中の使用のためには設計されていません。
- すべてのカスタム アクションの成功または失敗について、**MsiProcessMessage** を呼び出して Windows Installer ログに記録する。
- インストール中にシステム状態を変更するカスタム アクションは、アンインストール中に削除するか、元のシステム状態に復元しなくてはならない。また、システム状態を変更するカスタム アクションは、遅延およびロールバック カスタム アクションとの対で作成しなくてはならない。

  これは、カスタム アクション タイプ 19 (エラー、失敗、およびインストールの終了を表示する)、タイプ 35 (インストール ディレクトリを設定する)、または タイプ 51 (プロパティを設定する) には適用されません。
- インストールに含まれるすべてのカスタム アクションの正常な動作をドキュメント化しなくてはならない。これは、カスタム アクション タイプ 19 (エラー、失敗、およびインストールの終了を表示する)、タイプ 35 (インストール ディレクトリを設定する)、または タイプ 51 (プロパティを設定する) には適用されません。詳細については、「カスタム アクションの動作をドキュメント化する」を参照してください。

  様々な機能をサポートするために InstallShield プロジェクトに自動的に追加される各 InstallShield カスタム アクションについての情報は、「InstallShield カスタム アクション リファレンス」を参照してください。

カスタム アクションをリリース フラグに基づいて条件付きで起動する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

インストールがビルドされると、は自動的に ISReleaseFlags というプロパティを Property テーブルに追加します。このプロパティには、ビルド内のすべてのリリースフラグが含まれています。これらのフラグは [リリース] ビューの "リリース フラグ" 設定と同じ状態で表示されます。複数のフラグはコンマで区切られています。

カスタム アクションを、リリース フラグに基づいて次の方法を使って条件付きで起動させることができます。

"条件" 設定を構成する

カスタム アクションを条件付きで起動する最も簡単な方法は、シーケンスにカスタム アクションを 挿入する時に、"条件" 設定を構成する方法です。この場合、"条件" 設定に **ISReleaseFlags><"MyReleaseFlag"** などの条件を入力することができます。この条件は、**"MyReleaseFlag"** という部分文字列が ISReleaseFlags プロパティの値に含まれている場合に成功します。

MsiGetProperty の呼び出し

リリース フラグに基づいてカスタム アクションの条件を設定するより効果的な方法は、カスタム アクション内で **MsiGetProperty** を呼び出すよう作成する方法です。アクション起動を決定するだけでなく、そのアクションによって実行されるコードも指定することができます。たとえば、すべての場合に対して .DLL から一つの関数を呼び出しながら、関数の条件付論理を使ってそれぞれの場合に別の機能を提供することができます。

この方法によるカスタム アクションの条件付き起動の欠点は、スクリプトまたは DLL カスタム アクションしか使用できないことです。さらに、カスタム アクションは見えないところでも確実に実行します。アクションを実行すると、アクションは、指定したリリースフラグがビルドに含まれているかどうか確認します。そのフラグがあれば、カスタム アクションは続行します。フラグがなければ、アクションは終了します。

.msi データベースにファイルを配置し、実行時に抽出する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[サポート ファイル] ビュー (基本の MSI プロジェクトの場合) および [サポート ファイル/ビルボード] ビュー (InstallScript MSI プロジェクト) を使うと、インストール プログラムが使用するだけで、実際にインストールはされない一時ファイルを格納できます。これに該当するものには、ライセンスのテキストファイル (SdLicense で表示) や、インストーラーが呼び出す DLL ファイルなどがあります。実行時のサポート ファイルの抽出先は、Windows Installer プロパティ SUPPORTDIR に格納されています。



メモ: *基本の MSI インストールおよび InstallScript MSI インストール インストールの遅延/コミット/ロールバックのカスタム アクションは、一部のビルトイン Windows Installer プロパティ (*CustomActionData、ProductCode、*および* UserSID*) にのみアクセスすることができます。遅延/コミット/ロールバックの実行中に、カスタム アクションで他のプロパティ (例、*SUPPORTDIR*) にアクセスする場合は、それらを* CustomActionData *に渡す必要があります。詳細については、「遅延/コミット/ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。*



プロジェクト: *Windows Installer プロパティ* SUPPORTDIR *の値は、InstallScript システム変数 SUPPORTDIR の値とは同じではありませんので注意してください。*

カスタム アクション スクリプトの例

InstallScript

インストール時に特定のサポート ファイルにアクセスするには、基本の MSI プロジェクトと InstallScript MSI プロジェクトでは、Windows Installer プロパティ SUPPORTDIR を参照する必要があります。ファイルの完全パスを取得するには、**MsiGetProperty** を使用してパスをクエリしてから、そのファイル名を SUPPORTDIR の値にファイル名を付加します。カスタム アクションの設定では、これは単に [SUPPORTDIR] です。

InstallScript カスタム アクションでは、次のようなコードを使用することができます。

```
export prototype STRING GetSupportFilePathMSI(HWND);

function STRING GetSupportFilePathMSI(hMSI)
    STRING szSupportDir[MAX_PATH + 1];
    STRING szMyFile[MAX_PATH + 1];
    NUMBER nLength;
begin
    // バッファー サイズの初期値を設定します
    nLength = MAX_PATH + 1;
    MsiGetProperty(hMSI, "SUPPORTDIR", szSupportDir, nLength);

    // バッファー サイズ変数をリセットします
    nLength = MAX_PATH + 1;
    MsiGetProperty(hMSI, "MYFILE", szMyFile, nLength);

    // 完全ファイル パスを返します
    return szSupportDir ^ szMyFile;
end;
```

この例では、使用されているサポート ファイルの名前が MYFILE プロパティに格納されます。

C++

C++ では、次のコードが利用できます:

```
UINT _stdcall ShowSupportdir(MSIHANDLE hInstall)
{
    TCHAR szSupportDir[MAX_PATH + 1] = {'¥0'};
    DWORD dwBuff = sizeof(szSupportDir);
    MsiGetProperty(hInstall,"SUPPORTDIR",szSupportDir,&dwBuff);
    MessageBox(NULL,szSupportDir,"SUPPORTDIR is ...",MB_OK);
```

```
}
```

VBScript

VBScript では、次のコードが利用できます:

```
dim strSupportDir
strSupportDir = Session.Property("SUPPORTDIR")
```

MsiGetProperty

遅延/コミット/ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

基本の MSI インストールおよび InstallScript MSI インストール インストールの遅延/コミット/ロールバックのカスタム アクションは、一部のビルトイン Windows Installer プロパティ (CustomActionData、ProductCode、および UserSID) にのみアクセスすることができます。遅延/コミット/ロールバックの実行中に、カスタム アクション で他のプロパティにアクセスする場合は、それらを CustomActionData に渡す必要があります。これを行うには、すぐにプロパティを設定するタイプのカスタム アクションをスケジュールして、カスタム アクションの名前に一致するプロパティを設定します。このプロパティの値が、遅延/コミット/ロールバックのカスタム アクション内の CustomActionData プロパティから取得できるようになります。

CustomActionData を使用して、プロパティにアクセスする

以下の例は、遅延 InstallScript カスタム アクションで、Windows Installer プロパティ SUPPORTDIR にアクセスする方法です。



タスク

遅延 InstallScript カスタム アクションで SUPPORTDIR にアクセスするには、以下の手順に従います:

1. **[カスタム アクションとシーケンス]** ビューで、**GetSUPPORTDIR** という名前のプロパティ設定型カスタム アクション (type 51) を作成します。カスタム アクションの ***"プロパティ名"***、***"プロパティ値"***、および ***"インストール実行シーケンス"*** 設定を次のように構成し、他の設定は空白のままにしておきます。
   - プロパティ名: DisplaySupportDir
   - プロパティ値: [SUPPORTDIR]
   - インストール実行シーケンス: After InstallInitialize
2. **InstallScript** ビューで、**DisplaySupportDir** という名前で新しいカスタム アクションを作成します。
3. SUPPORTDIR の値を含むメッセージ ボックスを表示する次のコードを追加します。

```
function DisplaySupportDir(hMSI)
   STRING supportDirPath;
   NUMBER supportDirPathBuffer;
begin
```

```
    supportDirPathBuffer = MAX_PATH;
    if(MsiGetProperty(hMSI, "CustomActionData", supportDirPath, supportDirPathBuffer) == ERROR_SUCCESS) then
      SprintfBox(INFORMATION,"遅延実行","SUPPORTDIR の値は、%s",supportDirPath);
      SprintfBox(INFORMATION,"遅延実行","SUPPORTDIR の値は、%s",SUPPORTDIR);
    endif;
end;
```

4. **[カスタム アクションとシーケンス]** ビューで、**DisplaySupportDir** という名前の InstallScript カスタム アクションを作成します。カスタム アクションの **"関数名"**、**"スクリプト内実行"**、および **"インストール実行シーケンス"** 設定を次のように構成し、他の設定は空白のままにしておきます。

   - 関数名: DisplaySupportDir
   - スクリプト内実行: システム コンテキストの遅延実行
   - インストール実行シーケンス: After GetSUPPORTDIR



***重要・**ステップ 1 で入力するプロパティ名は、ステップ 4 で作成するカスタム アクションの名前と一致しなくてはなりません。*

CustomActionData を使用して、複数のプロパティにアクセスする

遅延/コミット/ロールバック カスタム アクションで、複数の Windows Installer プロパティにアクセスするには、CustomActionData でそれらのプロパティを "パック" してから、CustomActionData の値を取得したあと、遅延/コミット/ロールバック カスタム アクションを使って、それらの値を "アンパック" します。

たとえば、[INSTALLDIR]、[SUPPORTDIR]、および [SetupType] の値を取得する場合、type 51 カスタム アクションの Property Value 設定を次のように設定します。

[INSTALLDIR];[SUPPORTDIR];[SetupType]

各プロパティは、セミコロンで区切ります。

CustomActionData プロパティからの値を "アンパック" するには、次の VBScript のようなコードを使用します。

```
Dim PropArray
PropArray = Split(Session.Property("CustomActionData"), ";")
INSTALLDIR = PropArray(0)
SUPPORTDIR = PropArray(1)
SetupType = PropArray(2)
'変数を使用することができます...
```

InstallScript カスタム アクション



***プロジェクト・**この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

Windows Installer インストール内の InstallScript 関数を使用するには、最初に、スクリプトを呼び出すカスタム アクションを作成する必要があります。カスタム アクション ウィザードを利用して、カスタム アクションを最も簡単に作成することができます。

このウィザードの主なパネルは次のとおりです。

- [アクションの種類] パネル
- [アクション パラメーター] パネル
- [応答オプション] パネル
- [シーケンスに挿入] パネル

ウィザードを完了すると、タイプ 65536 のカスタム アクションが作成されます。このカスタム アクションタイプは、Windows Installer ヘルプライブラリ で定義されていません。InstallScript は Windows Installer のビルトイン機能ではなく、アクションタイプ 65536 は InstallShield によって作成されています。



プロジェクト: *InstallScript カスタム アクションは、InstallScript MSI プロジェクトのユーザー インターフェイス シーケンスでは使用しないでください。InstallScript MSI プロジェクトにユーザーインターフェースを提供するには、イベント ドリブン型のスクリプトを使用します。*

標準 DLLI ファイルの関数を呼び出す



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Windows Installer では、カスタム アクション向けに作成された DLL ファイル内 の関数に渡すことができるパラメーターに制限がありますが、InstallShield には解決策が用意されており、任意の個数のパラメーターを関数に渡して戻り値を格納できます。関数は必ず _stdcall 呼び出し規則を使用しなくてはなりません。この規則が使用されていない場合、複数のパラメーターを持つ関数は適切に動作しません。

カスタム アクション ウィザードの [アクションの種類] パネルで **[標準ダイナミック リンク ライブラリ内の関数の呼び出し]** を [種類] オプションに選択して、標準 DLL を呼び出すように指定できます。次のパネル、[関数定義] パネルで、関数のパラメーターと戻り値を指定できます。



メモ: *製品と共にインストールされる標準 DLL ファイルの関数を呼び出し、アクションが ("スクリプト内実行" 設定で) "遅延" とスケジュールされている場合、呼び出す DLL ファイルはコンポーネントのキーファイルでなくてはなりません。*

Binary テーブル内の標準 DLL ファイルの関数を呼び出すカスタム アクションは、"遅延実行" をスケジュールできません。

Windows Installer プロパティを関数の戻り値に設定する場合、アクションを即時実行に構成する必要があります。遅延、ロールバック、またはコミット実行にスケジュールされているアクションの戻り値プロパティを指定しようとすると、実行時に戻り値プロパティは無視されます。

フォーマットされた文字列としてのパラメーターと戻り値について

標準 DLL ファイル アクションの関数シグネチャを指定するとき、そのアクションの "関数シグネチャ" 設定は、以下の形式で関数名、引数、戻り値タイプ、および戻り値プロパティを表示します。

DataType1=[PropertyName1] DllName::FuncName(in DataType2="値", Direction DataType3=[PropertyName2])

次のテーブルは、フォーマットのそれぞれの部分についての説明です。

テーブル 6 - 関数の引数のフォーマット

引数	説明
DataType1=[PropertyName1]	DataType1 のタイプは、void、STRING、BOOL、NUMBER、HWND、HANDLE、または POINTER のどれかです。PropertyName1 は、[プロパティ マネージャー] ビュー からのプロパティ名です。 戻り値が void の場合、=[PropertyName1] は省略されます。
DllName	ピリオドと拡張子を除く DLL ファイル名を指定します。
FuncName	呼び出す関数の名前を入力します。
in DataType2="Value"	これは、定数である引数の形式です。DataType1 のタイプは、STRING、BOOL、NUMBER、HWND、HANDLE、または POINTER のいずれかです。Value は、この引数に渡すデータです。定数のデータ型は、常に in から始まります。
Direction DataType3=[PropertyName2]	これは、Windows Installer のプロパティ値を参照する引数の形式です。Direction には、*in*、*inout*、または *out* のいずれかを指定できます。これらのプロパティ タイプについては、引数のソースの下にある [関数定義] パネルを参照してください。 DataType3 では、STRING、BOOL、NUMBER、HWND、HANDLE、または POINTER のタイプを使用できます。PropertyName2 は、Direction の値に応じて、値が関数に渡されるプロパティ、または関数によって値が設定されるプロパティの名前です。

例

インストールで MessageBoxA を呼び出す で作成されたカスタム アクションの場合、Target の値は次のようになります。

void User32::MessageBoxA(in HWND=0, in STRING=[MESSAGEPROP], in STRING=[CAPTIONPROP], in NUMBER=1) と表示されます。

ここ例で、データタイプは *void* で、*in HWND=0* は数値 0 を表します。また *MESSAGEPROP* は、User32.dll の関数 MessageBoxA に渡される文字列を含むプロパティです。

Windows Installer DLL ファイル内の関数を呼び出す



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

Windows Installer のカスタム アクション専用に作成されている DLL 関数では、インストーラー データベースへのハンドルだけをパラメーターとして使用できます。.msi データベースから情報を読み込み、カスタム アクションで使用する方法について、以下に説明します。

カスタム アクション ウィザードの [アクションの種類] パネルで **[Windows Installer ダイナミック リンク ライブラリ内の関数の呼び出し]** を選択して、カスタム アクションが Windows Installer DLL ファイル (カスタム アクション タイプ 1 および 17) にあることを指定します。

別の方法としては、[アクションの種類] パネルで **[標準ダイナミック リンク ライブラリ内の関数の呼び出し]** を選択することができます。この場合、InstallShield で関数のパラメーターを指定することができます。



タスク ***Windows Installer 用に作成された DLL の関数にパラメーターを渡す場合、主に 3 つのステップが必要です。***

1. DLL ファイルを作成します。
2. カスタム アクションを作成してシーケンスの 1 つに挿入します。
3. [プロパティ マネージャー] ビューを使用して、パラメーターを渡します。以下に、これらのステップを詳しく説明します。

DLL ファイルの作成

カスタム アクションの DLL ファイル関数にデータを渡すには、データを渡す関数で、Installer プロパティの値を取得する **MsiGetProperty** 関数を呼び出す必要があります。以下の例では、**MYPROPERTY** というパブリック プロパティから値を取り出します。

```
UINT _stdcall MyActionName(MSIHANDLE hInstall)
{
    TCHAR szValue[51] = {0};
    DWORD dwBuffer = 50;

    MsiGetProperty(hInstall, TEXT("MYPROPERTY"), szValue, &dwBuffer);

    MessageBox(GetForegroundWindow( ),
        szValue,
        TEXT("MYPROPERTY の値"), MB_OK | MB_ICONINFORMATION);

    return ERROR_SUCCESS;
}
```

さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「MsiGetProperty」を参照してください。

DLL ファイルのビルドに C++ コンパイラを使用している場合は、コンパイラが DLL からエクスポートされた関数名を変更していないよう確認してください。たとえば Microsoft Visual C++ では、DLL ファイルからエクスポートする関数名を指定する .def ファイルを作成することができます。通常の .def (MyActions.def という名前) ファイルは次のような形式です。

```
LIBRARY MyActions
EXPORTS
   MyActionName
```

関数名修飾についての詳細は、コンパイラのマニュアルを参照してください。

カスタム アクションを作成してシーケンスに挿入する

2 番目のステップでは、カスタム アクションを作成して、それを [インストール] シーケンスに挿入します。カスタム アクションを作成するには、カスタム アクション ウィザードを使用し、カスタム アクションを配置するには、[カスタム アクションとシーケンス] ビューを使用します。

Property Manager を使ってパラメーターを渡す

タスク ***プロパティ マネージャーを使用して新しいパラメーターを渡すには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
2. **[新しいプロパティ]** ボタンをクリックします。ビューの下に新しい行が追加されます。
3. **Name** 列で、**MsiGetProperty** (この例では **MYPROPERTY**) を使用して取得するプロパティの名前を入力します。
4. **Value** 列で、渡す値を入力します。たとえば Web サイトに URL を渡す場合、**http://www.mycompany.com** と入力します。

カスタム アクションで DLL ファイル関数にパラメーターを渡す

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

カスタム アクション中の関数へパラメーターを渡すのには多くの理由があります。たとえば、Web 登録のために URL を渡したり、カスタムユーザーインターフェイスに UserName プロパティを渡したりする場合などです。ただし、Windows Installer は DLL ファイル関数へパラメーターを渡すことを直接にはサポートしません。

Windows Installer DLL ファイルのエントリポイント関数が持つことのできる唯一の引数はデータベースへのハンドルです。Windows Installer 用に書かれた DLL ファイルにパラメーターを渡す方法については、「Windows Installer DLL ファイル内の関数を呼び出す」を参照してください。



タスク ***InstallShield でこれを実現するには、以下の手順を実行します。***

1. カスタム ウィザードの **[アクションの種類]** パネルで、**[標準ダイナミックリンクライブラリの関数を呼び出す]** を選択します。
2. **[関数定義]** パネルで、InstallShield が関数に渡すパラメーターを指定します。

マネージ アセンブリのパブリック メソッドを呼び出す



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

Visual Basic .NET または C#のようなマネージ コードで書かれた .NET アセンブリ内のパブリック メソッドを呼び出すカスタム アクションの種類を使用します。このセクションには、次のトピックが含まれています:

- マネージ コード ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す
- .NET Core ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す
- アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する

マネージ コード ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

マネージ コード タイプのカスタム アクションは、Visual Basic .NET または C# などのマネージ コードで書かれた .NET アセンブリ内にあるパブリック メソッドを呼び出します。

プロジェクトにマネージ コード カスタム アクションを含めると、.NET アセンブリに C++ Windows Installer ラッパー DLL が作成されます。ラッパー DLL には、アセンブリのほか、アセンブリの仲介、ロード、実行を行うために必要な情報が含まれています。



メモ: *マネージ アセンブリの場所を* Binary *テーブルに指定すると、ラッパー (アセンブリが含まれています)* Binary *テーブルに格納されます。*

マネージコード カスタム アクションの実行時要件



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

マネージ コード タイプのカスタム アクションは、ターゲット システムに Microsoft .NET Framework が必要です。

InstallShield には .NET Framework の再配布可能ファイルが含まれています。.NET Framework がターゲット システムにない可能性がある場合、適切な .NET Framework 再配布可能ファイルをプロジェクトに追加することができます。手順については、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。

IS_CLR_VERSION プロパティを利用すると、カスタム アクションがダウンロードしてマネージ コードを実行するセミコロンで区切られた .NET Framework のバージョン一覧を識別することができます。ほとんどのシナリオで、このプロパティはインストール パッケージで設定されていません。コマンドラインで設定されます。バージョン 1.1 が必要であることを指定する場合、次のコマンドライン パラメーターを使用します:

```
IS_CLR_VERSION=v1.1.4322
```

プロパティの値には、.NET Framework の完全バージョン番号を指定する必要があることに注意してください。複数のバージョンが使用可能な場合、バージョンの一覧をセミコロンで区切って指定できます。ロードが可能であるバージョンの中から最初のものが使用されます。次のコマンドライン パラメーターの例では、カスタム アクションがバージョン 2.0 をロードしようと試みます。このバージョンが存在しない場合、バージョン 1.1 のロードが試みられます。バージョン 1.1 が見つからなかった場合、カスタム アクションは失敗します。

```
IS_CLR_VERSION=v2.0.50727;v1.1.4322
```

指定したバージョンがインストールされなかったとき、カスタム アクションで、インストールされている .NET Framework の中から一番新しいバージョンをロードする試みを行うように指定する場合、下の例のようにセミコロンをプロパティ値の最後に付加します:

```
IS_CLR_VERSION=v2.0.50727;v1.1.4322;
```

また、プロパティ値の最後に付加されたセミコロンは、指定されたバージョンが存在しないが、あるバージョンの .NET Framework が既にロードされている場合、それが既にインストールされている最新バージョンではなくても、現在ロードされているバージョンを使用することを示します。



メモ: *マネージ コード カスタム アクションの実行が遅延モードに設定されている場合、Windows Installer プロパティ* CustomActionData *を使って* IS_CLR_VERSION *プロパティと値を渡す必要があります。詳細については、「アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する」を参照してください。*



ヒント: *.NET Framework のバージョンの不一致が原因でカスタム アクションで問題が発生した場合、インストールの実行時に、エンドユーザーにコマンドラインで* IS_CLR_VERSION *プロパティを設定するように指示することも考えられます。遅延カスタム アクションの場合、*CustomActionData *で* IS_CLR_VERSION *プロパティが既に渡されていることに注意してください。詳細については、「アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する」を参照してください。*

32 ビット マネージ コード カスタム アクションと 64 ビット マネージ コード カスタム アクションの違い



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

プロジェクトでマネージコード カスタム アクションを含むリリースをビルドするとき、InstallShield はカスタム アクションと関連付けられたメインの .NET アセンブリのターゲット アーキテクチャ (32 ビットまたは 64 ビット) を判別しようとします。InstallShield は、アセンブリのポータブル実行形式ファイル (PE) を調査して、PE ファイルが 32 ビットであるか、64 ビットであるかを判別します。PE ファイルが Microsoft intermediate language (MSIL) コードを使用する場合、InstallShield はこれを 32 ビットとして取り扱います。InstallShield は、実行時にマネージ コードを実行するときに .NET Framework の適切なバージョン (32 ビットまたは 64 ビット) が使用されるようにリリースを構成します。

組み込まれたデフォルトの動作をオーバーライドするには、**[ダイレクト エディター]** ビューを使って、次のフィールドを含む新しいレコードを ISClrWrap テーブルに追加します。

テーブル 7 ・ マネージ コード カスタム アクションのデフォルト アーキテクチャをオーバーライドする

ISClrWrap テーブル列	説明
Action_	変更するマネージコード カスタム アクションの名前を入力します。
名前	次を入力します: **TargetPlatform**
値	適切なアーキテクチャを指定します。 • .NET Framework の 32 ビット バージョンを使用するには、次を入力します:**x86** • .NET Framework の 64 ビット バージョンを使用するには、次を入力します:**x64**

.NET Core ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

.NET Core タイプのカスタム アクションは、Visual Basic .NET または C#のようなマネージ コードで書かれた .NET アセンブリ内のパブリック メソッドを呼び出します。

.NET Core カスタム アクション タイプをプロジェクトに含んだ場合、InstallShield によって .NET アセンブリの C++ Windows Installer ラッパー DLL が作成されます。ラッパー DLL には、アセンブリのほか、アセンブリの仲介、ロード、実行を行うために必要な情報が含まれています。



メモ: *マネージ アセンブリの場所を* Binary *テーブルに指定すると、ラッパー (アセンブリが含まれています)* Binary *テーブルに格納されます。*

.NET Core カスタム アクションの実行時要件



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

.NET Core タイプのカスタム アクションを実行するには、次がターゲット システム上にインストールされている必要があります:

- Microsoft .NET Core 8.0
- Microsoft Visual C++ 2022 再配布可能ファイル

InstallShield には .NET Core の再配布可能ファイルが含まれています。ターゲット システムに .NET Core が搭載されていないことが予想される場合、プロジェクトに適切な .NET Core 再配布可能ファイルを追加することができます。手順については、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。

32 ビットと 64 ビット .NET Core カスタム アクションの違い



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

プロジェクトに .NET Core カスタム アクションを含むリリースをビルドするとき、InstallShield はカスタム アクションと関連付けられているメインの .NET アセンブリのターゲット アーキテクチャ (32 ビットまたは 64 ビット) を判別しようとします。InstallShield は、アセンブリのポータブル実行形式ファイル (PE) を調査して、PE ファイルが 32 ビットであるか、64 ビットであるかを判別します。PE ファイルが Microsoft intermediate language (MSIL) コードを使用する場合、InstallShield はこれを 32 ビットとして取り扱います。InstallShield は、実行時にマネージ コードを実行するために .NET Core 32 ビットまたは 64 ビットの適切なバージョンを使用するリリースを構成します。

組み込まれた既定の動作をオーバーライドするには、**[ダイレクト エディター]** ビューを使って、次のフィールドを含む新しいレコードを ISCIrWrap テーブルに追加します:

テーブル 8 ・ .NET Core カスタム アクションの既定のアーキテクチャをオーバーライドする

ISCIrWrap テーブル列	説明
Action_	変更する .NET Core カスタム カスタム アクションの名前を入力します。
名前	次を入力します: **TargetPlatform**
値	適切なアーキテクチャを指定します。 ・ .NET Core の 32 ビット バージョンを使用するには、次を入力します :**x86** ・ .NET Core の 64 ビット バージョンを使用するには、次を入力します :**x64**

アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する



プロジェクト・ *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

カスタム アクション ウィザードの [マネージ メソッドの定義] パネルで、マネージ アセンブリ カスタム アクションが呼び出すメソッドの引数を指定できます。[メソッド シグネチャ] ダイアログ ボックスでも、この情報を指定することができます。

デフォルトのメソッド シグネチャと利用について

デフォルトのメソッド シグネチャを利用すると、カスタム アクションは最高パラメーターを 1 つまで使ってメソッドを処理します。メソッドでパラメーターが使用されると、メソッドは MsiHandle および任意の戻り値に渡されます。メソッドが整数を戻すと、この整数は直接 Windows Installer に渡されます。つまり、このシナリオでメソッドが整数 1603 (ERROR_INSTALL_FAILURE) を戻すと、インストールは中止されます。

詳しい情報は、Windows Installer ライブラリの「Custom Action Return Values」を参照してください。

即時カスタム アクションにカスタム メソッド シグネチャを使用する

カスタム メソッド シグネチャを利用すると、カスタム アクションは指定した引数と共にメソッドを呼び出します。パラメーターの値が Windows Installer プロパティで、カスタム アクションが即時実行に設定されている場合、メソッドはすべての参照 (ref) および出力 (out) パラメーターの値を渡されたプロパティに格納します。カスタム アクションは、また、戻り値を文字列に変換し、それを戻り値のプロパティに格納します。戻り値プロパティが指定されていない場合、戻り値の値は無視されます。

カスタム シグネチャを使用している場合、カスタム アクションの戻り値は Windows Installer に渡されません。ただし、マネージ コードが未処理の例外をスローすると、カスタム アクションは ERROR_INSTALL_FAILURE を Windows Installer に戻します。つまり、カスタム アクションで Windows Installer についてインストールの失敗を示すには、マネージ コードで未処理の例外をスローする必要があります。

遅延/コミット/ロールバック カスタム アクションにカスタム メソッド シグネチャを使用する

遅延/コミット/ロールバック カスタム アクションは、一部のビルトイン Windows Installer プロパティ (CustomActionData、ProductCode、および UserSID) にのみアクセスすることができますこのため、カスタム メソッド シグネチャを使用するとき、遅延/コミット/ロールバックの実行中に、マネージ アセンブリ カスタム アクションで他のプロパティを渡す場合は、それらを CustomActionData を通して渡す必要があります。遅延/コミット/ロールバック マネージ アセンブリ カスタム アクションについては、次のガイドラインに留意してください:

- プロパティを渡すカスタム シグネチャの値は、次の形式で CustomActionData として渡される必要があります:

  PROPERTYNAME="プロパティの値"

  複数のプロパティ名プロパティの値のペアは、次の例のようにスペースで区切ります:

  PROPERTYNAME1="プロパティ値 1" PROPERTYNAME2="プロパティ値 2"

- 遅延/ロールバック/コミット カスタム アクションのマネージ アセンブリが製品と共にインストールされる場合、CustomActionData を利用して、マネージ アセンブリの場所を識別し、次の形式を使用します:

  #filekey.dll="アセンブリの場所"

  "アセンブリの場所" の部分には、**[#filekey.dll]** という形式を使用します。

- マネージ アセンブリのパスが 1 つまたは複数のプロパティを参照する場合、CustomActionData を使って、各プロパティと対応する値を渡します。

- CustomActionData を使って、次のプロパティと値を渡すことを検討してください:

  IS_CLR_VERSION="[IS_CLR_VERSION]"

  メモ・この情報はマネージ コード カスタム アクション タイプにのみ適用します。

  カスタム アクションが正しくないバージョンの .NET Framework をロードしてマネージ コードを実行しようとしたときに、実行時に問題が発生する場合、インストールの実行時に、エンドユーザーがコマンドラインで IS_CLR_VERSION を設定するように構成することができます。遅延/コミット/ロールバック マネージ コード カスタム アクションの場合、エンドユーザーが実行時にこのオーバーライドを使えるように IS_CLR_VERSION を CustomActionData に渡す必要があります。IS_CLR_VERSION プロパティについては、「マネージコード カスタム アクションの実行時要件」を参照してください。

カスタム メソッド シグネチャのパラメーター値を指定する

カスタム メソッド シグネチャを利用すると、パラメーターに次の種類の値を入力できます:

- 角かっこで囲まれた単一プロパティ (例、**[ProductName]**)。即時モードのマネージ アセンブリ カスタム アクションは ref および out パラメーターを更新することができます。InstallShield では、パラメーターの値に、複数のプロパティを組み合わせたり、プロパティと文字列を混合させて使用することはできません。

- リテラル ー たとえば、数字 **1** や **1603** などの文字列。

- 引用符で囲まれている明示的な文字列 (例、**「マイ プロジェクト名-1」**)。

- 変数 *MsiHandle* はインストールのハンドルです。

実行時に、プロパティが解決されたあと、または文字列が引用符内から分離されたあと、カスタム アクションはその文字列をそのメソッドの適切なパラメーターの種類のインスタンスに変換します。String 型のパラメーターは、そのままの形で渡され、すべて標準数値型は、文字列をその型に変えるパブリック スタティック Parse(string) メソッドを呼び出して処理されます。カスタム アクションは、すべての型のパブリック スタティック Parse(string) メソッドを呼び出し、その戻り値を渡されたパラメーターの値として使用します。IntPtr は、Parse メソッドがなくても処理されます。

プロセスの強制終了カスタム アクションの呼び出し



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *トランスフォーム*

Kill-Process タイプのカスタム アクションによって、実行時に、1 つまたは複数のプロセスが終了されます。以下は、プロジェクトで、このタイプのカスタム アクションの追加および構成方法の説明です。



タスク ***Kill-Process カスタム アクションをプロジェクトに追加するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、**[新しいプロセスの強制終了]** を選択します。プロセスの強制終了カスタム アクションがデフォルトの名前で追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このカスタム アクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。
4. 右側のペインで、アクションの設定を構成します。
   a. **"スクリプト内実行"** 設定で、アクションをトリガーするシーケンスの繰り返しを選択します。各オプションの詳細については、「スクリプト内実行」を参照してください。
   b. **[シーケンス]** 領域の設定を使って、プロセスを強制終了する位置にカスタム アクションをスケジュールします。代わりに、カスタム アクションを **[カスタム アクション]** エクスプローラーから、[シーケンス] エクスプローラーの下の適切なノードにドラッグします。
   c. **"関数名"** 設定で、適切な関数を選択します。
      - KillProcess— **"スクリプト内実行"** 設定で即時オプションのいずれかが選択されていて、特定の名前を持つプロセスを強制終了する場合、このオプションを選択します。

- KillProcessByID—**"スクリプト内実行"** 設定で即時オプションのいずれかが選択されていて、特定のプロセス ID (PID) を持つプロセスを強制終了する場合、このオプションを選択します。
- KillProcessDeferred—**"スクリプト内実行"** 設定で、遅延、コミット、またはロールバック オプションのいずれかが選択されていて、特定の名前を持つプロセスを強制終了する場合、このオプションを選択します。
- KillProcessByIDDeferred— **"スクリプト内実行"** 設定で、遅延、コミット、またはロールバック オプションのいずれかが選択されていて、特定の PID を持つプロセスを強制終了する場合、このオプションを選択します。

d. **"プロセス"** 設定で、強制終了するプロセスの実行可能ファイルの名前または PID を入力します。

関数名が次に設定されている場合:	次を使ってプロセスを指定します:
KillProcess または KillProcessDeferred	実行可能ファイル名
KillProcessByID または KillProcessByIDDeferred	PID

1 つ以上のプロセスを終了する場合、セミコロン (;) を使って、ファイル名または PID を区切ります。たとえば、**myfile1.exe** および **myfile2.exe** という名前のプロセスを終了する場合、次のように入力します:

myfile1.exe;myfile2.exe

ヒント: *"プロセス" 設定の値は、*ISTerminateProcesses *プロパティに書き込まれる場合があります。InstallShield 2015 以前から移行したプロジェクトなど、**"プロセス"** 設定に値が指定されていない追加の kill-process カスタム アクションがある場合、*ISTerminateProcesses *プロパティを共有して使用すると予期しない動作が発生する可能性があります。*

e. 必要に応じて、アクションの他の設定も構成します。

PowerShell カスタム アクションの呼び出し

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *トランスフォーム*

Windows PowerShell は、構成タスクのオートメーション化を可能にする .NET Framework ベースのコマンドライン シェルおよびスクリプト言語です。InstallShield では、この PowerShell スクリプト (.ps1) を実行するカスタム アクションがサポートされています。この種類のカスタム アクションは、インストールの実行時にシステムの構成タスクを実行するプロジェクトに追加することができます。PowerShell アクションには、ターゲット システム上に PowerShell 2.0 以降が必要です。

PowerShell カスタム アクションをプロジェクトに追加する

タスク

PowerShell カスタム アクションをプロジェクトに追加するには、以下の手順に従います:

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックし、**[新しい PowerShell]** をポイントして、次のいずれかのコマンドをクリックします:
   - **Binary テーブルに保存**—コード ベースを Binary テーブルに格納する場合、このコマンドを選択します。この場所は、ターゲット システムにコード ファイルをインストールしない場合に便利です。
   - **製品と一緒にインストール**—ターゲット システムにインストールされるスクリプト ファイルからコードを呼び出す場合、このコマンドを選択します。

   PowerShell カスタム アクションがデフォルトの名前で追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このカスタム アクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。
4. **"PowerShell スクリプト ファイル名"** 設定で、Binary テーブルに格納されているファイル リスト、またはプロジェクトに含まれているファイル リストから、PowerShell スクリプト ファイル (.ps1) を選択します。指定した場所が、Binary テーブルに格納されている場合、この設定で、省略記号ボタン (...) をクリックして、ファイルを参照します。
5. 必要に応じて、アクションの他の設定も構成します。

プロジェクトにカスタム アクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「アクションをシーケンスに挿入する」を参照してください。

PowerShell カスタム アクションの実行時要件

PowerShell は、デフォルトで、一部のオペレーティング システムにのみインストールされています。インストールで PowerShell カスタム アクションが起動されたときに PowerShell がインストールされていない場合、カスタム アクションは失敗します。

ターゲット システムに PowerShell がインストールされていることを確認する場合、プロジェクトに PowerShell の定義済みシステム検索を追加し、PowerShell カスタム アクションを構成して、システム検索で PowerShell がインストールされていることが判別された場合にのみ実行されるようにします。また、このシステム検索を使って、インストールする PowerShell のバージョンを判別したり、適切なターゲット システム上でのみ起動をトリガするカスタム アクションの条件を含めたりできます。デフォルトで、PowerShell の定義済みシステム検索は、POWERSHELLVERSION プロパティの値を PowerShell のバージョンに設定します。

Windows Installer プロパティ IS_CLR_VERSION を利用すると、カスタム アクションがダウンロードして PowerShell スクリプトを実行するセミコロンで区切られた .NET Framework のバージョン一覧を識別することができます。

POWERSHELLVERSION および IS_CLR_VERSION プロパティについての詳細は、「Windows Installer プロパティ リファレンス」を参照してください。

実行中の基本の MSI または InstallScript MSI インストールとの対話操作

実行中の基本の MSI または InstallScript MSI インストールとの対話操作を可能にする、いくつかの cmdlet があります:

テーブル 9 ・ 実行中の基本の MSI または InstallScript MSI インストールと対話操作を行う cmdlet

Cmdlet	説明
get-property -name [プロパティ名]	この cmdlet は Windows Installer プロパティの値を取得します。Name パラメーターは、位置指定パラメーターです。 次のサンプル コードは、MYPROPERTY という名前のプロパティの値を取得します。 $Value = get-property -name MYPROPERTY
set-property -name [プロパティ名] -value [値]	この cmdlet は Windows Installer プロパティの値を設定します。Name および Value パラメーターは位置指定パラメーターです。 次のサンプル コードは、MYPROPERTY プロパティの値を使って、NEWPROPERTY と呼ばれる 2 番目のプロパティの値を設定します。 $Value = get-property -name MYPROPERTY set-property -name NEWPROPERTY -value $Value
format-properties -Value [形式文字列]	この cmdlet は、形式化された式の値を展開します。Value パラメーターは、位置指定パラメーターです。 次のサンプル コードは、NEWPROPERTY と呼ばれる Windows Installer プロパティを含む文字列をフォーマットします。プロパティ名と、その周りの角かっこは、実行時にプロパティの値で置換されます。 $myFormat = format-properties -Value "これは、[NEWPROPERTY] が埋め込まれたテキスト文字列です。"
trace-info -LogMessage [長い文字列]	この cmdlet により、Windows Installer ログに必要な情報を書き込みし、それらをデバッグやその他の情報提供目的に使用できるようにします。LogMessage パラメーターは、位置指定パラメーターです。 次のサンプル コードは、文字列 **「これは PowerShell カスタム アクションです」**を Windows Installer ログに書き込みます。 trace-info -LogMessage “これは PowerShell カスタム アクションです”

スクリプトが成功したことを示すには、必ず 0 を戻します。たとえば、次の行でスクリプトを終了します:

exit(0)

実行可能ファイルを起動する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

サードパーティ ツールのインストール、Readme ファイルの表示、インストールする製品についての最新情報が記載された Web ページの表示など、インストールから実行可能ファイルを起動すると便利なことがあります。インストール内から実行可能ファイルを起動するには、[カスタム アクションとシーケンス] (基本の MSI、InstallScript MSI、MSI データベース、および トランスフォーム プロジェクトの場合) または [カスタム アクション] (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) にカスタム アクションを追加する必要があります。

カスタム アクション ウィザードを使用して、インストールから実行可能ファイルを起動するカスタム アクションを作成することができます。ウィザードの各パネルを下に示します。実行可能ファイルを起動するには各項目に値を入力する必要があります。

[基本情報] パネル

テーブル 10 - カスタム アクション ウィザードで [基本情報] パネル の設定を行う

パネル オプション	値
名前	カスタム アクションにわかりやすい名前を付けます。この名前は製品内部だけで、識別のために使用されます。
コメント	このカスタム アクションに関するコメントを入力します。

アクションの種類

テーブル 11 - カスタム アクション ウィザードの [アクションの種類] パネルにある設定

パネル オプション	値
Type	作成するカスタム アクションのタイプを選択します。この例では、[実行可能ファイルの起動] を選択します。
場所	選択は、インストール中に実行するターゲット実行可能ファイルの場所によって異なります。どのコンピューター上にも「メモ帳」はあるので、これがカスタム アクションのターゲットとなります。メモ帳は、ターゲット マシン上の Windows フォルダーにあることが保証されているため、Directory テーブルを使用してその位置をポイントすることができます。したがって、**[Directory テーブルに保存]** を選択します。

アクションのパラメーター

テーブル 12 - カスタム アクション ウィザードで [アクションのパラメーター] パネルの設定を行う

パネル オプション	値
ソース	Directory テーブルに保存されている実行可能ファイルを起動することを選択したため、現在 Directory テーブルにあるすべてのエントリを反映した選択リストが表示されます。これらのオプションの 1 つが **WindowsFolder** です。パスを直接入力するのではなく、このオプションを選択して「メモ帳」をポイントします。
ターゲット	このオプションでは、指定されたディレクトリ内にあるターゲットファイルを [ソース] オプションから入力する必要があります。**Notepad.exe** と入力して [次へ] ボタンをクリックし、続行します。

追加オプション

テーブル 13 • カスタム アクション ウィザードで [追加のオプション] パネルの設定を行う

パネル オプション	値
戻り値の処理	Windows Installer で、どのようにカスタム アクション スレッドの処理を制御するかを指定します。メインとカスタム アクションのスレッドが同期実行 (インストールがメイン インストール スレッドを再開する前にカスタム アクション スレッドが完了するのを待機する) か非同期 (インストールがメイン インストールの続行と同時にカスタム アクションを実行する) かを指定することができます。 この例では、**同期 (終了コードを確認)** オプションを選択します。

応答オプション

テーブル 14 • カスタム アクション ウィザードで [応答オプション] パネルの設定を行う

パネル オプション	値
スクリプト内実行	スクリプトの記述が表れたら、すぐにアクションを実行する場合には、[即時実行] を選択します。
実行スケジュール	カスタム アクションの記述が表れるたびにそれを実行する場合には、**[常に実行]** を選択します。

カスタム アクションのシーケンスへの挿入

カスタム アクションを作成後、[カスタム アクションとシーケンス] ビューでカスタム アクションをシーケンスに挿入する必要があります。

Msiexec.exe を使用して 2 番目の Windows Installer インストールを起動する



プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ネスト インストール カスタム アクション タイプを使用して 2 つ目の .msi パッケージを起動する代わりに、**Msiexec.exe** を起動するカスタム アクションを作成する方法があります。この方法で 2 つ目のインストールを起動すると、起動がその独自のプロセスで実行され、ターゲット システムに正しいエントリが作成されます。また、アンインストールプロセスは、この方法で 2 つ目のインストールを起動するとより効果的になります。レジストリ エントリは、正しくクリーンアップされ、参照カウントが正確に増加および減少されます。

カスタム アクションを作成する

まず、カスタム アクションを作成します。カスタム アクションを作成する最も簡単な方法は、カスタム アクション ウィザードを使用することです。

タスク ***ウィザードでカスタム アクションを作成するには、次の手順に従ってください。***

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、**[カスタム アクション ウィザード]** をクリックします。**カスタム アクション ウィザード**が開きます。
3. **[基本情報]** パネルで、カスタム アクションの名前とコメントを指定、**[次へ]** をクリックして次のステップに進んでください。
4. **[アクション タイプ]** パネルの **[タイプ]** リストで、**[実行可能ファイルを起動]** を選択します。**[場所]** リストで、**[Directory テーブルに保存]** を選択します。Directory テーブルを使用すると、SystemFolder などの定義済みフォルダーから選択できます。ここに、Msiexec.exe が保存されます。
5. **[アクションのパラメーター]** パネルで、起動する実行可能ファイル (**Msiexec.exe**) を参照します。**[ターゲット]** ボックスを使用すると、起動する実行可能ファイルの名前、およびそのファイルを渡すコマンドライン パラメーターを指定できます。例:

   msiexec.exe /i "[SOURCEDIR]AnotherSetup¥SecondSetup.msi" /qb

   このエントリの最初の部分、**msiexec.exe** は、起動する実行可能ファイルの名前です。次のセクション、**/i "[SOURCEDIR]AnotherSetup¥SecondSetup.msi"** は、Msiexec.exe に実行するパッケージを知らせます。この場合、最初のインストールのソースフォルダーの下にある、フォルダーのファイルを指しています。最後に、**/qb** は、Windows Installer サービスに、最小限のユーザー インターフェイスによるインストールの実行を通知します。プログレスバーだけが表示されます。
6. **[追加のオプション]** パネルでデフォルトのオプションを選択し、ウィザードが完了するまで **[次へ]** をクリックします。

カスタム アクションをシーケンスへ挿入する

カスタム アクションが作成されたら、それをシーケンスに挿入する必要があります。



タスク ***カスタム アクションを配列するには、以下の手順を実行します。***

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]**（基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合）または **[カスタム アクション]**（マージ モジュールおよび MSM データベース プロジェクトの場合）をクリックします。
2. **[シーケンス]** エクスプローラーで、**[インストール]** シーケンスの下にある **[ユーザー インターフェイス]** アイテムを展開します。このシーケンスにスケジュールされているアクションとダイアログが一覧表示されます。
3. **CostFinalize** を右クリックし、**[挿入]** をクリックします。[アクションの挿入] ダイアログ ボックスが開きます。
4. 新しいカスタム アクションを選択します。
5. **[OK]** をクリックします。

次に、起動するインストール パッケージが [アクション パラメーター] パネルで指定したフォルダーにあることを確認し、インストールをビルドして実行します。



メモ Msiexec.exe *を起動するカスタム アクションは [ユーザー インターフェイス] シーケンスに配置する必要があります。つまり、エンドユーザーがインストールをサイレント モードで実行すると、カスタム アクションは実行されません。*

ネスト インストール



プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*



重要 *Windows Installer ではネスト インストールの使用を避けるようにしてください。ネスト インストールが含まれている場合エンドユーザーが適切に処理することが困難であるため、時折アプリケーションの誤作動を引き起こします。マイクロソフト社では一般向けにリリースされる製品のインストールについて、ネスト インストールおよびネスト インストール カスタム アクションの使用を避けることを推奨しています。詳細については、Windows Installer ヘルプ ライブラリの「Concurrent Installations」を参照してください。*

ネスト インストールは、実行中のインストーラー（*「親製品」*と呼ぶ）から別の .msi パッケージ（*「子製品」*と呼ぶこともある）をインストールまたは削除するカスタム アクションの一種です。



メモ *ネストインストールのカスタム アクションを使用する前に、次の制約に注意してください。*

- *ネストインストールにはユーザー インターフェイスが表示されません。*

- *通常、子製品がエンド ユーザーのシステム上のコントロール パネルの中の [プログラムの追加と削除] に表示されることはなく、親製品がアンインストールされても、自動的にアンインストールされることはありません。*

ネスト インストール カスタム アクションを作成する



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*



***重要・** Windows Installer ではネスト インストールの使用を避けるようにしてください。ネスト インストールが含まれている場合エンドユーザーが適切に処理することが困難であるため、時折アプリケーションの誤作動を引き起こします。マイクロソフト社では一般向けにリリースされる製品のインストールについて、ネスト インストール およびネスト インストール カスタム アクションの使用を避けることを推奨しています。詳細については、Windows Installer ヘルプ ライブラリの「Concurrent Installations」を参照してください。*



タスク ***ウィザードでネストインストールカスタム アクションを作成するには、次の手順に従ってください。***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、**[カスタム アクション ウィザード]** をクリックします。**カスタム アクション ウィザード**が開きます。
3. **[基本情報]** パネルで、カスタム アクションの名前とコメントを指定、**[次へ]** をクリックして次のステップに進んでください。
4. **[アクション タイプ]** パネルの **[タイプ]** リストで、**[別の .msi パッケージを起動]** を選択します。**[場所]** リストで、**[メインのセットアップに含める]** を選択します。
5. **[アクション パラメーター]** パネルで、起動する .msi ファイルの場所を参照します。(便宜上、子製品のすべてのファイルは .msi パッケージに圧縮されていると仮定します。)**[ターゲット]** ボックスを使って、子インストールに渡すコマンドライン スイッチを指定することができます。たとえば、デフォルト設定と共に子インストールのすべての機能をインストールし、また親インストールが利用したのと同じ ALLUSERS プロパティの値を使うときは次のように **[ターゲット]** ボックスに入力します。

   ARPSYSTEMCOMPONENT=1 ADDDEFAULT=ALL ALLUSERS=[ALLUSERS]

   同じ式を使って、親と同じように子でも INSTALLDIR と同じ値を使用することができます。
6. **[追加オプション]** パネルのデフォルトのオプションを選択し、ウィザードが完了するまで **[次へ]** をクリックします

ネストされたインストール カスタム アクションをシーケンスへ挿入する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*



重要 *・Windows Installer ではネスト インストールの使用を避けるようにしてください。ネスト インストールが含まれている場合エンドユーザーが適切に処理することが困難であるため、時折アプリケーションの誤作動を引き起こします。マイクロソフト社では一般向けにリリースされる製品のインストールについて、ネスト インストールおよびネスト インストール カスタム アクションの使用を避けることを推奨しています。詳細については、Windows Installer ヘルプ ライブラリの「Concurrent Installations」を参照してください。*

カスタム アクションが作成されたら、それをシーケンスに挿入する必要があります。



タスク ***ネスト インストールをシーケンスに挿入するには、以下の手順を実行します。***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[シーケンス]** エクスプローラーで、**[インストール]** シーケンスの下にある **[ユーザー インターフェイス]** アイテムを展開します。このシーケンスにスケジュールされているアクションとダイアログが一覧表示されます。
3. **CostFinalize** を右クリックし、**[挿入]** をクリックします。[アクションの挿入] ダイアログ ボックスが開きます。
4. 新しいカスタム アクションを選択します。
5. **[条件]** ボックスで、**Not Installed** と入力します。これで、ネスト インストールは親製品がインストールされる初回インストール時にのみ実行されることを示します。
6. **[OK]** をクリックします。

子製品を削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

重要 *・Windows Installer ではネスト インストールの使用を避けるようにしてください。ネスト インストールが含まれている場合エンドユーザーが適切に処理することが困難であるため、時折アプリケーションの誤作動を引き起こします。マイクロソフト社では一般向けにリリースされる製品のインストールについて、ネスト インストールおよびネスト インストール カスタム アクションの使用を避けることを推奨しています。詳細については、Windows Installer ヘルプ ライブラリの「Concurrent Installations」を参照してください。*

子製品は、親製品が削除されたときも自動的に削除されません。ただし、親製品を削除したときに一緒に自動的に削除されるような 2 番目のネストインストールカスタム アクションを作成することができます。

タスク ***カスタム アクションを作成するには以下の手順を実行します。***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、**[カスタム アクション ウィザード]** をクリックします。**カスタム アクション ウィザード**が開きます。
3. **[基本情報]** パネルで、カスタム アクションの名前とコメントを指定、**[次へ]** をクリックして次のステップに進んでください。
4. **[アクション タイプ]** パネルの **[タイプ]** リストで、**[別の .msi パッケージを起動]** を選択します。**[場所]** リストで、**[アドバタイズされるアプリケーション、または既にインストールされているアプリケーション]** を選択します。
5. **[アクションのパラメーター]**パネルで、削除する .msi ファイルの場所を参照します。**[ターゲット]** ボックスでは、デフォルトのプロパティをそのまま残します。

   ALLUSERS=[ALLUSERS] REMOVE=ALL
6. **[追加オプション]** パネルのデフォルトのオプションを選択し、ウィザードが完了するまで **[次へ]** をクリックします

タスク ***カスタム アクションを配列するには、以下の手順を実行します。***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[シーケンス]** エクスプローラーで、**[インストール]** シーケンスの下にある **[実行]** アイテムを展開します。このシーケンスにスケジュールされているアクションとダイアログが一覧表示されます。
3. **InstallValidate** を右クリックし、**[挿入]** をクリックします。[アクションの挿入] ダイアログ ボックスが開きます。
4. 新しいカスタム アクションを選択します。
5. **[条件]** ボックスで、次のように入力します。

   REMOVE="ALL"
6. **[OK]** をクリックします。

このカスタム アクションによって、親製品がアンインストールされたときに子製品がアンインストールされます。

メモ- *[別の .msi パッケージの起動] するタイプのネスト インストール カスタム アクションを、[場所] の設定を **[ソース メディアに保存]** にして作成する場合、次の点に注意してください。*

- *子インストーラーは、Setup.exe インストールランチャの中に圧縮できません。*
- *子インストールのファイルが子の .msi データベース内で圧縮されていない場合、子インストールのファイルを親インストールのリリースフォルダーの Disk1 フォルダーに手動でコピーする必要があります。*

インストールで MessageBoxA を呼び出す

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***カスタム アクションで Windows API 関数 MessageBoxA() を呼び出してメッセージボックスを表示するには、以下の手順を実行します。***

パート A: カスタム アクション ウィザードを起動する

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、**[カスタム アクション ウィザード]** をクリックします。

パート B: カスタム アクションを開始する

1. **[基本情報]** パネルで、カスタム アクションの名前に**CA_Example** と入力します。**[次へ]** をクリックします。
2. **[アクションの種類]** パネルの [種類] リスト ボックスで、**[標準ダイナミック リンク ライブラリの関数を呼び出す]** を選択します。
3. **MessageBoxA** は、サポートされているすべての Windows プラットフォームに存在する User32.dll にエクスポートされるので、**[保存場所]** リストから **[インストール先のマシンの検索パス]** を選択します。続行するには **[次へ]** をクリックしてください。

パート C: 関数を定義する

MessageBoxA() の構文は次のとおりです。

```
int MessageBoxA (hWnd, lpText, lpCaption, uType);
```

この情報は、[関数定義] パネルで以下の手順で指定します。

1. **[名前]** ボックスで、**MessageBoxA** と入力します。
2. [**引数**] ボックスの最後の行をクリックして、最初のパラメーターを指定します。次のリストと同じ値になるよう、各パラメーターのフィールドを編集します。

   テーブル 15 ・ 引数ボックスのパラメーター

Type	ソース	値
HWND	定数	MsiWindowHandle
STRING	in プロパティ	MESSAGEPROP
STRING	in プロパティ	CAPTIONPROP
NUMBER	定数	1

   HWND タイプの **[値]** リストで、**MsiWindowHandle** を選択します。HWND タイプをこの値に設定すると、メッセージ ボックスが、インストール ウィンドウの手前に表示されます。MESSAGEPROP プロパティおよび CAPTIONPROP プロパティは、一覧からは使用できません。これらの名前は、**[値]** リストで入力すると、プロパティ マネージャーに追加されます。
3. **[戻り値の型]** リストで、**[void]** を選択します。(**MessageBoxA()** は数字を戻しますが、プロパティのその値のチェックについては、ここでは説明しません)。
4. **[次へ]** をクリックして **[アクションのパラメーター]** パネルに進みます。

パート D: MessageBoxA のソースを指定する

カスタム アクションウィザードで次に重要な設定は、[アクションのパラメーター] パネルの "ソース" フィールドです。MessageBoxA は User32.dll にあります。[参照] ボタンをクリックして、Windows システム フォルダーにある User32.dll を探します。

ウィザードの次のパネルでは、[次へ] をクリックしてデフォルト値をそのまま使用できます。続いて [概要] パネルで [完了] をクリックしてウィザードを閉じ、CA_Example をプロジェクトに追加できます。

パート E: プロパティの値を初期化する

MessageBoxA() の文字列パラメーターについては、プロパティは、上記手続きで説明説明されているように、提供されています。次に、これらのプロパティに値を入れます。

1. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックすると、製品のインストーラーのプロパティが表示されます。**カスタム アクション ウィザード**でこれらのプロパティを参照した結果として、MESSAGEPROP プロパティおよび CAPTIONPROP プロパティが作成されました。
2. **MESSAGEPROP** へ移動し、**Value** 列をクリックして値を入力します。値に「**このテキストが見えますか？** 」と入力します。
3. CAPTIONPROP プロパティを [**カスタム アクションの例**] に設定します。

パート F: カスタム アクションをシーケンスへ挿入する

カスタム アクションを実行するには、カスタム アクションをシーケンスに配置するか、またはカスタム アクションをダイアログのコントロール イベントの結果にする必要があります。

次の手順に従って、インストール プロジェクトの [インストール] シーケンスに CA_Example を挿入してください。

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[シーケンス]** エクスプローラーで、**[インストール]** シーケンスの下にある **[ユーザー インターフェイス]** アイテムを展開します。このシーケンスにスケジュールされているアクションとダイアログが一覧表示されます。
3. **MigrateFeatureStates** アクション (InstallWelcome ダイアログの直前のアクション) を右クリックして、**[挿入]** をクリックします。[**アクションの挿入**] ダイアログ ボックスが開きます。
4. ダイアログ ボックスの上にあるリストから、[**カスタム アクション**] (インストール プロジェクト) を選択します
5. カスタム アクションのリストから [**CA_Example**] を選択します。
6. **[OK]** をクリックします。

MigrateFeatureStates のすぐ後の [インストール] シーケンスに CA_Example が追加されます。

パート G: MessageBoxA の呼び出しをテストする

1. リリースをビルドします。(プロジェクトにコンポーネントやファイルを 1 つも追加していないと、出力ウィンドウでビルド エラーが表示されることがあります。)
2. **[ビルド]** メニューで **[実行]** をクリックします。

InstallShield がインストールを実行します。[ようこそ] ダイアログが表示されると、[カスタム アクションの例] というメッセージボックスが表示されます。

ターゲット システム上のファイルを検索する

[システム検索] ビュー

[システム検索] ビュー は、アプリケーションをインストールする前に特定のフォルダー、レジストリ、またはターゲット システム上の .ini ファイルを検索する機能を提供します。

ダイレクト エディター: 基本の MSI プロジェクト

Windows Installer は、ターゲット システム上のファイル検索の命令に、Signature、AppSearch および “locator” テーブルのレコードを使用します。Signature テーブルには、検出するファイルの情報を指定し、AppSearch テーブルには、検出された場合のファイル位置を完全パスで収めるプロパティを指定します。

たとえば、**FindMe.exe** というファイルを検索する場合は、ダイレクトエディターを使って Signature テーブルに次のレコードを追加します。

テーブル 16 ・ Signature テーブルのサンプル データ

テーブルの列	サンプル データ
Signature	findme_sig
FileName	FindMe.exe

Signature テーブルのその他のフィールドでは、オプションのバージョン、サイズ、データおよび言語情報を指定することができます。

AppSearch テーブルには、次のレコードを追加します。

テーブル 17 ・ AppSearch テーブルのサンプル データ

テーブルの列	サンプル データ	説明
プロパティ	LOCATION_OF_FINDME	パブリック プロパティにする必要があります。
Signature_	findme_sig	Signature テーブルで使用したのと同じ名前。

Windows Installer がファイルの検索を始める場所を指定することができる 4 つのロケータ テーブル、CompLocator、RegLocator、IniLocator、および DrLocator があります。特定のディレクトリ内のファイル検索には、DrLocator テーブルを使います。たとえば、ユーザーの Program Files ディレクトリ内にある **FindMe.exe** を検索する場合、DrLocator テーブルに次のレコードを追加します。

テーブル 18 ・ DrLocator テーブルのサンプル データ

テーブルの列	サンプル データ
Signature	findme_sig
Parent	
パス	[ProgramFilesFolder]
Depth	2

AppSearch アクションの実行後、**FindMe.exe** が見つかった場合は、パブリック プロパティ LOCATION_OF_FINDME にエンド ユーザーのシステム上でのこのファイルへの完全パスが収められ、ファイルが見つからなかった場合、このプロパティは未定義になります。

InstallScript: InstallScript MSI プロジェクト

FindFile および **FindAllFiles** 関数を使用すると、ターゲット システム上に存在するファイルを検索できます。たとえば、ユーザーの Program Files フォルダーから **FindMe.exe** というファイルを検索する場合、**OnAppSearch** イベント ハンドラー関数を次のように実装します。

```
function OnAppSearch()
    STRING svFoundFile;
begin
    FindAllFiles(PROGRAMFILES, "FindMe.exe", svFoundFile, RESET);
    MessageBox("FindMe.exe が検出されました: " + svFoundFile, INFORMATION);
end;
```

インストールが完了した後にアプリケーションを起動する

LaunchApplication 関数は、実行可能ファイルを起動します。たとえば、次のコードは、エンドユーザーがプログラムウインドウを閉じたとき、制御をスクリプトに戻して、エンドユーザーの INSTALLDIR フォルダーにある **Program.exe** のコピーを起動します。

```
LaunchApplication( INSTALLDIR ^ "Program.exe", "", "", SW_NORMAL,"",  LAAW_OPTION_WAIT );
```



プロジェクト: *InstallScript プロジェクトは INSTALLDIR ではなく TARGETDIR を利用します。*

InstallScript カスタム アクション内部からインストールを終了する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript カスタム アクションを含む基本の MSI*
- *InstallScript MSI*
- *InstallScript カスタム アクションを含むマージ モジュール*

ERROR_INSTALL_FAILURE という値を返す InstallScript エントリポイント関数によって、インストールが終了します。

スクリプトを通して ODBC プロパティを変更する

ODBC プロパティ (DBQ、 SystemDB など) をスクリプトを使用して変更する場合、これらのオプションを構成して、インストーラーのプロパティを使用します。そうすると、スクリプトからインストーラー プロパティを実行時に設定できます。たとえば、InstallShield インターフェイスで SystemDB を **[SYSTEM_DB_DIR]¥MyDatabase** に設定します。そうすると、スクリプトから **MsiSetProperty** を呼び出して、SYSTEM_DB_DIR の値を設定することができます。



ヒント: *OnMoving イベントの前に、できれば OnFirstUIBefore でプロパティ値を設定してください。その後になると、インストーラーは既に内部スクリプト情報を作成済みで、これを変更することはできないため手遅れになります。*

- *プロパティ名にはすべて大文字を利用します。*
- *Directory テーブルに SYSTEM_DB_DIR が存在する場合、ODBC 属性設定作成時に右かっこ (]) の後の円記号 (¥) を省略します。プロパティマネージャーで定義した場合、円記号が必ず必要です。*

VBScript カスタム アクションで INSTALLDIR プロパティを使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

VBScript カスタム アクションで INSTALLDIR プロパティを使用する必要がある場合、Session オブジェクトの Property プロパティを使用します。このオブジェクトはすべての VBScript カスタム アクションでアクセス可能です。

サンプルコードは次の通りです。

```
szInstallDir = Session.Property("INSTALLDIR")
```

詳細は、Windows Installer ヘルプライブラリの「Session オブジェクト」を参照してください。

アクション テキストを使う



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

エンド ユーザーに進行状況を伝えるために、インストールは一般的に、実行中の処理を説明する進行状況ダイアログで説明テキストを表示します。通常、インストールのステータスを表示する進行状況バーも同時に表示されます。標準アクションおよびカスタム アクションが発生すると、そのアクションについてのメッセージが進行状況ダイアログに表示されます。この機能は、実行するのに時間がかかるアクションで、特に役立ちます。インストールのログ記録が作成される場合、同じアクション テキストが記録されます。

また、アクションは処理の必要があるアクション データを Windows Installer に送ります。データが処理された後、進行状況ダイアログにその表示が可能となります。

たとえば、InstallFiles アクションを実行中、デフォルトで進行状況ダイアログにはアクション テキスト、「新しいファイルをコピーしています」が表示されます。このアクションテキストは、インストールのログ ファイルにも記録されます。アクションが発生するときに、進行中の処理についての詳細情報を提供するには、進行状況ダイアログにアクションの詳細を表示するコントロールを追加して、そのコントロールが ActionData イベントをサブスクライブするようにします。InstallFiles アクションの場合、アクションの詳細として、ターゲット システム上にインストールされる各ファイルの名前、ディレクトリ、およびサイズが含まれます。



プロジェクト: *InstallScript MSI インストールでは、状態ダイアログにアクション データを表示できません。ただし、そのテンプレートはインストールのログ ファイルに記録されます。*

アクションの説明とアクション データのテンプレートを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

InstallShield には、標準アクションとビルトイン InstallShield カスタム アクション用のデフォルトのアクションの説明が含まれています。また、適切な場合、これらのアクションの多くには、デフォルト テンプレートも用意されています。テンプレートは、アクション データ使用するフォーマットが示されています。

プロジェクトに含まれる標準アクションまたはビルトイン InstallShield カスタム アクションのデフォルトの説明またはテンプレートを変更しなくてはならない場合があります。また、独自のカスタム アクションのアクション テキストおよびアクション データを指定する必要がある場合があります。



タスク ***プロジェクトに含まれる説明およびアクション データ テンプレートを指定するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[アクション テキスト]** エクスプローラーを右クリックしてから、**[新規]** をクリックします。新しいアクション テキスト アイテムが追加されます。
3. アクション テキスト アイテムの名前として、アクションの名前を入力します。



重要: *[アクション テキスト] エクスプローラーの下にあるアクション テキスト項目の名前は、プロジェクトに含まれる標準およびカスタム アクションの名前と一致します。カスタム アクションの名前を変更する場合は、そのアクション テキスト項目の名前も変更しなくてはなりません。そうしないと、実行時にアクション テキストが表示されないか、インストールのログ ファイルに記録されません。ただし、プロジェクトに含まれるすべてのアクションに対してアクション テキストを作成する必要はありません。*

4. 右側のペインで、説明とテンプレートを指定します。詳細については、「アクション テキストの設定」を参照してください。



ヒント: *[アクション テキスト] エクスプローラーの下に既にリストされているアクションの説明またはテンプレートを編集するには、そのアクションを選択してから、右側のペインで適切な値を指定します。*

進行状況ダイアログにアクションの説明を表示する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

InstallScript MSI インストールでは、進行状況ダイアログ (STATUSEX ダイアログ) が Enable (STATUSEX) の呼び出しで有効化されている場合、自動的にこのダイアログにアクション テキストが表示されます。このダイアログを編集することはできません。

デフォルトで、インストールに含まれるアクションに説明が指定されている場合、その説明は進行状況ダイアログの進行状況バーの上に表示されます。このため、プロジェクトに含まれるこのダイアログに何らかの変更を加えない限り、次のステップを行う必要はありません。



タスク ***進行状況ダイアログにアクションの説明を表示するには、以下の手順に従います:***

1. **SetupProgress** ダイアログで、アクションの説明を表示するコントロールを追加するには、以下の手順に従います:
   a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
   b. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、**SetupProgress** ダイアログを展開します。
   c. **SetupProgress** ダイアログの下にある言語をクリックします。中央のペインにあるダイアログ エディターに、選択した言語のダイアログが表示されます。
   d. **[コントロール]** ツールバーで、**[テキスト領域]** コントロールをクリックします。
   e. ダイアログ上でアクションの説明を表示する位置に長方形を描きます。
2. コントロールを選択した状態で、右のペインでその設定を構成します:
   a. **(Name)** 設定に、次のように入力します:

      ActionText

   b. **Text** 設定から、値を削除します。
3. ActionText イベントのサブスクリプションを新しいコントロールに追加します:

a. **SetupProgress** ダイアログの下にある **[ダイアログ]** エクスプローラーで、**[動作]** アイテムをクリックします。中央のペインにあるグリッドに SetupProgress ダイアログのすべてのコントロールが表示されます。

b. 作成した **ActionText** コントロールを選択してから、右下のペインにある **[サブスクリプション]** タブをクリックします。

c. 右上のペインで、**Event** フィールドに **ActionText** が選択されたレコードを追加します。**Attribute** フィールドには、**Text** と入力します。

実行時に、インストールが完全または簡易ユーザー インターフェイスで実行されたとき、進行状況ダイアログに、各アクションが発生するたびにその説明が表示されます。あるアクションに説明が定義されていない場合、そのアクションが起動されたときに進行状況ダイアログで説明は表示されません。

進行状況ダイアログにアクション データを表示する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

InstallShield MSI インストールでは、進行状況ダイアログ (STATUSEX ダイアログ) で、アクション データを表示することはできません。ただし、InstallScript で Enable (INDVFILESTATUS) を呼び出すことで、同じような結果を得ることができます。INDVFILESTATUS 定数を **Enable** *関数を使って呼び出すと、***FeatureMoveData**、**CopyFile**、*または* **XCopyFile** *が呼び出されたときに、進行状況インジケーターが有効化されている場合、STATUSEX ダイアログに転送中の各ファイルの完全修飾ファイル名が表示されます。InstallScript MSI プロジェクトで、アクションにテンプレートを指定した場合、そのテンプレートがインストールのログ ファイルに書き込まれます。*

実行時にアクションが起動されるとき、進行中の処理についての詳細情報を提供するには、進行状況ダイアログにアクションの詳細を表示するコントロールを追加して、そのコントロールが ActionData イベントをサブスクライブするようにします。たとえば、InstallFiles アクションの実行中にアクションの詳細を表示するコントロールには、ターゲット システム上にインストールされている各ファイルの名前、ディレクトリ、およびサイズといった内容が含まれます。



タスク ***進行状況ダイアログにアクション データを表示するには、以下の手順に従います:***

1. **SetupProgress** ダイアログで、アクションの詳細を表示するコントロールを追加するには、以下の手順に従います:

   a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。

   b. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、**SetupProgress** ダイアログを展開します。

   c. **SetupProgress** ダイアログの下にある言語をクリックします。中央のペインにあるダイアログ エディターに、選択した言語のダイアログが表示されます。

   d. **[コントロール]** ツールバーで、**[テキスト領域]** コントロールをクリックします。

e. ダイアログ上でアクションの説明を表示する位置に長方形を描きます。

2. コントロールを選択した状態で、右のペインでその設定を構成します:

   a. **(Name)** 設定に、次のように入力します:

      ActionData

   b. **Text** 設定から、値を削除します。

3. ActionData イベントのサブスクリプションを新しいコントロールに追加します:

   a. **SetupProgress** ダイアログの下にある **[ダイアログ]** エクスプローラーで、**[動作]** アイテムをクリックします。中央のペインにあるグリッドに SetupProgress ダイアログのすべてのコントロールが表示されます。

   b. 作成した **ActionData** コントロールを選択してから、右下のペインにある **[サブスクリプション]** タブをクリックします。

   c. 右上のペインで、**Event** フィールドに **ActionData** が選択されたレコードを追加します。**Attribute** フィールドには、**Text** と入力します。

実行時に、インストールが完全ユーザー インターフェイスで実行されたとき、進行状況ダイアログに、各アクションが発生するたびにその詳細が表示されます。あるアクションにテンプレートが定義されていない場合、そのアクションが起動されたときに進行状況ダイアログで詳細が表示されません。

アクションの説明またはアクション データのテンプレートを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*



タスク ***プロジェクトに含まれる説明またはアクション データ テンプレートを削除するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[アクション テキスト]** エクスプローラーで、その説明またはテンプレートを削除するアクションを選択します。
3. **Description** 設定または **Template** 設定から、値を削除します。



メモ: *アクションの説明またはアクション テンプレートのどちらも使用しない場合、プロジェクトからアクション テキスト レコードを削除できます。その場合、[アクション テキスト] エクスプローラーで、削除するアクション テキスト アイテムを右クリックしてから、[削除] を選択します。*

以下のアクションのアクション テキストを削除することはできません:

- *GenerateScript*

- *Rollback*
- *RollbackCleanup*

さらに、これらの各アクションの "テンプレート" 設定の値は、常に **[1]** *とします。*

シーケンスを定義する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

インストールのシーケンスを定義することは、インストール パッケージ開発の重要な部分です。シーケンスは、Windows Installer で、インストール プロセスを制御する標準およびカスタム アクションが起動される順番を指定します。



プロジェクト: *基本の MSIおよびトランスフォーム プロジェクトでは、ダイアログが表示される順番もシーケンスで指定されます。*

InstallScript MSI プロジェクトでは、ユーザー インターフェイス ダイアログは、InstallScript コードで定義され、インストールのユーザー インターフェイス部分は InstallScript エンジンによって制御されます。

InstallShield の [カスタム アクションとシーケンス] ビューで、プロジェクトのシーケンスを定義することができます。プロジェクトのアクションとダイアログは、このビューの [シーケンス] エクスプローラーで、該当のシーケンスで起動されるタイミングにしたがって時系列で表示されます。各アクションとダイアログは、シーケンス上の番号が付与され、Windows Installer は、シーケンスを小さい番号から順に実行されます。

プロジェクトに追加したすべてのカスタム アクションとダイアログに手動で数値を割り当てることなく、[カスタム アクションとシーケンス] ビューを使用して、アクションまたはダイアログをシーケンスに挿入するか、シーケンス タイムラインを編集することができます。シーケンスについての詳しい情報についてはヘルプの同セクションを参照してください。



プロジェクト: *カスタム アクションまたはカスタム ダイアログをマージ モジュールで作成する場合、まずモジュールをインストール プロジェクトにインポートし、[カスタム アクションとシーケンス] ビューを使用してそれをシーケンスに追加する必要があります。*

[インストール] シーケンス



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

- *MSI データベース*
- *トランスフォーム*

[インストール] シーケンスは、エンドユーザーが新しい .msi ファイルをダブル クリックする操作などの、デフォルトのインストール モードでインストールを起動する場合に実行される一連のアクションです。実行されるアクションは、以下の 2 種類に分けられます。

- ユーザー インターフェイス
- 実行



プロジェクト: *以下に説明する [ユーザー インターフェイス] ダイアログは基本の MSI プロジェクト用のものです。InstallScript MSI プロジェクト については、InstallScript コードがインストールのユーザー インターフェイスを実行します。*

InstallScript MSI プロジェクトは、自動的に ISVerifyScriptingRuntime カスタム アクションを含みます。詳細については、「ISVerifyScriptingRuntime」を参照してください。

ユーザー インターフェイス

[ユーザー インターフェイス] シーケンスには、完全なユーザー インターフェイスをサポートするのに必要なアクションおよびダイアログがすべて含まれています。このシーケンスは、インストールをサイレント モードで実行する場合にはスキップされます。

標準のアクションに関する技術的な詳細は、Windows Installer ヘルプ ライブラリの「Standard Actions Reference」を参照してください。

テーブル 19 ・ [インストール] シーケンスのユーザー インターフェイス アクションとダイアログ

アクションまたはダイアログの名前	イベントの種類	説明
SetupCompleteError	ダイアログ	このダイアログは、インストールが致命的なエラーのために終了した場合に表示されます。
SetupInterrupted	ダイアログ	このダイアログは、ユーザーがインストールを終了した場合に表示されます。
SetupCompleteSuccess	ダイアログ	このダイアログは、インストールが正常に終了した場合に表示されます。
ISSetupFilesExtract	カスタム アクション	このカスタム アクションは、[サポート ファイル] ビューに追加した任意のファイルを抽出します。詳細については、「サポート ファイルを使用する」を参照してください。
ISSetAllUsers	カスタム アクション	ISSetAllUsers カスタム アクションは、Upgrade テーブル に 1 つ以上の記録があると、[ユーザー インターフェイス] シーケンスと [実行] インストール シーケンスの両方に挿入されます。

テーブル 19 • [インストール] シーケンスのユーザー インターフェイス アクションとダイアログ (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
AppSearch	標準アクション	このアクションは、ターゲット システム上のファイルを識別および検索するのに使用できます。このアクションに必要な Signature テーブルは、ダイレクト エディターで使用可能です。このアクションの使用についての詳細は、ナレッジベース記事 Q103147 を参照してください。
LaunchConditions	標準アクション	インストールを続行するために True を返す必要のある条件を評価します。条件のどれかが失敗した場合、エラーメッセージが表示され、インストールは終了します。起動条件は、[一般情報] ビューで編集できます。
SetupInitialization	ダイアログ	このダイアログは、セットアップの開始準備中に表示されます。このダイアログのデフォルトのテキストは、"インストールの準備をしています..." です。
FindRelatedProducts	標準アクション	このアクションを実行すると、インストーラーは各々のインストールした製品のアップグレード コードをパッケージの Upgrade テーブルにあるリストと比較します。コードが一致した場合、インストールしたパッケージの製品コードが Upgrade テーブル の ActionProperty 列に追加されます。
CCPSearch	標準アクション	CCPSearch アクションを使用すると、アップグレードできる製品がエンド ユーザーのシステムにあるかどうかを確認することができます。このアクションに必要な Signature テーブルは、ダイレクト エディターで使用可能です。
RMCCPSearch	標準アクション	RMCCPSearch アクションを使用すると、競合アップグレードできる製品がエンド ユーザーのシステムにあるかどうかを確認することができます。このアクションに必要な Signature テーブルは、ダイレクト エディターで使用可能です。
ValidateProductID	標準アクション	製品 ID の検証アクションは、ProductID プロパティを完全な製品識別子に設定する場合に使用します。この検証により、ユーザーに製品 ID の入力を要求し、違法な使用から製品を保護することができます。
CostInitialize	標準アクション	このアクションは、インストールの現在の設定に必要なディスク容量を判断する最初のステップです。コスティングを完了するには、CostInitialize アクションと CostFinalize アクションを組み合わせて使用する必要があります。

テーブル 19 • [インストール] シーケンスのユーザー インターフェイス アクションとダイアログ (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
FileCost	標準アクション	FileCost アクションは、インストールの現在の設定に必要なディスク容量を判断します。より新しいバージョンで上書きするファイルがあるかどうか、上書きがディスク容量にどのように影響するかを確認します。コスティングを完了するには、CostInitialize アクションと CostFinalize アクションを組み合わせて使用する必要があります。
IsolateComponents	標準アクション	このアクションは、コンポーネント (通常は DLL ファイル) を孤立した場所にインストールし、特定のアプリケーションだけがそのコンポーネントを使用するようにします。
setUserProfileNT	カスタム アクション	このカスタム アクションは USERPROFILE ディレクトリ識別子を初期化します。
SetAllUsersProfileNT	カスタム アクション	このカスタム アクションは ALLUSERSPROFILE ディレクトリ識別子を初期化します。
setAllUsersProfile2K	カスタム アクション	このカスタム アクションは ALLUSERSPROFILE ディレクトリ識別子を初期化します。
ResolveSource	標準アクション	ソースの位置を検出し、SourceDir プロパティを設定します。
CostFinalize	標準アクション	ディスク容量の計算アクションは、現在の構成のインストールで必要な合計ディスク容量を判断します。また、すべてのターゲットディレクトリが書き込み可能であることを確認します。ディスク容量の計算アクションの前に、ディスク容量計算の初期化アクションおよびダイナミック ディスク容量計算の初期化アクションを呼び出す必要があります。呼び出さなかった場合、アクションは失敗します。
SetARPReadme	カスタム アクション	SetARPReadme カスタム アクションは、[一般情報] ビューの "Readme" 設定で使用されるディレクトリ識別子を解決します。このカスタム アクションが必要な理由は、ARPREADME が Windows Installer プロパティであるため、自動的にフォーマットされないためです。
MigrateFeatureStates	標準アクション	このアクションは、アプリケーションのアップグレード中に使用されます。元のインストールの機能の状態がターゲットマシンから読み込まれ、その後アップグレードした機能に適用されます。

テーブル 19 ・[インストール] シーケンスのユーザー インターフェイス アクションとダイアログ (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
PatchWelcome	ダイアログ	パッチパッケージが完全ユーザー インターフェイスに適用されると、PatchWelcome ダイアログが表示されます。ここには、正しいオプションを使って REINSTALL および REINSTALLMODE を設定するためのコントロール イベントが含まれています。ただし、ユーザー インターフェイスが抑制されている場合、プロパティをコマンドラインで設定する必要があります。
InstallWelcome	ダイアログ	InstallWelcom ダイアログを使用すると、InstallScript MSI インストールが実行されたときに [ようこそ] パネルが表示されます。
SetupResume	ダイアログ	このダイアログは、それまでにキャンセルされたインストールが再開された場合に表示されます。
MaintenanceWelcome	ダイアログ	このダイアログは、エンド ユーザーがプログラムのインストール機能の変更、プログラムの削除、プログラムの再インストール、2 回目のインストールの実行、[プログラムの追加と削除] パネルでの現在の製品の選択を試みたときに表示されます。
SetupProgress	ダイアログ	このダイアログはインストールの進行状況を表示します。
ExecuteAction	標準アクション	このアクションは EXECUTEACTION プロパティを参照し、最初に呼び出すトップレベルのアクションを判断してそのアクションを呼び出します。トップレベルアクションには、INSTALL、ADVERTISE、ADMIN などのアクションがあります。通常、このアクションによって [インストール]–[実行] シーケンスが開始されます。
ISSetupFilesCleanup	カスタム アクション	このカスタム アクションは、[サポート ファイル] ビューでファイルを追加したとき表示されます。詳細については、「サポート ファイルを使用する」を参照してください。

実行

[実行] シーケンスには、マシンの状態を変更し、正常に機能するためにユーザー インターフェイスに依存しないすべてのアクションが含まれます。これらのアクションには、ファイル転送、コンポーネントおよび機能のパブリッシュ、COM サーバーの登録などがあります。[実行] シーケンスの多くは、シーケンスに挿入されたカスタム アクション以外、テストモードでインストールを実行したときにスキップされます。

テーブル 20 ・ [インストール] シーケンスでアクションとダイアログを実行する

アクションまたはダイアログの名前	イベントの種類	説明
ISSetupFilesExtract	カスタム アクション	このカスタム アクションは、[サポート ファイル] ビューに追加した任意のファイルを抽出します。詳細については、「サポート ファイルを使用する」を参照してください。
ISSetAllUsers	カスタム アクション	ISSetAllUsers カスタム アクションは、Upgrade テーブル に 1 つ以上の記録があると、[ユーザー インターフェイス] シーケンスと [実行] インストール シーケンスの両方に挿入されます。
AppSearch	標準アクション	このアクションは製品の以前のバージョンを識別、検索するのに使用できます。このアクションに必要な Signature テーブルは、ダイレクト エディターで使用可能です。このアクションの使用についての詳細は、ナレッジベース記事 Q103147 を参照してください。
LaunchConditions	標準アクション	インストールを続行するために True を返す必要のある条件を評価します。条件が満たされない場合、インストールはエラーメッセージを表示してインストールを終了します。起動条件は、[一般情報] ビューで編集できます。
FindRelatedProducts	標準アクション	このアクションを実行すると、インストーラーは各々のインストール済み製品のアップグレード コードをパッケージの Upgrade テーブル (ダイレクト エディター でサポートされている) にあるリストと比較します。コードが一致した場合、インストールしたパッケージの製品コードが Upgrade テーブル の ActionProperty 列に追加されます。
CCPSearch	標準アクション	CCPSearch アクションを使用すると、アップグレードできる製品がエンド ユーザーのシステムにあるかどうかを確認することができます。このアクションに必要な Signature テーブルは、ダイレクト エディターで使用可能です。
RMCCPSearch	標準アクション	RMCCPSearch アクションを使用すると、競合アップグレードできる製品がエンド ユーザーのシステムにあるかどうかを確認することができます。このアクションに必要な Signature テーブルは、ダイレクト エディターで使用可能です。

テーブル 20 ・ [インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
ValidateProductID	標準アクション	ValidateProductID アクションは、ProductID プロパティを完全な製品識別子に設定する場合に使用します。検証により、ユーザーに製品 ID の入力を要求し、違法な使用から製品を保護することができます。
CostInitialize	標準アクション	このアクションは、現在のインストール構成に必要なディスク容量を判断する最初のステップです。
FileCost	標準アクション	現在のインストール構成に必要なディスク容量を判断します。新規バージョンで上書きするファイルがあるかどうか、上書きがディスク容量にどのように影響するかを確認します。
IsolateComponents	標準アクション	このアクションは、コンポーネント (通常は DLL ファイル) を孤立した場所にインストールし、特定のアプリケーションだけがそのコンポーネントを使用するようにします。
CostFinalize	標準アクション	CostFinalize アクションは、現在のインストール構成に必要な合計ディスク容量を判断します。また、すべてのターゲットディレクトリが書き込み可能であることを確認します。
SetARPINSTALLLOCATION	カスタム アクション	SetARPINSTALLLOCATION カスタム アクションは、ARPINSTALLLOCATION プロパティをアプリケーションのプライマリ フォルダーの完全修飾パスに設定します。
SetODBCFolders	標準アクション	SetODBCFolders アクションは、ターゲット システムに既存の ODBC ドライバーがないか確認し、既存のドライバーの場所に新しい各ドライバーのターゲットディレクトリを設定します。
MigrateFeatureStatus	標準アクション	このアクションは、アプリケーションのアップグレード中に使用されます。元のインストールの機能の状態がターゲットマシンから読み込まれ、その後アップグレードした機能に適用されます。
InstallValidate	標準アクション	InstallValidate アクションは、現在のインストール構成に必要なディスク容量があるかどうかを判断します。
RemoveExistingProducts	標準アクション	RemoveExistingProducts アクションによって、コードが OLDPRODUCTS に表示される製品のサイレントアンインストールが実行されます。

テーブル 20 • [インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
InstallInitialize	標準アクション	InstallInitialize アクションは、エンド ユーザーのシステムに変更を加えるアクションの開始を知らせます。
AllocateRegistrySpace	標準アクション	インストーラーはこのアクションを実行する時に、システムに少なくとも AVAILABLEFREEREG プロパティで指定された使用可能なレジストリ空き領域があることを確認します。
ProcessComponents	標準アクション	ProcessComponents アクションは、コンポーネントの登録と登録解除を行います。このアクションはコンポーネントのキーパスおよびコンポーネントが持つその他のクライアントの登録、および登録解除も行います。
UnpublishComponents	標準アクション	アンインストール中に呼び出され、最初のインストールでパブリッシュされたコンポーネントを、他のアプリケーションでパブリッシュされたものであっても、パブリッシュ解除します。
MsiUnpublishAssemblies	標準アクション	MsiUnpublishAssemblies アクションは、オペレーティング システムのアセンブリのアンインストールを処理します。
UnpublishFeatures	標準アクション	UnpublishFeatures アクションは、インストールでパブリッシュされた機能への参照をすべて削除します。このようなリファレンスには、選択状態および機能コンポーネント マッピング情報を含むレジストリ エントリなどがあります。
StopServices	標準アクション	このアクションは、停止するように構成されている Windows サービスを停止します。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。
DeleteServices	標準アクション	このアクションは、削除するように構成されている Windows サービスを削除します。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。
UnregisterComPlus	標準アクション	このアクションは、COM+ アプリケーションを登録解除します。
SelfUnregModules	標準アクション	このアクションは、SelfReg テーブルのデータによって登録されたファイルを登録解除します。

テーブル 20 • [インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
UnregisterTypeLibraries	標準アクション	アンインストール中、アンインストールするようマークされた TypeLib テーブル内のすべてのファイルを登録解除します。このテーブルは、[COM 登録] 詳細設定で新規 TypeLib を作成した際に埋められます。
RemoveODBC	標準アクション	アンインストール中、ODBCDataSourceテーブル、ODBCTranslator テーブル、および ODBCDriver テーブルを参照し、アンインストールでどの ODBC リソースを削除するかを判断します。削除するようマークされたリソースは、アンインストールされます。
UnregisterFonts	標準アクション	アンインストール用に設定されたすべてのフォントに関する情報を登録解除します。
RemoveRegistryValues	標準アクション	RemoveRegistryValues アクションは、次の条件がすべて満たされている場合、エンド ユーザーのレジストリから値を削除します。 • 値が Registry テーブルに入力されている場合。 • 値がアンインストールのためにマークされいる場合。 • レジストリ エントリが属するコンポーネントが、[ソースから実行]、または [ローカルにインストール] に設定されている場合。
UnregisterClassInfo	標準アクション	このアクションは、アンインストールされている機能に属する COM クラスのシステムレジストリ情報を削除します。
UnregisterExtensionInfo	標準アクション	アンインストール中にエンド ユーザーのシステムから、拡張子に関するすべての情報を登録解除します。
UnregisterProgIdInfo	標準アクション	アクションは [ファイルの種類] 詳細設定に作成された ProgIDs すべてを登録解除します。
UnregisterMIMEInfo	標準アクション	アンインストールされている現在の機能の MIME テーブルを照会し、サーバーの MIME 情報を登録解除します。この情報は、ファイルの種類 詳細設定で入力します。

テーブル 20 • [インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
RemoveIniValues	標準アクション	このアクションは IniFile テーブルで (または [INI ファイルの変更] ビューを使って) コンポーネントに関連付けられた .ini 情報のみを削除します。このアクションは、IniFile テーブルに情報が存在することを確認した後、RemoveIniFile テーブルにリストされているすべての .ini ファイルの関連コンポーネントがアンインストール用にマークされ、かつローカルにインストールされた、またはソースから実行するように設定された場合に、これらの .ini ファイルを削除します。また、RemoveIniValue アクションでは、WriteIniValues アクションで書かれたすべての .ini ファイルも、関連コンポーネントがアンインストール用にマークされている場合、削除されます。
RemoveShortcuts	標準アクション	アンインストール用にマークされた機能の、アドバタイズされたショートカットをすべて削除します。また、アンインストールするようマークされたコンポーネントの、アドバタイズされていないショートカットもすべて削除されます。
RemoveEnvironmentStrings	標準アクション	コンポーネントが削除されると、このアクションは、WriteEnvironmentStrings アクションによってインストール中や再インストール中に環境変数に加えられた変更を元に戻します[環境変数] ビューを使って、環境変数の変更を指定できます。
RemoveDuplicateFiles	標準アクション	DuplicateFiles アクションで作成されたファイルを削除します。このアクションを正常に完了するには、複製されたファイルに関連したコンポーネントを、アンインストールするようマークしておく必要があります。
RemoveFiles	標準アクション	InstallFiles アクションでインストールされたファイルが、ソースから実行するかローカルにインストールされ、関連したコンポーネントがアンインストールするようマークされている場合に、これらのファイルを削除します。
RemoveFolders	標準アクション	Remove Folders アクションは、アンインストールするようマークされ、ソースから実行されたコンポーネントに関連している空のフォルダーを登録解除します。
CreateFolders	標準アクション	ローカルでインストールするよう設定されたコンポーネント用に、空のフォルダーを作成します。作成された新規フォルダーはその後、関連するコンポーネント GUID (コンポーネントの"コンポーネント コード" プロパティ 内) で登録されます。

テーブル 20 • [インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
MoveFiles	標準アクション	このアクションにより、ターゲット システムに既に存在するファイルを移動またはコピーすることができます。このアクションで使用する MoveFiles テーブルは、ダイレクト エディターで使用可能です。
InstallFiles	標準アクション	InstallFiles アクションでは、選択されたコンポーネント機能がインストールするようマークされている場合、その機能のファイルすべてをターゲットマシンにコピーします。ローカルにインストールされるコンポーネントに関連したファイルのみが、ターゲットマシンにコピーされます。
PatchFiles	標準アクション	Patch テーブルを照会して、インストールされたファイルに使用できるパッチを判断します。これらのファイルに対してビットワイズパッチ処理が行われます。
DuplicateFiles	標準アクション	DuplicateFiles アクションは、InstallFiles アクションでインストールされた特定のファイルのコピーを作成します。これらのファイルは、元のファイルと同じディレクトリに別の名前でコピーするか、別のディレクトリに同じ名前でコピーすることができます。このアクションで使用する DuplicateFiles テーブルは、ダイレクト エディターで使用可能です。
BindImage	標準アクション	インポートした DLL ファイル関数の仮想アドレスを、BindImage テーブルの指定どおりにファイルのインポートアドレス テーブルに書き込みます。BindImage テーブルは、ダイレクト エディター で使用できます。
CreateShortcuts	標準アクション	このアクションは [ショートカット] ビューまたは [セットアップのデザイン] ビューのショートカット エクスプローラーで指定したショートカットを作成します。
RegisterClassInfo	標準アクション	このアクションは、[COM 登録] 詳細設定で指定した、またはコンポーネント ウィザードで抽出された、あるいはコンポーネントの "ビルド時に COM 抽出" 設定で指定されたすべての COM クラス情報を登録します。
RegisterExtensionInfo	標準アクション	RegisterExtensionInfo アクションは、[ファイルの種類] 詳細設定で指定したすべての拡張子を登録します。
RegisterProgIdInfo	標準アクション	このアクションは、詳細設定で指定し、インストールするようマークされたクラスサーバーまたは拡張サーバーにリンクした ProgID をすべて登録します。

テーブル 20 ・[インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
RegisterMIMEInfo	標準アクション	このアクションは、[ファイルの種類] 詳細設定で指定し、インストールするようマークされたクラスサーバーまたは拡張サーバーにリンクした MIME タイプをすべて登録します。
WriteRegistryValues	標準アクション	レジストリデータに関連したコンポーネントがインストールするようマークされ、ソースから実行するよう設定されているかローカルでインストールされている場合に、そのレジストリデータをターゲット システムに書き込みます。このレジストリ情報は、[レジストリ] ビューで作成したデータと同じです。
WriteIniValues	標準アクション	このアクションに関連するコンポーネントがローカルでインストールするかソースから実行するよう設定されている場合に、.ini ファイルに情報を書き込みます。このアクションおよびこれに対応するテーブルは、[INI ファイルの変更] ビューに表示されます。
WriteEnvironmentStrings	標準アクション	コンポーネントがインストールされると、このアクションは Environment テーブル、または [環境変数] ビューで指定されたシステムの環境変数を変更します。
RegisterFonts	標準アクション	インストールに含むフォントを登録します。多くの場合、これらのフォントは、コンポーネント ウィザードを使用してセットアップに含みます。
InstallODBC	標準アクション	[ODBC リソース] ビューで指定した ODBC リソースのドライバー、トランスレーター、およびデータソースすべてをインストールします。
RegisterTypeLibraries	標準アクション	このアクションは、[COM 登録] 詳細設定またはコンポーネント ウィザード を使用して、インストールに作成したライブラリを登録します。
SelfRegModules	標準アクション	このアクションは、SelfReg テーブルに表示された自己登録モジュールを登録します。このアクションは、デフォルトのユーザー権限で実行されます。
RegisterComPlus	標準アクション	このアクションは、COM+ アプリケーションを登録します。
InstallServices	標準アクション	このアクションは、インストールするように構成されている Windows サービスをインストールします。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。

テーブル 20 ・[インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
MsiConfigureServices	標準アクション	このアクションは、Windows サービスの拡張カスタマイズオプションを構成します。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。 ***メモ・*** *このアクションは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*
StartServices	標準アクション	このアクションは、開始するように構成されているすべてのサービスを開始します。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。
RegisterUser	標準アクション	プログラムのユーザーを識別するため、ユーザー情報を登録します。
RegisterProduct	標準アクション	製品をインストーラーで登録し、インストーラーデータベースをターゲットマシンに保存します。
PublishComponents	標準アクション	このアクションは、アドバタイズされた機能に関連するコンポーネントすべてをパブリッシュします。
PublishFeatures	標準アクション	各機能のインストール状態を登録します。状態には、なし、アドバタイズ済み、インストール済みがあります。機能がインストールされると、PublishFeatures アクションによって機能とコンポーネントの関係がレジストリに書き込まれます。
PublishProduct	標準アクション	製品がアドバタイズされている場合、その製品をパブリッシュします。
ScheduleReboot	標準アクション	ScheduleReboot アクションをアクション シーケンスに挿入すると、インストールの最後に、システム再起動を促すメッセージがエンドユーザーに表示されます。ScheduleReboot アクションは通常、シーケンスの終わりに配置されます。
InstallFinalize	標準アクション	このアクションは、最後に処理される手順です。
RemoveExistingProducts	標準アクション	このアクションは、Upgrade テーブルにリストされたすべての製品コードをループにして、これらの製品を削除します。

テーブル 20 - [インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
ISSetupFilesCleanup	カスタム アクション	このカスタム アクションは、[サポート ファイル] ビューでファイルを追加したとき表示されます。詳細については、「サポート ファイルを使用する」を参照してください。

[アドバタイズ] シーケンス



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[アドバタイズ] シーケンスは、MsiExec.exe の /j コマンドライン オプションを使ってエンドユーザーがインストール プログラムを起動したときに実行されるアクションのリストです。このシーケンスにビルトインされているアクションの説明を、以下の表に示します。

ユーザー インターフェイス

入力規則 ICE78 では、[アドバタイズ]–[ユーザー インターフェイス] シーケンスが空になっている必要があるとしています。

実行

[実行] シーケンスには、正しく機能するためにユーザー インターフェイスを使用しないすべてのアクションが含まれます。これらのアクションには、ファイル転送、コンポーネントおよび機能のパブリッシュ、COM サーバーの登録などがあります。

標準のアクションに関する技術的な詳細は、Windows Installer ヘルプ ライブラリの「Standard Actions Reference」を参照してください。

テーブル 21 - [アドバタイズ] シーケンスでアクションを実行する

アクション名	イベントの種類	説明
CostInitialize	標準アクション	インストールの現在の設定に必要なディスク容量を判断する最初のステップです。
CostFinalize	標準アクション	現在の設定でインストールに必要な合計ディスク容量を判断します。
InstallValidate	標準アクション	現在の構成に必要なディスク容量があるかどうかを判断します。

テーブル 21・[アドバタイズ] シーケンスでアクションを実行する (続き.)

アクション名	イベントの種類	説明
InstallInitialize	標準アクション	エンドユーザーのシステムに変更を加えるアクションの最初にマークを付けます。
CreateShortcuts	標準アクション	[ショートカット] ビューまたは [セットアップのデザイン] ビューで指定したショートカットを作成します。
RegisterClassInfo	標準アクション	[COM 登録] 詳細設定で指定したすべての COM クラス情報を登録します。
RegisterExtensionInfo	標準アクション	[ファイルの種類] 詳細設定で指定したすべての拡張子を登録します。
RegisterProgIdInfo	標準アクション	コンポーネントがアドバタイズされている場合、COM 登録詳細設定で定義したすべての ProgID を登録します。
RegisterMIMEInfo	標準アクション	コンポーネントがアドバタイズされている場合、[ファイルの種類] 詳細設定で定義したすべての MIME タイプを登録します。
RegisterTypeLibraries	標準アクション	COM 登録詳細設定またはコンポーネントウィザードを使用して、インストールに作成したタイプライブラリを登録します。
PublishComponents	標準アクション	アドバタイズされた機能に関連するコンポーネントすべてをパブリッシュします。
MsiPublishAssemblies	標準アクション	共通言語ランタイムアセンブリと Win32 アセンブリのアドバタイズを管理します。このアクションは グローバルアセンブリキャッシュにアドバタイズまたはインストールされている機能を持つアセンブリを決定し、特定のアプリケーションから孤立した場所にアドバタイズまたはインストールされている親コンポーネントを持つアセンブリを判別するように MsiAssembly テーブルに問い合わせます。
PublishFeatures	標準アクション	各機能のインストール状態を登録します。状態には、なし、アドバタイズ済み、インストール済みがあります。機能がインストールされると、このアクションによって機能とコンポーネントの関係がレジストリに書き込まれます。
PublishProduct	標準アクション	製品がアドバタイズされている場合、その製品をパブリッシュします。
InstallFinalize	標準アクション	インストールまたはアンインストールの最終段階。

[管理] シーケンス



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[管理] シーケンスは、/a コマンドライン オプションを使ってユーザーがセットアップ プログラムを起動したときに実行されるアクションのリストです。[管理] シーケンスに組み込まれているアクションおよびダイアログを、以下の表に示します。

ユーザー インターフェイス

[ユーザー インターフェイス] シーケンスには、完全なユーザー インターフェイスをサポートするのに必要なアクションおよびダイアログがすべて含まれています。このシーケンスは、インストールをサイレント モードで実行する場合にはスキップされます。



プロジェクト: *以下に説明するユーザー インターフェイスアクションは基本の MSI プロジェクト用のものです。InstallScript MSI プロジェクト については、InstallScript コードがインストールのユーザー インターフェイスを実行します。*

標準のアクションに関する技術的な詳細は、Windows Installer ヘルプ ライブラリの「Standard Actions Reference」を参照してください。

テーブル 22 · [管理] シーケンスのユーザー インターフェイス アクション

アクション名	イベントの種類	説明
SetupCompleteError	ダイアログ	このダイアログは、セットアップが致命的なエラーのために終了した場合に表示されます。
SetupInterrupted	ダイアログ	このダイアログは、エンドユーザーによって終了されたインストールの終わりで表示されます。
SetupCompleteSuccess	ダイアログ	このダイアログは、インストールが正常に終了した場合に表示されます。
SetupInitialization	ダイアログ	このダイアログは、インストールの開始準備中に表示されます。このダイアログのデフォルトのテキストは、「インストールの準備をしています...」です。
CostInitialize	標準アクション	このアクションは、インストールの現在の設定に必要なディスク容量を判断する最初のステップです。

テーブル 22 • [管理] シーケンスのユーザー インターフェイス アクション (続き.)

アクション名	イベントの種類	説明
FileCost	標準アクション	インストールの現在の設定に必要なディスク容量を判断します。新規バージョンで上書きするファイルがあるかどうか、上書きがディスク容量にどのように影響するかを確認します。
CostFinalize	標準アクション	CostFinalize アクションは、現在の設定でインストールに必要な合計ディスク容量を判断します。また、すべてのターゲットディレクトリが書き込み可能であることを確認します。
AdminWelcome	ダイアログ	これは [管理] シーケンス中に最初に表示されるダイアログです。
SetupProgress	ダイアログ	このダイアログは、インストールの進行状況を表示します。
ExecuteAction	標準アクション	このアクションは [管理]–[実行] シーケンスを実行します。

実行

[実行] シーケンスには、正しく機能するためにユーザー インターフェイスを使用しないすべてのアクションが含まれます。これらのアクションには、ファイル転送、コンポーネントおよび機能のパブリッシュ、COM サーバーの登録などがあります。

テーブル 23 • [管理] シーケンスでアクションとダイアログを実行する

アクション名	イベントの種類	説明
CostInitialize	標準アクション	このアクションは、現在のインストール構成に必要なディスク容量を判断する最初のステップです。
FileCost	標準アクション	現在のインストール構成に必要なディスク容量を判断します。新規バージョンで上書きするファイルがあるかどうか、上書きがディスク容量にどのように影響するかを確認します。
CostFinalize	標準アクション	CostFinalize アクションは、現在の設定でインストールに必要な合計ディスク容量を判断します。また、すべてのターゲットディレクトリが書き込み可能であることを確認します。
InstallValidate	標準アクション	InstallValidate アクションは、現在のセットアップ設定に必要なディスク容量があるかどうかを判断します。
InstallInitialize	標準アクション	InstallInitialize アクションは、エンド ユーザーのシステムに変更を加えるアクションの開始を示します。

テーブル 23 - [管理] シーケンスでアクションとダイアログを実行する (続き.)

アクション名	イベントの種類	説明
InstallAdminPackage	標準アクション	このアクションは、インストール データベースを管理インストールの場所にコピーします。
InstallFiles	標準アクション	InstallFiles アクションでは、機能をインストールする場合、その機能に属するファイルすべてをターゲットマシンにコピーします。ローカルにインストールされるコンポーネントに関連したファイルのみが、ターゲットマシンにコピーされます。
InstallFinalize	標準アクション	インストールまたはアンインストールの最後のステップです。

[ユーザー インターフェイス] シーケンス



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

ユーザー インターフェイスには、デフォルトのユーザー インターフェイスに必要なすべてのアクションおよびダイアログが含まれています。これらのアクションやダイアログは、ターゲット システムを変更しません。一般的に、あとで実行するインストール作業のために、システム環境やエンドユーザーに関する情報を収集します。

デフォルトで、[管理] と [インストール] シーケンスは、それぞれ [ユーザー インターフェース] シーケンスを含みます。



プロジェクト- *InstallScript MSI プロジェクト用のユーザー インターフェイス シーケンスでは、InstallScript のカスタム アクションはサポートされていません。*

[実行] シーケンス



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[実行] シーケンスは、インストールがサイレント モードで実行されていない限り、[ユーザー インターフェイス] シーケンスの後で実行されます。[実行] シーケンスには、ターゲット システムを変更するすべての標準アクションとカスタム アクションが含まれています。たとえば、ファイル転送は [実行] シーケンス中にが処理されます。通常、このシーケンスはターゲット コンピューターに変更が加えられた際に行われるインストール作業の一部として考えることができます。

[インストール]、[アドバタイズ]、および [管理] の各シーケンスのアクションは、[実行] シーケンス内でそれぞれ異なったアクションを起動します。3 種類のシーケンスの目的がそれぞれ異なるために、起動するアクションも異なります。たとえば、[インストール] シーケンスはターゲット コンピューターに製品をインストールします。一方、[アドバタイズ] シーケンスは、ターゲット システム上にある製品のアドバタイズのショートカット、ファイルの種類の情報および COM サーバー データなどをインストールしますが、エンド ユーザーがショートカットを選択し、登録された文書を開くか、アドバタイズされた COM サーバーを起動するまで製品のファイルを転送しません。シーケンス名から推測できるように、これは製品のアドバタイズのために使用します。

アクションをシーケンスに挿入する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

シーケンスはインストールと関連付けられたすべてのアクションが実行されるタイミングを指示します。アクションは、任意のシーケンスで追加 、削除 、または順序変更を行うことができます。たとえば、インストールの一部として Readme ファイルを表示するために、インストールに Readme ファイルを起動するカスタム アクションを追加することができます。このアクションはシーケンスに挿入する必要があります。

アクションのシーケンスへの挿入は、シーケンスのアクションがいつ実行されるかによって決まります。多くのアクションは、そのカスタム アクションが機能する前に実行されている他のアクションに依存しています。たとえば、インストールに組み込む実行可能ファイルを起動する場合、この実行可能ファイルに依存するアクションを実行できるのは、シーケンス処理におけるファイルのインストールの終了後になります。



メモ: *InstallShield では、シーケンスに対する検証はされません。アクションが正常に機能しないシーケンスにある場合でも、インストールが実行されるまでエラーは発生しません。*

アクションの挿入



タスク ***アクションをシーケンスに挿入するには以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[シーケンス]** エクスプローラーで、新しいアクションの前にくるアクションまたはダイアログを右クリックして、**[挿入]** をクリックします。[アクションの挿入] ダイアログ ボックスが開き、シーケンスに追加することができるアクションおよびダイアログが一覧表示されます。
3. ダイアログ ボックスの上にあるリストから、挿入するアクションの種類を選択します。

4. アクションの一覧があるボックスで、挿入するアクションの種類を選択します。
5. **[OK]** をクリックします。

InstallShield では、[カスタム アクション] エクスプローラーからカスタム アクションを [シーケンス] エクスプローラーのシーケンスにドラッグ アンド ドロップすることもできます。



タスク ***ドラッグ アンド ドロップで、カスタム アクションをシーケンスに挿入するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[カスタム アクション]** エクスプローラーからカスタム アクションを、**[シーケンス]** エクスプローラーの下にあるシーケンス内の適切な位置までドラッグします。次いで、それをシーケンス内で直前にくるアイテムの上にドロップします。



メモ: *カスタム アクションは、同じシーケンス内で 2 度呼び出すことはできません。これは、カスタム アクションの名前が* CustomAction *テーブルでキーとして使用されているためです。したがって、カスタム アクションを、そのカスタム アクションを既に含んでいるシーケンスに挿入することはできません。*

アクションの挿入場所の決定

インストールに含むことができるカスタム アクションには、標準または Windows Installer 対応の DLL 関数の呼び出し、実行可能ファイルの起動、スクリプト (Jscript、VBScript、または InstallScript) の実行、プロパティまたはディレクトリの設定、および 2 番目の .msi インストールの起動など、多くのカテゴリがあります。カスタム アクションの構成によって、これらのタイプをさまざまな場面で呼び出すことができます。詳細は、次を参照してください:

- DLL 内の関数を呼び出すカスタム アクションをシーケンスする
- .exe ファイルを起動するカスタム アクションをシーケンスする
- スクリプトを呼び出すカスタム アクションをシーケンスする
- プロパティまたはディレクトリのプロパティを設定するカスタム アクションのシーケンス
- 2 番目の .msi パッケージを起動するカスタム アクションをシーケンスする

インクルードしたマージ モジュールからのカスタム アクションの検討

ダイレクト エディターで ModuleInstallExecuteSequence テーブルを変更して、マージ モジュール内のカスタム アクションの起動を制御することができます。インストール プロジェクトにマージ モジュールを追加する際、マージ モジュールの中に含められたすべてのカスタム アクションおよびダイアログは [カスタム アクションとシーケンス] ビューを通してインストールのシーケンスに挿入することができます。

カスタム アクションを別のシーケンスへコピーする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

- *MSI データベース*
- *トランスフォーム*

InstallShield では、ドラッグアンドドロップ操作で、カスタム アクションをあるシーケンスから別のシーケンスへコピーすることができます。



タスク ***カスタム アクションをあるシーケンスから別のシーケンスへコピーするには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[シーケンス]** エクスプローラーで、コピーするアクションを検索します。
3. カスタム アクションをあるシーケンスから別のシーケンスへドラッグしている間、CTRL を押し続けます。そのカスタム アクションの直前にくるアクションまたはダイアログの上にドロップします。



メモ: *カスタム アクションは、同じシーケンス内で 2 度呼び出すことはできません。これは、カスタム アクションの名前が* CustomAction *テーブルでキーとして使用されているためです。したがって、カスタム アクションを、そのカスタム アクションを既に含んでいるシーケンスに移動またはコピーすることはできません。*

ダイアログおよび標準アクションは、ドラッグアンドドロップ操作で、異なる種類のシーケンスへ移動することはできません。

シーケンスの順序を変更する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[カスタム アクションとシーケンス] ビューの [シーケンス] エクスプローラー内のアクションおよびダイアログは、起動されるタイミングにしたがって時系列で編成されます。各アクションとアイテムにはまた、他のアイテムのシーケンス番号に関連してシーケンス中の位置を識別する番号があります。アイテムは、小さい番号から順番に起動されます。

(基本の MSI、MSI データベース、およびトランスフォーム プロジェクトに) ダイアログを、または、カスタム アクションをプロジェクトに追加するとき、それをシーケンスの適切な位置に追加して、起動されるタイミングを指定することができます。



タスク ***シーケンス内で実行するアクションとダイアログの順序を変更するには、以下の手順を実行します。***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. 以下のいずれかを実行します。

- 新しいカスタム アクションをシーケンスするには、**[カスタム アクション]** エクスプローラーからそれを、**[シーケンス]** エクスプローラーの下にあるシーケンス内の適切な位置までドラッグします。次いで、それをシーケンス内で直前にくるアイテムの上にドロップします。
- アクションまたはダイアログを、シーケンス内の異なる位置へ(または、あるシーケンスから別のシーケンスへ) 移動するには、それを元の位置からドラッグして、シーケンス内で直前にくるアイテムの上にドロップします。



メモ: *カスタム アクションは、同じシーケンス内で 2 度呼び出すことはできません。これは、カスタム アクションの名前が* CustomAction *テーブルでキーとして使用されているためです。したがって、カスタム アクションを、そのカスタム アクションを既に含んでいるシーケンスに移動またはコピーすることはできません。*

ダイアログおよび標準アクションは、ドラッグアンドドロップ操作で、異なる種類のシーケンスへ移動することはできません。

シーケンスの追加後にカスタム アクションを変更しても、それをシーケンスに割り当て直す必要はありません。シーケンス内に置かれたアクションは、実際のアクションに対するポインターで、[カスタム アクションとシーケンス] ビューでアクションに変更を加えると、動的に更新されます。



ヒント: *また、以下の任意の方法を用いて、**[シーケンス]** エクスプローラー内のアクションおよびダイアログを移動することができます。*

- *アクションまたはダイアログを右クリックして、**[上へ移動]** または **[下へ移動]** をクリックします。*
- *アクションまたはダイアログをクリックしたあと、CTRL+シフト+上矢印、または、CTRL+シフト+下矢印を押します。*
- *"シーケンス番号" 設定を編集して、アクションまたはダイアログの位置を変更します。"シーケンス番号" 設定は、アクションまたはダイアログをクリックしたときに、右のグリッドに表示されます。アクションまたはダイアログのシーケンス内における順序を先にする場合は、シーケンス番号を小さくします。反対に、シーケンス内における順序を後にする場合は、シーケンス番号を大きくします。*

アクションをシーケンスから削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*



タスク ***アクションをシーケンスから削除するには、以下の手順を実行します。***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **シーケンス** エクスプローラーで、削除するアクションを含むシーケンスを展開します。
3. アクションを右クリックして、**削除**をクリックします。

アクションが削除されるのは指定したシーケンスからだけで、すべてのシーケンスまたはプロジェクトからは削除されません。インストール プロジェクトからのアクションの完全な削除は、[カスタム アクションとシーケンス] ビューの [カスタム アクション] エクスプローラーで行うことができます。

ロールバック カスタム アクションのシーケンス



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

ターゲット システムに直接変更を加えるインストール プロジェクトの中のカスタム アクションには、ロールバック同等のカスタム アクションが必要です。ロールバック カスタム アクションは、ロールバックが実行されたときにこれらの変更を元に戻すことができます。たとえば、ターゲット システムからファイルを削除するカスタム アクションがあるのに、削除したファイルを復元するロールバックカスタム アクションが存在しないと、ロールバックが終了しても、コンピューターが不安定な状態のままになることがあります。ロールバックカスタム アクションは、事前定義かカスタム アクションと同様に動作します。つまり、シーケンスで最初に発生したときには起動しません。ロールバックスクリプトに書き込まれ、ロールバックが起きたときにのみ起動します。

ロールバック カスタム アクションのシーケンスへの挿入時には、以下の事項に留意してください。

- ロールバックは、[実行] シーケンス中にのみ実行でき、[ユーザー インターフェイス] シーケンス中には実行できません。したがって、ロールバック アクションは、[実行] シーケンスで、InstallInitialize の後、および InstallFinalize の前に配置する必要があります。
- ロールバック カスタム アクションは、ロールバックするアクションの前にシーケンスする必要があります。その他のインスタンスでは、Windows Installer サービスはシーケンスを上から下へ実行するため、シーケンス番号が小さいほど、そのアクションが早く起動します。ロールバックスクリプトでは、サービスは反対方向に実行されます。したがって、必要なときにロールバック アクションを起動するには、ロールバックを行うアクションの前にそれをシーケンスします。

.exe ファイルを起動するカスタム アクションをシーケンスする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

カスタム アクションとして実行可能ファイルを起動することには、シーケンス中のどこにこのアクションが挿入されるかに関して、DLL ファイル関数の呼び出しよりもやや扱いにくい問題があります。インストールのどこで .exe ファイルを起動するのか、またどのシーケンス中でカスタム アクションを起動したいかなどについて考慮する必要があります。

.msi パッケージに格納されている実行可能ファイル

.msi パッケージに保存された .exe ファイルを起動する場合、カスタム アクションをシーケンスのどこにでも配置することができます。

テーブル 24 ・ .msi パッケージ内の .exe ファイルを起動するアクションの設定

アクションの種類	場所	スケジュール	Sequence
実行可能ファイルを起動する	.msi パッケージに保存	なし	シーケンス中の任意の場所

インストール済み実行可能ファイル

インストールの一部として .exe ファイルがインストールされる場合、選択したスケジューリングのプロパティ（遅延実行かまたは即時実行）に応じて以下の 2 組の規則のうちの 1 つに従う必要があります。すぐに実行の場合は、アクションを正常に動作させるために InstallFinalize アクションの後に置く必要がありあます。遅延実行の場合は、InstallFiles アクションと InstallFinalize アクションの間に挿入する必要があります。

テーブル 25 ・ インストールされた .exe ファイルを起動するアクションの設定

アクションの種類	場所	スケジュール	Sequence
実行可能ファイルを起動する	インストール済み	即時実行	InstallFinalize の後
実行可能ファイルを起動する	インストール済み	遅延実行	InstallFiles の後、InstallFinalize の前

ローカル実行可能ファイル

実行したい .exe ファイルが既にシステム上にある場合、アクションを CostFinalize アクションの呼び出しの後に挿入する必要があります。以上の事項を以下の表に整理して示します。

テーブル 26 ・ 既にインストールされた .exe ファイルを起動するアクションの設定

アクションの種類	場所	スケジュール	Sequence
実行可能ファイルを起動する	ローカル	なし	CostFinalize の後

DLL 内の関数を呼び出すカスタム アクションをシーケンスする

DLL ファイル内の関数を呼び出すカスタム アクションをシーケンスする場合、DLL ファイルの参照方法として、インストール時に DLL ファイルが配置されている場所に応じて 3 つの方法があります:

- DLL ファイルは .msi ファイルにストリーム入力できますが、インストールはされません。
- DLL ファイルはターゲット システムのパスに置くことができます。(標準の DLL ファイルに適用されますが、Windows Installer DLL ファイルLには適用されません)
- または、DLL ファイルを残りのインストール ファイルと一緒にインストールできます。

その他の考慮すべき事項として、スケジューリングがあります。2 つの主なスケジューリング方法は、即時実行と遅延実行です。即時実行は、Windows Installer が .msi ファイルを処理するときにカスタム アクションを実行する方法です。遅延実行は、アクションをキューに挿入し、スクリプトに書かれた順番に実行するようにインストーラーに指示する方法です。

.msi パッケージに格納されている DLL ファイル

.msi パッケージに保存された DLL ファイル から関数を呼び出す場合、カスタム アクションをシーケンスの任意の場所に配置することができます。

テーブル 27・.msi パッケージに格納されている DLL ファイル内の関数を呼び出すアクションの設定

アクションの種類	場所	スケジュール	Sequence
DLL 関数	.msi パッケージに保存	なし	任意の場所

システムのパスにある DLL ファイル

ターゲット システムに存在する DLL ファイルから関数を呼び出す場合も、カスタム アクションを一連のアクションの任意の場所に置くことができます。

テーブル 28・システムのパスで検索された DLL ファイル内の関数を呼び出すアクションの設定

アクションの種類	場所	スケジュール	Sequence
DLL 関数	ターゲット システム上	なし	任意の場所

インストール済み DLL ファイル

インストール中にターゲットコンピューターにインストールされるファイルから DLLI 関数を呼び出し、それを即時実行で実行するようにスケジューリングした場合は、そのアクションは InstallFinalize アクションの後にのみ配置できます。遅延実行中にアクションを起動したい場合は、そのアクションを InstallFiles アクションと InstallFinalize アクションの間に配置します。次の表はこの点を示したものです。

テーブル 29・既にインストールされた DLL ファイル内の関数を呼び出すアクションの設定

アクションの種類	場所	スケジュール	Sequence
DLL 関数	インストール済み	即時実行	InstallFinalize の後
DLL 関数	インストール済み	遅延実行	InstallFinalize の前、InstallFiles の後

スクリプトを呼び出すカスタム アクションをシーケンスする



プロジェクト: *VBScript と JScript の情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallScript 情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

カスタム アクションの中でスクリプトを使用すると、事実上どんなタスクを実行することも可能です。たとえば、外部アプリケーションを起動したり、レジストリ項目を作成したりできます。

VBScript と JScript コード

製品と一緒にインストール

インストール中にスクリプトがインストールされる場合、選択したスケジューリングのプロパティ (遅延実行かまたはすぐに実行) に応じて以下の 2 組の規則のうちの 1 つに従う必要があります。すぐに実行の場合は、アクションを正常に動作させるために InstallFinalize アクションの後に置く必要がありあます。遅延実行の場合は、InstallFiles アクションと InstallFinalize アクションの間に挿入する必要があります。

テーブル 30 ・ 製品と共にインストールされた VBScrilpt または JScript ファイルを呼び出すアクションの設定

アクションの種類	場所	スケジュール	Sequence
VBScript または JScript コードの実行	インストール済み	即時実行	InstallFinalize の後
VBScript または JScript コードの実行	インストール済み	遅延実行	InstallFiles の後、InstallFinalize の前

Binary テーブルに保存する

呼び出すスクリプト ファイルが Binary テーブルに保存されている場合、一連のアクションのどこにでもそのアクションを置くことができます。

テーブル 31 · Binary テーブルに格納されている VBScrilpt または JScript ファイルを呼び出すアクションの設定

アクションの種類	場所	スケジュール	Sequence
VBScript/JScript コードの実行	一時	なし	シーケンス中の任意の場所

ターゲット システム上に既存

呼び出すスクリプト ファイルがシステム上に既にある場合は、CostFinalize アクションの呼び出しの後にそのアクションを挿入する必要があります。

テーブル 32 ·

アクションの種類	場所	スケジュール	Sequence
VBScript/JScript コードの実行	ローカル	なし	CostFinalize の後

カスタム アクション中に保存されたディレクトリ

スクリプトをカスタム アクション中に直接置いた場合は、一連のアクション中のどの場所にもそのアクションを置くことができます。

InstallScript コード

InstallScript カスタム アクションは、InstallScript アクションのソースファイルが常に .msi パッケージにストリーム入力されるため、JScript または VBScript カスタム アクションと異なります。したがって、カスタム アクションはシーケンス番号として 2 と示されるアクションの後の任意の場所に挿入できます。この制限があるのは、スクリプト エンジンがシーケンス番号 2 の間に起動するためです。したがって、スクリプト エンジンが起動する前にスクリプトを呼び出した場合、システムはそれを処理できません。

カスタム アクションに、機能関数、またはセットアップの種類ダイアログ関数が含まれる場合、CostInitialize、FileCost、CostFinalize アクションの後に、カスタム アクションを挿入する必要があります。

プロパティまたはディレクトリのプロパティを設定するカスタム アクションのシーケンス



***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

プロパティまたはディレクトリを設定するカスタム アクションをシーケンスに配置する場合、まず、そのディレクトリまたはプロパティが必要とされる前にカスタム アクションを起動する必要があることに注意してください。たとえば、ターゲットコンピューター上にある EXE を起動したい場合、インストールで最初にそれを検索する必要があります。

実行可能ファイルが見つかった後、そのパスを Directory テーブルに入力し、その後、実行可能ファイルを起動することができます。しかし、実行可能ファイルの起動後にカスタム アクションを呼び出した場合、実行可能ファイルは見つかりません。

その他のシーケンスおよびスケジュール上の制限は次のとおりです。

- 検証規則 ICE12 によると、すべての タイプ-35 カスタム アクション (カスタム アクションウィザードで「**ディレクトリの設定**」と呼ばれる) を、シーケンスの標準 CostFinalize アクションの後に並べる必要があります。
- 同様に、ICE12 によると、Directory テーブルのプロパティの値を設定するすべてのタイプ-51 カスタム アクション (カスタム アクションウィザードで **プロパティの設定** と呼ばれる) を、標準 CostFinalize アクションの前にスケジュールする必要があります。
- プロパティを設定するすべてのカスタム アクションは、すぐに実行するようスケジューリングする必要があります。

2 番目の .msi パッケージを起動するカスタム アクションをシーケンスする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*



重要 *・Windows Installer ではネスト インストールの使用を避けるようにしてください。ネスト インストールが含まれている場合エンドユーザーが適切に処理することが困難であるため、時折アプリケーションの誤作動を引き起こします。マイクロソフト社では一般向けにリリースされる製品のインストールについて、ネスト インストールおよびネスト インストール カスタム アクションの使用を避けることを推奨しています。さらに詳しい情報は、MSDN Web サイトの「Concurrent Installations」を参照してください。*

ネスト インストールのカスタム アクション (カスタム アクション ウィザードの **[別の .msi パッケージを起動]** という名前) をシーケンスする場合、標準の CostFinalize アクションの後に、[実行] シーケンスにアクションを配置する必要があります。

また、該当する場合、インストール内に含まれる任意ファイルがメイン インストールで必要になる前に、カスタム アクションが起動するか確認してください。たとえば、2 番目の .msi ファイルの一部としてインストールされた実行可能ファイルを起動したい場合は、その実行可能ファイルが起動される前に必ずパッケージがインストールされていなければなりません。

ISSetAllUsers カスタム アクション

ISSetAllUsers がインストール パッケージのシーケンスに表示される理由

([アップグレード] ビューを使用して) インストールの Upgrade テーブルに 1 つ以上の記録を入力すると、InstallShield は [ユーザー インターフェイス] と [実行] インストール シーケンスの部分の両方に、ISSetAllUsers という .DLL カスタム アクションを挿入します。ご使用の製品がアップグレードとしてインストールされている場合、ISSetAllUsers カスタム アクションはインストールされているバージョンの ALLUSERS プロパティの値をチェックします。

ALLUSERS プロパティは [カスタマー情報] ダイアログ (基本の MSI プロジェクトの場合) に示され、初回インストール中にエンドユーザーによって **SdCustomerInformation** または **SdCustomerInformationEx** ダイアログ (InstallScirpt MSI プロジェクトの場合) に示されます。ISSetAllUsers カスタム アクションは、インストールされたバージョンの値を新規バージョンの値と比較します。値が異なると、ISSetAllUsers は新しいバージョンの ALLUSERS プロパティを、インストールされているバージョンと一致するように設定します。

アップグレードの場合、ALLUSERS プロパティはカスタム アクションを使用しなければ設定できないことに注意してください。

新しいインストールの ALLUSERS プロパティは、FindRelatedProducts アクションで正常にアップグレード インストールを行うことができるように、インストールされているバージョンのプロパティと一致する必要があります。さらに、以前のバージョンが一人の特定のユーザー用にインストールされていて、すべてのユーザー用にアップグレードがインストールされていると、インストールが結果的に破損して、正しくアンインストールされない場合があります。ISSetAllUsers は ALLUSERS プロパティをリセットしてこの問題を解決します。

ISSetAllUsers の役割

次の例で、ISSetAllUsers カスタム アクションの実際の役割を説明します。

1. My Application 1.0 を、ALLUSERSプロパティを *1* に設定してインストールします (ユーザーはインストール中にエンド ユーザー ダイアログで [**すべてのユーザー用にインストール**] を選択)。
2. My Application 2.0 がバージョン 1.0 のアップグレードとして作成され、Upgrade テーブルにバージョン 1.0 のアップグレードエントリが挿入されます。
3. エンドユーザーはターゲット システムにバージョン 2.0 をアップグレードとしてインストールします。
4. インストールの間、ISSetAllUsers は次の方法でインストールされているバージョンの ALLUSERS プロパティを調べ、新しいバージョンのプロパティと比較します。
    a. ISSetAllUsers はバージョン 2.0 の Upgrade テーブルの各エントリを繰り返し調べます。
    b. Upgrade テーブルの各アップグレード コードについて、ISSetAllUsers はターゲット システム上の関連製品を探します。
    c. Upgrade テーブルのバージョン制限と言語制限が (ステップ 4b で見つけた) インストールされている製品と一致する場合、ISSetAllUsers はインストールされているバージョンの ALLUSERSプロパティを確認します。

d. インストールされているバージョンの ALLUSERS 値が新しいバージョンと異なる場合、ISSetAllUsers は新しくインストールされたバージョンの ALLUSERS プロパティをこの値に設定し直します。



メモ: *一致する製品がターゲット システムにインストールされていない場合、ISSetAllUsers は何も実行しません。*

サポート ファイルを使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *スイート/アドバンスト UI*

サポート ファイルは、製品のインストール処理中にのみターゲット システムで使用できるファイルです。サポート ファイルはインストールが始まるとターゲット システムの一時ディレクトリにコピーされます。サポート ファイルはインストールが完了すると削除されます。サポート ディレクトリは、ダイナミック ファイルの場所を表し、ターゲット システムごとに異なります。また、同じシステムでもインストールするたびに異なります。

[サポート ファイル] ビュー (アドバンスト UI、基本の MSI、InstallScript オブジェクト、およびスイート/アドバンスト UI プロジェクトの場合)、または [サポート ファイル/ビルボード] ビュー (InstallScript プロジェクトおよび InstallScript MSI プロジェクトの場合) で、インストール中のみターゲット システム上で使用可能なファイルを追加および削除することができます。

サポート ファイルを追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *スイート/アドバンスト UI*

タスク ***サポート ファイルをインストール プロジェクトに追加するには、以下の手順に従います:***

1. 基本の MSI、InstallScript オブジェクト、およびスイート プロジェクトの場合: **[動作とロジック]** の下にある [ビューリスト] で、**[サポート ファイル]** をクリックします。

   InstallScript、および InstallScript MSI プロジェクトの場合: **[動作とロジック]** の下にある [ビューリスト] で、**[サポート ファイル/ビルボード]** をクリックします。

2. オプションで、ファイルに 1 つ以上のサブフォルダーを作成することもできます。その場合、**[サポート ファイル]** エクスプローラーで、[言語非依存] ノードまたは言語固有ノードのひとつを右クリックしてから、**[新しいフォルダー]** をクリックします。新しいフォルダーが追加されます。フォルダーに使用する名前を入力します。
3. **[サポート ファイル]** エクスプローラーで、追加するサポート ファイルが必要なアイテムをクリックします。
4. **[ファイル]** ペインで右クリックして **[ファイルの挿入]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
5. 含めるファイルを参照します。複数のファイルを選択するには、ファイルをクリックしながら CTRL キーを押します。
6. **[OK]** をクリックします。

InstallShield は、ファイルを [ファイル] ペインに追加します。

ヒント: *Windows エクスプローラーからファイルをドラッグして、[ファイル] ペインにドロップすることもできます。*

ライセンス ファイルの追加

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

[サポート ファイル/ビルボード] ビューで、使用許諾契約書を含むテキスト ファイルを追加できます。

タスク ***ライセンスファイルを追加するには、以下の手順を実行します。***

1. ビュー リストの [動作とロジック] の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. オプションで、ライセンス ファイルに 1 つ以上のサブフォルダーを作成することもできます。その場合、**[サポート ファイル]** エクスプローラーで、[言語非依存] ノードまたは言語固有ノードのひとつを右クリックしてから、**[新しいフォルダー]** をクリックします。新しいフォルダーが追加されます。フォルダーに使用する名前を入力します。
3. **[サポート ファイル]** エクスプローラーで、追加するライセンス ファイルを含めるアイテムをクリックします。

4. **[ファイル]** ペインで右クリックして **[ファイルの挿入]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
5. 含めるファイルを参照します。複数のファイルを選択するには、ファイルをクリックしながら CTRL キーを押します。
6. スクリプトの **SdLicense*** 関数の 1 つにある szLicenseFile パラメーターでファイルを呼び出します。

サポート ファイルの並べ替え



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

[サポート ファイル] ビュー (アドバンスト UI、基本の MSI およびスイート/アドバンスト UI プロジェクトの場合) または [サポート ファイル/ビルボード] ビュー (InstallScript プロジェクトと InstallScript MSI プロジェクトの場合) の [ファイル] ペインで、ファイルをソートする列の見出しをクリックしてファイルをソートすることができます。任意の列でソートすることができます。

Disk1 フォルダーへファイルとフォルダーを追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

Disk1 ノードでは、インストールメディアの Disk1 に入れるファイルとフォルダーを指定することができます。これらのファイルとフォルダーは、インストールの実行時、ターゲット システムに自動的にインストールはされません。代わりに、アプリケーションまたはインストールからインストール メディアにリンクさせることができます。

たとえば、アプリケーションと共に含める大きい再配布可能ファイルがあるとします。エンドユーザーが再配布可能ファイルにアクセスできるようにしたくても、アプリケーションのインストールには含めたくない場合があります。このような場合、このファイルを Disk1 フォルダーに入れることができます。



タスク ***Disk1 フォルダーにファイルまたはフォルダーを追加する場合、次の操作を実行します。***

1. 基本の MSI プロジェクトの場合: [動作とロジック] の下にある [ビューリスト] で、**[サポート ファイル]** をクリックします。

   InstallScript、および InstallScript MSI プロジェクトの場合: [動作とロジック] の下にある [ビューリスト] で、**[サポート ファイル/ビルボード]** をクリックします。

2. **[サポート ファイル]** エクスプローラーで、**Disk1** をクリックします。
3. **[ファイル]** ペインで右クリックして **[ファイルの挿入]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. 含めるファイルを参照します。複数のファイルを選択するには、ファイルをクリックしながら CTRL キーを押します。
5. **[OK]** をクリックします。

InstallShield は、ファイルを [ファイル] ペインに追加します。

Disk1 フォルダーからファイルまたはフォルダーを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

タスク ***Disk1 フォルダーからファイルまたはフォルダーを削除する場合、以下の手順にしたがいます:***

1. 基本の MSI プロジェクトの場合: [動作とロジック] の下にある [ビューリスト] で、**[サポート ファイル]** をクリックします。

   InstallScript、および InstallScript MSI プロジェクトの場合: [動作とロジック] の下にある [ビューリスト] で、**[サポート ファイル/ビルボード]** をクリックします。
2. **[サポート ファイル]** エクスプローラーで、**Disk1** をクリックします。
3. **[ファイル]** ペインでは、ファイルまたはフォルダーを右クリックして、**[削除]** をクリックします。

最後の Disk フォルダーにファイルとフォルダーを追加する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

最後の Disk ノードでは、インストールメディアの最後のディスクに入れるファイルとフォルダーを指定することができます。これらのファイルまたはフォルダーは、インストールの実行時、ターゲット システムに自動的にインストールはされません。代わりに、アプリケーションまたはインストールからインストール メディアにリンクさせることができます。

たとえば、アプリケーションと共に含める大きい再配布可能ファイルがあるとします。エンドユーザーが再配布可能ファイルにアクセスできるようにしたくても、アプリケーションのインストールには含めたくない場合があります。このような場合、このファイルを 最後のディスク フォルダーに入れることができます。

タスク

最後のディスクイメージフォルダーにファイルまたはフォルダーを追加する場合、次の操作を実行します。

1. ビュー リストの [動作とロジック] の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. **[サポート ファイル]** エクスプローラーで、**最後の Disk** をクリックします。
3. **[ファイル]** ペインで右クリックして **[ファイルの挿入]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. 含めるファイルを参照します。複数のファイルを選択するには、ファイルをクリックしながら CTRL キーを押します。
5. **[OK]** をクリックします。

InstallShield は、ファイルを [ファイル] ペインに追加します。

最後の Disk フォルダーからファイルとフォルダーを削除する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

タスク

最後のディスクイメージフォルダーからファイルまたはフォルダーを削除する場合、次の操作を実行します。

1. ビュー リストの [動作とロジック] の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. **[サポート ファイル]** エクスプローラーで、**最後の Disk** をクリックします。
3. **[ファイル]** ペインでは、ファイルまたはフォルダーを右クリックして、**[削除]** をクリックします。

その他の Disk フォルダーにファイルとフォルダーを追加する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

[その他] ノードを使うと、最初または最後のディスクの他にインストールメディアのディスクに入れるファイルとフォルダーを指定できます。これらのファイルまたはフォルダーは、インストールの実行時、ターゲット システムに自動的にインストールはされません。代わりに、アプリケーションまたはインストールからインストールメディアにリンクさせることができます。

タスク

ディスクイメージフォルダーにファイルまたはフォルダーを追加する場合、次の操作を実行します。

1. ビュー リストの [動作とロジック] の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. **[サポート ファイル]** エクスプローラーで、**[その他]** をクリックします。
3. **[ファイル]** ペインで右クリックして **[ファイルの挿入]** をクリックします。**[開く]** ダイアログ ボックスが開きます。

4. 含めるファイルを参照します。複数のファイルを選択するには、ファイルをクリックしながら CTRL キーを押します。
5. **[OK]** をクリックします。

InstallShield は、ファイルを [ファイル] ペインに追加します。



ヒント: *ディスクを指定するには、リリース ウィザードを実行して [一般オプション] パネルで [その他のディスクファイル] ボタンをクリックします。*

他の Disk フォルダーからファイルとフォルダーを削除する



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*



タスク ***ディスクイメージフォルダーからファイルまたはフォルダーを削除する場合、次の操作を実行します。***

1. ビュー リストの [動作とロジック] の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. **[サポート ファイル]** エクスプローラーで、**[その他]** をクリックします。
3. **[ファイル]** ペインでは、ファイルまたはフォルダーを右クリックして、**[削除]** をクリックします。

サポート ファイルを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *スイート/アドバンスト UI*



タスク ***プロジェクトからサポート ファイルを削除するには、以下の手順に従います:***

1. 基本の MSI、InstallScript オブジェクト、およびスイート プロジェクトの場合: **[動作とロジック]** の下にある [ビューリスト] で、**[サポート ファイル]** をクリックします。

   InstallScript、および InstallScript MSI プロジェクトの場合: **[動作とロジック]** の下にある [ビューリスト] で、**[サポート ファイル/ビルボード]** をクリックします。
2. **[サポート ファイル]** エクスプローラーで、削除するサポート ファイルを含むアイテムをクリックします。
3. **[ファイル]** ペインでは、ファイルまたはフォルダーを右クリックして、**[削除]** をクリックします。

エンドユーザー インターフェイスを定義する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

ドキュメントのこのセクションでは、エンドユーザー インターフェイスの異なる要素を定義することができる InstallShield の異なる機能について説明します。「エンドユーザー インターフェイスの定義」セクションの一部では、インストールのエンドユーザー インターフェイスのダイアログの作成および使用方法についても説明しています。他では、インストールのローカライズを行うための文字列などのトピックを扱っています。

このセクションには、次のトピックが含まれています:

- ダイアログの使い方
- ウィザード インターフェイスを使って作業する
- エンドユーザー インターフェイスのローカライズ
- ビルボードを表示する
- InstallScript および InstallScript MSI インストールで背景ウィンドウを表示する
- 実行時にリスト ボックスへデータを挿入する
- ファイル参照ダイアログを表示する
- InstallScript インストールでネットワーク参照ダイアログを表示する

ダイアログの使い方

このセクションでは、InstallShield の異なるプロジェクトの種類ごとに、エンドユーザー ダイアログの使い方について、基本的な情報からより高度な情報まで部分的に取り上げられています。



プロジェクト: *基本の MSI インストールでは通常、Windows Installer がランタイム ユーザー インターフェイスを制御します。InstallScript とInstallScript MSI インストールでは通常、InstallScript エンジンがランタイム エンド ユーザー インターフェイスを制御します。したがって、ドキュメントのこのセクションの一部は基本の MSI インストールに、一部は InstallScript とInstallScript MSI インストールに、また一部は 3 つのすべてのプロジェクト タイプに適用します。*

プロジェクトでダイアログを使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

ダイアログは、インストールのユーザー インターフェイスを提供します。エンドユーザーからの情報を要求し、インストール プロセスの進行状況についてフィードバックを提供します。

プロジェクトにおけるダイアログの作業方法は、作成するインストールの種類によって異なります。InstallScript または InstallScript MSI プロジェクトにダイアログを追加する手順は、基本の MSI プロジェクトにダイアログを追加する手順と異なります。

共通の操作

一部のダイアログ操作は、基本の MSI、InstallScript および InstallScript MSI プロジェクトで同一です。

- ダイアログ ウィザードを使用して新しいダイアログを作成する
- 他のプロジェクトで使用するダイアログを .isd ファイルへエクスポートする
- .isd ファイルからダイアログをインポートする
- すべてのダイアログを .rc ファイルへエクスポートする
- リソース .dll ファイルからダイアログをインポートする
- 他のプロジェクトへダイアログをエクスポートする

ダイアログ ウィザードを使用して新しいダイアログを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

InstallShield は、ウィザード パネルに従って新しいダイアログをインストールに追加し構成できるダイアログ ウィザードを提供しています。

既存のユーザー アカウントを作成または設定する機能を追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

多くのサーバー アプリケーションは、インストール中にユーザー アカウント情報を必要とします。サーバー アプリケーションが別のユーザーに制限されているリソースへアクセスできるようにするために、ユーザー アカウントを使えるようにしておくことが必要な場合が多々あります。InstallShield では、インストールに適切なランタイム ダイアログを追加して、インストール中に既存の Windows ユーザー アカウントの設定、または、新規アカウントの作成を行うことができます。

テーブル 1・ユーザー アカウント サポートの追加 — プロジェクト別の説明

プロジェクトの種類	追加情報
基本の MSI	基本の MSI プロジェクトでは、LogonInformation ダイアログで入力したユーザー名、パスワード、およびグループ情報は、次のプロパティに格納されます。 • IS_NET_API_LOGON_USERNAME • IS_NET_API_LOGON_GROUP • IS_NET_API_LOGON_PASSWORD
InstallScript	InstallScript プロジェクトでは、LogonInformation ダイアログでエンドユーザーが入力したユーザー名、パスワード、およびグループ情報は、次のグローバル変数に格納されます。 • IFX_NETAPI_USER_ACCOUNT • IFX_NETAPI_PASSWORD • IFX_NETAPI_GROUP
InstallScript MSI	InstallScript MSI プロジェクトでは、指定されたユーザー アカウントおよびパスワードは、次のグローバル変数およびプロパティに格納されます。 • IFX_NETAPI_USER_ACCOUNT • IFX_NETAPI_GROUP • IFX_NETAPI_PASSWORD • IS_NET_API_LOGON_USERNAME • IS_NET_API_LOGON_GROUP • IS_NET_API_LOGON_PASSWORD

LogonInformation ダイアログのサポートを基本の MSI プロジェクトへ追加する

タスク

LogonInformation ダイアログを基本の MSI プロジェクトへ追加するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** を右クリックしてから、**[新しいダイアログ]** を選択します。**新規ダイアログ ウィザード**が開きます。
3. ウィザードの各パネルを完成します。**[ダイアログ テンプレート]** パネルで、**[ログオン情報パネル、および関連する子ダイアログ]** を選択します。

新規ダイアログ ウィザードの各パネルを完了した後、インストール プロジェクトをビルドおよび実行します。LoginInformationDialog ダイアログが実行されたダイアログ シーケンスで、適切なダイアログが表示されます。

InstallScript プロジェクトまたは InstallScript MSI プロジェクトに LogonInformation ダイアログのサポートを追加する

タスク

InstallScript プロジェクトまたは InstallScript MSI プロジェクトに LogonInformation ダイアログのサポートを追加するには、次の手順を実行します。

1. LogonInformation ダイアログ セットを挿入する InstallScript コードの場所に移動します。通常、**OnFirstUIBefore** 内に追加します。例:

   ```
   Dlg_SdLogon:
   nResult = SdLogonUserInformation(szTitle, szMsg, szAccount, szPassword);
   if (nResult = BACK) goto Dlg_SdWelcome;
   ```

2. InstallScript MSI プロジェクトでは、次の関数を追加して Windows Installer のプロパティが InstallScript のグローバル変数と同じ値に設定されるようにします。**SdLogonUserInformation** を呼び出した後に追加することができます。

   ```
   OnLogonUserSetMsiProperties();
   ```

3. **[ビルド]** メニューで **[設定]** をクリックします。**[設定]** ダイアログ ボックスが開きます。
4. **[コンパイル/リンク]** タブをクリックします。
5. **[ライブラリ (.obl)]** ボックスに、新しいライブラリ ファイル (*.obl) の名前を入力します:

   ```
   NetApiRT.obl
   ```

メモ: *新しいライブラリ ファイル名は、*isrt.obl *ファイル名の前に追加します。*

InstallScript コードを追加した後、インストールをビルド並びに実行します。スクリプトが実行された時、適切なダイアログが表示されます。

基本の MSI、InstallScript および InstallScript MSI プロジェクトにおける LogonInformation ダイアログの制限

エンド ユーザーが LogonInformation ダイアログを使用しようとしたときに、ドメインの空白リストが表示されるか、「サーバーが見つかりません」エラーが表示されます。次の状況下で、この問題が発生します:

- ターゲットシステムがドメイン上に存在しない。
- ファイアーウォールでコンピュータ ブラウザ サービスが例外として構成されていない。
- ターゲット システム上で NetBIOS over TCP/IP が有効化されていない。
- ネットワーク上でブロードキャスト トラフィックが無効である。

他のプロジェクトで使用するダイアログを .isd ファイルへエクスポートする

エンドユーザーダイアログを別のプロジェクトで再利用するには、ダイアログを .isd ファイルとしてエクスポートしてから、必要に応じて別のプロジェクトにインポートすることができます。.isd ファイルは、ダイアログの動作とレイアウト情報を含みます。



タスク ***.isd ファイルへのダイアログのエクスポートは、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーでダイアログを右クリックしてから、**[ダイアログ ファイルへエクスポート]** を選択します。**[名前を付けて保存]** ダイアログ ボックスが開きます。
3. .isd ファイルを保存するフォルダーを参照します。必要であればファイル名を変更できます。
4. **[OK]** をクリックします。

これで .isd ファイルを使ってダイアログを別のインストール プロジェクトにインポートすることができます。

.isd ファイルからダイアログをインポートする

InstallShield では、ダイアログをインストール プロジェクトにインポートすることができます。ダイアログのファイル (.isd ファイル) はリポジトリに格納するか、その他の別の場所に格納することができます。



タスク ***.isd ファイルからダイアログをプロジェクトへインポートするには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーでダイアログを右クリックしてから、**[ダイアログのインポート]** を選択します。[ダイアログのインポート] ダイアログ ボックスが開きます。
3. 以下のいずれかを実行します。
   - **[レポジトリ アイテム]**ボックスで、プロジェクトに追加するダイアログをクリックします。
   - インポートするダイアログ ボックス (.isd ファイル) のファイルがリポジトリに格納されていない場合、**[参照]** ボタンをクリックしてファイルを選択します。
4. **[OK]** をクリックします。

サポートされている各言語のコピーがユーザーのプロジェクトに追加されます。

メモ- ダイアログをインポートするときには、ダイアログによって参照される文字列識別子、プロパティ、およびアクションが新しいインストール プロジェクト内にあることを確認してください。

競合の解決

ダイアログをインポートするときには、Windows Installer によって要求されるとおりにダイアログの名前とそのコントロールが固有のものであるかどうかが確認されます。既存のダイアログと同じ名前のダイアログはインポートできません。これは InstallShield が同一のダイアログをインポートしていると認識するためです。

すでにインストールプロジェクトに既に存在するコントロールまたは文字列識別子の名前と同じ名前を使う .isd ファイルをインポートする場合は、InstallShield がこの競合の解決方法を問い合わせます。既存の値を上書きするか、または既存のものを残すためにインポートされた値をスキップするかのどちらかを選択できます。

インストール中にダイアログを表示する

プロジェクトにダイアログを追加しても、そのダイアログがインストールに表示されるわけではありません。1 つ以上のプロジェクトのシーケンスにダイアログを挿入する処理については、「基本の MSI インストール中にダイアログを表示する」を参照してください。

ダイアログ ファイル (.isd) をリポジトリにパブリッシュする

別のプロジェクトで再利用、または他のユーザーと共有したい既存のダイアログ (.isd) がある場合、リポジトリにそれをパブリッシュすることができます。

タスク ***ダイアログをリポジトリにパブリッシュするには、次の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーでダイアログを右クリックしてから、**パブリッシュ ウィザード**を選択します。**パブリッシュ ウィザード**が開きます。
3. パブリッシュ ウィザードのパネルを完成します。

リポジトリからプロジェクトへダイアログをインポートした後、現在のダイアログと既存のリポジトリ ダイアログとは関連性を持ちません。ダイアログを変更してリポジトリへ再パブリッシュしても、プロジェクト内のインポート済みダイアログには影響しません。但し、リポジトリからプロジェクトへダイアログを再インポートすることができます。

すべてのダイアログを .rc ファイルへエクスポートする

ダイアログのリソースを Visual Studio のようなリソースエディターを使用して編集する場合、ダイアログを .rc ファイルへエクスポートします。



タスク ***ダイアログをリソース スクリプト (.rc) ファイルにエクスポートするには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** を右クリックしてから、**[ダイアログをリソース スクリプトへエクスポートする]** を選択します。

プロジェクトの場所にある <プロジェクト名>.rc というファイルに、フォルトの言語のダイアログのリソースがすべてコピーされます。

各ダイアログのレイアウトをできるだけ多く保存するために可能な限りの試みがなされます。しかし、ダイアログ エディター内のいくつかの情報についてはリソース エディター内で対応するものがないため、InstallShield では、以下の例外を除くダイアログ エディター内のリソースがエクスポートされます。

テーブル 2 ・ .rc ファイルへエクスポートされないリソース

例外	説明
文字列エントリ	すべての文字列エントリは、デフォルトの言語の値へ解決されます。
パス	テキストファイルやビットマップなど、あらゆるリソースへのパス変数を含むパスが、後にダイアログをインポートするのを簡単にするために保存されています。
選択ツリー	Windows Installer の選択ツリーは、.rc ファイルの標準ツリー コントロールとなります。
テキスト スタイル	ダイアログ エディターが、フォント情報 (ベース テキスト スタイルやテキスト スタイル) および最大文字長について別々のプロパティを持っている間は、これらのプロパティは Windows Installer テーブル内にあるようにグループ化されています。
動作	ダイアログのレイアウトと動作の完全なコピーを含む.isd ファイルとは違い、.rc ファイルはリソースのみを保存します。

リソースエディターでのダイアログの編集

ダイアログのジオメトリは、Windows のダイアログと同じように、リソース エディター、またはメモ帳でも編集できます。

ダイアログをプロジェクトへインポートする前に、.rc ファイルを .res ファイルへコンパイルし、.dll へビルドする必要があります。



タスク ***Visual Studio を使用している場合は、以下のステップに従って MyProject.rc を MyProject.dll へビルドします。***

1. リソース コンパイラ (Rc.exe) を使用して、次のコマンドライン ステートメントで **MyProject.rc** を **MyProject** へコンパイルします。

   rc MyProject.rc

2. Incremental Linker (Link.exe) を実行し、以下のコマンドを使用して .dll をビルドします。

```
link /DLL /NOENTRY /NODEFAULTLIB /MACHINE:iX86 /OUT:MyProject.dll MyProject.res
```

リソース .dll ファイルからダイアログをインポートする



タスク ***.dll ファイルからすべてのダイアログリソースをインポートするには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** を右クリックしてから、**[ダイアログをリソース DII からインポートする]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. インポートするダイアログリソースを含む .dll ファイルを参照します。
4. **[開く]** をクリックします。

各ダイアログがプロジェクトへ追加されます。名前付けの競合は、インポートしたダイアログに固有の番号を付加することによって解決されます。

インポートしたコントロールが、既に文字列エントリであるテキストを表示する場合、InstallShield では コントロールの Text プロパティに既存の文字列テーブルエントリが使用されます。



メモ: *ダイアログをインポートする際には、ダイアログによって参照されたプロパティが新規のインストール プロジェクト内にあるかどうかを確認してください。*

プロジェクトにダイアログを追加しても、そのダイアログが実行時にインストールで表示されるわけではありません。1 つ以上のインストールのシーケンスにダイアログを挿入する処理については、「基本の MSI インストール中にダイアログを表示する」を参照してください。

ダイアログは .rc ファイルへエクスポートされますが、InstallShield にそれらをインポートする前に、リソースを .dll ファイルへビルドする必要があります。詳細については、「すべてのダイアログを .rc ファイルへエクスポートする」を参照してください。

他のプロジェクトへダイアログをエクスポートする

InstallShield では、ダイアログを現在のプロジェクトからエクスポートして別の既存のプロジェクトへ挿入することができます。



タスク ***既存プロジェクトへダイアログをエクスポートするには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、ダイアログを右クリックしてから、**[プロジェクト ファイルへエクスポート]** を選択します。**[エクスポート情報]** ダイアログ ボックスが開きます。
3. 既存の .ism ファイルの保存場所を参照します。保存場所は、インストール プロジェクトまたはマージ モジュール プロジェクトのどちらかを使用できますが、ファイルは InstallShield の別のインスタンスでは開きません。
4. **[保存]** をクリックします。

このダイアログのコピーは、すべてのコントロール、すべてのダイアログの動作と共に特定の .ism ファイルに追加されます。ダイアログで使用されるすべての文字列テーブル、プロパティ、およびパス変数も新規プロジェクトにコピーされます。

対象の .ism ファイルに、異なるプロパティで同じ名前のダイアログがある場合は、[競合] ダイアログ ボックスが表示され、競合するダイアログを解決するオプションを選択できます。

リポジトリへダイアログをパブリッシュし、別のプロジェクトで再利用することもできます。詳細については、「リポジトリを使用してプロジェクトの要素を共有する」を参照してください。

プロジェクトからダイアログを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

基本の MSI プロジェクト



タスク ***基本の MSI プロジェクトからダイアログを削除するには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、ダイアログを右クリックし **[削除]** をクリックします。



メモ: *[カスタム アクションとシーケンス] ビューでダイアログを右クリックして [削除] を選択しても、プロジェクトからダイアログは削除されません。そのシーケンスから削除されるだけです。*

InstallScript および InstallScript MSI プロジェクト



タスク ***InstallScript または InstallScript MSI プロジェクトからダイアログを削除するには、次の手順に従います。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、ダイアログを右クリックし **[削除]** をクリックします。
3. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
4. インストールのエンドユーザー インターフェイスにダイアログを追加するスクリプトを削除します。

カスタム ダイアログの作成と構成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

異なる種類のプロジェクトでカスタム ダイアログを作成および構成する際、次のヒントを考慮してください。

InstallScript および InstallScript MSI プロジェクト

[ダイアログ] ビューに新規ダイアログを追加するとき、ダイアログのエントリが Dialog テーブルに追加されます。このテーブルにあるデータは、ダイレクト エディターを使って直接変更することができます。

ダイレクト エディターでは、新しいダイアログ用の ISResourceId フィールドの値を指定することができます。ISResourceId 値は、このダイアログを識別するスクリプト内で利用されます。

ダイアログをプロジェクトに追加した後、このダイアログをメモリにロードし、それを実行時に表示するためには **EzDefineDialog** や **WaitOnDialog** などの関数を呼び出す必要があります。Dialog テーブルで ISResourceId 値に対応する値は、**EzDefineDialog** の 4 番目の引数に使用してカスタム ダイアログを参照します。また **EzDefineDialog** の 2 番目の引数で、ヌル文字列ではなく ISUSER を指定した方が便利です。**EzDefineDialog** 関数は次のようになります。

```
EzDefineDialog("MyCustomDialog", ISUSER, "", 12005)
```

この例では、12005 は Dialog テーブルで指定されたとおり、このカスタム ダイアログの ISResourceId です。

カスタム ダイアログ関数を呼び出し、必要に応じてカスタム ダイアログを操作します。これらの関数についての詳細は、InstallScript 言語リファレンスに記載されています。

基本の MSI プロジェクト

基本の MSI プロジェクトの [ダイアログ] ビューでカスタム ダイアログを作成する場合、Windows Installer タイプのダイアログが作成されます。InstallScript カスタム ダイアログ関数は、Windows Installer タイプのダイアログと一緒に使用することはできません。このダイアログをインストールに表示するには、[次へ] と [戻る] コントロールを使用して 2 つのダイアログの間で連続して配置する必要があります。

ダイアログ エディターで変更を元に戻す



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

ダイアログ エディターは、各ダイアログに対して 50 アクションまで記憶しており、最近のものから順に変更を元に戻すことができます。また、プロジェクトを保存した後でも変更を元に戻すことができます。ただし、プロジェクトを閉じて再び開くと、元に戻す操作の履歴は消去されます。

変更を元に戻す



タスク ***変更を元に戻すには、以下の手順に従います:***

- Ctrl + Z を押します。
- **[編集]** メニューで **[元に戻す]** をクリックします。
- ツールバーの **[元に戻す]** ボタンをクリックします。

元に戻した変更をやり直す



タスク ***元に戻した変更をやり直して有効にするには以下の手順に従います:***

- Ctrl+Y を押します。
- **[編集]** メニューで **[やり直す]** をクリックします。
- ツールバー上の **[やり直す]** ボタンをクリックします。

ダイアログ エディターの使用中は、[元に戻す] と [やり直す] ボタンおよびメニュー コマンドが有効です。アクションを元に戻すには、そのダイアログで実行したアクションの履歴から移動します。繰り返す場合、前に移動します。

たとえば、ボタンのサイズを変更して、Tab Index プロパティを変更した場合、[元に戻す] をクリックすると、Tab Index プロパティの直前の値が復元されます。また、[元に戻す] を 2 回クリックすると、ボタンが元のサイズに戻ります。次に、[やり直す] を 2 回クリックすると、ボタンのサイズおよびそのタブ索引の変更が再び反映されます。



メモ: *ダイアログ エディターで変更された文字列エントリを元に戻すことはできません。*

InstallScript および InstallScript MSI プロジェクトでダイアログを使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield では、スクリプトで特定の関数を呼び出してエンドユーザー ダイアログを表示できます。[ダイアログ] ビューには、プロジェクトに追加するカスタム ダイアログおよび標準ダイアログがリストされています。

このセクションでは、InstallScript および InstallScript MSI プロジェクトで、ダイアログがある基本的な関数の作成および実行方法が説明されています。

InstallScript および InstallScript MSI インストールでダイアログを表示する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallScript または InstallScript MSI インストールのユーザー インターフェイスのほとんどは、**OnFirstUIBefore** や **OnFirstUIAfter** などのイベント ハンドラーで定義されます。**OnFirstUIBefore** に含まれる次のサンプル InstallScript コード は、**SdWelcome** および **SdLicense2** 用のコードです:

```
Dlg_SdWelcome:
    szTitle = "";
    szMsg = "";
    nResult = SdWelcome( szTitle, szMsg );
    if (nResult = BACK) goto Dlg_Start;

Dlg_SdLicense2:
    szTitle = "";
    szOpt1 = "";
    szOpt2 = "";
    szLicenseFile = SUPPORTDIR ^ "License.rtf";
    nResult = SdLicense2Ex( szTitle, szOpt1, szOpt2, szLicenseFile, bLicenseAccepted, TRUE );
    if (nResult = BACK) then
        goto Dlg_SdWelcome;
    else
        bLicenseAccepted = TRUE;
    endif;
```

各ダイアログ関数は、エンド ユーザーがどのボタンをクリックしてダイアログを終了したかを示す整数を返します。エンド ユーザーがダイアログでクリックした [戻る] ボタンを処理するために、ダイアログの直後には、その戻り値を定数 BACK と比較する if ステートメントが、前のダイアログの直前にジャンプする goto ステートメントを使って続きます。前述のコードサンプルでは、エンド ユーザーが **SdLicense2** ダイアログで [戻る] ボタンをクリックした場合、goto ステートメントが Dlg_SdWelcome ラベルにジャンプして、エンド ユーザーには **SdWelcome** ダイアログが表示されます。このため、2 つのダイアログ (たとえば、**SdWelcome** と **SdLicense2**) の間にダイアログ関数を挿入する場合、if ステートメント、ラベル、および goto ステートメントを適切に調整する必要があります。



タスク

インストールのユーザー インターフェイスの一部としてエンド ユーザーにダイアログを表示すには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. ダイアログ関数を、適切なスクリプト イベント ハンドラーに追加します。
3. 目的とするダイアログの動作に応じて、ダイアログ関数のパラメーターを変更します。

利用可能なパラメーターについては、ダイアログ関数に関するドキュメントを参照してください:

- ダイアログ関数
- ダイアログのカスタマイズ関数

4. たとえば、if および goto ステートメントを使って、ダイアログの流れを指定します。

InstallScript と InstallScript MSI プロジェクトのダイアログに含まれるテキストを変更する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ほとんどのダイアログ関数は、ダイアログのタイトル領域に表示されるテキストを定義する szTitle と呼ばれる文字列引数を受け付けます。たとえば、**SdRegisterUser** の呼び出しは次のとおりです:

```
SdRegisterUser( szTitle, szMsg, szName, szCompany );
```

デフォルトで、パラメーター szTitle はヌル文字列 (“”) として定義され、ダイアログがデフォルトのタイトル テキストを使用することを示します。**SdRegisterUser** では、デフォルト タイトルが次のスクリーンショットのように表示されます。

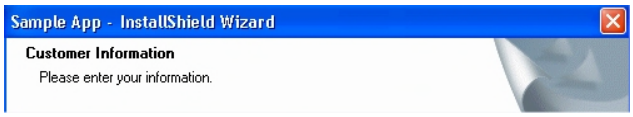


図 1: SdRegisterUser ダイアログのデフォルトのタイトル テキスト

タイトルを変更するには、szTitle パラメーターに表示する特定の文字列を入力します。タイトルは 2 つのセクションに分かれていて、タイトル領域の上部にある太字テキストと、メイン タイトルの下にある標準テキストがあります。

szTitle の値内で、この 2 つのセクションを指定するために、セクションの間に改行文字 ¥n を配置します。

```
SdRegisterUser (“新しいタイトル¥nこれは、新しいサブタイトルです。”,
    szMsg, szName, szCompany);
```

スクリプトを再コンパイルしてからインストールを実行すると、タイトルが次のサンプル スクリーンショットのように表示されます。

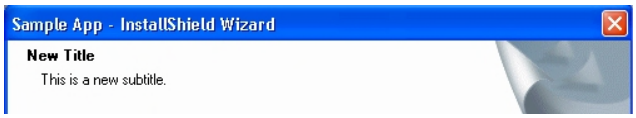


図 2: SdRegisterUser ダイアログの変更済みのタイトル テキスト

ダイアログに表示される他のテキストを変更する場合、表示されるテキストを含むパラメーター (たとえば、szMsg) が通常 1つ以上あります。ダイアログ ボックスのタイトルと同様に、メッセージ パラメーターのヌル文字列は、ダイアログが InstallShield で提供されているデフォルト メッセージ テキストを使用することを示します。

インストールに複数の UI 言語が含まれている場合、InstallScript でハードコード化された文字列の代わりに文字列 ID を使用できます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

InstallScript と InstallScript MSI プロジェクトのダイアログに含まれるデフォルト イメージについての背景情報



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

コントロール ID が 1200 であるスタティック ダイアログ ユーザー インターフェイス コントロールには、2 種類のイメージ (内部ダイアログのデフォルト バナー イメージ、および [ようこそ] ダイアログといった外部スキン ダイアログの左側に表示されるイメージ) が使用されます。デフォルト イメージは、透明色の使用が可能な .png イメージです。これらのコントロールのテキスト フィールドは、イメージを識別するのにデフォルトで次の構文が使用されます:

@@ResourceID;ScaleFactor

ResourceID は、.png イメージ (リソース タイプ PNG として格納される) またはビットマップ イメージ (リソース タイプ RT_BITMAP として格納される) のいずれかを検出するためのリソース識別番号を示します。*ScaleFactor* は、イメージが意図されている DPI スケール率を示します。

たとえば、スケール因子は 100% (96 DPI)、125% (120 DPI)、150% (144 DPI)、200% (192 DPI) となります。イメージに指定されているスケール因子が 200 の場合、200% DPI スケール以下で実行中のターゲット システム上で、イメージは縮小表示されます。200% ターゲットシステム上で、これは 1:1 の割合で表示されます。イメージに指定されているスケール因子が 100 の場合、200% DPI スケールで実行中のターゲット システム上で、イメージは拡大表示されます。

ビットマップ イメージ (.bmp) を識別する以前のフォーマットも使用できますが、スケーリングまたは .png イメージがサポートされていません:

@BitmapResourceID;TransparentFlag;3-DFlag;<unused>;TransparentColorKey

BitmapResourceID は、ビットマップ イメージのリソース ID 番号を示します。*TransparentFlag* は、1 (true) または 0 (false) のいずれかで示され、ビットマップを表示する際、*TransparentColorKey* フィールドで指定した色が透明かそうでないかを示します。*TransparentColorKey* は、ビットマップの透過色を示す RGB 値を指定します。

InstallScript および InstallScript MSI プロジェクトで新しいカスタム ダイアログを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

カスタム ダイアログを作成するには、次の一般的な手順を行う必要があります:

1. 新規ダイアログ ウィザードを使って、プロジェクトに新しいカスタム ダイアログを追加します。詳細については、「新規ダイアログ ウィザードを使って、InstallScript または InstallScript MSI プロジェクトに新しいカスタム ダイアログを追加する」を参照してください。
2. ダイアログにコントロールを追加します。詳細については、「InstallScript または InstallScript MSI プロジェクトのダイアログにコントロールを追加する」を参照してください。
3. ダイアログをメモリにロード、画面に表示、ダイアログのコントロールとのユーザーの対話を処理し、エンド ユーザーの操作が終了したときにダイアログを閉じるスクリプト関数を作成します。詳細については、「InstallScript を使用してカスタム ダイアログを実装する」を参照してください。

新規ダイアログ ウィザードを使って、InstallScript または InstallScript MSI プロジェクトに新しいカスタム ダイアログを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

タスク ***新しいカスタムダイアログをプロジェクトに追加するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** を右クリックしてから、**[新しいダイアログ]** を選択します。**新規ダイアログ ウィザード**が開きます。
3. ウィザードパネルに従い、新しいダイアログを作成します。**[ダイアログ テンプレート]** パネルで、**NewScriptBasedDialog** テンプレートまたは **NewSkinnableDialog** テンプレートを選択します。

   NewScriptBasedDialog および NewSkinnableDialog テンプレートには、コントロール ID 値 2 を持つ非表示コントロールが含まれています。エンド ユーザーが InstallScript ダイアログの右上にある閉じるボタンをクリックしてインストールをキャンセルできるようにするには、ダイアログにコントロール ID プロパティがこの値に設定されているボタン コントロールを含まなくてはなりません。ブランク ダイアログ テンプレートには、このコントロールは含まれていません。詳細については、「InstallScript を使用してカスタム ダイアログを実装する」を参照してください。

新しいカスタム ダイアログをプロジェクトに追加した後、ダイアログ コントロールを追加できます。

InstallScript または InstallScript MSI プロジェクトのダイアログ レイアウトを編集する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ダイアログ エディターを使って、選択したダイアログのコントロールを追加、削除、または並べ替えることができます。またダイアログ エディターでは、ダイアログまたはそのコントロールのプロパティを変更することもできます。InstallShield ダイアログ エディターは、他のリソースエディターと同様に機能します。

ダイアログ エディターを使ってダイアログのレイアウトや外観を変更するとき、現在のプロジェクトに含まれるダイアログのコピーを変更しているだけで、その他のプロジェクトには変更が反映されません。



タスク ***ダイアログ エディターにアクセスするには、以下の手順を実行します。***

- **[ダイアログ]** ビューで、右側のダイアログ プレビュー ペインに表示されるダイアログをクリックし、編集するダイアログの言語リンクをクリックします。
- **[ダイアログ]** ビューで、ダイアログを右クリックし **[編集]** をクリックします。

中央ペインにダイアログ エディターが表示されます。



メモ *エンド ユーザーがデスクトップ ディスプレイ テーマを使用している Windows プラットフォームでインストールを実行すると、そのテーマがインストールのエンドユーザー ダイアログを表示するときに使用されます。*

InstallScript または InstallScript MSI プロジェクトのダイアログにコントロールを追加する



プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield の [コントロール] ツールバーを使用して、コントロールをダイアログに追加することができます。



タスク ***ダイアログにコントロールを追加するには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、それから新しいダイアログ コントロールを追加するダイアログ アイテムを展開します。
3. 展開したダイアログ アイテムの下にある言語をクリックします。中央のペインにあるダイアログ エディターに、選択した言語のダイアログが表示されます。
4. **[コントロール]** ツールバーで、追加するコントロールをクリックします。
5. ダイアログ上のコントロールを表示する位置に長方形を描きます。
6. 右側のペインで、コントロールのプロパティを設定します。

[ダイアログ エディター] にコントロールを追加した後、ユーザーとコントロール間の対話を InstallScript を使って処理します。詳細については、「InstallScript を利用してダイアログ コントロールを処理する」を参照してください。



メモ: *Microsoft Visual Studio で作成されたプロジェクトの場合、ツールボックスを使用してダイアログ コントロールを追加できます。*

コントロールのプロパティを設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ダイアログ自体とそのすべてのコントロールには、ダイアログ エディター内にプロパティシートがあります。これらのプロパティでは、コントロールのサイズや位置、そのデフォルトのテキスト (文字列 ID) またはダイアログがモーダルかどうかを判別します。

プロパティシートの 1 つを編集するには、ダイアログ上でコントロールを選択するか、プロパティシート上方のボックスからコントロールまたはダイアログの名前を選択します。プロパティはコントロールの種類に応じてさまざまです。

- ダイアログ
- チェック ボックス
- プッシュ ボタン
- 編集フィールド
- コンボ ボックス
- テキスト領域
- リスト ボックス
- ラジオ ボタン
- ビットマップ
- グループ ボックス
- 線
- ラジオ ボタン グループ
- 選択ツリー
- 進行状況バー
- リスト ビュー
- アイコン

ビットマップの上部にコントロールを表示する

プロジェクト- この情報は、次のプロジェクト タイプに適用します:

- *InstallScript*
- *InstallScript MSI*

ビットマップその他のダイアログ コントロールのプロパティを設定するとき、コントロールがビットマップの上に配置されるように指定することができます。

タスク ***ビットマップ上にコントロールを表示するには、次の手順に従います:***

- ビットマップの **Tab Index** プロパティの値には、ビットマップの上に配置されるすべてのコントロールの **Tab Index** プロパティの値よりも小さい値を使用します。これを構成する最も簡単な方法は、ビットマップ コントロールの **Tab Index** プロパティを 0 に設定することです。
- すべてのコントロール (ビットマップを含む) では、**Tab Stop** のプロパティを **True** に設定しておく必要があります。

InstallScript を使用してカスタム ダイアログを実装する

プロジェクト- この情報は、次のプロジェクト タイプに適用します:

- *InstallScript*
- *InstallScript MSI*

カスタム ダイアログ作成の次のステップは、ダイアログ コントロールとのエンドユーザーの対話を処理する InstallScript 関数の作成です。

タスク ***InstallScript を使用して、InstallScript プロジェクトでカスタム ダイアログを実装するには、次の手順を実行します。***

1. 最初に、カスタム ダイアログのコードを含む InstallScript ソース ファイルの新規作成を行います。ファイルに **CustomDialog.rul** という名前を付けます。

   ヒント - *他のプロジェクトでダイアログ スクリプトを使用する場合は、.rul ファイルを別の場所にコピーします。他のプロジェクトで、**InstallScript** ビューを開き、**InstallScript** エクスプローラーで **[ファイル]** を右クリックして、**[スクリプト ファイルの挿入]** をクリックします。*

2. CustomDialog.rul 内で、カスタム ダイアログ関数用のプロトタイプと実装を配置します。次のコードを CustomDialog.rul に追加します。

```
prototype NUMBER CustomDialog( );
function NUMBER CustomDialog( )
begin
end;
```

3. スクリプトをコンパイルしたときに、次の行を Setup.rul 上部付近に挿入して、メインのスクリプトの Setup.rul に CustomDialog 関数のコードが含まれていること示します:

```
#include "CustomDialog.rul"
```

4. CustomDialog.rul スクリプトで、ダイアログに追加したコントロールの数値 ID を格納する定数を定義します。標準ダイアログから [戻る]、[次へ] および [キャンセル] ボタンをコピーした場合、CustomDialog.rulの上部付近に次の行を追加することができます。

```
// コントロール識別子
#define BUTTON_NEXT    1
#define BUTTON_BACK    12
```

一般に、ダイアログのコントロールの数値 ID は、コントロールの Control Identifier プロパティにリストされている数値で、ダイアログ エディターでコントロールを選択するとプロパティ リストに表示されます。



***重要・**エンド ユーザーが InstallScript ダイアログの右上にある閉じるボタンをクリックしてインストールをキャンセルできるようにするには、Control Identifier プロパティに値 2 を持つボタン コントロールをそのダイアログに含まなくてはなりません。このボタンはダイアログ内の任意の場所に配置でき、必要であればボタンを非表示にすることも可能です。これは、ダイアログにこの ID を持つボタンが含まれていない場合、インストールはダイアログの閉じるボタンがクリックされたときに生成されるメッセージを渡しません。*

たとえばチェック ボックス、編集フィールド、またはリストボックスといった、エンドユーザーが対話可能なその他の各コントロールに追加定数を定義する必要があります。

5. CustomDialog 関数は、**EzDefineDialog** 関数を使用して、カスタム ダイアログをメモリにロードします。

```
EzDefineDialog(
  "CustomDialog", // ダイアログのニックネーム
  ISUSER,         // ダイアログのリソースを含む DLL
  "CustomDialog", // [ダイアログ] ビューのダイアログの名前
  0);             // ダイアログの数値リソース ID (ここでは使用されない
```

EzDefineDialog と共に使う引数については、EzDefineDialog を参照してください。



***ヒント・**ダイアログ エディターで表示されるとおり、ダイアログのリソース ID はダイアログ名となります。数値リソース ID を指定する必要がある場合、カスタム ダイアログの Dialog テーブルにある **ISResourceId** 列に数値 ID を追加することができます。Dialog テーブルには、ダイレクト エディターを使ってアクセスできます。*

6. カスタム ダイアログのスクリプトでメッセージ ループを作成します。メッセージ ループは繰り返し **WaitOnDialog** 関数を呼び出します。この関数が呼び出されると、ユーザーがインタラクトするコントロールの数値コントロール ID が戻されます。通常のメッセージループは次のように表示されます。

```
// ユーザーが [次へ]、[戻る]、または [キャンセル] ボタンを使用してダイアログ ボックスを
// 終了するまで WaitOnDialog を繰り返し呼び出す
while (!bDone)

// ユーザーがコントロールと対話するのを待ち、
// コントロールの ID を戻す
nCtrl = WaitOnDialog("CustomDialog");

// switch ステートメントを使用して、異なるコントロールを処理する
```

```
switch (nCtrl)

    case CONTROL1:
        // ユーザーが CONTROL1 をクリックしたときにアクションを実行する

    case CONTROL2:
        // ユーザーが CONTROL2 をクリックしたときにアクションを実行する

    // その他のコントロール用の case ステートメント

endswitch;

endwhile;
```

たとえば、CustomDialog は現在 **[次へ]**、**[戻る]**、および **[キャンセル]** ボタンを含みます。そのメッセージループは次のように表示される可能性があります。

```
while (!bDone)

nControl = WaitOnDialog("CustomDialog");

switch (nControl)

    case DLG_INIT:
        // このダイアログの [戻る]、[次へ]、および [キャンセル] ボタンと有効/無効状態を
        // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
        // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
        // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
        hwndDlg = CmdGetHwndDlg("CustomDialog");
        SdGeneralInit("CustomDialog", hwndDlg, 0, "");

    case BUTTON_BACK:
        // ユーザーが [戻る] をクリック
        nReturn = BUTTON_BACK;
        bDone = TRUE;

    case BUTTON_NEXT:
        // ユーザーが [次へ] をクリック
        nReturn = BUTTON_NEXT;
        bDone = TRUE;


    デフォルト:
        // 標準のコントロール処理を確認します
        if (SdIsStdButton(nControl) && SdDoStdButton(nControl)) then
            bDone = TRUE;
        endif;

endswitch;

endwhile;
```

7. エンドユーザーが **[戻る]** または **[次へ]** をクリックすることによってダイアログを終了した場合、**EndDialog** と **ReleaseDialog** を使用して、画面とメモリからダイアログを削除します。

```
EndDialog("CustomDialog");
ReleaseDialog("CustomDialog");
```

メイン スクリプトでダイアログを使用するには、Setup.rul の **OnFirstUIBefore** イベント ハンドラーなど、CustomDialog 関数に呼び出しを追加します。

```
Dlg_SdWelcome:
    szTitle = "";
    szMsg   = "";
    nResult = SdWelcome(szTitle, szMsg);

Dlg_CustomDlg:
    nResult = CustomDialog( );
    if (nResult = BUTTON_BACK) goto Dlg_SdWelcome;

// 以下省略
```

エンドユーザーが **[戻る]** または **[次へ]** をクリックした場合、スクリプトは 1 つ前、または 1 つ後のダイアログを表示します。ユーザーが [**キャンセル**] をクリックした場合、標準の確認ダイアログが表示されます (**OnCanceling** イベント ハンドラーが処理)。

メモ- *ダイアログ コントロールの機能の実装に関する情報は、「InstallScript を使ったダイアログ コントロールの処理」を参照してください。*

特殊メッセージ

コントロール ID を返す以外にも、**WaitOnDialog** 関数はいくつかの特殊メッセージを返します。

- ダイアログが画面に表示される直前に、**WaitOnDialog** はメッセージ定数の DLG_INIT を返します。DLG_INIT case ステートメントでは、チェック ボックスやラジオ ボタンのデフォルトのステータスを設定したり、リストボックスやコンボボックス コントロールにデータを入力してデフォルトの選択を設定したり、編集フィールドの初期テキストを設定することができます。
- エラーが発生すると、**WaitOnDialog** は DLG_ERR 定数を返します。

InstallScript を利用してダイアログ コントロールを処理する

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallScript および InstallScript MSI インストール プロジェクトでは、InstallScript を使用して、カスタム ダイアログに追加したコントロールを処理します。

チェック ボックスコントロールの使用

ボタンのクリックを処理することのほかに、カスタム ダイアログは通常、チェック ボックスなどのダイアログ コントロールにおけるエンド ユーザーの選択を取得できる機能も必要です 。



メモ: InstallShield には、AskOptions という標準ダイアログがあります。これは最大 9 つまでのチェック ボックスやラジオボタンを表示するので、エンドユーザーにチェック ボックスを表示するためだけにカスタム ダイアログを作成する必要はありません。

押しボタンと同様に、ダイアログ ボックスに追加する各チェック ボックス コントロールに対し、通常は数値コントロール ID に象徴的な名前を割り当てる #define ステートメントをスクリプトに追加します。たとえば、チェック ボックス コントロールをカスタム ダイアログ ボックスに追加すると、コントロールの数値 ID はコントロールの Control Identifier プロパティに表示されます。数値 ID が 1302 の場合は、次の行をスクリプトに追加します。

```
#define MYCHECKBOX1 1302
```

ほとんどのタイプのコントロールに対し、InstallScript は **CtrlGet** 関数と **CtrlSet** 関数を定義します。これらの関数は、コントロールの現在の状態またはデータを取得あるいは設定するために使用できます。たとえば、**CtrlGetState** を使用するとチェック ボックス コントロールの現在の状態を取得でき、**CtrlSetState** を使用するとチェック ボックス コントロールの現在の状態を設定できます。ダイアログのメッセージ ループの DLG_INIT case で、**CtrlSetState** を呼び出してチェック ボックスの初期選択状態を設定することができます。次のコードは、チェック ボックスが予め選択された状態で表示されるように設定します。

```
case DLG_INIT:
    // このダイアログの [戻る]、[次へ]、および [キャンセル] ボタンと有効/無効状態を初期化して、
    // ID 700-724 および 202 の %P、%VS、%VI を IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、
    // および IFX_INSTALLED_DISPLAY_VERSION でそれぞれ置換します。
    hwndDlg = CmdGetHwndDlg("CustomDialog");
    SdGeneralInit("CustomDialog", hwndDlg, 0, "");
    CtrlSetState("CustomDialog", MYCHECKBOX1, BUTTON_CHECKED);
```

同様に、ダイアログのメッセージ ループの外に次のコードを追加すると、チェック ボックスの最終的な選択状態を検出することができます。

```
nState = CtrlGetState("CustomDialog", MYCHECKBOX1);
if (nState = BUTTON_CHECKED) then
    // チェックボックスが選択されている
else
    // チェックボックスが選択されていない
endif;
```

編集フィールドの使用

編集コントロールをカスタム ダイアログに追加することもできます。これにより、エンドユーザーは 1 行のテキストを入力できるようになります。InstallShield には、**SdShowDlgEdit1**、**SdShowDlgEdit2**、および **SdShowDlgEdit3** と呼ばれる標準ダイアログがあります。これは、エンドユーザーがテキストを入力できる 1、2、または 3 つの編集フィールドを表示します。

編集コントロールに格納されているテキストは **CtrlGetText** を使用して読み取ることができます。たとえば、リソース ID が 10000 の編集コントロールからテキストを読み取って svEdit という文字列変数に格納するには、次のコードを使用します。

```
CtrlGetText("CustomDialog", 10000, svEdit);
```

同様に、**CtrlSetText** を使用すると、編集コントロールに格納される (メッセージ ループの switch ステートメントの DLG_INIT ブロック内) 初期のテキストを設定することができます。



*ヒント: 編集コントロールの**パスワード**属性を利用して、編集フィールドにエンドユーザーが入力する文字を非表示に設定することができます。**番号**属性を利用して、編集フィールドにエンドユーザーが数値のみを入力できるように設定することができます。*

リストボックスとコンボボックスのコントロールの使用

リストボックスとコンボボックスのコントロールは LIST タイプの変数と関連付ける必要があります。リストボックスのコントロールを表示する前に、文字列リスト変数にデータを入力し、ダイアログの DLG_INIT ハンドラーで **CtrlSetList** を使用してリストボックスまたはコンボ ボックスのコントロールと関連付ける必要があります(コンボ ボックスのコントロールに対しては、通常 Drop-Down List プロパティを True に設定し、Height プロパティをコントロールのドロップダウンリスト部分の高さに設定します)。

リスト ボックスまたはコンボ ボックスのコントロールに初期選択を設定するには、ダイアログのDLG_INIT ハンドラーで **CtrlSetCurSel** を呼び出します。また、ユーザーの現在の選択を取得するには、**CtrlGetCurSel** を呼び出します。たとえば、次のコードは使用可能な各ドライブの文字をリスト変数に入力し (GetValidDrivesList を使用)、リストをコンボ ボックス コントロールに関連付け、ダイアログを終了する前にコンボ ボックスからエンドユーザーの選択を読み取ります。

```
function NUMBER CustomDialog( )
  NUMBER nReturn;
  NUMBER nControl;
BOOL   bDone;
       // コンボボックスリストと現在の選択のための変数
  LIST listDrives;
  STRING svDrive;
begin
    nReturn = EzDefineDialog("CustomDialog", ISUSER, "CustomDialog", 0);

    bDone = FALSE;

    // コンボボックス項目を含むリストの作成
    listDrives = ListCreate(STRINGLIST);
    // 使用可能な全ドライブ名をリストに入力
    GetValidDrivesList(listDrives, -1, -1);

    while (!bDone)

    nControl = WaitOnDialog("CustomDialog");

    switch (nControl)

        case DLG_INIT:
            // このダイアログの [戻る]、[次へ]、および [キャンセル] ボタンと有効/無効状態を初期化して、
            // コントロール ID 700-724 および 202 上で %P、%VS、%VI を
            // IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION および IFX_INSTALLED_DISPLAY_VERSION で
            // それぞれ置換します。
            hwndDlg = CmdGetHwndDlg("CustomDialog");
            SdGeneralInit("CustomDialog", hwndDlg, 0, "");
            CtrlSetState("CustomDialog", MYCHECKBOX1, BUTTON_CHECKED);
            // コンボボックスとリストの関連付け
            CtrlSetList("CustomDialog", MYCOMBOBOX1, listDrives);
            // リストからの最初のドライブ名の取得...
            ListGetFirstString(listDrives, svDrive);
```

```
            // ...そしてそれを現在の選択に設定
            CtrlSetCurSel("CustomDialog", MYCOMBOBOX1, svDrive);
        // その他のコントロール用の case ステートメント

    endswitch;

    endwhile;

    // エンドユーザーの選択を取得し、メッセージボックスに表示
    CtrlGetCurSel("CustomDialog", MYCOMBOBOX1, svDrive);
    MessageBox("次のドライブを選択しました: " + svDrive, INFORMATION);

    EndDialog("CustomDialog");
    ReleaseDialog("CustomDialog");

    return nReturn;
end;
```

リスト ボックスとコンボ ボックス コントロールは Sorted プロパティがあります。これを [はい] に設定すると、リスト コントロールの内容をアイテムの表示名 別に並べ替えることができます。

InstallScript または InstallScript MSI プロジェクトでダイアログの動作を編集する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ダイアログの動作を変更するには、希望する動作に応じてダイアログ関数のパラメーターを変更します。利用可能なパラメーターについては、ダイアログ関数に関するドキュメントを参照してください:

- ダイアログ関数
- ダイアログのカスタマイズ関数

"Sd ダイアログ スタティック テキスト" フィールドに製品名、製品バージョン、または、インストール済みバージョンを含むフィールドを追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield では、Sd ダイアログ スタティック テキスト フィールドに、%P、%VS、または %VI (場合により、追加スペースとして表示されます)というプレースホルダーを含む製品名、製品バージョン、または、インストールされたバージョンをグローバルに配置することができます。実行時に、システム変数 IFX_PRODUCT_DISPLAY_NAME, IFX_PRODUCT_DISPLAY_VERSION と IFX_INSTALLED_DISPLAY_VERSION の値がスタティック テキスト フィールドで %P、%VS、および %VI の代わりに表示されます。

プレース フォルダーを含むスタティック テキスト フィールドを追加する場合、701 から 799 の範囲内で (そのダイアログに対して) 一意の ID をスタティック テキスト フィールドに与えます。ID が 701 から 799 のとき、InstallShield はスタティック テキスト フィールドをスキャンしてプレースホルダーの存在を確認します。プレースホルダーが見つかると、InstallShield はそれを対応するシステム変数の値で置き換えます。

ダイアログで HTML コントロールを使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield では、InstallScript と InstallScript MSI プロジェクトに含まれるダイアログで HTML コントロールをサポートします。HTML コントロール を使って、ダイアログ コントロールで HTML マークアップを使用できます。ダイアログで Web ページ、インストール済みの HTML ファイル、および HTML サポートファイルへのリンクを含めることができます。エンド ユーザーが実行時のダイアログでハイパーリンクをクリックしたときに、インターネット ブラウザーで HTML ページを開くか、InstallScript コードを使って定義した別の動作をトリガーできます。HTML コントロールを使って、外観を制御するためのスタイルを含む、有効な HTML マークアップを使用できます。

HTML コンテンツがインストール済みの HTML ファイルまたは HTML サポート ファイルである場合、HTML コントロールを使って、それを直接ダイアログに表示することも可能です。

HTML コントロールを作成するには、まずスタティック テキスト コントロールを作成します。ダイアログにスタティック テキストコントロールを配置してから、それを HTML コントロールに変換できます。



タスク

ダイアログの任意のスタティック テキスト コントロールを HTML コントロールに変換するには、以下の手順の 1 つを実行します:

- [ダイアログ] ビューで、コントロールの Text プロパティを次のとおりに設定します:

  [html] *HTML マークアップ*

- InstallScript を使って、コントロールのテキストを **[html]** に設定して、後に HTML マークアップを続けます。

実行時に、InstallScript エンジンがコントロールを HTML コントロールに変換します。

例

次の例では、スタティック テキスト コントロールを HTML コントロールに変換する方法をデモンストレーションします。サンプル HTML コードを使って、コントロールの色とフォントを設定して、ダイアログと一致させます。



タスク

コントロールの Text プロパティを使って、スタティック テキスト コントロールを HTML コントロールに変換するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、HTML コントロールを含めるダイアログを右クリックして、**[編集]** をクリックします。ダイアログ エディターに、プロジェクトのデフォルト言語でダイアログが表示されます。

3. HTML コントロールに変換するスタティック テキスト コントロールをクリックします。InstallShield の右側に設定が表示されます。
4. **Text** プロパティで、省略記号ボタン (...) をクリックしますプロジェクトに含まれるすべての現在の文字列のリストが表示されます。
5. **[編集]** ボタンをクリックします。
6. 文字列 ID の値を次のように設定します:

   [html]<style type="text/css">html,body {padding:0; margin:0; background-color:ButtonFace} * {font-size: 8pt; font-family: "MS Sans Serif";} </style> Web サイトにアクセス

実行時、インストールはデフォルトの InstallScript ダイアログ フォント (8 ポイント MS Sans Serif) を使い、背景色を Windows ダイアログの色 (通常は灰色) に設定します。コンテンツが白色のダイアログ領域にある場合、*background-color:ButtonFace* スタイル部分を省略して、HTML コントロールの背景が白色になるようにします。



ヒント: *[ダイアログ] ビューを通してスタティック コントロールを変換する代わりに、InstallScript ビューでダイアログのスクリプト関数を編集することもできます。たとえば、ダイアログ メッセージ ループの DLG_INIT ケースに次のコードを追加します:*

*CtrlSetText(szDlg, HTML_CTRL_ID, "[html]<style type=¥"text/css¥">html,body {padding:0; margin:0; background-color:ButtonFace} * {font-size: 8pt; font-family: "MS Sans Serif";} </style>Web サイトにアクセス site");*

その他のスクリプト サンプルは、「 CtrlGetUrlForLinkClicked Example」を参照してください。

HTML コントロールにあるリンクのクリック イベントの処理方法

CtrlSetText 関数を使って HTML コントロール コンテンツを実行時に設定して、コントロールの作成に成功すると、**CtrlSetText** が 0 を返します。失敗すると、**CtrlSetText** が ISERR_GEN_FAILURE を返します。

HTML コントロールにあるリンク イベントを処理するため、**WaitOnDialog** 関数は、HTML コントロールのコントロール ID を返します (これは、元のスタティック テキスト コントロールのコントロール ID と同じです)。カスタム ダイアログ スクリプト関数に HTML コントロール ID を処理する case ステートメントを含めて、**CtrlGetUrlForLinkClicked** 関数を呼び出してクリックされたリンクの URL を取得できます。スクリプトはこのリンクを使って、たとえばインターネット ブラウザーを起動してリンクにナビゲートするなどの任意のアクションを行うことができます。

エンド ユーザーがアンカー タグを使った HTML コントロールのリンクをクリックすると、次の動作が発生します:

- アンカー タグにターゲット属性が含まれていない、またはターゲット値が新しいウインドウに表示されない場合、コントロール ID がダイアログ スクリプトに戻ります。これによって、スクリプトがリンク クリック自体を処理できます。
- アンカー タグにターゲット属性が含まれていて、その値が新しいウィンドウに表示される _blank またはそれに等しい値であるとき、メッセージはスクリプトに送られず、URL が自動的に起動されます。

HTML コントロールに関する特別考慮

ダイアログに HTML コントロールを追加する場合は、以下の詳細に注意してください。

HTML コントロールの実行時の要件

HTML コントロールは、完全に機能するためには Internet Explorer 5.5 以降を条件とします。ターゲット システムに以前のバージョンが存在するとき、HTML コントロールのコンテンツを設定する唯一の方法は、HTML ファイルを **CtrlSetText** に提供する方法です。ターゲット システム上で、Internet Explorer がデフォルトのブラウザーである必要はありません。

HTML ファイルの HTML コントロール サポート

HTML コントロールで HTML ファイルを使用する場合、ファイルへのフル パスを **CtrlSetText** 関数を使って指定しなくてはなりません。そうでない限り、InstallScript エンジンがファイルの場所を認識できません。サンプル InstallScript は次の通りです:

```
CtrlSetText(szDlg, HTML_CTRL_ID, "[html]file://" + SUPPORTDIR^"aboutpage.htm");
```

電子メール アドレスに HTML コントロールを使用する

電子メール アドレスをポイントするハイパーリンクを追加するには、次のようなコードを使用します:

```
CtrlSetText(szDlg, HTML_CTRL_ID, "[html]<a href=¥"mailto:support@mycompany.com?Subject=Example%20Line¥">サポートに電子メールを送信</a>");
```

エンド ユーザーがこのようなリンクをクリックしたとき、ターゲット システムの電子メール アプリケーションで新しい電子メール メッセージが自動的に開きます。mailto リンクでは、スクリプトにメッセージは送信されませんので、ご注意ください。

フォーマットされていない HTML テキスト コンテンツ

HTML コントロールのコンテンツ セットはそのままの形式で渡されます。コンテンツのフォーマットは行われません。コンテンツをフォーマットする必要がある場合、スクリプトでコンテンツ文字列を操作してからコンテンツ文字列を **CtrlSetText** に渡します。

スタティック テキスト コントロールのテキスト プロパティ文字数制限

スタティック テキスト コントロールのテキスト プロパティの文字数制限は 256 文字です。したがって、コントロールのテキスト コンテンツの始めに **[html]** を追加してスタティック テキスト コントロールを HTML コントロールに変換する場合、入力可能な最大文字数は 256 文字です。これにはハイパーリンク テキストおよび指定する HTML マークアップ文字のすべてが含まれます。コンテンツに 256 文字以上を入力するには、**CtrlSetText** 関数を使ってコントロールを作成します。

デフォルトのダイアログに戻す



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ダイアログを編集してから、後で変更内容をすべて元に戻す場合、元のデフォルト ダイアログに復元することができます。

タスク ***デフォルトのダイアログに元に戻すには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. 編集したアクションを右クリックして、**[元に戻す]** をクリックします。

リソース コンパイラとリソース リンカ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallScript または InstallScript MSI プロジェクトでダイアログに変更を加えるには、リソース コンパイラとリソース リンカをシステムにインストールしておく必要があります。InstallShield には、これらのタイプのツールが両方搭載されています。

タスク ***リソース コンパイラとリソース リンカの位置の確認は、次の手順で行います。***

1. **[ツール]** メニューから **[オプション]** を選択します。**Options** ダイアログ ボックスが開きます。
2. **[リソース]** タブをクリックします。

[リソース] タブには、コンパイラとリンカの場所の一覧があります。

タスク ***リソースコンパイラまたはリソースリンカを既にシステム上にあるものと置き換えるには、以下の手順に従います:***

1. **[ツール]** メニューから **[オプション]** を選択します。**Options** ダイアログ ボックスが開きます。
2. **[リソース]** タブをクリックします。
3. **[リソースのコンパイラの場所]** ボックス、または、**[リソースのリンカの場所]** ボックスの横にある [参照] ボタンをクリックして、プログラム ファイルまで移動します。

ダイアログ サンプラー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ダイアログ サンプラーは、InstallScript と InstallScript MSI プロジェクトにおけるビルトイン ダイアログのサンプル、および、スクリプト ダイアログ (sd) 関数によって呼ばれるダイアログを表示する InstallShield ウィザードです。異なる 2 つの ダイアログ サンプラーを起動することができます。



タスク ***ダイアログ ウィザードを起動するには、以下の手順を実行します。***

InstallShield の **[ツール]** メニューで **InstallScript** をポイントして、**[標準ダイアログ サンプラー]** または **[スキン ダイアログ サンプラー]** をクリックします。

[プログラム] メニューにある InstallShield プログラムフォルダーのショートカットを使用して、これらのサンプラーを起動することもできます。

ダイアログ スキン



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

概要

ダイアログ スキンを使うと、エンドユーザー ダイアログを異なる外見とカラー スキームを使って表示できます。スキンされたダイアログは標準ダイアログよりやや大きめで、コントロールの場所も異なります。スキン付きダイアログを適切に表示するには、ターゲットマシンのデスクトップ領域が 800 x 600 ピクセル以上 (大きなフォントが利用されている場合は 1024 x 768 ピクセル) で、システムが 16-bit 色以上を表示可能でなくてはなりません。

制限

プロジェクトでダイアログのスキンを使用する場合、いくつか制約があります。

- 標準ダイアログを [ダイアログ エディタ] 内で一度編集すると、そのダイアログにダイアログ スキンを適用または削除することはできません。したがって、編集するダイアログのスキンを指定するときは、まずスキンを指定してから、編集してください。
- 標準ナビゲーション ボタン ([次へ]、[キャンセル]、[戻る]、および [終了]) の場所はすべてのスキンで共通です。.skin ファイルはその位置を制御します。ダイアログ エディターで移動しても、ランタイムの位置に影響はありません。また、いくつかのダイアログでは [参照] ボタン (例 **SdAskDestPath**) を再配置できない場合があります。
- スキン ダイアログではウィンドウ ビルボードを使用できません。ウィンドウ スタイルのビルボードについての詳細は、「InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルおよびファイルの種類」を参照してください。
- プロジェクトにカスタム ダイアログを追加する場合、新しいダイアログは、他のエンドユーザー ダイアログと同じサイズである必要があります。カスタム ダイアログのサイズが異なる場合、正しく表示されないことがあります。たとえば上記の位置関係の問題により、ナビゲーション ボタンが実行時にエンドユーザーに表示されないことがあります。



メモ: *カスタム ダイアログにダイアログ スキンを指定する場合、スキンが適切に表示されるためにダイアログ エディターでカスタム ダイアログを作成する前にスキンの設定を行う必要があります。*

ダイアログ スキンの指定



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ダイアログ スキンを使用して、インストールのエンドユーザー ダイアログの外観を変更することができます。1 つのプロジェクトにつき、1 つのスキンを指定することができます。プロジェクト内のすべてのダイアログは、選択したスキンを利用して表示されます。



メモ- *標準ダイアログを [ダイアログ エディタ] 内で一度編集すると、そのダイアログにダイアログ スキンを適用または削除することはできません。したがって、編集するダイアログのスキンを指定するときは、まずスキンを指定してから、編集してください。*

カスタム ダイアログにダイアログ スキンを指定する場合、スキンが適切に表示されるためにダイアログ エディターでカスタム ダイアログを作成する前にスキンの設定を行う必要があります。

ダイアログ スキンの選択



タスク ***ダイアログ スキンを選択するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[スキン]** を開いて利用可能なダイアログ スキンオプションを表示します。
3. スキンをクリックして、右側のパネルにプレビュー ダイアログを表示します。
4. 表示されたスキンを選択するには、プレビュー ペインの **[選択]** をクリックします。[**ダイアログ**] エクスプローラーで、選択したスキンのアイコンに赤いチェック記号が表示されます。

Blue スキン以外のすべてのダイアログ スキンは、フェード グラフィックに .gif ファイルを利用します。Blue スキンは .bmp ファイルを利用するのでファイルサイズが大きくなります。.gif ファイルを利用したとき、16 ビット カラーシステム上での Blue スキンの外観はその他のカラーに比べると鮮明ではありません。16-bit カラーシステムをサポートする Blue スキンを利用する場合で、ファイルサイズの大きさが関係ない場合は、Blue オプションを選択して下さい。より小さいサイズには、グラフィックに .gif ファイルを利用する BlueTC (True Color) オプションを選択してください。

ダイアログスキンの選択をクリアする



タスク ***ダイアログ スキンの選択をクリアするには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[スキン]** アイテムを開いて利用可能なダイアログ スキンオプションを表示してから、**<なし>** アイテムを選択します。
3. プレビュー パネルの **[選択]** をクリックします。

基本の MSI プロジェクトでダイアログを使用する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

このセクションでは、基本の MSI プロジェクトで、ダイアログがある他の基本的な関数の作成および実行方法が説明されています。



タスク

新しいダイアログを作成して、それをインストールで表示するプロセスは、次のように複数のタスクに分割できます:

1. ダイアログをプロジェクトに追加する。
2. ダイアログのレイアウトを編集する。
3. コントロールの動作を定義する (表示する条件、インタラクションがトリガーするイベント、およびサブスクライブするイベント)。
4. ダイアログを表示する (別のダイアログのコントロール内の NewDialog または SpawnDialog イベントを使用するか、またはダイアログをプロジェクトのシーケンスに挿入します)

基本の MSI インストール中にダイアログを表示する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

次に説明するいずれかの方法で、エンドユーザー ダイアログを表示できます。

- [ユーザー インターフェイス] シーケンスにダイアログを挿入する。
- 別のダイアログの動作を介してダイアログを起動する。

シーケンスへダイアログを挿入する



プロジェクト: *マージ モジュール内でダイアログを作成する場合、ダイアログをシーケンスに挿入するにはモジュールをインストール プロジェクトと関連付ける必要があります。*



タスク

ダイアログをシーケンスにスケジュールするには、以下の手順に従います:

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[シーケンス]** エクスプローラーで、ダイアログが通常表示される 3 つの上位シーケンス (**[インストール]**、**[アドバタイズ]**、または、**[管理]**) から 1 つを展開して、その **[ユーザー インターフェイス]** シーケンスを表示します。
3. **[ユーザー インターフェイス]** シーケンスを展開して、既存のアクション、ダイアログ、およびカスタム アクションの順序付けを確認します。

4. シーケンス内で作業中のダイアログの前にくるアクションまたはダイアログを右クリックして、**[挿入]** をクリックします。[アクションの挿入] ダイアログ ボックスが開きます。
5. リストで、**ダイアログ**を選択します。
6. ダイアログのリストから、新しいダイアログを選択します。
7. **[OK]** をクリックします。

インストールが完全ユーザー インターフェイス (/q コマンドライン パラメーターで定義) で実行され、このダイアログに対するすべての条件が満たされると、ダイアログが選択したシーケンス内に表示されます。

InstallShield でダイアログを [実行] シーケンスに挿入できても、挿入すると ICE13 違反になります。

ダイアログを別のダイアログの動作を介して起動する

ほとんどのダイアログは、NewDialog (ウィザードで別のダイアログを表示する) または SpawnDialog (モデルダイアログを表示する) コントロールイベントの結果として表示されます。ダイアログの動作 (通常は [次へ] ボタン) でこのイベントを選択することによって、イベントを正常に実行するダイアログを指定できます。コントロールイベントに条件を追加することによって、直前のダイアログでの選択の結果に依存して別々のダイアログを表示できます。

タスク ***たとえば、InstallWelcome ダイアログの後に WelcomeBitmap ダイアログを表示するには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**InstallWelcome** ダイアログを拡張してから、そのダイアログの **[動作]** サブノードをクリックします。
3. コントロール リストおよび対応するコントロールの種類を含むペインで、**[次へ]** ボタンを選択します。
4. **"イベント"** 設定で、**[新しいイベント]** ボタンをクリックしてから **[NewDialog]** をクリックします。**"イベント"** 設定の下に、いくつかの **"NewDialog"** サブ設定が追加されます。
5. **"NewDialog"** 設定に、次のように入力します: **1**

   1 の条件は、ダイアログがすべての条件の下に作成されなくてはならないことを示します。
6. **"ダイアログ名"** 設定で、次を入力します: **WelcomeBitmap**

メモ *InstallWelcome ダイアログが [次へ] ボタンから別のダイアログを起動した場合は、上記の手順を繰り返して WelcomeBitmap の [次へ] ボタンで次のダイアログを表示するようにします。*

[カスタム アクションとシーケンス] ビューで新しいダイアログを表示する詳しい方法については、「[カスタム アクションとシーケンス] ビューでエンドユーザー ダイアログの順序の表示する (基本の MSI プロジェクト)」を参照してください。

基本の MSI プロジェクトで新しいダイアログを作成する

プロジェクト *この情報は、基本の MSI プロジェクトに適用します。*

タスク ***新規で基本の MSI プロジェクトを作成するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** を右クリックしてから、**[新しいダイアログ]** をクリックします。ダイアログ ウィザード が起動し、新規ダイアログ作成手順を案内します。

次のステップでは、ダイアログのレイアウトおよびコントロールの動作を編集します。

ヒント: *ダイアログを別のプロジェクトからインポートして追加することもできます。*

エンドユーザー インターフェイスでダイアログを表示する

プロジェクトにダイアログを追加しても、そのダイアログがインストールに表示されるわけではありません。1 つ以上のプロジェクトのシーケンスにダイアログを挿入する処理については、「基本の MSI インストール中にダイアログを表示する」を参照してください。

基本の MSI プロジェクトでダイアログ レイアウトを編集する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

ダイアログ エディターを使用して、ダイアログの修正、コントロールの編集、およびダイアログとコントロールの表示プロパティの設定をすべてビジュアル環境で行うことができます。

メモ: *エンド ユーザーがデスクトップ ディスプレイ テーマを使用している Windows プラットフォームでインストールを実行すると、そのテーマがインストールのエンドユーザー ダイアログを表示するときに使用されます。*

[ダイアログ] ビューは、各プロジェクトのサポート言語のダイアログのバージョンを管理します。レイアウトを編集するには、言語固有のバージョンを選択しなければなりません。コントロールのサイズに加える変更を除き、これらのバージョンは変わりません。詳細については、「各言語ごとのダイアログを変更する」を参照してください。

ダイアログ エディターを開く

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

タスク ***ダイアログ エディターでダイアログを開くには、次のいずれかを実行します。***

- **[ダイアログ]** ビューの中のダイアログの名前の下にある、ダイアログの言語固有のバージョンを選択します。
- **[カスタム アクションとシーケンス]** ビューで、ダイアログを右クリックして、**[レイアウトの編集]** をクリックします。

ダイアログ エディターは、[ダイアログ] ビューの右ペインで開きます。

基本の MSI プロジェクトでダイアログにコントロールを追加する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield の [コントロール] ツールバーを使用して、コントロールをダイアログに追加することができます。



タスク ***ダイアログにコントロールを追加するには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、それから新しいダイアログ コントロールを追加するダイアログ アイテムを展開します。
3. 展開したダイアログ アイテムの下にある言語をクリックします。中央のペインにあるダイアログ エディターに、選択した言語のダイアログが表示されます。
4. **[コントロール]** ツールバーで、追加するコントロールをクリックします。
5. ダイアログ上のコントロールを表示する位置に長方形を描きます。
6. 右側のペインで、コントロールのプロパティを設定します。



ヒント: *Microsoft Visual Studio 内から InstallShield プロジェクトを編集している場合、ツールボックスを使用してコントロールを追加できます。*

コントロールの設定を構成する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

ダイアログ自体とそのすべてのコントロールには、ダイアログ エディター内に設定のグリッドが表示されます。これらの設定では、コントロールのサイズや位置、そのデフォルトのテキスト (文字列 ID など) またはダイアログがモーダルかどうかなどの特徴を判別します。

コントロールの設定を編集するには、ダイアログ上でコントロールを選択します。設定は、コントロールの種類に応じてさまざまです。

- ビットマップ
- ダイアログ
- チェック ボックス
- プッシュ ボタン
- 編集フィールド

- コンボ ボックス
- テキスト領域
- リスト ボックス
- ラジオ ボタン
- ビットマップ
- グループ ボックス
- ビルボード
- 線
- ラジオ ボタン グループ
- 選択ツリー
- 進行状況バー
- リスト ビュー
- スクロール可能テキスト
- アイコン
- ディレクトリ リスト
- ディレクトリ コンボ
- ボリューム コスト リスト
- ボリューム選択コンボ
- マスク編集
- パス編集

ビットマップ上にコントロールを表示する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*



タスク ***ビットマップ上にコントロールを表示するには、次の手順に従います:***

- ビットマップの **Tab Index** プロパティの値には、ビットマップの上に配置されるすべてのコントロールの **Tab Index** プロパティの値よりも小さい値を使用します。これを構成する最も簡単な方法は、ビットマップ コントロールの **Tab Index** プロパティを 0 に設定することです。
- すべてのコントロール (ビットマップを含む) では、**Tab Stop** のプロパティを **True** に設定しておく必要があります。

基本の MSI プロジェクトでダイアログ レイアウトを編集する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

[ダイアログ] ビューで、実行時のコントロールの動作を指定する設定を構成できます。コントロールの動作関連の設定では、次の種類の機能を構成できます:

- コントロールで対話するエンドユーザーが起動すイベント
- コントロールがサブスクライブする必要がある Windows Installer のイベント
- コントロールを表示する条件



タスク ***コントロールの動作設定にアクセスするには、以下の手順に従います:***

1. 以下のいずれかを実行します。
   - **[ダイアログ]** ビューで、変更するダイアログのダイアログ ノードを拡張してから、**[動作]** サブノードをクリックします。
   - **[カスタム アクションとシーケンス]** ビューで、ダイアログを右クリックして、**[動作の編集]** をクリックします。
2. 選択されたダイアログのすべてのコントロールの一覧が表示される中央ペインで、構成するコントロールをクリックします。設定が、右のペインに表示されます。
3. 必要に応じて **"イベント"**、**"サブスクリプション"**、および **"条件"** 設定を構成します。

基本の MSI ダイアログでコントロール イベントを使用する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

コントロール イベントを使うと、ダイアログの各コントロールに対してカスタム機能を提供することができます。コントロール イベントでは、カスタム アクションの起動や新しいダイアログの生成、また進行状況バーの表示などを行うことができます。多くのコントロール イベントは、パブリッシュされた情報を使用しています。また、情報を使用するには、コントロール イベント情報を サブスクライブしなければならないコントロールもあります。

イベントとコントロールを関連付けるには、[ダイアログ] ビューでダイアログの [動作] 領域を使用します。詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」を参照してください。コントロールのサブスクリプションに関する詳細は、「基本の MSI ダイアログでサブスクリプションを使用する」を参照してください。

このイベントの条件を指定するには、"イベント" 設定で省略記号ボタン (...) をクリックします。

次は使用可能なコントロール イベント一覧です:

テーブル 3 ・ コントロールのイベントの種類

コントロール イベント	説明
AddLocal	指定の機能 (または、**[すべて]** が指定された場合にはすべての機能) をローカルにインストールすることを設定します。 機能を指定するには、この設定のサブ設定を構成してください。
AddSource	指定の機能 (または、**[すべて]** が指定された場合にはすべての機能) をソースからインストールすることを設定します。 機能を指定するには、この設定のサブ設定を構成してください。
CheckExistingTargetPath	指定のパスが書き込み可能であることを検証し、また指定のパスが存在しない限り、追加コントロールイベントがトリガされないように防ぎます。 パスを含むプロパティを指定するには、この設定のサブ設定を構成します。
CheckTargetPath	指定された値が有効なパスでない場合、追加コントロール イベントがトリガされないように防ぎます。 パスを含むプロパティを指定するには、この設定のサブ設定を構成します。
DirectoryListNew	ディレクトリ リスト コントロールに、新規フォルダーの作成が必要であることを通知します。インストーラーは新しいフォルダーを作成すると、ユーザーはフォルダーに新しい名前を付けることができます。
DirectoryListOpen	ディレクトリ リスト コントロールに、選択されたディレクトリを開くことを通知します。ディレクトリが選択されていない場合、追加のコントロール イベントがトリガされないように防ぎます。
DirectoryListUp	ディレクトリ リスト コントロールに、1 階層上へ移動して現在のディレクトリの親を選択することを通知します。
DoAction	選択されたコントロールがアクティブになると、カスタム アクションをトリガします。 カスタム アクションを指定するには、この設定のサブ設定を構成してください。
EnableRollback	インストールのロールバック機能を有効または無効にします。 イベントがロールバックを有効にするかどうかを指定するには、この設定のサブ設定を構成してください。True の値はロールバックを有効にし、False はロールバックを無効にします。

テーブル 3 ・ コントロールのイベントの種類 (続き.)

コントロール イベント	説明
EndDialog	モーダル ダイアログを閉じます。このイベントは、プッシュ ボタン コントロールまたは選択ツリー コントロールに使用できます。 このイベントで発生させる動作を指定するには、この設定のサブ設定を構成してください。 標準ダイアログで使用可能なオプション: • **終了** – ダイアログ シーケンスが閉じると、コントロールによって UserExit の値がインストーラーに返されます。このオプションは、子ウィンドウがある場合は使用できません。 • **再試行** – ダイアログ シーケンスが閉じると、コントロールによって Suspend の値がインストーラーに返されます。このオプションは、子ウィンドウがある場合は使用できません。 • **無視** – ダイアログ シーケンスが閉じると、コントロールによって Finished の値がインストーラーに返されます。このオプションは、子ウィンドウがある場合は使用できません。 • **戻す**–ダイアログが終了して、コントロールが親ウィンドウに戻ります。親ウィンドウが存在しない場合、コントロールによって Success の値がインストーラーに返されます。 エラー ダイアログで使用可能なオプション: • ErrorOk • ErrorCancel • ErrorAbort • ErrorRetry • ErrorIgnore • ErrorYes • ErrorNo

テーブル 3 ・ コントロールのイベントの種類 (続き.)

コントロール イベント	説明
MsiLaunchApp	 ***メモ・*** *Windows Installer 5 で、このイベントがサポートされています。以前のバージョンの Windows Installer はこのイベントを無視します。したがって、Windows Installer 4.5 以前が搭載されているシステム上でインストールを実行するときに、このイベントを使用する場合、ダイアログ コントロールに条件を追加することで、Windows Installer 4.5 以前が搭載されたシステム上では、これが表示されないようにします。* 指定されたファイルを実行します。 通常このイベントは、SetupCompleteSuccess ダイアログ上のチェック ボックス コントロールに使用します。エンド ユーザーはチェック ボックスを使って、インストールの終わりにアプリケーションを実行するかどうかを選択できます。チェック ボックス コントロールには、アンインストール中にコントロールが表示されるのを防ぐ条件を含みます。 ファイルを起動するコマンドラインを指定するには、この設定のサブ設定を構成します。
MsiPrint	 ***メモ・*** *Windows Installer 5 で、このイベントがサポートされています。以前のバージョンの Windows Installer はこのイベントを無視します。したがって、Windows Installer 4.5 以前が搭載されているシステム上でインストールを実行するときに、このイベントを使用する場合、ダイアログ コントロールに条件を追加することで、Windows Installer 4.5 以前が搭載されたシステム上では、これが表示されないようにします。* エンド ユーザーがコントロールの内容を印刷できるようにします。 このイベントを、スクロール可能なテキスト コントロールを含むダイアログにあるプッシュ ボタン コントロールに追加できます。エンド ユーザーがプッシュ ボタン コントロールをクリックすると、スクロール可能なテキスト コントロールの内容が印刷されます。
NewDialog	別のダイアログに進みます。 新しいダイアログを指定するには、この設定のサブ設定を構成してください。
Reinstall	指定の機能 (または、**[すべて]** が指定された場合にはすべての機能) を再インストールします。 機能を指定するには、この設定のサブ設定を構成してください。
Remove	指定の機能 (または、**[すべて]** が指定された場合にはすべての機能) の削除が選択されていることを示します。 機能を指定するには、この設定のサブ設定を構成してください。

テーブル 3 ・ コントロールのイベントの種類 (続き.)

コントロール イベント	説明
Reset	ダイアログのすべてのコントロールが行なったプロパティの変更を元の値にリセットします。
RMShutdownAndRestart	再起動マネージャーが使用中のファイルを含むすべてのアプリケーションをシャットダウンし、インストールの終わりにそれらを再起動させることを示します。
SelectionBrowse	エンド ユーザーが強調表示されたアイテムのパスを変更できるようにする、指定された [参照] ダイアログを起動します。 ダイアログを指定するには、この設定のサブ設定を構成してください。
SetInstallLevel	INSTALLLEVEL プロパティを指定の値に設定します。 適切なインストール レベルを指定するには、この設定のサブ設定を構成してください。
SetProperty	プロパティの値を設定します。 プロパティの値を指定するには、この設定のサブ設定を構成してください。
SetTargetPath	選択されたパスを確認および設定します。 パスを含むプロパティを指定するには、この設定のサブ設定を構成します。
SpawnDialog	現在のダイアログを閉じずに、指定されたモーダルダイアログを起動します。
SpawnWaitDialog	条件式が True 評価されるまで待機するダイアログを起動してから、ダイアログを閉じます。
ValidateProductID	ProductID プロパティを設定します。

基本の MSI ダイアログでサブスクリプションを使用する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

Windows Installer サービスは、実行時にインストールに多くの情報を提供します。インストール中の最も便利な情報収集方法はサブスクリプションです。サブスクライブとは、一般的に、パブリッシュまたはサービスに自分の名前を入力することと定義づけられます。Windows Installer では、アクションが通常情報をパブリッシュし、ダイアログ コントロールがその情報をサブスクライブします。

サブスクリプションの一般的な使用例

この関係を表す一番いい例は、進行状況バーです。InstallFiles アクションは、移動したファイルの割合と、残りのファイルの割合に関する情報をパブリッシュします。進行状況バーがこの情報をサブスクライブすると、インストールのファイル転送プロセス状況を正確に表示することができます。

Windows Installer は、「ティック」と呼ばれる進行状況を追跡します。進行状況バーがこの情報をサブスクライブしている場合、**ticksSoFar** および **totalTicks** の値を渡します。進行状況バーはこの情報を使用してインストールの全進行状況を表示します。

サブスクリプションは進行状況バー以外にも使用されます。シリアル番号をインストールの一部として検証するカスタム アクションがあることがあります。このアクションは検証の成否をパブリッシュできます。ダイアログで [次へ] ボタンをクリックするとこの情報をサブスクライブできます。アクションが検証結果として失敗をパブリッシュすると、ボタンは有効になりません。アクションが成功をパブリッシュすると、ボタンは有効になります。

サブスクリプションの種類

イベントとコントロールのサブスクリプションを設定するには、[ダイアログ] ビューでダイアログの [動作] 領域を使用します。詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」を参照してください。

次はコントロールがサブスクライブ可能なイベントの一覧です:

テーブル 4 ・ コントロールがサブスクライブ可能なイベントの種類

イベント	説明
ActionData	最終アクションについての情報を表示します。 テキスト コントロールはこのイベントをサブスクライブすることにより、インストール中にコピーされたファイル名などの情報を表示できます。
ActionProgress	監視中のアクションについての進行状況を表示します。 進行状況バー コントロールが、このイベントをサブスクライブできます。 アクションを指定するには、この設定のサブ設定を構成してください。
ActionText	現在のアクションの名前を表示します。 テキストコントロールはこのイベントをサブスクライブすることにより、インストール中に実行された現在のアクションの説明を表示できます。
IgnoreChange	現在のディレクトリ内のフォルダを開かずに強調表示します。
ScriptInProgress	Windows Installer がインストールの実行スクリプトをコンパイル中に、情報テキストを表示します。
SelectionAction	強調表示されたアイテムについて説明する UI 文字列を表示します。 テキスト コントロールで、このイベントをサブスクライブして UI 文字列を表示できます。
SelectionDescription	テキスト コントロール内の UI 文字列を表示します。UI 文字列は、選択ツリー コントロールで選択された機能について説明します。
SelectionNoItems	説明テキストを削除するか、不要なボタンを無効にします。

テーブル 4 ・ コントロールがサブスクライブ可能なイベントの種類 (続き.)

イベント	説明
SelectionPath	選択ツリー コントロールで選択されたアイテムのパスをパブリッシュします。 テキスト コントロールで、このイベントをサブスクライブしてパスを表示できます。
SelectionPathOn	選択パスが現在選択されている機能と関連付けられているかどうかを示すブール値をパブリッシュします。
SelectionSize	強調表示されたアイテムのサイズをパブリッシュします。
SetProgress	インストールの進行状況についての情報をパブリッシュします。
TimeRemaining	現在処理中のシーケンスの残りの秒数を概算でパブリッシュします。 テキスト コントロールで、このイベントをサブスクライブして残りの時間を表示できます。

基本の MSI ダイアログでコントロール イベントをトリガーする



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

イベントは、エンド ユーザーがコントロールを使用した時に発生する定義済みの動作です。イベントは、[ダイアログ] ビューの [動作] 領域 で定義されます。使用可能な各イベントに関する詳細は、「基本の MSI ダイアログでコントロール イベントを使用する」を参照してください。

コントロールのイベントを表示するには、コントロール名のリストからイベントを選択して、右のペインにある″イベント″ 設定を構成します。

ダイアログ コントロールに新しいイベントを追加する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*



タスク

ダイアログコントロールに新しいイベントを追加するには、以下の手順を実行します。

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、変更するダイアログのダイアログ ノードを拡張してから、**[動作]** サブノードをクリックします。
3. 選択されたダイアログのすべてのコントロールの一覧が表示される中央ペインで、構成するコントロールをクリックします。設定が、右のペインに表示されます。

4. **"イベント"** 設定で、**[新しいイベント]** ボタンをクリックしてから、追加するイベントの種類を選択します。**"イベント"** 設定の下に新しい一連の行が追加されます。
5. イベントの条件を指定するには、"イベント" 設定で省略記号ボタン (...) をクリックします。**[条件ビルダー]** ダイアログ ボックスが開き、ここでアクションに対して満たさなくてはならない条件を指定できます。

   任意の条件下でイベントをトリガする場合、次の条件を入力します : 1

   1
6. イベント設定にサブ設定がある場合、必要に応じてサブ設定を構成します。

ダイアログ コントロールのイベントを並べ替える



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

ユーザー インターフェイス コントロールに構成されたイベントは、実行時にコントロールの "イベント" 設定で表示された順番で起動されます。



タスク ***イベントを移動するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、変更するダイアログのダイアログ ノードを拡張してから、**[動作]** サブノードをクリックします。
3. 選択されたダイアログのすべてのコントロールの一覧が表示される中央ペインで、並べ替えるイベントを含むコントロールをクリックします。設定が、右のペインに表示されます。
4. 設定のグリッドの **[イベント]** 領域で、移動させたいイベントのイベント設定を見つけます。
5. 移動するイベントのイベント設定で、**[イベントの移動]** ボタンをクリックして、**[上に移動]** または **[下に移動]** をクリックします。

イベントをダイアログ コントロールから削除する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*



タスク ***コントロールからイベントを削除するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、変更するダイアログのダイアログ ノードを拡張してから、**[動作]** サブノードをクリックします。

3. 選択されたダイアログのすべてのコントロールの一覧が表示される中央ペインで、削除するイベントを含むコントロールをクリックします。設定が、右のペインに表示されます。
4. 設定のグリッドの **[イベント]** 領域で、削除したいイベントのイベント設定を見つけます。
5. ”イベント” 設定で、**[イベントの削除]** ボタンをクリックします。

ダイアログ ボタンからカスタム アクションを起動する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

ダイログ上のボタンからカスタム アクションを起動するには、コントロールの DoAction イベントを使用します。DoAction イベントをセットアップするには、カスタム アクションを作成する必要があります。

カスタム アクションを作成したら、このアクションを起動する DoAction イベントを設定できます。



タスク

プッシュボタン コントロールからカスタム アクションを起動するには、次の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、変更するダイアログのダイアログ ノードを拡張してから、**[動作]** サブノードをクリックします。
3. 選択されたダイアログのすべてのコントロールの一覧が表示される中央ペインで、カスタム アクションをトリガするプッシュボタン コントロールをクリックします。設定が、右のペインに表示されます。
4. **”イベント”** 設定で、**[新しいイベント]** ボタンをクリックしてから **[DoAction]** をクリックします。InstallShield によって、**”イベント”** 設定の下に新しい行のセットが追加されます。
5. イベントの条件を指定する場合: **”DoAction”** 設定で省略記号ボタン (...) をクリックします。**[条件ビルダー]** ダイアログ ボックスが開き、ここでアクションに対して満たさなくてはならない条件を指定できます。
6. **”アクション”** サブ設定で、コントロールと関連付けるアクションの名前を指定します。

インストールが期待通りに動作するのを確認するために、インストールのビルドとテストを行います。

[カスタム アクションとシーケンス] ビューでエンドユーザー ダイアログの順序の表示する (基本の MSI プロジェクト)



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。InstallScript または InstallScript MSI プロジェクトのエンドユーザー ダイアログの順序を表示するには、InstallScript ビュー を使用してスクリプトを調べます。*

[カスタム アクションとシーケンス] ビューで、インストールの実行時にエンドユーザーに表示されるプライマリおよび次のダイアログの順序を表示できます。プライマリ ダイアログは、最上位のダイアログで、[インストール] シーケンスに挿入されています。[カスタム アクションとシーケンス] ビューでプライマリ ダイアログの順序を変更できます。

次のダイアログは、直前のダイアログの [次へ] ボタンのコントロールイベントのアクションにより起動されるダイアログです。次のダイアログは [カスタム アクションとシーケンス] ビューで表示できますが、[インストール] シーケンスの一部ではないので、[カスタム アクションとシーケンス] ビューで順序を変更することはできません。次のダイアログの順番を変更するには、[ダイアログ] ビューでダイアログの下にある [動作] ノードを使用しなくてはなりません。詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」を参照してください。

初期の [シーケンス] エクスプローラー

[カスタム アクションとシーケンス] ビューを初めて開いたとき、[シーケンス] エクスプローラーで、すべてのプライマリ ダイアログおよびアクションのリストが、インストール中に実行される順序で示されます。多数の [次へ] ダイアログには、プラス記号 (+) が横に付いています。

拡張 [シーケンス] エクスプローラー

インストール シーケンスに挿入されているプライマリダイアログの順序を表示するほか、[シーケンス] エクスプローラーで次のダイアログの順序を表示することもできます。

プライマリ ダイアログの隣りのプラス記号 (+) をクリックすると、リストが展開され、実行時に表示されるプライマリ ダイアログの次のダイアログ (ある場合) の順序が表示されます。特定のダイアログに複数の [次へ] ボタンのコントロールイベントがある場合は、可能性のあるすべての次のダイアログがプライマリダイアログの下に表示されます。



図 3: シーケンス内のダイアログの順番を表示する

次のダイアログの情報パネル

次のダイアログをクリックすると、右側のペインにダイアログの情報が表示されます。右のペインにはまた、ヘルプ情報のほか、動作エディターおよびレイアウト エディターへのハイパーリンクが表示されます。選択されたダイアログのより詳しい情報は、[ダイアログ] ビューで見ることができます。

コンテキストメニューのコマンド

プライマリ ダイアログを右クリックすると、コンテキスト メニューによりコマンドが表示され、シーケンスの編集、ダイアログ シーケンス ビューの更新、ダイアログのレイアウトや動作の編集ができます。次のダイアログを右クリックすると、コンテキスト メニューによりコマンドが表示され、ダイアログのレイアウトや動作を編集できます。

動作の編集

タスク ***[カスタム アクションとシーケンス] ビューの [シーケンス] エクスプローラー内から [ダイアログの動作エディター] に移動するには、以下のいずれかの方法を実行します。***

- 編集する次のダイアログを右クリックして、コンテキストメニューから**動作の編集** をクリックします。
- 編集するダイアログ名前をクリックします。次に、右のペインの **[アクション項目]** セクションで **[ダイアログの動作を編集する]** をクリックします。

詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」を参照してください。

レイアウトの編集

タスク ***[カスタム アクションとシーケンス] ビューの [シーケンス] エクスプローラー内から [ダイアログのレイアウトエディター] に移動するには、以下のいずれかの方法を実行します。***

- 編集する次のダイアログを右クリックして、コンテキスト メニューから **[レイアウトの編集]** をクリックします。
- 編集するダイアログ名前をクリックします。次に、右のペインの **[アクション項目]** セクションで **[ダイアログのレイアウトを編集する]** をクリックします。インストールが複数の言語で作成されている場合は、リストで適切な言語を選択する必要があります。

詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」を参照してください。

CustomSetup ダイアログのオプション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*
- *トランスフォーム*

CustomSetup ダイアログには、ターゲット システム、インストール中の機能、および Windows Installer のインストール オプションについての情報と緊密に統合された、洗練されたエンドユーザー インターフェイスがあります。これによって、エンドユーザーがインストールを最大限に制御することができます。

このダイアログが提供する多くのオプションおよび情報は、下記で説明されているようにセットアップのデザインで設定された機能のプロパティによって決定されます。

[アドバタイズ] オプション

アドバタイズ機能によって、インストールが最初に実行された後に、必要に応じてファイルをインストールすることができます。CustomSetup ダイアログで、エンドユーザーが機能をクリックすると、**[必要な場合にインストール]** オプションを選択して、その機能を後からインストールするように指定することができます。

ただし、このデフォルトのオプションが表示されるのは、機能の "アドバタイズ" 設定に [アドバタイズを許可する] または [アドバタイズを優先する] を選択した場合のみです。Windows Installer が機能をアドバタイズするオプションを表示しないようにする場合、この設定で [アドバタイズを許可しない] を選択します。

機能を非表示にする

機能の "表示" 設定 を [非表示] に設定すると、CustomSetup ダイアログに機能またはサブ機能は表示されず、エンド ユーザーはインストール オプションを変更できなくなります。

すべてのサブ機能を表示する

機能の "表示" 設定も、以下のオプションを持つダイアログが最初に表示されたときに、サブ機能が展開されるかどうかを制御します。

テーブル 5 ・ 機能の "表示" 設定のオプション

オプション	説明
閉じて表示する	デフォルトでサブ機能が閉じた状態で、機能が CustomSetup ダイアログで表示されます。
展開して表示する	デフォルトでサブ機能が展開された状態で、機能が CustomSetup ダイアログで表示されます。

機能のデフォルトのインストール先フォルダーを設定する

CustomSetup ダイアログで機能を選択し、[詳細] ボタンをクリックすることによって、ユーザーは機能がインストールされるフォルダーを表示して編集できます。機能の "インストール先" 設定は、表示されるデフォルト値であり、実際には [機能の詳細] ダイアログにユーザーが新しい値を入力するか新しいパスを参照すると、その新しい値に再び割り当てられます。

機能のインストール場所に影響する要因についての総合的な説明は、「機能のインストール先を設定する」を参照してください。

機能に名前を付ける

"表示名" を設定して、CustomSetup ダイアログに表示される別の名前を指定します。

機能の説明を表示する

機能を選択したときに CustomSetup ダイアログの下部に表示される説明は、機能の"説明" 設定から取得されます。

機能の順序を変更する

CustomSetup ダイアログで機能がエンドユーザーに対して表示される順序を変更することができます。機能が表示される順序は、[セットアップのデザイン] ビューでの順序から取得されます。詳細については、「機能の並べ替え」を参照してください。

機能のインストールを必須にする

機能の"必須" 設定を [はい] に設定すると、エンドユーザーに対して **[使用不能にする]** オプションが表示されず、機能をインストールしなければならなくなります。

ダイアログのテーマ



プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

ダイアログ テーマは、エンドユーザー ダイアログに統一感のとれた個性的な印象を与えることができる、あらかじめ定義されている 1 セットのイメージです。

ボタンをクリックするだけで、プロジェクトに提供されているテーマから 1 つ選んで、Setup.exe 初期化ダイアログを含む、プロジェクトで使用されているすべての内部および外部ダイアログに適用することができます。それぞれのダイアログは [ダイアログ] ビュー内から簡単にプレビューすることができ、選択したテーマがどのように見えるかを実際に確認することができます。



メモ: *InstallShield では現在、独自のダイアログ テーマを作成することはできませんが、多くのテーマが用意されています。詳細については、「提供されているテーマと対応するダイアログのサイズ」を参照してください。*

幅の広いダイアログ テーマにおける実行時の要件



プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

幅の広いダイアログ テーマを使用する場合、ターゲット システムの画面解像度は最低 640 x 480 ピクセル (大きいフォントが使用されている場合、1024 x 768) でなければなりません。解像度がそれより低い場合、水平スクロールバーが実行時に、幅の広いテーマで表示される各ダイアログの底辺に追加されます。

ダイアログ テーマのプレビュー



プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

プロジェクトに異なるテーマを選択すると、そのテーマは、[ダイアログ] ビューにあるすべての内部および外部ダイアログに適用されます。ダイアログの表示に、インストールをビルドして実行する必要はありません。



タスク ***プロジェクトでテーマを選択する前にプレビューするには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**InstallWelcome** テーブルをクリックします。
3. **[テーマ]** フォルダーの下に表示されているテーマから 1 つクリックします。

右のペインに、選択されたテーマでサンプル外部ダイアログのスクリーン ショットが表示されます。



タスク ***プロジェクトのダイアログが選択したテーマでどのように見えるかをプレビューするには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、それから新しいダイアログ コントロールを追加するダイアログ アイテムを展開します。
3. 展開したダイアログ アイテムの下にある言語をクリックします。

中央のペインに、選択されたテーマが適用された状態でダイアログのプレビューが表示されます。

ダイアログ テーマの選択または変更



プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

ダイアログ テーマを使用して、インストールのエンドユーザー ダイアログの外観を変更することができます。1 つのプロジェクトにつき、1 つのテーマを選択することができます。



ヒント: *幅の広いテーマから標準サイズのテーマに変更すると、ダイアログの左端または右端に配置されたコントロールの位置が変わる可能性があります。*

プロジェクトに選択したテーマを変更して、カスタム外部ダイアログも追加する場合、まずテーマを変更してから、カスタム外部ダイアログを追加します。これを逆にすると、ダイアログが適切に表示されない場合があります。



タスク ***プロジェクトに使用されているダイアログ テーマを変更するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**InstallWelcome** テーブルをクリックします。
3. 使用するテーマを右クリックして、**[選択]** をクリックします。

選択されたテーマが、プロジェクトのダイアログに適用されます。さらに [ダイアログ] エクスプローラーでは、選択されたテーマのアイコンに赤いチェック マークが表示されます。

ヒント: また、[ダイアログ] エクスプローラーで使用するテーマを選択して、ボタン 1 つでテーマを変更することもできます。右のペインで [選択] ボタンをクリックします。

InstallShield でテーマがどのように適用されるかについての背景事情

ダイアログ テーマを別のテーマへ切り替えると、InstallShield はそのテーマを、特定の幅と高さに一致するプロジェクト内のすべてのダイアログに適用します。したがって、プロジェクトでダイアログの幅または高さを変更したあとで、テーマを変更すると、そのテーマはカスタマイズされた幅または高さを持つダイアログには適用されません。

たとえば、標準幅のテーマを他のテーマに変更した場合、そのテーマは標準の幅と高さ (374 x 266) を持つすべてのダイアログに適用されます。幅の広いテーマを他のテーマに変更した場合、そのテーマは 584 x 274 のすべてのダイアログに適用されます。

外部ダイアログ スタイルを使用するダイアログに関する背景情報

デフォルトでは、すべての外部ダイアログに、ダイアログの左側にイメージが表示される同じスタイルのダイアログが使用されます。InstallWelcome ダイアログと SetupCompleteSuccess ダイアログは、外部ダイアログの例です。

テーマを選択したあと、カスタム外部ダイアログをプロジェクトに追加すると、選択されたテーマが新しいカスタム外部ダイアログに使用されます。ただし、プロジェクトのテーマをあとで変更すると、そのテーマはカスタム外部ダイアログで適切に表示されない場合があります。このタイプの問題を解決するには、「テーマをカスタム外部ダイアログに適用する」をご覧ください。

テーマをカスタム外部ダイアログに適用する

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

プロジェクトに選択したテーマを変更して、カスタム外部ダイアログも追加する場合、まずテーマを変更してから、カスタム外部ダイアログを追加する方法がより簡単です。そうしなかった場合、ダイアログが適切に表示されない場合があります。新しく選択されたテーマの内部ダイアログイメージが部分的にカスタム外部ダイアログに追加されます。

カスタム外部ダイアログをプロジェクトに追加したあと、テーマを変更する場合、カスタム外部ダイアログの名前を、InstallShield Program Files フォルダーにインストールされている適切な .theme ファイルに追加する必要があります。そのあと、テーマをプロジェクトに適用できます。

タスク ***カスタム外部ダイアログをプロジェクトに追加したあとでテーマを変更するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield Program Files フォルダーで、ダイアログで使用するテーマの .theme ファイルを見つけます。デフォルト保存先は次の場所です:

   C:¥Program Files¥InstallShield¥2022¥Support¥Themes

3. メモ帳などのテキスト エディターで .theme ファイルを開きます。たとえば、Global テーマを使用する場合、Global.theme ファイルを開きます。
4. ファイルの **<Include>** または **<Exclude>** セクションで、次の行を追加します:

   <Name>*NameOfDialog***</Name>**

   NameOfDialog には、ダイアログの名前が入ります。
5. InstallShield でプロジェクトを開きます。
6. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
7. **[ダイアログ]** エクスプローラーで、**InstallWelcome** テーブルをクリックします。
8. 使用するテーマを右クリックして、**[選択]** をクリックします。

ロゴまたは他のイメージを外部ダイアログを追加する



プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

外部ダイアログは、インストールの始めと終わりに表示されるダイアログで、通常 Welcome ダイアログと Completion ダイアログがこれにあたります。一部のテーマの外部ダイアログには、会社または製品のロゴを配置することができます。たとえば、Monitor テーマと Circles テーマの外部ダイアログには、ダイアログの左側に空白部分があります。これらのダイアログにビットマップ コントロールを追加して、エンドユーザー インターフェイスをカスタマイズすることができます。



タスク ***外部ダイアログにイメージを追加するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、イメージを追加するダイアログのダイアログ アイテムを展開します。
3. 展開したダイアログ アイテムの下にある言語をクリックします。中央のペインにあるダイアログ エディターに、選択した言語のダイアログが表示されます。
4. **[コントロール]** ツールバー上で、**[ビットマップ]** ボタンをクリックします。
5. ダイアログ エディターで、ビットマップ コントロールが開始する箇所をクリックし、左マウス ボタンを押したまま、コントロールが終わる箇所までマウスをドラッグします。左マウス ボタンを解放します。
6. 右のペインでビットマップ コントロールのプロパティを構成します。

   "ファイル名" プロパティをクリックして、省略記号ボタン (...) をクリックして、使用するビットマップ ファイルを参照します。

必要に応じて、この手続きをプロジェクトのそれぞれの言語および外部ダイアログについて繰り返します。以下は、すべての新しい基本の MSI プロジェクトでデフォルトで提供されている外部ダイアログです:

- AdminWelcome
- InstallWelcome

- MaintenanceWelcome
- PatchWelcome
- SetupCompleteError
- SetupCompleteSuccess
- SetupInitialization
- SetupInterrupted
- SetupResume
- SplashBitmap

提供されているテーマと対応するダイアログのサイズ



プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*



エディション: *一部のテーマは、InstallShield の Premier Edition と Professional Edition の両方で提供されていますが、Premier Edition のみで提供されているものもあります。*

一部のテーマのダイアログのサイズは、標準の幅と高さ (374 x 266) に設定されています。幅の広いテーマのダイアログ サイズは 584 x 274 に設定されています。次のテーブルは、各ダイアログのサイズとそのテーマを含む InstallShield エディションを示します。

テーブル 6 ・ InstallShield のダイアログ テーマ

名前	InstallShield エディション	ダイアログの幅	ダイアログの高さ
Circles テーマ (大)	Premier および Professional	584	274
Classic テーマ	Premier および Professional	374	266
Cooperation テーマ (大)	Premier	584	274
Filmstripテーマ (大)	Premier	584	274
Global テーマ	Premier および Professional	374	266
InstallShield Blue テーマ	Premier および Professional	374	266

テーブル 6 • InstallShield のダイアログ テーマ (続き.)

名前	InstallShield エディション	ダイアログの幅	ダイアログの高さ
InstallShield Blue テーマ (大)	Premier および Professional	584	274
InstallShield Silver テーマ	Premier	374	266
Monitor テーマ	Premier	374	266
Pastel Wheat テーマ	Premier	374	266
Theater テーマ (大)	Premier	584	274

Circles テーマ (大)

プロジェクト• *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

以下は、Circle テーマ (大) のサンプル外部ダイアログと内部ダイアログです。

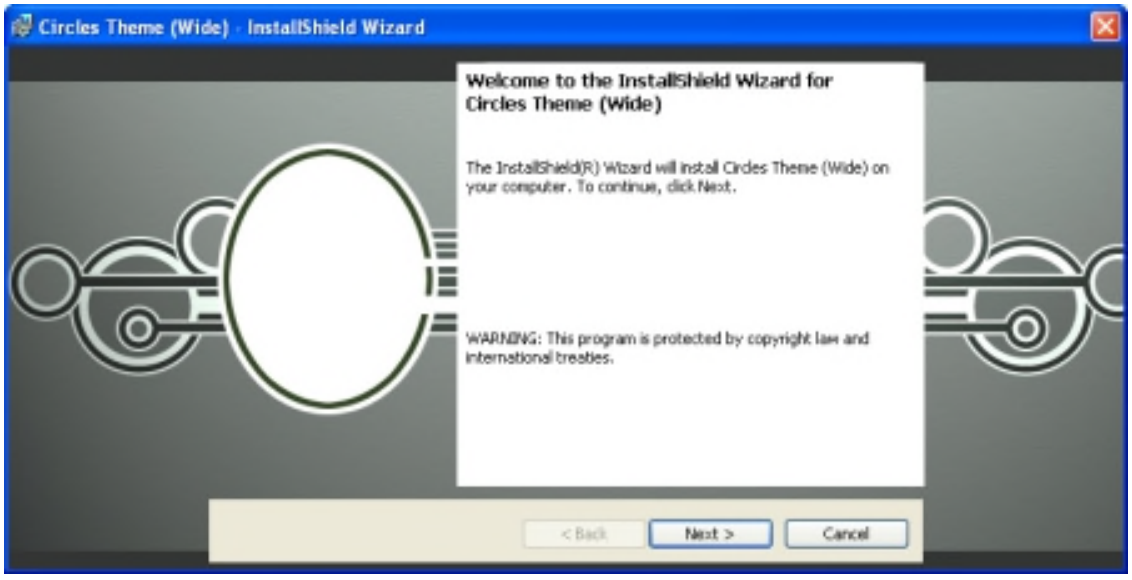


図 4: Circle テーマ (大) のサンプル外部ダイアログ



図 5: Circle テーマ (大) のサンプル内部ダイアログ

外部ダイアログに会社または製品ロゴを追加する方法については、「ロゴまたは他のイメージを外部ダイアログを追加する」をご覧ください。

Classic テーマ

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

以下は、Classic テーマのサンプル外部ダイアログと内部ダイアログです。

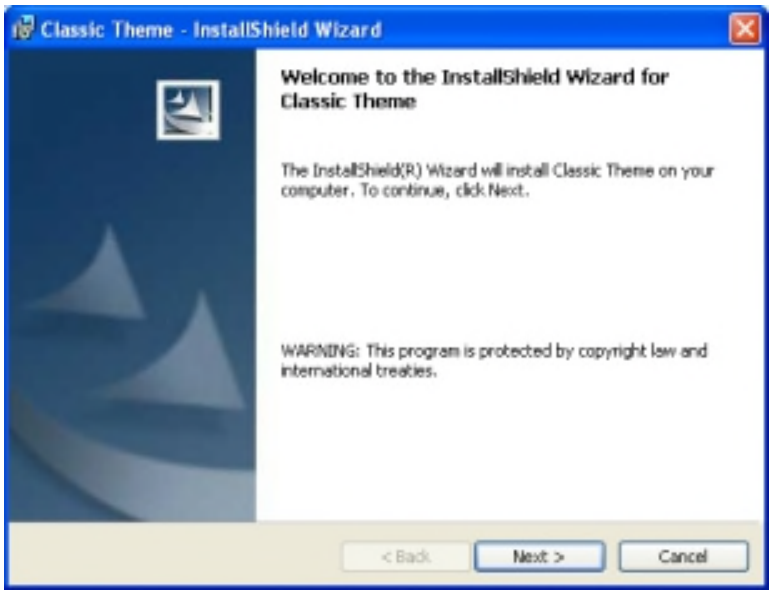


図 6: Classic テーマのサンプル外部ダイアログ

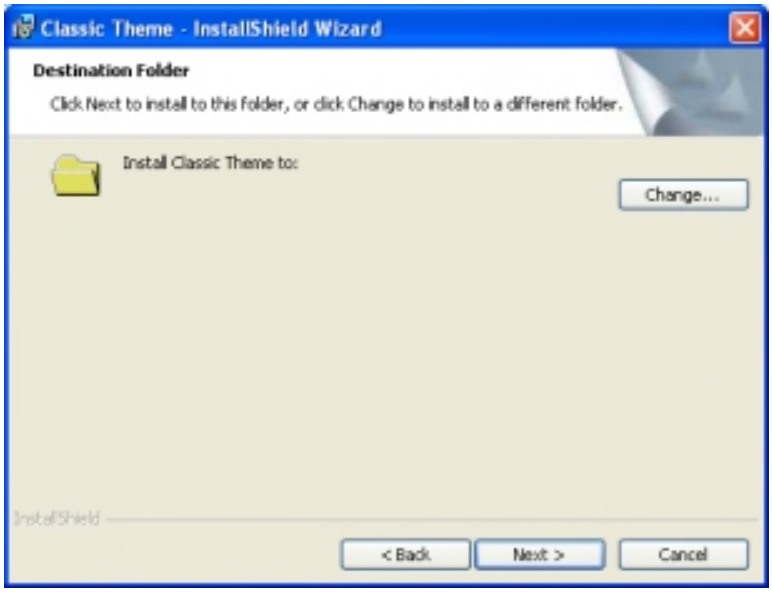


図 7: Classic テーマのサンプル内部ダイアログ

Cooperation テーマ (大)



プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

エディション: *このテーマは、InstallShield の Premier Edition で提供されています。*

以下は、Cooperation テーマ（大）のサンプル外部ダイアログと内部ダイアログです。



図 8: Cooperation テーマ（大）のサンプル外部ダイアログ

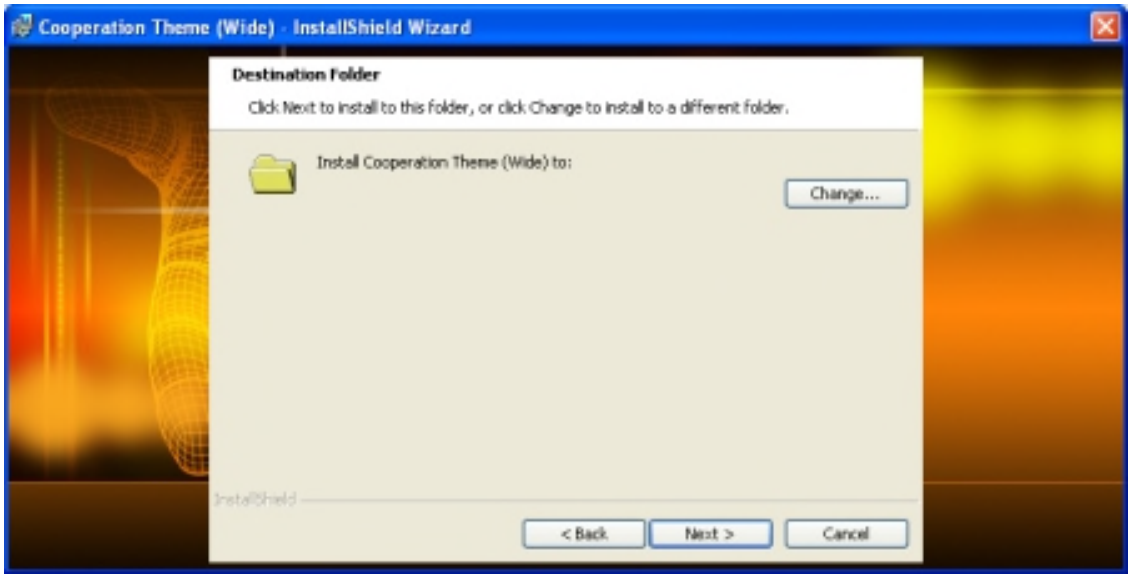


図 9: Cooperation テーマ（大）のサンプル内部ダイアログ

Filmstripテーマ（大）

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*



エディション: *このテーマは、InstallShield の Premier Edition で提供されています。*

以下は、Filmstrip テーマ (大) のサンプル外部ダイアログと内部ダイアログです。

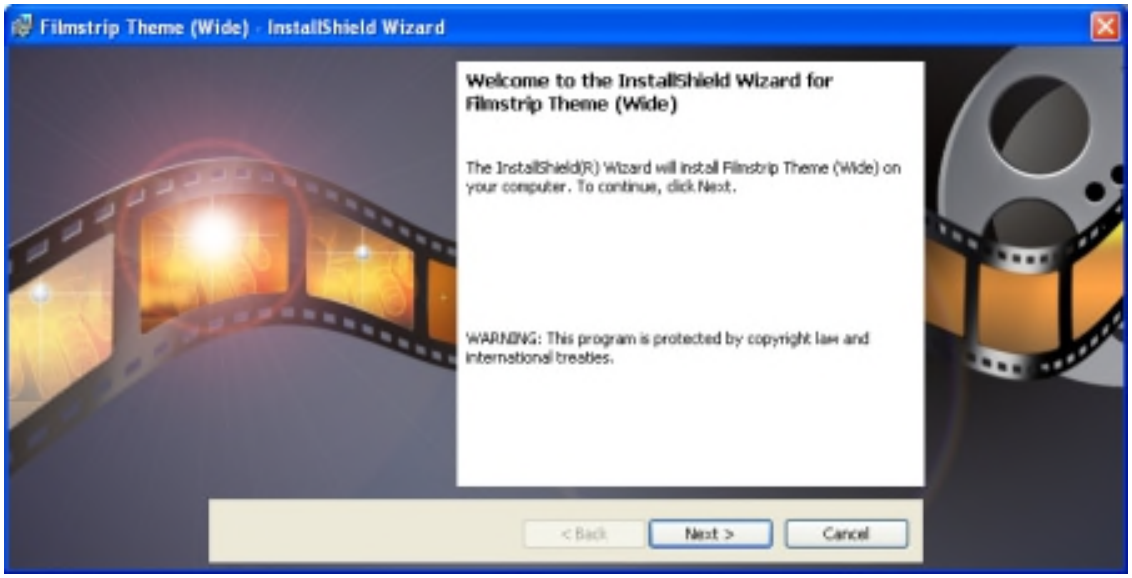


図 10: Filmstrip テーマ (大) のサンプル外部ダイアログ



図 11: Filmstrip テーマ (大) のサンプル内部ダイアログ

Global テーマ

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

以下は、Global テーマのサンプル外部ダイアログと内部ダイアログです。

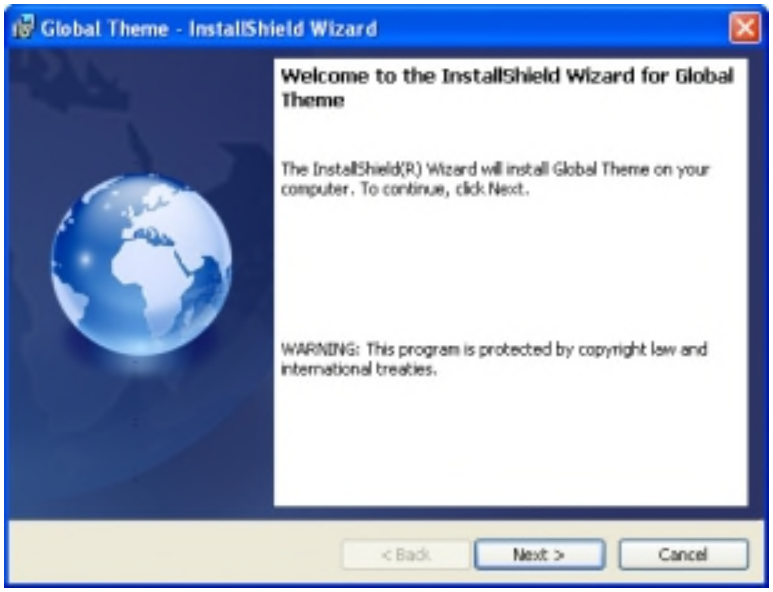


図 12: Global テーマのサンプル外部ダイアログ

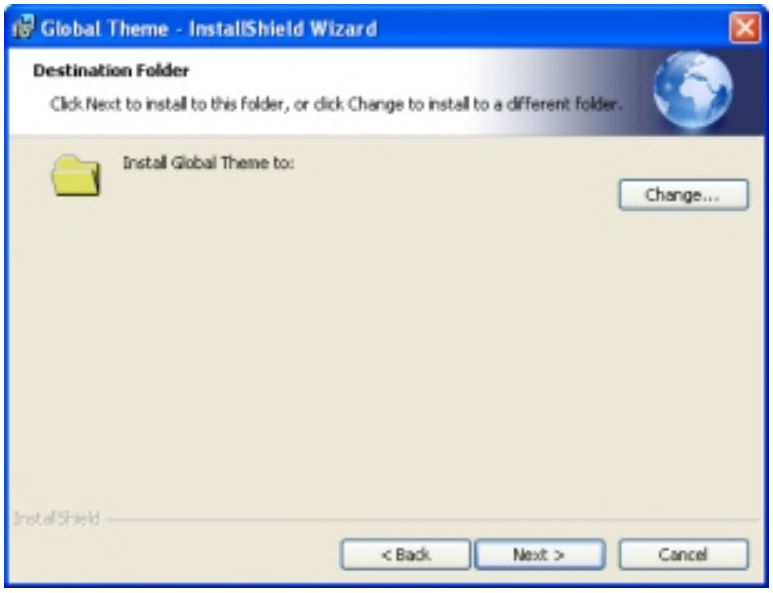


図 13: Global テーマのサンプル内部ダイアログ

InstallShield Blue テーマ

プロジェクト- *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

以下は、InstallShield Blue テーマのサンプル外部ダイアログと内部ダイアログです。



図 14: InstallShield Blue テーマのサンプル外部ダイアログ

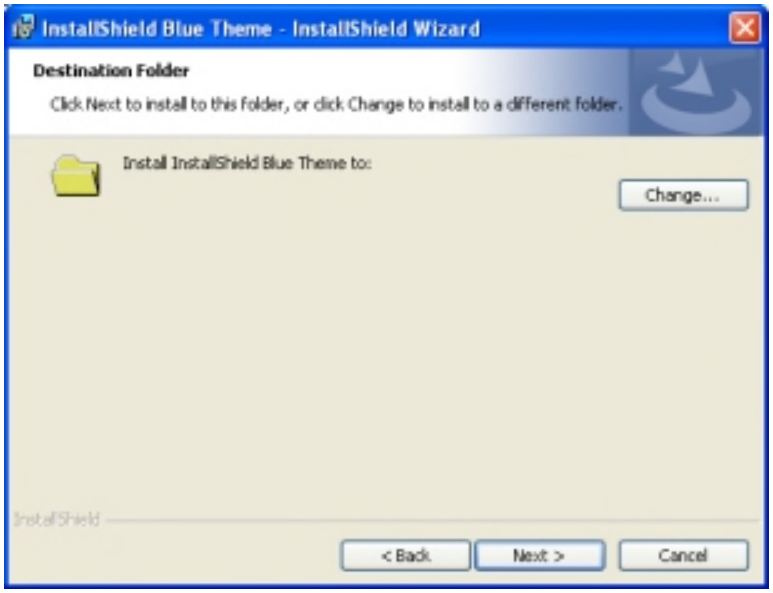


図 15: InstallShield Blue テーマのサンプル内部ダイアログ

InstallShield Blue テーマ（大）

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

以下は、InstallShield Blue テーマのサンプル外部ダイアログと内部ダイアログです。

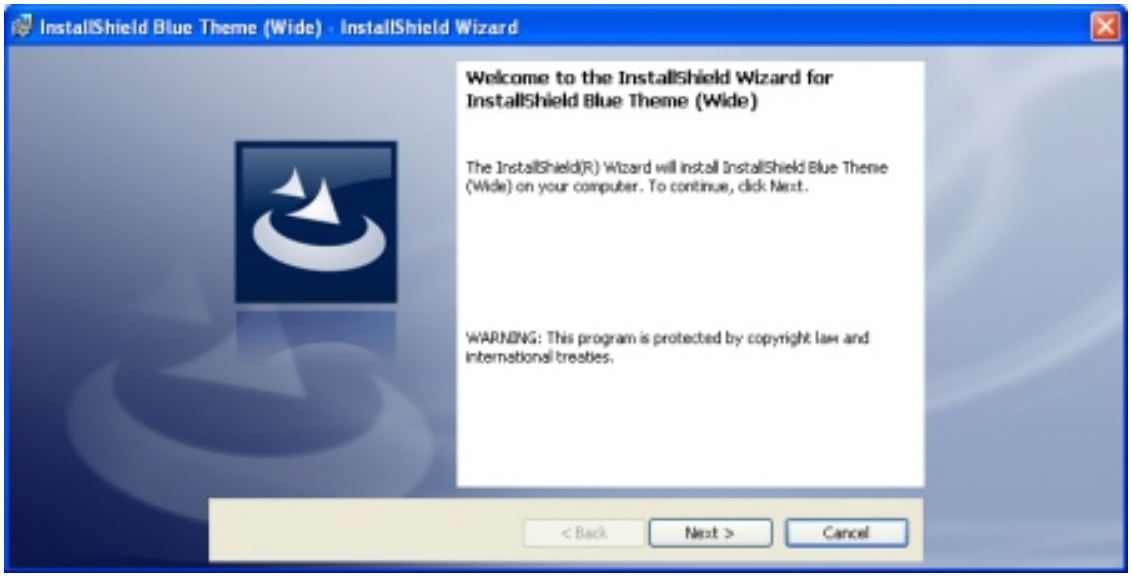


図 16: InstallShield Blue テーマ（大）のサンプル外部ダイアログ

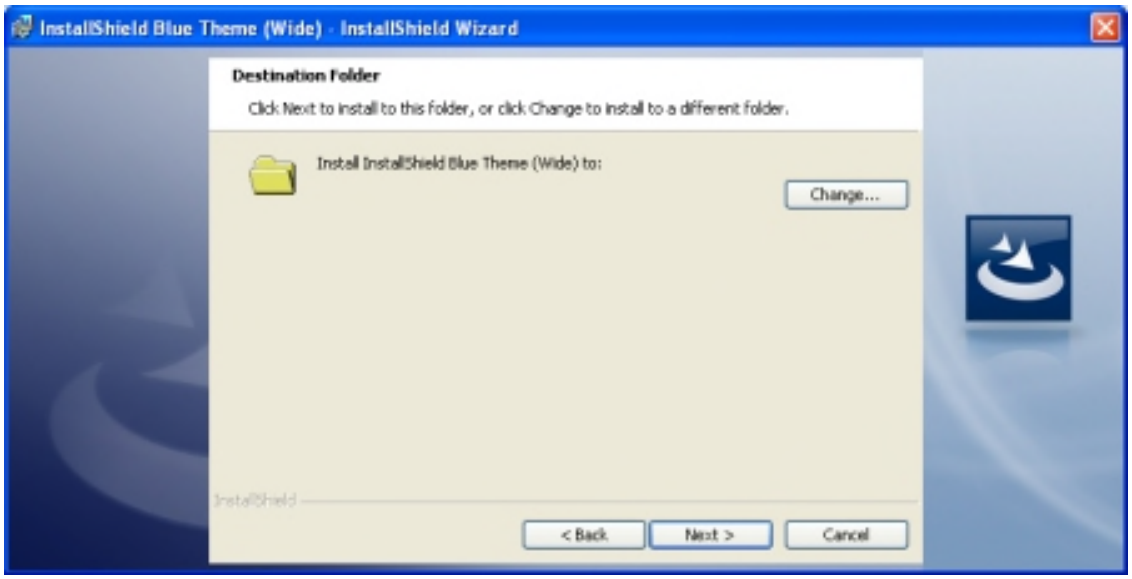


図 17: InstallShield Blue テーマ（大）のサンプル内部ダイアログ

InstallShield Silver テーマ

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

エディション: *このテーマは、InstallShield の Premier Edition で提供されています。*

以下は、InstallShield Silver テーマのサンプル外部ダイアログと内部ダイアログです。



図 18: InstallShield Silver テーマのサンプル外部ダイアログ



図 19: InstallShield Silver テーマのサンプル内部ダイアログ

Monitor テーマ

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

エディション: *このテーマは、InstallShield の Premier Edition で提供されています。*

以下は、Monitor テーマのサンプル外部ダイアログと内部ダイアログです。

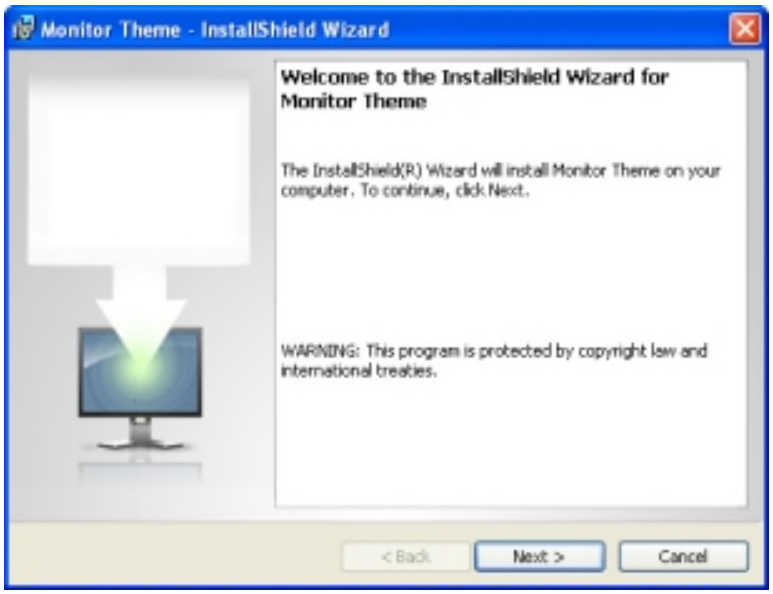


図 20: Monitor テーマのサンプル外部ダイアログ

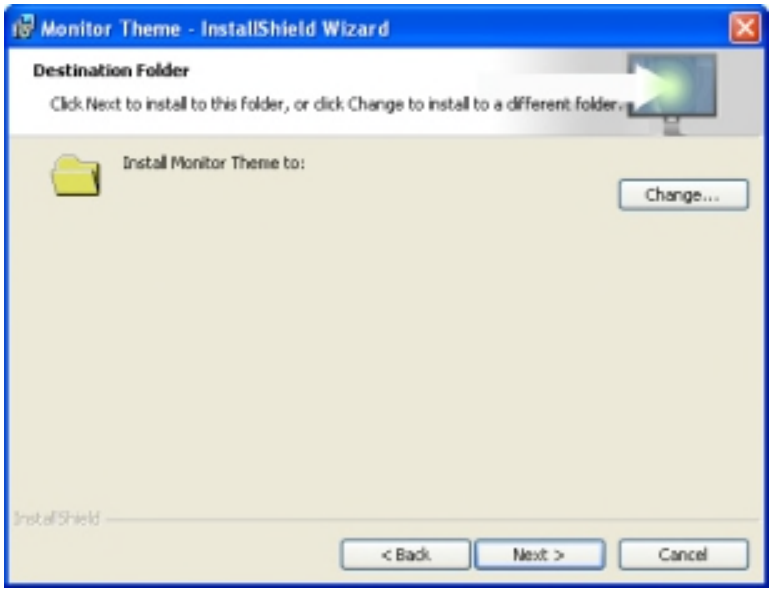


図 21: Monitor テーマのサンプル内部ダイアログ

外部ダイアログに会社または製品ロゴを追加する方法については、「ロゴまたは他のイメージを外部ダイアログを追加する」をご覧ください。

Pastel Wheat テーマ

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

エディション: *このテーマは、InstallShield の Premier Edition で提供されています。*

以下は、Pastel Wheat テーマのサンプル外部ダイアログと内部ダイアログです。

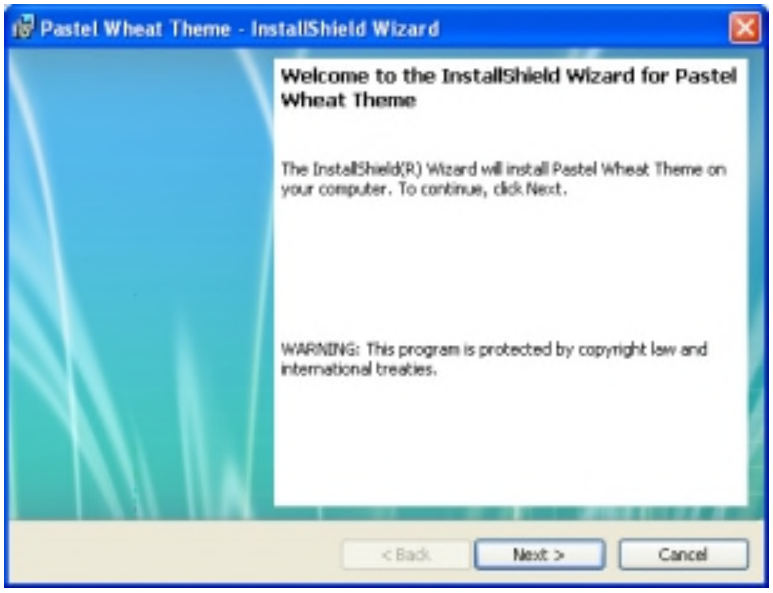


図 22: Pastel Wheat テーマのサンプル外部ダイアログ

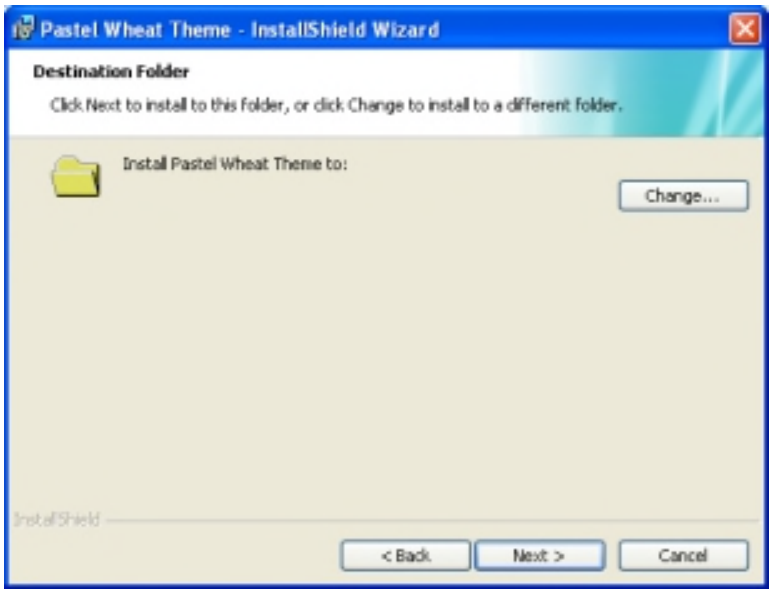


図 23: Pastel Wheat テーマのサンプル内部ダイアログ

Theater テーマ (大)

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

エディション: *このテーマは、InstallShield の Premier Edition で提供されています。*

以下は、Theater テーマ (大) のサンプル外部ダイアログと内部ダイアログです。

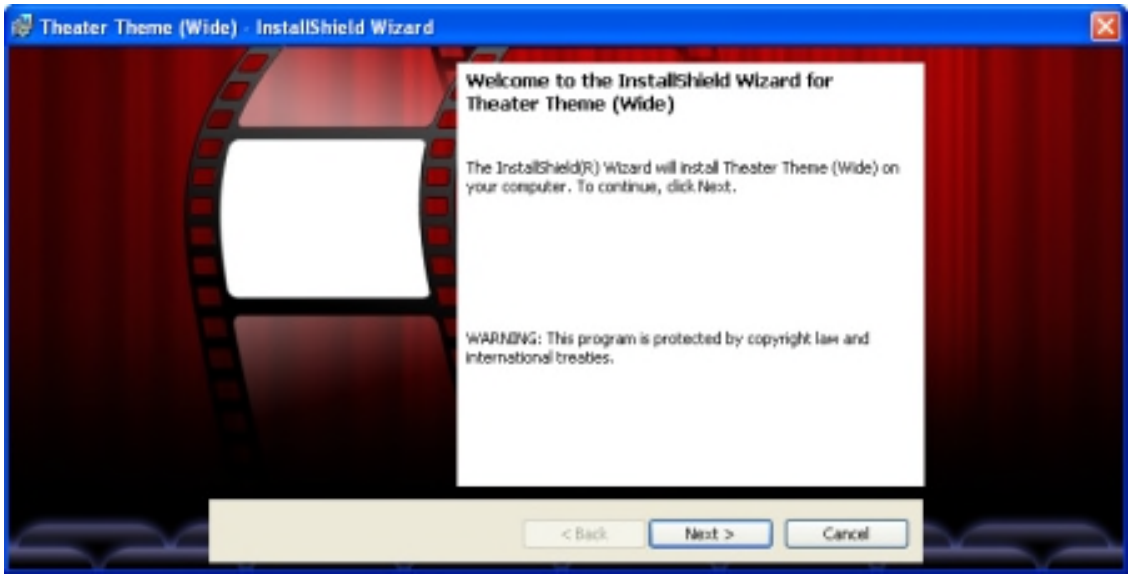


図 24: Theater テーマ (大) のサンプル外部ダイアログ

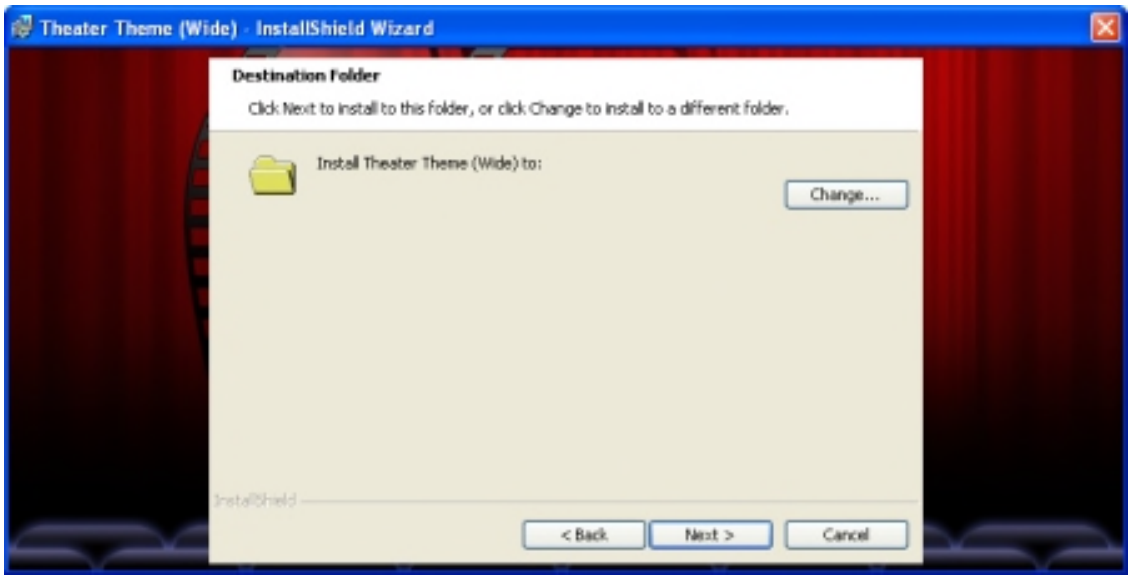


図 25: Theater テーマ (大) のサンプル内部ダイアログ

外部ダイアログに会社または製品ロゴを追加する方法については、「ロゴまたは他のイメージを外部ダイアログを追加する」をご覧ください。

右から左に記述される言語のダイアログ サポート

エディション: *InstallShield Premier Edition には、右から左に読まれるアラビア語とヘブライ語のダイアログ サポートが含まれます。*

プロジェクト: *次のプロジェクト タイプに、右から左に記述される言語のダイアログ サポートが含まれます:*

- *基本の MSI*
- *マージ モジュール*

InstallShield には、右から左に記述される言語であるアラビア語 (サウジアラビア) とヘブライ語のサポートが含まれています。デフォルトのエンド ユーザー ダイアログ文字列のすべてが、これらの言語で利用できます。

これらの言語は右から左に読まれるため、InstallShield にはアラビア語とヘブライ語ダイアログのミラー サポートが含まれます。これによって、アラビア語とヘブライ語のダイアログには、右から左方向へのレイアウトが使用されます。たとえば、英語やその他の左から右に読まれる言語のダイアログで右側にあるボタンは、右から左に読まれる言語のダイアログでは左側に移動されます。これは InstallShield の [ダイアログ] ビューの [ダイアログ エディター] ペインで処理され、実行時にも同様に処理されます。

適切な場合、ビルトイン ダイアログ テーマで表示されるダイアログ イメージの逆向きバージョンが表示されます。逆向きバージョンは、イメージ ファイル名の .bmp または .jpg 部分の直前に *_mirror* が付いています。たとえば、左から右に読み書きされる言語のイメージ名が **banner.jpg** の場合、右から左に読み書きされる言語の対応するイメージの名前は **banner_mirror.jpg** となります。これらの 2 つのファイルは同じフォルダーに配置され、ダイアロ

グの右から左に読まれる言語バージョンには、自動的に banner_mirror.jpg が使用されます。イメージを逆向きにすべきではない場合、そのイメージの *_mirror.jpg または *_mirror.bmp バージョンは含まれず、ダイアログの右から左方向バージョンは、イメージのミラー イメージを表示しません。



ヒント: *ランタイム ダイアログにカスタム イメージを使用する場合で、プロジェクトが右から左に記述される言語のサポートを含むときは、これらの右から左に記述される言語用にミラー イメージ バージョンを作成する必要がある可能性があります。[ダイアログ] ビューで、ダイアログの右から左方向レイアウトをプレビューして、カスタム イメージがどのように表示されるかを確認できます。適切な場合、カスタム イメージの左から右方向バージョンが含まれるフォルダーに、イメージの *_mirror.jpg または *_mirror.bmp バージョンを追加します。*

[ファイルを開く] ダイアログを起動する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield は、基本の MSI インストールのダイアログの 1 つから [ファイルを開く] ダイアログを起動するサポートを含みます。エンド ユーザーがダイアログの 1 つから [参照] ボタンをクリックすると、[ファイルを開く] ダイアログが起動します。[ファイルを開く] ダイアログを使って、エンド ユーザーはファイルを参照できます。エンド ユーザーがファイルを選択して [開く] ボタンをクリックすると、[ファイルを開く] ダイアログが閉じて、インストールがダイアログの編集フィールドにその完全パスとファイル名を書き込みます。インストールはまた、IS_BROWSE_FILEBROWSED プロパティの値を、エンド ユーザーが選択したファイルのパスとファイル名に設定します。

インストールに [ファイルを開く] ダイアログのサポートを組み込む場合、次の機能を指定するいくつかのプロパティを定義できます。

- このダイアログで表示されるデフォルト パスを指定することができます。
- [ファイルを開く] ダイアログの **[ファイルの種類]** ドロップダウン リストに表示する文字列を指定できます。
- エンド ユーザーがファイルを選択するためにフォルダーを参照するときに表示される、ファイルの拡張子を指定できます。その他のファイル拡張子を持つファイルは表示されず、またそれらを開くこともできません。
- [ファイルを開く] ダイアログが使用するデフォルトのファイル拡張子を指定できます。エンド ユーザーが拡張子を入力しなかった場合、[ファイルを開く] ダイアログがファイル名にこのデフォルト拡張子を追加します。

以下は、ダイアログの要件です:

- [ファイルを開く] ダイアログで、エンド ユーザーが新しいファイルを作成することはできません。つまり、ファイルが既存しないとき、エンド ユーザーが [ファイルを開く] ダイアログで新しいファイル名を手作業で入力することはできません。
- [ファイルを開く] ダイアログが起動するダイアログに複数の編集フィールド コントロールが含まれている場合、ファイルへの完全パスを含む編集フィールド コントロールの Tab Stop プロパティは最小値を持たなくてはなりません。



タスク　　エンド ユーザーダイアログに [ファイルを開く] ダイアログ機能を追加するには、以下の手順を実行します。

1. **FileBrowse** という名前の新しい .dll カスタム アクションをプロジェクトに追加します:

   a. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。

   b. **[カスタム アクション]** エクスプローラーを右クリックして、**[新しい MSI DLL]** をポイントしてから、**[Binary テーブルに保存]** をクリックします。新しいカスタム アクションが、*NewCustomActionN* という名前で追加されます (ここで *N* は連続番号です)。

   c. カスタム アクションの名前を **FileBrowse** に変更します。

   d. 右のペインで、このカスタム アクションについて次の設定を構成します:

      - DLL Filename: <ISRedistPlatformDependentFolder>¥FileBrowse.dll
      - 関数名: FileBrowse
      - 戻り値の処理: 同期 (終了コードを無視)
      - スクリプト内実行: 即時実行
      - 実行スケジュール: Always execute

      他のすべての設定については、デフォルト値をそのまま使用します。"MSI タイプ番号" 設定の値は **65** にします。

2. [ファイルを開く] ダイアログを起動するダイアログを作成または編集して、その動作とレイアウトを構成します:

   a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。

   b. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** アイテムを展開します。

   c. 既存のダイアログを選択するか、新しいダイアログを作成します。

      既存のダイアログそ選択して、そのダイアログに複数の編集フィールド コントロールが含まれている場合、ファイルへの完全パスを含む編集フィールド コントロールの **Tab Stop** プロパティは最小値を持たなくてならない点にご注意ください。

   d. このダイアログの下にある、レイアウトを構成する言語をクリックします。

   e. エンド ユーザーが実行時に選択する、完全パスとファイル名を含む編集フィールド コントロールを追加します。コントロールに関連付けられたプロパティ名がプロンプトされたとき、**IS_BROWSE_FILEBROWSED** を入力します。

   f. 編集フィールド コントロールのとなりに、プッシュボタン コントロールを追加します。このボタンが [ファイルを開く] ダイアログを起動します。

   g. プッシュボタン コントロールを選択してから、右側のグリッドで必要に応じてそのプロパティを編集します。たとえば、ボタンで表示するテキストを指定するには、**Text** プロパティの値を追加します。

   h. **[ダイアログ]** エクスプローラーで、構成するダイアログの下にある **[動作]** アイテムをクリックします。

   i. ダイアログ コントロールがリストされる中央のペインで、作成したプッシュボタン コントロールをクリックします。設定が、右のペインに表示されます。

j. **"イベント"** 設定で、**[新しいイベント]** ボタンをクリックしてから **[DoAction]** をクリックします。
InstallShield によって、**"イベント"** 設定の下に新しい行のセットが追加されます。

k. **"DoAction"** 設定に、次のように入力します:

1

l. **"アクション"** サブ設定で、次のアクション名を指定します:

FileBrowse

m. **"イベント"** 設定で、**[新しいイベント]** ボタンをクリックしてから **[SetProperty]** をクリックします。
InstallShield によって、**"イベント"** 設定の下に新しい行が追加されます。

n. **"SetProperty"** 設定に、次のように入力します:

1

o. **"プロパティ"** サブ設定で、次のアクション名を指定します:

IS_BROWSE_FILEBROWSED

p. **"値"** サブ設定で、次のアクション名を指定します:

[]IS_BROWSE_FILEBROWSED]



***メモ・** SetProperty イベントは、ダイアログの "編集" フィールド コントロールに表示される MSI プロパティ値を更新するため、ファイルの参照ウィンドウでファイルが選択された後に表示される MSI プロパティ値が更新されます。*

3. [ファイルを開く] ダイアログの動作を指定するいくつかのプロパティと、[ファイルを開く] ダイアログを起動するダイアログを構成します:

a. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。

b. IS_BROWSE_FILEBROWSED プロパティを探します。デフォルト値は 0 です。以下のいずれかを実行します。

- エンド ユーザーが最初に表示したときに、ダイアログの編集フィールド コントロールを空白のままにするには、IS_BROWSE_FILEBROWSED プロパティが含まれる行を右クリックしてから、**[プロパティの削除]** を選択します。
- 編集フィールド コントロールにデフォルトのパスとファイル名を表示するには、IS_BROWSE_FILEBROWSED プロパティの値をパスとファイル名に変更します。



***メモ・** このプロパティの値を手動で変更しなかったとき、または削除した場合、[ファイルを開く] ダイアログを起動するダイアログの編集フィールド コントロールのデフォルト値は 0 に設定されます。*

c. オプションとして、IS_BROWSE_FILEEXT と呼ばれるプロパティを追加し、エンド ユーザーがファイルを選択するためにフォルダーを参照する際に表示されるファイルの拡張子を識別するフィルター文字列に、その値を設定することもできます。その他のファイル拡張子を持つファイルは表示されず、またそれらを開くこともできません。

フィルター文字列には、有効なファイル名文字と、ワイルドカードとしてアスタリスク (*) の組み合わせが可能です。

複数のファイル拡張子を指定する場合は、セミコロンで区切ります。スペースは使用できません。たとえば、エンド ユーザーが .exe と .dll ファイルを選択できるようにするには、IS_BROWSE_FILEEXT プロパティの値として、以下の文字列を入力します:

***.exe;*.dll**

この例では、[ファイルを開く] ダイアログでエンドユーザーは .exe と .dll ファイルを選択できます。その他のファイル タイプは表示されません。

このプロパティを設定しなかった場合、[ファイルを開く] ダイアログでエンド ユーザーは任意のファイル タイプを選択できます。

d. オプションとして、IS_BROWSE_FILETYPE と呼ばれるプロパティを追加し、その値を [ファイルを開く] ダイアログの **[ファイルの種類]** ドロップダウン リストに表示する文字列に設定します。ドロップダウン リストに表示可能なオプションは 1 つだけです。

たとえば、エンド ユーザーが .tt または .doc ファイルを選択できるようにするには、IS_BROWSE_FILETYPE プロパティの値として次の文字列を入力します:

テキスト文書 (*.txt); Word 文書 (*.doc)

このプロパティを設定しなかった場合、[ファイルを開く] ダイアログの **[ファイルの種類]** ドロップダウン リストは空白です。

e. オプションとして、IS_BROWSE_DEFAULTEXTENSION と呼ばれるプロパティを追加して、その値を [ファイルを開く] ダイアログが使用するデフォルトのファイル拡張子に設定します。エンド ユーザーが拡張子を入力しなかった場合、[ファイルを開く] ダイアログがファイル名にこのデフォルト拡張子を追加します。たとえば、デフォルトのファイルカクチョウシとして .exe を使用するには、IS_BROWSE_DEFAULTEXTENSION プロパティの値として次の文字列を入力します:

exe

f. オプションとして、IS_BROWSE_INITIALDIR と呼ばれるプロパティを追加して、その値を [ファイルを開く] ダイアログが使用するデフォルト パスに設定します。たとえば、**C:¥Program Files¥My Product** を使用するには、IS_BROWSE_INITIALDIR プロパティの値として次の文字列を入力します:

C:¥Program Files¥My Product

実行時、エンド ユーザーが新しいプッシュボタン コントロールをクリックしたとき、[ファイルを開く] ダイアログが開きます。エンド ユーザーはファイルを参照して選択できます。インストールは IS_BROWSE_FILEBROWSED プロパティの値を、エンド ユーザーが選択したファイルのパスとファイル名に設定し、[ファイルを開く] ダイアログを起動したダイアログの編集フィールド コントロールにそのパスとファイル名を表示します。

LicenseAgreement ダイアログでエンドユーザーが EULA を最初から最後までスクロールするのを必須にする



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield では、エンドユーザーが次のいずれかの方法でスクロール可能な EULA コントロール内にあるエンドユーザー使用許諾契約 (EULA) テキストの終わりに到達するまで、LicenseAgreement ダイアログ上の [次へ] ボタンを無効にすることができます。

- スクロール バーを使用する。
- スクロール可能な EULA コントロールにフォーカスがあるとき、PAGE DOWN を押す。

- スクロール可能な EULA コントロールにフォーカスがあるとき、CTRL+PAGE DOWN を押す。
- スクロール可能な EULA コントロールにフォーカスがあるとき、DOWN ARROW を押す。
- スクロールを右クリックして、[最下部] をクリックする。

ユーザーはまた、[次へ] ボタンが有効にされる前に **[ソフトウェア使用許諾契約に同意します]** オプションを選択する必要があります。

デフォルトで LicenseAgreement ダイアログでは、エンドユーザーは **[同意する]** オプションを選択しなければなりません。エンドユーザー が EURA テキストの最後まで読むことも必須にする場合、次のタスクを実行します。



タスク

エンドユーザーがスクロール可能な EULA コントロール内の EURA テキストの最後まで到達することを必須にする場合、以下の手順に従います:

1. **WatchScroll** という名前の新しい .dll カスタム アクションをプロジェクトに追加します:
   a. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
   b. **[カスタム アクション]** エクスプローラーを右クリックして、**[新しい MSI DLL]** をポイントしてから、**[Binary テーブルに保存]** をクリックします。新しいカスタム アクションが、*NewCustomActionN* という名前で追加されます (ここで *N* は連続番号です)。
   c. カスタム アクションの名前を **WatchScroll** に変更します。
   d. 右のペインで、このカスタム アクションについて次の設定を構成します:
      - DLL Filename: <ISRedistPlatformDependentFolder>¥EulaScrollWatcher.dll
      - 関数名: WatchScroll
      - 戻り値の処理: 非同期 (終了コードを待機)
      - スクリプト内実行: 即時実行
      - 実行スケジュール: Always execute

      他のすべての設定については、デフォルト値をそのまま使用します。"MSI タイプ番号" 設定の値は **129** にします。
2. 適切なカスタム アクションが起動されるように LicenseAgreement ダイアログを編集します:
   a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
   b. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、**LicenseAgreement** アイテムを展開し、**[動作]** をクリックします。
   c. LicenseAgreement コントロールの一覧がある中央のペインで、**Memo** という名前の ScrollableText コントロールをクリックします。このコントロールが EULA のテキストを含むコントロールです。設定が、右のペインに表示されます。
   d. **"イベント"** 設定で、**[新しいイベント]** ボタンをクリックしてから **[DoAction]** をクリックします。InstallShield によって、**"イベント"** 設定の下に新しい行のセットが追加されます。
   e. **"DoAction"** 設定に、次のように入力します:

      Not LicenseViewed AND Not ISLicenseWatching
   f. **"アクション"** サブ設定で、次のアクション名を指定します:

WatchScroll

g. LicenseAgreement コントロールの一覧がある中央のペインで、**Next** という名前の PushButton コントロールをクリックします。

h. **"条件"** 設定で、**[新しい条件]** ボタンをクリックしてから **[Disable]** をクリックします。**"Disable"** 設定が追加されます。

i. **"Disable"** 設定に、次のように入力します:

AgreeToLicense <> "はい" OR Not LicenseViewed

j. **"条件"** 設定で、**[新しい条件]** ボタンをクリックしてから **[Enable]** をクリックします。**"Enable"** 設定が追加されます。

k. **"Enable"** 設定に、次のように入力します:

AgreeToLicense = "はい" AND LicenseViewed

実行時に、EULA コントロールがインストールによって非同期カスタム アクションで監視されます。このカスタム アクションの実行中、ISLicenseWatching プロパティが設定されます。カスタム アクションの完了すると、ISLicenseWatching プロパティは削除されます。これにより、エンドユーザーが EULA のテキストをスクロールするときに砂時計フリッカーが表示されるのを防ぐことができます。一旦エンドユーザーが EULA の終わりまで到達すると、インストールにより LicenseViewed プロパティが設定され、[次へ] ボタンの条件に基づき必要に応じて [次へ] ボタンが手動で有効にされます。[次へ] ボタンは Control テーブルに格納されているテキストで検索されます。したがって、コントロールの名前が **Next** であるかぎり、前述の手順をすべての言語トランスフォームで採用することができます。

[印刷] ボタンをダイアログに追加する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield X 以降で作成された基本の MSI プロジェクトでは、LicenseAgreement ダイアログに [印刷] ボタンが含まれます。このボタンを利用して、エンドユーザーがダイアログのスクロール テキスト コントロールの内容を印刷することを可能にします。このボタンの イベント はカスタム アクション ISPrint を実行します。このアクションは基本の MSI プロジェクトに含まれます。次は [印刷] ボタンを別のダイアログに追加する、および InstallShield DevStudio 9.0 以前で作成した 既存プロジェクトに追加する 手順です。

[印刷] ボタンを別のダイアログに追加する

カスタム アクション IS Print が LicenseAgreement 以外のダイアログで適切に作動させるためには、ユーザー定義の Windows Installer プロパティ IS_PRINT_DIALOG の値を ダイアログの名前に設定しなくてはなりません。(IS_PRINT_DIALOG が既存するプロパティではない場合、ISPrint は LicenseAgreement ダイアログ ボックスのスクロール テキスト コントロールの内容を印刷します。)



タスク ***[印刷] ボタンを別のダイアログへ追加するには、以下の手順を実行します。***

1. ダイアログ ボックスでボタンコントロールを作成し、オプションでその **"テキスト"** プロパティを**[印刷(&P)]** に設定します。詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」を参照してください。
2. [印刷] ボタンに DoAction イベントを追加し、イベントの **"アクション"** サブ設定で ISPrint を選択します。詳細については、「基本の MSI ダイアログでコントロール イベントをトリガーする」を参照してください。
3. ダイアログ ボックスおよびその前後のダイアログの [戻る] および [次へ] ボタンのイベントから IS_PRINT_DIALOG の値を変更します。
   a. どのダイアログが [印刷] ボタンを追加するダイアログの前に表示されるのかを判別します。これは、NewDialog イベントの引数で確認して、ダイアログの [戻る] ボタンを探すか、[カスタム アクションとシーケンス] ビューを開いて、ダイアログの順序を表示することで確認することができます。
   b. SetProperty イベント を追加します。**SetProperty** 設定で、イベント条件として次を指定します:

      1

   c. **[プロパティ名]** ボックスで、次のように入力します。

      IS_PRINT_DIALOG

   d. "値" 設定で、[印刷] ボタンを追加するダイアログの名前を指定します。
   e. [印刷] ボタンを追加するダイアログのあとにも [印刷] ボタンを含むダイアログがある場合、次を実行します。
      i. どのダイアログが [印刷] ボタンを追加するダイアログの後に表示されるのかを確認します。これは、NewDialog イベントの引数を確認して、ダイアログの [次へ] ボタンを探すか、[カスタム アクションとシーケンス] ビューを開いてダイアログの順序を表示することで確認することができます。
      ii. SetProperty イベントを次のダイアログの [戻る] ボタンに追加して、その **"プロパティ名"** 設定を **IS_PRINT_DIALOG**、**"値"** 設定を [印刷] ボタンを追加するダイアログの名前に設定します。

      次のダイアログが [印刷] ボタンを含まない場合、または次のダイアログ ボックスが LicenseAgreement ダイアログの場合は、[印刷] ボタンを追加するダイアログの [次へ] ボタンに SetProperty イベントを追加して、その **"プロパティ名"** 設定を **IS_PRINT_DIALOG**、"値" 設定を **LicenseAgreement** に設定します。
   f. [印刷] ボタンを追加するダイアログの前に表示される任意のダイアログに [印刷] ボタンが含まれているとき、また直前のダイアログに [印刷] ボタンがない、あるいはそれが LicenseAgreement ダイアログである場合、[印刷] ボタンを追加するダイログの [戻る] ボタンに SetProperty イベントを追加し、イベントの **"プロパティ名"** 設定を **IS_PRINT_DIALOG**、**"値"** 設定を **LicenseAgreement** に設定します。

既存プロジェクトへ [印刷] ボタンを追加する



タスク ***InstallShield DevStudio 9.0 以前を使って作成した既存プロジェクトへ [印刷] ボタンを追加するには、以下の手順を実行します。***

1. ISPrint カスタム アクションを作成するには、以下手順を実行します。
   a. カスタム アクション ウィザードを開始します。

b. **[基本情報]** パネルの **[名前]** ボックスに **ISPrint** を入力します。

c. **[アクションの種類]** パネルの **[種類]** ボックスで、**[Windows Installer ダイナミック リンク ライブラリの関数を呼び出す]** を選択します。

d. **[アクション パラメーター]** パネルで、**[参照]** ボタンをクリックし、フォルダーの Redist¥Language Independent¥i386 サブフォルダーにある SetAllUsers.dll ファイルを参照します。**[開く]** をクリックします。

プロジェクトでパス変数の使用が構成されている場合、InstallShield はパスの一部に定義済みパス変数 <ISRedistPlatformDependentFolder> を使用します。

e. **[アクションのパラメーター]** パネルの **[ターゲット]** ボックスで、**PrintScrollableText** を入力します。

f. 以降すべてをデフォルトの設定にしてウィザードを完成します。

2. ダイアログ ボックスでボタンコントロールを作成し、オプションでその **"テキスト"** プロパティを**[印刷(&P)]** に設定します。詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」を参照してください。

3. [印刷] ボタンに DoAction イベントを追加し、イベントの **"アクション"** サブ設定で **ISPrint** を選択します。詳細については、「基本の MSI ダイアログでコントロール イベントをトリガーする」を参照してください。

4. Windows Installer プロパティ IS_PRINT_DIALOG を作成して、その値をダイアログ ボックスの名前に設定します。詳細については、「Windows Installer ベースのプロジェクトにおけるプロパティの作成」を参照してください。

メモ: *ISPrint カスタム アクションがコントロールイベントによって実行される場合、Windows Installer 制限のため、カスタム アクションのログ記録は通常通りに インストールログ記録 へは記録されません。情報は ISPrintLogmNoten 形式のプロパティの値にログ記録されます。*

Windows Vista 以降のシステムの再起動を最小限にする

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

Windows ロゴ: *インストール終了後のシステム再起動は、エンドユーザーにとって不都合なものです。Windows ロゴ プログラムの要件の 1 つに、エンドユーザーがインストール完了後自動的にアプリケーションを閉じて再起動を行うことができるオプションを含まなくてはならないという項目があります。*

この要件をサポートするため、すべての基本の MSI プロジェクトには、デフォルトで [MSI 再起動マネージャー – 使用中のファイル] ダイアログが含まれています。インストール中に更新が必要なファイル (複数可) が他のアプリケーションによって使用中の場合、Windows Vista 以降のシステム上では [使用中のファイル (再起動マネージャー)] ダイアログが表示されます。ダイアログには、エンドユーザーが選択できる次の 2 つのオプションが含まれます。

- エンドユーザーは選択で、インストールの完了後、自動的にファイルを使用中のアプリケーションを閉じて、再起動することができる。
- エンドユーザーは、アプリケーションを閉じない選択ができる。インストールの終わりで再起動が必要。

エンド ユーザー エクスペリエンスを最適化するため、アプリケーションには再起動マネージャー API の利用が推奨されます。再起動マネージャーは、エンド ユーザーがアプリケーションを停止した時点から正確に、また効果的にこれを再開します。詳しい情報は、「About Restart Manager (再起動マネージャー)」および MSDN Web サイトで再起動マネージャーに関するその他の文書を参照してください。

Using Windows Installer with Restart Manager (Windows Installer ヘルプ ライブラリ)

ダイアログのコントロール

基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、またはマージ モジュール プロジェクトのいずれの場合でも、定義済みダイアログおよびカスタム ダイアログのレイアウトと動作は、さまざまな共通のコントロールを使って変更することができます。

テーブル 7 ・ ダイアログのコントロール

コントロールの種類	プロジェクトの種類	説明
ビルボード	基本の MSI、マージ モジュール	ビルボード コントロールは、コントロールイベントに応答して更新可能なデータを表示するために使用されます。たとえば、ビルボードを使用して、遅延しているカスタム アクションの進行を表示できます。  ***プロジェクト*** *インストールに* Setup.exe *ランチャーが含まれている場合、起動ツールがファイル転送処理中にビルボードを表示するように設定できます。これは、Windows Installer コントロールであるビルボード コントロールの代替となります。この* Setup.exe *ビルボード サポートは、基本の MSI、InstallScript、および InstallScript MSI インストールで使用できます。この種類のビルボードについての詳しい情報は、「ビルボードを表示する」を参照してください。*
ビットマップ	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ビットマップ コントロールはイメージを表示します。
チェック ボックス	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	チェック ボックス コントロールは、エンド ユーザーが選択またはクリアできるチェック ボックスを表示します。

テーブル 7 ・ ダイアログのコントロール (続き.)

コントロールの種類	プロジェクトの種類	説明
コンボ ボックス	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コンボ ボックス コントロールは、定義済みの値を表示するドロップ ダウン リストを含みます。このボックスはまた、エンド ユーザーが値を入力できる編集フィールドです。
ダイアログ	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログ コントロールは、構成された一連の設定と関連付けられています。
ディレクトリ コンボ	基本の MSI、マージ モジュール	ディレクトリ コンボは、ディレクトリ リストやパス編集 コントロールと一緒に使用して、参照ダイアログを作成します。ディレクトリ コンボは、使用中のシステムにマッピングしたドライブのリストを表示します。
ディレクトリ リスト	基本の MSI、マージ モジュール	ディレクトリ リストは、ディレクトリ コンボやパス編集 コントロールと一緒に使用して、参照ダイアログを作成します。ディレクトリ リストにはディレクトリ リスト コントロールで選択されたドライブ下にあるフォルダーが表示され、パス編集コントロールの値が入力されます。
編集フィールド	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	編集フィールド コントロールは、エンド ユーザーが文字列または整数を入力できるテキスト ボックスです。
グループ ボックス	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	グループ ボックスは、あるエリア内の複数のコントロールを囲むのに使用します。また、グループ ボックスには、そこに含まれるコントロール間の関係を表現するのに使用できるラベルがあります。

テーブル 7 - ダイアログのコントロール (続き.)

コントロールの種類	プロジェクトの種類	説明
ハイパーリンク	基本の MSI、マージ モジュール	ハイパーリンク コントロールは HTML リンクを表示します。このリンクを実行時にクリックすると、ターゲット システム上のデフォルト ブラウザでページが開きます。  ***重要*** *- Windows Installer 5 以降は、この新しいハイパーリンク コントロールをサポートします。*  ***プロジェクト*** *- InstallScript プロジェクト、または InstallScript MSI プロジェクトで HTML コントロールをダイアログに追加する方法については、「ダイアログで HTML コントロールを使用する」を参照してください。*
アイコン	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	アイコン コントロールは、アイコンの画像を表示します。
線	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	行コントロールは長さと太さが調節可能な行を作成します。行を使ってダイアログの領域を区分したり、グラフィカルな要素を追加します。
リスト ボックス	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	リスト ボックス コントロールは、エンド ユーザーが定義済みオプション リストから単一のオプションを選択できる、標準のリスト ボックスです。
リスト ビュー	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	リスト ビュー コントロールは、オプションの単一の列を表示します。各オプションの横にアイコンが表示されます。
マスク編集	基本の MSI、マージ モジュール	マスク編集コントロールは、基本的にエンド ユーザーが指定されたフォーマットで情報を入力できる編集フィールド コントロールです。

テーブル 7 ・ ダイアログのコントロール (続き.)

コントロールの種類	プロジェクトの種類	説明
パス編集	基本の MSI、マージ モジュール	パス編集コントロールは、ディレクトリ コンボ コントロールや ディレクトリ リスト コントロールと一緒に使用して、参照ダイ アログを作成します。パス編集には、ディレクトリ コンボおよ びディレクトリ リストでエンド ユーザーによる選択によって構 成された完全なパスが表示されます。これは、エンドユーザーに よる編集に沿ったものである場合も、そうでない場合もありま す。
進行状況バー	基本の MSI、 InstallScript、 InstallScript MSI、 InstallScript オブ ジェクト、マージ モジュール	進行状況バーは、コントロール イベントに応じて満たされる動 的なグラフィック バーです。
プッシュ ボタン	基本の MSI、 InstallScript、 InstallScript MSI、 InstallScript オブ ジェクト、マージ モジュール	プッシュ ボタン コントロールは、エンド ユーザーがクリックし たときにコマンドが実行されるボタンです。
ラジオ ボタン	基本の MSI、 InstallScript、 InstallScript MSI、 InstallScript オブ ジェクト、マージ モジュール	ラジオ ボタン コントロールは、エンド ユーザーがその中から 1 つだけを選択できる、2 つまたはそれ以上のオプションの 1 つで す。ラジオ ボタンは、ラジオ ボタン グループに挿入しなくては なりません。これは、ラジオ ボタン グループ コントロールの一 部として機能します。
ラジオ ボタン グ ループ	基本の MSI、 InstallScript、 InstallScript MSI、 InstallScript オブ ジェクト、マージ モジュール	ラジオ ボタン グループ コントロールは、ラジオ ボタン コント ロールを収容するコンテナです。ラジオ ボタン コントロールと そのラジオ ボタンは、単一のコントロールとして動作します。
スクロール可能テキ スト	基本の MSI、マージ モジュール	スクロール可能テキストは、ダイアログに収まりきらないテキス トの長い文字列を表示します。このコントロールに含まれている スクロールバーを使って、エンド ユーザーがテキストをスク ロールできます。LicenseAgreement ダイアログは、スクロール可 能テキスト コントロールを一般的に含むダイアログの例です。

テーブル 7 ・ ダイアログのコントロール (続き.)

コントロールの種類	プロジェクトの種類	説明
選択ツリー	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	選択ツリーは、CustomSetup ダイアログで見られるような、ユーザーが機能の選択状態を変更できる特殊なコントロールです。
テキスト領域	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	テキスト領域コントロールは、テキストを表示します。
ボリューム コスト リスト	基本の MSI、マージ モジュール	ボリューム コスト リストコントロールには、各ボリュームまたはドライブに関連付けられたディスク容量の要件が表示されます。
ボリューム選択コンボ	基本の MSI、マージ モジュール	ボリューム選択コンボ コントロールによって、エンドユーザーがボリューム、またはドライブをアルファベット順のリストから選択できます。

ビルボード コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

インストールに Setup.exe *ランチャーが含まれている場合、起動ツールがファイル転送処理中にビルボードを表示するように設定できます。これは、Windows Installer コントロールであるビルボード コントロールの代替となります。この* Setup.exe *ビルボード サポートは、基本の MSI、InstallScript、および InstallScript MSI インストールで使用できます。この種類のビルボードについての詳しい情報は、「ビルボードを表示する」を参照してください。*

ビルボード コントロールは、コントロールイベントに応答して更新可能なデータを表示するために使用されます。ビルボードはこの情報を表示する別のコントロールを含むことができますが、テキスト、ビットマップ、およびアイコンを含むそれらのコントロールは、Windows Installer にリンクされていない、スタティックコントロールである必要があります。たとえば、ビルボードを使用して、遅延しているカスタム アクションの進行を表示できます。

ビルボードは、ダイアログ エディター内で完全にはサポートされていません。ビルボードが Windows Installer アクションとインタラクトしてその他のコントロールを表示する場合、プロジェクトの Billboard および BBControl テーブルを変更しなくてはなりません。これらのテーブルの変更には、[ダイレクト エディター] ビューを使用します。

[ダイアログ] ビューにあるダイアログのビルボード コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 8 ・ビルボード コントロールの設定

設定	説明
Name	このビルボードの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。

テーブル 8 ・ ビルボード コントロールの設定 (続き.)

設定	説明
Text	この設定には、ビルボードのコントロールの初期値 (BBControl テーブルを参照) に使用されるテキストが含まれます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ビルボードのコントロールにビットマップまたはアイコンが含まれている場合、この値はコントロールに最初に表示されるファイルに対する Binary テーブルに入る外部キーです。
Text Style	ビルボードのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

Billboard Table (Windows Installer ヘルプ ライブラリ)
BBControl Table (Windows Installer ヘルプ ライブラリ)

ビットマップ コントロール



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

- *InstallScript オブジェクト*
- *マージ モジュール*

ビットマップ コントロールはイメージを表示します。

[ダイアログ] ビューにあるダイアログのビットマップ コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 9 ・ ビットマップ コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このビットマップの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンドユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
File Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに使用するイメージ ファイルのパスと名前を入力するか、この設定の省略記号ボタン (...) をクリックして、そのファイルを参照します。ビルド時にそのファイルがリリースに追加されます。ファイルはインストールでバイナリ リソースとして格納されなくてはなりません。

テーブル 9 ・ ビットマップ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Stretch to Fit	基本の MSI、マージ モジュール	コントロールの領域に収まるようイメージのサイズを変更するには、True を選択します。 イメージがコントロールのサイズよりも小さい場合に中央に配置するか、コントロールのサイズよりも大きい場合にそれをコントロールのサイズにトリミングする場合は、False を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。

テーブル 9 - ビットマップ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

メモ: *InstallScript プロジェクトと基本の MSI プロジェクトで、[ダイアログ] エクスプローラの* ***SdWelcome*** *および* ***InstallWelcome*** *ダイアログでビットマップ コントロールを使用する場合、ダイアログ ボックス上のダイアログ画像の位置が変更されます:*

- *InstallScript プロジェクトの場合、ダイアログ画像はダイアログ ボックスの左側にのみ表示されます。*
- *基本 MSI プロジェクトの場合、ダイアログ画像はダイアログ ボックス全体に表示されます。*

[チェック ボックス] ボタン



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

チェック ボックス コントロールは、エンド ユーザーが選択またはクリアできるチェック ボックスを表示します。

[ダイアログ] ビューにあるダイアログのチェック ボックス コントロールを選択すると、右側のペインに以下の設定が表示されます。



プロジェクト: *(基本の MSI およびマージ モジュール プロジェクト) ダイアログ上に初めてこのタイプのコントロールを作成したとき、InstallShield によって、Windows Installer プロパティの名前を付けるようプロンプトが表示されます。InstallShield は、この名前をコントロールの Property 設定の値として使用します。実行時に、インストールはエンド ユーザーの選択に基づいてこのプロパティの値を設定します。詳細については、「Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方」を参照してください。*

テーブル 10 ・ チェック ボックス コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このチェック ボックスの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。

テーブル 10 • チェック ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Control Style	基本の MSI、マージ モジュール	このコントロールをテキスト ラベル、アイコン、またはビットマップのいずれを使用してマークするかを指定します。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
File Name	基本の MSI、マージ モジュール	この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。 このコントロールに使用するイメージ ファイルのパスと名前を入力するか、この設定の省略記号ボタン (...) をクリックして、そのファイルを参照します。ビルド時にそのファイルがリリースに追加されます。ファイルはインストールでバイナリ リソースとして格納されなくてはなりません。 この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。

テーブル 10 ・ チェック ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Icon Size	基本の MSI、マージ モジュール	アイコン ファイルが複数のリソースを持つことを想定して、コントロールに使用するイメージのサイズを指定します。**[最初のイメージを使用]** オプションを選択すると、ファイル内の最初のイメージが表示され、Stretch to Fit 設定での指定に関わらず、コントロールの大きさに合うよう引き伸ばされます。 この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。
Indirect Property	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) をクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Property	基本の MSI、マージ モジュール	エンド ユーザーがこのチェック ボックスを選択したときに設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それを Value 設定に割り当てます。

テーブル 10 • チェック ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Property Is Integer	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールのプロパティ (Property 設定で指定します) に整数値が含まれる場合は、True を選択します。プロパティの値が文字列の場合、False を選択します。
Push Button	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このチェック ボックスをプッシュ ボタン コントロールに変更するには、True を選択し、そうでない場合は、False を選択します。
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Stretch to Fit	基本の MSI、マージ モジュール	コントロールの領域に収まるようイメージのサイズを変更するには、True を選択します。 イメージがコントロールのサイズよりも小さい場合に中央に配置するか、コントロールのサイズよりも大きい場合にそれをコントロールのサイズにトリミングする場合は、False を選択します。 この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。

テーブル 10 - チェック ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この設定は、コントロールのラベルに使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。

テーブル 10 ・ チェック ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Value	基本の MSI、マージ モジュール	チェック ボックスが選択されると、Property 設定で指定されたプロパティがこの値に設定されます。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

コンボ ボックス コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

コンボ ボックス コントロールは、定義済みの値を表示するドロップ ダウン リストを含みます。このボックスはまた、エンド ユーザーが値を入力できる編集フィールドです。

[ダイアログ] ビューにあるダイアログのコンボ ボックス コントロールを選択すると、右側のペインに以下の設定が表示されます。



プロジェクト: *(基本の MSI およびマージ モジュール プロジェクト) ダイアログ上に初めてこのタイプのコントロールを作成したとき、InstallShield によって、このコンボボックスに属するすべての項目を判別するための Windows Installer プロパティの名前を付けるようプロンプトが表示されます。InstallShield は、この名前をコントロールの Property 設定の値として使用します。実行時に、インストールはエンド ユーザーの選択に基づいてこのプロパティの値を設定します。詳細については、「Windows Installer プロパティおよびアドバンスト UI またはス*

イート/アドバンスト UI プロパティの使い方」を参照してください。(Windows Installer コンボ ボックスでは、エンド ユーザーは 1 つの項目しか選択できないことに注意してください。)

テーブル 11 ・コンボ ボックス コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコンボ ボックスの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Code Page	基本の MSI、マージ モジュール	コントロールでインストーラーのパッケージで定義されたコード ページからのフォントを使用する場合は、[データベース] を選択します。コントロールでターゲット システムのデフォルト コード ページのフォントを使用するには、[ユーザーのシステム] を選択します。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Drop-Down List	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールをドロップダウン リストにするには、True を選択します。コントロールを編集可能なコンボ ボックスにするには、False を選択します。

テーブル 11 • コンボ ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。  ***ヒント:*** *コンボ ボックス コントロールのドロップダウンリスト部分の高さを指定する、"Height" 設定に十分な大きさの数字 を指定するようにしてください。*
Indirect Property	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェックボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Items	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールにリストするオプションを指定するには、この設定で省略記号ボタン (...) をクリックして、[リスト アイテム] ダイアログ ボックスを開きます。 新しいアイテムをリストに追加するには、[リスト アイテム] ダイアログ ボックスにある [追加] ボタンをクリックします。各アイテムについて、コンボ ボックスに表示されているテキストおよび値 (このアイテムが選択されたときにプロパティに割り当てられる値) を入力する必要があります。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。

テーブル 11 ・ コンボ ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Left Scrollbar	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Max. Length	基本の MSI、マージ モジュール	エンドユーザーがコンボ ボックスに入力できる文字数を指定します。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) をクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Property	基本の MSI、マージ モジュール	エンド ユーザーがこのコンボボックスに値を入力した場合、またはコンボボックスから値を選択した場合に設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値 (Items 設定 で指定します) に割り当てます。
Property Is Integer	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールのプロパティ (Property 設定で指定します) に整数値が含まれる場合は、True を選択します。プロパティの値が文字列の場合、False を選択します。 Item 設定のすべての値が、Property Is Integer 設定で選択した値に基づいて整数または文字列のどちらかに設定されていることを確認してください。
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。

テーブル 11 • コンボ ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Sorted	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True を選択すると、コンボボックスの項目はアルファベット順に並べ替えられます。False の値を選択すると、各項目に対して [List Items] ダイアログ ボックスで指定した順序が保持されます。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。

テーブル 11 - コンボ ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

ダイアログ コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

[ダイアログ] ビューの [ダイアログ エディター] にあるダイアログを選択すると、右側のペインに以下の設定が表示されます。

コントロールと異なり、ダイアログ エディターからダイアログの名前を変更することはできません。名前を変更するには、[ダイアログ] ビューでダイアログを右クリックして [名前の変更] をクリックします。

テーブル 12 - ダイアログ コントロールの設定

設定	プロジェクトの種類	説明
Caption	基本の MSI、マージ モジュール	ダイアログのタイトルとして使用する名前を指定します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 12 • ダイアログ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
コメント	基本の MSI、マージ モジュール	ダイアログのコメントを入力します。入力したコメントは、参考用にプロジェクト ファイルに保存され、インストールでは使用されません。
Custom Palette	基本の MSI、マージ モジュール	カスタム パレットが必要になるのは、ダイアログに Windows Installer が作成するデフォルトのカラーパレットとは異なるカラーパレットを使用したイメージが含まれる場合のみです。ダイアログに異なるカラー パレットを使用するイメージが含まれていない場合、この値を False のままに残します。
Error Dialog	基本の MSI、マージ モジュール	ダイアログがエラー ダイアログとして機能する場合は、True を選択します。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	インストーラー ユニットのダイアログの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Keep Modeless	基本の MSI、マージ モジュール	この値が True で、ダイアログが DoAction イベントにより起動された場合、他のすべてのダイアログはそのままです。この場合に Keep Modeless 設定が False のとき、その他のダイアログは表示されません。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログを配置する位置が画面左端からのパーセンテージで指定します。値が 50 の場合、ダイアログは横方向の中央に表示されます。 ダイアログがインストール ウィザードの一部であり、前のダイアログが異なる位置にあったか、またはエンドユーザーが移動した場合には、この値は無視されます。
Left Scrollbar	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True を選択して、ダイアログの左側にスクロールバー (存在する場合) を設定します。デフォルト値が False の場合は、スクロールバーが右側に残ります。 縦長のスクロールバーがある場合に、これをダイアログの左側に表示するには、True を選択します。デフォルト値 False の場合は、スクロールバーが右側に配置されます。
Minimize	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True は、エンドユーザーがこのダイアログを最小化できることを意味します。False はオプションがタイトル バーにないことを意味します。

テーブル 12 • ダイアログ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Modal	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログがインストール ウィザードの一部であり、このダイアログの上に他のダイアログを配置しないようにする場合、True に設定します。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) をクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Resource Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この読み取り専用の設定は、ダイアログのリソース ID を表示します。これはダイアログに固有な数値 ID です。
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合キャプションはコントロールの左に寄せて配列されます。True に設定するとキャプションは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Text Style	InstallScript、InstallScript MSI	ダイアログに使用するフォントを指定します。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログを配置する位置を、画面上端からのパーセンテージで指定します。値が 50 の場合、ダイアログは縦方向の中央に表示されます。 ダイアログがインストール ウィザードの一部であり、前のダイアログが異なる位置にあったか、またはエンドユーザーが移動した場合には、この値は無視されます。
Track Disk Space	基本の MSI、マージ モジュール	何らかのアクションを実行する前、エンドユーザーに対してドライブの容量が足りない、または OutOfDiskSpace プロパティの値を確認する警告を表示するコントロールがこのダイアログにある場合は True を選択します。

テーブル 12 - ダイアログ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合ダイアログは表示され、False の場合は非表示になります。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	インストーラー ユニットのダイアログの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

[ディレクトリ コンボ] コントロール



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

ディレクトリ コンボは、ディレクトリ リストやパス編集 コントロールと一緒に使用して、参照ダイアログを作成します。ディレクトリ コンボは、使用中のシステムにマッピングしたドライブのリストを表示します。

ディレクトリ コンボをダイアログに最初に作成する場合、InstallShield はWindows Installer のプロパティを指定するプロンプトを表示します。このプロパティは、それに伴うディレクトリ リストとパス編集では同じにする必要があります。実行時に、インストールはエンド ユーザーの選択に基づいてこのプロパティの値を設定します。詳細については、「Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方」を参照してください。

[ダイアログ] ビューにあるダイアログのディレクトリ コンボ コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 13 - ディレクトリ コンボ コントロールの設定

設定	説明
Name	このディレクトリ コンボの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのテキストに使用されます。

テーブル 13 ・ ディレクトリ コンボ コントロールの設定 (続き.)

設定	説明
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Indirect Property	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。

テーブル 13 ・ ディレクトリ コンボ コントロールの設定 (続き.)

設定	説明
Property	エンド ユーザーがこのディレクトリ コンボの値を選択する場合に設定されるプロパティの名前を入力します。このコントロールはディレクトリ リストに表示されるパスの最初の部分を入力するので、ディレクトリ コンボ、ディレクトリ リスト、およびパス編集を参照ダイアログで一緒に使用する場合は同じプロパティを使用しなければなりません。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値に割り当てます。
Right-Aligned	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Show CD-ROM	ディレクトリコンボに CD-ROM のボリュームを含めるには、True を選択します。
Show Fixed	ディレクトリコンボにハード ドライブを含めるには、True を選択します。
Show Floppy	ディレクトリコンボにフロッピー ドライブを含めるには、True を選択します。
Show RAMDisk	ディレクトリコンボに RAM のボリュームを含めるには、True を選択します。
Show Remote	ディレクトリコンボにマップされたネットワーク ドライブを含めるには、True を選択します。
Show Removable	ディレクトリコンボにリムーバブル ドライブを含めるには、True を選択します。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。

テーブル 13 ・ ディレクトリ コンボ コントロールの設定 (続き.)

設定	説明
Text	この設定は、コントロールのテキストに使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

[ディレクトリ リスト] コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

ディレクトリ リストは、ディレクトリ コンボやパス編集 コントロールと一緒に使用して、参照ダイアログを作成します。ディレクトリ リストにはディレクトリ リスト コントロールで選択されたドライブ下にあるフォルダーが表示され、パス編集コントロールの値が入力されます。

ディレクトリ リストをダイアログに最初に作成する場合、InstallShield はWindows Installer のプロパティを指定するプロンプトを表示します。このプロパティは、それに伴うディレクトリ コンボとパス編集では同じにする必要があります。実行時に、インストールはエンド ユーザーの選択に基づいてこのプロパティの値を設定します。詳細については、「Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方」を参照してください。

[ダイアログ] ビューにあるダイアログのディレクトリ リスト コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 14 ・ ディレクトリ リスト コントロールの設定

設定	説明
Name	このディレクトリ リストの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Indirect Property	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。

テーブル 14 • ディレクトリ リスト コントロールの設定 (続き.)

設定	説明
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Property	エンド ユーザーがこのディレクトリ リストから値を選択する場合に設定されるプロパティの名前を入力します。このコントロールはパス編集に表示されるパスを入力し、ディレクトリ コンボでの選択に応じて変更されます。したがって、参照ダイアログでこの 3 つのコントロールを使用する場合には、すべてのコントロールに対して同じプロパティを使用する必要があります。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値に割り当てます。
Right-Aligned	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	この設定は、コントロールのテキストに使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 14 - ディレクトリ リスト コントロールの設定 (続き.)

設定	説明
Text Style	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

[編集フィールド] コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

編集フィールド コントロールは、エンド ユーザーが文字列または整数を入力できるテキスト ボックスです。

[ダイアログ] ビューにあるダイアログの編集フィールド コントロールを選択すると、右側のペインに以下の設定が表示されます。



プロジェクト: *(基本の MSI およびマージ モジュール プロジェクト) ダイアログ上に初めてこのタイプのコントロールを作成したとき、InstallShield によって、Windows Installer プロパティの名前を付けるようプロンプトが表示されます。InstallShield は、この名前をコントロールの Property 設定の値として使用します。実行時に、インストールはエンド ユーザーの選択に基づいてこのプロパティの値を設定します。詳細については、「Windows*

Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方」を参照してください。

テーブル 15 ・ 編集フィールド コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	この編集フィールドの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージモジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォントスタイルがコントロールのラベルに使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージモジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。

テーブル 15 • 編集フィールド コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Indirect Property	基本の MSI、マージ モジュール	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Max. Length	基本の MSI、マージ モジュール	エンド ユーザーがこのコントロールに入力できる最大文字数を指定します。
MultiLine	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールを複数行のコントロールとするには、True を選択します。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。

テーブル 15 • 編集フィールド コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Password	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	このコントロールをパスワード コントロールのように動作させ、各文字の代わりにアスタリスク (*) だけを表示するには、True を選択します。このコントロールを通常の編集フィールドコントロールとして使用するには、False を選択します。
Property	基本の MSI、マージモジュール	エンド ユーザーがこのコントロールにテキストを入力したときに設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値に割り当てます。
Property Is Integer	基本の MSI、マージモジュール	このコントロールのプロパティ (Property 設定で指定します) に整数値が含まれる場合は、True を選択します。プロパティの値が文字列の場合、False を選択します。
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。

テーブル 15 • 編集フィールド コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	基本の MSI、マージ モジュール	この設定は、コントロールで使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。

テーブル 15 • 編集フィールド コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Width	基本の MSI、マージ モジュール	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

[グループ ボックス] コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

グループ ボックスは、あるエリア内の複数のコントロールを囲むのに使用します。グループ ボックスを囲む個別のコントロールと一緒にグループ ボックスを移動および調整するには、グループ ボックスと各コントロールを選択する時に CTRL キーを押します。また、グループ ボックスには、そこに含まれるコントロール間の関係を表現するのに使用できるラベルがあります。

[ダイアログ] ビューにあるダイアログのグループ ボックス コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 16 • グループ ボックス コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このグループ ボックスの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。

テーブル 16 • グループ ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。

テーブル 16 - グループ ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この設定はグループ ボックスのラベルに使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 16 ・ グループ ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

ハイパーリンク コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

InstallScript プロジェクト、または InstallScript MSI プロジェクトで HTML コントロールをダイアログに追加する方法については、「ダイアログで HTML コントロールを使用する」を参照してください。

プロジェクト内のダイアログにハイパーリンクを追加するには、ハイパーリンク コントロールを使用します。ハイパーリンク コントロールは HTML リンクを表示します。このリンクを実行時にクリックすると、ターゲット システム上のデフォルト ブラウザでページが開きます。



重要: *Windows Installer 5 以降は、この新しいハイパーリンク コントロールをサポートします。このコントロールを Windows Installer の以前のバージョンを持つシステムで表示されるダイアログに使用すると、ランタイム エラー 2885 が発生して、インストールが中止されます。このため、ダイアログ上でハイパーリンク コントロールを使用するとき、インストールが Windows Installer 4.5 以前をターゲットとする場合、プロジェクトにハイパーリンク コントロールを含むバージョンと、含まないバージョンの 2 種類のダイアログを含めることをお勧めします。ダイアログに条件を追加して、ターゲット マシンの Windows Installer バージョンによってダイアログを表示または非表示とします。*

ハイパーリンク コントロールを含むダイアログに使用できるサンプル条件は以下の通りです:

VersionMsi >= "5.00"

ハイパーリンク コントロールを一切含まない代替のダイアログに使用できるサンプル条件は以下の通りです:

VersionMsi < "5.00"

[ダイアログ] ビューにあるダイアログのハイパーリンク コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 17 ・ ハイパーリンク コントロールの設定

設定	説明
Name	このハイパーリンク コントロールの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Code Page	コントロールでインストーラーのパッケージで定義されたコード ページからのフォントを使用する場合は、[データベース] を選択します。コントロールでターゲット システムのデフォルト コード ページのフォントを使用するには、[ユーザーのシステム] を選択します。
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。

テーブル 17 • ハイパーリンク コントロールの設定 (続き.)

設定	説明
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Format as Bytes	インストールが "テキスト" 設定に入力された値をバイトで表示する場合は、True を選択します。True を選択した場合、"テキスト" 設定には、数値のみを単位を付けずに指定する必要があります。そうすると、実行時に実行時に数値が 512 で除算され、サイズに応じて KB、MB、GB の正しい数字で表示されます。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
No Prefix	"テキスト" 設定にアンパサンドが含まれている場合に、アンパサンド文字 (&) として表示するには、True を選択します。 "テキスト" 設定でアンパサンド文字を使って、後続の文字を下線付きにして表示したい場合 (たとえば、「**クリック(&C)**」は「クリック(C)」と表示されます) には False を選択します。
No Text Wrap	このコントロールでテキストの折り返しを許可するかどうかを示します。True を選択した場合にテキストがテキスト領域の幅を越えると、文字列の末端が切り詰められて省略記号ボタン (...) に置き換わります。False に設定するとテキストが折り返され、テキスト領域の高さが十分ではない場合にはテキスト領域が拡張されます。
Property	エンド ユーザーがこのハイパーリンクをクリックしたときに設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。
Right-Aligned	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。

テーブル 17 ・ ハイパーリンク コントロールの設定 (続き)

設定	説明
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	この設定は、コントロールで使用するテキストを含みます。この設定でハイパーリンクに HTML マークアップを使用します。この設定のサンプル エントリは、以下のとおりです: **ABC 社の最新情報を見るには、ABC Company Web site を参照してください。** たとえば、**ABC 社の Web サイト** テキストは、**http://www.abccompany.com** Web サイトをポイントするハイパーリンクとして表示されます。 ハイパーリンクに唯一サポートされているプロトコールは HTML です。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Transparent	このコントロールを透過して背景を表示するには、True を選択します。 色付きのビットマップ上にコントロールを配置する場合、コントロールを透明にすることは避けたほうがよい場合があります。エンド ユーザーがディスプレイの配色を変更した場合、テキストが見えなくなる場合があります。たとえば、エンド ユーザーがアクセシビリティの理由でハイコントラスト パラメーターを設定した場合、このコントロールのテキストが見えなくなる場合があります。

テーブル 17 ・ ハイパーリンク コントロールの設定 (続き.)

設定	説明
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

アイコン コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

アイコン コントロールは、アイコンの画像を表示します。

[ダイアログ] ビューにあるダイアログのアイコン コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 18 ・ アイコン コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このアイコンの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。

テーブル 18 - アイコン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
File Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに使用するアイコン ファイルのパスと名前を入力するか、この設定の省略記号ボタン (...) をクリックして、そのファイルを参照します。ビルド時にそのファイルがリリースに追加されます。ファイルはインストールでバイナリ リソースとして格納されなくてはなりません。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Icon Size	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	アイコン ファイルが複数のリソースを持つことを想定して、コントロールに使用するイメージのサイズを指定します。**[最初のイメージを使用]** オプションを選択すると、ファイル内の最初のイメージが表示され、Stretch to Fit 設定での指定に関わらず、コントロールの大きさに合うよう引き伸ばされます。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。

テーブル 18 • アイコン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Stretch to Fit	基本の MSI、マージ モジュール	コントロールの領域に収まるようアイコンのサイズを変更するには、True を選択します。 アイコンがコントロールのサイズよりも小さい場合に中央に配置するか、コントロールのサイズよりも大きい場合にそれをコントロールのサイズにトリミングする場合は、False を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。

テーブル 18 ・ アイコン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

行コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

行コントロールは長さと太さが調節可能な行を作成します。行を使ってダイアログの領域を区分したり、グラフィカルな要素を追加します。

[ダイアログ] ビューにあるダイアログの行コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 19 ・ 行コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この行の名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。

テーブル 19 • 行コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。

テーブル 19 • 行コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

リスト ボックス コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

リスト ボックス コントロールは、エンド ユーザーが定義済みオプション リストから単一のオプションを選択できる、標準のリスト ボックスです。

[ダイアログ] ビューにあるダイアログのリスト ボックス コントロールを選択すると、右側のペインに以下の設定が表示されます。



プロジェクト: *(基本の MSI およびマージ モジュール プロジェクト) ダイアログ上に初めてこの種類のコントロールを作成したとき、InstallShield によって、このリスト ボックスに表示されるすべての項目を判別するための Windows Installer プロパティの名前を付けるようプロンプトが表示されます。InstallShield は、この名前をコントロールの Property 設定の値として使用します。実行時に、インストールはエンド ユーザーの選択に基づいてこのプロパティの値を設定します。詳細については、「Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方」を参照してください。*

テーブル 20 ・ リスト ボックス コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このリスト ボックスの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。

テーブル 20 • リスト ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Code Page	基本の MSI、マージ モジュール	コントロールでインストーラーのパッケージで定義されたコード ページからのフォントを使用する場合は、[データベース] を選択します。コントロールでターゲット システムのデフォルト コード ページのフォントを使用するには、[ユーザーのシステム] を選択します。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Indirect Property	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。

テーブル 20 • リスト ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Items	基本の MSI、マージ モジュール	このコントロールにリストするオプションを指定するには、この設定で省略記号ボタン (...) をクリックして、[リスト アイテム] ダイアログ ボックスを開きます。 新しいアイテムをリストに追加するには、[リスト アイテム] ダイアログ ボックスにある [追加] ボタンをクリックします。各アイテムについて、リスト ボックスに表示されているテキストおよび値 (このアイテムが選択されたときにプロパティに割り当てられる値) を入力する必要があります。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Property	基本の MSI、マージ モジュール	エンド ユーザーがこのコントロールからオプションを選択したときに設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択のテキストに割り当てます。
Property Is Integer	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールのプロパティ (Property 設定で指定します) に整数値が含まれる場合は、True を選択します。プロパティの値が文字列の場合、False を選択します。

テーブル 20 • リスト ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sorted	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True を選択すると、リスト ボックスの項目はアルファベット順に並べ替えられます。False の値を選択すると、各項目に対して [List Items] ダイアログ ボックスで指定した順序が保持されます。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。

テーブル 20 • リスト ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Text	基本の MSI、マージ モジュール	この設定は、スクリーン リーダーで使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

リスト ビュー コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

リスト ビュー コントロールは、オプションの単一の列を表示します。各オプションの横にアイコンが表示されます。

[ダイアログ] ビューにあるダイアログのリスト ビュー コントロールを選択すると、右側のペインに以下の設定が表示されます。



プロジェクト: *(基本の MSI およびマージ モジュール プロジェクト) ダイアログ上に初めてこの種類のコントロールを作成したとき、InstallShield によって、このリスト ビューに表示されるすべての項目を判別するための Windows Installer プロパティの名前を付けるようプロンプトが表示されます。InstallShield は、この名前をコントロールの Property 設定の値として使用します。実行時に、インストールはエンド ユーザーの選択に基づいてこのプロパティの値を設定します。詳細については、「Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方」を参照してください。*

テーブル 21 ・ リスト ビュー コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このリスト ビューの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。

テーブル 21 ・リスト ビュー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Icon Size	基本の MSI、マージ モジュール	アイコン ファイルが複数のリソースを持つことを想定して、コントロールに使用するイメージのサイズを指定します。**[最初のイメージを使用]** オプションを選択すると、ファイル内の最初のイメージが表示され、Stretch to Fit 設定での指定に関わらず、コントロールの大きさに合うよう引き伸ばされます。
Indirect Property	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。

テーブル 21・リスト ビュー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Items	基本の MSI、マージ モジュール	このコントロールにリストするオプションを指定するには、この設定で省略記号ボタン (...) をクリックして、[リスト アイテム] ダイアログ ボックスを開きます。 新しいアイテムをリストに追加するには、[リスト アイテム] ダイアログ ボックスにある [追加] ボタンをクリックします。各アイテムについて、リスト ビュー コントロールに表示されているテキストおよび値 (このアイテムが選択されたときにプロパティに割り当てられる値) を入力する必要があります。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Property	基本の MSI、マージ モジュール	エンド ユーザーがこのコントロールから値を選択したときに設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値に割り当てます。
Property Is Integer	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールのプロパティ (Property 設定で指定します) に整数値が含まれる場合は、True を選択します。プロパティの値が文字列の場合、False を選択します。

テーブル 21 ・リスト ビュー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sorted	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True を選択すると、コンボボックスの項目はアルファベット順に並べ替えられます。False の値を選択すると、各項目に対して [List Items] ダイアログ ボックスで指定した順序が保持されます。
Stretch to Fit	基本の MSI、マージ モジュール	コントロールの領域に収まるようイメージのサイズを変更するには、True を選択します。 イメージがコントロールのサイズよりも小さい場合に中央に配置するか、コントロールのサイズよりも大きい場合にそれをコントロールのサイズにトリミングする場合は、False を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。

テーブル 21・リスト ビュー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	基本の MSI、マージ モジュール	この設定は、コントロールで使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。

テーブル 21 ・ リスト ビュー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

マスク編集コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

マスク編集コントロールは、基本的にエンド ユーザーが指定されたフォーマットで情報を入力できる編集フィールド コントロールです。

[ダイアログ] ビューにあるダイアログのマスクされた編集コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 22 ・ マスクされた編集コントロールの設定

設定	説明
Name	このマスクされた編集フィールドの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。

テーブル 22 ・ マスクされた編集コントロールの設定 (続き.)

設定	説明
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Indirect Property	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Mask	マスクの形式を入力します。詳細については、Windows Installer ヘルプ ライブラリの「MaskedEdit Control」を参照してください。
Max. Length	エンド ユーザーがこのコントロールに入力できる最大文字数を指定します。
Property	エンド ユーザーがマスクされた編集コントロールに入力する値で設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値に割り当てます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。

テーブル 22・マスクされた編集コントロールの設定 (続き.)

設定	説明
Text Style	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

パス編集コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

パス編集コントロールは、ディレクトリ コンボ コントロールやディレクトリ リスト コントロールと一緒に使用して、参照ダイアログを作成します。パス編集には、ディレクトリ コンボおよびディレクトリ リストでエンド ユーザーによる選択によって構成された完全なパスが表示されます。これは、エンドユーザーによる編集に沿ったものである場合も、そうでない場合もあります。

[ダイアログ] ビューにあるダイアログのパス編集コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 23・パス編集コントロールの設定

設定	説明
Name	このパス編集フィールドの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。

テーブル 23 ・ パス編集コントロールの設定 (続き.)

設定	説明
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	このプロパティは将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Indirect Property	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。

テーブル 23 - パス編集コントロールの設定 (続き.)

設定	説明
Property	エンドユーザーがこのパス編集に入力する場合や、ディレクトリリストから選択する場合に設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値に割り当てます。
Right-Aligned	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	この設定は、コントロールのラベルに使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。

テーブル 23 ・ パス編集コントロールの設定 (続き.)

設定	説明
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

[進行状況バー] コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

進行状況バーは、コントロール イベントに応じて満たされる動的なグラフィック バーです。

[ダイアログ] ビューにあるダイアログの進行状況バーコントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 24 ・ 進行状況バー コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この進行状況バーの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。

テーブル 24 ・ 進行状況バー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Appearance	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	適切なオプションを選択します: • **継続**—進行状況バーは、継続するバーで構成されます。 • **セグメント化**—進行状況バーは、一連の長方形で構成されます。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、コントロールにこのフォント スタイルが使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。

テーブル 24 • 進行状況バー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	基本の MSI、マージ モジュール	この設定は、コントロールで表示されるテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。

テーブル 24 • 進行状況バー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

[プッシュ ボタン] コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

プッシュ ボタン コントロールは、エンド ユーザーがクリックしたときにコマンドが実行されるボタンです。

[ダイアログ] ビューにあるダイアログのプッシュ ボタン コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 25 • プッシュ ボタン コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このプッシュ ボタンの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Control Style	基本の MSI、マージ モジュール	このコントロールをテキスト ラベル、アイコン、またはビットマップのいずれを使用してマークするかを指定します。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。

テーブル 25 • プッシュ ボタン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
File Name	基本の MSI、マージ モジュール	このコントロールに使用するイメージ ファイルのパスと名前を入力するか、この設定の省略記号ボタン (...) をクリックして、そのファイルを参照します。ビルド時にそのファイルがリリースに追加されます。ファイルはインストールでバイナリ リソースとして格納されなくてはなりません。 この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。 この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Icon Size	基本の MSI、マージ モジュール	アイコン ファイルが複数のリソースを持つことを想定して、コントロールに使用するイメージのサイズを指定します。**[最初のイメージを使用]** オプションを選択すると、ファイル内の最初のイメージが表示され、Stretch to Fit 設定での指定に関わらず、コントロールの大きさに合うよう引き伸ばされます。 この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。

テーブル 25 - プッシュ ボタン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
UAC シールド アイコンの表示	基本の MSI、マージ モジュール	エンド ユーザーが Windows Vista 以降のシステムでインストールを実行しているとき、コントロールに [ユーザー アカウント制御 (UAC)] シールド アイコンを含める場合は、True を選択します。シールド アイコンによって、昇格された権限が必要かもしれないことがエンドユーザーに通知されます。 UAC シールド アイコンをコントロールに含めない場合は、False を選択します。
Stretch to Fit	基本の MSI、マージ モジュール	コントロールの領域に収まるようイメージのサイズを変更するには、True を選択します。 イメージがコントロールのサイズよりも小さい場合に中央に配置するか、コントロールのサイズよりも大きい場合にそれをコントロールのサイズにトリミングする場合は、False を選択します。 この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。

テーブル 25 • プッシュ ボタン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Text	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この設定は、コントロールで使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。

テーブル 25 ・ プッシュ ボタン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

[ラジオ ボタン] コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ラジオ ボタン コントロールは、エンド ユーザーがその中から 1 つだけを選択できる、2 つまたはそれ以上のオプションの 1 つです。ラジオ ボタンは、ラジオ ボタン グループに挿入しなくてはなりません。これは、ラジオ ボタン グループ コントロールの一部として機能します。

ラジオ ボタン コントロールとそのラジオ ボタンは、単一のコントロールとして動作します。ラジオ ボタン グループ内にある個別のボタンを表示または非表示にすることは不可能です。1 つのグループ内のボタンのスタイルはすべて同じです。たとえば、すべてのボタンにテキストを使うか、すべてのボタンにビットマップを使います。

ラジオ ボタン グループを削除すると、グループのすべてのラジオ ボタンも削除されるので注意してください。また、ラジオ ボタン グループ コントロールの名前を変更したとき、そののラジオ ボタンはすべて削除されます。

[ダイアログ] ビューにあるダイアログのラジオ ボタン コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 26 ・ ラジオ ボタン コントロールの設定

設定	プロジェクトの種類	説明
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。

テーブル 26 • ラジオ ボタン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。 グループの中の最初のラジオ ボタンのコントロール ID は、ラジオ ボタン グループのコントロール ID です。2 番目以降のラジオ ボタンのコントロールは、ラジオ ボタン グループの識別子を 1 ずつ加算したものになります。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Order	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ラジオ ボタンをラジオ ボタン グループで表示する順番を示す番号を指定します。この設定は一意の正の整数が設定されなければなりませんが、このグループ内の他のラジオ ボタンの Order 設定は連続的である必要はありません。

テーブル 26 ・ ラジオ ボタン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Text	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この設定は、ラジオ ボタンのラベルに使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Value	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ユーザーがラジオ ボタンを選択したときに、それと関連付ける値を入力します。  ***プロジェクト:*** *基本んお MSI プロジェクトの場合: デフォルトでボタンの 1 つを選択にするには、ラジオ ボタン グループの Windows Installer プロパティを [プロパティ マネージャー] ビュー に入力します。スコープ内でプロパティをパブリックにする場合は、すべて大文字で入力する必要があります。その後、デフォルトで表示するボタンの値を指定します。たとえば、ラジオ ボタンのひとつが 104 という値を持ち、ラジオ ボタン グループがプロパティ* GRP_PROPERTY1 *を使う場合、次の情報をプロパティ マネージャーで入力して、このラジオ ボタンをデフォルト選択にします。* • *Name: GRP_PROPERTY1* • *値: 104*

テーブル 26 ・ ラジオ ボタン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

[ラジオ ボタン グループ] コントロール



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

基本の MSI とマージ モジュール プロジェクトの場合、ラジオ ボタン グループ内にある個別のボタンを表示または非表示にすることは不可能です。1 つのグループ内のボタンのスタイルはすべて同じです。たとえば、すべてのボタンにテキストを使うか、すべてのボタンにビットマップを使います。

ラジオ ボタン グループ コントロールは、ラジオ ボタン コントロールを収容するコンテナです。ラジオ ボタン コントロールとそのラジオ ボタンは、単一のコントロールとして動作します。ラジオ ボタン グループを削除すると、グループのすべてのラジオ ボタンも削除されるので注意してください。また、ラジオ ボタン グループ コントロールの名前を変更したとき、そののラジオ ボタンはすべて削除されます。

[ダイアログ] ビューにあるダイアログのラジオ ボタン グループ コントロールを選択すると、右側のペインに以下の設定が表示されます。



***プロジェクト・** (基本の MSI およびマージ モジュール プロジェクト) ダイアログ上に初めてこの種類のコントロールを作成したとき、InstallShield によって、このラジオ ボタン グループに表示されるすべてのボタンを判別するための Windows Installer プロパティの名前を付けるようプロンプトが表示されます。InstallShield は、この名前をコントロールの Property 設定の値として使用します。実行時に、インストールはエンド ユーザーの選択に基づいて*

このプロパティの値を設定します。詳細については、「Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方」を参照してください。

テーブル 27 ・ ラジオ ボタン グループ コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このラジオ ボタンの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。 この設定は、ラベルにイメージを含むラジオ ボタン グループには影響しません。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。 グループの中の最初のラジオ ボタンのコントロール ID は、ラジオ ボタン グループのコントロール ID です。2 番目以降のラジオ ボタンのコントロールは、ラジオ ボタン グループの識別子を 1 ずつ加算したものになります。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。

テーブル 27 • ラジオ ボタン グループ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Has Border	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ラジオ ボタン グループの境界を表示するには、True を選択します。境界を省略するには、False を選択します。 Sunken 設定を使って、境界の種類をより詳細に変更できます。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Indirect Property	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェックボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。

テーブル 27 - ラジオ ボタン グループ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Property	基本の MSI、マージ モジュール	エンド ユーザーがこのグループ内のラジオ ボタンの 1 つを選択すると設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 この設定を使ってデフォルト選択をラジオ ボタン グループに設定する方法については、ラジオ ボタンの Value 設定を参照してください。
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。

テーブル 27 ・ ラジオ ボタン グループ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Text	基本の MSI、マージ モジュール	この設定は、ラジオ ボタン グループのラベルに使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 この設定は、ラベルにイメージを含むラジオ ボタン グループには影響しません。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。 この設定は、ラベルにイメージを含むラジオ ボタン グループには影響しません。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。

テーブル 27 ・ ラジオ ボタン グループ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

[スクロール可能テキスト] コントロール



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

スクロール可能テキストは、ダイアログに収まりきらないテキストの長い文字列を表示します。このコントロールに含まれているスクロールバーを使って、エンド ユーザーがテキストをスクロールできます。LicenseAgreement ダイアログは、スクロール可能テキスト コントロールを一般的に含むダイアログの例です。

[ダイアログ] ビューにあるダイアログのスクロール可能テキスト コントロールを選択すると、右側のペインに以下の設定が表示されます。



***メモ・** 多くのコントロールのテキスト値とは異なり、Windows Installer ではスクロール可能テキスト コントロール内のプロパティ名またはその他の値を解決しません。そのため、ファイルのテキストは書かれたとおりに表示されます。*

テーブル 28 ・ スクロール可能テキスト コントロールの設定

設定	説明
Name	このスクロール可能テキスト コントロールの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトル バーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。

テーブル 28 • スクロール可能テキスト コントロールの設定 (続き.)

設定	説明
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
File Name	このコントロールに使用する .rtf ファイルのパスと名前を入力するか、この設定の省略記号ボタン (...) をクリックして、そのファイルを参照します。ビルド時にそのファイルがリリースに追加されます。ファイルはインストールでバイナリ リソースとして格納されなくてはなりません。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Right-Aligned	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。

テーブル 28 • スクロール可能テキスト コントロールの設定 (続き.)

設定	説明
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

[選択ツリー] コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

選択ツリーは、CustomSetup ダイアログで見られるような、ユーザーが機能の選択状態を変更できる特殊なコントロールです。

[ダイアログ] ビューにあるダイアログのツリー コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 29 ・ Selection Tree Control の設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この選択ツリーの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。

テーブル 29 • Selection Tree Control の設定 (続き.)

設定	プロジェクトの種類	説明
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Indirect Property	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Property	基本の MSI、マージ モジュール	選択ツリーでのプロパティの名前を入力します。エンド ユーザーが参照ダイアログでプロパティを設定することができます。(このダイアログでは、パス編集、ディレクトリ コンボ、およびディレクトリ リスト、コントロールを使用しています。)

テーブル 29 ・ Selection Tree Control の設定 (続き.)

設定	プロジェクトの種類	説明
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	基本の MSI、マージモジュール	この設定は、スクリーン リーダーで使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 29 • Selection Tree Control の設定 (続き.)

設定	プロジェクトの種類	説明
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

[テキスト領域] コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

- *InstallScript オブジェクト*
- *マージ モジュール*

テキスト領域コントロールは、テキストを表示します。

[ダイアログ] ビューにあるダイアログのテキスト領域コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 30 ・ テキスト エリア コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このテキスト エリアの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、テキストにこのフォント スタイルが使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Code Page	基本の MSI、マージ モジュール	コントロールでインストーラーのパッケージで定義されたコード ページからのフォントを使用する場合は、[データベース] を選択します。コントロールでターゲット システムのデフォルト コード ページのフォントを使用するには、[ユーザーのシステム] を選択します。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。

テーブル 30 ・ テキスト エリア コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Format as Bytes	基本の MSI、マージ モジュール	インストールが "テキスト" 設定に入力された値をバイトで表示する場合は、True を選択します。True を選択した場合、"テキスト" 設定には、数値のみを単位を付けずに指定する必要があります。そうすると、実行時に実行時に数値が 512 で除算され、サイズに応じて KB、MB、GB の正しい数字で表示されます。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) をクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
No Prefix	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	"テキスト" 設定にアンパサンドが含まれている場合に、アンパサンド文字 (&) として表示するには、True を選択します。 "テキスト" 設定でアンパサンド文字を使って、後続の文字を下線付きにして表示したい場合 (たとえば、「**クリック(&C)**」は「クリック(C)」と表示されます) には False を選択します。
No Text Wrap	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールでテキストの折り返しを許可するかどうかを示します。True を選択した場合にテキストがテキスト領域の幅を越えると、文字列の末端が切り詰められて省略記号ボタン (...) に置き換わります。False に設定するとテキストが折り返され、テキスト領域の高さが十分ではない場合にはテキスト領域が拡張されます。

テーブル 30 ・ テキスト エリア コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
プロパティ	基本の MSI、マージ モジュール	テキストの初期値を提供するプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この設定は、コントロールで内部に表示されるテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 30 ・ テキスト エリア コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Transparent	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールを透過して背景を表示するには、True を選択します。 色付きのビットマップ上にコントロールを配置する場合、コントロールを透明にすることは避けたほうがよい場合があります。エンド ユーザーがディスプレイの配色を変更した場合、テキストが見えなくなる場合があります。たとえば、エンド ユーザーがアクセシビリティの理由でハイコントラスト パラメーターを設定した場合、このコントロールのテキストが見えなくなる場合があります。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

[ボリューム コスト リスト] コントロール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

ボリューム コスト リストコントロールには、各ボリュームまたはドライブに関連付けられたディスク容量の要件が表示されます。

リストには、5 つの列があります。[ダイレクト エディター] ビューでアクセス可能な UIText テーブルの各列のヘッダーは構成可能です。UIText テーブルは、文字列エントリの値を識別し、適切な場合に翻訳された文字列が実行時に表示されるようにします。各列における UIText テーブルのキーと英語のデフォルト値を以下に示します:

- VolumeCostVolume—Volume
- VolumeCostSize—Disk Size
- VolumeCostAvailable—Available
- VolumeCostRequired—Required
- VolumeCostDifference—Differences

[ダイアログ] ビューにあるダイアログのボリューム リスト コスト コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 31・Volume Cost List コントロールの設定

設定	説明
Name	このボリューム コスト リストの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Available Col Width	Available 列におけるデフォルトの幅をインストーラー単位で指定します (システム フォントの高さの 1/12 に定義されています)。この列をゼロに設定するか空白のままにすると、列は非表示となります。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルをボリューム コスト リストのコンテンツに使用します。
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。

テーブル 31 • Volume Cost List コントロールの設定 (続き.)

設定	説明
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Differences Col Width	Differences 列におけるデフォルトの幅をインストーラー単位で指定します (システム フォントの高さの 1/12 に定義されています)。この列をゼロに設定するか空白のままにすると、列は非表示となります。
Disk Size Col Width	Differences 列におけるデフォルトの幅をインストーラー単位で指定します (システム フォントの高さの 1/12 に定義されています)。この列をゼロに設定するか空白のままにすると、列は非表示となります。
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Required Col Width	Required 列におけるデフォルトの幅をインストーラー単位で指定します (システム フォントの高さの 1/12 に定義されています)。この列をゼロに設定するか空白のままにすると、列は非表示となります。
Right-Aligned	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Show CD-ROM	コントロールに CD-ROM のボリュームを含めるには、True を選択します。
Show Fixed	コントロールにハード ドライブを含めるには、True を選択します。
Show Floppy	コントロールにフロッピー ドライブを含めるには、True を選択します。
Show RAMDisk	コントロールに RAM のボリュームを含めるには、True を選択します。
Show Remote	コントロールにマップされたネットワーク ドライブを含めるには、True を選択します。

テーブル 31 • Volume Cost List コントロールの設定 (続き.)

設定	説明
Show Removable	コントロールにリムーバブル ドライブを含めるには、True を選択します。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	この設定は、コントロールのラベルに使用するテキストを含みます。ラベルは表示されませんが、スクリーン リーダによって使用されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Volume Col Width	Volume 列におけるデフォルトの幅をインストーラー単位で指定します (システム フォントの高さの 1/12 に定義されています)。この列をゼロに設定するか空白のままにすると、列は非表示となります。

テーブル 31 ・ Volume Cost List コントロールの設定 (続き.)

設定	説明
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

[ボリューム選択コンボ] コントロール



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

ボリューム選択コンボ コントロールによって、エンドユーザーがボリューム、またはドライブをアルファベット順のリストから選択できます。

[ダイアログ] ビューにあるダイアログのボリューム選択コンボ コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 32 ・ ボリューム選択コンボ コントロールの設定

設定	説明
Name	このボリューム選択コンボの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのテキストに使用されます。
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。

テーブル 32 ・ ボリューム選択コンボ コントロールの設定 (続き.)

設定	説明
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Indirect Property	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Property	エンド ユーザーがこのコントロールから値を選択したときに設定されるプロパティの名前を入力します。このコントロールはディレクトリ リストに表示されるパスの最初の部分を入力するので、ボリューム選択コンボ、ディレクトリ リスト、およびパス編集を参照ダイアログで一緒に使用する場合は同じプロパティを使用しなければなりません。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値に割り当てます。
Right-Aligned	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Show CD-ROM	コントロールに CD-ROM のボリュームを含めるには、True を選択します。
Show Fixed	コントロールにハード ドライブを含めるには、True を選択します。
Show Floppy	コントロールにフロッピー ドライブを含めるには、True を選択します。
Show RAMDisk	コントロールに RAM のボリュームを含めるには、True を選択します。
Show Remote	コントロールにマップされたネットワーク ドライブを含めるには、True を選択します。
Show Removable	コントロールにリムーバブル ドライブを含めるには、True を選択します。

テーブル 32 ・ ボリューム選択コンボ コントロールの設定 (続き.)

設定	説明
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	この設定は、コントロールで使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

コントロールのコピーと貼り付け

ダイアログ エディターでは、標準的な Windows キーボード ショートカットやコンテキスト (右クリック) メニューを使用してコントロールをコピーし、同じダイアログ内で、またはあるダイアログから別のダイアログに貼り付けることができます。



タスク ***あるダイアログから別のダイアログにプッシュ ボタンをコピーするには以下の手順に従います:***

1. コピーするボタンが含まれるダイアログのレイアウトを開きます。
2. プッシュボタンを選択して CTRL+C を押します。
3. ダイアログ エディターで 2 番目のダイアログを開き、CTRL+V を押してコントロールをダイアログに貼り付けます。

プッシュボタンは、元のボタンと同じ相対サイズと位置を使って貼り付けられ、他のプロパティも、元のプロジェクトと同一です。プッシュ ボタンのコピーには、コントロールの種類に基づいて一意の名前が付けられます。ただし、新規コントロールの動作の情報は (基本の MSI プロジェクトに) コピーされないので 、編集の必要があります。



メモ: *ラジオ ボタン グループは特殊なケースです。このグループをコピーすると、そのラジオ ボタンのすべてが自動的にコピーされます。特定のラジオ ボタンだけをコピーすることはできません。*

コントロールの切り取りと貼り付け

ダイアログ エディターでは、標準的な Windows キーボード ショートカットやコンテキスト (右クリック) メニューを使用してコントロールをあるダイアログから別のダイアログに移動することができます。



タスク ***コントロールをあるダイアログから切り取って、別のダイアログに貼り付けるには、以下の手順を実行します。***

1. コントロールを右クリックしてから **[切り取る]** をクリックするか、CTRL+X を押します。**[コントロールの削除]** ダイアログ ボックスが表示されて、コントロールを削除すると、関連した条件やアクションも同時に削除されることを通知するメッセージが表示されます。
2. コントロールを引き続き現在のダイアログから切り取る場合は **[はい]** をクリックします。
3. ダイアログ エディターで 2 番目のダイアログを開き、CTRL+V を押してコントロールをダイアログに貼り付けます。

コントロールを新しい場所に貼り付けたら、コントロールのプロパティと動作を指定します。

ウィザード インターフェイスを使って作業する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*

- *スイート/アドバンスト UI*



***エディション**- アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ドキュメントのこのセクションでは、アドバンスト UI およびスイート/アドバンスト UI プロジェクトのウィザード インターフェイスを使った作業について説明します。ウィザード インターフェイス要素は、ウィザード ページおよび 2 番目のウィンドウ（ポップアップ ウィンドウとも呼ばれます）で構成されます。InstallShieldでは、様々なスタイル、ブラシ、およびコントロール テーマを定義ならびにカスタマイズして、ウィザード インターフェイスを効率よくフォーマットできます。また、InstallShield を使ってウィザード ページおよび 2 番目のウィンドウを追加、変更または削除し、各要素が表示されるタイミングをスケジュールできます。

ウィザード インターフェイスの要素



***プロジェクト**- この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



***エディション**- アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

次の 2 つのスクリーンショットで、ウィザード インターフェイスの基本要素を示します。



図 26: アドバンスト UI またはスイート/アドバンスト UI インストールのサンプル ウィザード ページ



図 27: アドバンスト UI およびスイート/アドバンスト UI インストールのサンプル 2 番目のウィンドウ

InstallShield では、独自のウィザード アイコン、ヘッダー イメージ、およびウィザード キャプションを使用できます。また、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのユーザー インターフェイスのスタイル、ブラシ、その他をヘッダー、本文、およびナビゲーション領域で定義およびカスタマイズすることも可能です。

スタイル、ブラシ、コントロール テーマ、および最小ウィザード ページを使ってウィザード インターフェイスをカスタマイズする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、様々なユーザー インターフェイス関連のスタイル、ブラシ、およびコントロールのテーマを定義およびカスタマイズして、個別のユーザー インターフェイス要素を手動で編集することなく、ウィザード インターフェイスの外観を簡単に変更することができます。スタイル、ブラシ、およびコントロールのテーマを使用して、インストール全体を通してウィザード インターフェイスのフォーマットを統一することができます。

スタイルウィザード インターフェイスのスタイル

InstallShield には、ウィザード UI 上のテキスト属性 (たとえば、テキストの色、サイズ、フォント名) を定義する多くのビルトイン テキスト スタイルが含まれています。また、InstallShield には 1 つのビルトイン フォント セットが含まれています。フォント セット とは、フォント名、サイズ、太さなどの属性を含むフォントのコレクションです。フォント セットに含まれる各フォントについて、それが適用される言語を指定できます。これによって、プロジェクトでサポートする各言語に異なるフォントを選択できます。フォント セットは、テキスト スタイルと組み合わせて使用します。テキスト スタイルはフォント セットを参照しますが、オプションでフォント セットのレベルで定義されている様々なフォント属性をオーバーライドすることが可能です。任意のビルトイン テキスト スタイルとフォント セットを変更して、必要に応じて独自のテキスト スタイルとフォント セットを作成できます。詳しくは、次を参照してください:

- カスタム スタイルをウィザード インターフェイスに定義する
- テキスト スタイルをウィザード インターフェイスのテキストに適用する

ブラシウィザード インターフェイスのブラシ

InstallShield には、いくつかのビルトン ブラシが用意されています。ブラシを使って、ウィザード インターフェイスの様々な要素、たとえばウィザード ページやコントロール背景に純色、グラデーション、またはイメージを指定します。任意のビルトイン ブラシを変更して、必要に応じて独自のテキスト スタイルとフォント セットを作成できます。詳しくは、次を参照してください:

- カスタム ブラシをウィザード インターフェイスに定義する
- ウィザード インターフェイスのヘッダー、本文、およびナビゲーション領域に背景ブラシを指定する
- ウィザード ページまたは 2 番目のウィンドウのデフォルト 背景ブラシをオーバーライドする

ウィザード インターフェイスのコントロールのテーマ

InstallShield では、ウィザード インターフェイス コントロールのテーマを定義することができます。コントロールのテーマは、コントロールの様々な部分の色（テキストの色、背景の色、および境界線の色）や、クリック済みまたはホバー状態といった、コントロールの様々な状態に使用する色を集めたものです。デフォルトで、ユーザー インターフェイスのコントロールに使用するコントロールのテーマを指定できます。ユーザー インターフェイス内のウィザード ページやウィンドウに含まれる特定のコントロールのテーマをオーバーライドすることもできます。コントロールのテーマを使用すると、インストールのユーザー インターフェイス内のその他のアイテムが 3D 効果なしで平面的に表示されます。これは Windows Store アプリケーションのスタイルに似ています。

次の種類のコントロールで、コントロールのテーマを使用できます:

- ボタン（ナビゲーション ボタンを含む）
- チェック ボックス
- ラジオ ボタン
- コマンド リンク

詳しくは、次を参照してください:

- カスタム コントロール テーマをウィザード インターフェイスに定義する
- テーマをウィザード インターフェイスのコントロールに適用する

ウィザード インターフェイスのルック アンド フィールをカスタマイズする

[ウィザード インターフェイス] ビューの [ウィザード ページ] ノードを使って、デフォルトのプロジェクト全体を通して使用されるスタイル、ブラシ、およびコントロールのテーマを選択することができます。このビューにある [ウィザード ページ] ノードを使って、ウィザード キャプション、ウィザード アイコン、ヘッダー イメージ、およびナビゲーション バーの高さのデフォルト値をオーバーライドし、その他のプロジェクト全体を通して使用する UI 関連のデフォルト値を指定することもできます。

スタイルの特定の使用方法をオーバーライドしたい場合は、特定のウィザード ページ、2 番目のウィンドウ、およびウィザード コントロールで使用可能な "本文の背景" 設定などのページ固有、およびコントロール固有の設定を使用します。たとえば、InstallationWelcome ウィザード ページの "本文の背景" 設定を使って、プロジェクトのデフォルト背景を InstallationWelcome ウィザード ページでのみオーバーライドすることができます。

カスタム スタイルをウィザード インターフェイスに定義する



***プロジェクト**- この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



***エディション**- アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、カスタム フォント セットとテキスト スタイルをプロジェクトに追加してから、これらのフォント セットとテキスト スタイルを様々なウィザード インターフェイス要素に適用することができます。フォント セット とテキスト スタイルは、フォーマットの一貫性を保ち、ウィザード インターフェイス全体にテキスト フォーマットの変更を適応するのに役立ちます。



タスク ***カスタム スタイルを定義するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで **[スタイル]** を右クリックしてから、**[フォント セットの追加]** または **[テキスト スタイルの追加]** をクリックします。

   フォント セット とは、フォント名、サイズ、太さなどの属性を含むフォントのコレクションです。フォント セットに含まれる各フォントについて、それが適用される言語を指定できます。これによって、プロジェクトでサポートする各言語に異なるフォントを選択できます。

   テキスト スタイルはフォント セットを参照しますが、オプションでフォント セットのレベルで定義されている様々なフォント属性をオーバーライドすることが可能です。

   InstallShield によって **[スタイル]** の下に新しいスタイルが追加されます。必要に応じてスタイルの名前を変更します。
3. 新しいスタイルを選択してから、右のペインで設定を構成します。

フォント セットを追加した後、[ウィザード インターフェイス] ビューで定義された任意のテキスト スタイルの "フォント一覧" 設定で、これを選択できます。後で、そのフォント セットに基づいたテキスト スタイルに使用するフォントの一覧を変更したい場合、フォント セットを選択して、その設定を編集します。

テキスト スタイルを追加すると、[ウィザード インターフェイス] ビューで様々なスタイル関連の設定にそれらを選択できます。後でそのスタイルを使用するウィザード インターフェイス要素の外観を変更したい場合は、スタイルを選択してその設定の値を編集できます。

スタイルに加えた変更は、そのスタイルを使用するすべてのウィザード インターフェイス要素に反映されます。

テキスト スタイルをウィザード インターフェイスのテキストに適用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、様々なウィザード インターフェイス要素にテキスト スタイルを適用することができます。テキスト スタイルが使用されているウィザード インターフェイス要素の外観を後で変更したい場合は、そのテキスト スタイルが使用されている各要素を個別に編集する代わりに、テキスト スタイルを簡単に変更することができます。

テキスト スタイルをテキストに適用するとき、ウィザード インターフェイス全体に適用するか、各ウィザード ページのコントロール レベルで適用するかを選択できます。



タスク ***テキスト スタイルをテキスト全体に適用するには、次の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を選択します。InstallShield の右側のペインに、グローバル ウィザード インターフェイス設定が表示されます。
3. 次の設定で、適切なテキスト スタイルを選択します:
   - ヘッダー テキスト スタイル
   - ナビゲーション テキスト スタイル

InstallShield によって、プロジェクト全体で該当する UI 要素のテキスト スタイルが変更されます。



タスク ***テキスト スタイルを特定のテキストに適用するには、次の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** ノードまたは **[2 番目のウィンドウ]** ノードを展開して、スタイルを適用するテキストが含まれているウィザード ページまたはウィンドウを見つけます。
3. スタイルを適用するテキストが含まれているコントロールを選択します。
4. 右側のペインにある **"テキスト スタイル"** 設定を拡張してから、そのサブ設定を使って適切なテキスト スタイルを指定します。詳細については、これらの設定を選択したときに右下のペインに表示されるインライン ヘルプを参照してください。

選択されたコントロールのテキスト スタイルが変更されます。

カスタム ブラシをウィザード インターフェイスに定義する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ブラシを使って、ウィザード インターフェイスの様々な要素、たとえばウィザード ページやコントロール背景に純色、グラデーション、またはイメージを指定します。InstallShield では、カスタム ブラシをプロジェクトに追加してから、これらのブラシを様々なウィザード インターフェイス要素に適用することができます。

タスク

カスタム スタイルを定義するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ブラシ]** を右クリックしてから、**[ブラシの追加]** をクリックします。InstallShield によって **[ブラシ]** の下に新しいブラシが追加されます。必要に応じてブラシの名前を変更します。
3. 新しいブラシを選択してから、右のペインで設定を構成します。

ブラシを追加すると、[ウィザード インターフェイス] ビューで様々な背景関連の設定にそれを選択できます。たとえば、ウィザード ページのナビゲーション領域の背景にデフォルトで使用するブラシを指定するには、"ナビゲーション背景" 設定を使用します。

ウィザード インターフェイスのヘッダー、本文、およびナビゲーション領域に背景ブラシを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

デフォルトで、InstallShield ではウィザード ページのヘッダー、本文、およびナビゲーション領域に異なるブラシを使用できます。場合によっては、これらの 3 つの領域にわたって、単一のブラシ (たとえば、単一の背景イメージ、または垂直方向のグラデーション) のみを使用したいときがあります。次の手順は、3 つの異なるブラシまたは単一のブラシを使ってカスタマイズを行う方法を説明します。

ヒント: *ブラシの定義についての情報は、「カスタム ブラシをウィザード インターフェイスに定義する」を参照してください。*

ヘッダー、本文、およびナビゲーション領域に 3 つの異なるブラシを使用する

タスク

ヘッダー、本文、およびナビゲーション領域に 3 つの異なるブラシを使用するには、以下の手順い従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を選択します。"グローバル ウィザード インターフェイス" 設定が、右のペインに表示されます。
3. **"デフォルトの本文背景"** 設定で、ウィザード インターフェイスの本文領域で背景として使用するブラシを選択します。
4. **"ヘッダー背景"** 設定で、ウィザード インターフェイスのヘッダー領域で背景として使用するブラシを選択します。
5. **"ナビゲーション背景"** 設定で、ウィザード インターフェイスのナビゲーション領域で背景として使用するブラシを選択します。

重要: *前述の背景の設定でブラシを選択するとき、"完全なウィザード背景" 設定のすべての値を削除してください。ウィザード インターフェイスでは完全なウィザード背景に単一のブラシを使用するか、ヘッダー、本文、ナビゲーション領域に個別のブラシを使用することができます。これらの 4 つの領域に同時にブラシを指定することはできません。*

ヘッダー、本文、およびナビゲーション領域にわたって単一のブラシを使用する

タスク

ヘッダー、本文、およびナビゲーション領域に単一のブラシを使用するには、以下の手順い従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を選択します。"グローバル ウィザード インターフェイス" 設定が、右のペインに表示されます。
3. **"完全なウィザード背景"** 設定で、ウィザード インターフェイスのナビゲーション背景、本文およびナビゲーション領域に使用するブラシを選択します。

重要: *前述の設定でブラシ を選択するとき、その他の背景設定 ("デフォルトの本文背景"、"ヘッダー背景"、および "ナビゲーション背景") のすべての値を削除してください。ウィザード インターフェイスでは完全なウィザード背景に単一のブラシを使用するか、ヘッダー、本文、ナビゲーション領域に個別のブラシを使用することができます。これらの 4 つの領域に同時にブラシを指定することはできません。*

ウィザード ページまたは 2 番目のウィンドウのデフォルト 背景ブラシをオーバーライドする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ほとんどの場合、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのユーザー インターフェイス内の各ウィザード ページと 2 番目のウィンドウには、同じ背景が使用されます。一部の場合において、プロジェクト内の特定のウィザード ページまたは 2 番目のウィンドウに設定されているデフォルトの背景をオーバーライドすることもあります。その場合、該当するウィザード ページまたはウィンドウの背景に異なるブラシを選択することができます。



タスク

特定のウィザード ページまたは 2 番目のウィンドウにブラシを適用するには、次の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** ノードまたは **[2 番目のウィンドウ]** ノードを展開して、ブラシを適用するウィザード ページまたはウィンドウをクリックします。
   InstallShield の右側のペインに選択されたウィザード ページまたはウィンドウの設定が表示されます。
3. **"本文背景"** 設定で、選択されたウィザード ページまたは 2 番目のウィンドウに使用するブラシを選択します。

選択されたウィザード ページまたは 2 番目のウィンドウの背景が変更されます。

カスタム コントロール テーマをウィザード インターフェイスに定義する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、ウィザード インターフェイス コントロールのテーマを定義することができます。コントロールのテーマは、コントロールの様々な部分の色（テキストの色、背景の色、および境界線の色）や、クリック済みまたはホバー状態といった、コントロールの様々な状態に使用する色を集めたものです。デフォルトでユーザー インターフェイスのコントロールに使用するコントロール テーマを指定することができます。このテーマは、ユーザー インターフェイス上のすべてのナビゲーション ボタン（たとえば、[戻る] および [次へ] ボタン）のテーマとなります。ユーザー インターフェイス内のウィザード ページやウィンドウに含まれる特定のコントロールのテーマをオーバーライドすることもできます。コントロールのテーマを使用すると、インストールのユーザー インターフェイス内のその他のアイテムが 3D 効果なしで平面的に表示されます。これは Windows Store アプリケーションのスタイルに似ています。

次の種類のコントロールで、コントロールのテーマを使用できます:

- ボタン（ナビゲーション ボタンを含む）
- チェック ボックス
- ラジオ ボタン
- コマンド リンク



タスク ***コントロール テーマを定義するには、次の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[コントロール テーマ]**を右クリックしてから、**[コントロール テーマの追加]** をクリックします。InstallShield によって **[コントロール テーマ]** の下に新しいコントロール テーマが追加されます。必要に応じて、コントロール テーマの名前を変更します。
3. コントロール テーマを選択してから、右のペインで設定を構成します。

コントロール テーマを追加すると、[ウィザード インターフェイス] ビューで様々なテーマ関連の設定にそれを選択できます。そのコントロール テーマに基づくコントロールの外観を変更するには、そのコントロール テーマを選択してから、設定を編集します。

コントロール テーマに加えた変更は、そのコントロール テーマを使用するすべてのウィザード インターフェイス要素に反映されます。

テーマをウィザード インターフェイスのコントロールに適用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、ボタン、チェックボックス、ラジオボタン、およびコマンド リンクにコントロール テーマを適用することができます。コントロール テーマが使用されているウィザード インターフェイス要素の外観を後で変更したい場合は、そのコントロール テーマが使用されている各要素を個別に編集する代わりに、コントロール テーマを簡単に変更することができます。

いくつかの個別のコントロールにテーマを適用して、その他のコントロールをそのままにしておくか、プロジェクト全体にテーマを適用して、いくつかの個別のコントロールのテーマをオーバーライドすることができます。プロジェクト全体のテーマを変更しても、コントロールごとのテーマが指定されている場合、それが変更されることはありません。

コントロール テーマをプロジェクト全体のナビゲーション ボタンに適用できますが、個別のコントロールに適用することはできませんので、ご注意ください。

タスク ***コントロール テーマをウィザード インターフェイス全体に適用するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を選択します。InstallShield の右側のペインに、グローバル ウィザード インターフェイス設定が表示されます。
3. ***"デフォルト コントロール テーマ"*** 設定で、ボタン (ナビゲーション ボタンを含む)、チェックボックス、ラジオボタン、およびコマンド リンクなどのコントロールにデフォルトで使用するコントロールの適切なテーマを選択します。

コントロールごとのテーマが選択されていないすべてのコントロールのテーマが変更されます。

ヒント: *インストールで特定のテーマで定義されている色ではなく、ユーザー定義のシステム色を使用するには、"デフォルト コントロール テーマ" 設定で* ***(テーマなし)*** *オプションを選択します。*

タスク ***デフォルト テーマの選択をオーバーライドしてコントロールにテーマを適用するには、次の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** ノードまたは **[2 番目のウィンドウ]** ノードを展開して、テーマを適用するコントロール (ボタン、チェックボックス、ラジオボタン、コマンド リンク) が含まれているウィザード ページまたはウィンドウを見つけます。
3. テーマを適用するコントロールを選択します。
4. 右側のペインで、**"コントロール テーマ"** 設定を見つけて、選択されたコントロールに使用するテーマを選択します。

選択されたコントロールのテーマが変更されます。

ヒント: *選択されたコントロールのデフォルトのコントロール テーマがオーバーライドされないようにするは、"コントロール テーマ" 設定で [テーマなし] オプションを選択するか、この設定を空白のままに残します。*

最小ウィザード ページのカスタマイズ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield を使って、最小ウィザード ページのテーマをカスタマイズすることができます。カスタマイズされたテーマは、インストール中に表示される 2 番目のダイアログにも適用されます。

- ダーク
- ライト

[最小ウィザード ページ] 領域では、ウィザード ページのキャプション、背景色および [閉じる] および [最小化] ボタンのアイコンをカスタマイズすることもできます。

個別のウィザードページでは、次に記述される追加のカスタマイズも可能です:

- 進行状況バーの色をカスタマイズする
- ライセンス ファイルまたは URL を指定する
- インストール ディレクトリを変更するオプション

テキスト スタイル最小ウィザード インターページのテキストに適用する

[ウィザード インターフェイス] ビューの [スタイル] 領域では、ウィザードの 1 ヶ所以上に適用するフォーマット情報のまとまりを構成して、素早く外観を変更することができます。次は、ダークおよびライト テーマでサポートされている既定のテキストのスタイルです:

- ダーク テーマ – HeaderDark、CaptionDark、および BodyDark
- ライト テーマ – HeaderLight、CaptionLight、および BodyLight

ここに記述されているスタイルは、最小ウィザード ページのテキストに適用される既定のフォント セットおよびテキスト スタイルを指定します。これらのスタイルは、インストール作成者の必要に応じて、さらにカスタマイズを行うことができます。現時点で、最小ウィザード ページでサポートされているカスタム スタイルはありません。

最小ウィザード ページ セットアップのコントロール テーマを選択する

選択されたテーマに基づいて、最小ウィザード ページにはデフォルトで 2 つのコントロール テーマがサポートされています。これらを使って、最小ウィザード ページ セットアップに表示されるボタンの外観を制御することができます。プロパティ

- MinimalPanelLight
- MinimalPanelDark

最小ウィザード ページでサポートされているカスタム コントロール テーマはありません。

最小ウィザード ページのアイコンを選択

[最小ウィザード ページ] 領域では、製品ロゴ、[閉じる] および [最小化] ボタンのアイコンをカスタマイズすることができます。**[サポート ファイル]** 領域には、最小ウィザード ページに表示されるすべてのデフォルト テーマ固有のアイコン ファイルが 100、150 および 200 倍率で含まれています。

制限

スイート プロジェクトにおける最小ウィザード ページのサポートに関する制限事項:

- 最小ウィザード ページのレイアウトのカスタマイズはサポートされていません。ページおよびコントロールの追加/削除を行うことはできません。
- 最小ウィザード ページ上のボタン クリックのアクションを指定することはできません。
- インストールのパスワード保護は、現在サポートされていません。
- セットアップのインストール中、タスクバーに表示される進行状況は、サポートされていません。
- Windows 7 上の InstallShield IDE で最小ウィザード ページをプレビューすることはできません。

ウィザード インターフェイスのフォーマットを選択する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield ではインストールのウィザード インターフェイスに、いくつかの異なるウィザード フォーマットを選択できます:

- **Glass**—インストールはユーザー選択のシステム色を使って、ウィザード インターフェイスのキャプション バー、ヘッダー、およびナビゲーション領域を表示します。また、ウィザード インターフェイスの同じ領域には、Windows Vista で使用できるガラス効果 (透明感) を使うこともできます。このフォーマットを使用すると、ウィザード インターフェイスのキャプション バー、ヘッダー、およびナビゲーション領域に定義されているすべてのブラシは、インストールによって無視されます。

  一部のオペレーティング システムは、このサポートを完全にサポートしません。したがって、インストールが Glass フォーマットの代わりに Wizard 97 フォーマットを使用する場合もあります。

  デフォルトでは、これが設定されています。

- Aero—このフォーマットは、一般的に Glass フォーマットと同じ外観を持ちます。

  一部のオペレーティング システムは、このサポートを完全にサポートしません。したがって、インストールが Aero フォーマットの代わりに Wizard 97 フォーマットを使用する場合もあります。

- Wizard 97—このフォーマットは、従来型のウィザード フォーマットです。インストールはウィザード インターフェイスの様々な領域に定義されているブラシを使います。

- Wizard Lite—このフォーマットは Wizard 97 フォーマットに類似していますが、ヘッダー (タイトルが表示される領域、およびウィザード ページ上部の角に表示されるイメージ) を含みません。インストールはウィザード インターフェイスの別の領域に定義されているブラシを使います。

一部の状況下で、インストールは Glass または Aero フォーマットの代わりに Wizard 97 フォーマットを使用します:

- Windows 8 および Windows Server 2012 システムは、透明感をサポートしません。これらのシステムでは、Glass フォーマットを持つウィザード インターフェイスは Wizard 97 フォーマットを使用します。ただし、Aero フォーマットを持つウィザード インターフェイスは Aero を使用しますが、ガラス効果はありません。

- Windows 7、Windows Vista、Windows Server 2008 R2、および Windows Server 2008 では、エンド ユーザーが有効化/無効化できる透明感サポートが使用できます。これらのシステムでは、透明感が無効化されている場合、Glass フォーマットを使ったウィザード インターフェイスと Aero フォーマットを使ったウィザード インターフェイスでは、Wizard 97 フォーマットが使用されます。また、これらのウィザード インターフェイスでは、ターゲット システム上のデスクトップ コンポジション機能が無効化されている場合、Wizard 97 フォーマットが使用されます。

- Windows XP および Windows Server 2003 上では、Glass フォーマットのウィザード インターフェイスと Aero フォーマットのウィザード インターフェイスは Wizard 97 フォーマットを使用します。

InstallShield の [ウィザード インターフェイス] ビューのウィザード インターフェイス エディターでは、インターフェイスの編集可能なプレビューを表示するときに Glass フォーマットまたは Aero フォーマットを使用しません。アドバンスト UI またはスイート/アドバンスト UI プロジェクトでこれらのフォーマットのどちらかを選択した場合、ウィザード インターフェイス エディターでは、編集可能なプレビューを表示するときに Wizard 97 フォーマットを使用します。

タスク ***ウィザード インターフェイスのウィザード ページに使用するフォーマットを選択するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を選択します。
3. 右側のページに表示される、**"ウィザード フォーマット"** 設定で、適切なオプションを選択します。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで新しい空白のウィザード ページを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*

- *スイート/アドバンスト UI*



***エディション**- アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、プロジェクトに新しいブランク ウィザード ページを追加できます。



タスク ***アドバンスト UI またはスイート/アドバンスト UI プロジェクトで新しい空白のウィザード ページを作成するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を右クリックしてから、**[ブランク ページを追加]** をクリックします。

[ウィザード ページ] の下に新しいウィザード ページが追加されます。必要に応じて、ウィザード ページの名前を変更します。右側のペインで設定を構成して、中央ペインでページを参照およびレイアウトを変更します。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、定義済みのウィザード ページを追加する



***プロジェクト**- この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



***エディション**- アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに様々な定義済みウィザード ページを追加できます。以下に例を示します:

- エンド ユーザーが .msi パッケージのインストール ディレクトリを選択できるページ
- エンド ユーザーがカスタマー情報およびシリアル番号を入力できるページ
- 機能ツリーおよび機能の説明とサイズを表示してエンド ユーザーがインストールする機能を選択できるページ
- データベース ログイン ページを使って、エンド ユーザーはデータベース サーバー ログイン情報 (データベース サーバー名、認証資格情報、データベース カタログ名など) を入力し、スイートに含まれる 1 つ以上の .msi パッケージがターゲットとするデータベース サーバーへの接続を設立することができます。



タスク

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、定義済みのウィザード ページを追加するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を右クリックしてから、**[定義済みページの追加]** をクリックします。**ユーザー インターフェイス ウィザード**が開きます。.
3. ウィザードのパネルを必要に応じて完成します。

新しい定義済みのページががプロジェクトに追加されます。右側のペインで設定を構成して、中央ペインでページを参照およびレイアウトを変更します。



メモ- *プロジェクトに [インストール フォルダーを参照] と同様のウィザード ページ (BrowseFolder と呼ばれる) を追加すると、InstallShield によってプレースホルダー イメージ コントロールが追加され、ページにロック イメージを表示することができます。コントロールの "リソース" 設定を構成してこのコントロールと共に表示するファイルを指定するか、イメージを除外する場合は、イメージ コントロールを削除します。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで新しい 2 番目のウィンドウを作成する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション- *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、プロジェクトに新しいブランクの 2 番目のウィンドウを追加できます。



タスク

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで新しい空白の 2 番目のウィンドウを作成するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[2 番目のウィンドウ]** を右クリックしてから、**[ブランク ウィンドウを追加]** をクリックします。

2 番目のウィンドウ リストの最後に新しいウィンドウが追加されます。必要に応じてウィンドウの名前を変更します。右側のペインで設定を構成して、中央ペインでページを参照およびレイアウトを変更します。

シーケンス ウィザード ページ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[ウィザード インターフェイス] ビューを使って、アドバンスト UI または スイート/アドバンスト UI インストールで実行時にウィザード ページを表示する順番を指定します。ウィザード ページは、このビューの [ウィザード ページ] ノードの下に、実行時にインストールが表示する順番でリストされます。



タスク

プロジェクト内でウィザード ページをシーケンスするには、次の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード ページ]** ノードの下にある **[ウィザード インターフェイス]** エクスプローラーで、順番を変更するウィザード ページを右クリックしてから、**[上へ移動]** または **[下へ移動]** をクリックします。

リスト内で、ウィザード ページが上または下に移動します。

ウィザード ページまたは 2 番目のウィンドウのレイアウトおよび動作を編集する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[ウィザード インターフェイス] ビューを使って、インストールに含まれるウィザード ページおよび 2 番目のウィンドウのレイアウトならびに動作を編集します。

ウィザード ページまたは 2 番目のウィンドウにコントロールを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[ウィザード インターフェイス] ビューでウィザード ページまたは二次ウィンドウを選択すると、ウィザード インターフェイスのプレビュー ペインのすぐ上に表示されるツールバーに、新しいコントロールを追加できるボタンやコントロールが追加表示されます。

タスク

ウィザード インターフェイスのコントロールを追加するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、変更するウィザード ページまたは 2 番目のウィンドウをクリックします。中央ペインのウィザード エディターに、ページまたは 2 番目のウィンドウが表示されます。
3. ウィザードのインターフェイスのプレビュー ペインの上にあるツールバーで、適切な [新しいコントロール] ボタン ([ラベル]、[テキスト ボックス]、[ボタン]、または [**コンボボックス**] ボタン) をクリックして、コントロール ボタンの横にあるドロップダウン リストで表示されているコントロールの種類から 1 つ選択します。利用可能なそれぞれのコントロールの種類についての詳細は、「[ウィザード インターフェイス] ビューのツールバー」を参照してください。

   ページまたは 2 番目のウィンドウの本文領域の左上付近にコントロールが追加されます。また、選択されたウィザード ページまたはウィンドウの下にあるコントロールにサブノードが追加されます。
4. コントロールの中央にカーソルを配置してから、それを適切な場所にドラッグして移動させます。
5. 右側のペインで、コントロールの設定を構成します。

ヒント: *ウィザード ページのノードの下にリストされているコントロールの順番は、タブの順番に一致します。*

1 つのページで 1 つ以上のラジオボタン コントロール グループを使用している場合、各グループの最初のラジオボタンの "スタイル" 設定に WS_GROUP が含まれていることを確認してください。

ウィザード ページまたは 2 番目のウィンドウに含まれるコントロールのタブの順番を変更する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

多くの場合、エンド ユーザーは TAB キーを使って、ウィザード ページまたはウィンドウ上にあるコントロール間のフォーカスを移動させることができます。デフォルトで、タブの順番は、ウィザード ページまたはウィンドウに追加されたコントロールの順番となります。



タスク

ウィザード ページまたはウィンドウに含まれるコントロールのタブの順番を変更するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、タブの順番を変更するウィザード ページまたは 2 番目のウィンドウのノードを拡張します。

   ページまたはウィンドウ上にある各コントロールには、それぞれサブノードがあります。コントロールはタブの順番に従ってリストされています。つまり、ページ ノードの下にある最初のコントロールが、そのページでフォーカスされる最初のコントロールです。
3. タブの順番にシーケンスするコントロールを右クリックして、**[上に移動]** または **[下に移動]** をクリックします。
4. ページの下にリストされているコントロールが適切なタブの順番に並ぶまで、コントロールを移動させます。

一部の状況において、コントロールがタブの順番に含まれていない場合があります。たとえば、無効なコントロールについては、エンド ユーザーがタブをクリックしてもコントロール間でフォーカスが移動しません。

ウィザード インターフェイスにおけるキーボード ショートカットの指定とアンパサンド (&) の使用



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ウィザード インターフェイス文字列で、アンパサンド (&) はキーボード ショートカットを指定する特殊な文字の場合がありますが、リテラルなアンパサンドの表示を意味することもあります。ボタン、チェックボックス、およびラジオ ボタン コントロールでは、単一のアンパサンドは常に、コントロールをクリックまたは選択するのに使用されるキーボード ショートカットを指定します。また、2 重のアンパサンド (&&) は、単一のアンパサンドが表示されることを意味します。ラベル コントロールの場合、アンパサンドの解釈は、それが使用されているラベル コントロールのスタイルによって異なります。

テーブル 33 ・ ウィザード インターフェイスのラベル コントロールでアンパサンドを使用する

ラベル コントロールのサンプル文字列エントリ	コントロールの "スタイル" 設定における SS_NOPREFIX の状態	ウィザード インターフェイスで表示される文字列	注
新しい && 強化された製品のシリアル番号(&S) を入力してください	SS_NOPREFIX = False	新しい & 強化された製品のシリアル番号(S) を入力してください	エンド ユーザーが ALT キーを押すと、文字列の「S」に下線が表示されます。キーボード ショートカット ALT + S は、タブの順序でこのラベルの直後に表示されるコントロールにフォーカスを移動させます。 「新しい」と「強化された」の間にある 2 重のアンパサンドは、単一のアンパサンドが表示されることを意味します。
新しい & 強化された製品のシリアル番号を入力してください	SS_NOPREFIX = True	新しい & 強化された製品のシリアル番号を入力してください	SS_NOPREFIX 定数の True 値は、コントロールの文字列内のアンパサンドがキーボード ショートカットとして使用されないように防ぎます。

デフォルトでは、ウィザード インターフェイスに新しいラベル コントロールを追加すると、ラベル コントロールの "スタイル" 設定の下にある SS_NOPREFIX サブ設定が False に設定されています。必要に応じて、この値を変更できます。

ウィザード インターフェイスにキーボード ショートカットを持たないラベル コントロールを含む場合、そのコントロールの SS_NOPREFIX サブ設定を True に選択する方法を考慮してください。これによって、コントロールの文字列エントリに含まれる任意のアンパサンドが、誤ってキーボード ショートカットとして解釈されることが無い様にできます。

製品にアンパサンドが含まれている場合、ウィザード インターフェイスに含まれる 2 つの製品名プロパティを使用することができます (1 つめは、単一のアンパサンドを使用するプロパティ、もう 1 つはエスケープされたアンパサンド、つまり二重アンパサンドを使用する値を持つプロパティ)。製品名を持つ各コントロールの種類とスタイルに基づいて、必要に応じて適切な製品名プロパティを使用します。例:

テーブル 34 ・ アンパサンドを含む文字列の 2 つの異なるプロパティの使用方法サンプル

プロパティ	プロパティ値	このプロパティの使用例
[ProductName]	新しい & 強化された製品	• SS_NOPREFIX = True であるラベル コントロール • [ウィザード ページ] ノードの "ウィザード キャプション" 設定 – この値は、ウィザード ページのキャプション バー内のキャプションとして使用されます。この設定のデフォルト値: [ProductName] – InstallShield Wizard • ウィザード ページの "タイトル" 設定 – タイトル テキストは、ウィザード ページのヘッダー領域に表示されます。
[EscapedProductName]	新しい && 強化された製品	• SS_NOPREFIX = False であるラベル コントロール • SS_NOPREFIX スタイル設定をサポートせず、単一のアンパサンドを常にキーボード ショートカットの指定と解釈するチェックボックス、コマンド リンク、およびラジオボタンなどのコントロール

ウィザード ページでナビゲーション ボタンを使う



***プロジェクト・** この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



***エディション・** アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトのウィザード ページには、以下のビルトイン ナビゲーション ボタンを 1 つ以上含めることができます:

テーブル 35 ・ ウィザード ページのビルトイン ナビゲーション ボタン

ボタン	デフォルト動作の説明
Next	[表示] 条件が False 評価されるページをすべてスキップして、インターフェイスの次のウィザード ページに移動します。 現在のウィザード ページがインターフェイスで最後のページの場合、このボタンをクリックしても何も動作しません。 [次へ] ナビゲーション ボタンは、アクティブ ページの設定アクションを使うコントロールの動作とよく似ていて、特定のページへ移動する動作をトリガします。
戻る	[表示] 条件が False 評価されるページをすべてスキップして、インターフェイスの前のウィザード ページに移動します。 現在のウィザード ページがインターフェイスで最初のページの場合、このボタンをクリックしても何も動作しません。 [戻る] ナビゲーション ボタンは、[アクティブ ページの設定] アクションを使うコントロールの動作とよく似ていて、特定のページへ移動する動作をトリガします。
Cancel	質問 IDS_SUITE_CONFIRMCANCEL を含むメッセージ ボックスを表示します。[はい] の場合、ウィザードをキャンセルして最後のウィザード ページに移動します。 ほとんどの場合、この時点でインストールをキャンセルしても何ら効果がないため、最後のウィザード ページには [キャンセル] ボタンを表示しません。[キャンセル] ボタンが表示されている場合、それをクリックするとウィザードが閉じます。
Install	インストールを開始して次のウィザード ページに移動します。 [インストール] ナビゲーション ボタンは、[インストール] アクションを使うコントロールと似たような動作をし、インストールを開始してから特定のウィザード ページに移動します。
Finish	インストールまたはメンテナンスが正しく完了した後に再起動が必要な場合は、再起動をするかどうか尋ねてから (IDS_SUITE_ASKREBOOT)、適切な応答をトリガします。どちらの場合も (すぐにまたは遅延の再起動が必要、または不要な場合)ウィザードが閉じます。

ウィザード ページにナビゲーション ボタンを含める、またはナビゲーション ボタンを除外する

アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [ウィザード インターフェイス] ビューでウィザード ページを選択すると、ビューの右側のグリッドに選択されたページの設定が表示されます。"ナビゲーション領域" の設定を使って、そのページのナビゲーション ボタンを構成できます。



タスク ***ナビゲーション ボタンを含める、または除外するには、以下の手順を実行します:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、変更するボタンを含むウィザード ページをクリックします。
3. ビューの右側にある **[ナビゲーション]** 領域を使って、以下の処理を行います:
   - ウィザード ページに特定のボタンを含めるには、そのボタンの設定で **[はい]** を選択してから、そのボタンの設定の下にあるサブ設定を必要に応じて構成します。
   - ウィザード ページから特定のボタンを除外するには、そのボタンの設定で **[いいえ]** を選択してから、そのボタンの設定の下にあるサブ設定を必要に応じて構成します。

ナビゲーション ボタンのデフォルトの動作を変更する処理については「ウィザード ページのナビゲーション ボタンのデフォルト動作をオーバーライドする」を参照してください。

ウィザード ページのナビゲーション ボタンのデフォルト動作をオーバーライドする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ナビゲーション ボタンの "クリック" イベント は、"アクティブ ページの設定" アクションまたは [インストール] アクションを含み、そのアクションをウィザード インターフェイス内で移動させると、デフォルトの動作が阻止されます。デフォルトの動作は、アクションの条件が False のために、スケジュールされたアクションがスキップされた場合でも阻止されます。

たとえば、[インストール] ボタンのデフォルトの動作をオーバーライドしたいとき、インターフェイス内で次のウィザード ページに移動する代わりに、インストールを特定のウィザード ページに進めたい場合があります。このシナリオでは、[インストール] ボタンの "クリック" 設定に [インストール] アクションを追加して、次に続く特定のページを指定しなくてはなりません。その後、アクションに条件を追加すると、条件が [インストール] アクションの実行を阻止して、[インストール] ボタンのデフォルトの動作が抑制されます。つまり、エンド ユーザーが [インストール] ボタンをクリックしても、インストールは開始しません。(この場合、適切な場合は [ナビゲーション] ボタンの "有効" 設定に条件を定義して、[インストール] ボタンを確実に無効にしてください。効果のない [インストール] ボタンをクリックできる状態にしておくと、エンド ユーザーが混乱する可能性があります。)

ISuiteUIExtension インターフェイス上の対応するメソッドを実際に呼び出さない限り、拡張 DLL アクションが、デフォルト アクションを阻止することはできません。アドバンスト UI またはスイート/アドバンスト UI エンジンが、拡張アクションがメソッドの呼び出しを選択したかどうかを判別することはできません。拡張アクション

を使ってデフォルト動作をオーバーライドする必要がある場合で、特定の条件下でのみナビゲートまたはインストールするときは、False 条件 (空白の None グループなど) を持つ [アクティブ ページの設定] アクションまたは [インストール] アクションを追加します。

ウィザード インターフェイスのコンボ ボックスおよびリスト ボックス コントロールを設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、ウィザード インターフェイスに定義済みの選択可能なオプションを含む 1 つ以上のコンボ ボックス コントロールおよびリスト ボックス コントロールを追加できます。その手順には、コントロールの設定を構成して選択可能なオプションのコンテンツを定義する処理が含まれます。また、2 つのプロパティ (コントロールで表示するオプションのリストを定義するプロパティおよび実行時にエンド ユーザーが選択するオプションを格納するためにインストールが使用するプロパティ) も使用します。



タスク

定義済みオプションを含むコンボ ボックス コントロールまたはリスト ボックス コントロールを設定するには、以下の手順に従います:

1. 2 つのプロパティ (コントロールで表示するオプションのリストを定義するプロパティおよび実行時にエンド ユーザーが選択するオプションを格納するためにインストールが使用するプロパティ) をコントロールに関連付けます。
   a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
   b. **[ウィザード インターフェイス]** エクスプローラーで、変更を行うウィザード ページまたは 2 番目のウィンドウを拡張してから、構成したいコンボ ボックス コントロールまたはリスト ボックス コントロールを選択します。
   c. **"プロパティ"** 設定で、コントロールと関連付けるアドバンスト UI またはスイート/アドバンスト UI のプロパティの名前 (たとえば、SELECTIONPROPERTY) を入力します。実行時に、インストールはこのプロパティを使って、エンド ユーザーが選択する値を格納します。
   d. **"コンテンツ プロパティ"** 設定で、このコントロールでリストするオプションを識別するのに使用するアドバンスト UI またはスイート/アドバンスト UI プロパティの名前を入力します。(例、CONTROLOPTIONS)
2. コンボ ボックス コントロールまたはリスト ボックス コントロールに表示するオプション リストを定義します:

a. **"コンテンツ プロパティ"** 設定で、省略記号ボタン (...) をクリックします。**[オプション リストの編集]** ダイアログ ボックスが開きます。

b. **[テキスト]** 列に、コントロールで表示するオプションの 1 つを入力します。

この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

c. **[値]** 列に、手順 2b で入力したリスト オプション テキストと関連付ける値を入力します。エンド ユーザーが **[テキスト]** 列に入力されているリスト オプションを選択したときに、このプロパティ値 (手順 1c で入力) が、SELECTIONPROPERTY に格納されます。

d. 追加でテキスト値のペアを追加する場合、**[新規]** ボタンをクリックすると InstallShield によって追加される行のフィールドに必要な値を入力します。

たとえば、次のようなデータを入力します:

テキスト	値
ID_Option1	Value1
ID_Option2	Value2
ID_Option3	Value3

e. 適切な場合、リスト オプションの順番を並べ替えます。

インストールでは、このダイアログ ボックスでリストされている順番で、リストオプションが表示されます。リスト オプションの順番を並べ替えるには、移動させたい行を選択してから、必要に応じて [上] または [下] ボタンをクリックします。連続する複数の行を選択するには、最初の行をクリックしてから SHIFT を押しながら最後の行をクリックします。連続しない複数の行を選択するには、選択する行の 1 つをクリックしてから CTRL を押しながら追加する各行をクリックします。

3. **[OK]** をクリックします。

[コンボ ボックス コントロール] または [リスト ボックス] コントロールのデフォルト オプションを指定する

コンボ ボックス コントロールまたはリスト ボックス コントロールのデフォルト オプションを指定するには、[プロパティ マネージャー] ビューを使ってアドバンスト UI またはスイート/アドバンスト UI プロパティの値をデフォルトの選択オプションに対応するプロパティ値と等しく設定します。つまり、デフォルトで *ID_Option3* を選択するには、[プロパティ マネージャー] ビューに SELECTIONPROPERTY プロパティを追加してから、その値を *Value3* に設定します。コントロールのデフォルトを空白にするには、[プロパティ マネージャー] ビューでプロパティを定義付けないままとします。

コンボ ボックス コントロールまたはリスト ボックス コントロールからのエンド ユーザー データを使った作業

実行時に、エンド ユーザーはコンボ ボックスまたはリスト ボックスから単一の選択を行うことができます。これによって、コントロールのプロパティ（前述の手順では SELECTIONPROPERTY）が選択されたオプションに関連付けられた値、たとえば *Value1*、*Value2*、または *Value3* に設定されます。SELECTIONPROPERTY プロパティの値に基づいて、様々な種類の動作をトリガすることができます。例:

- ウィザード ページまたは 2 番目のウィンドウでコントロールを有効または無効できます。そのためには、コントロールの"有効" 設定を使って条件のプロパティの種類を定義し、コントロールの有効状態をプロパティの特定の値と関連付けます。
- ウィザード ページまたは 2 番目のウィンドウで特定のコントロールを表示または非表示できます。そのためには、コントロールの"表示" 設定を使って条件のプロパティの種類を定義し、コントロールの有効状態をプロパティの特定の値と関連付けます。
- 特定のウィザード ページまたは 2 番目のウィンドウを表示またはスキップすることができます。そのためには、ウィザード ページの 2 番目のウィンドウの "表示" 設定を使って条件のプロパティの種類を定義し、ウィザード インターフェイスの有効状態をプロパティの特定の値と関連付けます。
- アドバンスト UI またはスイート/アドバンスト UI プロパティ セットの値を、アドバンスト UI またはスイート/アドバンスト UI インストール内のパッケージで使用されている特定のプロパティに使用することができます。そのためには、[パッケージ] ビューのグリッドにある [操作] 領域で使用できるサブ設定で、コマンドラインおよびサイレント コマンドライン設定を構成します。詳細については、「アドバンスト UI およびスイート/アドバンスト UI 形式の式を使用して、コマンドラインを動的に構成する」を参照してください。

ウィザード ページまたはウィンドウに含まれるコントロールの検証を構成する



***プロジェクト**- この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



***エディション**- アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、一部の様々なウィザードのインターフェイス コントロールに対して、検証を実装することができます。例:

- エンドユーザーが、テキスト ボックスでシリアル番号を指定されたフォーマットで入力したことを確認できます。
- テキスト ボックスの検証を構成して、エンドユーザーがテキスト ボックスで指定したフォルダーが正しいかどうかを確認できます。
- エンドユーザーが特定の矛盾する組み合わせで、チェックボックスを選択するのを防ぐことができます。

コントロールの検証によって、そのコントロールに関連付けられているプロパティがチェックされます。たとえば、エンドユーザーがチェックボックスをクリックしたとき、それに関連付けられているプロパティは、潜在的に、2 つの新しい値の 1 つに変更されます。検証では、この新しい値が適切であるかどうかを確認する方法が提供されます。

様々な種類のコントロールによって、検証が異なるイベントでトリガされます:

テーブル 36・異なる種類の検証のトリガ

コントロールの種類	検証をトリガするイベント
テキスト ボックス **パスワード ボックス**	コントロールのエントリに変更が発生した時、検証がトリガされます。
ボタン、 **コマンド リンク、** **チェック ボックス、** **ラジオ ボタン、** **イメージ ボタン**	エンドユーザーがコントロールをクリックした時、検証がトリガされます。
コンボ ボックス、 **リストボックス、** **チェック済みリストボックス、** **機能選択ツリー**	コントロールの選択に変更が発生した時、検証がトリガされます。

Text Style 設定のサブ設定を使って、コントロールに異なるテキスト スタイルを指定することにより、エンドユーザーに、異なるコントロールのステータス (デフォルト、有効、または無効) を視覚的に示すことができます。

また、[イベント] 領域のサブ設定を使って、エンドユーザーがコントロールをクリックまたは使用したときに、アクションがトリガされるようにもできます。コントロールに構成されたアクションと検証がある場合、検証はアクションの前に実行されます。アクションに関する詳細は、「ウィザード インターフェイス内の要素のアクションを構成する」を参照してください。



タスク　***コントロールの検証を構成するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、変更するウィザード ページまたは 2 番目のウィンドウを選択します。
3. ウィザード エディターで、検証に必要なコントロールを選択します。
4. **"検証"** 設定で、**[検証]** ボタンをクリックしてから発生させる検証の種類をポイントします。

   あるタイプの検証では、作成した DLL ファイル内の関数を呼び出すことができます。プロジェクトの [サポート ファイル] ビューに DLL ファイルが存在する場合、それらのファイルがこのリストで表示されますので、検証関数の名前を、呼び出す DLL 内の関数で上書きすることができます。

次のテーブルは、使用可能な検証の種類について説明します。同じ種類の検証は、特定の種類のコントロールに適用されます。たとえば、値の長さの検証は、テキスト ボックスおよびパスワード ボックス コントロールにのみ適用されます。

テーブル 37 • ウィザード インターフェイスで使用されるコントロールのための検証の種類

検証の種類	説明
既存ファイル	このタイプの検証は、このコントロールに関連付けられているプロパティの値に、存在するファイルの完全修飾パスと名前が含まれているかどうかを確認します。通常、プロパティの値は、エンドユーザーがこのコントロールでパスとファイル名を入力したときに設定されます。
既存フォルダー	このタイプの検証は、このコントロールに関連付けられているプロパティの値に、存在するフォルダーの完全修飾パスと名前が含まれているかどうかを確認します。通常、プロパティの値は、エンドユーザーがこのコントロールでパスとフォルダー名を入力したときに設定されます。
既存フォルダーの新しいファイル	このタイプの検証は、このコントロールに関連付けられているプロパティの値に、存在しないファイルの完全修飾パスと名前が含まれているかどうかを確認します。通常、プロパティの値は、エンドユーザーがこのコントロールでパスとファイル名を入力したときに設定されます。 この種類の検証には、ファイルのパスが存在している必要ありますので注意してください。
値の長さ	このタイプの検証は、このコントロールに関連付けられているプロパティの値に、特定の数の文字が含まれているかどうかを確認します。通常、プロパティの値は、エンドユーザーがこのコントロールでテキストを入力したときに設定されます。 この種類の検証を構成するには、この設定の下のサブ設定値に最小値と最大値を指定します。
値のマスク	このタイプの検証は、このコントロールに関連付けられているプロパティの値が、特定のフォーマットに一致しているかどうかを確認します。通常、プロパティの値は、エンドユーザーがこのコントロールで情報（シリアル番号など）を入力したときに設定されます。 この種類の検証を構成するには、この設定の下の "パターン" 設定に必要なフォーマットを指定します。数字の代わりに、次のいずれかの文字を入力します: **# % @** 文字の代わりに、次のいずれかの文字を入力します: **& ^ ? '** その他のすべての入力された文字は正確に一致している必要があります。

テーブル 37 • ウィザード インターフェイスで使用されるコントロールのための検証の種類 (続き.)

検証の種類	説明
DLL アクションを参照	このタイプの検証は、[サポート ファイル] ビューで作成し、プロジェクトに追加した DLL ファイル内の関数を呼び出します。たとえば、DLL ファイルを使用して、プロパティの値の有効性 (エンドユーザーがコントロールで入力した内容) をチェックできます。

ウィザード インターフェイス内の要素のアクションを構成する



プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、ウィザード ページまたは 2 番目のウィンドウが開いているまたは閉じているときに起動されるアクションを定義することができます。この機能によって、コンボボックスの値を動的に取得するなど、インストールで任意の初期設定処理が可能となります。また、実行時にウィザード インターフェイスが表示される前に実行されるアクションをスケジュールすることも可能です。

また、InstallShield では、エンド ユーザーが様々なウィザード インターフェイス コントロールを使用したときにトリガされるアクションを定義できます。たとえば、LicenseAgreement ページに [印刷] ボタンを追加して、そのボタン コントロールの印刷アクションを定義できます。つまり、エンド ユーザーが [印刷] ボタンをクリックすると、[印刷] ダイアログ ボックスが開いて、エンド ユーザーが使用許諾契約を印刷できるようにします。

コントロールに構成されたアクションと検証がある場合、検証はアクションの前に実行されます。アクションは、検証が成功したときも、失敗したときも実行されます。これらのアクションはまた、スプラッシュ ページが抑制されたとき、イメージ ボタンがフォーカスされたとき、または解除されたとき、および、エンドユーザーが、リッチ テキスト ボックスの下端までスクロールしたときも実行されます。検証に関する詳細は、「ウィザード ページまたはウィンドウに含まれるコントロールの検証を構成する」を参照してください。



タスク

ウィザード ページ、2 番目のウィンドウ、またはコントロールのアクションを構成するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、変更するウィザード ページまたは 2 番目のウィンドウを選択します。
3. 次のいずれかの手順に従って、コントロールのアクションを構成します:

- ナビゲーション コントロールの 1 つ ([次へ]、[戻る]、[キャンセル]、[インストール]、または [完了] ボタン) にアクションを構成するには、右側のペインでアクションを変更したいコントロールの設定を拡張します。
- その他のコントロールのアクションを構成するには、ウィザード エディターで、アクションを変更したいコントロールを選択します。

4. 以下のいずれかを実行します。
   - 選択されたウィザード ページまたはウィンドウが開いたときに実行するアクションをスケジュールする場合: **"開始ページ"** 設定で **[新しいアクション]** ボタンをポイントしてから構成するアクションの種類を選択します。
   - 選択されたウィザード ページまたはウィンドウが閉じたときに実行するアクションをスケジュールする場合: **"終了ページ"** 設定で **[新しいアクション]** ボタンをポイントしてから構成するアクションの種類を選択します。
   - コントロールのアクションをスケジュールする場合: **"クリック"** 設定、または **"イベント"** 設定の下にあるアクション関連の設定で **[新しいアクション]** ボタンをポイントしてから、構成するアクションの種類をクリックします。
5. アクションの下のサブ設定を必要に応じて構成します。

複数のアクション ステートメントを入力した場合、アクションは、"アクション" 設定で表示されている順番で実行されます。

あるタイプのアクションでは、作成した DLL ファイル内の関数を呼び出すことができます。プロジェクトの [サポート ファイル] ビューに DLL ファイルが存在する場合、それらのファイルがこのリストで表示されますので、アクション関数の名前を、呼び出す DLL 内の関数で上書きすることができます。

次のテーブルは、使用可能なアクションの種類について説明します。

テーブル 38 ・ コントロールのアクション

アクションの種類	説明
プロパティの設定	この種のアクションは、特定のプロパティを特定の値に設定します。
インストール	この種のアクションは、インストールを開始して特定のウィザード ページに移動します。
開く	この種類アクションは、ファイルまたは Web ページを開きます。
印刷	この種のアクションは、[印刷] ダイアログ ボックスを起動して、エンド ユーザーが指定するファイルを印刷できるようにします。

テーブル 38・コントロールのアクション（続き.）

アクションの種類	説明
フォルダーの参照	この種のアクションは、エンド ユーザーがコントロールをクリックしたときに [フォルダーを参照] ダイアログ ボックスを起動します。 エンド ユーザーがこのダイアログ ボックスでフォルダーを選択してから [OK] ボタンを クシックすると、ダイアログ ボックスが閉じて、インストールは特定のプロパティの値をエンド ユーザーが選択したフォルダーの完全パスに設定します。 この種のアクションを構成するには、このアクションの下にあるサブ設定を構成します。
ファイルの参照	この種のアクションは、エンド ユーザーがコントロールをクリックしたときに [ファイルの参照] ダイアログ ボックスを起動します。 エンド ユーザーがこのダイアログ ボックスでファイルを選択してから [開く] ボタンを クリックすると、ダイアログ ボックスが閉じて、インストールは特定のプロパティの値をエンド ユーザーが選択したファイルの完全パスとファイル名に設定します。 この種のアクションを構成するには、このアクションの下にあるサブ設定を構成します。
新しいファイルの参照	この種のアクションは、エンド ユーザーがコントロールをクリックしたときに [新しいファイルの参照] ダイアログ ボックスを起動します。ダイアログ ボックスを使って、エンド ユーザーは新しいファイルを保存する場所を参照できます。 エンド ユーザーがこのダイアログ ボックスでファイルを指定してから [保存] ボタンを クリックすると、ダイアログ ボックスが閉じて、インストールは ″プロパティ″ サブ設定で指定されたプロパティの値をエンド ユーザーが指定するファイルの完全パスとファイル名に設定します。 この種のアクションを構成するには、このアクションの下にあるサブ設定を構成します。
アクティブ ページの設定	この種のアクションは、特定のウィザード ページに移動します。
ウィンドウを表示	この種のアクションは、2 番目のウィンドウをモーダル ウィンドウとして表示します。

テーブル 38 ・ コントロールのアクション (続き.)

アクションの種類	説明
ウィンドウを閉じる	この種類のアクションは、メインのウィザード ページまたは 2 番目のウィンドウを閉じるか、場合によっては 2 番目のウィンドウを条件付きで閉じます。 [ウィンドウを閉じる] アクションには、次の定義済みリターン コード ID に対応するパラメータを使用できます: IDOK、IDCANCEL、IDABORT、IDRETRY、IDIGNORE、IDYES、IDNO、および IDCLOSE [ウィンドウを閉じる] アクションの動作は、ウィザード ページおよび 2 番目のウィンドウで次のように多少異なります: • ウィザード ページの場合、[ウィンドウを閉じる] アクションはそのリターン コード パラメータが IDCANCEL に設定されている場合、エンド ユーザーがウィザードをキャンセルできるプロンプトを表示 (また、エンド ユーザーが [はい] を指定した場合にウィザードを中断します)。その他のリターン コード ID の場合、ウィザードがすぐに閉じます。 • 2 番目のウィンドウの場合、[ウィンドウを閉じる] アクションは 2 番目のウィンドウを閉じます。また特殊な場合、たとえば 2 番目のウィンドウが ISRMFilesInUse および ISRMFileInUse の場合は指定されたリターン コード値が戻されます。 InstallShield では現在、指定されたリターン コード ID によって異なるカスタム動作を含む、次の 2 番目のウィンドウが提供されています: • ISDownloadProgress • ISPromptForSourceMedia • ISFilesInUse • ISRMFilesInUse • ISUpgradeParcel • ISSQLBrowse
イベントの停止	この種類のアクションは、後に続くアクションの処理を条件付きで停止します。たとえば、このアクションを使ってボタンのデフォルト動作を抑制することができます。
スイート アクションの実行	 ***プロジェクト:*** *この機能は、スイート/アドバンスト UI プロジェクトで使用できます。* この種類のアクションは、インストールでアクションを起動します。スイート/アドバンスト UI プロジェクトの [イベント] ビューで定義されている任意のアクションを選択できます。詳細については、「スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する」を参照してください。

テーブル 38 ・ コントロールのアクション (続き.)

アクションの種類	説明
Web 配置パブリッシュ プロファイルのロード	 ***プロジェクト・** この機能は、スイート/アドバンスト UI プロジェクトで使用できます。* この種類のアクションは、エンド ユーザーが Web 配置パッケージに使用する構成設定を含むパブリッシャー プロファイル ファイル (.publishsettings) を参照およびロードすることができる 2 番目のウィンドウを起動します。詳細については、「Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する」を参照してください。

テーブル 38 ・ コントロールのアクション（続き.）

アクションの種類	説明
データベース	 ***プロジェクト・** この機能は、スイート/アドバンスト UI プロジェクトで使用できます。* 次のデータベース関連のオプションが使用できます: ・ **データベース メタデータの構成** ― データベース メディア（接続文字列、ODBC ドライバー、タイムアウト値等）を初期化し、アクティブな構成として別のアクションが参照できるよう、拡張 DLL に構成の名前を保存します。 ・ **SQL ログインのプロパティをオーバーライド** ― 指定されたデータベース サーバーへの接続を確立するとき、SQL ログインの設定を識別するために使用されるスイートのプロパティを定義します。たとえば、**[データベース ログインの検証]** アクションが呼び出されると、スイート SQL ランタイムは **[SQL ログインのプロパティをオーバーライド]** アクションによって指定されたスイートのプロパティの値を読み込んで、[スイート UI] ウィザード ページに入力された SQL ログインの設定を取得してから、指定されたデータベース サーバーへの接続を設立しようとします。 ・ **データベース カタログの参照**―使用可能なデータベース カタログ リストを取得して、その結果をスイート プロパティ（**ISSQLBrowseList**）に保存します。このアクションは、**[データベースメタデータの構成]** および **[SQL ログインのプロパティをオーバーライド]** アクションによって識別されたメタデータおよびプロパティを使用します。 ・ **データベース サーバーの参照**―使用可能なデータベース サーバー リストを取得して、その結果をスイート プロパティ（**ISSQLBrowseList**）に保存します。このアクションは、**[データベースメタデータの構成]** および **[SQL ログインのプロパティをオーバーライド]** アクションによって識別されたメタデータおよびプロパティを使用します。 ・ **登録済みデータベース サーバーの参照**―ODBC DSN に登録されているデータベース サーバー リストを取得して、そのリストをスイート プロパティ（**ISSQLBrowseLocalList**）に保存します。このアクションは、**[データベースメタデータの構成]** および **[SQL ログインのプロパティをオーバーライド]** アクションによって識別されたメタデータおよびプロパティを使用します。 ・ **データベース ログインの検証** ― **[データベース メタデータの構成]** および **[SQL ログインのプロパティをオーバーライド]** によって識別されたデータベース メタデータおよびプロパティを使って、データベース ログイン資格情報を検証する [検証] アクションを追加します。 ・ **SQL 文字列の実行** ― **[データベース メタデータの構成]** および **[SQL ログインのプロパティをオーバーライド]** によって識別されたデータベース メタデータおよびプロパティを使って、SQL ステートメントを実行します。

テーブル 38 ・ コントロールのアクション (続き.)

アクションの種類	説明
DLL アクションを参照	このタイプのアクションは、[サポート ファイル] ビューで作成し、プロジェクトに追加した DLL ファイル内の関数を呼び出します。たとえば、DLL ファイルを使って、カスタム動作をトリガすることができます。

ウィザード インターフェイスにおけるイメージ、テキスト ファイル、その他のオブジェクトの使用



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



***エディション・** アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトのウィザード インターフェイスで使用するイメージ、テキスト ファイル、その他のファイルはすべて、プロジェクトにサポートファイルとして含める必要があります。サポート ファイルとは、インストール処理中のみの使用目的でターゲット システムにインストールされるファイルです。これらのファイルは、インストールの完了時に自動的にターゲット システムから削除されます。

プロジェクトが複数言語をサポートする場合、言語固有のサポート ファイルを [サポート ファイル] ビューの適切な言語固有の領域にに追加できます。実行時、アドバンスト UI またはスイート/アドバンスト UI インストールは、適切な言語固有のサポートファイルを表示します。

詳細については、「サポート ファイルを使用する」を参照してください。

ウィザード インターフェイスにおける DPI 対応



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



***エディション・** アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける DPI 対応には、2 つの基本事項があります:

- エンジンが使用するアイテムは、システムの DPI 設定にしたがって調整されます。つまり、ターゲット システムで DPI 200% が設定されている場合、チェックボックス コントロールも同様に調整されます。
- エンジンは、ウィザード インターフェイスに含まれたイメージその他のリソースを表示するとき、スケール因子と言語を考慮します。

たとえば、エンジンがスケール因子を 150 と判断したとき、適切な言語が英語である場合は、ファイル名またはパス名に *scale-150* の文字列を含むリソースを探します。ターゲット システム上で image.png を検索するとき、次にリストされている順番でパスを検索します:

1. [SUPPORTDIR]0409¥**scale-150**¥image.png
2. [SUPPORTDIR]0409¥image.**scale-150**.png
3. [SUPPORTDIR]0409¥image.png
4. [SUPPORTDIR]**scale-150**¥image.png
5. [SUPPORTDIR]image.**scale-150**.png
6. [SUPPORTDIR]¥image.png

正しい DPI よりも一致する言語が優先します。

InstallShield はアドバンスト UI およびスイート/アドバンスト UI プロジェクトに含まれるビルトイン デフォルト イメージに、scale-150 および scale-200 イメージを含みます。プロジェクトにカスタム リソースを含める場合、[サポート ファイル] ビューを使って適切なスケールのイメージを追加します。

エンドユーザー インターフェイスのローカライズ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallShield には、エンド ユーザーに表示するテキスト文字列を指定するための多くの設定があります。たとえば、機能の "表示名" と "説明" 設定に値を入力すると、実行時に CustomSetup ダイアログで表示される機能名と説明を構成することになります。1 つまたは複数の言語にローカライズが必要な設定に値を入力するたびに、プロジェクトがサポートする各言語用に InstallShield が自動的に文字列エントリを作成します。各文字列エントリは、言語非依存の識別子と対応する言語固有の値で構成されます。実行時に、インストールは適切な翻訳済み文字列値を表示します。ハードコード化されたテキストの代わりに文字列エントリを使用することで、インストール中に表示される全ての言語固有文字列から、インストールのコードを完全に切り離すことができます。

プロジェクトのローカライズ処理を簡素化するために、インストール処理中、実行時に表示されるすべてのテキスト文字列は、統合された 1 つのビュー、[文字列エディター] ビューに表示されます。このビューを使って、ボタン テキストから機能の説明まで全ての文字列を編集できます。またこのビューでは、各言語の文字列エントリをファイルにエクスポートして、ファイルにリストされている値を翻訳してから、翻訳済みファイルをプロジェクトにインポートすることができます。

複数言語をサポートするプロジェクトで文字列エントリを処理する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*



エディション: *InstallShield Premier Edition では、複数言語インストールの作成をサポートします。*



メモ: *プロジェクトに言語を追加するには、[一般情報] ビューの "セットアップ言語" 設定を使用します。*

プロジェクトは、ローカライズ可能な各テキスト文字列およびプロジェクトでサポートする各言語用の文字列エントリを含みます。[文字列エディター] ビューは、プロジェクト内の言語非依存文字列 ID の一覧と対応する言語固有の値が表示されます。

複数言語サポートを含めるプロジェクトでは、文字列エントリに関する次の詳細にご注意ください:

- InstallShield ですべてのローカライズ可能な設定の値 (たとえば、機能の "表示名" と "説明" 設定) は、プロジェクトのデフォルト言語を使用します。プロジェクトのデフォルト言語は、[文字列エディター] ビューで指定できます。これらのローカライズ可能な設定の値を編集すると、InstallShield が [文字列エディター] ビューでデフォルト言語の文字列エントリの値を同時に更新します。

  同様に、[文字列エディター] ビューを使ってプロジェクトのデフォルト言語の文字列値を変更すると、InstallShield が、その文字列エントリの値を表示するその他の InstallShield ビューの設定を同時に更新します。

- プロジェクトでサポートする各言語は、同じ一連の文字列 ID を使用します。

  したがって、プロジェクトに文字列エントリを追加すると、InstallShield は各言語に対して同じ文字列 ID を使用します。文字列 ID の名前を変更すると、すべての言語の文字列 ID も変更されます。文字列エントリを削除すると、各言語の文字列エントリが削除されます。

- 新しい言語をプロジェクトに追加すると、プロジェクト内の他の言語で利用可能な文字列 ID と同じ文字列 ID がその言語にも追加されます。デフォルトの文字列エントリ (つまり、組み込みダイアログやその他のユーザー インターフェイス要素で使用できる文字列エントリ) には、既に翻訳済みの文字列値が追加されます。カスタム文字列エントリには、新しい言語用のデフォルト言語からの文字列が使用されます。必要に応じて、新しい言語用の文字列値をどれでも変更することができます。

プロジェクト作成の終了後、テキスト ファイルに各言語の文字列エントリをエクスポートして、翻訳することができます。テキスト ファイルの文字列エントリを翻訳した後、それをプロジェクトにインポートできます。

InstallShield で文字列エントリを使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*



メモ: *Windows アプリ パッケージのローカリゼーションを行うには、InstallShield が搭載されているマシン上に Windows 10 SDK もインストールする必要があります。これがインストールされていない場合、InstallShield はデフォルト言語のみを含む Windows アプリ パッケージ (.appx) をビルドします。*

プロジェクトを通して文字列をハードコード化する代わりに、InstallShield でローカライズ済みテキストを受け入れる領域では、文字列エントリを使用できます。文字列エントリを選択する方法は、それらを使用する場所によって異なります。

様々なビューでの設定

InstallShield のビュー内にある設定で値を入力するときに、その値がエンド ユーザーに表示されるテキスト文字列である場合、自動的にその設定の文字列 ID が使用されます。InstallShield は、文字列値の前に中括弧で囲まれた文字列 ID を配置します。機能の "表示名" 設定の値の例を以下に示します:

{ID_STRING24}My New Feature

この例では、**ID_STRING24** が、"表示名" 設定で使用される文字列 ID です。**My New Feature** は、プロジェクトのデフォルト言語の文字列値です。



タスク ***ローカライズ可能な文字列エントリを受け付ける設定での処理には、以下のいずれかを行います:***

- プロジェクト内の何処にも存在しない新しいテキスト文字列を入力するには、使いたいテキスト文字列を入力します。入力した値には、自動的に新しい文字列 ID が割り当てられます。

  プロジェクトが複数の言語をサポートする場合、プロジェクトの言語すべてに新しい文字列 ID が追加され、入力した文字列値がすべての言語の値として使用されます。

- プロジェクトで使用されている既存する文字列エントリの一覧を参照するには、設定の値をクリックしたときに右側に表示される省略記号ボタン (...) をクリックします。既存の文字列エントリを選択、または新しい文字列エントリを作成できる [文字列の選択] ダイアログ ボックスが開きます。



***注意・**既存の文字列値を編集するのと、InstallShield 設定でテキストを入力して新しい文字列 ID を入力するのは異なる点に注意してください。設定に対して文字列 ID を選択した後に設定のローカライズ可能なテキストを削除しても、単に現在値が削除されるだけです。これでは、現在の文字列 ID は新しい ID に置き換えられません。*

ダイアログ エディター

ダイアログ エディターには、エンドユーザーに表示される文字列を使用できるたくさんのコントロールがあります。たとえば、テキストプロパティやツールチッププロパティは常にローカライズ可能テキストが必要になります。

ダイアログ エディターで行う文字列エントリの処理は、InstallShield のその他の場所で文字列 ID を選択するのと似ています。



タスク ***ダイアログ エディターとローカライズ可能テキストを使用するコントロールのプロパティを使用するには、以下のいずれかを行います:***

- コントロールのプロパティ シートをクリックして、文字列を編集します。すると、現在の言語の文字列エントリが変更されます。最初に文字列を選択せずにテキスト値を入力すると、InstallShield がプロジェクトの新しい文字列 ID を作成します。

  プロジェクトが複数の言語をサポートする場合、プロジェクトの言語すべてに新しい文字列 ID が追加され、入力した文字列値がすべての言語の値として使用されます。

- ローカライズ可能なプロパティの値を編集するためにクリックすると、省略記号ボタン (...) がプロパティ シート内に表示されます。省略記号ボタンをクリックして、現在の言語の文字列エントリを表示します。

ダイアログ コントロールのプロパティと、InstallShield 内のその他のビューにある設定との最も大きな違いは、その他のビューでは常にデフォルト言語の文字列が表示される点です。ただし、ダイアログのレイアウトを編集するとき、最初にダイアログの言語を選択しなくてはなりません。すると、ダイアログとプロパティシートの文字列はすべて編集しているダイアログで使用中の言語で表示されます。

InstallScript スクリプト エディター ペイン

InstallScript ビューの InstallScript スクリプト エディター ペインでは、[文字列の選択] ダイアログ ボックスを使って、文字列エントリを選択および編集することができます。



タスク ***InstallScript で文字列 ID を使用するには、以下の手順を実行します。***

1. スクリプト内で文字列 ID を挿入する場所にカーソルを置きます。
2. **[編集]** メニューで、**[挿入]** をポイントして **[文字列エントリ]** を選択します。**[文字列の選択]** ダイアログ ボックスが開きます。
3. 以下のいずれかを実行します。
   - 既存の文字列エントリを使って、スクリプトで使用する文字列エントリを含む行をクリックします。
   - 新しい文字列エントリを作成するには、**[新規作成]** ボタンをクリックします。**[文字列エントリ]** ダイアログ ボックスが開き、ここで新しい文字列 ID と値を作成できます。
4. **[OK]** をクリックします。

InstallShield は、スクリプト中に (@) 記号に続けて文字列 ID を配置します。

詳細は、「文字列定数演算子(@)」を参照してください。



ヒント: *ハードコード化されたユーザー インターフェイス文字列を直接 InstallScript コードに含めることもできますが、この方法は回避することをお勧めします。InstallScript コード内にハードコード化された文字列は、Unicode として格納されません。したがって、正しいコード ページが設定されたシステム上でインストールが実行された場合にのみ、言語が正しく表示されます。[文字列エディタ] ビューを使ってプロジェクトに文字列を追加して、InstallScript コードからその文字列に関連付けられた文字列 ID を参照することで、この問題を回避できます。*

ビルド時と実行時に文字列エントリに起きる処理

InstallShield は、文字列エントリを InstallShield プロジェクト ファイル (.ism) の ISString テーブルに格納します。ほとんどの場合、リリースを作成するとき、ビルド時に文字列 ID ではなく文字列値が使用されます。つまり、文字列 ID はリリースに組み込まれず、文字列 ID は実行時に使用されません。

InstallShield がビルド時に文字列 ID を使用する唯一の例外は、プロジェクトに InstallScript コードが含まれている場合です。この場合、InstallShield がビルドするリリースには、すべての文字列エントリを含む文字列テーブルが含まれます。これらの文字列テーブルによって、ハードコード化されたテキスト文字列の代わりに、InstallScript コード内に文字列 ID を使用できるようになります。実行時、インストールは必要に応じて、文字列 ID を文字列値と置き換えます。

文字列エントリを追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[文字列エディター] ビューでは、後で InstallShield 内の設定や、プロジェクトに含まれるダイアログ、および InstallScript コードで参照することが可能な新しい文字列エントリを作成できます。



タスク ***新しい文字列エントリをプロジェクトに追加するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. 以下のいずれかを実行します。
   - **[新しい文字列エントリ]** ボタンをクリックします。
   - INSERT キーを押す。

   [文字列エントリ] ダイアログ ボックスが開きます。

3. **[文字列 ID]** ボックスで、文字列エントリに使用する言語非依存の文字列 ID を指定するか、自動的にボックスに挿入される新しいデフォルト文字列 ID のままに残します。
4. **[値]** ボックスに、文字列エントリに使用するローカライズ可能なテキスト文字列を指定します。
5. **[コメント]** ボックスに、オプションとして、文字列エントリについて内部で使用するメモを指定します。コメントは実行時には表示されません。
6. **[OK]** をクリックします。

[文字列エディター] ビューで、新しい文字列エントリ用の新しい行が追加されます。



ヒント: *プロジェクトが複数の言語をサポートする場合、プロジェクトの言語すべてに文字列エントリが追加され、入力した文字列値がすべての言語の値として使用されます。必要な場合は、言語の文字列値を編集できます。*



メモ: *InstallShield では、ローカライズ可能なテキストを受け入れる設定には必ず文字列 ID を使用しなくてはならないため、既存の文字列 ID を選択せずに設定に値を入力すると、プロジェクトに新しい文字列エントリが自動的に追加されます。*

文字列エントリを編集する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[文字列エディター] ビューには、プロジェクト内の文字列エントリを変更できる、スプレッドシート形式のテーブルが表示されます。



タスク ***[文字列エディター] ビューで文字列エントリを編集するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. 変更する文字列エントリを見つけます。
3. 以下のいずれかを実行します。
   - テーブル セル内のテキストをすべて上書きするには、テーブル セルをクリックしてから、新しい ID、値、またはコメントを入力します。
   - テーブル セル内の特定の位置にカーソルを配置するには、その場所をダブルクリックします。次に、変更を入力します。

文字列 ID の名前を変更すると、プロジェクト内の全ての言語に対する文字列 ID も変更されますので、ご注意ください。

文字列エントリを編集すると、変更を行った日付と時刻が [更新日時] 列に表示されます。[文字列エディター] ビュー内の文字列エントリは、更新日時順に並べ替えることができます。これは、文字列が最後に翻訳された以降に変更された文字列を識別するのに、便利です。



ヒント: *InstallShield の他のビュー内にあるローカライズ可能な設定を編集中に、その設定にある省略記号ボタン (...) をクリックできます。[文字列の選択] ダイアログ ボックスが開き、ここで選択した設定に使用する文字列エントリを指定できます。*



プロジェクト: *基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトでは、一部の文字列値は角括弧でかこまれた Windows Installer プロパティを含みます (例、***Install [ProductName]***)。実行時に、このプロパティと括弧はプロパティ値で置き換えられます。*

*文字列値の中には、中括弧でかこまれたフォント情報を含むものもあります (例、***{&MSSansBold8}OK***)。フォント情報は、実行時に文字列を表示するのに使用するスタイルの詳細を示します。*

文字列 ID のインスタンスを検索する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

文字列 ID は InstallShield プロジェクト全体で使用されます。特定の文字列 ID がプロジェクト内の何処で使用されているかを確認したい場合、[文字列エディター] ビューで検索を行うことができます。



タスク ***プロジェクト内の特定の文字列 ID をすべて見つけるには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. 検索を行う文字列 ID を含む行を選択します。

   連続する複数の行を選択するには、最初の行をクリックしてから SHIFT を押しながら最後の行をクリックします。連続しない複数の行を選択するには、最初の行をクリックしてから CTRL を押しながら追加するそれぞれの行をクリックします。
3. **[選択した文字列をプロジェクトで検索]** ボタンをクリックします。

InstallShield が、検索結果を [出力] ウィンドウの [結果] タブに表示します。[結果] タブは、文字列 ID が使用されている回数を表示します。また、[ダイレクト エディター] ビュー内のどの場所に文字列 ID が使用されているかも示します。

文字列 ID がプロジェクト内で使用されていない場合、[結果] タブには 0 個見つかったと表示されます。



メモ: *InstallShield は、文字列 ID のインスタンスをプロジェクトの InstallScript ファイル (.rul) では検索しません。*

文字列エントリの検索と置換



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[文字列エディター] は、標準の検索/置換機能をサポートします。この機能は、プロジェクト内で特定の文字列を検索して、改訂済みの文字列と置換するときに便利です。

文字列エディターで文字列を検索する



タスク ***文字列エディターで文字列を検索するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. **[文字列の検索]** ボタンをクリックします。**[検索]** ダイアログ ボックスが開きます。
3. **[検索する文字列]** ボックスで、検索する文字列を入力します。
4. 必要に応じて、その他のオプションを選択します。
5. **[次を検索]** をクリックします。

InstallShield が、指定された文字列の最初のインスタンスを検出します。

文字列エディターで文字列を検索/置換する



タスク ***文字列エディターで文字列を検索/置換するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. **[検索/置換]** ボタンをクリックします。**[置換]** ダイアログ ボックスが開きます。

3. **[検索する文字列]** ボックスで、置換するテキストを入力します。
4. **[置換後の文字列]** ボックスで、新しい文字列を入力します。
5. 必要に応じて、その他のオプションを選択します。
6. **[次を検索]**、**[置換]**、または **[すべて置換]** をクリックします。

InstallShield が必要に応じて文字列を置換します。

文字列エントリを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

プロジェクト内の特定の文字列エントリが不要になった場合は、それを削除できます。

文字列エントリを削除すると、プロジェクトに含まれる各言語からその文字列エントリが削除されます。



タスク ***文字列エントリをプロジェクトから削除するには、次の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. 削除する文字列エントリを見つけます。
3. **[選択した文字列を削除]** ボタンをクリックするか、または DELETE キーを押します。

ローカライズ可能な設定から文字列 ID を削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallShield のビュー内にある設定で値を入力するときに、その値がエンド ユーザーに表示されるテキスト文字列である場合、自動的にその設定の文字列 ID が使用されます。InstallShield は、文字列値の前に中括弧で囲まれた文字列 ID を配置します。

設定の値を削除すると、実際は現在の言語用の文字列値が削除されます。そのような場合、設定には空の値を持つ文字列 ID が使用されます。

つまり、ローカライズ可能な設定から文字列 ID を削除する場合、それを異なる文字列 ID に置き換えるか、新しい文字列 ID を追加しなくてはなりません。

文字列 ID の置換についての詳細は、「文字列エントリを編集する」を参照してください。新しい文字列 ID の作成方法については、「文字列エントリを追加する」を参照してください。

ビルボードを表示する



プロジェクト: *ビルボードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

ビルボードをプロジェクトに追加して、インストール処理中にエンド ユーザーに対して情報を提供できます。ビルボードは、エンド ユーザーと連絡を取ったり、広告、教育、およびエンターテインメントを提示するために使用することができます。たとえば、ビルボードを使ってインストール中の製品に含まれる新しい機能の概要や貴社の他の製品についての情報を提供できます。各ビルボードは、貴社のグラフィック担当者がファイル転送の外観を完全にカスタマイズできるファイルです。



プロジェクト: *ビルボードのサポートは、使用しているプロジェクトの種類によって異なります。*

このセクションには、次のトピックが含まれています:

- 基本の MSI インストールでビルボードを表示する
- InstallScript および InstallScript MSI インストールでビルボードを表示する

基本の MSI インストールでビルボードを表示する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

ビルボードをプロジェクトに追加して、インストール処理中にエンド ユーザーに対して情報を提供できます。ビルボードは、エンド ユーザーと連絡を取ったり、広告、教育、およびエンターテインメントを提示するために使用することができます。たとえば、ビルボードを使ってインストール中の製品に含まれる新しい機能の概要や貴社の他の製品についての情報を提供できます。各ビルボードは、貴社のグラフィック担当者がファイル転送の外観を完全にカスタマイズできるファイルです。

基本の MSI プロジェクトにおけるビルボード ファイルの種類



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield では、ビルボード用に様々な種類のファイルがサポートされています:

- Adobe Flash アプリケーション ファイル (.swf)
- イメージ (.bmp、.gif、.jpg、および .jpeg)

Flash アプリケーション ファイルを表示するために必要な Adobe Flash Player がターゲット システムに存在しない場合、インストールはそれを検知して Flash ビルボードの代わりにイメージ ビルボードを表示します。このため、Flash ビルボードをプロジェクトに含める場合は、1 つ以上のイメージ ビルボードもプロジェクトに含めることが推奨されます。



メモ: *プロジェクトに複数のイメージ ビルボードを追加することができますが、Adobe Flash アプリケーション ファイル ビルボードの場合は 1 つだけしか追加できません。*

基本の MSI プロジェクトにおけるビルボードの種類



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield は、異なる種類のビルボードをサポートします。たとえば、インストールが全画面背景を使用し、ビルボードを前景に、また小さい進行状況ボックスを画面の右下に表示するスタイルがあります。別のスタイルでは、インストールがビルボードを表示する標準サイズのダイアログを表示します。このダイアログの下の部分に、進行状況バーが表示されます。

各ビルボード タイプの説明とサンプル スクリーンショットは、次のとおりです。

全画面表示、右下に小さい進行状況ボックスを表示する

全画面表示、右下に小さい進行状況ボックスを表示するタイプのビルボードは、インストールが標準エンドユーザー ダイアログを表示するときに、全画面の背景も表示します。ファイルの転送中、インストールが全画面背景を使用し、ビルボードを前画面に、また小さい進行状況ボックスを画面の右下に表示します。

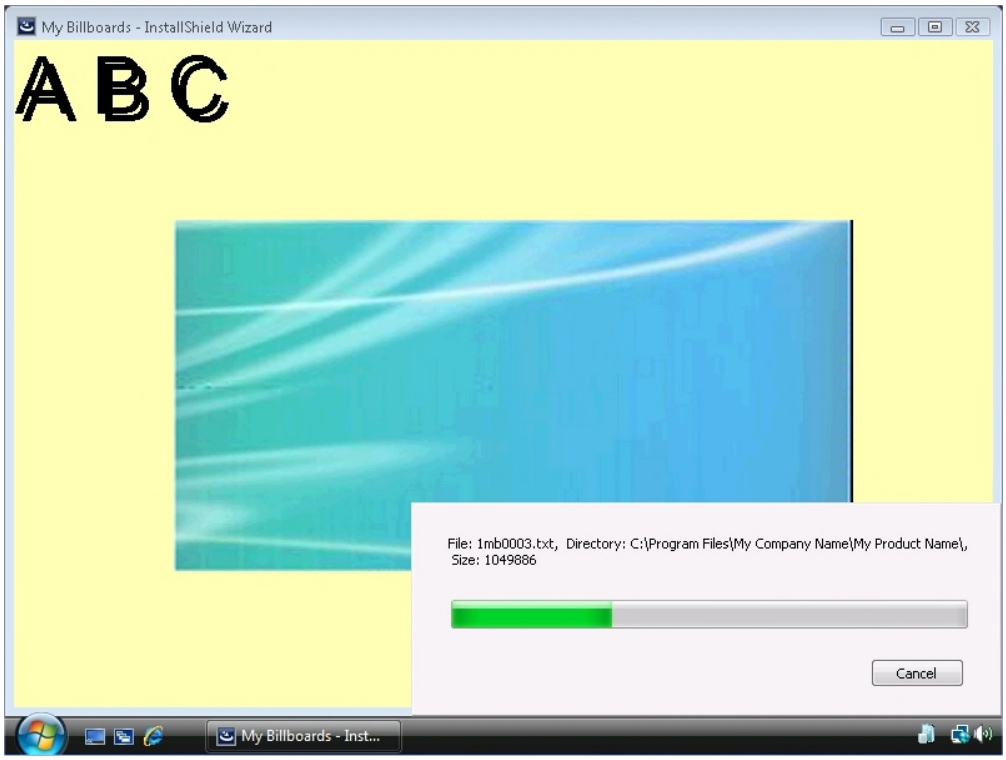


図 28: 全画面表示、右下に小さい進行状況ボックスを表示する

サンプル スクリーンショットでは、ビルボードは中央にある青緑色の長方形です。構成可能なビルボード設定の一部は次のように設定されています:

- 原点ー 中央揃え
- タイトル ー A B C
- フォント ー 48 pt. Arial
- 背景色 ー 黄色

ウィンドウ表示、標準の進行状況を表示する

ウィンドウ表示、標準の進行状況を表示するタイプののビルボードでは、ファイルの転送中、インストールはビルボードを表示する標準サイズのダイアログを表示します。このダイアログの下の部分に、進行状況バーが表示されます。このスタイルの場合、インストールは背景を表示しません。

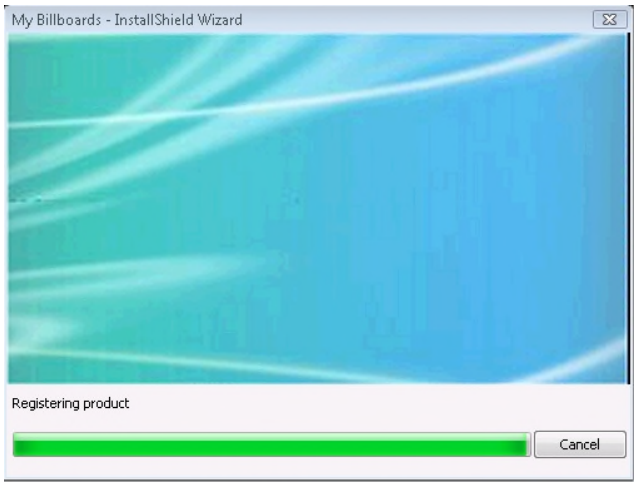


図 29: ウィンドウ表示、標準の進行状況を表示する

サンプル スクリーンショットでは、ビルボードは青緑色の長方形です。そのサイズは、幅が 544 ピクセルで、高さが 281 ピクセルです。

ウィンドウ表示、右下に小さい進行状況ボックスを表示する(ビルボードなし)

ウィンドウ表示、右下に小さい進行状況ボックスを表示する(ビルボードなし)タイプのビルボードでは、ファイル転送中にインストールは小さい進行状況ボックスを画面の右下に表示します。ビルボードまたは背景は一切表示しません。

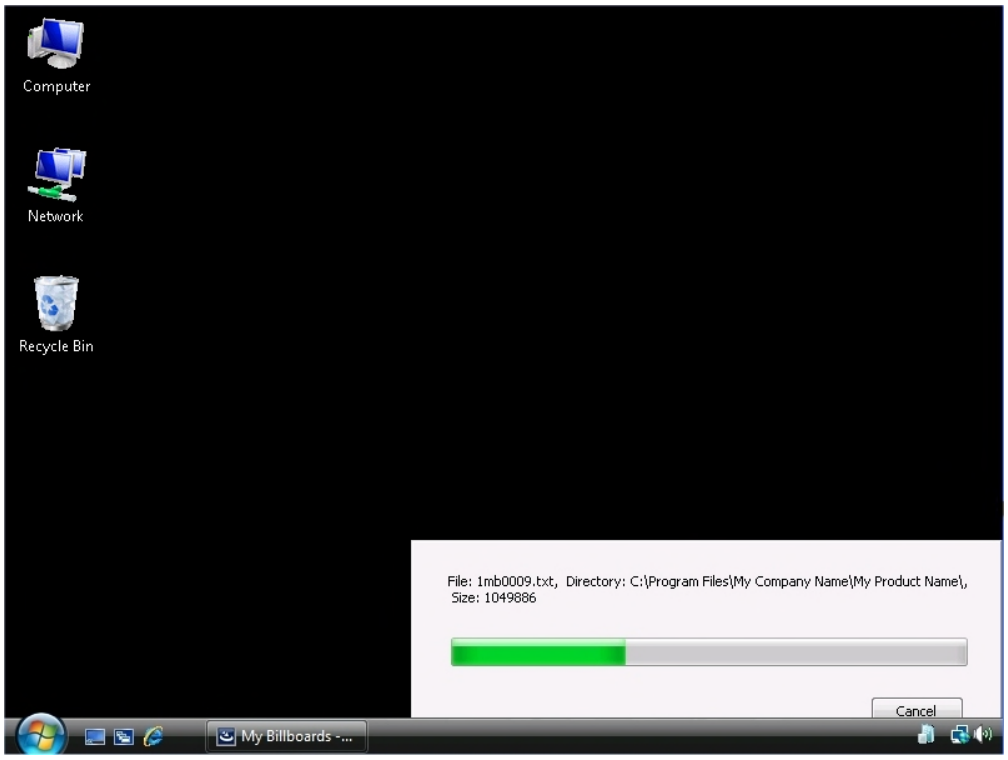


図 30: ウィンドウ表示、右下に小さい進行状況ボックスを表示する(ビルボードなし)

サンプル スクリーンショットに見られるように、進行状況バーが表示されますが、ビルボードは表示されません。黒色の背景は、エンド ユーザーのデスクトップです。

基本の MSI プロジェクトで使用するビルボードの種類を指定する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield は、異なる種類のビルボードをサポートします。



タスク ***インストールで使用するビルボードの種類を指定するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ビルボード]** をクリックします。
2. 中央のペインで、**[ビルボード]** エクスプローラーをクリックします。右側のペインに **"ビルボードの種類"** 設定が表示されます。
3. **"ビルボードの種類"** 設定で、適切なビルボードの種類を選択します。

各ビルボード タイプのサンプルは、「基本の MSI プロジェクトにおけるビルボードの種類」を参照してください。

基本の MSI プロジェクトに Adobe Flash アプリケーション ファイル ビルボードを追加する



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield を使って、ファイル転送処理中に Flash アプリケーション ファイル ビルボードを表示できます。Flash アプリケーション ファイルは、ビデオ、動画、音声、インタラクティブ インターフェイス、ゲーム、テキスト、その他の .swf ファイルがサポートするあらゆる要素で構成されます。Flash ビデオ ファイル (.flv) や MP3 オーディオ ファイルは .swf ファイルに埋め込んで、ファイル転送中にターゲット システム上のローカルで使用できるようにすることが推奨されます。.swf ファイルは Web サイト上に配置された外部ファイルを参照することが可能ですが、この外部実装ではエンド ユーザーがインターネットに接続されていることが必須となります。



タスク ***Adobe Flash アプリケーション ファイル ビルボードをインストール プロジェクトに追加するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ビルボード]** をクリックします。
2. **[ビルボード]** エクスプローラーで、**[Adobe Flash アプリケーション ファイル(.swf)]** を右クリックしてから、**[新しいビルボード]** を選択します。新しいビルボード が **NewBillboard1** という名前で作成されます。
3. ビルボードの名前を入力します。この名前はインストールを作成するときにアイテムを識別するために使用されます。この名前はインストール時には表示されません。
4. 右側のペインで、ビルボードの設定を構成します。



メモ- *.swf ファイルの作成に使用した Flash またはその他のツールのバージョンがターゲット システムにインストールされている Flash Player よりも新しい場合、ターゲット システム上で一部の Flash 機能が予定どおりに動作しない可能性があります。*

Flash アプリケーション ファイルを表示するために必要な Adobe Flash Player がターゲット システムに存在しない場合、インストールはそれを検知して Flash ビルボードの代わりにイメージ ビルボードを表示します。このため、Flash ビルボードをプロジェクトに含める場合は、1 つ以上のイメージ ビルボードもプロジェクトに含めることが推奨されます。

基本の MSI プロジェクトにイメージ ビルボードを追加する



プロジェクト- *この情報は、基本の MSI プロジェクトに適用します。*

ファイル転送処理中に 1 つのイメージ ビルボードのみを表示したり、一連のイメージ ビルボードで、各ビルボードが特定の時間表示されるよう設計したりすることができます。InstallShield は、.bmp、.gif、.jpg、および .jpeg イメージ ファイルをサポートします。



メモ- *動画 .gif ファイルはサポートされていません。ビルボードで動画を使用したい場合は、Adobe Flashアプリケーション ファイル ビルボードの使用をご検討ください。*



タスク ***イメージ ビルボードをインストールに追加するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ビルボード]** をクリックします。
2. **[ビルボード]** エクスプローラーで、**[イメージ]** を右クリックしてから、**[新しいビルボード]** をクリックします。新しいビルボード が **NewBillboard1** という名前で作成されます。
3. ビルボードの名前を入力します。この名前はインストールを作成するときにアイテムを識別するために使用されます。この名前はインストール時には表示されません。
4. 右側のペインで、ビルボードの設定を構成します。

基本の MSI プロジェクトでビルボードの設定を構成する



プロジェクト- *この情報は、基本の MSI プロジェクトに適用します。*

Adobe Flash アプリケーション ファイル ビルボード、またはイメージ ビルボードをプロジェクトに追加するとき、その設定を構成する必要があります。

タスク ***ビルボードの設定を構成するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ビルボード]** をクリックします。
2. 中央ペインの **[ビルボード]** エクスプローラーで構成するビルボードを選択します。右側のペインにビルボードが表示されます。
3. 必要に応じて設定を構成します。

ビルボードの各設定についての詳細は、「Adobe Flash アプリケーション ファイル ビルボードとイメージ ビルボードの設定」を参照してください。

リリースをビルドまたは起動せずにビルボードをプレビューする

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield では、リリースをビルドおよび実行せずに、実行時にビルボードがどのように表示されるのかをプレビューできます。

ビルボードをプレビューすると、そのビルボードに現在構成されている背景色、位置、および関連設定を使ったビルボードの外観を確認できます。

タスク ***ビルボードをプレビューするには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ビルボード]** をクリックします。
2. 中央ペインの **[ビルボード]** エクスプローラーでプレビューするビルボードを右クリックしてから、**[ビルボードのプレビュー]** を選択します。

InstallShield が、実行時に表示されるビルボードのプレビューを表示します。

プレビューを止めるには、[プレビュー] ウィンドウにある [キャンセル] ボタンをクリックします。

ヒント: *ビルボードのプレビューを使うと、Flash またはイメージ ビルボードが、選択された異なるビルボード タイプではどのように表示されるのかを確認するのに特に便利です。ビルボードをプレビューし、ビルボード タイプの変更してから、再度ビルボードをプレビューすることができます。*

基本の MSI プロジェクトでビルボードの順序を設定する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

イメージ ビルボードは、[ビルボード] ビューで表示されているのと同じ順序で、上から下に順番に表示されます。

タスク ***イメージ ビルボードが実行時に表示される順番を変更するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ビルボード]** をクリックします。
2. **[ビルボード]** エクスプローラーで、移動するビルボードを 1 つ右クリックし、**[上に移動]** または **[下に移動]** をクリックします。

すべてのビルボードが正しく並べ替えられるまで最後のステップを繰り返します。

ビルボードを含む基本の MSI インストールにおける実行時の動作

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

重要: *インストールにビルボードが含まれている場合、インストールには* Setup.exe *セットアップランチャーを含めなくてはなりません。セットアップランチャーが実行時にビルボードを表示するため、これが必須となります。[リリース] ビューにあるリリースについての Setup.exe タブでは、セットアップランチャーを使用するかどうかなどの情報を指定することができます。詳細については、「リリースの [Setup.exe] タブ」を参照してください。*

インストールに Flash ビルボードと 1 つ以上のイメージ ビルボードが含まれている場合、実行時のファイル転送処理中に Flash ビルボードまたはイメージ ビルボードのうち 1 つのビルボード タイプのみが表示されます。

- Flash Player がターゲット システムに存在する場合、インストールは Flash ビルボードを表示します。
- Flash Player が存在しない場合、インストールはイメージ ビルボードを表示します。

実行時の動作は、インストールが Flash ビルボードかイメージ ビルボードのどちらを表示するかによって、多少異なります:

- **インストールが Flash ビルボードを表示する場合** — ファイル転送が完了すると、インストールは Flash ビルボードに割り当てられた時間が経過するまで、それを表示し続けます。割り当てられた時間が経過すると、インストールはビルボードの表示を終了して、適切な SetupComplete ダイアログを表示します。

  ファイル転送が Flash ビルボードに割り当てられた時間よりも長くかかった場合、インストールはファイル転送が終了するまで、Flash ビルボードを表示し続けます。

- **インストールがイメージ ビルボードを表示する場合** — ファイル転送が完了すると、その他のビルボードがスケジュールされていても、また現在のビルボードに割り当てられた時間が経過していなくても、インストールはイメージ ビルボードの表示を終了します。次に、インストールは適切な SetupComplete ダイアログを表示します。

  ファイルの転送時間が、ビルドボードに割り当てられた時間を超える場合、インストールはファイル転送が終了するまでビルボードを表示し続けます。[ビルボード] ビューの "ビルボードのループ" 設定に [いいえ] が選択されている場合、インストールがファイルの転送を終了する前に最後のビルボードに到達したとき、インストールはファイルの転送が終了するまで最後のイメージ ビルボードを表示し続けます。次に、インストールは適切な SetupComplete ダイアログを表示します。この設定に [はい] が選択されている場合、インストールがファイルの転送を終了する前に最後のビルボードに到達したとき、インストールは最初のビルボードから再び表示を開始します。必要な場合、ファイルの転送が終了して SetupComplete ダイアログが表示されるまでループが継続します。

メモ: *.swf ファイルの作成に使用した Flash またはその他のツールのバージョンがターゲット システムにインストールされている Flash Player よりも新しい場合、ターゲット システム上で一部の Flash 機能が予定どおりに動作しない可能性があります。*

基本の MSI プロジェクトからビルボードを削除する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

タスク

インストールからビルボードを削除するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ビルボード]** をクリックします。
2. **[ビルボード]** エクスプローラーで、削除するビルボードを右クリックして **[削除]** を選択します。

InstallScript および InstallScript MSI インストールでビルボードを表示する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ビルボードをプロジェクトに追加して、インストール処理中にエンド ユーザーに対して情報を提供できます。ビルボードは、エンド ユーザーと連絡を取ったり、広告、教育、およびエンターテインメントを提示するために使用することができます。たとえば、ビルボードを使ってインストール中の製品に含まれる新しい機能の概要や貴社の他の製品についての情報を提供できます。各ビルボードは、貴社のグラフィック担当者がファイル転送の外観を完全にカスタマイズできるファイルです。

InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルおよびファイルの種類

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield は、2 種類のビルボードをサポートします。1 つめは、全画面の背景ウィンドウを含み、2 つめはこれを含みません。両方のスタイルで進行状況バーが表示されます。

以下に、両方の種類のビルボードの説明とサンプル、および適切なサポート対象ファイルの種類を示します。

ウィンドウ表示のビルボードで進行状況を表示する

このスタイルのビルボードでは、インストールが標準サイズのダイアログを表示します。ファイルの転送中、ダイアログ上部にビルボードが表示され、ダイアログの下部に進行状況バーが表示されます。このスタイルのビルボードの場合、インストールは背景ウィンドウを表示しません。

このスタイルのビルボードでは、以下の種類のファイルを使用できます:

- Adobe Flash アプリケーション ファイル (.swf)
- イメージ (.bmp、.gif、.jpg、および .jpeg)

Flash アプリケーション ファイルを表示するために必要な Adobe Flash Player がターゲット システムに存在しない場合、インストールはそれを検知して Flash ビルボードの代わりにイメージ ビルボードを表示します。このため、Flash ビルボードをプロジェクトに含める場合は、1 つ以上のイメージ ビルボードもプロジェクトに含めることが推奨されます。

メモ- *進行状況を表示するウィンドウ表示のビルボードでは、プロジェクトに複数のイメージ ビルボードを追加することができますが、Adobe Flash アプリケーション ファイル ビルボードの場合は 1 つだけしか追加できません。*

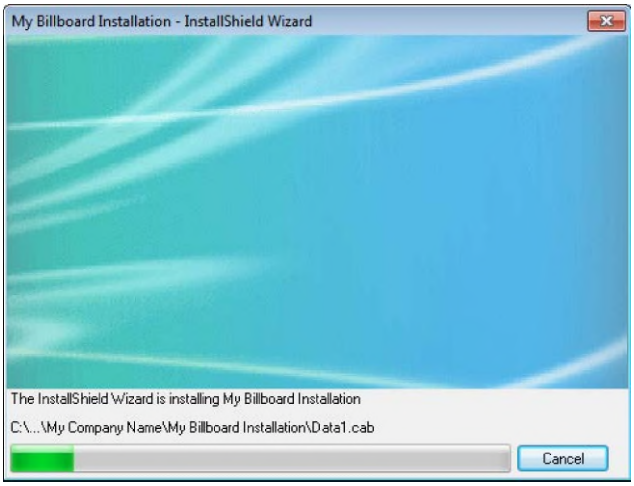


図 31: ウィンドウ表示のビルボードで進行状況を表示する

サンプル スクリーンショットでは、ビルボードは青緑色の長方形です。そのサイズは、幅が 544 ピクセルで、高さが 281 ピクセルです。

メモ- *スキン ダイアログではウィンドウ表示のビルボードを使用できません。スキンについての詳細は、「ダイアログ スキン」を参照してください。*

全画面背景ウィンドウ表示ビルボード

このスタイルのビルボードでは、インストールが標準のエンド ユーザー ダイアログを表示するとき、全画面の背景ウィンドウも表示します。ファイルの転送中、全画面の背景ウィンドウの手前にビルボードが表示され、標準サイズの進行状況ダイアログがビルボードの手前に表示されます。このビルボード スタイルには、背景ウィンドウは不要です。

このスタイルのビルボードでは、イメージ ファイル (.bmp、.gif、.jpg、および .jpeg) を使用できます。

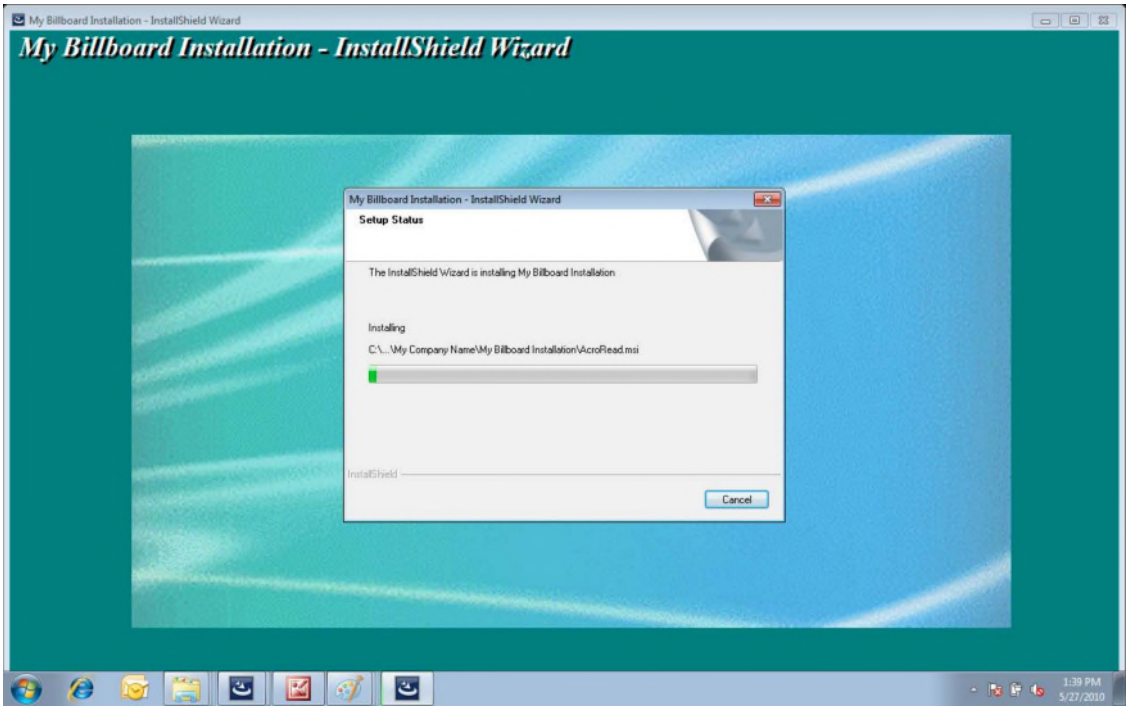


図 32: 全画面背景ウィンドウ表示ビルボード

サンプル スクリーンショットでは、ビルボードは進行状況ダイアログの後ろにある青緑色の長方形イメージで、背景ウィンドウの手前にあります。

InstallScript または InstallScript MSI プロジェクトにおけるビルボード ファイルの名前

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

メモ: *ここで説明されているAdobe Flash アプリケーション ファイル サポートは、使用するビルボード スタイルがウィンドウ表示スタイルで進行状況を含む場合に適用します。全画面背景ウィンドウ表示ビルボード スタイルには適用しません。Flash ファイル サポートを全画面表示で背景ウィンドウを含むスタイルと共に使用するには、* **PlayMMedia** *関数を使用します。このサポートを使うと、以下の命名規則は Adobe Flash ファイルに適用しません。詳細については、「PlayMMedia」を参照してください。この実装で背景ウィンドウを表示する方法については、「InstallScript および InstallScript MSI インストールで背景ウィンドウを表示する」をご参照してください。*

プロジェクトに複数のイメージ ビルボードを追加することができますが、Adobe Flash アプリケーション ファイル ビルボードの場合は 1 つだけしか追加できません。

Flash アプリケーション ファイルを表示するために必要な Adobe Flash Player がターゲット システムに存在しない場合、インストールはそれを検知して Flash ビルボードの代わりにイメージ ビルボードを表示します。このため、Flash ビルボードをプロジェクトに含める場合は、1 つ以上のイメージ ビルボードもプロジェクトに含めることが推奨されます。

ビルボードをプロジェクトに追加する最初の手順は、適切な場合は Adobe Flash アプリケーション ファイルと、インストールのビルボードとして使用するイメージ ファイル (.bmp、.gif、.jpg、または .jpeg) の作成です。ファイルに正しい名前を付けたら、ビルボードをプロジェクトに追加できます。Flash アプリケーション ファイルを表示するために必要な Adobe Flash Player がターゲット システムに存在しない場合、インストールはそれを検知して Flash ビルボードの代わりにイメージ ビルボードを表示します。このため、Flash ビルボードをプロジェクトに含める場合は、1 つ以上のイメージ ビルボードもプロジェクトに含めることが推奨されます。

ビルボードファイルには、指定の命名規則を使用する必要があります。各ファイル名は **bbrd** で始まり、ビルボード番号 (1 から 99) が続きます。最後に、サポート対象のファイル拡張子 (.swf、.bmp、.gif、.jpg、または .jpeg) で終わります。プロジェクトには、最多で 1 つの Adobe Flash ファイル (.swf) を含むことができます。.swf ファイルを含める場合、その名前にはプロジェクトに含まれるビルボード ファイルのリストで 1 番低い番号を含めなくてはなりません (たとえば、bbrd1.swf)。

ビルボードは、ファイル名の順番に表示されます。たとえば、bbrd2.bmp といういう名前のビルボード ファイルは、bbrd4.gif の前に表示されます。

ファイル名番号が連続番号である必要はありません。つまり、プロジェクトに bbrd1.jpg、bbrd3.jpg、および bbrd5.jpg を追加すると、実行時に各イメージが順番通りに表示されます。

ヒント: *各イメージ ビルボードを表示する時間は、インストール プロジェクトに含まれているイメージ ビルボードの数によって異なります。表示時間の割合は、おおよそ 1 をビルボードの数で割ったものです。たとえば、インストールに 4 つのビルボードがある場合、各ビルボードが表示される時間は、ファイル転送時間の 25% の長さになります。*

InstallScript または InstallScript MSI プロジェクトに Adobe Flash アプリケーション ファイル ビルボードを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

メモ: *ここで説明されているAdobe Flash アプリケーション ファイル サポートは、使用するビルボード スタイルがウィンドウ表示スタイルで進行状況を含む場合に適用します。全画面背景ウィンドウ表示ビルボード スタイルには適用しません。Adobe ファイル サポートを全画面の背景ウィンドウと共に使用する場合、***PlayMMedia*** *関数を使用します。詳細については、「PlayMMedia」を参照してください。この実装で背景ウィンドウを表示する方法については、「InstallScript および InstallScript MSI インストールで背景ウィンドウを表示する」をご参照してください。*

InstallShield を使って、ファイル転送処理中に Flash アプリケーション ファイル ビルボードを表示できます。Flash アプリケーション ファイルは、ビデオ、動画、音声、インタラクティブ インターフェイス、ゲーム、テキスト、その他の .swf ファイルがサポートするあらゆる要素で構成されます。Flash ビデオ ファイル (.flv) や MP3 オーディ

オ ファイルは .swf ファイルに埋め込んで、ファイル転送中にターゲット システム上のローカルで使用できるようにすることが推奨されます。.swf ファイルは Web サイト上に配置された外部ファイルを参照することが可能ですが、この外部実装ではエンド ユーザーがインターネットに接続されていることが必須となります。



タスク ***Adobe Flash アプリケーション ファイル ビルボードをインストール プロジェクトに追加するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. **[サポートファイル]** エクスプローラーで、ビルボード (**言語非依存**、または言語固有のどちらでも可) を含む **[ビルボード]** 項目をクリックします。右側に [**ファイル**] ペインが表示されます。
3. **[ファイル]** ペインの任意の場所を右クリックして、**[ファイルの挿入]** をクリックするか、**[ファイル]** ペインにカーソルを置いて Insert キーを押します。**[開く]** ダイアログが開きます。
4. **bbrd1.swf** と名づけられた Adobe Flash アプリケーション ファイル ビルボード ファイルを選択してから、**OK** をクリックします。

ファイルがプロジェクトへ追加されます。



メモ: *.swf ファイルの作成に使用した Flash またはその他のツールのバージョンがターゲット システムにインストールされている Flash Player よりも新しい場合、ターゲット システム上で一部の Flash 機能が予定どおりに動作しない可能性があります。*

Flash アプリケーション ファイルを表示するために必要な Adobe Flash Player がターゲット システムに存在しない場合、インストールはそれを検知して Flash ビルボードの代わりにイメージ ビルボードを表示します。このため、Flash ビルボードをプロジェクトに含める場合は、1 つ以上のイメージ ビルボードもプロジェクトに含めることが推奨されます。

InstallScript または InstallScript MSI プロジェクトにイメージ ビルボードを追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ファイル転送処理中に 1 つのイメージ ビルボードのみを表示したり、一連のイメージ ビルボードで、各ビルボードが特定の時間表示されるよう設計したりすることができます。InstallShield は、.bmp、.gif、.jpg、および .jpeg イメージ ファイルをサポートします。



メモ: *動画 .gif ファイルはサポートされていません。ビルボードで動画を使用したい場合は、Adobe Flashアプリケーション ファイル ビルボードの使用をご検討ください。*



タスク ***イメージ ビルボードをプロジェクトに追加するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. **[サポートファイル]** エクスプローラーで、ビルボード (**言語非依存**、または言語固有のどちらでも可) を含む **[ビルボード]** 項目をクリックします。右側に [**ファイル**] ペインが表示されます。
3. **[ファイル]** ペインの任意の場所を右クリックして、**[ファイルの挿入]** をクリックするか、**[ファイル]** ペインにカーソルを置いて Insert キーを押します。**[開く]** ダイアログが開きます。
4. ビルボード ファイルを選択して [**OK**] をクリックします。

ファイルがプロジェクトへ追加されます。



ヒント: *各イメージ ビルボードを表示する時間は、インストール プロジェクトに含まれているイメージ ビルボードの数によって異なります。表示時間の割合は、おおよそ 1 をビルボードの数で割ったものです。たとえば、インストールに 4 つのビルボードがある場合、各ビルボードが表示される時間は、ファイル転送時間の 25% の長さになります。*

InstallScript または InstallScript MSI プロジェクトでコードを追加または変更してビルボードを表示する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

実行時にビルボードを表示するために必要な InstallScript コードは、使用するビルボードのスタイルによって異なります。

- **ウィンドウ表示のビルボードで進行状況を表示する**—実行時に、進行状況を表示するウィンドウ表示のビルボードを表示するには、STATUSBBRD 定数を**Enable** 関数と一緒に使います。
- **全画面背景ウィンドウ表示ビルボード**—全画面背景ウィンドウと共に表示されるビルボードを表示するには、FULLWINDOWMODE および BACKGROUND 定数を **Enable** 関数と一緒に使います。詳しくは、「InstallScript および InstallScript MSI インストールで背景ウィンドウを表示する」をご覧ください。

  Adobe Flash ビルボードをこのスタイルのビルボードと共に使用する場合、**PlayMMedia** 関数も使用する必要があります。

これらの異なるスタイルのビルボードについての詳細は、「InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルおよびファイルの種類」を参照してください。

InstallScript または InstallScript MSI プロジェクトでビルボードの順序を設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

イメージ ビルボードは、ファイル名の数字の順番に表示されます。bbrd2.bmp という名前のビルボードファイルは、bbrd3.bmp より先に、また bbrd1.bmp というファイルの後に表示されます。命名規則についてのガイドラインは、「InstallScript または InstallScript MSI プロジェクトにおけるビルボード ファイルの名前」を参照してください。

タスク ***ビルボードが表示される順番を変更するには、以下の手順を実行します。***

1. 順番を変更するビルボード ファイルを削除します。(詳細については、「InstallScript または InstallScript MSI プロジェクトでビルボードの順序を設定する」を参照してください。
2. Windows エクスプローラーを使用して、適切な順番に来るようにビルボード ファイルの名前を変更します。
3. 「InstallScript または InstallScript MSI プロジェクトにイメージ ビルボードを追加する」の説明にしたがって、名前を変更したファイルを追加します。

ファイル名番号が連続番号である必要はありません。つまり、プロジェクトに bbrd1.jpg、bbrd3.jpg、および bbrd5.jpg を追加すると、実行時に各イメージが順番通りに表示されます。

InstallScript または InstallScript MSI インストール中に特殊効果を持つビルボードを表示する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

SetDisplayEffect 関数を使うと、ビルボードが最初にメインのインストールウィンドウに表示されたときに特殊効果を使うことができます。ファイル転送中に **PlaceBitmap** を使ってビットマップを表示するか、ビルボードを表示する前に、この関数オプションの 1 つを選択します。

InstallScript または InstallScript MSI プロジェクトでビルボードを画面上の異なる場所に移動させる

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

メモ: *ここで説明されているAdobe Flash アプリケーション ファイル サポートは、使用するビルボード スタイルが全画面背景ウィンドウ表示ビルボードの場合 (***PlayMMedia** *関数を使用します) に適用します。ウィンドウ表示スタイルで進行状況を含む場合 (STATUSBBRD 定数を* **Enable** *関数と共に使用します) には適用しません。これらの 2 つのスタイルのビルボードについての詳細は、「InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルおよびファイルの種類」を参照してください。*

ビルボードはデフォルトでメイン インストール ウィンドウの中央に表示されます。別の場所を指定するには、BILLBOARD オプションと一緒に **PlaceWindow** 関数を呼び出します。たとえば画面左上隅から 10 ピクセルの所にビルボードを配置するには、次のように呼び出します。

```
PlaceWindow (BILLBOARD, 10, 10, UPPER_LEFT);
```

ビルボードはファイル転送中にのみ表示されるため、ファイル転送を開始する前に **PlaceWindow** を呼び出すようにしてください。

InstallScript または InstallScript MSI プロジェクトでビルボードの順序を設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*



タスク ***インストール プロジェクトからビルボードを削除するには次の操作を実行します。***

1. ビュー リストの **[動作とロジック]** の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. **[サポートファイル]** エクスプローラーで、削除するビルボード（**言語非依存**、または言語固有のどちらでも可）を含む **[ビルボード]** 項目をクリックします。右側に [**ファイル**] ペインが表示されます。
3. ビルボード ファイルを右クリックして、**[削除]** をクリックします。

InstallScript および InstallScript MSI インストールで背景ウィンドウを表示する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

デフォルトで、インストールは背景ウィンドウを表示しません。**PlayMMedia** 関数を使って Adobe Flash アプリケーション ファイル（.swf）または AVI ファイルを表示するためには、インストールが背景ウィンドウを表示する必要があります。また、1 つの種類のビルボードに対してのみ背景ウィンドウの表示が必要な場合もあります。



ヒント: *Flash ファイル ビルボードおよびイメージ ビルボードは、背景ウィンドウなしで表示できます。詳細については、「InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルおよびファイルの種類」を参照してください。*

背景ウィンドウを表示する方法は、使用するプロジェクトの種類によって異なります。

InstallScript プロジェクト

InstallScript プロジェクトで背景ウィンドウを表示するには、次のコード行の初めについている 2 つのスラッシュ (//) を削除して、スクリプトの **OnShowUI** イベント ハンドラー関数を修正します:

```
//if ( LoadStringFromStringTable( "TITLE_MAIN", szTitle ) < ISERR_SUCCESS ) then // タイトル文字列のロード
//    szTitle = IFX_SETUP_TITLE;
//endif;
//SetTitle( szTitle, 24, WHITE );
//Enable( FULLWINDOWMODE );
// Enable( BACKGROUND );
//SetColor( BACKGROUND, RGB( 0, 128, 128 ) );
```

InstallScript MSI プロジェクト

InstallScript プロジェクトで背景ウィンドウを表示するには、次のコード行の初めについている 2 つのスラッシュ (//) を削除して、適切なスクリプトの UI イベント ハンドラー (たとえば、**OnFirstUIBefore** または **OnMaintUIBefore**) を修正します:

```
// SetTitle( @PRODUCT_NAME, 24, WHITE );
// SetTitle( @PRODUCT_NAME, 0, BACKGROUNDCAPTION );
// Enable( FULLWINDOWMODE );
// Enable( BACKGROUND );
// SetColor(BACKGROUND,RGB (0, 128, 128));
```

実行時にリスト ボックスへデータを挿入する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

基本の MSI プロジェクト

実行時にリストボックスのアイテムを設定するには、SQL クエリを使って .msi データベース用の一時レコードを作成する必要があります。具体例については、Revenera コミュニティの「Populating A List Box Dynamically At Run Time (実行時にリストボックスを動的に入力する)」を参照してください。

詳細は、Revenera コミュニティの「Accessing the MSI Database at Run Time (実行時にMSIデータベースにアクセスする)」を参照してください。

InstallScript MSI プロジェクト

CtrlSetList 関数は、ダイアログ上の ListBox コントロールに、文字列リスト変数を関連付けます。例:

```
function FillListBox( )
    LIST listDays;
begin
    listDays = ListCreate(STRINGLIST);
    ListAddString(listDays, "Monday", AFTER);
    ListAddString(listDays, "Wednesday", AFTER);
    ListAddString(listDays, "Friday", AFTER);
```

```
    CtrlSetList("DialogName", nListBoxId, listDays);
end;
```

ファイル参照ダイアログを表示する

GetOpenFileName API を呼び出して、ファイル参照ダイアログを表示する例については、ナレッジベース記事 Q104325 を参照してください。

InstallScript インストールでネットワーク参照ダイアログを表示する



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

SelectDirEx 関数を利用して、ユーザーがネットワークコンピューターを参照できるダイアログを表示することができます。

```
prototype NetBrowse( );

function NetBrowse( )
    STRING svSelectedDir[MAX_PATH + 1];
    NUMBER nReturn;
begin
    svSelectedDir = PROGRAMFILES;

    nReturn = SelectDirEx("", "Select a directory:", "", "",
        BIF_EDITBOX | BIF_RETURNONLYFSDIRS, svSelectedDir);

    if (nReturn = OK) then
        MessageBox("選択したディレクトリは: " + svSelectedDir, INFORMATION);
    elseif (nReturn = CANCEL) then
        MessageBox("ユーザーがキャンセルをクリック", INFORMATION);
    else
        MessageBox("ダイアログ表示エラー", WARNING);
    endif;
end;
```

サーバーの構成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

インストールを作成しているとき、ターゲット システムにインストールされるテクノロジに対してサーバー側のサポートを提供する必要があることに気がつくことがあります。InstallShield では、サーバー側のインストールまたは COM+ サーバー アプリケーションのアプリケーション プロキシ管理を簡単に構成することができます。InstallShield では、インターネット インフォメーション サービス、SQL、およびコンポーネント サービスがサポートされています。

このセクションには、次のトピックが含まれています:

- SQL サポートの構成
- スイート/アドバンスト UI プロジェクトで、SQLLogin 定義済みウィザード ページを追加する
- COM+ アプリケーションとコンポーネントの管理
- インターネット インフォメーション サービス

SQL サポートの構成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*



メモ: *スイート/アドバンスト UI プロジェクトにおける SQL サポートの構成についての詳細は、「スイート/アドバンスト UI プロジェクトで、SQLLogin 定義済みウィザード ページを追加する」を参照してください。*

InstallShield では、Microsoft SQL Server、Microsoft Windows Azure、MySQL、および Oracle のサポートが提供されています。[SQL スクリプト] ビューは、ユーザー インターフェイス内ですべての SQL スクリプトをサーバー接続および設定によって管理並びに編成するための拠点です。InstallShield 内の SQL サポートにより、次の処理が可能です。

- SQL サーバーへ接続する。
- カタログスキーマおよび/またはデータをインポートする。

- SQL スクリプトとコンポーネントを関連付ける。
- 必要な SQL サーバー/スクリプトプロパティ（サーバー名、データベース名、認証メソッドなど）を設定する。
- インストールまたはアンインストール中に実行するための SQL スクリプトの設定する。
- SQL スクリプトを編集する。
- SQL Server、MySQL、または Oracle の特定バージョンを要求またはターゲットする、もしくはその両方。
- SQL スクリプトのテキスト置換を定義する。
- Microsoft SQL Server Management Studio または Microsoft SQL Server Query Analyzer でスクリプトを開く。



メモ: *データベースのインポート機能は、Microsoft SQL Server Database で利用できます。Oracle ユーザーの方は、Oracle データベース ユーティリティの Oracle Web ページで、InstallShield と共に利用可能なユーティリティについての情報をご覧いただけます。*

使用中のシステムに Microsoft SQL Server Management Studio または Microsoft SQL Server Query Analyzer がインストールされている場合、プロジェクトに追加した新しい SQL スクリプトを開いて、スクリプトをテスト、編集、または構文チェックを行うことができます。これらのツールの 1 つを起動して InstallShield 内でスクリプトを開くには、[SQL スクリプト] ビューでスクリプトを右クリックしてから [Microsoft SQL Server Management Studio でスクリプトを開く] をクリックします。InstallShield が次のツールから 1 つを検索して、最初に検出されたツールを起動します:

1. *Microsoft SQL Server 2008 Management Studio (Express;* ssms.exe *を含む任意のエディション)*
2. *Microsoft SQL Server 2005 Management Studio* (SqlWb.exe)
3. *Microsoft SQL Server 2005 Management Studio Express* (ssmsee.exe)
4. *Microsoft SQL Server 2000 Query Analyzer* (isqlw.exe)

SQL ログインの設定に Windows Installer プロパティを使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

[SQL スクリプト] ビューを使って新しい SQL データベース接続をプロジェクトに追加すると、InstallShield はデフォルトで次の Windows Installer プロパティをプロジェクトに追加します。

テーブル 1 ・ デフォルト SQL ログインのプロパティ

プロパティ	説明
IS_SQLSERVER_AUTHENTICATION	このプロパティは、指定されたカタログの接続に使用する認証の種類を識別します。デフォルトのプロパティは IS_SQLSERVER_AUTHENTICATION です。プロパティ値には、次の数値が有効です: • 0 – 現在のユーザーの Windows 認証情報 • 1 – サーバー認証 このプロパティは、選択された SQL 接続の [詳細] タブにある "認証タイプのプロパティ名" 設定のデフォルト値です。
IS_SQLSERVER_DATABASE	このプロパティは、インストール中にそのプログラムへの接続を作成する SQL カタログの名前を識別します。このプロパティは、選択された SQL 接続の [詳細] タブにある "ターゲット カタログのプロパティ名" 設定のデフォルト値です。
IS_SQLSERVER_PASSWORD	このプロパティは、サーバー認証に使用するパスワードを識別します。このプロパティは、選択された SQL 接続の [詳細] タブにある "サーバー認証パスワードのプロパティ名" 設定のデフォルト値です。
IS_SQLSERVER_SERVER	このプロパティは、(Microsoft SQL Server および MySQL の場合) ターゲット サーバーのインスタンス名、または (Oracle の場合) URL 接続文字列またはローカル ネット サービス名を識別します。このプロパティは、選択された SQL 接続の [詳細] タブにある "ターゲット サーバーのプロパティ名" 設定のデフォルト値です。
IS_SQLSERVER_USERNAME	このプロパティは、サーバー認証に使用するログイン ID を識別します。このプロパティは、選択された SQL 接続の [詳細] タブにある "サーバー認証ログイン ID のプロパティ名" 設定のデフォルト値です。

既存の接続でこれらの Windows Installer プロパティの 1 つをオーバーライドする場合、プロパティ マネージャーで新しいプロパティを追加します。そのあと、[SQL スクリプト] ビューで、その接続を選択します。[詳細] タブで、該当の一覧から新しいプロパティの名前を選択します。

製品が後で使用できるように、ターゲットシステム上にこれらの任意のプロパティの値を保存したい場合、それが可能です。以下は、これらのプロパティの使用例です:

- [レジストリ] ビューで、データが **[IS_SQLSERVER_SERVER]** のレジストリ値を作成します。実行時に、インストールがレジストリ値を作成するとき、そのレジストリ値のデータは、SQL カタログの名前に設定されます。
- [テキスト ファイルの変更] ビューを使って、実行時に処理を行うテキスト文字列の置換を構成します。このビューで、製品と共にインストールされるファイルを説明するテキスト ファイル リファレンスを追加してから、検索/置換の基準を指定します。検索/置換の基準には、SQL Server マシンの名前の場所に **[IS_SQLSERVER_DATABASE]** と入力します。実行時、インストールがテキスト ファイルを編集するとき、SQL Server マシンの名前がテキストファイルに書き込まれます。



ヒント: *デフォルトの SQL ランタイム動作をオーバーライドするためにプロジェクトで定義づけることが可能な追加 Windows Installer プロパティのリストは、「デフォルトの SQL ランタイム動作をオーバーライドする」を参照してください。*



プロジェクト: *基本の MSI インストールでは、エンド ユーザーはビルトイン SQLLogin ダイアログを使って、前述のプロパティを構成できます。基本の MSI プロジェクトで SQL 接続の [詳細] タブにある SQL プロパティのいずれかを変更した場合に、[ダイアログ] ビューの SQLLogin ダイアログ内にある対応するプロパティが自動的に更新されることはありません。従って、ダイアログのプロパティを手動で変更して、SQL 接続の [詳細] タブで選択したプロパティと一致させる必要があります。*

新しい SQL 接続が同じ Windows Installer のプロパティを共有するかどうかを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

InstallShield では、デフォルトで、プロジェクトに追加した新しい SQL データベース接続が、同じ Windows Installer のプロパティを共有するかどうかを指定することができます。

たとえば、すべての接続で同じ Windows Installer のデフォルト プロパティを共有させたいとき、[SQL スクリプト] ビューで接続の 1 つをプロジェクトに追加し、**MyConnection** のカタログ名を指定することができます。2 つ目の接続をプロジェクトに追加したとき、その同じ **MyConnection** のカタログ名が 2 つ目の接続に使用されます。いずれか一方のカタログ名を変更すると、両方とも同じ Windows Installer プロパティに基づいているため、もう一方の接続のカタログ名も自動的に更新されます。



タスク ***すべての SQL 接続が同じ Windows Installer のプロパティを共有するかどうかを指定するには、以下の手順に従います。***

1. **[ツール]** メニューから **[オプション]** を選択します。**Options** ダイアログ ボックスが開きます。
2. **[SQL スクリプト]** タブをクリックします。
3. **[新しい接続に一意の Windows Installer プロパティを生成する]** チェック ボックスを目的に従って選択またはクリアします:
   - 追加したすべての新しい接続で Windows Installer プロパティを共有する場合、このチェック ボックスをクリアします。
   - 追加した新しい接続で異なる Windows Installer プロパティを使用する場合、このチェック ボックスを選択します。

チェック ボックスが選択されている状態で 2 つ目の接続をプロジェクトに追加すると、その 2 つ目の接続に新しい Windows Installer プロパティ セットが作成されます。

チェック ボックスがクリアされている状態で 2 つ目の接続を追加すると、その 2 つ目の接続にデフォルトの Windows Installer プロパティが使用されます。Windows Installer のデフォルトのプロパティは、プロジェクトに追加した 1 つ目の接続用に追加されたプロパティです。



ヒント: *既存の接続の Windows Installer プロパティをオーバーライドする場合、プロパティ マネージャーで新しいプロパティを追加します。そのあと、[SQL スクリプト] ビューで、その接続を選択します。[詳細] タブで、該当の一覧から新しいプロパティの名前を選択します。*



プロジェクト: *基本の MSI プロジェクトの ビルトイン SQLLogin ダイアログは、プロジェクトに追加された 1 つ目の SQL 接続用の Windows Installer のデフォルト プロパティと共に動作するようにデザインされています。**[新しい接続に一意の Windows Installer プロパティを生成する]** オプションを選択した場合、新しい Windows Installer のプロパティと動作するように、各接続ごとにダイアログを複製する必要があります。*

InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallScript 言語には、*SQLRT* というプレフィックスで始まる多数のビルトイン SQL ランタイム (SQLRT) 関数が含まれています。一部の関数は InstallScript プロジェクトでのみ、または InstallScript MSI プロジェクトでのみ提供されています。また、両方のプロジェクトで提供されている関数もあります。

SQLRT 関数は、SQL 情報を [SQL スクリプト] ビューで構成する必要があります。この構成情報は、SQL ランタイム関数が適切に動作するように、SQLRT.ini ファイルに書き込まれます。InstallScript プロジェクトの場合、SQLRT 関数は SQLRT.obl ファイルに存在し、SQLRT.dll ファイルを呼び出します。InstallScript MSI プロジェクトの場合、SQLRT 関数は SQLCONV.obl ファイルに存在し、ISSQLSRV.dll ファイルを呼び出します。これらのサポート ファイルは、[SQL スクリプト] ビューを使用したとき、自動的にプロジェクトに追加されます。

SQLRTInitialize2 関数は SQL サーバー ランタイムを初期化します。InstallScript プロジェクトの場合、**SQLRTInitialize2** 関数は、OnSQLServerInitialize イベント ハンドラーの実行中、自動的に呼び出されます。InstallScript MSI プロジェクトの場合、**SQLRTInitialize2** 関数は、OnSQLLogin イベント ハンドラーの実行中、自動的に呼び出されます。SQLRT 関数の 1 つを OnSQLServerInitialize または OnSQLLogin イベントの前に呼び出す必要がある場合、**SQLRTInitialize2** 関数をまず呼び出す必要があります。



メモ: *InstallShield の以前のバージョンでは、**SQLRTInitialize** 関数は InstallScript プロジェクトで、SQL サーバー ランタイムを初期化するために使用されていました。この関数は、**SQLRTInitialize2** 関数によって置き換えられました。*

新しい SQL 接続の追加



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

SQL スクリプト ビューでは、スクリプトは接続によって編成されます。これは接続が確立されるまでスクリプトがサーバー上で実行することができないためです。したがって、プロジェクトに SQL Script を追加する前に、まず SQL 接続を作成する必要があります。



タスク

新しい SQL 接続を作成するには、以下の手順を実行します:

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。

エクスプローラーに、新しい接続が追加されます。右のペインにあるタブを利用して、この接続に関連付けられている設定を構成します。



メモ: *SQLLogin と SQLBrowse ダイアログを通して、エンド ユーザーはエイリアス名を使って SQL Server データベースに接続および参照できます。*

Microsoft から Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) と呼ばれる新しい OLE DEB プロバイダーがリリースされました。今後、新しいプロバイダーは最新のサーバー機能で更新されます。このプロバイダーは、TLS 1.2 のみの環境もサポートします。

InstallShield 2019 R3 以降は、すべての新しい SQL Server 接続にデフォルトとして MSOLEDBSQL を使用します。これは、[要件] タブで [データベース サーバーのドライバーを選択します] ドロップダウンから異なる SQL ドライバーを選択して変更することができます。



重要: *Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) は、InstallShield 2019 R3 以降に前提条件として含まれています。*

詳細については、[要件] タブ の「データベース サーバーの SQL ドライバーを選択する」を参照してください。

SQL Server データベースのデフォルトの TCP/IP ネットワーク ライブラリを別のプロトコルで上書きする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript*
- *InstallScript MSI*

デフォルトで、InstallShield インストールは SQL Server データベースへの接続時に、TCP/IP ネットワーク ライブラリを使用します。別のプロトコルを使用する場合、必要に応じてこのデフォルト動作をオーバーライドすることができます。



タスク ***SQL Server データベースへの接続時に、デフォルトの TCP/IP ネットワーク ライブラリを上書きするには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。エクスプローラーに、新しい接続が追加されます。右のペインにあるタブを利用して、この接続に関連付けられている設定を構成します。
3. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
4. **[テーブル]** エクスプローラーで、Property テーブルをクリックします。
5. IS_SQLSERVER_NETLIB_MS フィールドを見つけます。デフォルトの値は次のとおりです。

   Network Library=DBMSSOCN

   DBMSSOCN は、TCP/IP ネットワーク ライブラリのモジュール名 (拡張子を除く) を示します。
6. 必要に応じて、デフォルトの DBMSSOCN 名を置き換えます。例:
   - Named Pipes ネットワーク ライブラリを使用するには、**DBNMPNTW** を指定します。
   - SPX/IPX を使用するには、**DBMSSPXN** を指定します。
   - Banyan Vines を使用するには、**DBMSVINN** を指定します。
   - Multi-Protocol (Windows RPC) を使用するには、**DBMSRPCN** を指定します。

実行時に、SQL Server データベースに接続する際、指定されたプロトコルが使用されます。

SQL Server Express LocalDB のインスタンスに接続するときの要件



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

インストールで、SQL Server Express LocalDB のインスタンスへの接続をサポートする場合、ターゲット システム上に、SQL Server Native Client 11 ODBC ドライバーが存在している必要があります。当該のドライバーが存在していることを確認するために、インストールに Microsoft SQL Server 2012 Native Client 前提条件を含めることができます。このドライバーを使用するには、プロジェクトの構成も必要になります。

タスク ***SQL Server Express LocalDB のインスタンスへの接続をサポートするために、プロジェクトに適切な SQL Server Native Client 前提条件を含めるには、以下の手順に従います:***

1. 基本の MSI および InstallScript MSI プロジェクトの場合、**[アプリケーション データ]** の下にあるビュー リストで、**[再配布可能ファイル]** をクリックします。

   InstallScript プロジェクトの場合: **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。

2. 再配布可能ファイルのリストで、[**Microsoft SQL Server 2012 Native Client**] チェック ボックスを選択します。

InstallShield 前提条件ファイル (.prq) で定義された条件が満たされた場合のみ、InstallShield 前提条件が起動します。条件の一覧を参照するには、[再配布可能ファイル] ビューまたは [前提条件] ビューにある SQL Server Native Client 前提条件をクリックしてから、右側の詳細ペインに表示される説明を確認します。これらのビューで [詳細を表示] ボタンをクリックして、詳細の表示/非表示を切り替えることができます。

タスク ***InstallShield インストール プロジェクトを構成して、インストールで、SQL Server Native Client 11 ODBC ドライバーが使用されるようにするには、以下の手順に従います::***

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーで、ISSQLDBMetaData テーブルをクリックします。
3. **[ISSQLDBMetaData]** 列で、まず **[MSSQLServer]** 行を見つけます。
4. **[AdoDriverName]** 列の値を、次の値で置き換えます:

   sqlncli11

実行時、SQL Server Express LocalDB データベースへに接続する際、SQL Server Native Client 11 ODBC ドライバーが使用されます。

新しい SQL スクリプトの追加

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

タスク ***新しい SQL 接続の作成後、新しい SQL スクリプトを追加するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. 新しい接続を右クリックしてから、**[新しいスクリプト]** をクリックします。

新しいスクリプトをプロジェクトに追加すると、新しいコンポーネントがそのスクリプトに追加されます。このスクリプトを機能に関連付けます。機能が存在しない場合、SQL スクリプトを追加したときに [新しい機能の作成] ダイアログ ボックスが表示され、新しい機能を作成するように要求してきます。スクリプトと機能間の関連付けは、[SQL スクリプト] ビューにある [全般] タブの **[SQL スクリプトが属する機能の選択]** 領域を利用して後で変更することができます。



ヒント: *また、それらをインポートまたは挿入することで、プロジェクトにスクリプトを追加することもできます。詳細については、「SQL スクリプトの挿入およびインポート」を参照してください。*



メモ: *使用中のシステムに Microsoft SQL Server Management Studio または Microsoft SQL Server Query Analyzer がインストールされている場合、プロジェクトに追加した新しい SQL スクリプトを開いて、スクリプトをテスト、編集、または構文チェックを行うことができます。これらのツールの 1 つを起動して InstallShield 内でスクリプトを開くには、[SQL スクリプト] ビューでスクリプトを右クリックしてから [Microsoft SQL Server Management Studio でスクリプトを開く] をクリックします。InstallShield が次のツールから 1 つを検索して、最初に検出されたツールを起動します:*

- *Microsoft SQL Server 2008 Management Studio (Express;* ssms.exe *を含む任意のエディション)*
- *Microsoft SQL Server 2005 Management Studio* (SqlWb.exe)
- *Microsoft SQL Server 2005 Management Studio Express* (ssmsee.exe)
- *Microsoft SQL Server 2000 Query Analyzer* (isqlw.exe)

SQL スクリプトの挿入およびインポート



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

InstallShield では、複数のプロジェクトで SQL スクリプト ファイル (.sql) を再利用することができます。[SQL スクリプト] ビューで、プロジェクトにスクリプト ファイルを挿入したりインポートしたりすることができます。

- スクリプト ファイルを挿入すると、ファイルの現在の保存場所へのリンクが作成されます。
- スクリプト ファイルをインポートすると、プロジェクトのスクリプト ファイルを格納しているフォルダーにそのファイルがコピーされます。インポートしたスクリプト ファイルは、システム上またはリポジトリに保存することができます。

SQL スクリプト ファイルの挿入

タスク ***SQL スクリプト ファイルを挿入するには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. SQL 接続をまだ追加していない場合、**[SQL スクリプト]** エクスプローラーで、SQL 接続の追加を行います。
3. SQL 接続を右クリックしてから、**[スクリプト ファイルの挿入]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. 挿入する SQL スクリプト ファイル (.sql) を選択します。
5. **[開く]** をクリックします。

SQL スクリプト ファイルのインポート

タスク ***SQL スクリプト ファイルをインポートするには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. SQL 接続をまだ追加していない場合、**[SQL スクリプト]** エクスプローラーで、SQL 接続の追加を行います。
3. SQL 接続を右クリックしてから、**[スクリプト ファイルのインポート]** をクリックします。[SQL スクリプト ファイルのインポート] ダイアログ ボックス が開きます。
4. 以下のいずれかを実行します。
   - **[レポジトリ アイテム]** ボックスで、プロジェクトに追加する SQL スクリプト ファイル (.sql) をクリックします。
   - インポートするスクリプト ファイルがリポジトリに格納されていない場合、[参照] ボタンをクリックしてファイルを選択します。
5. **[OK]** をクリックします。

新しいスクリプトをプロジェクトに挿入またはインポートすると、新しいコンポーネントがそのスクリプトに追加されます。このスクリプトを機能に関連付けます。機能が存在しない場合、SQL スクリプトを追加したときに [新しい機能の作成] ダイアログ ボックスが表示され、新しい機能を作成するように要求してきます。スクリプトと機能間の関連付けは、[SQL スクリプト] ビューにある [全般] タブの **[SQL スクリプトが属する機能の選択]** 領域を利用して後で変更することができます。

SQL Server データベースのインポートと SQL スクリプト ファイルの生成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*

- *InstallScript MSI*



タスク ***既存の Microsoft SQL Server データベースをインポートして、それからスクリプト ファイルを生成するには、以下の手順を実行します。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーを右クリックしてから、**[データベース インポート ウィザード]** をクリックします。**データベース インポート ウィザード**が開きます。
3. データベース インポート ウィザードのパネルを完成します。

データベース インポート ウィザードが起動してデータベース設定のインポート手順を案内し、これらの設定および選択されたオプションに基づいて SQL スクリプト ファイルを生成します。



メモ: *データベースのインポート機能は、Microsoft SQL Server Database で利用できます。データベース インポート ウィザードで生成されたスクリプトは、他の種類のデータベースとは互換性を持ちません。*

SQL スクリプト ファイルの編集



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

スクリプト ファイルは、作成から挿入、インポートまですべて終えた時点で、InstallShield 内で編集することができます。



プロジェクト: *InstallScript プロジェクトで、InstallShield にアップグレードする前に、OnFirstUIBefore をオーバーライドしていたスクリプトを使って作業する必要があり、そのスクリプトが OnSQLServerInitialize を呼び出さない場合、そのコードをスクリプト ファイルに追加する必要があります。*



タスク ***InstallShield でスクリプト ファイルを編集するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、編集するスクリプト ファイルをクリックします。
3. **[スクリプト]** タブをクリックすると、スクリプト ファイルのコンテンツが表示されます。
4. 必要に応じてファイルを編集します。

プロジェクトをビルドするたびにスクリプトを再作成する場合、[SQL スクリプト] ビュにあるスクリプトのための [データベースのインポート] タブで [ビルド時に SQL スクリプトを再作成する] オプションを選択できます。既存のスクリプト ファイルのバックアップは必ず最初にとるようにしてください。



メモ: *使用中のシステムに Microsoft SQL Server Management Studio または Microsoft SQL Server Query Analyzer がインストールされている場合、プロジェクトに追加した新しい SQL スクリプトを開いて、スクリプトをテスト、編集、または構文チェックを行うことができます。これらのツールの 1 つを起動して InstallShield 内でスクリプトを開くには、[SQL スクリプト] ビューでスクリプトを右クリックしてから [Microsoft SQL Server Management Studio でスクリプトを開く] をクリックします。InstallShield が次のツールから 1 つを検索して、最初に検出されたツールを起動します:*

- *Microsoft SQL Server 2008 Management Studio (Express;* ssms.exe *を含む任意のエディション)*
- *Microsoft SQL Server 2005 Management Studio* (SqlWb.exe)
- *Microsoft SQL Server 2005 Management Studio Express* (ssmsee.exe)
- *Microsoft SQL Server 2000 Query Analyzer* (isqlw.exe)

SQL スクリプト ファイルのバージョン番号を指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

[SQL スクリプト] ビューでスクリプトの作成またはインポートを行ったあと、SQL スクリプト ファイルへスキーマ バージョン番号を割り当てることができます。スキーマ バージョン情報を指定すると、適切なときにだけ SQL スクリプトの起動をトリガーするのに便利です。



タスク ***SQL スクリプト ファイルのスキーマ バージョンを指定するには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、構成する SQL スクリプトをクリックします。
3. **[全般]** タブをクリックします。
4. **[スキーマ バージョン (xxxxx.xxxxx.xxxxx.xxxxx)** ボックスで、SQL スクリプトのスキーマ バージョン番号を指定します。

インストールは、ターゲット データベースにある現在のスキーマ バージョンを確認します。スキーマ バージョンは、InstallShield というカスタム テーブルの ISSchema 列に格納されます。SQL スクリプトへスキーマ バージョン番号を指定すると、インストールは、スクリプト スキーマ バージョン番号が現在のスキーマ バージョン番号より大きいときのみ、スクリプトを実行します。一旦スクリプトが実行されると、インストールはターゲット データベースにある現在のスキーマ バージョンを更新して、新しいスキーマ バージョン番号を反映します。

SQL スクリプトのスキーマ バージョン (xxxx.xxxxx.xxxx) を指定しなかった場合、スクリプトは常に起動されます。

たとえば、インストールで、新しい接続が **TestDB** という名前のデータベースに対して作成されてから **TestScript** と呼ばれるスクリプトが作成され、そのスクリプトが **12345.54321.12345.54321** というスキーマ バージョン番号を持つとします。**TestDB** データベースに、カスタム InstallShield テーブルが作成され、そのテーブルの InstallShield 列にスキーマ バージョンが格納されます。同じシステムで別のインストールが実行され、かつこのインストールにも **TestScript** という名前の SQL スクリプトがある場合、この SQL スクリプトは、もう 1 つのスクリプトのスキーマ バージョン番号がターゲット データベースの InstallShieldテーブルにある ISSchema 列に格納されているバージョン番号よりも大きいときのみ実行されます。スクリプトが実行されると、インストールは ISSchema 列を新しいスキーマ バージョン番号で更新します。

ヒント: *"スキーマ バージョン" 設定で番号を指定して、InstallShield によって、ターゲット データベースにそのスキーマ バージョン番号を格納するための InstallShield テーブルが追加されると、データは、アンインストールが実行されたときも自動的に削除されません。従って、インストールが変更内容をロールバックできるようにするためには、アンインストール時にテーブルを削除するカスタム スクリプトを作成して、InstallShield テーブルを削除する必要があります。*

接続に関連付けられている複数 SQL スクリプトの実行順序を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

プロジェクトで 1 つ以上の SQL スクリプトを SQL 接続に追加する場合、インストールがターゲット マシンでこれらの SQL スクリプトを実行する順序を指定できます。順序を指定するための手順は、Windows Installer ベースのプロジェクトと InstallScript ベースのプロジェクトで異なります。

基本の MSI プロジェクト、DIM、InstallScript MSI プロジェクトで SQL スクリプトの順序を指定する

基本の MSI、DIM、および InstallScript MSI プロジェクトの [SQL スクリプト] ビューで、接続の SQL スクリプトが一覧表示される順序は、SQL スクリプトが実行される順序です。たとえば、[SQL スクリプト] ビューで接続を作成し、その接続に 1 つ以上の SQL スクリプト追加した場合、SQL 接続のすぐ下に表示されているスクリプトが、インストールで一番最初に実行されます。

タスク ***接続の SQL スクリプトが実行される順番を変更するには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、移動する SQL スクリプト右クリックし、**[上に移動]** または **[下に移動]** をクリックします。

順序は、次回インストールをビルドして実行したとき、更新されます。

InstallScript プロジェクトで SQL スクリプトの順序を指定する

InstallScript インストールでは、デフォルトで、SQL スクリプトのバッチ モードが無効になっています。したがって、接続の SQL スクリプトが実行される順序は、インストールが関連付けられているコンポーネントを処理する順番に対応します。

このデフォルト動作をオーバーライドして、SQL スクリプトを実行する順序を再指定できるようにするには、[SQL スクリプト] ビューでバッチ モードを有効にします。



タスク

バッチ モードを有効または無効にするには、以下の手順に従います。

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーを右クリックして、**[バッチ モード]** をクリックします。

[バッチ モード] コマンドにチェック マークが付いている場合、バッチ モードは有効になっています。チェック マークが付いてない場合、バッチ モードは無効です。

SQLRTGetBatchList 関数は、バッチ モードが有効になっているとき実行する必要がある SQL スクリプトに関連付けられているコンポーネントの一覧を返します。詳細については、「SQLRTGetBatchList」を参照してください。

デフォルトの SQL ランタイム動作をオーバーライドする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

次の Windows Installer のプロパティを定義して、デフォルトのランタイム動作をオーバーライドすることができます:

テーブル 2 · SQL の Windows Installer プロパティ

プロパティ	説明
IS_SQLSERVER_CONNECTIONS_TO_VALIDATE	SQLLogin ダイアログの [次へ] ボタンをクリックしたときテストされる接続をオーバーライドします。複数の接続は、セミコロンで区切って指定します。デフォルトで、ISSQLConnection テーブルにあるすべての接続が検証されます。
IS_SQLSERVER_CXNS_ABSENT_FROM_INSTALL	インストールまたはアンインストール中にスキップする 1 つまたは複数の SQL 接続を指定します。1 つ以上の SQL 接続を指定するには、各接続をセミコロン (;) で区切ります。すべての SQL 接続をスキップするには、このプロパティの値を **ALL** に設定します。このプロパティは、SQL スクリプト作成エラーが原因で製品をアンインストールできないとき便利です。

テーブル 2 ・ SQL の Windows Installer プロパティ (続き.)

プロパティ	説明
IS_SQLSERVER_DO_NOT_USE_REG	レジストリに格納されている SQL Server ログイン情報を使用しないように指定します。SQLLogin ダイアログは、メンテナンスおよびアンインストール モードでは表示されないので、インストールはインストール中に指定されたログイン情報を格納します。この動作が必要ない場合は、IS_SQLSERVER_DO_NOT_USE_REG プロパティを設定してください。
IS_SQLSERVER_LOCAL_ONLY	SQL Server 参照コンボ ボックスおよびリスト ボックス コントロールでローカル SQL Server のみ表示するよう指定します。 ***メモ:*** *ローカル SQL Server とリモート SQL Server または SQL Server エイリアスのみを表示するには、*IS_SQLSERVER_LOCAL_ONLY *プロパティと、*IS_SQLSERVER_REMOTE_ONLY *プロパティまたは*IS_SQLSERVER_ALIAS_ONLY *プロパティのどちらかを設定します。*
IS_SQLSERVER_REMOTE_ONLY	SQL Server 参照コンボ ボックスおよびリスト ボックス コントロールでリモート SQL Server のみ表示するよう指定します。 ***メモ:*** *リモート SQL Server とローカル SQL Server または SQL Server エイリアスのみを表示するには、*IS_SQLSERVER_REMOTE_ONLY *プロパティと、*IS_SQLSERVER_LOCAL_ONLY *プロパティまたは*IS_SQLSERVER_ALIAS_ONLY *プロパティのどちらかを設定します。*
IS_SQLSERVER_ALIAS_ONLY	SQL Server 参照コンボ ボックスおよびリスト ボックス コントロールで SQL Server エイリアスのみ表示するよう指定します。 ***メモ:*** *リモート SQL Server エイリアスとローカル SQL Server または リモート SQL Servers のみを表示するには、*IS_SQLSERVER_ALIAS_ONLY *プロパティと、*IS_SQLSERVER_LOCAL_ONLY *プロパティまたは*IS_SQLSERVER_REMOTE_ONLY *プロパティのどちらかを設定します。*



プロジェクト: *基本の MSI プロジェクトの場合、すべての接続は SQLLogin ダイアログにリンクされています。複数の SQLLogin ダイアログを表示するには、[ダイアログ] ビューから SQL ダイアログをコピーし、デフォルトの SQL ダイアログに似せてその動作とイベントを変更します。必ず、新しいプロパティを作成して、[SQL スクリプト] ビューにある接続の [詳細] タブでそれらを設定するようにしてください。作成した SQL ダイアログのコピーを変更する場合、それらの新しいプロパティを使用できます。*

SQL ランタイム エラーの処理



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*



タスク ***インターフェイスで SQL スクリプトのエラー処理プロパティを設定するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、エラー処理を追加する SQL スクリプト ファイルをクリックします。
3. **[ランタイム]** タブをクリックします。
4. **[スクリプト エラー処理]** 領域で、一覧からオプションを 1 つ選択するか、**[カスタム]** ボタンをクリックして、スクリプトのデフォルト エラー処理をオーバーライドします。選択可能なオプションは以下のとおりです:
   - エラー時に、次のスクリプトへ移動する
   - エラー時に、次のステートメントへ移動する
   - エラー時に、インストールを中止する

SQL スクリプトにデータベース サーバー タイプの条件を設定



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

InstallShield では、Microsoft SQL Server および MySQL 両方をターゲットとし、かつ異なるデータベース サーバー テクノロジごとに複数の SQL スクリプトをもつ単一接続を作成することができます。ただし、接続は最初に検出されたデータベース サーバーのタイプに基づいて作成されます。したがって、スクリプトは、検出されたデータ

ベース サーバー タイプに対してのみ実行され、検出されなかったデータベース サーバー タイプ固有のスクリプトは失敗します。たとえば、ランタイムが Microsoft SQL Server を検出すると, Microsoft SQL Server に関連付けられたスクリプトが実行され、MySQL 固有のスクリプトは失敗します。

この動作の回避策として、インストールが検出されなかったデータベース サーバー タイプのスクリプトを実行したときに SQL スクリプト エラーが起こるように SQL に条件を設定することをお勧めします。条件を設定することで、スクリプト作成エラーにカスタム エラー処理を設定し、接続の次のスクリプトへ移動することができます。以下は、スクリプトにデータベース サーバー タイプの条件を設定するための手順です。



タスク ***Microsoft SQL Server をターゲットにするスクリプトにデータベース サーバー タイプの条件を設定するには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、条件を作成するスクリプト ファイルをクリックします。
3. **[スクリプト]** タブをクリックします。
4. スクリプトの始めに、次のステートメントを追加します。

   SELECT @@ROWCOUNT
5. **[ランタイム]** タブをクリックします。
6. **[スクリプト エラーの処理]** 領域で、**[カスタム]** ボタンをクリックします。**[カスタム エラー処理]** ダイアログ ボックスが開きます。
7. **[ここをクリックして新規の項目を追加します]** 行をクリックします。
8. **[エラー番号]** 列で、**1193** と入力します。これは、MySQL がスクリプトの始めで追加された指定のシステム変数を持っていなかったときに、MySQL より返されるエラー番号です。
9. **[動作]** 列で、**[エラー時に、次のスクリプトへ移動する]** をクリックします。
10. **[プロジェクト全体]** 列で、**[いいえ]** を選択します。
11. **[OK]** をクリックします。



タスク ***MySQL をターゲットにするスクリプトにデータベース サーバー タイプの条件を設定するには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、条件を作成するスクリプト ファイルをクリックします。
3. **[スクリプト]** タブをクリックします。
4. スクリプトの始めに、次のステートメントを追加します。

   SELECT @@table_cache
5. **[ランタイム]** タブをクリックします。
6. **[スクリプト エラーの処理]** 領域で、**[カスタム]** ボタンをクリックします。**[カスタム エラー処理]** ダイアログ ボックスが開きます。
7. **[ここをクリックして新規の項目を追加します]** 行をクリックします。

8. **[エラー番号]** 列で、**137** と入力します。これは、Microsoft SQL Server がスクリプトの始めで追加された指定のシステム変数を持っていなかったときに、Microsoft SQL Server より返されるエラー番号です。
9. **[動作]** 列で、**[エラー時に、次のスクリプトへ移動する]** をクリックします。
10. **[プロジェクト全体]** 列で、**[いいえ]** を選択します。
11. **[OK]** をクリックします。

Windows Installer ベースのインストールで SQL スクリプトを条件付きで起動する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

インストールで、SQL スクリプトを、ターゲット システムである条件が満たされた場合のみ起動するようにした方がよい場合があります。たとえば、作成した SQL スクリプトで、エンドユーザーが管理者権限を所持している必要があるとします。エンドユーザーが管理者権限を所持していない場合、この SQL スクリプトは起動されません。



タスク

SQL スクリプトの条件を作成するには、以下の手順に従います。

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、条件を作成するスクリプト ファイルをクリックします。
3. **[ランタイム]** タブをクリックします。
4. **[スクリプト条件]** 領域で、**[条件ステートメントを指定する]** チェック ボックスを選択します。
5. 省略記号ボタン (...) をクリックします。**[条件ビルダー]** ダイアログ ボックスが開きます。
6. 1 つまたは複数の条件を作成します。

SQL スクリプトはコンポーネントの状態に関連付けられています。[SQL スクリプト] ビューで設定された条件は、SQL スクリプトのコンポーネントの条件です。コンポーネントの条件がターゲット システムで満たされた場合、インストールで SQL スクリプトがインストールされます。インストールされる予定ではない SQL スクリプトがターゲット システムにインストールされる場合、インストールのログ ファイルを生成すると、SQL スクリプトが間違ってインストールされる理由を判別しやすくなります。



ヒント: *条件および条件の構文の詳細については、「条件ステートメントのビルド」および「条件ステートメントの構文」を参照してください。*

InstallScript インストールで SQL スクリプトを条件付きで起動する



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

インストールで、SQL スクリプトを、ターゲット システムである条件が満たされた場合のみ起動するようにした方がよい場合があります。

InstallShield は、1 セットのデフォルトのグローバル イベント ハンドラーを生成します。各イベント ハンドラーは InstallScript 言語でスクリプトされた関数です。次の SQL 関連のイベントは、InstallShield フレームワークによって自動的に呼び出されます。

- OnSQLServerInitialize
- OnSQLComponentInstalled

OnSQLServerInitialize は, OnFirstUIBefore より呼び出され, OnSQLComponentInstalled は、ファイル転送中に、インストールされた各コンポーネントに対して呼び出されます。



メモ: *InstallShield にアップグレードする前に OnFirstUIBefore をオーバーライドしていたスクリプトを使って作業する必要があり、そのスクリプトが OnSQLServerInitialize を呼び出さない場合、そのコードをスクリプト ファイルに追加する必要があります。*

スクリプトで、OnSQLServerInitialize と OnSQLComponentInstalled を変更し、異なる事柄についてチェックを行うことができます。たとえば、下のサンプル コードではユーザーが管理者特権を持つかどうかを確認をすることができます。

```
function OnSQLComponentInstalled(szComponent)
string sMessage;
string sData;
number nResult;
begin

    if( Is( USER_ADMINISTRATOR, sData ) ) then

        nResult = SQLRTComponentInstall( szComponent );

        if( nResult = SQL_ERROR_ABORT ) then

            sMessage = SdLoadString( IDS_IFX_SQL_ERROR_RUN_FAILED );
            MessageBox( sMessage, MB_OK );
            abort;

        endif;
    else
        //管理者権限がないユーザーです。スクリプトを実行しません。
    endif;
end;
```



メモ: *InstallShield インターフェイスで、スクリプトが失敗したときの動作を構成するには、[SQL スクリプト] エクスプローラーで [SQL スクリプト] をクリックして、[ランタイム] タブに移動します。[スクリプト エラー処理] セクションを使用して、次のオプションの中の一つを選択することができます。*

- *エラー時に、次のスクリプトへ移動する*
- *エラー時に、次のステートメントへ移動する*
- *エラー時に、インストールを中止する*

SQL スクリプトが完全 SQL サーバーに対してのみ実行されるように要求する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

SQL スクリプト サポートを含むインストールでは、エンドユーザーはデフォルトで、Microsoft SQL Server Desktop Engine (MSDE) および SQL Server Express Edition で SQL スクリプトを実行することができます。エンドユーザーが完全 SQL サーバーでのみ SQL スクリプトを実行できるようにする場合、[SQL スクリプト] ビューを使用して、インストールの任意のデータベース接続対してそれを構成することができます。



タスク

SQL スクリプト ファイルが MSDE または SQL Server Express Edition を持つターゲット システム上で実行されるのを防ぐには、次の手順を実行します。

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、構成する SQL 接続をクリックします。
3. **[要件]** タブで、**[Microsoft SQL Server Desktop Engine / SQL Server Express へのインストールを許可する]** チェック ボックスをクリアします。

Windows Installer プロパティを使って SQL スクリプトの文字列を動的に置換する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

基本の MSI、DIM、InstallScript MSI プロジェクトでは、Windows Installer のプロパティを使用して、実行時の SQL スクリプトのテキストの置換を指定することができます。これによって、エンド ユーザーがターゲット システム上で起動される SQL スクリプトで使用される情報を指定することができるように設定できます。Windows Installer は MsiFormatRecord を使って、実行時に SQL スクリプトのプロパティを解決します。

例

次の手順は、ユーザーがビルトイン SQL ログイン ダイアログ（基本の MSI インストールの SQLLogin ダイアログ、または InstallScript MSI インストールの SQLServerSelectLogin2 ダイアログ）で入力する情報を含むカスタム SQL スクリプトを使って、実行時にデータベースを作成する方法を説明します。Windows Installer プロパティが、データベース名とそのターゲット場所に使用されます。



タスク

エンド ユーザーが実行時に指定する情報を含む SQL スクリプトを使ってデータベースを作成するには、以下の手順に従います:

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。InstallShield 新しい SQL 接続が追加されます。
3. SQl 接続をクリックしてから、**[全般]** タブをクリックします。
4. **[存在しない時、カタログを作成する]** チェック ボックスをクリアします。
5. **[SQL スクリプト]** エクスプローラーで、新しい接続を右クリックして、**[新しいスクリプト]** をクリックします。新しい SQL スクリプトが SQL 接続に追加されます。
6. SQL スクリプトをクリックして、**[スクリプト]** タブをクリックします。
7. ［スクリプト］ペインで、次のように入力します:

```
if not exists(select name from master.dbo.sysdatabases where name = '%DBNAME%')
begin
CREATE DATABASE %DBNAME%
ON
( NAME = %DBNAME%_dat,
    FILENAME = '%DBPATH%%DBNAME%dat.mdf',
    SIZE = 10,
    MAXSIZE = 50,
    FILEGROWTH = 5 )
LOG ON
( NAME = %DBNAME%_log,
    FILENAME = '%DBPATH%%DBNAME%log.ldf',
    SIZE = 5MB,
    MAXSIZE = 25MB,
    FILEGROWTH = 5MB ) ;
end
GO
```

8. SQL スクリプトの 2 つのテキスト置換を構成します:
   a. **[テキスト置換]** タブをクリックしてから、**[追加]** ボタンをクリックします。**[検索/置換]** ダイアログ ボックスが開きます。
   b. **[検索文字列]** ボックスに次を入力します:

      %DBNAME%

   c. **[置換文字列]** ボックスに次を入力します:

      []S_SQLSERVER_DATABASE]

   d. **[OK]** をクリックします。**[検索/置換]** ダイアログ ボックスが閉じます。

e. **[追加]** ボタンをクリックします。**[検索/置換]** ダイアログ ボックスが開きます。

f. **[検索文字列]** ボックスに次を入力します:

%DBPATH%

g. **[置換文字列]** ボックスに次を入力します:

[INSTALLDIR]

h. **[OK]** をクリックします。**[検索/置換]** ダイアログ ボックスが閉じます。

9. **[ランタイム]** タブをクリックします。

10. **[インストール中にスクリプトを実行]** チェック ボックスをクリアして、**[ログイン中にスクリプトを実行]** チェック ボックスを選択します。

11. リリースをビルドします。

MsiFormatRecord (Windows Installer ヘルプ ライブラリ)

InstallScript テキスト置換を使って SQL スクリプトの文字列を動的に置換する



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript プロジェクトでは、テキスト置換文字列変数を使用して、実行時に、SQL スクリプトで使用されているテキストの置き換えを指定することができます。これによって、エンド ユーザーがターゲット システム上で起動される SQL スクリプトで使用される情報を指定することができるように設定できます。InstallScript ランタイム コードは **TextSubSubstitute** 関数を使用して、文字列変数を SQL スクリプト内の適切な値で置き換えます。

例

次の手順は、エンド ユーザーが **SQLServerSelectLogin2** ダイアログに入力する情報を含むカスタム SQL スクリプトを使って、実行時にデータベースを作成する方法を説明します。InstallScript テキスト置換が、データベース名とそのターゲット場所に使用されます。



タスク

エンド ユーザーが実行時に指定する情報を含む SQL スクリプトを使ってデータベースを作成するには、以下の手順に従います:

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。

2. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。InstallShield 新しい SQL 接続が追加されます。

3. SQl 接続をクリックしてから、**[全般]** タブをクリックします。

4. **[存在しない時、カタログを作成する]** チェック ボックスをクリアします。

5. **[SQL スクリプト]** エクスプローラーで、新しい接続を右クリックして、**[新しいスクリプト]** をクリックします。新しい SQL スクリプトが SQL 接続に追加されます。

6. SQL スクリプトをクリックして、**[スクリプト]** タブをクリックします。

7. ［スクリプト］ペインで、次のように入力します:

```
if not exists(select name from master.dbo.sysdatabases where name = '%DBNAME%')
begin
CREATE DATABASE %DBNAME%
ON
( NAME = %DBNAME%_dat,
    FILENAME = '%DBPATH%¥%DBNAME%dat.mdf',
    SIZE = 10,
    MAXSIZE = 50,
    FILEGROWTH = 5 )
LOG ON
( NAME = %DBNAME%_log,
    FILENAME = '%DBPATH%¥%DBNAME%log.ldf',
    SIZE = 5MB,
    MAXSIZE = 25MB,
    FILEGROWTH = 5MB ) ;
end
GO
```

8. SQL スクリプトの 2 つのテキスト置換を構成します:

   a. **[テキスト置換]** タブをクリックしてから、**[追加]** ボタンをクリックします。**[検索/置換]** ダイアログ ボックスが開きます。

   b. **[検索文字列]** ボックスに次を入力します:

      %DBNAME%

   c. **[置換文字列]** ボックスに次を入力します:

      <MYDATABASENAME>

   d. **[OK]** をクリックします。**[検索/置換]** ダイアログ ボックスが閉じます。

   e. **[追加]** ボタンをクリックします。**[検索/置換]** ダイアログ ボックスが開きます。

   f. **[検索文字列]** ボックスに次を入力します:

      %DBPATH%

   g. **[置換文字列]** ボックスに次を入力します:

      <TARGETDIR>

   h. **[OK]** をクリックします。**[検索/置換]** ダイアログ ボックスが閉じます。

9. **[ランタイム]** タブをクリックします。

10. **[インストール中にスクリプトを実行]** チェック ボックスをクリアして、**[ログイン中にスクリプトを実行]** チェック ボックスを選択します。

11. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。

12. Database Name コントロールを含むダイアログの OnSQLServerInitialize イベントでダイアログ コードを見つけ、InstallScript 関数 **TextSubSetValue** の呼び出しを追加します。たとえば、ユーザー名をエンド ユーザーが SQLServerSelectLogin2 ダイアログで指定する名前にする場合、次のコードに示されている **TextSubSetValue** の呼び出しを追加します:

```
// ダイアログ ボックスを表示する（接続名なし）
// ダイアログを入れ替える場合、コメントアウト解除
```

```
// nResult = SQLServerSelectLogin2( szConnection, szServer, szUser, szPassword, bWinLogin, szDB, FALSE, TRUE );

// ダイアログ ボックスを表示する (接続名を含む)
// ダイアログを入れ替える場合、コメントアウト
nResult = SQLServerSelectLogin2( szConnection, szServer, szUser, szPassword, bWinLogin, szDB, TRUE, TRUE );
TextSubSetValue ("<MYDATABASENAME>", szDB, FALSE);
```

13. リリースをビルドします。

Windows Installer ベースのインストールで SQL サーバー側インストールを要求する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

インストールが SQL Server マシン上でのみ実行されるように構成する方法として、レジストリ情報のシステム検索を実行し、その結果をプロパティに格納したあと、そのプロパティを設定する条件に使用するという方法があります。以下は、その手順です。



タスク

Windows Installer ベースのインストールが SQL サーバー マシン上でのみ実行されるように構成するには、以下の手順に従います。

1. **[動作とロジック]** の下のビュー リストで、**[システム検索]** をクリックします。
2. グリッドを右クリックして、**[追加]** をクリックします。**システム検索ウィザード**が開きます。
3. **[検索内容]** パネルで、**[レジストリ エントリ]** をクリックし、**[次へ]** をクリックします。
4. **[検索方法を指定してください]** パネルで、以下を行います。
   a. **[レジストリ ルート]** 一覧で、**HKEY_LOCAL_MACHINE** をクリックします。
   b. **[レジストリ キー]** ボックスで、次の文字列を入力します。

      Software¥Microsoft¥Microsoft SQL Server

   c. **[レジストリ値]** ボックスで、次の文字列を入力します。

      InstalledInstances

   d. **[次へ]** をクリックします。
5. **[値の処理方法]** パネルで、以下を行います。
   a. **[値を保存するプロパティ]** ボックスで、次を入力します。

      SQLSERVERFOUND

   b. **[追加のオプション]** 領域で、**[プロパティに値を保存し、インストール条件でこのプロパティを使用]** オプションを選択します。
   c. **[完了]** をクリックします。**条件ビルダー**が開きます。

6. 条件を確認し、レジストリ エントリがシステム上で見つからなかったとき、エンドユーザーに表示するメッセージを入力します。たとえば、次のようなメッセージを入力します。

   このマシン上に Microsoft SQL Server が見つかりませんでした。このインストールはサーバー マシン上でのみ実行されるよう設計されています。

7. **[OK]** をクリックします。

エントリが [システム検索] グリッドに追加されます。

InstallScript プロジェクトのサーバー側インストールを要求する



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript プロジェクトで、サーバーサイド インストールを強制する一つの方法として、インストール プロジェクトが特定のレジストリ キーと値を検索し、値が検索されたときのみそれ自身をインストールするという形でプロジェクトをインストールするという方法があります。InstallScript プロジェクトで、これをどのように行うかの例は、以下のサンプル コードを参照してください。

```
function OnBegin()
    string sKey, sValue, sData;
    string sMsg;
    number nType, nSize, nResult;
begin

    RegDBSetDefaultRoot ( HKEY_LOCAL_MACHINE );
    sKey = "Software¥¥Microsoft¥¥Microsoft SQL Server";
    sValue = "InstalledInstances";
    nResult = RegDBGetKeyValueEx( sKey, sValue, nType, sData, nSize );

    if( nResult < 0 ) then

        //SQL Server レジストリキーがありません
        sMsg = "このマシン上に Microsoft SQL Server が見つかりませんでした。¥n" +
            "このインストールはサーバー マシン上でのみ実行されるよう設計されています。";
        MessageBox( sMsg, SEVERE );
        abort;

    endif;

end;
```

SQL スクリプト ファイルをリポジトリにパブリッシュする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

別のプロジェクトで再利用、または他のユーザーと共有したい既存の SQL スクリプト ファイルがある場合、リポジトリにそれをパブリッシュすることができます。

タスク ***SQL スクリプト ファイルをリポジトリにパブリッシュするには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーでパブリッシュするスクリプト ファイルを右クリックしてから、**[パブリッシュ ウィザード]** をクリックします。**パブリッシュ ウィザード**が開きます。
3. パブリッシュ ウィザードのパネルを完成します。

リポジトリからプロジェクトへスクリプト ファイルをインポートした後、現在のスクリプト ファイルと既存のリポジトリ スクリプト ファイルとは関連性を持ちません。スクリプト ファイルを変更してリポジトリへ再パブリッシュしても、プロジェクト内のインポート済みスクリプト ファイルには影響しません。ただし、リポジトリからプロジェクトへスクリプト ファイルを再インポートすることができます。

MySQL ODBC ドライバーをインストールする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

インストールが実行される前に、MySQL ODBC ドライバーのインストールおよび起動を行う場合、MySQL Connector 3.51 用の InstallShield 前提条件をプロジェクトに含めます。

タスク ***インストールが実行される前に、MySQL ODBC ドライバーのインストールおよび起動を行うには、以下の手順に従います:***

1. 基本の MSI および InstallScript MSI プロジェクトの場合、**[アプリケーション データ]** の下にあるビュー リストで、**[再配布可能ファイル]** をクリックします。

   InstallScript プロジェクトの場合: **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. 再配布可能ファイルのリストで、[**MySQL Connector ODBC 3.51**] チェック ボックスを選択します。

ヒント: *MySQL Connector ODBC 3.51 セットアップ前提条件は、そのインストーラーをダウンロードして、InstallShield 前提条件をシステムに追加した場合のみ使用可能です。詳しい手順については、「MySQL Connector ODBC 前提条件を含める」を参照してください。*

アンインストール中に、SQL データベースを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*



メモ: *現在接続中のデータベースは削除することができません。*

SQL データベースをアンインストール中に削除する必要がある場合、SQL スクリプトを使ってこれを実現できます。以下は、プロジェクトと SQL スクリプトを構成して、Microsoft SQL Server データベースを削除する方法です。



タスク ***アンインストール中に、Microsoft SQL Serve データベースを削除するには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. SQL 接続を追加します。
3. **[全般]** タブにある **[カタログ名]** ボックスで、**Master** と入力します。

   Master は、すべての Microsoft SQL Server システムに存在するシステム データベースの名前です。現在接続されているデータベースは削除することができないため、Master データベースに接続して、それから接続されていない別のデータベースを削除することができます。
4. サーバーの認証情報を入力します。
5. 新しい Script ファイルを追加します。
6. 次を SQL スクリプト ファイルに追加します。

   DROP DATABASE *DatabaseName*
   Go

   DatabaseName は削除するデータベースの名前です。



ヒント: *上記に代わる別の手順として、SQL スクリプトでも同じ操作を実行することができます。これを実現するには、次のコードを SQL スクリプトに入力します。*

USE Master
DROP DATABASE DatabaseName
Go

DatabaseName は削除するデータベースの名前です。

Microsoft SQL Azure Database サーバーへの接続と SQL スクリプトの実行



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

ターゲット システムにおける Microsoft Windows Azure SQL Database Server に接続するための要件

Microsoft Windows SQL Azure SQL データベース サーバーへの接続には、インストールを実行するターゲット システム上で SQL Server Native Client 10.0 ODBC ドライバーが必要です。このドライバーがターゲット システムっ上に存在していることを確認するには、プロジェクトに Microsoft SQL Server 2012 Native Client 10.00.2531 の InstallShield 前提条件を含めます。



タスク

Microsoft SQL Server 2008 Native Client 10.00.2531 前提条件をプロジェクトに含めるには、以下の手順に従います:

1. 基本の MSI および InstallScript MSI プロジェクトの場合、**[アプリケーション データ]** の下にあるビュー リストで、**[再配布可能ファイル]** をクリックします。

   InstallScript プロジェクトの場合: **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。

2. 再配布可能ファイル リストで、ターゲット システムのアーキテクチャが 32 ビットか 64 ビットかに応じて、適切な **[Microsoft SQL Server 2008 Native Client 10.00.2531]** チェック ボックス (複数可) を選択します。

InstallShield 前提条件ファイル (.prq) で定義された条件が満たされた場合のみ、InstallShield 前提条件が起動します。条件の一覧を参照するには、[再配布可能ファイル] ビューまたは [前提条件] ビューにある SQL Server Native Client 前提条件をクリックしてから、右側の詳細ペインに表示される説明を確認します。これらのビューで [詳細を表示] ボタンをクリックして、詳細の表示/非表示を切り替えることができます。

実行時の Windows Azure SQL データベースの接続に使用するユーザー名を指定する

インストールで SQLLogin ダイアログを使用し、かつ SWindows Azure SQL データベースをターゲットする場合、エンドユーザーが SQLLogin ダイアログの [ログイン ID] ボックスで文字列を入力するとき、次のフォーマットの使用が必要になります:

DatabaseUserName@ServerName

参考例:

MyUserName@wbzdh64drd

Oracle のインスタンスに接続して SQL スクリプトを実行する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*



***メモ**- Oracle 11g InstaInt Client がターゲット システム上で必要な時にインストールがこれをダウンロードする場合、Oracle に問い合わせて、それが可能かどうかを確認してください。可能な場合、独自の Web サイトでダウンロードをホストして、InstallShield 前提条件にそのダウンロード パスを追加しなくてはなりません。Revenera では、このインストールのダウンロードを提供していません。*

Oracle のインスタンスに接続する場合、インストールを実行するエンドユーザーのマシン上に次の要素が必要です:

- Microsoft ODBC for Oracle
- x86 プロジェクトには、Oracle Instant Client ソフトウェアの 32 ビット バージョンが必要です (ターゲット システムが 64 ビットの場合も含む)
- x64 プロジェクトには、64 ビットのターゲット システム上で Oracle Instant Client ソフトウェアの 64 ビット バージョンが必要です。

最新版の MDAC (Microsoft Data Access Components) は、Microsoft ODBC for Oracle ドライバーをサポートします。ただし、Microsoft ODBC for Oracle ドライバーには、Oracle データベースサーバーと応答するために Oracle Instant Client が必要となります。このため、Oracle 11g Instant Client 前提条件を含めることで、インストール時にターゲット マシン上で Oracle Instant Client の構成を可能にできます。Oracle はファイルのインストールを提供しません。InstallShield Program Files フォルダーにある Oracle Instant Client インストール プロジェクトを使って、簡単にファイルをインストールすることができます。手順については、「Oracle Instant Client をダウンロードして、そのための .msi パッケージを作成する」を参照してください。

SQL スクリプトに Unicode 文字が含まれている場合、Unicode 文字を含む SQL スクリプトの実行をサポートする Oracle ODBC Instant Client を使用する必要があります。Microsoft ODBC for Oracle は、これをサポートしません。ODBC Instant Client インストールを Oracle 11g Instant Client 前提条件と共に含めて、ターゲット マシン上で Oracle Instant Client を構成し、また ODBC Instant Client のセットアップも行います。詳細については、「Oracle Basic Instant Client および Oracle ODBC Instant Client をダウンロードして、両方の .msi パッケージ および InstallShield 前提条件を作成する」を参照してください。Microsoft ODBC for Oracle の代わりに ODBC Instant Client を使用する場合は、プロジェクトを適切に構成するようにしてください。詳細については、「Oracle Database Server 上で Unicode 文字を含む SQL スクリプトを実行する」を参照してください。

Oracle Instant Client をダウンロードして、そのための .msi パッケージを作成する



***プロジェクト**- この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

Oracle 11g Instant Client サポートを基本の MSI、InstallScript、または InstallScript MSI プロジェクトに追加する前に、Oracle Instant Client を Oracle の Web サイトからダウンロードしてください。Oracle では、ファイルのインストールを提供していません。したがって、.msi パッケージは自分で作成する必要があります。.msi パッケージは、InstallShield Program Files フォルダーにある Oracle Instant Client インストール プロジェクトを利用することで、簡単に作成することができます。

.msi パッケージを作成した後、それを使用中のマシンの適切な場所に追加してから、この Oracle Instant Client .msi パッケージを使用する InstallShield 前提条件に追加できます。

メモ *SQL スクリプトに Unicode 文字が含まれている場合、Unicode 文字を含む SQL スクリプトの実行をサポートする Oracle ODBC Instant Client を使用する必要があります。Microsoft ODBC for Oracle は、これをサポートしません。ODBC Instant Client インストールを Oracle 11g Instant Client 前提条件と共に含めて、ターゲット マシン上で Oracle Instant Client を構成し、また ODBC Instant Client のセットアップも行います。詳細については、「Oracle Basic Instant Client および Oracle ODBC Instant Client をダウンロードして、両方の .msi パッケージ および InstallShield 前提条件を作成する」を参照してください。*

タスク ***Oracle Instant Client をダウンロードして、.msi パッケージを作成するには、以下の手順に従います。***

1. http://www.oracle.com/technetwork/database/features/instant-client/index-097480.html から、Oracle Instant Client バージョン 11.1.0.x (32 ビット Windows プラットフォーム用の基本パッケージ) をダウンロードします。ダウンロード ファイルは、instantclient-basic-win32-11.1.0.7.0.zip という名前です。

   重要 *• InstallShield における Oracle サポートには、ターゲット システムが 32 ビットまたは 64 ビット システムのいずれの場合においても、Oracle Instant Client の 32 ビット バージョンのインストールが必要です。*

2. ファイルを C ドライブのルートに展開します。ファイルを展開すると、すべてのファイルが次の場所に追加されます:

   C:¥instantclient_11_1

3. InstallShield で Oracle Instant Client インストール プロジェクトを開きます。プロジェクトのパスは、次のとおりです。

   InstallShield Program Files フォルダー¥Support¥Oracle Instant Client¥instantclient-win32-11_1.ism

   このプロジェクトは Oracle Instant Client のバージョン 11.1.0.7 のために Reveneraが作成したものですが、このプロジェクトを、より新しいバージョンの Oracle 11g Instant Client (例、11.2.01) に使用することも可能です。

4. Oracle 11g Instant Client のインストール中、**Oracle11g Instant Client 11.1.0.7** という名前が実行時に表示されます。この名前は、.msi ファイルにも使用されます。この名前を変更して、異なるバージョン番号 (例、11.2.0.1、11.2.0.x など) を反映させるには、以下の手順に従います:

   a. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。

   b. **"製品名"** 設定で、既存のテキストを必要に応じて編集します。

5. 11.1.0.7 以後にリリースされたバージョンを使用していて、依存関係が必要な場合、そのファイルまたはマージ モジュールをプロジェクトに追加します。

6. ツールバー上で、**[ビルド]** ボタンをクリックします。

Oracle 11g Instant Client を作成中の InstallShield プロジェクトに含めることができるように、Oracle 11g Instant Client .msi ファイルがビルドされ、次のディレクトリに追加されます。

InstallShield Program Files フォルダー¥SetupPrerequisites¥oracle



ヒント: *11.1.0.7 以後にリリースされたバージョンを使用している場合、InstallShield 前提条件エディターで Oracle 11g Instant Client 11.1.0.7 前提条件を開きます。前提条件の名前を変更して、適切なバージョン番号を反映させます。また、条件も更新して、適切なバージョン番号を反映させます。*

Oracle 11g Instant Client をインストールに追加する方法については、「Oracle 11g Instant Client をインストールする」を参照してください。

Oracle Basic Instant Client および Oracle ODBC Instant Client をダウンロードして、両方の .msi パッケージ および InstallShield 前提条件を作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

Oracle 11g Instant Client および Oracle ODBC Instant Client のサポートを基本の MSI、InstallScript、または InstallScript MSI プロジェクトに追加する前に、Oracle の Web サイトから Oracle Basic Instant Client および Oracle ODBC Instant Client をダウンロードする必要があります。Oracle では、ファイルのインストールを提供していません。したがって、.msi パッケージは自分で作成する必要があります。.msi パッケージは、InstallShield Program Files フォルダーにある Oracle Instant Client インストール プロジェクトを利用することで、簡単に作成することができます。

.msi パッケージを作成した後、それを使用中のマシンの適切な場所に追加してから、この Oracle .msi パッケージを使用する InstallShield 前提条件に追加できます。



タスク ***Basic Instant Client および ODCBC Instant Client をダウンロードして .msi パッケージを作成するには、以下の手順に従います:***

1. http://www.oracle.com/technetwork/database/features/instant-client/index-097480.html から、Oracle Instant Client バージョン 11.1.0.x (32 ビット Windows プラットフォーム用の基本パッケージ) および Instant Client Package for ODBC バージョン 11.1.0.x をダウンロードします。ダウンロード ファイルの名前は instantclient-basic-win32-11.1.0.7.0.zip および instantclient-odbc-win32-11.1.0.7.0.zip です。

   重要: *InstallShield における Oracle サポートには、ターゲット システムが 32 ビットまたは 64 ビット システムのいずれの場合においても、ターゲット システムに Basic Instant Client および ODBC Instant Client の 32 ビット バージョンが必要です。*

2. ファイルを C ドライブのルートに展開します。ファイルを展開すると、すべてのファイルが次の場所に追加されます:

   C:¥instantclient_11_1

ファイルが C:¥instantclient11_1¥instantclient11_1 に展開されないことを確認してください。

3. Oracle Instant Client インストール プロジェクトのコピーを作成します。プロジェクトのパスは、次のとおりです。

   InstallShield Program Files フォルダー¥Support¥Oracle Instant Client¥instantclient-win32-11_1.ism

   instantclient-win32-odbc11_1.ism のコピーを作成して、元の .ism ファイルと同じディレクトリに配置します。この新しいファイルのパスは次のとおりです:

   InstallShield Program Files フォルダー¥Support¥Oracle Instant Client¥instantclient-win32-odbc11_1.ism

   instantclient-win32-11_1.ism ロジェクトは Oracle Instant Client のバージョン 11.1.0.7 のために Revenera が作成したものですが、このプロジェクトを、より新しいバージョンの Oracle 11g Instant Client および ODBC Instant Client (例、11.2.01) に使用することも可能です。

4. InstallShield で作成した新しいプロジェクトを開きます。

5. 製品コード、アップグレード コード、および製品名を更新します:

   a. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。

   b. **"製品コード"** 設定で、**[新しい GUID の生成]** ボタン (**[..]**) をクリックします。

   c. **"アップグレード コード"** 設定で、**[新しい GUID の生成]** ボタン (**[..]**) をクリックします。

   d. **"製品名"** 設定で、既存のテキストを必要に応じて変更します。

      Oracle11g Instant Client – ODBC 11.1.0.7

      入力した名前は、実行時、Oracle のインストール中に表示されます。この名前を変更して、異なるバージョン番号 (例、11.2.0.1 または 11.2.0.x など) を反映させることができます。

6. 11.1.0.7 以後にリリースされたバージョンを使用していて、依存関係が必要な場合、そのファイルまたはマージ モジュールをプロジェクトに追加します。

7. ODBC Instant Client インストールを起動するカスタム アクションを追加します:

   a. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。

   b. 中央のペインで、**[カスタム アクション]** エクスプローラーを右クリックして、**[新しい EXE]** をポイントしてから、**[ディレクトリを参照するパス]** をクリックします。新しいカスタム アクションが追加されます。

   c. 次のように右側のペインでカスタム アクションの設定を構成します:

      - 作業ディレクトリ: INSTALLDIR
      - ファイル名とコマンド ライン: [INSTALLDIR]odbc_install.exe
      - スクリプト内実行: システム コンテキストの遅延実行
      - インストール実行シーケンス: 次の後: InstallODBC
      - インストール実行条件: Not Installed

8. エンド ユーザーが Instant Client をアンインストールするときに ODBC Instant Client をアンインストールするカスタム アクションを追加します:

   a. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。

b. 中央のペインで、**[カスタム アクション]** エクスプローラーを右クリックして、**[新しい EXE]** をポイントしてから、**[ディレクトリを参照するパス]** をクリックします。新しいカスタム アクションが追加されます。

c. 次のように右側のペインでカスタム アクションの設定を構成します:

- 作業ディレクトリ: INSTALLDIR
- ファイル名とコマンド ライン: [INSTALLDIR]odbc_uninstall.exe
- スクリプト内実行: システム コンテキストの遅延実行
- インストール実行シーケンス: 次の後: RemoveODBC
- インストール実行条件: REMOVE="ALL"

9. .msi パッケージ ファイルの名前を指定します:

a. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。

b. **[リリース]** エクスプローラーの中央ペインで、**[製品構成 1]** と名づけられた製品構成をクリックします。

c. **"MSI パッケージ ファイル名"** 設定に、次の名前を入力します:

Oracle11g Instant Client – ODBC 11.1

InstallShield は、作成する Basic Instant Client および ODBC Instant Client に .msi パッケージ用に入力された名前を使用します。このファイルが、作成する InstallShield 前提条件によって起動されます。

10. ツールバー上で、**[ビルド]** ボタンをクリックします。

Oracle11g Instant Client – ODBC 11.1.msi を作成中の InstallShield プロジェクトに含めることができるように、Oracle .msi ファイルがビルドされ、次のディレクトリに追加されます:

InstallShield Program Files フォルダー¥SetupPrerequisites¥oracle



タスク ***前述の手順に従って作成した .msi パッケージを起動する InstallShield 前提条件を作成するには、以下の手順に従います:***

1. Oracle11g Instant Client の 既存の InstallShield 前提条件 (.prq) のコピーを作成します。前提条件のパスは、次のとおりです:

InstallShield Program Files フォルダー¥SetupPrerequisites¥Oracle11g Instant Client 11.1.0.7.prq

コピーを Oracle11g Instant Client – ODBC 11.1.0.7.prq と名づけて、元の .prq ファイルと同じディレクトリに配置します。この新しいファイルのパスは次のとおりです:

InstallShield Program Files フォルダー¥SetupPrerequisites¥Oracle11g Instant Client – ODBC 11.1.0.7.prq

2. **[ツール]** メニューで **[前提条件エディター]** をクリックします。**InstallShield 前提条件エディター** が開きます。

3. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。

4. 新しい Oracle11g Instant Client – ODBC 11.1.0.7.prq ファイルを参照してから、**[開く]** ボタンをクリックします。

5. **[プロパティ]** タブにある **"InstallShield 前提条件の固有の ID"** 設定に次を入力します:

Oracle11g Instant Client – ODBC 11.1.0.7

異なるバージョンを使用する場合は、適切な ID に更新してください。

6. InstallShield 前提条件の条件を構成します:
   a. **[条件]** タブをクリックします。
   b. 各既存の条件をクリックしてから、**[削除]** ボタンをクリックします。
   c. **[追加]** ボタンをクリックします。**[前提条件設定]** ダイアログ ボックスが開きます。
   d. **[レジストリキーの存在の有無]** オプションを選択します。
   e. **"確認するレジストリ キー名を指定します"** 設定に次を入力します:

      HKEY_LOCAL_MACHINE¥SOFTWARE¥ODBC¥ODBCINST.INI¥Oracle in instantclient11_1

      異なるバージョンを使用する場合は、適切な条件を構成してください。
   f. **[32 ビット]** オプションを選択します。
   g. **[指定されたレジストリ キーが "存在しない" とき]** オプションを選択してから、**[OK]** をクリックします。
7. InstallShield 前提条件に含める適切な .msi ファイルを指定します:
   a. [**含めるファイル**] タブをクリックします。
   b. 既存の **Oracle11g Instant Client 11.1.msi** を選択してから、**[更新]** ボタンをクリックします。**[新規ファイル]** ダイアログ ボックスが開きます。
   c. 前述の手順に従ってビルドした Oracle11g Instant Client – ODBC 11.1.msi ファイルを選択してから、**[OK]** をクリックします。
8. InstallShield 前提条件が起動するファイルを指定します:
   a. **[実行するアプリケーション]** タブをクリックします。
   b. **"起動するアプリケーションを指定する"** 設定で、.msi ファイルを選択します。
9. **[ファイル]** メニューで、**[保存]** をクリックします。

Oracle 11g Instant Client および ODBC Instant Client をインストールに追加する方法については、「Oracle 11g Instant Client をインストールする」を参照してください。

Microsoft ODBC for Oracle の代わりに Oracle ODBC Instant Client を使用する場合は、プロジェクトを適切に構成するようにしてください。詳細については、「Oracle Database Server 上で Unicode 文字を含む SQL スクリプトを実行する」を参照してください。

Oracle 11g Instant Client をインストールする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

インストールが実行される時に Oracle 11g Instant Client をインストールする場合、Oracle 11g Instant Client 前提条件をプロジェクトに含めます。

InstallShield 前提条件を含める前に、Oracle Basic Instant Client をダウンロードして、.msi パッケージを作成する必要があります。Oracle ODBC Instant Client のインストールも必要なとき、たとえば Oracle で Unicode サポートを使用する場合は、Basic Instant Client および ODBC Instant Client をダウンロードし、.msi パッケージをビルドし、InstallShield 前提条件を作成してから (Oracle 11g Instant Client をテンプレートとして使用できます)、必要に応じて設定を更新します。詳細については、次を参照してください。

- Oracle Instant Client をダウンロードして、そのための .msi パッケージを作成する
- Oracle Basic Instant Client および Oracle ODBC Instant Client をダウンロードして、両方の .msi パッケージおよび InstallShield 前提条件を作成する



重要: *InstallShield における Oracle サポートには、ターゲット システムが 32 ビットまたは 64 ビット システムのいずれの場合においても、Oracle Instant Client の 32 ビット バージョンのインストールが必要です。*



タスク ***Oracle 11g Instant Client 前提条件をプロジェクトに含めるには、以下の手順に従います:***

1. 基本の MSI および InstallScript MSI プロジェクトの場合、**[アプリケーション データ]** の下にあるビュー リストで、**[再配布可能ファイル]** をクリックします。

   InstallScript プロジェクトの場合: **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。

2. 再配布可能ファイルのリストから [**Oracle11g Instant Client 11.1.0.7**] チェック ボックス (または、任意の Oracle 前提条件のチェック ボックス) を選択します。

InstallShield 前提条件ファイル (.prq) で定義された条件が満たされた場合のみ、InstallShield 前提条件が起動します。条件の一覧を参照するには、[再配布可能ファイル] ビューまたは [前提条件] ビューにある Oracle 前提条件をクリックしてから、右側の詳細ペインに表示される説明を確認します。これらのビューで [詳細を表示] ボタンをクリックして、詳細の表示/非表示を切り替えることができます。



メモ: *Oracle 11g InstaInt Client がターゲット システム上で必要な時にインストールがこれをダウンロードする場合、Oracle に問い合わせて、それが可能かどうかを確認してください。可能な場合、独自の Web サイトでダウンロードをホストして、InstallShield 前提条件にそのダウンロード パスを追加しなくてはなりません。Revenera では、このインストールのダウンロードを提供していません。*

Oracle Database Server 上で Unicode 文字を含む SQL スクリプトを実行する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

InstallShield の [SQL スクリプト] ビューで使用できるデフォルトの SQL スクリプト サポートを使用するインストールの場合、Microsoft ODBC for Oracle を使用して、ターゲット システム上の Oracle データベースとの接続が行われます。一部の状況において、このデフォルトの動作をオーバーライドして、Microsoft ODBC for Oracle の代わりに、Oracle ODBC Instant Client を使用した方がよい場合もあります。

たとえば、SQL スクリプトに Unicode 文字が含まれている場合、Unicode 文字を含む SQL スクリプトの実行をサポートする Oracle ODBC Instant Client を使用する必要があります。Microsoft ODBC for Oracle は、これをサポートしません。



タスク

InstallShield プロジェクトで、インストールが Microsoft ODBC for Oracle ではなく Oracle ODBC Instant Client を使用するように構成するには、以下の手順に従います:

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーで、**ISSQLDBMetaData** テーブルをクリックします。
3. **Oracle** レコードについて、以下の手順を行います:
   a. AdoDriverName フィールドを以下の値に変更します:

      {Oracle in instantclient11_1}

   b. AdoCxnServer フィールドを以下の値に変更します:

      DBQ=

実行時にインストールは Oracle ODBC Instant Client を使って Oracle データベースに接続し、SQL スクリプトを実行します。



ヒント: *Oracle ODBC Instant Client は、Oracle Instant Client ソフトウェアの ODBC パッケージがインストールされている場合にターゲット システム上で使用可能です。インストールに追加する方法については、「Oracle Basic Instant Client および Oracle ODBC Instant Client をダウンロードして、両方の .msi パッケージ および InstallShield 前提条件を作成する」を参照してください。*

カスタマイズした SQL スクリプトを実行して SQL Server データベースを作成するサンプル プロジェクトを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

以下は、カスタマイズした SQL スクリプトを使用して SQL Server データベースを作成するインストールを作成する方法です。



タスク ***カスタマイズした SQL スクリプトを実行して、ターゲット マシン上で SQL Server データベースを作成するインストールを作成するには、以下の手順に従います。***

1. 基本の MSI プロジェクトまたは InstallScript MSI プロジェクトを新規作成します。
2. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
3. 次の名前を持つ新しいプロパティを作成します。

   IS_SQLSERVER_DATABASE2
4. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
5. 新しい SQL 接続を追加し、構成します。
   a. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。新しい接続が **NewConnection1** というデフォルト名で作成されます。
   b. **[SQL スクリプト]** エクスプローラーで、**NewConnection1** をクリックし、**[全般]** タブをクリックします。
   c. **[デフォルト ターゲット サーバー名 (オプション)]** ボックスで、**TESTSQLSERVER** と入力します。
   d. **[存在しない時、カタログを作成する]** チェック ボックスをクリアします。
   e. **[接続方法]** 領域で、**[次のログイン ID およびパスワードを使用したサーバー認証]** オプションを選択します。
   f. **[ログイン ID]** ボックスで、**sa** と入力します。**[パスワード]** ボックスは空白にできません。
   g. **[要件]** タブをクリックします。
   h. **Microsoft SQL Server** チェック ボックスが選択されていて、**MySQL** と **Oracle** チェック ボックスは選択されていないことを確認してください。
6. NewConnection1 の新しい SQL スクリプトを追加し、構成します。
   a. **[SQL スクリプト]** エクスプローラーで、**NewConnection1** を右クリックして、**[新しいスクリプト]** をクリックします。
   b. スクリプトの名前を **NewScript1** に変更します。
   c. **[SQL スクリプト]** エクスプローラーで、**NewScript1** をクリックし、**[スクリプト]** タブをクリックします。
   d. スクリプト エディター ペインで、次のスクリプトを追加します。

      CREATE DATABASE [TestDB] ON (NAME = N' TestDB', FILENAME = N'C:¥Program Files¥Microsoft SQL Server¥MSSQL¥data¥testdb.mdf' , SIZE = 3, FILEGROWTH = 10%) LOG ON (NAME = N' TestDB_log', FILENAME = N'C:¥Program Files¥Microsoft SQL Server¥MSSQL¥data¥testdb.ldf' , SIZE = 1, FILEGROWTH = 10%) COLLATE SQL_Latin1_General_CP1_CI_AS
   e. **[ランタイム]** タブをクリックします。
   f. **[スクリプトの実行]** 領域で、**[ログイン中にスクリプトを実行する]** チェック ボックスを選択し、**[インストール中にスクリプトを実行する]** および **[アンインストール中にスクリプトを実行]** チェック ボックスが選択されていないことを確認してください。
7. ２ つ目の新しい SQL 接続を追加し、構成します。

a. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。新しい接続が **NewConnection2** というデフォルト名で作成されます。

b. **[SQL スクリプト]** エクスプローラーで、**NewConnection2** をクリックし、**[詳細]** タブをクリックします。

c. **[ターゲット カタログ プロパティ名]** リストから、**IS_SQLSERVER_DATABASE2** を選択します。

d. **[全般]** タブをクリックします。

e. **[カタログ名]** ボックスで、**TestDB** と入力します。

f. **[存在しない時、カタログを作成する]** チェック ボックスをクリアします。

g. **[デフォルト ターゲット サーバー名 (オプション)]** ボックスで、**TESTSQLSERVER** と入力します。

h. **[接続方法]** 領域で、**[次のログイン ID およびパスワードを使用したサーバー認証]** オプションを選択します。

i. **[ログイン ID]** ボックスで、**sa** と入力します。**[パスワード]** ボックスは空白にできません。

j. **[要件]** タブをクリックします。

k. **Microsoft SQL Server** チェック ボックスが選択されていて、**MySQL** と **Oracle** チェック ボックスは選択されていないことを確認してください。

8. NewConnection2 の新しい SQL スクリプトを追加し、構成します。

a. **[SQL スクリプト]** エクスプローラーで、**NewConnection2** を右クリックして、**[新しいスクリプト]** をクリックします。

b. スクリプトの名前を **NewScript2** に変更します。

c. **[SQL スクリプト]** エクスプローラーで、**NewScript2** をクリックし、**[スクリプト]** タブをクリックします。

d. スクリプト エディター ペインで、次のスクリプトを追加します。

```
CREATE TABLE TestTable (TestColumn1 CHAR NOT NULL PRIMARY KEY)
```

e. **[ランタイム]** タブをクリックします。

f. **[スクリプトの実行]** 領域で、**[インストール中にスクリプトを実行する]** チェック ボックスを選択し、**[ログイン中にスクリプトを実行する]** および **[アンインストール中にスクリプトを実行]** チェック ボックスが選択されていないことを確認してください。

インストールを実行すると、**TestDB** データベースが作成され、**TestTable** という名前のテーブルがデータベースに追加されます。



プロジェクト: *DIM プロジェクトを使用して、その DIM をインストール プロジェクトに追加しても、上記の手順と同様の結果を得ることができます。*

カスタマイズした SQL スクリプトを実行して Oracle スキーマを作成するサンプル インストールを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

以下は、カスタマイズした SQL スクリプトを使用して、Oracle スキーマを作成するインストールを作成する方法です。



タスク

カスタマイズした SQL スクリプトを実行して、ターゲット マシン上で Oracle スキーマを作成するインストールを作成するには、以下の手順に従います。

1. 基本の MSI プロジェクトまたは InstallScript MSI プロジェクトを新規作成します。
2. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
3. 次の名前を持つ 3 つの新しいプロパティを作成します。

   IS_SQLSERVER_DATABASE2

   IS_SQLSERVER_USERNAME2

   IS_SQLSERVER_PASSWORD2
4. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
5. 新しい SQL 接続を追加し、構成します。
   a. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。新しい接続が **NewConnection1** というデフォルト名で作成されます。
   b. **[SQL スクリプト]** エクスプローラーで、**NewConnection1** をクリックし、**[全般]** タブをクリックします。
   c. **[デフォルト ターゲット サーバー名 (オプション)]** ボックスで、次のように入力します。

   //sch01jsmith.installshield.com:1521/orcl

   d. **[存在しない時、カタログを作成する]** チェック ボックスをクリアします。
   e. **[接続方法]** 領域で、**[次のログイン ID およびパスワードを使用したサーバー認証]** オプションを選択します。
   f. **[ログイン ID]** ボックスで、**scott** と入力します。
   g. **[パスワード]** ボックスで、**scott** と入力します。
   h. **[要件]** タブをクリックします。
   i. **Oracle** チェック ボックスが選択されていて、**Microsoft SQL Server** と **MySQL** チェック ボックスがクリアされていることを確認してください。
6. NewConnection1 の新しい SQL スクリプトを追加し、構成します。
   a. **[SQL スクリプト]** エクスプローラーで、**NewConnection1** を右クリックして、**[新しいスクリプト]** をクリックします。

b. スクリプトの名前を **NewScript1** に変更します。

c. **[SQL スクリプト]** エクスプローラーで、**NewScript1** をクリックし、**[スクリプト]** タブをクリックします。

d. スクリプト エディター ペインで、次のスクリプトを追加します。

```
CREATE TABLESPACE TEST_TS LOGGING DATAFILE 'test01.dbf' SIZE 1M AUTOEXTEND ON NEXT 2M MAXSIZE
UNLIMITED
Go
CREATE USER TEST_USER IDENTIFIED BY MYPSWD DEFAULT TABLESPACE TEST_TS QUOTA UNLIMITED on TEST_TS
Go
GRANT CONNECT TO TEST_USER
Go
GRANT DBA TO TEST_USER
Go
ALTER USER TEST_USER DEFAULT ROLE ALL
Go
```

e. **[ランタイム]** タブをクリックします。

f. **[スクリプトの実行]** 領域で、**[ログイン中にスクリプトを実行する]** チェック ボックスを選択し、**[インストール中にスクリプトを実行する]** および **[アンインストール中にスクリプトを実行]** チェック ボックスが選択されていないことを確認してください。

7. ２ つ目の新しい SQL 接続を追加し、構成します。

a. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。新しい接続が **NewConnection2** というデフォルト名で作成されます。

b. **[SQL スクリプト]** エクスプローラーで、**NewConnection2** をクリックし、**[詳細]** タブをクリックします。

c. **[ターゲット カタログ プロパティ名]** リストから、**IS_SQLSERVER_DATABASE2** を選択します。

d. **[サーバー認証ログイン ID プロパティ名]** リストから、**IS_SQLSERVER_USERNAME2** を選択します。

e. **[サーバー認証パスワード プロパティ名]** リストから、**IS_SQLSERVER_PASSWORD2** を選択します。

f. **[全般]** タブをクリックします。

g. **[カタログ名]** ボックスで、**TEST_USER** と入力します。

h. **[存在しない時、カタログを作成する]** チェック ボックスをクリアします。

i. **[デフォルト ターゲット サーバー名 (オプション)]** ボックスで、次のように入力します。

//sch01jsmith.installshield.com:1521/orcl

j. **[接続方法]** 領域で、**[次のログイン ID およびパスワードを使用したサーバー認証]** オプションを選択します。

k. **[ログイン ID]** ボックスで、**TEST_USER** と入力します。

l. **[パスワード]** ボックスで、**MYPSWD** と入力します。

m. **[要件]** タブをクリックします。

n. **Oracle** チェック ボックスが選択されていて、**Microsoft SQL Server** と **MySQL** チェック ボックスがクリアされていることを確認してください。

8. NewConnection2 の新しい SQL スクリプトを追加し、構成します。

a. **[SQL スクリプト]** エクスプローラーで、**NewConnection2** を右クリックして、**[新しいスクリプト]** をクリックします。

b. スクリプトの名前を **NewScript2** に変更します。

c. **[SQL スクリプト]** エクスプローラーで、**NewScript2** をクリックし、**[スクリプト]** タブをクリックします。

d. スクリプト エディター ペインで、次のスクリプトを追加します。

CREATE TABLE TestTable (TestColumn1 CHAR NOT NULL PRIMARY KEY)

e. **[ランタイム]** タブをクリックします。

f. **[スクリプトの実行]** 領域で、**[インストール中にスクリプトを実行する]** チェック ボックスを選択し、**[ログイン中にスクリプトを実行する]** および **[アンインストール中にスクリプトを実行]** チェック ボックスが選択されていないことを確認してください。

インストールを実行すると、**TEST_USER** スキーマで **TEST_USER** ユーザーと **TestTable** テーブルが作成されます。

プロジェクト: *DIM プロジェクトを使用して、その DIM をインストール プロジェクトに追加しても、上記の手順と同様の結果を得ることができます。*

スイート/アドバンスト UI プロジェクトで、SQLLogin 定義済みウィザード ページを追加する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

SQL サーバーは、特に InstallShield スイート インストールで提供される複数パッケージ サポートを活用する多くのアプリケーションで不可欠です。以前、InstallShield SQL サポートが使用できるのは基本の MSI、InstallScript、および InstallScript MSI プロジェクトのみでした。今回より、SQL サポートがスイート/アドバンスト UI プロジェクトに追加されており、以下の機能を使用することができます:

- 新しい SQLLogin 定義済みウィザード ページを追加
- SQL ステートメントを直接スイートから実行

SQLLogin ウィザード ページを使って、エンド ユーザーはデータベース サーバー ログイン情報 (データベース サーバー名、認証資格情報、データベース カタログ名など) を入力し、スイートに含まれる 1 つ以上の .msi パッケージがターゲットとするデータベース サーバーへの接続を設立することができます。

SQLLogin ウィザード ページをプロジェクトに追加すると、以下が可能となります:

- スイート プロジェクト内の SQL ログインの設定を識別するプロパティを指定してから、これらのプロパティを受け取る .msi パッケージを選択する

- .msi パッケージ内で SQL ログインの設定を識別するプロパティを指定する
- データベース テクノロジ (Microsoft SQL Server、Microsoft Windows Azure、MySQL、または Oracle) を選択して、ターゲットにする ODBC ドライバーを選択する
- OnBegin でアクションを使って、更新済みの SQL ドライバーが確実にスケジュールされるようにする

さらに、スイート/アドバンスト UI プロジェクトでは今回より、ユーザー インターフェイスから SQL データベース サーバー上の SQL ステートメントを直接実行することができます。これによって、インストール続行前に SQL データベース サーバーを調査することができます。これによって、スイート プロパティで SQL クエリの結果にアクセスが可能となります。

次は、SQLLogin 定義済みウィザード ページをプロジェクトに追加する方法を示します。別の方法として、スイート プロジェクト内の SQL サポートを手動で構成して、インストールのニーズに従って UI をカスタマイズすることもできます。



タスク ***スイート/アドバンスト UI プロジェクトで、SQLLogin 定義済みウィザード ページを追加して、オプションで SQL ステートメントを直接実行するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を右クリックしてから、**[定義済みページの追加]** をクリックします。**ユーザー インターフェイス ウィザード**が開きます。.
3. **[タスク ページ]** リストから、**[データベース サーバーのログイン情報を入力]** を選択します。
4. **[後続ページ]** リストを使って、ウィザード ページ シーケンスでデータベース サーバー ウィザード ページをスケジュールする場所を指定します。実行時に、このリストで選択したページが、新しいデータベース サーバー ウィザード ページの直後に表示されます。
5. ユーザー インターフェイス ウィザードの [タスク構成] パネルには、次のオプションがあります:
   a. スイート プロジェクト内の "SQL ログイン" の設定を識別するプロパティを指定する

      ***メモ・** 1 つのスイート プロジェクトには、異なる SQL サーバーを必要とする、または共有する必要のある複数パッケージが含まれている場合があります。*

   b. プロパティを受け取る .msi パッケージを選択する。
   c. データベース テクノロジおよびドライバーを選択する。

   InstallShield は、指定済みの "タスク構成" プロパティに対応する "開始ページ" 設定を追加します。
6. オプションで、実行時にスイート/アドバンスト UI プロジェクトから 直接実行する SQL ステートメントを追加する場合、プロジェクトに **[SQL 文字列の実行]** アクションを追加してから、必要に応じてそのサブ設定を指定します。SQL クエリ結果には、**[データ プロパティ]** サブ設定で指定するスイート プロパティからアクセスが可能です。

COM+ アプリケーションとコンポーネントの管理



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

[コンポーネントサービス] ビューでは、インストールパッケージ用の COM+ アプリケーションとコンポーネントを管理することができます。COM+ サーバー アプリケーションとアプリケーション プロキシの両方を管理することができます。COM+ アプリケーション プロキシはサーバー アプリケーション属性のサブセットで構成され、これはクライアン ト コンピューターからアプリケーションが存在するマシンへのリモート アクセスを可能にします。

InstallShield のコンポーネント サービスに関する次の情報をお読みください。

- COM+ システム アプリケーション以外のみプロジェクトに追加することができます。したがって、InstallShield は [コンポーネント サービス] ビュー の [COM+ アプリケーション] エクスプローラーの下に、COM+ システム アプリケーション以外のみを表示します。
- ローカル マシンにインストールされている COM+ アプリケーションのみが、[コンポーネント サービス] ビューに表示され、プロジェクトに追加することが可能です。
- アプリケーションプロキシは COM+ サーバーアプリケーションでのみ利用可能で、ライブラリアプリケーションでは利用できません。

[コンポーネント サービス] ビューの外観は、[コントロール パネル] の [コンポーネント サービス] 管理ツールに似ています。

COM+ サーバー アプリケーションの管理



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*



タスク ***COM+ サーバー アプリケーションのインストール設定を構成するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[コンポーネント サービス]** をクリックします。
2. **[コンポーネント サービス]** エクスプローラーで、構成する COM+ アプリケーションをまだ選択していない場合は、それを選択します。
3. **[インストール]** タブで、**[サーバー]** チェック ボックスを選択します。



メモ - [プロキシ]チェック ボックス、または [サーバー] チェック ボックス、もしくはその両方を選択してください。これらのチェック ボックスの 1 つをクリアすると、もう 1 つのチェック ボックスは選択された状態が保たれますが、変更することはできません。これは誤って両方のチェック ボックスをクリアしないよう防ぐためです。

4. **[ビルド時にクライアントマシンから COM+ 設定をリフレッシュする]** チェック ボックスを必要に応じて選択します。
5. インストールが COM+ アプリケーション プロキシ サポートを含まない場合、[**プロキシ**] チェック ボックスをクリアします。

COM+ サーバー アプリケーションをプロジェクトへ追加すると、対応するコンポーネントが、プロキシ コンポーネントと関連付けられている各機能に対して作成されます。このコンポーネントは COM+ アプリケーションのサーバー .dll ファイルを含みます。



***メモ** - 選択された COM+ アプリケーションがサーバーとプロキシ インストールの両方を含む場合、インストール条件を追加して適切なアプリケーションがターゲット マシンへインストールされるように設定することができます。詳細については、「サーバーとクライアント マシンをターゲットにした COM+ アプリケーションを含める」を参照してください。*

COM+ アプリケーション プロキシの管理



***プロジェクト** - この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

COM+ アプリケーション プロキシ サポートは、クライアント マシンからサーバー アプリケーションが存在するマシンへのリモート アクセスを可能にするシステム設定を構成しますクライアント アプリケーションをインストールに追加しなくてはならない場合もあります。



タスク ***COM+ アプリケーション プロキシのインストール設定を構成するには、以下の手順を実行します。***

1. **[サーバー構成]** の下のビュー リストにある **[コンポーネント サービス]** をクリックします。
2. **[コンポーネント サービス]** エクスプローラーで、構成する COM+ アプリケーションをまだ選択していない場合は、それを選択します。
3. **[インストール]** タブで、**[プロキシ]** チェック ボックスを選択します。
4. インストールが COM+ サーバー アプリケーションを含まない場合は、**[サーバー]** チェック ボックスをクリアします。



***メモ・[プロキシ]**チェック ボックス、または **[サーバー]** チェック ボックス、もしくはその両方を選択してください。これらのチェック ボックスの 1 つをクリアすると、もう 1 つのチェック ボックスは選択された状態が保たれますが、変更することはできません。これは誤って両方のチェック ボックスをクリアしないよう防ぐためです。*

5. **[リモート サーバー名]** ボックスには、アプリケーションが常駐するリモート サーバー コンピューターの名前を指定します。正しい名前を入力するか、またはデフォルトの [REMOTESERVERNAME] プロパティを利用することが可能です。このデフォルト値はインストールで COM+ アプリケーションの **[プロキシ]** チェック ボックスを選択したときに自動的に作成されます。

   ***メモ・**REMOTESERVERNAME *プロパティのデフォルト値は、InstallShieldで COM+ アプリケーションを インストール プロジェクトへ追加するために使用されるマシンの名前です。*[REMOTESERVERNAME] *プロパティの値を変更するには、[プロパティ マネージャー] ビューを使用します。*

   *エンドユーザーがリモート サーバーを指定できるようにしたい場合は、**[ダイアログ]** ビューのエンドユーザー ダイアログにリモート サーバー編集フィールド コントロールを追加します。このコントロールの **[プロパティ]** 値に* **REMOTESERVERNAME***を設定します。*

6. **[分散 COM をクライアント マシンで有効にする]** チェック ボックスを必要に応じて選択します。DCOM (分散コンポーネント オブジェクト モデル) がすべてのクライアント マシン上で既に有効であることが分かっていて、クライアント マシン上での管理者権限を持たないことが分かっている場合、このチェック ボックスをクリアします。

   このチェック ボックスを選択した場合、インストール時に Y が HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Ole レジストリキーの EnableDCOM エントリに書き込まれて、DCOM が有効となります。

   メモ・エンドユーザーは [コントロール パネル] にある [コンポーネント サービス] 管理ツールを利用して、マシン上で DCOM を有効または無効にすることができます。ただし、そのマシンの DCOM が無効になっている場合、アプリケーション プロキシはクライアント マシン上では動作しません。そのため、[クライアント マシン上で 分散 COM を有効にする] チェック ボックスの選択をお勧めします。

   エンドユーザーがアプリケーション プロキシ サポートをアンインストールする場合、EnableDCOM レジストリ エントリは、インストール プロセスでこのレジストリ エントリがターゲット マシンで Y に変わったとしても、変更はされません。

COM+ アプリケーション プロキシをプロジェクトへ追加すると、対応するコンポーネントが、サーバー コンポーネントに関連付けられている各機能に対して作成されます。このコンポーネントは COM+ アプリケーションのサーバー .dll ファイルを含みます。これらのサーバー ファイルはクライアント マシン上のタイプ ライブラリにインストールされます。



メモ・選択された COM+ アプリケーションがサーバーとプロキシ インストールの両方を含む場合、インストール条件を追加して適切なアプリケーションがターゲット マシンへインストールされるように設定することができます。詳細については、「*サーバーとクライアント マシンをターゲットにした COM+ アプリケーションを含める*」を参照してください。

サーバーとクライアント マシンをターゲットにした COM+ アプリケーションを含める



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

適切なコンポーネントがターゲット マシンにインストールされるようにインストール条件を作成することで、インストール時に、サーバー アプリケーションをサーバーにインストールし、アプリケーション プロキシ サポートをクライアント マシンにインストールすることができます。実際の手順を以下に例を挙げて説明します。



タスク

サーバーとクライアント マシン両方をターゲットにした COM+ アプリケーションをインストールに含めるには、以下の手順を実行してください。

1. **[プロパティ マネージャー]** ビューで、COMPLUSTYPE という名前で新しいプロパティを作成します。
2. **[ダイアログ]** ビューで、新規または既存のエンドユーザー ダイアログにラジオ ボタン グループ コントロールを追加し、このコントロールの **Property** 値に **COMPLUSTYPE** を設定します。このラジオ ボタン グループを利用して、エンドユーザーはサーバーまたはプロキシのどちらのインストールを実行するのかを指定できます。
3. ラジオ ボタン グループに **[サーバー]** オプション用のラジオ ボタンを追加し、このコントロールの **Value** プロパティを [Server] に設定します。
4. ラジオ ボタン グループに **[プロキシ]** オプション用のラジオ ボタンを追加し、このコントロールの **Value** プロパティを [Proxy] に設定します。
5. **[コンポーネント サービス]** ビューで、構成する COM+ アプリケーションを選択します。
6. **[インストール]** タブで、[サーバー インストール] および [アプリケーション プロキシ] のパラメーターを設定します。
7. **[サーバー]** チェック ボックスの下の **[条件]** ボックスに、次を入力します:

   COMPLUSTYPE="サーバー"
8. **[プロキシ]** チェック ボックスの下の **[条件]** ボックスに、次を入力します:

   COMPLUSTYPE="プロキシ"

インターネット インフォメーション サービス



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

インターネットインフォメーションサービス (IIS) は Microsoft が開発した Web サーバーです。Web ベースのアプリケーションをビルドおよび配布、Web サイトの管理、およびインターネットまたはイントラネットへ情報をパブリッシュするための安定したプラットフォームを提供します。

InstallShield の [IIS 構成] ビューでは、新しい IIS Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張を作成および管理することができます。

InstallShield における IIS サポートのバージョン固有情報



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

以下は、特定のバージョンの IIS に関する情報です。

- IIS は Windows 2000 Server 以降および Windows XP 以降のシステムに含まれています。IIS 6 は Windows Server 2003 システムでのみ利用できます。IIS 7 は Windows Vista と Windows Server 2008 システムで提供されています。IIS 7.5 は Windows 7 と Windows Server 2008 R2 システムで提供されています。IIS は自動的にインストールされません (デフォルト)。
- [IIS 構成] ビューにある一部の Web サイトと仮想ディレクトリの設定は、特定のバージョンの IIS に適用します。これらの設定についてのバージョン固有の情報は、InstallShield のインライン ヘルプ ペインに表示されます。バージョン固有のプロパティが構成されていて、ターゲット システムに対応するバージョンの IIS がないとき、IIS はバージョン固有のプロパティを無視します。

  たとえば、IIS 7 と IIS 6 は、アプリケーションまたは仮想ディレクトリの "アプリケーション保護" プロパティをサポートしません。[IIS 構成] ビューで、このプロパティは、アプリケーションまたは仮想ディレクトリの [アプリケーションの設定] 領域にある設定を使って構成されます。[IIS 構成] ビューでこの設定を選択すると、右下に表示されるヘルプ ペインに、この設定が IIS 6 以降には適用しないことが示されます。この設定が選択されているときに、エンドユーザーが製品を IIS 6 以降があるターゲット システムにインストールすると、"アプリケーション保護" 設定は無視されます。
- Web サービス拡張、アプリケーション プール、およびすべての関連プロパティは、IIS 6 があるマシン上でのみ提供されています。

  IIS 7 があるシステムでは、Web サービス拡張に、[IIS メタベースと IIS 6 構成の互換性] 機能がインストールされている必要があります。この機能がインストールされていない場合、インストールのログ ファイルに、この機能が必須である可能性があることを通知するエラー2147467259 が表示される場合があります。

  Windows Vista または Windows Server 2008 システムに、[IIS メタベースと IIS 6 構成の互換性] 機能がインストールされているかどうかを判別する方法については、「ターゲット システムに IIS 6 以前が搭載されているか、または IIS 6 メタベースの互換性機能があるかどうかを判別する」を参照してください。

- Windows Vista 以降と Windows Server 2008 以降のシステムでは、インストール プロジェクトで [IIS 構成] ビューの "コンテンツのソース パス (ローカルまたは UNC)" 設定で指定された物理パスにある構成ファイルに個別の Web サイト、アプリケーション、および仮想ディレクトリの設定が保管されます。したがって、各 Web サイト、アプリケーション、または仮想ディレクトリは、一意の物理パスを持ちます。一意の物理パスを持たなかった場合、同じ物理パスを持つ 2 つの異なる仮想ディレクトリがあるとき、たとえばテスト環境などで予期しない動作に遭遇する可能性があります。

  たとえば、同じ物理パスがある 2 つの仮想ディレクトリがあり、ディレクトリの参照が片方のみで有効にされているとき、作成された 2 つ目の仮想ディレクトリの "ディレクトリの参照" 設定は、1 つ目の仮想ディレクトリの設定をオーバーライドします。

- Windows Server 2003 を持つシステムでは、IIS 6 がインストールされていない場合、他の IIS ディレクトリおよびサイトは上記にかかわらず作成されます。IIS 6 固有の設定はスキップされます。
- IIS 5.1 for Windows XP Professional は、一回につき 1 つの Web サイトのみサービス可能です。これは IIS 5.1 の制限によるものです。
- InstallShield はバージョン 5 およびそれ以降の IIS をサポートします。

インストールする IIS のバージョンを判別する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

[IIS 構成] ビューで Web サイトを追加すると、ターゲット システムにインストールされる IIS バージョンを検索するためのエントリが [システム検索] ビューに追加されます。IIS_VERSION プロパティは (ダイレクト エディターにある) RegLocator と AppSearchテーブルのエントリによって設定され、インストールされた IIS のバージョンを判別します。

IIS_VERSION プロパティには、IIS のバージョン番号が含まれています。インストールされている IIS バージョンを判別する必要がある場合、このプロパティの値を確認します。

IIS サポートの実行時要件

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield インストールの IIS サポートは、ターゲット マシンに IIS がインストールされていて、エンド ユーザーが管理者権限を持つ場合のみ動作します。

IIS の設定を含むパッケージのインストール実行中、InstallShield インストールはターゲットマシンに IIS があるかどうかを確認します。IIS がインストールされていない場合、インストールはエンドユーザーにダイアログを表示して IIS がインストールされていないことを通知します。ダイアログで、エンドユーザーは [中止]、[再試行]、[無視] のいずれかを選択することができます。

- エンドユーザーが [中止] を選択すると、インストールが終了します。
- エンドユーザーが IIS をインストールしてから、[再試行] を選択したとき、インストールは IIS の存在を再確認し、インストールを続行します。エンドユーザーが IIS をインストールしていないにもかかわらず [再試行] を選択したとき、インストールは IIS の存在を再確認し、ダイアログを再度表示します。
- エンドユーザーが [無視] を選択すると、インストールは続行しますが、IIS Web サイト、アプリケーション、仮想ディレクトリ、および他の IIS 要素は構成されません。



メモ: *InstallShield では、インストールを実行している ターゲット マシン以外のターゲット マシンでの Web サイトの作成はサポートされていません 。*



プロジェクト: *InstallScript プロジェクトでアプリケーション プールを作成する場合で、アプリケーション プールに仮想ディレクトリ、アプリケーション、または Web サイトに関連付けるには、同じコンポーネント内でアプリケーション プールと仮想ディレクトリ、アプリケーション、または Web サイトを作成しなくてはなりません。これを行うことで、仮想ディレクトリ、アプリケーション、または Web サイトが作成される前にアプリケーション プールが作成されるようになります。これは、IIS 6 以降の要件で、基本の MSI、DIM、InstallScript MSI、またはマージ モジュール プロジェクトの制限事項ではありません。*

Web サーバーで CMD コマンドが SSI #exec ディレクティブに使用されるのを許可するかどうかを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

サーバー側インクルード (SSI) ディレクティブは、コンテンツを Web ページに挿入するようにWeb サーバーに指示します。#exec タイプのディレクティブによって、Web サーバーは Web ページにシェル コマンドの出力を含めることができます。

IIS Web サーバーを構成して、#exec ディレクティブの CMD コマンドがシェル コマンドの実行に使用されるのを防いだり、CMD コマンドがこのタイプのコマンドの実行に使用されることを許可することができます。HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥W3SVC¥Parameters レジストリ キーの SSIEnableCmdDirective レジストリ値によって、CMD コマンドが許可されているかどうかが判別されます。

InstallShield では、インストール時に、ターゲット システム上で SSIEnableCmdDirective レジストリ値をどう構成するかを指定することができます。インストールで SSIEnableCmdDirective レジストリ値を変更しない場合、そのように指定することもできます。

セキュリティに関する懸念により、デフォルトの SSIEnableCmdDirective 値は FALSE (0) になっています。FALSE (0) 値により、エンドユーザーによって承認されていないサーバー側での実行可能ファイルの実行を防ぐことができます。



タスク ***Web サーバーで CMD コマンドが SSI #exec ディレクティブに使用されるのを許可するかどうかを指定するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. 中央のペインで、**[Web サイト]** エクスプローラーをクリックします。右側のペインにショートカットの設定が表示されます。
3. **"SSIEnableCmdDirective レジストリ値"** 設定で、適切なオプションを選択します。
   - **無視する** – ターゲット システム上の SSIEnableCmdDirective レジストリ値を変更しません。デフォルトでは、これが設定されています。
   - **FALSE (0)**–ターゲット システム上の SSIEnableCmdDirective レジストリ値を 0 に設定します。これにより、サーバー側インクルードの #exec CMD ディレクティブがシェル コマンドの実行に使用されるを防ぐことができます。この値を選択すると、IIS Web サーバーに #exec CMD ディレクティブに依存するアプリケーションが存在した場合、インストール プロジェクトの Web サイトおよび仮想ディレクトリがインストールされたあと、これらのアプリケーションが誤作動を起こす可能性があります。
   - **TRUE (1)**–ターゲット システム上の SSIEnableCmdDirective レジストリ値を 1 に設定します。これにより、サーバー側インクルードの #exec CMD ディレクティブがシェル コマンドの実行で使用できるようになります。

FALSE または TRUE オプションを選択すると、値 (FALSE の場合 0、TRUE の場合 1) がINSTALLSHIELD_SSI_PROP プロパティに格納されます。

インストールにある 1 つまたは複数の Web サイト、仮想ディレクトリ、アプリケーション プール、または Web サービス拡張がターゲット システムにインストールされて、**"SSIEnableCmdDirective レジストリ値"** 設定で FALSE または TRUE オプションが選択されている場合、SSIEnableCmdDirective レジストリ値がターゲット システムで更新されます。



メモ: *製品のインストール中に SSIEnableCmdDirective レジストリ値が変更された場合でも、ターゲット システムから製品がアンインストールされるときに、SSIEnableCmdDirective レジストリ値が変更されることはありません。*

Web サイトの作成とアプリケーションまたは仮想ディレクトリの追加



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield の [IIS 構成] ビューで、プロジェクトに IIS Web サイトを追加します。このビューではまた、Web サイトにアプリケーションおよび仮想ディレクトリを追加することもできます。Web サイトをコンポーネントに関連付けると、アプリケーションまたは仮想ディレクトリを持たない Web サイトを追加することもできます。



タスク ***実行時にターゲット システム上で Web サイトを作成するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーを右クリックして、**[Web サイトの追加]** をクリックします。InstallShield が新しい Web サイトを追加します。
3. 設定を構成する Web サイトを選択します。



ヒント: *InstallShieldでは、プロジェクトで Web サイトの TCP ポートとサイト番号を指定できます。これらの設定は、実行時に新しい Web サイトが作成されるか、既存の Web サイトが更新されるのかを判断するのに役立ちます。詳細については、「TCP ポート番号とサイト番号の構成」を参照してください。*



タスク ***実行時にターゲット システム上でアプリケーションを作成するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、アプリケーションを含める Web サイトを右クリックして、**[新しいアプリケーション]** をクリックします。新しいリリースが追加されます。
3. 設定を構成するアプリケーションを選択します。



タスク ***実行時にターゲット システム上で仮想ディレクトリを作成するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、ア仮想ディレクトリを含める Web サイトを右クリックして、**[新しい仮想ディレクトリ]** をクリックします。新しい仮想ディレクトリが追加されます。
3. 設定を構成する仮想ディレクトリを選択します。



メモ: *Web サイト、アプリケーション、および仮想ディレクトリをコンポーネントと機能に関連付ける方法については、「IIS サポートの機能とコンポーネントの関連付け」を参照してください。*

IIS Web サイトのスキャンをして、その設定を InstallShield プロジェクトにインポートする



エディション: *IIS データを InstallShield プロジェクトにインポートする機能は、InstallShield Premier Edition のみで使用できます。*



プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield には、既存の IIS Web サイトをスキャンし、その Web サイトについての IIS データを記録して、それらの設定を InstallShield プロジェクトの [IIS 構成] ビューにインポートするサポートが含まれています。データをプロジェクトにインポートした後、必要に応じて IIS 設定を変更して、IIS Web サイトを作成または管理するインストールをビルドすることができます。

IIS スキャナー (IIScan.exe) は、IIS Web サイトをスキャンして、InstallShield の [IIS 構成] ビューで構成可能な設定の値を記録するコマンドライン ツールです。IIScan.exe は、すべての値を含む XML ファイルを作成します。この XML ファイルを使って、[IIS 構成] ビューに値をインポートできます。

IIScan.exe は、以下の場所にインストールされています:

InstallShield Program Files フォルダー¥System¥iisscan.exe



タスク ***既存の Web サイトをスキャンして、[IIS 構成] ビューで構成可能なすべての IIS データを記録するには、以下の手順に従います:***

1. IIScan.exe ツールを、スキャンする IIS Web サイトを持つサーバー上に配置します。
2. コマンドライン プロンプトで、特定の Web サイトをスキャンする IIS スキャナーを起動するコマンドライン ステートメントを入力します。次は、サンプル ステートメントです:

   iisscan.exe –website "Site Name" –outfile "C:¥PathToFile¥FileName.xml"

   IIScan.exe に関する詳細については、「IIScan.exe」を参照してください。

IIS スキャナーは、指定の Web サイトに関する IIS 情報を含む XML ファイルを作成します。



タスク ***XML ファイルからのデータを InstallShield プロジェクトにインポートするには、以下の手順に従います:***

1. IIS スキャナーが作成した XML ファイルを取得して、それを InstallShield を持つマシンからアクセス可能な場所に配置します。
2. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
3. **[Web サイト]** エクスプローラーを右クリックして、**[Web サイトのインポート]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. IIS スキャナーが作成した XML ファイルを選択してから、**[開く]** をクリックします。

InstallShield が Web サイトとその設定を [IIS 構成] ビューにインポートします。インポート中に Web サイトの設定、仮想ディレクトリ、およびアプリケーションすべてがこのビューで構成されます。



ヒント: *InstallShield では、特定の IIS データ (たとえば Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、またはそれらの設定) が [IIS 構成] ビューにインポートされないように防止するためのフィルターを構成できます。詳細については、「Web サイトおよびその設定を InstallShield プロジェクトにインポートするときに IIS データをフィルターする」を参照してください。*

Web サイトおよびその設定を InstallShield プロジェクトにインポートするときに IIS データをフィルターする



エディション: *IIS データを InstallShield プロジェクトにインポートする機能は、InstallShield Premier Edition のみで使用できます。*



プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

IIS スキャナーが生成した XML ファイルを使って IIS Web サイトとその設定を [IIS 構成] ビューにインポートするとき、インストールで構成する必要がない IIS 設定もインポートされる場合があります。XML ファイルを使って IIS 設定をインポートするたびにこれらの設定が構成されないようにするためには、Filters.xml を編集します。このファイルを使って、インポート時に無視する任意の IIS 設定を指定できます。

Filters.xml は、以下の場所にあります:

InstallShield Program Files フォルダー¥Support

このファイルは編集後も同じ場所になくてはなりません。そうでない場合、IIS インポートが正しく動作しません。



ヒント: Filters.xml *ファイルを使って、COM 抽出中にどのレジストリ項目を除外するのかを制御することもできます。詳細については、「COM 抽出のレジストリ変更をフィルターする」を参照してください。*

Filters.xml *ファイルを使って、依存関係のスキャン中にどのファイルを除外するまたは含めるのかを制御することもできます。詳細については、「依存関係スキャナーでファイルをフィルターする」を参照してください。*

IIS データの除外

Filters.xml ファイルの <Exclude> 要素では、インポートしない各 IIS アイテムのサブ要素 (Web サイト、アプリケーション、仮想ディレクトリ、およびアプリケーション プール) を追加します。また、インポートしない各 IIS 設定のサブ要素を追加することもできます。ここでリストされたアイテムと設定は、プロジェクトの [IIS 構成] ビューには追加されません。

<Exclude> 要素における IIS データ指定の例

TestAppPool と名付けられたアプリケーション プールが [IIS 構成] ビューにインポートされないように防ぐには、<Exclude> 内に以下の行を追加します。

```
<IisScanItem name="TestAppPool" type="4"/>
```

type は必須です。次のテーブルは、使用可能な IIS アイテムの種類の一覧です:

テーブル 3 ・ <IisScanItem> サブ要素の type 属性のとして使用可能な属性

属性値	説明
1	Web サイト
2	アプリケーション
3	仮想ディレクトリ
4	アプリケーション プール

指定の値を持つ IIS 設定が [IIS 構成] ビューにインポートされないように防ぐには、<Exclude> 内に以下の行を追加します。

```
<IisScanProperty name="PropertyName" value="PropertyValue"/>
```

<Exclude> 要素内にこの行があると、InstallShield は **PropertyValue** と等しい、またはこれを含む値を持つ **PropertyName** と名付けられた設定をブロックします。



メモ・ *プロパティ名と値は Windows Installer プロパティまたはその値ではありません。これらは、InstallShield の [IIS 構成] ビューで構成可能な設定の名前と値の省略形です。*



重要・ *その場合、XML コードが適切に書かれていることを再確認してください。不適切な場合には、すべてのフィルターが失敗します。多くの場合、インターネット エクスプローラーで* Filters.xml *ファイルを開いて、不適切に書かれた XML コードを確認することができます。<Filters>、<Include>、および <Exclude> 要素は縮小および展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。*

<Exclude> または <Include> 要素にサブ要素を追加した場合、それらが誤ってコメントアウトされたセクションに配置されていないことを確認してください。InstallShield は Filters.xml *ファイルのコメントアウト部分を無視します。*

次のサンプル XML コードで、Filters.xml ファイルの形式を説明します。

```
<Filters>
 <Include>
  <!—この要素にファイルを追加する方法
  —>
  <File name="mfc42.dll" We="needthis"/>
 <Include>
 <Exclude>
  <!—この要素にファイルを追加する方法
  —>
```

```
 <Registry key="HKEY_CLASSES_ROOT¥Interface¥{00020404-0000-0000-C000-000000000046}"/>
 <File name="12520437.cpx" path="[SystemFolder]" wrp="4.0-10.0" />
 <File name="12520850.cpx" path="[SystemFolder]" wrp="4.0-10.0" />
 <IisScanProperty name="AppMappings" value="*;;*;0" />
 <IisScanProperty name="CustomErrors" value="C:¥WINDOWS¥help¥iisHelp¥common" />
</Exclude>
</Filters>
```

ネスト仮想ディレクトリの作成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

既存の仮想ディレクトリの下に、仮想サブディレクトリを作成することができます。

また、インストールの一部としてインストールされる仮想ディレクトリの下にも仮想サブディレクトリを作成することができます。親仮想ディレクトリは、仮想サブディレクトリの前にインストールされる必要があります。



プロジェクト: *InstallScript の場合、親仮想ディレクトリが必ず子仮想ディレクトリの前にインストールされるようにするには、親仮想ディレクトリを含むコンポーネントが、子仮想ディレクトリを含むコンポーネントの前にインストールされる必要があります。*



タスク ***既存の仮想ディレクトリの下に仮想ディレクトリを作成するには、以下の手順を実行します。***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、ネスト仮想ディレクトリを含める Web サイトを選択します。
3. 新しい Web サイトを右クリックして **[新しい仮想ディレクトリ]** を選択します。新しい仮想ディレクトリが追加されます。
4. **[全般]** タブをクリックします。
5. **"名前"** 設定で、既存のディレクトリ名と、作成するネスト仮想サブディレクトリの名前を指定します。2 つの名前はスラッシュで区切ります。

   たとえば、**VirtualDirectory** という名前の既存の仮想ディレクトリの下に **MySubdirectory** という名前の仮想ディレクトリを作成する場合、次のように入力します:

   VirtualDirectory/MySubdirectory



メモ: *親ディレクトリがターゲット システムに既に存在しない場合、エンドユーザーが IIS マネージャーでそのディレクトリを開いたとき、ターゲット システムでエラーが表示されます。*

TCP ポート番号とサイト番号の構成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShieldでは、プロジェクトで Web サイトの TCP ポートとサイト番号を指定できます。これらの設定は、実行時に新しい Web サイトが作成されるか、既存の Web サイトが更新されるのかを判断するのに役立ちます。これらはまた、[インターネット インフォーメーション サービス] ビューで構成された Web サイトの設定が、ターゲット システムの Web サイトに適用されるかどうかにも影響します。



タスク

Web サイトに TCP ポート番号とサイト番号を指定するには、以下の手順に従います:

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、構成する Web サイトを選択します。
3. **[Web サイト]** タブをクリックします。
4. **[TCP ポート番号]** ボックスと **[サイト番号]** ボックスに、適切な番号を入力します。

実行時の動作

実行時に、インストールが次の規則に基づいて Web サイト、アプリケーション、および仮想ディレクトリを作成します:

テーブル 4 · 各種サンプル "TCP ポート番号" および "サイト番号" 設定値の実行時の結果

InstallShield の "TCP ポート番号" 設定	InstallShield の "サイト番号" 設定	実行時の結果
0	ゼロ以外の値	"TCP ポート番号" 設定が 0 で、"サイト番号" 設定が 0 以外のとき、インストールは Web サイトのアプリケーションおよび仮想ディレクトリをシステムの最初のサイト番号にインストールします。指定されたサイト番号は無視されます。 たとえば、ターゲット システム上の最初のサイト番号が 1 のとき、"サイト番号" 設定に異なるゼロ以外の番号 (たとえば、3) が指定された場合でも、インストールは Web サイトのアプリケーションと仮想ディレクトリをサイト番号 1 にインストールします。 インストールは、[IIS 構成] ビューで構成された Web サイト設定を、ターゲット システム上の Web サイトに一切適用しません。

テーブル 4 - 各種サンプル "TCP ポート番号" および "サイト番号" 設定値の実行時の結果 (続き.)

InstallShield の "TCP ポート番号" 設定	InstallShield の "サイト番号" 設定	実行時の結果
80 (ゼロ以外の値)	0 (デフォルト値)	指定された TCP ポートがターゲット システムに存在するとき、インストールは TCP ポートで実行中の Web サイト (この例では、ポート 80) にアプリケーションと仮想ディレクトリをインストールします。インストールは、[IIS 構成] ビューで構成された Web サイト設定を、ターゲット システム上の Web サイトに一切適用しません。 TCP ポートがターゲット システムに存在しない場合、プロジェクトで構成された Web サイトの設定を使って新しい Web サイトが作成されます。またインストールは、Web サイトのアプリケーションと仮想ディレクトリをインストールします。
81 (ゼロ以外の値)	3 (ゼロ以外の値)	指定された TCP ポートおよびサイト番号がターゲット システムに存在するとき、インストールは TCP ポートで実行中の Web サイト (この例では、ポート 3) にアプリケーションと仮想ディレクトリをインストールします。インストールは、[IIS 構成] ビューで構成された Web サイト設定を、ターゲット システム上の Web サイトに一切適用しません。 TCP ポートがターゲット システムに存在するが、サイト番号が存在しない場合、インストールは新しい Web サイトおよびそのアプリケーションと仮想ディレクトリを、既存のポートに新しいサイト番号を使ってインストールします。またインストールは、[IIS 構成] ビューで設定された Web サイトのプロパティを構成します。 TCP ポートがターゲット システムに存在しないが、サイト番号が存在する場合、インストールは新しい Web サイトおよびそのアプリケーションと仮想ディレクトリをこの TCP ポートにインストールします。またインストールは、[IIS 構成] ビューで設定された Web サイトのプロパティを構成します。

IIS Web サイトとそのアプリケーションおよび仮想ディレクトリを、次に利用できる新しいサイト番号にインストールする

次の手順は、IIS Web サイトとそのアプリケーション、および仮想ディレクトリを、次の利用できる新しいサイト番号にインストールする方法を説明します。手続きのは、プロジェクトの種類によって異なりますので注意してください。



タスク ***基本の MSI、DIM、InstallScript MSI、またはマージ モジュール プロジェクトで、次に利用できる新しいサイト番号にインストールするには、以下の手順に従います:***

1. 新しい Windows Installer プロパティを作成して、その値を **INSTALLSHIELD_IIS_NEXT_NEW_SITE_NUMBER** に設定します:
   a. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
   b. **[新しいプロパティ]** ボタンをクリックします。ビューの下に新しい行が追加されます。
   c. **[名前]** 列に、新しいプロパティを入力します。例:

      MYPROPERTY

   d. **[値]** 列で、次の値を入力します:

      INSTALLSHIELD_IIS_NEXT_NEW_SITE_NUMBER
2. Web サイトのサイト番号のプロパティを指定します:
   a. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
   b. **[Web サイト]** エクスプローラーで、構成する Web サイトを選択します。
   c. **"サイト番号"** 設定に、ステップ 1c で作成したプロパティを入力します。プロパティはすべて大文字で、角括弧で囲みます。例:

      [MYPROPERTY]

実行時、インストールは Web サイトとそのアプリケーション、および仮想ディレクトリを次に利用できる新しいサイト番号にインストールします。



タスク ***InstallScript インストール中、次に使用できる新しいサイト番号にインストールするには、以下の手順に従います:***

1. 新しい文字列エントリを作成して、その値を INSTALLSHIELD_IIS_NEXT_NEW_SITE_NUMBER に設定します:
   a. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
   b. **[新しい文字列エントリ]** ボタンをクリックします。**[文字列エントリ]** ダイアログ ボックスが開きます。
   c. **[文字列識別子]** ボックスに、新しい変数名を入力します。変数名はすべて大文字でなくてはなりません。例:

      MYPROPERTY

   d. **[値]** ボックスに、次の値を入力します:

      INSTALLSHIELD_IIS_NEXT_NEW_SITE_NUMBER
2. Web サイトのサイト番号の変数を指定します:
   a. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
   b. **[Web サイト]** エクスプローラーで、構成する Web サイトを選択します。
   c. **"サイト番号"** 設定に、ステップ 1c で作成した変数を入力します。すべて大文字でなくてはなりません。

実行時、インストールは Web サイトとそのアプリケーション、および仮想ディレクトリを次に利用できる新しいサイト番号にインストールします。

Web サイトの IIS ホスト ヘッダー名を指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield では、ホスト ヘッダー名を指定して、インストール中にインストールされる IIS Web サイトを識別することができます。ホスト ヘッダー (ドメイン名とも呼ばれます) を利用して、複数の Web サイトを Web サーバー上の IP アドレスに割り当てることができます。



タスク ***Web サイトのホスト ヘッダー名を指定するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、ホスト ヘッダー名を指定する Web サイトを選択します。
3. **"ホスト ヘッダー名"** 設定で、使用するホスト ヘッダー名を入力します。例:

   www.mycompany.com

Web サイトの SSL 証明書を指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

サーバー証明書を利用して、ユーザーは Web サーバーの認証および Web コンテンツの有効性の確認を行うことができると共に、セキュリティで保護された接続を確立することができます。では、実行時にインストールできるように、インストールに Web サイトのサーバー証明書を含めることができます。

タスク ***Web サイトにインストールする SSL 証明書を指定するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、SSL 証明書を指定する Web サイトを選択します。
3. **"SSL 証明書"** 設定で、省略記号ボタン (...) をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. インストールするセキュリティ証明書ファイル (.cer または .pfx) を選択して、**[開く]** をクリックします。
5. 証明書にパスワードが必要な場合、**"SSL 証明書パスワード"** 設定でパスワードを指定します。

.cer ファイルが Binary テーブルに格納されます。実行時、インストールで Web サイトと仮想ディレクトリがインストールされるとき、SSL 証明書もインストールされます。

実行時に IIS 証明書ファイルを設定する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

エンド ユーザーが SSL 証明書に提供する IIS 証明書ファイルを参照し、インストール実行時にパスワードを入力できる、**[IIS の SSL を構成する]** ダイアログ (IISBrowseSSLCertificate) をインストーラーに含めるかどうかを選択できます。

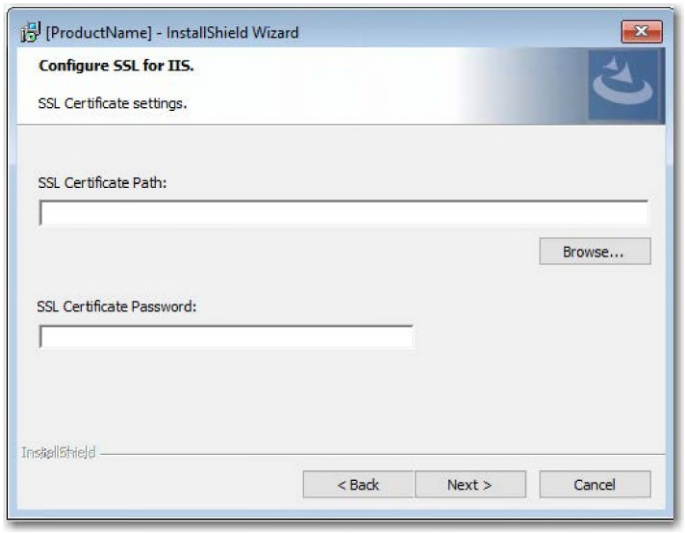


図 1: IIS ダイアログの SSL を構成する

インストーラーに **[IIS の SSL を構成する]** ダイアログ を追加するには、次の手順に従います:



タスク ***インストーラーに [IIS の SSL を構成] ダイアログ を追加するには:***

1. **[サーバー構成]** の下の **ビュー リスト** にある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーを右クリックして、**[Web サイトの追加]** をクリックします。InstallShield が新しい Web サイトを追加します。
3. 新しい Web サイトを選択して **[セキュリティ > セキュリティで保護された通信]** の下にある **"SSL 証明書"** および **"SSL 証明書パスワード"** プロパティに移動します。
4. **"SSL 証明書"** および **"SSL 証明書パスワード"** プロパティを次の値に設定します:

プロパティ	値
SSL 証明書	[IS_IIS_WEBCERTPATH]
SSL 証明書のパスワード	[IS_IIS_WEBCERTPASSWORD]

5. **[ユーザー インターフェイス > ダイアログ]** ビューを開いて、ダイアログ シーケンスに **IISBrowseSSLCertificate** ダイアログを追加します。

   [編集] ボックス (**IISWebCertPassword** と **IISWebCertPath**) およびプッシュボタン (**BrowseCertificate**) イベントの **IISBrowseSSLCertificate** ダイアログで、ユーザーが構成した SSL 証明書およびパスワードのプロパティ名がプロパティ名を更新する必要があります。

ファイルを IIS 仮想ディレクトリに追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*



タスク ***ファイルを IIS 仮想ディレクトリに追加する:***

1. IIS Web サイトをプロジェクトに追加していない場合は、それを行います。InstallShield が自動的に定義済みのパス **[IISROOTFOLDER]** を **[ファイルとフォルダー]** ビューへ追加します
2. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
3. **[機能]** リストで、ファイルに関連付ける機能を選択します。
4. **[インストール先コンピューターのフォルダー]** ペインで、ファイルを **[IISROOTFOLDER]** フォルダー、または **[IISROOTFOLDER]** フォルダーのサブフォルダーに追加します。
5. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。

6. 新しい仮想ディレクトリを作成します。
7. **[Web サイト]** エクスプローラーで、作成した仮想ディレクトリをクリックします。
8. **"コンテンツ ソース パス (ローカルまたは UNC)"** 設定で省略記号ボタン (...) をクリックします。**[ディレクトリの参照]** ダイアログ ボックスが開きます。基本の MSI または InstallScript MSI プロジェクトでは、このダイアログ ボックスで Windows Installer プロパティ (たとえば、[IISROOTFOLDER]) を選択するか、または新しいプロパティを作成できます。InstallScript プロジェクトでは、このダイアログ ボックスで InstallScript 変数 (たとえば、<IISROOTFOLDER>) を選択するか、または新しい変数を作成できます。デフォルトでは、これらのファイルは IISROOTFOLDER に格納されています。
9. **[ファイルとフォルダー]** ビューで追加した新しいファイルを含むディレクトリと同じターゲット ディレクトリを入力します。
10. **[OK]** をクリックします。

インストール時、ファイルはターゲット ディレクトリ フォルダーへコピーされます。さらに IIS が存在する場合、ターゲット システム上のフォルダーに仮想ディレクトリが構成されます。

[IIS 構成] ビューからアプリケーションと仮想ディレクトリを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*



タスク ***アプリケーションまたは仮想ディレクトリをインストールから削除するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、アプリケーションまたは仮想ディレクトリを右クリックしてから **[削除]** をクリックします。

アプリケーション プールの追加



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

アプリケーション プールは、1 つ以上の Web アプリケーションを 1 つ以上のワーカー プロセスにルートするときの方法を指定するひとまとまりの構成設定です。InstallShield では、アプリケーション プールを作成して、Web サイトおよび仮想ディレクトリをその作成したアプリケーション プールに関連付けることができます。



プロジェクト- *InstallScript プロジェクトでアプリケーション プールを作成する場合で、アプリケーション プールに仮想ディレクトリ、アプリケーション、または Web サイトに関連付けるには、同じコンポーネント内でアプリケーション プールと仮想ディレクトリ、アプリケーション、または Web サイトを作成しなくてはなりません。これを行うことで、仮想ディレクトリ、アプリケーション、または Web サイトが作成される前にアプリケーション プールが作成されるようになります。これは、IIS 6 以降の要件で、基本の MSI、DIM、InstallScript MSI、またはマージ モジュール プロジェクトの制限事項ではありません。*



メモ- *アプリケーション プールは、IIS 6.0 以降がインストールされているマシンでのみ利用可能です。*



タスク ***[IIS 構成] ビューで新しいアプリケーション プールを追加するには、以下の手順を実行します。***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[アプリケーション プール]** エクスプローラーを右クリックして、**[アプリケーション プールの追加]** をクリックします。InstallShield が新しいアプリケーション プール項目を追加します。
3. 新しいアプリケーション プール アイテムの名前を変更して、その設定を構成します。

[IIS 構成] ビューからアプリケーション プールを削除する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*



タスク ***アプリケーション プールをインストールから削除するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[アプリケーション プール]** エクスプローラーで、アプリケーション プールを右クリックし **[削除]** をクリックします。

Web サービス拡張の追加



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

Web サービス拡張は、基本の IIS 機能をさらに拡張して、静的コンテンツの提供以上のことをできるようにする IIS 機能です。Web サービス拡張の例として、アクティブ サーバー ページ (.asp)、ASP.NET、およびサーバー側インクルード (SSI) などがあります。

InstallShield では、Web サービス拡張をインストールに追加することができます。



メモ: *サービス拡張は、IIS 6.0 以降がインストールされているマシンでのみ利用可能です。*

IIS 7 があるシステムでは、Web サービス拡張に、[IIS メタベースと IIS 6 構成の互換性] 機能がインストールされている必要があります。詳細については、「InstallShield における IIS サポートのバージョン固有情報」を参照してください。



タスク

Web サービス拡張をインストールに追加するには、以下の手順に従います:

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サービス拡張]** エクスプローラーを右クリックして、**[Web サービス拡張の追加]** をクリックします。InstallShield が Web サービス拡張を追加します。
3. 新しいWeb サービス拡張アイテムの名前を変更して、その設定を構成します。

[IIS 構成] ビューから Web サービス拡張を削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*



タスク ***Web サービス拡張をインストールから削除するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サービス拡張]** エクスプローラーで、Web サービス拡張を右クリックして、**[削除]** をクリックします。

IIS サポートの機能とコンポーネントの関連付け



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

コンポーネントと IIS データ (Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、または Web サービス拡張) は、一対多の関係を持ちます。したがって、1 つのコンポーネントに複数の IIS 要素を関連付けることが可能です。コンポーネントがインストールされている間、コンポーネントに関連付けられているすべての Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プールおよび Web サービス拡張が実行時に作成されます。



プロジェクト: *InstallScript プロジェクトで Web サイトがコンポーネントに関連付けられている場合、その Web サイトに追加されるアプリケーションまたは仮想ディレクトリも同じコンポーネントに関連付けなくてはなりません。したがって、1 つまたは複数のアプリケーション、または仮想ディレクトリを含む Web サイトのコンポーネントを変更しようとすると、InstallShield はメッセージ ボックスを表示して、Web サイトのアプリケーションおよび仮想ディレクトリのすべてに対して、同じコンポーネントの変更が行われることを通知します。このメッセージ ボックスは、Web サイトに含まれる任意のアプリケーションまたは仮想ディレクトリのコンポーネントを変更しようとした場合にも表示されます。どちらの場合も、メッセージ ボックスはコンポーネントの変更を続行するか、キャンセルするか選択肢を提供します。*

必須ではありませんが、基本の MSI、DIM、InstallScript MSI、およびマージ モジュール プロジェクトでは、Web サイトのすべてのアプリケーションおよび仮想ディレクトリと同様に、Web サイトを同じコンポーネントに保持することが推奨されます。

InstallScript プロジェクトでアプリケーション プールを作成する場合で、アプリケーション プールに仮想ディレクトリ、アプリケーション、または Web サイトに関連付けるには、同じコンポーネント内でアプリケーション プールと仮想ディレクトリ、アプリケーション、または Web サイトを作成しなくてはなりません。これを行うことで、仮想ディレクトリ、アプリケーション、または Web サイトが作成される前にアプリケーションプールが作成されるようになります。これは、IIS 6 以降の要件で、基本の MSI、DIM、InstallScript MSI、またはマージ モジュール プロジェクトの制限事項ではありません。

Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プールまたは Web サービス拡張を含めるコンポーネントは、必要に応じていつでも [IIS 構成] ビューで変更することができます。

タスク

Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、または Web サービス拡張をプロジェクト内の異なるコンポーネントに関連付けるには、以下の手順に従います。

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. エクスプローラーで、コンポーネントに関連付ける Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、または Web サービス拡張をクリックします。
3. **[コンポーネント]** 設定で、選択した IIS データを含める既存のコンポーネントの名前を選択します。省略記号ボタン (...) ボタンをクリックして、既存のコンポーネントを選択することもできますし、新しいコンポーネントをこの IIS データに作成することもできます。

ヒント: *プロジェクトからコンポーネントを削除した場合、コンポーネントに関連付けられたすべての Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張もプロジェクトから削除されます。*

Web サイト、アプリケーション、および仮想ディレクトリのアンインストール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

インストールによって作成された Web サイトは、次の条件の両方が True とならない限り、削除されません:

- Web サイトにアプリケーションまたは仮想ディレクトリが含まれていない。
- [IIS 構成] ビューにある Web サイトの ***"アンインストール時に削除する"*** 設定の値が [はい] である。

メモ: *Web サイトがコンポーネントと関連付けられている場合、Web サイトの* ***"アンインストール時に削除する"*** *設定は、そのコンポーネントの "パーマネント" 設定 (基本の MSI、DIM、InstallScript MSI、またはマージ モジュール プロジェクトの場合) またはそのコンポーネントの "アンインストール" 設定 (InstallScript プロジェクトの場合) に対応します。つまり、Web サイトの* ***"アンインストール時に 削除する"*** *設定を選択またはクリアすると、自動的にコンポーネントの "パーマネント" 設定または "アンインストール" 設定が適切に更新されます。*

コンポーネントがアンインストールされると、そのすべての Web サイト、アプリケーション、および仮想ディレクトリもアンインストールされます。パーマネントと構成されている場合、コンポーネントはアンインストールされません。詳細については、「アンインストール時にコンポーネントのファイルと他の関連データをアンインストールするかどうかを指定する」を参照してください。

Web サービス拡張とアプリケーション プールのアンインストール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*



メモ: *サービス拡張、アプリケーション プール、および関連プロパティは、IIS 6.0 以降がインストールされているマシンでのみ利用可能です。*

IIS 7 があるシステムでは、Web サービス拡張に、[IIS メタベースと IIS 6 構成の互換性] 機能がインストールされている必要があります。詳細については、「InstallShield における IIS サポートのバージョン固有情報」を参照してください。

インストール内の Web サービス拡張およびアプリケーション プールと同じ名前を持つ、IIS Manager にインストールされている Web サービス拡張およびアプリケーション プールはどれも、それらのコンポーネントがパーマネントとマークされていない限り、関連付けられている機能がアンインストールされるときに必ずアンインストールされます。Web サービス拡張またはアプリケーション プールがインストール以外によって作成された場合でも、(Web サービス拡張およびアプリケーション プールの) 名前がインストールに含まれるものと一致する場合は、アンインストール時にターゲット システムから削除されます。唯一の例外は DefaultAppPool というアプリケーション プールで、これはアンインストール時に削除されることはありません。



タスク

アンインストール中に Web サービス拡張をアンインストールするように構成するには、次の手順を実行します。

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サービス拡張]** エクスプローラーで、Web サービス拡張を選択します。
3. **"コンポーネントをパーマネントとしてマークする"** 設定で、**[いいえ]** を選択します。



タスク

アンインストール中にアプリケーション プールをアンインストールするように構成するには、次の手順を実行します。

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[アプリケーション プール]** エクスプローラーで、アプリケーション プールを選択します。
3. **"コンポーネントをパーマネントとしてマークする"** 設定で、**[いいえ]** を選択します。

"コンポーネントをパーマネントとしてマークする" 設定で [はい] が選択されていと、Web サービス拡張またはアプリケーション プールはアンインストールの実行後もターゲット マシンに残ります。



ヒント: *"コンポーネントをパーマネントとしてマークする" 設定の値を変更すると、Web サービス拡張またはアプリケーション プールを含むコンポーネントの "パーマネント" 設定 (基本の MSI、DIM、InstallScript MSI、およ*

びマージ モジュール プロジェクトの場合)、または "アンインストール" 設定 (InstallScript プロジェクトの場合) の値に反映します。したがって、"コンポーネントをパーマネントとしてマークする" 設定で [はい] を選択して、かつコンポーネントにファイル、ショートカット、レジストリ エントリなどの他のデータが含まれている場合、IIS データおよびコンポーネントの他のデータはアンインストール時にアンインストールされません。

Web サイトまたはアプリケーションの ASP.NET バージョンを設定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield では、インストールにある Web サイトまたはアプリケーションの ASP.NET バージョンを設定することができます。ASP.NET バージョンを指定すると、Web サイトまたはアプリケーションが作成された後、ASP.NET IIS の Registration Tool (Aspnet_regiis.exe) がインストールで実行され、Web サイトまたはアプリケーションを指定したバージョンにマッピングします。

Web サイトに ASP.NET バージョンを指定すると、IIS はその値を実行時に作成された Web サイトおよびすべてのそのアプリケーションに使用します。



重要: *マイクロソフト社は、Aspnet_regiis.exe ツールの機能に制限があるため、Windows Vista または Windows Server 2008 システムでの使用を推奨していません。結果として、場合により、アプリケーションのマッピングを [IIS 構成] ビューで手動で定義する必要があります。詳細については、「Web サイト、アプリケーション、または仮想ディレクトリのアプリケーションのマッピングを定義する」を参照してください。*

ASP.NET 3.0 には Aspnet_regiis.exe ツールは含まれていません。したがって、ASP.NET バージョンを バージョン 3 の ASP.NET に設定できません。



タスク

プロジェクトで Web サイトまたはアプリケーションの ASP.NET バージョンを指定するには、以下の手順に従います:

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、ASP.NET バージョンを指定する Web サイトまたはアプリケーションを選択します。Web サイトまたはアプリケーションの設定が右側に表示されます。
3. **"ASP .NET バージョン"** 設定で、アプリケーションに必要な .NET Framework のバージョン番号を完全な形で入力するか、一覧から選択します。

   たとえば、バージョン 2 の ASP.NET を指定する場合、**2.0.50727** と入力します。バージョン 1.1 の ASP.NET を指定するには、**1.1.4322** と入力します。



ヒント: *.NET Framework を持つ Windows の 64 ビット バージョンでインストールが実行可能な場合、Web サイトまたはアプリケーションの "ASP.NET プラットフォーム" 設定を使って、どの ASP.NET プラットフォーム (32 ビットまたは 64 ビット) を使って ASP.NET バージョンに Web サイトまたはアプリケーションをマップするのかを指定できます。システム上で Enable32BitAppOnWin64 プロパティが True に設定されていない限り、64 ビット システム上で 32 ビット ASP.NET を登録することはできません。*

繰り返しまたは 1 回限りの ASP.NET 登録を行うかどうかを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

[IIS 構成] ビューの **"アプリケーション"** 設定にある **[ASP.NET 登録]** 設定を使って、繰り返しまたは 1 回限りの ASP .NET 登録を行うかどうかを指定できます。この機能を使って、同じ Web サイトに ASP.NET アプリケーションおよび ASP.NET コア アプリケーションの両方をインストールすることができます。



タスク ***プロジェクトでアプリケーションの ASP.NET 登録を指定するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、ASP.NET 登録を指定する アプリケーションを選択します。アプリケーションの設定が右側に表示されます。
3. **"ASP .NET 登録"** 設定で次のオプションから 1 つを選択します:
   - **再帰的** – 指定されたアプリケーションおよびすべてのサブアプリケーションのスクリプト マップとアプリケーション プールの割り当てを更新します。
   - **非再帰的** – 特定のアプリケーションのスクリプト マップとアプリケーション プールの割り当てを更新します。サブアプリケーションは変更されません。

64 ビット プラットフォーム上で IIS 6 をサポートする際の考慮



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

IIS インストールが .NET Framework を持つ Windows の 64 ビット バージョンで実行可能な場合、Web サイトまたはアプリケーションに [IIS 構成] ビューの "ASP.NET プラットフォーム" 設定を使用できます。この設定を使って、どの ASP.NET プラットフォームを使って Web サイトまたはアプリケーションを ASP.NET バージョンにマップするのかを指定できます。デフォルトで、この設定は構成されていません。この設定の値を変更しなかった場合、インストールは 32 ビット サポートが不要であると判断します。

"ASP.NET プラットフォーム" と "ASP.NET バージョン" 設定で構成した値によって、デフォルトで実行時に以下の動作が発生します:

- ターゲット システムが Windows 32 ビット バージョンを持つ場合、指定された ASP.NET の 32 ビットバージョンが使用されます。
- ターゲット システムが Windows 64 ビット バージョンを持つ場合で、ASP.NET プラットフォームに 64 ビットが指定されるとき、ターゲット システム上の Enable32bitAppOnWin64 プロパティは False でなくてはなりません。そうでない場合、32 ビット バージョンをサポートするように構成されているシステム上で ASP.NET の 64 ビット バージョンを使用すると、ASP.NET が破損する原因となります。このため、ASP.NET サービスの破損を避けるためにインストールは中止されます。
- ターゲット システムが Windows 64 ビット バージョンを持つ場合で、ASP.NET プラットフォームに 32 ビットが指定されるとき、ターゲット システム上の Enable32bitAppOnWin64 プロパティは True でなくてはなりません。そうでない場合、64 ビット バージョンをサポートするように構成されているシステム上で ASP.NET の 32 ビット バージョンを使用すると、ASP.NET が破損する原因となります。このため、ASP.NET サービスの破損を避けるためにインストールは中止されます。

[IIS 構成] ビューでアプリケーション プールの "32 ビット アプリケーションを有効にする" 設定を構成して、IIS 7 が搭載されたターゲット システム上の Enable32bitAppOnWin64 プロパティを設定できます。実行時、インストールはインストールするアプリケーション プールに適切な Enable32bitAppOnWin64 プロパティを設定します。

IIS 6 が搭載されたシステム上では、Enable32bitAppOnWin64 プロパティは IIS サーバー上のすべての Web サイトとアプリケーションに反映するグローバル プロパティです。インストール実行時にプロパティの値を変更すると、そのサーバー上にインストール済みの Web サイトとアプリケーションが失敗する原因となるため、InstallShield で作成されたインストールは、ターゲット システムに IIS 6 が搭載されている場合、実行時に Enable32bitAppOnWin64 プロパティを変更しません。

実行時、IIS 6 を持つシステム上でインストールが Enable32bitAppOnWin64 プロパティをチェックするように設定できます。そのとき、製品の要件およびチェック結果に基づいて、たとえば 32 ビット IIS 6 アプリケーションまたは 64 ビット IIS アプリケーションを含む特定のコンポーネントのインストールをスキップして、残りのファイル転送を続行できます。

InstallShield には、Enable32bitAppOnWin64 プロパティがどのように設定されているかを検出できるサンプル Windows Installer DLL ファイルが含まれています。DLL ファイルのリリースおよびデバッグ バージョン、およびファイルの作成に使用された Visual Studio 2008 C++ プロジェクトは以下の場所にインストールされています:

InstallShield Program Files フォルダー¥Samples¥WindowsInstaller¥Detect IIS6 Compatibility

プロジェクトで Windows Installer DLL ファイルを使用するためには、まず DetectIIS6Compat.dll ファイルで **DetectIIS6AppPool32BitSupport** 関数を呼び出すカスタム アクションを作成する必要があります。



タスク

IIS 6 が搭載されているシステム上の 32 ビット アプリケーションのサポートをチェックするカスタム アクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI の場合) または **[カスタム アクション]** (DIM、マージ モジュール プロジェクトの場合) をクリックします。
2. 中央のペインで、**[カスタム アクション]** エクスプローラーを右クリックして、**[新しい MSI DLL]** をポイントしてから、**[Binary テーブルに保存]** をクリックします。新しいカスタム アクションが追加されます。
3. アクションの名前を、たとえば **DetectIIS6AppPool32BitSupport** に変更します。
4. 右側のペインでカスタム アクションの設定を構成します:
   - **"DLL ファイル名"** 設定で、DetectIIS6Compat.dll ファイルへのパスを指定します。通常のパスは以下のとおりです:

     <ISProductFolder>¥Samples¥WindowsInstaller¥Detect IIS6 Compatibility¥DetectIIS6Compat.dll
   - "**関数名**" 設定で、次の名前を入力します:

     DetectIIS6AppPool32BitSupport
   - **"スクリプト内実行"** 設定で、**[即時実行]** を選択します。
   - 1 つまたは複数のシーケンス設定で適切なオプションを選択して、必要に応じて、カスタム アクションをスケジュールします。
   - "条件" 設定に次の条件を入力します:

     VersionNT<600

     Windows Server 2008 以降または Windows Vista 以降は IIS 7 以降が搭載されているため、この条件は、カスタム アクションがこれらのオペレーティング システムで実行されないように保証します。

実行時、特定の条件下でカスタム アクションは Windows Installer プロパティ ISIIS6APPPOOLSUPPORTS32BIT を設定します。

テーブル 5 ・ IIS 6 システム上で 32 ビット アプリケーションのサポートをチェックする Windows Installer プロパティ

ターゲット システム環境	結果
32 ビット Windows、IIS 6 (Enable32bitAppOnWin64 の値は関係ありません。)	ISIIS6APPPOOLSUPPORTS32BIT は 1 に設定されます。
64 ビット Windows、IIS 6、Enable32bitAppOnWin64=true	ISIIS6APPPOOLSUPPORTS32BIT は 1 に設定されます。
64 ビット Windows, IIS 6、Enable32bitAppOnWin64=false	ISIIS6APPPOOLSUPPORTS32BIT は設定されません。

条件ステートメントで ISIIS6APPPOOLSUPPORTS32BIT プロパティを使用できます。たとえば、64 ビット アプリケーションをサポートしない IIS 6 サーバー上で製品の使用が不可能な場合、[一般情報] ビューの "インストール条件" 設定に次の条件ステートメントを入力します:

Not ISIIS6APPPOOLSUPPORTS32BIT

別の方法として、エラー カスタム アクションを作成することもできます。実行時、そのカスタム アクションの条件が True の場合、エラーが表示されます。したがって、64 ビット アプリケーションをサポートしない IIS 6 サーバーで製品の使用が不可能な場合、エラー カスタム アクションに次の条件を使用します:

ISIIS6APPPOOLSUPPORTS32BIT=1

32 ビット アプリケーション サポートが有効化されている IIS 6 サーバー上にのみコンポーネントをインストールする場合、[コンポーネント] ビューの "条件" 設定に次の条件ステートメントを入力します:

ISIIS6APPPOOLSUPPORTS32BIT=1

Web サイト、アプリケーション、または仮想ディレクトリのアプリケーションのマッピングを定義する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield では、ファイル名拡張子とこれらのファイルを処理するアプリケーション間のマッピングを定義することができます。



タスク ***アプリケーションのマッピングの追加/編集/削除を行うには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、構成する Web サイト、アプリケーション、または仮想ディレクトリを選択します。
3. **"アプリケーションのマッピング"** 設定で、省略記号ボタン (...) をクリックします。**[アプリケーションのマッピング]** ダイアログ ボックスが開きます。
4. 以下のいずれかを実行します。
   - 新しいマッピングを追加するには、**[追加]** ボタンをクリックします。**[アプリケーション拡張子マッピング]** ダイアログ ボックスが開きます。詳細については、「[アプリケーション拡張子マッピング] ダイアログ ボックス」を参照してください。
   - 既存のマッピングを変更するには、編集するマッピングを選択して、**[編集]** ボタンをクリックします。
   - 既存のマッピングを削除するには、それを選択して、**[削除]** ボタンをクリックします。

Web サイト、アプリケーション、または仮想ディレクトリのアプリケーションのマッピングを定義する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShieldWeb サイト、アプリケーション、または仮想ディレクトリのタイムアウトのパラメーターを指定する



タスク ***タイムアウトのパラメーターを指定するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、構成する Web サイト、アプリケーション、または仮想ディレクトリを選択します。
3. **"アプリケーションの設定"** 領域で、**[構成]** ボタンをクリックします。**[アプリケーションのマッピング]** ダイアログ ボックスが開きます。
4. **"セッション タイムアウト(分)"** と **"ASP スクリプト タイムアウト(秒)"** 設定で、適切なタイムアウト値を指定します。

ターゲット システムに IIS 6 以前が搭載されているか、または IIS 6 メタベースの互換性機能があるかどうかを判別する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

実行時に、インストールによって、ターゲット システムに IIS メタベースおよび IIS 6 構成との互換性機能がインストールされているかどうか、または IIS 以前がインストールされているかを検出できます。製品の要件と検出結果に基づいて、インストールを終了およびエラー メッセージを表示できます。

たとえば、Web サービス拡張は IIS 6 を持つシステムにインストールできます。IIS 7 を持つシステムでは、Web サービス拡張は IIS メタベースおよび IIS 6 構成との互換性機能がインストールされている場合のみインストール可能です。このため、IIS 6 が存在するか、IIS 6 との互換性機能がインストールされていることをインストールが検証するように構成することが必要な場合があります。これらのどちらの条件も False の場合、インストールが終了してエラー メッセージが表示されます。

InstallShield には、以下のどちらかが True であることを検出するためのサンプル Windows Installer DLL ファイルが含まれています:

- ターゲット システムには IIS 7 が搭載されていて、IIS 6 メタベースおよび IIS 6 構成の互換性機能がインストールされている。
- ターゲット システムに IIS 6 以前が搭載されている。

DLL ファイルのリリースおよびデバッグ バージョン、および ファイルの作成に使用された Visual Studio 2008 C++ プロジェクトは以下の場所にインストールされています:

InstallShield Program Files フォルダー¥Samples¥WindowsInstaller¥Detect IIS6 Compatibility

プロジェクトで Windows Installer DLL ファイルを使用するためには、まず DetectIIS6Compat.dll ファイルで **DetectIIS6Interfaces** 関数を呼び出すカスタム アクションを作成する必要があります。



タスク

IIS メタベースおよび IIS 6 構成との互換性機能と IIS 6 以前をチェックするカスタム アクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI の場合) または **[カスタム アクション]** (DIM、マージ モジュール プロジェクトの場合) をクリックします。
2. 中央のペインで、**[カスタム アクション]** エクスプローラーを右クリックして、**[新しい MSI DLL]** をポイントしてから、**[Binary テーブルに保存]** をクリックします。新しいカスタム アクションが追加されます。
3. アクションの名前を、たとえば **DetectIIS6Interfaces** に変更します。
4. 右側のペインでカスタム アクションの設定を構成します:
   - **"DLL ファイル名"** 設定で、DetectIIS6Compat.dll ファイルへのパスを指定します。通常のパスは以下のとおりです:

     <ISProductFolder>¥Samples¥WindowsInstaller¥Detect IIS6 Compatibility¥DetectIIS6Compat.dll
   - **"関数名"** 設定で、次の名前を入力します:

     DetectIIS6Interfaces
   - **"スクリプト内実行"** 設定で、**[即時実行]** を選択します。
   - 1 つまたは複数のシーケンス設定で適切なオプションを選択して、必要に応じて、カスタム アクションをスケジュールします。

IIS 6 メタベースおよび IIS 6 構成の互換性機能がインストールされているか、ターゲットシステムに IIS 6 以前が搭載されている場合、Windows Installer プロパティ ISIISMETABASECOMPATPRESENT の値は 1 に設定されます。ターゲット システムに IIS 7 以降が搭載されていて、互換性機能がインストールされていない場合、プロパティは設定されません。

条件ステートメントで ISIISMETABASECOMPATPRESENT を使用できます。たとえば、IIS 6 メタベースおよび IIS 6 構成の互換性機能または IIS 6 以前が搭載されていないシステム上で製品の使用を不可能にする場合、[一般情報] ビューの "インストール条件" 設定に次の条件ステートメントを入力します:

ISIISMETABASECOMPATPRESENT

別の方法として、エラー カスタム アクションを作成することもできます。実行時、そのカスタム アクションの条件が True の場合、エラーが表示されます。したがって、IIS 6 メタベースおよび IIS 6 構成の互換性機能または IIS 6 以前が搭載されていないシステム上で製品の使用を不可能にする場合、エラー カスタム アクションに以下の条件を使用します:

Not ISIISMETABASECOMPATPRESENT

IIS 6 メタベースおよび IIS 6 構成の互換性機能または IIS 以前が搭載されているシステム上にのみコンポーネントをインストールする場合、[コンポーネント] ビューの "条件" 設定に次の条件ステートメントを入力します:

ISIISMETABASECOMPATPRESENT=1

IIS の詳細設定を構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*



メモ: *詳細 IIS 設定は IIS 6 以前に適用します。IIS 7 は、これらの設定を無視します。*

[IIS 構成] ビュー内の [Web サイト] エクスプローラーで Web サイト、アプリケーション、または仮想ディレクトリを選択したときに表示される右側のペインには、IIS の最も一般的な設定が含まれています。右側ペインの下にある [詳細] 領域では、その他の領域では表示されない IIS 設定を構成できます。



タスク ***[IIS 構成] ビューに表示されていない設定を構成するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、詳細設定を構成する Web サイト、アプリケーション、または仮想ディレクトリを選択します。
3. **"その他の IIS プロパティ"** 設定で、省略記号ボタン (...) をクリックします。**[その他の IIS プロパティ]** ダイアログ ボックスが開きます。
4. プロパティー覧で、値を変更するプロパティを選択して、**[値の変更]** ボタンをクリックします。**[IIS メタデータ値の編集]** ダイアログ ボックスが開きます。
5. **[値]** ボックスで、新しい値を指定します。

詳細プロパティのデフォルト値を変更すると、そのプロパティと、指定した値、および追加の必須情報が ISIISProperty テーブルに追加されます。詳細については、「ISIISProperty テーブル」を参照してください。

Web サイト、アプリケーション、または仮想ディレクトリのカスタム エラー メッセージを定義する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

エンドユーザーが Web サイトに接続しようとして、HTTP (Hypertext Transfer Protocol) エラーが発生した場合、エンド ユーザーのブラウザーはエラーを説明するデフォルトのメッセージを表示します。HTTP エラー コードをファイルまたは URL (Uniform Resource Locator) へマッピングすることで、インストールが IIS を構成して、デフォルトのエラー メッセージの代わりにカスタム エラー メッセージを表示するよう設定することができます。



タスク

Web サイト、アプリケーション、または仮想ディレクトリのカスタムエラーメッセージを構成するには、以下の手順を実行します。

1. カスタムエラーメッセージを含むファイルを作成し、インストールに追加します。
2. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
3. HTTP エラーメッセージをカスタマイズする Web サイト、アプリケーション、または仮想ディレクトリを選択します。Web サイト、アプリケーション、または仮想ディレクトリの設定が、右側に表示されます。
4. **"カスタム エラー"** 設定で、省略記号ボタン (...) をクリックします。**[カスタム エラー]** ダイアログ ボックスが開きます。
5. 変更する HTTP エラーコードを選択し、**[プロパティの編集]** ボタンをクリックします。**[新しいプロパティの追加]** ダイアログ ボックスが開きます。
6. ファイルへエラーコードをマップするには、以下の手順を実行します。
   a. **[メッセージの種類]** リストから **[ファイル]** を選択します。
   b. [**ファイル**] ボックスで、インストール内のカスタム エラー メッセージをポイントするパスおよびファイル名を入力するか、[参照] ボタンを押してファイルを指定します。

   URL へエラー コードをマップするには、以下の手順を実行します。
   a. [**メッセージの種類**] 一覧から URL を選択します。
   b. **URL** ボックスで、カスタム エラー メッセージへの URL を入力します。

Windows Installer のプロパティを使用して、IIS 設定を動的に変更する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript MSI*

基本の MSI および InstallScript MSI プロジェクトでは、Windows Installer のプロパティを使用して、実行時に IIS 設定を動的に構成することができます。これにより、エンドユーザーがターゲット マシンにインストールしている仮想ディレクトリの名前、TCP ポート、サイト番号、または Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張の他の IIS 設定を指定できるようにすることができます。

Windows Installer は MsiFormatRecord を使って、実行時にこのプロパティを解決します。このプロパティとその値は、アンインストールと修復の時に使用できるように、インストールによって次のレジストリ キーに書き込まれます:

HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Uninstall¥InstallShield Uninstall Information¥*{ProductCode}*

したがって、Web サイトがエンド ユーザーによって指定されたサイト番号にインストールされた場合、Web サイトとそのアプリケーション、およびその仮想ディレクトリは、このサイト番号から正常にアンインストールされるようになります。



ヒント: *IIS データをレジストリに格納しないほうが良い場合、プロジェクトで* IS_IIS_DO_NOT_USE_REG *プロパティを設定します。*

[IIS 構成] ビューのパスワードにプロパティを使用すると、パスワードは暗号化されてレジストリに格納されます。

例

以下は、エンドユーザーがインストール中に、Web サイトに匿名アクセスするときに使用されるユーザー名を指定できるようにする手順です。文字列の入力が許可されている [IIS 構成] ビューあるすべての IIS 設定では、ハードコード化されている値にプロパティを代入することができます。このビューでプロパティを指定するとき、プロパティは角かっこで囲み、プロパティ名はすべて大文字にする必要があります (例、**[MYPROPERTY]**)。

手順のステップ 5 はプロジェクトの種類によって若干異なります。これは、基本の MSI インストールのユーザー インターフェイスは Windows Installer が制御し、InstallScript MSI インストールのユーザー インターフェイスは InstallScript エンジンが制御するためです。



タスク ***エンドユーザーがユーザー名を指定できるようにするには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、エンドユーザーが匿名アクセスにユーザー名を指定できる Web サイトをクリックします。
3. **"匿名アクセスを有効にする"** 設定で、**[はい]** を選択します。
4. **"匿名ユーザー名"** 設定で、次のように入力します:

   [MYPROPERTY]
5. ダイアログでプロパティを使用します。この部分の手続きは、使用しているプロジェクトの種類によって異なります。
   - 基本の MSI プロジェクトの場合:
     a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。

b. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、User Name コントロールを含めるダイアログの下にある言語をクリックします。代わりに、新しいダイアログを追加することもできます。

c. **Edit Field** コントロールをダイアログに追加し、**Property** プロパティを次にように設定します:

MYPROPERTY

- InstallScript MSI プロジェクトの場合:

a. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。

b. User Name コントロールを含めるダイアログの OnFirstUIBefore イベントでダイアログ コードを見つけ、Windows Installer API 関数 **MsiSetProperty** の呼び出しを追加します。たとえば、ユーザー名をエンド ユーザーが CustomerInformation ダイアログで指定する名前にする場合、次のコードに示されている **MsiSetProperty** の呼び出しを追加します:

```
Dlg_SdCustomerInformation:
    nResult = SdCustomerInformation(szTitle, svName, svCompany, nUser);
    MsiSetProperty(ISMSI_HANDLE, "MYPROPERTY", svName);
    if (nResult = BACK) goto Dlg_SdWelcome;
```

6. リリースをビルドします。

MsiFormatRecord (Windows Installer ヘルプ ライブラリ)

InstallScript テキスト置換を使用して、IIS 設定を動的に変更する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript プロジェクトでは、テキスト置換文字列変数を使用して、実行時に IIS 設定を動的に構成することができます。これにより、エンドユーザーがターゲット マシンにインストールしている仮想ディレクトリの名前、TCP ポート、サイト番号、または Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張の他の IIS 設定を指定できるようにすることができます。InstallScript ランタイム コードは **TextSubSubstitute** 関数を使用して、文字列変数を適切な値で置き換えます。

例

以下は、エンドユーザーがインストール中に、Web サイトに匿名アクセスするときに使用されるユーザー名を指定できるようにする手順です。文字列の入力が許可されている [IIS 構成] ビューあるすべての IIS 設定では、ハードコード化されている値にテキスト置換文字列変数を代入することができます。このビューで指定する文字列変数は山かっこで囲む必要があります; 例、**<MYPROPERTY>**。指定する文字列変数は大文字と小文字を区別します。

タスク ***エンドユーザーがユーザー名を指定できるようにするには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、エンドユーザーが匿名アクセスにユーザー名を指定できる Web サイトをクリックします。
3. **"匿名アクセスを有効にする"** 設定で、**[はい]** を選択します。
4. **"匿名ユーザー名"** 設定で、次のように入力します:

<MYPROPERTY>

5. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
6. User Name コントロールを含むダイアログの OnFirstUIBefore イベントでダイアログ コードを見つけ、InstallScript 関数 **TextSubSetValue** の呼び出しを追加します。たとえば、ユーザー名をエンド ユーザーが CustomerInformation ダイアログで指定する名前にする場合、次のコードに示されている **TextSubSetValue** の呼び出しを追加します:

```
Dlg_SdRegisterUser:
    szMsg = "";
    szTitle = "";
    //{{IS_SCRIPT_TAG(Dlg_SdRegisterUser)
    nResult = SdRegisterUser( szTitle, szMsg, szName, szCompany );
    TextSubSetValue("<MYPROPERTY>", szName, TRUE);
    //}}IS_SCRIPT_TAG(Dlg_SdRegisterUser)
    if (nResult = BACK) goto Dlg_SdLicense2;
```

7. リリースをビルドします。

ターゲット マシン上で Web サービスを配布する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

ターゲット システムへ Web サービスを配布するには、Web サービス専用のファイルを特定の場所にコピーし、そのフォルダーに仮想ディレクトリ名を割り当てる必要があります。これにより、HTTP を通して Web サービスにアクセスできるようになります。



ヒント: *仮想ディレクトリをプロジェクトに追加する方法については、「Web サイトの作成とアプリケーションまたは仮想ディレクトリの追加」をご覧ください。*



タスク ***ターゲット マシンへ Web サービスを配布するには、以下の手順に従います。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. ターゲット システムにファイルをインストールするためのフォルダー (ターゲット ディレクトリ) を選択します。そのフォルダーにファイルを追加します。
3. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
4. **[Web サイト]** エクスプローラーで、Web サイトに関連付けられている仮想ディレクトリを選択します。

5. **"コンテンツ ソース パス (ローカルまたは UNC)"** 設定で省略記号ボタン (...) をクリックします。**[ディレクトリの参照]** ダイアログ ボックスが開きます。**[ファイルとフォルダー]** ビューで追加した新しいファイルを含むディレクトリと同じターゲット ディレクトリを入力します。

インストール時、ファイルはターゲット ディレクトリ フォルダーへコピーされます。さらに IIS が存在する場合、ターゲット システム上のフォルダーに仮想ディレクトリが構成されます。

Web アプリケーションのフォーム認証を有効化する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Web サイトの **[IIS 構成]** ビューで **[セキュリティ > 認証済みアクセス]** セクションの下にある **"フォーム認証"** 設定を使って、Web アプリケーションのフォーム認証を設定します。**[フォーム認証]** オプションを **[はい]** に設定してフォーム認証を有効化します。

ASP.NET フォーム ベースの認証は、多くの要求を受け取るパブリック Web サーバー上にあるサイトまたはアプリケーションに適しています。この認証モードでは、オペレーティング システムが提供している認証方法に頼ることなく、アプリケーション レベルでのクライアント登録および認証を管理することができます。



重要: *フォーム認証は、Web サーバーにユーザー名とパスワードをテキスト形式で送信します。ホームページ以外のアプリケーション内のすべてのページ、およびログオン ページには、Secure Sockets Layer (SSL) 暗号化を使用してください。*



タスク ***Web サイトのフォーム認証を有効化するには、次の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、Web サイトを選択します。
3. **[セキュリティ > 認証アクセス]** の下にある **"フォーム認証"** 設定を **[はい]** に設定します。

IISROOTFOLDER サポートの追加

IISROOTFOLDER は、ターゲット システム上にある Web サーバーのルート ディレクトリを判断するのに利用される InstallShield ディレクトリ変数です。インストール プロジェクトで IIS 機能を使用していて、Web サイトが既に追加されている場合 IISROOTFOLDER は自動的に追加されます。



メモ: *[ファイルとフォルダー] ビューで IISROOTFOLDER ディレクトリに追加されたファイルはすべて、ターゲット マシン上の Web サーバーのルート ディレクトリにインストールされます。IIS がインストールされていない場合、ファイルはルート フォルダーにコピーされます。*

IIS_WEBSITE_NAME プロパティ

IIS_WEBSITE_NAME プロパティは現在使用されていません。これらのプロパティが以前のプロジェクト バージョンから存在している場合、アップグレーダはこれらを自動的に処理します。アップグレーダは Web サイトを作成して、サイト番号フィールドを [IIS_WEBSITE_NAME] に設定します。新しい Web サイトには、任意のプロパティまたはハードコード化された番号を使用できます。

IIS_PORT_NUMBER プロパティ

IIS_PORT_NUMBER プロパティは現在使用されていません。これらのプロパティが以前のプロジェクト バージョンから存在している場合、アップグレーダはこれらを自動的に処理します。アップグレーダは Web サイトを作成して、ポート番号フィールドを [IIS_PORT_NUMBER] に設定します。新しい Web サイトを作成すると、任意のプロパティまたはハードコード化された番号を使用できます。

メンテナンスおよびアンインストールのためのインストールを作成する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield では、エンドユーザーがインストールを再実行して、プログラム機能の変更したり、アプリケーションの再インストールまたは削除をすることができます。エンドユーザーがコントロール パネルの [プログラムの追加と削除] でアプリケーションを選択すると、インストールによって以下を実行できるダイアログ ボックスが表示されます。

- 以前にインストールされなかった機能の個別インストールおよびアンインストール。
- 最初のインストールで選択された設定でアプリケーションを再インストール
- アプリケーションのアンインストール

アプリケーションの変更、修正、アンインストールを行う場合、オペレーティング システムにがアプリケーションの存在を認知している必要があります。このため、インストールはオペレーティング システムにアプリケーションを登録し、従って簡単にメンテナンスまたはアンインストールができます。[一般情報] ビューで必要な情報を入力します。

すべての InstallShield インストールでは、メンテナンス (変更および修正) はデフォルトで自動的に処理されます。

Windows Installer ベースのインストールの場合、アンインストールは自動的に処理されますが、唯一の例外は、アンインストール中それ自身で効果を取り消すか、または、アンインストール中のみ実行される別のカスタム アクションによってその効果を取り消させる必要があるカスタム アクションがあるということです。

InstallScript または InstallScript MSI インストールでは、数多くのスクリプト関数のアクションは自動アンインストールによってログされますが、ログされないスクリプト関数アクションは、アンインストールのスクリプトで対応する必要があります。

このセクションには、次のトピックが含まれています:

- アンインストールのための InstallScript インストールを作成
- アンインストール時またはインストール時のみにスクリプト コードを実行する
- アンインストール用にログされた InstallScript 関数
- メンテナンス/アンインストール機能
- 製品が作成したファイルの削除
- 製品が作成したレジストリデータの削除

アンインストールのための InstallScript インストールを作成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

アンインストール機能の構成は、その結果生じる操作の複雑性を考えると、比較的単純です。**CreateInstallationInfo** (または **InstallationInfo**) および **MaintenanceStart** (または **DeinstallStart**) の関数を呼び出します。イベント指向のスクリプトでは、**CreateInstallationInfo** および **MaintenanceStart** がデフォルト **OnMoveData** イベント ハンドラーコードで呼び出されます。

他にも、アンインストール機能の構成に重要な役割を果たす関数があります。たとえば、**RegDBCreateKeyEx** 関数を使って作成されたすべてのキーは、ロギングが無効ではない限り、アンインストール用に記録されます。また、**Disable** 関数を呼んでロギングを無効にする場合、必ず **Enable** を呼んで、アンインストール用に記録が必要なその後に続くアクションのために再度ロギングを有効化するようにしてください。

MaintenanceStart (または **DeinstallStart**) 必要なレジストリキーおよびそのための必要な値を作成します。このキーの下に追加のレジストリキーを作成する場合、 RegDBSetItem を呼ぶか、または、カスタム値の場合、次のようなコードを使用してください。

```
RegDBSetDefaultRoot( HKEY_USER_SELECTABLE );
RegDBSetKeyValueEx( "Software¥¥Microsoft¥¥Windows¥¥Current Version¥¥Uninstall¥¥" + INSTANCE_GUID, "カスタム値", nType, szValue, nSize);
```



メモ: *アンインストールは、初回インストールが* **FeatureTransferData** *または* **FeatureMoveData** *を呼び時のみ実行可能です。(イベント指向のスクリプトでは、***FeatureTransferData** *関数が自動的に呼び出されます)。インストールが他の手法 (例、***XCopyFile** *等) によってファイルをインストールするときに、エンドユーザーにアンインストールを実行できるようにする場合、すべてのコンポーネントを選択解除し、***FeatureTransferData** *または* **FeatureMoveData** *を呼び出します。この呼び出しが実行されている間、***SdShowMsg** *を呼んでメッセージを表示することもできます。*

アンインストール時に実行するスクリプト コードを含める



メモ: **DeinstallStart** *を呼んでアンインストールを有効にすると、インストール スクリプトはアンインストール中に実行されません。アンインストール時にスクリプト コードを実行するには、代わりに* **MaintenanceStart** *を呼びます。(イベントベースのインストールでは、***MaintenanceStart** *がデフォルトの* **OnMoveData** *イベント ハンドラーコードで呼び出されます。)*

イベントベースのスクリプト

イベント指向のスクリプトでは、次のイベント ハンドラー関数がアンインストール中に自動的に呼び出されます。

- **OnBegin**
- **OnMaintUIBefore**

- **OnMoving**
- <機能名>**Uninstalling** (各インストール済みの機能に対して)
- <機能名>**Uninstalled** (各インストール済みの機能に対して)
- **OnMoved**
- **OnMaintUIAfter**
- **OnEnd**

オーバーライド可能なデフォルトコードを持つこれらのイベント ハンドラー関数のコードはアンインストール中に実行されます。アンインストール、または、インストール中のみ特定のイベント ハンドラーを実行する手順については、アンインストール時またはインストール時のみにスクリプト コードを実行するを参照してください。

手続き型スクリプト

手続き型のスクリプト(プログラム ブロックを使用したスクリプト)で、**DeinstallStart** を使わずに **MaintenanceStart** を使ってアンインストールを有効にすると、スクリプトはアンインストール中に実行されます。アンインストール、または、インストール中のみ特定のスクリプトを実行する手順については、アンインストール時またはインストール時のみにスクリプト コードを実行するを参照してください。

アンインストール時またはインストール時のみにスクリプト コードを実行する



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

DeinstallStart ではなく **MaintenanceStart** を呼び出してアンインストールを可能にすると、アンインストールおよびメンテナンス インストール中にインストールスクリプトが実行されます。(イベントベースのスクリプトでは、**MaintenanceStart** がデフォルトの **OnMoveData** イベント ハンドラーコードで呼び出されます。)コードをアンインストール中、またはメンテナンスインストール中のみ実行するには、コードを次のような if-then ステートメントで囲みます。

```
if MAINTENANCE then
    /* このコードは、
    アンインストール または メンテナンス インストール */
endif;
```

コードを最初のインストール中のみ実行するには、コードを次のような if-then ステートメントで囲みます。

```
if !MAINTENANCE then
    /* このコードは、
    初回のインストール */
endif;
```

コードを上のどちらの if-then ステートメントで囲まない場合、アンインストール、メンテナンスインストール、および初回のインストール中に実行されます。ただし、イベント指向のスクリプトでは、以下のような例外があります。

- 次のようなイベント ハンドラーのコードは、アンインストール中およびメンテナンスインストール中のみ実行されます。

  OnMaintUIBefore

  OnMaintUIAfter

- 次のようなイベント ハンドラーのコードは、最初のインストール中のみ実行されます。

  OnCCPSearch

  OnAppSearch

  OnFirstUIBefore

  OnFirstUIAfter

アンインストール用にログされた InstallScript 関数



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI (InstallScript ユーザー インターフェイス (UI) のスタイルが、外部 UI ハンドラーとして InstallScript エンジンを使用する従来型のスタイルの場合)*

この情報は、InstallScript UI に新しいスタイル (InstallScript エンジンを埋め込み UI ハンドラーとして使用するスタイル) が使用されている InstallScript MSI プロジェクトには適用しません。詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

一般的に、ログ記録が有効な状態で作成または置換されたすべてのファイル、フォルダー、レジストリ エントリ、.ini ファイル エントリ、ショートカット フォルダー、およびサービスは、アンインストール用にログ記録されます。また、InstallScript インストールおよび InstallScript MSI インストール中にターゲット システムで変更を行う InstallScript 関数もアンインストール用にログ記録されます。デフォルトでインストールがアンインストール用にログ記録を行う一般的な関数の例は以下の通りです:

- AddFolderIcon
- AddProfString
- CopyFile
- CreateInstallationInfo
- CreateProgramFolder
- FeatureMoveData
- FeatureTransferData
- RegDBSetAppInfo
- RegDBSetItem
- ReplaceProfString
- ServiceAddService

- ServiceStartService
- VerSearchAndUpdateFile
- VerUpdateFile
- WriteProfString
- XCopyFile

ログ記録は、InstallScript コード内でDisable (LOGGING); および Enable (LOGGING); 関数を呼び出して有英語効または無効にできます。詳細については、「Disable」および「 Enable」を参照してください。

メンテナンス/アンインストール機能



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

メンテナンス/アンインストール機能はメンテナンス設定とアンインストールに必要なファイルをインストールします。ファイルのターゲット先はシステム変数 *DISK1TARGET* の値によって指定されます。この機能は、 .cab に自動的に配置され、InstallShield インターフェイスでは表示されません。

実行時にメンテナンス/インストール機能は、全セットアップの種類用に自動選択されます。**FeatureUpdate** を呼び出すと、この機能は選択解除されます。メンテナンス/インストール機能は、**FeatureSelectItem** を呼び出すことで選択/選択解除され、システム変数 *DISK1COMPONENT* を関数の第 2 引数として渡します。たとえば、次のコードは選択解除する必要があります。

```
FeatureSelectItem( MEDIA, DISK1COMPONENT, FALSE );
```

製品が作成したファイルの削除



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

デフォルトでは、インストール後の製品によって作成されたファイルは、ユーザーのデータであると見なされるので、アプリケーションをアンインストールする際にも削除されません。

基本の MSI プロジェクト

RemoveFile テーブルにレコードを作成して (ダイレクト エディターを使用)、削除するファイルを指定することができます。たとえば、アプリケーションが *INSTALLDIR* の指定先に Product.ini というファイルを作り、メインの実行可能プログラムを含むコンポーネントの削除時に、このファイルも同時に削除させる場合は、RemoveFile テーブルに次のレコードを追加します。

テーブル 1 · RemoveFile テーブルに追加する追加

FileKey	remove_file
Component_	ProgramFiles
FileName	Product.ini
DirProperty	INSTALLDIR
InstallMode	2

InstallScript および InstallScript MSI プロジェクト

スクリプトで、アンインストールのイベント ハンドラー関数の中から DeleteFile および DeleteDir 関数を呼び出すことで、指定したファイルやディレクトリを削除できます。

製品が作成したレジストリデータの削除



***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

デフォルトでは、製品のアンインストーラーは、インストール プログラムが作成したデータのみを削除します。

基本の MSI プロジェクト

レジストリエクスプローラーは特別なアンインストールフラグを使って、アンインストール時に削除するレジストリデータを管理します。特に、レジストリキーで “-” フラグ (キー全体をアンインストール) を使用すると、キーおよびそのすべての値とサブキーがアンインストール時に削除されます。



***メモ·** RemoveRegistry テーブル (ダイレクト エディターで表示) は、製品のインストール時にのみ削除するデータを指定します。詳細については、Windows Installer ヘルプライブラリを参照してください。*

InstallScript MSI プロジェクト

スクリプトで、アンインストールのイベント ハンドラー関数の中から RegDBDeleteValue および RegDBDeleteKey 関数を呼び出すことで、レジストリ値やキーを削除できます。

トランザクション処理を使って複数パッケージ インストールを構成する



プロジェクト- *次のプロジェクト タイプで、トランザクション処理を使った複数パッケージのインストールがサポートされています:*

- *基本の MSI*
- *InstallScript MSI*

Windows Installer 4.5 は、トランザクション処理を使った複数パッケージのインストールをサポートします。パッケージを連鎖させることによって、単一のトランザクションとして処理します。トランザクションに含まれる 1 つまたは複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は全てのパッケージについてロールバックを開始して、システムを以前の状態に復元します。

ドキュメントのこのセクションでは、InstallShield で連鎖されたパッケージとしてインストールに含める .msi パッケージを指定する方法について説明します。また、連鎖するパッケージに渡すプロパティなどの設定の構成方法についても説明します。このセクションには、次のトピックが含まれています:

- トランザクション処理を使った複数パッケージ インストールについての概要
- 新しい連鎖 .msi パッケージをプロジェクトに追加する
- リリース フラグを連鎖 .msi パッケージに割り当てる
- プロジェクトから連鎖 .msi パッケージを削除する



メモ- *Windows Installer 4.5 は、トランザクション処理を使った複数パッケージのインストールをサポートします。以前のバージョンは、連鎖 .msi パッケージのいずれも起動しません。*

インストールで Windows Installer 4.5 をインストールする方法は、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

トランザクション処理を使った複数パッケージ インストールについての概要



プロジェクト- *次のプロジェクト タイプで、トランザクション処理を使った複数パッケージのインストールがサポートされています:*

- *基本の MSI*
- *InstallScript MSI*

インストールに 1 つまたは複数の .msi パッケージが連鎖するパッケージとして追加されていて、エンド ユーザーが Windows Installer 4.5 を搭載したシステム上でこのインストールを実行すると、Windows Installer はすべてのパッケージを単一のトランザクションでインストールすることができます。トランザクションに含まれる 1 つまたは

複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は全てのパッケージについてロールバックを開始して、システムを以前の状態に復元します。

トランザクション処理を行うとき、Windows Installer は .msi パッケージから様々なスクリプトの状態をバッチ処理します。たとえば、Windows Installer が [実行] シーケンス中に InstallInitialize アクションを開始するとき、その時点で Windows Installer は全てのパッケージの即時アクションを実行し、また全ての遅延アクションをインストール スクリプトに配置します。次に、Windows Installer は全てのパッケージのインストール スクリプトに含まれる遅延アクション (コミットまたはロールバック アクションを除く) を実行します。さらに、Windows Installer はロールバック アクションをロールバック スクリプトにコピーします。この時点まで各パッケージの製品が正しくインストールされると、Windows Installer は各パッケージのすべてのコミットアクションを実行します。そうでない場合、Windows Installer はロールバック スクリプトを実行して、ターゲット システムを元の状態に復元します。

この機能についての詳細は、Windows Installer ヘルプ ライブラリの「Multiple-Package Installations」を参照してください。

メモ: *Windows Installer 4.5 は、トランザクション処理を使った複数パッケージのインストールをサポートします。以前のバージョンは、連鎖 .msi パッケージのいずれも起動しません。*

インストールで Windows Installer 4.5 をインストールする方法は、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

新しい連鎖 .msi パッケージをプロジェクトに追加する

プロジェクト: *次のプロジェクト タイプで、トランザクション処理を使った複数パッケージのインストールがサポートされています:*

- *基本の MSI*
- *InstallScript MSI*

InstallShield では、ビルド済みの Windows Installer パッケージ (.msi ファイル) を連鎖 .msi パッケージとしてプロジェクトに追加することができます。Windows Installer 4.5 は、トランザクション処理を使った複数パッケージのインストールをサポートします。パッケージを連鎖させることによって、単一のトランザクションとして処理します。トランザクションに含まれる 1 つまたは複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は全てのパッケージについてロールバックを開始して、システムを以前の状態に復元します。

[リリース] ビューの [連鎖 .msi パッケージ] 領域で、メイン インストールに連鎖させる 1 つまたは複数の .msi パッケージをプロジェクトに追加できます。

連鎖 .msi パッケージは、インストール中にマナー アップグレードをサポートします。

メモ: *Windows Installer 4.5 は、トランザクション処理を使った複数パッケージのインストールをサポートします。以前のバージョンは、連鎖 .msi パッケージのいずれも起動しません。*

インストールで Windows Installer 4.5 をインストールする方法は、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。



タスク ***.msi パッケージを連鎖するパッケージとして追加するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[連鎖 .msi パッケージ]** エクスプローラーを右クリックしてから、**[新しい連鎖パッケージ]** をクリックします。InstallShield は、デフォルト名を持つ新しい置換アイテムを追加します。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。

   名前は実行時には表示されません。これはメイン インストールに含まれる異なる連鎖 .msi パッケージ間でパッケージを区別するのに内部で使用される名前です。
4. 新しい連鎖されたパッケージ アイテムを選択します。InstallShield の右側に設定が表示されます。
5. **"インストール (実行時のパス)"** 設定で、**[参照]** ボタンをクリックします。**[ファイルの参照]** ダイアログ ボックスが開きます。
6. 追加する Windows Installer パッケージ (.msi ファイル) を選択してから、**[開く]** をクリックします。選択するパッケージは、**MsiInstallProduct** を使ったインストールが可能な .msi パッケージでなくてはなりません。InstallScript MSI パッケージは選択できませんので、ご注意ください。

InstallShield は、.msi パッケージ (および、.msi パッケージが圧縮されていない場合は非圧縮ファイル) を製品のメイン .msi パッケージにストリームするかどうかを問い合わせるプロンプトを表示します。

- ファイルのストリームを指定した場合、InstallShield は .msi パッケージの名前を **"インストール (実行時のパス)"** 設定に追加します。また、自動的にファイルの名前 (圧縮された .msi パッケージの場合) またはエントリ *.*(非圧縮の .msi パッケージの場合) が **[ストリームされたファイル]** ボックスに追加されます。ワイルドカード エントリ *.* は、.msi パッケージと同じフォルダーにある全てのファイルとともに .msi パッケージにストリームされます。
- ファイルのストリームを行わないことを指定した場合、InstallShield は次のパスを **"インストール (実行時のパス)"** 設定に追加します:

  [SourceDir]FileName.msi

  このパスは、必要に応じて変更が可能です。たとえば次のようなパスを使用し、また **[インストール後にストリームされたファイルを削除する]** チェック ボックスをクリアしたい場合があります:

  [LocalAppDataFolder]{ProductCodeGUID}¥FileName.msi

  この例では、.msi パッケージはローカル システムにキャッシュされ、それはメンテナンスで使用することができます。

  ヒント・*Windows Installer では、.msi パッケージのファイル サイズに制限があるため、ファイルを .msi パッケージにストリームしなくない場合があります。*

7. 必要に応じて、連鎖 .msi パッケージの設定を構成します。詳細については、「連鎖 .msi パッケージの設定」を参照してください。

メイン インストールにいくつかの .msi パッケージが含まれている場合、InstallShield はメイン インストールの ISChainPackage テーブルの Order 列を使って、実行時に連鎖するパッケージを起動する順番を決定します。アンインストールは Order 番号が高い方から低い方へ逆の順番で、また全てのインストールは Order 番号が低い方から高い方へ順番に処理されます。

メモ- [リリース] ビューの [連鎖 .msi パッケージ] 領域で .msi パッケージを指定した場合、メイン インストールのリリースをビルドするときに InstallShield は連鎖パッケージをビルドしません。

リリース フラグを連鎖 .msi パッケージに割り当てる

***プロジェクト-** 次のプロジェクト タイプで、トランザクション処理を使った複数パッケージのインストールがサポートされています:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

特定のビルドから除外する連鎖 .msi パッケージに、1 つまたは複数のリリース フラグを割り当てることができます。たとえば、連鎖 .msi パッケージを必要とする特別なアドオンを含んだ製品の特別なエディションにのみ含める連鎖 .msi パッケージがある場合、その連鎖パッケージにフラグを付けることで、必要な場合にのみそれがビルドに含まれるようにします。

タスク

インストール プロジェクトに追加した連鎖 .msi パッケージにリリース フラグを追加するには、次の手順に従います。

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[連鎖 .msi パッケージ]** エクスプローラーで、リリース フラグを含めるパッケージを選択します。
3. **"リリース フラグ"** 設定で、文字列を入力します。文字列には、文字または数字を使用できます。1 つの連鎖パッケージに複数のフラグを付けるには、コンマを使ってフラグを区切ります。

リリース フラグに基づいて連鎖 .msi パッケージをフィルターする方法についての詳細は「リリース フラグ」を参照してください。

プロジェクトから連鎖 .msi パッケージを削除する

***プロジェクト-** 次のプロジェクト タイプで、トランザクション処理を使った複数パッケージのインストールがサポートされています:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*



タスク ***プロジェクトから連鎖 .msi パッケージを削除するには、以下の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[連鎖 .msi パッケージ]** エクスプローラーで、プロジェクトから削除する連鎖 .msi パッケージを右クリックしてから、**[削除]** を選択します。

InstallShield がプロジェクトから連鎖 .msi パッケージを削除します。InstallShield は、.msi パッケージ、またはそれに属するファイルをファイル システムからは削除しませんので、ご注意ください。

インストールのビルド、テスト、および配布

インストール プロジェクトの機能、コンポーネント、ファイル、ショートカット、レジストリ エントリ、エンド ユーザー ダイアログ、およびその他の要素を構成が完了すると、インストールのリリースを作成しビルドできます。InstallShield では、1 つのプロジェクトに対して複数リリースの作成が可能で、異なるメディア タイプおよび異なる要件の組み合わせにしたがって各リリースを構成することができます。リリースをビルドすると、インストールのコンテンツがパッケージされ、配布メディアにコピーして、必要に応じて配布または配置することができるディスク イメージが作成されます。

テストは安定したインストールを作成するために最も重要です。InstallShield では、リリースのエンドユーザー インターフェイスのみを部分的にテスト実行することが可能ですInstallShield のボタンをクリックするだけで、インストールを実行することもできます。この方法でインストールを実行した場合、エンド ユーザーのマシン上とまったく同様にインストールが実行します。ファイルはすべて転送され、ショートカットおよびレジストリ エントリが作成され、ユーザー インターフェイスが表示されます。

製品をリリースする前に、インストール プロジェクトを検証した方がよい場合があります。Windows Installer プロジェクトを検証するには、一連の内部整合性評価プログラム (ICE) のルールをインストール プロジェクトに適用します。これらの ICE はマイクロソフトが作成したもので、アクションが正常に実行するために必要な有効データベースがインストール パッケージに含まれていることを確認するのに利用します。

InstallShield のデバッグ ツールを利用して、InstallScript スクリプトまたは Windows Installer リリースをデバッグして問題の原因を追究することができます。スクリプトをデバッグする際、スクリプトをステートメント毎に実行し、実行ポイントがスクリプトの中を移動するのを確認しながらコントロールの流れを追います。また、スクリプト実行中どの時点でも、スクリプト中の変数の値を監視することもできます。Windows Installer リリースをデバッグするとき、InstallShield は実行を一時停止するブレーク ポイントに到達するまで各アクションおよびダイアログ ボックスを実行します。この時点で、プロパティを表示して設定することができます。

このセクションには、次のトピックが含まれています:

- リリースの作成とビルド
- インストールのテストと実行
- プロジェクトの検証
- インストールまたはアンインストールがターゲット システムを再開するタイミングについて
- Windows Installer ベースのインストールのデバッグ
- インストールを複数のディスクまたは CD に分割する
- セットアップランチャーの作成
- インストールを配布する
- 64 ビット セットアップ ランチャーのビルド
- Windows アプリ パッケージの配布

リリースの作成とビルド

InstallShield でプロジェクトのデザインが完了すると、構成およびビルドしてエンドユーザーへ配布できるリリースを作成できます。リリースは、リリース ウィザード、または [リリース] ビューに表示される設定で指定されたオプションに基づいてビルドされます。

製品構成の使用

***プロジェクト・** 以下は、製品構成をサポートするプロジェクトの種類です:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

ビルドした各リリースは、製品構成のメンバーです。製品構成により、製品名、製品コード、パッケージコードなど、類似したプロパティを共有するリリースを一緒にグループ化することができます。

製品の構成により、InstallShield の外部でフォルダー構造を設定しなくてもビルドの履歴を保持することができます。たとえば開発中の製品が、同じバージョンでも複数のリリースになるような場合がありえます。製品を再ビルドするごとに各リリースを上書きすることを、避けたい場合もあります。製品構成により、過去のビルド履歴の保持と新しいリリースの組み込みを同時に実現できます。

以前は、このような履歴を InstallShield の外部に保管しなければなりませんでした。その場合、ドライブ上にディレクトリ構造を作成して各フォルダーに手作業でラベルを付け、そのディレクトリにリリースをコピーしていました。製品構成により、これらの作業は InstallShield で自動的に行われます。

タスク

製品構成を作成するには、以下の手順に従います。

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーを右クリックし、**[新しい製品構成]** をクリックします。

InstallShield が [リリース] エクスプローラーに新しい製品構成を追加します。[全般] タブで製品構成の設定が一覧表示されます。

各製品構成の設定については、製品構成の [全般] タブ を参照してください。

***ヒント・** リリース ウィザード を使って製品構成を作成することもできます。(-a パラメーターを使って) コマンドラインから既存の製品構成を指定することができますが、コマンドラインを通してリリースの設定を定義することはできません。*

ビルドに適切なアーキテクチャ検証の種類を選択する

***プロジェクト・** 以下は、製品構成をサポートするプロジェクトの種類です:*

- *基本の MSI*
- *マージ モジュール*

WOW64 (32-bit Windows-on-Windows) サポートを含まない x64 Windows ベースのシステムがより一般的なので、InstallShield でリリースをビルドするときにアーキテクチャの検証を実行することが推奨されます。アーキテクチャの検証を使って、インストールがターゲット システムのアーキテクチャと一致しない製品ファイルをインストールしようとしたり、ランタイム バイナリを使用したりする問題を引き起こす可能性のある状況を検出することができます。

たとえば、エンド ユーザーが WOW64 サポートが搭載されていない 64 ビット Windows Server Core システムでインストールを実行することが可能な場合、アーキテクチャの検証を使って、インストールに含まれるすべての x86 ビット の製品ファイル、または x86 カスタム アクション ファイルを識別できます。x86 バイナリを WOW64 サポートを搭載しない x64 ターゲット システムにロードすることはできません。

また、単一のプロジェクトに x64 と x86 製品ファイル (または x64 と x86 カスタム アクション) が混在し、個別の x86 .msi パッケージと x64 .msi パッケージを生成する場合、アーキテクチャの検証を使って、x86 リリースに含まれる任意の x64 製品またはカスタム アクション ファイルを識別することができます。これらのファイルを x86 ターゲット システムでロードすることはできません。

アーキテクチャの検証を使って、ビルトイン InstallShield カスタム アクション DLL の x86 バージョンのみを含む x86 専用 .msi パッケージ、またはビルトイン InstallShield カスタム アクション DLL の x64 バージョンのみを含む x64 専用 .msi パッケージを生成することができます。この機能を有効化するため、InstallShield では 2 つの定義済みパス変数 (ISRedistPlatformDependentFolder および ISRedistPlatformDependentExpressFolder) が使用されます。デフォルトで、これらのパス変数の値は、InstallShield カスタム アクション DLL の x86 バージョンを含むサブフォルダーです。サブフォルダーは、*InstallShield Program Files Folder*¥Redist にあります。InstallShield は、必要な場合、ビルド時にこれらのパス変数の値を変更して、InstallShield カスタム アクション DLLの (x86 バージョンの代わりに) x64 バージョンをリリースに含みます。これらのファイルは *InstallShield Program Files フォルダー*¥Redist の異なるサブフォルダーにあります。

アーキテクチャの検証の概要

InstallShield は、次の異なるレベルでビルド時のアーキテクチャ検証をサポートします:

- **なし** – ビルド時のアーキテクチャ検証を行いません。
- **厳密な検証をしない**—この種類の検証を使って、x86 および x64 .msi パッケージ (Template Summary プロパティで指定) をビルドでき、両方の種類のパッケージで x86 と x64 製品ファイルとカスタム アクション ファイル を混合することができます。

  ビルド時に、ISRedistPlatformDependentFolder と ISRedistPlatformDependentExpressFolder パス変数は、x86 InstallShield カスタム アクション DLL を含む x86 の場所をポイントします。InstallShield は、プロジェクトにこれらのカスタム アクションを必要とするサポートが含まれている場合、これらの x86 カスタム アクション DLL をビルドに含みます。

  [厳密な検証をしない] オプションは、Template Summary プロパティで指定されているアーキテクチャが、リリースに含まれている 1 つ以上のカスタム アクション ファイルのアーキテクチャと一致しない場合に、ビルド エラーをトリガしません。

  [厳密な検証をしない] オプションは、Template Summary プロパティで指定されているアーキテクチャが、リリースに含まれている 1 つ以上の製品ファイルのアーキテクチャと一致しない場合に、ビルド警告をトリガしません。

  [厳密な検証をしない] オプションがデフォルトの選択です。
- **厳密な検証をする**—この種類の検証を行うと、Template Summary プロパティで Intel (x86 用) または x64 のどちらが指定されているかによって、InstallShieldは x86 専用または x64 専用 .msi パッケージをビルドしようとします。

  Template Summary プロパティで Intel が指定されている場合、ISRedistPlatformDependentFolder および ISRedistPlatformDependentExpressFolder パス変数は x86 InstallShield カスタム アクション DLL を含む x86 の場所をポイントします。ただし、Template Summary プロパティで x64 が指定されていて、厳密な検証を行う場合、これらのパス変数は x64 InstallShield カスタム アクション DLL を含む x64 の場所をポイントします。プロジェクトにこれらのカスタム アクションのいずれかを必要とするサポートが含まれている場合、InstallShield は適切な x86 または x64 バージョンの DLL をビルドに追加します。

厳密な検証を行うオプションは、"テンプレートの概要" プロパティで指定されているアーキテクチャが、リリースに含まれている 1 つ以上のカスタム アクション ファイルのアーキテクチャと一致しない場合に、ビルド エラーをトリガする場合があります。したがって、x86 パッケージをビルド時に検証中、InstallShield はパッケージに含まれる各 x64 カスタム アクションに対してビルド エラーを生成します。x64 パッケージをビルド時に検証中、InstallShield はパッケージに含まれる各 x86 カスタム アクションに対してビルド エラーを生成します。

[厳密な検証を行う] オプションは、Template Summary プロパティで指定されているアーキテクチャが、リリースに含まれている 1 つ以上の製品ファイルのアーキテクチャと一致しない場合に、ビルド警告をトリガする場合があります。したがって、x86 パッケージをビルド時に検証中、InstallShield はパッケージに含まれる各 x64 製品ファイルに対してビルド警告を生成します。x64 パッケージをビルド時に検証中、InstallShield はパッケージに含まれる各 x86 製品ファイルに対してビルド警告を生成します。

***ヒント・** Template Summary プロパティに関する情報は、「"テンプレート概要" プロパティを使用する」を参照してください。*

64 ビット オペレーティング システムをターゲットとする処理についてのヒントは、「64 ビットのオペレーティング システムを基本の MSI および InstallScript MSI インストールを使ってターゲットする」を参照してください。

アーキテクチャ検証の種類を選択する

InstallShield では、製品の [リリース] ビューで定義されている各製品構成に使用するアーキテクチャ検証の種類を指定できます。

タスク ***ビルド時に実行するアーキテクチャ検証の種類を選択するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、構成する製品構成を選択します。
3. 次いで、**[全般]** タブの **"アーキテクチャの検証"** 設定で、適切なオプションを選択します。選択可能なオプションは以下のとおりです:
   - なし
   - 厳密な検証をしない
   - 厳密な検証をする

厳密なアーキテクチャの検証におけるビルド エラーと警告のトラブルシューティング

厳密なアーキテクチャの検証を使用して、Template Summary プロパティに x64 を指定する .msi パッケージをビルドする場合、ビルド時に InstallShield が次のようなエラーおよび警告を生成する可能性があります:

エラー -7319: 32 ビットの標準 DLL カスタム アクション MyCustomAction を 64 ビット限定のパッケージに含めることはできません。

警告 -7326: 32 ビット PE ファイル C:¥SourceFiles¥x86¥Myx86File.exe を64 ビット限定のパッケージに含んでいます。

多くの場合、x64 ターゲット システム上で x86 ファイルを実行することができます。ただし、x64 システムが WOW64 をサポートしない場合、x86 ファイルを実行することができないこともあります。

要件によって、ビルド エラーまたは警告で参照されている x86 ファイルまたはカスタム アクションを x64 ファイルまたはカスタム アクションで置換することが推奨されます。その他の状況下では、ビルド エラーまたは警告を無視することができます。たとえば、コンポーネントが x64 システムにインストールされないように防ぐ条件がファイルのコンポーネントに含まれている場合、対応するビルド警告を無視することができます。一部の状況において、要件を再検討し、WOW64 をサポートしない x64 ターゲット システムをサポートしない結論もあります。

同様に、x86 .msi パッケージをビルドするときに厳密なアーキテクチャの検証を使用する場合、次のようなビルド エラーと警告が発生する可能性があります:

エラー –7320: 64 ビットの標準 DLL カスタム アクション MyCustomAction を 32 ビット限定のパッケージに含めることはできません。

警告 –7327: 64 ビット PE ファイル C:¥SourceFiles¥x64¥Myx64File.exe を32 ビット限定のパッケージに含んでいます。

x86 システムで x64 ファイルまたはカスタム アクションを実行することはできません。要件によっては、プロジェクトを変更するか、ビルド エラーと警告を無視することができます。一部の場合、要件を再検討するかもしれません。

リリースに関する作業



プロジェクト: *インストール プロジェクト: インストール プロジェクトをビルドする前に、機能、コンポーネント、ファイル、ショートカット、レジストリ エントリ、エンドユーザー インターフェイスなどの、プロジェクトの各要素に対して設定のデザインと構成が完了していることを確認します。*

マージ モジュール プロジェクト: パッケージをビルドする前に、コンポーネント、ファイル、ショートカット、レジストリ エントリなど、マージ モジュールのすべての要素について設定のデザインと構成が完了していることを確認します。

InstallShield でプロジェクトのデザインが完了すると、構成およびビルドしてエンドユーザーへ配布できるリリースを作成できます。1 つのプロジェクトに対して複数リリースの作成したり、異なるメディア タイプおよび異なる要件の組み合わせにしたがって各リリースを構成したりすることができます。InstallShield では、次のような方法を利用して、リリースの作成またはビルドを行うことができます。

- リリース ウィザードを使用する。(このメソッドは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトでは使用できません。)
- [リリース] ビューを使用する。
- ISCmdBld.exe を使用してコマンドライン ビルドからビルドする。
- オートメーション インターフェイスでリリースをビルドする。(このメソッドは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトでは使用できません。)

Visual Studio 内部から InstallShield リリースをビルドする方法については、「Microsoft Visual Studio でリリースをビルドする」を参照してください。



ヒント: *リリースをビルドする際、Windows エクスプローラーが* Disk1 *フォルダーまたはサブフォルダーをポイントしていないことを確認してください。Windows エクスプローラーが* Disk1 *フォルダーをポイントしていると、ビルド プロセスが失敗してエラーが生成されます。*

リリース ウィザードを使用して、リリースの作成およびビルドを行う



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

リリース ウィザードでは、製品のリリースのビルド、および、リリースの設定の指定を簡単に行うことができます。



タスク ***リリース ウィザードを起動して、インストール パッケージをビルドするには、以下のいずれかを実行します。***

- ツールバーの **[リリース ウィザード]** ボタンをクリックします。
- **[ビルド]** メニューで **[リリース ウィザード]** をクリックします。
- **[リリース]** ビューで、リリース名を右クリックしてから **[リリース ウィザード]** をクリックします。

リリース ウィザードのパネルの指示に従います。ウィザードから返されるエラーの一覧については、「ビルド エラーと警告」を参照してください。

[リリース] ビューを使用して、リリースの作成およびビルドを行う



プロジェクト: *この情報は、次のプロジェクトに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[リリース] ビューでは、リリースの設定の指定やリリースのビルドを行うことができます。リリースの作成とビルドの手順は、使用しているプロジェクトの種類によって異なります。

基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトにおけるリリースの作成とビルド



タスク ***リリースの作成とビルドを行うには、以下の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで新しいリリースを含める製品構成を右クリックして、**[新しいリリース]** をクリックします。新しい製品構成の作成方法については、「製品構成の使用」を参照してください。InstallShield によって、製品構成の下に新しい行が追加されます。設定がタブ上に表示されます。
3. リリースの設定を適宜構成します。
4. リリースを右クリックして、**[ビルド]** をクリックします。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*



タスク ***リリースの作成とビルドを行うには、以下の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーを右クリックしてから、**[新しいリリース]** をクリックします。新しいリリースが追加されます。設定がタブ上に表示されます。
3. リリースの設定を適宜構成します。
4. リリースを右クリックして、**[ビルド]** をクリックします。

InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトにおけるリリースの作成とビルド



タスク ***リリースの作成とビルドを行うには、以下の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーを右クリックして、**[新しいリリース]** をポイントし、作成するメディアの種類をクリックします。エクスプローラーに、新しいリリースが追加されます。設定がタブ上に表示されます。
3. リリースの設定を適宜構成します。
4. リリースを右クリックして、**[ビルド]** をクリックします。

リリースの場所の設定



プロジェクト: *この情報は、次のプロジェクトに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

インストールのディスク イメージ フォルダーは、リリースの場所にビルドされます。圧縮されていないアプリケーション ファイルの保存に必要な、すべての追加のファイルやフォルダーは、ディスク イメージ フォルダーのサブフォルダーに配置されます。リリースの場所は、プロジェクトの場所のサブフォルダーです。

ビルドしたリリースを表示するには、製品名の下にある [ディスク イメージ] をクリックして、[リリース] ビューでそのリリース名をクリックします。リリースの場所を変更していない場合、ビルドされたインストール パッケージは、次のフォルダーに配置されます。

InstallShieldプロジェクト フォルダー¥*プロジェクト名*¥*製品構成*¥*リリース名*¥DiskImages¥Disk1

リリースの場所は、リリース ウィザードの [詳細設定] パネル、または [リリース] ビューの "リリースの場所" プロパティのいずれかで設定できます。

コマンドラインからのリリースのビルド



プロジェクト: *この情報は、次のプロジェクトに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

Windows Installer ベースのプロジェクト、InstallScript プロジェクト、およびアドバンスト UI またはスイート/アドバンスト UI プロジェクトでは、ISCmdBld.exe を使って、コマンドラインからリリースをビルドできます。また、プロジェクトに InstallScript が含まれている場合、リリースがビルドされる前に、ISCmdBld.exe によってコンパイルされます。

ISCmdBld.exe を使ったインストールのビルドは、バッチ ファイルからビルドをするときに便利です。また ISCmdBld.exe を使用してコマンドラインからリリースをビルドすると、リリースをビルドするのに加えて .ipr ファイルと .ipo ファイル (InstallShield Professional で作成されたオブジェクト プロジェクト ファイル) が .ism ファイルに変換されます。



ヒント: *バイナリまたは XML フォーマットの .ism ファイルは、どちらもコマンドライン ビルドで利用できます。コマンドライン ビルドは、アドバンスト UI およびスイート/アドバンスト UI プロジェクト ファイル (.issuite) でも使用できます。*

Standalone Build で ISCmdBld.exe を使って、コマンドラインからリリースをビルドできます。詳細については、「Standalone Build」を参照してください。

ISCmdBld.exe を使用して、コマンドラインからリリースをビルドする



プロジェクト: *この情報は、次のプロジェクトに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

ISCmdBld.exe は、デフォルトで、InstallShield フォルダーの System サブフォルダーに配置されます。ISCmdBld.exe は、インストールされた場所から移動させないでください。

ISCmdBld.exe の構文および利用可能なコマンドライン パラメーターについては、「ISCmdBld.exe」を参照してください。

.ini ファイルでコマンドライン ビルド パラメーターを渡す



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

コマンドラインを使ったビルド中に多くのパラメーターを渡す場合や、同じパラメーターを継続的に渡す場合、.ini ファイルを使うと便利な場合があります。以下は、IsCmdBld.exe を実行して、**MySetup.ini** ファイルで指定したパラメーターと共に リリースをビルドするステートメントの例を示します。

```
ISCmdBld.exe –i "C:¥InstallShield 2022 Projects¥MySetup.ini"
```

コマンドラインでパラメーターを渡すときと同じ情報を .ini ファイルに含める必要があります。このファイルには 4 つのセクションがあります:

- **[Project]**—このセクションには、製品構成の名前およびプロジェクト ファイル (.ism) へのパスの エントリを含めます。パッチをビルドしている場合、ビルド中のパッチ構成の名前のエントリを含めます。
- **[Release]**—このセクションには、圧縮の種類 (圧縮または非圧縮)、ビルド フラグ、Setup.exe の設定、およびリリース名などのリリース構成情報のエントリを含めます。
- **[Mode]**—このセクションには、Silent=yes (リリースのビルド中に、ビルド エラーまたは警告メッセージを抑制する場合) のような、使用可能なオプション エントリを含めます。このセクションでは、ログ ファイルを作成するかどうかを指定することもできます。
- **[BuildLocation]**—このセクションでは、リリースの出力場所をオプションで指定することができます。

一部のセクションは必須ではありません。コマンドラインから直接パラメーターを渡すことに関しては、サイレント ビルドやビルドの場所などの要件のためのパラメーターはオプションです。下の .ini ファイルの例では、これらのパラメーターは [Mode] セクションと [BuildLocation] セクションにあります。デフォルトを使用する場合は、.ini ファイルからこれらのエントリを省略してください。デフォルトでは、ログ ファイルは作成されず、インストールもサイレント モードで実行されません。また、リリースは [オプション] ダイアログ ボックスの [ファイルの場所] タブで指定されたプロジェクトの場所に作成されます。

サンプル .ini ファイル

次のテーブルは、サンプル .ini ファイルに含まれている 4 つのセクションからのサンプル エントリです。サンプル エントリは、各テーブルの最初の列に表示されています。その他の列は、対応するコマンドライン パラメーターと説明です。

[Project] セクションのエントリ

テーブル 1 ・ サンプル エントリ – [Project] セクション (.ini ファイル)

エントリ	対応する ISCmdBld.exe コマンドライン パラメーター	説明
Name="C:¥InstallShield 2022 Projects¥Othello.ism"	–p	.ism ファイルへのパス。
BuildLabel=Label1	–a	ビルドするリリースの製品構成の名前。パラメーターが存在しない場合は、作成されます。
Product=Othello		プロジェクトの名前。このエントリを使って、[一般情報] ビューで指定された値をオーバーライドできます。
PatchConfigName="Version 1.2"	–patch_config	ビルドする [パッチのデザイン] ビューのパッチ構成の名前。
CompileScript=no	–n	Setup.rul がビルドプロセスの一部としてコンパイルされないようにします。  ***メモ・*** *CompileOnly を CompileScript と共に使用することはできません。*

[Release] セクションのエントリ

テーブル 2 ・ サンプル エントリ – [Release] セクション (.ini ファイル)

エントリ	対応する ISCmdBld.exe コマンドライン パラメーター	説明
BuildFlags=flags	–f	カンマで区切られたインストールに含まれるリリース フラグ。
Configuration=COMP	–c	圧縮、非圧縮の違い。
Name=Othello Beta	–r	リリース名。
SetupEXE=yes	–e	Setup.exe ファイルを作成します。

[Mode] セクションのエントリ

テーブル 3 ・ サンプル エントリ – [Mode] セクション (.ini ファイル)

エントリ	対応する ISCmdBld.exe コマンドライン パラメーター	説明
CubFile="C:¥Program Files¥InstallShield¥2022¥Support¥Validation¥MyValidation.cub"	–m	インストールまたはマージ モジュール パッケージのビルドが完了したあと、指定された .cub ファイルを使って検証を実行します。完全パスと名前を .cub ファイルに含めます。 複数の .cub ファイルを使用するには、各パスをセミコロン (;) で区切ります。長いファイル名は引用符で囲んでください。 たとえば、次のエントリは、検証に InstallShield 検証スイート – Windows 8 (ISWin8.cub) および完全 MSI 検証スイート (darice.cub) が必要であることを示します: CubFile="C:¥Program Files¥InstallShield¥2022¥Support¥Validation¥ISWin8.cub;C:¥Program Files¥InstallShield¥2022¥Support¥Validation¥darice.cub"
Silent=yes	–s	サイレント モードで実行します。
StopOnFirstError=yes	–x	最初にエラーが発生した時点でビルドを中止します。
CompileOnly=no	–q3	Setup.rul のみをコンパイルして、以前にビルドされていれば、Setup.inx を .msi ファイル の**Binary** テーブルにストリーミングします。 ***メモ:*** *CompileOnly、BuildTablesRefreshFiles、および BuildTablesOnly エントリは、相互に排他的です。1 つだけ [はい] に設定することができます。* *CompileOnly を CompileScript と共に使用することはできません。*
BuildTablesRefreshFiles=no	–q2	Windows Installer テーブルをビルドしてファイルをリフレッシュします。 ***メモ:*** *CompileOnly、BuildTablesRefreshFiles、および BuildTablesOnly エントリは、相互に排他的です。1 つだけ [はい] に設定することができます。*

テーブル 3 - サンプル エントリ – [Mode] セクション (.ini ファイル) (続き.)

エントリ	対応する ISCmdBld.exe コマンドライン パラメーター	説明
BuildTablesOnly=yes	–q1	リリースの Windows Installer テーブルのみをビルドします。 ***メモ-*** *CompileOnly、BuildTablesRefreshFiles、および BuildTablesOnly エントリは、相互に排他的です。1 つだけ [はい] に設定することができます。*
UpgradeOnly=no	–u	リリースをビルドではなく、アップグレードすることができます。
Verbose=yes	–v	エンジン ログ ファイルを生成。
WarningAsError=yes	–w	ビルド警告すべてをビルドエラーとして扱う。
MSIVersion=2.0	–g	ターゲット システムで必要な Windows Installer のバージョン。 ***メモ-*** *このエントリは Standalone Build のみに適用します。*
DotNetVersion=	–j	ターゲット システムに必要な Microsoft .NET Framework の最小バージョン。このパラメーターはオプションで、デフォルトでは InstallShield がサポートする .NET Framework の最新バージョンに設定されています。 ***メモ-*** *このエントリは Standalone Build のみに適用します。*
DotNetPath="C:¥..."	–t	ビルド マシン上の Microsoft .NET Framework へのパス。 ***メモ-*** *このエントリは Standalone Build のみに適用します。*

テーブル 3 ・ サンプル エントリ – [Mode] セクション (.ini ファイル) (続き.)

エントリ	対応する ISCmdBld.exe コマンドライン パラメーター	説明
SkipValidators=yes	–h	通常ビルドの終わりで実行されるアップグレード検証ツールをオフにすることができます。 ***メモ・** このエントリは Standalone Build のみに適用します。*
MergeModulePath="C:¥..."	–o	マージ モジュール (.msm ファイル) のパスを検索します。 詳細については、「マージ モジュールを含むディレクトリを指定する」を参照してください。 ***メモ・** このエントリは Standalone Build のみに適用します。*
PrerequisitePath="C:¥..."	–prqpath	InstallShield 前提条件ファイル (.prq) の検索パス。 詳細については、「InstallShield 前提条件を含むディレクトリを指定する」を参照してください。 ***メモ・** このエントリは Standalone Build のみに適用します。*
LicCheckTimeOut="40"	–licCheckTimeOut <秒>	このパラメーターを使って、ライセンス サーバーからライセンスを使用可能かどうかチェックする時間帯を設定することができます。

[BuildLocation] セクションのエントリ

テーブル 4 ・ サンプル エントリ – [BuildLocation] セクション (.ini ファイル)

エントリ	対応する ISCmdBld.exe コマンドライン パラメーター	説明
Path="C:¥InstallShield 2022 Projects¥Othello"	–b	リリースの出力場所。

.ini ファイルのエントリについての詳細は、「コマンドラインからのリリースのビルド」を参照してください。

ISBuild.exe を使用してコマンドラインからのビルドする



プロジェクト *この情報は、InstallScript プロジェクトに適用します。*



重要 *ISBuild.exe が使用できなくなったため、ISBuild.exe の代わりに ISCmdBld.exe を使ってビルドをリリースしてください。*

ISBuild.exe (レガシー InstallScript プロジェクトに提供されているコマンドライン ツール) は、ISCmdBld.exe に呼び出されるシェルです。ISCmdBld.exe は、.ipr ファイルと .ipo ファイル (InstallShield Professional を使って作成されたオブジェクト プロジェクト) を .ism ファイルに変換処理して、リリースをビルドします。

ISBuild.exe で使用できるコマンドライン パラメーターの一覧については、「ISBuild.exe」を参照してください。

コマンドラインからの自己展開型実行可能ファイルをビルドする



プロジェクト *この情報は、InstallScript プロジェクトに適用します。*

ReleasePackager.exe を使用して、コマンドラインから自己展開型実行可能ファイルをビルドすることができます。バッチ ファイルからビルドを行うときに便利です。

ReleasePackager.exe は、デフォルトで InstallShield フォルダーの System サブフォルダーに配置されます。ReleasePackager.exe を、インストールされた場所から移動しないでください。

ReleasePackager.exe の構文と利用可能な引数およびオプションについては、「ReleasePackager.exe」を参照してください。

リリースの再ビルド



プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

リリース ウィザードで設定したデフォルトのエントリを使用して、いつでもリリースを再ビルドできます。再ビルドされるのは、現在強調表示されているリリースです。



タスク ***リリースを再ビルドするには、以下のいずれかを行ってください。***

- **[ビルド]** ボタンをクリックします。

- **[ビルド]** メニューで **[ビルド]** をクリックします。
- **[リリース]** ビューで、リリース名を右クリックしてから **[ビルド]** をクリックします。

また、リリース ウィザードを再度実行して、リリースを再ビルドすることもできます。ウィザードでは InstallShield プロジェクト ファイルのリリース設定が保持されるので、リリース自体を削除した場合でも、最後にリリースをビルドした際に設定したオプションがすべてウィザードに表示されます。

新しいインストール パッケージを表示するには、[リリース] ビューで適切なリリースのディスク イメージをクリックします。デフォルトのリリースの場所を変更していない場合、ビルドされたインストール パッケージは、次のフォルダーに配置されます。

C:¥InstallShield 2022 プロジェクト¥*プロジェクト名*¥*プロジェクト構成*¥*リリース名*¥DiskImages¥Disk1



メモ *リリースをビルドする際、Windows エクスプローラーが* Disk1 *フォルダーまたはサブフォルダーをポイントしていないことを確認してください。*Disk1 *フォルダーがポイントされていると、ビルド処理は終了しません。Windows Explorer がサブフォルダーにアクセスしていると、ビルドでエラーが発生します。*

クイック ビルドを実行する



プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

インストールをテストするとき、ファイルに変更がなければインストールのすべてを頻繁にビルドする必要はありません。InstallShield では、[テーブルのビルドのみ] と [テーブルのビルドとファイルのリフレッシュ] の 2 つのクイック ビルド オプションが提供されています。

テーブルのみをビルドする

このオプションの名前が示すように、Windows Installer のテーブルだけがビルドされます。このセットアップをまだビルドしていない場合、新しい .msi ファイルが作成されますが、インストールにはファイルが追加されません。インストールを既にビルドしている場合、.msi ファイルは、すべてのテーブルがビルドされたとき更新されますが、ファイルは転送されません。このオプションを使用して、インストールのユーザー インターフェイスをテストすることができます。



タスク ***インストールの Windows Installer テーブルのみをビルドするには、以下のいずれかを実行します。***

- **[ビルド]** メニューで **[テーブルのみをビルド]** をクリックします。

- **[リリース]** ビューで、ビルドするリリースを右クリックしてから **[テーブルのみをビルド]** をクリックします。

テーブルのビルドとファイルのリフレッシュ

すべてのインストール ファイルを新しい場所に転送せずに、ほぼ完全にビルドを行う場合、.msi ファイルを再ビルドして、Files テーブルを更新するビルドを実行できます。これにより、新しいファイルや変更されたファイルがインストールに含まれます。変更されたファイルは、サイズまたはタイムスタンプが、ビルドにすでに含まれるコピーと異なる場合にのみ更新されます。削除されたファイルへの参照は、インストールから削除されますが、ファイルは、ビルドの場所に残ります。この種類のビルドは、ビルドが完全に実行された後でのみ実行されます。また、この種類のビルドは、メディアが非圧縮のネットワークイメージである場合だけ作動します。

タスク

Windows Installer テーブルのみをビルドして、インストールのファイルをリフレッシュするには、以下のいずれかを実行します。

- **[ビルド]** メニューで **[テーブルのビルドとファイルのリフレッシュ]** をクリックします。
- **[リリース]** ビューで、ビルドするリリースを右クリックしてから **[テーブルのビルドとファイルのリフレッシュ]** をクリックします。

バッチ ビルドを実行する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

構成の異なる複数のリリースを同時にビルドする必要がある場合があります。この機能をバッチビルドと呼びます。

タスク

InstallShield でバッチ ビルドを実行するには、次の手順を実行します。

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. 基本の MSI、InstallScript MSI、マージ モジュール プロジェクトの場合、[製品構成] を右クリックして **[バッチ ビルド]** を選択します。

   InstallScript または InstallScript オブジェクト プロジェクトの場合、**[リリース]** エクスプローラーを右クリックして **[バッチ ビルド]** を選択します。

[バッチ ビルド] ダイアログが開き、ビルドする製品構成とリリースを選択するようメッセージが表示されます。

ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する



エディション: *この機能は、InstallShield の Premier Edition で提供されています。*



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *スイート/アドバンスト UI*

プロジェクトによって異なる情報がある場合は、その内容が記載されています。

InstallShield では、ビルド プロセスの様々な段階で実行するコマンドを指定できます。たとえば、基本の MSI プロジェクトを作成中の場合、ビルド時に InstallShield が作成する .msi パッケージを、デジタル署名が行われて Setup.exe ファイルにストリームされる前に変更したい場合があります。ビルド中、適切な時点で .msi パッケージに必要な変更を行うためのスクリプトを実行するコマンドを指定できます。

コマンドを入力するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンド用に定義された以下の変数を使用することもできます:

テーブル 5 ・ ビルド イベント変数

変数	プロジェクトの種類	説明
<ISReleaseName>	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、スイート/アドバンスト UI	この変数は、InstallShield がビルド中のリリースの名前を示します。
<ISProductConfigName>	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	この変数は、InstallShield がビルド中のリリースを含む製品構成の名前を示します。(InstallScript プロジェクトの場合、この名前は常に *Media* です。)
<ISReleasePath>	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、スイート/アドバンスト UI	この変数は、InstallShield がビルド中のリリースがビルドされる場所を示します。これは、[リリース] ビューにあるリリースの [ビルド] タブにある "リリースの場所" 設定で行います。

テーブル 5 ・ ビルド イベント変数 (続き.)

変数	プロジェクトの種類	説明
<ISReleaseUsesShallowFolderPaths>	基本の MSI、InstallScript MSI	この変数は True または False に設定され、InstallShield がリリースをビルドする際にシャロー フォルダー ディレクトリ構造を使用するかどうかを示します。

ビルド時に、InstallShield は <ISReleaseName>、<ISProductConfigName>、<ISReleasePath>、および <ISReleaseUsesShallowFolderPaths> の一時環境変数を設定します。InstallShield はまた、プロジェクトに含まれるすべての定義済みパス変数の一時環境変数も設定します。バッチ ファイル内のこれらの環境変数の値の 1 つを、その変数名をパーセント記号 (%) で囲んで参照することができます。例:

set PATH = %<ISReleaseName>%;%PATH%

ビルドが終了すると、設定された一時環境変数は削除されます。



タスク ***ビルド前、ビルド中、またはビルド後に実行する 1 つ以上のコマンドを指定するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、設定を行うリリースを選択します。
3. **[イベント]** タブをクリックします。
4. 以下の 1 つ以上の設定で、InstallShield が実行するコマンドを入力します:
   - ビルド前のイベント
   - ファイルの準備イベント
   - 圧縮前のイベント (このイベントは、スイート/アドバンスト UI プロジェクトでは使用できません。)
   - ビルド後のイベント

   コマンド プロンプトから実行されるのと同じ要領で各コマンドを入力します。つまり、たとえばスペースを含むパスを指定する場合には、そのパスを引用符で囲みます。

   各設定の詳細については、「リリースの [イベント] タブ」を参照してください。

   これらの任意の設定に 1 つ以上のコマンドを指定するには、この設定で省略記号ボタン (...) をクリックします。1 つ以上のコマンドを入力することができるダイアログ ボックスが開きます。1 行につき 1 つのコマンドを入力します。

1 つのイベント設定に対して 1 つ以上のコマンドを入力すると、InstallShield はビルド時に、その設定でリストされている順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。



ヒント: *詳細ビルド ログには、ビルド時に実行される各ビルド イベントが表示されます。ビルド イベントのトラブルシューティングには、詳細ログ ファイルを生成して、"Launching prebuild イベント"、"Launching precompression build イベント"、および "Launching postbuild イベント" などの文字列を検索します。ログ ファイルのこれらの領域には、ビルド時に実行されたコマンドについての詳細がリスト表示されます。*

詳細ビルド ログを生成するには、コマンドラインから IsCmdBld.exe *を起動するときに、–v パラメーターを渡します。*

ファイルの準備イベントを使って、署名されていない InstallShield ファイルに署名を行う



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[メディア] の **[リリース]** ビューで、リリースの **[イベント]** タブにある **"ファイルの準備イベント"** 設定を使って、ユーザーは InstallShield ビルド処理中に生成されたすべての内部ファイルに署名を行う独自のカスタム スクリプトを呼び出すことができます。これは、独自のスクリプトまたは実行可能ファイルを実行して InstallShield のサインアウトを構成したい場合に便利です。

ファイルの準備イベント コマンドは、プロジェクトのビルド中に、InstallShield の内部サポートファイルが準備され、それらのファイルが MSI または Setup.exe file にストリームされる前に呼び出されます。

プロジェクトのビルド中、InstallScript MSI プロジェクトではファイルの準備イベント コマンドが 2 回呼び出されますが、その他のプロジェクトでは一回だけ呼び出されます。次の InstallShield プロジェクトをビルド中に署名されていない InstallShield ファイルの場所:

テーブル 6 ・ 署名されていないファイルの場所

プロジェクト	署名されていないファイルの場所
基本の MSI	• _XXXXXXXXXX.clr ー プロジェクトに管理対象カスタム アクションまたは PowerShell カスタム アクションが含まれている場合、内部フォルダーで署名なしの cir ファイルを使用できます。このフォルダーは、プロジェクトのビルド中に製品構成フォルダー内に作成されます。 • XXXX.tmp ー プロジェクトでアプリケーションまたはショートカットのアイコンを設定するように構成されている場合、内部フォルダーに署名されていない一時ファイルが作成されます。 • ISSetup.dll ー プロジェクトに InstallShield カスタム アクションが含まれている場合、内部フォルダーに署名されていない dll ファイルが作成されます。
InstallScript	• _isuser_0xNNNN.dll ー InstallScript プロジェクト ダイアログが変更された場合、Script File フォルダーの隣にあるプロジェクト フォルダーに署名されていない dll ファイルが作成されます。 • ISSetup.dll ー 署名されていない dll が Disk1 フォルダーに作成されます。

テーブル 6 ・ 署名されていないファイルの場所

プロジェクト	署名されていないファイルの場所
InstallScript MSI	• _isuser_0xNNNN.dll ― InstallScript MSI プロジェクト ダイアログが変更された場合、Script File フォルダーの隣にあるプロジェクト フォルダーに署名されていない dll ファイルが作成されます。 • _XXXXXXXXXX.clr ― プロジェクトに管理対象カスタム アクションまたは PowerShell カスタム アクションが含まれている場合、内部フォルダーで署名なしの cir ファイルを使用できます。このフォルダーは、プロジェクトのビルド中に製品構成フォルダー内に作成されます。 • XXXX.tmp ― プロジェクトでアプリケーションまたはショートカットのアイコンを設定するように構成されている場合、内部フォルダーに署名されていない一時ファイルが作成されます。 • ISSetup.dll ― 署名されていない dll ファイルが内部フォルダーに作成されます。この DLL は InstallScript MSI プロジェクトに必要です。 ***メモ・*** *ファイルの準備イベント コマンドは、InstallScript MSI プロジェクトで 2 回呼び出されます。*
スイート/アドバンスト UI	• ISSetup.dll ― プロジェクトに InstallScript イベントが含まれている場合、内部フォルダーに署名されていない dll ファイルが作成されます。このフォルダーは、プロジェクトのビルド中にリリース フォルダー内に作成されます。 • Setup64.exe ― プロジェクトに x64 パッケージまたは条件が含まれている場合、内部フォルダーに署名されていない exe ファイルが作成されます。

ビルドエラーと警告を解決する

プロジェクト・ *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

リリースをビルドすると、InstallShield ユーザー インターフェイスの下の部分に出力ウィンドウが開きます。このウィンドウの [タスク] タブには、ビルド プロセス中に発生したすべてのビルドエラーおよび警告が一覧で表示されます。



タスク ***発生したビルドエラーおよび警告を解決する方法を調べるには、次のどちらかを行なってください。***

- **[出力]** ウィンドウの **[タスク]** タブでエラーコードをクリックすると、インターネット ブラウザーを起動し、選択されたビルド エラーまたは警告に関するナレッジ ベース記事およびドキュメントのトピックを表示することができます。
- ビルド エラーと警告の解決方法に関するヒントを見ることができる「ビルド エラーと警告」をご覧ください。

コマンドラインからプロジェクトの設定を変更する



プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallShield オートメーション インターフェイスでは、VBScript スクリプトや Visual Basic アプリケーションなどを使って、無人のビルド プロセスから多数のプロジェクト設定をクエリおよび変更できます。

オートメーション インターフェイスを使ってプロジェクトにアクセスするスクリプトは、次のような枠組となります (このスクリプトを Framework.vbs という名前のテキストファイルにコピーしてから、ファイルのアイコンをダブルクリックすることができます)。

```
Set oProject = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
oProject.OpenProject "C:¥MySetups¥MyProject.ism"

' ここでクエリと変更を行う

oProject.SaveProject ' プロジェクトを変更する場合のみ必要
oProject.CloseProject
```

ビルドのキャンセル



プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*

- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*



タスク ***既に起動したビルドをキャンセルするには、以下の手順に従います:***

ツールバー上の **[ビルドの中止]** ボタンをクリックします。

Standalone Build

InstallShield には、Standalone Build モジュールがあり、InstallShield のインストールをコンパイルする部分 (および含める再配布可能ファイル) のみインストールすることができます。Standalone Build と再配布可能ファイルのビルド マシンへのインストール方法については、「Standalone Build をビルド マシンにインストールする」を参照してください。

InstallShield 2025 Standalone Build は、同じマシン上で別のメジャー バージョンの Standalone Build と共存することができます。ただし、異なるフォルダーに格納されている必要があります。

プロジェクトのアップグレード

Standalone Build は、InstallShield Developer 7.0 またはそれ以降を使って作成されたプロジェクトを自働的にアップグレードします。ただし、InstallShield for Windows Installer を使って作成されたプロジェクトをアップグレードすることはできません。

コマンドライン ビルドの使用

コマンドラインから Standalone Build を実行することができます。詳細については、「スタンドアロン コマンドライン ビルド」を参照してください。

スタンドアロン オートメーション インターフェイスの利用

Standalone Build は、オートメーション インターフェイスのスタンドアロン バージョン (スタンドアロン オートメーション インターフェイス) を含みます。

Standalone Build をビルド マシンにインストールする



エディション: *複数言語インストールのサポートは、InstallShield Premier Edition のみで提供されています。*



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

- *マージ モジュール*
- *スイート/アドバンスト UI*

Standalone Build のインストール

Standalone Build のインストールは、単一の圧縮済み実行可能ファイルで構成されています。複数言語インストールのビルドに必要なすべてのファイルが含まれています。



タスク

Standalone Build をビルド マシンにインストールするには、次の手順を実行します。

1. Standalone Build インストールの実行可能ファイルを見つけるには、次の手順を実行します。
   - InstallShield DVD をお持ちの場合、ファイルは DVD に含まれているので DVD ブラウザーを使って見つけることができます。
   - InstallShield をダウンロードされた場合、Standalone Build インストール ファイルは InstallShield のダウンロードおよびライセンス ガイド (http://www.installshield.com/instructions/sab.asp) の手順にに従ってダウンロードできます。
2. 実行可能ファイルをダブルクリックして、インストールを実行します。Standalone Build のライセンシング構成に関する詳細は、InstallShield ダウンロード & ライセンス ガイド (http://www.installshield.com/instructions/sab.asp) を参照してください。

インストール中、[カスタム セットアップ] ダイアログが表示され、デフォルトで無効化されている Standalone Build の 2 つの機能が表示されます。これらの機能を使用するには、有効化します。

- **オートメーション インターフェイス**—Standalone Build でオートメーション インターフェイスを使用するのに必要なファイルをインストールおよび登録します。
- **InstallScript オブジェクト サポート**—InstallScript オブジェクトを含む InstallScript プロジェクトのビルドを可能にする InstallScript オブジェクトのサポート ファイルをインストールおよび登録します。

Standalone Build のインストール中に登録されるファイル

Standalone Build をインストールすると、以下のマージ モジュールで既存しないものがインストールおよび登録されます。

- ATL 3.0 (ATL.msm)
- Microsoft C ランタイム ライブラリ (MSVCRT.msm)
- Microsoft C Runtime Library 7.1 (VC_User_CRT71_RTL_X86_——.msm)
- MSXML3 (msxml3_wim32.msm)
- MSXML4 (msxml4sxs32.msm または msxml4sys32.msm)
- Visual C++ 8.0 ATL (x86) WinSXS MSM (Microsoft_VC80_ATL_x86.msm)
- Visual C++ 8.0 ATL.Policy (x86) WinSXS MSM (policy_8_0_Microsoft_VC80_ATL_x86.msm)
- Visual C++ 8.0 CRT (x86) WinSXS MSM (Microsoft_VC80_CRT_x86.msm)
- Visual C++ 8.0 CRT.Policy (x86) WinSXS MSM (policy_8_0_Microsoft_VC80_CRT_x86.msm)

Standalone Build インストールはまた、Microsoft Scripting Runtime Library (scrrun.dll) が既存しない場合に、これをインストールおよび登録します。これは、[SystemFolder] にインストールされます。

Standalone Build をインストールする代わりに、1 台目のマシンから別のマシンに Standalone Build ファイルをコピーした場合で、コピー先のマシンにこれらのマージ モジュールに含まれているテクノロジーと scrrun.dll ファイルが既存しないときは、手動でインストールしなくてはなりません。このマージ モジュール テクノロジー用のインストール パッケージは、Microsoft Web サイト (http://www.microsoft.com) から取得できます。

オプションのオートメーション サポート機能のインストール中に登録されるファイル

Standalone Build インストールのオートメーション インターフェイス機能は、Standalone Build でオートメーション インターフェイスを使用するときに必要なファイルをインストールおよび登録します。登録する必要があるファイルの一覧は次のとおりです:

- *Program Files フォルダー*¥Common Files¥InstallShield¥Shared¥IsmAuto.tlb
- *Program Files フォルダー*¥Common Files¥InstallShield¥Shared¥Ismmupdater.tlb
- *Program Files フォルダー*¥Common Files¥InstallShield¥Shared¥ISWIBuild.tlb
- *Program Files フォルダー*¥Common Files¥InstallShield¥Shared¥IsAppServices.tlb
- *Program Files フォルダー*¥Common Files¥InstallShield¥Shared¥IsUpgrade.tlb
- *Standalone Build Program Files フォルダー*¥System¥ISWiAutomationAutomation Interface Version.dll

Standalone Build をインストールする代りに、1 台目のマシンから別のマシンに Standalone Build ファイルをコピーした場合、コピー先のマシンの Standalone Build でオートメーション インターフェイスを使用するには、これらのファイルを手動で登録しなくてはなりませんん。

オプションの InstallScript オブジェクト サポート機能のインストール中に登録されるファイル

InstallScript オブジェクト サポート機能は、InstallScript オブジェクトを含む InstallScript プロジェクトのビルドを可能にする InstallScript オブジェクトのサポート ファイルをインストールおよび登録します。登録する必要があるファイルの一覧は次のとおりです:

- *Standalone Build Program Files フォルダー*¥System¥DObjAutomation Interface Version.dll
- *Standalone Build Program Files フォルダー*¥System¥ISBE.dll
- *Standalone Build Program Files フォルダー*¥System¥ISMKAutomation Interface Version.dll
- *Standalone Build Program Files フォルダー*¥System¥StockWiz.dll

Standalone Build をインストールする代りに、1 台目のマシンから別のマシンに Standalone Build ファイルをコピーした場合、コピー先のマシンで InstallScript オブジェクトを含む InstallScript プロジェクトをビルドするには、これらのファイルを手動で登録しなくてはなりませんん。

Standalone Build のファイルを手動で登録する

前述の Standalone Build 用の DLL ファイルの 1 つを手動で登録するには、System32 フォルダーにインストールされている Regsvr32.exe を使用します。

.tlb ファイルの 1 つを手動で登録するには、以下のファイルを使用します:

Standalone Build Program Files フォルダー¥System¥RegTypLib.exe

Standalone Build 用に再配布可能ファイルをビルド マシンへ追加する

マージ モジュール、オブジェクト、および InstallShield 前提条件を含み、再配布可能ファイルのほとんどは非常に多くのハード ディスク容量を使用するため、Standalone Build と共に自動的にはインストールされません。これによって、作成中の InstallShield プロジェクトに必要な再配布可能ファイルのみをビルド マシンにインストールすることができます。唯一の例外は .NET Framework 2.0 再配布可能ファイルで、これは自動的にインストールされます。



ヒント: *Standalone Build は、InstallShield がプログラム ファイルに使用するディレクトリ構造と同じ基本のディレクトリ構造を使用します。これにより、InstallShield があるマシンから再配布可能ファイルや他のファイルを Standalone Build があるマシンにコピーするとき、同じ相対パスを使用できるようになりました。たとえば、ファイルが InstallShield があるマシンの* InstallShield Program Files フォルダー¥Modules¥i386 *ディレクトリにある場合、そのファイルを Standalone Build があるマシンの* Standalone Build Program Files フォルダー¥Modules¥i386 *ディレクトリにコピーします。*



タスク ***.NET 再配布可能ファイルをビルド マシンに追加するには、次の手順を実行します。***

1. InstallShield の完全バージョンを持つマシン上で、必要な再配布可能ファイルを参照します。通常ファイルは、InstallShield Program Files フォルダー¥Redist または Redist フォルダーのサブフォルダーに配置されています。InstallShield 再配布可能ファイルがマシン上で利用できる状態では無い場合、まずこれをダウンロードします。詳細については、「再配布可能ファイル ダウンローダー ウィザード」を参照してください。
2. InstallShield を持つマシンから、ビルドマシン上にある Standalone Build Program Files フォルダー へ再配布可能ファイルをコピーします。



タスク ***基本の MSI または InstallScript MSI プロジェクト用の再配布可能ファイルをビルド マシンに追加するには、次の手順を実行します。***

1. InstallShield の完全バージョンを持つマシン上で、必要な再配布可能ファイルを参照します。InstallShield 再配布可能ファイルがマシン上で利用できる状態では無い場合、まずこれをダウンロードします。詳細については、「再配布可能ファイルをコンピューターにダウンロードする」を参照してください。
   - InstallShield マージ モジュール の 2 つのデフォルトの場所は、次の通りです。

     InstallShield Program Files フォルダー¥Modules¥i386

     Common Files¥Merge Modules
   - InstallShield オブジェクトのデフォルトの場所は、次の通りです。

     InstallShield Program Files フォルダー¥Objects
   - すべての InstallShield セットアップ前提条件ファイルは、次の場所に格納されています:

     InstallShield Program Files フォルダー¥SetupPrerequisites
   - 独自のマージ モジュールを InstallShield でビルドした場合、[オプション] ダイアログ ボックスの [マージ モジュール] タブにリストされている場所のどこかに格納されています。
2. 再配布可能ファイルを InstallShield を持つマシンからビルド マシンの適切な場所へコピーします。
   - マージ モジュールを次のフォルダーに追加します。

Standalone Build Program Files Folder¥Modules¥i386

- オブジェクトを次のフォルダーに追加します。

  Standalone Build Program Files Folder¥Objects

- InstallShield セットアップ前提条件を次のフォルダーに追加します:

  Standalone Build Program Files Folder¥SetupPrerequisites



タスク ***InstallScript プロジェクト用の InstallShield 前提条件をビルド マシンに追加するには、以下の手順に従います:***

1. InstallShield の完全バージョンがインストールされているマシン上で、必要な InstallShield 前提条件を参照します。InstallShield 再配布可能ファイルがマシン上で利用できる状態では無い場合、まずこれをダウンロードします。詳細については、「再配布可能ファイルをコンピューターにダウンロードする」を参照してください。

   すべての InstallShield セットアップ前提条件ファイルは、次の場所に格納されています:

   InstallShield Program Files フォルダー¥SetupPrerequisites

2. InstallShield 前提条件を InstallShield がインストールされているマシンからビルド マシンの適切な場所へコピーします。

   Standalone Build Program Files Folder¥SetupPrerequisites



タスク ***ビルド マシンに InstallScript プロジェクト用の InstallShield 前提条件の 1 つを追加するには、以下の手順を実行します:***

1. 再配布可能ファイルのインストールをダウンロードします。インストールは FlexNet Connect を通して提供されます。詳細については、「InstallShield のアップデートを取得する」を参照してください。
2. 再配布可能ファイル インストールをビルド マシン上で実行します。InstallShield ウィザードが [インストール先の選択] ダイアログを表示したら、次の場所を参照します。

   Standalone Build Program Files Folder¥ObjectsPro

   ObjectsPro フォルダーがまだ作成されていない場合、このフォルダーを追加する必要があります。

InstallScript オブジェクト インストールは、必要なオブジェクト ファイルをインストールおよび登録します。



タスク ***ビルドマシンに InstallScript 再配布可能ファイルまたはサードパーティ再配布可能ファイルの 1 つを追加するには、次の手順を実行します。***

1. 再配布可能ファイルを参照します。
2. これをビルド マシン上の次の場所に配置します。

   Standalone Build Program Files Folder¥ObjectsPro

   ObjectsPro フォルダーがまだ作成されていない場合、このフォルダーを追加する必要があります。

スタンドアロン コマンドライン ビルド

Standalone Build では、ISCmdBld.exe を使って、コマンドラインからリリースをビルドすることができます。手順については、「コマンドラインからのリリースのビルド」を参照してください。

ISCmdBld.exe がサポートするコマンドライン パラメーターの一覧については、「ISCmdBld.exe」を参照してください。

スタンドアロン オートメーション インターフェイス

Standalone Build は、オートメーション インターフェイスのスタンドアロン バージョン (InstallShield スタンドアロン オートメーション インターフェイス) を含みます。このインターフェイスは、InstallShield と共にインストールされる ISWiAutomationAutomation Interface Version.dll と同じファイルを使用しますが、インストールされる場所は異なります。



メモ *Standalone Build を InstallShield と同じマシンにインストールする場合、最初にインストールされた* ISWiAutomation*Automation Interface Version*.dll *ファイルが登録されます。詳細については、「Standalone Build と InstallShield を同一のマシン上にインストールする」を参照してください。*

Standalone Build の将来のバージョンには、新しい ProgID および GUID が与えられます。これにより、複数のバージョンのスタンドアロン オートメーション インターフェイスを同じビルド マシンにサイドバイサイドでインストールすることができます。

ISWiAutomationAutomation Interface Version.dll の登録

スタンドアロン オートメーション インターフェイスは、ISWiAutomationAutomation Interface Version.dll というファイルに含まれています。Standalone Build を Standalone Build のインストールを起動してインストールした場合、このファイルは既に登録されています。インストールを実行する代りに、Standalone Build のファイルをコピーした場合、System32 フォルダーにインストールされている Regsvr32.exe を使って、このファイルを手動で登録する必要があります。ISWiAutomationAutomation Interface Version.dllファイルは、Standalone Build Program Files フォルダーの System サブフォルダーにインストールされます。

Standalone Build と InstallShield を同一のマシン上にインストールする

ほとんどの場合、InstallShield がインストールされているマシン上に Standalone Build がインストールされることはありません。同じマシン上に両方をインストールする場合は、スタンドアロン オートメーション インターフェイスは InstallShield が使用するのと同じ ISWiAutomationAutomation Interface Version.dll ファイル (ただし、異なる場所にインストールされている) を使用する点にご注意ください。また、スタンドアロン オートメーション インターフェイスと InstallShield は同じ ProgID (IswiAutoAutomation Interface Version.ISWiProject) を使用します。このため、適切な ProgID を反映させるためにコードを変更する必要なく、スタンドアロン オートメーション インターフェイスと InstallShield の両方で同じオートメーション スクリプトを使用できます。

InstallShield または Standalone Build のどちらかを最初にインストールしたとき、インストールが ISWiAutomationAutomation Interface Version.dll を登録します。次にもう 1 つの方のツールをインストールすると、2 番目のインストールは ISWiAutomationAutomation Interface Version.dll が既に登録済みであることを検出して、再登録を行いません。ISWiAutomationAutomation Interface Version.dll の 1 つのインスタンスが登録されているかぎり、Standalone Build と InstallShield の両方でオートメーション インターフェイスを使用できます。

最初にインストールしたインスタンスをアンインストールすると、インストールは ISWiAutomationAutomation Interface Version.dll を登録解除します。このため、残された製品でオートメーション インターフェイスを使用するためには ISWiAutomationAutomation Interface Version.dll を手動で登録しなくてはなりません。たとえば、InstallShield をインストールしてから Standalone Build をインストールした後、InstallShield をアンインストールした場合、Standalone Build 用に ISWiAutomationAutomation Interface Version.dll ファイルを手動で登録しなくてはなりません。これを怠った場合、スタンドアロン オートメーション インターフェイスを使用することはできません。

ISWiAutomationAutomation Interface Version.dll を手動で登録するには、System32 フォルダーにインストールされている Regsvr32.exe を使用します。ISWiAutomationAutomation Interface Version.dll ファイルは、InstallShield および Standalone Build Program Files フォルダーの System サブフォルダーにインストールされます。

Microsoft ビルド エンジン (MSBuild)

InstallShield は、.NET Framework に含まれている Microsoft ビルド エンジン (MSBuild) をサポートします。MSBuild サポートを利用して、Visual Studio がインストールされていないビルド ラボ環境で InstallShield プロジェクトと共に Visual Studio ソリューションをビルドすることができます。

概要

MSBuild は、Visual Studio に依存するビルドを削除するように設計された拡張ビルド フレームワークです。.NET Framework は InstallShield Standalone Build に似た役割で機能し、コマンドラインまたは MSBuild の他のホストからプロジェクトまたはソリューションをビルドすることができます。MSBuild についての詳細は、MSDN ライブラリを参照してください。

MSBuild のタスク

MSBuild の柔軟性および拡張性は、タスクと呼ばれる細かくグループ分けされた内部ビルド手順を通して制御されます。MSBuild と共に発送されるタスクの 1 つは Csc と呼ばれ、Visual C# プロジェクトからのコードをコンパイルすることができます。InstallShield は、InstallShield という名前の MSBuild タスクをインストールします。このタスクは、InstallShield プロジェクトと、プロジェクトのデフォルトのビルド ステップを提供するターゲット ファイルをビルドします。このカスタマイズされたタスクとターゲット ファイルによって、MSBuild は InstallShield プロジェクトを Visual Studio ソリューションの一部としてビルドするのに必要なすべてのアクションを実行することができます。

以下のテーブルは、InstallShield タスクのパラメーターの説明です。[プロジェクトの種類] 列には、該当のプロジェクト タイプが表示されます。

テーブル 7 ・ MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
InstallShieldPath	文字列	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	このパラメーターは、InstallShield アプリケーションを含むフォルダーへのパスを指定します。

テーブル 7 ・ MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
プロジェクト	文字列	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	このパラメーターは、プロジェクト ファイル (.ism) の場所を指定します。
ProductConfiguration	文字列	基本の MSI、InstallScript MSI、マージ モジュール	このパラメーターは、リリースの製品構成を指定します。
ReleaseConfiguration	文字列	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	このパラメーターは、リリース名を指定します。 ***メモ・*** *ReleaseConfiguration または PatchConfiguration を指定します。これらのいずれかを指定しなかった場合、または両方を指定した場合、ビルド エラーが発生します。*
PatchConfiguration	文字列	基本の MSI、InstallScript MSI	このパラメーターは、ビルド中の [パッチのデザイン] ビューでのパッチ構成の名前を指定します。 ***メモ・*** *ReleaseConfiguration または PatchConfiguration を指定します。これらのいずれかを指定しなかった場合、または両方を指定した場合、ビルド エラーが発生します。*
OutDir	文字列	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	このパラメーターは、出力先フォルダーとファイルの保存先フォルダーへの完全修飾パスを指定します。
MergeModulePath	String[ ]	基本の MSI、InstallScript、InstallScript MSI	このパラメーターは、プロジェクトで参照するマージ モジュール (.msm ファイル) を含むフォルダーを指定します(複数指定可)。 InstallShield では、マージ モジュールを含むフォルダーを指定するその他の方法も提供されています。詳細については、「マージ モジュールを含むディレクトリを指定する」を参照してください。

テーブル 7 ・ MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
PrerequisitePath	String[ ]	基本の MSI、InstallScript、InstallScript MSI	このパラメーターは、プロジェクトで参照される InstallShield 前提条件ファイル (.prq ファイル) を含むフォルダーをセミコロンで区切って指定します(複数指定可)。 InstallShield では、InstallShield 前提条件ファイル ファイルを含むフォルダーを指定するその他の方法も提供されています。詳細については、「InstallShield 前提条件を含むディレクトリを指定する」を参照してください。
ReleaseFlags	String[ ]	基本の MSI、InstallScript MSI	このパラメーターを使用して、リリースに含めるリリース フラグを指定できます。複数のフラグはカンマで区切ります。
PathVariables	ITaskItem[ ]	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	このパラメーターを使用して、プロジェクトのパス変数をオーバーライドすることができます。このパラメーターを使用して、パス変数へのパスを指定したり、PathVariable サブ要素を使用して、パス変数の名前の値を定義することができます。 同じ Visual Studio ソリューション フォルダー内にある姉妹プロジェクトのファイルを参照するには、VSSolutionFolder と呼ばれる定義済みのパス変数を使用します。詳細については、「Visual Studio ソリューションで VSSolutionFolder パス 変数を使用する」を参照してください。 MSBuild ITaskItem[ ] プロパティについての詳細は、MSDN ライブラリを参照してください。
PreprocessorDefines	ITaskItem[ ]	基本の MSI、InstallScript、InstallScript MSI	このパラメーターを使用して、プロジェクトのプリプロセッサ定義を追加またはオーバーライドすることができます。このパラメーターを使用して、プリプロセッサ定義の定義を指定したり、Token サブ要素を使用して、プリプロセッサ定義の名前の値を定義することができます。 MSBuild ITaskItem[ ] プロパティについての詳細は、MSDN ライブラリを参照してください。

テーブル 7 • MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
OutputGroups	ITaskItem[ ]	基本の MSI、InstallScript、InstallScript MSI、マージモジュール	このパラメーターは、Visual Studio からのプロジェクト出力グループを指定します。このパラメーターを使用して、プロジェクト出力グループのソースの場所へのパスを指定したり、次のリストにあるサブ要素を使用して、これらの追加の値を定義することができます。 • **Name**—プロジェクトの名前 • **OutputGroup**—プロジェクト出力グループの名前 • **TargetPath**—プロジェクト出力グループのターゲット パス (ソースの場所とは異なります) MSBuild ITaskItem[ ] プロパティについての詳細は、MSDN ライブラリを参照してください。

テーブル 7 ・ MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
ビルド	文字列	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	このパラメーターは、実行するビルドのタイプを指定します。次のビルドのタイプから選択することができます。 • **完全**—このオプションを選択すると、全リリースがビルドされます。 • **コンパイル**—このオプションを選択すると、Setup.rul スクリプト ファイルがコンパイルされます。また、基本の MSI およびマージ モジュール プロジェクトの場合、このオプションは ISSetup.dll を .msi パッケージの Binary テーブルにストリームします。 • **テーブル**—データベース用の Windows Installer テーブルのみをビルドします。このセットアップをまだビルドしていない場合、新しい .msi ファイルが作成されますが、インストールにはファイルが追加されません。インストールをすでにビルドしている場合、すべてのテーブルがビルドされたときに .msi ファイルが更新されますが、ファイルは転送されません。理想的には、このオプションは、インストールのユーザー インタフェースをテストするときに使用します。このオプションは、基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトで適用します。 • **TablesAndFiles**—基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトの場合、Windows Installer テーブルをビルドしてファイルを更新します。この種類のビルドは、ビルドが完全に実行された後でのみ実行されます。また、この種類のビルドは、メディアが非圧縮のネットワークイメージである場合だけ作動します。 InstallScript プロジェクトの場合、前回のビルドの実行以後に変更になった部分のみを再ビルドします。このパラメーターが使用されない場合、ファイル メディア ライブラリ全体が再ビルドされます。 • **UpgradeOnly**—このオプションによって、InstallShield の以前のバージョンで作成された基本の MSI プロジェクトをアップグレードすることができるようになります。ビルドはできません。

テーブル 7 ・ MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
BuildCompressed	ブール値	基本の MSI、InstallScript MSI、マージ モジュール	このパラメーターは、リリースを単一ファイルに圧縮するか、または圧縮せずに複数ファイルとして残すかを指定します。
BuildSetupExe	ブール値	基本の MSI、InstallScript MSI、マージ モジュール	基本の MSI および InstallScript MSI プロジェクトの場合、このパラメーターは、インストールと同時に Setup.exe ファイルを作成するかどうかを指定します。 マージ モジュール プロジェクトの場合、このオプションを使って、マージ モジュールをビルドするかどうか、ビルドする場合、それをマージ モジュール フォルダーにコピーするかどうか、または、コピーせずに単にマージ モジュールをビルドするかどうかを指定できます。
ProductVersion	文字列	基本の MSI、InstallScript MSI、マージ モジュール	このパラメーターは、製品バージョンを指定します。これは、製品バージョンのビルド バージョン (3 番目のフィールド) を増加するときに、特に便利です。 有効な製品バージョン番号については、「製品バージョンを指定する」を参照してください。
PropertyOverrides	ITaskItem[ ]	基本の MSI、InstallScript MSI、マージ モジュール	このパラメーターを利用すると、Windows Installer プロパティの値をオーバーライドしたり、プロパティが存在しないとき、それを新規作成したりできます。このパラメーターを利用するには、値に新しいプロパティ値があり、かつメタデータ Property がそのプロパティの名前であるアイテムのプロパティー覧を含めます。 このパラメーターは、InstallShield タスクの PropertyOverrides プロパティに InstallShieldPropertyOverrides ItemGroup パススルーとして露出されます。
RunMsiValidator	ITaskItem[ ]	基本の MSI、InstallScript MSI、マージ モジュール	このパラメーターを利用すると、ビルドが完了したあと、インストール パッケージおよびマージ モジュールを検証に使用する .cub ファイルを指定できます (複数可)。 検証の詳しい情報については、「プロジェクトの検証」を参照してください。

テーブル 7 ・ MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
RunUpgradeValidation	ブール値	基本の MSI、InstallScript MSI	このパラメーターは、アップグレードの検証を実行するかしないかを指定します。
StopOnFirstError	ブール値	基本の MSI、InstallScript MSI、マージモジュール	このパラメーターは、一番最初のエラーが発生した時点で、ビルドを停止するかどうかを指定します。
MsiVersion	文字列	基本の MSI、InstallScript MSI、マージモジュール	このパラメーターは、ターゲット マシン上でインストールが許可する Windows Installer の最小バージョンを指定します。
DotNetFrameworkVersion	文字列	基本の MSI、InstallScript MSI	このパラメーターは、ターゲット マシン上でインストールが許可する .NET Framework の最小バージョンを指定します。
DotNetUtilPath	文字列	基本の MSI、InstallScript MSI	Regasm.exe と InstallUtilLib.dll は .NET Framework の各バージョンに含まれるユーティリティです。このパラメーターは、.NET インストーラー クラスおよび COM Interop を含むリリースのビルド時に使用する、これらのファイルの 32 ビット バージョンを含むディレクトリのパスを指定します。これは、.NET Framework 再配布可能ファイルへのパスではありません。 ***メモ:*** *InstallShield を 64 ビット システムで使用する場合、*ISCmdBld.exe *はこのパラメーターに指定されたパスに基づいて対応する .NET Framework の 64 ビット バージョンの場所を判別し、適切な場合、*ISCmdBld.exe *は* Regasm.exe *および* InstallUtilLib.dll *の 64 ビットの場所を使用します。*
Disk1Folder	出力文字列	基本の MSI、InstallScript、InstallScript MSI、マージモジュール	このパラメーターは、出力ファイルの場所を指定します。

テーブル 7 • MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
SummaryInfoComments	文字列	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	ビルド番号を含めるなど、ビルド時に **概要情報ストリーム** コメントを設定するには、このパラメーターを使用します。 このプロパティを使って追加されたコメントは、ビルド済みのインストーラーの **[プロパティ]** ダイアログ ボックスで参照することができます。
MSIPackageFileName	文字列	基本の MSI、InstallScript MSI	このパラメーターを使ってビルド時に MSI パッケージ ファイル名を指定します。.msi ファイルに使用するファイル名を指定します (ピリオドまたは拡張子は除きます)。
InstallShieldDigitalCertificatePassword	文字列	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	このパラメーターはオプションで、デジタル証明書パスワードを指定します。指定されなかった場合、ファイルの署名にはプロジェクトで構成されたパスワードが使用されます。

MSBuild スクリプト

MBuild は、1 つのファイル形式 (XML ビルド スクリプト) を理解します。XML ビルド スクリプトは、Visual C# および Visual Basic .NET プロジェクト (.csproj および .vbproj) のプロジェクト ファイル形式として使用されます。MSBuild はまた、ソリューション ファイルおよび Visual C++ プロジェクト ファイル フォーマット (.sln と .vcproj) を処理するための内部フックも備えています。

InstallShield の Visual Studio との統合では、MSBuild と互換性のある XML フォーマット プロジェクト ファイル (.isproj) が使用されています。これにより、MSBuild は、InstallShield プロジェクトを含む Visual Studio ソリューションをシームレスにビルドすることができます。スタンドアロン環境でソリューションをビルドするには、ビルド マシンに InstallShield Standalone Build をインストールします。

.isproj ファイルのカスタマイズ

InstallShield プロジェクトの変更を .isproj ファイルに反映させるには、.isproj ファイルの上部に PropertyGroup 要素または ItemGroup 要素 を追加するか、または既存の PropertyGroup または ItemGroup 要素を更新します。そのあと、InstallShield 関連のパラメーターを必要に応じて追加します。

.isproj ファイルからの、以下のサンプルコードは、次の処理を行う方法を説明します:

- 製品バージョンを設定する。

- 製品名を設定する。
- カスタム パブリック プロパティ MY_PROPERTY を *My Value* に設定する。
- **MyPathVariableName** と呼ばれるパス変数を *C:¥MyPath* の新しい値でオーバーライドします。
- InstallShield 前提条件に以下の検索パス、<ISProductFolder>¥SetupPrerequisites および <ISProjectFolder>¥MyCustomPrerequisites を指定する。

```
<PropertyGroup>
  <InstallShieldProductVersion>1.2.3</InstallShieldProductVersion>
</PropertyGroup>
<ItemGroup>
  <InstallShieldPropertyOverrides Include="新しい製品">
    <Property>ProductName</Property>
  </InstallShieldPropertyOverrides>
  <InstallShieldPropertyOverrides Include="値">
    <Property>MY_PROPERTY</Property>
  </InstallShieldPropertyOverrides>
  <InstallShieldPathVariableOverrides Include="C:¥MyPath">
    <PathVariable>MyPathVariableName</PathVariable>
  </InstallShieldPathVariableOverrides>
  <InstallShieldPrerequisitePath Include="&lt;ISProductFolder&gt;¥SetupPrerequisites" />
  <InstallShieldPrerequisitePath Include="&lt;ISProjectFolder&gt;¥MyCustomPrerequisites" />
</ItemGroup>
```

MSBuild を使用して、コマンドラインからリリースをビルドする

メモ *MSBuild を使って、InstallShield プロジェクトを含む Visual Studio ソリューションをビルドするには、.NET Framework 3.5 以降が必要です。*

MSBuild により、Visual Studio がインストールされていないマシン上で、簡単にコマンドラインからリリースをビルドすることができます。唯一マシンにインストールされている必要のあるコンポーネントは、.NET Framework と InstallShield Standalone Buildです。マシン上に Visual Studio ソリューションのコピーを作成して、MSBuild を実行します。

タスク ***コマンドラインから MSBuild を使用するには、以下の手順に従います。***

1. [コマンドライン プロンプト] ウィンドウを開きます。
2. MSBuild.exe を含むディレクトリへ変更します:

   C:¥Windows¥Microsoft.NET¥Framework¥*Version Folder*¥
3. Visual Studio 統合プロジェクトのリリース ビルドを行うコマンドライン ステートメントを入力します。例:

   MSBuild.exe C:¥*My Visual Studio Solution を含むフォルダー*¥*My Solution*.sln /property:Configuration=Release

リリースの設定を構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

リリースを作成すると、デフォルトが設定されます。[リリース] ビューで、すべてのリリース設定を編集することができます。リリース ウィザードでも、リリースの設定を編集することができます。



タスク ***リリース設定を編集するには、次の手順を実行します。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、設定を構成するリリースを選択します。設定が、右のペインのタブ上に表示されます。
3. タブの 1 つから設定を選択して、値を変更します。設定についての情報が、[ヘルプ] パネルに、または**[リリース]** ビュー内で F1 を押すと、表示されます。

配布する単一実行可能ファイルの作成



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

ビルドを単一の実行可能ファイルとしてパッケージする場合、リリース ウィザードの [一般オプション] パネルにある **[単一の実行可能ファイルを作成する]** オプション、または、**[リリース]** ビューの **"単一 .exeファイル名"** 設定を、次の手順にしたがって使用します:



メモ: *作成した単一実行可能ファイルでは、Setup.exe で使用できるすべてのコマンドライン パラメーターを使用できます。*



タスク ***リリース ウィザードを使用してビルドを単一実行可能ファイルにパッケージするには、以下の手順に従います。***

1. リリース ウィザードを起動します。
2. **[全般オプション]** パネルに移動します。
3. **[単一実行可能ファイルの作成]** チェック ボックスを選択します。

4. デフォルトで、ファイル名 **<プロジェクト名>.exe** が **[ファイル名]** ボックスに入ります。自己展開型ファイルに別の名前を付ける場合は、コンボ ボックスに新しいファイル名を入力するか、ファイル名を定義するパス変数を選択します。
5. [**Icon**] ボックスには、InstallShield がビルド時に Setup.exe ファイルを作成するときに使用するアイコンを含むファイルの完全修飾名を指定します。

   ファイルを指定するには、明示的パスまたはパス変数を入力するか、省略記号ボタンをクリックして **[アイコンの変更]** ダイアログ ボックスを開いて **[参照]** ボタンをクリックするとファイルを選択できます。

   デフォルトではインデックス 0 のアイコンが使われます。別のアイコンを指定するには、**[アイコンの変更]** ダイアログ ボックスでアイコンを選択するか、アイコンのインデックスまたはリソース ID (それぞれ前にマイナス (-) 記号がついている) をファイル名に付け足します。たとえば、**C:¥Temp¥MyLibrary.dll,2** は 2 というインデックスを持つアイコンを、**C:¥Temp¥MyLibrary.dll,-100** は 100 と言うリソース ID を持つアイコンを示します。
6. **[概要]** パネルで、**[リリースのビルド]** チェック ボックスを選択し、**[完了]** ボタンをクリックして、**リリース ウィザード**を完成します。



タスク ***[リリース] ビューの設定を使って、ビルドを単一実行可能ファイルとしてパッケージするには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、単一実行可能ファイルとしてパッケージするリリースを選択します。
3. **Setup.exe** タブを作成します。
4. **"単一 .exe ファイル名"** 設定で、ファイル名または値がファイル名を定義するパス変数 (山かっこで囲みます。例、**<MY EXE FILENAME>**) を入力します。
5. [**Setup.exe アイコン ファイル**] 設定には、InstallShield がビルド時に Setup.exe ファイルを作成するときに使用するアイコンを含むファイルの完全修飾名を指定します。

   ファイルを指定するには、明示的パスまたはパス変数を入力するか、省略記号ボタン (...) をクリックして **[アイコンの変更]** ダイアログ ボックスを開いて **[参照]** ボタンをクリックするとファイルを選択できます。

   デフォルトではインデックス 0 のアイコンが使われます。別のアイコンを指定するには、**[アイコンの変更]** ダイアログ ボックスでアイコンを選択するか、アイコンのインデックスまたはリソース ID (それぞれ前にマイナス (-) 記号がついている) をファイル名に付け足します。たとえば、**C:¥Temp¥MyLibrary.dll,2** は 2 というインデックスを持つアイコンを、**C:¥Temp¥MyLibrary.dll,-100** は 100 と言うリソース ID を持つアイコンを示します。
6. リリースをビルドします。

実行可能ファイルにデジタル署名を付加する方法についての詳細は、「デジタル署名とセキュリティ」を参照してください。自己展開型実行可能ファイルを起動するためにエンドユーザーにパスワードの入力を要求する手続きの詳細については、「インストールのパスワード保護」を参照してください。

ディスク イメージにファイルを非圧縮のまま残す



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*



タスク

機能の一部または全部を非圧縮のままディスク イメージに残すには、リリース ウィザードの [メディア レイアウト] パネルを使用します:

1. **リリース ウィザード**を起動し、**[メディア レイアウト]** パネルに移動します。
2. 以下のいずれかを選択します :
   - すべての機能ファイルを圧縮せずにディスクイメージに配置するには、**[CDROM フォルダー]** オプションを選択します。機能のすべてのファイルが機能の **CD-ROM フォルダー**のプロパティで指定したフォルダーのディスク イメージに配置されます。フォルダーが機能に指定されていない場合、機能のファイルはディスク イメージのルートに配置されます。
   - 一部の機能のファイルを圧縮せずにディスク イメージに配置するには、以下の手順を実行します。
     a. **[カスタム]** を選択して、**[次へ]** をクリックします。**[カスタム メディア レイアウト]** パネルが開きます。
     b. **[キャビネットの機能]** ボックスで、 機能の横にあるチェック ボックスを選択または選択解除します。機能のファイルがキャビネット ファイルに格納されている場合、チェック ボックスを選択解除します。機能のファイルが機能の "**CD-ROM フォルダー**" プロパティで指定したフォルダーのディスク イメージに格納されている場合は、チェック ボックスを選択します。フォルダーが指定されていない場合、機能のファイルはディスクイメージのルートに配置されます。



***ヒント・** 機能のほとんどをキャビネット ファイルに格納する場合は、**[すべてクリア]** をクリックしてから、ディスク イメージに非圧縮のまま残す機能を選択します。反対に、機能のほとんどを非圧縮でディスク イメージに残す場合は、**[すべて選択]** ボタンをクリックしてから、キャビネット ファイルに格納する機能を選択解除します。*

3. **リリース ウィザード**を完了します。



***メモ・** 機能のコンポーネントの "圧縮" 設定で [はい]、[いいえ] のどちらを選択しても、機能に非圧縮 (CD-ROM フォルダー) オプションを選択すると、機能のファイルは配布メディアで圧縮されません。*

Windows Vista 以降のプラットフォームでのセットアップランチャーの必要実行レベルを指定する



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

InstallShield では、Windows Vista 以降のプラットフォーム上でインストール (セットアップ ランチャー、InstallShield 前提条件、.msi ファイル、および任意のアドバンスト UI またはスイート/アドバンスト UI パッケージ) を実行するために、インストールの Setup.exe ファイルが必要とする最低実行レベルを指定することができます。プロジェクトの個別のリリースに対して設定することが可能です。

タスク ***リリースの必要実行レベルを指定するには、次の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、設定を行うリリースを選択します。
3. **Setup.exe** タブをクリックします。
4. **"必要実行レベル"** 設定に、適切なオプションを選択します。

選択可能なオプションは、以下のとおりです。

- **管理者** ー Setup.exe の実行には、管理者権限が必要です。管理者は、Setup.exe の実行を承認する必要があります。非管理者は、管理者としての認証が必要になります。
- **最高権限** ー Setup.exe の実行には、管理者権限が推奨されます。管理者は、Setup.exe の実行を同意(コンセント)する必要があります。非管理者は、管理者権限を持たずに Setup.exe を実行します。これは、InstallScript プロジェクトと InstallScript MSI プロジェクトのデフォルト オプションです。
- **起動者** ー Setup.exe の実行に、管理者権限は必要ありません。したがって、管理者権限を持たないユーザーも Setup.exe を実行することができます。Setup.exe は、資格情報または同意 (コンセント) を求める UAC メッセージを表示しません。これは、アドバンスト UI、基本の MSI、およびスイート/アドバンスト UI プロジェクトのデフォルト オプションです。

アドバンスト UI、InstallScript、InstallScript MSI、スイート/アドバンスト UI プロジェクト、および"セットアップ ランチャー" 設定が [はい] に設定されている基本の MSI プロジェクトでは、InstallShield は Windows アプリケーション マニフェストをリソースとして Setup.exe ランチャーに埋め込みます。このマニフェストは選択された実行レベルを指定します。Windows Vista よりも古いバージョンのオペレーティング システムでは、必要実行レベルは適用されません。実行レベルは、マニフェストで次のように定義されています:

```
<requestedExecutionLevel
  level="asInvoker"
  uiAccess="false"/>
```

レベル属性の他の有効な値は、**highestAvailable** と **requireAdministrator** です。

基本の MSI プロジェクトで、"セットアップ ランチャー" 設定が [いいえ] に設定されている場合、InstallShield は Windows アプリケーション マニフェストを Setup.exe ランチャーに埋め込みません。

基本の MSI プロジェクトで必要実行レベルを昇格することの利点は、Setup.exe を実行するための権限の昇格が、必要な場合 1 度で済むという点です。またこれらの権限はインストールに含まれるすべての InstallShield 前提条件および .msi パッケージの [実行] シーケンスにも適用できるため、承認を得るためのプロンプトを複数回にわたって行う必要がありません。たとえば、すべての InstallShield 前提条件のうち 2 つが管理者権限を必要とする場合、この設定を [管理者] に変更します。そうすることにより、インストール中、Windows Installer が Setup.exe ファイルを実行する前、プロンプトはエンドユーザーに対して一度のみ表示されます。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトにも似たような利点があり、Setup.exe を実行するための権限の昇格が、必要な場合 1 度で済みます。またこれらの権限はアドバンスト UI またはスイート/アドバンスト UI パッケージのすべてにも適用できるため、承認を得るためのプロンプトを複数回にわたって行う必要がありません。

ただし、権限を昇格して、インストールの終わりでもアプリケーションを起動する場合、この昇格された権限はアプリケーションに適用されますので注意してください。ほとんどの場合、Windows Vista 以降では、昇格された権限を使用したアプリケーションの実行は推奨されていません。



重要: *InstallShield は昇格された権限で実行されます。InstallShield 内からインストールを起動すると、昇格された権限がインストールに引き継がれるため、インストールは自動的に昇格された権限を持ちます。Windows Vista 以降を使用するエンド ユーザーに対して表示される動作を反映しない可能性があります。したがって、開発システムに Windows Vista 以降を使用する場合、InstallShield 内からではなく、リリース フォルダーを開いてインストールを直接起動することを考慮してください。*

リリース ファルダに素早くアクセスしてリリースを直接起動するには、標準ツールバーまたは [ツール] メニューの [リリースフォルダーを開く] をクリックします。

エンド ユーザーのインストール エクスペリエンスは、インストールが必要とする権限によってのみ実行された場合、安全性がより一層向上します。アプリケーションは、システム管理者のみによる実行が必須の場合を除き、最も低い権限で実行されることが理想的です。

InstallShield 前提条件が昇格された権限で実行されるときに、製品をアドバタイズするかどうか指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

構成方法によって、InstallShield 前提条件を含むインストールがインストール中のいくつかの時点で、Windows Vista 以降のシステム上で昇格された権限のユーザー アカウント制御 (UAC) プロンプトを表示することができます:

1. エンド ユーザーが Setup.exe ファイルを起動するとき
2. Setup.exe ファイルが、昇格された権限を必要とするセットアップ前提条件を起動するとき
3. Setup.exe ファイルが、昇格された権限を必要とする機能前提条件を起動するとき
4. Windows Installer が .msi パッケージの [実行] シーケンスを開始するとき

[リリース] ビューでインストールの "必要実行レベル" 設定で [管理者] を選択すると、通常 Windows Vista 以降は UAC プロンプトを一度だけ表示します。これは、エンド ユーザーが Setup.exe ファイルを起動したときに表示されます。ただし、この設定で [起動者] を選択した場合、Windows Vista 以降はインストール中に複数回にわたって UAC プロンプトを表示する可能性があります。たとえば、Windows Vista 以降は昇格された権限を必要とするセットアップ前提条件に UAC プロンプトを表示してから、.msi パッケージの [実行] シーケンスで別の UAC プロンプトを表示することがあります。作成中のインストールがこのシナリオに該当する場合、インストールをアドバタイズしてから .msi パッケージを実行し、エンドユーザーが .msi パッケージの UAC プロンプトを避けることができるようにした方がよい場合があります。作成中のインストールがこのシナリオに該当しない場合、.msi パッケージのアドバタイズを避けることが推奨されます。アドバタイズを行うと、UAC プロンプトが回避されなくなるためです。



ヒント: *[一般情報] ビューの "管理者権限が必要" 設定で、.msi パッケージが管理者権限を必要とするかどうかを指定します。InstallShield 前提条件エディターの [動作] タブでは、前提条件が管理者権限を必要とするかどうかを指定します。Windows Vista における UAC プロンプトの表示に関連する他の InstallShield 設定については、「インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する」を参照してください。*



タスク ***.msi パッケージをアドバタイズするかどうかを指定するには、以下の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、設定を行うリリースを選択します。
3. **Setup.exe** タブをクリックします。
4. **"前提条件が昇格必要時のアドバタイズ"** 設定で、適切なオプションを選択します。

選択可能なオプションは、以下のとおりです。

- **アドバタイズ: サイレント** – インストールのセットアップ前提条件が、昇格された権限で正常にインストールされたとき、.msi パッケージがアドバタイズされ、サイレント (ユーザー インターフェイスがない状態) で実行されます。これにより、資格情報または同意を求める追加の UAC プロンプトが、メイン インストールの [実行] シーケンスで不要となります。
- **アドバタイズ: 完全 UI** – インストールのセットアップ前提条件が、昇格された権限で正常にインストールされた場合、.msi パッケージがアドバタイズされ、完全ユーザー インターフェイスと共に実行されます。これにより、メイン インストールの [実行] シーケンスで追加の UAC プロンプトが不要となります。
- **アドバタイズなし**–.msi パッケージはアドバタイズされません。エンド ユーザーがインストールを実行したとき、1 つまたは複数の UAC プロンプトを表示して、セットアップ前提条件をインストールできます。.msi パッケージの [実行] シーケンスにも昇格が必要な場合は、Windows Installer が [実行] シーケンスを開始する前に追加 UAC プロンプトを表示できます。



重要: *これらのアドバタイズ オプションが動作するためには、パッケージでアドバタイズがサポートされている必要があります。アドバタイズは瞬間的ではないので、インストールに多少の遅延が生じます。また、アドバタイズの後、インストールの主要部分が完了する前にエンドユーザーが [キャンセル] をクリックすると、予期しない動作が発生することがあります。たとえば、製品のアドバタイズ ショートカットが、メインのインストールが始まる前にデスクトップに表示されてしまうことがあります。また、この時、混乱したユーザーがメインのインストールをキャンセルすると、パッケージがアドバタイズされたまま、完全にインストールされないということも場合によって起きます。したがって、状況によっては、この設定を [アドバタイズなし] のままにすることで、追加の UAC プロンプトを許可し、製品のアドバタイズを避ける方が安全な場合もあります。*

一般的に目標とされているのは、インストールが UAC プロンプトを一回のみ表示することです。"前提条件が昇格必要時のアドバタイズ" 設定のアドバタイズ オプションは、この目的を容易に実現できるように設けられていますが、すべての場合においてこの目的の実現を保証するものではありません。たとえば、インストールにより再起動が実行された場合、インストールのプロセスは、再起動後、常に制限された権限に戻ります。昇格がセットアップ前提条件、機能前提条件、または .msi パッケージのいずれに必要であるかにかかわらず、その後の権限の昇格により、追加の UAC プロンプトが表示されることもあります。再起動が最後の前提条件と .msi ファイルの間に来た場合、.msi パッケージはアドバタイズされません。次は、この設定でいずれかのアドバタイズ オプションを選択した際に発生する可能性がある異なる結果の例です。

- 前提条件は昇格を必要とし、ターゲット マシンで正常にインストールされる。製品はアドバタイズされ、インストールされる。製品インストールは、UAC プロンプトを表示しない。
- インストールの前提条件の 1 つが再起動を必要とする可能性がある。前提条件の再起動が実行されたあと、追加の前提条件はインストールされず、インストールされた他の前提条件も昇格を必要としない。このシナリオでインストールされた最後の前提条件は、昇格された前提条件の単純な成功例とは異なるため、.msi パッケージのアドバタイズは実行されない。
- インストールの前提条件は、すべてターゲット マシンに存在するためインストールをする必要がない。したがって、.msi パッケージのアドバタイズは実行されない。
- インストールの前提条件は、すべて昇格を必要としない。したがって、.msi パッケージのアドバタイズは実行されない。
- 昇格された前提条件は正常にインストールされる。ただし、この場合、パッケージは既にインストールされている製品のマイナー アップグレード。つまり、パッケージの製品コードが、既にターゲット マシンに存在する製品の製品コードと一致している。.msi パッケージのアドバタイズは実行されない。UAC プロンプトは、適切なデジタル証明書が以前のインストールおよびパッチに含まれているかどうかによって、表示される可能性もあるし、表示されない可能性もある。

インストールのパスワード保護



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

セキュリティをさらに向上させるため、インストール パッケージをパスワードで保護することができます。インストールをパスワードで保護すると、パッケージをインストールするエンドユーザーは、インストールを開くために大文字小文字の区別があるパスワードを入力する必要があります。

リリース ウィザードの [パスワードと著作権] パネルでパスワード保護をアクティベートできます。

リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield では、インストールに含まれている InstallShield 前提条件のランタイムの場所を指定できます。

タスク ***リリースの InstallShield 前提条件の配置場所を指定するには、以下の手順に従います。***

- **リリース ウィザード**の**InstallShield [前提条件]** パネル を使用します。
- **[リリース]** ビューでリリースを選択します。次いで、**[Setup.exe]** タブの **"InstallShield 前提条件の場所"** 設定で、適切なオプションを選択します。

タスク ***各 InstallShield 前提条件に対して異なる場所を指定するには、次の手順に従います:***

1. **[再配布可能ファイル]** ビュー (基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合)、または **[前提条件]** ビュー (InstallScript プロジェクトの場合) で、各 InstallShield 前提条件の適切な場所を指定します。詳細については、「特定の InstallShield 前提条件の実行時の場所を指定する」を参照してください。
2. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
3. ビルドするリリースを選択します。
4. **Setup.exe** タブをクリックします。
5. **InstallShield"前提条件の場所"** 設定で、**[個々の選択に従う]** を選択します。

ヒント: *または、リリース ウィザードの [InstallShield 前提条件] パネルで、[個々の選択に従う] オプションを選択することもできます。*

InstallShield 前提条件が別の前提条件の依存ファイルとしてプロジェクトに追加される場合、前提条件依存ファイルの場所は、それを必要とする前提条件の場所設定に従います。

InstallShield 前提条件を含むリリースのビルドで次の両方が当てはまるとき、1 つまたは複数のビルド エラーが発生する可能性があります。

- 前提条件が、Setup.exe から抽出されるとき、または (エンド ユーザーのコンピューターに Web からダウンロードされる代わりに) ソース メディアからコピーされるときの InstallShield 前提条件の場所を指定してください。
- 前提条件ファイルはコンピューターにはありません。

ビルドエラーを除去するには、プロジェクトから InstallShield 前提条件を削除するか、インターネットからコンピューターに InstallShield 前提条件をダウンロードするか、またはリリースの InstallShield 前提条件の場所をダウンロードオプションに変更してから、リリースを再ビルドします。

リリース レベルで、アドバンスト UI またはスイート/アドバンスト UI パッケージのランタイムの場所を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれているパッケージのランタイムの場所を指定できます。

ヒント: *アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、アップデート セットアップ ランチャーの構成を行う場合、パッケージのランタイムの場所は、セットアップ ランチャーから抽出するか、Web からダウンロードします。アップデート セットアップ ランチャーは、ソース メディアに格納されているパッケージには依存できません。*

詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート」を参照してください。

タスク ***リリースのパッケージの配置場所を指定するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. ビルドするリリースを選択します。
3. **Setup.exe** タブをクリックします。
4. **"パッケージの場所"** 設定で、適切なオプションを選択します。

タスク ***各パッケージに対して異なる場所を指定するには、次の手順に従います:***

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. それぞれの InstallShield 前提条件に対して、適切な場所を指定します。詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに対してランタイムの場所を指定する」を参照してください。
3. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
4. ビルドするリリースを選択します。
5. **Setup.exe** タブをクリックします。
6. **"パッケージの場所"** 設定で、**[個々の選択に従う]** を選択します。

パッケージが別のパッケージの依存関係としてプロジェクトに追加される場合、パッケージの依存関係ファイルの場所は、それを必要とするパッケージの場所設定に従います。

パッケージを含むリリースのビルドで次の両方が当てはまるとき、1 つまたは複数のビルド エラーが発生する可能性があります。

- パッケージの場所で、パッケージが Setup.exe から抽出されるか、または (エンド ユーザーのコンピューターに Web からダウンロードされる代わりに) ソース メディアからコピーされるかを指定します。

- パッケージ ファイルは、使用中のコンピューター上にはありません。

ビルドエラーをなくすには、プロジェクトからパッケージを削除してコンピューターに追加するか、またはリリースのパッケージの場所をダウンロード オプションに変更してから、リリースを再ビルドします。

スイート でパッケージのアンインストール順を指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[リリース] ビューの **[Setup.exe]** タブにある **"アンインストール順"** プロパティを使って、スイート プロジェクトに含まれるパッケージのアンインストール順を指定することができます。



タスク ***スイート プロジェクトでパッケージのアンインストール順を指定するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. ビルドするリリースを選択します。
3. **Setup.exe** タブをクリックします。
4. **"アンインストール順"** 設定で次のオプションから 1 つを選択します:
   - **パッケージ順と同じ** – (プロジェクトで定義されている) パッケージのインストール順と同じ順番でパッケージをアンインストールします。
   - **パッケージ順と逆** – (プロジェクトで定義されている) パッケージのインストール順と逆の順番でパッケージをアンインストールします。

オートメーション インターフェイスにおける UninstallOrder メソッド

オートメーション インターフェイスで UninstallOrder メソッドを使って **"アンインストール順"** プロパティを設定することもできます。次の値のうち 1 つを指定します。

- euoForward(0) – (プロジェクトで定義されている) パッケージのインストール順と同じ順番でパッケージをアンインストールします。
- euoReverse(1) – (プロジェクトで定義されている) パッケージのインストール順と逆の順番でパッケージをアンインストールします。

デジタル署名とセキュリティ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

インストールとアプリケーション ファイルにデジタル署名をすることで、インストールやアプリケーション内のコードが、発行時以来、改ざんまたは変更されていないことをエンドユーザーに保証することができます。

[署名] タブを使って、InstallShield がファイルに署名するときに使用するデジタル署名に関する情報 (証明機関より付与されたデジタル証明書ファイルを含む) を指定します。

[署名] タブでまた、ビルド時にデジタル署名をするインストール内のファイルを指定することもできます。InstallShield は、作業中のプロジェクトの種類に応じて、リリースに含まれる次の任意およびすべてのファイルに署名することができます。

- 基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの Windows Installer パッケージ (.msi ファイル)
- マージ モジュール プロジェクトのマージ モジュール パッケージ (.msm ファイル)
- アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI プロジェクトの Setup.exe ファイル
- InstallScript プロジェクトのメディア ヘッダー ファイル
- リリースの任意のファイル (アプリケーション ファイルを含む)



Windows ロゴ: *インストールのすべての実行可能ファイル (.exe、.dll、.ocx、.sys、.cpl、.drv、および .scr ファイル) は、Windows ロゴ プログラムに準拠するためにデジタル署名が必要です。*

[署名] タブにある設定に関する詳しい情報は、「リリースの [署名] タブ」を参照してください。

証明機関

証明機関とは、デジタル証明書 (デジタル ID とも呼ばれます) を発行、管理する VeriSign のような組織です。証明機関は、指定された基準に基づいて要求側の ID を検証し、デジタル証明書を発行します。デジタル証明書を取得するには、証明機関に会社と製品に関する特定の情報を提供する必要があります。

証明機関の一覧については、MSDN Web サイトの「Microsoft Root Certificate Program Members」を参照してください。

SHA-1 と SHA-256 証明書の違い

InstallShield では、インストールおよびファイルをビルド時に署名する際、HA-256 または SHA-1 ハッシュ アルゴリズムを使ったデジタル証明書を使用できます。

SHA-1 はセキュリティの脆弱性があるため、SHA-256 の使用が推奨されます。Microsoft は、Windows では 2016 年 1 月以降に SHA-1 証明書を使って署名およびタイムスタンプが追加されているアイテムを信頼しないことを発表しました。さらに、証明書を発行する組織である証明機関では、SHA-1 証明書が段階的に廃止されます。したがって、InstallShield プロジェクトに含まれる任意の SHA-1 証明書は、SHA-256 証明書と差し替えることが推奨されます。最新情報および特定の詳細については、証明機関にお問い合わせください。

プロジェクトで、SHA-256 証明書を使った署名が構成されている場合、InstallShield はビルド時に署名を行うファイルの署名に SHA-256 ハッシュを使用します。プロジェクトで SHA-1 証明書を使った署名が構成されている場合、InstallShield は SHA-1 ハッシュを使用します。SHA-1 証明書を使用すると、SHA-1 の使用についてアラートするビルド警告 -7346 が発生します。

InstallShield 2016 から、Windows Installer および InstallScript プロジェクトの SHA-256 デジタル証明書サポートが次のように強化されています:

- ダイジェスト タイプを指定できる機能
- Settings.xml で SHA-256 タイムスタンプ サーバーを指定することができます。
- 証明書ストアにある類似した名前の証明書も処理します



重要: *2016 年 1 月以降に作成またはタイムスタンプが付けられたすべての新しい署名は、SHA-256 に基づく必要があります。SHA-1 証明書を使って署名されているすべてのファイルを継続してサポートするためには、2016 年 1 月以前の日時を使ったタイムスタンプを含める必要があります。これらのファイルは、すべての現在のバージョンの Windows ですべての SHA-1 サポートが停止される 2020 年 1 月 14 日まで、MOTW (Mark of the web) システムを使って引き続き使用することができます。*

証明書ファイルまたは証明書ストアにある証明書を使ってデジタル署名を生成する

ファイルおよびインストールの署名に使用するデジタル署名情報を指定するとき、InstallShield では次のオプションから選択できます:

- 使用中のマシンにある .pfx 証明書ファイルを指定できます。
- 証明書を含む証明書ストアを参照できます。

オプション 1 ー.pfx ファイル

.pfx (Personal Information Exchange) ファイルを使って、インストールおよびアプリケーションにデジタル署名を行います。次のツールを利用して、.pvk ファイルと .spc ファイルから .pfx ファイルを作成することができます:

- PVK2PFX.exe—Windows Platform SDK の一部で、Microsoft Visual Studio 2005 にも含まれています。
- pvkimprt.exe—この PVK Digital Certificate Files Importer ツールは、Microsoft Web サイト (http://www.microsoft.com/downloads/details.aspx?FamilyID=F9992C94-B129-46BC-B240-414BDFF679A7&displaylang=EN) からダウンロードすることができます。

.pfx ファイルは通常、パスワードに関連付けられています。

オプション 2 ー 証明書ストアにある証明書

証明書ストアにデジタル証明書を格納する場合、使用する証明書を含む証明書ストアをプロジェクトで参照することができます。この方法で署名を行う場合、ストア名 (Personal、Trusted Root Certification Authorities、Enterprise Trust、Intermediate Certification Authorities)、ストアの場所 (ユーザー、マシン)、および特定の証明書を識別するためのサブジェクトといった情報を指定する必要があります。

ストアにパスワード付きでインポートされた証明書をプロジェクトで使用するように構成すると、ビルド時、InstallShield がプロジェクトのファイルに署名を行うときに、Windows がパスワードをプロンプトします。Windows が使用する強力なキー保護のため、InstallShield がパスワードを暗号化サービス プロバイダーに提供することはできません。

デジタル署名にタイムスタンプを追加する

リリースのデジタル署名情報を指定すると、InstallShield はデフォルトで、ビルド時にデジタル署名にタイムスタンプを追加します。デジタル証明書のタイムスタンプは、ファイルが署名された日時を記録します。これによって、署名が行われた時点で証明書の期限が切れていないことを証明することができます。信頼されたタイムスタンプ サーバーからのタイムスタンプは、一般的に証明書の有効期限よりも長く、デジタル署名を有効な状態に保ちます。

InstallShield が使用するデフォルトのタイムスタンプ サーバーを変更する方法、またはタイムスタンプ機能を無効にする方法については、「デジタル署名のタイムスタンプ サーバーを変更する」を参照してください。

ビルド時にリリースとそのファイルにデジタル署名を行う



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

では、リリースに構成できるデジタル署名の設定があります。ビルド時に、InstallShield は構成した設定を使用して、インストール パッケージ、Setup.exe ファイル、および、リリースにある定義した基準を満たす他のすべてのファイルに署名します。



タスク ***リリースとそのファイルのデジタル署名を構成するには、以下の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、署名するリリースをクリックします。
3. **[署名]** タブをクリックします。
4. 次の設定を適切に構成します。
   - **署名の種類** – ビルドが生成したファイルにデジタル署名を行う方法を指定します。**[カスタム]** オプションを選択すると、**"パス"** および **"引数"** 設定が有効になります。
   - **パス** – 使用する署名ツールの場所を指定します。署名ツールの場所を指定するには、この設定で省略記号ボタン (...) をクリックします。
   - **引数** – 署名ツール構成のコマンドライン引数を指定します。

- **証明書 URL** ー 完全修飾 URL を入力します (例、**http://www.mydomain.com**)。この URL は、エンドユーザーが製品や組織、会社に関しての情報を得るためのリンク先として、デジタル署名の中で使用されます。
- **デジタル証明書情報** ー この設定の省略記号ボタン (...) をクリックします。[証明書の選択] ダイアログボックスが開いて、.pfx ファイル、証明書ファイル (EV がエクスポートした .cer ファイル) の場所を指定するか、証明書を含む証明書ストアについての情報を指定することができます。
- **証明書パスワード**—ストアにパスワード付きでインポートされた証明書をプロジェクトで使用するように構成すると、ビルド時、InstallShield がプロジェクトのファイルに署名を行うときに、Windows がパスワードをプロンプトします。Windows が使用する強力なキー保護のため、InstallShield がパスワードを暗号化サービス プロバイダーに提供することはできません。
- **出力ファイルに署名する** — 署名を行うファイルを指定します (Setup.exe、.msi パッケージ、その両方、またはどちらにも署名しない)。**パッケージ内のファイルに署名する**
- **64 ビット署名を使用** ー 64 ビット署名フレームワークを使用してパッケージに署名を行うかどうかを指定します。
- **署名の説明** ー "署名出力ファイル" 設定で指定されたファイルに使用する署名の説明を指定します。
- **パッケージのファイルに署名** ー **[サポートファイル]** ビューで構成された InstallShield サポートファイル (英語、言語非依存、および詳細ファイル) を含む、インストール プロセス中にのみ使用されるすべてのファイルにデジタル署名するかどうかを指定します。

  [はい] オプションを選択する場合、***"パッケージ内のファイルに署名する"*** 設定の下にある他の設定を使って、署名を行うファイルとファイル パターンおよび署名を行わない項目を指定します。

  ***メモ・** 署名から除外するように指定されたファイルとファイル パターンは、署名するように設定されたものよりも優先されます。たとえば、**"含める"** 設定および **"除外する"** 設定に ***.exe** を指定すると、InstallShield は .exe ファイルに署名を行いません。*

***ヒント・** [署名] タブにある設定に関する詳しい情報は、「リリースの [署名] タブ」を参照してください。*

ビルド時に、InstallShield は [署名] タブで指定されたファイルに署名を行います。リリースが、マージ モジュールを含むインストール用の場合、ファイルは、マージ モジュールがマージされる前に署名されます。

コマンドラインからビルドされたリリースにデジタル署名を行う

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

ビルド時に InstallShield でアプリケーションに署名を行う代わりに、iSign アプリケーション (iSign.exe) を使用して、InstallScript プロジェクトのリリースをビルドした後、コマンドラインからそのリリースにデジタル署名を行うことができます。

iSign.exe は、次のディレクトリに保存されています:

InstallShield Program Files フォルダー¥System

iSign.exe の構文および利用可能なコマンドライン パラメーターについては、「iSign.exe」を参照してください。

リリース フラグ



プロジェクト: *次のプロジェクト タイプは、リリース フラグのサポートを含みます:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

リリースをビルドする際、リリースのタイプによって機能、InstallShield 前提条件、および連鎖 .msi パッケージを含めることも除外することもできます。たとえば、製品のトライアル版を作成していて、ビルドにすべての機能を含めない場合、機能にフラグを付けて、これらの機能をリリースの 製品構成 の下でこれらを指定することができます。



タスク

リリース フラグに基づいて機能、InstallShield 前提条件、および連鎖 .msi パッケージをフィルターするには、以下の手順に従います。

1. リリース フラグを機能、InstallShield 前提条件、および連鎖 .msi パッケージに割り当てます。

   この方法についての詳細は、「リリース フラグを機能に割り当てる」、「リリース フラグを InstallShield 前提条件に割り当てる」、または「リリース フラグを連鎖 .msi パッケージに割り当てる」を参照してください。

2. 適切な場合にリリースに含めるフラグを指定します。

トライアル版と完全バージョンのシナリオを比較すると、リリース フラグを使用する理由がよく分かります。次のテーブルでは、4 つの異なる機能の例を示します。どの機能が製品で使用できるかは、エンドユーザーが入手するバージョンによって決まります。

この例で、完全バージョンにはアプリケーションの実行可能ファイル、ヘルプ ファイル、スペルチェッカー、およびアドオン パックが連鎖 .msi パッケージとして含まれています。しかし、トライアル版にはアドオン パックは含まれませんが、アップグレード パックが含まれます。実質的に 1 つの製品であるものに対し、異なるインストール プロジェクトを 2 つ作成する必要はなく、特定のバージョンに固有の機能にフラグを付けることができます。たとえば、アドオン機能は **[完全]** としてフラグされ、アップグレード機能は **[トライアル]** としてフラグされます。

テーブル 8 ・ サンプル リリース フラグ

機能、InstallShield 前提条件、または連鎖 .msi パッケージ	バージョン	リリース フラグ
Windows Installer 4.5 再配布可能ファイル (InstallShield 前提条件)	すべてのバージョン	なし
Program_Executable (機能)	すべてのバージョン	なし
Help_Files (機能)	すべてのバージョン	なし

テーブル 8 • サンプル リリース フラグ (続き.)

機能、InstallShield 前提条件、または連鎖 .msi パッケージ	バージョン	リリース フラグ
Spellchecker (機能)	完全バージョン	完全
Add_Ons (連鎖 .msi パッケージ)	完全バージョン	完全
Upgrade_Pack (機能)	トライアル バージョン	Trial

リリースのビルドにリリース ウィザードを使用した場合、リリースにフラグの付いた機能、InstallShield 前提条件、および連鎖 .msi パッケージを含めるかどうかを選択できるオプションがあります。デフォルトでは、リリースにはすべての機能、InstallShield 前提条件、および連鎖 .msi パッケージが含まれます。リリース ウィザードの [フィルターの設定] パネル内でリリース フラグに **Full** を指定した場合、フラグがつかないすべての機能および InstallShield 前提条件と一緒に **Spellchecker** 機能と **Add_Ons** 連鎖 .msi パッケージのみがリリース リリースにビルドされます。アップグレード パックは、インストールに含まれません。

ヒント• *デフォルトでは、すべての機能、InstallShield 前提条件、および連鎖 .msi パッケージがリリースに含まれます。[リリース] ビューまたはリリース ウィザードのどちらかでリリース フラグを指定すると、フラグが付いていない機能、InstallShield 前提条件、および連鎖 .msi パッケージ、並びに指定されたフラグが付いた機能、InstallShield 前提条件、および連鎖 .msi パッケージがインストールに含まれます。[リリース] ビューまたはリリース ウィザードで指定されていないリリース フラグがついた機能、InstallShield 前提条件、または連鎖 .msi パッケージはリリースに含まれません。*

製品構成フラグとリリース フラグの違い

プロジェクト• *以下のプロジェクト タイプには、製品構成フラグとリリースフラグのサポートが含まれています:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

インストールに含めるフラグをリリース (*「リリース フラグ」*) と製品構成 (*「製品構成フラグ」*) の両方について指定できます。リリース フラグは、リリースの設定とリリース ウィザードの [フィルターの設定] パネルの両方で表示されますが、製品構成フラグは [リリース] ビューの製品構成の [全般] タブでのみ編集が可能です。

製品構成フラグは、リリース フラグを補足します。つまり、製品に対して指定するすべての機能フラグは、ビルド中の各リリースのフラグと組み合わせられ、指定されたフラグが付いたすべての機能、InstallShield 前提条件、および連鎖 .msi パッケージがリリースにビルドされます。製品構成レベルでは、どのフラグが含まれているかが確認できないので、このウィザードでフラグを指定するときは注意してください。

製品構成とリリースフラグとの関係を分かりやすく説明するため、以下の機能、InstallShield 前提条件、および連鎖 .msi パッケージを含むプロジェクトを想定します。

テーブル 9 ・ サンプル機能、InstallShield 前提条件、および連鎖 .msi パッケージ

機能、InstallShield 前提条件、または連鎖 .msi パッケージ	バージョン	リリース フラグ
Windows Installer 4.5 再配布可能ファイル (InstallShield 前提条件)	すべてのバージョン	なし
Program_Executable (機能)	すべてのバージョン	なし
Help_Files (機能)	すべてのバージョン	なし
Spellchecker (機能)	完全バージョン	完全
Add_Ons (連鎖 .msi パッケージ)	完全バージョン	完全
Upgrade_Pack (機能)	トライアル バージョン	Trial

次の表は、製品構成フラグとリリース フラグの組み合わせによって、どの機能、InstallShield 前提条件、および連鎖 .msi パッケージがインストールに含まれるかを説明します。

テーブル 10 ・ 製品構成フラグとリリース フラグを組み合わせる

製品構成フラグ	リリース フラグ	インストールに含まれる機能、InstallShield 前提条件、および連鎖 .msi パッケージ
<なし>	<なし>	Windows Installer 4.5 再配布可能ファイル、Program_Executable、Help_Files、Spellchecker、Add_Ons、Upgrade_Pack
<なし>	完全	Windows Installer 4.5 再配布可能ファイル、Program_Executable、Help_Files、Spellchecker、Add_Ons
Trial	<なし>	Windows Installer 4.5 再配布可能ファイル、Program_Executable、Help_Files、Upgrade_Pack
完全	Trial	Windows Installer 4.5 再配布可能ファイル、Program_Executable、Help_Files、Spellchecker、Add_Ons、Upgrade_Pack

リリース フラグに基づくフィルター



プロジェクト: *以下のプロジェクト タイプには、製品構成フラグとリリースフラグのサポートが含まれています:*

- *基本の MSI*

- *InstallScript MSI*
- *スイート/アドバンスト UI*

リリース フラグを機能および InstallShield 前提条件に割り当てると、割り当てたフラグに基づいてこれらを含めるリリースを作成することができます。デフォルトでは、すべての機能、InstallShield 前提条件、および連鎖 .msi パッケージがリリースに含まれます。[リリース] ビューまたはリリース ウィザードのどちらかでフラグを指定すると、フラグが付いていない機能、InstallShield 前提条件、および連鎖 .msi パッケージ、並びに指定されたフラグが付いた機能、InstallShield 前提条件、および連鎖 .msi パッケージがインストールに含まれます。フラグが付いていない機能、InstallShield 前提条件、および連鎖 .msi パッケージのみを含めるには、存在しないフラグを指定します。たとえば、**NoFlags** の様に指定します。こうすると、フラグが付いていない機能、InstallShield 前提条件、および連鎖 .msi パッケージだけがリリースに組み込まれます。

サブ機能を含めることはできますが、この親機能を含めることはできません。このような場合、サブ機能はリリースに最上位の機能として組み込まれ、この親はリリースからは除外されます。

リリース ウィザードまたは [リリース] ビューで、機能、InstallShield 前提条件、および連鎖 .msi パッケージをフィルターすることができます。この 2 つの方法を、以下に説明します。

リリース ウィザードでのフィルター

リリースの最も簡単な作成方法は、リリース ウィザードを使用することです。ウィザードの [フィルターの設定] パネルで、リリースに含めるフラグを指定できます。

[リリース] ビューでのフィルター

[リリース] ビューを使用して、含めるフラグを指定することもできます。フラグを製品構成レベルとリリースの両方で、フラグを含めることができます。詳細については、「製品構成フラグとリリース フラグの違い」を参照してください。



タスク ***製品構成フラグを指定するには、以下の手順を実行します。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、変更する製品構成をクリックします。設定が、右のペインにある **[全般]** タブで表示されます。
3. **"製品設定フラグ"** 設定に、含めるフラグを入力します。複数のフラグを含める場合は、それぞれカンマで区切ってください。



タスク ***リリース フラグを指定するには、以下の手順を実行します。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、変更するリリースをクリックします。設定が、右のペインのタブ上に表示されます。
3. "リリース フラグ" 設定に、このリリースに含めるフラグを入力します。複数のフラグを含める場合は、それぞれカンマで区切ってください。

1 つのプロジェクトから複数の製品構成のインストールを作成する



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

評価版リリースと完全リリースなど、2 つ以上の構成の製品をリリースしたい場合があります。単一プロジェクト内で、異なるプロジェクト機能のサブセットを持つリリースをビルドできます。プリプロセッサ命令を使用した単一のインストール スクリプトがあると、異なるランタイム動作をするリリースをビルドすることができます。



タスク ***ビルド時にリリースの含める機能を判別するには、以下の手順に従います。***

1. **リリース ウィザード**を起動し、**[機能]** パネルに移動します。
2. **[以下に含める機能を指定]** オプションを選択します。
3. **[機能]** エクスプローラーで、製品構成に含める機能のチェック ボックスを選択し、含めない機能のチェック ボックスを選択解除します。
4. **リリース ウィザード**の他のパネルを完成します。



タスク ***プリプロセッサ命令を使ってリリースのランタイム動作を決めるには、以下の手順に従います。***

1. スクリプトで、プリプロセッサ命令を使用して異なる製品構成に異なるコードを実行します。例 :

```
#ifdef EVAL_RELEASE
    /*評価固有コード */
#elif FULL_RELEASE
    /*完全リリース固有コード */
#endif
```

2. [**コンパイラ プリプロセッサ定義**] ボックス (**リリース ウィザード**の [**一般オプション**] パネル) または、"**コンパイラ プリプロセッサ定義**" 設定 ([**リリース**] ビューの [**ビルド**] タブ) は、リリースに実行させるコードに対応するプリプロセッサ定数を指定します。
3. リリースをビルドします。

標準 Web インストールと One-Click Install の違い



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

アプリケーションをインターネットからダウンロードする際、多くのエンドユーザーはダウンロード方法やアプリケーションを正しく実行する方法を説明した、長く複雑な説明書に直面します。標準的なダウンロードでは 10 以上の手順が必要となります。この作業は時間がかかる上に、必ずしも成功するとは限りません。アプリケーションのダウンロードに失敗したエンドユーザーは製品サポートのヘルプに頼らなければならないか、完全にあきらめて二度とアプリケーションにアクセスしなくなるでしょう。これは顧客にとっても、開発チームにとっても時間を無駄にする結果となります。

そこで、One-Click Install セットアップを使用すると、エンドユーザーはアプリケーションを必要最小限の操作でダウンロードすることができ、すぐにアプリケーションを開始することが可能です。以下に示す標準セットアップと One-Click Install セットアップのインストール手順を比べてみてください。

標準インターネット インストールの手順

標準的なインターネット インストールは以下のように行われます。

1. エンドユーザーが Web ページにアクセスし、アプリケーションをダウンロードするためのリンクをクリックします。
2. ダイアログが開きます。ソフトウェアの使用許諾契約に同意することを求めるダイアログが表示されます。
3. ここでエンド ユーザーは、インストール実行可能ファイルをディスクに保存することができます。エンド ユーザーは、インストールの実行ファイルに後でアクセスするため、この場所を覚えておく必要があります。
4. ファイル転送が開始されます。
5. エンドユーザーは、インストール実行可能ファイルを検索して、それを起動します。
6. インストール実行可能ファイルが展開、および実行されます。
7. アプリケーションを適切に実行するために他のリソースが必要な場合、エンドユーザーはこのときにダウンロードする必要があります。
8. インストールが実行され、ユーザーは案内に沿ってインストールを開始します。
9. エンドユーザーは、アプリケーション実行可能ファイルのディスクへの場所など、いくつか選択を行います。
10. アプリケーション ファイルの転送が開始されます。
11. エンドユーザーは、ディスク上のアプリケーション実行可能ファイルの位置を特定し、アプリケーションを起動します。

One-Click Install インストールの手順

One-Click Install セットアップを使用すると、エンドユーザーはインストールを素早く簡単に行うことができます。One-Click Install セットアップは以下のように行われます。

1. エンドユーザーが Web ページにアクセスし、[ダウンロード] ボタンをクリックします。
2. ダイアログが開きます。ダイアログにファイル転送の進行状況が表示されます。インストールがダウンロードされると、すぐにインストールが実行されます。
3. エンドユーザーは、アプリケーションの使用を開始します。

One-Click Install セットアップでは、エンドユーザーが実行可能ファイルを探す必要がなく Web ページからの移動は不要です。One-Click Install セットアップにより、エンドユーザーはダウンロードに苦労することなくアプリケーションの使用に専念することができます。

InstallScript プロジェクトの One-Click Install インストール



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

One-Click Install インストールには、適切なコマンドラインを使って、Setup.exe ファイルのダウンロードと起動を行うセットアップ プレーヤー (Setup.ocx) が含まれています。これにより、限定された権限で Windows Vista 以降を使用しているエンドユーザーもインストールを実行できるようになります。インストールのマニフェストで指定された必要実行レベルにより昇格された権限が必要な場合、適切なユーザー アカウント制御 (UAC) プロンプトが Setup.exe ファイルが起動されたときに表示されます。エンドユーザーのシステムに以前のバージョンの Windows がある場合、マニフェストは無視されますが、もう一方の動作は同じです。

One-Click Install インストールの Setup.exe ファイルに COM 情報は含まれていません。

他のインストールと同様、インターネット インストールは、アップデート、メンテナンス モード、複数インスタンス インストール、サイレント インストールのような機能を利用することができます。つまり、InstallShield インストール用に提供されている機能がすべて利用できるということです。

次の InstallScript 関数は、インターネットで有効です:

テーブル 11・インターネットで有効な InstallScript 関数

関数	説明
CopyFile	有効な URL によって指定されたファイルをコピーしたり、または CGI や ASP の要求を送信したりします。
ExistsDir	有効な URL によって指定されたフォルダーの存在を確認します。
GetFileInfo	有効な URL によって指定されたファイルに関する情報を取得します。
GetLine	有効な URL によって指定されたファイルから行を読み取ります。
Is	次を確認します。 ・ インストールがインターネットから実行されているかどうか ・ 有効な URL によって指定されたファイルやフォルダーの存在 ・ 指定された文字列が URL かどうか ・ URL の有効性
OpenFile	有効な URL によって指定されたファイルを開きます。
ParseUrl	指定した URL の一部を取得します。
ReadBytes	有効な URL によって指定されたファイルからバイトを読み取ります。
SeekBytes	有効な URL によって指定されたファイル内でファイルポインターを再配置します。

One-Click Install オブジェクト



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

One-Click Install のセットアップ プレーヤー内のオブジェクトは、Player または Ether です。Player オブジェクトは、オブジェクト モデルの中で最上位を占め、次に Ether と続きます。

Player オブジェクト



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

セットアップ プレーヤーを使用できるようにするには、インスタンスを作成する必要があります。詳細については、「Setup Player を使用する」を参照してください。

Playerには、メソッドとプロパティが 1 つずつあります。

- Open (URL)—データ .cab ファイルの場所を指定し、Ether オブジェクトのリファレンスを返します。引数は、Web サーバー、UNC パス、またはローカル ドライブのファイルの場所へのパスを示す URL から構成されます。
- LastError—メソッド (通常 Open()) によってスローされたエラー コードを示す整数値。

例

```
Player.Open("http://www.mydomain.com/myproduct");
Player.Open("file://¥¥server¥myproduct");
Player.Open("file://c:/my installations/myproduct/media/default");
```

LastError メソッド



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

これは、Player オブジェクトの LastErrorメソッドです。Player のメソッド (通常、Open()) によってスローされるエラーのエラーコードを示す整数値。

構文

```
var nError = player.LastError;
```

例

```
var ether = Player.Open("http://www.installshield.com")
```

パラメーター

なし。

戻り値

テーブル 12 ・ LastError メソッドの戻り値

リターン コード	説明
-2147186688	アクセス許可は付与されていません。[Java セキュリティ] ダイアログ ボックスの [拒否] をクリックすると、エラーがスローされます。
-2147186686	Setup Player ActiveX コントロールのアップデートかインストールができませんでした。APPLETタグの Codebase 属性で指定されたURLからダウンロードする機能を確認します。
-2147186685	ISPlayer のインスタンス作成か、ISPlayer の初期化ができませんでした。
-2147186684	player.Openを呼び出せませんでした。
-2147186683	Player.Open はアプレットの初期化のタイムアウトを呼び出しました。
-2147166892	要求したURLがありません。player.Open 呼び出し中に指定したURLで使用可能なインストール ファイルを確認してください。
-2147166895	保護されたリソースにアクセスできません。認証が必要で、エンドユーザーが正しい情報を入力していないインストールへのアクセスを行った場合、このエラーが返されます。
-2147012889	サーバー名が解決されませんでした。player.Open の呼び出しで使用される URL を確認してください。ドメイン名のスペル ミスが、このエラーの一番の原因となっていますので注意してください。

***メモ・** リストの最初の 6 つのエラーについては、エンドユーザーが Java コンソールで詳しいエラー情報を確認できます。*

Open メソッド

***プロジェクト・** この情報は、InstallScript プロジェクトに適用します。*

Player オブジェクトの Open メソッドに、インストールのキャビネット ファイルの URL を指定する文字列引数を使うことができます。

構文

var ether = Player.Open("<URL 文字列>")

例

```
var ether = Player.Open("http://www.installshield.com")
```

パラメーター

テーブル 13 • Open メソッドの有効なパラメーター

パラメーター	説明
<URL 文字列>	インストールのキャビネット ファイルの URL を指定します。

戻り値

Open メソッドは、Ether オブジェクトの参照を返します。

Ether オブジェクト

Ether オブジェクトは、次のメソッドをサポートします:

テーブル 14 • Ether オブジェクトがサポートするメソッド

メソッド	説明
GetLastError	インストールの結果を示す数値を返します。
IsPlaying	インストールが実行中かどうかを示すブール値を返します。
Play	インストールを開始します。
SetProperty	プロパティを一定の値に設定します。

GetLastError メソッド

Ether オブジェクトの Play メソッドを呼んだあと、GetLastError メソッドを呼び出してインストールの結果を取得します。GetLastError により、エラーが発生したかどうか、インストールが正常に完了したかどうか、またはキャンセルされたかどうかが分かります。

構文

```
ether.GetLastError( );
```

例

GetLastError は、次の例のようにポールすることをお勧めします。

```
var intervalID = 0;

function startInstall() {
    if (ether)
    {
```

```
        ether.Play();
        if (intervalID == 0)
            intervalID = window.setInterval("IsSetupFinished()", 3000);
    }
}

function IsSetupFinished()
{
    var nResult = ether.GetLastError();
    if (nResult != 0)
    {
        if (nResult == SETUP_FINISHED) // セットアップが完了しました
            /* to do */ ;
        if (nResult == SETUP_ERR_CANCELLED) // セットアップがキャンセルされました
            /* to do */ ;

        clearInterval(intervalID);
        intervalID = 0;
    }
}
```

パラメーター

なし。

戻り値

GetLastError の全戻り値の一覧は、リリースの Disk Images フォルダーにある Disk1¥_graphics¥Utilities.js にあり、次でも確認することができます。

- SETUP_RUNNING
- SETUP_FINISHED
- SETUP_ERR_GENERAL
- SETUP_ERR_LOADMEDIA
- SETUP_ERR_INSTALLKERNEL
- SETUP_ERR_STARTKERNEL
- SETUP_ERR_OPENCAB
- SETUP_ERR_INSTALLSUPPORT
- SETUP_ERR_SETTEXTSUB
- SETUP_ERR_INITINFO
- SETUP_ERR_GETSETUPDRIVER
- SETUP_ERR_INITPROPERTIES
- SETUP_ERR_RUNINSTALL
- SETUP_ERR_UNINSTALLSUPPORT
- SETUP_ERR_EXTRACTBOOT

- SETUP_ERR_DOWNLOADFILE
- SETUP_ERR_CANCELLED

IsPlaying メソッド

Ether オブジェクトの IsPlaying メソッドは、インストールが実行されているかどうかを示すブール値を返します。

構文

var <変数> = ether.IsPlaying();

例

var bPlaying = ether.IsPlaying();

パラメーター

なし。

戻り値

テーブル 15 ・ IsPlaying メソッドの戻り値

戻り値	説明
true	インストールが実行中であることを示します。
FALSE	インストールが実行されていないことを示します。

Play メソッド

Ether オブジェクトの Play メソッドは、インストールを起動します。

このメソッドの使用例を見るには、リリース ウィザードの [インターネット オプション] パネルまたはリリース プロパティ グリッドの [インターネット] ページで「デフォルトの Web ページの作成」オプションを使ってメディアをビルドしてから、生成された Web ページ (Setup.htm) でコードを調べます。

構文

ether.Play();

パラメーター

なし。

戻り値

なし。

SetProperty メソッド

Ether オブジェクトの SetProperty メソッドは、2 つの引数を受け取ります。最初の引数は文字列で、is::CmdLine プロパティの名前を表します。2 番目は Setup.exe コマンドライン パラメーターを指定する文字列です。

構文

ether.SetProperty("is::CmdLine","Setup.exe command-line options");

例

ether.SetProperty("is::CmdLine","-l0x0407");

パラメーター

次のテーブルには、SetProperty メソッドのパラメーターが含まれています。

テーブル 16 ・ SetProperty メソッドのパラメーター

プロパティ名	プロパティ値
is::CmdLine	Setup.exe コマンドライン パラメーターを指定する文字列。サポートされているパラメーターの一覧は、「Setup.exe および Update.exe コマンドライン パラメーター」をご覧ください。

戻り値

なし。

One-Click Install インストールのデジタル署名

エンドユーザーが One-Click Install インストールを実行すると、セットアップ プレーヤー (Setup.ocx) は常に次のファイルのデジタル署名を検証します:

- Setup.ocx
- Setup.exe
- Data1.hdr
- ISSetup.dll

エンドユーザーが One-Click Install インストールを実行すると、セットアップ プレーヤー (Setup.ocx) は常に次のファイルのデジタル署名を検証します: すべてのファイルが適切に署名されていて、証明書がまだ完全に信頼されていない場合、セットアップ プレーヤーはプロンプトを表示して、エンドユーザーの承認を求めます (プロンプトは、ダウンロードした実行可能ファイルを実行するときに表示されるプロンプトに似ています)。

InstallShield は、Setup.ocx ファイルの署名を行います。Setup.exe ファイル、Data1.hdr ファイル、および ISSetup.dll ファイルの署名に使用する証明書は同じである必要があります。Setup.exe ファイルとメディア ヘッダーを署名するオプションを選択すると、この要件は自動的に満たされます。

古いプロパティとメソッドの置換

InstallShield のインターネット インストール用簡易化モデルでは、InstallShield Professional 6.x で Ether オブジェクトとそのサブオブジェクトが持っていたメソッドとプロパティを消去します。以下は、そのメソッドとプロパティのリスト、および、Professional 6.x インストール Web ページを InstallShield インストールと共に動作させるため変換する際、それらのをどう置換するかの説明です。

AskDestPath

相当するダイアログをインストール中に表示するには、インストールの スクリプトで InstallScript 関数 **AskDestPath**を呼び出します。Web ページからインストールへのあて先パスを渡すには、"スクリプト定義" プロパティを設定します。

CommandLine

次のような行を、

ether.CommandLine = "<文字列値>";

次のように変更します。

ether.SetProperty("is::CmdLine","<文字列値>");Features, SetState

相当する機能をインストール中に提供するには、インストールのスクリプトで InstallScript 関数 **SdFeatureTree** を呼び出します。Web ページからインストールに機能の選択を渡すには、スクリプト定義のプロパティを設定して、インストール スクリプトでその値を使用して **FeatureSelectItem** を条件付きで呼び出します。

Features, GetState

インストール中に機能の選択状態を取得するには、インストールのスクリプトで InstallScript 関数 **FeatureIsItemSelected** を呼び出します。Web ページは、機能の選択状態を取得できません。

FileToOpen

インストール中にファイルを開くには、インストールのスクリプトで InstallScript 関数 **LaunchApplication** を呼び出します。Web ページからインストールへファイル名を渡すには、スクリプト定義のプロパティを設定します。

GetDownloadSize, FileLevelCosting

インストール中に必要なディスク空き容量を取得するには、インストールのスクリプトで InstallScript 関数 **FeatureGetTotalCost** を呼び出します。(**SdFeatureTree** のような機能のダイアログは、必要なディスク空き容量を表示します。)Web ページは必要なディスク空き容量を取得できません。

言語

次のような行を、

ether.Language = "<言語コード>";

次のように変更します。

ether.SetProperty("is::CmdLine","-l<言語コード>");LegacyMode

デフォルトで、インターネット インストールは *レガシー モード* で実行されます。つまり、インストールのユーザー インターフェイス イベントが適切に生成された状態でということです。Web ページで ether.LegacyMode=false; の設定効果を複製する方法が 2 つあります。

- 次の行を使って、サイレントでインストールを実行します。

  ether.SetProperty("is::CmdLine","-s");

- [スクリプト定義] プロパティを設定、または Is(WEB_BASED_SETUP) の戻り値を確認して、OnShowUI ハンドラーでインストールのユーザー インターフェイス イベント ハンドラー関数を条件付で呼び出します。

ProductName

Web ページは、この情報を取得できません。

GetTargetDir

Web ページは、この情報を取得できません。

GetTextSub

Web ページは、この情報を取得できません。

IsInstalled

Web ページは、この情報を取得できません。

SetSetupType

インストール中にセットアップの種類を指定するには、インストール スクリプトで InstallScript 関数 **FeatureSetupTypeSet**を呼び出します。インストール中にセットアップの種類選択ダイアログを表示するには、インストール スクリプトで、**SdSetupType** を呼び出します。Web ページからインストールにセットアップの種類を渡すには、スクリプト定義プロパティを設定します。

SetTargetDir

インストール中にターゲット ディレクトリを指定するには、インストール スクリプトでシステム変数 TARGETDIR に値を割り当てます。Web ページからインストールにターゲット ディレクトリを渡すには、スクリプト定義プロパティを設定します。

SetTextSub

インストール中にテキスト置換を指定するには、インストール スクリプトで "TextSub オブジェクトの値" プロパティに値を割り当てます。Web ページからインストールにテキスト置換を渡すには、[スクリプト定義] プロパティを設定します。

デフォルトの Web ページ (Setup.htm)



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

Setup.htm という名前にデフォルトの Web ページは、次のメソッドいずれかを使用して One-ClickInstall インストールをビルドした際に、ディスク イメージ フォルダー内に作成されます。

- リリース ウィザードの [インターネットのオプション] パネルで **[セットアップのデフォルト Web ページを作成]** オプションを選択。
- [リリース] ビューの [インターネット] タブにある "デフォルト Web ページの作成" 設定で [はい] を選択。

このページをそのまま使用する、自由にカスタマイズする、または Setup Player の使用例として参照することもできます。

上記の Web ページの作成に関する設定は、[リリース] ビューの "ClickOnce Install の生成" 設定が [いいえ] に設定されているとき、無効になりますので注意してください。

One-Click Install インストール



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

ホスト システム上で HTTP 1.1 をサポートする Web サーバーが必要です。FTP は One-Click Install インストールをサポートしませんので注意してください。

Setup Player の別の呼び出し方法



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

Web ページからセットアップ プレーヤーを呼び出す必要はありません。他のソースから Player を呼び出すことができます。たとえば、ログイン スクリプト、Visual Basic、Visual C/C++ アプリケーションなどです。ActiveXコントロールをサポートするプログラムツールは、Playerを呼び出すことができます。

Playerを呼び出すためのサンプルVisual Basic スクリプト

```
Dim player, ether
Set player = CreateObject("Setup.Player.Automation Interface Version")
Set ether = player.open("http://www.mydomain.com/mysetup")

If ether.IsPlaying() Then
    MsgBox "セットアップが実行中です。"
Else
    MsgBox "セットアップが実行していません。"
End If

Set player = Nothing
Set ether = Nothing
```

HTTPS と One-Click Install インストール



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

One-Click Install インストールは、HTTPS (secure hypertext transfer protocol) の下、つまり SSL (Secure Sockets Layer) を使った HTTP の下で実行することができます。HTTPS と共に実行するためにインストールが必要な唯一の変更は、インターネット対応関数に渡される URL を http:// から https:// へ変更することです。

One-Click Install インストールの作成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

リリース ウィザードを使用して、One Click Install インストールを作成することができます。



タスク ***リリース ウィザードを使用して、One Click Install を作成するには、以下の手順を実行します。***

1. **リリース ウィザード**を起動します。
2. **[インターネットのオプション]** パネルに移動します。
3. **[One-Click Install の生成]** オプションを選択します。
4. **[セットアップにデフォルトの Web ページを作成]** チェック ボックスを選択するか、**[起動する URL の入力]** ボックスでカスタム Web ページの場所を指定します。
5. 必要に応じて、他のオプションを指定します。
6. [リリース ウィザード] パネルの残りの部分を完了させます。
7. リリースをビルドします。



タスク ***基本の MSI または InstallScript MSI プロジェクトの One-Click Install インストールを作成するには、以下の手順を実行します。***

1. **リリース ウィザード**を起動します。
2. **[メディアの種類]** パネルに移動します。
3. **[メディアの種類]** 一覧で、**Web** をクリックします。
4. **[One Click Install]** パネルに移動し、**[One-Click Install の生成]** オプションを選択します。
5. .htm または .cab ファイルのファイル名を指定します。
6. One-Click Install がサポートするブラウザーを指定します。
7. [リリース ウィザード] パネルの残りの部分を完了させます。
8. リリースをビルドします。

Setup Player を使用する



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

セットアップ プレーヤーは、Setup.exe ファイルをダウンロードしてから、適切なコマンドラインを使って起動する Setup.ocx ファイルです。Setup.ocx は Setup.exe 以外の名前を持つセットアップ実行可能ファイルの起動もサポートします。これが正しく動作するためには、Setup.ini ファイルの [Startup] セクションに次のキーが含まれている必要があります:

LauncherName=*SetupLauncher*

SetupLauncher は、代替セットアップ ランチャーのファイル名です。

Web ページで Setup Player を使用する例として、リリース ウィザードの [インターネットのオプション] パネルにある [デフォルトの Web ページを作成] オプション、または [リリース] ビューの [インターネット] タブにある "デフォルトの Web ページを作成" 設定を使用してリリースをビルドします。それから、生成された Web ページ (Setup.htm) でコードを調べます。

Player のインスタンスを作成したら、Player 内で各オブジェクトに関連付けられたメソッドおよびプロパティを使用して、インストールをカスタマイズできます。

起動された ClickOnce Install インストールにデータを渡す

起動された One-Click Install インストールにデータを渡すには、is::CmdLine parameter と CMDLINE スクリプト変数を使用します。

インターネット インストールをサイレントで実行する

インターネット インストールをサイレントで実行するには、Ether オブジェクトの SetProperty メソッドを呼び出して Setup.exe コマンドライン オプションを以下の例のように渡します。

```
ether.SetProperty("is::CmdLine","-s -f1¥"C:¥¥My Folder¥¥Mydir.iss¥"");
```

```
ether.SetProperty("is::CmdLine","-s -f1¥"C:¥¥My Folder¥¥Mydir.iss¥" -f2¥"C:¥¥My Folder¥¥Mydir.log¥"");
```

次の事項に注意してください。

- サイレントのインターネット インストールを実行する際、-f1 オプションを指定する必要があります。インターネット インストールには、応答ファイル用のデフォルトの場所がありません。
- サイレントのインターネット インストールを実行または記録する際、-f1 または -f2 オプションを使用する場合、完全修飾絶対パスを指定しなければなりません。これらのオプションに、URL は使用できません。
- **SetProperty** への 2 番目の引数には、エスケープ シーケンス (¥¥ と ¥") を使用して、円記号と引用符を上記の例にしたがって指定する必要があります。

ランタイム言語の設定



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*



エディション: *複数言語サポートは、InstallShield の Premier Edition で使用することができます。*

次のメソッドで言語を指定すると、InstallShield で指定したデフォルトの言語、またはターゲット システムのデフォルトの言語にかかわらず、One-Click Install インストールはこの言語で実行されます。このメソッドを使用しない場合、One-Click Install インストールは、他のインストールと同じ方法でランタイム言語を判別します。



タスク ***ランタイム メッセージを追加するには、以下の手順を実行します。***

1. 言語のサポートをプロジェクトに追加します。
   a. **[プロジェクト]** メニューで、**[設定]** をクリックします。**[プロジェクトの設定]** ダイアログ ボックスが開きます。
   b. **[言語]** タブをクリックします。
   c. インストールがターゲットにする、言語のチェック ボックスを選択します。
2. リリースに言語を追加します。
   a. **[ビルド]** メニューで **[リリース ウィザード]** をクリックします。
   b. **[セットアップ言語]** パネルから、適切な言語を選択します。
3. Play メソッドを呼び出す前に、InstallShieldの Web ページで、SetProperty メソッドを呼び出して、適切な言語識別子を指定します。

   たとえば、英語の言語 ID は 0009 です: 英語のランタイム言語を設定するには、以下のコードを使用します:

   ```
   ether.SetProperty("is::CmdLine","-l0009");
   ```

デバッグ モードで One-Click Install インストールを実行する



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

デバッグモードで One-Click Install を実行するには、ether.Play を呼び出す前に以下を入力します。

```
ether.SetProperty( "is::CmdLine", "/d")
```

InstallScript からユーザーを認証する



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

One-Click Install 実行後に認証が必要な場合、InstallScript を使用して直ちにユーザーを認証できます。認証が必要なとき、Web サーバーからインストールに HTTP エラーコード 401 が送信されて、InternetError イベントが発生します。OnInternetError イベント ハンドラーでユーザーにユーザー名とパスワードを要求して、このイベントに対応します。

これを簡単に処理するには、WinInet.dll の **InternetErrorDlg** API を使用し、OnInternetError から IDRETRY 値を返します。これによって、インストールが HTTP 要求を再度発行してファイルを求めます。ユーザーが間違った情報を入力した場合、再入力を求めるプロンプトが表示されます。

Setup.ini



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

Setup.ini は、InstallScript ベース プロジェクトのビルド プロセス中にインストールの一部の要素を制御するために作成される初期化ファイルです。ビルド プロセスは、Setup.ini に一定のキー名と値のみを挿入します。ビルド プロセスが Setup.ini を作成すると、<プロジェクト フォルダー>¥Media¥<メディア名>¥Disk Images の Disk1 フォルダーに配置されます。

さらに Setup.ini をカスタマイズする場合は、テキスト エディターで変更します。希望の変更を行うバッチ ファイルまたは実行可能ファイルを起動するリリース ウィザードの [ポストビルドのオプション] パネル、または [リリース] ビューの "バッチ ファイルの実行" プロパティを使用して、このプロセスを自動化することができます。Setup.ini を、以前のメディアビルドからの変更コピーでただ単に上書きしないでください。上書きするとインストールが正しく動作しないことがあります。

Setup.ini には、次の 2 つのあらかじめ定義されたセクションがあります。

- [Startup]
- [Mif]

追加セクションを Setup.ini に追加して、情報をセットアップ スクリプトに渡すことができます。それから **GetProfString** と **GetProfInt** 関数を呼び出して Setup.ini からの情報をインストールに転送するとができます。

Setup.ini ファイルの例を示します。

```
[Startup]

EnableLangDlg=Y

Product=My Application

ProductGUID=23EAFFCA-361D-11D3-8B0F-00105A9846E9

CompanyName=自分の会社名

Skin=Setup.skin

SmallProgress=N

SplashTime=5

CheckMD5=N

CmdLine=/f1Test1.iss

LauncherName=MyInstall.exe


[Mif]

Type=SMS

Filename=Ishield
```

SerialNo=IS50-32XYZ-12345

Locale=DEU



メモ- (Disk1 を削除した後に) 後でファイルにアクセスする必要がある場合は、インストールの始めに Setup.ini *をサポート フォルダー (SUPPORTDIR) にコピーする必要があります。*

[Startup] セクション



***プロジェクト-** この情報は、InstallScript プロジェクトに適用します。*

Setup.ini の [Starup]セクションでは、次のキー名を使用できます。

- EnableLangDlg
- 製品
- ProductGUID
- CompanyName
- Skin
- SmallProgress
- SplashTime
- CheckMD5
- CmdLine
- LauncherName

EnableLangDlg

EnableLangDlg キー名は、インストール初期設定中に [言語] ダイアログ ボックスを表示するかどうかをインストールに指示します。[言語] ダイアログを使用して、エンドユーザーはインストールの実行時に使用する言語を使用可能な言語から選択することができます。この値は、**リリース ウィザード**の **[セットアップの言語]** パネル、または **[リリース]** ビューの **"言語ダイアログ"** プロパティで設定できます。

[言語] ダイアログについての詳細は、「インストールがユーザー インターフェイスで使用する言語を判別する方法」を参照してください。

製品

Product キー名は、[スタートアップ メッセージ] ダイアログ ボックスで、テキスト文字列の最初に表示されるアプリケーション名または製品名を識別します。

ProductGUID

ProductGUID キー名は、Setup.exe が初期化中にこのデータにアクセスできるように、インストールの GUID を指定します。メディア ビルダーは自動的に、Setup.ini に正しい値を入力します。値を変更しないでください。

CompanyName

CompanyName キー名は、会社の名前を指定します。

Skin

Skin キー名は、インストールがエンドユーザー ダイアログの表示に使用するスキン ファイルの名前を指定します。このキー名が Setup.ini ではない場合、スキンは使用されません。スキンは、リリース ウィザードの ユーザー インターフェイス パネル、または [リリース] ビューの "スキンの指定" プロパティで指定できます。

SmallProgress

SmallProgress キー名は、インストール初期化ダイアログを、InstallShield Professional バージョン 6.31 以前で作成したインストールによって表示されていた小さいボックスにするか、残りのエンドユーザー ダイアログ ボックスと同じ大きいボックスにするかを指定します。この値を Y に設定すると小さいダイアログ ボックスが表示され、N に設定すると大きいダイアログ ボックスが表示されます。(インストールが [セキュリティ]、[セットアップの保存および/または実行]、[セットアップ言語の選択] または [正規の製品が検出されました] ダイアログ ボックスを表示する場合、インストール初期化ダイアログ ボックスは、このキーの値に関係なく、残りのエンドユーザー ダイアログ ボックスと同じ大きさになります。)この値は、リリース ウィザードの ユーザー インターフェイス パネル、または [リリース] ビューの "小さい初期化ダイアログ" プロパティで設定できます。

SmallProgress が N に設定されていると、インストール初期化ダイアログ ボックス(および [セットアップ言語の選択] ダイアログ ボックスがあればこれも該当)は、スタートアップ グラフィックが閉じるまで表示されません。スタートアップ グラフィックの表示時間を指定するには、SplashTime 値を設定します。

SplashTime

SplashTime キー名は、スタートアップ グラフィックが表示される時間を秒単位で指定します。

CheckMD5

CheckMD5 キー名は、各ファイルの MD5 ハッシュ値をファイル展開中にインストール ヘッダーファイルに保存されている値と比較するかどうかをインストールに指示します。一般オプション – [詳細] ダイアログ ボックスの [MD5 シグネチャの検証] チェック ボックスでこの値を設定できます。



ヒント: *MD5 確認は破損したファイルを検出します。これは インターネットでのインストール中に便利です。MD5 確認を行わなかった場合、ファイル転送処理速度は速くなります。*

CmdLine

CmdLine キー名は、コマンドラインからパラメーターが渡されない時に Setup.exe に渡すコマンドライン パラメーターを識別します。



メモ: *Setup.exe を起動するときにコマンドライン プロンプトで /v パラメーターを使用する場合、CmdLine キー名に指定されたパラメーターは一切無視されます。*

使用可能なコマンドライン パラメーターの詳細については、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

LauncherName

Setup.exe の名前を変更すると、ファイルの新しい名前は LauncherName キー名によって指定されます。このキー名は、別のインストールが **DoInstall** 関数を使ってこのインストールを起動する場合に必要です。

[Mif] セクション



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

[Mif] セクションがある場合、インストールは自動的に、インストール .mif ファイルを Temp フォルダーに作成します。Setup.exe の –m コマンドライン オプションを使用し、オプションで –m1 と –m2 のオプションを使用することによって .mif ファイルを作成することもできます。

Setup.ini の [Mif] セクションの下には、キー名があります。

- Type
- Filename
- SerialNo
- ロケール

Type

このキーを SMS に設定します。

Filename

このファイル名キーはオプションです。作成される .mif ファイルの代替名を提供します。このキーが含まれていないと、インストールは Setup.ini の [Startup] セクションの下にある AppName キーを .mif ファイル名として使用しようとします。AppName キーも存在しない場合、インストールは Status.mif というデフォルトの名前を持つファイルを作成します。

.mif ファイルには .mif 拡張子が必要なので、ファイル名には拡張子は含めません。ファイル名にはパスも含まれません。パスはデフォルトで Temp フォルダーに置かれます。

SerialNo

SerialNo キーもオプションです。このキーの情報が提供されると、.mif ファイルの [シリアル番号] セクションに配置されます。このキーが存在しない場合、インストールは引用符 (“”) を代わりに置きます。

ロケール

.mif ファイルに指定したロケールを含めます。English (ENU) がデフォルトですが、ロケール文字列の全リストについては Microsoft のマニュアルを参照してください。

次は、自動的に .mif ファイルを作成するインストールの Setup.ini ファイルの例です。

```
[Startup]

AppName=InstallShield
```

[Mif]

Type=SMS

Filename=IShield

SerialNo=IS50-32XYZ-12345

Locale=DEU

リリースのクローンを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield では、[リリース] ビューでリリースをクローンまたはコピーすることができます。これにより、すべてのプロパティがクローンされたリリースにあったのときと同じ状態でリリースを作成することができます。従って、ほとんど相違点がないリリースを複数作成する場合、リリースをクローンして必要に応じて設定のみ変更するだけですみます。



タスク ***リリースをコピーまたはクローンするには、以下の手順を実行します。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. コピーするリリースを右クリックして **[クローン]** を選択します。

*「リリース x のコピー」*という名前のリリースが作成されます。リリースのコピーには、オリジナルのリリースと同じプロパティがすべてあります。リリースの名前を変更し、必要に応じリリースの設定を変更します。

Windows アプリ パッケージの作成



プロジェクト: *Windows アプリの作成は、基本の MSIのプロジェクト タイプで使用できます:*



重要: *デスクトップ拡張 (デスクトップ ブリッジ) を含む Windows アプリ パッケージ (.appx) のインストールおよびテストを行うには、Windows 10 Anniversary Update が必要です。さらに、Windows アプリ パッケージに署名を行う、Windows アプリ ロゴのバリエーションを活用する (「Windows アプリ ロゴの考慮事項」 を参照)、あるいはローカライズされた Windows アプリ パッケージをビルドするためには、InstallShield を Windows 10 マシン上にインストールするか、Windows 10 SDK と共にインストールする必要があります。*

Windows 8.x および 10 上にアプリを配布およびインストールする為に使用される Windows アプリ パッケージ (.appx/.msix) は、シンプルでセキュリティ保護されたパッケージ フォーマットで、UWP (ユニバーサル Windows プラットフォーム) で使用可能な唯一のフォーマットです。Windows アプリ パッケージの利点:

- 高い可用性、信頼性、および耐久性によって、アプリケーションが長期間にわたってエラーなしで継続的に動作し続けます。
- 必要最小限の構成とカスタマイズ不要な UI によるスタティック ビルドを使ったスムーズなインストール経験
- Windows ストアを使ってアプリケーションを販売または提供できるオプション
- UWP API を使用できる機能だけでなく、ライブ タイルなどの UWP 機能を活用
- Windows Nano Server 上で サポートを持つ唯一のパッケージ フォーマット

InstallShield は、Windows アプリ パッケージ フォーマットおよびそのデスクトップ/サーバー拡張の作成をサポートし、Windows アプリ フォーマットに適合しないアイテムを識別するためのテストを提供します。[リリース] ビューでリリースを選択するとき、**[Appx/MSIX]** という名前の新しいリリースごとに提供されるタブに、Windows アプリ パッケージを作成する設定があります。ここで、Windows アプリ パッケージのビルド プロセスに影響する様々な主要な設定を指定できます。

Windows アプリ パッケージは、**[Appx/MSIX]** タブの **[表示プロパティ]** 領域にある設定を使ってローカライズできます。Windowsアプリ パッケージのビルドに使用できるサポート対象言語は、リリースの [ビルド] タブにある **"UI 言語"** 設定の言語と一致します。ただし、ビルド マシンには MakePRI.exe（Windows 10 SDK 内）が必要です。MakePRI.exe がビルド マシン上で見つからない場合、InstallShield はデフォルト言語のみを使って Windows アプリ パッケージを生成します。

次の手順は、Windows アプリ パッケージの作成方法を説明します。



タスク ***基本の MSI プロジェクトを Windows アプリ パッケージ (.appx) フォーマットに変換するには、次の手順に従います:***

1. .msi リリースのビルドの一環として、Windowsアプリ パッケージをビルドする基本の MSI プロジェクトを開きます。
2. **[リリース]** エクスプローラーで、製品構成の下にある Windows アプリ パッケージのビルドを行うリリースの名前をクリックします。
3. **[Appx/MSIX]** タブをクリックして、**"WINDOWS アプリ パッケージのビルド"** 設定で **[はい]** を指定します。
4. **Appx、MSIX または両方**のパッケージ タイプを選択してください。
5. 次の手順に従ってください:
   - **[Appx/MSIX]** タブの **[全般]** 領域を使って、デスクトップ拡張またはサーバー拡張を含めるかどうかを選択するなど、ビルド処理に影響を及ぼす主要な設定を構成します。
   - **[Appx/MSIX]** タブの **[パッケージ ID のオーバーライド]** 領域を使って、Windows に対してパッケージを一意に識別します。
   - **[Appx/MSIX]** タブの **[表示プロパティ]** 領域を使ってエンドユーザーに対して、パッケージを識別します。

   これらの設定についての詳細は、「リリースの [Appx/MSIX] タブ」を参照してください。
6. Windows アプリ パッケージで作成されたタイルを構成するには、**[ショートカット]** ビューを開いて、**[WINDOWS アプリ パッケージ タイルのオーバーライド]** 領域で設定を構成します。**「Windows アプリ パッケージ タイルのオーバーライド"** 設定についての詳細は、「ショートカットの設定」を参照してください。
7. リリースをビルドします。

8. .msi パッケージ内で Windows アプリ パッケージ フォーマットに適さないアイテムの存在をスキャンするには、InstallShield MSIX/UWP アプリ適合性スイートを実行します。**[ビルド]** メニューから **[検証]** をポイントしてから、**[InstallShield MSIX/UWP アプリ適合性スイート]** をクリックします。詳細については、「InstallShield MSIX/UWP アプリ適合性スイート」を参照してください。

   Windows アプリ パッケージの配布方法についての詳細は、「Windows アプリ パッケージの配布」を参照してください。

インストールのテストと実行

インストールを配布する前にテストを行い、顧客に頼らずに問題をすべて見つけることはとても重要です。InstallShield では、(ファイルをターゲット システムにコピーする手間をかけずに) エンドユーザー ダイアログをテストしたり、転送するファイルを含むインストール全体を実行することができます。

- インストールのテスト実行
- InstallShield インターフェイスからインストールを実行する
- サイレント モードでインストールを実行する
- InstallScript MSI プロジェクトの MSI サイレント モード インストール
- コマンドラインを使ったサイレント アンインストール (InstallScript MSI のみ)
- スイート インストールをコマンドラインから最小ユーザー インターフェイスを使って実行する
- コマンドラインからアップデートのダウンロードを行わずにスイートのサイレント インストールを実行する

インストールのテスト実行



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

インストールをテスト実行してエンドユーザー ダイアログを確認することができます。テスト実行によって、完全ユーザー インターフェイスが表示されますが、ターゲット システムにファイルがコピーされたり、レジストリが更新されることはありません。



タスク ***プロジェクトをテスト実行するには、以下のいずれかを行ってください。***

- **[リリース]** ビューで、リリース名を右クリックしてから **[セットアップのテスト]** をクリックします。
- ツールバーにある**[テスト]** ボタンをクリックします。

InstallShield インターフェイスからインストールを実行する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

リリース ウィザードを実行してインストールのリリースをビルドすると、それを InstallShield 内から実行することができます。



タスク ***ビルドしたインストール パッケージを実行するには、以下のいずれかを実行します。***

- ツールバー上の **[実行]** ボタンをクリックします。
- **[リリース]** ビューで、リリース名を右クリックしてから **[セットアップの実行]** をクリックします。
- **[ビルド]** メニューで、**[ReleaseName の実行]** をクリックします。*ReleaseName* は、現在 **[リリース]** ビューでフォーカスされているリリースの名前です。

InstallShield は、[リリース] ビューで現在強調表示されているリリースを実行します。

[オプション] ダイアログ ボックスで [インストール前にアンインストールする] チェック ボックスを選択すると、これまでに実行されたすべてのリリースが現在のリリースが実行される前にアンインストールされます。このオプションは、[オプション] ダイアログ ボックスの [プリファレンス] タブから使用できます。

InstallShield インターフェイスからインストールを起動するとき、コマンドライン パラメーターを指定することはできません。Windows Installer にパラメーターを渡すには、コマンドラインからインストールを起動する必要があります。詳細については、「MsiExec.exe コマンドラインのパラメーター」を参照してください。

サイレント モードでインストールを実行する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

サイレント インストールはエンド ユーザー インターフェイスなしで実行されるインストールです。インストールをサイレントで実行する場合、InstallShield では、基本の MSI、InstallScript MSI および InstallScript プロジェクトにサイレント インストールを作成できます。

基本の MSI のサイレント インストール

基本の MSI インストールをサイレントで実行するには、コマンドラインで次のように入力します。

msiexec /i Product.msi /qn

リリース設定に Setup.exe が含まれている場合、次のコマンドを実行することができます:

Setup.exe /s /v"/qn"

基本の MSI インストールは、応答ファイルの作成や読み込みは行いません。基本の MSI プロジェクトのインストール プロパティを設定するには、コマンドラインから次のように入力します。

msiexec /i Product.msi /qn INSTALLDIR=D:¥ProductFolder USERNAME="お客様各位"

InstallScript MSI および InstallScript サイレント インストール

InstallScript MSI および InstallScript プロジェクトでは、エンドユーザーの対応を記録する応答ファイルに記録を作成する必要があります。この応答ファイルは、インストールが実行できるように、Setup.exe に渡されます。従来のサイレント インストールは、通常のインストールとほとんど同じように動作します。このインストールは、通常のインストールと同じスクリプト論理に従っています。

Setup.exe を使用せずに InstallScript MSI インストールをインストールする必要がある場合、MSI サイレント モードを使用することができます。

InstallScript MSI と InstallScript のサイレント インストール



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield Silent では、サイレント インストールとしても知られる自動化された電子的なソフトウェアの配布が可能です。InstallShield Silent を使用すると、エンド ユーザーがインストールを監視したり、ダイアログ ボックスに情報を入力する必要はありません。InstallShield Silent インストールは、エンドユーザーの操作なしで自動的に実行されます。

マシン上にアプリケーションの複数のインスタンスがあり、そのマシン上でアプリケーションのサイレント アップデート インストールを実行する場合、最初に見つかったインスタンスにアップデートが適用され、[正規の製品が検出されました] ダイアログ ボックスは表示されません。

InstallShield Silent を起動させるには、Setup.exe –s コマンドライン パラメーターを使用します。



***Windows ロゴ・** Windows 95 ロゴ要件に準拠するため、デフォルトのインストール オプションが選択される応答ファイルを、サイレントインストールで作成する必要があります。*

インストールを Setup.exe –r パラメーターで実行して、インストール オプションを選択し、InstallShield Silent 応答ファイルを自動的に記録するか、独自のファイルを作成することができます。応答ファイルの実例については、InstallShield インストールの Disk1 にある Setup.iss ファイルを参照してください。応答のファイル形式の説明については、「応答ファイルを作成する」を参照してください。

Setup.exe を使用せずに InstallScript MSI インストールをインストールする必要がある場合、MSI サイレント モードを使用することができます。



タスク ***サイレント インストールを作成するには、以下の手順を実行します。***

1. インストールを作成します。
2. 応答ファイルを作成します。
3. サイレント インストールを実行します。
4. エラーを確認します。

サイレント インストールの作成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

サイレント モードで実行されるインストールを作成するには、通常の方法でインストールを作成してから、非サイレント モードでスクリプトをテストします。

インストール スクリプトのロジックを変更し、インストールが サイレント モードが実行されているかどうかによって、簡単にフロー制御を含めることができます。InstallShield では、*MODE* というシステム変数が提供されています。*MODE* 変数には、インストールの現在のモードを識別する次の定数のうちの 1 つが含まれます。

- **NORMALMODE**—インストールが通常モードで実行されていることを示します。
- **RECORDMODE**—Setup.exe によって、Windows フォルダーにサイレント インストール ファイル (.iss ファイル) が自動的に生成されることを示します。.iss ファイルは、インストールの入力値を記録したものです。
- **SILENTMODE**—インストールがサイレント モードで実行されていることを示します。

詳細については、「MODE」を参照してください。



ヒント: *すべての InstallShield ビルトインおよび Sd ダイアログは、InstallShield サイレント応答ファイル (.iss ファイル) に格納された値を自動的に処理します。カスタム ダイアログを作成する場合、* **SilentReadData** *を呼び出し、ダイアログの戻り値をサイレント モードで処理する必要があります。*

インストールを作成または編集したら、応答ファイルを作成してサイレント インストールの作成を続けます。

応答ファイルを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

通常の (非サイレント モード) インストールは、ダイアログ ボックスへの応答という形でエンドユーザーから必要な入力を受け取ります。逆に、サイレント インストールは、ユーザーに入力をプロンプトしません。したがって、別のソースからエンドユーザー インプットを取得する必要があります。その方法が、InstallShield サイレント応答ファイル (.iss ファイル) です。

応答ファイルには、通常のインストール実行時にエンドユーザーがダイアログへの応答として入力するだろうと思われる情報が含まれています。InstallShield サイレントでは、応答ファイルから実行時に必要な情報を読み込みます。

応答ファイルの形式は .ini ファイル形式に類似していますが、拡張子には .iss が付きます。応答ファイルはデータ エントリを含むセクションから構成されたプレーン テキスト ファイルです。

InstallShield サイレント応答ファイルの作成するには、InstallShield を使って自動的に応答ファイルを記録/作成する方法と、直接応答ファイルを作成する方法の 2 種類があります。

応答ファイルの記録

InstallShield で自動的に応答ファイルを作成するオプションがあります。インストールを、Setup.exe –r コマンドライン パラメーターを使用して単純に実行します。InstallShield は、Setup.iss にインストールの設定オプションをすべて記録し、このファイルを Windows フォルダーに格納します。

すべての InstallShield ビルトインおよび Sd ダイアログ関数は、InstallShield が記録モードで実行されたとき (Setup –r)、Setup.iss ファイルに値を書き込むように設計されています。カスタム ダイアログを作成する場合は、**SdMakeName** と **SilentWriteData** を呼び出し、インストールが記録モードで実行されたときにセクションとダイアログ データを応答ファイルに追加します。<InstallShield の場所>Include フォルダー内にある Sd ダイアログのソース コードを見て、これらの関数を使用して Setup.iss に書き込む例を参照してください。**SdMakeName** および **SilentWriteData** を呼び出した際に Setup.iss に追加するデータの詳細については、次のセクションをお読みください。

応答ファイルを手動で作成する

応答ファイルは、手動で初めから作成することもできます。すでに示したように、Setup.iss ファイルは .ini ファイルと類似しています。InstallShield 応答ファイルのセクションは、次の順序で作成しなければなりません。

1. サイレント ヘッダー セクション
2. アプリケーション ヘッダー セクション
3. ダイアログ シーケンス セクション
4. ダイアログ データ セクション (ダイアログ 1 つにつき 1 セクション)

セクション名は **[InstallShield Silent]** というように、角かっこで囲みます。

データ エントリはセクション名の後に続き、次のように <名前=値> という形式のペアにします:

```
Dlg0={23EAFFCA-361D-11D3-8B0F-00105A9846E9}-Welcome-0
```



タスク ***応答ファイルを作成するには、以下の手順を実行します。***

1. テキスト エディターを使用し、Setup.iss という名前のテキストファイルを作成します。
2. Setup.iss にサイレント ヘッダーを入力します。
3. Setup.iss にアプリケーション ヘッダーを入力します。

4. Setup.iss にダイアログ シーケンスを入力します。
5. Setup.iss にダイアログ データを入力します。
6. Setup.iss を保存して閉じます。

この形式に慣れ親しめるよう、サンプル応答ファイルが付属されています。

応答ファイルのサイレント ヘッダー



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

すべての応答ファイルは、応答ファイル サイレント ヘッダーで始まります。応答ファイル サイレント ヘッダーを使って、InstallShield はそのファイルが有効な InstallShield 応答ファイルであることを識別することができます。また、応答ファイルが InstallShield の適切なバージョンを使って作成されたインストールに対応することを確認するのにも利用されます。

サイレントヘッダーのフォーマットは、次の通りです。Setup.iss ファイルの始まり部分に次の行を入力します。

```
[InstallShield Silent]
Version=v7.00
File=Response File
```

Version=v7.00 ラインは、InstallShieldSilent 応答ファイルのバージョンを示します。InstallShield のバージョンではありません。すべての応答ファイルに **v7.00** を使用します。

応答ファイルを作成するとき、サイレント インストールの実行に使用される応答ファイルを作成する InstallShield と同じバージョンを使用してください。

応答ファイルのアプリケーション ヘッダー



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

応答ファイル アプリケーション ヘッダーは、応答ファイル サイレント ヘッダーのすぐ後に続く応答ファイルの中の 2 番目の情報ブロックです。応答ファイル アプリケーション ヘッダーを使用して、応答ファイルを視覚的に見分けることができます。ヘッダーは、インストール スクリプトや Setup.exe によって使用されます。アプリケーション ヘッダーを使用して、応答ファイルがどのインストールのものかを正確に識別することができます。これはダイアログ データを見ただけでは、概して判断が付きにくいものです。

アプリケーションヘッダーのフォーマットは、次の通りです。名前、バージョン、および会社に割り当てられた値は、インストール スクリプトの **CreateInstallationInfo** 関数への呼び出しでレジストリに書き込まれた値から取得されます。(イベントベースのスクリプトでは、**CreateInstallationInfo** がデフォルト OnMoveData イベント ハンドラー コードで呼び出されます。サイレント ヘッダーの下の Setup.iss ファイルに、次の行を入力します。

```
[Application]
Name=<CreateInstallationInfo からの ProductKey>
```

Version=<CreateInstallationInfo からの VersionKey>
Company=<CreateInstallationInfo からの CompanyKey>

応答ファイルのダイアログ シーケンス



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

サイレント ヘッダーとアプリケーション ヘッダーの後にある応答ファイルの情報の 3 番目のブロックは、応答ファイルのダイアログ シーケンスです。ダイアログ シーケンス セクションでは、通常のインストールで使用する必要があるダイアログ (カスタム ダイアログを含む) が、表示される順にすべて一覧表示されます。ダイアログ ボックスは、[<PRODUCT_GUID>-DlgOrder] というセクション ヘッダーの下に一覧表示されます。

ダイアログの順番付けシーケンスは 0 から始まります。リストに表示できるダイアログの数に制限はありません。

ダイアログの順序と数は、重要な意味があります。InstallShield Silent が実行されるとき、ダイアログの数または順序のいずれかが、非サイレント インストールのダイアログの順序または数に一致しない場合、サイレント インストールに失敗し、ログ ファイル にエラーが記録されます。ダイアログの発生ごとに、エントリを 1 つ作成します。特定のダイアログは、インストールで複数回表示される場合、複数回一覧表示されます。

以下は、ダイアログ シーケンス エントリの形式です。

Dlg<#>=<PRODUCT_GUID>-<DialogIdentifier>

Dlg<#> の部分は、Dlg という文字列と、シーケンス番号から構成されます。インストールの最初のダイアログは、Dlg0 です。この後の各ダイアログでは、<#> の値が 1 ずつ増加します。

DialogIdentifier の形式は、*functionname*-<#>です。 *functionname* は、ダイアログを作成した関数の名前で、<#> は、インストールにおけるその特定のダイアログのシークエンス番号を意味します。

カスタム ダイアログの場合、<DialogIdentifier> の *functionname* の部分に一意の英数名を使用できます。たとえば、カスタム ダイアログを MyDialog と名前を付けることができます。インストールの 10 番目のダイアログが MyDialog というカスタム ダイアログの 2 番目である場合、ダイアログ シーケンス セクションで、次のようなエントリができます。

Dlg9=<PRODUCT_GUID>-MyDialog-1

<DialogIdentifier> 式は、ダイアログのダイアログ データ セクションを識別するために使用されます。

ダイアログ シーケンス セクションは、常に次の形式のステートメントで終了してください。

Count=<ダイアログの数>

ステートメントは、ダイアログ シーケンス セクションにリストされているダイアログの正確な数を指定します。すべてのエントリをカウントします。ダイアログの順番付けは 0 で開始するため、Count の値は、ダイアログ シーケンスに対する <#> の最大値よりも常に 1 つ多くなります。

次の例は、Welcome と **AskOptions** の 2 つのダイアログです。下の例のように、ダイアログ シーケンスのリストを Setup.iss に入力します。

[{23EAFFCA-361D-11D3-8B0F-00105A9846E9}-DlgOrder]
Dlg0={23EAFFCA-361D-11D3-8B0F-00105A9846E9}-Welcome-0

```
Dlg1={23EAFFCA-361D-11D3-8B0F-00105A9846E9}-AskOptions-0
Count=2
```

応答ファイルのダイアログ データ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

応答ファイルの最後の情報ブロックは、応答ファイルのダイアログ データです。応答ファイルのダイアログ データは、[ダイアログ シーケンス] セクションで識別された各ダイアログから返された値を含む、セクションのコレクションです。各ダイアログには、独自のセクションがあります。リストされた値は、エンドユーザーの入力主導の標準のインストールでダイアログが戻す値と同じになります。また、カスタム ダイアログのダイアログデータセクションを作成することもできます。

以下は、ダイアログ データ セクションの形式です。

```
[<PRODUCT_GUID>-<DialogIdentifier>]
Result=value
Keyname1=value
Keyname2=value
```

[<PRODUCT_GUID>-<DialogIdentifier>] セクション ヘッダーによって、特定のダイアログ ボックスが識別され、ダイアログ データ エントリ リストが続きます。<DialogIdentifier> は、[ダイアログ シーケンス] セクションにダイアログを一覧表示するために使用される式と同じです。

データ エントリ項目は、keyname=value という形式になります。キー名は、ダイアログが戻す値の名前で、応答ファイルに記録されます。カスタム ダイアログを含むすべてのダイアログによって、キー名 Result の値が戻され、ダイアログを終了するためにクリックされたボタンを反映します。多数のダイアログによって、変数の値が設定されるか戻されます。

たとえば、非サイレント インストールでは、**AskDestPath** ダイアログによって、svDir パラメーターにインストール先が返されます。スクリプトの行は、次のようになります。

```
nResult = AskDestPath( szTitle, szMsg, svDir, 0 );
```

サイレント インストールの場合、Setup.iss ファイルの対応するダイアログ データ エントリは、次のようになります。

```
[{23EAFFCA-361D-11D3-8B0F-00105A9846E9}-AskDestPath-0]
Result=1
szPath=C:¥Program Files¥InstallShield¥2022
```

上の例では、**Result=1** は、ダイアログの [次へ] ボタンをクリックするのに等しく、**szPath=C:¥Program Files¥InstallShield¥2022** は、**AskDestPath** の svDir パラメーターに返される値です。

スクリプトで使用される変数の名前は、Setup.iss ファイルに関して意味がありません。ただし、Setup.iss ダイアログ データ セクションでは、ビルトイン ダイアログ ボックスと Sd ダイアログに、パラメーターにマップする独自のキー名のセットがそれぞれあります。キー名は重要で、各ダイアログに対して定義されたキー名と同じでなければなりません。次の「ダイアログ データのキー名リスト」を参照してください。

カスタム ダイアログを使用する場合、各ダイアログの Result キー名に対するダイアログ データ エントリ、およびカスタム ダイアログによって設定または返される他の値に対するエントリを作成する必要があります。カスタム ダイアログに対する keyname=value 式の作成方法の指針として、下のダイアログ データのキー名リストを使用

してください。**GetProfString** または **GetProfInt** を呼び出して、カスタム ダイアログに対するダイアログ データ情報を確認します。**GetProfString** および **GetProfInt** によって、.iss ファイル、セクション、およびキー名を指定することができ、指定したキー名に割り当てられた値が返されます。

Result キー名の標準値

カスタム ダイアログを含むすべてのダイアログによって、キー名 Result の値が戻され、ダイアログを終了するためにクリックされたボタンを示します。ボタンが押されたことが示されない場合、Result の標準値は以下のようになります。

- 12 – [戻る] ボタン
- 1 – [次へ] または [OK] ボタン

コンポーネントとサブコンポーネントの選択の記録

あるダイアログ関数を使用すると、エンドユーザーがコンポーネントやサブコンポーネントを選択できるようになります。コンポーネントとサブコンポーネントの選択を応答ファイルに記録するために使用されるダイアログ データのキー名エントリには、type、count、および <#> の3 種類があります(次を参照)。

コンポーネント選択の各セットおよびサブコンポーネント選択の各セットには、1 つの type キー名エントリ、1 つの count キー名エントリ、および各コンポーネントまたはサブコンポーネントの選択を記録するために必要な数の <#> キー名エントリがあります。

コンポーネントの選択を記録するためにキー名を作成する際、以下のように、type、count、および <#> キー名エントリの前に "Component" という語を入力します。

```
Component-typeComponent-countComponent-0
```

サブコンポーネントの選択を記録するためにキー名を作成する際、以下のように、type、count、および <#> キー名エントリの前に、サブコンポーネントが属するコンポーネントの名前を入力します。

```
Program Files-typeProgram Files-countProgram Files-0Program Files-1
```

完全な値エントリを作成するには、等号記号を使用して、値をキー名に付加します(各種類のキー名に割り当てられる値のタイプは、下に説明されています)。次の例は、**Program Files** と **Binary Files** という 2 つのコンポーネントと、**Executables** と **Support Elements** という **Program Files** の 2 つのサブコンポーネントの完全な値エントリです。

```
Component-type=string
Component-count=2
Component-0=Program Files
Component-1=Binary Files
Program Files-type=string
Program Files-count=2
Program Files-0=Executables
Program Files-1=Support Elements
```

type キー名によって、コンポーネントまたはサブコンポーネントリストのデータ型が示されます。InstallShield ダイアログは現在、コンポーネント リストとサブコンポーネント リストに、文字列リストしか使用しないため、Component-type=string のように、type は常に「**string**」に等しくなります。今後のバージョンでは、番号リストが使用される可能性もあり、その場合、type は **number** に等しくなります。

count キー名エントリ

count は、特定のコンポーネントまたはサブコンポーネントエントリに対する選択の数に等しくなります。たとえば、**ComponentDialog** ダイアログで 2 つのコンポーネントがインストールに選択された場合、カウント ダイアログ データ エントリは **Component-count=2** のようになります。**Program Files** コンポーネントの 2 つのサブコンポーネントが選択された場合、**Program Files-count=2** エントリが作られます。

<#> キー名エントリ

<#> キー名エントリの数字部分は、0 で開始して 1 ずつ増加する続き番号で、記録された各コンポーネントまたはサブコンポーネントの選択を番号付けします。番号付けは 0 から開始するため、最大値は、常に count の値より 1 つ少なくなります。

<#> キー名エントリに割り当てられる値は、選択されたコンポーネントまたはサブコンポーネントの表示名 (コンポーネント リストまたはサブコンポーネント リストがビルドされた際に、2 番目のパラメーターとして **ComponentAddItem** に渡された文字列) です。

たとえば、**ComponentDialog** ダイアログによって、**Program Files**、**Help Files**、**Sample Files**、および **Utilities** のコンポーネントの選択がエンドユーザーに提供される例を考えてみましょう。エンドユーザーが **Program Files** と **Help Files** を選択した場合、**ComponentDialog** ダイアログのこのインスタンスに対するダイアログ データ セクションには、2 つのリスト項目エントリが含まれ、次のようになります。

```
[{23EAFFCA-361D-11D3-8B0F-00105A9846E9}-ComponentDialog-0]
szDir=C:¥MYAPP¥FILES
Component-type=string
Component-count=2
Component-0=Program Files
Component-1=Help Files
Result=
```

次の例では、サブコンポーネントリストの選択が記録される方法について説明しています。例は、**SdComponentMult** ダイアログのインスタンスの場合です。例では、**Program Files** および **Help Files** の 2 つのコンポーネントが選択されていることが示されています。また、**Main Files**、**Aux.Files**、**Main Help**、および **Tutorial Files** の 4 つのサブコンポーネントも選択されたことを示します。**Main Files** と **Aux.Files** は、**Program Files** のサブコンポーネントであり、**Main Help** と **Tutorial Files** は、**Help Files** のサブコンポーネントです。

```
[{23EAFFCA-361D-11D3-8B0F-00105A9846E9}-SdComponentMult-0]
Component-type=string
Component-count=2
Component-0=Program Files
Component-1=Help Files
Program Files-type=string
Program Files-count=2
Program Files-0=Main Files
Program Files-1=Aux.ファイル
Help Files-type=string
Help Files-count=2
Help Files-0=Main Help
Help Files-1=Tutorial Files
Result=1
```

ダイアログ データのキー名リスト

InstallShield ダイアログに対するダイアログ データのキー名は、次のテーブルに一覧表示されています。最初の列には、ダイアログ名が含まれています。2 番目の列には、各ダイアログに適用できるキー名が一覧表示されています。3 番目の列には、キー名に関連した値の説明が含まれています。

テーブル 17 • ダイアログ データのキー名

ダイアログ	キー名	説明
AskDestPath-<#>	結果	標準値
AskDestPath-<#>	szPath	編集フィールドで設定されたパス
AskOptions-<#>	結果	標準値 戻り値の数はスクリプト呼び出しに基づいて変更できるため、AskOptions は特殊です。<#> が選択変数の数である Sel-<#> という形式のキー名のシーケンスを使用できます。番号は 0 で始まります。Sel-<#> エントリの数は、**AskOptions** への特定の呼び出しの変数の数 (チェック ボックス/ラジオ ボタン) に一致します。次の例を参照してください: • **Sel-0**—**AskOptions** の最初の選択変数にマップします。 • **Sel-1**—**AskOptions** の 2 番目の選択変数にマップします。 • **Sel-2**—**AskOptions** の 3 番目の選択変数にマップします。
AskPath-<#>	結果	標準値
AskPath-<#>	szPath	"インストール先" 編集フィールドに入力されたパス
AskText-<#>	結果	標準値
AskText-<#>	szText	編集フィールドからのテキスト
AskYesNo-<#>	結果	1 = エンドユーザーが [はい] ボタンをクリックしました。 0 = ユーザーが [いいえ] ボタンをクリックしました。
ComponentDialog-<#>	結果	標準値
ComponentDialog-<#>	szDir	[インストール先] ディレクトリ フィールドに入力されたパス
ComponentDialog-<#>	Component-type	文字列 (現在使用できる唯一の値です)
ComponentDialog-<#>	Component-count	コンポーネントの選択の合計数
ComponentDialog-<#>	Component-<#>	選択した項目のリスト内の番号 (番号付けは 0 で開始します)

テーブル 17・ダイアログ データのキー名 (続き.)

ダイアログ	キー名	説明
MessageBox-<#>	結果	1 = ユーザーが [OK] ボタンをクリックしました。
RebootDialog-<#>	結果	0 (Result は常に 0 です)
RebootDialog-<#>	Choice	マシンを再起動する前の最後の再起動選択モードを示します。ラジオ ボタンにマップし、以下の可能値があります。 • 602 = "はい、今すぐコンピューターを再起動します。" • 0 = "いいえ、あとでコンピューターを再起動します。" に設定します。
SdAskDestPath-<#>	結果	標準値
SdAskDestPath-<#>	szDir	[インストール先] ディレクトリ フィールドに入力されたパス
SdAskOptions-<#>	結果	標準値
SdAskOptions-<#>	Component-type	文字列 (現在使用できる唯一の値です)
SdAskOptions-<#>	Component-count	コンポーネントの選択の合計数
SdAskOptions-<#>	Component-<#>	選択した項目のリスト内の番号 (番号付けは 0 で開始します)
SdAskOptionsList-<#>	結果	標準値
SdAskOptionsList-<#>	Component-type	文字列 (現在使用できる唯一の値です)
SdAskOptionsList-<#>	Component-count	コンポーネントの選択の合計数
SdAskOptionsList-<#>	Component-<#>	選択した項目のリスト内の番号 (番号付けは 0 で開始します)
SdBitmap-<#>	結果	標準値
SdComponentDialog-<#>	結果	標準値
SdComponentDialog-<#>	szDir	[インストール先] ディレクトリ フィールドに入力されたパス
SdComponentDialog-<#>	Component-type	文字列 (現在使用できる唯一の値です)

テーブル 17 • ダイアログ データのキー名 (続き.)

ダイアログ	キー名	説明
SdComponentDialog-<#>	Component-count	コンポーネントの選択の合計数
SdComponentDialog-<#>	Component-<#>	選択した項目のリスト内の番号 (番号付けは 0 で開始します)
SdComponentDialog2-<#>	結果	標準値
SdComponentDialog2-<#>	Component-type, <サブコンポーネント>-type	文字列 (現在使用できる唯一の値です)
SdComponentDialog2-<#>	Component-count, <サブコンポーネント>-count	コンポーネントまたはサブコンポーネントの選択の合計数
SdComponentDialog2-<#>	Component-<#>, <サブコンポーネント>-<#>	選択した項目のリスト内の番号 (番号付けは 0 で開始します)
SdComponentDialogAdv-<#>	結果	標準値
SdComponentDialogAdv-<#>	szDir	[インストール先] ディレクトリ フィールドに入力されたパス
SdComponentDialogAdv-<#>	Component-type	文字列 (現在使用できる唯一の値です)
SdComponentDialogAdv-<#>	Component-count	コンポーネントの選択の合計数
SdComponentDialogAdv-<#>	Component-<#>	選択した項目のリスト内の番号 (番号付けは 0 で開始します)
SdComponentMult-<#>	結果	標準値
SdComponentMult-<#>	Component-type, <サブコンポーネント>-type	文字列 (現在使用できる唯一の値です)

テーブル 17 • ダイアログ データのキー名 (続き.)

ダイアログ	キー名	説明
SdComponentMult-<#>	Component-count, <サブコンポーネント>-count	コンポーネントまたはサブコンポーネントの選択の合計数
SdComponentMult-<#>	Component-number, <サブコンポーネント>-<#>	選択した項目のリスト内の番号 (番号付けは 0 で開始します)
SdComponentTree-<#>	結果	標準値
SdConfirmNewDir-<#>	結果	1 = エンドユーザーが [はい] ボタンをクリックしました。 0 = ユーザーが [いいえ] ボタンをクリックしました。
SdConfirmRegistration-<#>	結果	1 = エンドユーザーが [はい] ボタンをクリックしました。 0 = ユーザーが [いいえ] ボタンをクリックしました。
SdDisplayTopics-<#>	結果	標準値
SdFinish-<#>	結果	1 = 完了
SdFinish-<#>	bOpt1	1 = "はい、README ファイルを開きます。" が選択されました。 0 = "はい、README ファイルを開きます。" が選択されていません。
SdFinish-<#>	bOpt2	1 = "はい、今すぐ [app name] を再起動します。" が選択されました。 0 = "はい、今すぐ [app name] を再起動します。" が選択されていません。
SdFinishEx		特殊処理を必要とします。
SdFinishReboot-<#>	結果	1 = 完了
SdFinishReboot-<#>	BootOption	0 = Windows をマシンで再起動しない 2 = Windows を再起動 3 = マシンの再起動
SdLicense-<#>	結果	12 = エンドユーザーが [戻る] ボタンをクリックしました。 1 = エンドユーザーが [はい] ボタンをクリックしました。

テーブル 17 • ダイアログ データのキー名 (続き.)

ダイアログ	キー名	説明
SdOptionsButtons-<#>	結果	12 = エンドユーザーが [戻る] ボタンをクリックしました。 または、[次へ] ボタンがクリックされたことを示します。 101 = オプション ボタン 1 (上) が選択されました。 102 = オプション ボタン 2 が選択されました。 103 = オプション ボタン 3 が選択されました。 104 = オプション ボタン 4 が選択されました。
SdRegisterUser-<#>	結果	標準値
SdRegisterUser-<#>	szName	"名前" フィールドに入力されたテキスト
SdRegisterUser-<#>	szCompany	"会社" フィールドに入力されたテキスト
SdRegisterUserEx-<#>	結果	標準値
SdRegisterUserEx-<#>	szName	"名前" フィールドに入力されたテキスト
SdRegisterUserEx-<#>	szCompany	"会社" フィールドに入力されたテキスト
SdRegisterUserEx-<#>	szSerial	"シリアル (番号)" フィールドに入力されたテキスト
SdSelectFolder-<#>	結果	標準値
SdSelectFolder-<#>	szFolder	"プログラム フォルダー" フィールドに入力されたフォルダー名
SdSetupType-<#>	結果	12 = エンドユーザーが [戻る] ボタンをクリックしました。 または、[次へ] ボタンがクリックされたことを示します。 301 = 標準のラジオ ボタンが現在選択されています。 302 = コンパクト ラジオ ボタンが現在選択されています。 303 = カスタム ラジオ ボタンが現在選択されています。
SdSetupType-<#>	szDir	"インストール先" 編集フィールドに入力されたパス
SdShowDlgEdit1-<#>	結果	標準値
SdShowDlgEdit1-<#>	szEdit1	svEdit1 フィールドに入力されたテキスト
SdShowDlgEdit2-<#>	結果	標準値
SdShowDlgEdit2-<#>	szEdit1	svEdit1 フィールドに入力されたテキスト
SdShowDlgEdit2-<#>	szEdit2	svEdit2 フィールドに入力されたテキスト

テーブル 17 • ダイアログ データのキー名 (続き.)

ダイアログ	キー名	説明
SdShowDlgEdit3–<#>	結果	標準値
SdShowDlgEdit3–<#>	szEdit1	svEdit1 フィールドに入力されたテキスト
SdShowDlgEdit3–<#>	szEdit2	svEdit2 フィールドに入力されたテキスト
SdShowDlgEdit3–<#>	szEdit3	svEdit3 フィールドに入力されたテキスト
SdShowFileMods–<#>	結果	標準値
SdShowFileMods–<#>	nSelection	"セットアップで実施する機能を選択してください" ラジオボタン選択: 101 = セットアップに AUTOEXEC.BAT ファイルを変更させる。 102 = AUTOEXEC.NEW ファイルに指定された変更を保存する。 103 = 変更しない
SdShowInfoList–<#>	結果	標準値
SdStartCopy–<#>	結果	標準値
SdWelcome–<#>	結果	標準値
SdWelcomeMaint–<#>	結果	301 = [次へ] ボタンがクリックされたとき、[変更] ボタンが選択されたことを示します。 302 = [次へ] ボタンがクリックされたとき、[修復] ボタンが選択されたことを示します。 303 = [次へ] ボタンがクリックされたとき、[削除] ボタンが選択されたことを示します。
SelectDir–<#>	結果	標準値
SelectDir–<#>	szDir	このダイアログによって選択されたパス
SelectDirEx–<#>	結果	標準値
SelectDirEx–<#>	szDir	このダイアログによって選択されたパス
SelectFolder–<#>	結果	標準値
SelectFolder–<#>	szResultFolder	このダイアログによって選択されたフォルダー

テーブル 17 • ダイアログ データのキー名 (続き.)

ダイアログ	キー名	説明
SetupType-<#>	結果	12 = エンドユーザーが [戻る] ボタンをクリックしました。 または、[次へ] ボタンがクリックされたことを示します。 301 = 標準のラジオ ボタンが現在選択されています。 302 = コンパクト ラジオ ボタンが現在選択されています。 303 = カスタム ラジオ ボタンが現在選択されています。
SprintfBox-<#>	結果	1 = ユーザーが [OK] ボタンをクリックしました。
Welcome-<#>	結果	標準値

SdFinishEx の特殊処理

SdFinishEx は、システム変数 *BATCH_INSTALL* の値に応じて **SdFinish** または **SdFinishReboot** のどちらかを呼び出します。このため、*SdFinishEx* を応答ファイルで処理することはできません。*SdFinishEx* を呼び出すスクリプトのサイレント インストールを作成するには、次のようなスクリプト コードを利用します:

```
if (MODE = NORMALMODE) then
    SdFinishEx (szTitle, szMsg1, szMsg2, szOpt1, szOpt2, bvOpt1, bvOpt2);
else
    if !BATCH_INSTALL then
        /* 再起動が必要ないサイレント インストール中に
        bvOpt1 と bvOpt2 に持たせる値を設定します。
        例: */
        bvOpt1 = FALSE;
        bvOpt2 = TRUE;
    else
        /* サイレント インストールが再起動を必要なとき、
        即再起動する場合、
        次のようなシステム関数を呼びます:*/
        System (SYS_BOOTMACHINE);
    endif;
endif;
```

サンプル応答ファイル



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

次の応答ファイルは、サイレント InstallShield インストール用の .iss ファイルです。

```
[InstallShield Silent]
Version=v7.00
File=Response File

[Application]
```

```
Name=InstallShield
Version=10.50.000
Company=ソフトウェア会社名

[{77AB941D-5876-11D4-A4A2-006067620F66}-DlgOrder]
Dlg0={77AB941D-5876-11D4-A4A2-006067620F66}-SdBitmap-0
Count=8
Dlg1={77AB941D-5876-11D4-A4A2-006067620F66}-Welcome-0
Dlg2={77AB941D-5876-11D4-A4A2-006067620F66}-SdRegisterUser-0
Dlg3={77AB941D-5876-11D4-A4A2-006067620F66}-AskDestPath-0
Dlg4={77AB941D-5876-11D4-A4A2-006067620F66}-SetupType-0
Dlg5={77AB941D-5876-11D4-A4A2-006067620F66}-SelectFolder-0
Dlg6={77AB941D-5876-11D4-A4A2-006067620F66}-SdStartCopy-0
Dlg7={77AB941D-5876-11D4-A4A2-006067620F66}-SdFinish-0

[{77AB941D-5876-11D4-A4A2-006067620F66}-SdBitmap-0]
Result=1

[{77AB941D-5876-11D4-A4A2-006067620F66}-Welcome-0]
Result=1

[{77AB941D-5876-11D4-A4A2-006067620F66}-SdRegisterUser-0]
szName=John Doe
szCompany=ソフトウェア会社名
Result=1

[{77AB941D-5876-11D4-A4A2-006067620F66}-AskDestPath-0]
szPath=C:¥Program Files¥製品名バージョン
Result=1

[{77AB941D-5876-11D4-A4A2-006067620F66}-SetupType-0]
Result=301
szDir=C:¥Program Files¥製品名バージョン

[{77AB941D-5876-11D4-A4A2-006067620F66}-SelectFolder-0]
szResultFolder=製品名バージョン
Result=1

[{77AB941D-5876-11D4-A4A2-006067620F66}-SdStartCopy-0]
Result=1

[{77AB941D-5876-11D4-A4A2-006067620F66}-SdFinish-0]
Result=1
bOpt1=1
bOpt2=0
```

サイレント インストールの実行



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

インストールと応答ファイルを作成した後、次の操作を行います:

1. [サポート ファイル/ビルボード] ビューで、(デフォルトで Windows フォルダーに格納されている) 応答ファイルをアドバンス ファイル の下にある Disk1 フォルダーに追加します。
2. リリースをビルドします。インストールに自己展開型実行可能ファイルを作成する場合、リリース ウィザードの [一般オプション] パネルにある "セットアップ コマンドライン" プロパティか、または [リリース] ビューの "セットアップ コマンドライン" プロパティに –s を入力します。

これで、InstallShield Silent を使ってインストールをサイレント モードで実行する準備が整いました。インストールをサイレント モードで実行する際、メッセージが表示されない点に注意してください。代わりに、Setup.log という名前のログ ファイルによって、インストールが正常に終了したかどうかを含むインストール情報がキャプチャされます。ログファイルを確認して、インストールの結果を判断することができます。(一部のインストール初期化エラーでは、ログファイルの名前が Setupexe.log で、インストールがインターネットから実行される場合はそれが SUPPORTDIR に作成され、それ以外の場合は SRCDIR に作成されることがありますのでご注意ください。)

InstallShield Silent を起動するには、–s オプションで Setup.exe を実行します。自己展開型実行可能ファイルを作成した場合、上記手順の 2 で –s オプションを含んでいるため、単に起動するだけです。

InstallShield には、–f1 および –f2 スイッチがあり、応答ファイルの名前と場所、およびログファイルの場所を指定できます。詳細については、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

サイレント インストールが正常に終了したかどうかを検証するには、Setup.log の [ResponseResult] セクションの ResultCode 値を確認します。InstallShield は、ResultCode キー名のあとに、適切な戻り値を書き込みます。

Setup.log ファイルを使用してエラーを確認する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

エンドユーザーが Setup.exe を /s 引数で実行した場合に生成される Setup.log は、サイレント インストール ログファイルのデフォルトの名前です。Setup.log は、デフォルトで応答ファイル Setup.iss のあるディレクトリに作成されます。/f1 および /f2 スイッチを Setup.exe に使用することで、Setup.logに異なる名前と場所を指定することができます。

Setup.log ファイルには 3 つのセクションがあります。最初のセクション [InstallShield Silent] は、サイレント インストールで使用した InstallShield Silent のバージョンを識別します。また、ファイルがログファイルであることも表します。

2 番目の Application セクションは、インストールされたアプリケーションの名前とバージョン、および会社名を識別します。

3 番目の ResponseResult セクションには、サイレント インストールが成功したかどうかを示す結果コードが含まれます。[ResponseResult] セクションの ResultCode キー名に、整数値が割り当てられます。インストールは、ResultCode キーに以下の戻り値のいずれかを設定します。



プロジェクト: *InstallScript MSI インストールでは、応答ファイルの読み取りまたは書き込み時に初期化プロセスはありません。そのため、InstallScript MSI インストールで発生する可能性のあるエラーは、サイレント インス*

トールの場合は -3、記録インストールの場合は -6 です。以下の表は、該当する適切なプロジェクトの種類を示します。

テーブル 18 ・ Setup.log ファイルの値を返す

結果コード	プロジェクトの種類	説明
0	InstallScript、InstallScript MSI	成功
-1	InstallScript	一般エラー。
-2	InstallScript	モードが間違っています。
-3	InstallScript、InstallScript MSI	必要なデータが Setup.iss ファイルに見つかりません。
-4	InstallScript	実行に十分なメモリがありません。
-5	InstallScript	ファイルが存在しません。
-6	InstallScript、InstallScript MSI	応答ファイルに書き込むことができません。
-7	InstallScript	ログファイルに書き込むことができません。
-8	InstallScript	InstallShield Silent 応答 (.iss) ファイルのパスが間違っています。
-9	InstallScript	リストの種類が無効です (文字列 /番号)。
-10	InstallScript	データ 型が無効です。
-11	InstallScript	セットアップ中に不明のエラーが発生しました。
-12	InstallScript	ダイアログ ボックスに不具合があります。
-51	InstallScript	指定したフォルダーを作成することができません。
-52	InstallScript	指定したファイルまたはフォルダーにアクセスできません。
-53	InstallScript	選択したオプションは無効です。

サイレント インストールが成功した場合、Setup.log ファイルの内容は次のようになります。

```
[Application]
Name=Sample App 3000
Version=1.00.0000
Company=Sample Software Corporation
Lang=0409

[ResponseResult]
```

ResultCode=0

InstallScript MSI プロジェクトの MSI サイレント モード インストール



プロジェクト: *この情報は、InstallScript MSI プロジェクトに適用します。*

Setup.exe を使用せずに InstallScript MSI プロジェクトをサイレントでインストールする必要がある場合、MSI サイレント モードを使用することができます。

InstallScript MSI プロジェクトの MSI サイレント インストールを起動する

MSI サイレント インストールは、次のような場合に起動されます。

- コマンドラインで次のように入力してインストールを起動した場合：

  msiexec product.msi /qn

- アドバタイズされたショートカットからインストールを起動した場合。
- 要求時インストールでインストールを起動した場合。
- アップデート パッケージ実行中に製品が削除された場合。

従来のサイレントモードとは異なり、MSI サイレントモードはスクリプトで提供される通常の論理に従いません。単に InstallExecuteSequence テーブルを通して実行されます。(このテーブルを表示するには、ダイレクト エディターに移動します。)その結果、次を含む一部のイベントは呼び出されません。

- すべての UI インストール イベント—OnFirstUIBefore と OnFirstUIAfter
- すべての機能イベント

OnMsiSilentInstall イベント

このイベントの機能

InstallScript MSI ベースのインストールでは、製品がターゲット システムにインストールされていない場合、InstallShield は OnMsiSilentInstall イベント ハンドラーを起動します。これは、インストールがアドバタイズされたショートカットからアクティベートされた場合、または、インストールが次のコマンドラインを使用して起動された場合に発生します。

msiexec product.msi /qn

MSI サイレントインストールモードをサポートする場合は、OnMsiSilentInstall イベントを上書きする必要があります。これにより、通常 OnFirstUIBefore、OnFirstUIAfter、および機能イベント ハンドラーで実行されるタスクを実行できます。

イベントの上書き

デフォルトでは、**OnMsiSilentInstall** によりメッセージが表示され、インストールは中止されます。このイベント ハンドラーは、この関数の独自の実装を作成することにより上書きできます。次にこの機能のプロトタイプを示します。

external prototype OnMsiSilentInstall(HWND hInstall);

ここで、hInstall はインストールのハンドラーです。

最も簡単な方法は、インストールが中止しないようにこのイベントの空の本文を実装することです。例:

```
function OnMsiSilentInstall(hInstall)
begin
  //何もしないで、インストールを継続します。
end;
```

MSI サイレント インストール、およびアドバタイズされたショートカットのアクティベーションにより **OnMsiSilentInstall** が再び起動されます。オンデマンド インストール、自動修復、およびアンインストール モードでは呼び出されません。

サイレント インストールが実行されているモードの検出

InstallShield スクリプトから従来のサイレント インストールが実行されているかどうかを検出するには 次を使用します。

```
if (MODE = SILENTMODE)
```

InstallShield スクリプトから MSI サイレント モード インストール (/q、アドバタイズ、自動修復、アンインストールまたはオンデマンド インストールを含む) が実行されているかどうかを検出するには 、Windows Installer API 関数の **MsiGetProperty** を使用して Windows Installer プロパティ ISSETUP_UISEQUENCE_PROCESSED を確認します。このプロパティが設定されていない場合は、サイレント インストールです。(InstallUISequence が実行されないことを示します。)

コマンドラインを使ったサイレント アンインストール (InstallScript MSI のみ)

プロジェクト: *この情報は、InstallScript MSI プロジェクトに適用します。*

InstallScript MSI インストールを使ってインストールされた製品をサイレント アンインストールするには、応答ファイルを使います。

タスク ***応答ファイルを使ってアンインストールを実行するには、以下の手順に従います:***

1. Setup.exe を /r 引数と共に実行してアンインストール用の応答ファイル (.ss) を準備します。

   Setup.exe /r

2. 次の行をコマンドラインに入力します (イタリック体で表示されているアイテムは、製品のアンインストールに固有のデータを示します):

   Setup.exe /s /f1"*FullPath¥YourResponseFile.iss*"

スイート インストールをコマンドラインから最小ユーザー インターフェイスを使って実行する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

進行状況パネルを表示するのみの最小 UI モードでスイート インストールを実行するコマンドライン パラメーターを使用できます。

スイート インストールを最小 UI モードで実行するには、コマンドラインで /passive パラメーターを使用します:

Setup.exe /passive

最小 UI モードを使ってアンインストールするには、次のコマンドを使用します:

Setup.exe /passive /remove

コマンドラインからアップデートのダウンロードを行わずにスイートのサイレント インストールを実行する



プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

/skipupdate というコマンドライン パラメーターを使ってスイート/アドバンスト UI サイレント インストールを構成し、自動的にユーザーによるダウンロード可能なアップデートの適用を強制する代わりに、インストーラーがアップデートのダウンロードをスキップしてサイレント インストールを続行するように指示します。

ユーザーによるアップデートのダウンロードを強制せずにスイート/アドバンスト UI サイレント インストールを実行するには、次のコマンドラインで /skipupdate パラメーターを使用します:

Setup.exe /silent /skipupdate

このコマンドライン パラメーターは、サイレント モードによるスイート/アドバンスト UI インストール機能を強化します。

プロジェクトの検証

プロジェクトを検証するには、一連の内部整合性評価プログラム (ICE) のルールをインストールまたはマージ モジュール パッケージに適用します。ICE はアクションが正常に実行するために必要な有効データベースがパッケージに含まれているかどうかを判断するのに役立ちます。インストール プロジェクトがこれらの ICE のいずれかに違反する場合、InstallShield は違反の対象となった ICE 規則をレポートし、問題を解決するための追加情報を提供します。

InstallShield では、Microsoft 作成による多くの ICE を利用できます。また、Revenera が作成したカスタム InstallShield ICE (ISICE) も、InstallShield インストールとマージ モジュール検証スイートの一部で利用することができます。ISICE は、パッケージを Windows ベースのインストール用のベスト プラクティスに対して検証するのに使用できます。



Windows ロゴ: *インストールがマイクロソフトの Windows ロゴ プログラムの要件を満たしているかどうかを判別するために、プロジェクトの検証が役に立つ場合があります。従って、Windows ロゴの認証に関心があるユーザーは、**InstallShield 検証スイート – Windows 8** および**完全 MSI 検証スイート**を使ってインストール パッケージの検証を行うことをお勧めします。InstallShield でマージ モジュールを作成した場合、**InstallShield マージ モジュー***

ル検証スイート – Windows 8 およびマージ モジュール検証スイートを利用して、マージ モジュールを検証することができます。詳細については、「Windows ロゴ プログラムの要件」を参照してください。

Windows ロゴの取得に関する詳細は、MSDNを参照してください。



***エディション:** InstallShield Premier Edition には、次の一連の検証スイートが含まれています。*

InstallShield Premier Edition には、次の一連の検証スツールが含まれています:

- ***InstallShield 仮想化適合性スイート**—このスイートに含まれる ISVICE 検証ツールを使って、Microsoft App-V 4.x、Microsoft App-V 5.x、Microsoft Server App-V、VMware ThinApp、および Citrix XenApp との適合性を確認することで、製品が仮想化に必要な準備が整っているかどうかを判断することができます。顧客に仮想バージョンの提供を考慮する場合に、検証スイートを使用して、製品をどのようにビルドするか豊富な情報に基づいた意思決定を行うことができます。*
- ***InstallShield ベスト プラクティス スイート**—インストールがベスト プラクティス ガイドラインに違反している場合、このスイートの ISBP 検証ツールによってアラートされます。*
- ***InstallShield MSIX/UWP アプリ適合性スイート**—このスイートに含まれる ISUWP 検証ツールは、.msi パッケージで Windows アプリ パッケージ フォーマットに適さないアイテムの存在をスキャンして、既知の Windows アプリのバリエーション (ユニバーサル アプリ、デスクトップ ブリッジ、Windows ストア、WSA、および Nano Server) に対する適合性を示すレポートを提供します。*

InstallShield では、アップグレード検証とパッチ検証のエンジンも提供しています。このエンジンへは、アップグレード検証ウィザードを通してアクセスできます。

インストール パッケージまたはマージ モジュールを検証する

InstallShield 内からインストール パッケージまたはマージ モジュールの検証方法には 2 種類あります:

- InstallShield を構成して、リリースのビルド時に毎回インストール パッケージとマージ モジュールの検証を行うようにすることができます。これは、デフォルトで無効になっています。詳細については、「ビルド時に検証を行うかどうかを指定する」を参照してください。

  ビルド時に検証を行う場合、複数の検証スイートを指定できます。

- ビルド済みのリリースに対してオンデマンドで検証を行うことができます。詳しくは、「インストール パッケージまたはマージ モジュールをオン デマンドで検証する」をご覧ください。

  オンデマンドで検証を行う場合、検証スイートは 1 回の実行につき 1 つのみ指定できます。



***ヒント:** 代わりの方法として、ビルドが完了したあと、ISCmdBld.exe を使ってコマンドラインからインストール パッケージやマージ モジュールを検証することもできます。詳細については、「ISCmdBld.exe」を参照してください。*



***プロジェクト:** MSI データベース プロジェクトまたは MSM データベース プロジェクト場合、パッケージの検証はオンデマンドでのみ行うことができます。*



メモ: インストールまたはマージ モジュールに適用する検証の警告を見るには、オンデマンドで検証を実行する必要があります。このタイプの検証メッセージは、ビルド時に検証を実行しても見ることができません。InstallShield では、どちらの検証方法でも、他のタイプの検証メッセージ (エラーおよび失敗) が報告されます。異なるタイプの検証メッセージについては、「検証結果を表示する」を参照してください。

ビルド時に検証を行うかどうかを指定する

InstallShieldでは、リリースが正常にビルドされるたびにインストール パッケージおよびマージ モジュールを検証するかどうかを指定することができます。InstallShield ではまた、検証に使う検証スイートも指定できます (複数可)。



タスク ***検証の設定を構成するには、***

1. **[ツール]** メニューから **[オプション]** を選択します。**Options** ダイアログ ボックスが開きます。
2. **[検証]** タブをクリックします。
3. InstallShield がビルド時に行う検証の種類に当てはまるチェック ボックスを選択します。ビルド時ではなくオンデマンドで検証を行う場合は、チェック ボックスの選択を解除します。
4. **[次を使用して検証を実行する]** チェック ボックスまたは **[次を使用してマージモジュールを検証する]** チェック ボックスを選択したとき、実行する検証タイプも選択します (複数可)

   新しい、またはカスタム検証ツール (.cub file) を追加するには、**[参照]** ボタンをクリックして、適切な検証ツールを作成します。



メモ: インストールまたはマージ モジュールに適用する検証の警告を見るには、オンデマンドで検証を実行する必要があります。このタイプの検証メッセージは、ビルド時に検証を実行しても見ることができません。InstallShield では、どちらの検証方法でも、他のタイプの検証メッセージ (エラーおよび失敗) が報告されます。

検証中に実行する ICE、ISICE、ISVICE および ISBP を指定する

InstallShieldでは、インストール パッケージおよびマーモジュールの検証に使用する ICE、ISICE、ISVICE、ISUWP および ISBP の項目をカスタマイズすることができます。



タスク ***検証中に実行する ICE、ISICE、ISVICE、ISUWP および ISBP を指定するには、以下の手順に従います:***

1. **[ツール]** メニューから **[オプション]** を選択します。**Options** ダイアログ ボックスが開きます。
2. **[検証]** タブをクリックします。
3. **[カスタマイズ]** ボタンをクリックします。**[検証設定のカスタマイズ]** ダイアログ ボックスが表示されます。
4. **[カスタマイズする CUB ファイルを選択]** リストで、カスタマイズするスイートをクリックします。
5. ICE 一覧で、インストール パッケージまたはマージ モジュールの評価に使用する各検証ツールのチェック ボックスを選択します。検証に使用しない検証ツールの各チェック ボックスの選択を解除します。

連続する複数の検証ツールを構成するには、最初のファイルを選択してから SHIFT キーを押したまま最後のファイルを選択します。そのあと、該当する検証ツールのチェック ボックスを選択またはクリアします。

6. **[OK]** をクリックします。

検証スイートの ICE、ISICE、ISVICE、ISUWP および ISBP のリストをカスタマイズすると、(ビルド時またはオンデマンドで) 検証が実行されるたびに選択された ICE、ISICE、ISVICE、ISUWP または ISBP のみが使用されます。

インストール パッケージまたはマージ モジュールをオン デマンドで検証する

InstallShield では、ビルド プロセスとは切り離してインストール パッケージまたはマージ モジュールを検証することができます。これは、ビルドに成功したあと検証が毎回実行されないように InstallShield を構成してあるけれども、ある時点で製品が Windows ロゴ コンプライアンスまたはベスト プラクティス コンプライアンスに準拠しているかどうかテストするために検証を行う必要がある場合、特に便利です。



タスク ***リリースをオン デマンドで検証するには、以下の手順に従います。***

1. 適切なリリースのビルドを完了します。
2. **[ビルド]** メニューで **[検証]** をポイントして、実行する検証タイプをクリックします。

検証結果を表示する

検証スキャンの結果は、出力ウィンドウの [検証] タブで表示されます。また、この結果は、リリース フォルダー にある ValidationFiles フォルダーに XML ドキュメントとして保存されます。このファイルは、ビルド ディレクトリ、または、[リリース] ビューに移動し、リリースの下で Validations フォルダーを選択すると表示できます。

検証メッセージ

検証メッセージは 3 つに分類されています。

- **エラー**—重複コンポーネント GUID など、データベースに関する問題を説明します。
- **警告**—特定の状況下で発生する可能性がある、データベースに関する問題を説明します。
- **失敗**—検証ツールも実行できなくなるほどの深刻な問題がデータベースにあるときに表示されます。

プロジェクトのスキャン結果が検証メッセージを含む場合、メッセージおよび関連するコードも出力ウィンドウの [タスク] タブに一覧表示されます。



メモ: *インストールまたはマージ モジュールに適用する検証の警告を見るには、オンデマンドで検証を実行する必要があります。このタイプの検証メッセージは、ビルド時に検証を実行しても見ることができません。InstallShield では、どちらの検証方法でも、他のタイプの検証メッセージ (エラーおよび失敗) が報告されます。これらの 2 つの検証方法については、「インストール パッケージまたはマージ モジュールを検証する」を参照してください。*



ヒント: 出力ウィンドウの [タスク] タブの検証コードをクリックすると、対応するナレッジ ベース記事または HelpNet のトピックを表示することができます。

また、[タスク] タブで検証メッセージをクリックすると、その検証メッセージに対応するダイレクト エディター の領域にすぐ移動することができます。この機能は、ICE、ISICE、ISVICE、ISUWP および ISBP で提供されています。

ICE

検証ツールにより、プロジェクトが各内部整合性評価プログラム (ICE) に準拠しているかどうかが確認されます。これらの ICE は、Msival2.exe (Microsoft Windows Installer Platform SDK の一部) が、インストール およびマージ モジュール パッケージの Windows ロゴ コンプライアンスを検証するときに使用されます。

インストール プロジェクトまたはマージ モジュール プロジェクトがこれらの ICE のいずれかに違反する場合、InstallShield は違反の対象となった ICE 規則をレポートし、問題を解決するための追加情報を提供します。

特定の ICE については、Windows Installer ヘルプ ライブラリの「ICE Reference」を参照してください。

ISICE

InstallShield 検証スイートは、プロジェクトが Windows ロゴ プログラムに準拠していることを確認するのに役立つ多数の InstallShield ICE (ISICE) で構成されています。

InstallShield で使用できる ISICE リストは次の表のとおりです。

テーブル 19 ・ ISICE

ISICE	プロジェクトの種類	説明
ISICE01	基本の MSI、InstallScript MSI、MSI データベース	デフォルトでアプリケーションが ProgramFiles またはユーザーの AppData フォルダーにインストールされることを検証します。
ISICE02	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	インストールまたはマージ モジュールに含まれているすべての実行可能ファイルにデジタル署名が行われていることを検証します。
ISICE03	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	ネストされたインストール カスタム アクション (タイプ 7、23、および 39) が存在しないことを検証します。

テーブル 19 • ISICE (続き.)

ISICE	プロジェクトの種類	説明
ISICE04	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	標準 .msi テーブルにカスタム列が追加されていないことを検証します。
ISICE05	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	「MSI」で始まるプロパティまたはカスタム テーブルが存在しないことを検証します (大文字と小文字を区別しません)。
ISICE06	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	インストールに Windows Resource Protection で保護されているファイルが含まれていないことを検証します。
ISICE07	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	MSI コンポーネントが適切なインストールおよびアンインストールにオーサされていることを検証します。
ISICE08	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	遅延カスタム アクションが偽装ビット セットを含んでいないことを検証します。
ISICE09	基本の MSI、InstallScript MSI、MSI データベース	インストールに MsiPatchCertificate テーブル エントリが含まれていることを検証します。
ISICE10	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	各カスタム アクションに [カスタム アクション] ビューの [ヘルプ ファイル パス] 設定で指定されたパスが含まれていることを検証します。
ISICE11	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	.exe ファイルがマニフェストを必要な情報と共に埋め込んだことを検証します。

テーブル 19 • ISICE (続き.)

ISICE	プロジェクトの種類	説明
ISICE12	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	保護されたレジストリ キーが変更されていないことを検証します。
ISICE13	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	.hlp ファイルまたは WinHelp ランタイム ファイルがインストールに含められていないことを検証します。この ICE は現在 ISBP17 です。
ISICE14	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	古い形式の API がインポートされていないことを検証します。この ICE は現在 ISBP18 です。
ISICE15	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	使用されていない API がインポートされていないことを検証します。この ICE は現在 ISBP19 です。
ISICE16	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	16 ビット コンポーネントが 64 ビット システムをターゲットにするインストールで配布されていないことを検証します。
ISICE17	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	インストールまたはマージ モジュールが再起動を強制しないことを検証します。
ISICE18	基本の MSI、MSI データベース	MsiRMFilesInUse ダイアログが含まれていることを検証します。
ISICE19	基本の MSI、InstallScript MSI、MSI データベース	アップグレードに関するプロパティ、Upgrade テーブルが存在すること、インストールがダウングレードを防ぐことを検証します。

テーブル 19 ・ ISICE (続き.)

ISICE	プロジェクトの種類	説明
ISICE20	基本の MSI、InstallScript MSI, MSI データベース	マシン全体に渡る設定が権限のないインストールによってインストールされることを検証します。
ISICE21	基本の MSI、InstallScript MSI, MSI データベース	アプリケーションが実行時に VB6 に依存しないことを検証します。
ISICE22	基本の MSI、InstallScript MSI, MSI データベース	アプリケーションがシステム起動時に自動的に開始しないことを検証します。
ISICE23	基本の MSI、InstallScript MSI, MSI データベース	アプリケーションが強く適切な ACL を設定することを検証します。
ISICE24	基本の MSI、InstallScript MSI, MSI データベース	インストールが 16 ビット コンポーネントをインストールしないことを検証します。
ISICE25	基本の MSI、InstallScript MSI, MSI データベース	インストールが Windows ディレクトリまたはサブディレクトリにいずれのファイルもインストールしないことを検証します。
ISICE26	基本の MSI、InstallScript MSI, MSI データベース	アプリケーションが、アクセス許可を適切に制御して、Windows セキュリティ機能をサポートすることを検証します。

***ヒント・** 場合によって、次の検証エラー メッセージが ISICE で表示されることがあります:*

コンポーネント [2] のファイル [1] が見つかりませんでした。このファイルのコンテンツに対するすべてのテストが無効である可能性があります。

指定されたファイルがないとき、また、関連する ISICE がファイルに対して検証できなかったとき、このエラーが発生します。たとえば、実行可能ファイルが不足しているとき、ISICE11 はファイルが埋め込みマニフェストがあるかどうかを検証することができません。

上記の検証エラー メッセージが表示された場合、不足しているファイルについての問題を解決し、プロジェクトの検証を再度実行します。

ISICE01

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

アプリケーションはデフォルトで、ProgramFiles フォルダーまたは AppData フォルダーにインストールされなくてはなりません。現在のデフォルトの場所は [1] です。

[1] が製品のデフォルト インストールの場所です。

説明

ユーザーは必要な場所に製品をインストールすることができ、デフォルトでファイルが保存される場所には一貫性を持たせる必要があります。

ISICE01 は、製品のデフォルト インストール先が Program Files フォルダーまたは Application Data フォルダーであることを検証します。異なる場所が設定されている場合、検証中にこのエラー メッセージが表示されます。¥

修正アクション

デフォルトの場所を変更します。詳細については、「デフォルトの製品インストール先フォルダー (INSTALLDIR) の設定」を参照してください。

ISICE02



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のファイル [1] がデジタル署名されていません。Windows Vista 以降に配布されたすべての [3] ファイルは、署名が必要です。

[1] は実行可能ファイルの名前、[2] はそのファイルを含むコンポーネントの名前、そして、[3] は署名が必要なファイルの種類です。

説明

ISICE02 は、インストールに含まれるすべての実行可能ファイルがデジタル署名されたことを検証します。これには、次の拡張子を持つファイルを含まれます: exe、dll、ocx、sys、cpl、drv、および scr。インストールに含まれる実行可能ファイルにデジタル署名が行われていない場合、検証中にこのエラー メッセージが表示されます。

修正アクション

インストールに含まれるすべての実行可能ファイルがデジタル署名されたことを確認してください。詳細については、「デジタル署名とセキュリティ」を参照してください。

ISICE03



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ (エラー)

ネストされたカスタム アクション [1] (タイプ [2]) は無効です。

[1] はプロジェクトに含まれるカスタム アクションの名前で、[2] はカスタム アクションの Windows Installer タイプ番号です。

説明

ISICE03 は、インストールにネストされたインストール カスタム アクションが含まれていないことを検証します。Windows Installer ではネスト インストールの使用を避けるようにしてください。ネスト インストールが含まれている場合エンドユーザーが適切に処理することが困難であるため、時折アプリケーションの誤作動を引き起こします。マイクロソフト社では一般向けにリリースされる製品のインストールについて、ネスト インストールおよびネスト インストール カスタム アクションの使用を避けることを推奨しています。

詳細については、Windows Installer ヘルプ ライブラリの「Concurrent Installations」を参照してください。

修正アクション

このエラーを解決するためには、ネストされたインストール カスタム アクションをプロジェクトから削除します。[カスタム アクションとシーケンス] ビュー (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または [カスタム アクション] ビュー (マージ モジュール プロジェクトおよび MSM データベース プロジェクトの場合) におけるネスト インストール カスタム アクションの MSI タイプ番号は、7、23、または 39 です。

ネストされたインストールの代替として、InstallShield 前提条件を作成してから、それをインストール プロジェクトに追加するという方法があります。詳細については、「InstallShield 前提条件を定義する」を参照してください。

Summary List of All Custom Action Types (Windows Installer ヘルプ ライブラリ)

ISICE04



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ (エラー)

標準 MSI テーブル [1] が、schema.msi で定義された MSI スキーマと一致しません。

[1] はプロジェクトに含まれるテーブルの名前です。

説明

ISICE04 は、インストールが標準テーブルにカスタム テーブル列を追加しないことを検証します。標準テーブルに列を追加できるのは、Windows Installer の今後のバージョンだけです。

修正アクション

このエラーを解決するには、エラーメッセージで通知されたテーブルに追加されたカスタム テーブル列すべてを削除します。これは、ダイレクト エディターからテーブルをエクスポートしてテキスト エディターでこれを編集してから、変更済みのテーブルをインポートする方法で行います。

データベース テーブル (Windows Installer ヘルプ ライブラリ)

ISICE05



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

MSI プロパティ [1] が予約された文字で始まっています。MSI プロパティ名を「MSI」(大文字と小文字を区別しません) で始めることはできません。

[1] はプロジェクトに含まれるプロパティの名前です。

メッセージ 2(エラー)

テーブル [1] が予約された文字で始まっています。カスタム テーブル名を「MSI」(大文字と小文字を区別しません) で始めることはできません。

[1] はプロジェクトに含まれるテーブルの名前です。

説明

ISICE05 は、インストールに「**MSI**」(大文字と小文字を区別しません) で始まる名前を持つカスタム プロパティまたはカスタム テーブルが含まれていないことを検証します。「**MSI**」で始まる名前は、今後の新しい標準プロパティおよびテーブル用に予約されています。

修正アクション

プロパティ関連のエラーを解決するためには、[プロパティ マネージャー] を使ってエラー メッセージで通知されたカスタム プロパティの名前を変更します。

テーブル関連のエラーを解決するためには、ダイレクト エディター を使ってテーブルをエクスポートし、エクスポートされた .idt ファイルのテーブル名をテキスト エディターを使って編集してから、再びダイレクト エディターを使って更新済みのテーブルをインポートします。その後、元のカスタム テーブルを削除します。

Naming Custom Tables, Properties, and Actions (Windows Installer ヘルプ ライブラリ)

ISICE06



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

コンポーネント [2] のファイル [1] は保護された Windows ファイルです。保護されたファイルを Windows Vista 以降に配布することはできません。

[1] はプロジェクトに含まれるファイル名で、[2] はそのファイルを含むコンポーネントの名前です。

メッセージ 2(警告)

コンポーネント [2] のファイル [1] は、保護された Windows ファイルと同じファイル名前です。たたし、異なるパスへ配布されているように見えます。

[1] はプロジェクトに含まれるファイル名で、[2] はそのファイルを含むコンポーネントの名前です。

説明

ISICE06 は、作成中のインストーラーが Windows Resource Protection (WRP) で保護されているファイルをインストールしないことを検証します。WRP は、Windows Vista 以降のマシン上で保護されたシステム リソースを更新する場合、必ず Microsoft 承認済みのインストールまたはアップデート方法を通してそれが行われるように設計されています。

インストールが保護された Windows ファイルと同じファイル名を持つファイルをインストールするとき、そのファイルがその保護された Windows ファイルと同じ場所にインストールされない場合、警告メッセージ 2 が表示されます。

修正アクション

この検証エラーを解決するには、エラー メッセージで通知されたファイルをプロジェクトから削除します。

検証警告に遭遇したとき、メッセージで通知されたファイルが保護されたファイルかどうかを判別します。

- ファイルが保護されたファイルでなく、他の保護されたファイルと同じ名前を持つファイルである場合、この警告は無視することができます。
- ファイルが保護されたファイルの場合、それをプロジェクトから削除し、この警告を解決します。

製品が新しいバージョンのシステム コンポーネントを必要とする場合、ターゲット マシン上のコンポーネントは Microsoft サービス パックまたは必要なシステム コンポーネントを含む Microsoft 承認済みのインストール パッケージを通して更新されなくてはなりません。システム コンポーネントをリパッケージしてはいけません。

ISICE07



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

コンポーネント [1] の Component.ComponentId がヌルです。正しくインストールおよびアンインストールを行うためには、すべてのコンポーネントに有効な ComponentId が必要です。ヌルのままにする場合、その理由を文書化する必要があります。

[1] は [コンポーネント] ビューの [コンポーネント コード] 設定で指定された GUID を持たない、プロジェクトに含まれるコンポーネントの名前です。

メッセージ 2(エラー)

コンポーネント [1] は COM データを含みますが、複数のファイルがあります。各 COM コンポーネントは、COM ファイルを 1 つだけ含みます。

[1] はプロジェクト内で複数の COM サーバーを含むコンポーネントの名前です。

メッセージ 3(エラー)

コンポーネント [1] には、[デスクトップ] または [スタート] メニュー ショートカットのターゲットとして複数のファイルが含まれています。

[1] は、プロジェクト内でショートカットのターゲットとして指定された複数のファイルを含むコンポーネントの名前です。

メッセージ 4(エラー)

コンポーネント [1] は複数の機能に関連付けられ、次のテーブルで参照されています: [2]。これらの参照には、コンポーネントは 1 つの機能だけに関連付けられている必要があります。

[1] は複数の機能に関連付けられているプロジェクト内のコンポーネントの名前で、Class、Extension、MsiAssembly、PublishComponent、TypeLib テーブルのいずれかによって参照されています。[2] は、このコンポーネントへの参照を含むカンマ区切りのテーブル一覧です。

説明

ISICE07 は、インストールに含まれるコンポーネントがコンポーネント規則に従っていることを検証します。これによって、ある製品のインストーがターゲット システム上にあるその他の製品に支障をきたさないようにします。これらの規則はまた、アンインストールされる製品に関係のあるすべてのリソースが Windows Installer によって正しく確実に削除されるように支援します。

修正アクション

エラー 1 を解決するには、[コンポーネント] ビューを開いてエラーメッセージで通知されたコンポーネントを選択してから、[コンポーネント コード] 設定をクリックします。画面の右下に表示される [ヘルプ] ペインで、[GUID 生成] をクリックします。[コンポーネント コード] 設定を空白にしておかなくてはならない理由がある場合、Windows ロゴ プログラムの申請書にその内容を記述します。

エラー 2 を解決するには、COM サーバーのどれかを移動させてコンポーネントを分離し、各 COM サーバーがそのコンポーネントのキー パスとなるようにします。

エラー 3 を解決するには、[ショートカット] ビューを使って、エラー メッセージで通知されたコンポーネントに関連付けられたショートカットのうち 1 つを残して、それ以外をすべて削除します。

エラー 4 を解決するには、[セットアップのデザイン] ビューまたは [機能] ビューを使用して、関連付けられている機能のうち 1 つを残して、それ以外をすべて削除します。

ISICE08



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ (エラー)

カスタム アクション [1] は偽装を使用します。Windows Vista 以降上でセットアップを実行する場合、偽装を使うことはできません。

[1] はプロジェクトに含まれるカスタム アクションの名前です。

説明

ISICE08 は、インストールに偽装を使うカスタム アクションが含まれていないことを検証します。スクリプト内実行タイプ設定の [ターミナル サーバーで使用可能] オプションのひとつを選択した場合 ([カスタム アクションとシーケンス] ビュー、[カスタム アクション] ビュー、またはカスタム アクション ウィザードの [応答オプション] パネルで設定)、ターミナル サーバー マシン上でマシン毎のインストールを実行中に、カスタム アクションはエンドユーザーを偽装します。

修正アクション

この検証エラーを解決するためには、[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) を開いて、エラー メッセージで通知されたカスタム アクションをクリックします。"スクリプト内実行" 設定の値を "ターミナル サーバーで使用可能" を含まないオプションのひとつに変更します。

ISICE09



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

UAC (ユーザー アカウント制御) パッチには、MsiPatchCertificate テーブル エントリが必要です。

説明

ISICE09 は、インストールに MsiPatchCertificate テーブル エントリが含まれていることを検証します。ベースとなるインストールでは、この テーブルは署名者証明書を提供します。署名者証明書は、後に続くパッチが管理者以外によって適用された場合、またはシステム権限を使用しない管理者によって適用された場合に、デジタル署名を検証するのに使われます。

修正アクション

この検証エラーを解決するためには、プロジェクトに MsiPatchCertificate テーブルを追加します。詳しくは、「非管理者パッチのインストールを準備する」をご覧ください。

ISICE10



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ (エラー)

カスタムアクション [1] (タイプ [2]) について、テーブル [3] に記述されていません。

[1] はプロジェクトに含まれるカスタム アクションの名前で、[2] は Windows Installer タイプ番号です。[3] は、リストされているカスタム アクションのエントリが不足しているテーブルの名前です。

説明

各カスタム アクションの意図された動作は、Windows ロゴ プログラムに基づいて文書化する必要があります。これは、企業環境でシステム管理者が製品を配布する場合、特にカスタム アクションの動作を把握する必要がある場合に便利です。

ISICE10 は、インストールに含まれる各カスタム アクションが文書化されているかどうかについて、CustomAction テーブルの各エントリに対応する ISCustomActionReference テーブル エントリがあることを確認する方法で検証を行います。

修正アクション

この検証エラーを解決するためには、[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) を開いてエラー メッセージで通知されたカスタム アクションを選択してから、"ヘルプ ファイル パス" 設定を使ってカスタム アクションの動作を説明する文書へのパスを指定します。"ヘルプ ファイル パス" 設定の値を指定するとき、ISCustomActionReference テーブルにそのカスタム アクション用の行がない場合、InstallShield がこれを追加します。

また、リリースの製品構成を構成して、ビルド時に InstallShield が、各カスタム アクション ヘルプ ファイルのコンテンツを .msi ファイルにストリームするかどうかを指定することができます。詳細については、[リリース] ビューの [全般] タブ (製品構成) の [カスタム アクションのヘルプを含める] 設定の説明を参照してください。

カスタム アクションがインストール プロジェクトによって消費されるマージ モジュールである場合、マージ モジュール プロジェクトの [カスタム アクション] ビューでパスを指定してから、マージ モジュールを再ビルドします。

ISICE11

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

コンポーネント [2] の EXE [1] にマニフェストがありません。

[1] はマニフェストを持たないプロジェクト内の実行可能ファイルの名前で、[2] は、その実行可能ファイルを含むコンポーネントの名前です。

メッセージ 2(エラー)

コンポーネント [2] の EXE [1] のマニフェストに requestedExecutionLevel level 設定がありません。

[1] は Windows XP 用に作成されたあと、Windows Vista 以降用に更新されていない可能性があるマニフェストを持つプロジェクト内の実行可能ファイルの名前です。マニフェストは、実行可能ファイルの実行レベルを定義しません。[2] は実行可能ファイルを含むコンポーネントの名前です。

メッセージ 3(エラー)

コンポーネント [2] の EXE [1] のマニフェストに requestedExecutionLevel uiAccess 設定がありません。

[1] は Windows XP 用に作成されたあと、Windows Vista 以降用に更新されていない可能性があるマニフェストを持つプロジェクト内の実行可能ファイルの名前です。マニフェストは、実行可能ファイルの UI アクセシビリティ値を定義しません。[2] は実行可能ファイルを含むコンポーネントの名前です。

メッセージ 4(警告)

コンポーネント [2] の EXE [1] は、マイクロソフトからのウェーバーを必要とする昇格された権限 (highestAvailable) が必要であるとマークされています。

[1] は、昇格された権限が必要であると指定するマニフェストを持つプロジェクト内の実行可能ファイルの名前で、[2] は、その実行可能ファイルを含むコンポーネントの名前です。

メッセージ 5(警告)

コンポーネント [2] の EXE [1]は、マイクロソフトからのウェーバーを必要とする昇格された権限 (requireAdministrator) が必要であるとマークされています。

[1] は、昇格された権限が必要であると指定するマニフェストを持つプロジェクト内の実行可能ファイルの名前で、[2] は、その実行可能ファイルを含むコンポーネントの名前です。

メッセージ 6(警告)

コンポーネント [2] の EXE [1] は、マイクロソフトからのウェーバーを必要とする uiAccess が必要であるとマークされています。

[1] は、より上位の権限を使用して情報をデスクトップ上の他のウィンドウ上に渡すために、実行可能ファイルが UI 保護レベルをバイパスできることを指定しているマニフェストを持つプロジェクト内の実行可能ファイルの名前です。このとき、uiAccess の値には True が設定されています。[2] は実行可能ファイルを含むコンポーネントの名前です。

説明

ISICE11 は、インストール内のすべての .exe ファイルに実行レベルを定義する埋め込みマニフェストがあることを検証します。実行レベルは、マニフェストで次のように定義されています:

```
<requestedExecutionLevel
  level="asInvoker"
  uiAccess="false"/>
```

レベル属性の他の有効な値は、highestAvailable と requireAdministrator です。レベルの値が highestAvailable または requireAdministrator に設定した場合、マイクロソフトからのウェーバーを申請して、Windows ロゴ証明書を取得する必要があります。

uiAccess 属性は、より上位の権限を使用して情報をデスクトップ上の他のウィンドウ (例、オンスクリーン キーボード) 上に渡すために、実行可能ファイルが UI 保護レベルをバイパスすることができるかどうか示します。この属性は、UI アクセシビリティ アプリケーションのときのみ True に設定してください。UI アクセシビリティの値を True に設定した場合、マイクロソフトからのウェーバーを申請して、Windows ロゴ証明書を取得する必要があります。

修正アクション

検証エラー (メッセージ1 から 3) を解決するには、インストール内の実行可能ファイルを、レベルの値と uiAccess の値が適切に設定されている埋め込みマニフェストを持つ実行可能ファイルで置き換えます。

検証警告 (メッセージ 4 から 6) は、製品に Windows ロゴ プログラムの証明を受けたいが、requestedExecutionLevel 要素を、現在実行可能ファイルのマニフェストで定義されているように保持したい場合、場合によって、マイクロソフトから要件免除を取得する必要があることを通知します。

ISICE12



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のレジストリ キー [1] は保護されたレジストリ キーです。保護されたレジストリ キーは、Windows Vista 以降上で変更されない可能性もあります。

[1] は Trusted Installer グループのみが Windows Vista 以降のシステム上で変更できるレジストリ キーで、[2] はそのレジストリ キーを含むコンポーネントの名前です。

説明

ISICE12 は、インストールが Windows Vista 以降のシステム上で Windows リソース保護 (WRP) によって保護されているレジストリ キーを変更しようとしないことを検証します。Trusted Installer グループのみが変更できるレジストリ キーは WRP によって保護されています。

修正アクション

この検証エラーを解決するには、プロジェクトから保護されているレジストリ キーを削除するか、またはそのレジストリ キーを WRP で保護されていないレジストリの一部に移動します。

Using Windows Installer and Windows Resource Protection (Windows Installer ヘルプ ライブラリ)

ISICE16



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のファイル [1] は、16 ビット ファイルです。64 ビット システムをターゲットするインストールには、16 ビット ファイルが含まれている可能性があります。

[1] は、プロジェクトに含まれる 16 ビット ファイルの名前で、[2] は、そのファイルを含むコンポーネントの名前です。

説明

インストールが 64 ビット プラットフォームをターゲットにする場合、ISICE16 は、インストールに 16 ビット ファイルが含まれていないことを検証します。64 ビット プラットフォームは 16 ビット ファイルをサポートしません。

修正アクション

この検証エラーを解決するには、この 16 ビット ファイルを 64 ビット または 32 ビット ファイルで置き換えます。

Windows Installer on 64-bit Operating Systems (Windows Installer Help Library)

ISICE17



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

シーケンス [1] に [2] アクションが含まれています。[2] アクションを使用するには、マイクロソフトからのウェーバーが必要です。

[2] はプロジェクトに含まれるアクションの名前で、[1] は、そのアクションを含むシーケンスの名前です。

メッセージ 2(エラー)

コントロール [1] (ダイアログ [2] 内) は [3] アクションを実行します。[3] アクションを使用するには、マイクロソフトからのウェーバーが必要です。

[3] は、プロジェクトに含まれるアクションの名前です。[2] は、アクションを起動するイベントを含むコントロールの名前です。[2] は、コントロールを持つダイアログの名前です。

説明

ISICE17は、インストールがアクションを使用して、再起動を強制しないことを検証します。現在、検証の対象となるアクションは、ForceReboot のみです。

修正アクション

エラーは、製品に Windows ロゴ プログラムの認定を受けたいが、インストールに指定したアクションを保持したい場合、場合によってマイクロソフトから要件免除を取得する必要があることを通知します。

この検証エラーを解決するには、アクションをメッセージで表示されたシーケンスまたはダイアログから削除します。

プロジェクトが、システムの再起動数を最小化する MsiRMFilesInUse ダイアログを含み、アプリケーションが Restart Manager API を使用するように適切にインストルメントされていることを確認してください。詳細については、「Windows Vista 以降のシステムの再起動を最小限にする」を参照してください。

ForceReboot Action (Windows Installer ヘルプ ライブラリ)

ISICE18



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

ファイル [1] が Dialog テーブルにありません。

[1] は、プロジェクトに必要なダイアログの名前です。

メッセージ 2(エラー)

ダイアログ [1] のタイトルに、[2] または [ProductName] が含まれていません。

[1] はプロジェクト内のダイアログ の名前で、[2] は [ProductName] 変数の値です。

説明

ISICE18 は、インストールが MsiRMFilesInUse ダイアログを含み、ダイアログのタイトル バーに製品名が含まれていることを検証します。

修正アクション

このエラーを解決するには、MsiRMFilesInUse ダイアログがプロジェクトにあることを確認します。すべての基本の MSI プロジェクトには、デフォルトでこのダイアログが含まれています。このダイアログの詳細は、「Windows Vista 以降のシステムの再起動を最小限にする」を参照してください。

ダイアログを基本の MSI プロジェクトに再度追加する場合、このダイアログを含む別の基本の MSI プロジェクトを開くか、または新しい基本の MSI プロジェクトを作成します。次いで、ダイアログを、それを必要とするプロジェクトにエクスポートします。詳細については、「他のプロジェクトへダイアログをエクスポートする」を参照してください。

エラー 2 を解決するには、製品名または [ProductName] プロパティを MsiRMFilesInUse ダイアログの [キャプション] プロパティに追加します。

Using Windows Installer with Restart Manager (Windows Installer ヘルプ ライブラリ)

ISICE19



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

テーブル [1] が、インストール パッケージにありません。

[1] は、パッケージに必要なテーブルの名前です。

メッセージ 2(エラー)

UpgradeCode [1] は、適切にスケジュールされたタイプ 19 カスタム アクションを使用してダウングレードの検知および防止を行いません。パッケージはダウングレードをできないようにする必要があります。

[1] は、UpgradeCode プロパティの値です。

メッセージ 3(エラー)

プロパティ [1] が SecureCustomProperties にリストされていません; これが原因で、このプロパティがダウングレードの防止のために正しく使用されなくなる可能性があります。

[1] は、プロパティの名前です。

メッセージ 4(エラー)

プロパティ [1] には、有効な GUID を指定してください。[2] は有効な GUID ではありません。

[1] は、値に有効な GUID が必要なプロパティの名前です。[2] は、現在このプロパティに使用されている無効な値です。

メッセージ 5(エラー)

列 [1] (Upgrade テーブル内) には、有効な GUID が必要です。[2] は有効な GUID ではありません。

[1] は、Upgrade テーブルの列の名前です、[2] は、指定された列のエントリの値です。

メッセージ 6(エラー)

プロパティ [1] には、有効な数値のバージョンをメジャー.マイナー.ビルド というフォーマットで指定してください。[2] は、このフォーマットに沿っていないバージョンです。

[1] は、値にメジャー.マイナー.ビルド というフォーマットのバージョン番号を指定する必要のあるプロパティです。[2] は、現在このプロパティに使用されている無効な値です。

メッセージ 7(エラー)

Upgrade テーブル内の列 [1] には、有効な数値のバージョンをメジャー.マイナー.ビルド というフォーマットで指定してください。[2] は、このフォーマットに沿っていないバージョンです。

[1] は、Upgrade テーブルの列の名前です、[2] は、指定された列のエントリの値です。

メッセージ 8(エラー)

プロパティ [1] に、ヌル値および空白は指定できません。

[1] は、値がヌル値または空白になっているプロパティの名前です。

メッセージ 9(エラー)

プロパティ [1] (Upgrade テーブル内) に、ヌル値および空白は指定できません。

[1] は、Upgrade テーブルの列の名前です。

メッセージ 10(エラー)

プロパティ [1] を指定してください。[2] はデフォルトの値です。

[1] は、カスタマイズされていないプロジェクト内のプロパティの名前です。[2] は、InstallShield がこのプロパティに使用するデフォルト値です。

メッセージ 11(エラー)

プロパティ [1] には、有効な数値の LANGID が必要です; [2] は、数値の LANGID ではありません。

[1] は、値に有効な言語 ID が必要なプロパティの名前です。[2] は、現在このプロパティに使用されている無効な値です。

説明

ISICE19 は、アップグレードに関するプロパティ、および Upgrade テーブルが存在することを検証します。ISICE19 はまた、インストールが以前のパッケージを新しいパッケージで上書きインストールするのを防ぐかどうかを判断するのに役立ちます。

修正アクション

エラー 1 と 2 を解決するには、Upgrades ビューにメジャー アップグレード アイテムがあること、製品の現在のバージョンによって将来のバージョンが上書きインストールされないようにメジャー アップグレード アイテムが適切に構成されていること、および、製品に適切に構成、スケジュールされたタイプ 19 のカスタム アクションが含まれていることを確認してください。詳しい手順については、「現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ」を参照してください。詳細は、Windows Installer ヘルプ ライブラリの「Preventing an Old Package from Installing Over a Newer Version」を参照してください。

エラー 3 を解決するには、指定されたプロパティを SecureCustomProperties プロパティの値に追加します。プロパティの値は、[プロパティ マネージャー] ビューを使用して設定することができます。複数の値を追加するには、それぞれをセミコロンで区切ります。詳しい情報は、Windows Installer ライブラリの「SecureCustomProperties Property」を参照してください。

エラー 4 と 5 を解決するには、既存の値を有効な GUID で置き換えます。値を変更するには、[一般情報] ビューを開きます。次いで、エラー メッセージにリストされている設定をクリックします。有効な GUID を入力します。新しい一意の GUID を使用するには、この設定で **[新しい GUID の生成]** ボタンをクリックします。詳細については、「GUID」を参照してください。

エラー 6 と 7 を解決するには、指定した値を有効なバージョン番号で置き換えます。出力ウィンドウのエラーメッセージをクリックすると、無効な値があるダイレクト エディターの場所に移動することができます。バージョン番号には、**aaa.bbb.ccccc** という形式で、数字のみを利用します。ここで、*aaa* はメジャーバージョン番号、*bbb* はマイナーバージョン番号、そして *ccccc* はビルド番号を表します。*aaa* と *bbb* の最大値は 255 です。*ccccc* の最大値は、65,535 です。詳細については、「製品バージョンを指定する」を参照してください。

エラー 8 を解決するには、指定されたプロパティに値を入力します。プロパティの値は、[プロパティ マネージャー] ビューを使用して設定することができます。

エラー 9 を解決するには、指定された列に値を入力します。ダイレクト エディター ビューを使用して、値を設定することができます。出力ウィンドウのエラー メッセージをクリックすると、修正が必要な値があるダイレクト エディターの場所に移動することができます。各 Windows Installer テーブルについて学ぶには、Windows Installer ヘルプ ライブラリの「Database Tables」を参照してください。

エラー 10 を解決するためには、指定されたプロパティのデフォルト値を適切な値で置き換えます。この値は、プロパティ マネージャーを使用して変更することができます。

エラー 11 を解決するためには、指定されたプロパティの既存の値を有効な言語 ID で置き換えます。

Upgrade Table (Windows Installer ヘルプ ライブラリ)

ISICE20



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

行 [1] (テーブル [2]): ルート [3] のレジストリ キーをインストール、変更、または削除するには、昇格された権限が必要です。ただし、パッケージは昇格された権限を要求しません。

[1] は、プロジェクトのレジストリの変更に関する情報を含む行の名前です。[2] は、レジストリに関するデータを含むテーブルの名前です。[3] は、レジストリ データの定義済みルート キーで、指定された行に Root 列でリストされているものと同じです。

メッセージ 2(エラー)

行 [1] (テーブル [2]): ディレクトリ [3] のファイルをインストール、変更、または削除するには、昇格された権限が必要です。ただし、パッケージは昇格された権限を要求しません。

[1] は、エンドユーザーがシステム権限がないとアクセスできないディレクトリに関する情報を含む行の名前です。[2] は、レジストリ データを含むテーブルの名前です。[3] は、ディレクトリの名前です。

メッセージ 3(エラー)

環境変数 [1] は、システム環境変数ですが、このパッケージは昇格された権限を要求しません。

[1] は、プロジェクトに含まれているシステム環境変数の名前です。

説明

[概要情報] ビューの "管理者権限が必要" 設定がプロジェクトで [いいえ] に設定されている場合、ISICE20 は、インストールが管理者権限を必要とするタスク (システム レジストリまたはフォルダーの場所への書き込みなど) を実行しようとしないことを検証します。

修正アクション

ISICE20 エラーを解決するには、以下のいずれかを実行します。

- [一般情報] ビューの "管理者権限が必要" 設定で [はい] を選択する。詳細については、「概要情報ストリーム データを入力する」を参照してください。
- システムの場所でデータをインストール、変更、または削除しないようにプロジェクトに変更を加える。たとえば、プロジェクトでレジストリ キーが HKEY_LOCAL_MACHINE に追加される場合、[レジストリ] ビューを使用して、そのレジストリ キーを HKEY_CURRENT_USER に移動します。プロジェクトで、システム環境変数が追加または変更される場合、[環境変数] ビューを使用して、環境変数の [種類] 設定を [システム] から [ユーザー] へ変更します。

Authoring Packages without the UAC Dialog Box (Windows Installer ヘルプ ライブラリ)
Installing a Package with Elevated Privileges for a Non-Admin (Windows Installer ヘルプ ライブラリ)
Guidelines for Packages (Windows Installer ヘルプ ライブラリ)

ISICE21



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のファイル [1] は、VB6 ランタイムに依存します。

[1] は、Visual Basic 6 ランタイムを必要とするファイルの名前です。[2] はそのファイルを含むコンポーネントの名前です。

説明

ISICE21 は、製品が Visual Basic 6 ランタイムに依存しないことを検証します。

修正アクション

この検証エラーを解決するために、次のオプションを考慮してください:

- 製品が現在使用していないコンポーネントにファイルが含まれている場合、プロジェクトからそのファイルとコンポーネントを削除します。ファイルの作成には、Visual Basic .NET などの比較的新しい言語の使用を考慮してください。
- プロジェクトからファイルとそのコンポーネントを削除してから、Visual Basic 6 ランタイムに依存しない新しいバージョンのファイルとコンポーネントと置換します。

プロジェクトでこれらのオプションのいずれも不可能な場合、インストールが Windows ロゴ プログラムの認証を受けることはできません。

ISICE22



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

コンポーネント [2] のショートカット [1] がスタートアップ フォルダーをターゲットとします。

[1] はプロジェクトに含まれるショートカット名で、[2] はそのファイルを含むコンポーネントの名前です。

メッセージ 2(エラー)

コンポーネント [2] のレジストリ キー [1] は Run キーを設定します。

[1] はプロジェクトに含まれるレジストリ キーで、[2] はそのショートカットを含むコンポーネントの名前です。

説明

ISICE 22 は、ターゲット システムが開始するときに製品が自動的に開始しないことを検証します。

エラー メッセージ 1 は、インストールが [スタート] メニューのスタートアップ フォルダー内にショートカットを作成する場合に発生します。

エラー メッセージ 2 は、インストールが以下のようなレジストリの場所で Run キーを設定する場合に発生します。

- HKEY_LOCAL_MACHINE¥Software¥Microsoft¥Windows¥CurrentVersion
- HKEY_CURRENT_USER¥Software¥Microsoft¥Windows¥CurrentVersion
- HKEY_LOCAL_MACHINE¥Software¥Wow6432Node¥Microsoft¥Windows¥CurrentVersion
- HKEY_CURRENT_USER¥Software¥Wow6432Node¥Microsoft¥Windows¥CurrentVersion

修正アクション

この検証エラーを解決するには、ショートカットまたはレジストリ キーおよび、適切な場合はコンポーネントをプロジェクトから削除します。

ISICE23



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] の LockObject [1] は、セキュリティで保護されていないアクセス許可を設定します。

[1] は、アクセス許可が構成されるファイルの名前、ディレクトリ、またはレジストリ キーです。[2] はそのファイル、ディレクトリ、またはレジストリ キーを含むコンポーネントの名前です。

説明

ISICE23 は、ファイル、ディレクトリ、およびレジストリ キーに対する不適切なアクセス許可の構成によって、インストールがターゲット システムをセキュリティ上の危険にさらさないことを検証します。

たとえば、インストールによってインストールされる特定のフォルダーについて、すべてのユーザーに対する書き込みアクセス許可が割り当てられる場合、ISICE23 はエラーをトリガして、インストールによってターゲット システムがさらされるセキュリティの脆弱性をアラートします。

修正アクション

この検証エラーを解決するには、エラーに記述されているファイル、ディレクトリ、またはレジストリ キーに強力かつ適切なアクセス許可を使用します。詳細については、次を参照してください。

- ファイルとフォルダーのアクセス許可を構成する
- レジストリ キーのアクセス許可を構成する

ISICE24



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のファイル [1] は、16 ビット ファイルです。Windows 8 システムをターゲットするインストールには、16 ビットファイルを含むことはできません。

[1] は、インストール内の 16 ビット ファイルの名前です。[2] はそのファイルを含むコンポーネントの名前です。

説明

ISICE24 は、インストールに 16 ビット ファイルが含まれていないことを検証します。Windows の 64 ビット バージョンは 16 ビット ファイルをサポートしません。

修正アクション

この検証エラーを解決するには、この 16 ビット ファイルを 64 ビット または 32 ビット ファイルで置き換えます。

ISICE25



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [1] のインストール先は、Windows フォルダーまたはそのサブフォルダーの 1 つです。

[1] は、コンポーネントの名前です。

説明

ISICE25 は、インストールが Windows フォルダーまたはそのサブフォルダーのいずれにも直接書き込まないことを検証します。

修正アクション

この検証エラーを解決するには、次のいずれか 1 つを実行します:

- コンポーネントのインストール先を変更します。
- コンポーネントがフォントをインストールする場合、ベストプラクティスに従ってプロジェクトにフォントファイルが含まれていることを確認してください。詳しくは、「フォントのインストール」をご覧ください。
- コンポーネントがデバイス ドライバーをインストールする場合、ベストプラクティスに従ってプロジェクトにドライバーが含まれていることを確認してください。詳しくは、「デバイスドライバー設定を構成する」をご覧ください。

ISICE26



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のファイル [1] には、オプション ヘッダー [3] が設定されていません。

[1] は、ファイルの名前で、[2] は、そのファイルを含むコンポーネントの名前です。[3] は、ファイルのコンパイル時には使用されなかったパラメーター (SafeSEH、NXCOMPAT、または DYNAMICBASE) です。

説明

ISICE26 は、インストールが Windows セキュリティ機能をサポートすることを検証します。インストールがインストールする実行可能ファイルの PE ヘッダーをチェックして、次のコマンドライン パラメーターを使ってコンパイルされていることを検証します:

- /SafeSEH—安全な例外処理を確認
- /NXCOMPAT—データの実行を防止
- /DYNAMICBASE—ASLR (Address Space Layout Randomization)

単一ファイルで、前述のパラメーターを 1 つずつ複数の ISICE26 エラーをトリガすることが可能です。

修正アクション

この検証エラーを解決するには、製品のソースコードのビルド オプションを確認して、必要に応じて編集します。Visual Studio の場合、これらは "リンカ" 設定の [詳細プロパティ] ページで構成されます。

InstallShield 仮想化整合性スイート



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

InstallShield には、InstallShield 仮想化適合性スイートという名前の一連の検証ツールが含まれています。このスイートに含まれる InstallShield 仮想化適合性検証ツール (ISVICE) は、様々な仮想化テクノロジについてプロジェクトの適合性をチェックします。

InstallShield で使用できる ISVICE のリストは次の表のとおりです。

テーブル 20 · ISVICE

ISICE	プロジェクトの種類	テクノロジ	説明
ISVICE01	基本の MSI、InstallScript MSI、MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	通常、Windows オペレーティング システムの一部であるファイルの存在を確認します。
ISVICE02	基本の MSI、InstallScript MSI、MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	デバイス ドライバー ファイルの存在を確認します。

テーブル 20 ・ （続き.)ISVICE

ISICE	プロジェクトの種類	テクノロジ	説明
ISVICE03	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	ClickOnce 配布ファイルの存在を確認します。
ISVICE04	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、ThinApp、XenApp	ASP.NET/IIS アプリケーションの一部であるファイルの存在を確認します。
ISVICE05	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、ThinApp、XenApp	WMI プロバイダーの一部であるファイルの存在を確認します。
ISVICE06	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	J2EE アプリケーション サーバーの一部であるファイルの存在を確認します。
ISVICE07	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	仮想化を行うと動作しない可能性のあるアプリケーションの一部であるファイルの存在を確認します。
ISVICE08	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	仮想化を行うと失敗することが報告されているアプリケーションの一部であるファイルの存在を確認します。
ISVICE09	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、ThinApp、XenApp	ASP.NET/IIS アプリケーションがインストール済みであることを示す InstallShield IIS サポートの存在をチェックします。
ISVICE10	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	MsiDriverPackages テーブルを使ってインストールされたドライバーの存在を確認します。

テーブル 20 ・ （続き.)ISVICE

ISICE	プロジェクトの種類	テクノロジ	説明
ISVICE11	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、ThinApp、XenApp	パッケージに少なくとも 1 つのショートカットが含まれていることを確認します。
ISVICE12	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	シェル拡張の存在を確認します。
ISVICE13	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、Server App-V、ThinApp、XenApp	URL プロトコル ハンドラーの存在を確認します。
ISVICE14	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、Server App-V、ThinApp、XenApp	デフォルト プログラム リストのサポートの存在を確認します。
ISVICE15	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、ThinApp、XenApp	ブート スタート サービスの存在を確認します。
ISVICE16	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、Server App-V、XenApp	インストールされたファイルの合計サイズが 4GB 未満であることを確認します。
ISVICE17	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、ThinApp、XenApp	Windows Installer COM+ テーブルの使用状況を確認します。
ISVICE18	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、ThinApp、XenApp	COM DLL サロゲート ファイルの存在を確認します。
ISVICE19	基本の MSI、InstallScript MSI, MSI データベース	ThinApp、XenApp	64 ビット パッケージであるかどうかを確認します。ThinApp および XenApp は 64 ビット アプリケーションをサポートしません。

ISVICE01



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE01 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- Server App-V
- ThinApp
- XenApp

メッセージ (エラー)

ファイル [1]が OS と厳密に統合されたファイルのようです。Internet Explorer、Windows Media Player、または .NET Framework のフレームワークなどのアプリケーションの一部であるファイルは、仮想化には向いていません。これらのファイルは仮想ではなく、ローカル マシンにインストールしてください。)

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE01 は、Windows オペレーティング システムの一部であるファイルの存在を確認します。リリースにこの種類のファイルが 1 つ以上含まれる場合、検証中にこのエラーが表示されます。

修正アクション

エンド ユーザーが製品の仮想バージョンを希望していて、作業中の InstallShield プロジェクトに .NET Framework その他の Windows と密接に統合されているアイテムが含まれている場合、プロジェクトからこれらのアイテムを削除することが推奨されます。エンド ユーザーが仮想バージョンの製品を使用する前に、ローカル マシンにこれらのプロジェクトに必要なテクノロジをインストールする手順を説明するドキュメントの提供を考慮してください。

ISVICE02



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE02 は、次の仮想化テクノロジに適用します:

- App-V 4.x

- App-V 5.x
- Server App-V
- ThinApp
- XenApp

メッセージ (エラー)

ファイル [1] がデバイス ドライバーのようです。プリンター ドライバーや USB デバイス ドライバーなど、システム レベルのドライバーは、仮想環境からは使用できません。このドライバーを抽出して、ローカル マシンにインストールし、パッケージの残りの部分を仮想化することができる場合があります。)

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE02 は、デバイス ドライバー ファイルの存在を確認します。プリンター ドライバーや USB デバイス ドライバーなど、システム レベルのドライバーは、仮想環境からは使用できません。

修正アクション

エンド ユーザーが製品の仮想バージョンnを希望していて、InstallShield プロジェクトにデバイス ドライバーが含まれている場合、プロジェクトからデバイス ドライバーを削除して、ドライバー用の個別のインストールを作成することを考慮してください。エンド ユーザーは製品の仮想バージョンを使用する前に、ローカル マシンにそのドライバーをインストールできます。

ISVICE03



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE03 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- App-V 5.x
- Server App-V
- ThinApp
- XenApp

メッセージ (エラー)

ファイル [1] が ClickOnce 配置ファイルのようです。ClickOnce はユーザーごとのインストール フォーマットなので、マシンごとの特性を持つ仮想パッケージとは互換性を持たない場合があります。また、ClickOnce アプリケーションは自動的にアップデートを行うため、仮想アプリケーション管理システムでは、無効なバージョン エラーの原因となる可能性があります。)

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE03 は、ClickOnce 配置ファイルの存在を確認します。プロジェクトに ClickOnce 配置 ファイルが含まれているとき、ISVICE03 が発生します。

修正アクション

ClickOnce 配置ファイルがアプリケーションにおける重要性を検討して、ClickOnce インストールが予定通りに動作することがどれほど重要であるのかを判断してください。ClickOnce 配置 ファイルがそれほど重要でない場合、製品の仮想バージョンは少しだけ機能を限定して配布することが安全策と言えます。ClickOnce 配置ファイルが重要な場合、製品の仮想バージョンが正しく機能しない場合があります。

ISVICE04



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE04 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- ThinApp
- XenApp

メッセージ (エラー)

指定されたファイル [1] が ASP.NET/IIS アプリケーションの一部のようです。デスクトップ アプリケーションの仮想化で、これはサポートされていません。)

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE04 は、ASP.NET/IIS アプリケーションの一部であるファイルの存在を確認します。

修正アクション

ASP.NET または IIS アプリケーションが製品の重要な部分を構成しない限り、またはターゲット システム上で個別にローカルでインストールすることが可能な場合、この検証エラーを無視することができます。

ISVICE05



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE05 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- ThinApp
- XenApp

メッセージ (エラー)

ファイル [1] が WMI プロバイダーのようです。デスクトップ アプリケーションの仮想化で、これはサポートされていません。)プロバイダーが、パッケージの重要な部分を構成しない限り、パッケージの残りの部分を仮想アプリケーションとして使用することが可能な場合があります。)

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE05 は、WMI (Windows Management Instrumentation) プロバイダーの一部であるファイルの存在を確認します。

修正アクション

WMI プロバイダーが製品の重要な部分を構成しない限り、またはターゲット システム上で個別にローカルでインストールすることが可能な場合、この検証エラーを無視することができます。

ISVICE06



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE06 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- Server App-V
- ThinApp

- XenApp

メッセージ (エラー)

ファイル [1] が J2EE アプリケーション サーバーの一部のようです。デスクトップ アプリケーションの仮想化で、これはサポートされていません。)

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE06 は、J2EE アプリケーション サーバーの一部であるファイルの存在を確認します。

修正アクション

J2EE アプリケーションが製品の重要な部分を構成しない限り、またはターゲット システム上で個別にローカルでインストールすることが可能な場合、この検証エラーを無視することができます。

ISVICE07



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE07 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- Server App-V
- ThinApp
- XenApp

メッセージ (警告)

ファイル [1] は、仮想化には不向きであるとされているアプリケーションの一部のようです。ただし、適切ではない部分をローカルにインストールして、パッケージの残りの部分だけを仮想化することが可能です。

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE07 は、仮想化を行うと動作しない可能性のあるアプリケーションの一部であるファイルの存在を確認します。これには、SQL Server などのアプリケーションからのファイルが含まれます。

修正アクション

サポートされていないアプリケーションが製品の重要な部分を構成しない限り、またはターゲット システム上で個別にローカルでインストールすることが可能な場合、この検証エラーを無視することができます。

ISVICE08



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE08 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- Server App-V
- ThinApp
- XenApp

メッセージ (エラー)

ファイル [1] は、仮想化には不向きであるとされているアプリケーションの一部のようです。

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE08 は、仮想化を行うと動作しない可能性のあるアプリケーションの一部であるファイルの存在を確認します。これには、ウィルス対策ソフトウェアや Exchange Server などのサーバー ソフトウェアからのファイルが含まれます。

修正アクション

サポートされていないアプリケーションが製品の重要な部分を構成しない限り、またはターゲット システム上で個別にローカルでインストールすることが可能な場合、この検証エラーを無視することができます。

ISVICE09



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE09 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- ThinApp

- XenApp

メッセージ (エラー)

IIS [1] アイテム [2] がパッケージに含まれています。デスクトップ アプリケーションの仮想化で、これはサポートされていません。)

[1] はプロジェクトに含まれる IIS アイテムの種類の名前で、[2] はそのアイテムの名前です。

説明

ISVICE09 は ASP.NET/IIS アプリケーションがインストール済みであることを示す InstallShield IIS サポートの存在をチェックします。

修正アクション

サポートされていない ASP.NET または IIS アプリケーションが製品の重要な部分を構成しない限り、またはターゲット システム上で個別にローカルでインストールすることが可能な場合、この検証エラーを無視することができます。

ISVICE10



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE10 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- Server App-V
- ThinApp
- XenApp

メッセージ (エラー)

コンポーネント [1] には、MsiDriverPackages (DIFxApp) テーブルを使ってインストールされたデバイス ドライバーが含まれています。プリンター ドライバーや USB デバイス ドライバーなど、システム レベルのドライバーは、仮想環境からは使用できません。このドライバーを抽出して、ローカル マシンにインストールし、パッケージの残りの部分を仮想化することができる場合があります。)

[1] は、プロジェクトに含まれているデバイス ドライバーの名前です。

説明

ISVICE10 は、MsiDriverPackages テーブルを通してインストールするように構成されたドライバーの存在を確認します。

修正アクション

エンド ユーザーが製品の仮想バージョンを希望していて、InstallShield プロジェクトにドライバーが含まれている場合、プロジェクトからドライバーを削除して、ドライバー用の個別のインストールを作成することを考慮してください。エンド ユーザーは製品の仮想バージョンを使用する前に、ローカル マシンにそのドライバーをインストールできます。

ISVICE11



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE11 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- App-V 5.x
- ThinApp
- XenApp

メッセージ (App-V 4.x and App-V 5.x の警告。ThinApp および XenApp のエラー)

このパッケージには、ショートカットが含まれていません。仮想アプリケーションへのエントリ ポイントを定義するために、通常、ショートカットが必要です。

説明

ISVICE11 パッケージに少なくとも 1 つのショートカットが含まれていることを確認します。ショートカットは、仮想パッケージ内のアプリケーションを起動するときに最も見つけやすいエントリ ポイントです。ほとんどの仮想パッケージには、1 つ以上のショートカットが含まれています。

修正アクション

パッケージが別の仮想パッケージの依存関係である場合、この ISVICE は無視できます。

パッケージがプラグインとして使用される場合、プラグインの対象となるアプリケーションへのショートカットを作成する必要があるかもしれません。

エンド ユーザーが仮想アプリケーションを個別に起動できるようにする場合、メインの実行可能ファイルへのショートカットの追加を考慮してください。

ISVICE12



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript MSI*
- *MSI データベース*

ISVICE12 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- App-V 5.x
- Server App-V
- ThinApp
- XenApp

メッセージ (App-V 4.x、App-V 5.x、ThinApp、および XenApp のエラー。Server App-V の警告。)

このパッケージには、シェル拡張 (キー [1]) が含まれています。シェル拡張は、Windows Explorer を拡張するため、仮想パッケージからロードすることはできません。この拡張がアプリケーションにとって大変重要な場合、仮想化した後にアプリケーションが機能しないことがあります。ただし、拡張が重要ではない場合、アプリケーションを使用できることもあります。

[1] は、プロジェクトに含まれているシェル拡張のキーです。

説明

ISVICE12 は、シェル拡張の存在を確認します。

修正アクション

シェル拡張が製品における重要性を検討して、シェル拡張が予定通りに動作することがどれほど重要であるのかを判断してください。シェル拡張がそれほど重要でない場合、製品の仮想バージョンは少しだけ機能を限定して配布しても恐らく問題はないと言えます。シェル拡張が重要な場合、製品の仮想バージョンが正しく機能しない場合があります。

ISVICE13



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE13 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- Server App-V
- ThinApp
- XenApp

メッセージ (警告)

このパッケージは URL プロトコル (フレンドリ名 [1]) を登録します。

[1] は、URL プロトコルの表示名です。

説明

ISVICE13 は、URL プロトコル ハンドラーの存在を確認します。

修正アクション

URL プロトコルの登録が製品における重要性を検討して、製品が予定通りに動作することがどれほど重要であるのかを判断してください。URL プロトコルの登録がそれほど重要でない場合、製品の仮想バージョンは少しだけ機能を限定して配布しても恐らく問題はないと言えます。URL プロトコルの登録が重要な場合、製品の仮想バージョンが正しく機能しない場合があります。

製品に URL プロトコル ハンドラーが含まれているため、App-V 5.x が製品に適した仮想化テクノロジであるかどうかを検証してください。

ISVICE14



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE14 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- Server App-V
- ThinApp
- XenApp

メッセージ (警告)

This package registers its capabilities in the Default Programs list (key '%1').

[1] はデフォルト プログラム リストに含まれるアイテムの登録情報に対応するキーの名前です。

説明

ISVICE14 は、デフォルト プログラム リストのサポートの存在を確認します。

修正アクション

デフォルト プログラム リストの登録が、製品で重要ではない場合、この検証エラーを無視することができます。

製品に デフォルト プログラム リストのアイテムを登録するサポートが含まれているため、App-V 5.x が製品に適した仮想化テクノロジであるかどうかを検証してください。

ISVICE15



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE15 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- App-V 5.x
- ThinApp
- XenApp

メッセージ (エラー)

このパッケージには、起動時に開始するサービス (名前 [1]) が含まれています。仮想サービスは、仮想アプリケーションが有効である間だけに限られているため、起動時に開始しなくてはならないサービスは、仮想化には向いていません。このサービスを抽出して、ローカル マシンにインストールすることで、パッケージの残りの部分を仮想化することが可能な場合があります。)

[1] はプロジェクトに含まれるサービスの名前です。

説明

ISVICE15 は、ブート スタート サービスの存在を確認します。

修正アクション

サービスが製品における重要性を検討して、製品が予定通りに動作することがどれほど重要であるのかを判断してください。サービスがそれほど重要でない場合、製品の仮想バージョンは少しだけ機能を限定して配布しても恐らく問題はないと言えます。

サービスが重要であるが、それを残りのプロジェクトから切り離すことができる場合、プロジェクトから削除して、サービス用の個別のインストールを作成してください。エンド ユーザーは、製品の仮想バージョンを使用する前に、ローカル マシンにサービスをインストールできます。

サービスが重要な場合で、プロジェクトから切り離すことができないとき、製品の仮想バージョンは正しく機能しない場合があります。

ISVICE16



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE16 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- Server App-V
- XenApp

メッセージ (エラー)

This package contains more than 4 GB of files.Since the target virtualization technology has a 4 GB size limit, this application cannot be successfully virtualized.(このパッケージには、4 GB 以上のファイルが含まれています。ターゲットの仮想化テクノロジには 4 GB のサイズ制限があるため、このアプリケーションを正しく仮想化することができません。)

説明

ISVICE16 は、インストールされたファイルの合計サイズが 4GB 未満であることを確認します。App-V 4.x、Server App-V、および XenApp は、4 GB 以上のファイルを含むパッケージをサポートしません。

修正アクション

App-V 5.x または ThinApp は 4 GB 以上のサイズをサポートするので、App-V 5.x または ThinApp が製品に適した仮想化テクノロジであるかどうかを検証してください。

ISVICE17



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE17 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- App-V 5.x
- ThinApp
- XenApp

メッセージ (エラー)

このパッケージには COM+ データ (コンポーネント [1]) が含まれているので、アプリケーションを仮想化した場合に正しく機能しない可能性があります。

[1] はプロジェクトに含まれるコンポーネントの名前です。コンポーネントには COM+ データが含まれています。

説明

ISVICE17 は、Windows Installer COM+ テーブルの使用状況を確認します。

修正アクション

COM+ アプリケーションが製品における重要性を検討して、製品が予定通りに動作することがどれほど重要であるのかを判断してください。COM+ アプリケーションがそれほど重要でない場合、製品の仮想バージョンは少しだけ機能を限定して配布しても恐らく問題はないと言えます。COM+ アプリケーションが重要な場合、製品の仮想バージョンが正しく機能しない場合があります。

ISVICE18



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE18 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- ThinApp
- XenApp

メッセージ (エラー)

このパッケージには、サポートされていない COM DLL サロゲート (レジストリ キー [1]) が含まれているので、アプリケーションを仮想化した場合に正しく機能しない可能性があります。

[1] はプロジェクトに含まれるレジストリ キーの名前です。

説明

ISVICE18 は、COM DLL サロゲート ファイルの存在を確認します。

修正アクション

COM DLL サロゲートが製品における重要性を検討して、製品が予定通りに動作することがどれほど重要であるのかを判断してください。DLL サロゲートがそれほど重要でない場合、製品の仮想バージョンは少しだけ機能を限定して配布しても恐らく問題はないと言えます。DLL サロゲートが重要な場合、製品の仮想バージョンが正しく機能しない場合があります。

ISVICE19



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE19 は、次の仮想化テクノロジに適用します:

- ThinApp
- XenApp

メッセージ (エラー)

The package is 64-bit.ThinApp and XenApp do not support 64-bit applications.(パッケージが 64 ビットです。ThinApp および XenApp は 64 ビット アプリケーションをサポートしません。

説明

ISVICE19 は、64 ビット パッケージをビルドしているかどうかを確認します。ThinApp および XenApp は 64 ビット アプリケーションをサポートしません。

修正アクション

製品に 64 ビット サポートが含まれているため、App-V または Server App-V が製品に適した仮想化テクノロジであるかどうかを検証してください。

InstallShield MSIX/UWP アプリ適合性スイート



プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield には、InstallShield MSIX/UWP アプリ適合性スイートという名前の一連の検証ツールが含まれています。このスイートに含まれる InstallShield Windows アプリ適合性検証ツール (ISUWP) は、.msi パッケージ内で Windows アプリ パッケージ (.appx) フォーマットに適さないアイテムの存在をスキャンします。このスイートにアクセスするには、**[ビルド]** メニューから **[検証]** をポイントしてから、**[InstallShield MSIX/UWP アプリ適合性スイート]** をクリックします。スイートは、問題が見つかったすべてのテストを [リリース] ビューに表示し、レポート内の各問題に関連付けられた列に既知の Windows アプリ バリエーション (ユニバーサル アプリ、デスクトップ ブリッジ、Windows ストア、WSA、および Nato Server) への適合性が表示されます。従来型の CUB の場合、これらの列は空白のままです。InstallShield で利用できる ISUWP 検証ツールは次の通りです:

テーブル 21 ・ ISUWP

ISUWP	プロジェクトの種類	テクノロジ	説明
ISUWP01	基本の MSI	Windows アプリ	Windows サービスの存在を確認します。
ISUWP02	基本の MSI	Windows アプリ	MsiDriverPackages テーブルを使ってインストールされたドライバーの存在を確認します。
ISUWP03	基本の MSI	Windows アプリ	ショートカットの存在を確認します。InstallShield では、アプリケーションを識別するためにショートカットが必要です。アプリケーションを識別するショートカット無しでは、アプリを開始する方法が無いためパッケージが無効となります。

テーブル 21 • ISUWP

ISUWP	プロジェクトの種類	テクノロジ	説明
ISUWP04	基本の MSI	Windows アプリ	Windows ストア アプリケーションの複数のショートカットの存在を確認します。InstallShield はショートカットをアプリケーションに変換し、Windows ストアでは複数アプリケーションを使用できない場合があります。
ISUWP05	基本の MSI	Windows アプリ	Windows Nano Server アプリケーションのショートカットの存在を確認します。InstallShield はショートカットがパッケージによってインストールされた .exe をポイントする場合、そのショートカットをタイルに変換しますが、Windows Nano Server はアプリケーション タイルを表示できません。
ISUWP06	基本の MSI	Windows アプリ	16 ビット ファイルの存在を確認します。すべてのサーバー インストールは 64 ビット プラットフォームで、64 ビット プラットフォームは 16 ビット ファイルをサポートしません。
ISUWP07	基本の MSI	Windows アプリ	32 ビット ファイルの存在を確認します。すべてのサーバー インストールは 64 ビット プラットフォームで、64 ビット プラットフォームは 32 ビット ファイルをサポートしません。
ISUWP08	基本の MSI	Windows アプリ	昇格された権限を必要とするアプリケーションの存在を確認します。
ISUWP09	基本の MSI	Windows アプリ	UI 保護レベルをバイパスできるアプリケーションの存在をチェックします。
ISUWP10	基本の MSI	Windows アプリ	COM 登録の存在を確認します。
ISUWP11	基本の MSI	Windows アプリ	カスタム アプリケーション ユーザー モデル ID を設定する実行可能ファイルの存在を確認します。
ISUWP12	基本の MSI	Windows アプリ	Windows Nano Server と互換性を持たない .NET アセンブリの存在を確認します。
ISUWP13	基本の MSI	Windows アプリ	.NET Framework の最新版をサポートしない .NET アセンブリの存在を確認します。
ISUWP14	基本の MSI	Windows アプリ	ASP.NET/IIS アプリケーションの一部であるファイルの存在を確認します。

テーブル 21 • ISUWP

ISUWP	プロジェクトの種類	テクノロジ	説明
ISUWP15	基本の MSI	Windows アプリ	ASP.NET/IIS アプリケーションがインストール済みであることを示す InstallShield IIS サポートの存在を確認します。
ISUWP16	基本の MSI	Windows アプリ	SQL スクリプトを実行するビルトイン InstallShield サポートの存在を確認します。
ISUWP17	基本の MSI	Windows アプリ	テキスト ファイルを変更するビルトイン InstallShield サポートの存在を確認します。
ISUWP18	基本の MSI	Windows アプリ	XML ファイルを変更するビルトイン InstallShield サポートの存在を確認します。
ISUWP19	基本の MSI	Windows アプリ	スケジュール タスクを作成するビルトイン InstallShield サポートの存在を確認します。
ISUWP20	基本の MSI	Windows アプリ	予約済み、または禁止されているファイル タイプの関連付けの存在を確認します。

ISUWP01

ISUWP01 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

このパッケージには 1 つ以上の Windows サービスが含まれています。Windows Server 以外のデバイスをターゲットとする Windows アプリ パッケージは、Windows サービスをサポートしません。

説明

ISUWP01 は Windows サービスの存在を確認します。

修正アクション

アプリケーションが 1 つ以上の Windows サービスを必要とする場合、サーバー拡張および WSA が必要です。デスクトップ拡張を含めることは、このシナリオではサポートされていません。

ISUWP02

ISUWP02 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

コンポーネント [1] には、MsiDriverPackages (DIFxApp) テーブルを使ってインストールされたデバイス ドライバーが含まれています。Windows アプリ パッケージは、デバイス ドライバーをサポートしません。

[1] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISUWP02 は、MsiDriverPackages テーブルを使ってインストールされたドライバーの存在を確認します。

修正アクション

ドライバーが必要な場合、このアプリケーションは Windows アプリだけではサポートできません。

ISUWP03

ISUWP03 は、Windows アプリのシナリオに適用します。

メッセージ (警告)

このパッケージには、ショートカットが含まれていません。アプリケーションへのエントリ ポイントを定義するために、通常、ショートカットが必要です。

説明

ISUWP03 は、ショートカットの存在を確認します。InstallShield では、アプリケーションを識別するためにショートカットが必要です。アプリケーションを識別するショートカット無しでは、アプリを開始する方法が無いためパッケージが無効となります。

ISUWP04

ISUWP04 は、Windows アプリのシナリオに適用します。

メッセージ (警告)

このパッケージには 1 つ以上のショートカットが含まれています。Windows ストアでは、複数のアプリケーションを使用できない場合があります。

説明

ISUWP04 は、Windows ストア アプリケーションの複数のショートカットの存在を確認します。InstallShield はショートカットをアプリケーションに変換し、Windows ストアでは複数アプリケーションを使用できない場合があります。

修正アクション

このアプリケーションを Windows ストアに配置したい場合、アプリケーションを複数の Windows パッケージに分割しなくてはならない場合があります。

ISUWP05

ISUWP05 は、Windows アプリのシナリオに適用します。

メッセージ (警告)

このパッケージには 1 つ以上のショートカットが含まれています。Windows Nano Server はアプリケーション タイルを表示できません。

説明

ISUWP05 は、Windows Nano Server アプリケーションのショートカットの存在を確認します。InstallShield はショートカットがパッケージによってインストールされた .exe をポイントする場合、そのショートカットをタイルに変換しますが、Windows Nano Server はアプリケーション タイルを表示できません。

修正アクション

この警告は、予定どおりに動作しない可能性のある既知のアイテムのアラートを行います。

ISUWP06

ISUWP06 は、Windows アプリのシナリオに適用します。

メッセージ (警告)

コンポーネント [2] のファイル [1] は、16 ビット ファイルです。64 ビット システムは 16 ビット アプリケーションをサポートしません。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISUWP06 は、16 ビット ファイルの存在を確認します。すべてのサーバー インストールは 64 ビット プラットフォームで、64 ビット プラットフォームは 16 ビット ファイルをサポートしません。

修正アクション

修正アクションは、実行時にファイルがロードおよび実行する場合のみ必要です。16 ビットまたは 32 ビットのデータ ファイルは問題ありません。フラグされた 16 ビット ファイルが実行時にロードおよび実行する場合、これを 64 ビット ファイルに置換します。

ISUWP07

ISUWP07 は、Windows アプリのシナリオに適用します。

メッセージ (警告)

コンポーネント [2] のファイル [1] は、32 ビット ファイルです。64 ビット システムは 32 ビット アプリケーションをサポートしない場合があります。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISUWP07 は、32 ビット ファイルの存在を確認します。すべてのサーバー インストールは 64 ビット プラットフォームで、64 ビット プラットフォームは 32 ビット ファイルをサポートしません。

修正アクション

修正アクションは、実行時にファイルがロードおよび実行する場合のみ必要です。16 ビットまたは 32 ビットのデータ ファイルは問題ありません。フラグされた 32 ビット ファイルが実行時にロードおよび実行する場合、これを 64 ビット ファイルに置換します。

ISUWP08

ISUWP08 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

コンポーネント [2] の [1] ファイルは、昇格された権限を必要とするアプリケーション実行可能ファイルです。Windows ストアでは、管理者権限を必要とするアプリケーションを使用できません。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISUWP08 は、昇格された権限を必要とするアプリケーションの存在を確認します。

修正アクション

昇格された権限の要件を削除します。アプリは昇格された権限を使って、ファイルまたはレジストリ キーの処理におけるベスト プラクティスを妨害する場合がありますが、そのような動作は Windows アプリでは使用できません。

ISUWP09

ISUWP09 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

コンポーネント [2] の [1] ファイルは、デスクトップ上でより高度な権限を要するウィンドウへ打ち込むため、UI 保護レベルのバイパスが許可されているアプリケーション実行可能ファイルです。Windows アプリ パッケージでは、UI 保護レベルのバイパスが許可されているアプリケーションはサポートされていません。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISUWP09 は、UI 保護レベルをバイパスできるアプリケーションの存在をチェックします。

修正アクション

UI 保護レベルのバイパスを必要とするアプリケーションはほとんどありませんが、それが必要な場合、現在 Windows アプリではサポートされていません。アプリケーションが UI 保護レベルをバイパスする必要がある場合、Windows アプリ パッケージ以外の異なるパッケージの使用を考慮してください。

ISUWP10

ISUWP10 は、Windows アプリのシナリオに適用します。

メッセージ (警告)

メッセージ 1(警告)

テーブル [1] には、1 つ以上のエントリが含まれています。Windows アプリ パッケージに含まれるアプリケーションは、COM サーバーを別のアプリケーションに公開することができません。

[1] はプロジェクトに含まれるテーブルです。

メッセージ 2(警告)

コンポーネント [2] のレジストリ キー [1] はCOM データです。Windows アプリ パッケージに含まれるアプリケーションは、COM サーバーを別のアプリケーションに公開することができません。

[1] はプロジェクトに含まれるレジストリ キーの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISUWP10 COM 登録の存在を確認します。

修正アクション

COM 登録が問題の場合、またはそうでない場合があります。COM の内部的な使用はデスクトップ ブリッジでサポートされていますが、(他のアプリによる使用に向けて) パブリック登録をサポートする拡張子は現在サポートされていません。後者の場合、Windows アプリ パッケージは適していません。

ISUWP11

ISUWP11 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

コンポーネント [2] にあるファイル [1] は、非推奨の API [3] をインポートします。Windows アプリ パッケージに含まれるアプリケーションは、カスタム アプリケーション ユーザー モデル ID を設定することができません。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

[3] は、API の名前です。

説明

ISUWP11 カスタム アプリケーション ユーザー モデル ID を設定する実行可能ファイルの存在を確認します。

修正アクション

問題のある API 呼び出しのコードベースを削除する必要があります。Windows アプリはアプリケーション ユーザー モデル ID を設定します。

ISUWP12

ISUWP12 は、Windows アプリのシナリオに適用します。

メッセージ

メッセージ 1(エラー)

コンポーネント [2] のファイル[1] は、.NET Framework に対してコンパイルされた .NET アセンブリです。Windows Nano Server .NET コア アプリケーションのみをサポートします。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

メッセージ 2(エラー)

コンポーネント [2] のファイル [1] は、[3] に対してコンパイルされた .NET アセンブリです。Windows Nano Server .NET コア アプリケーションのみをサポートします。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

[3] はプロジェクトに含まれる Framework の名前です。

説明

ISUWP12 Windows Nano Server と互換性を持たない .NET アセンブリの存在を確認します。

修正アクション

Nano Server をサポートする場合、.NET コアに対してアプリを再ビルドします。

ISUWP13

ISUWP13 は、Windows アプリのシナリオに適用します。

メッセージ (警告)

コンポーネント [2] のファイル[1] は、.NET Framework の最新版をサポートしない .NET アセンブリです。Windows ストア アプリケーションは、常に .NET Framework の最新バージョン上で実行します。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISUWP13 .NET Framework の最新版をサポートしない .NET アセンブリの存在を確認します。

修正アクション

App.config を再ビルドまたは調整します。

ISUWP14

ISUWP14 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

指定されたファイル [1] が ASP.NET/IIS アプリケーションの一部のようです。これは Windows アプリに含まれるアプリケーションではサポートされていません。

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISUWP14 は、ASP.NET/IIS アプリケーションの一部であるファイルの存在を確認します。

修正アクション

ASP.NET/IIS アプリケーションの一部であるファイルは Windows アプリではサポートされていないため、パッケージからこのファイルを削除するか、代替のパッケージ フォーマットを使用する必要があります。

ISUWP15

ISUWP15 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

IIS [1] アイテム [2] がパッケージに含まれています。これは Windows アプリに含まれるアプリケーションではサポートされていません。

[1] はプロジェクトに含まれる IIS アイテムの種類の名前で、[2] はそのアイテムの名前です。

説明

ISUWP15 は ASP.NET/IIS アプリケーションがインストール済みであることを示す InstallShield IIS サポートの存在を確認します。

修正アクション

ASP.NET/IIS アプリケーションの一部であるファイルは Windows アプリではサポートされていないため、パッケージからこのファイルを削除するか、代替のパッケージ フォーマットを使用する必要があります。

ISUWP16

ISUWP16 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

このパッケージには、インストール中に実行する 1 つ以上の SQL スクリプトが含まれています。Windows アプリ パッケージは、SQL スクリプトの実行をサポートしません。

説明

ISUWP16 は、SQL スクリプトを実行するビルトイン InstallShield サポートの存在を確認します。

修正アクション

Windows アプリ パッケージの使用をサポートするためには、フラグされた構成を、インストールからアプリケーションに移動する必要があります。

ISUWP17

ISUWP17 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

このパッケージには、インストール中に実行される 1 つ以上の InstallShield テキスト ファイルの変更アイテムが含まれています。Windows アプリ パッケージは、テキスト ファイルの変更をサポートしません。

説明

ISUWP17 は、テキスト ファイルを変更するビルトイン InstallShield サポートの存在を確認します。

修正アクション

Windows アプリ パッケージの使用をサポートするためには、フラグされた構成を、インストールからアプリケーションに移動する必要があります。

ISUWP18

ISUWP18 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

このパッケージには、インストール中に実行される 1 つ以上の InstallShield XML ファイルの変更アイテムが含まれています。Windows アプリ パッケージは、XML ファイルの変更をサポートしません。

説明

ISUWP18 は、XML ファイルを変更するビルトイン InstallShield サポートの存在を確認します。

修正アクション

Windows アプリ パッケージの使用をサポートするためには、フラグされた構成を、インストールからアプリケーションに移動する必要があります。

ISUWP19

ISUWP19 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

このパッケージには、インストール中に実行される 1 つ以上の InstallShield スケジュール タスク アイテムが含まれています。 Windows アプリ パッケージは、スケジュール タスクの作成をサポートしません。

説明

ISUWP19 は、スケジュール タスクを作成するビルトイン InstallShield サポートの存在を確認します。

修正アクション

Windows アプリ パッケージの使用をサポートするためには、フラグされた構成を、インストールからアプリケーションに移動する必要があります。

ISUWP20

ISUWP20 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

コンポーネント [2] の拡張子 [1] は、Windows アプリ パッケージに含まれるアプリケーションに予約済み、または禁止されているファイル タイプです。

[1] はプロジェクトに含まれる拡張子の名前で、[2] はその拡張子を含むコンポーネントの名前です。

説明

ISUWP20 は、予約済み、または禁止されているファイル タイプの関連付けの存在を確認します。

修正アクション

フラグされた予約済み、または禁止されているファイル タイプの関連付けを削除して、Windows アプリ パッケージの使用をサポートします。

InstallShield ベスト プラクティス スイート



エディション: *InstallShield ベスト プラクティス スイートは、InstallShield の Premier Edition で提供されています。*



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

一部の ISBP は、基本の MSI プロジェクトと MSI データベース プロジェクトに適用します。

InstallShield には、InstallShield ベスト プラクティス スイートという名前の 1 セットの検証ツールが含まれています。インストールがベスト プラクティス ガイドラインに違反している場合、このスイートの InstallShield ベスト プラクティス (ISBP) 検証ツールによって警告されます。

以下は、InstallShield で提供されている ISBP の一覧です。

テーブル 22 ・ ISBP

ISICE	プロジェクト	説明
ISBP01	基本の MSI、InstallScript MSI、MSI データベース	"ALL" という名前の機能がないことを検証します。
ISBP02	基本の MSI、InstallScript MSI、MSI データベース	"DATABASE" という名前のディレクトリがないことを検証します。
ISBP03	基本の MSI、MSI データベース	50 ユニット未満の ComboBox がないことを確認します。
ISBP04	基本の MSI、MSI データベース	ダイアログで使用されているプロパティがセキュリティで保護された、または制限付きのパブリック プロパティであることを確認します。
ISBP05	基本の MSI、MSI データベース	NULL 値の ControlEvent 条件がないことを確認します。
ISBP06	基本の MSI、InstallScript MSI、MSI データベース	InstallUISequence カスタム アクションが InstallExecuteSequence にもシーケンスされていることを確認します。
ISBP07	基本の MSI、InstallScript MSI、MSI データベース	すべての機能が関連付けられているコンポーネントを持ち、すべてのコンポーネントが機能に関連付けられていることを確認します。
ISBP08	基本の MSI、InstallScript MSI、MSI データベース	ARPINSTALLLOCATION が InstallExecuteSequence の CostFinalize の後に設定されていることを確認します。
ISBP09	基本の MSI、InstallScript MSI、MSI データベース	LIMITUI が ARPNOMODIFY なしに設定されていないことを確認します。
ISBP10	基本の MSI、InstallScript MSI、MSI データベース	AppSearch プロパティがセキュリティで保護された、または制限付きのパブリック プロパティであることを確認します。
ISBP11	基本の MSI、InstallScript MSI、MSI データベース	コンパイル済みの .NET アセンブリが配布されていないことを確認します。

テーブル 22 • ISBP (続き.)

ISICE	プロジェクト	説明
ISBP12	基本の MSI、InstallScript MSI、MSI データベース	自己登録型のファイルがないことを確認します。
ISBP13	基本の MSI、MSI データベース	ダイアログ コントロールによって設定され、インストールで使用されるプロパティにデフォルト値があることを確認します。
ISBP14	基本の MSI、InstallScript MSI、MSI データベース	各ファイルに正しいバージョン情報または MsiFileHash エントリがあることを確認します。
ISBP15	基本の MSI、MSI データベース	RadioButtonGroup に Text が定義されていることを確認します。
ISBP16	基本の MSI、InstallScript MSI、MSI データベース	インストール先が 64 ビットであるコンポーネントがそれぞれ 64 ビット コンポーネントとマークされていることを確認します。
ISBP17	基本の MSI、InstallScript MSI、MSI データベース	.hlp ファイルまたは WinHelp ランタイム ファイルがインストールに含められていないことを検証します。
ISBP18	基本の MSI、InstallScript MSI、MSI データベース	古い形式の API がインポートされていないことを検証します。
ISBP19	基本の MSI、InstallScript MSI、MSI データベース	使用されていない API がインポートされていないことを検証します。
ISBP20	基本の MSI、InstallScript MSI、MSI データベース	Registry テーブルに含まれるレジストリ エントリが、HKLM¥Software などのルート レベルのレジストリ キーや、削除された場合にターゲット マシン上でのトラブルの原因となるような、その他のキーの削除を行わないことを検証します。

ISBP01



プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

Feature ALL conflicts with the installation meta-feature ALL.(機能 ALL がインストール メタ機能 ALL と競合します。)

説明

ISBP01 は、インストールが **ALL** という名前が大文字、小文字、または大文字と小文字の両方で指定されている機能がないことを確認します。機能のコンテキストで、Windows Installer は **ALL** という単語を機能の状態プロパティ (ADVERTISE、REINSTALL および ADDLOCAL など) の有効値として使用するために予約しています。機能の状態プロパティには、特定の機能名のほか、すべての機能を意味する **ALL** という単語を指定することができます。したがって、プロジェクト内の個々の機能に **ALL** という名前を指定することはできません。

修正アクション

このエラーを解決するには、機能の名前を ALL 以外の名前に変更します。詳細については、「機能の作成」を参照してください。

このエラーはまた、出力ウィンドウで ISBP01 エラー メッセージをクリックしても解決することができます。[ダイアログ エディター] ビューで Feature テーブルが表示され、**ALL** という名前の機能を含む行がハイライトされます。機能の名前を変更するには、現在の名前を選択して、新しい名前を入力します。

ISBP02



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

Directory DATABASE conflicts with Windows Installer's undocumented directory DATABASE.(ディレクトリ DATABASE が Windows Installer の文書化されていないディレクトリ DATABASE と競合します。)

説明

ISBP02 は、インストールが **DATABASE** という名前が大文字、小文字、または大文字と小文字の両方で指定されている機能がないことを確認します。Windows Installer は、このディレクトリ名を予約語として使用します。

修正アクション

このエラーを解決するには、[出力] ウィンドウで ISBP01 エラー メッセージをクリックします。[ダイアログ エディター] ビューで Directory テーブルが表示され、**DATABASE** という名前のディレクトリを含む行がハイライトされます。ディレクトリの名前を変更するには、現在の名前を選択して、新しい名前を入力します。

ISBP03



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*

メッセージ (警告)

ComboBox [1] (ダイアログ [2]) の高さが 50 以下です。ComboBox が低すぎる場合、複数のアイテムを一度に表示できません。

[1] はコンボ ボックス コントロールの名前で、[2] はそのコントロールを持つダイアログの名前です。

説明

コンボ ボックス コントロールの高さが 50 インストーラー ユニット未満の場合、エンドユーザーはこのボックスで一度に 1 アイテムしか見ることができなくなることがあります。ユーザビリティを改善するために、高さを 50 以上にすることをお勧めします。

修正アクション

この警告を解決するには、[出力] ウィンドウで ISBP03 警告メッセージをクリックします。[ダイアログ エディター] ビューで Control テーブルが表示され、コンボボックス コントロールを含む行がハイライトされます。このコントロールの Height 列の値を 50 以上に変更します。

ISBP04



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*

メッセージ

メッセージ 1(警告)

コントロール [2] (ダイアログ [1] 内) はプライベート プロパティ [3] を使用しています。この値は、[実行] シーケンスで利用できません。

[1] はプロジェクト内のダイアログの名前で、[2] はそのダイアログにあるコントロールの名前で、[3] はそのコントロールによって設定また使用されているプライベート プロパティの名前です。

メッセージ 2(警告)

コントロール [2] (ダイアログ [1] 内) はセキュリティで保護されていないカスタム プロパティ [3] を使用しています。この値は、[実行] シーケンスで利用できません。

[1] はプロジェクト内のダイアログの名前で、[2] はそのダイアログにあるコントロールの名前で、[3] はそのコントロールによって使用されているプロパティの名前です。

説明

インストールの [ユーザー インターフェイス] シーケンスでプライベート プロパティを設定し、その値を [実行] シーケンスに渡すことはできません。したがって、ISBP04 は、インストールにプライベート プロパティを使用するダイアログ コントロールが含まれていないことを検証します。ダイアログ コントロールでプライベート プロパティが使用されている場合、場合によって、プロパティの値が [実行] シーケンスに渡されることが許可されるように手続きをとる必要があるというメッセージが警告 1 によって表示されます。

また、[実行] シーケンスで昇格された権限が必要なインストールの [ユーザー インターフェイス] シーケンスでパブリック プロパティを設定してあるときに、そのプロパティの値を [実行] シーケンスに渡す場合、プロパティは SecureCustomProperties プロパティの値として一覧されているか、制限付きパブリック プロパティである必要があります。したがって、ISBP04 はまた、インストールにカスタム パブリック プロパティを使用するダイアログ コントロールがある場合、カスタム パブリック プロパティが **SecureCustomProperties** プロパティの値として一覧されていることも確認します。カスタム パブリック プロパティが **SecureCustomProperties** プロパティの一覧にない場合、場合によってプロパティの値が [実行] シーケンスに渡されることが許可されるように手続きをとる必要があるというメッセージが警告 2 によって表示されます。

修正アクション

[実行] シーケンスでプロパティを使用する必要がない場合、この警告メッセージは無視することができます。ただし、このシーケンスでプロパティを使用する必要ある場合、この警告を解決する必要があります。

警告 1 を解決する必要がある場合、プライベート プロパティをパブリック プロパティに変更します。この変更を行うには、まず [ダイアログ] ビューでダイアログ コントロールを見つけ、コントロールの Property プロパティの文字列をすべて大文字にします (パブリック プロパティはすべて大文字)。また、プロジェクト内の他の場所で使用されているプロパティ名もすべて同様に変更します。

警告 2 を解決する必要がある場合、[プロパティ マネージャー] ビューを開きます。SecureCustomProperties プロパティの Value 列で、警告メッセージで言及されているプロパティの名前を追加します。このプロパティに複数のプロパティが含まれている場合、各プロパティをセミコロン (;) で区切ります。

Private Properties (Windows Installer ヘルプ ライブラリ)
Public Properties (Windows Installer ヘルプ ライブラリ)
SecureCustomProperties Property (Windows Installer ヘルプ ライブラリ)

ISBP05



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*

メッセージ (警告)

コントロール [2] (ダイアログ [1] 内) に NULL 値の条件が含まれています。コントロール イベントが常に実行されるように、1 の条件を使用します。

[1] はプロジェクト内のダイアログの名前で、[2] はそのダイアログにあるコントロールの名前です。

説明

ISBP05 は、各ダイアログ コントロールの各イベントの条件が Null 値ではないことを確認します。この検証により、Windows Installer によってコントロールのイベントが適切な状況で起動されるようになります。

修正アクション

コントロールの他のイベントが起動されなかった場合にのみ Null 値の条件があるイベントが Windows Installer によって起動される必要がある場合、この警告メッセージは無視することができます。

この警告を解決する必要がある場合、まず [ダイアログ] ビューでダイアログを見つけます。ダイアログの下にある [動作] をクリックします。コントロールの一覧で、警告メッセージで言及されているコントロールをクリックします。右下のペインに [イベント] タブが表示されていることを確認します。右上に表示されているペインのグリッド内で、イベントの条件を入力します。Windows Installer によってこのイベントが起動されるようにするには、条件として 1 を入力します。

ControlEvent Table (Windows Installer ヘルプ ライブラリ)
Conditional Statement Syntax (Windows Installer ヘルプ ライブラリ)

ISBP06



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (警告)

カスタム アクション [1] は InstallUISequence でスケジュールされていますが InstallExecuteSequence ではスケジュールされていません。このカスタム アクションは、サイレントまたは簡易 UI インストールでは実行されません。

[1] はプロジェクトに含まれるカスタム アクションの名前です。

説明

[インストール] シーケンス中 [実行] シーケンスにスケジュールされたアクションは、サイレント インストール、基本の UI インストール、または簡易 UI インストールでは実行されません。

修正アクション

サイレント インストール、基本の UI インストール、および簡易 UI インストール中 Windows Installer によってカスタム アクションが起動される必要がある場合、カスタム アクションを UI シーケンスに加えて、またはその代わりに、[インストール]-[実行] シーケンスにスケジュールすることも考えられます。

アクションの再スケジュールするには、まず [カスタム アクションとシーケンス] ビューを開いて、警告メッセージの中で言及されているアクションを見つけます。次いで、そのアクションを [インストール] シーケンスの [ユーザー インターフェイス] シーケンスから [インストール] シーケンスの [実行] シーケンスにドラッグします。

両方のシーケンスでアクションをスケジュールするには、[インストール] シーケンスの [ユーザー インターフェイス] シーケンスのアクションを [インストール] シーケンスの [実行] シーケンスにコピーします。[カスタム アクションとシーケンス] ビューでこれを行うには、CTRL を押しながら、アクションを前のシーケンスから後のシーケンスにドラッグします。

サイレント インストール、基本の UI インストール、または縮小 UI インストール中、このカスタム アクションを Windows Installer によって起動しない場合、この警告は無視できます。

ISBP07



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(警告)

機能 [1] にコンポーネントがありません。

[1] は、プロジェクトに含まれる機能の名前です。

メッセージ 2(警告)

コンポーネント [1] は機能に関連付けられていません。

[1] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISBP07 は、プロジェクト内のすべてのコンポーネントが最低 1 つの機能に属していることを確認します。コンポーネントが最低 1 つの機能に関連付けられれていない場合、Windows Installer はそのコンポーネントをインストールすることができません。

ISBP07 はまた、プロジェクト内のすべての機能が最低 1 つのコンポーネントを含んでいることを確認します。コンポーネントには、ファイル、ショートカット、レジストリ エントリなどのインストール要素が含まれています。機能にコンポーネントがない場合、機能にインストールするものが何もないということになります。

修正アクション

これらの警告のいずれかを解決するには、[セットアップのデザイン] ビューを使用して、コンポーネントを機能に関連付けます。詳細については、「既存コンポーネントを機能に関連付ける」を参照してください。

ISBP08



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (警告)

InstallExecuteSequence の CostFinalize の後に ARPINSTALLLOCATION を設定する Type 51 アクションがないようです、。

説明

ARPINSTALLLOCATION プロパティは、製品のプライマリ インストール先への完全修飾パスを指定します。Windows Installer はこの値を Uninstall レジストリ キーに書き込みます。

ARPINSTALLLOCATION プロパティは通常、プロパティ設定型のカスタム アクション (タイプ 51) によって設定されます。

SetARPINSTALLLOCATION カスタム アクションは、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに自動的に追加されるビルトイン InstallShield カスタム アクションです。このカスタム アクションをプロジェクトから削除すると、ISBP08 警告に遭遇する場合があります。

修正アクション

この警告を解決するには、プロジェクトにカスタム アクションを次の設定と共に追加します。

- プロパティ名: ARPINSTALLLOCATION
- プロパティ値: [INSTALLDIR]
- 実行スケジュール: Always execute
- インストール実行シーケンス: After CostFinalize

  ***メモ・** ベスト プラクティスに従うには、機能の移行がメジャー アップグレード アイテムで選択されている場合、CostFinalize の後のすべてアクションを MigrateFeatureStates の後にシーケンスします。*

- インストール実行条件: Not Installed

残りの設定はすべてデフォルト値のままにします。

詳細については、「[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でカスタム アクションを作成する」を参照してください。

ARPINSTALLLOCATION Property (Windows Installer ヘルプ ライブラリ)

ISBP09

***プロジェクト-** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

The property LIMITUI has been set, but ARPNOMODIFY has not.This can lead to undesirable behavior with Add or Remove Programs.(プロパティ LIMITUI は設定されていますが、ARPNOMODIFY は設定されていません。これは [プログラムの追加と削除] での正しくない動作につながります。)

説明

インストールで LIMITUI プロパティを設定した場合、**ARPNOMODIFY** プロパティも設定する必要があります。

LIMITUI プロパティは "ユーザー インターフェイス レベル" を [基本] に設定します。基本の UI で実行されたインストールは、エンドユーザーに進捗状況メッセージのみを表示します。エンドユーザーは、機能を選択したり、フィードバックを提供したりすることをできません。

ARPNOMODIFY プロパティは、エンドユーザーが [プログラムの追加と削除] で製品を変更しようとするのをできないようにします。

修正アクション

このエラー解決するには、[プロパティ マネージャー] ビューで **ARPNOMODIFY** プロパティをプロジェクトに追加し、その値を 1 に設定します。詳細については、「Windows Installer ベースのプロジェクトにおけるプロパティの作成」を参照してください。

User Interface Levels (Windows Installer ヘルプ ライブラリ)

ISBP10



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (警告)

AppSearch [1] はセキュリティで保護されていないカスタム プロパティ [2] を使用しています。この値は、[実行] シーケンスで利用できません。

[1] はプロジェクトに含まれるシステム検索の名前で、[2] はそのシステム検索によって使用されるプロパティの名前です。

説明

[実行] シーケンスで昇格された権限が必要なインストールの [ユーザー インターフェイス] シーケンスにパブリック プロパティが設定されている場合、そのプロパティの値を [実行] シーケンスに渡すためには、プロパティが SecureCustomProperties プロパティの値としてリストされているか、または制限付きパブリック プロパティである必要があります。したがって、ISBP10 は、プロジェクトの AppSearch テーブルにカスタム パブリック プロパティがある場合、そのカスタム パブリック プロパティが **SecureCustomProperties** プロパティの値として一覧されていることを確認します。カスタム パブリック プロパティが **SecureCustomProperties** プロパティの一覧にない場合、場合によってプロパティの値が [実行] シーケンスに渡されることが許可されるように手続きをとる必要があるというメッセージが警告によって表示されます。

修正アクション

[実行] シーケンスでプロパティを使用する必要がない場合、この警告メッセージは無視することができます。ただし、このシーケンスでプロパティを使用する必要ある場合、この警告を解決する必要があります。

この警告を解決するには、[プロパティ マネージャー] ビューを開きます。SecureCustomProperties プロパティの Value 列で、警告メッセージで言及されているプロパティの名前を追加します。このプロパティに複数のプロパティが含まれている場合、各プロパティをセミコロン (;) で区切ります。

SecureCustomProperties Property (Windows Installer ヘルプ ライブラリ)
AppSearch Table (Windows Installer ヘルプ ライブラリ)

ISBP11



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (警告)

ファイル [1] は .NET アセンブリのプリコンパイル済みネィティブ イメージのようです。これにより、ターゲット マシンでプリコンパイルするよりも効率が悪くなる可能性があります。

[1] は .NET アセンブリのプリコンパイル済みネィティブ イメージの名前です。

説明

ISBP11 は、プロジェクトに .NET アセンブリのプリコンパイル済みネィティブ イメージが含まれていないことを確認します。

修正アクション

この警告を解決するには、メッセージの中で言及されているネィティブ イメージ ファイルを適切な .NET アセンブリ ファイルで置き換えます。

アセンブリがインストール中にマシン コードにコンパイルされるようにする場合、ファイルのコンポーネントの "NET プリコンパイル アセンブリ" 設定で [はい] を選択します。

プリコンパイル、またはジャストインタイプ コンパイルでは、一部のコードが実行中に呼び出されない可能性があることを考慮に入れています。プリコンパイルは実行中、必要に応じてアセンブリを変化し、後の呼び出しでアクセスできるように結果のコードを格納します。ターゲット マシンでのプリコンパイルでは、プロセスにおいて、実行されているマシンの正確なアーキテクチャを有効に利用することができます。

ISBP12



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

カスタム アクション [1] は RegSvr32 を通して自己登録を呼び出しているようです。ベスト プラクティスでは、COM および Registry テーブル データをインストーラー パッケージに作成することが推奨されています。

[1] は、インストール時に RegSvr32.exe を使用してファイルを登録する可能性があるプロジェクト内のカスタム アクションの名前です。

メッセージ 2(エラー)

ファイル [1] は自己登録型です。ベスト プラクティスでは、COM および Registry テーブル データをインストーラー パッケージに作成することが推奨されています。

[1] は、自己登録型としてマークされているファイルの名前です。

説明

InstallShield では、COM サービスが自己登録するように指定することができますが、好まれる COM サービスの登録方法は、ビルド時またはデザイン時にファイルから COM 情報を抽出する方法です。この方法を使用した場合、InstallShield は COM クラス情報を .msi データベースの Class、ProgID、および Registry テーブルに書き込みます。自己登録には、いくつかの制限があります。詳しくは、「COM サーバーの自己登録」を参照してください。

修正アクション

エラー 1 を解決するには、エラー メッセージで参照されたカスタム アクションを削除して、そのファイルの COM 抽出を適切に構成します。

エラー 2 を解決するには、このファイルの COM 抽出を構成します。

詳細については、「COM 情報を COM サーバーから抽出する」を参照してください。

ISBP13



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*

メッセージ (エラー)

プロパティ [1] にデフォルト値がなく、ダイアログ コントロールで設定されていて、テーブル [2] で使用されています。サイレントまたは簡易 UI インストールで、作成されない可能性があります。

[1] はプロジェクト内のプロパティの名前で、[2] は、そのプロパティを使用するテーブルの名前です。)

説明

ほとんど場合、**Property** テーブルの各プロパティにはデフォルト値が必要です。エンドユーザーがインストールをサイレント、簡易 UI、または基本の UI モードでインストールを実行したとき、デフォルト値が使用されます。

修正アクション

プロパティにデフォルト値を追加する必要がない場合、このエラー メッセージは無視することができます。

このエラーを解決するには、[プロパティ マネージャー] ビューを開きます。エラー メッセージで指定されたプロパティの Value 列に値を入力します。

ISBP14



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

ファイル [1] に、ファイル バージョンも MsiFileHash レコードもありません。パッチのインストール時に、ソース パッケージが必要になる可能性があります。

[1] は、プロジェクトに含まれるファイルの名前です。

メッセージ 2(エラー)

ファイル [1] はバージョン [2] として記録されていますが、実際のバージョンは [3] です。パッチのインストール時に、ソース パッケージが必要になる可能性があります。

[1] は、プロジェクトに含まれるファイルの名前です。[2] は、ファイルに構成されたバージョン番号ですが、[3] が実際のバージョン番号です。

説明

ISBP14 は、プロジェクトの各ファイルが正しいバージョン情報または MsiFileHash エントリをもっていることを確認します。バージョン付きファイルの正しいバージョン情報およびバージョンがないファイルの MsiFileHash エントリを使用することにより、Windows Installer は必要のないファイル コピーを検出して、除去することができます。また、以前のバージョンの製品にパッチを適用するときに、ソース パッケージが必要ならないようにします。

修正アクション

エラー 1 を解決する方法として、ファイルのバージョン番号をファイルに追加することができます。ファイルにバージョンがない場合、ファイル ハッシュが使用されるようにファイルを構成します。

InstallShield では、ファイルのバージョン情報をオーバーライドすることができます。ファイルのプロパティがオーバーライドされると、エラー 2 が発生する場合があります。このエラーを解決するために、バージョンのオーバーライドを削除することもできます。

ファイルの設定を構成するには、まず [ファイルとフォルダー] ビューを開きます。**[インストール先コンピューターのファイル]** ペインで、該当のファイルを右クリックしてから、[プロパティ] を選択します。[プロパティ] ダイアログ ボックスが開き、バージョン情報の構成とファイル ハッシュを使用するかどうかの指定を行うことができます。

File Table (Windows Installer ヘルプ ライブラリ)

ISBP15



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*

メッセージ (エラー)

Control [2] on the Dialog [1] is a borderless RadioButtonGroup with overlapping controls and data in the Text column.This may cause repaint issues on some systems.(コントロール [2] (ダイアログ [1] 内) は、Text 列にオーバーラップするコントロールとデータがある境界線のない RadioButtonGroup です。これにより、一部のシステムで再描画の問題が発生する可能性があります。)

説明

ISBP15 は、ダイアログに境界線のないラジオ ボタン グループ コントロールがあり、かつ 1 つまたは複数の他のコントロールがそのラジオ ボタン グループとオーバーラップするとき、ラジオ ボタン グループの Text 属性に何も指定されていないことを確認します。指定されている場合、Windows Installer の一部のバージョンで再描画に関する問題が発生する可能性があります。

修正アクション

このエラーを解決するには、[出力] ウィンドウで ISBP15 エラー メッセージをクリックします。[ダイアログ エディター] ビューで Control テーブルが表示され、ラジオボタン コントロールを含む行がハイライトされます。ラジオ ボックス コントロールの Text 列の文字列を削除します。

また、ラジオ ボックス コントロールが境界線を使用するように変更することもできます。この変更を行うには、まず [ダイアログ] ビューを開いて、エラー メッセージで言及されているダイアログを見つけます。ラジオ ボタン コントロールをクリックして、Has Border プロパティを True に変更します。

RadioButtonGroup Control (Windows Installer ヘルプ ライブラリ)

ISBP16



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

Component [1] installed to 64-bit location [2] is not marked with the 64-bit component attribute.This may allow files to be installed to an incorrect location.[1] is the name of a component, and [2] is the destination folder for the component's files.(コンポーネント [1] (64 ビットの場所 [2] にインストール済み) が、コンポーネント属性で 64 ビットとマークされていません。このため、ファイルが不適切な場所にインストールされる可能性があります。

[1] はコンポーネントの名前で、[2] はコンポーネントのファイルのインストール先フォルダーです。)

説明

コンポーネントが 64 ビットとして構成されていない場合、Windows Installer はそのコンポーネントのファイルを適切な 64 ビットの場所にインストールしない可能性があります。

修正アクション

コンポーネントが 64 ビットであると指定するには、コンポーネントの "64 ビット コンポーネント" 設定で [はい] を選択します。

"64 ビット コンポーネント" 設定、および追加のコンポーネントの設定に関する詳細については、「コンポーネントの設定」を参照してください。

ISBP17



プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

コンポーネント [2] のファイル [1] は、WinHelp で、Windows Vista 以降に配布されない可能性があります。

[1] は、プロジェクト内の .hlp ファイルの名前で、[2] は、その .hlp ファイルを含むコンポーネントの名前です。

メッセージ 2(エラー)

コンポーネント [2] のファイル [1] は、WinHelp ランタイムで、Windows Vista 以降に配布されない可能性があります。

[1] はプロジェクトに含まれる Windows Help アプリケーション ファイルの名前で、[2] は、そのファイルを含むコンポーネントの名前です。

説明

ISBP17 は、プロジェクトに .hlp ファイルが含まれていないことを検証します。Windows Vista 以降は、この種類のファイルをサポートしていません。

また、ISBP17 は、WinHlp32.exe と WinHelp.exe がプロジェクトに含まれていないことも検証します。これらのアプリケーションは再配布できません。

修正アクション

この検証エラーを解決するためには、プロジェクトからヘルプ ファイルを削除します。代案として、ヘルプ システムを別のファイル形式 (例、.chm、.htm、.xml など)に変換することも考えられます。この場合、Windows API からの呼び出しも、新しいヘルプ ファイルに変更する必要があります。

ISBP18



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のファイル [1] は、古い形式の API [3] をインポートします。

[1] は、プロジェクトに含まれるファイルの名前で、[2] は、そのファイルを含むコンポーネントの名前です。[3] は、そのファイルが使用する古い形式の API の名前です。

説明

ISBP18 は、プロジェクトに含まれるすべての .dll ファイルおよび .exe ファイルが旧式のカーネル API を使用しないことを検証します。

修正アクション

この検証エラーを解決するには、このファイルを古い形式の API を使用しない更新済みのバージョンで置き換えます。更新済みのバージョンを作成するとき、場合によって、.exe または .dll ファイルを上書きし、再ビルドする必要があります。exe または .dll ファイルがサードパーティ ベンダーからのものである場合、そのベンダーより更新済みのバージョンを取得してください。

ISBP19



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のファイル [1] は、非推奨の API [3] をインポートします。

[1] は、プロジェクトに含まれるファイルの名前で、[2] は、そのファイルを含むコンポーネントの名前です。[3] は、そのファイルが使用する非推奨の API の名前です。

説明

ISBP19 は、プロジェクトに含まれるすべての .dll ファイルおよび .exe ファイルが非推奨のカーネル API を使用しないことを検証します。

修正アクション

この検証エラーを解決するには、このファイルを非推奨の API を使用しない更新済みのバージョンで置き換えます。更新済みのバージョンを作成するとき、場合によって、.exe または .dll ファイルを上書きし、再ビルドする必要があります。exe または .dll ファイルがサードパーティ ベンダーからのものである場合、そのベンダーより更新済みのバージョンを取得してください。

ISBP20



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

レジストリ エントリ [1] の Name 列に、アスタリスクが含まれています。これは、Windows Installer がレジストリ キー [2] を削除する原因となります。

[1] は、アスタリスク (*) を含むレジストリ エントリ行の Registry テーブルでの Registry 列の名前です。[2] は、アンインストール中に削除されるレジストリ キーの名前です。

メッセージ 2(警告)

レジストリ エントリ [1] は、ソフトウェア アプリケーションとのインストールで推奨されないキーにインストールします。このエントリをインストールすると、ターゲットマシン ([2]) に悪影響が及ぶ可能性があります。

[1] は、ターゲットシステム上でトラブルの原因となる可能性のあるレジストリ エントリ行の Registry テーブルでの Registry 列の名前です。[2] は、レジストリ キーの名前です。

説明

ISBP20 は、製品がターゲット マシンから削除されるときに、HKEY_LOCAL_MACHINE¥SOFTWARE などのルート レベルのレジストリ キーを削除するようなレジストリ エントリがプロジェクトに含まれていないことを検証します。次のテーブルは、アンインストール中に問題を引き起こすようなレジストリ データの例を示します。この例では、HKEY_LOCAL_MACHINE¥SOFTWARE 全体と、そのサブ キーおよび値のすべてが削除されます。

テーブル 23 ・ ISBP20 エラーをトリガーするレジストリ データの例

レジストリ	ルート	キー	名前	値	コンポーネント
Registry2	2	SOFTWARE	*		MyComponent

ISBP20 は、ターゲット マシン上で悪影響を及ぼすような他のレジストリ エントリがプロジェクトに含まれていないことを検証します。

修正アクション

この検証エラーまたは警告を解決するには、[出力] ウィンドウで ISBP20 メッセージをクリックします。[ダイレクト エディター] ビューに Registry テーブルが表示され、問題を引き起こす可能性のあるレジストリ エントリを含む行がハイライト表示されます。レジストリ エントリを削除するには、その行を右クリックしてから [行の削除] を選択します。

Registry テーブル (Windows Installer ヘルプ ライブラリ)

インストールまたはアンインストールがターゲット システムを再開するタイミングについて



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI — InstallScript ユーザー インターフェイス (UI) のスタイルが、外部 UI ハンドラーとして InstallScript エンジンを使用する従来型のスタイルの場合*

この情報は、InstallScript UI に新しいスタイル (InstallScript エンジンを埋め込み UI ハンドラーとして使用するスタイル) が使用されている InstallScript MSI プロジェクトには適用しません。詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

BATCH_INSTALL システム変数がゼロ以外の値に設定されているとき、それはターゲット システムが再起動した後に実行が必要な操作が 1 つ以上あることを意味します。たとえば、ファイルがターゲット システム上に既に存在し、またそれがロックされているために、インストールがそのファイルをインストールできないと判断した場合、BATCH_INSTALL がゼロ以外の値に設定されるときがあります。

インストールに含まれるファイルがロックされているためにインストールできないとき、インストールは次回システムが再開するときにそのファイルを更新する必要があることをオペレーティング システムに自動的に通知します。追加の Windows ファイルがロードされる前にファイルを更新しなくてはならない可能性があるため、ファイルは Windows がロードされるときに更新されます。なお、InstallScript インストールではファイルは更新されません。

BATCH_INSTALL は、システムが再開した後にロックされたファイルを削除するためにアンインストール中に設定されます。ただし、再起動後にキャッシュされたインストール ファイルを使用することはできません。したがって、再起動関連の処理で有効なのは、オペレーティング システムが実行する処理に限ります。

If BATCH_INSTALL は、ファイル転送が終了したときにゼロ以外の値に設定され、以下が発生します:

- インストールは、ファイルの自己登録を一切行わない。これは、ファイルが正しく自己登録されるためにインストールのファイル全てが完全にインストールおよび更新されていなくてはならない場合があるためです。インストールはこれらのファイルが他の自己登録ファイルの依存関係である可能性を考慮し、インストールできなかったファイルが自己登録型であるかどうかに関わらず、ファイルの自己登録を一切行わない点にご注意ください。この場合、インストールは自己登録の必要があるファイルを記録して、再開した後にインストールがこれらのファイルを自己登録できるようにします。

- フレームワークは自動的に **SdFinishReboot** (OnFirstUIAfter または OnMaintUIAfter イベント内) を呼び出して、エンド ユーザーがインストール完了後にシステムを自動的に再開するためのオプションを提供します。
- インストールが完了すると適切な RunOnce レジストリを作成し、再開後にインストールが実行するようにします。再開後、インストールは /reboot パラメーターを使って実行します。

  ALLUSERS スクリプト変数が 1 に設定された場合、RunOnce エントリは HKEY_LOCAL_MACHINE の下に作成されます。その他の場合、このエントリは HKEY_CURRENT_USER の下に作成されます。Windows は、これらの両方の場所で RunOnce エントリをサポートします。

  RunOnce レジストリ値の名前は、**InstallShield セットアップ %n** で、**%n** は一意のキー名として使用可能な最も小さい整数です。レジストリ値のデータは、再起動の後にインストールを実行する完全なコマンド ラインです。コマンド ラインには、Setup.exe のパスとファイル名が含まれます。

  再起動の後、インストールは Disk1 の場所である DISK1TARGET から実行します。したがって、再起動の後にインストールを実行する場合、インストールに必要な Disk1 ファイルがインストール済みでなくてはなりません。

  再起動の後にキャッシュされたインストール ファイルを使用することはできないため、これはアンインストールには適用しませんので、ご注意ください。

インストールが再開の後に実行すると (つまり、/reboot パラメーターを使って実行すると)、以下が発生します:

- 登録が必要であるとインストールによって記録された自己登録ファイルはすべて登録されます。
- **OnRebooted** イベント ハンドラーが呼び出されます。

アンインストールでは、インストールは再開の後に実行しません。したがって、これらの処理は一切行われません。

Windows Installer ベースのインストールのデバッグ

MSI デバッガーを使用して、Windows Installer ベースのリリースをデバッグすることができます。MSI デバッガーを使用して、パッケージの [ユーザー インターフェイス] と [実行] シーケンスの順番に従って Windows Installer のプロパティを表示/設定することができます。

ブレークポイントに到達するまで各アクションおよびダイアログが実行され、ブレークポイントに到達した時点で実行が停止します。ここで、ウォッチ ウィンドウと変数 ウィンドウにプロパティを表示し、設定することができます。最後に、残りのアクションを順番に実行するか、または、デバッガーを中止します。



メモ- *MSI デバッガーと InstallScript デバッガーは目的がまったく異なる点にご注意ください。セットアップのデザインについての詳しい情報*

MSI デバッガーを開始する

MSI デバッガーを開始する前に、InstallShield でリリースをビルドし、[MSI デバッガー] ビューを開く必要があります。さらに、特定のアクションまたはダイアログでのインストール状態を表示できるように、常にブレークポイントを設定します。

MSI デバッガーを開いてデバッグを開始すると、その時点で [リリース] ビューで強調されているリリースがデバッグされます。リリースをまだビルドしていない場合、またはリリースが Setup.exe で圧縮されている場合、MSI デバッガーには何も表示されず、ツールバーやメニュー項目も使用できません。パッケージをデバッグするには、ビルドされたリリースに切り替えるか、または必要なオプションで再ビルドします。



タスク ***デバッグを開始するには、以下のいずれかを行います。***

- MSI デバッガーのツールバーで **[MSI デバッガー]** ボタンをクリックする。
- MSI デバッガーのツールバーで **[ステップ オーバー]** ボタンをクリックする。
- **[ビルド]** メニューで **[MSI デバッガー]** をポイントし、**[MSI デバッガーの開始]** をクリックします。
- **[ビルド]** メニューで **[MSI デバッガー]** をポイントし、**[ステップ オーバー]** をクリックします。
- F5 を押す。
- F11 を押す。

インストールのデバッグが実行されます。ブレーク ポイントに到達すると再び MSI デバッガーが強調され、インストールの実行は停止します。MSI Debugger で、次のいずれかの操作を実行できます。

- Windows Installer のプロパティを参照および設定する。
- 残りのインストールを順番に実行する。
- MSI デバッガーおよび/またはインストールを終了する。

MSI デバッガーでブレークポイントを設定する



タスク ***MSI デバッガーでブレーク ポイントを設定するには、以下の手順を実行します:***

1. **[リリース]** ビューで、デバッグを行うリリースを選択します。
2. **[MSI デバッガー]** ビューを開きます。MSI デバッガーでは、2 つのリストが表示されます。最初に [ユーザー インターフェイス] シーケンスにおける標準およびカスタム アクションがすべて表示され、次にインストールのダイアログがすべて表示されます。
3. ブレークするアクションまたはダイアログを含む行にカーソルを置きます。
4. 次のいずれかの方法で、この行にブレークポイントを設定します。
   - MSI デバッガーのツールーバー上の **[ブレークポイントの切り替え]** ボタンをクリックします。
   - **[ビルド]** メニューで **[MSI デバッガー]** をポイントし、**[ブレークポイントの切り替え]** をクリックします。
   - F9 を押す。

ブレークポイントは、デバッグを開始する前、またはデバッグ セッションを開いている間に設定できます。

ブレークポイントに到達したら、プロパティを表示して設定するか、またはデバッグを続行します。

ヒント: アクションに条件が存在する場合は、各アクションの後にその条件がリストされます。条件付のアクションには慎重にブレークポイントを設定します。条件の評価が True にならないとアクションは実行されず、その部分は飛ばされます。

ブレークポイントを削除する

タスク ***ブレークポイントを削除するには、次の手順に従います:***

1. **[追加ツール]** の下のビュー リストにある **[MSI デバッガー]** をクリックします。
2. ブレークポイントを削除するには、ブレークポイントのある行にカーソルを置き、**[ブレークポイントの切り替え]** ボタンをクリックするか F9 を押します。

タスク ***デバッガーですべてのブレークポイントの設定を解除するには、以下のいずれかを行います。***

- [**すべてのブレークポイントの削除**] ボタンをクリックする。
- SHIFT +F5 を押す。

MSI デバッガーでプロパティの表示と設定を行う

インストールを順番に実行しているために MSI デバッガーがブレークポイント、アクションまたはダイアログで停止した場合、Windows Installer のプロパティを表示し、その値を ウォッチ ウィンドウ、または変数ウィンドウで実行時に変更できます。

変数ウィンドウ

変数ウィンドウには、データベースの Property テーブル (InstallShield の [プロパティ マネージャー] ビュー に表示されます) にあるすべてのプロパティとその現在の値が表示されます。

インストールの実行中にプロパティの値を変更するには、[プロパティ値] 列を編集します。

変数ウィンドウが表示されない場合は、[表示] メニューから [変数] を選択します。

ウォッチ ウィンドウ

ウォッチ ウィンドウで、プロパティの名前を入力して、インストールの任意の時点でプロパティの値を確認することができます。たとえば、TARGETDIR が実行時に何に解決されるかを見るには、Name 列に **「TARGETDIR」** と入力します。

インストールの実行中にプロパティの値を変更するには、[プロパティ値] 列を編集します。

ウォッチ ウィンドウが表示されない場合は、[表示] メニューで [監視] をクリックします。

MSI デバッガーでアクションにステップインする

MSI デバッガーでカスタム アクションにステップインすると、コードをデバッグできます。次の種類のカスタム アクションにステップインしたとき、登録済みのデバッガーを起動できます。

- Windows Installer .dll ファイル カスタム アクション
- 標準 .dll ファイル カスタム アクション
- Visual Basic Script または JavaScript カスタム アクション
- InstallScript カスタム アクション



タスク

MSI デバッガーで現在のアクションにステップインするには、次のいずれかを行います。

- MSI デバッガーのツールバーで **[ステップイン]** ボタンをクリックする。
- **[ビルド]** メニューで **[MSI デバッガー]** をポイントし、**[ステップ イン]** をクリックします。
- F11 を押す。

ダイアログが開き、登録済みのデバッガーを起動するかどうかたずねます。[はい] を選択するとデバッガーが起動します。[いいえ] を選択して、次のカスタム アクションにステップ オーバーします。



メモ: *InstallShield ではエントリ ポイントのソース コードが提供されていないため、登録済みデバッガーが起動されると一部のマシン コードが表示されます。コードを見るには、[ステップイン] ボタンを 2 度クリックする必要があります。*

MSI デバッガーでアクションを順番に実行する



タスク

MSI デバッガーで現在のアクションまたはダイアログを順番に実行してデバッグを続行するには、次のいずれかを実行します:

- MSI デバッガーのツールバーで **[ステップ オーバー]** ボタンをクリックする。
- **[ビルド]** メニューで **[MSI デバッガー]** をポイントし、**[ステップ オーバー]** をクリックします。
- F10 を押す。

MSI デバッガーが標準またはカスタム アクションのブレークポイントに到達すると、インストールのデバッグは停止し、MSI デバッガーが強調されます。ここで、インストーラーのプロパティをウォッチし、設定することができます。

引き続きアクションを順番に実行するには、F10 キーを押し続けます。セットアップ ウィザードの各ダイアログで [次へ] をクリックしないとシーケンスが進行しない場合もあります。アクションの戻り値を表示するには、アクション実行後にカーソルをアクションの上に置きます。戻り値はツール ヒントとして表示されます。アクションが正常に実行されると、ツール ヒントには ERROR_SUCCESS のように表示されます。

アクションにブレークポイントを設定すると、MSI デバッガーは後続のアクションでブレークすることになりますが、ダイアログにブレークポイントを設定すると単にそのダイアログに移動するだけで、次のブレークポイントまで介入することはありません。

MSI デバッガーを停止する



タスク

リリースのデバッグを中止するには、以下のいずれかを行います:

- MSI デバッガーのツールバーで **[デバッガーの中止]** ボタンをクリックする。
- **[ビルド]** メニューで **[MSI デバッガー]** をポイントし、**[MSI デバッガーの停止]** をクリックします。
- SHIFT +F5 を押す。

デバッグ セッションを終了しようとするときにデバッガーがダイアログ上のブレイクポイントで停止した場合、InstallShield は、「デバッグを中止するには、ダイアログをすべて閉じてください。」というメッセージをメッセージ ボックスに表示します。開いているダイアログを閉じるには、まず F11 を押して、現在のブレークポイントをステップ オーバーします。それから開いているすべてのダイアログを閉じます。

インストールをキャンセルしても MSI デバッグを中止できます。

インストールを複数のディスクまたは CD に分割する

プログラムが大きくなるにつれて、ディスクの分割の必要性が増します。このような場合、何年も前には複数のフロッピー ディスクで製品を出荷していました。現在の標準は CD-ROM または DVD を使用することです。CD または DVD の容量はフロッピーディスクよりも非常に大きいですが、多くの製品ではそれ以上のスペースを必要とします。マルチメディア チュートリアル、大きなヘルプ ライブラリ、画像の多いプログラムなどが製品に入っていると、標準 CD サイズである 650 MB を超える場合があります。

CD、DVD、カスタムサイズのメディア ディスクにかかわらず、インストールが 1 枚のディスクに収まりきらない場合は、複数のディスクに分割する必要があります。

リリース ウィザードを使用する

複数のディスクにインストールを分割する方法を定義する方法は、リリース ウィザードを使用することです。ウィザードは、選択したメディアのサイズにかかわりなくリリース作成を手順を追って説明します。ここで複数のディスクにファイルを分割する方法を指定できます。また、これらのファイルに適用する圧縮も指定することができます。

ウィザードでは、InstallShield が必要に応じてインストールを複数のディスク上に自動的に分割するように設定することも、またインストール ファイルの分割方法をカスタマイズすることもできます。インストールを複数のディスクに分割する方法をカスタマイズする場合は、インストールの問題を防ぐために「ディスク分割規則」を参照してください。

制限

Windows Installer サービスの制限のために、リムーバブル ディスク以外のディスク上で複数ディスク インストールを実行することはできません。たとえば、インストールが 2 つの CD に分割されている場合、インストールをテストするためには、物理的にこの CD を作成する必要があります。インストールを固定のディスクから実行しようとすると、このインストールは失敗します。

複数ディスク インストールを圧縮する

Setup.exe も .msi ファイルも複数のディスクに分割することができないので、ソース ファイルはこれらのファイルと別に保管する必要があります。サイズ制限のないネットワーク インストールを作成してすべてのファイルを圧縮するように指定した場合で、.msi ファイルまたはSetup.exeの作成を選択すると、これらの圧縮ファイルは .msi または Setup.exe の中に配置されます。その他のすべてのメディア タイプでは、Setup.exe、.msi ファイル、およびすべてのソースファイルが格納されてい .cab ファイル は個別にメディアに配置されます。

たとえば、1.5 MB のファイル、Setup.exe、および Windows NT 用のインストール ファイルがそれぞれ格納された 3 つの機能を含むインストールがあって、サイズが 2 MB のカスタム メディア タイプを作成するとします。ビルドによって複数のディスクが作成されます。ディスク 1 には Setup.exe、InstMsiW.exe (Windows NT マシンに Windows Installer サービスをインストールするロジックが含まれています)、Setup.ini (Setup.exe を含むインストールに必須) および .msi ファイルが含まれます。残りのディスクには、ソースファイルの圧縮コピーを保管する .cab ファイルが格納されます。

ディスク分割規則

リリース ウィザードで、インストールを必要な数のディスクに自動的に分割させるように選択することができます。ウィザードは以下の規則にしたがいます。すべての複数ディスク インストールはこの規則にしたがう必要があります。複数のディスクにインストールを分割する方法をカスタマイズする場合、これらの規則にしたがいます。

- **Setup.exe は、ディスク 1 上にある必要があります。**この規則は、インストールの出荷にフロッピー ディスクを使用し、Setup.exe を出荷する必要があるときに適用します。Setup.exe のサイズ ー Windows 9x、Windows NT、またはその両方など選んだプラットフォームによりますが、可能範囲は 1.31 MB から 2.58 MB です。Windows Installer サービスのインストール ファイルもビルドに含まれています。選択したプラットフォームに応じて InstMsiW.exe または InstMsiA.exe と呼ばれるこのファイルも、ディスク 1 上に置く必要があります。
- **ディスク 1 上に置く必要のある 2 番目のファイルは、ビルドする .msi ファイルです。**.msi ファイル、Setup.exe および Windows Installer インストール ファイルの合計サイズは、標準 1.4 MB フロッピーディスクの最大容量を超えています。インストールをフロッピー ディスクで配布する場合、2 つのオプションがあります。
  - Setup.exe を含めずにインストールを配布する。このオプションは、InstallScript MSI プロジェクトの作成時には使用できません。
  - 2 つ目のオプションは、フロッピーディスク以外のメディアでセットアップを配布することです。
- **インストールのすべてのトランスフォームも最初のディスクに置く必要があります。**たとえば、複数のランタイム言語を使用するインストールを作成する場合には、それぞれの言語に対して .mst ファイルが作成されます。このファイルはインストール データベースに適用され、それによって [言語選択] ダイアログで選択された言語を提供します。
- **インストールに含まれる再配布可能ファイル (マージ モジュールを含む) は、最初のディスク上に置く必要があります。**再配布可能ファイルは、スクリプトのビルド時にインストール データベースへストリームされます。このプロセスは、再配布可能ファイルを別のディスクに配置した場合は完了できません。

カスタム ディスク分割

リリース ウィザードを使うと、ディスクの分割に関する設定を行うパネルが表示されます。必要に応じて、ウィザードによって自動的にディスクでインストールを分割するか、ディスク分割を手動で定義することもできます。各ファイルの配置場所を指定することはできませんが、各機能が存在するディスクを指定することはできます。コンポーネントおよびファイルは、機能と同じディスクに自動的に配置されます。自分でレイアウトするかウィザードで自動的に行うかに関わらず、InstallShield が自動的に従うガイドラインが他にもあります。

各ディスクに配置する機能の定義に加え、各ディスクのボリュームラベルをカスタマイズし、ディスク プロンプトを作成して、ウィザードでディスクのサイズを強制するよう選択することができます。ディスクのレイアウトをカスタマイズする手順については、リリース ウィザードのドキュメントで説明しています。

ディスクのプロンプト

ディスクのプロンプトは、インストールを続行するために別のディスクをドライブに挿入する必要があるときに、エンドユーザーに表示されるメッセージです。標準のディスクのプロンプトがありますが、製品に合わせてプロンプトを作成することもできます。

文字列は、実際には、以下のようにプロジェクト全体の多数のソースで表示されます。

1. 終了プロンプトは、ダイレクト エディターの Error テーブルにある error 1302 として発信されます。
2. この値はプロジェクトの文字列エントリー覧からのもので、日本語のデフォルト値は、「次のディスクを挿入してください: *[2]*」です。
3. Windows Installer は、DiskPrompt プロパティの値で *[2]* を解決します。この値はプロパティ マネージャーで *[1]* に設定されています。
4. 最後に、Windows Installer は、"ディスクのプロンプト" フィールドに入力した文字列に対して、*[1]* を評価します。

ほとんどの場合、ディスク ボリュームと同じ名前を入力します。たとえば、デフォルト値が変更しなかったと想定して、**Disk8** のディスク プロンプトを入力すると、ユーザーに「次のディスクを挿入してください : Disk8」というプロンプトが表示されます。

ディスク番号の表示は、疑問符で代用させることもできます。この疑問符は、自動的に正しいディスク番号に置き換えられます。エンドユーザーに対して「次のディスクを挿入してください: DISK4」とプロンプトする場合、4 番目のディスク名に **DISK?** と入力します。ディスクのプロンプトは、インストールの各ディスクに対して指定する必要があります。

ボリューム ラベル

ボリューム ラベルは、インストールが配置されているディスクの名前です。インストールで新しいディスクが要求されると、Windows Installer エンジンによってディスクのボリューム ラベルがインストール ータベースのボリューム ラベルと一致するかどうかが確認されます。ラベルが一致しない場合、サービスは、異なるディスクがドライブに挿入されていると解釈します。ディスク分割オプションをカスタマイズする際に、ボリュームラベルを変更できます。デフォルトのボリュームラベルは、DISK1、DISK2、DISK3 などです。インストールをビルドする際、これらのディレクトリが出力場所に作成されていることを確認できます。インストールを保存したディスクのボリューム ラベルは、ウィザードで指定された名前と同じである必要があります。



***注意** ・インストールがビルドされた後にボリューム ラベルを変更することはできません。正しいボリューム名を指定しなかった場合、Windows Installer サービスはディスクを認識できず、インストールに失敗します。*

ディスク番号の指定は、疑問符で代用させることができます。たとえば、ディスクのボリューム ラベルを **InstallShield Disk 1** や **InstallShield Disk 2** などと設定する場合、すべてのディスクのボリューム ラベルを **InstallShield Disk ?** と変更しておきます。この疑問符は、ビルド時にディスク番号に置き直されます。

ディスク サイズを強制する

"ディスクサイズの強制" オプションを使って、現在のレイアウトが指定したディスクに確実に合うようにできます。このオプションが選択されておらず、作成するディスクが選択したメディアのサイズを超えた場合、ウィザードにより自動的に別のディスクが作成され、機能が分割されます。このオプションを選択した場合にディスクサイズがメディアサイズを超えると、ビルドは失敗し、ビルドエラー 1531 が発生します。その後、機能のサイズを小さくするか、別のレイアウトを試すことができます。

追加情報

インストールのディスク分割を指定し終わってから、インストールに機能を追加すると、これらの機能はインストールの最後のディスクに自動的に追加されます。リリース ウィザードを使用して、いつでもインストールに新しいディスクを追加できます。

セットアップランチャーの作成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

InstallScript インストールは、常にセットアップランチャーを含みます。

アドバンスト UI およびスイート/アドバンスト UI インストールでは、常にセットアップ ランチャーが作成されます。セットアップ ランチャーによって、.msi、.msp、InstallScript、および .exe パッケージが実行されます。詳細については、「アドバンスト UI およびスイート/アドバンスト UI インストールの作成」を参照してください。

InstallShieldでは、インストールに Setup.exe セットアップ ランチャーを含めるかどうかを指定できます。Setup.exe セットアップ ランチャーは、次のような場合に必要です:

- 必要に応じて、自動的にターゲット システムの Windows Installer エンジンを更新またはインストールする。
- 複数言語のインストール プロジェクトをビルドしていて、[言語の選択] ダイアログを表示する場合。
- プロジェクトに InstallShield 前提条件が含まれている場合。
- プロジェクトに .NET Framework が含まれている場合。
- InstallScript MSI プロジェクトのリリースをビルドしていて、ユーザー インターフェイスに従来のスタイルを用いる場合。
- 製品構成に製品の複数インスタンスをインストールするサポートが含まれているリリースをビルドして、かつインスタンスの選択ダイアログを適宜表示する場合。
- ビルボードを含む基本の MSI プロジェクトのリリースをビルドする場合。

Setup.exe セットアップ ランチャーは、上記のシナリオを制御するブートストラップ アプリケーションです。

[リリース] ビューにあるリリースの Setup.exe タブでは、Setup.exe ランチャーを使用するかどうかなどの情報を指定することができます。詳細については、「リリースの [Setup.exe] タブ」を参照してください。



ヒント: セットアップランチャーの要件は、リリース ウィザードの [セットアップランチャー] パネルでも指定できます。

Windows Installer と Setup.exe

Windows Installer がターゲット システムに存在しない可能性がある、または、インストールが特定のバージョンの Windows Installer でのみ使用可能な機能に依存している可能性がある場合、InstallShield では、Windows Installer をインストールする再配布可能ファイルをインストールに含めるオプションが提供されます。このオプションを選択すると、InstallShield はターゲット システム上で Windows Installer の存在を確認する Setup.exe ランチャーを作成します。Windows Installer がインストールされていない場合や、または新しいバージョンをインストールする必要がある場合、Setup.exe は Windows Installer インストールを起動してからインストール パッケージを起動します。

詳細については、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

複数言語のサポートと Setup.exe

[リリース] ビューの [ビルド] タブで選択した言語関連の設定に応じて、Setup.exe を実行するエンドユーザーはインストールを実行する言語を選択することができます。

複数の言語でインストールを配布する際に、実行するインストールの言語バージョンの選択オプションをエンドユーザーに提供する場合、Setup.exe が常に必要になります。複数言語のサポートに関する詳しい情報は、「複数言語インストールの作成」をご覧ください。

InstallShield 前提条件と Setup.exe

InstallShield 前提条件を含むプロジェクトでは、Setup.exe によってターゲット システムが InstallShield 前提条件を満たしているかどうかが確認されるため、Setup.exe が必須になります。条件が満たされた場合、Setup.exe は InstallShield 前提条件をインストールします。詳細については、「インストール プロジェクトに含まれている InstallShield 前提条件を利用する」を参照してください。

.NET Framework と Setup.exe

.NET Framework を含むプロジェクトでは、Setup.exe がターゲット システムを検索して .NET Framework が存在するかどうかを確認するため、Setup.exe が必要です。適切なバージョンの .NET Framework が存在しない場合、Setup.exe によってそのバージョンがインストールされます。

.NET Framework の追加については、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。

InstallScript MSI プロジェクトと Setup.exe

従来のスタイルの InstallScript MSI インストールでは、Setup.exe が InstallScript エンジンを開始してユーザー インターフェイスを表示するため、Setup.exe が必須になります。この従来のスタイルでは、InstallScript エンジンが外部 UI ハンドラーの役割を果します。

InstallScript MSI インストールで新しいスタイルの UI を使用すると、InstallShield が InstallScript エンジンを .msi パッケージ内部に埋め込むため、セットアップランチャーが必須ではありません。

2 つのスタイルに関する詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

複数インスタンスのサポートと Setup.exe

製品を構成して、エンドユーザーがターゲット システムで製品を各コンテキストで複数回インストールできるようにする場合、インスタンスの選択ダイアログを適宜表示するには、Setup.exe セットアップ ランチャーが必ず必要になります。Setup.exe ファイルがインスタンスの選択ダイアログを表示します。

複数インスタンスのサポートに関する詳しい情報は、「製品の複数のインスタンスをインストールする」をご覧ください。

ビルボードと Setup.exe

ビルボードは、実行時に Setup.exe によって表示されるため、ビルボードを含む基本の MSI プロジェクトには、Setup.exe が必要です。

ビルボードに関する詳細については、「基本の MSI インストールでビルボードを表示する」を参照してください。

セットアップランチャーのファイルのプロパティをカスタマイズする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

プロジェクトによって異なる情報がある場合は、その内容が記載されています。

InstallShield では、Setup.exe セットアップ起動プログラムのバージョン リソースにカスタム情報を使用できます。情報が、セットアップ起動プログラムの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンドユーザーが Setup.exe ファイルが右クリックして、[プロパティ] をクリックしたときに表示されます。

Setup.exe のプロパティを構成するための InstallShield の設定

次のテーブルは、Windows の [前提条件] ダイアログ ボックスに含まれている様々なプロパティと、それらを構成するのに使用できる InstallShield で対応する設定の一覧です。



メモ: *[プロパティ] ダイアログ ボックスは、Windows のバージョンによって異なります。たとえば、Windows 7 システムでは、バージョン リソース情報は [プロパティ] ダイアログ ボックスの [詳細] タブに表示されます。一方、Windows XP システムでは、バージョン リソース情報は、同じダイアログボックスの [バージョン] タブに表示されます。*

また、Windows の一部のバージョンは、[プロパティ] ダイアログ ボックスの一部の設定を表示しません。

テーブル 24 • Setup.exe のプロパティの情報ソース

Setup.exe プロパティ	Setup.exe プロパティを構成するための InstallShield 設定
会社名	このフィールドの値を含む設定は、使用するプロジェクトの種類によって異なります。 • 基本の MSI および InstallScript MSI プロジェクトでは、[一般情報] ビューの "発行者" 設定です。 • InstallScript プロジェクトでは、[一般情報] ビューの "会社名" 設定です。 • アドバンスト UI またはスイート/アドバンスト UI プロジェクトでは、InstallShield で提供されたカスタム値、またはデフォルトの InstallShield 固有の名前が使用されます。 InstallShield を使って、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの "会社名" フィールドに、独自のカスタム値を使用するには、以下の手順に従います: 1. **[リリース]** ビューで、構成するリリースを選択します。 2. **"カスタム バージョンのプロパティを使用する"** 設定での **Setup.exe** タブで **[はい]** を選択します。 3. **"会社名"** 設定に、Setup.exe ファイルのプロパティ内の "会社名" フィールドに使用するテキストを入力します。 "カスタム バージョンのプロパティを使用する" 設定に [いいえ] を選択するか、"会社名" 設定を空白のままにすると、InstallShield は [一般情報] ビュー "発行者" 設定を使用します。この設定が空白の場合、InstallShield はデフォルトの InstallShield 固有の名前を使用します。
製品名	このフィールドの値を含む設定は、使用するプロジェクトの種類によって異なります。 • 基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合、[リリース] ビューにある製品構成の "製品名" 設定です。設定が空白の場合、InstallShield は [一般情報] ビューの "製品名" 設定を使います。 • アドバンスト UI、InstallScript、およびスイート/アドバンスト UI プロジェクトでは、[一般情報] ビューの "製品名" 設定です。

テーブル 24 • Setup.exe のプロパティの情報ソース (続き.)

Setup.exe プロパティ	Setup.exe プロパティを構成するための InstallShield 設定
製品バージョン	このフィールドの値を含む設定は、使用するプロジェクトの種類によって異なります。 • 基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合、[リリース] ビューにある製品構成の "製品バージョン" 設定です。設定が空白の場合、InstallShield は [一般情報] ビューの "製品バージョン" 設定を使います。 • InstallScript プロジェクトでは、[一般情報] ビューの "製品バージョン" 設定です。 • アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、[一般情報] ビューの "製品名" 設定です。この値は、[リリース] ビューを使用して変更することができます。 アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、[一般情報] ビューの値をオーバーライドするには、以下の手順に従います: 1. **[リリース]** ビューで、構成するリリースを選択します。 2. **"カスタム バージョンのプロパティを使用する"** 設定での **Setup.exe** タブで **[はい]** を選択します。 3. **"バージョン"** 設定に、Setup.exe ファイルのプロパティ内の "バージョン" フィールドに使用するテキストを入力します。 "カスタム バージョンのプロパティを使用する" 設定に [いいえ] を選択するか、"バージョン" 設定を空白のままにすると、InstallShield は [一般情報] ビューの "バージョン" 設定を使用します。この設定が空白の場合、InstallShield はデフォルトの InstallShield 固有の値を使用します。

テーブル 24 • Setup.exe のプロパティの情報ソース (続き.)

Setup.exe プロパティ	Setup.exe プロパティを構成するための InstallShield 設定
ファイル バージョン	このフィールドの値を含む設定は、使用するプロジェクトの種類によって異なります。 • 基本の MSI、InstallScript MSI、InstallScript: **[リリース]** ビューの **[カスタム バージョンのプロパティを使用する]** の下にある Setup.exe の **"ファイル バージョン"** 設定 を使用します。 この設定が空白の場合、InstallShield はデフォルトで "製品バージョン" の値を使用します。  ***メモ:*** *ファイル バージョンは、常に 4 つのフィールドが含まれます。"ファイルバージョン" に 4 フィールドよりも少ないフィールドを指定すると、残りのフィールドには 0 が挿入されます。たとえば、ファイル バージョンとして 1.1 を指定すると、Setup.exe のバージョン リソースで使用されるファイル バージョンは 1.1.0.0 となります。* **"ファイル バージョン"** を上書きする場合、**"カスタム バージョンのプロパティを使用する"** を **[はい]** に設定して、**"ファイル バージョン"** フィールドにバージョンを入力してください。 1. **[リリース]** ビューで、構成するリリースを選択します。 2. **Setup.exe** タブで **"カスタム バージョンのプロパティを使用する"** 設定に **[はい]** を選択します。 3. **"ファイル バージョン"** フィールドに、Setup.exe ファイルのプロパティ内の "ファイル バージョン" フィールドに使用するテキストを入力します。 "カスタム バージョンのプロパティを使用する" 設定に [いいえ] を選択した場合、または "ファイル バージョン" 設定を空白のままに残した場合、InstallShield は基本の MSI および InstallScript MSI の場合は **[製品の構成]** ビューから、および InstallScript の場合は **[一般情報]** ビューから製品バージョンの "ファイル バージョン" 設定を使用します。この設定が空白の場合、はデフォルトの InstallShield 固有の値を使用します。

テーブル 24 • Setup.exe のプロパティの情報ソース (続き.)

Setup.exe プロパティ	Setup.exe プロパティを構成するための InstallShield 設定
著作権情報	InstallShield は、提供されたカスタム値、またはデフォルトの InstallShield 著作権情報を使用します。 "著作権情報" フィールドの値に独自の値を使用するには、以下の手順に従います: 1. **[リリース]** ビューで、構成するリリースを選択します。 2. **"カスタム バージョンのプロパティを使用する"** 設定での **Setup.exe** タブで **[はい]** を選択します。 3. **" ランチャーの著作権"** 設定に、Setup.exe ファイルのプロパティ内の "著作権" フィールドに使用するテキストを入力します。 "カスタム バージョンのプロパティを使用する" 設定に [いいえ] を選択するか、"起動ツールの著作権" 設定を空白のままにすると、InstallShield はデフォルトの InstallShield 著作権情報を使用します。
ファイルの説明	InstallShield は、提供されたカスタム値またはデフォルトの InstallShield Setup.exe の説明を使用します。 "説明" フィールドの値に独自の値を使用するには、以下の手順に従います: 1. **[リリース]** ビューで、構成するリリースを選択します。 2. **"カスタム バージョンのプロパティを使用する"** 設定での **Setup.exe** タブで **[はい]** を選択します。 3. **"ファイルの説明"** 設定に、Setup.exe ファイルのプロパティ内の "説明" フィールドに使用するテキストを入力します。  ***プロジェクト:*** *アドバンスト UI、InstallScript またはスイート/アドバンスト UI プロジェクトの場合: "カスタム バージョンのプロパティを使用する" 設定に [いいえ] を選択するか、"ファイルの説明" 設定を空白のままにすると、InstallShield はデフォルトの InstallShield Setup.exe の説明を使用します。* *基本の MSI または InstallScript MSI プロジェクトの場合: "カスタム バージョンのプロパティを使用する" 設定に [いいえ] を選択すると、InstallShield はデフォルトの InstallShield Setup.exe の説明を使用します。[はい] を選択した場合に "ファイルの説明" 設定を空白のままにすると、InstallShield は [リリース] ビュー内にある製品構成の "コメント" 設定の値を使用します。この設定が空白の場合、InstallShield は [一般情報] ビューの "[概要情報ストリーム] のコメント" 設定を使います。この設定が空白の場合、InstallShield はデフォルトの InstallShield Setup.exe の説明を使用します。*
言語	このフィールドには、[リリース] ビュー内の現在ビルド中のリリースの "デフォルト言語" 設定で指定された言語が使用されます。

テーブル 24 • Setup.exe のプロパティの情報ソース (続き.)

Setup.exe プロパティ	Setup.exe プロパティを構成するための InstallShield 設定
元のファイル名	InstallShield は、提供されたカスタム値、またはデフォルトの InstallShield 名 setup.exe のどちらかを使用します。 ″元のファイル名″ フィールドの値に独自の値を使用するには、以下の手順に従います: 1. [メディア] の下にあるビュー リストで、**[リリース]** をクリックします。 2. **[リリース]** エクスプローラーを右クリックし、**[新しい製品構成]** をクリックします。 リリース エクスプローラーに、新しい製品構成が追加されます。 InstallShield の [全般] タブに製品構成の設定が一覧表示されます。 3. **″セットアップ ファイル名″** 設定の **[全般]** タブに、″元のファイル名″ フィールドに使用するテキストを入力します。 InstallShield がビルド時に生成する元のファイル名に使用するファイル名 (.exe ファイル拡張子を除く) を指定します。

Setup.exe のプロパティ ダイアログ ボックスのサンプル

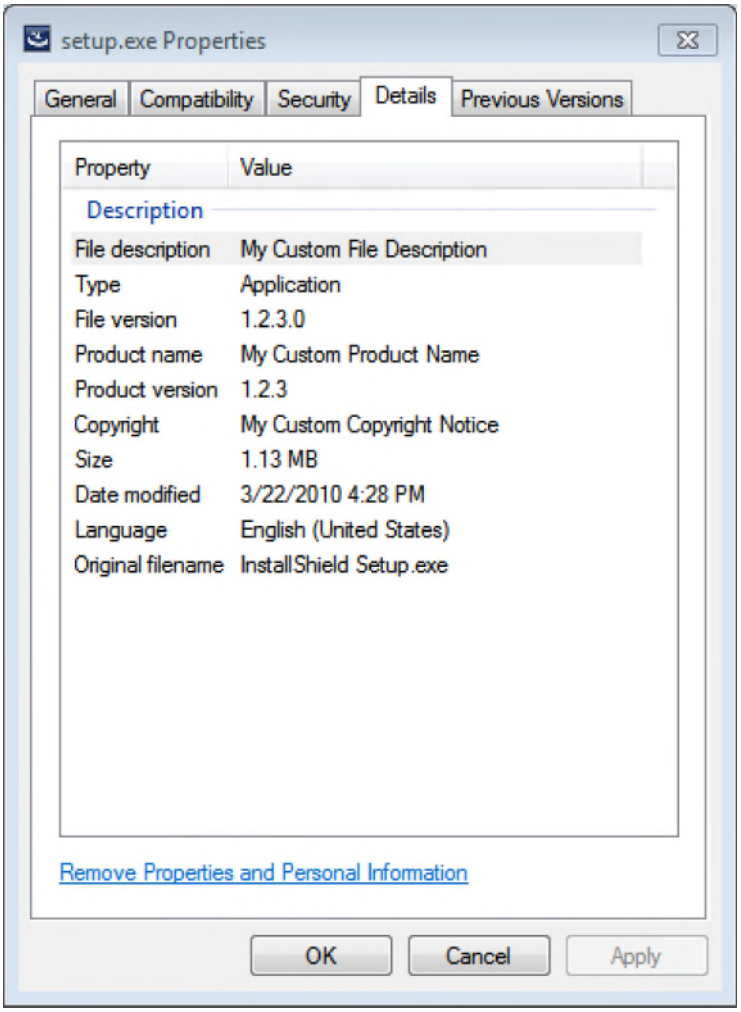


図 1: Windows 7 マシン上における Setup.exe のプロパティのサンプル

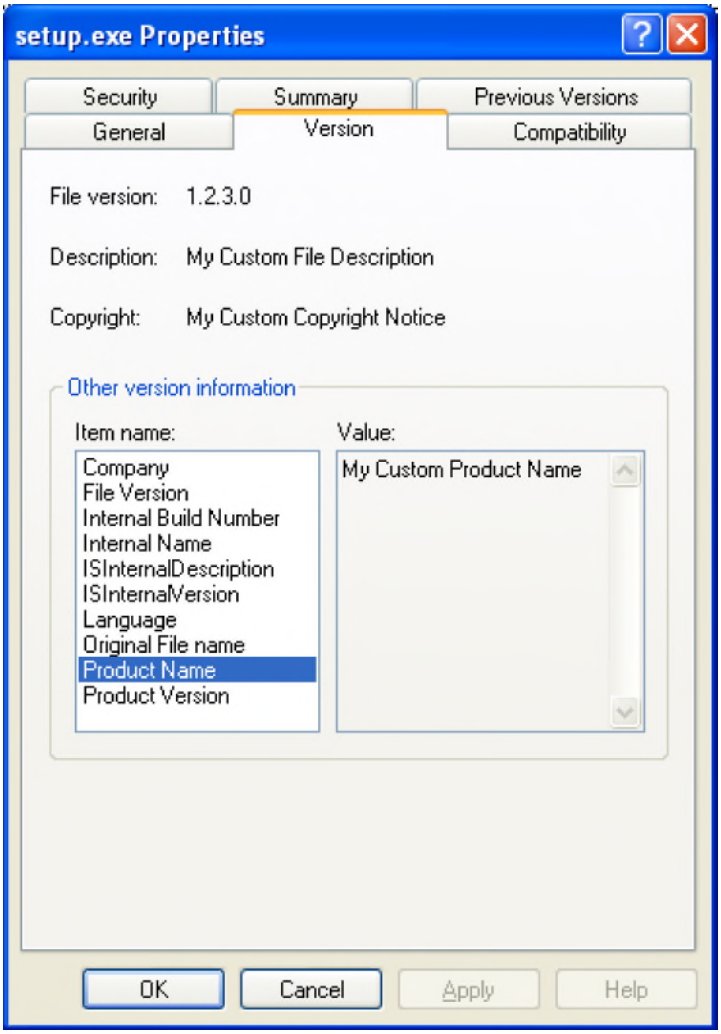


図 2: Windows XP マシン上における Setup.exe のプロパティのサンプル

セットアップランチャーのファイルのアイコンを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

InstallShield では、Setup.exe セットアップ ランチャーに使用するアイコンを指定できます。アイコンには、.exe、.dll、または .ico ファイルを使用できます。

エンド ユーザーが Windows Explorer で Setup.exe ファイルを参照したときにも、このアイコンが表示されます。アイコンはまた、Setup.exe の [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Setup.exe ファイルを右クリックして、[プロパティ] をクリックしたときに表示されます。

アイコンを指定しなかった場合、InstallShield は Setup.exe ファイルのデフォルト アイコンを使用します。



タスク ***Setup.exe ファイルのアイコンを指定するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、構成するリリースを選択します。
3. **Setup.exe** タブをクリックします。
4. [**Setup.exe アイコン ファイル**] 設定には、InstallShield がビルド時に Setup.exe ファイルを作成するときに使用するアイコンを含むファイルの完全修飾名を指定します。

   ファイルを指定するには、明示的パスまたはパス変数を入力するか、省略記号ボタン (...) をクリックして **[アイコンの変更]** ダイアログ ボックスを開いて **[参照]** ボタンをクリックするとファイルを選択できます。

   デフォルトではインデックス 0 のアイコンが使われます。別のアイコンを指定するには、**[アイコンの変更]** ダイアログ ボックスでアイコンを選択するか、アイコンのインデックスまたはリソース ID (それぞれ前にマイナス (-) 記号がついている) をファイル名に付け足します。たとえば、**C:¥Temp¥MyLibrary.dll,2** は 2 というインデックスを持つアイコンを、**C:¥Temp¥MyLibrary.dll,-100** は 100 と言うリソース ID を持つアイコンを示します。

リリースを次回ビルドするとき、InstallShield が Setup.exe ファイルに指定されたアイコンを使用します。

インストールを配布する

インストールを作成したあと、場合により、指定した場所へ配布する必要があります。これにはネットワーク ドライブ、CD、ローカル ドライブの別の場所、または FTP サイトが可能です。インストールを配布すると、インストールをビルドしたときに作成されたディスク イメージが、指定の場所にコピーされます。

フォルダーまたは FTP サイトにリリースを自動的に配布する

リリースのビルドとテストが完了すると、残る作業は、それを適切な場所に配布するのみです。リリースを適切な場所に手動でコピーすることもできますし、[リリース] ビューの [イベント] タブを使って、InstallShield がリリースを適切な場所 (ローカル/ネットワークの場所、または FTP サイト) へ自動的にコピーするように構成することもできます。



タスク ***InstallShield を構成して、リリースを特定の場所へ自動的に配布するようにするには、以下の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、構成するリリースを選択します。
3. **[イベント]** タブをクリックします。

4. 設定を適切に構成します。**[イベント]** タブにある設定に関する詳しい情報は、「リリースの [イベント] タブ」を参照してください。



メモ: *インストールが 1 つのディスクのみで構成される場合は、リリースの保存場所に Disk1 フォルダーの内容がコピーされますが、フォルダーそのものはコピーされません。インストールが複数のディスクで構成される場合は、フォルダーおよびその内容がリリースの保存場所にコピーされます。*



プロジェクト: *InstallScript および InstallScript Object プロジェクトの場合、リリースがビルドされるたびに、InstallShield が、自動的にそのリリースを [イベント] タブで指定した場所にコピーします。*

その他のプロジェクトの種類の場合、[リリース] ビューでリリースを右クリックして、[配布] をクリックすると、リリースに関連するファイルがすべて指定された場所にコピーされます。

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

[イベント] タブでフォルダーおよび FTP サイトを指定した場合、ファイルは FTP ロケーションにのみコピーされます。

基本の MSI、InstallScript MSI、またはマージ モジュール プロジェクトで、ビルドするたびに、ビルド エンジンがリリースを指定の場所にコピーするように設定する場合、"ビルド後、配布する" 設定で [はい] を選択します。

ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する



エディション: *仮想マシンへのリリース配布機能は、InstallShield Premier Edition でご利用いただけます。*



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

インストールのビルドが成功するたびに InstallShield が仮想マシン (VM) を指定のスナップショットに戻し (指定されている場合)、VM の電源をオンにしてインストールを VM にコピーし、テストが可能な状態になるようプロジェクトのリリースを構成できます。また、これらのテスト準備を必要なときにオンデマンドで行うこともできます。このテスト準備機能により、手作業で行う処理を減らし、テストにかかる時間を短縮できます。

VM は Microsoft Hyper-V Server、VMware ESX または ESXi Server、または VMware Workstation を指定できます。

VM に配布するための要件

サポートされている VM の1 つに配布する場合、InstallShield では必要な仮想化テクノロジとの対話が必要です。VMWare 仮想化テクノロジ (VMware ESX、ESXi Server oまたは VMware Workstation) を使用する場合、InstallShield と同じマシン上に VMware VIX API が インストールされていなくてはなりません。マシン上に VMware Workstation をインストールするか、http://www.vmware.com/support/developer/vix-api から VMware VIX API をダウンロードおよびインストールすることができます。

VMware Workstation を使用する場合、InstallShield と同じマシン上に VMware Workstation をインストールすることが推奨されます。これによって、InstallShield が特定の VMware Workstation バージョン用に設計された VIX API のバージョンを使用します。VIX API の新しいバージョンも使用できる場合がありますが、現在使っている VMware Workstation にバンドルされている VIX API のバージョンを InstallShield で使用する方法が最も推奨されます。

VM 関連の設定を構成する

タスク

ビルド時またはオンデマンドで VM に配布できるよう、リリースの VM 関連設定を構成するには、以下の手順に従います:

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、構成するリリースを選択します。
3. **[イベント]** タブをクリックします。
4. **[仮想マシン]** 領域を展開します。
5. ***"構成"*** 設定を使って、選択されたリリースを VM に配布するときに使用する VM 構成設定グループの名前を選択します。
   - 新しい VM 構成を作成するには、この設定で省略記号ボタン (...) をクリックします。
   - 既存の VM 構成を編集するには、その構成を選択してから省略記号ボタン (...) をクリックします。

   どちらの場合も **[仮想マシンの構成を編集]** ダイアログが開いて、マシン全体に適用する VM の詳細を構成することができます。このダイアログ ボックスの使い方については、「[仮想マシンの構成を編集] ダイアログ ボックス」を参照してください。

   [仮想マシンの構成を編集] ダイアログ ボックスにある "VM 構成" 設定を指定すると、InstallShield によって指定されたデータが VMConfigurations.xml という名前のファイルに書き込まれます。この VMConfigurations.xml ファイルは、マシン全体に適用するファイルで、一度構成を行うと他の InstallShield プロジェクトでも使用できます。また、他のチームメンバーと共有することも可能です。また、このファイルをビルド マシンで使用することも可能です。詳細については、「リリースの配布用の仮想マシン設定を共有する」を参照してください。
6. 必要に応じて、**仮想マシン**の残りの設定を構成します。

   これらの残りの設定を使って VM の詳細を構成すると、[仮想マシンの構成を編集] ダイアログ ボックスで入力済みの、マシン全体に適用される値が上書きされます。

   [仮想マシン] 領域にある各設定についての詳細は、「リリースの [イベント] タブ」を参照してください。

ヒント: *ビルドが成功するたびに指定された VMにリリースを配布するには、"ビルド後に配布" 設定を [はい] に選択します。*

オンデマンドでビルド済みのリリースをステージングする場合、リリースを右クリックしてから [VM に配布] をクリックします。

インターネットで配布するインストールを準備する

ユーザーがソフトウェアを受け取る方法は急激に変化しています。インターネット テクノロジがそれほど発展していなく、高速のインターネット接続が導入される以前は、すべてのソフトウェアが、フロッピー ディスクや CD-ROM などのリムーバブル メディアで発送されていました。ソフトウェアは現在、インターネットから直接ダウンロードされることがほとんどです。この時間およびコストを節約できるソフトウェア配布プロセスを利用するため、より簡単にダウンロードおよびインストールできる方法で、インストールをパッケージする必要があります。

Web 用のインストールには、満たす必要のある条件がいくつかあります。

圧縮サイズ

現在、多くの人が、高速ケーブル モデムや DSL 回線を使用してインターネットに接続していますが、低速度モデムを使用している人も多くいます。アプリケーションのダウンロードに必要なオンライン時間が増加するため、パッケージ サイズは、低速の接続を使用している人にとって非常に重要です。

自己展開

多くのファイル圧縮ユーティリティは、アプリケーション ファイルを解凍するための特別なクライアント側アプリケーションが必要です。別のユーティリティでは、このような必要性のために、エンドユーザーのダウンロードやインストール プロセスが複雑になります。インストール プロセスをシンプルにするには、使用する圧縮ユーティリティが他のアプリケーションを必要としないように、自己展開できる必要があります。

デジタル署名

顧客がソフトウェアをダウンロードおよびインストールするときの安全を確保するため、アプリケーションパッケージをデジタルで署名できます。デジタル署名により、ソフトウェア作成者または企業がエンドユーザーに識別され、発行以来アプリケーション コードが変更または改ざんされていないことが保証されます。アプリケーションにデジタル署名を付加する詳しい方法については、「デジタル署名とセキュリティ」を参照してください。

使いやすさ

インターネット配布のインストールのパッケージにおいて最も重要なことは、使いやすさです。すべてのエンドユーザーが、インストール ファイルを保存する場所を指定したり、コンピューターを検索してこれらのファイルを自分で探すことは望んでいるとは限りません。インストールを圧縮パッケージにシームレスに統合し、必要なインストール開始手順を 1 つだけにする必要があります。

プロキシ サーバーのサポート

特定のファイルがターゲット システム上で必要な場合のみ、インストールがそのファイルをダウンロードするように構成できます。たとえば、Windows Installer エンジン、.NET Framework、および一部の InstallShield 前提条件が、一部またはほとんどのターゲット システム上に既存する可能性があります。これらのファイルをインストールに埋め込む代わりに、必要なファイルだけを実行時にダウンロードするようにプロジェクトを構成することができます。こうすることで、インストール全体のサイズを抑えることができます。

エンド ユーザーがプロキシ サーバーを使ってインターネットにアクセスする場合で、インストールがファイルをダウンロードするように構成されているとき、インストールはダウンロード中に、Internet Explorer で手動で構成されたシステム プロキシ設定を使用します。これは、ターゲット システム上で別のブラウザーがデフォルトとして設定されている場合でも同じです。

InstallShield は、Internet Explorer の "設定を自動的に検出する" 設定をサポートしませんので、ご注意ください。(エンドユーザーが使用している Internet Explorer で、LAN 接続に対して [設定を自動的に検出する] チェック ボックスが選択されているときに、インストールでファイルのダウンロードが必要な場合、ファイルのダウンロードができないため、インストールは失敗します。エンドユーザーが使用している Internet Explorer で、LAN 接続に対して [設定を自動的に検出する] チェック ボックスが選択されている可能性があるとき、ダウンロードされるように構成する代わりに、すべてのファイルをインストールに埋めこんだ方が良い場合があります。ファイルが埋めこまれている場合、失敗は避けられます。)ただし、InstallShield は Internet Explorer の LAN 接続用にセットアップされた自動構成スクリプト機能をサポートします。

InstallScript インストールと共に配布される再配布可能ファイル

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

次のファイルは、InstallShield によって自動的に作成され、(エンドユーザー使用許諾契約で述べられているように) ソフトウェア配布と共に配布することが可能です。メディアのビルドは、ディスク イメージ フォルダーに必要なため、自動的に次のファイルを含みます。

メイン メディア ファイル

- data<2, 3, 4, ⋯>.cab
- data1.cab
- data1.hdr
- ISSetup.dll
- layout.bin
- setup.exe
- setup.ini
- setup.inx

オプション ファイル

- setup.isn—スキン ファイル
- setup.isn—Skin Filesetup.htm—One Click Install .htm ページ

64 ビット セットアップ ランチャーのビルド



プロジェクト- *この情報は、基本の MSI プロジェクトに適用します。*

InstallShieldでは、インストールに 64 ビット**Setup.exe** セットアップ ランチャーを含めるかどうかを指定できます。64 ビット **Setup.exe** セットアップ ランチャーは、インストールが 64 ビット システムをターゲットとする場合のみ必要です。デフォルトで、InstallShield は 32 ビット システムだけでなく 64 ビットシステム上の WOW64 上で実行可能な 32 ビット セットアップ ランチャーをビルドします。一部の 64 ビット ターゲット システム、たとえば Windows Server Core システム は、WOW64 (32-bit Windows-on-Windows) をサポートしない場合があります。これらの 64 ビット ターゲット システムは 32 ビット セットアップランチャーを実行できません。

[リリース] ビューにある製品構成の **全般** タブでは、64 ビット セットアップ ランチャーをビルドするかどうかなどの情報を指定することができます。

64 ビット セットアップ ランチャーをビルドしたとき、対応する 64 ビット MSI パッケージがデフォルトでビルドされることはありません。64 ビット MSI パッケージをビルドするには、適切な 64 ビット値 (x64 または Intel64) を使って ″テンプレート概要″ を構成する必要があります。ピュア 64 ビット MSI パッケージは、インストール中に 32 ビット製品ファイルをコピーまたは 32 ビット カスタム アクションを実行することができません。これは、32 ビット バイナリを WOW64 サポートなしで 64 ビット ターゲット システム上にロードすることができないためです。**[リリース]** ビューにある製品構成の ”InstallShield アーキテクチャの検証” 設定を使って、ピュア 32 ビットまたは 64 ビット MSI パッケージをビルドすることができます。InstallShield では、リリース フラグの構成に基づいて、単一プロジェクトから 2 つのインストーラー (32 ビットおよび 64 ビット) を作成することができます。

64 ビット MSI パッケージのビルドについての詳細は、「64 ビットのオペレーティング システムを基本の MSI および InstallScript MSI インストールを使ってターゲットする」を参照してください。

64 ビット セットアップ ランチャーを含む 32 ビット MSI パッケージをビルドする

MSI パッケージで 64 ビット セットアップ ランチャー オプションを含む 32 ビット パッケージのビルドが構成された場合、InstallShield は次の警告メッセージを表示します:

ISDEV: 警告 -7372: セットアップ ランチャーでは 64 ビットのビルドが構成されていますが、テンプレートの概要では 64 ビット MSI パッケージのビルドが構成されていません。

これは 64 ビット セットアップ ランチャーがピュア 32 ビット システム上での実行に失敗し、32 ビット MSI パッケージがピュア 64 ビット システム上でのインストールに失敗することを通知することが目的です。この警告を回避するため、インストーラーがピュア 32 ビット システムおよび 64 ビット システム上の WOW64 をターゲットにする必要があるときは、64 ビット セットアップ ランチャーを使用しないでください。または、MSI テンプレートの概要で、ピュア 64 ビット システムをターゲットにする 64 ビット MSI パッケージを生成するように変更してください。

64 ビット セットアップ ランチャーを使用する InstallScript カスタム アクションのビルド

InstallScript カスタムアクションは、32 ビットセットアップ ランチャーに加えて、64 ビットセットアップ ランチャーでサポートされています。インストール要件に応じて、**[カスタムアクションとシーケンス] ビューの [アクション]** 領域で、**″64 ビット InstallScript を実行″** 設定を構成できます。

InstallScript カスタムアクションが 64 ビット DLL ファイルを読み込む必要がある場合、または 64 ビット ファイル システムと 64 ビットレジストリの場所にアクセス/構成する必要がある場合、**"64 ビットInstallScriptを実行する"** 設定を **[はい]** に設定することをお勧めします。これによって、InstallScript カスタム アクションが 64 ビット プロセスで実行され、64 ビットシステム リソースと正しく対話することが保証されます。

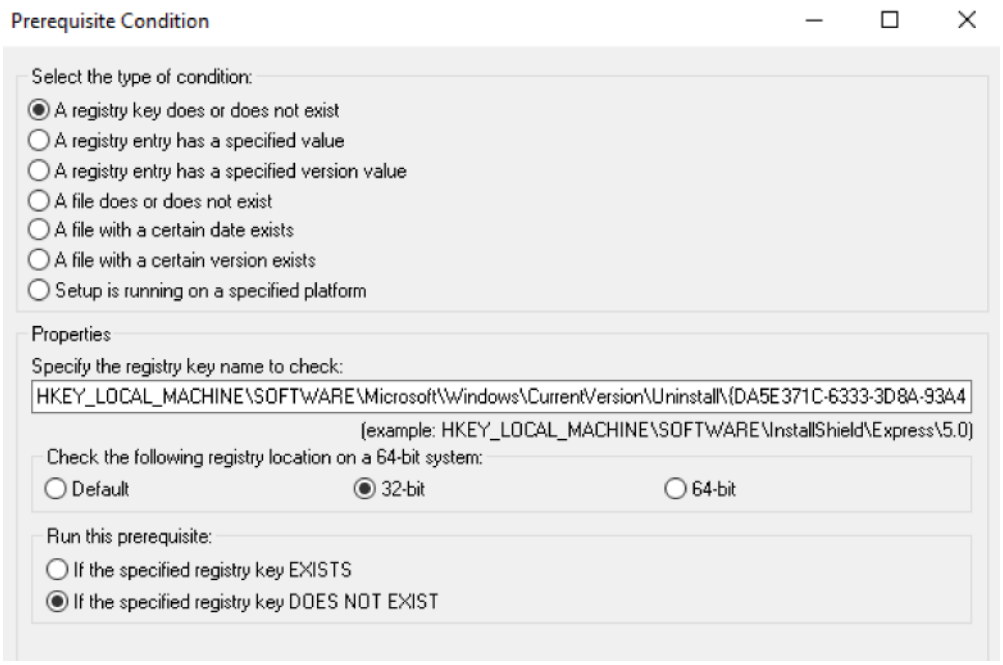
詳細については、「カスタム アクションの設定」を参照してください。

InstallShield 前提条件および 64 ビット セットアップ ランチャー

一般的に InstallShield 前提条件を含むプロジェクトでは、**Setup.exe** によってターゲット システムで InstallShield 前提条件が満たされているかどうかが確認されるため、**Setup.exe** が必須になります。条件が満たされている場合、**Setup.exe** が InstallShield 前提条件をインストールします。

64 ビット セットアップ ランチャーを使用する InstallShield 前提条件を含むプロジェクトのビルドを構成する場合、必ず前提条件のインストーラーが 64 ビットであり、ピュア x64 システムにインストールが可能で、前提条件がレジストリまたはフォルダーの場所で 64 ビットの場所を確認するように構成します。

場合によって、64 ビット セットアップ ランチャーを使う 32 ビット前提条件のインストーラーをビルドしたい場合、32 ビットの場所を確認する条件を構成します。前提条件に含まれるほとんどのレジストリ条件はデフォルトの場所を確認するように構成されています (32 ビット セットアップ ランチャーの場合は 32 ビットの場所、64 ビット セットアップ ランチャーの場合は 64 ビットの場所)。ただし、32 ビットの前提条件インストーラーは、32 ビット レジストリハイブの下にレジストリ キーを作成します。この問題を解決するためには、32 ビットの場所を確認するようにレジストリの場所を InstallShield 前提条件エディターで明示的に変更する必要があります。前提条件は次の通りです:



Windows アプリ パッケージの配布



プロジェクト・ *この情報は、基本の MSI プロジェクトに適用します。*



重要・ *デスクトップ拡張 (デスクトップ ブリッジ) を含む Windows アプリ パッケージ (.appx) のインストールおよびテストを行うには、Windows 10 Anniversary Update が必要です。Windows アプリ パッケージにデジタル署名を行う場合、InstallShield を Windows 10 または Windows 10 SDK がインストールされているマシン上にインストールする必要があります。*



ヒント・ *Windows アプリ パッケージを正しくサイドロードするためには、サイドローディングを有効化する必要があります。サイドローディングに関する詳細は、MSDN 記事「Enable your device for deployment 」を参照してください。*

次のテーブルは、Windows アプリパッケージの配布方法についての概要を説明します:

テーブル 25 ・ Windows アプリの配布方法

配布方法	説明	配布方法の設定	署名の考慮
Windows ストア	Windows アプリ パッケージを Windows ストアにアップロードします。詳細については、「Windows アプリのパッケージ化」(MSDN 記事) を参照してください。 Windows ストアを通して配布されるパッケージにはすべて、コンテンツを制限する追加ポリシーが適用されます。	Windows ストア	この種類のアプリは、パッケージが Microsoft によって署名されるため、Microsoft にアップロードする前に署名する必要がありません。
スイート/アドバンスト UI プロジェクトにサイドロードを追加	サイドローディング Windows アプリ パッケージをスイート/アドバンスト UI プロジェクトに追加する方法は「サイドローディング Windows アプリ パッケージ (.appx| msix) をスイート/アドバンスト UI プロジェクトに追加する」に説明されています。 サイドローディング アプリが良く利用されるのは、エンタープライズ環境で、会社組織内部のみでアプリが配布される場合です。 WINDOWS アプリ パッケージ (.appx) の **[Windows アプリ パッケージの対象]** 条件や、終了条件あるいはパッケージの対象条件にある **[Windows タイプが存在]** 条件といった条件の確認、およびその他の条件確認を行い、この種類の配布方法を採用します。この種類のシナリオでは、エンド ユーザーが条件付で .msi パッケージまたは Windows アプリ パッケージをインストールできる柔軟性を提供できます。詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類」を参照してください。	直接配布 (サイドロード)	パッケージがサイドロードされ、**また**スイートに追加される場合、Windows アプリ パッケージはスイート/アドバンスト UI プロジェクトにビルドされる前に署名を行う必要があります。
サイドロードおよび Windows アプリ パッケージ (.appx) としての配布	カスタマに対し、Windows アプリ パッケージ (.appx) ファイルをダブルクリックでインストール開始可能な実行可能インストーラーとして配布します。 サイドローディング アプリが良く利用されるのは、エンタープライズ環境で、会社組織内部のみでアプリが配布される場合です。	直接配布 (サイドロード)	パッケージがサイドロードされる場合、カスタマが独自に署名することを希望しない限り、一般的に署名が必要です。パッケージのダブルクリック インストールを行う前に、あらかじめ署名が必要です。

トランスフォームの作成



プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

トランスフォーム (.mst ファイル) は、2 つの .msi データベース間の差分を含むシンプルな Windows Installer データベースです。トランスフォームを使うと、管理者は、インストール パッケージを配布する際、変更した設定をデータベースに適用することができます。

インストールのカスタマイズ

たとえば、ユーザーは会社の部署ごとに異なる方法でインストールをカスタマイズしなくてはならない場合があります。通常のオフィススイートは、スプレッドシート プログラム、ワードプロセッサ、およびプレゼンテーション ツールを備えています。会計部ではスプレッドシートとプレゼンテーション プログラムだけを必要とし、書類作成部ではワードプロセッサとスプレッドシートだけを必要とするような場合があります。その他の部署にはアプリケーションの全スイートが必要となるかもしれません。

その場合各社員ごとに手作業でセットアップしなくても、ユーザーはスイート全体のオリジナルインストールを使用して各部署の必要に応じてカスタマイズし、その後 2 つのパッケージ間のトランスフォームを作成することができます。トランスフォームは、個別の製品構成についてそれぞれ作成する必要があります。

トランスフォームの作成

InstallShield では、製品の複数の構成を容易に作成できるようにウィザードがその手順をガイドします。以下の 2 つの方法のいずれかを利用してトランスフォームを作成することができます。

2 つの .msi パッケージを比較してトランスフォームを作成する

トランスフォームを作成する一つ目の手法では、ベース .msi パッケージはすべてのカスタマイズの起点としての役割を果たし、ターゲットパッケージはインストールに反映させたいすべての変更を含みます。一度ベース .msi パッケージ とターゲット .msi パッケージが準備されると、トランスフォームウイザードを使用してトランスフォームを作成することができます。ウィザードを終了すると InstallShield は 2 つのパッケージを比較し、パッケージ間の差分を含むトランスフォーム (.mst ファイル)を作成します。

単一の .msi パッケージを基にトランスフォームを作成する

別のトランスフォームの作成方法として、新規トランスフォームプロジェクトの作成があります。新規トランスフォーム プロジェクトを作成すると、トランスフォーム オープン ウィザード が起動します。トランスフォーム オープン ウィザードを使用して、ダイレクト MST モードで既存の .msi パッケージを開き必要に応じてプロジェクトを編集することができます。InstallShield は、 ベース .msi パッケージとダイレクト MST モードで加えた変更との差分を評価し、その差分を含むトランスフォーム プロジェクト (.mst ファイル) を作成します。

この手法でトランスフォームを作成すると、トランスフォーム プロジェクトに組み込む追加トランスフォーム プロジェクトを指定することができます。また、エンドユーザーへのデフォルトのユーザー インターフェイス応答も指定することができます。

トランスフォームの適用

トランスフォームの作成後は、アプリケーションがインストールされているコンピューター別に実行時にトランスフォームを適用できます。たとえば、ターゲット コンピューターが経理部門のコンピューターかどうかを検査することができます。それが経理部門にある場合、経理部用のトランスフォームが元のインストールに適用され、スプレッドシートとプレゼンテーション ツールのみがインストールされます。

2 つの .msi パッケージを比較してトランスフォームを作成する



プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*



メモ: *トランスフォーム プロジェクトが新規ファイル (オリジナル .msi ファイルには含まれないファイル) を含む場合、これらのファイルはインストール実行中には見つかりません。この問題を避けるため、トランスフォーム プロジェクトをベース .msi パッケージと同じ場所に配置するか、または手動でソース ファイルをベース .msi パッケージの場所へコピーします。*



タスク ***2 つの .msi パッケージを比較してトランスフォームを作成するには、以下の手順を実行してください。***

1. InstallShield で、ベースの .msi パッケージを開きます。
2. プロジェクトを新しい名前で保存します。この新しいプロジェクトがターゲット プロジェクトとなります。詳細については、「新しい名前と場所でプロジェクトを保存する」を参照してください。
3. インストールに反映させたい変更をすべて加え、リリースをビルドします。
4. **[ツール]** メニューで、**[トランスフォームを作成/適用]** をクリックします。トランスフォームウィザードが開きます。
5. トランスフォーム ウィザードのパネルを完成して、トランスフォームを作成します。

単一の .msi パッケージを基にトランスフォームを作成する



プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*



メモ: *トランスフォーム プロジェクトが新規ファイル (オリジナル .msi ファイルには含まれないファイル) を含む場合、これらのファイルはインストール実行中には見つかりません。この問題を避けるため、トランスフォーム プロジェクトをベース .msi パッケージと同じ場所に配置するか、または手動でソース ファイルをベース .msi パッケージの場所へコピーします。*



タスク ***単一の .msi パッケージを基にトランスフォームを作成するには、以下の手順を実行してください。***

1. **[ファイル]** メニューで、**[新規]** をクリックします。**[新規プロジェクト]** ダイアログ ボックスが開きます。
2. **[Windows Installer]** タブで **[トランスフォーム]** をクリックします。
3. **[プロジェクト名]** ボックスで、トランスフォーム プロジェクトの名前を入力します。

4. **[場所]** ボックスに、トランスフォーム プロジェクトが保存される場所へのパスを入力する、または **[参照]** をクリックして場所を探します。
5. **[OK]** をクリックします。**トランスフォーム オープン ウィザード**が開きます。
6. **トランスフォーム オープン ウィザード**のパネルを完成してトランスフォームを作成します。

トランスフォームの適用



***プロジェクト・** この情報は、トランスフォーム プロジェクトに適用します。*

トランスフォームを適用すると、ベース .msi パッケージを変更して、トランスフォーム (.mst ファイル) に含めた変更を組み込むことになります。

トランスフォームを適用する方法として 2 つの方法が以下に記述されています。

コマンドラインからのトランスフォームの適用

1 つの方法としては、コマンドラインを Msiexec.exe または Setup.exe に渡す方法があります。コマンドラインからトランスフォームを適用すると、.msi パッケージのデータベースは、実行時に変更されます。トランスフォームを実行時に適用するには、次のコマンドライン ステートメントのように、TRANSFORMS プロパティで目的のトランスフォームを指定します。

```
msiexec /i "ProductName.msi" TRANSFORMS="YourTransform.mst"
```

複数のトランスフォームはセミコロンで区切ります。



***注意 ・** Windows Installer はセミコロンをファイル名の区切り文字として解釈するため、トランスフォーム名にセミコロンを使用しないでください。*

コマンドラインで 2 つ以上のトランスフォームを指定した場合、トランスフォームは指定した順番で適用されます。たとえば、ベース パッケージに機能が 1 つだけあって、2 つ目の機能を追加する 1 つのトランスフォームと、2 つ目の機能を変更するもう 1 つのトランスフォームがあったとします。機能を追加するトランスフォームは、2 つ目の機能を変更するトランスフォームの前に指定しなければなりません。その順番で指定しない場合、変更は正しく行われません。

トランスフォームの適用にトランスフォーム ウィザードを利用する

トランスフォーム ウィザード を使ってトランスフォームを適用する際、トランスフォーム ファイルに含まれる変更すべてを反映したインストール パッケージ (.msi パッケージ) を作成します。これは、あらゆるユーザーのアプリケーションをカスタマイズしているネットワーク管理者にとって便利な方法です。



タスク** **トランスフォーム ウィザードを起動するには、次を実行します。

[ツール] メニューで、**[作成/適用]** をクリックします。

トランスフォームの編集



プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

InstallShield を使用すると、.msi パッケージを、InstallShield (.ism) プロジェクトに変換することなく編集することができます。変更点はトランスフォーム (.mst) ファイルとして保存できます。

編集するため既存するトランスフォームを開く

InstallShield で、既存の .mst ファイルを参照することができます。.mst ファイルを開く前に、InstallShield はこのファイルを適用する ベース .msi を必要とします。また、編集するために .mst ファイルを開く前にベース .msi ファイルに適用するその他の .mst ファイルの名前も必要です。

InstallShield で、.mst ファイルを初めて開いたとき、トランスフォーム オープン ウィザードが起動します。このウィザードでは、 .mst ファイル、ベース .msi ファイル、および編集の前に適用する追加トランスフォーム指定することができます。このウィザードの最後のパネル、[応答トランスフォームの作成] パネルでは、そのトランスフォームが応答トランスフォームであるかどうかを指定できます。応答トランスフォームには、インストールのユーザー インターフェイス部分のデフォルトの応答も含まれています。応答トランスフォーム作成を選択すると、インストールのユーザー インターフェイス要素が実行され、インストール ウィザードの各パネルのオプションをカスタマイズすることができます。

トランスフォーム オープン ウィザードが完了したとき、トランスフォームが適用され、.mst ファイルが InstallShield のダイレクト MST モード (ダイレクト編集モードに似ています) で開きます。ウィザードで指定されたデータは、次回 .mst ファイルを開いたとき、 トランスフォーム プロジェクトが InstallShield で直接開くように保存されます。



タスク ***既存のトランスフォームを開くには、以下の手順を実行してください。***

1. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログが開きます。
2. 必要な .mst ファイルを選択します。
3. **[開く]** をクリックします。**トランスフォーム オープン ウィザード**が開きます。

既存のトランスフォーム ファイルを InstallShield トランスフォーム ウィザードで開く



プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

InstallShield トランスフォーム ウィザードで、既存のトランスフォーム ファイルを開くことができます。トランスフォーム ウィザードで初めて開いたときと同様に、トランスフォーム ファイルのベースとなる MSI パッケージを選択するためのプロンプトが表示されます。これによって、複数の MSI パッケージに共通の汎用トランスフォーム ファイルを使用することができます。

InstallShield トランスフォーム ウィザードで既存のトランスフォーム ファイルを開くには、Windows エクスプローラーでトランスフォーム ファイルを右クリックしてから、コンテキスト メニューから **[InstallShield トランスフォーム ウィザードで開く]** を選択します。

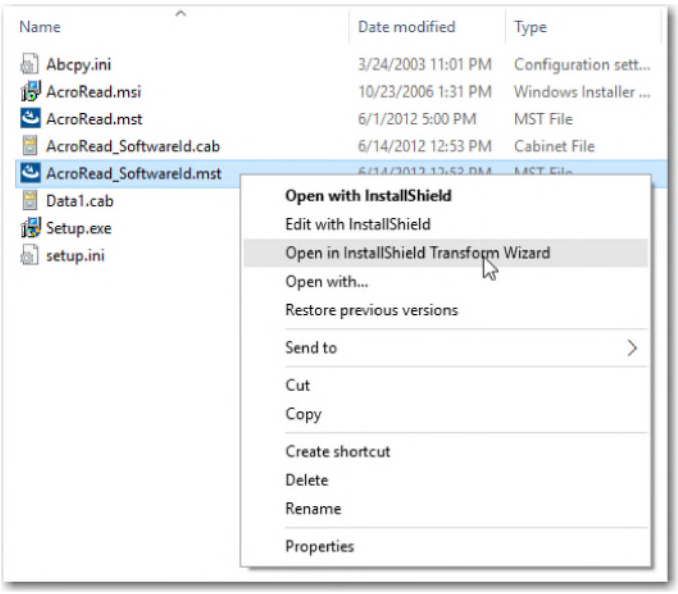


図 1: [InstallShield トランスフォーム ウィザードで開く] オプション

.msi ファイルをトランスフォームとして保存する

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

ダイレクト編集モードで .msi ファイルを編集中、そのファイルを .mst ファイルとして保存することができます。

タスク ***.msi ファイルを .mst ファイルとして保存するには、以下の手順を実行します。***

1. **[ファイル]** メニューで、**[名前を付けて保存]** をクリックします。**[名前を付けて保存]** ダイアログ ボックスが開きます。
2. **[ファイルの種類]** ボックスで **[Windows Installer トランスフォーム (*.mst)]** を選択します。

最後に保存してからそれ以降に変更された部分を指定された .mst ファイルに保存します。

トランスフォームを .msi ファイルとして保存する

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

.mst ファイルを .msi ファイルとして保存することもできます。

タスク ***トランスフォームを .msi ファイルとして保存するには、以下の手順を実行します。***

1. まだトランスフォーム プロジェクトを開いていない場合、InstallShield で開きます。
2. **[ファイル]** メニューで、**[名前を付けて保存]** をクリックします。**[名前を付けて保存]** ダイアログ ボックスが開きます。
3. **[ファイルの種類]** ボックスで **[Windows Installer パッケージ (*.msi)]** を選択します。

.msi ファイルが作成されます。ベース .msi ファイルのデータすべて、編集した .mst ファイル、およびその他適用された .mst ファイルが含まれます。プロジェクトは ダイレクト モード MSI プロジェクトとして取り扱われます。詳細については、「ダイレクト編集モードで .msi/.msm データベースを編集する」を参照してください。

応答トランスフォーム

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

応答トランスフォームは、管理者がユーザー インターフェイス シーケンスを実行して希望の値を入力するだけでインストールのデフォルト値を設定することができる種類のトランスフォームです。たとえば、インストールされる機能、アプリケーションのインストール場所、および会社名のデフォルト値等を含んだ応答トランスフォームを作成することができます指定するデフォルト値はトランスフォームの開始位置として利用されます。

応答トランスフォームを作成するには、トランスフォーム オープン ウィザードを利用します。

応答トランスフォームの作成

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

タスク ***応答トランスフォームを作成するには、以下の手順に従います:***

1. トランスフォーム オープン ウィザードを開きます。
2. 必要に応じてウィザードの各パネルを完成します。[応答トランスフォームの作成] パネルで **[応答トランスフォームの作成]** を選択します。
3. **[コマンドラインのプロパティ (オプション)]** ボックスで、応答トランスフォームに渡すコマンドラインのプロパティ (セミコロンで分かれた一対のプロパティ名/値) を指定します。
4. **[完了]** をクリックします。インストールのユーザー インターフェイス要素が実行されます。
5. 必要に応じてインストールウィザードの各パネルで利用可能なオプションをカスタマイズして、シミュレート インストールを完成します。ファイルは転送されず、コンピューターに変更を加えることもありません。
6. インストールを最後まで進み、**[インストール]** をクリックします。

シミュレート インストールは終了し、InstallShield はトランスフォーム プロジェクトの [プロパティ] テーブルの中の値としてシミュレート インストールの間に加えられたすべての変更を保存します。これらの値はインストールでデフォルト値として使用されます。

応答トランスフォームの変更



プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*



タスク ***既存のトランスフォームプロジェクト内の個別応答を変更するには、以下の手順に従います:***

1. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
2. 応答を変更するトランスフォーム プロジェクト (.mst ファイル) を選択します。
3. **[開く]** をクリックします。InstallShield はダイレクト MST モードでトランスフォーム プロジェクトを開きます。
4. **ダイレクト エディター**で、**Property** テーブルをクリックします。
5. 必要に応じて任意の応答プロパティの値を変更します。COMPANYNAME と USERNAME はよく変更されるプロパティの代表例です。

また、別の方法として、トランスフォームの作成時と同様の手順を実行することもできます。これらの手順に従うと、インストールのユーザー インターフェイスを起動し、適切な応答を選択することができます。

応答トランスフォームのテスト



プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

応答に加えた変更を確認するために、次のいずれかを実行してトランスフォームのユーザー インターフェイス部分をテストすることができます。

- **[ビルド]** メニューで **[テスト]** をクリックします。
- **[テスト]** ボタンをクリックします。
- CTRL+T を押します。

サーバーの場所の構成



プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

エンド ユーザーが製品をネットワーク サーバーまたは Web サイトからインストールする場合で、機能をそのサーバーまたは Web サイトから実行できるようにするには、初回インストールの後、製品によるサーバーまたは Web サイト上の .msi パッケージおよび関連ファイルへのアクセスが必要になる場合があります。また、インストールが問題のあるファイルを自動的にサーバーまたは Web サイトからコピーすることができるため、一部のファイルが削除されているまたは破損している場合も、アクセスが必要になることがあります。

InstallShield では、トランスフォームに製品のインストール パッケージへのネットワークまたは URL ソース パスの一覧を指定することができます。このリストにあるパスは SOURCELIST プロパティに格納されています。実行時に Windows Installer は、エンド ユーザーの既存の製品のソース リストの最後に、このリストを追加します。

SOURCELIST プロパティ (Windows Installer ヘルプ ライブラリ)

サーバーの場所の追加



プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*



タスク

サーバーの場所としてネットワーク サーバー パスまたは Web サイト URL を追加するには、以下の手順に従います:

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"サーバーの場所"** 設定で、省略記号ボタン (...) をクリックします。**[サーバーの場所]** ダイアログ ボックスが開きます。
3. 必要に応じて、サーバーの場所を指定します。
4. **[OK]** をクリックします。

InstallShield が、"サーバーの場所" 設定の値を指定された場所で更新します。

サーバーの場所の変更



プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*



タスク

サーバー リストにある場所を変更するには、以下の手順を実行します:

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"サーバーの場所"** 設定で、省略記号ボタン (...) をクリックします。**[サーバーの場所]** ダイアログ ボックスが開きます。
3. 必要に応じて、サーバーの場所を変更します。
4. **[OK]** をクリックします。

InstallShield が、必要に応じて "サーバーの場所" 設定の値を更新します。

サーバーの場所の削除



プロジェクト- *この情報は、トランスフォーム プロジェクトに適用します。*



タスク ***サーバーの場所のパスを削除するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"サーバーの場所"** 設定で、省略記号ボタン (...) をクリックします。**[サーバーの場所]** ダイアログ ボックスが開きます。
3. **[ソース パス]** リストで、削除するパスを選択します。
4. **[削除]** ボタンをクリックします。
5. **[OK]** をクリックします。

サーバーの場所の順番を変更する



プロジェクト- *この情報は、トランスフォーム プロジェクトに適用します。*

インストールがネットワーク サーバーまたは Web サイト上に格納されているソース パスへのアクセスを必要とする場合、サーバーの場所一覧で最初にアクセス可能なパスを使用します。InstallShield では、必要に応じてパスの順序を並べ替えることができます。



タスク ***サーバーの場所の順番を変更するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"サーバーの場所"** 設定で、省略記号ボタン (...) をクリックします。**[サーバーの場所]** ダイアログ ボックスが開きます。
3. **[ソース パス]** リストで、移動するパスを選択します。
4. **[上へ移動]** と **[下へ移動]** ボタンをクリックして、必要に応じて順番を変更します。
5. **[OK]** をクリックします。

5

アドバンスト UI およびスイート/アドバンスト UI インストールの作成



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI インストールは、インストールと InstallShield 前提条件を合わせてパッケージ化し、カスタマイズ可能な統合ユーザー インターフェイスを持つ単一インストールを作成するブートストラップ アプリケーションです。これらのインストールでは、条件付きで、必要に応じて、ターゲット システムでパッケージを起動するセットアップ ランチャー (Setup.exe) を使用します。

以下は、スイート/アドバンスト UI インストール内の各パーツを示したダイアグラムです。このインストールでは、複数のプライマリ パッケージ (.exe パッケージ、サイドローディング Windows アプリ パッケージ (.appx)、および Windows Installer トランザクションを含む) および複数の InstallShield 前提条件パッケージを実行するサポートが提供されています。アドバンスト UI インストールのパーツは、アドバンスト UI インストールでは、プライマリ パッケージは 1 つのみ実行できるけれども、InstallShield 前提条件パッケージは複数実行できるという点を除き、類似しています。

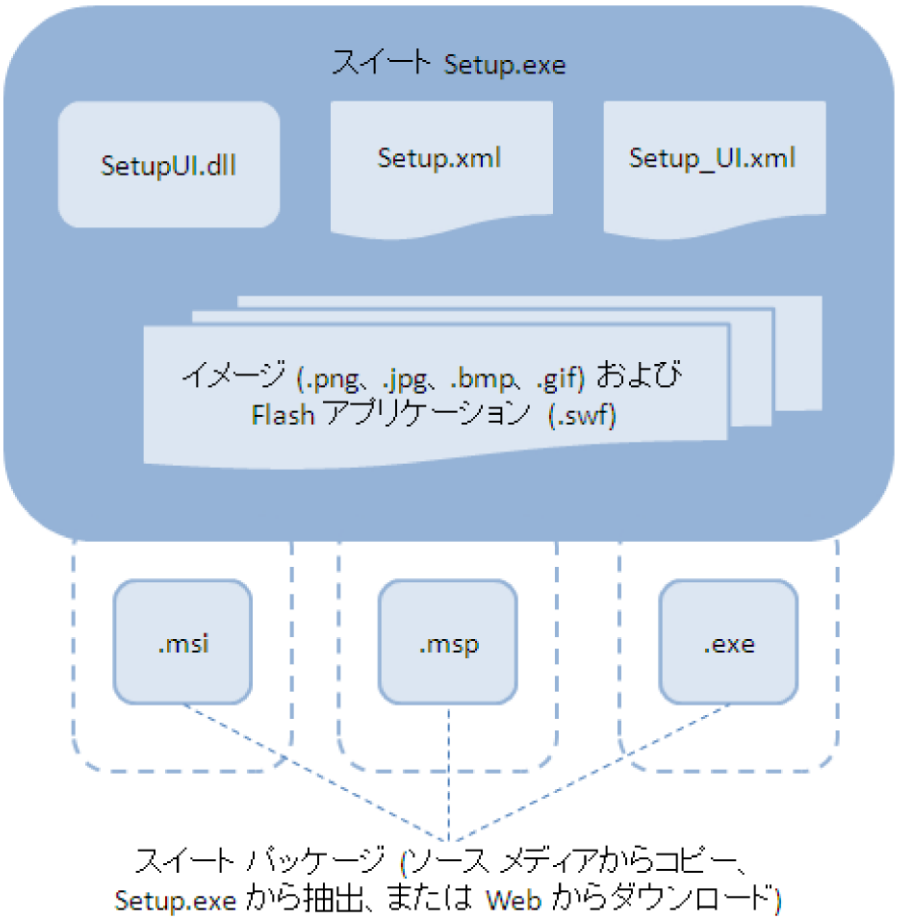


図 1: スイート/アドバンスト UI インストールのパーツ

ビルド時に InstallShield は .issuite ファイル (アドバンスト UI プロジェクトの xml プロジェクト ファイル、または、スイート/アドバンスト UI プロジェクトの XML プロジェクト ファイル) の様々なセクションを使用して、Setup.xml および Setup_UI.xml を作成し、これらを、アドバンスト UI またはスイート/アドバンスト UI パッケージとアドバンスト UI または スイート/アドバンスト UI インストールの実行に必要なその他のファイルと共にバンドルします。作成されたアドバンスト UI またはスイート/アドバンスト UI インストールには、次のファイルが含まれています:

- Setup.exe – アドバンスト UI またはスイート/アドバンスト UI エンジンが含まれています。
- Setup.xml – アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージ、パッケージの起動条件、およびアドバンスト UI またはスイート/アドバンスト UI インストールのウィザード ユーザー インターフェイスで使用される文字列エントリが記述されています。
- Setup_UI.xml – ウィザード インターフェイスのスタイル設定、レイアウト情報、その他の UI パラメーターを定義します。

- SetupUI.dll – アドバンスト UI およびスイート/アドバンスト UI インストールに使用されるアドバンスト UI およびスイート/アドバンスト UI ライブラリで構成されています。

InstallShield はまた、アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージを集めて、適切な場合はパッケージと関連する非圧縮ファイルのフォルダー構造を作成します。また、InstallShield には、アドバンスト UI またはスイート/アドバンスト UI インストールのサポート ファイルも組み込まれています。これらのサポート ファイルは、ウィザード ページで表示される使用許諾契約や画像ファイルです。

エンド ユーザーがアドバンスト UI またはスイート/アドバンスト UI インストールを起動すると、アドバンスト UI またはスイート/アドバンスト UI エンジンは Setup.xml ファイルを読み込み、SetupUI.dll ファイルをロードして、スイートならびにそのパッケージに定義された条件を評価し、Setup.xml ファイルの指示に従い、適切な場合は、ターゲット システムで必要なパッケージとファイルをダウンロードし、実行する必要があるパッケージを実行して、リソースをクリーンアップします。

アドバンスト UI またはスイート/アドバンスト UI インストールには、ターゲット システム上で Windows Installer 3.1 以降が必要になりますのでご注意ください。また、Windows XP 以降または Windows Server 2003 以降も必要です。

アドバンスト UI またはスイート/アドバンスト UI タイプのプロジェクトに関する詳しい情報は、ドキュメントのこのセクションを参照してください。

アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、インストールのブートストラップ アプリケーションを開発したり、完全にカスタマイズ可能なユーザー インターフェイスを作成することができます。次のテーブルは、これら 2 つのプロジェクト タイプの異なる機能と特徴の比較です。

テーブル 1 ・ アドバンスト UI およびスイート/アドバンスト UI プロジェクトの相違点

比較カテゴリ	アドバンスト UI プロジェクト	スイート/アドバンスト UI プロジェクト
InstallShield の Edition Premier および Professional Edition の違いついては、「InstallShield の他のエディションからアップグレードする」をご覧ください。	InstallShield Professional	InstallShield Premier

テーブル 1 ・ アドバンスト UI およびスイート/アドバンスト UI プロジェクトの相違点 (続き.)

比較カテゴリ	アドバンスト UI プロジェクト	スイート/アドバンスト UI プロジェクト
プロジェクト ファイルの拡張子	.issuite	.issuite
サポートされているプライマリ パッケージのファイル形式 適切なパッケージのファイル形式の選択については、「パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン」をご覧ください。	.msi パッケージ .msp パッチ InstallScript パッケージ	.msi パッケージ .msp パッチ InstallScript パッケージ .exe パッケージ .appx パッケージ Web 配置パッケージ (.zip) Windows Installer トランザクション スイート/アドバンスト UI インストールに、基本の MSI および InstallScript プロジェクト (.ism) をパッケージとして含めることもできます。ビルド時に InstallShield は、InstallShield プロジェクト内で指定されたリリースを最初にビルドして、生成するスイート/アドバンスト UI インストールにそれらをパッケージとして含めます。 サイドローディング Windows アプリ パッケージを含むスイート/アドバンスト UI インストールを作成およびビルドするには、InstallShield または Standalone Build が搭載されているマシン上に Windows Vista 以降または Windows Server 2008 以降が必要です。
プロジェクトに含めることができるプライマリ パッケージの数 詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるプライマリ パッケージと依存パッケージの違い」を参照してください。	1 この機能を利用すると、単一のパッケージに対して、最新のカスタマイズ可能なユーザー インターフェイスを作成することができます。	1 つ以上 この機能を利用すると、単一のパッケージまたは複数のパッケージに対して、最新のカスタマイズ可能なユーザー インターフェイスを作成することができます。 また、この機能を使うと、複数の個別のインストールを、統合されたユーザー インターフェイスを使用する単一のインストールにパッケージ化することもできます。

テーブル 1 ・ アドバンスト UI およびスイート/アドバンスト UI プロジェクトの相違点 (続き.)

比較カテゴリ	アドバンスト UI プロジェクト	スイート/アドバンスト UI プロジェクト
InstallShield 前提条件をプロジェクトに含めることができるサポート 詳細については、「InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める」を参照してください。	必要に応じて、InstallShield 前提条件を、.msi パッケージ、.msi パッケージ、.msp パッケージ、および .exe パッケージとして (前提条件に対して実行されるように設定されたファイルの種類に応じます)、数に制限なく含めることができます。	必要に応じて、InstallShield 前提条件を、.msi パッケージ、.msi パッケージ、.msp パッケージ、および .exe パッケージとして (前提条件に対して実行されるように設定されたファイルの種類に応じます)、数に制限なく含めることができます。
プライマリ パッケージとセカンダリ パッケージを識別できるサポート 詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるプライマリ パッケージと依存パッケージの違い」を参照してください。	制限付きで柔軟性あり。1 つの .msi、.msp、または InstallScript パッケージをプライマリ パッケージとしてプロジェクトに追加できます。 プロジェクトに含める InstallShield 前提条件はすべて、依存パッケージとして追加されます。これは変更不可能です。	完全に柔軟性あり。任意のサポートされている種類のパッケージをプロジェクトに追加して、それらがプライマリ パッケージまたは依存パッケージであるかを容易に指定できます。 プロジェクトに含めた InstallShield 前提条件はすべて、依存パッケージとして追加されますが、それを上書きして、プライマリ パッケージとしてマークすることも可能です。
アクションを作成して、適切なランタイム イベント中にそれらの実行をスケジュールできる機能のサポート 詳細については、「スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する」を参照してください。	利用できません。	スイート/アドバンスト UI プロジェクトには、[イベント] ビューが含まれており、実行時にターゲット システム上で発生するアクションを定義することができます。[イベント] ビューで特定のイベント中に発生させるアクションをスケジュールするか、[パッケージ] ビューでパッケージに割り当てることができます。さらに、ウィザード ページまたは 2 番目のウィンドウを開くまたは閉じるとき、またはエンド ユーザーがコントロールを使用したとき (たとえば、ボタンをクリックしたとき) に発生させるアクションをスケジュールすることもできます。

テーブル 1・アドバンスト UI およびスイート/アドバンスト UI プロジェクトの相違点 (続き.)

比較カテゴリ	アドバンスト UI プロジェクト	スイート/アドバンスト UI プロジェクト
ビルド イベントのサポート 詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。	利用できません。	スイート/アドバンスト UI プロジェクトでは、InstallShield でリリースのビルドが開始する前、および、InstallShield がリリースをビルドして署名をした後で実行するコマンドをスケジュールすることができるリリース設定が提供されています。
ランタイム言語サポート 詳細については、「InstallShield 実行時の言語サポート」を参照してください。	アドバンスト UI インストールのエンドユーザー インターフェイスは、InstallShield をインストールしたときに選択した 1 言語に限られます。	エンドユーザーのテキストを複数の言語で表示する単一インストールを作成できます。翻訳済みの文字列を利用して、ウィザード ページおよびメッセージを 34 の追加言語の 1 つに変更することができます。
ビルド時に InstallShield が初期化する仮想マシンへインストールを配布できる機能 詳細については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。	利用できません。	インストールのビルドが成功するたびに InstallShield が仮想マシン (VM) を指定のスナップショットに戻し、VM の電源をオンにしてスイート/アドバンスト UI インストールを VM にコピーし、テストが可能な状態にします。この機能を使って手作業で行う処理を減らし、テストにかかる時間を節約できます。

アドバンスト UI またはスイート/アドバンスト UI インストールで機能、パッケージ、前提条件、およびファイルを編成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロ*

ジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI インストールはブートストラップ アプリケーションで、通常、その主要な目的はターゲット システム上で必要に応じて条件付きでパッケージを起動することです。アドバンスト UI またはスイート/アドバンスト UI インストールでは、製品が適切に機能するようにサードパーティまたはカスタム再配布可能ファイルをインストールする InstallShield 前提条件を実行する必要がある場合もあります。

機能は、エンド ユーザーから見て、アドバンスト UI またはスイート/アドバンスト UI インストールの個別にインストール可能な最小構成単位です。各パッケージと、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれる InstallShield 前提条件は、プロジェクト内の機能と関連付けなくてはなりません。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるパッケージと InstallShield 前提条件の追加および構成については、ドキュメントのこのセクションを確認してください。このセクションには、次のトピックが含まれています:

- パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン
- .msi パッケージ、.msp パッチ、トランザクションをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する
- InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトの InstallScript パッケージについての特別考慮
- スイート/アドバンスト UI プロジェクトに InstallShield プロジェクト (.ism) をパッケージとして追加する
- InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトの InstallScript パッケージについての特別考慮
- スイート/アドバンスト UI プロジェクトに InstallShield プロジェクト (.ism) をパッケージとして追加する
- 実行可能パッケージ (.exe) をスイート/アドバンスト UI プロジェクトに追加する
- Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する
- サイドローディング Windows アプリ パッケージ (.appx| msix) をスイート/アドバンスト UI プロジェクトに追加する
- サイドローディング Windows アプリ バンドル (.appxbundle | .msixbundle) をスイート/アドバンスト UI プロジェクトに追加する
- Windows パッケージ マネージャー パッケージをスイート/アドバンスト UI プロジェクに追加する
- アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージにおけるスタティック ファイルとダイナミック ファイルの違い
- InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるプライマリ パッケージと依存パッケージの違い
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、パッケージとトランザクションのインストール順序を指定する

- アドバンスト UI またはスイート/アドバンスト UI プロジェクトの設定を構成する
- アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージを機能に関連付ける
- 異なるアドバンスト UI およびスイート/アドバンスト UI インストール間で、共通パッケージを共有する
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに対してランタイムの場所を指定する
- アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージ操作を構成する
- アドバンスト UI または スイート/アドバンスト UI インストールのパッケージにカスタム フォルダー名を使用する

パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

推奨されるファイル フォーマット

アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージ用に使用するファイル形式は、ブートストラップ アプリケーションまたはセットアップ ランチャー (Setup.exe) を含まないファイル形式をお勧めします。アドバンスト UI またはスイート/アドバンスト UI プロジェクトには、次のパッケージの種類が推奨されています:

- .msi パッケージ
- .msp パッチ
- InstallScript パッケージ パッケージは、InstallShield 2012 Spring 以降でビルドされた非圧縮の InstallScript インストールでなければなりません。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおける、これらの種類のパッケージの使用については、次」を参照してください:

- .msi パッケージ、.msp パッチ、トランザクションをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する
- InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する

スイート/アドバンスト UI インストールには、IIS サーバーおよびクラウドに Web 配置パッケージ (.zip) を展開できるサポートが含まれています。詳細については、「Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する」を参照してください。

さらに、InstallShield にはサイドローディング Windows アプリ パッケージ (.appx) をスイート/アドバンスト UI プロジェクトに追加するためのサポートが含まれています。アプリのサイドローディングとは、Windows Store を介さずにアプリケーションをインストールするプロセスです。詳細については、「サイドローディング Windows アプリ パッケージ (.appx| msix) をスイート/アドバンスト UI プロジェクトに追加する」を参照してください。
Windows アプリ パッケージは限られた状況下でのみサポートされているため、通常これらのパッケージは .msi パッケージなどのより一般的なインストール形式と共に使用されます。

InstallShield では、実行可能パッケージ (.exe) をスイート/アドバンスト UI プロジェクトに追加できます。以下に例を示します:

- Windows Installer ベースのインストール用に InstallShield でビルドされたセットアップ ランチャー 実行可能ファイル
- Windows Installer ベースのパッチ用に InstallShield でビルドされたセットアップ ランチャー 実行可能ファイル
- InstallScript インストール用のセットアップ ランチャー ファイル (これには、InstallShield 2012 以前でビルドされた InstallScript インストールおよび圧縮済み InstallScript インストールが含まれます。)
- InstallShield 以外のツールでビルドされたセットアップ ランチャー実行可能ファイル

.msi、.msp、および InstallScript パッケージは、さまざまな理由で、セットアップ ランチャー実行可能ファイルよりも利点があります:

- たとえば、アドバンスト UI およびスイート/アドバンスト UI セットアップ ランチャーのウィザード インターフェイスでは、ターゲット システム上で実行中、推奨されるファイル フォーマットの進行状況を段階的に表示できますが、実行可能パッケージの進行状況は表示できません。
- パッケージとして推奨されるファイル フォーマットの 1 つをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加したとき、アドバンスト UI またはスイート/アドバンスト UI インストールで、以前のバージョンのパッケージが新しいバージョンを上書きするのを防ぐための適切な対象条件が自動的に追加されます。実行可能パッケージをスイート/アドバンスト UI プロジェクトに追加する場合、このようなダウングレードを防ぐ対象条件を手入力で定義する必要があります。
- パッケージとして推奨されるファイル フォーマットの 1 つをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する場合、そのパッケージに対して、ターゲット システムでパッケージが既にインストールされているかどうかを評価する判別条件を定義する必要はありません。これは、このようなシナリオは、アドバンスト UI またはスイート/アドバンスト UI インストールでブロックされるからです。実行可能パッケージをスイート/アドバンスト UI プロジェクトに追加する場合、この種類の判別条件は、パッケージに対して手入力で定義する必要があります。
- アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーでは、アドバンスト UI またはスイート/アドバンスト UI インストールのウィザード インターフェイスが優先されるため、推奨されるファイル フォーマットのユーザー インターフェイスは自動的に抑制されます。プロジェクトに実行可能パッケージが含まれている場合、そのユーザー インターフェイスを手動で抑制し、アドバンスト UI またはスイート/アドバンスト UI インストールで、そのユーザー インターフェイスをサイレントで起動する必要があります。そうしなかった場合、インストールの実行時に、アドバンスト UI またはスイート/アドバンスト UI のユーザー インターフェイスと .exe パッケージのユーザー インターフェイスの 2 つの異なるユーザー インターフェイスが表示されることになります。.msi パッケージ、.msp パッチ、トランザクションをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する

詳細については、「実行可能パッケージ (.exe) をスイート/アドバンスト UI プロジェクトに追加する」を参照してください。

メディアの種類の要件

アドバンスト UI またはスイート/アドバンスト UI インストールでは、複数のディスク (CD や DVD など) に分割されたパッケージをインストールできません。また、ネットワークなどの固定ディスクからの複数ディスク パッケージもインストールすることができません。従って、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含めるパッケージ (.msi パッケージ、.msp パッケージ、InstallScript パッケージ、.exe パッケージ、Web 配置パッケージ、または Windows アプリ パッケージ(.appx)) はどれも、単一ディスク リムーバブル メディア タイプ、ネットワーク イメージ タイプ、または Web タイプである必要があります。

.msi パッケージ、.msp パッチ、トランザクションをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

プロジェクト固有の違いについては、必要に応じて記述されています。



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、次の種類の Windows Installer ベースのインストールを、アドバンスト UI およびスイート/アドバンスト UI プロジェクトに追加することができます:

- Windows Installer パッケージ (.msi パッケージ)
- Windows Installer 修正プログラム (パッチ) (.msp パッケージ)

InstallShield ではまた、Windows Installer トランザクションをスイート/アドバンスト UI プロジェクトに含めることもできます (アドバンスト UI プロジェクトには追加できません)。

トランザクションは、Windows Installer 4.5 以降の機能を使った、ターゲット システム上でトランザクション処理を使って連鎖させる Windows Installer パッケージ (.msi) および Windows Installer パッチ (.msp) で構成されます。パッケージを連鎖させることによって、単一のトランザクションとして処理します。各スイート/アドバンスト UI インストールには、複数の個別のトランザクションを含めることができます。トランザクションに含まれる 1 つまたは複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は現在のトランザクションに含まれる連鎖パッケージ全てのロールバックを開始して、システムを以前の状態に復元します。

トランザクション処理なしに実行する .msi または .msp パッケージを追加する

タスク

トランザクション処理なしで実行する .msi または .msp パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックして、適切なコマンド (**新しい Windows Installer パッケージ (.msi)** または **新しいパッチ パッケージ (.msp)**) をクリックします。**[このパッケージのインストール ファイルを選択します]** ダイアログ ボックスが開きます。
3. アドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加したいパッケージ ファイルを選択してから、**[開く]** をクリックします。**[このパッケージのファイルを追加する]** ダイアログ ボックスが開きます。
4. パッケージ ファイルに追加のフォルダーとファイルを含めるかどうかを指定します。含める場合、ダイナミック リンク ファイルを含めるかどうかも指定します。

InstallShield によって、[パッケージ] エクスプローラーにパッケージおよび追加のファイルおよびフォルダーが追加されます。

トランザクション処理を使って実行するパッケージを追加する

タスク

トランザクション処理を使って実行するパッケージをスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[新しいトランザクション]** をクリックします。**[パッケージ]** エクスプローラーに、**[トランザクション]** フォルダーが追加されます。
3. **[パッケージ]** エクスプローラーで、**トランザクション** フォルダーを右クリックして、適切なコマンド (**新しい Windows Installer パッケージ (.msi)** または **新しいパッチ パッケージ (.msp)**) をクリックします。**[このパッケージのインストール ファイルを選択します]** ダイアログ ボックスが開きます。
4. スイート/アドバンスト UI プロジェクトに追加したいパッケージ ファイルを選択してから、**[開く]** をクリックします。**[このパッケージのファイルを追加する]** ダイアログ ボックスが開きます。
5. パッケージ ファイルに追加のフォルダーとファイルを含めるかどうかを指定します。含める場合、ダイナミック リンク ファイルを含めるかどうかも指定します。

InstallShield によって、[パッケージ] エクスプローラーにパッケージおよび追加のファイルおよびフォルダーが追加されます。

重要: *スイート/アドバンスト UI インストールからトランザクション内のパッケージを使用するには、Windows Installer 4.5 以降が必要になります。トランザクション内のパッケージが、以前のバージョンの Windows Installer があるターゲット システム上のスイート/アドバンスト UI インストールによって起動された場合、スイート/アドバンスト UI インストールは失敗します。従って、スイート/アドバンスト UI インストールに含まれるパッケージにトランザクション処理を使用する場合、以 下のタスクから 1 つ以上を実行することが推奨されます:*

- *以前のバージョンがあるターゲット システム用のスイート/アドバンスト UI プロジェクトに Windows Installer 4.5 再配布可能ファイルをパッケージとして含めて、トランザクション処理を使用するパッケージの前にスケジュールする。*
- *Windows Installer 4.5 以降が搭載されていないシステム上でスイート/アドバンスト UI インストールが実行することを防ぐ終了条件を作成する。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールの終了条件を定義する」を参照してください。*
- *トランザクションに含まれる各パッケージに Windows Installer 4.5 以降の存在を確認する対象条件を作成して、トランザクションに含まれるこれらのパッケージが起動しないように防ぐ。詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。*

InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。スイート/アドバンスト UI とアドバンスト UI プロジェクト タイプの違い関する説明は、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」をご覧ください。*

InstallScript インストールを、実行可能パッケージとしてではなく InstallScript パッケージとしてアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するための要件

InstallShield では、InstallScript パッケージが次の要件を満たした場合にのみ、InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加することができます。

- InstallScript パッケージは非圧縮です。
- InstallShield 2012 Spring 以降が、InstallScript パッケージまたはアドバンスト UI またはスイート/アドバンスト UI インストールを作成するために使われている。
- InstallScript パッケージで、イベント ベースのスクリプトが使用される。program...endprogram スタイルのスクリプトは使用しない。program...endprogram ブロックを使用するスクリプトをイベント ベースのスクリプトに変換する必要がある場合、「OnShowUI」を参照できます。

これらの条件のいずれかが満たされなかった場合、InstallScript インストールを、InstallScript パッケージとしてではなく、実行可能ファイル パッケージ (.exe) として、スイート/アドバンスト UI プロジェクトに追加することができます。ただし、InstallScript インストールを、実行可能パッケージ (.exe) としてではなく、InstallScript パッケージとして含める方が優先されます。詳細については、「パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン」を参照してください。

メモ: *InstallShield 2012 以前の InstallScript インストールをパッケージとしてアドバンスト UI またはスイート/アドバンスト UI インストールに含める場合、アドバンスト UI またはスイート/アドバンスト UI インストールで、InstallScript パッケージが起動されたとき、ランタイム エラーが発生します。*

InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときの手順

タスク

InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[InstallScript パッケージ (.hdr)]** をクリックします。**[このパッケージのインストール ファイルを選択します]** ダイアログ ボックスが開きます。
3. InstallScript インストールの data1.hdr ファイルを選択して、**[開く]** をクリックします。data1.hdr ファイルは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトにパッケージとして含める非圧縮 InstallScript インストールの残りのファイルと同じフォルダーに格納します。

ヒント: *InstallScript インストールの data1.hdr ファイルを選択すると、InstallScript パッケージが起動されたときに必要な .cab ファイル、InstallScript ファイル (.rul)、および、その他のファイルがアドバンスト UI またはスイート/アドバンスト UI インストールに含められます。*

InstallShield によって、[パッケージ] エクスプローラーにパッケージおよび追加のファイルおよびフォルダーが追加されます。

ヒント: *アドバンスト UI およびスイート/アドバンスト UI プロジェクトに InstallScript パッケージを含める手順についての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトの InstallScript パッケージについての特別考慮」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの InstallScript パッケージについての特別考慮

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。スイート/アドバンスト UI とアドバンスト UI プロジェクト タイプの違い関する説明は、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」をご覧ください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトに InstallScript パッケージを含める場合、または スイート/アドバンスト UI プロジェクトに InstallScript プロジェクト (.ism) を含める場合、以下の詳細に注意が必要です。

ランタイム ユーザー インターフェイス、イベント、および関数

アドバンスト UI またはスイート/アドバンスト UI インストールで InstallScript パッケージが起動されたとき、それ自身のユーザー インターフェイス (UI) が表示され、InstallScript パッケージの UI は自動的に抑制されます。アドバンスト UI またはスイート/アドバンスト UI インストールでは、InstallScript パッケージに対して、進行状況も表示されます。これらの変更を可能にするために、アドバンスト UI またはスイート/アドバンスト UI インストールでは、デフォルトで、いくつかのアドバンスト UI およびスイート/アドバンスト UI 固有の InstallScript イベントおよび関数が使用され、一部の標準 InstallScript イベントおよび関数は無視されます。アドバンスト UI およびスイート/アドバンスト UI 固有の InstallScript イベントによって、ファイルの転送がどう開始されるかも補強されています。

InstallScript Setup.exe ファイルから起動される (スイート/アドバンスト UI インストールから起動されない) 標準の InstallScript インストールでは、ほとんどのイベントは OnShowUI イベントから直接呼ばれます。InstallScript パッケージを起動するアドバンスト UI またはスイート/アドバンスト UI インストールでは、OnShowUI が OnSuiteShowUI によって置き換えられます。OnSuiteShowUI を使ってデフォルトで起動されるアドバンスト UI およびスイート/アドバンスト UI 固有のイベントなどに関する詳しい情報は、「OnSuiteShowUI」をご覧ください。

InstallScript 言語には、InstallScript パッケージとそれを実行するアドバンスト UI またはスイート/アドバンスト UI インストールの間の対話を可能にするアドバンスト UI およびスイート/アドバンスト UI 固有の関数がいくつか含まれています。詳しくは、「アドバンスト UI およびスイート/アドバンスト UI の対話関数」を参照してください。

次の事項に注意してください。

- アドバンスト UI またはスイート/アドバンスト UI インストールでは、InstallScript パッケージの標準 InstallScript UI が抑制されるため、すべてのスクリプト ダイアログ (例、**DefineDialog**、**EzDefineDialog**) で、エラーが返されます。従って、ダイアログの呼び出しは、アドバンスト UI およびスイート/アドバンスト UI 固有のイベントまたは機能の転送イベントで行う必要があります。
- アドバンスト UI およびスイート/アドバンスト UI インストールでは、すべてのスクリプトから提供されたメッセージ ボックス関数 (例、**MessageBox**、**SprintfBox**) は SuiteErrorReport 関数を介して送られます。これらの関数への呼び出しは、エラー ケース以外、ユーザー入力を要求する方法として使用することはできません。すべてのユーザー入力は、アドバンスト UI またはスイート/アドバンスト UI のウィザード ページから取得する必要があります。この設定は、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [ウィザード インターフェイス] ビューで構成することができます。この構成により、アドバンスト UI またはスイート/アドバンスト UI インストール内のすべてのパッケージで、一貫した UI 使用感を提供することができます。
- 非 InstallScript UI (DLL または 他の方法によって実装されたカスタム UI など) は、アドバンスト UI およびスイート/アドバンスト UI 固有の InstallScript イベントまたは機能転送イベントから呼び出すことはできません。すべてのユーザー入力は、アドバンスト UI インストールまたはスイート/アドバンスト UI インストールの UI で取得する必要があります。

アドバンスト UI またはスイート/アドバンスト UI インストールからのデータをアドバンスト UI またはスイート/アドバンスト UI インストール内の InstallScript パッケージに渡す

コマンドラインを使って、アドバンスト UI またはスイート/アドバンスト UI インストールからのデータをスイート内の InstallScript パッケージに渡すには、CMDLINE 変数を使用します。詳細については、「CMDLINE」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI インストールのプロパティの値を InstallScript パッケージに渡すには、InstallScript 関数 SuiteGetProperty または SuiteFormatString を使用します。

アドバンスト UI またはスイート/アドバンスト UI インストールから起動するためのランタイム検出

InstallScript インストールが、アドバンスト UI またはスイート/アドバンスト UI インストール内の InstallScript パッケージとして実行されているかどうかを判別するには、InstallScript コードで SUITE_HOSTED 変数を使用します。詳細については、「SUITE_HOSTED」を参照してください。

スイート/アドバンスト UI プロジェクトに InstallShield プロジェクト (.ism) をパッケージとして追加する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、スイート/アドバンスト UI プロジェクトに基本の MSI および InstallScript プロジェクト ファイル (.ism) をパッケージとして追加できます。1 つ以上の InstallShield プロジェクト パッケージを含むスイート/アドバンスト UI プロジェクトのリリースをビルドすると、関連する InstallShield プロジェクト内の指定されたリリースが最初にビルドされて、生成されるスイート/アドバンスト UI インストールにそれらがパッケージとして含まれます。InstallShield は基本の MSI プロジェクトおよび InstallScript プロジェクトの出力を .msi パッケージおよび .hdr パッケージとしてスイート/アドバンスト UI インストールに追加します。

スイート/アドバンスト UI プロジェクトに基本の MSI または InstallScript プロジェクトを含める場合、それらの基本の MSI または InstallScript パッケージはスイート/アドバンスト UI プロジェクトと同じバージョンの InstallShield で保存されていなくてはなりません。

タスク

基本の MSI プロジェクトまたは InstallScript プロジェクトをスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[プロジェクト パッケージ (.ism)]** をクリックします。**[このパッケージのインストール ファイルを選択します]** ダイアログ ボックスが開きます。

3. スイート/アドバンスト UI プロジェクトに追加したい .ism ファイルを選択してから、**[開く]** をクリックします。

[パッケージ] エクスプローラーにプロジェクトが追加されます。右側のペインで設定を構成します。"製品構成" 設定 (基本の MSI プロジェクト パッケージの場合) および "リリース" 設定 (基本の MSI または InstallScript プロジェクト パッケージの場合) を使って、InstallShield でスイート/アドバンスト UI インストールに含めるリリースを選択します。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトに InstallScript パッケージを含める手順についての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトの InstallScript パッケージについての特別考慮」を参照してください。

実行可能パッケージ (.exe) をスイート/アドバンスト UI プロジェクトに追加する



プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、実行可能パッケージ (.exe) をスイート/アドバンスト UI プロジェクトに追加できます。このフォーマットのスイート/アドバンスト UI パッケージは、優先されるフォーマット タイプの 1 つが使用できないときのみ推奨されます。詳細については、「パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン」を参照してください。



タスク ***.exe パッケージをスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[新しい実行可能パッケージ (.exe)]** をクリックします。**[このパッケージのインストール ファイルを選択します]** ダイアログ ボックスが開きます。
3. スイート/アドバンスト UI プロジェクトに追加したい .exe ファイルを選択してから、**[開く]** をクリックします。**[このパッケージのファイルを追加する]** ダイアログ ボックスが開きます。
4. パッケージ ファイルに追加のフォルダーとファイルを含めるかどうかを指定します。含める場合、ダイナミック リンク ファイルを含めるかどうかも指定します。

InstallShield によって、[パッケージ] エクスプローラーにパッケージおよび追加のファイルおよびフォルダーが追加されます。

スイート/アドバンスト UI プロジェクトで実行可能パッケージの構成するときの注意事項

次のタスクは、スイート/アドバンスト UI プロジェクトで実行可能パッケージを構成するときに推奨されます。

- スイート/アドバンスト UI インストールに対して異なるシナリオごとの動作を指示するための条件を定義します (たとえば、パッケージが既にインストール済みかどうかの検証、要件や依存関係の確認など)。詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。
- 実行可能パッケージのユーザー インターフェイスを抑制し、スイート/アドバンスト UI インストールをサイレントで起動させます。そうしなかった場合、インストールの実行時に、スイート/アドバンスト UI ユーザー インターフェイスと .exe パッケージのユーザー インターフェイスの 2 つの異なるユーザー インターフェイスが表示されることになります。詳細については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」を参照してください。

Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する



プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、Web 配置パッケージ (.zip) をスイート/アドバンスト UI プロジェクトに追加できます。実行時、スイート/アドバンスト UI インストールは、エンド ユーザーの入力に基づいて IIS Web サーバーまたはクラウド上に Web 配置パッケージを配置します。Web 配置パッケージは、Visual Studio などの IIS または Web アプリケーション開発環境を使って作成できます。



タスク ***Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[新しい Web 配置パッケージ (.zip)]** をクリックします。**[このパッケージのインストール ファイルを選択します]** ダイアログ ボックスが開きます。
3. スイート/アドバンスト UI プロジェクトに追加したい Web 配置パッケージ ファイルを選択してから、**[開く]** をクリックします。**[このパッケージのファイルを追加する]** ダイアログ ボックスが開きます。
4. パッケージ ファイルに追加のフォルダーとファイルを含めるかどうかを指定します。含める場合、ダイナミック リンク ファイルを含めるかどうかも指定します。通常、Web 配置パッケージは自己完結型で、.zip ファイルのみを必要とします。

InstallShield によって、[パッケージ] エクスプローラーにパッケージおよび追加のファイルおよびフォルダーが追加されます。

スイート/アドバンスト UI プロジェクトで Web 配置パッケージを構成するときの注意事項

Web 配置パッケージのライフサイクル

Web 配置パッケージは一般的に、初回インストールと同様、または既存の古いバージョンあるいは同一バージョンを上書きする場合と同様に動作するように設計されています。その結果、Web 配置パッケージではアンインストールを行うことができません。つまり、スイート/アドバンスト UI インストールはこの種類のパッケージのメンテナンスを行うことができません。1 つ以上の Web 配置パッケージ、または 1 つ以上の従来型パッケージ（メンテナンス オプションが含まれている）をプライマリ パッケージとして含むスイート/アドバンスト UI インストールの作成を避けることが推奨されます。

スイート/アドバンスト UI プロパティと、ハードコード化された Web 配置構成値の設定との違い

[パッケージ] ビューに Web 配置パッケージを選択すると、右側のペインにパッケージに関連する設定が表示されます。右側のペインの [共通] タブにある [構成] 領域には、Web 配置の構成関連の設定、たとえばインストール先（多くの場合はリモート サーバー）や資格情報などが含まれています。さらに、[構成] 領域には Web 配置パッケージのパラメーター XML ファイルで定義されるパラメーターの設定も含まれています。デフォルトでは、スイート/アドバンスト UI プロパティが構成設定すべての構成設定に使用され、エンド ユーザーはウィザード インターフェイスまたはコマンドラインを使って実行時にそれらを設定することができます。

プロパティのデフォルト値を含める場合、[プロパティ マネージャー] ビューを使って、対応するプロパティの値を指定します。

一部の状況において、1 つ以上のパラメーターにハードコード化された値を使用したい場合があります。たとえば、特定の Web 配置パラメーターを特定の値に設定するとき、そのパラメーターのデフォルト プロパティ名をハードコード化された値に置換する必要があります。

Web 配置パッケージを構成するためのウィザード インターフェイスについて

Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加すると、そのパッケージ用の定義済みの Web 配置ウィザード ページが自動的にプロジェクトに追加されます。Web 配置ウィザード ページを使って、エンド ユーザーは特定のパッケージをローカル IIS サーバー、リモート サーバー、またはクラウドベースのサーバーのいずれに配置するのかを指定することができます。また、ビルトイン Web 配置関連アクションを使うと、エンド ユーザーがパブリッシャー プロファイル ファイル（.publishsettings）から構成設定をロードすることも可能です。

エンド ユーザーが Web 配置パッケージのパラメーター XML ファイルで定義されているパラメーターを構成できるようにするには、必要に応じてウィザード インターフェイスにコントロールを追加して、それらのコントロールを [パッケージ] ビューの対応するパラメーター設定に指定されているプロパティに関連付けます。

サイドローディング Windows アプリ パッケージ（.appx|msix）をスイート/アドバンスト UI プロジェクトに追加する



プロジェクト- *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

メモ: *サイドローディング Windows アプリ パッケージを含むスイート/アドバンスト UI インストールを作成およびビルドするには、InstallShield または Standalone Build が搭載されているマシン上に Windows Vista 以降または Windows Server 2008 以降が必要です。*

InstallShield では、サイドローディング Windows アプリ パッケージ (.appx) をスイート/アドバンスト UI プロジェクトに追加できます。

アプリのサイドローディングとは、Windows Store を介さずにアプリケーションをインストールするプロセスです。この種類のアプリケーションは、企業環境で配布されることがあります。サイドロード アプリケーションの要件は以下の通りです:

- ターゲット システムに Windows 8 以降、または Windows Server 2012 以降が搭載されている。
- ターゲット システム上で **"すべての信頼されているアプリケーションのインストールを許可する"** グループ ポリシー設定が有効になっている。

サイドロードおよびこれらの種類のアプリケーションを実行するためのその他の要件は、MSDN Web サイトに掲載されている Windows ドキュメントを参照してください。

スイート/アドバンスト プロジェクトでは、ARM プロセッサ アーキテクチャをターゲットとする Windows アプリ パッケージを含める機能はサポートされていません。

タスク

Windows アプリ パッケージ (.appx) をスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[新しいサイドローディング Windows アプリ パッケージ (.appx)]** をクリックします。**[このパッケージのインストール ファイルを選択します]** ダイアログ ボックスが開きます。
3. スイート/アドバンスト UI プロジェクトに追加したい Windows アプリ パッケージを選択してから、**[開く]** をクリックします。**[このパッケージのファイルを追加する]** ダイアログ ボックスが開きます。
4. パッケージ ファイルに追加のフォルダーとファイルを含めるかどうかを指定します。含める場合、ダイナミック リンク ファイルを含めるかどうかも指定します。通常、Windows アプリ パッケージは自己完結形式で、Windows アプリ パッケージおよび証明書ファイル (.cer) のみを必要とします。

InstallShield によって、[パッケージ] エクスプローラーにパッケージおよび追加のファイルおよびフォルダーが追加されます。

サイドロード アプリケーション パッケージをスイート/アドバンスト UI プロジェクトに追加すると、InstallShield によってそのパッケージに Windows 8 以降または Windows Server 2012 以降の存在を確認する対象条件が自動的に追加されます。この対象条件は、スイート/アドバンスト UI インストールが Windows の以前のバージョンが搭載されているシステム上にパッケージがインストールされないように防ぎます。

サイドローディング Windows アプリ バンドル (.appxbundle | .msixbundle) をスイート/アドバンスト UI プロジェクトに追加する

プロジェクト- *この情報は、スイート/アドバンスト UI に適用します。*

エディション- *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

メモ- *サイドローディング Windows アプリ バンドルを含むスイート/アドバンスト UI インストールを作成およびビルドするには、InstallShield または Standalone Build が搭載されているマシン上に Windows 8 以降または Windows Server 2016 以降が必要です。*

InstallShield では、サイドローディング Windows アプリ バンドル (.msixbundle) をスイート/アドバンスト UI プロジェクトに追加できます。

アプリのサイドローディングとは、Windows Store を介してアプリ バンドルを取得せずにインストールするプロセスです。この種類のアプリ バンドルは、企業環境で配布されることがあります。サイドロード アプリ バンドルの要件は以下の通りです:

- ターゲット システムに Windows 10、または Windows Server 2019 以降が搭載されている。
- ターゲット システム上で **"すべての信頼されているアプリケーションのインストールを許可する"** グループ ポリシー設定が有効になっている。

サイドロードおよびこれらの種類のアプリ バンドルを実行するためのその他の要件は、MSDN Web サイトに掲載されている Windows ドキュメントを参照してください。

タスク

Windows アプリ バンドル (.msixbundle) をスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. [パッケージ] エクスプローラーを右クリックしてから、**[新しい UWP アプリ バンドル (.appxbundle | .msixbundle)]** をクリックします。
   [このパッケージのインストール ファイルを選択します] ダイアログ ボックスが開きます。
3. スイート/アドバンスト UI プロジェクトに追加したい Windows アプリ バンドルを選択してから、**[開く]** をクリックします。
   このパッケージの **[ファイルを追加する]** ダイアログ ボックスが開きます。
4. パッケージ ファイルに追加フォルダーおよびファイルを含めるかどうかを指定します。その場合、ダイナミックにリンクされたファイルを含めるかどうかを指定します。通常、Windows アプリ バンドルは自己完結形式で、Windows アプリ バンドルおよび証明書ファイル (.cer) のみを必要とします。

5. InstallShieldによって、[パッケージ] エクスプローラーにバンドルおよび追加のファイルおよびフォルダーが追加されます。

メモ- *サイドロード アプリケーション バンドルをスイート/アドバンスト UI プロジェクトに追加すると、InstallShield によってそのバンドルに Windows 8 以降または Windows Server 2016 以降の存在を確認する対象条件が自動的に追加されます。この対象条件は、スイート/アドバンスト UI インストールが Windows の以前のバージョンが搭載されているシステム上にパッケージがインストールされないように防ぎます。*

Windows パッケージ マネージャー パッケージをスイート/アドバンスト UI プロジェクトに追加する

プロジェクト- *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション- *アドバンスト UI プロジェクト タイプは、InstallShield で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

Windows パッケージ マネージャー (WPM) パッケージをスイート/アドバンスト UI プロジェクに追加することができます。Windows パッケージ マネージャー (WPM) パッケージをインストールするには、ターゲット システム上に Windows パッケージ マネージャー (WPM) ソリューションがインストールされている必要があります。Windows パッケージ マネージャー(WPM) ソリューションは、アプリ インストーラー MSIX バンドルでインストールされます。Windows パッケージ マネージャー (WPM) パッケージが スイート/アドバンスト UI プロジェクトに追加されると、InstallShield は、アプリ インストーラー MSIX バンドル パッケージを依存ファイルとして追加します。

ターゲットマシン上でアプリ インストーラーが使用できない場合、InstallShield はアプリ インストーラー MSIX バンドル パッケージをインストールします。Windows パッケージ マネージャー は、Windows 10 1809 からアプリ インストーラーを通してサポートされています。

タスク

Windows パッケージ マネージャーをスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[新しい Windows パッケージ マネージャー パッケージ]** をクリックします。

InstallShield の **[パッケージ]** エクスプローラーにパッケージが追加され、右側にあるペインの **[共通]** タブにある **[全般]** 領域で、その設定を構成することができます。**[全般]** 領域を使って、スイート/アドバンスト UI インストールを使ってパッケージ起動する条件とその詳細を指定します。



メモ: *スイート/アドバンスト UI プロジェクトをビルドする前に、**[全般]** 領域の下にある **"WPM パッケージ ID"** 設定で有効な WPM パッケージ ID が不足している場合、Windows パッケージ マネージャー (WPM) パッケージが赤いアイコンで表示されます。*

スイート/アドバンスト UI プロジェクトで Windows パッケージ マネージャー パッケージを構成するときの注意事項

スイート/アドバンスト UI インストールに対して異なるシナリオごとの動作を指示するための条件を定義する必要があります (たとえば、パッケージが既にインストール済みかどうかの検証、要件や依存関係の確認など)。詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージにおけるスタティック ファイルとダイナミック ファイルの違い



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、.msi、.msp、または .exe パッケージを追加または構成するとき、パッケージに、パッケージ ファイルの近くに保存されている追加ファイルが必要かどうかを示すことができます。たとえば、追加するパッケージが非圧縮の .msi パッケージである場合、その他のファイル (近くのサブフォルダーに保存されている .cab ファイルや非圧縮データ ファイルなど) をパッケージとファイルに含める必要が、場合によって、あります。

InstallShield では、パッケージの追加ファイルをプロジェクトに追加するとき、いくつかの方法がサポートされています:

- **すべての追加ファイルにスタティック リンクを使用する**—この方法は、パッケージの作成作業がすべて完了し、変更の可能性がなくなった場合に便利です。この方法は、アドバンスト UI またはスイート/アドバンスト UI インストールを設計中、すべての追加ファイルに対して、名前変更の可能性がほぼなくなった場合にも便利です。ビルド時に、[パッケージ] ビューに表示されているパッケージのすべてのスタティック ファイルが、アドバンスト UI またはスイート/アドバンスト UI インストールにコピーされます。
- **すべての追加ファイルにダイナミック リンクを使用する**—この方法では、追加ファイルのソースの場所を指定します。ビルド時に、ソースの場所に保存されているファイルがすべて、アドバンスト UI またはスイート/アドバンスト UI インストールにコピーされます。

この方法は、パッケージとアドバンスト UI またはスイート/アドバンスト UI インストールの両方を作成しているとき、および、ビルドとビルドの間で、パッケージに必要な追加ファイルのリストに変更が発生する可能性があるときに便利です。

- **追加のファイルにスタティックとダイナミック ファイル リンクを組み合わせて使用する**—この方法では、デザイン時に名前が分かっているすべての重要な追加ファイルに対してスタティック リンクを使い、その他の追加ファイルに対し、ダイナミック リンクを使うことができます。ビルド時に、[パッケージ] ビューに表示されているパッケージのすべてのスタティック ファイルが、アドバンスト UI またはスイート/アドバンスト UI インストールにコピーされます。また、必要に応じて、すべてのダイナミック ファイルもコピーされます。



メモ: *InstallShield では、パッケージ ファイル (.ms、.msp、InstallScript パッケージ、.exe、その他のパッケージ ファイル) に対して、常にスタティック リンクが含まれます。従って、パッケージ ファイルの名前は、ビルドが実行される度に変わることはありません。*



重要: *ダイナミック ファイル リンクの使用は注意が必要です。ダイナミック リンクが参照するソース ファイルからダイナミック リンクがあるファイルを誤って削除してしまった場合、そのファイルは、次回リリースをビルドしたとき、リリースに含まれません。このとき、ビルドの警告やエラーも表示されません。製品は問題なくインストールされる場合がありますが、誤って削除されたダイナミック リンクがあるファイルがインストールされないため、適切に動作しない可能性があります。*

InstallShield で、スタティックのソース場所とダイナミックのソース場所を識別する

スタティック リンク フォルダーが [パッケージ] ビューで表示されたとき、フォルダー アイコンは普通の黄色のフォルダーで表示されます。

ダイナミック リンク フォルダーが [パッケージ] ビューで表示されたときは、フォルダーのアイコン上に、ダイナミックにリンクされていることを示すイメージが表示されます。



InstallShield では、ダイナミック ファイルのイメージの他に、ダイナミックにリンクされているサブフォルダーを 1 つ以上含むフォルダーのアイコン上に、矢印も追加されます。



どちらの特殊フォルダー アイコンをクリックしても、コンテンツのソース場所などの設定を構成できます。また、ビルド時に使用されるフィルターを設定して、特定のファイルをビルドに含めたり、ビルドから除外したりできます。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトでダイナミック リンクを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[パッケージ] ビューを使って、プロジェクト内のパッケージに必要な追加ファイルのダイナミック リンクを作成できます。

ダイナミック ファイル リンクは、パッケージをプロジェクトに追加するときに含めることができます。また、ダイナミック ファイル リンクを、後でプロジェクトに追加することもできます。以下は、その両方の手順です。

パッケージを追加するときにダイナミック ファイル リンクを作成する

[パッケージ] ビューでプロジェクトにパッケージを追加すると、[このパッケージのファイルを追加する] ダイアログ ボックスが表示されます。このダイアログ ボックスで、追加するパッケージ ファイルに、追加するパッケージの近くにあるその他のファイルとフォルダーが必要であるかどうかを指定できます。たとえば、非圧縮の .msi パッケージをプロジェクトに追加する場合、場合によって、その他のファイルおよび、.msi パッケージによってターゲット システムにインストールされるファイルを含むフォルダーを含める必要があります。これらのフォルダーおよびその他のファイルは通常、.msi パッケージが含まれている同じフォルダーに保存されます。

[このパッケージのファイルを追加する] ダイアログ ボックスが表示されたら、次のいずれかを実行します:

- スタティック ファイルのみ含める場合、このダイアログ ボックスの **[ビルド時に見つかったファイルをダイナミックに追加する]** チェックボックスをクリアします。
- ダイナミック ファイルを含める場合は、スタティック ファイルを含める含めないにかかわらず、このダイアログ ボックスの **[ビルド時に見つかったファイルをダイナミックに追加する]** チェックボックスをクリアします。

次に、パッケージ ファイルに相対してソース ファイルがどこに存在するか表しているオプションを選択します。異なるオプションについての詳細は、「[このパッケージのファイルを追加する] ダイアログ ボックス」を参照してください。

パッケージが追加された後、ダイナミック ファイル リンクを作成する

タスク

パッケージがプロジェクトに追加されてから、追加ファイルのダイナミック ファイル リンクを追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、追加ファイルを必要とするパッケージを見つけて、対応するダイナミック リンクを持たせるフォルダーをクリックします。**[ファイル]** ペインで、現在のファイルとダイナミック フォルダーのリストが表示されます。
3. **[ファイル]** ペインで右クリックして **[ダイナミック リンクの追加]** をクリックします。**[フォルダーの参照]** ダイアログ ボックスが開きます。
4. ダイナミックに含めるファイルを含むフォルダーを参照して、**[OK]** をクリックします。

ダイナミック ファイル リンクの構成

いったんダイナミック ファイル リンクをプロジェクトに追加すると、ダイナミック ファイル リンクにファイルを含めるとき、および、除外するときのフィルター基準などの設定を構成することができます。詳しくは、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ダイナミック リンク フォルダーのフィルターを定義する」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ダイナミック リンク フォルダーのフィルターを定義する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

いったんダイナミック ファイル リンクをプロジェクトに追加すると、ダイナミック ファイル リンクにファイルを含めるとき、および、除外するときのフィルター基準を指定することができます。



タスク

ダイナミック ファイル リンクのフィルター基準を指定するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、ダイナミック リンク フォルダーを選択します。
3. **[ファイル]** ペインで、フィルター基準を構成するフォルダーをクリックします。
4. **[ダイナミック リンク]** ペインで、**"フィルター"** 設定の下にあるサブ設定を確認します。
5. 以下のいずれかを実行します。
   - フィルターを追加するには、以下の手順を実行します。
     a. **"フィルター"** 設定で、**[新しいフィルター]** ボタンをクリックして、追加するフィルターの種類に応じて、**[含めるファイル]**、**[除外するファイル]**、**[含めるフォルダー]**、または **[除外するフォルダー]** を選択します。新しい行が追加され、フィルターを定義できます。
     b. 新しい行で、ファイルまたはフォルダーの名前またはパターンを指定します。

        ワイルドカード文字の指定には、アスタリスク (*) を使用します。たとえば、すべての .htm ファイルを含める場合、**"含めるファイル"** 設定で ***.htm** と入力します。すべてのファイルを含める場合、***.*** と入力できます。

        パスでルート フォルダー (パッケージ ファイルを含むフォルダー) を参照する場合、ドットと円記号を使用します:

.¥

たとえば、**MyDirectory** という名のサブフォルダー内 (**MyDirectory** は、パッケージを含むフォルダーのサブフォルダーです) にある **ReadMe.txt** ファイルを参照する場合、次のようなフィルターを使うことができます:

.¥MyDirectory¥ReadMe.txt

- フィルターを変更するには、変更するフィルターの設定で、フィルターの適切な名前またはパターンを入力します。
- フィルター基準の順番を変更するには、移動するフィルターの設定で、**[フィルターの移動]** ボタンをクリックして、**[上に移動]** または **[下に移動]** をクリックします。
- フィルターを削除するには、削除するフィルターの設定で、**[このフィルターを削除]** ボタンをクリックします。

アドバンスト UI またはスイート/アドバンスト UI インストールのビルド時におけるフィルター基準の評価

InstallShield では、ビルド時に、ダイナミック リンク フォルダーに設定されたフィルターが使われて、特定のパッケージのリリースに含めるフォルダーとファイル、および、除外するフォルダーとファイルが判別されます。

InstallShieldで、ダイナミック リンクのソース フォルダー (パッケージを含むフォルダー) が確認されるとき、フィルターは、"フィルター" 設定の下に表示されている順番で上から下へと確認されます。最初にファイルに一致したファイル フィルターが、ファイルをビルドに含めるかどうかを判別するフィルターです。同様に、最初にフォルダーに一致したフォルダー フィルターが、フォルダーのファイルをビルドに含めるかどうかを判別するフィルターです。

たとえば、リスト内の最初のフィルターが次のような場合、**ReadMe.txt** という名前のファイルを含め、ソースフォルダー内にあるすべてのファイルが含められます:



図 2: *.**.* フィルターが ReadMe.Txt フィルターの上に表示されているため、ReadMe.txt "除外するファイル" フィルターは、*.* "含めるファイル" フィルターによってオーバーライドされます。

ReadMe.txt 除外フィルターを *.* インクルード フィルターの上に移動した場合、ビルドに **ReadMe.txt** という名前のファイルを除いてすべてのファイルが含められます。

ダイナミック リンク フォルダーにフィルターが含まれていない場合、ソース フォルダー内のすべてのファイルが含められますが、サブフォルダー内のファイルは無視されます。

InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*

- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield 前提条件 は、製品が必要とする製品、またはテクノロジ フレームワークのためのインストールです。既存の InstallShield 前提条件 (.prq ファイル) は、数に制限なく、アドバンスト UI およびスイート/アドバンスト UI プロジェクトに追加することができます。InstallShield 前提条件を自作で作成して、それをアドバンスト UI およびスイート/アドバンスト UI プロジェクトに追加することもできます。



タスク ***InstallShield 前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含めるには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[前提条件 (.prq) のインポート]** をクリックします。



メモ: *スイート プロジェクトで Windows Package Manger (WPM) 前提条件のインポートはサポートされていません。*

InstallShield 前提条件は、[パッケージ] エクスプローラーに、前提条件に対して実行するように設定されたファイルの種類に応じて、.msi パッケージ、.msp パッケージ、.exe パッケージとして追加されます。

InstallShield 前提条件に依存関係がある場合 (つまり、1 つまたは複数の .prq ファイルが、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する InstallShield 前提条件で、依存関係として指定されている場合)、依存関係にある前提条件が [パッケージ] エクスプローラーに別のパッケージとして自動的に追加されます。



ヒント: *InstallShield 前提条件をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加すると、前提条件パッケージの設定に対して、デフォルトの値が自動的に構成されます。これらの設定は、アドバンスト UI またはスイート/アドバンスト UI プロジェクトでパッケージの設定を変更するのと同様、必要に応じて変更が可能です。詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトの設定を構成する」を参照してください。*



メモ: *アドバンスト UI およびスイート/アドバンスト UI パッケージで Windows Package Manger (WPM) および アプリ パッケージの前提条件はサポートされていません。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトにパッケージとして含まれている InstallShield 前提条件の検出条件と評価条件を確認する

InstallShield 前提条件をアドバンスト UI またはスイート/アドバンスト UI プロジェクトをインポートしたとき、すぐにパッケージの検出条件と評価条件を確認し、必要に応じて変更を加えます。

たとえば、Windows オペレーティング システムのバージョン 5.0 以降すべてをターゲットする InstallShield 前提条件をインポートした場合、前提条件がプロジェクトに追加された時にデフォルトで構成される条件を変更した方が良い場合があります。たとえば、複数存在するプラットフォーム条件を、OS バージョン条件のサブ設定が **5.0–**（バージョン 5.0 以降という意味）に設定されている単一のプラットフォーム条件で置き換えることができます。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるプライマリ パッケージと依存パッケージの違い



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

プロジェクト固有の違いについては、必要に応じて記述されています。



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクト内の各パッケージは、プライマリ パッケージまたは依存パッケージとして識別される必要があります。パッケージの種類によって、そのパッケージがターゲット システムにあることにより、アドバンスト UI またはスイート/アドバンスト UI インストールが初回インストール モードとして実行されるか、またはメンテナンス モードで実行されるかが決定されるかどうかが識別できます:

- **プライマリ パッケージ**—プライマリ パッケージは、アドバンスト UI またはスイート/アドバンスト UI インストールのメインの部分です。

  実行時、ターゲット システム上からインストールのプライマリ パッケージのすべてが不足している場合、インストールは初回インストールとして実行します。ターゲット システム上にスイート インストールのいずれかのプライマリ パッケージが存在する場合、インストールはメンテナンス モードで実行します。

- **依存パッケージ**—依存パッケージがターゲット システムにあるかないかは、インストールが実行されるときのモードに影響しません。

[パッケージ] ビューでパッケージを選択したときに使用できる "パッケージの種類" では、パッケージがプライマリ パッケージであるか依存パッケージであるかが示されます。

スイート/アドバンスト UI プロジェクトでは、プロジェクト内のパッケージに対して設定されている "パッケージの種類" 設定の値を必要に応じて変更することができます。ただし、アドバンスト UI プロジェクトでは、プライマリ パッケージの数が 1 つのみサポートされているため、この設定は読み取り専用になっています。

InstallShield 前提条件をアドバンスト UI プロジェクトまたはスイート/アドバンスト UI プロジェクトにインポートすると、InstallShield 前提条件が依存パッケージとしてプロジェクトに追加されます。スイート/アドバンスト UI プロジェクトでは、適宜 "パッケージの種類" 設定を使って、デフォルト値を上書きし、パッケージを

プライマリ パッケージとして指定することができます。ただし、InstallShield の Professional Edition では、アドバンスト UI プロジェクト内の InstallShield 前提条件は、常に依存パッケージとして識別され、デフォルトの動作を上書きすることはできません。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、パッケージとトランザクションのインストール順序を指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

プロジェクト固有の違いについては、必要に応じて記述されています。



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

スイート/アドバンスト UI プロジェクトに、1 つ以上のパッケージ、トランザクション、または InstallShield 前提条件が含まれている場合、[パッケージ] ビューを使って、実行時のターゲット システムおける、パッケージのインストール順序とトランザクションの起動順序を指定できます。

同様に、アドバンスト UI プロジェクトに、パッケージおよび 1 つ以上の InstallShield 前提条件が含まれている場合、[パッケージ] ビューを使って、実行時のターゲット システムおける、パッケージのインストール順序を指定できます。

この [パッケージ] ビューでは、パッケージ、InstallShield 前提条件 パッケージ、およびトランザクション (サポートされている場合) が、アドバンスト UI またはスイート/アドバンスト UI インストールの実行時と同じ順番で表示されています。



タスク

ターゲット システムにパッケージをインストールする順番を変更するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、移動するパッケージまたはトランザクションを右クリックし、**[上に移動]** または **[下に移動]** をクリックします。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトの設定を構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI インストールにパッケージを追加するとき、設定を構成して、表示名や、アドバンスト UI またはスイート/アドバンスト UI インストールで、パッケージが起動されるときの条件などの情報を指定できます。



タスク

アドバンスト UI またはスイート/アドバンスト UI プロジェクトでパッケージを構成するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、構成するパッケージをクリックします。
3. **[共通]** タブにある設定を使って、パッケージの条件を定義、およびその他の設定を構成します。

パッケージの設定についての詳細は、「共通」を参照してください。

実行時に、[パッケージ] ビューでリストされたパッケージに定義した条件とパッケージの順番に従って、Setup.exe ファイルが各パッケージを起動します。



ヒント: *構成中のパッケージが .msi パッケージである限り、MSI パッケージの対象条件および検出条件を定義する際に、現在のパッケージの製品コード、パッケージ コード、および製品バージョンのプレースホルダーとしてアスタリスクを使用することができます。*

また、構成中のパッケージが .msi パッケージである限り、MSI アップグレードの対象条件および検出条件を定義する際に、現在の製品のアップグレード コードおよび最小バージョン番号/最大バージョン番号のいずれかのプレースホルダーとしてアスタリスクを使用することができます。

アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージを機能に関連付ける



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロ*

ジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれる各パッケージは、プロジェクト内の機能と関連付けなくてはなりません。

タスク ***ターゲット システムの再起動を必要とするアドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージを関連付けるには、次の手順に従います:***

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、構成するパッケージを選択します。
3. **[機能]** タブをクリックします。
4. パッケージを含める各機能のチェック ボックスを選択します。

異なるアドバンスト UI およびスイート/アドバンスト UI インストール間で、共通パッケージを共有する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、複数のアドバンスト UI およびスイート/アドバンスト UI インストール間で、共通パッケージを共有することができます。2 以上のアドバンスト UI およびスイート/アドバンスト UI インストールが 1 つのパッケージを共有している場合に、アドバンスト UI およびスイート/アドバンスト UI 製品のすべてが削除されるまで、ターゲット システムにそのパッケージが保持されるようにします。

パッケージを共有としてマークするかどうかを判別するときのガイドライン

アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージを構成するとき、これを共有とマークすることができます。パッケージを共有とマークする場合、パッケージを共有するすべてのアドバンスト UI およびスイート/アドバンスト UI プロジェクトで、必ず同じパッケージ GUID を使用してください。

共有パッケージの機能は、通常プライマリ パッケージではなく、依存パッケージで使用できます。.NET Framework をインストールするパッケージなど、広く配布されるパッケージは、ほとんどの場合ターゲット システムにインストールされたままとなるため、通常共有とマークされません。

共有機能はほとんどの種類のパッケージで使用できます (.msi、.msp、.exe、.appx、InstallScript、基本の MSI プロジェクト、および InstallScript プロジェクト)。ただし、通常アンインストールされることがない Web 配布パッケージでは使用できません。

パッケージが共有かどうかを指定する

タスク ***アドバンスト UI またはスイート/アドバンスト UI プロジェクトでパッケージの共有を指定するには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、構成するパッケージを選択します。
3. **"共有"** 設定で、適切な値を選択します。
   - **いいえ**—パッケージは別のアドバンスト UI またはスイート/アドバンスト UI プロジェクトと共有されません。これがデフォルトの値です。
   - **はい**—パッケージは別のアドバンスト UI またはスイート/アドバンスト UI プロジェクトと共有されます。

     "共有" 設定に **[はい]** を選択した場合、パッケージを共有するすべてのアドバンスト UI およびスイート/アドバンスト UI プロジェクトで必ず同じ パッケージ GUID を使用してください。**"パッケージ GUID"** 設定を使って、パッケージの GUID を確認および変更することができます。

共有パッケージの実行時の動作

実行時、アドバンスト UI またはスイート/アドバンスト UI インストールが既存しない共有パッケージをインストールするとき、インストールは次のレジストリ キーの下にある共有パッケージのパッケージ GUIDの 新しいレジストリ キーを作成します:

- 32 ビット システムの場合: HKEY_LOCAL_MACHINE¥SOFTWARE¥InstallShield¥SuiteInstallers¥Parcels
- 64 ビット システムの場合: HKEY_LOCAL_MACHINE¥SOFTWARE¥Wow6432Node¥InstallShield¥SuiteInstallers¥Parcels

インストールはまた、スイート GUID をパッケージ GUID レジストリ キーのレジストリ データとして追加します。別のアドバンスト UI またはスイート/アドバンスト UI インストールで同じ共有パッケージをインストールする必要がある場合、これらのアドバンスト UI またはスイート/アドバンスト UI インストールのスイート GUID がパッケージ GUID レジストリ キーのレジストリ データに追加されます。任意のアドバンスト UI またはスイート/アドバンスト UI プロジェクトのアンインストール中に、スイート GUID がパッケージ GUID レジストリ キーから削除されます。共有パッケージおよびパッケージ GUID レジストリ キーは、最後のアドバンスト UI またはスイート/アドバンスト UI 製品が削除されるまでターゲット システムに残ります。

2 つのアドバンスト UI またはスイート/アドバンスト UI インストールが、同じパッケージの異なるバージョンを共有する場合、どちらのアドバンスト UI またはスイート/アドバンスト UI インストールが先に実行されたかにかかわらず、新しいバージョンのパッケージは両方が実行された後にターゲット システムにインストールされたバージョンです。また、インストールは必要に応じて Parcels レジストリ キーをアップデートします。共有パッケージとパッケージ GUID レジストリ キーは、最後のアドバンスト UI またはスイート/アドバンスト UI 製品が削除されるまでターゲット システムに残ります。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに対してランタイムの場所を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、プロジェクト内の各パッケージに対して、異なるランタイムの場所を指定することができます。たとえば、パッケージとそのファイルをソース メディアに格納するか、必要な場合に Web からダウンロード可能にできます。または、パッケージをアドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルに圧縮して、実行時に必要な場合に抽出することができます。

タスク

プロジェクト内の各パッケージに対して、異なるランタイムの場所を指定するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、構成するパッケージを選択します。
3. **[場所]** リストで、適切なオプションを選択します。

指定した場所は、リリース レベルでオーバーライドすることができます。個々のパッケージに選択した値のオーバーライドを禁止するには、リリース レベルの "パッケージの場所" 設定を [個々の選択に従う] に設定します。詳細については、「リリース レベルで、アドバンスト UI またはスイート/アドバンスト UI パッケージのランタイムの場所を指定する」を参照してください。

ヒント: *アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、アップデート セットアップ ランチャーの構成を行う場合、パッケージのランタイムの場所は、セットアップ ランチャーから抽出するか、Web からダウンロードします。アップデート セットアップ ランチャーは、ソース メディアに格納されているパッケージには依存できません。*

詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージ操作を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

アドバンスト UI プロジェクトでは、.exe パッケージが InstallShield 前提条件からプロジェクトに含められた場合のみサポートされています。

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI の操作とは、アドバンスト UI またはスイート/アドバンスト UI インストールが実行されているときの状態を意味します。操作には、インストール、削除、修復、変更、および保守という異なる種類があります。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトにパッケージを追加すると、そのパッケージに対して、様々な操作設定を構成することができます。設定を使って、操作ごとに、それらがパッケージに適用可能かどうかを指定できます。また、設定を使って、モード別に、アドバンスト UI またはスイート/アドバンスト UI のインストールで、パッケージが起動されたときに使用するコマンドラインを指定することもできます。コマンドラインは、アドバンスト UI またはスイート/アドバンスト UI インストールがユーザー インターフェイスと共に実行されたときのコマンドライン、または、サイレント (ユーザー インターフェイスなし) で実行されたときのコマンドラインと、別々に指定することができます。

インストール操作

アドバンスト UI またはスイート/アドバンスト UI インストールでは、パッケージをターゲット システムにインストールするとき、インストール操作が実行されます。インストール操作は、初回インストール、および、メンテナンス モードでインストールする機能が選択されたときに発生します。すべてのパッケージには、通常、インストール操作が設定されています。

アドバンスト UI またはスイート/アドバンスト UI インストールで、.msi または .msp、または InstallScript パッケージに対してインストール操作が実行されるとき、それらのパッケージは自動的にサイレントで起動されます。ただし、アドバンスト UI またはスイート/アドバンスト UI インストールで、.exe パッケージに対してインストール操作が実行されるときは、異なる種類の .exe パッケージで、ユーザー インターフェイスなしの異なる方法が使用される可能性があるため、パッケージを自動的にサイレントで起動することはできません。アドバンスト UI またはスイート/アドバンスト UI インストールで、.exe パッケージをサイレントで実行する必要がある場合、そのパッケージのインストール操作のサブ設定で適切なコマンドラインを入力します。

エンドユーザーによって構成可能なアドバンスト UI またはスイート/アドバンスト UI のパッケージが存在する場合、エンドユーザーがプロパティを構成できるウィザード ページを作成し、コマンドラインを使って、その関連付けられたプロパティをそれらのパッケージに渡します。

削除操作

アドバンスト UI またはスイート/アドバンスト UI インストールでは、パッケージをターゲット システムからアンインストールするとき、削除操作が実行されます。プライマリ パッケージとしてマークされているパッケージには、通常、削除操作が設定されています。逆に、依存パッケージとしてマークされているパッケージは、通常ターゲット システムにそのまま残されるため、概して削除操作は設定されていません。.NET Framework をインストールするパッケージは、依存パッケージとしてマークされるパッケージの 1 つの例です。パッケージの "パッケージの種類" 設定では、パッケージがプライマリ パッケージであるか、または依存パッケージであるかが識別されます。

メンテナンスの実行時に、アドバンスト UI またはスイート/アドバンスト UI の起動が無効にされたとき、また、アドバンスト UI またはスイート/アドバンスト UI 製品がアンインストールされたとき、削除操作が実行されます。インストール時同様、パッケージは、.exe パッケージのコマンドラインを使って、サイレントで実行します。

削除操作は、.msp パッケージには適用できませんので注意してください。また、.msp パッケージを削除するとき、ベースの .msi パッケージを削除する必要があることにも注意してください。

修復操作

アドバンスト UI またはスイート/アドバンスト UI インストールでは、一部のパッケージでサポートされている修復機能を実行する修復操作を実行することができます。修復機能をサポートするアドバンスト UI またはスイート/アドバンスト UI パッケージを構成する場合、パッケージの修復操作設定を構成して、修復機能を有効にします。アドバンスト UI またはスイート/アドバンスト UI インストールでは、アドバンスト UI またはスイート/アドバンスト UI インストールがメンテナンスの一部として修復されるとき、および、パッケージが既にインストールされている場合にのみ、修復操作が実行されます。

.msi パッケージに対して修復操作を構成する場合、アドバンスト UI またはスイート/アドバンスト UI インストールでは、修復機能が必要に応じて自動的に起動することが可能なため、サイレントで修復を実行する特別のコマンドラインを指定する必要はありません。ただし、.exe パッケージに対して修復操作を構成する場合、適切なコマンドラインを入力する必要があります。たとえば、基本の MSI の Setup.exe パッケージの場合、"修復操作" 設定の下にある "EXE コマンドライン" と "EXE サイレント コマンド ライン" サブ設定で次のコマンドラインを入力できます:

[SystemFolder]msiexec.exe /f {8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC} /qn

カッコ内の GUID は、Windows Installer パッケージの製品コードです。

修復操作は、.msp パッケージに適用できませんので注意してください。

変更操作

アドバンスト UI またはスイート/アドバンスト UI インストールでは、機能選択の実行に変更操作が実行されます。機能の選択を提供するアドバンスト UI またはスイート/アドバンスト UI パッケージを構成する場合、パッケージの変更操作設定を構成して、変更を有効にします。変更操作は、アドバンスト UI またはスイート/アドバンスト UI インストールの機能の選択で、パッケージの機能の追加または削除が指示されたときに実行されます。

変更操作は、.msp パッケージに適用できませんので注意してください。

保守操作

アドバンスト UI またはスイート/アドバンスト UI インストールでは、次のようなシナリオで、メンテナンス操作が実行されます:

- エンドユーザーが、[プログラムの追加と削除] のエントリから製品の変更を選択したとき。

- エンドユーザーが、アドバンスト UI またはスイート/アドバンスト UI インストールを再実行したとき。

保守操作は、操作設定から制御できないという点で、他の種類の操作と異なります。

アドバンスト UI または スイート/アドバンスト UI インストールのパッケージにカスタム フォルダー名を使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI インストールをビルドしたとき、インストールに含まれている各パッケージに対してフォルダーが 1 つ作成されます。これらのフォルダーは、アドバンスト UI またはスイート/アドバンスト UI のセットアップ ランチャーと同じフォルダー内で作成されます。デフォルトで、各フォルダーの名前には、InstallShield で生成された GUID が使用されます。

InstallShield の [パッケージ] ビルドでは、今回から、これらのパッケージ フォルダーに対して、この GUID 名をオーバーライドして、ユーザーに分かりやすい名前を使用できるようになりました。



タスク

パッケージ フォルダーの名前に任意のテキストを指定するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、フォルダー名をカスタマイズするパッケージのノードを展開します。
3. **[パッケージ ファイル]** フォルダーを右クリックして、**[名前の変更]** をクリックします。

   メモ・ *デフォルト名 (**パッケージ ファイル**) をそのまま使用する場合、パッケージ フォルダーの名前に GUID が使用されます。*
4. フォルダーの新しい名前を入力します。複数のパッケージ フォルダー名をカスタマイズする場合、各フォルダー名に異なる名前を使用してください。

ビルド時に、リリースが作成された時、パッケージ フォルダーの名前に、GUID ではなく、入力されたテキストが使用されます。

スイート/アドバンスト UI インストールの動作をカスタマイズする



プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

スイート/アドバンスト UI インストールを作成するとき、製品の要件およびエンド ユーザーのニーズに応じてカスタマイズすることが重要です。InstallShield では、スイート/アドバンスト UI インストールで、インストールに含めるパッケージの範囲を超えた様々なランタイム タスクを実行する機能を拡張することができます。詳細については、次を参照してください。

- スイート/アドバンスト UI インストール中に Windows 役割と機能を有効化する
- スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する

スイート/アドバンスト UI インストール中に Windows 役割と機能を有効化する



プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

一部の Windows の機能は、ターゲット システム上でデフォルトで無効です。また、エンド ユーザーまたは IT 管理者は、一部の Windows の機能を無効にしたい場合があります。そのような Windows の役割と機能を手動で有効または無効にするには、[コントロール パネル] の [プログラムと機能] の **[Windows の機能の有効化または無効化]** を使います。

スイート/アドバンスト UI インストール内の特定のパッケージが、ターゲット システム上で 1 つ以上の Windows の役割または機能が有効化されていることを必要とする場合、スイート/アドバンスト UI プロジェクトでパッケージを構成中に、必要な役割と機能を指定できます。実行時、インストールされる対象のパッケージが無効化されている 1 つ以上の Windows の役割または機能を必要とする場合、スイート/アドバンスト UI インストールは、パッケージを起動する前にこれらの役割および機能を有効化します。

たとえば、スイート/アドバンスト UI プロジェクトで Windows のインターネット インフォメーション サービス機能が有効であることを必要とする IIS Web サイトをインストールパッケージがあります。同じプロジェクト内の別のパッケージでは、PowerShell 機能が有効であることが必要です。プロジェクトのこれらのパッケージの設定

を構成するとき、必要な Windows 機能を指定します。実行時、パッケージがターゲット システム上でインストールの対象となっているが、必要な Windows 機能のいずれかが無効である場合、スイート/アドバンスト UI インストールは、起動前にこれらの無効化されている Windows 機能を有効にします。パッケージがインストールの対象ではない場合、必要な Windows 機能は有効化されません。

スイート/アドバンスト UI インストールがインストール中に有効化した任意の Windows の役割と機能は、アンインストール中に削除されません。そのため、ターゲット システム上でその他の製品が必要とするこれらの役割および機能の破損を回避することができます。

InstallShield には、いくつかの Windows 機能のビルトインサポートが搭載されています:

- インターネット インフォメーション サービス
- PowerShell
- .NET Framework 3.x

InstallShield ではまた、パッケージに必要な上記以外の Windows 役割と機能を指定することもできます。



タスク

特定のパッケージがインストールの対象であるときに有効化する Windows の役割または機能を指定するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、構成するパッケージをクリックします。
3. **[共通]** タブにある **"Windows の機能"** 設定で **[Windows の機能を追加]** ボタンをクリックしてから、有効な機能オプションの 1 つを選択します:
   - **インターネット インフォメーション サービス**
   - **PowerShell**
   - **Microsoft .NET Framework 3.x**
   - **カスタム** – オプションを使って、この設定にオプションとしてリストされていない任意の Windows の機能を指定できます。

   InstallShield は、**"Windows 機能"** 設定の下に新しい **"Windows の機能"** 行を追加し、任意のビルトイン機能オプション (インターネット インフォメーション サービス、PowerShell、または Microsoft .NET Framework 3.x) の必要に応じて構成します。**[カスタム]** オプションでは、**[Windows の機能]** 行には機能名のプレースホルダ—文字列が含まれます。
4. ビルトイン オプションの 1 つを選択した場合、InstallShield によって **"Windows 機能"** 設定に追加された値をそのままに残します。

   [カスタム] オプションを選択した場合: **"Windows の機能"** 設定に、有効にする Windows の役割または機能の名前を入力します。カスタム オプションに入力する名前はすべて、Deployment Image Servicing and Management (DISM.exe) および Package Manager (Pkgmgr.exe) などのツールが Windows の特定のバージョンでロールまたは機能を識別するのに使用する文字列です。

   各 Windows のバージョンで使用可能な Windows の役割および機能の一覧は、Microsoft TechNet を参照してください。

Windows の役割および機能を有効化する際の特別考慮

様々な Windows バージョンで Windows の役割と機能が使用可能であるかどうか

Windows の役割と機能を有効化できるサポートは、Windows Vista 以降、または Windows Server 2008 以降を搭載するターゲット システム上で使用できます。指定のターゲット システム上で有効化することができる Windows の役割および機能のリストは、存在する Windows のバージョンによって異なります。また、異なるバージョンの Windows では、同じロールまたは機能に異なる文字列を使用する場合があります。

スイート/アドバンスト UI インストールがターゲット システム上で使用できない Windows の役割または機能を有効化しようと試みた場合、スイート/アドバンスト UI インストールは失敗して、ログファイルにエラー、0x800f080c が書き込まれます。

したがって、必要な Windows の役割と機能を指定する一部の状況下で、パッケージの "対象条件" 設定を使って、パッケージに選択している必要な Windows の役割または機能に対応する、適切なオペレーティング システムのバージョンを識別したい場合があります。有効にする Windows の役割または機能の名前にそれぞれ異なる文字列を使った複数バージョンの Windows をターゲットとするには、複数回にわたってパッケージをプロジェクトに追加する方法を考慮してください。パッケージの各インスタンスで、"Windows の機能" および "対象条件" 設定を使って、ターゲット オペレーティング システムに対応する適切な文字列を指定します。

Windows の一部のバージョンで、一部のロールおよび機能が無効化されないだけでなく、インストールもされない場合があります。スイート/アドバンスト UI インストールがターゲット システム上にインストールされていない Windows の役割または機能を有効化するように構成した場合、スイート/アドバンスト UI インストールはその役割または機能のインストールを試みます。ロールまたは機能をインストールできないとき、スイート/アドバンスト UI インストールは失敗します。このシナリオが当てはまる Windows の機能の例として .NET Framework 3.x 機能があり、これは Windows 8 システム上ではデフォルトでインストールされません。この機能のペイロードは、Windows アップデートを通してのみ使用可能です。インターネットへの接続が利用できないとき、または Windows アップデートの代わりに Windows Server Update Service を通してアップデートを取得するように構成されている場合、スイート/アドバンスト UI インストールで .NET Framework 3.x 機能をインストールおよび有効化することはできません。

必要な Windows の役割または機能の依存関係を識別する

一部の Windows の役割および機能は、その他の Windows の役割または機能に依存します。無効化されているロールまたは機能に依存関係があるかどうかを判別する方法の 1 つとして、次のコマンドを権限が昇格されたコマンド プロンプト ウィンドウ (cmd.exe のショートカットを右クリックしてから **[管理者として実行]** をクリック) から実行する方法があります:

```
dism /online /enable-feature FEAT-NAME
```

この例で、FEAT_NAM はチェックする無効化された機能の名前です。前述のコマンドが動作する場合、機能には依存関係がありません。これが失敗するとき、エラーに依存関係がリストされます。

パッケージで有効化するロールまたは機能が、ターゲット システムで無効化されている可能性のある 1 つ以上のロールまたは機能を必要とする場合、必ずそのパッケージの "Windows の機能" 設定で依存関係の機能も指定してください。

Windows の役割または機能を有効化するランタイムの動作

スイート/アドバンスト UI インストールで、一部のターゲット システム上で Windows の役割または機能を有効化する場合、スイート/アドバンスト UI インストールの InstallationProgress ウィザード ページで有効化処理中の役割または機能の進行状況を表示できます。この進行状況の情報は、Windows 8 以降または Windows Server 2012

以降が搭載されたターゲット システム上で使用できます。これらのターゲット システム上で、スイート/アドバンスト UI インストールは DISM API を使ってロールおよび機能を有効化し、これらの API には進行状況の情報をレポートする機能をサポートします。

Windows Vista、Windows Server 2008、Windows 7、および Windows Server 2008 R2 ターゲット システム上で、スイート/アドバンスト UI インストールは Pkgmgr.exe を使ってロールおよび機能を有効化します。Pkgmgr.exe は、進行状況の情報をレポートする機能をサポートしません。したがって、これらのシステム上ではロールおよび機能の有効化処理中に進行状況バーを更新することはできません。

Windows 7 および Windows Server 2008 R2 システム上で DISM.exe を使用できますが、これらの Windows バージョンでは API サポートが提供されていません。また、API DLL 依存関係は、ターゲット システムで再配布することができません。

スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する



プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

スイート/アドバンスト UI インストールで、インストールに含めるパッケージの範囲を超えた様々な実行時タスクが必要な場合があります。たとえば、スイート/アドバンスト UI インストールで、以下の 1 つ以上の操作が必要な状況が考えられます。

- スイート/アドバンスト UI インストールのユーザー インターフェイスが表示される前にアプリケーションをインストールする。

  この状況が必要な例として、スイート/アドバンスト UI インストール内のパッケージの 1 つが Oracle データベースをインストールする SQL スクリプトを実行する時があります。このパッケージを実行する前に、スイート/アドバンスト UI インターフェイスで、エンド ユーザーがネットワーク上で使用可能なサーバーの一覧から適切なサーバーを選択できるウィザード ページを表示したい場合があります。この種類の UI サポートを使用するには、ターゲット システム上に ODBC ドライバーがインストールされていることが必要です。

- 特定の製品、テクノロジ、フォルダー、ファイル、レジストリ エントリ、その他のアイテムの存在の有無をターゲット システム上で検索する。

  ターゲット システムを検索するサポートを使って、特定の条件が満たされているかどうかに基づいて動作をトリガすることができます。たとえば、ターゲット システム上で特定のファイルが不足している場合、スイート/アドバンスト UI プロパティを適切に設定してから、そのプロパティを使ってスイート/アドバンスト UI インストールに含まれる 1 つ以上のパッケージでプロパティを設定できます。

- スイート/アドバンスト UI インストール内のパッケージを実行する前後にターゲット システムを構成する。

一部の状況、たとえばスイート/アドバンスト UI インストールがサーバー上の IIS アプリケーションをインストールする場合、ターゲット システム上で IIS 関連の Windows の役割および機能を有効化したい場合があります。その他、エンド ユーザーがスイート/アドバンスト UI インストールに含まれるパッケージのいずれを実行するか選択した内容によって、ターゲット システム上の構成ファイル、レジストリ データ、その他の項目を編集したい場合があります。

- エンド ユーザーによって、特定のウィザード インターフェイス コントロールを使用したとき、またはウィザード ページ または 2 番目のウィンドウが開いた/閉じたときにトリガされる初期設定を行う

  たとえば、ウィザード インターフェイス上のコンボ コンボ ボックスにアクションを追加して、コンボ ボックスに値をダイナミックに挿入したい場合があります。

スイート/アドバンスト UI インストールの機能を拡張して、前述のタスクその他を実行できるようにするには、スイート/アドバンスト UI プロジェクトの [イベント] ビューを使って、実行時に実行可能ファイルの実行、DLL 関数の呼び出し、PowerShell スクリプトの実行、スイート/アドバンスト UI プロパティの設定、InstallScript コードの実行、またはマネージ アセンブリのパブリック メソッドの呼び出しを行うアクションを作成することができます。

タスク ***アクションを作成してスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:***

1. インストールで必要な機能を処理する独自の .exe ファイル (適切な場合) 、DLL ファイル、PowerShell スクリプト、InstallScript ファイル (.rul)、またはマネージ アセンブリ (Visual Basic .NET または C# といったマネージ コードで作成) を作成ます。

   InstallScript アクションの場合、プロジェクトの [InstallScript] ビューを使います。

2. [イベント] ビューで、ステップ 1 で作成したファイルを使用するアクションを作成します。詳細については、「アクションをスイート/アドバンスト UI プロジェクトに追加する」を参照してください。

3. アクションを実行するタイミングを決めて、スケジュールします。そのためには、アクションをイベントに追加するか、ウィザード インターフェイス要素に追加するか、またはそれをプロジェクト内の特定のパッケージに含まれるイベントに割り当てます。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

スイート/アドバンスト UI インストールに含まれるアクションの .exe ファイルでの作業について

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

スイート/アドバンスト UI インストールで .exe アクションを使って、インストールに含まれている (一時サポート ファイル、または製品とともにインストールされる) またはターゲット システムに既存する実行可能ファイルを起動できます。

次に、スイート/アドバンスト UI インストールに .exe アクションを追加する例を説明します。

- 独自の、またはサード パーティが作成した Setup.exe インストールを起動する .exe アクションを追加する。
- 既存ディレクトリのアクセス許可を設定する .exe アクションを追加する。
- ReadMe ファイルを、それに関連付けられたアプリケーション (メモ帳、または Adobe Reader) で開く .exe アクションを追加する。
- 製品に含まれるデータ ファイルを開く .exe アクションを追加する。

.exe アクションを スイート/アドバンスト UI プロジェクトに追加する最初の手順は、(適切な場合) .exe ファイルを作成または取得するか、実行可能ファイルが起動するドキュメントその他のファイルを作成することです。

.exe アクションは、実行中のインストール セッションにアクセスすることができません。したがって、(コマンド ライン引数を除いて) スイート/アドバンスト UI プロパティを .exe ファイルに渡したり、.exe ファイルから戻したりはできません (レジストリやファイルなどの外部ストレージ経由の場合を除く)。

タスク

スイート/アドバンスト UI インストールで実行可能ファイルを実行するアクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーを右クリックしてから、**[新しい EXE]** をクリックします。**[アクション]** エクスプローラーに実行可能ファイル アクションが追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このアクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。
4. 必要に応じて、アクションの設定を構成します。

プロジェクトにアクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

メモ: *.exe アクションからインタラクティブなアプリケーションを起動する場合、スイート/アドバンスト UI インストールがサイレント (ユーザー インターフェイスなし) で実行中に、それが起動することを防ぐための条件をアクションに追加した場合があります。この種類の条件で* ISSilentInstall *プロパティを使用できます。スイート/アドバンスト UI インストールがサイレント (ユーザー インターフェイスなし) で実行している場合、このプロパティは True に設定します。*

別の方法として、インストールのユーザー インターフェイスからインタラクティブなアプリケーションを起動したい場合もあります。つまり、.exe アクションを使って PDF ファイルを起動する代わりに、インストールのユーザー インターフェイスに含まれるウィザード ページの 1 つから PDF ファイルを開く機能を追加することができます。

スイート/アドバンスト UI インストールに含まれるアクションの DLL ファイルでの作業について

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DLL アクション用の DLL ファイルを作成する

スイート/アドバンスト UI プロジェクトに DLL アクションを追加する最初の手順は、DLL の作成です。DLL は、Visual C++ の最近のバージョン、または、COM および DLL のエクスポートをサポートしているすべてのツールや言語を使って作成できます。

スイート/アドバンスト UI エンジンでは、各アクションにエントリ ポイントが必要です。エントリ ポイントを指定するには、それを [イベント] ビューにあるアクションの "関数名" 設定に入力します。関数は、次のように定義します:

```
HRESULT _stdcall MyFunction(IDispatch *pAction);
```

この関数をアドバンスト UI またはスイート/アドバンスト UI インストールから呼び出せるようにするには、関数を適切にエクスポートする DLL をビルドするときに、定義ファイル (.def) を含める必要があります。次は、サンプル .def ファイルのコンテンツです。LIBRARY の後の名前は、DLL に使用した名前です。

```
LIBRARY "MyActionLibrary"
EXPORTS
    MyAction
```

スイート/アドバンスト UI インストールは、インストールに含まれるイベントまたはパッケージのアクションの実行がスケジュールされているときに関数のエントリ ポイントを呼び出します。エントリ ポイント関数は、アクションが正しく終了したことを示す ERROR_SUCCESS を戻します。アクションが失敗した場合、ゼロ以外の値が戻されます。値は、Windows Installer カスタム アクションが戻す値と同じです。

スイート/アドバンスト UI インストールでは任意のアクションを実行できますが、アクションはそれに構成されている権限を使って実行されます。一部のアクションでは、管理者権限が必要な場合があります (IIS 7.x 構成データの読み取りなど)。その場合、アクションが必要なデータに正しくアクセスできるように、アクションが管理者権限を要求するように構成する必要があります。つまり、[イベント] ビューにあるアクションの "管理者権限が必要" 設定で [はい] を選択する必要があります。別の方法として、スイート/アドバンスト UI インストールの Setup.exe ファイルを管理者権限を使って実行するか、管理者マニフェストを含みます。ただし、管理者権限を昇格するのは、できるだけ短時間に抑える必要があります。

アクションのエントリ ポイントに渡される IDispatch インターフェイスは、ISuiteExtension2 インターフェイス (すべての ISuiteExtension メソッドとプロパティを含む) を実装します。ISuiteExtension2 へのポインターを取得するには、IDispatch インターフェイスで QueryInterface メソッドを呼び出します。ISuiteExtension2 インターフェイスでは、アクション関数を使って、スイート/アドバンスト UI プロジェクトでアクションに定義された属性パラメーターにアクセスすることができます。プロジェクト内の各アクションは、それぞれのアクション インスタンスに固有の異なるインターフェイス ポインターが渡されます。したがって、エントリ ポイント関数に渡されたインターフェイス ポイントを常に使用し、それを他のアクション DLL の呼び出し用に保存しないようにしてください。

インターフェイスは、次のように定義されています。

```
interface ISuiteExtension2 : IDispatch {
  [propget, id(1), helpstring("Attribute")]
    HRESULT Attribute([in]BSTR bstrName, [out, retval]BSTR *bstrValue);
```

```
  [id(2), helpstring("method LogInfo")]
    HRESULT LogInfo([in]BSTR bstr);

  [propget, id(3), helpstring("Property")]
    HRESULT Property([in]BSTR bstrName, [out, retval]BSTR *bstrValue);

  [propput, id(3), helpstring("Property")]
    HRESULT Property([in]BSTR bstrName, [in]BSTR bstrValue);

  [id(4), helpstring("FormatProperty")]
    HRESULT FormatProperty([in]BSTR bstrValue, [out, retval]BSTR *bstrReturnValue);

  [id(5), helpstring("ResolveString")]
    HRESULT ResolveString([in]BSTR bstrStringId, [out, retval]BSTR *bstrReturnValue);

  [id(6), helpstring("ProgressMessage")]
    HRESULT SendProgressMessage([in]BSTR bstrMsg, [in]INT iCurrent, [in]INT iMax,
                                [in]EnumProgressFlags eFlags);
};
```

スイート/アドバンスト UI プロジェクトでは、関数プロトタイプを使って、ISuiteExtension2 COM インターフェイス ポインターがアクションに渡されます。これにより、次の関数にアクセスできるようになります:

- **LogInfo**

  HRESULT LogInfo([in]BSTR bstr);

  このメソッドにより、スイート/アドバンスト UI デバッグ ログに必要な情報を書き込みし、それらをデバッグやその他の情報提供目的に使用できるようにします。bstr パラメーターには、ログに書きこまれる文字列が含まれています。

- **get_Property**

  HRESULT get_Property([in]BSTR bstrName, [out, retval]BSTR *bstrValue);

  このメソッドによって、現在実行中のスイート/アドバンスト UI インストールで定義されているプロパティの値が取得されます。定義されていないプロパティは、空の値を返します。bstrName パラメーターによって、値が取得されるプロパティの名前が指定されます。プロパティの値は、bstrValue パラメーターで返されます。

- **put_Property**

  HRESULT put_Property([in]BSTR bstrName, [in]BSTR bstrValue);

  このメソッドを使って、現在実行中のスイート/アドバンスト UI インストールで、新しいプロパティの値を設定したり、既存のプロパティの値を変更したりできます。空の値を渡すと、その結果として、プロパティが削除されます。bstrName パラメーターによって、設定するプロパティの名前が指定されます。bstrName パラメーターによって、設定するプロパティの名前が指定されます。

- **FormatProperty**

  HRESULT FormatProperty([in]BSTR bstrValue, [out, retval]BSTR *bstrReturnValue);

  この方法を使うと、bstrValue パラメーターで提供された文字列内に組み込まれた形式化された式を解決することができます。フォーマットされた値は、bstrReturnValue パラメーターで戻されます。形式化された式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

- **ResolveString**

```
HRESULT ResolveString([in]BSTR bstrStringId, [out, retval]BSTR *bstrReturnValue);
```

このメソッドはスイート/アドバンスト UI の文字列 ID を、現在選択されている UI 言語で実行中されているスイート/アドバンスト UI インストールの対応する文字列値に解決します。bstrStringId パラメーターによって、解決する文字列 ID が指定され、解決された文字列は bstrReturnValue で返されます。指定された文字列 ID が存在しない場合、文字列は空の状態で戻されます。

- **SendProgressMessage**

```
HRESULT SendProgressMessage([in]BSTR bstrMsg, [in]INT iCurrent, [in]INT iMax,
                            [in]EnumProgressFlags eFlags);
```

このメソッドは、スイート/アドバンスト UI インストールで InstallationProgress ウィザード ページのステータス メッセージと進行状況バーを更新します。進行状況バーを更新するには、eFlags の epfProgressValid を指定し、ステータス メッセージを更新するには、epfMessageValid を指定します。ステータス メッセージと進行状況バーの両方を更新するには、両方のフラグと共にビット単位の OR 演算子 (|) を使用します。

get_Attribute メソッドはスイート/アドバンスト UI インストールの DLL アクションでは使用できないので、これを呼び出すことはできません。

作成した DLL から ISuiteExtension2 インターフェイスにアクセスするには、#import を使って、InstallShield と共にインストールされている SetupSuite.exe ファイルからタイプ ライブラリ情報を組み込むことができます。次は、そのパスです:

InstallShield Program Files フォルダー¥Redist¥Language Independent¥i386¥SetupSuite.exe"

たとえば、VC++ プロジェクト内のタイプ ライブラリ (例、stdafx.h) をインポートするには、次のステートメントを使用します:

```
#import "C:¥Program Files¥InstallShield¥2022¥Redist¥Language Independent¥i386¥SetupSuite.exe" no_namespace raw_interfaces_only named_guids
```

InstallShield が別の場所にインストールされている場合、#import ステートメントのパスを、それに従って変更する必要があります。

DLL アクションを追加する



タスク

スイート/アドバンスト UI インストールの実行中に DLL ファイルの関数を呼び出すアクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーを右クリックしてから、**[新しい DLL]** をクリックします。**[アクション]** エクスプローラーに DLL アクションが追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このアクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。
4. 必要に応じて、アクションの設定を構成します。

プロジェクトにアクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

スイート/アドバンスト UI インストールに含まれるアクションの PowerShell スクリプトでの作業について

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

Windows PowerShell は、構成タスクのオートメーション化を可能にする .NET Framework ベースのコマンドライン シェルおよびスクリプト言語です。InstallShield では、PowerShell スクリプト (.ps1) を実行するスイート/アドバンスト UI インストールのアクションを含めることができます。この種類のアクションは、インストールの実行時にシステムの構成タスクを実行するプロジェクトに追加することができます。PowerShell アクションには、ターゲット システム上に PowerShell 2.0 以降が必要です。

PowerShell アクションを追加する

タスク

スイート/アドバンスト UI インストールで PowerShell スクリプトを実行するアクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーを右クリックしてから、**[新しい PowerShell]** をクリックします。**[アクション]** エクスプローラーに PowerShell アクションが追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このアクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。
4. 必要に応じて、アクションの設定を構成します。

[アクション] エクスプローラー、または [イベント] エクスプローラーで PowerShell アクションを選択すると、InstallShield の右側に [PowerShell スクリプト] タブが表示されます。このタブを使ってスクリプトを編集できます。

プロジェクトにアクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

PowerShell アクションの実行時要件

スイート/アドバンスト UI プロパティ IS_CLR_VERSION を使って、PowerShell スクリプトを実行するためにアクションがロードする、セミコロン区切りの .NET Framework バージョン一覧を識別することができます。IS_CLR_VERSION プロパティについては、「 アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」を参照してください。

実行中のスイート/アドバンスト UI インストールとのインタラクション

実行中の スイート/アドバンスト UI インストールとのインタラクションを可能にする、いくつかの cmdlet があります:

テーブル 2 ・ 実行中のスイート/アドバンスト UI インストールとインタラクションする cmdlet

Cmdlet	説明
get-suiteproperty -name [プロパティ名]	cmdlet がスイート/アドバンスト UI プロパティの値を取得します。Name パラメーターは、位置指定パラメーターです。 次のサンプル コードは、MYPROPERTY という名前のプロパティの値を取得します。 $Value = get-suiteproperty -name MYPROPERTY
set-suiteproperty -name [プロパティ名] -value [値]	この cmdlet がスイート/アドバンスト UI プロパティの値を設定します。 次のサンプル コードは、MYPROPERTY プロパティの値を使って、NEWPROPERTY と呼ばれる 2 番目のプロパティの値を設定します。 $Value = get-suiteproperty -name MYPROPERTY set-suiteproperty -name NEWPROPERTY -value $Value
format-suiteproperties -Value [format string]	この cmdlet は、形式化された式の値を展開します。Name および Value パラメーターは位置指定パラメーターです。 次のサンプル コードは、NEWPROPERTY と呼ばれるスイート/アドバンスト UI プロパティを含む文字列を解決します。プロパティ名と、その周りの角かっこは、実行時にプロパティの値で置換されます。 $myFormat = format-suiteproperties -Value "これは、[NEWPROPERTY] が埋め込まれたテキスト文字列です。"
resolve-suitestring -StringId [文字列 ID]	この cmdlet は、スイート/アドバンスト UI 文字列識別子を、現在選択されている UI 言語でそれに対応する文字列値で解決します。指定された文字列 ID が存在しない場合、文字列は空の状態で戻されます。StringId パラメーターは、位置指定パラメーターです。 次のサンプル コードは、**IDS_MY_MESSAGE** と呼ばれる文字列識別子を解決します: $StringValue = resolve-suitestring -StringId IDS_MY_MESSAGE

テーブル 2 ・ 実行中のスイート/アドバンスト UI インストールとインタラクションする cmdlet (続き.)

Cmdlet	説明
trace-suiteinfo -LogMessage [長い文字列]	この cmdlet により、スイート/アドバンスト UI デバッグ ログに必要な情報を書き込みし、それらをデバッグやその他の情報提供目的に使用できるようにします。LogMessage パラメーターは、位置指定パラメーターです。 次のサンプル コードは、**IDS_MY_MESSAGE** と呼ばれる文字列識別子を解決してから、その値をスイート/アドバンスト UI デバッグ ログ記録に書き込みます。 $StringValue = resolve-suitestring -StringId IDS_MY_MESSAGE trace-suiteinfo -LogMessage $StringValue

スクリプトが成功したことを示すには、必ず 0 を戻します。たとえば、次の行でスクリプトを終了します:

exit(0)

プロジェクトの [イベント] ビューで PowerShell アクションを追加した後、[アクション] エクスプローラーまたは [イベント] エクスプローラーでそれを選択し、InstallShield の右に表示される [PowerShell スクリプト] タブを使ってスクリプトを編集できます。

スイート/アドバンスト UI インストールに含まれるプロパティを設定するアクションでの作業について

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、スイート/アドバンスト UI プロジェクトに、実行時にスイート/アドバンスト UI プロパティを設定するアクションを追加できます。

タスク

スイート/アドバンスト UI インストールで プロパティを設定するアクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーを右クリックしてから、**[新しいプロパティ セット]** をクリックします。**[アクション]** エクスプローラーにプロパティ アクションが追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このアクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。

4. 必要に応じて、アクションの設定を構成します。

プロジェクトにアクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

スイート/アドバンスト UI インストールに含まれる InstallScript コードを実行するアクションでの作業について



プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、スイート/アドバンスト UI プロジェクトに、実行時に InstallScript コードを呼び出すアクションを追加できます。スイート/アドバンスト UI ベースの InstallScript アクションは、基本の MSI プロジェクトの InstallScript カスタム アクションとほぼ同じ要領で機能します。

スイート/アドバンスト UI プロジェクトに IInstallScript アクションを追加するには、まず始めにインストールから実行する InstallScript コードを作成します。この作業には、プロジェクトの [InstallScript] ビューを使います。コードを作成したら、プロジェクトの [イベント] ビューを使って InstallScript アクションを追加します。



タスク

スイート/アドバンスト UI インストールで InstallScript コードを実行するアクションを追加するには、以下の手順に従います:

1. InstallScript コードをプロジェクトに追加するには、以下の手順に従います:
   a. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
   b. **InstallScript** エクスプローラーで、**[ファイル]** を右クリックして、**[新しいスクリプト ファイル]** を選択します。**[ファイル]** ノードの下に Setup.rul という名前の新しいファイルが追加されます。
   c. 新しい .rul ファイルを選択してから、実行時に起動するコードを右側のペインで作成します。

      スイート/アドバンスト UI アクションで InstallScript 関数を呼び出す場合、次の例のように、**export** キーワードを使って関数をプロトタイプ化します。

      ```
      export prototype MyFunction(OBJECT);
      ```

2. InstallScript コードを呼び出す InstallScript アクション を追加します:
   a. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
   b. **[アクション]** エクスプローラーを右クリックしてから、**[新しい InstallScript]** をクリックします。**[アクション]** エクスプローラーに InstallScript アクションが追加されます。
   c. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このアクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。

d. 必要に応じて、アクションの設定を構成します。

プロジェクトにアクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

スイート/アドバンスト UI アクションの InstallScript コードを書くためのヒント

一部の InstallScript 関数は InstallScript アクションのエントリ ポイントに渡される ISuiteExtension オブジェクトをラップします。これらの InstallScript 関数を使って、InstallScript アクションが実行中のスイート/アドバンスト インストールとインタラクトできるようにします。

- **SuiteFormatString**
- **SuiteGetProperty**
- **SuiteLogInfo**
- **SuiteResolveString**
- **SuiteSetProperty**

これらの関数についての詳細は、「スイート/アドバンスト UI およびアドバンスト UI の対話関数」を参照してください。

スイート/アドバンスト UI インストール内のアクションから呼び出される InstallScript 関数を使用する際、次の詳細にご注意ください:

- InstallScript UI 関連の関数を、スイート/アドバンスト UI プロジェクトで使用することはできません。これを使用すると、スクリプトのコンパイル エラーが発生するか、コンパイルが完了しても、実行時にエラーが発生します。。InstallScript アクションで表示される UI はすべて、スイート/アドバンスト UI インストールの UI が表示されるときまで遅延させることが推奨されます。
- InstallScript アクションは、スイート/アドバンスト UI プロパティを読み書きして、それらをスイート UI とのコミュニケーション手段として使用します。(スイート/アドバンスト UI アクションは、基本の MSI カスタム アクションと同様に、UI にアクセスすることができません。)ビルトイン プロパティの一覧は、「アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」を参照してください。
- InstallScript 実行時パス変数 ((PROGRAMFILES、WINDIR など) は使用できません。これらの変数は通常、「*バインドされた変数*」として知られるスクリプト エンジンの概念を使って実装されます。バインドされた変数は、変数そのものがテキストサブ名値のマッピングに直接関連付けられています。これらの変数に対応するテキストサブは引き続き使用できます。たとえば、<PROGRAMFILES> は、InstallScript 関数 **TextSubGetValue** または **TextSubSubstitute** を通して解決することができます。パス型スイート/アドバンスト UI プロパティも、スイート/アドバンスト UI エンジンを通して定義され、**SuiteGetProperty** で読み込むことができます。
- LAAW_PARAMETERS 構造体変数の szStatusText メンバーを、スイート/アドバンスト UI インストールで呼び出される InstallScript アクションに使用することはできません。

InstallScript デバッガーを使って、スイート/アドバンスト UI インストールの InstallScript アクションをデバッグできます。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング」を参照してください。

スイート/アドバンスト UI インストールでマネージ コード アクション を使用する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield を使って、実行時に Visual Basic .NET または C# などのマネージ コードで書かれたマネージ アセンブリでパブリック メソッドを呼び出すスイート/アドバンスト UI プロジェクトに、マネージ コード アクション タイプを追加することができます。

マネージ コード アクション用のマネージ アセンブリを作成する

スイート/アドバンスト UI プロジェクトにマネージ コード アクションを追加する最初の手順は、マネージ アセンブリの作成です。マネージ アセンブリは、任意の最近のバージョンの .NET 言語 (Visual Basic .NET または C# など) を使って作成できます。

スイート/アドバンスト UI エンジンでは、各マネージ コード アクションにつき 1 つのエントリ ポイントが必要です。この場合のエントリ ポイントとは、スタティック メソッドか、パラメーターなしで構成可能なクラス上にあるインスタンス メソッドのどちらかです。

エントリ ポイントを指定するには、**[イベント]** ビューにあるアクションの **"クラス"** 設定および **"メソッド"** 設定を使います。

スイート/アドバンスト UI インストールは、インストールに含まれるアクションの実行がスケジュールされているときにエントリ ポイントを呼び出します。エントリ ポイント メソッドは、アクションが正しく終了したことを示す 数値 0 を戻します。アクションが失敗した場合、ゼロ以外の値が戻されます。戻されるエラーの値は、Windows Installer カスタム アクションで定義されている戻り値と同じです。

アクションのエントリ ポイントに渡されるインターフェイスは、オブジェクトとして渡される ISuiteExtension インターフェイスを実装します。戻り値の型は 32 ビット整数型でなくてはなりません。

スイート/アドバンスト UI 拡張インターフェイスを使用する場合、たとえば Visual Studio で C# コードを作成するときに次のようなコードを含めることができます:

```
[Guid("BAFAEAED-08C6-4679-B94E-487A4D89DE63")]
[TypeLibType(4288)]
public interface ISuiteExtension
  {
    [DispId(1)]
    string get_Attribute(string bstrName);
    [DispId(2)]
    void LogInfo(string bstr);
    [DispId(3)]
    string get_Property(string bstrName);
    [DispId(3)]
    void set_Property(string bstrName, string bstrValue);
    [DispId(4)]
```

```
        string FormatProperty(string bstrValue);
        [DispId(5)]
        string ResolveString(string bstrStringId);
    }
```

マネージ コードで次のメソッドを使用できます:

- **LogInfo** — このメソッドにより、スイート/アドバンスト UI デバッグ ログに必要な情報を書き込み、それらをデバッグやその他の情報提供目的に使用できるようにします。bstr パラメーターには、ログに書きこまれる文字列が含まれています。
- **get_Property** — このメソッドによって、現在実行中のスイート/アドバンスト UI インストールで定義されているプロパティの値が取得されます。定義されていないプロパティは、空の値を返します。bstrName パラメーターによって、値が取得されるプロパティの名前が指定されます。プロパティの値は、bstrValue パラメーターで返されます。
- **set_Property** — このメソッドを使って、現在実行中のスイート/アドバンスト UI インストールで、新しいプロパティの値を設定したり、既存のプロパティの値を変更したりできます。空の値を渡すと、その結果として、プロパティが削除されます。bstrName パラメーターによって、設定するプロパティの名前が指定されます。bstrName パラメーターによって、設定するプロパティの名前が指定されます。
- **FormatProperty**—この方法を使うと、bstrValue パラメーターで提供された文字列内に組み込まれた形式化された式を解決することができます。フォーマットされた値は、bstrReturnValue パラメーターで戻されます。形式化された式で使用可能な構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。
- **ResolveString** — このメソッドはスイート/アドバンスト UI の文字列 ID を、現在選択されている UI 言語で実行中されているスイート/アドバンスト UI インストールの対応する文字列値に解決します。bstrStringId パラメーターによって、解決する文字列 ID が指定され、解決された文字列は bstrReturnValue で返されます。指定された文字列 ID が存在しない場合、文字列は空の状態で戻されます。

次のサンプル C# コードは、TestCLRAction、TestCLRClass の "クラス" 設定値、TestCLRMethod の "メソッド" 設定値に対応します:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using Microsoft.WindowsAzure.Common;
using Microsoft.WindowsAzure.Management;
using Microsoft.WindowsAzure.Management.WebSites;
using Microsoft.WindowsAzure.Management.WebSites.Models;
using Microsoft.WindowsAzure;
using System.Windows.Forms;

namespace TestCLRAction
{
    [Guid("BAFAEAED-08C6-4679-B94E-487A4D89DE63")]
    [TypeLibType(4288)]
    public interface ISuiteExtension
    {
        [DispId(1)]
        string get_Attribute(string bstrName);
```

```
    [DispId(2)]
    void LogInfo(string bstr);
    [DispId(3)]
    string get_Property(string bstrName);
    [DispId(3)]
    void set_Property(string bstrName, string bstrValue);
    [DispId(4)]
    string FormatProperty(string bstrValue);
    [DispId(5)]
    string ResolveString(string bstrStringId);
}

public class TestCLRClass
{
    const UInt32 ERROR_INSTALL_FAILURE = 1603;
    const UInt32 ERROR_SUCCESS = 0;

    ISuiteExtension GetExtensionFromDispatch(object pDispatch)
    {
        if (MarshalIsComObject(pDispatch) == false)
            throw new System.ContextMarshalException("無効な
            ディスパッチ オブジェクトが CLR メソッドに渡されました");

        return (ISuiteExtension)pDispatch;
    }
    public UInt32 TestCLRMethod(object pDispatch)
    {
        try
        {
            ISuiteExtension suiteExtension = GetExtensionFromDispatch(pDispatch);
            suiteExtension.set_Property("TESTPROP", "hello, world");
        }
        catch (System.ContextMarshalException)
        {
            return ERROR_INSTALL_FAILURE;
        }

        return ERROR_SUCCESS;
    }
}
```



メモ: *スイート/アドバンスト UI インストールでは任意のアクションを実行することが可能ですが、アクションは構成された特権を使って実行されます。一部のアクションでは、管理者権限が必要な場合があります (IIS 7.x 構成データの読み取りなど)。その場合、アクションが必要なデータに正しくアクセスできるように、アクションが管理者権限を要求するように構成する必要があります。つまり、[イベント] ビューにあるアクションの "管理者権限が必要" 設定で [はい] を選択する必要があります。別の方法として、スイート/アドバンスト UI インストールの Setup.exe ファイルを管理者権限を使って実行するか、管理者マニフェストを含みます。ただし、特権の昇格はできるだけ短時間に限ります。*

マネージ コード アクションを追加する



タスク

スイート/アドバンスト UI インストールの実行中にマネージ アセンブリのパブリック メソッドを呼び出すアクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーを右クリックしてから、**[新しいマネージ コード]** をクリックします。**[アクション]** エクスプローラーに マネージ コード アクションが追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このアクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。
4. 必要に応じて、アクションの設定を構成します。

プロジェクトにアクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

マネージ コード アクションの実行時要件

スイート/アドバンスト UI プロパティ IS_CLR_VERSION を使って、マネージ コードを実行するためにアクションがロードする、セミコロン区切りの .NET Framework バージョン一覧を識別することができます。IS_CLR_VERSION プロパティについては、「 アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」を参照してください。

マネージ コード アクションに依存関係がある場合、[サポート ファイル] ビューを使って、それらをプロジェクトに追加します。実行時、スイート/アドバンスト UI エンジンは、.NET Framework に [SETUPSUPPORTDIR] 内に依存関係があることを指定します。

スイート/アドバンスト UI インストールで .NET Core アクションを使用する



プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield を使って、実行時に Visual Basic .NET または C# などのマネージ コードで書かれたマネージ アセンブリでパブリック メソッドを呼び出すスイート/アドバンスト UI プロジェクトに、.NET Core アクション タイプを追加することができます。

.NET Core アクションのマネージ アセンブリを作成する

スイート/アドバンスト UI プロジェクトに .NET Core アクションを追加する場合、最初にマネージ アセンブリを作成します。マネージ アセンブリは、任意の最近のバージョンの .NET 言語 (Visual Basic .NET または C# など) を使って作成できます。

スイート/アドバンスト UI エンジンは、各 .NET Core アクションにエントリ ポイントが必要です。この場合のエントリ ポイントとは、スタティック メソッドか、パラメーターなしで構成可能なクラス上にあるインスタンス メソッドのどちらかです。

エントリ ポイントを指定するには、**[イベント]** ビューにあるアクションの **"クラス"** 設定および **"メソッド"** 設定を使います。

スイート/アドバンスト UI インストールは、インストールに含まれるアクションの実行がスケジュールされているときにエントリ ポイントを呼び出します。エントリ ポイント メソッドは、アクションが正しく終了したことを示す 数値 0 を戻します。アクションが失敗した場合、ゼロ以外の値が戻されます。戻されるエラーの値は、Windows Installer カスタム アクションで定義されている戻り値と同じです。

アクションのエントリ ポイントに渡されるインターフェイスは、オブジェクトとして渡される ISuiteExtension インターフェイスを実装します。戻り値の型は 32 ビット整数型でなくてはなりません。

スイート/アドバンスト UI 拡張インターフェイスを使用する場合、たとえば Visual Studio で C# コードを作成するときに次のようなコードを含めることができます:

```
[Guid("BAFAEAED-08C6-4679-B94E-487A4D89DE63")]
[TypeLibType(4288)]
public interface ISuiteExtension
  {
    [DispId(1)]
    string get_Attribute(string bstrName);
    [DispId(2)]
    void LogInfo(string bstr);
    [DispId(3)]
    string get_Property(string bstrName);
    [DispId(3)]
    void set_Property(string bstrName, string bstrValue);
    [DispId(4)]
    string FormatProperty(string bstrValue);
    [DispId(5)]
    string ResolveString(string bstrStringId);
  }
```

マネージ コードで次のメソッドを使用できます:

- **LogInfo** ― このメソッドにより、スイート/アドバンスト UI デバッグ ログに必要な情報を書き込みし、それらをデバッグやその他の情報提供目的に使用できるようにします。bstr パラメーターには、ログに書きこまれる文字列が含まれています。
- **get_Property** ― このメソッドによって、現在実行中のスイート/アドバンスト UI インストールで定義されているプロパティの値が取得されます。定義されていないプロパティは、空の値を返します。bstrName パラメーターによって、値が取得されるプロパティの名前が指定されます。プロパティの値は、bstrValue パラメーターで返されます。
- **set_Property** ― このメソッドを使って、現在実行中のスイート/アドバンスト UI インストールで、新しいプロパティの値を設定したり、既存のプロパティの値を変更したりできます。空の値を渡すと、その結果として、プロパティが削除されます。bstrName パラメーターによって、設定するプロパティの名前が指定されます。bstrName パラメーターによって、設定するプロパティの名前が指定されます。

- **FormatProperty**—この方法を使うと、bstrValue パラメーターで提供された文字列内に組み込まれた形式化された式を解決することができます。フォーマットされた値は、bstrReturnValue パラメーターで戻されます。形式化された式で使用可能な構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。
- **ResolveString** — このメソッドはスイート/アドバンスト UI の文字列 ID を、現在選択されている UI 言語で実行中されているスイート/アドバンスト UI インストールの対応する文字列値に解決します。bstrStringId パラメーターによって、解決する文字列 ID が指定され、解決された文字列は bstrReturnValue で返されます。指定された文字列 ID が存在しない場合、文字列は空の状態で戻されます。

次のサンプル C# コードは、TestCLRAction、TestCLRClass の "クラス" 設定値、TestCLRMethod の "メソッド" 設定値に対応します:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using Microsoft.WindowsAzure.Common;
using Microsoft.WindowsAzure.Management;
using Microsoft.WindowsAzure.Management.WebSites;
using Microsoft.WindowsAzure.Management.WebSites.Models;
using Microsoft.WindowsAzure;
using System.Windows.Forms;

namespace TestCLRAction
{
    [Guid("BAFAEAED-08C6-4679-B94E-487A4D89DE63")]
    [TypeLibType(4288)]
    public interface ISuiteExtension
    {
        [DispId(1)]
        string get_Attribute(string bstrName);
        [DispId(2)]
        void LogInfo(string bstr);
        [DispId(3)]
        string get_Property(string bstrName);
        [DispId(3)]
        void set_Property(string bstrName, string bstrValue);
        [DispId(4)]
        string FormatProperty(string bstrValue);
        [DispId(5)]
        string ResolveString(string bstrStringId);
    }

    public class TestCLRClass
    {
        const UInt32 ERROR_INSTALL_FAILURE = 1603;
        const UInt32 ERROR_SUCCESS = 0;

        ISuiteExtension GetExtensionFromDispatch(object pDispatch)
        {
            if (MarshalIsComObject(pDispatch) == false)
```

```
            throw new System.ContextMarshalException("無効な
            ディスパッチ オブジェクトが CLR メソッドに渡されました");

        return (ISuiteExtension)pDispatch;
    }
    public UInt32 TestCLRMethod(object pDispatch)
    {
        try
        {
            ISuiteExtension suiteExtension = GetExtensionFromDispatch(pDispatch);
            suiteExtension.set_Property("TESTPROP", "hello, world");
        }
        catch (System.ContextMarshalException)
        {
            return ERROR_INSTALL_FAILURE;
        }

        return ERROR_SUCCESS;
    }
}
```

メモ: *スイート/アドバンスト UI インストールでは任意のアクションを実行することが可能ですが、アクションは構成された特権を使って実行されます。一部のアクションでは、管理者権限が必要な場合があります (IIS 7.x 構成データの読み取りなど)。その場合、アクションが必要なデータに正しくアクセスできるように、アクションが管理者権限を要求するように構成する必要があります。つまり、[イベント] ビューにあるアクションの "管理者権限が必要" 設定で [はい] を選択する必要があります。別の方法として、スイート/アドバンスト UI インストールの Setup.exe ファイルを管理者権限を使って実行するか、管理者マニフェストを含みます。ただし、特権の昇格はできるだけ短時間に限ります。*

.NET Core アクションを追加する

タスク

スイート/アドバンスト UI インストールの実行中にマネージ アセンブリのパブリック メソッドを呼び出すアクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーを右クリックしてから、**[新しい .NET Core アクション]** をクリックします。
   [アクション] エクスプローラーに .NET Core アクションが追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このアクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。
4. 必要に応じて、アクションの設定を構成します。

プロジェクトにアクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

.NET Core アクションの実行時要件

.NET Core タイプのカスタム アクションを使用するには、ターゲット システム上に Microsoft .NET Core 8.0 が必要です。

.NET Core アクションに依存ファイルがある場合、**[サポート ファイル]** ビューを使ってこれらをプロジェクトに追加します。実行時、スイート/アドバンスト UI エンジンが .NET Core に対して、関連ファイル が [SETUPSUPPORTDIR] にあることを指定します。

アクションをスイート/アドバンスト UI プロジェクトに追加する



プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*



タスク ***アクションをスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーを右クリックしてから、追加するアクションの種類をクリックします。

   PowerShell アクションを作成する場合、**[開く]** ダイアログ ボックスが開いて、使用する PowerShell スクリプト ファイル (.ps1) を選択することができます。
3. オプションで、新しいアクションを選択し、F2 キーを押し、新しい名前を入力して名前を変更できます。
4. アクションの設定を構成します。

実行時にアクションが起動またはロードできない場合、インストールは中止します。

スイート/アドバンスト UI アクションの設定を構成する



プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*



タスク ***スイート/アドバンスト UI プロジェクトでアクションを構成するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーで、構成するアクションの種類をクリックします。
3. 右側に表示されているグリッド内の設定を構成します。

実行時にアクションが起動またはロードできない場合、インストールは中止します。

スイート/アドバンスト UI アクションをスケジュールする



プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

プロジェクトにアクションを追加すると、それを実行時に起動するタイミングをスケジュールできます。スイート/アドバンスト UI エンジンがアクションを起動する前に評価する 1 つ以上の条件を定義することもできます。

InstallShield では、アクションを種類の異なる場所でスケジュールすることができます:

- スイート/アドバンスト UI が管理する 1 つ以上のビルトイン イベントの実行中にアクションをスケジュールできます。たとえば、スイート/アドバンスト UI インストールがロードされた直後にスイート/アドバンスト UI エンジンは OnBegin と呼ばれるイベントの実行をトリガします。このイベントは、OnBegin イベントが発生するときに実行する操作を指定します。
- スイート/アドバンスト UI エンジンがパッケージをインストール、削除、変更、または修復する前後にアクションの実行をスケジュールできます。
- アクションをウィザード インターフェイスの要素 (たとえば、ウィザード ページまたはウィザード ページのコントロール) と関連付けることができます。

詳細については、「スイート/アドバンスト UI インストールにおけるイベントの種類」を参照してください。

イベント中にアクションをスケジュールする

[イベント] ビューの [イベント] エクスプローラーの下にリストされているイベントとアクションは、実行時に起動される順番に時系列で編成されています。アクションをイベントに追加する際、アクションを適切なイベントに追加してアクションが起動されるタイミングを指定します。

タスク ***イベント中にアクションをスケジュールするには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[イベント]** エクスプローラーで、そのアクションの直前に位置するイベントまたはアクションを右クリックして、**[既存アクションを追加]** をポイントしてから、スケジュールするアクションの名前をクリックします。**[イベント]** エクスプローラーにアクションが追加されます。
3. 右側に表示されているグリッド内の設定を構成します。設定の 1 つを使って、アクションが起動されるかどうかを制御する、条件ステートメントをビルドすることができます。この条件の定義方法に関する詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」をご覧ください。

   アクションが特定のモードでのみ実行することを示すには、"プロパティの比較" 条件でスイート/アドバンスト UI プロパティ ISInstallMode を使用します。このプロパティ、およびその他のビルトイン プロパティの詳細については、「アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」を参照してください。

ヒント: *[イベント] ビューでは、ドラッグアンドドロップ編集がサポートされています:*

- *新しいアクションをシーケンスするには、[アクション] エクスプローラーからそれを、[イベント] エクスプローラーの下にある適切なイベントにドラッグします。次いで、それをシーケンス内で直前にくるアイテムの上にドロップします。*
- *アクションを、イベント内の別の位置へ (または、あるイベントから別のイベントへ) 移動するには、それを元の位置からドラッグして、シーケンス内で直前にくるアイテムの上にドロップします。*
- *アクションをあるイベントから別のイベントへコピーするには、CTRL を押しながら、そのアクションをもう 1 つのイベントまでドラッグし、直前にくるアクションまたはイベントの上にドロップします。*

スイート/アドバンスト UI エンジンがパッケージをインストール、削除、変更、または修復する前後にアクション行をスケジュールする

タスク ***パッケージがインストール、削除、変更、または修復する前後にアクションをスケジュールするには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、アクションと関連付けるパッケージをクリックします。
3. **[共通]** タブで、**[イベント]** 領域が展開されていない場合、これを展開します。
4. 以下のいずれかを実行します。
   - パッケージがインストール、削除、変更、または修復する前にアクションをスケジュールする場合: **"構成済みパッケージ"** 設定で、**[新しいアクション]** ボタンをポイントしてから、スケジュールするアクションの名前をクリックします。InstallShield によって、**"アクション"** 設定が追加され、その値が選択されたアクションの名前に設定されます。
   - パッケージがインストール、削除、変更、または修復した後にアクションをスケジュールする場合: **"構成済みパッケージ"** 設定で、**[新しいアクション]** ボタンをポイントしてから、スケジュールするアク

ションの名前をクリックします。InstallShield によって、**"アクション"** 設定が追加され、その値が選択されたアクションの名前に設定されます。

5. オプションで、アクションの起動時により詳細な制御を行うための条件ステートメントをビルドしたい場合は、**[条件の追加]** ボタンをクリックします。この条件の定義方法に関する詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」をご覧ください。

   インストールに含まれる特定のパッケージが、特定の状態で実行している場合のみアクションを実行することを示すには、[パッケージの操作] 条件を作成します。

   アクションが特定のモードでのみ実行することを示すには、"プロパティの比較" 条件でスイート/アドバンスト UI プロパティ ISInstallMode を使用します。このプロパティ、およびその他のビルトイン プロパティの詳細については、「アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」を参照してください。

ウィザード インターフェイス内の要素にアクションを追加する

InstallShield では、ウィザード ページまたは 2 番目のウィンドウが開いているまたは閉じているときに起動されるアクションを定義することができます。また InstallShield では、エンド ユーザーがボタンをクリックしたときなど、様々なウィザード インターフェイス コントロールを使用するとトリガされるアクションを定義できます。ウィザード インターフェイスを使ってアクションをスケジュールする方法については、「ウィザード インターフェイス内の要素のアクションを構成する」を参照してください。

スイート/アドバンスト UI インストールにおけるイベントの種類



プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

スイート/アドバンスト UI インストールは、特定の順番で一連のイベントを管理するスイート/アドバンスト UI インストールによって駆動します。これらのイベントは、プロジェクトに追加されたアクションを実行します。

次のテーブルは、スイート/アドバンスト UI インストールの [イベント] ビューにリストされている各イベントについて説明します:

テーブル 3 · [イベント] ビューにリストされているイベント

イベント名	説明
OnBegin	OnBegin イベントはスイート/アドバンスト UI インストールの最初のイベントです。このイベントにスケジュールされているアクションは、そのアクションに定義されている条件ステートメントが False 評価されない限り、すべてのモード (インストール、メンテナンス、修復など) で実行されます。 アクションが特定のモードでのみ実行することを示すには、"プロパティの比較" 条件でスイート/アドバンスト UI プロパティ ISInstallMode を使用します。このプロパティ、およびその他のビルトイン プロパティの詳細については、「アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」を参照してください。 OnBegin 中にスケジュールされたアクションは、スイート/アドバンスト UI インストールがウィザード インターフェイスを表示する前に実行します。このイベント中に長時間実行するアクションは、エンド ユーザーがインストールを起動した時点からウィザード インターフェイスが初めて表示されるまでに長い遅延をもたらす可能性があります。その結果、インストールが正しく起動されていないと、エンド ユーザーが勘違いする場合があります。これを避けるために、OnBegin 中に長時間実行するアクションをスケジュールしないことが推奨されます。 また、起動時に独自のユーザー インターフェイスを表示するアクションも避けて下さい。スイート/アドバンスト UI インストールのウィザード インターフェイスは、既に開いている別のウィンドウの背面に表示され、エンド ユーザーを混乱させる可能性があります。
OnResuming	OnResuming イベントは、ターゲット マシンが再起動された後、スイート/アドバンスト UI インストールが再開するときに、そのアクションを起動します。
OnStaging	OnStaging イベントは、スイート/アドバンスト UI インストールがステージングされる直前に、そのアクションを起動します。つまり、(適切な場合) スイート/アドバンスト UI インストールがパッケージをダウンロードして、(適切な場合) Setup.exe ファイルからパッケージを抽出し、BrowseStageFolder ウィザード ページでエンド ユーザー が指定するディレクトリ、またはコマンドラインを使って **ISRootStagePath** プロパティで設定されたディレクトリにパッケージをコピーする前にアクションを起動します。
OnStaged	OnStaged イベントは、(適切な場合) スイート/アドバンスト UI インストールがパッケージをダウンロードし、Seup.exe ファイルから抽出して、ステージング ディレクトリにコピーした後に、そのアクションを起動します。
OnPackagesConfiguring	PackagesConfiguring イベントは、スイート/アドバンスト UI インストールがインストール、変更、修正、または削除される直前に、そのアクションを起動します。

テーブル 3 ・ [イベント] ビューにリストされているイベント (続き.)

イベント名	説明
OnPackagesConfigured	OnPackagesConfigured イベントは、スイート/アドバンスト UI インストールがインストール、変更、修正、または削除された時に、そのアクションを起動します。
OnRebooting	OnRebooting イベントは、スイート/アドバンスト UI インストールがマシンの再起動をトリガする直前に、そのアクションを起動します。 OnRebooting 中にスケジュールされたアクションは、短時間で実行しなくてはなりません。そうでない場合、前に実行されているアクションによって再起動が開始され、再起動イベントが完了する前に Windows がスイート/アドバンスト UI プロセスをシャットダウンした場合、アクションが完了せずに失敗することがあります。
OnEnd	OnEnd イベントは、スイート/アドバンスト UI インストールが正常に終了する直前に、そのアクションを起動します。エンド ユーザーがスイート/アドバンスト UI インストールをキャンセルした場合、またはインストールでエラーが発生した場合、OnEnd イベントは実行しません。

スイート/アドバンスト UI エンジンは、実行済み/未実行のイベントの状況を管理しません。そのため、再起動によってスイート/アドバンスト UI インストールが再開した後に、アクションが再実行されることを避けるため、アクションの [プロパティの比較] 条件を定義しなくてはならない場合があります。これらの条件で ISOnRebooted プロパティを使用できます。スイート/アドバンスト UI インストールが再起動の後に再開する場合、このプロパティは True に設定します。

アクションに "機能の操作" 条件または "パッケージの操作" 条件を定義する場合、アクションをスケジュールできる最初のイベントは OnStaging です。

アドバンスト UI またはスイート/アドバンスト UI インストールの終了条件を定義する



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



***エディション・** アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、アドバンスト UI またはスイート/アドバンスト UI インストールがインストールの終了の前に、様々な条件に応じて表示する終了エラー メッセージを指定できます。たとえば、アドバンスト UI またはスイート/アドバンスト UI インストールに Windows Vista 以降が必要な場合、エンド ユーザーに対して要件を通知する終了メッセージを設定できます。また、終了メッセージに条件を定義して、ターゲット システムに Windows の以前のバージョンが搭載されているかどうかを評価することもできます。エンドユーザーがアドバンスト UI またはスイート/アドバンスト UI インストールを起動したとき、定義された条件が True となった場合、エラー メッセージが表示されます。エンドユーザーがエラー メッセージ ボックスを閉じると、インストールは終了します。



タスク

アドバンスト UI またはスイート/アドバンスト UI インストールの終了条件およびメッセージを定義するには、以下の手順に従います:

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"終了条件"** 設定で、**[新しい条件]** ボタンをクリックします。**"終了条件"** 設定の下に終了メッセージのサブ設定が追加されます。
3. **"終了メッセージ"** 設定に、構成中の条件が実行時に True 評価された場合にアドバンスト UI またはスイート/アドバンスト UI インストールで表示するエラー メッセージを入力します。
4. **"終了メッセージ"** 設定を拡張してから、そのサブ設定を使って実行時にアドバンスト UI またはスイート/アドバンスト UI インストールが評価を行う条件ステートメントを定義します。
   - 最初のサブ設定では、その設定内のリストを使って、構成している条件、または条件グループに使用するオプションを選択します。選択可能なオプションは以下のとおりです:
     - **Any**—Any 条件グループは、論理演算子 OR と同じ要領で動作します。Any グループに含まれる任意の条件が True 評価された場合、条件グループ全体が True 評価されます。Any グループに含まれる条件のいずれも True 評価されなかった場合、条件グループ全体が False 評価されます。
     - **All**—All 条件グループは、論理演算子 AND と同じ要領で動作します。All グループが True 評価されるためには、All グループに含まれる条件のすべてが True 評価される必要があります。
     - **None**—None 条件グループは、論理演算子 NOR と同じ要領で動作します。None グループに含まれる条件のいずれも True 評価されなかった場合、条件グループ全体が True 評価されます。None 条件グループに含まれる任意の条件が True 評価された場合、条件グループ全体が False 評価されます。None 条件グループが 1 つの条件ステートメントで構成される場合、これは論理演算子 NOT と同じ要領で動作します。
   - サブ条件グループを追加するには、この設定で **[新しい条件]** ボタンをクリックしてから、必要に応じて **Any**、**All**、または **None** を選択します。条件の定義に使用できる新しい行が追加されます。
   - 条件を追加するには、この設定で **[新しい条件]** ボタンをクリックしてから、使用可能な条件の種類から 1 つを選択します。条件の定義に使用できる新しい行が追加されます。

条件の構成についての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。

各条件設定についての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI インストールのインストール モードまたはメンテナンス モードをトリガする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI インストールは、次のいずれかのモードで実行されます:

- インストール モード。初回インストールとして動作するインストール。
- メンテナンス モード。インストールで、エンド ユーザーが製品の削除、修復、変更を選択できるウィザード ページが表示されます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、インストール モードおよびメンテナンス モードの条件が自動的に作成されます。モード条件は、 内では編集できません。また、モード条件はプロジェクト ファイル (.issuite) でのみ表示されます。これらのモードは、アドバンスト UI またはスイート/アドバンスト UI インストールを初回インストール モードで実行するか、メンテナンス モードで実行するかを判別します。

インストール モードの条件

インストール モードの条件は、次の要因に基づきます:

- **検出条件**—インストール モード条件の一部は、スイート/アドバンスト UI プロジェクトに含まれるすべてのプライマリ パッケージ (アドバンスト UI プロジェクトの場合、1 つのプライマリ パッケージ) の検出条件から構成されています。検出条件が True に評価されたプライマリ パッケージが存在しない場合 (既にインストールされているプライマリ パッケージが存在しない場合)、インストール モード条件の検出条件の部分は True と評価されます。検出条件が True に評価されたプライマリ パッケージが 1 つ以上存在する場合 (既にインストールされているプライマリ パッケージが最低 1 つ存在する場合)、インストール モード条件の検出条件の部分は False と評価されます。
- **インストール済みスイート条件**—インストール モード条件のもう片方の部分は、インストール済みスイート条件で構成されています。この条件により、同じバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールがターゲット システムに存在しないとき、インストールは初回インストール モードでトリガされることがあります。インストール モード条件のンストール済みスイート条件は、適切な場合、アドバンスト UI またはスイート/アドバンスト UI インストールのスイート GUID およびバージョンを、ターゲット システムに既にインストールされているアドバンスト UI またはスイート/アドバンスト UI インストールのスイート GUID およびバージョンと比較します。これらの値が異なる場合、インストール モード条件のンストール済みスイート条件によって、アドバンスト UI またはスイート/アドバンスト UI インストールが存在していないということが示唆され、これにより、初回インストールが実行されることがあります。[インストール済みスイート] タイプの条件チェックに関する詳細は、「特定のバージョンのアドバンスト UI またはス

イート/アドバンスト UI インストールが既にインストールされているかどうかを判別する」を参照してください。

検出条件が False に評価されたプライマリ パッケージが 1 つ以上存在する場合、または、インストール済みスイート条件で、アドバンスト UI またはスイート/アドバンスト UI インストールがまだインストールされていないと判断された場合、インストール モード条件は True として評価され、インストールが初回インストールとして実行されます。

メンテナンス モードの条件

メンテナンス モード条件は、スイート/アドバンスト UI プロジェクトに含まれるすべてのプライマリ パッケージ（アドバンスト UI プロジェクトの場合、1 つのプライマリ パッケージ）の検出条件に基づきます。検出条件が True に評価されたプライマリ パッケージが 1 つ以上存在する場合（既にインストールされているプライマリ パッケージが最低 1 つ存在する場合）、モード条件は True として評価され、インストールはメンテナンス モードで実行されます。

スイート/アドバンスト UI プロジェクトと Web 配置パッケージに関する特別考慮

Web 配置パッケージを検出することはできません。これらのパッケージは一般的に、初回インストールと同様、または既存の古いバージョンあるいは同一バージョンを上書きする場合と同様に動作するように設計されています。その結果、Web 配置パッケージではアンインストールを行うことができません。つまり、スイート/アドバンスト UI インストールはこの種類のパッケージのメンテナンスを行うことができません。1 つ以上の Web 配置パッケージ、または 1 つ以上の従来型パッケージ（メンテナンス オプションが含まれている）をプライマリ パッケージとして含むスイート/アドバンスト UI インストールの作成を避けることが推奨されます。

アドバンスト UI またはスイート/アドバンスト UI インストールの [プログラムの追加または削除] エントリ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、アドバンスト UI またはスイート/アドバンスト UI インストールで、[プログラムの追加または削除] エントリを作成するかどうかを指定できます。このエントリを使って、エンド ユーザーは必要に応じてアドバンスト UI またはスイート/アドバンスト UI インストールの管理、変更、または削除を行うことができます。アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [一般情報] ビューには、適切な動作を指定できる "[プログラムの追加と削除] エントリの表示" 設定があります。

アドバンスト UI またはスイート/アドバンスト UI インストール全体に対して単一のエントリのみを表示する場合、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含めるパッケージからのエントリを非表示にしてください。

たとえば .msi パッケージの場合、[パッケージ] ビューにあるパッケージのコマンドライン設定を使って、Windows Installer プロパティ ARPSYSTEMCOMPONENT の値を 1 に設定できます。

[プログラムの追加または削除] エントリを表示" 設定は、アドバンスト UI またはスイート/アドバンスト UI 製品が [プログラムの追加または削除] にエントリされるかどうかが決定される唯一の要因ではないので注意してください。メンテナンス モード条件によっても、エントリされるかどうかが決定されます。メンテナンス モード条件は、インストール内のすべてのプライマリ パッケージの検出条件をすべて組み合わせた条件です。アドバンスト UI またはスイート/アドバンスト UI インストール内のすべてのプライマリ パッケージに対する検出条件が、初回インストール後、False と評価された場合、アドバンスト UI またはスイート/アドバンスト UI 製品の [プログラムの追加または削除] エントリは追加されません。アドバンスト UI またはスイート/アドバンスト UI インストール内のすべてのプライマリ パッケージに対する検出条件が、アドバンスト UI またはスイート/アドバンスト UI 製品が削除された後で True に評価された場合、既存の [プログラムの追加または削除] エントリはそのまま残されます。

コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI インストールプロジェクトの [パッケージ] ビューでパッケージを構成するとき、コマンドラインおよびサイレント コマンドライン設定を使って、パッケージを起動したときにアドバンスト UI またはスイート/アドバンスト UI インストールで使用するコマンドラインを指定できます。これらの設定は、[パッケージ] ビューのグリッド内にある [操作] 領域のサブ設定で使用できます。

アドバンスト UI またはスイート/アドバンスト UI インストールがユーザー インターフェイスと共に実行される場合、アドバンスト UI またはスイート/アドバンスト UI の Setup.exe ファイルは、"MSI コマンドライン" 設定、"MSP コマンドライン" 設定、"コマンドライン" 設定、または "EXE コマンドライン" 設定 (パッケージの種類によって異なる) に入力されたコマンドラインを使用してパッケージを起動します。インストールがサイレント (ユーザー インターフェイスなし) で実行されるとき、アドバンスト UI またはスイート/アドバンスト UI の Setup.exe ファイルは、"MSI サイレント コマンドライン" 設定、"MSP サイレント コマンドライン" 設定、"サイレント コマンドライン" 設定、または "EXE サイレント コマンドライン" 設定に入力されたコマンドラインを使用してパッケージを起動します。

これらの設定に値を入力するとき、構成するパッケージの種類によって以下の背景情報を参照してください。

.msi パッケージ

.msi パッケージのインストールおよび削除操作を構成している場合、使用できるコマンドライン パラメータの種類は、Windows Installer プロパティです。アドバンスト UI またはスイート/アドバンスト UI インストールは MsiInstallProduct を使って、.msi パッケージをインストール モードまたは削除モードで起動します。この関数には、コマンドラインとして Windows Installer プロパティのみを使用できます。

この種類のパッケージの修復および変更操作を構成している場合、使用できるコマンドライン パラメータの種類は、Windows Installer 機能プロパティのみです。

- ADDDEFAULT
- ADDLOCAL
- ADDSOURCE
- ADVERTISE
- COMPADDDEFAULT
- COMPADDLOCAL
- COMPADDSOURCE
- FILEADDDEFAULT
- FILEADDLOCAL
- FILEADDSOURCE
- MSIDISABLELUAPATCHING
- MsiPatchRemovalList
- MSIRESTARTMANAGERCONTROL
- MSIDISABLERMRESTART
- MSIRMSHUTDOWN
- MSIPATCHREMOVE
- PATCH
- REINSTALL
- REINSTALLMODE
- REMOVE

アドバンスト UI またはスイート/アドバンスト UI インストールは MsiConfigureProductEx を使って .msi パッケージを修復 モードまたは変更モードで起動します。この関数には、Windows Installer プロパティのみを使用できます。

パッケージの "MSI コマンドライン" 設定および "MSI サイレント コマンドライン" 設定に Windows Installer プロパティを使用している場合、以下の形式を使用してください:

MYPROPERTYNAME=MyPropertyValue

/l コマンドライン オプションをパッケージに渡してログ ファイルを生成することはできません。インストールをログ記録する場合、[パッケージ] ビューで、パッケージに対してログを有効にできます。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート」を参照してください。別の方法として、インストールをログ記録したい場合は、logging system policyまたは MsiLogging プロパティのどちらかを使用します。

アドバンスト UI またはスイート/アドバンスト UI インストールでは、.msi パッケージは常にサイレントで起動されますので注意してください。そのため、.msi パッケージのユーザー インターフェイスを隠すコマンドライン パラメータを渡す必要はありません。

.msp パッケージ

.msp パッケージのインストール操作を構成している場合、使用できるコマンドライン パラメータの種類は、Windows Installer プロパティのみです。アドバンスト UI またはスイート/アドバンスト UI インストールは MsiApplyPatch を使って .msp パッケージをターゲット システムに適用します。この関数には、コマンドラインとして Windows Installer プロパティのみを使用できます。

.msp パッケージを使ってすべての機能をアップデートするには、以下のようなコマンドライン プロパティを入力します:

REINSTALLMODE=vomus REINSTALL=ALL

.msp パッケージに含まれている特定の機能のみをアップデートするには、アップデートするコンマ区切りの機能リストに REINSTALL を設定します。

REINSTALLMODE=vomus REINSTALL=Feature1,Feature3,Feature5

アドバンスト UI またはスイート/アドバンスト UI インストールでは、.msp パッケージは常にサイレントで起動されますので注意してください。そのため、.msp パッケージのユーザー インターフェイスを隠すコマンドライン パラメータを渡す必要はありません。

InstallScript パッケージ

InstallScript パッケージの [インストール] 操作と [変更] 操作を構成する場合、コマンドラインで、アドバンスト UI およびスイート/アドバンスト UI のプロパティ ISFeatureInstall と ISFeatureRemove を使用できます。InstallScript 関数 FeatureConfigureFeaturesFromSuite は、これらのプロパティを使って、InstallScript パッケージの機能を設定します。

"コマンドライン" 設定と "サイレント コマンドライン" 設定では、次のように、これらのプロパティに、複数の機能名をカンマ区切りで設定することができます:

ISFeatureInstall=Feature1,Feature2 ISFeatureRemove=Feature3

上記の例では、**Feature1** と **Feature2** は、インストールされるように選択されています。**Feature3** は、存在していれば、削除されるように選択されています。

機能が、両方のプロパティの値で使用されている場合、**ISFeatureRemove** プロパティに設定された機能は、**ISFeatureInstall** プロパティに優先されます。

アドバンスト UI およびスイート/アドバンスト UI インストールでは、デフォルトで、InstallScript パッケージはサイレントで起動されますので注意してください。従って、InstallScript パッケージのユーザー インターフェイスを非表示にするコマンドライン パラメーターを渡す必要はありません。また、サイレント応答ファイルを使用する必要もありません。

.exe パッケージ

.exe パッケージのインストール、削除、修復、および変更操作を構成している場合、.exe ファイルがサポートしている任意のコマンドライン パラメーターを入力できます。

エンド ユーザーによる操作なしですべてのカスタマイズを完了する場合、.exe パッケージをサイレントで実行するコマンドライン パラメーターを含めることができます。

Windows Installer ベースのインストール用に InstallShield でビルドされた Setup.exe ファイルをサイレントで実行する

アドバンスト UI またはスイート/アドバンスト UI インストールで、Windows Installer ベースの Setup.exe ファイルをサイレントで実行する場合、"EXE コマンドライン" 設定と "EXE サイレント コマンド" 設定で、次の構文を使用します:

```
Setup.exe /s /v"/qn"
```

Windows Installer ベースのパッチ用に InstallShield でビルドされた Update.exe ファイルをサイレントで実行する

アドバンスト UI またはスイート/アドバンスト UI インストールで、Windows Installer ベースの Update.exe パッチをサイレントで実行する場合、"EXE コマンドライン" 設定と "EXE サイレント コマンド" 設定で、次の入力します:

```
Update.exe /s /v"/qn"
```

InstallScript インストールのセットアップ ランチャー ファイルをサイレントで実行する

アドバンスト UI またはスイート/アドバンスト UI インストールで、InstallScript の Setup.exe インストールをサイレントで起動する場合、サイレント応答ファイルを作成して、それを、スイート プロジェクトの [サポート ファイル] ビューに、サポート ファイルとして追加します。[インストール] 操作の "EXE コマンドライン" 設定と "EXE サイレント コマンドライン" 設定で、次の構文を使用します:

```
Setup.exe /s /f1"[SETUPSUPPORTDIR]¥Setup.iss"
```

アンインストールで、元の Setup.exe ファイルをサイレントで起動する代わりに、アンインストール用にサイレント応答ファイルを作成して、それを、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [サポート ファイル] ビューに、サポート ファイルとして追加します。[削除] 操作の "EXE コマンドライン" 設定と "EXE サイレント コマンドライン" 設定で、次の構文を使用します:

```
"[ProgramFilesFolder]InstallShield Installation Information¥{PRODUCT-GUID-HERE}¥Setup.exe" /s /f1"[SETUPSUPPORTDIR]¥Uninstall.iss" -remove_only -runfromtemp
```

アドバンスト UI およびスイート/アドバンスト UI 形式の式を使用して、コマンドラインを動的に構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI 形式の式を使って、アドバンスト UI またはスイート/アドバンスト UI インストールで、アドバンスト UI またはスイート/アドバンスト UI パッケージを起動するときに使用するコマンドラインを動的に構成することができます。形式化された式は、プロパティ名、環境変数リファレンス、その他の特殊文字列から構成されます。実行時に、インストールはこれらの式の値を拡張します。

一部のシナリオでは、単一のアドバンスト UI またはスイート/アドバンスト UI で形式化された式を使ってパッケージのほとんどまたはすべてを構成できます。その他のシナリオでは、各パッケージに異なるアドバンスト UI またはスイート/アドバンスト UI の形式化された式が必要です。

これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

次のサンプル シナリオは、アドバンスト UI またはスイート/アドバンスト UI プロパティを形式化された式として使用して、インストール内で値をパッケージに渡す方法を説明します。

アドバンスト UI またはスイート/アドバンスト UI プロパティを使って、アドバンスト UI またはスイート/アドバンスト UI インストール内の複数のパッケージに同じ値を渡す

アドバンスト UI またはスイート/アドバンスト UI インストールを作成するとき、アドバンスト UI またはスイート/アドバンスト UI プロパティを使って Windows Installer プロパティを設定し、各 .msi パッケージをインストールする方法を変更する場合があります。たとえば、複数言語のインストールを作成している場合、インストールする言語を制御する言語トランスフォームを選択する必要があります。この特定の例で、言語 (つまりプロパティ値) は恐らくアドバンスト UI またはスイート/アドバンスト UI プロジェクトの各パッケージの言語と同じです。

アドバンスト UI プロジェクトでは、アドバンスト UI またはスイート/アドバンスト UI プロパティ ISSelectedLanguage は、アドバンスト UI またはスイート/アドバンスト UI インストールに使用する言語を識別します。リリースでエンド ユーザーが言語を選択できるように構成する場合、InstallationLanguage ウィザード ページがエンド ユーザーに対して言語のプロンプトを行います。インストールは ISSelectedLanguage プロパティをユーザーが選択した言語の 10 進数言語 ID を含む文字列に設定します。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれる .msi パッケージを適切な言語で実行するには、各 .msi パッケージのコマンドライン設定に入力する値に以下を含みます:

TRANSFORMS="[ISPREREQDIR]¥[ISSelectedLanguage].mst"

.mst ファイルは、.msi パッケージと同じフォルダーに配置します。

基本の MSI .exe パッケージのコマンドライン設定に次の構文を使います:

/L[ISSelectedLanguage]

アドバンスト UI またはスイート/アドバンスト UI プロパティを使って、アドバンスト UI またはスイート/アドバンスト UI インストール内の複数のパッケージに異なる値を渡す

アドバンスト UI またはスイート/アドバンスト UI のプロパティを使って、アドバンスト UI またはスイート/アドバンスト UI インストール内の各パッケージまたは一部のパッケージに、異なるプロパティの値を渡すことができます。たとえば、エンド ユーザーが INSTALLDIR を各パッケージごとに異なる場所にオーバーライドできるようにしたい場合が考えられます。この例では、INSTALLDIR の値は各パッケージごとに異なります。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれるパッケージの 1 つで、プロジェクトに 定義済み BrowseFolder ウィザード ページを追加する と、InstallShield は [パッケージ] ビューのインストール操作で、以下をパッケージのコマンドライン設定に追加します。

INSTALLDIR="[ISInstallDir_PackageDisplayName]"

適切な場合、そのコマンドラインをオーバーライドすることができます。パッケージが .msi パッケージである場合、そのコマンドラインはパッケージの INSTALLDIR 値をエンド ユーザーがそのパッケージの BrowseFolder ウィザード ページで指定したパスに設定します。

パッケージが基本の MSI または InstallScript MSI .exe パッケージの場合、デフォルトのコマンドラインを変更してこの構文を使用します:

/v"INSTALLDIR=¥"[ISInstallDir_PackageDisplayName]¥""

アドバンスト UI またはスイート/アドバンスト UI インストールのエンドユーザー インターフェイスを定義する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI インストールのウィザード インターフェイス要素は、ウィザード ページおよび 2 番目のウィンドウ (ポップアップ ウィンドウとも呼ばれます) で構成されます。アドバンスト UI またはスイート/アドバンスト UI プロジェクトには、標準の定義済みウィザード ページおよび 2 番目のウィンドウが多く含まれています。ページやウィンドウを追加および削除、またそのレイアウトや動作をカスタマイズすることができます。

アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーでは、アドバンスト UI またはスイート/アドバンスト UI インストールのウィザード インターフェイスが優先されるため、.msi、.msp、および InstallScript パッケージ フォーマットのユーザー インターフェイスは自動的に抑制されます。アドバンスト UI またはスイート/アドバンスト UI プロジェクトに .exe パッケージが含まれている場合、そのユーザー インター

フェイスを手動で抑制し、アドバンスト UI またはスイート/アドバンスト UI インストールで、そのユーザー インターフェイスをサイレントで起動する必要があります。そうしなかった場合、インストールの実行時に、アドバンスト UI またはスイート/アドバンスト UI のウィザード ページと .exe パッケージのユーザー インターフェイスの 2 つの異なるユーザー インターフェイスが表示されることになります。詳細については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI インストールのエンドユーザー インターフェイスの定義についての詳細は、「ウィザード インターフェイスを使って作業する」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI インストールでハイコントラストモードをサポート

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンストUIまたはスイート/アドバンスト UI インストール中、各ウィザードページには、ウィザード アイコン、ウィザード キャプション、ヘッダー イメージ、ヘッダー テキスト、本文テキスト、ナビゲーション ボタン、ナビゲーション ボタン ラベルなどのコア要素が表示されます。ヘッダー テキスト、本文テキスト、およびナビゲーション ボタンのラベルの読みやすさは、ターゲット システムに設定されているコントラスト モードによって異なります。InstallShield は、一般的に、各ウィザード ページのヘッダー テキスト、本文テキスト、およびナビゲーション ボタンのラベルが、すべてのコントラスト モード、特にエンドユーザーのシステムで構成された高コントラスト モードではっきりと読み取り可能であるように構成されています。InstallShield は、ターゲットまたはエンド ユーザーのシステムで設定されたコントラスト モードにかかわらず、自動的にアクセシビリティと読みやすさを維持します。

一般的に、単色とグラデーションは、各ウィザード ページのヘッダーテキスト、本文テキスト、およびナビゲーション ボタン用に、**ウィザードインターフェイス** エクスプローラーのブラシを使用して最初に指定されますが、これらの定義された単色とグラデーションは、エンドユーザーのシステムで設定された高コントラスト モードの色によって上書きされます。InstallShield は、一貫したアクセシビリティを確保するために、この動作を完全にサポートしています。

アドバンスト UI またはスイート/アドバンスト UI インストールのエンドユーザー インターフェイスの定義についての詳細は、「ウィザード インターフェイスを使って作業する」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトの UI で機能ステータスおよびその他の機能データを参照する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト インストールの実行時に、機能関連の情報に依存する特定の UI 動作をトリガする必要がある場合があります。この情報の使用例を次に示します:

- 特定の機能のインストールが選択されなかった場合に、特別な対応ウィザード ページを省略することができます。
- 特定の機能が既にインストールされている場合に、ウィザード ページのチェックボックスを無効にすることができます。
- 拡張条件 DLL を作成して、インストールが実行時に行う機能関連のチェックするカスタム条件を定義できます。
- ターゲット システム上で各機能が必要とする空き容量を表示するカスタム機能ツリー ウィザード ページを表示するように、インストールを構成できます。

アドバンスト UI またはスイート/アドバンスト UI インストールの実行中にチェックすることができる、様々な機能関連の疑似プロパティを以下に示します。いずれも、*名前*部分を機能の名前に置換します。

テーブル 4 ・ アドバンスト UI またはスイート/アドバンスト UI インストールの UI における機能関連の疑似プロパティ

疑似プロパティ	種類	説明
FEATURE[*name*].actionState	読み取り/書き込み	これは、指定された機能の次の動作を示します。次の値が可能です: ・ **インストール**—この状態は、機能のインストールが設定されていることを示します。この状態は機能が既にインストール済みでない場合のみ有効です。 ・ **削除**—この状態は、機能の削除が設定されていることを示します。この状態は機能が既にインストールされている場合のみ有効です。 ・ **ヌル (空白文字列)**—空白文字列の状態は動作を要求しません。これは常に有効です。
FEATURE[*name*].installState	読み取り専用	これは、指定された機能の現在の状態を示します。次の値が可能です: ・ **0**—機能は現在ターゲット システムに存在しません。 ・ **1**—機能の一部がターゲット システムに存在します。つまり、この機能の一部のパッケージが存在します。 ・ **2**—一部の機能の子機能のみがターゲット システムに存在します。 ・ **3**—機能が、現在ターゲット システムに存在します。
FEATURE[*name*].displayName	読み取り専用	これは、実行時に適切なウィザード ページ上で指定された機能に表示される名前を示します。
FEATURE[*name*].description	読み取り専用	これは、実行時にウィザード ページ上に表示される特定の機能の説明を示します。
FEATURE[*name*].cost	読み取り専用	これは、指定された機能がターゲット システム上で必要とする空き容量を示します。

読み取り/書き込み疑似プロパティ FEATURE[name].actionState は、例えば拡張条件やウィザード インターフェイスのコントロールの ”動作” 設定を通して設定できます。

これらの任意の疑似プロパティは、拡張条件またはウィザード インターフェイスのコントロールの ”表示” または ”有効化” 設定のバインド条件を通して読み込むことができます。

アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの様々な領域で、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む形式化された式を埋め込むことができます。実行時、インストールはこれらの式の値を拡張します。次の種類の形式化された式がサポートされています。

- **プロパティ**—プロパティの値を解決するには、プロパティ名を角括弧で囲みます。たとえば、**[MYPROPERTY]** は実行時に MYPROPERTY というプロパティの値に解決します。プロパティが定義されていない場合、この式は空白文字列に解決します。
- **再帰的なプロパティ**—プロパティの解決済みの値を解決するには、プロパティ名を追加の角括弧のセットで囲みます。たとえば **[[PROPERTY1]]** は、**PROPERTY1** に名前が格納されているプロパティの値に解決します。つまり、**PROPERTY1** に **PROPERTY2** という値が格納されている場合、**PROPERTY2** に格納されている値に解決されます。
- **エスケープされた文字**—式に特殊文字 (リテラル括弧、コンマ、または角括弧など) を含めるには、特殊文字の前に円記号を付けて、結果となる文字列を角括弧で囲みます。たとえば、**[¥[]Text[¥]]** と入力すると **[Text]** に置換されます。
- **ヌル文字**—式にヌル文字を埋め込むには、チルダを角括弧で囲む、つまり**[~]** を埋め込みます。
- **環境変数**—環境変数の値を使用するには、環境変数の名前の前にパーセント記号を付けて、結果文字列を角括弧で囲みます。たとえば、**[%PATH]** は PATH 環境変数の値として解決します。
- **オブジェクト式**—ターゲット システム上でファイル、レジストリ エントリ、オペレーティング システム、その他の項目についての情報を検索するには、オブジェクト式を使用します。これによって、ターゲット システム固有の条件に基づいて、実行時にアドバンスト UI またはスイート/アドバンスト UI の多くの設定をダイナミックに構成することができます。オブジェクト式は、規則 **[@Object(Parameters, ...).Property(Parameters, ...)]** を使用します。オブジェクト式の書き方についての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトでオブジェクト式を書いて、ターゲット システムを検索する」を参照してください。

プロパティ式の中など、その他の形式化された式内にオブジェクト式を埋め込むことができます。オブジェクト式のパラメーター内に、プロパティ式およびその他の形式化された式を埋め込むこともできます。詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトでオブジェクト式を書いて、ターゲット システムを検索する」を参照してください。

構文エラー、オブジェクト式のオブジェクト エラー、および無効な形式化された式は、実行時に空白文字列として評価されます。一部の状況において、解析エラーまたは形式化された式の評価についての情報は、アドバンスト UI またはスイート/アドバンスト UI インストールのデバッグ ログ ファイルに記録されます。

形式化された式を使用できる、アドバンスト UI およびスイート/アドバンスト UI プロジェクトの領域

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの様々な領域にある多くの設定で、形式化された式を使用できます。一部の種類のアクションでは、形式化された式を使用することができます。InstallShield の次の領域は、形式化された式を指定できる場所の例です。

- [パッケージ] ビュー: パッケージのファイルを起動するために使用するコマンドラインを入力する設定

  詳細については、「アドバンスト UI およびスイート/アドバンスト UI 形式の式を使用して、コマンドラインを動的に構成する」を参照してください。

- [プロパティ マネージャー] ビュー: プロパティの [値] 列[形式化済み] チェックボックスを確実に選択して、プロパティの値の形式化された式が実行時に解決されるようにします。実行時の解決処理は、インストール中に OnBegin イベントのアクションが実行される前、早い段階で発生します。

  詳細については、「[プロパティ マネージャー] ビュー」を参照してください。

- たとえば、[機能] ビューの機能の "表示名" 設定や、[ウィザード インターフェイス] ビューの UI コントロールの "テキスト" 設定など、文字列エントリを使用する設定。

- 終了条件、検出条件、インストール条件、その他の種類の条件を構成するためのテキストを入力することができるサブ設定の一部。たとえば、"ファイルの比較"、"レジストリの比較"、および "プロパティの比較" 条件確認の "比較対象" サブ設定内。

  詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類」を参照してください。

- プロパティを設定するアクションの [イベント] ビューの場合: "プロパティ値" 設定内"プロパティ値の形式化" 設定で [はい] を選択して、プロパティの値の形式化された式が実行時に解決されるようにします。

  詳細については、「スイート/アドバンスト UI インストールに含まれるプロパティを設定するアクションでの作業について」を参照してください。

- ISuiteExtension ポインターを受け取る任意のアクション: FormatProperty メソッドの bstrValue パラメーター

  詳細については、次を参照してください。

  - アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、拡張条件 DLL を作成する
  - スイート/アドバンスト UI インストールに含まれるアクションの DLL ファイルでの作業について
  - スイート/アドバンスト UI インストールでマネージ コード アクションを使用する
  - スイート/アドバンスト UI インストールで .NET Core アクションを使用する

- InstallScript アクション コード: InstallScript 関数 **SuiteFormatString** の szValue パラメーター

  詳細については、「SuiteFormatString」を参照してください。

- PowerShell アクションの場合: format-suiteproperties cmdlet は形式化された式の値を展開します。

  詳細については、「スイート/アドバンスト UI インストールに含まれるアクションの PowerShell スクリプトでの作業について」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでオブジェクト式を書いて、ターゲット システムを検索する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの様々な設定では、ファイル、レジストリ エントリ、その他のアイテムについての情報をターゲット システムでクエリするオブジェクト式を埋め込むことができます。オブジェクト式には、この規則を使用します:

[@Object(Parameters, ...).Property(Parameters, ...)]

各オブジェクト式には、オブジェクト固有プロパティの集まりである、オブジェクトへの参照が含まれています。オブジェクトとプロパティにはパラメーターを含めることができます。

たとえば、次の Platform オブジェクト式は、アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のマシンのアーキテクチャ (x86、x64、IA64、ARM、または不明) を取得します:

[@Platform.Architecture]

スケジュール済みの処理が完了した後、次の Package オブジェクト式は、パッケージ GUID 0F283EC0-E9D1-4D18-801D-0014F6CA96DF を持つパッケージの終了コードを取得します。

[@Parcel(0F283EC0-E9D1-4D18-801D-0014F6CA96DF).ExitCode]

オブジェクト式書き込みについてのガイドライン

オブジェクト式を書き込む際には、次のガイドラインに留意してください:

- 構文エラー、オブジェクト式のオブジェクト エラー、および無効な形式化された式は、実行時に空白文字列として評価されます。一部の状況において、解析エラーおよび形式化された式の評価についての情報は、アドバンスト UI またはスイート/アドバンスト UI インストールのデバッグ ログ ファイルに記録されます。
- プロパティ式その他のオブジェクト式のような、別の形式化された式内にオブジェクト式を埋め込むことができます。

  次の式では、Registry オブジェクト式が File オブジェクト式にパラメーターの一部として埋め込まれています。

  [@File([@Registry(HKLM¥Software¥MyProduct).KeyValue(MyProductPath)]¥MyProduct.exe).Version]

  MyProduct.exe ファイルが **MyProductPath** 値データに指定された場所にあるとき、File オブジェクト式はファイルのバージョンを返します。ファイルがその場所に見つからなかった場合、またはレジストリ値が存在しなかった場合、File オブジェクト式は空白文字列を返します。

- オブジェクト式のパラメーター内に、プロパティ式およびその他の形式化された式を埋め込むことができます。

  次の File オブジェクト式は、32 ビットの ProgramFilesFolder にある MyProduct.exe と呼ばれるファイルの最終更新日時を取得します。(*false* のパラメーター値は、インストールが 64 ビットの場所を確認しないことを示します。)

  [@File([ProgramFilesFolder]MyCompany¥MyProduct.exe,false).Date(modified)]

- アドバンスト UI およびスイート/アドバンスト UI インストールの形式化された式のパーサーに、引用符は不要です。たとえば、オブジェクト式内のパラメーターに 1 つ以上のスペースが含まれるパスまたは別のアイテムが含まれる場合、そのパスを引用符で囲む必要はありません。
- オブジェクト式のパラメーター内にリテラル角括弧、丸括弧、またはコンマを含む必要がある場合、パラメーター内に特殊文字を含む値に解決する形式化されたプロパティ式 ([SpecialCharacterString])、またはエスケープ文字式 (A[¥,] B[¥,] および C) を埋め込みます。
- オブジェクト式を使ってシステム検索を行うことができます。ある検索の出力を、異なるオブジェクト式のパラメーターへのインプットとして使用することができます。
- オブジェクト式の実行時の解決には、2 つの異なる段階 (解析と評価) があります。実行時、任意の形式化された式が解析されるとき 、まず始めにオブジェクト式のパラメーターが解析されます。すべての形式化された式が解析されたあと、解析された出力が評価されます。その他の解析は、最初の解析が完了した後に始まります。

  したがって、式 [MYPROP] の値が [MYPROP] の場合、[MYPROP] がターゲット システム上で評価される解析された出力となります。この状況で、再帰ルックアップが発生することはありません。
- サンプル オブジェクト式、および使用可能な各オブジェクトについての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用可能な式のオブジェクト リファレンス」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI インストール中に、ユーザー アカウント制御のプロンプト回数を最小限に抑える



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

特定の詳細の一部は、スイート/アドバンスト UI プロジェクト タイプにのみ適用します。これらの違いについては、必要に応じて記述されています。



エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

Windows Vista 以降およびユーザー アカウント制御 (UAC) の目的は、ユーザーが常に標準ユーザーとして実行できるようにすることです。昇格が必要になることは、ほとんどありませんが、必要となる場合、できるだけ短い時間に限られる必要があります。

InstallShield のいくつかの異なる領域にある設定は、アドバンスト UI またはスイート/アドバンスト UI インストールで昇格された権限の UAC の同意または資格情報プロンプトをトリガするかどうかに影響します。さらに、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージの設定は、昇格された権限の UAC プロンプトをエンド ユーザーに表示するかどうかにも影響します。Windows Vista 以降のシステムおよび Windows Server 2008 以降のシステム上でエンド ユーザーがインストールを実行する際の UAC の動作を適切に定義するためには、これらの異なる設定をよく理解することが必要です。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおける UAC 関連の設定

アドバンスト UI またはスイート/アドバンスト プロジェクトの次の設定は、UAC プロンプトを表示するかどうかを決定するのに役立ちます:

- **必要実行レベル**—[リリース] ビューにあるこの設定を使用して、インストールの Setup.exe ファイルが必要とする最小実行レベルを指定します。InstallShield は、Setup.exe ランチャーに埋め込まれるアプリケーション マニフェストで選択した値 (管理者、最高権限、または起動者) を使用します。詳細については、「Windows Vista 以降のプラットフォームでのセットアップランチャーの必要実行レベルを指定する」を参照してください。
- **昇格された権限が必要**—[パッケージ] ビューのパッケージでこの設定を使って、パッケージに昇格されたシステム権限が必要かどうかを指定します。
- **管理者権限が必要**—この設定を使って、[イベント] ビューのプロジェクトに追加したアクションが管理者権限を必要とするかどうかを指定します。



***プロジェクト・** スイート/アドバンスト UI プロジェクトには [イベント] ビューが含まれていて、アクションをイベントに割り当てることができます。*

アドバンスト UI またはスイート/アドバンスト UI インストールに含まれているパッケージの UAC 関連の設定

アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれるパッケージで、昇格された権限のための UAC プロンプトをトリガする様々な方法があります。詳細については、「インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI インストールの UAC 関連のランタイム動作

Windows Vista 以降および Windows Server 2008 以降における、次の UAC 関連の動作にご注意ください:

- "必要実行レベル" が [起動者] に設定されている場合で、プロジェクト内のアクションには管理者権限が不要、またインストール内のパッケージにも管理者権限が不要なとき、インストール中にエンド ユーザーに UAC プロンプトが表示されることはありません。インストール全体 (UI の表示、アクションの起動、およびパッケージの実行) は、昇格されないプロセスで実行します。
- "必要実行レベル" が [起動者] に設定されている場合で、プロジェクト内のアクションには管理者権限が不要、インストール内の 1 つ以上のパッケージで管理者権限が必要な場合、インストール中エンド ユーザーに UAC プロンプトが 1 回表示されます (さらに、再起動の度に追加の UAC プロンプトが表示されます) 。UAC

プロンプトは、エンド ユーザーが UI 上でインストール ボタンをクリックした後に表示されます。"昇格された権限が必要" に [はい] が設定されているパッケージは、新しい個別のプロセスで昇格された権限で起動されます。"昇格された権限が必要" に [いいえ] が設定されている、UI のその他の部分、アクション、およびパッケージは、元の昇格されないプロセスで起動されます。

- "必要実行レベル" が [起動者] に設定されている場合で、プロジェクト内の 1 つ以上のアクションで管理者権限が必要な場合、インストール中エンド ユーザーに UAC プロンプト 1が 回表示されます (さらに、再起動の度に追加の UAC プロンプトが表示されます)。UAC プロンプトのタイミングは、昇格が最初に必要な時によって異なります。

  たとえば、一番最初にスケジュールされた管理者権限が必要なアクションが OnBegin イベント中に発生するように構成されている場合 (パッケージが実行される前)、エンド ユーザーが Setup.exe ファイルを起動したすぐ後に UAC プロンプトが表示されます。エンド ユーザーが同意または認証情報を入力した後、インストールは新しい昇格されたプロセスでこのアクション、およびそれに続く昇格が必要なアクションとパッケージを起動します。後に続く昇格が不要なアクションとパッケージは、UI の残りの部分と共に、元の昇格されないプロセスで起動されます。

  ただし、最初にスケジュールされている管理者権限が必要なアクションが、権限の昇格が必要なパッケージの後に発生するように構成されている場合、エンド ユーザーが UI 上でインストール ボタンをクリックした後に UAC プロンプトが表示されます。エンド ユーザーが同意または認証情報を入力した後、インストールは新しい昇格されたプロセスでパッケージ、およびそれに続く昇格が必要なアクションとパッケージを起動します。後に続く昇格が不要なアクションとパッケージは、UI の残りの部分と共に、元の昇格されないプロセスで起動されます。

アドバンスト UI またはスイート/アドバンスト UI パッケージにおけるターゲット システムの再起動

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield はいくつかの方法を使って、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージを実行した後にターゲット システムを再起動するかどうかを判別します。

.exe パッケージにおけるレジストリ キーの変更

アドバンスト UI またはスイート/アドバンスト UI に含まれる .exe パッケージが実行された後に次のレジストリ キーのどれかが変更されたとき、ターゲットマシンの再起動が必要な場合があります。

- HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥RunOnce

- HKEY_CURRENT_USER¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥RunOnce
- (64 ビット) HKEY_LOCAL_MACHINE¥Software¥Microsoft¥Windows¥CurrentVersion¥RunOnce
- HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥RunOnceEx
- (64 ビット) HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥RunOnceEx
- HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Control¥Session Manager、PendingFileRenameOperations の値

レジストリキーは パッケージが実行された後に考慮されます。これらの数字が異なる場合、ファイルがシステムの再起動を試みると同時にインストールを終了するものと見なされます。

アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに指定されたリターン コード

アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージの終了コードが、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [パッケージ] ビューにある "再起動コード" 設定でパッケージの適切な操作の種類 (インストール、削除、修復、または変更) に指定されたリターン コードと一致する場合、ターゲット システムの再起動が必要な場合があります。

リターンコード 1641 および 3010

アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージで実行される .msi ファイルに再起動ステートが戻された場合、ターゲット システムの再起動が必要な場合があります。標準 Windows Installer の再起動リターンコードは以下の通りです。

- ERROR_SUCCESS_REBOOT_INITIATED (1641)
- ERROR_SUCCESS_REBOOT_REQUIRED (3010)

パッケージに指定された動作

アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [パッケージ] ビューにあるパッケージの "再起動の要求" 設定を使って、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに前述の条件が当てはまる場合、または当てはまらない場合の動作を指定できます。各操作の種類 (インストール、削除、修復、および変更) に対して異なる動作を指定できます。

詳細については、「再起動が必要なアドバンスト UI またはスイート/アドバンスト UI パッケージの動作を指定する」を参照してください。

再起動が必要なアドバンスト UI またはスイート/アドバンスト UI パッケージの動作を指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで新しいパッケージを構成するとき、または既存パッケージの設定を変更するとき、ターゲット システムの再起動が必要と見られるときに、アドバンスト UI またはスイート/アドバンスト UI インストールが行う処理を指定できます。たとえば、ターゲット システムを再起動する前にインストールがエンド ユーザーに対してプロンプトを行う場合、または再起動そのものをスキップする場合が考えられます。アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [パッケージ] ビューでは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれる各パッケージに対して、また各操作の種類 (インストール、削除、修復、または変更) に対して適切な対応を行うために必要な柔軟性が提供されています。

タスク

ターゲット システムの再起動を必要とするアドバンスト UI またはスイート/アドバンスト UI パッケージの動作を指定するには、次の手順に従います:

1. 再起動動作を構成したいパッケージを含むアドバンスト UI またはスイート/アドバンスト UI プロジェクトを開きます。
2. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
3. **[パッケージ]** エクスプローラーで、構成するパッケージを選択します。
4. 適切な操作 (**インストール**、**削除**、**修復**、または**変更**) の下にある **"再起動の要求"** 設定で適切なオプションを選択します。

次のテーブルは、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージが行う様々な再起動関連の動作のための適切なオプションについて説明します。

テーブル 5 ・ アドバンスト UI またはスイート/アドバンスト UI パッケージの再起動動作

アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージの動作	"再起動の要求" 設定のガイドライン
エンド ユーザーにプロンプトを表示して、エンド ユーザーの応答に従ってターゲット システムを再起動します	"再起動の要求" 設定で以下のオプションから 1 つを選択することを考慮します: ・ **コンピューターの再起動を許可する**—このオプションは、ほとんどの状況で使用できます。 ・ **常にコンピューターを再起動する**—このオプションは、エンド ユーザーがターゲット マシンを再起動しないオプションを選択している場合に、エンド ユーザーによるパッケージのプロンプトに対する応答と競合する可能性があります。この状況が発生する恐れがある場合に、常にシステムを再起動させるためには、パッケージのプロンプトを抑制して、パッケージの "再起動の要求" 設定で適切なオプションを選択してください。

テーブル 5 ・ アドバンスト UI またはスイート/アドバンスト UI パッケージの再起動動作 (続き.)

アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージの動作	"再起動の要求" 設定のガイドライン
常に再起動をトリガする。エンドユーザーにプロンプトを行う、または行わない。	"再起動の要求" 設定で以下のオプションから 1 つを選択することを考慮します: ・ **コンピューターの再起動を許可する**—このオプションは、ほとんどの状況で使用できます。 ・ **常にコンピューターを再起動する**—このオプションは、エンドユーザーがターゲット マシンを再起動しないオプションを選択している場合に、エンド ユーザーによるパッケージのプロンプトに対する応答と競合する可能性があります。この状況が発生する恐れがある場合に、常にシステムを再起動させるためには、パッケージのプロンプトを抑制して、パッケージの "再起動の要求" 設定で適切なオプションを選択してください。
リターン コードまたはレジストリを使って、再起動が必要であることを示す。エンド ユーザーにプロンプトを行わない。	"再起動の要求" 設定で以下のオプションから 1 つを選択することを考慮します: ・ **再起動の要求を遅延して、Setup.exe を終了する、またはコンピューターを再起動する**—後続のパッケージのためにターゲットシステムを再起動する必要がある場合は、このオプションを選択します。 ・ **常に再起動の要求を表示して、Setup.exe を終了する、またはコンピューターを再起動する**—ターゲット システムを即時に再起動する必要がある場合は、このオプションを選択します。 ・ **再起動の要求を無視する**—パッケージの再起動要求が不要な場合は、この無視オプションを選択します。 ・ **再起動の要求を遅延して、Setup.exe を終了する、またはコンピューターを再起動する**—後続のパッケージに再起動が不要な場合は、このオプションを選択します。アドバンスト UI またはスイート/アドバンスト UI インストールに含まれる 1 つ以上のパッケージで再起動が必要な場合、このオプションを使用すると再起動の回数を最小限に抑えることができます。 後続のパッケージを実行する前に再起動が必要な場合で、そのパッケージの実行前に必ず再起動が行われる保証がない場合、この遅延オプションの選択は避けてください。
再起動を必要とするが、エンド ユーザーにプロンプトを表示せず、再起動の必要性も表示しない。	"再起動の要求" 設定で以下のオプションの選択を考慮してください: ・ **常に再起動の要求を表示して、Setup.exe を終了する、またはコンピューターを再起動する**—再起動が必要な場合は、このオプションを選択します。

アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



エディション: *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI インストールがインストール モードまたはメンテナンス モードのどちらで実行されているかによって、インストールは Web サイトでホストされているアドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーが更新されているかどうかを確認し、それをダウンロードするか、エンド ユーザーにダウンロードのオプションを提供します:

- インストールが [インストール] モードで実行中の場合、Web サイトでホストされているアドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーが更新されているかどうかを自動的に確認します。アップデートが利用可能な場合、インストールはそれをダウンロードしてから起動します。
- インストールが [メンテナンス] モードで実行中の場合、Web サイトでホストされているアドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーが更新されているかどうかを確認します。アップデートが利用可能な場合、インストールで表示されるメンテナンス ウィザードに **[アップデート]** ボタンが含まれます。エンド ユーザーがこのボタンをクリックすると、スイート エンジンがアップデートをダウンロードして起動します。

アプリケーションがターゲット マシン上に配置されたとき、アプリケーションのアップデートを定期的に確認するパラメーターをビルド前に構成する方法についての詳細は、「アプリのアップデートを確認する」を参照してください。

ダウンロード可能なアップデートをサポートするための要件と推奨事項

アドバンスト UI またはスイート/アドバンスト UI インストールにダウンロード可能なアップデートのサポートを追加して、アドバンスト UI またはスイート/アドバンスト UI プロジェクト用に、ベースのセットアップ ランチャーとアップデート用セットアップ ランチャーを作成する場合、次のガイドラインが推奨されます。

- isupdate.xml という名前のメタデータ ファイルをホストするのに使用する URL を決定します。

  ビルド時に InstallShield は isupdate.xml ファイルを作成して、プロジェクトのビルド フォルダー内の UpdateMetadata サブフォルダーに配置します。この isupdate.xml ファイルは、アドバンスト UI またはスイート/アドバンスト UI インストールの異なるバージョンとそのダウンロード場所を識別します。このファイルをサイトにアップロードする前に、必要に応じて手動でアップデートすることができます。

  ベースのアドバンスト UI またはスイート/アドバンスト UI プロジェクトで、アップデート URL パスを指定する必要があります。

- アップデートのリリース準備ができたら、isupdate.xml とアップデートをサイトにアップロードします。

- 最低限として、アップデート セットアップ ランチャーは、デジタル署名が必要です。エンドユーザー側での使用感を考慮した場合、アップデート セットアップ ランチャーとベース セットアップ ランチャーに同一のデジタル署名を使用することをお勧めします。
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、アップデート セットアップ ランチャーの構成を行うとき、パッケージのランタイムの場所は、セットアップ ランチャーからの抽出か、Web からのダウンロードである必要があります。アップデート セットアップ ランチャーは、ソース メディアに格納されているパッケージには依存できません。

  ベース セットアップ ランチャーは、ソース メディアに格納されているパッケージを使用することができます。

  アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに対してランタイムの場所を指定する方法についての詳細は、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに対してランタイムの場所を指定する」を参照してください。

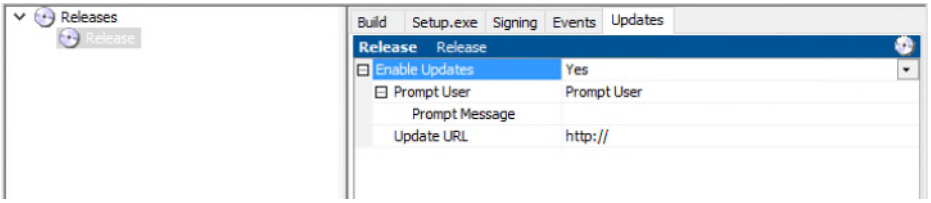
  アドバンスト UI またはスイート/アドバンスト UI プロジェクトでは、すべてのパッケージのランタイムの場所を上書きできます。詳細については、「リリース レベルで、アドバンスト UI またはスイート/アドバンスト UI パッケージのランタイムの場所を指定する」を参照してください。
- アドバンスト UI またはスイート/アドバンスト UI インストールには、[プログラムの追加と削除] エントリが必ず必要です。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールの [プログラムの追加または削除] エントリ」を参照してください。
- アップデート アドバンスト UI またはスイート/アドバンスト UI プロジェクトのスイート GUID は、ベース プロジェクトのスイート GUID と一致しなくてはなりません。この設定は [一般情報] ビューにあります。

ダウンロード可能なアップデートのサポートを有効化する

タスク

ダウンロード可能なアップデートのサポートをベースのアドバンスト UI またはスイート プロジェクトで有効化するには、以下の手順に従います:

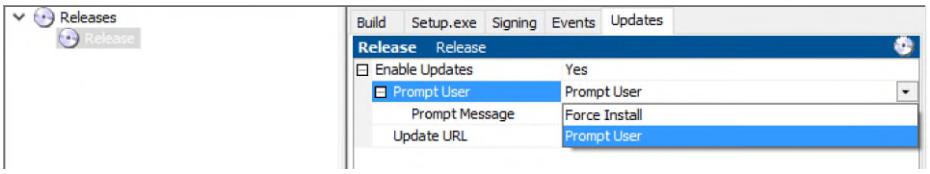
1. 将来的にダウンロード可能なアップデートを有効にするアドバンスト UI またはスイート/アドバンスト UI プロジェクトを開きます。
2. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
3. **[リリース]** エクスプローラーで、構成するリリースを選択します。
4. 次いで、**[アップデート]** タブの ***"アップデートの有効化"*** 設定で、適切な値を選択します。
   - **いいえ** – ダウンロード可能なアップデートを無効化します。
   - **はい** – ダウンロード可能なアップデートを有効化します。



ダウンロード可能なアップデートのより新しいバージョンを構成する

タスク ***ダウンロード可能なアップデートのサポートをベースのアドバンスト UI またはスイート プロジェクトで構成するには、以下の手順に従います:***

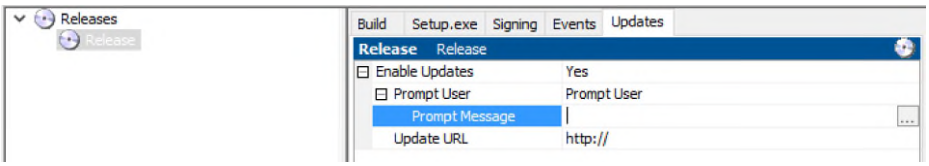
1. アドバンスト UI またはスイート/アドバンスト UI プロジェクトを開きます。
2. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
3. **[リリース]** エクスプローラーで、構成するリリースを選択します。
4. 次いで、**[アップデート]** タブの **"ユーザーに確認"** 設定で、適切な値を選択します。
   - **強制インストール** – エンド ユーザーはスイート パッケージ (使用可能な場合) の新しいバージョンを強制的にダウンロードおよびインストールすることができます。
   - **ユーザーに確認する** – エンド ユーザーがより新しいバージョンをダウンロードするか、スキップして現在のインストール処理を続行するかどちらかのオプションを受け取ります。



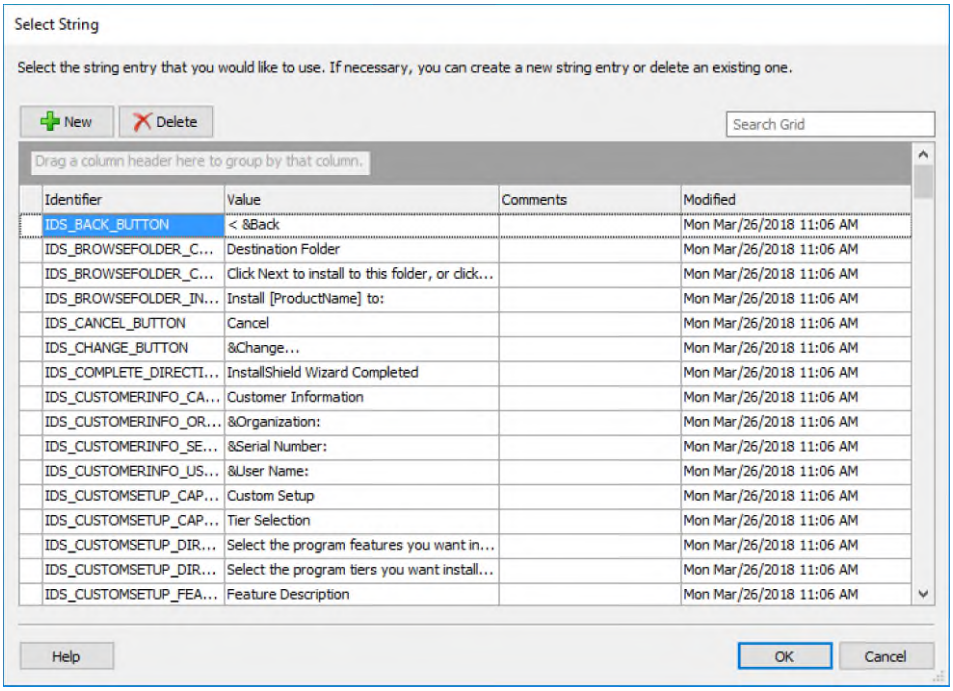
確認メッセージを構成する

タスク ***ベースのアドバンスト UI またはスイート プロジェクトにより新しいバージョンの確認メッセージを構成するには、以下の手順に従います:***

1. **[アップデート]** タブにある **"確認メッセージ"** 設定で、
   - エンド ユーザーに表示するメッセージを入力するか、
   - 省略記号ボタン (...) をクリックして、新しいメッセージを作成、または文字列一覧からローカライズ済みのメッセージを選択します。



2. **[アップデート]** タブにある **"確認メッセージ"** 設定で、**[文字列の選択]** ダイアログを表示する省略記号ボタン (...) をクリックします。
3. **[文字列の選択]** ダイアログで **[新規作成]** をクリックして新しい文字列を作成してから **[OK]** をクリックします。



メモ・より新しいバージョンをダウンロードするか、スキップして現在のインストール処理を続行するための [はい] または [いいえ] オプションを表示する確認メッセージがエンド ユーザーに表示されます。

ダウンロード可能なアップデートのサポートを追加する

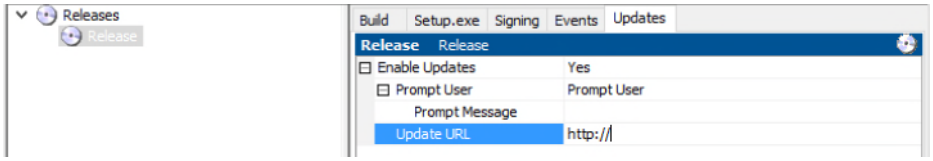
タスク ***ダウンロード可能なアップデートのサポートをベースのアドバンスト UI またはスイート プロジェクトに追加するには、以下の手順に従います:***

1. 将来的にダウンロード可能なアップデートを有効にするアドバンスト UI またはスイート/アドバンスト UI プロジェクトを開きます。
2. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
3. **[リリース]** エクスプローラーで、構成するリリースを選択します。
4. **[アップデート]** タブにある **"アップデート URL"** 設定で、ターゲット システムでダウンロード可能にするアップデート セットアップ ランチャーのパスとして今後使用する絶対パス URL (**http://** または **https:/** で始まります) を入力します。

   isupdate.xml を "アップデート URL" 設定で指定されているのと同じパスにアップロードする必要があります。たとえば、"アップデート URL" 設定の値が **http://MyWebSite.com/path/setup.exe** の場合、メタデータの URL は **http://MyWebSite.com/path/isupdate.xml** となります。

   "アップデート URL" 設定で指定されているセットアップ ランチャーの場所は、isupdate.xml ファイルに適切なパスを指定する方法でオーバーライドすることができます。ただし、isupdate.xml ファイルが "アップデート URL" で指定されている .exe ファイルと同じフォルダーに存在しなくてはなりません。そうでない場合、エンドユーザーがダウンロード可能なアップデートを使用することはできません。

5. **[署名]** タブでは、デジタル署名情報を入力して、Setup.exe ランチャーが署名されるようにリリースを構成することをお勧めします。



セットアップランチャーを作成する

タスク

アドバンスト UI またはスイート/アドバンスト UI プロジェクトのアップデート セットアップ ランチャーを作成するには、以下の手順に従います:

1. 更新するアドバンスト UI またはスイート/アドバンスト UI プロジェクトを開き、必要に応じて、更新します。
2. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
3. **[リリース]** エクスプローラーで、構成するリリースを選択します。
4. **[署名]** タブで、デジタル署名情報を入力して、Setup.exe ランチャーが署名されるようにリリースを構成します。ここでは、ベースの Setup.exe ランチャーの署名に使用したのと同じデジタル署名を使用することをお勧めします。
5. リリースをビルドします。

   InstallShield は isupdate.xml ファイルを作成して、プロジェクトのビルド フォルダー内の UpdateMetadata サブフォルダーに配置します。XML ファイルには、アドバンスト UI またはスイート/アドバンスト UI インストールの異なるバージョンの場所を識別する次のような要素が含まれています:

   <Update Url="http://www.mysite.com/MyProduct/Setup.exe" Version="1.00.0000" Id="{8E23F36C-5EA8-475D-8EAE-09FA36103975}"/>

   このファイルをサイトにアップロードする前に、必要に応じて編集することができます。

   たとえば、起動後にアップデート セットアップ ランチャーへのパスを変更する場合、Url 属性の値を変更してから指定の場所に isupdate.xml ファイルを再びアップロードします。

   同じ isupdate.xml を使って複数製品のアップグレードを定義したい場合、各製品の Update 要素をそのファイルに追加することができます。
6. isupdate.xml ファイルとアップデート セットアップ ランチャーをサイトにアップロードします。

ダウンロード可能なアップデートの実行時の動作

ダウンロード可能なアップデートをサポートするインストールの実行時の動作は、インストールが実行中のモードによって異なります。

インストール モードの動作

ベースのアドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーがインストール モードで実行するとき、セットアップ ランチャーはアップデート URL パスにある isupdate.xml ファイルを確認します。

1. isupdate.xml ファイルが存在し、指定のパスにある新しいバージョンのインストールが参照されている場合、ベース アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーはそれを自動的にダウンロードしてデジタル署名を検証します。
2. isupdate.xmlが新しいバージョンのインストールを参照しない場合、またはisupdate.xml が存在しない場合、アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーはベース プロジェクトで構成されているアップデート URL パスにダウンロードがあるかどうかを確認します。ダウンロード可能なアイテムが存在する場合、自動的にダウンロードが実行され、デジタル署名が確認されます。
3. ベース プロジェクトで構成されているアップデート URL パスにダウンロードが存在しない場合、ベース アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーは、アップデートをダウンロードしないまま続行します。

メンテナンス モードの動作

アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーがメンテナンス モードで実行するとき、アドバンスト UI または スイート/アドバンスト UI セットアップ ランチャーはアップデート URL パスにある isupdate.xml ファイルを確認します。

1. isupdate.xml ファイルが存在し、指定のパスにある新しいバージョンのインストールを参照する場合、インストールは ISUpdateAvailable プロパティを 1 に設定して、[アップデート] ボタンが含まれまれている MaintenanceUpdateWelcome ウィザード ページを表示します。また、インストールは ISUpdateVersion プロパティをダウンロード可能なスイート/アドバンスト UI またはアドバンスト UI インストールのバージョンと等しく設定します。

   エンド ユーザーが [アップデート] ボタンをクリックすると、セットアップ ランチャーがそれをダウンロードしてからデジタル署名を検証します。
2. isupdate.xml ファイルにインストールの新しいバージョンへの参照が含まれていない場合、またはisupdate.xml ファイルが存在しない場合、インストールは [アップデート] ボタンを含まない標準のメンテナンス ウィザード ページ、MaintenanceWelcome を表示します。

デジタル署名の検証

アドバンスト UI または スイート/アドバンスト UI インストールがアップデートをダウンロードした場合、それを起動する前にデジタル署名が検証されます。

1. アップデート セットアップ ランチャーのデジタル署名がベースのセットアップ ランチャの署名と一致すると、アップデート セットアップ ランチャーが自動的に実行されます。
2. デジタル署名が一致しなかった場合、または、ベース セットアップ ランチャーがデジタル署名されていない場合、エンドユーザーがアップデート セットアップ ランチャーの実行を続けるかどうかを選択できるセキュリティ警告が表示されます。
3. アップデート セットアップ ランチャーにデジタル署名がない場合、インストールは失敗し、デバッグ ログによって、デジタル署名問題が報告されます。

[新しいアップデート利用可能] ダイアログのカスタマイズ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

[新しいアップデート利用可能] ダイアログ ボックスは、アプリケーションのアップデートまたは新しいバージョンがターゲット システムでダウンロード可能になったときに表示される実行時のインターフェイスです。このダイアログ ボックスは通常、**[メディア]** ビューの **[リリース]** エクスプローラの **[アップデート]** タブにある **[アプリのアップデートを確認する]** 領域で指定されたアップデート構成に基づいて、スケジュールどおりにトリガされます。**[新しいアップデート利用可能]** ダイアログ ボックスには、通常、次の内容が表示されます:

- アップデートが利用可能であることを示すメッセージ。
- **[後で通知]**、**[このアップデートをスキップ]**、**[今すぐダウンロード]**などのアクション ボタン。

アドバンスト UI またはスイート/アドバンスト UI インストーラのインストール中に表示される **[新しいアップデート利用可能]** ダイアログ ボックスを設定するには、isupdate.xml ファイルに以下のプロパティを必要な値とともに追加し、更新した XML ファイルを指定の場所に再アップロードします:

- DialogBackgroundColor— このプロパティを使って、**[新しいアップデート利用可能]** ダイアログ ボックスの背景色を指定します。
- DialogHeaderFontSize — このプロパティを使って、**[新しいアップデート利用可能]** ダイアログ ボックスのダイアログ ヘッダーのフォントサイズを定義します。読みやすくするため、28 以上のサイズを定義することが推奨されます。
- DialogBodyFontSize— このプロパティを使って、**[新しいアップデート利用可能]** ダイアログ ボックスのダイアログ テキストおよびボタン テキストのフォント サイズを定義します。読みやすくするため、18 以上のサイズを定義することが推奨されます。
- DialogTextColor— このプロパティを使って、**[新しいアップデート利用可能]** ダイアログ ボックスのダイアログ テキスト、ボタン テキスト、ダイアログ ヘッダーの色を指定します。

色の値は、0xRRGGBBのような 16 進数形式で定義する必要があります:

- 0x は後に続く文字が 16 進数であることを示します。
- RR、GG、および BB は 00 から FF までの範囲で、それぞれ赤、緑、および青を意味します。

  例:

  - 0xFF0000 は赤を意味します。
  - 0x00FF00 は緑を意味します。
  - 0x0000FF は青を意味します。
  - 0xFFFFFF は白を意味します。

- 0x000000 は黒を意味します。

以下の例に示すように、それぞれの値を持つプロパティを isupdate.xml ファイルに追加することができます:

<Update Url="http://www.mysite.com/MyProduct/Setup.exe" Version="1.00.0000" Id="{8E23F36C-5EA8-475D-8EAE-09FA36103975}"DialogBackgroundColor="0xFF0000" DialogHeaderFontSize="38" DialogBodyFontSize="24" DialogTextColor="0xFFFFFF"/>

Windows タスク スケジューラでのアプリケーション更新スケジュール タスクの管理



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI のインストーラが実行されると、通常、ターゲット システム上の Windows タスク スケジューラにスケジュールタスクが作成されます。これらのタスクは、アプリケーションの更新を管理するように設定され、インストールされたアプリケーションが定期的に更新をチェックして適用できるようにします。これらのタスクの頻度と動作は、**[アップデート]** タブの **[アプリのアップデートを確認する]** 領域で定義した設定によって決まります。

ただし、インストール中にアプリケーション更新のスケジュール タスクの作成をスキップしたい場合は、インストール ダイアログに表示されるチェック ボックスを選択できます。このタイプのチェック ボックスでは、最初に、**[ユーザー インターフェイス]** ビューの **[ウィザード インターフェイ スエクスプローラ]** の任意のウィザード ページの下にチェックボックス コントロールを追加し、ISSKIPAPPUPDATES プロパティを使用して構成する必要があります。以下の手順では、プロジェクトに必要なチェックボックス コントロールを追加して構成する方法について説明します。



タスク

インストール中にアプリケーションの更新をスキップするチェックボックス コントロールをプロジェクトに追加して設定するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[プロパティ マネージャー]** をクリックします。
2. **[新しいプロパティ]** ボタンをクリックしてプロパティを作成し、**[名前]** 列に **ISSKIPAPPUPDATES** と入力します。
3. **[値]** 列に、新しいプロパティの値を入力します。デフォルト値は **False** です。
4. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
5. **[ウィザード インターフェイス]** エクスプローラで、**[ウィザード ページ]** ノードを展開し、インストール中にチェックボックスを表示するウィザード ページ(**InstallationWelcome** ウィザード ページなど)の下にチェックボックス・コントロールを追加します。

6. チェックボックス コントロールの **"プロパティ"** 設定を **ISSKIPAPPUPDATES=true** のみに設定します。
7. チェックボックス コントロールの **"テキスト"** 設定を設定して、インストール ダイアログのチェックボックスに必要なテキストを表示します。たとえば、この設定を **[インストール中に更新をスキップする]** に設定できます。
8. チェックボックスコントロールの **"幅"** 設定を必要に応じて設定し、インストール ダイアログで定義済みのテキストを含むチェックボックスを見やすくします。たとえば、この設定を **200** に設定できます。

インストール プロセス中、**"テキスト"** 設定で指定した定義済みテキストのチェックボックスが定義済みインストール ダイアログに表示されます。このチェック ボックスを選択すると、ターゲット システムへのインストール中に Windows タスク スケジューラでアプリケーション更新のスケジュール タスクの作成を妨げるまたはスキップすることができます。逆に、このチェックボックスの選択を解除すると、これらのタスクの作成が許可されます。デフォルトで、このチェック ボックスは選択されていません。

メモ: *ターゲットシステムへのインストール中に Windows タスクス ケジューラでアプリケーション更新のすべてのスケジュール済みタスクの作成をスキップするには、ターゲットシステムでインストーラを実行する前に、ISSKIPAPPUPDATES プロパティを使用してチェックボック スコントロールを構成する必要があります。*

アドバンスト UI またはスイート/アドバンスト UI インストールをパスワードで保護する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、アドバンスト UI またはスイート/アドバンスト UI インストールのセットアップ起動プログラムをパスワードで保護することができます。セットアップ起動プログラムをパスワードで保護すると、インストールを実行するエンド ユーザーは、大文字小文字の区別があるパスワードを入力する必要があります。

タスク

アドバンスト UI またはスイート/アドバンスト UI インストールをパスワードで保護するには、以下の手順に従います:

1. セットアップ起動プログラムにパスワード保護を追加します:
   a. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
   b. **[リリース]** エクスプローラーで、構成するリリースを選択します。
   c. **Setup.exe** タブをクリックします。セットアップ起動プログラムの設定が表示されます。

d. **"ランチャーをパスワードで保護"** 設定で、**[はい]** を選択します。

e. **"ランチャーのパスワード"** 設定に、エンド ユーザーが指定しなくてはならない、大文字と小文字を区別するパスワードを指定します。

2. オプションで、エンド ユーザーがパスワードを入力できる定義済みのウィザード ページを追加することもできます。

a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。

b. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を右クリックしてから、**[定義済みページの追加]** をクリックします。**ユーザー インターフェイス ウィザード**が開きます。.

c. **[タスク ページ]** リストで、**[インストールのパスワードを入力]** を選択します。

d. **[後続ページ]** リストを使って、ウィザード ページ シーケンスでデータベース サーバー ウィザード ページをスケジュールする場所を指定します。実行時に、このリストで選択したページが、新しいパスワード ウィザード ページの直後に表示されます。

メモ・ *定義済みのパスワード ウィザード ページをプロジェクトに追加すると、InstallShield によってプレースホルダー イメージ コントロールが追加され、ページにロック イメージを表示することができます。コントロールの "リソース" 設定を構成してこのコントロールと共に表示するファイルを指定するか、イメージを除外する場合は、イメージ コントロールを削除します。*

デフォルトで、定義済みパスワード ウィザード ページは、ISPassword プロパティを使って、エンド ユーザーが入力したパスワード値を格納します。アドバンスト UI またはスイート/アドバンスト UI インストールは、ISPassword プロパティの値を Setup.xml ファイルのパスワードに格納されているパスワードと比較します。エンド ユーザーが入力したパスワードとプロジェクトで構成されたパスワードとが一致した場合、インストールが次のウィザード ページに進みます。

サイレント アドバンスト UI および スイート/アドバンスト UI インストールの場合、エンド ユーザーは /password コマンドライン パラメーターを使ってコマンドラインからパスワードを指定することができます。詳細については、「アドバンスト UI およびスイート/アドバンストの Setup.exe コマンドラインのパラメーター」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング

プロジェクト・ *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション・ *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロ*

ジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

以下は、アドバンスト UI またはスイート/アドバンスト UI インストールに関する問題を解決するときに役立つ背景情報です。

アドバンスト UI またはスイート/アドバンスト UI インストールを、メンテナンス モードで実行される可能性がある理由

アドバンスト UI またはスイート/アドバンスト UI インストールは、次のシナリオで、メンテナンス モードで実行されることがあります:

- エンドユーザーが、[プログラムの追加と削除] から製品を修正する選択をしたとき。
- エンドユーザーが、アドバンスト UI またはスイート/アドバンスト UI インストールを再実行したとき。
- アドバンスト UI またはスイート/アドバンスト UI インストール内に、検出条件が適切に構成されていないプライマリ パッケージが 1 つ以上存在するとき。アドバンスト UI またはスイート/アドバンスト UI パッケージにおける条件の構成については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」をご覧ください。
- アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージが、依存パッケージではなく、プライマリ パッケージとして間違って構成され、かつ、パッケージが既にターゲット システムにインストールされているとき。

  InstallShield は、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれるすべてのプライマリ パッケージの検出条件に基づいて、メンテナンス モード条件を自動的に作成します。アドバンスト UI またはスイート/アドバンスト UI エンジンは、これらの条件を使って、アドバンスト UI またはスイート/アドバンスト UI インストールがメンテナンス モードで実行するかどうかを判断します。検出条件が True に評価されたプライマリ パッケージが 1 つ以上存在する場合 (つまり、既にインストールされているように見えるプライマリ パッケージが最低 1 つ存在する場合)、メンテナンス モード条件は True として評価され、アドバンスト UI またはスイート/アドバンスト UI インストールはメンテナンス モードで実行されます。

  [パッケージ] ビューの "パッケージの種類" 設定を利用して、パッケージがプライマリ パッケージか依存パッケージかを指定できます。

  モード条件に関する詳しい情報は、「アドバンスト UI またはスイート/アドバンスト UI インストールのインストール モードまたはメンテナンス モードをトリガする」をご覧ください。

トラブルシューティングのためのアドバンスト UI またはスイート/アドバンスト UI インストールのログ

アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティングは、まずデバッグ ログ ファイルの検証から始めます。デバッグ ログ ファイルでは、実行時にアドバンスト UI またはスイート/アドバンスト UI インストールの操作がどのように展開されたかについての詳しい情報が提供されています。デバッグ ログ ファイルを作成するには、/debuglog コマンドライン パラメーターを使って、コマンドラインからアドバンスト UI またはスイート/アドバンスト UI インストールを実行します。Setup.exe ファイルと同じディレクトリで InstallShield.log という名前のログ ファイルを生成するには、コマンドライン パラメーターのみを渡します。Setup.exe ファイルが読み取り専用の場所にあるとき、この処理は実行できません。例:

```
Setup.exe /debuglog
```

ログ ファイルの名前と場所を指定するには、パスと名前を次の要領で渡します。

```
Setup.exe /debuglog"C:¥PathToLog¥setupexe.log"
```

アドバンスト UI またはスイート/アドバンスト UI のデバッグ ログ ファイルには、次の情報が含まれています:

- アドバンスト UI またはスイート/アドバンスト UI エンジンからの初期化情報
- プロパティの値とプロパティの値の変更
- 機能のアクションとインストール状態に関する情報
- パッケージの検出状態と適格性に関する情報
- パッケージのコマンドラインと起動の結果情報
- InstallScript パッケージの場合—InstallScript 関数 SuiteLogInfo によってログ記録された情報

アドバンスト UI またはスイート/アドバンスト UI プロジェクトのデフォルトのアドバンスト UI またはスイート/アドバンスト UI のユーザー インターフェイスでも、情報はデバッグ ログ ファイルにログされますが、一般的に、これらの情報は無視することができます。(ログの UI 情報は通常 UI DLL: というプレフィックスがついています。)

機能、パッケージの状態、および、アドバンスト UI またはスイート/アドバンスト UI のモード情報がログ ファイルに書き込まれるとき、状態の数値のみ書き込まれます。次のテーブルは、数値とそれが表す状態の説明です。

テーブル 6 ・ 機能の操作ステータス

ログの値	状態
0	機能に対して発生する操作はありません。
1	機能がインストールされます。
2	機能が削除されます。

テーブル 7 ・ 機能のインストール状態

ログの値	状態
0	機能は現在インストールされていません。
3	機能はインストールされています。

テーブル 8 ・ パッケージ状態

ログの値	インストール状態	操作状態
0	パッケージはインストールされていません。	—
1	パッケージは現在インストールされています。	インストール操作が実行されます。
2	—	変更操作が実行されます。

テーブル 8 • パッケージ状態 (続き.)

ログの値	インストール状態	操作状態
3	—	修復操作が実行されます。
4	—	削除操作が実行されます。
5	—	パッケージに対して発生する操作はありません。

テーブル 9 • アドバンスト UI またはスイート/アドバンスト UI インストールのモード

ログの値	インストール モード
0	初回インストール
1	メンテナンス/UI メンテナンス モードの選択肢
2	メンテナンス/変更
3	メンテナンス/削除
4	メンテナンス/修復
5	ステージングのみ (インストールは、アドバンスト UI またはスイート/アドバンスト UI インストールがステージングされる、または圧縮解除および特定の場所にコピーされる、ステージ モードで実行中。インストール内のいずれのパッケージも、このモードでは実行しません。)

デバッグ方法 (例)

デバッグ ログ ファイルは、アドバンスト UI またはスイート/アドバンスト UI インストールで予期されなかった動作が認められた様々な異なる状況で役に立ちます。たとえばここで、スイート/アドバンスト UI プロジェクトに、.exe ファイルと .msi ファイルという 2 つのパッケージが含まれていると仮定します。スイート/アドバンスト UI インストールでは、それぞれのファイルに対して、それ自身の機能が関連付けられています。あるマシンで実行中に、両方の機能が選択されているにもかかわらず、.exe パッケージのみインストールされます。この動作のトラブルシューティングを開始するには、まず、この問題が発生しているマシン上でデバッグ ログ ファイルを生成します。ログ ファイルが生成されたら、両方の機能が実際にインストール用に選択されていたかどうかを検証します。ログ ファイルに、各機能について次のような情報が記録されています:

9-21-2011[03:07:04 PM]: Engine: determining suite feature states

9-21-2011[03:07:04 PM]: Initializing state for feature 'NewFeature'

9-21-2011[03:07:04 PM]: Default action state 1 for mode 0

9-21-2011[03:07:04 PM]: Initial feature state: 1

9-21-2011[03:07:04 PM]: Final feature state: 1

...

9-21-2011[03:07:04 PM]: Initializing state for feature 'NewFeature1'

9-21-2011[03:07:04 PM]: Default action state 1 for mode 0

9-21-2011[03:07:04 PM]: Initial feature state: 1

9-21-2011[03:07:04 PM]: Final feature state: 1

上記の情報では、機能 NewFeature と NewFeature1 の初期状態が示されています。両方の機能とも初期状態は、インストールが選択されたことを示す 1 であったことが分かります。従って、スイート/アドバンスト UI インストール内の機能は、問題であるようには見えません。さらに続けてログを見ていくと、次の情報が提供されています:

9-21-2011[03:07:10 PM]: Engine: setting parcel states as determined by feature selections

9-21-2011[03:07:10 PM]: Feature NewFeature setting parcel states, parent override: no, override state: 0

9-21-2011[03:07:10 PM]: Requesting action state 1 for parcel '{BA3EAE7A-0701-4CE0-9224-4F5C0F135792}'

9-21-2011[03:07:10 PM]: Containing feature is request state change to parcel {BA3EAE7A-0701-4CE0-9224-4F5C0F135792}, feature request: 1

スイート/アドバンスト UI インストールのこの時点までで、UI DLL は、UI 選択段階が完了し、インストールを開始するために必要な情報がすべて収集しています。ここで、スイート/アドバンスト UI エンジンは、関連付けられている機能を基に、インストールするパッケージ (ログ ファイルでは*パーセル*という言葉を使用しています) を判別します。この情報は、{BA3EAE7A-0701-4CE0-9224-4F5C0F135792} というパッケージが (この GUID は、アドバンスト UI またはスイート/アドバンスト UI プロジェクト内の各パッケージで設定された "パッケージ GUID" 設定と同じ値です)、NewFeature (インストールが選択されている機能) という機能からインストール (action state 1) するように要求されていることを示しています。さらに見ていくと、次の情報が提供されています:

9-21-2011[03:07:10 PM]: Parcel is ineligible or the current parcel state and install mode to not allow parcel configuration

この行では、パッケージ (期待通りにインストールされない .msi パッケージ) が非対象になってなっていることが分かります。アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [パッケージ] ビューで、提供されたパッケージ ID でこのパッケージを探すと、パッケージに対象条件が設定されています。条件は、パッケージは、特定の .msi 製品コードとパッケージ コードを持つパッケージがインストールされている場合、または、特定の製品コードがある .msi パッケージがインストールいる場合、および、製品バージョンが、.msi パッケージの製品バージョンより大きくない場合に対象となるように設定されています。この条件は、より新しいバージョンが既にインストールされている場合、パッケージがインストールされるのを防ぐ目的で設定されています。

この動作が発生したコンピューターをさらに検証すると、マシン上で新しいバージョンが既に存在し、これが原因で、スイート/アドバンスト UI インストール内の .msi パッケージが対象外なったことが分かります。

スイート/アドバンスト UI インストールにおける InstallScript アクションのデバッグ

スイート/アドバンスト UI インストールで InstallScript アクション の問題をトラブルシューティングするには、InstallScript デバッガーが利用できます。スイート/アドバンスト UI インストールの InstallScript アクションをデバッグするには、コマンドラインから /d パラメーターを使ってスイート/アドバンスト UI インストールを実行します。



メモ: *InstallScript コードをデバッグするには、デバッグ情報ファイルの* Setup.dbg *が使用可能である必要があります。また、開発マシン以外のシステムでインストールをデバッグする場合、特定のファイルを開発マシンからデ*

バッグ マシンにコピーする必要があります。詳細については、「任意のコンピューターでのインストールのデバッグ」を参照してください。

InstallScript デバッガーで次のコマンドを使って、InstallScript カスタム アクションを実行します:

Setup.exe /d

インストールが元のコンパイル場所であるパスにある場合、デバッグ シンボル ファイル (.dbg) へのパスも指定する必要があります:

Setup.exe /d"Path-to-Setup.dbg"

アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート



***プロジェクト**- この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*



***エディション**- アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

/log コマンドライン パラメーターを使って、アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動する場合、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージ設定の構成で、ログ ファイルを生成するかどうかを指定できます。パッケージの種類に応じて、提供されている設定を構成して、その他の情報 (ログ ファイルが作成されるときにパッケージに渡すログ オプションなど) も指定できます。

パッケージにログ機能がサポートされていない場合、そのパッケージに対して、ログ機能を無効にしておくことをお勧めします。

パッケージのログ機能を有効にする



タスク

アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージにログのサポートを有効にするには、以下の手順に従います:

1. ログを有効にするパッケージを含むアドバンスト UI またはスイート/アドバンスト UI プロジェクトを開きます。
2. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。

3. **[パッケージ]** エクスプローラーで、構成するパッケージを選択します。
4. **[共通]** タブにある **"ログの有効化"** 設定で、**[はい]** を選択します。
5. .msi または .msp パッケージで、必要に応じて、次のオプションのサブ設定を構成します:

   - **ログ ファイル**—ログ ファイルの名前を指定します。ファイルのパスを使用しないでください。アドバンスト UI またはスイート/アドバンスト UI の /log コマンドライン パラメーターによって、エンドユーザーは、パッケージのログ ファイル用のディレクトリを指定することができます。

     この設定を空白のままにしておいた場合、インストールで作成されたログ ファイルに、*PackageGUID*.log という名前が使用されます。*PackageGUID* は、[パッケージ] ビューの "パッケージ GUID" 設定でパッケージに割り当てられた GUID です。

   - **ログ オプション**—ログ ファイルを生成するとき、パッケージで使用する Windows Installer ログ /L フラグを指定します。たとえば、すべてを詳細に記録するログ ファイルを作成する場合は、この設定で、次のように入力します:

     *v

     その他の使用可能なフラグについては、/L の説明を参照してください。

     この設定を空白のままにしておくと、アスタリスク (*) と v フラグがログ ファイルの生成時に使用されます。

   .exe パッケージの場合、次のサブ設定を構成します:

   - **ログのコマンドライン**—アドバンスト UI またはスイート/アドバンスト UI インストールで、ログ機能を有効にするために .exe パッケージに渡すコマンドラインを指定します。サポートされている適切なフラグを含めます。構成中の .exe パッケージでログ機能がサポートされている場合、ファイルが作成されるディレクトリのパスの代わりに、アドバンスト UI またはスイート/アドバンスト UI のプロパティ ISLogDir を参照するパスを角かっこで囲んで含めます。

     たとえば、InstallShield でビルドされた Setup.exe ファイルによって実行される .msi パッケージについて、すべてを詳細に記録するログ ファイルを生成する場合、次のコマンドラインを入力します:

     /v"/l*v ¥"[ISLogDir]FileName.log¥""

     アドバンスト UI またはスイート/アドバンスト UI インストールによって [ISLogDir] が、ログ ファイルを含むフォルダーへのパスで置き換えられます。アドバンスト UI またはスイート/アドバンスト UI の /log コマンドライン パラメーターを使用すると、エンドユーザーは、パッケージのログ ファイル用のディレクトリを指定することができます。

アドバンスト UI またはスイート/アドバンスト UI インストールの実行中に、パッケージ ログ ファイルを生成する

パッケージ ログ ファイルを作成するには、/log コマンドライン パラメーターを使って、コマンドラインからアドバンスト UI またはスイート/アドバンスト UI インストールを実行します。/log パラメーターを使用すると、ログが有効されたアドバンスト UI またはスイート/アドバンスト UI インストール内の各パッケージに対して、ログ ファイルが作成されます。

特定のフォルダー内でパッケージ ログ ファイルを生成するには、フォルダーのパスを /log パラメーターと一緒に、アドバンスト UI またはスイート/アドバンスト UI の Setup.exe ファイルに渡します。オプションで、/log パラメーターとパスをコロンで区切ることもできます。たとえば、次のどちらのコマンドラインを使用しても、ログ ファイルは PathToLogFiles フォルダーに作成されます:

Setup.exe /log"C:¥PathToLogFiles"

Setup.exe /log:"C:¥PathToLogFiles"

ログ ファイルの場所へのパスは、事前に存在している必要があります。

パッケージのログ ファイルを *%TEMP%* ディレクトリに生成する場合、パスを空白のままにしておきます。例:

Setup.exe /log

アドバンスト UI またはスイート/アドバンスト UI インストールのプロパティ ISLogDir は、パッケージ ログ ファイルを含むディレクトリへのパスを格納します。

ヒント: *アドバンスト UI またはスイート/アドバンスト UI インストールでは、logging system policy または MsiLogging プロパティを使用して、Windows Installer ベースのパッケージのログ ファイルを生成することもできます。*

64 ビット セットアップ EXE のビルド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShieldでは、アドバンスト UI およびスイート/アドバンスト UI プロジェクトでインストールに 64 ビット **Setup.exe** セットアップランチャーを含めるかどうかを指定できます。64 ビット **Setup.exe** セットアップ ランチャーは、インストールが 64 ビット システムをターゲットとする場合のみ必要です。デフォルトで、InstallShield は 32 ビット システムだけでなく 64 ビットシステム上の WOW64 上で実行可能な 32 ビット セットアップ ランチャーをビルドします。一部の 64 ビット ターゲット システム、たとえば Windows Server Core システム- は、WOW64 (32-bit Windows-on-Windows) をサポートしない場合があります。これらの 64 ビット ターゲット システムは 32 ビット セットアップランチャーを実行できません。

[リリース] ビューにあるリリースの **[Setup.exe]** タブを使って、64 ビット セットアップ ランチャーをビルドするための情報を指定できます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの 64 ビット セットアップ EXE の構成にともない、WOW64 サポートを搭載していないピュア 64 ビット オペレーティング システムをターゲットすることが可能なピュア 64 ビット セットアップ EXE をビルドするためには、64 ビット パッケージおよび 64 ビット アクション イベントを構成する必要があります。ピュア 64 ビット スイート セットアップ EXE は、インストール中に 32 ビットパッケージをインストールまたは 32 ビット アクション イベントを実行することができません。これは、32 ビット バイナリを WOW64 サポートなしで 64 ビット ターゲット システム上にロードすることができないためです。ただし、InstallShield では、オペレーティング システムのアーキテクチャに基づいて選択的にパッケージをインストールするためのプラットフォーム条件を構成することが可能です。この機能によって、単一のスイート セットアップ インストーラーに複数のプラットフォームをターゲットとするパッケージを含めることができます。64 ビット セットアップ ランチャーをビルドするとき、32 ビット DLL、23 ビット マネージ コア、32 ビット .NET Core、および InstallScript アクション イベントはサポートされません。

条件ステートメントの構成方法については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。

64 ビット セットアップ EXE ランチャーを含む 32 ビット パッケージをビルドする

パッケージに 64 ビット セットアップ EXE ランチャー オプションを持つ 32 ビット Windows インストーラー パッケージ (.msi) および 32 実行可能パッケージ (.exe) が含まれている場合、InstallShield でアドバンスト UI およびスイート/アドバンスト UI プロジェクトのビルドを行うと次の警告メッセージが表示されます。

- ISDEV: 警告 -7373: 64 ビット セットアップ (x86 Windows Installer パッケージ - 'パッケージ名' を含む) をビルドしています。
- ISDEV: 警告 -7373: 64 ビット セットアップ (x86 実行可能パッケージ - 'パッケージ名' を含む) をビルドしています。

これは 64 ビット セットアップ ランチャーがピュア 32 ビット システム上での実行に失敗し、32 ビット パッケージがピュア 64 ビット システム上でのインストールに失敗することを通知することが目的です。このエラーを回避するため、インストーラーがピュア 32 ビット システムおよび 64 ビット システム上の WOW64 をターゲットにする必要があるときは、64 ビット セットアップ EXE ランチャーを使用しないでください。または、すべての 32 ビット パッケージをピュア 64 ビット システムをターゲットにする 64 ビット パッケージに移行することも可能です。

64 ビット セットアップ ランチャーを使った InstallScript イベント アクションのビルド

InstallScript イベント アクションは、64 ビット セットアップ EXE ランチャーでサポートされていません。InstallShield では、アドバンスト UI およびスイート/アドバンスト UI プロジェクトに InstallScript イベント アクションが含まれている場合、次のエラーが発生します:

ISDEV: エラー -7371: InstallScript カスタム アクションは、64 ビット セットアップ ランチャーでサポートされていません。

InstallScript イベント アクションが必要な場合、アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは 64 ビット セットアップ ランチャーを使用しないビルドを構成することができます。これは、InstallScript のランタイムが 32 ビット セットアップ EXE ランチャーを必要とする 32 ビットのままであるためです。または、InstallScript イベント アクションが不要な場合、プロジェクトから削除して 64 ビット セットアップ EXE ランチャーのビルドを続けることもできます。

64 ビット セットアップ ランチャーで 32ビット DLL、32 ビット マネージ コード、および 32 ビット.NET Core イベント アクションをビルドする

64 ビット セットアップ EXE ランチャーで 32ビット DLL、32 ビット マネージ コード、および 32 ビット.NET Core イベント アクションはサポートされていません。InstallShield のビルドでは、32 ビット DLL および 32 ビット マネージ コード イベント アクションを含む 64 ビット セットアップ EXE ランチャーのビルド中に次のエラーが発生します。

ISDEV: エラー -7374: 64 ビット セットアップ (x86 DLL アクション 'アクション名') をビルドしています。

ISDEV: エラー -7374: 64 ビット セットアップ (x86 EXE アクション 'アクション名' を含む) をビルドしています。

ISDEV: エラー -7392: x86 .NET Core アクション 'アクション名' を含む 64 ビット セットアップをビルドしています。

ISDEV: エラー -7339: x86 .マネージ コード アクション 'アクション名' を含む 64 ビット セットアップをビルドしています。

このエラーを解決するには、64 ビット セットアップ ランチャーに 64ビット DLL、32 ビット マネージ コード、および 64 ビット.NET Core イベント アクションを含みます。

64 ビット セットアップ ランチャーを使った 32 ビット EXE イベント アクションのビルド

InstallShield では、アドバンスト UI およびスイート/アドバンスト UI プロジェクトで 64 ビット セットアップ EXE ランチャーおよび 32 ビット EXE イベント アクションを含むビルドが構成されている場合、ビルド中に次のエラー メッセージが表示されます。

ISDEV: エラー –7375: 64 ビット セットアップ (x86 EXE アクション 'アクション名' を含む) をビルドしています。

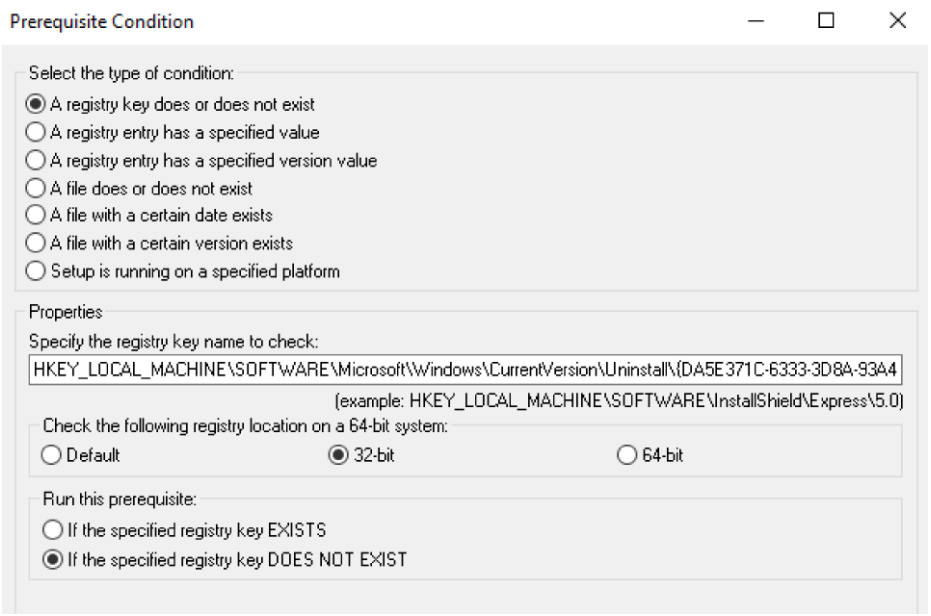
これは 64 ビット セットアップ ランチャーがピュア 32 ビット システム上での実行に失敗し、32 ビット EXE アクション イベントがピュア 64 ビット システム上での実行に失敗することを通知することが目的です。このエラーを回避するため、インストーラーがピュア 32 ビット システムおよび 64 ビット システム上の WOW64 をターゲットにする必要があるときは、64 ビット セットアップ EXE ランチャーを使用しないでください。または、すべての 32 ビット EXE アクション イベントをピュア 64 ビット システムをターゲットにする 64 ビット EXE に移行することも可能です。

InstallShield 前提条件および 64 ビット セットアップ ランチャー

一般的に InstallShield 前提条件を含むプロジェクトでは、Setup.exe によってターゲット システムで InstallShield 前提条件が満たされているかどうかが確認されるため、Setup.exe が必須になります。条件が満たされている場合、Setup.exe が InstallShield 前提条件をインストールします。

64 ビット セットアップ ランチャーを使用する InstallShield 前提条件を含むプロジェクトのビルドを構成する場合、必ず前提条件のインストーラーが 64 ビットであり、ピュア x64 システムにインストールが可能で、前提条件がレジストリまたはフォルダーの場所で 64 ビットの場所を確認するように構成します。

場合によって、64 ビット セットアップ ランチャーを使う 32 ビット前提条件のインストーラーをビルドしたい場合、32 ビットの場所を確認する条件を構成します。前提条件に含まれるほとんどのレジストリ条件はデフォルトの場所を確認するように構成されています (32 ビット セットアップ ランチャーの場合は 32 ビットの場所、64 ビット セットアップ ランチャーの場合は 64 ビットの場所)。ただし、32 ビットの前提条件インストーラーは、32 ビット レジストリハイブの下にレジストリ キーを作成します。この問題を解決するためには、32 ビットの場所を確認するようにレジストリの場所を InstallShield 前提条件エディターで明示的に変更する必要があります。前提条件についての情報は次の通りです:



32 ビット拡張条件 DLL および UI リソース DLL

64 ビット セットアップ ランチャーでは、実行時に 32 ビット拡張条件 DLL および UI リソース 拡張 DLL のロードでエラーが発生します。プロジェクトに 32 ビット拡張 DLLがバインドされている場合、対応する 64 ビット拡張 DLL をビルドして、64 ビット セットアップ ランチャーのビルドを構成する必要があります。

6

InstallShield 前提条件およびその他の再配布可能ファイルをデザインする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript オブジェクト*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield は、InstallShield 前提条件、マージ モジュール、および InstallScript オブジェクトとしてプロジェクトに組み込むことができる多くのサード パーティ再配布可能ファイルを提供します。独自の再配布可能ファイルを作成して、他のインストール開発者に配布することもできます。これらの再配布可能ファイルの種類ごとに、以下のセクションで説明されています。

InstallShield 前提条件



プロジェクト: *次のプロジェクト タイプは、InstallShield 前提条件のサポートを含みます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield には、InstallShield 前提条件を、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに、パッケージとして含めることができるサポートも含まれています。詳細については、「InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める」を参照してください。

InstallShield 前提条件 は、製品が必要とする製品、またはテクノロジ フレームワークのためのインストールです。InstallShield で提供されている InstallShield 前提条件の一例として、Java Runtime Environment (JRE) および SQL Server Express Edition があります。任意の既存の InstallShield 前提条件をインストール プロジェクトに追加して、その設定の多くを構成することができます。また、独自の InstallShield 前提条件を作成して、それをプロジェクトに追加することもできます。



プロジェクト: *Windows Installer ベースのインストールでは、プロジェクトに InstallShield 前提条件を含めると、複数のインストールを連鎖することができるため、1 度に 1 つの実行シーケンスのみしか実行できない Windows Installer 制限をバイパスできます。*Setup.exe *セットアップランチャーは、連鎖を管理するブートストラップ アプリケーションとしての役割を果たします。*



メモ: *Windows Installer 4.5 のチェーンとは異なり、InstallShield 前提条件のインストールは単一のトランザクションとして処理されません。つまり、正常に完了したインストールは、後続の前提条件のインストールが失敗した場合でもロールバックされることはありません。Windows Installer 4.5 のチェーン サポートの詳細については、「トランザクション処理を使って複数パッケージ インストールを構成する」を参照してください。*

独自の InstallShield 前提条件の作成方法についての詳細だけでなく、InstallShield で提供されている InstallShield 前提条件の構成方法については、「InstallShield 前提条件を定義する」を参照してください。

InstallShield 前提条件の種類

基本の MSI、InstallScript、または InstallScript MSI プロジェクトに InstallShield 前提条件を追加すると、デフォルトで*セットアップ前提条件*タイプの InstallShield 前提条件とみなされます。つまり、製品がインストールされる前にターゲット マシン上にインストール済みでなくてはならない、ベース アプリケーションまたはコンポーネントとして処理されます。前提条件がシステム上にインストール済みでない場合、メイン インストールが実行する前にセットアップ前提条件のインストールが実行します。

基本の MSI プロジェクトでは、InstallShield 前提条件をメイン インストールに含まれる機能に関連付けることができます。InstallShield 前提条件が 1 つまたは複数の機能に関連付けられているとき、これは*機能前提条件*と呼ばれます。機能前提条件は、実行時にその前提条件を含む 1 つまたは複数の機能がインストールされるとき、その前提条件がシステム上にインストール済みではない場合にインストールされます。したがって、機能の条件がターゲット システム上で満たされていない場合、またはエンド ユーザーが機能のインストールを行わないことを選択した場合、その機能はインストールされません。その結果、インストールされる別の機能にも機能前提条件が関連付けられていない限り、関連付けられた機能前提条件はいずれもインストールされません。

これら 2 種類の InstallShield 前提条件についての詳細は、「セットアップ前提条件と機能前提条件の違い」を参照してください。

マージ モジュール



プロジェクト: *この情報は、マージ モジュール プロジェクトに適用します。*

InstallShield では、独自の マージ モジュールを作成して、インストール プロジェクトで使用したり、他のインストール開発者が使用できるように配布したりすることができます。

詳細については、「マージ モジュールのデザイン」を参照してください。



メモ: *[再配布可能ファイル] ビューに含まれているマージ モジュールの多くは Microsoft またはその他のサードパーティによるものです。InstallShieldでは、これらのモジュールを無料配布することによって、インストール プロジェクトの作成を支援します。ただし、サードパーティが作成したモジュールに存在する問題を InstallShield が修正したり直すことはできません。サード パーティが作成したモジュールに関する問題は、ベンダーへお問い合わせください。*

InstallScript オブジェクト



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

InstallShield では、独自の InstallScript オブジェクトを作成して、他のインストール プロジェクトで使用したり、他のインストール開発者が使用できるように配布したりすることができます。

詳細については、「InstallScript オブジェクトを作成する」を参照してください。

InstallShield 前提条件を定義する



プロジェクト: *次のプロジェクト タイプは、InstallShield 前提条件のサポートを含みます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

これら全てのプロジェクト タイプは、InstallShield 前提条件タイプのセットアップ前提条件のサポートを含みます。基本の MSI プロジェクトは、機能前提条件タイプのサポートを含みます。

InstallShield には、InstallShield 前提条件を、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに、パッケージとして含めることができるサポートも含まれています。詳細については、「InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める」を参照してください。



メモ: *Windows Package Manger 前提条件を、スイート プロジェクトでインポートすることはできません。*

WPM 前提条件のランタイムは、Windows 10 以降のワークステーションでサポートされています。その他のオペレーティング システムの場合、WPM 前提条件はランタイム中にスキップされます。

InstallShield の InstallShield 前提条件エディターを利用すると、InstallShield に含まれているすべての前提条件の設定を変更することができます。このツールを使ってカスタム前提条件を作成し、プロジェクトに追加することもできます。

独自の InstallShield 前提条件を指定したり既存の前提条件を変更することで、たとえば次のようなオプションの設定が可能になります。

- 前提条件ファイルをターゲットマシンへダウンロードする URL
- 前提条件をインストールする条件
- 特定の前提条件が依存している別の前提条件
- 前提条件のコマンドライン
- インストールがサイレントモードで実行中の場合に使われるコマンドライン
- 前提条件をインストールした後にターゲットマシンを再起動するかどうか
- 前提条件をインストールするために管理者権限が必要かどうか
- エンド ユーザーが前提条件をインストールするかどうかを選択するためのメッセージ ボックスを表示するかどうか
- Windows Package Manager からの前提条件の構成をサポートします。

このセクションには、次のトピックが含まれています:

- InstallShield 前提条件を作成する
- 既存の InstallShield 前提条件を変更する
- InstallShield 前提条件のプロパティを設定する
- nstallShield Windows Package Manager 前提条件の作成

- .prq ファイルの代替 URL を指定する
- InstallShield 前提条件のインストール条件を設定する
- InstallShield 前提条件のファイルを指定する
- InstallShield 前提条件をダウンロードする URL を指定する
- InstallShield 前提条件をインストールするためのパラメーターを指定する
- InstallShield 前提条件のインストールでターゲットマシンを再起動する
- InstallShield 前提条件のインストール動作を指定する
- InstallShield 前提条件の依存関係を指定する
- InstallShield 前提条件を保存する

InstallShield 前提条件を作成する



タスク ***新しい InstallShield 前提条件を作成するには、以下の手順に従います:***

1. **[ツール]** メニューで **[前提条件エディター]** をクリックします。**InstallShield 前提条件エディター**が開きます。
2. 各タブにある設定を適宜構成します。
3. **[ファイル]** メニューで、**[保存]** をクリックします。

既存の InstallShield 前提条件を変更する



タスク ***既存の InstallShield 前提条件を変更するには、以下の手順に従います:***

1. **[ツール]** メニューで **[前提条件エディター]** をクリックします。**InstallShield 前提条件エディター**が開きます。
2. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. InstallShield 前提条件ファイル (.prq) を参照および選択してから、**[開く]** をクリックします。
4. 必要に応じてフォントの変更を行います。
5. **[ファイル]** メニューで、**[保存]** をクリックします。

InstallShield 前提条件のプロパティを設定する

InstallShield 前提条件エディターの [プロパティ] タブでは、InstallShield 前提条件ファイルの説明および一意の ID を指定することができます。

タスク ***InstallShield 前提条件のプロパティを設定するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[プロパティ]** タブをクリックします。
3. **InstallShield 前提条件の固有の ID**ボックスで前提条件ファイルに一意のファイル拡張子を入力するか、デフォルト値をそのまま利用します。これは前提条件アプリケーションの名前、コンポーネント、または GUID 名前、どちらでも構いません。

   メモ・新しい前提条件を作成するために [InstallShield 前提条件エディター] を開けるたびに、新しい GUID が生成され、InstallShield 前提条件の固有の IDボックスにリストされます。

4. **[説明]** ボックスに InstallShield 前提条件の概要を入力します。この説明は **[再配布可能ファイル]** ビューで InstallShield 前提条件ファイルを選択したときに、**概要**タイトルの下に表示されます。

nstallShield Windows Package Manager 前提条件の作成

タスク ***新しい InstallShield Windows Package Manager 前提条件を作成するには、次の手順に従います。***

1. **[ツール]** メニューで **[前提条件エディター]** をクリックします。**InstallShield 前提条件エディター** が開きます。
2. 次の手順に従って、インストールするパッケージの **[Windows Package Manager]** タブの設定を構成します:
   1. **[Windows Package Manager からインストール]** チェックボックスを選択します。
   2. いストールする WPM パッケージの**パッケージ ID** を指定します。
   3. 最新版か、特定のバージョンを指定します。
   4. 追加のパラメーターを指定します。
3. **[ファイル]** メニューで、**[保存]** をクリックします。

.prq ファイルの代替 URL を指定する

次の条件がどちらも true の場合、.prq ファイルに代替 URL を指定することもできます。

- インストールで、1 つまたは複数の InstallShield 前提条件を、Setup.exe ファイルまたはソース メディアに含めず、Web からダウンロードするように指定した。詳細については、「リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する」を参照してください。
- 後日、InstallShield 前提条件のアプリケーション ファイルをダウンロードするための、違う URL にエンドユーザーをリダイレクトする可能性がある。

この場合、代替 URL が、Setup.exe インストールと共に含めた .prq ファイルに指定されます。エンドユーザーがインストールを実行すると、ターゲットマシンは Setup.exe .prq ファイルで指定した代替 URL を参照して、その .prq ファイルをダウンロードし、代替 .prq ファイルで指定した URL ファイルを使用して必要条件アプリケーションに必要なファイルをダウンロードします。



タスク ***.prq ファイルに代替 URL を指定するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[プロパティ]** タブをクリックします。
3. **[前提条件ファイルをダウンロードする場合、.prq をダウンロードするための別のロケーション]** ボックスで、.prq ファイルの代替 URL を入力します。例:

   http://www.mywebsite.com/MyPrq.prq



メモ: *InstallShield 前提条件は、FTP URL をサポートしません。*

InstallShield 前提条件のインストール条件を設定する

ターゲット マシン上に InstallShield 前提条件が既にインストールされているかどうかを判断するインストール条件を指定する必要があります。これを行わなかった場合、ターゲット システムは前提条件が適切にインストールされなかったと見なして作動するため、問題が発生します。また、オペレーティング システム、レジストリ、またはファイル要件を指定するインストール条件を作成することも可能です。InstallShield 前提条件エディターの [条件] タブでインストール条件を設定することができます。

InstallShield 前提条件が (すべてではなく) いくつかのオペレーティング システム上にインストールされる場合、前提条件の複数オペレーティング システム条件を作成することができます。ターゲットマシンが 1 つの独立したオペレーティング システム条件を満たす場合、その前提条件用のオペレーティング システムが満たされます。

実行時に条件が True と評価すると、条件が満たされます。InstallShield 前提条件は、以下が当てはまる場合にターゲット マシンにインストールされます:

- ターゲット マシンがすべてのオペレーティング システム条件、および [条件] タブにリストされているその他すべての条件を満たしている。
- 機能前提条件のみ (つまり、メイン インストールに含まれる 1 つまたは複数の機能に関連付けられている InstallShield 前提条件) ー 機能前提条件を含む機能は、必ずインストールされなくてはなりません。したがって、機能の条件がターゲット システム上で満たされていない場合、またはエンド ユーザーが機能のインストールを行わないことを選択した場合、その機能はインストールされません。その結果、インストールされる別の機能にも機能前提条件が関連付けられていない限り、関連付けられた機能前提条件はいずれもインストールされません。



メモ: *Windows サービス パックなどの前提条件のインストールがサポートされていないため、オペレーティング システム バージョン情報に変更がないものと想定されます。したがって、オペレーティング システム条件は前提条件が正しくインストールされているかどうかを検証する際に考慮されません。*

InstallShield 前提条件の新しいインストール条件を追加する



タスク ***InstallShield 前提条件のインストール条件を新たに追加するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[条件]** タブをクリックします。
3. **[追加]** をクリックします。**[前提条件設定]** ダイアログ ボックスが開きます。
4. 条件の種類によって適切なオプションを選択し、必要に応じて条件のプロパティを設定します。

InstallShield 前提条件エディターが、[条件] タブに条件を追加します。

実行時に条件が True と評価すると、条件が満たされます。InstallShield 前提条件は、以下が当てはまる場合にターゲット マシンにインストールされます:

- ターゲット マシンがすべてのオペレーティング システム条件、および [条件] タブにリストされているその他すべての条件を満たしている。
- 機能前提条件のみ (つまり、メイン インストールに含まれる 1 つまたは複数の機能に関連付けられている InstallShield 前提条件) ー 機能前提条件を含む機能は、必ずインストールされなくてはなりません。したがって、機能の条件がターゲット システム上で満たされていない場合、またはエンド ユーザーが機能のインストールを行わないことを選択した場合、その機能はインストールされません。その結果、インストールされる別の機能にも機能前提条件が関連付けられていない限り、関連付けられた機能前提条件はいずれもインストールされません。

たとえば、いくつかの異なる条件を持つ InstallShield 前提条件を含んでいると想定します:

- ターゲット マシンに特定のファイルが存在しない。この例で、このファイルが不足している場合、製品が必要とするアプリケーションがインストールされることを示します。そうでない場合、製品はインストールされません。
- ターゲットマシンに Windows XP がある。
- ターゲットマシンに Windows Vista がある。

この例では、ターゲット マシンで Windows XP または Windows Vista が実行されている場合で、InstallShield 前提条件がどの機能にも関連付けられていないとき、さらに指定されたファイルが存在しない場合のみ、前提条件がインストールされます。InstallShield 前提条件が、インストールによって実行時にインストールされる機能と関連付けられている場合にも当てはまります。



メモ: *Windows サービス パックなどの前提条件のインストールがサポートされていないため、オペレーティング システム バージョン情報に変更がないものと想定されます。したがって、オペレーティング システム条件は前提条件が正しくインストールされているかどうかを検証する際に考慮されません。*

InstallShield 前提条件のオペレーティング システム条件を追加する

InstallShield 前提条件が (すべてではなく) いくつかのオペレーティング システム上にインストールされる場合、前提条件の複数オペレーティング システム条件を作成することができます。ターゲットマシンが 1 つの独立したオペレーティング システム条件を満たす場合、その前提条件用のオペレーティング システムが満たされます。



タスク ***InstallShield 前提条件に新しいインストール要件を追加するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[条件]** タブをクリックします。
3. **[追加]** ボタンをクリックします。**[前提条件設定]** ダイアログ ボックスが開きます。
4. [セットアップが指定のプラットフォーム上で実行されている] を選択します。
5. 以下のいずれかを実行します。
   - [**前提条件を実行するプラットフォームを選択します**] ボックスで、ターゲット システム上で前提条件に必要なオペレーティング システムを選択する。
   - **[前提条件を実行するプラットフォームを選択します]** ボックスで、**[カスタム]** を選択する。次いで、オペレーティング システムの設定を必要に応じて構成します。

   ヒント・*既にある定義済みのオペレーティング システム条件からの値を使用して、新しいカスタム条件を作成することもできます。そのためには、****[前提条件を実行するプラットフォームを選択します]*** *ボックスで適切な定義済みオペレーティング システムを選択するか、設定を* ***[カスタム]*** *に変更します。****[カスタム]*** *領域で追加の値を指定し、オペレーティング システム条件を続けて構成します。*
6. この前提条件が True 評価されるサービス パックの番号を指定するには、**[サービス パック]** ボックスを使用します。たとえば、ターゲット システムにサービス パック 2、3、または 4 が必要な場合、最初のボックスに 2、2 番目のボックスに 4 を入力します。将来のサービス パックもすべて含めるには、2 番目のボックスを空白のままに残します。サービス パック 3 のみをターゲットとする場合は、両方のボックスに 3 と入力します。
7. **[OK]** をクリックします。

InstallShield 前提条件エディターが、[条件] タブにオペレーティング システム条件を追加します。



メモ・ *InstallShield 前提条件エディターは、****[前提条件を実行するプラットフォームを選択します]*** *ボックスで選択された値を .prq ファイルに格納する代わりに、選択した値に関連付けられている基のオペレーティング システムの設定を格納します。したがって、[カスタム] オプションを選択したあと、[カスタム] 領域で設定の変更または構成を行わなかった場合、選択された条件の [前提条件設定] ダイアログ ボックスが再度開かれたとき、InstallShield 前提条件エディターは、このボックスの値を [カスタム] に設定しません。代わりに、ボックスの値を定義済みのオペレーティング システム オプションに設定します。*

InstallShield 前提条件のインストール条件を変更する



タスク ***既存の条件を変更するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[条件]** タブをクリックします。

3. 変更する条件を選択します。
4. **[変更]** ボタンをクリックします。**[前提条件設定]** ダイアログ ボックスが開きます。
5. 必要に応じて条件を変更します。

InstallShield 前提条件エディターが、[条件] タブの変更済み条件を更新します。

InstallShield 前提条件からインストール条件を削除する



タスク ***InstallShield 前提条件から条件を削除するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[条件]** タブをクリックします。
3. 削除する条件を選択します。
4. **[削除]** をクリックします。

InstallShield 前提条件のファイルを指定する

新しい InstallShield 前提条件を作成する時、前提条件と共に含む必要があるインストール ファイルを指定しなくてはなりません。既存する InstallShield 前提条件に必要なファイルのリストを変更することもできます。InstallShield 前提条件エディターの [含めるファイル] タブでファイルを指定します。



タスク ***InstallShield 前提条件に必要なファイルをひとつ追加するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. [**含めるファイル**] タブをクリックします。
3. **[追加]** をクリックします。**[新規ファイル]** ダイアログ ボックスが開きます。
4. **[ファイル]** ボックスで、追加するファイルの名前とパスを入力します。ファイルを参照して見つけるには、省略記号 (...) ボタンをクリックします。
5. エンドユーザーが Web からファイルをダウンロードすることが可能な場合、URL を **[ファイルへの URL]** ボックスで指定します。この URL は、インストール作成者が **[再配布可能ファイル]** ビューを利用してインターネットからローカル マシンへ InstallShield 前提条件をダウンロードする時に InstallShield が利用する URL と同じです。

   たとえば、そのファイルの URL が **http://www.mywebsite.com/Folder1/Notepad.exe** の場合、次をこのボックスに入力します。

   http://www.mywebsite.com/Folder1



メモ: *InstallShield 前提条件は、FTP URL をサポートしません。*

タスク ***InstallShield 前提条件に必要なファイルを一度に複数追加するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、**[含めるファイル]** タブをクリックします。
2. **[複数ファイルの追加]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. 追加するファイルを指定します。 連続する複数のファイルを選択するには、最初のファイルを選択してから SHIFT キーを押しながら最後のファイルを選択します。連続しない複数のファイルを選択するには、最初のファイルを選択してから CTRL キーを押しながらその他の各ファイルを選択します。
4. **[開く]** をクリックします。

メモ: *この手法でファイルを追加すると、URL(ファイルをダウンロードできる場所)は設定されません。複数ファイルに URL を設定する場合は、「InstallShield 前提条件をダウンロードする URL を指定する」を参照してください。*

タスク ***既存するファイルの設定を変更するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、**[含めるファイル]** タブをクリックします。
2. ファイル名、パス、または URL を変更するファイルを指定します。
3. **[変更]** をクリックします。**[新規ファイル]** ダイアログ ボックスが開きます。
4. 必要に応じて設定を変更します。

タスク ***InstallShield 前提条件からファイルを削除するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、**[含めるファイル]** タブをクリックします。
2. 削除するファイルを選択します。
3. **[削除]** をクリックします。

InstallShield 前提条件をダウンロードする URL を指定する

エンドユーザーはインストールに含まれるセットアップ前提条件に必要なファイルのダウンロードが可能でなくてはなりません。InstallShield 前提条件エディターの [含めるファイル] タブで必ず URL を指定してください。

メモ: *InstallShield 前提条件は、FTP URL をサポートしません。*

タスク ***InstallShield 前提条件に必要なファイル (複数可) のURL を指定するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[含めるファイル]** タブをクリックします。

3. その URL を指定するファイル（複数可）を指定します。連続する複数のファイルを選択するには、最初のファイルを選択してから SHIFT キーを押しながら最後のファイルを選択します。連続しない複数のファイルを選択するには、最初のファイルを選択してから CTRL キーを押しながらその他の各ファイルを選択します。
4. [**ファイルの URL を設定する**] をクリックします。**[ファイル URL の設定]** ダイアログ ボックスが開きます。
5. **[ファイルの親フォルダーへの URL]** ボックスで、選択されたファイルの親ディレクトリの URL を入力します。この URL は、インストール作成者が **[再配布可能ファイル]** ビューを利用してインターネットからローカル マシンへ InstallShield 前提条件をダウンロードする時に InstallShield が利用する URL と同じです。

   たとえば、**Notepad.exe** および **Paint.exe** というファイルを選択して、これらのファイルの URL がそれぞれ **http://www.mywebsite.com/Folder1/Notepad.exe** および **http://www.mywebsite.com/Folder1/Paint.exe** となっている場合、次の通りこのボックスに入力します。

   http://www.mywebsite.com/Folder1
6. **[OK]** をクリックします。



ヒント: *上記手順とは別の方法として、[含めるファイル] タブにある前提条件ファイルのリストに各ファイルを追加する際に URL を指定することも可能です。詳細については、「InstallShield 前提条件のファイルを指定する」を参照してください。*

InstallShield 前提条件をインストールするためのパラメーターを指定する

InstallShield 前提条件エディターの [実行するアプリケーション] タブでは、ターゲット マシン上への前提条件のインストール方法を指定することができます。



タスク ***InstallShield 前提条件をインストールするためのパラメーターを指定するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[実行するアプリケーション]** タブをクリックします。
3. **[起動するアプリケーションを指定してください]** リストで、InstallShield 前提条件がインストールされている場合にターゲット マシンで起動するファイル（通常、Setup.exe セットアップランチャーまたは .msi ファイル）を選択します。**[含めるファイル]** タブで指定したファイルのみがこの一覧に含まれます。

   メイン インストールが基本の MSI インストールまたは InstallScript MSI インストールの場合で、.msi ファイルを指定して、[動作] タブで進行状況の表示を指定した場合、Setup.exe セットアップランチャーは進行状況メッセージをキャプチャして、実行時に MsiExec.exe の代わりに Windows Installer API を使って .msi パッケージを起動します。

   その他の種類のファイルを指定した場合、または進行状況が表示されない .msi ファイルを指定した場合、Setup.exe セットアップランチャーは実行時に Open 動詞（.msi と .exe ファイルの場合）、またはデフォルトの動詞（その他の種類のファイル）のどちらかを使ってファイルを実行します。



*プロジェクト・.msi ファイルを指定し、[動作] タブで進行状況の表示を指定した場合、*Setup.exe *セットアップランチャーは進行状況メッセージをキャプチャして、実行時に* MsiExec.exe *の代わりに Windows Installer API を使って .msi パッケージを起動します。*

その他の種類のファイルを指定した場合、進行状況が表示されない .msi ファイルを指定した場合、またはメイン インストールが InstallScript インストールの場合、Setup.exe セットアップランチャーは実行時に Open 動詞 (.msi と .exe ファイルの場合)、またはデフォルトの動詞 (その他の種類のファイル) のどちらかを使ってファイルを実行します。

4. Windows Installer エンジン、.NET Framework またはその両方をこの InstallShield 前提条件のインストールの前にインストールする必要がある場合、[**Windows Installer エンジンおよび/または .NET Framework を最初にインストールする必要がある**] チェック ボックスを選択します。



***メモ・**このチェック ボックスを選択しても Windows Installer エンジンまたは .NET Framework をインストールに追加しません。これらがインストールに含まれている場合、InstallShield 前提条件がインストールされる前にこれらをインストールしなくてはならないことを指定するだけです。Windows Installer エンジンまたは .NET Framework をインストールに含める場合、それらをプロジェクトに追加する必要があります。詳しくは、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」または「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」をご覧ください。*

5. 必要に応じて、**[アプリケーションのコマンドラインを指定する]** ボックスで **[起動するアプリケーションを指定する]** リストで選択したファイルのコマンドラインを入力します。このボックスにファイル名を含めないでください。

   詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。

6. 必要に応じて、**[セットアップがサイレントモードで実行中にアプリケーションのコマンドラインを指定する]** ボックスに適切なコマンドラインを入力します。このボックスにファイル名を含めないでください。

   詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。

7. 選択された InstallShield 前提条件アプリケーションがインストール後にターゲットマシンの再起動を要求する場合、**[再起動が必要な場合、アプリケーションが戻すコードを指定してください(十進法)]** ボックスに戻りコードを入力します。



***ヒント・**複数のリターンコードが存在する場合、各コードはカンマで区切ってリストしてください。*

- *InstallShield 前提条件として起動するファイルの戻りコードが分からない場合は、ファイルの作成者に問い合わせてください。*
- *再起動が必要な InstallShield 前提条件の詳細は、「InstallShield 前提条件のインストールでターゲットマシンを再起動する」を参照してください。*

InstallShield 前提条件のコマンドライン パラメーターを指定する

InstallShield 前提条件エディターの [実行するアプリケーション] タブでは、ターゲット システム上で InstallShield 前提条件が起動されたときに使用されるコマンドライン パラメーターを指定できます。



タスク

InstallShield 前提条件のインストールが起動されたときに使用するコマンドライン パラメーターを指定するには、以下の手順に従います:

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[実行するアプリケーション]** タブをクリックします。
3. **[アプリケーションのコマンドラインを指定する]** ボックスで、**[起動するアプリケーションを指定する]** リストで選択したファイルの有効な任意のコマンドライン パラメーターを入力します。このボックスにファイル名を含めないでください。
4. メインのインストールがサイレント モードで実行中される場合に、InstallShield 前提条件に使用するコマンドラインを指定するには、**[セットアップがサイレントモードで実行するときのアプリケーションのコマンドラインを指定してください]** ボックスで任意の有効なコマンドライン パラメーターを入力します。このボックスにファイル名を含めないでください。

   この設定に入力されたコマンドライン パラメーターは、その前提条件が表示しないように構成されている場合、つまり [動作] タブで [**前提条件をインストール一覧に表示しない**] チェック ボックスが選択されている場合にも使用されます。



メモ: *InstallShield 前提条件を含むインストールを /s コマンドライン パラメーターを使って起動すると、前提条件のインストールは自動的にサイレントで実行されません。必要に応じて、[実行するアプリケーション] タブの "**セットアップがサイレントモードで実行中にアプリケーションのコマンドラインを指定する**" 設定で、InstallShield 前提条件用の有効なサイレント コマンドライン パラメーターを指定してください。*

*[**セットアップがサイレント モードで実行するときのアプリケーションのコマンドラインを指定してください。**] ボックスを空白のままに残した場合で、次の条件の 1 つまたは両方が True の場合、InstallShield 前提条件にコマンドライン パラメーターは渡されません:*

- *メイン インストールがサイレントで実行されている。*
- *前提条件について、[動作] タブにある [**前提条件をインストール一覧に表示しない**] チェック ボックスが選択されている。*

このため、標準のコマンドライン パラメーターを指定する場合、サイレント コマンドライン パラメーターも指定することが推奨されます。

セットアップランチャーが InstallShield 前提条件にコマンドライン パラメーターを渡す仕組み

InstallShield 前提条件のインストールが .msi パッケージで、[動作] タブで進行状況の表示を選択した場合、セットアップランチャーは MsiExec.exe の代わりに Windows Installer API を使って 、[実行するアプリケーション] タブで指定されたコマンドライン パラメーターで .msi パッケージを起動します。これらの状況下では、コマンドラインに以下の種類のパラメーターをどれでも入力できます:

- 次のフォーマットで表記された Windows Installer プロパティ名と値:

  PROPERTY=VALUE

- /L (/L*v などの任意の変数も使用可能)
- /log
- /q (/qb+! などの任意の変数も使用可能)

- /quiet

進行状況の表示を指定すると、デフォルトで、前提条件の .msi パッケージのユーザー インターフェイスは表示されません。この動作をオーバーライドするには、コマンドライン パラメーターとして /qf を指定できます。

InstallShield 前提条件のインストールがその他の種類のファイルである場合、または進行状況が表示されない .msi パッケージである場合、セットアップランチャーは実行時に Open 動詞 (.msi と .exe ファイルの場合)、またはデフォルトの動詞 (その他の種類のファイル) のどちらかを使ってファイルを実行します。したがtって、進行状況が表示されない .msi パッケージの場合、MsiExec.exe が受け付けるコマンドライン パラメーターであればどれでも使用できます。サポートされているコマンドライン パラメーターのリストは、Windows Installer ヘルプ ライブラリの「Command-Line Options」と「Standard Installer Command-Line Options」を参照してください。

InstallShield 前提条件のインストールが、InstallShield で作成された Setup.exe ファイルの場合、InstallShield 前提条件がプロジェクトの機能と関連付けられているかどうかによって、[実行するアプリケーション] タブで指定した前提条件コマンドライン パラメーターの形式とは異なります。InstallShield 前提条件が機能と関連付けられていない場合、それは *セットアップ前提条件*と呼ばれます。InstallShield 前提条件が 1 つまたは複数の機能と関連付けられている場合、それは *機能前提条件*と呼ばれます。



メモ: *「セットアップ前提条件」と「機能前提条件」の違いについての詳細は、「セットアップ前提条件と機能前提条件の違い」を参照してください。*

セットアップ前提条件に関して、セットアップランチャーは以下のプロパティについてコマンドライン パラメーターを使ったプロパティの置換をサポートします。

- SETUPEXEDIR – セットアップランチャー ファイルへのパスを識別します。たとえば、セットアップランチャーへのパスが C:¥MySetups¥MyApp¥Setup.exe とすると、[SETUPEXEDIR] の値は C:¥MySetups¥MyApp になります。
- SETUPEXENAME – プロジェクトがビルドされるときに作成される、セットアップランチャー ファイルの名前を識別します。セットアップランチャー ファイルの名前が変更された場合、インストールは実行時に、このプロパティ値を更新します。以下のパスは、このファイルへの完全パスを示します:

  [SETUPEXEDIR]¥[SETUPEXENAME]
- ISPREREQDIR – 実行中の InstallShield 前提条件へのパスを識別します。このパスは、末尾のスラッシュを含みます。このプロパティを使って、InstallShield 前提条件内のファイルを参照することができます(例、**[ISPREREQDIR]MyConfigFile.ini**) 。
- ProductLanguage – データベースに書き込まれていない、ユーザー インターフェイスに含まれる任意の文字列用にインストールが使用する言語を識別します。

機能前提条件に関して、[起動するアプリケーション] タブで指定したコマンドライン パラメーターは、実行時にフォーマットされたテキスト フィールドと同様にフォーマットされます。したがって、Setup.exe がサポートする標準コマンドライン パラメーターだけでなく、前述のプロパティもコマンドラインで使用することができます。サポートされているコマンドライン パラメーターの一覧は、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

InstallShield 前提条件のインストールでターゲットマシンを再起動する

InstallShield はいくつかの方法を使って、InstallShield 前提条件の実行後にターゲット マシンの再起動が必要かどうかを決定します。

レジストリキーの変更

InstallShield 前提条件が実行された後に次のレジストリキーのどれかが変更された場合、ターゲットマシンの再起動が必要な可能性があります。

- HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥RunOnce
- HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥RunOnceEx
- HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Control¥Session Manager¥PendingFileRenameOperations

レジストリキーは InstallShield 前提条件が実行された後に考慮されます。これらの数字が異なる場合、ファイルがシステムの再起動を試みると同時にインストールを終了するものと見なされます。

.prq ファイルで指定されたリターンコード

.exe ファイルの exit コードが InstallShield 前提条件 (.prq) ファイルのリターン コードと一致する場合、ターゲットマシンの再起動が必要な可能性があります。.prq ファイルのリターン コードは InstallShield 前提条件エディターの **[再起動が必要な場合、アプリケーションが戻すコードを指定してください]** に入力されます。詳細については、「InstallShield 前提条件をインストールするためのパラメーターを指定する」を参照してください。

リターンコード 1641 および 3010

InstallShield 前提条件用に実行される .msi パッケージに reboot ステートが返された場合、ターゲットマシンの再起動が必要な可能性があります。標準 Windows Installer の再起動リターンコードは以下の通りです。

- ERROR_SUCCESS_REBOOT_INITIATED (1641)
- ERROR_SUCCESS_REBOOT_REQUIRED (3010)

.prq ファイルで指定された動作

InstallShield 前提条件エディターの [動作] タブに表示されるフィールドの 1 つを利用して、前述の条件が特定の InstallShield 前提条件に適合するかどうかによって後に続く動作を指定することができます。詳細については、「再起動を必要とする InstallShield 前提条件の動作を指定する」を参照してください。

InstallShield 前提条件のインストール動作を指定する

新しい InstallShield 前提条件を作成する、または既存の InstallShield 前提条件を変更するとき、エンドユーザーが前提条件のインストールをスキップすることができるかどうかを指定できます。また、ターゲット マシンの再起動の要否に応じて、前提条件のインストールをどのように進めるかを指定することもできます。InstallShield 前提条件エディターの [動作] タブでは、こういった種類の InstallShield 前提条件のインストール動作を指定することができます。

InstallShield 前提条件に管理者権限が必要であることを指定する

Windows Vista 以降の環境では、エンド ユーザーは通常、管理者権限がない標準ユーザーとしてシステムを実行します。InstallShield 前提条件が管理者権限を必要とする場合、Windows Vista は、ユーザー アカウント制御 (UAC) プロンプトを表示して、エンドユーザーに同意 (コンセント) または認証情報を提供するように要求します。

InstallShield 前提条件に管理者権限が必要な場合、または、InstallShield 前提条件をマシンごとにインストールする場合、InstallShield 前提条件エディターの [動作] タブでそれを指定することができます。



タスク ***InstallShield 前提条件をインストールするために管理者権限が必要であると指定するには、次の手順を実行します。***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[動作]** タブをクリックします。
3. **[前提条件は管理者権限を必要とする]** チェック ボックスを選択します。



メモ: *この設定は Windows Vista 以降のシステム上で実行されるインストールに適用します。以前のバージョンの Windows はこの設定を無視します。*

InstallShield 前提条件のインストールの要否をエンド ユーザーが選択できるようにする

InstallShield 前提条件のインストールの要否をエンド ユーザーが選択できるメッセージ ボックスを表示することができます。InstallShield 前提条件エディターの [動作] タブでこれを指定します。



タスク ***エンド ユーザーが InstallShield 前提条件のインストールの要否を選択できるようにするには、次の手順に従います。***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[動作]** タブをクリックします。
3. **"ユーザーは前提条件をオプションでスキップすることができる"**チェック ボックスを選択します。

ターゲットシステムで [インストールされるセットアップ前提条件] リストに、前提条件の名前を含めるかどうかを指定する

ターゲット システムに 1 つまたは複数のセットアップ前提条件のインストールが必要な場合、通常は実行時にメイン インストールが実行する前にセットアップ前提条件ダイアログが表示されます。エンド ユーザーがこのダイアログで [インストール] ボタンをクリックすると、セットアップ前提条件がインストールされます。

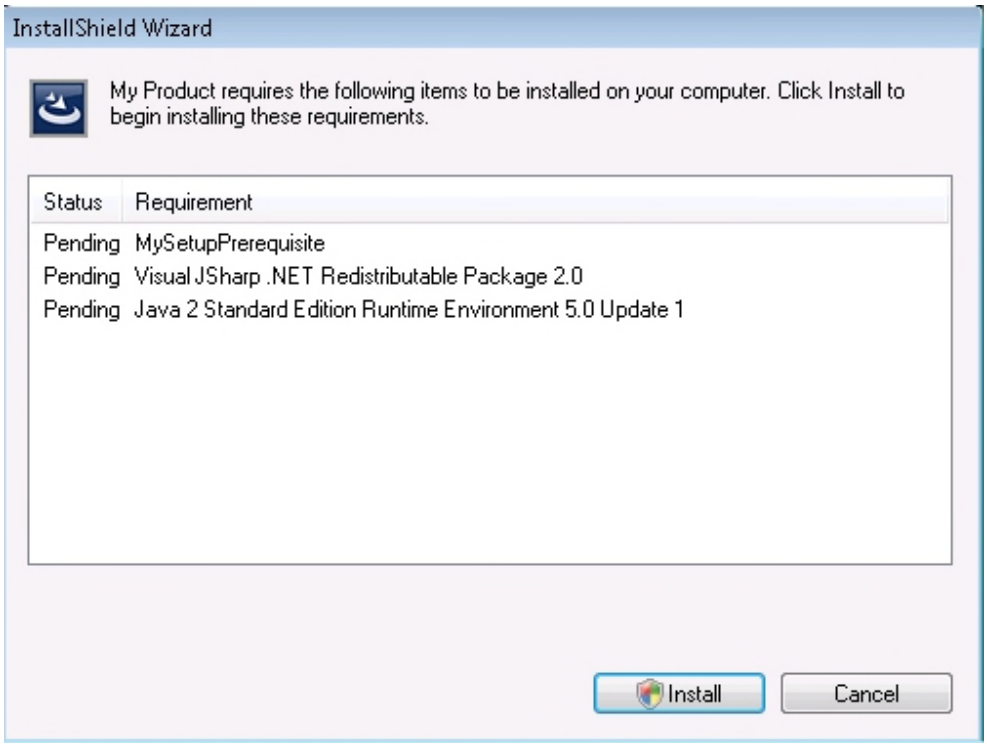


図 1: [インストールするセットアップ前提条件] リストを表示するサンプル [セットアップ前提条件] ダイアログ

InstallShield 前提条件エディターの [動作] タブで、前提条件ダイアログの一覧にセットアップ前提条件を表示しないかどうかを指定できます。前提条件を隠した場合、インストールが必要な前提条件の 1 つとしてリストされていなくても、条件がそれを必要したときに前提条件がインストールされます。

インストールに含まれるすべてのセットアップ前提条件が隠されている場合、インストールは実行時にセットアップ前提条件ダイアログを表示しません。

タスク

実行時に、セットアップ前提条件ダイアログでセットアップ前提条件をリストするかどうかを指定するには、以下の手順に従います:

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[動作]** タブをクリックします。
3. 以下のいずれかを実行します。
   - 前提条件を一覧に表示しない場合、**[前提条件をインストール一覧に表示しない]** チェック ボックスを選択します。

     セットアップ前提条件が非表示の場合、標準のコマンドライン パラメーターではなく、サイレント コマンドライン パラメーターを使って起動されます。詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。
   - セットアップ前提条件を一覧に表示する場合、**[前提条件をインストール一覧に表示しない]** チェック ボックスをクリアします。



メモ- [前提条件をインストール一覧に表示しない] *チェック ボックスが選択またはクリアされているかに関わらず、機能前提条件が実行時にセットアップ前提条件ダイアログに表示されることはありません。つまり、InstallShield 前提条件をプロジェクトに追加して、それを機能に関連付けた場合、実行時、メイン インストールが実行する前に表示されるセットアップ前提条件ダイアログで、機能前提条件は一覧に表示されません。機能前提条件に関する詳細は、「セットアップ前提条件と機能前提条件の違い」を参照してください。*

実行時に、InstallShield 前提条件のインストールの進行状況を表示するかどうかを指定する

前提条件エディターの [動作] タブでは、実行時に Windows Installer によって表示されるインストールの進行状況メッセージと共に、InstallShield 前提条件のインストールがインストールの実際の進行状況を表示するステータス バーを表示するかどうかを指定します。この機能は、前提条件が .msi ファイルを起動する場合のみ使用が可能で、Setup.exe ファイルを起動する場合は使用できません。さらに、InstallShield 前提条件を含むインストールは基本の MSI インストールまたは InstallScript MSI インストールでなくてはなりません。

進行状況が表示されない場合、または InstallShield 前提条件を含むインストールが InstallScript インストールの場合、前提条件のインストールは実行時ダイアログの「**[PrerequisiteName] をインストールしています**」メッセージを表示して、ステータス バーは前提条件のインストールが完了するまでループします。



タスク

InstallShield 前提条件のインストールについて進行状況メッセージとステータス バーを表示するかどうかを指定するには、以下の手順に従います:

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[動作]** タブをクリックします。
3. 以下のいずれかを実行します。
   - 進行状況を表示するには、**[前提条件のウィンドウに進行状況を表示する (未加工の MSI ファイルのみ)]** チェック ボックスを選択します。
   - 進行状況を表示しない場合は、**[前提条件のウィンドウに進行状況を表示する (未加工の MSI ファイルのみ)]** チェック ボックスをクリアします。



メモ- *進行状況の表示を選択した指定した場合、一部のコマンドライン パラメーターしかサポートされません。コマンドライン パラメーターは、InstallShield 前提条件エディターの [実行するアプリケーション] タブで指定されます。詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。*

前提条件の進行状況の表示を指定すると、デフォルトで、前提条件の .msi パッケージのユーザー インターフェイスは表示されませんので、ご注意ください。この動作をオーバーライドするには、InstallShield 前提条件エディターの [実行するアプリケーション] タブで、コマンドライン パラメーターとして /qf を指定できます。

InstallShield 前提条件のインストールに伴う可能性のある問題の対処方法について

InstallShield 前提条件のインストール実行後、引き続き InstallShield 前提条件のインストールが必要であると表示される。原因として、次の理由が考えられます。

- InstallShield 前提条件が正しくインストールされなかった。
- InstallShield 前提条件に不正確な条件が指定されていた。

たとえば、ターゲット マシンに特定のファイルが存在しない場合に InstallShield 前提条件をインストールする必要があると条件付けたとします。InstallShield 前提条件がインストールされた後にもそのファイルが不足している場合、その条件は適切ではなかったと考えられます。InstallShield 前提条件エディターの [動作] タブでは、これらの状況下でインストールをどのように進めるのかを指定することができます。



タスク ***InstallShield 前提条件のインストールに関連して起こりうる問題の対処には、次の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[動作]** タブをクリックします。
3. **前提条件をインストールした後も、その前提条件のインストールがまだ条件で要求された場合** リストで、適切な項目をクリックします。

再起動を必要とする InstallShield 前提条件の動作を指定する

新しい InstallShield 前提条件を作成または既存の InstallShield 前提条件を変更するとき、ターゲット マシンの再起動の要否に従って 前提条件のインストール処理をどのように進めるのかを指定することができます。たとえば、マシンを再起動する前にインストールがエンド ユーザーに対してプロンプトを行う場合、または再起動そのものをスキップする場合が考えられます。InstallShield 前提条件エディターの [動作] タブでは、各 InstallShield 前提条件に適切な応答を行うための柔軟性を提供します。



タスク ***ターゲット マシンの再起動を要求する InstallShield 前提条件の動作を指定するには、次の手順に従います。***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[動作]** タブをクリックします。
3. **前提条件が再起動を必要とする場合**リストで、適切な項目をクリックします。

使用可能なオプションについて詳しい情報は、[動作] タブを参照してください。

InstallShield 前提条件の依存関係を指定する

新しい InstallShield 前提条件を作成する、または既存の InstallShield 前提条件を変更するとき、この InstallShield 前提条件が依存する他の 前提条件 (.prq ファイル) を指定することができます。依存関係は、InstallShield 前提条件エディターの依存関係タブで指定します。

InstallShield 前提条件の依存関係を追加する



タスク ***InstallShield 前提条件の依存関係を追加するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[依存関係]** タブをクリックします。
3. **[追加]** ボタンをクリックします。**[新しい依存関係]** ダイアログ ボックスが開きます。
4. **[ファイル]** ボックスで、現在の InstallShield 前提条件に必要な .prq ファイルの名前とパスを入力します。.prq ファイルを参照して見つけるには、省略記号 (...) ボタンをクリックします。InstallShield 前提条件ファイル (.prq) は、次の場所に格納されています:

   InstallShield Program Files フォルダー¥SetupPrerequisites
5. **[OK]** をクリックします。

InstallShield 前提条件の依存関係を変更する



タスク ***既存の依存関係の設定を変更するには、以下の手順を実行します:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[依存関係]** タブをクリックします。
3. 変更する依存関係を選択します。
4. **[変更]** をクリックします。**[新しい依存関係]** ダイアログ ボックスが開きます。
5. 必要に応じて設定を変更します。

InstallShield 前提条件の依存関係を削除する



タスク ***前提条件から依存関係を削除するには、以下の手順を実行します:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[依存関係]** タブをクリックします。
3. 削除する依存関係を選択します。
4. **[削除]** をクリックします。

InstallShield 前提条件を保存する

新しい InstallShield 前提条件を作成したとき、または InstallShield の InstallShield 前提条件エディターを利用して既存の前提条件を変更したとき、InstallShield 前提条件 (.prq) ファイルを保存しなくてはなりません。



タスク ***InstallShield 前提条件エディターで開いている InstallShield 前提条件を保存するには、以下の手順に従います:***

1. **[ファイル]** メニューで、**[名前を付けて保存]** をクリックします。**[名前を付けて保存]** ダイアログ ボックスが開きます。
2. InstallShield 前提条件ファイルの名前と .prq 拡張子を指定しますInstallShield 前提条件ファイル (.prq) は、次の場所に格納されています:

   InstallShield Program Files フォルダー¥SetupPrerequisites
3. **[保存]** ボタンをクリックします。

SetupPrerequisites に格納されている InstallShield 前提条件は、[再配布可能ファイル] ビューにリストされています。

マージ モジュールのデザイン



プロジェクト: *この情報は、マージ モジュール プロジェクトに適用します。*

マージ モジュールを利用することにより、複数の開発チームが互いに独立したコンポーネントについて作業できるようになります。各チームは、自分たちの作業が他のコンポーネントにどう影響するかを考慮せずに、そのインストールデータベースに値を設定できます。各コンポーネントの開発が完全に終了すると、マージ モジュールデータベースの各コンポーネントを中央の製品インストールデータベースにマージできます。

InstallShield では、独自の マージ モジュールを作成して、インストール プロジェクトで使用したり、他のインストール開発者が使用できるように配布したりすることができます。

マージ モジュールを作成する最も重要なステップは、その設計にあります。上手く設計されたマージ モジュールにするためには、ファイルをプロジェクトの開発者的視点から見た構成による、複数のコンポーネントに分割する必要があります。機能、コンポーネント、ファイルを含む完全なインストールを作成する場合とは違い、マージ モジュールには、コンポーネントのみが含まれます (コンポーネントの中にファイルがあります)。

マージ モジュールの作成には、主に以下のものが関係します。

テーブル 1・マージ モジュール作成の概観

マージ モジュール要素	説明
コンポーネント	アプリケーション データをグループ化できます。コンポーネントは、プロジェクトを開発者側の視点から見た構成であり、ファイル、レジストリ エントリ、およびショートカットなどのデータを含みます。コンポーネント ビューでは、マージ モジュールのコンポーネントを設計します。
ファイル、レジストリデータ およびショートカット	上記で示した階層構造を完成するためにコンポーネントに追加される要素です。
詳細設定	詳細設定を指定して、コンポーネントのパブリッシュ、COM サーバー、ファイル拡張サーバー、および MIME の種類の登録を行なうことができます。コンポーネントの詳細設定を使って、レジストリにアプリケーションパスのエントリを作成することもできます。

このセクションには、次のトピックが含まれています:

- マージ モジュール プロジェクトを作成する
- マージ モジュールのデフォルトのインストール先フォルダーを指定する
- マージ モジュール言語を選択する
- マージ モジュールに依存関係を追加する
- マージ モジュールへ除外を追加する
- マージ モジュールをマージ モジュール プロジェクト内からリポジトリにパブリッシュする

マージ モジュール プロジェクトを作成する



プロジェクト: *この情報は、マージ モジュール プロジェクトに適用します。*

マージ モジュール (.msm ファイル) には、個別機能をインストールするために必要なロジックとファイルのすべてが含まれています。たとえば、多くのアプリケーションには、Visual Basic ランタイム DLL が必要です。コンポーネントにファイルを含めてインストール要件を調べる必要はなく、プロジェクトの機能に Visual Basic 仮想マシン マージ モジュールを添付するだけでこれを実行できます。

マージ モジュールプロパティには、以下の項目が含まれます。

- 製品名
- 製品バージョン
- 言語
- INSTALLDIR

マージ モジュールの依存関係と除外は、[一般情報] ビューにある "モジュールの依存関係" および "モジュールの除外" 設定で指定できます。

マージ モジュールを作成するには、まずマージ モジュール プロジェクトを作成します。



タスク

マージ モジュール プロジェクトを作成するには、以下の手順に従います:

1. **[ファイル]** メニューで、**[新規]** をクリックします。
2. **[Windows Installer]** タブをクリックします。
3. [プロジェクトの種類] ボックスで、**[マージ モジュール プロジェクト]** を選択します。
4. **[プロジェクト名]** ボックスに、プロジェクトの名前を入力します。

   重要 - *新しいマージ モジュール プロジェクトを作成するとき、[新規プロジェクト] ダイアログ ボックスで指定する名前は 35 文字以下でなくてはなりません。これは、入力した名前がマージ モジュールファイルの* ModuleSignature *テーブルにある ModuleID フィールドで GUID と共に利用されるため、また ModuleID フィールドのオブジェクト名部分の最大文字数が 35 文字のためです。*
5. **[場所]** ボックスに、プロジェクトを保存する場所を指定するか、**[参照]** ボタンをクリックして、その場所に移動します。
6. **[OK]** をクリックします。

マージ モジュールのデフォルトのインストール先フォルダーを指定する



プロジェクト: *この情報は、マージ モジュール プロジェクトに適用します。*

[一般情報] ビューの "INSTALLDIR" 設定に入力された値は、マージ モジュールのすべてのファイルのデフォルト フォルダーとなります。したがってその値は、デフォルトのコンポーネントインストール先フォルダーである、Windows Installer フォルダープロパティ **INSTALLDIR** に割り当てられます。

このマージ モジュールのユーザーが、デフォルトのインストール先ディレクトリを上書きできるようにするには、"INSTALLDIR" 設定に **[TARGETDIR]** を入力します。詳細については、「マージ モジュールのインストール先をオーバーライドする」を参照してください。

インストール先フォルダーに関するその他の考慮事項

マージ モジュール プロジェクトに含まれる各コンポーネントにも "インストール先" 設定があります。コンポーネントの "インストール先" 設定は、[一般情報] ビューの "INSTALLDIR" 設定をオーバーライドします。したがって、マージ モジュールの全ファイルがデフォルトでマージ モジュールのインストール先フォルダーにインストールされるようにする場合、すべてのコンポーネントの "インストール先" 設定に [INSTALLDIR] を入力します。

マージ モジュール言語を選択する



***プロジェクト・** この情報は、マージ モジュール プロジェクトに適用します。*

[一般情報] ビューの "言語" 設定では、マージ モジュールの言語を選択できます。たとえば、モジュールでユーザー インターフェイスをインストールする場合、そのユーザー インターフェイスを作成する言語を指定します。モジュール言語の設定と一致しないマシン上でモジュールを実行しても、モジュールはインストールされません。



***ヒント・** すべての言語でモジュールを実行できるようにするには、言語非依存を選択します。*

マージ モジュールに依存関係を追加する



***プロジェクト・** この情報は、マージ モジュール プロジェクトに適用します。*

マージ モジュールのなかには、それ自体で独立したものもありますが、正しく機能するために他のマージ モジュールを必要とするものもあります。たとえば、マージ モジュールには Visual Basic ランタイム ライブラリが必要なことがあります。その場合、そのモジュールを依存関係として新しいマージ モジュールに追加することが必要になります。

ファイルをマージ モジュールに追加する場合、そのファイルがシステムに格納された別のマージ モジュールに含まれていれば、セットアップベストプラクティス違反となります。この場合、マージ モジュールを依存関係として追加する必要があります。



***注意・** 典型的な基本の MSI プロジェクトまたは InstallScript MSI プロジェクトを作成中、マージ モジュール内で利用可能なファイルを追加すると、自動的にマージ モジュールがプロジェクトに追加されます。マージ モジュールの作成中に別のマージ モジュール内で利用可能なファイルを追加すると、そのマージ モジュールを依存関係として設定するように注意を促すセットアップ ベストプラクティス アラートが表示されます。InstallShield はマージ モジュール プロジェクトに依存関係を作成しません。*

[一般情報] ビューに入力した依存関係情報は、プロジェクトの ModuleDependency テーブルに書き込まれます。このテーブルは、ダイレクト エディターを使って参照できます。

再配布可能ファイル ギャラリーからの依存関係の追加

マージ モジュールが [オプション] ダイアログ ボックスの [マージ モジュール] タブで指定されたパスのいずれかに存在する場合、それは再配布可能ファイルギャラリー内にあります。



タスク ***再配布可能ファイル ギャラリーから 1 つ以上の依存関係を追加するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"モジュールの依存関係"** 設定で、省略記号ボタン (...) をクリックします。**[モジュールの依存関係]** ダイアログ ボックスが開きます。
3. 作成中のマージ モジュールが必要とするマージ モジュールのチェック ボックスを選択します。
4. **[OK]** をクリックします。

InstallShield が [一般情報] ビューのグリッドへ依存関係を追加します。必要に応じて、依存関係の設定を変更できます。

再配布可能ファイル ギャラリーに存在しない依存関係を追加する



タスク ***マシンにない依存関係を追加するには、次の手順を実行します。***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"モジュールの依存関係"** 設定で、**[新しいモジュールの依存関係を追加]** ボタンをクリックします。

InstallShield が [一般情報] ビューのグリッドへ依存関係を追加します。必要に応じて、各設定を構成します。



メモ: *"モジュール ID" 設定を入力するとき、InstallShield はそれが正しいかどうかをチェックしませんので、ご注意ください。インストールされたプログラムを実行するまで、失敗するかどうかわかりません。*

マージ モジュールへ除外を追加する



プロジェクト: *この情報は、マージ モジュール プロジェクトに適用します。*

ときどき、作成中のモジュールに関連付けられているモジュールから、特定のマージ モジュールを除外することが必要なことがあります。旧バージョンのモジュールが新しいモジュールと互換性がないためにこのような作業が必要になることがあります。結果的に、新しいモジュールが関連するどれかのセットアップからそのモジュールを除外する必要が出てきます。

再配布可能ファイル ギャラリーからの除外を追加する

マージ モジュールが [オプション] ダイアログ ボックスの [マージ モジュール] タブで指定されたパスのいずれかに存在する場合、それは再配布可能ファイルギャラリー内にあります。



タスク ***再配布可能ファイル ギャラリーからの除外を追加するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"モジュールの除外"** 設定で、省略記号ボタン (...) をクリックします。**[モジュールの除外]** ダイアログ ボックスが開きます。
3. 作成中のマージ モジュールと互換性のないマージ モジュールのチェック ボックスを選択します。
4. **[OK]** をクリックします。

InstallShield が [一般情報] ビューのグリッドへ除外を追加します。必要に応じて、除外の設定を変更できます。

再配布可能ファイル ギャラリーに存在しない新しい除外を追加する



タスク ***マシンに存在しない除外を追加するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"モジュールの除外"** 設定で、**[新しいモジュールの除外を追加]** ボタンをクリックします。

InstallShield が [一般情報] ビューのグリッドへ除外を追加します。必要に応じて、各設定を構成します。



メモ: *"モジュール ID" 設定を入力するとき、InstallShield はそれが正しいかどうかをチェックしませんので、ご注意ください。インストールされたプログラムを実行するまで、失敗するかどうかわかりません。*

マージ モジュールをマージ モジュール プロジェクト内からリポジトリにパブリッシュする



プロジェクト: *この情報は、マージ モジュール プロジェクトに適用します。*

Windows Installer ベースのインストールで再利用可能な、または他のユーザーと共有可能なマージ モジュールを設計中の場合、それをリポジトリへパブリッシュすることができます。InstallShield では、ビルドの度にマージ モジュールがリポジトリへパブリッシュされるよう、リリースを構成することが可能です。



タスク ***マージ モジュール プロジェクト内部から、マージ モジュールをリポジトリへパブリッシュするよう設定するには、次の手順に従ってください。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、設定を行うリリースを選択します。

3. **[イベント]** タブをクリックします。
4. **"マージ モジュールのパブリッシュ"** 設定で、**[リポジトリにパブリッシュ]** オプションを選択します。
5. **"リポジトリ名"** 設定で、適切なリポジトリを選択します。アクセス可能なすべてのリポジトリは、この設定にリストされます。
6. **"表示名"** 設定に、リポジトリにパブリッシュするマージ モジュールの名前を指定します。この名前は、[再配布可能ファイル] ビューでマージ モジュールの名前として表示されます。
7. **"発行者"** 設定に、発行者の名前を入力します。
8. **"説明"** 設定に、マージ モジュールの説明を入力します。

選択したリリースをビルドすると常に、InstallShield によって指定されたリポジトリへマージ モジュールがパブリッシュされます。

ビルド時にマージ モジュールがインストールへ組み込まれます。リポジトリ マージ モジュールに変更を加えてからリポジトリへ再パブリッシュすると、プロジェクトのインストールが次回再ビルドされたときに、古いバージョンのリポジトリマージ モジュールを持つプロジェクトはすべて更新されます。

InstallScript オブジェクトを作成する

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

InstallShield では、独自の InstallShield オブジェクトを作成して、他のインストール プロジェクトで使用したり、他のインストール開発者が使用できるように配布したりすることができます。オブジェクトによって、InstallShield の以前のバージョンにあったテンプレートが置き換えられます。

メモ: *作成するオブジェクトは、InstallShield X 以降、InstallShield DevStudio、または InstallShield Professional 6.1 以降でのみ利用することができます。*

オブジェクトの作成は、標準インストール プロジェクトの作成と多少異なります。唯一の違いはスコープの重要性です。オブジェクトを作成するとき、オブジェクトのスコープがその利用目的に対して最適な状態にして下さい。オブジェクトのスコープはその有用性に影響します。スコープが狭すぎるとオブジェクトがニーズを十分に満たさないものになることがあります。スコープが広すぎるとオブジェクトが厄介で実用性に欠けるものになることがあります。

次のセクションでは、オブジェクトの作成プロセスの手順が説明されています。

オブジェクトの作成

現在、オブジェクトの作成方法には 2 通りあります。[新規のプロジェクト] ダイアログ ボックスから新規の InstallScript Object プロジェクトを作成するか、既存の InstallScript プロジェクトを InstallScript オブジェクトに変換することができます。

タスク

InstallScript プロジェクトを InstallScript オブジェクトに変換するには、以下の手順を実行します。

1. **[ファイル]** メニューで、**[名前を付けて保存]** をクリックします。**[名前を付けて保存]** ダイアログ ボックスが開きます。
2. プロジェクトのコピーの作成先となるフォルダーを選択します。新しいオブジェクトを元のプロジェクトと同じフォルダーに保存することはできません。
3. **[ファイル名]** ボックスで、新規プロジェクトの名前を入力します。
4. **[ファイルの種類]** リストで、**[InstallScript オブジェクト プロジェクト (*.ism)]** を選択します。
5. **[保存]** をクリックします。

指定した名前を持つオブジェクト プロジェクト ファイル (..ism ファイル) とすべてのプロジェクトのサブフォルダーが指定の場所に作成されます。プロジェクトが InstallShield で開きます。

オブジェクトのデザイン

オブジェクトの目的は、別個の機能をインストールするためです。たとえば、アプリケーションを実行するためには、最新バージョンの DirectX をインストールする必要がある場合があります。必要なすべてのファイルを個別にインストール プロジェクトに含めるかわりに、InstallShield に付属の DirectX オブジェクトを含めることができます。この同じオブジェクトに、Visual Basic ランタイム .dll ファイルや、MDAC サポートが含まれていると想定してください。オブジェクトが、ニーズに対して大きすぎます。必要のないファイルをインストールし、インス

トール プロセスの所要時間がかかるだけです。したがって、オブジェクトをできるだけ合理化する必要があります。4 つの異なるテクノロジを1つのオブジェクトに含めないでください。4 つの別のプロジェクトを作成するようにしてください。

オブジェクトは従来のインストールと同じ構造をとりますが、オブジェクトはセットアップの種類をサポートしない点で、大きく異なります。オブジェクトの構成の最高レベルにあるのは機能です。機能の下にはコンポーネントが続きます。さらに、オブジェクトはシェル オブジェクトもサポートしていません。オブジェクトからシェルオブジェクトを作成する場合は、スクリプトから行う必要があります。

詳細については、「オブジェクトの設計」を参照してください。

オブジェクトのビルド

オブジェクトのビルドは、インストール プロジェクトのビルドに非常によく似ています。この作業を完了する最も簡単な方法は、リリース ウィザードを使うことです。オブジェクトのビルドとインストール プロジェクトのビルドの主な違いは、オブジェクトのメディアの種類を指定する必要がないことです。ウィザードでは、CD-ROM 形式のメディアのみ作成できます。

詳細については、「ユーザー インターフェイスからオブジェクトをビルドする」を参照してください。

オブジェクトのテスト

どのようなソフトウェア作業においても、一般にリリースする前にソフトウェアをテストすることが重要です。オブジェクトのテストは簡単です。標準インストール プロジェクトを作成してから、オブジェクトを追加し、ビルドして実行します。すべてが予定通りに実行されると、次に、オブジェクトをプロジェクトに含めます。予定通りに行かなければ、デバッグする必要があります。「オブジェクトのテストとデバッグ」を参照してください。(オブジェクトプロジェクトがあり、関連したインストール プロジェクトを InstallShieldの別のインスタンスで開いて、オブジェクトを変更および再ビルドした場合、それらの変更を統合するには関連したインストール プロジェクトを閉じてから再度開く必要があります。)

オブジェクトの配布

ここで使用されている*オブジェクトの配布*とは、そのオブジェクトをインストールに含めて、ターゲットマシンでそのオブジェクトからファイルをインストールすることではありません。ここで言う*オブジェクトの配布*とは、オブジェクトを他のユーザーのマシンにおいてインストール プロジェクトで使用できるようにするインストールを作成することを意味します。オブジェクト内に含まれているロジックはインストールしたくないので、通常行うようにロジックをコンポーネントにただ追加することはできません。オブジェクトを簡単に登録するには、InstallShield オブジェクト インストーラー オブジェクトを使用することができます。このオブジェクトを機能と関連付ける際、配布するローカル ギャラリーからオブジェクトを選択するよう求めるプロンプトが表示されます。

詳細については、「オブジェクトの配布」を参照してください。

オブジェクトの設計



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトのデザインとはコンポーネントとファイル グループをいかに配置するかを意味し、オブジェクトの実際の内容を扱うこの作業は、安定した使用可能なオブジェクトを実現するために克服しなくてはならない最大の難関です。

オブジェクトのデザインには、いくつかの重要点があります。それぞれを以下で説明します。

オブジェクトのプロジェクト設定

標準のインストール プロジェクトとオブジェクトの違いは、そのプロジェクト設定にあります。オブジェクト作成用のタブは、[プロジェクトの設定] ダイアログ ボックスで利用可能です。これらのタブで、オブジェクトの開発言語、表示名、オブジェクトに関連付けられているアイコンとヘルプトピックを定義することができます。また、作成するオブジェクトに、好みのウィザード インターフェイスを選択することみできます。

詳細については、「オブジェクトのウィザードの設計」を参照してください。

コンポーネントの管理

プロジェクトは、時間をかけて適切に各機能に分割することが重要です。機能を設計する場合、機能のローカリゼーションや条件選択などの状況を考慮に入れる必要があります。詳細については、「オブジェクト用機能の設計」を参照してください。

プロパティとメソッド

プロパティとメソッドを使うと、デザイン時と実行時に、ユーザーがオブジェクトと相互作用することができます。詳細については、「プロパティの作成」および「メソッドの作成」を参照してください。

デザインタイム ウィザードの作成

オブジェクトにプロパティを含める場合、オブジェクトのユーザーが必要に応じてプロパティを簡単に設定できるように、グラフィカル インターフェイスを提供すると良いでしょう。詳細については、「オブジェクトのウィザードの設計」を参照してください。

ランタイム ユーザー インターフェイスの作成

作成したオブジェクトで、このオブジェクトを含むインストールのエンドユーザーにダイアログを表示できます。オブジェクトのランタイム ユーザー インターフェイスの作成と、オブジェクトを含むインストールからのインターフェイスの呼び出しについての詳細は、「オブジェクトの実行時のユーザー インターフェイスを作成する」を参照してください。

オブジェクトのデザイン環境をローカライズする

InstallShield インターフェイスは英語と日本語にローカライズされています。オブジェクトをこれらの言語にローカライズする方法についての詳細は、「オブジェクトのデザインタイム環境のローカライズ」を参照してください。

オブジェクトのランタイム動作をインターナショナル化する

アプリケーションを複数言語で実行するようにしている場合は、インターナショナル化されたオブジェクトを準備する必要があります。言語固有のファイルをオブジェクトと一緒にインストールすることは、通常のインストールとは若干異なります。詳細については、「インターナショナル化のためのオブジェクトの作成」を参照してください。

実行時に、オブジェクトとインストールとの互換性を持たせる

オブジェクトはオブジェクトを含むインストールによってサポートされている、すべてのオペレーティング システムと言語をサポートしています。詳細については、「実行時に、オブジェクトとインストールとの互換性を持たせる」を参照してください。

オブジェクト用機能の設計



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

機能のデザインの良し悪しは、作業全体に影響するため、注意が必要です。機能のデザインに落ち度がある場合、オブジェクトをインターナショナル化したり、ファイルを選択してオブジェクトにビルドしたり、または条件を付けてファイルをインストールするなどの作業をより困難なものにします。このセクションでは、オブジェクトの機能レイアウトを正確に効率良くデザインするためのガイドラインを示します。

オブジェクトの機能を選択してインストール

通常、オブジェクトの機能の選択について条件を設定します。たとえば、DCOM をインストールするオブジェクトを作成する場合、まず、そのターゲット システムがすでに DCOM をインストールしているかどうかを確認する必要があります。DCOM がすでにインストールされている場合、新たにオブジェクトを作成する作業は時間の無駄であり、また面倒な問題を起こす可能性があります。オブジェクトの機能を、オブジェクトを含むプロジェクトに悪影響を与えずに、条件を付けて選択するには、次のガイドラインに従ってオブジェクトを設計する必要があります。

オブジェクトはその機能を選択しませんが、必要な場合に機能の選択を解除します。(オブジェクトの機能はデフォルトで選択されます。)使用するオブジェクトが 1 つ、または複数の機能を選択する場合、選択する機能がすでにエンドユーザーやプロジェクトの内部ロジックによって選択を解除されていても、そのオブジェクトを含む機能が選択されます。

しかし、オブジェクトによる機能の選択解除は、ある問題を引き起こす可能性もあります。下記にその問題と解決法について示します。

プロジェクトに2つの主機能があるとします(機能 A と B)。B には、サブ機能はありませんが、機能 C をもつオブジェクトがあります。InstallScript エンジンは、子機能が選択されていない時、自動的に機能を選択解除します。したがって、機能 C が選択されていない時、DCOM オブジェクトは選択されません。DCOM オブジェクトが選択されていない場合、機能 B も選択されません。メイン機能に関連付けられたコンポーネントがある場合、この動作が原因で問題が生じることがあります。この例では機能 B がこれに該当します。コンポーネントを含む機能が選択されていないため、直接関連付けられたコンポーネントもインストールされません。

DCOM が選択されているかどうかに関わらずコンポーネントが選択されるようにするには、オブジェクト内にサンプル機能機能を作成します。機能 C が選択されない場合も、サンプル機能が選択されます。サンプル機能が選択されるため、機能 B も選択され、そのコンポーネントと共にインストールされます。

言語依存関係のインストール

複数言語をターゲットにするインストールを開発するには、同じファイルの別の言語バージョンを使用することがよくあります。これらのファイルに共通する名前がついていると、InstallShield でインストールをビルドして、すべてのインストール ファイルをメディアにコピーする際に、問題が発生する可能性があります。リリース ウィザードを実行する際に、機能内のすべてのファイルをコピーし、リリース上の同じディレクトリに保管します。同じ名前が付いた 2 つのファイルが別々のコンポーネントに存在する場合、その 2 つのコンポーネントが同じ機能に属するとそのうちの 1 つが上書きされてしまいます。この問題を解決するには、ターゲットとする各言語の機能を別に作成する必要があります。詳細については、「インターナショナル化のためのオブジェクトの作成」を参照してください。

オブジェクト ファイルの管理



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトをビルドする際には、そのすべてのファイルは、圧縮されていない形式でリリースへ送られます。その結果、同じ名前をもつ複数のファイルを含むオブジェクトが問題を引き起こすことがあります。同じ名前のファイルが同じ機能内にある場合は、それらのうちの一つだけがリリースへ送られることになります。同じ名前をもつ残りのファイルは、リリースへ送られた最後のファイルによって上書きされてしまいます。

同じ名前のファイルをもつ主な理由は、複数の言語をサポートするためです。その場合、サポートする各言語について別々の機能を作成することを推奨します。詳細については、「インターナショナル化のためのオブジェクトの作成」を参照してください。

問題の防止をより確実にするためには、すべての機能について CD-ROM フォルダーを定義します。CD-ROM フォルダーを定義することで、各機能のファイルが送られるリリース上の場所を指定します。CD-ROM フォルダーを定義するには、[機能] ビューへ移動し、"CD-ROM フォルダー" プロパティでフォルダーの名前を入力するだけです。

同じ名前があるファイルをもつ機能には、CD-ROM フォルダーを定義しなければなりません。(例: Setup.inx というコンパイル済みのスクリプト ファイル、または [サポート ファイル/ビルドード] ビューの [アドバンス ファイル] ノードの [Disk1] サブノードに配置したファイル)

オブジェクトのウィザードの設計



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトにプロパティを含める際、オブジェクトのユーザーにインターフェイスを提供して、そのプロパティを変更できるようにすることをお勧めします。InstallShield では、作成したプロパティに基づいてストック ウィザードを作成したり、Visual Basic や Visual C++ を使用して独自のウィザードを作成できます。

InstallShield ストック ウィザード

InstallShield ストック ウィザードは、オブジェクトの社内使用に最適です。ストックウィザードには、すべての読み取り専用プロパティが表示され、読み書きが可能なプロパティには編集フィールドが表示されます。

ストック ウィザードにはいくつかの制限がありますが、カスタム ウィザードを使用して回避することができます。たとえば、ストック ウィザードをご自分のニーズに合わせてカスタマイズすることはできません。表示するプロパティを選択したり、ウィザードの表示方法をカスタマイズすることもできません。ウィザード用にカスタマイズしたインターフェイスを作成する場合、Visual C++ や Visual Basic で独自のウィザードを作成する必要があります。ストック ウィザードのもう 1 つの欠点は、配列プロパティをサポートしていないことです。このように、ストックウィザードにはいくつかの制限がありますが、簡単に実装および使用できます。

カスタム ウィザード

ウィザードによりプロフェッショナルな印象を持たせたり、機能性を向上、または表示をカスタマイズする必要がある場合、Visual C++ または Visual Basic を使用して独自のウィザードを作成します。これらのアプリケーションを InstallShield インストールを実行する前にインストールした場合、InstallShield オブジェクト ウィザードを作成する機能が追加されます。InstallShield をインストールした後にこれらのプログラムをインストールした場合は、InstallShield インストールをメンテナンス モードで実行し、ウィザード機能を Visual Studio アプリケーションに追

加する必要があります。Visual C++ で InstallShield オブジェクト ウィザードを作成するには、[ファイル] メニューから [新規作成] を選択します。その後、プロジェクトの種類のリストから [InstallShield オブジェクト ウィザード] を選択します。Visual Basic の場合は、[新規プロジェクト] ダイアログ ボックスから [InstallShield オブジェクト ウィザード] を選択します。

Visual C++ で InstallShield オブジェクト ウィザードを作成すると、VC ウィザード フレームワーク プロジェクトをベースとした新しい ATL-MFC が表示されます。Visual Basic の場合は、テンプレートプロジェクトが作成されます。各プロジェクトの種類内のコメント化されたコードは、オブジェクトプロパティの取得方法と保存方法を示します。詳細については、「カスタム オブジェクト ウィザード」を参照してください。

ウィザードの作成が完了したら、コンパイルします。

メモ: *カスタムウィザードがあるオブジェクトを配布する場合、配布インストール プロジェクトのウィザード .dll に依存関係を含めるようにしてください。[セットアップのデザイン] ビューまたは [機能] ビューの InstallShield オブジェクトインストーラー機能の上に配置する機能に必要なファイルを含めてください。スタティック(静的) ウィザード .dll を作成すると依存関係を使用せずに済みます。*

InstallShield ストック ウィザード、またはカスタム ウィザードのどちらを使用するかを指定する

タスク ***InstallShield ストック ウィザードまたは作成したカスタム ウィザードのどちらを使用するかを指定するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. 以下のいずれかを実行します。
   - ストック ウィザードを使用する場合: ***"オブジェクト ウィザード"*** 設定で、**[InstallShield オブジェクト ストック ウィザードを使用する]** を選択します。
   - 独自のカスタム ウィザードを使用する場合: ***"オブジェクト ウィザード"*** 設定で、省略記号ボタン (...) をクリックします。**[マイ カスタム オブジェクトの参照ウィザード]** ダイアログ ボックスが表示されます。オブジェクト用に作成したウィザード ファイルを選択します。

プロジェクトのリリースをビルドするとき、指定したオブジェクトがウィザードに含まれます。

カスタム オブジェクト ウィザード

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクト用に InstallShield ですでに定義されているストック ウィザードではなくて、独自のウィザードを使用する場合、インプロセスの COM サーバーを提供する必要があります。このサーバーは、IDevWizard インターフェースを実装しなければなりません。このインターフェースは 、オブジェクトが最初にプロジェクトに含められる際と、[オブジェクト] ビューの [機能] ペインのすでに含められているオブジェクトのコンテキスト メニューから [変更] が選択される際に、InstallShieldによって呼び出されます。このインターフェースはオブジェクトのプロパティの値を表示したり、これらの値を変更するのに便利です。

このインターフェースは、次の 2 つの引数と共に Display メソッドを提供する必要があります。

- オブジェクトのプロパティへのアクセスを提供する IDispatch ポインター。
- 既存のオブジェクトのプロパティに変更を加えられているかどうか、既存のオブジェクトが存在するかしないか、新しいオブジェクトを作成する必要があるかどうかなどを通知するブール戻り値。

C++ 言語の場合、このインターフェースは下記のように定義されます。

```
interface IDevWizard : IDispatch
{
HRESULT Display([in] IDispatch *pObject,
                [out, retval] VARIANT_BOOL* Confirmed);
};
```

サーバーは、ISetupDesignObjectBuild インターフェースも実装できます。InstallShield リリース ビルダーは、このインターフェイスを呼び出して、オブジェクトのどの機能がそのオブジェクトを含むプロジェクトのビルドから除外されるべきかを判別します。(このインターフェイスが実装されない場合、すべてのオブジェクトの機能がビルドに含まれます。)

このインターフェイスは、インストール作成者がオブジェクトのウィザードで選択を実行するため、ビルドされたプロジェクトからこれからインストールされないオブジェクトの機能を除外する際に便利です。たとえば、オブジェクトがオプションで様々なドライバーに対するサポートを提供し、プロジェクトの作成者がそのドライバーのうちの幾つかを含めないと選択する場合、そのドライバーを含む機能をビルドされたプロジェクトから除外することができます。

このインターフェースは、実装されている場合、次の 3 つの引数と共に IsIncluded メソッドを提供する必要があります。

- オブジェクトのプロパティへのアクセスを提供する IDispatch ポインター。
- 機能を指定する文字列値 (例、**Feature 1¥Subfeature 1A**)。
- このコンポーネントをビルドに含める必要があるかどうかを通知するブール戻り値。

C++ 言語の場合、このインターフェースは下記のように定義されます。

```
interface ISetupDesignObjectBuild : IDispatch
{
HRESULT IsIncluded([in] IDispatch *pObject,
                   [in] BSTR FeaturePath,
                   [out, retval] VARIANT_BOOL* Included);
};
```

メモ- *カスタムウィザードがあるオブジェクトを配布する場合、配布インストール プロジェクトのウィザード .dll に依存関係を含めるようにしてください。[セットアップのデザイン] ビューまたは [機能] ビューの InstallShield オブジェクトインストーラー機能の上に配置する機能に必要なファイルを含めてください。スタティック(静的) ウィザード .dll を作成すると依存関係を使用せずに済みます。*

オブジェクトのデザインタイム環境のローカライズ

プロジェクト- *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

InstallShield インターフェイスでローカライズする各言語について、別々の表示名、短い名前、.htm ファイル、およびオブジェクトのアイコンを指定できます。このローカライズは、[プロジェクトの設定] ダイアログ ボックスの [言語固有オブジェクト プロパティ] タブで行います。

[言語固有オブジェクト プロパティ] タブの最初のオプションは [IDE 言語] リスト ボックスです。サポートする最初の言語を選択し、その言語が英語でない場合は [デフォルトを使用する] ボックスのチェックを外します。[言語固有オブジェクトのプロパティ] タブ の残りを入力します。

サポートしている言語が 1 つだけの場合、このダイアログを閉じることができます。しかし、さらに言語を追加してサポートする場合、その言語をリストから選択して、それに必要な情報を提供します。



メモ: *独自のウィザードを作成する場合は、サポートする各言語に対して、ウィザードの実行可能ファイルでランタイム文字列を提供します。*

インターナショナル化のためのオブジェクトの作成



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

ターゲットマシンに基づいて様々なファイルをインストールするオブジェクトをデザインする作業は、同様の機能によってインストールをデザインする作業と類似しています。オブジェクトと標準インストール プロジェクトの重要な違いは、オブジェクトをインターナショナル化する作業はインストールをデザインするよりも微妙に複雑であるというところにあります。この違い、つまり、リリースビルダーは非圧縮したオブジェクトのプロジェクト ファイルをディスク イメージに格納するので、インターナショナル化したファイルは機能レベルで処理する必要があります。次のセクションでは、オブジェクトをインターナショナル化するのに必要な手順と方法について説明します。

オブジェクトに言語を追加する

InstallShield の Premier Edition がある場合、[プロジェクト設定] ダイアログ ボックスの [言語] タブ を使用してオブジェクトにサポート言語を関連付けることができます。

言語固有のファイル グループを設計する

インターナショナル化されたオブジェクトを作成する上で最も重要な手順は、使用する言語固有のデータをコンポーネントから分離させることです。この処理をわかりやすいように例を示して説明します。英語、フランス語、ドイツ語の 3 つの言語をサポートし、そのうち英語をデフォルトの言語にする場合、少なくともコンポーネントが 4 つ必要になります。最初のコンポーネントには英語のみのファイルが格納されます。このコンポーネントの言語プロパティでは、英語と間接的にサポートしている言語すべてを設定する必要があります。この操作を実行することにより、オブジェクトを使用可能にして、英語、フランス語、ドイツ語以外の言語でもインストールできるようになります。ターゲットマシンがサポートされる 3 言語 (英語、フランス語、ドイツ語) 以外の言語で稼動している場合、英語のファイルがインストールされます。

フランス語とドイツ語のコンポーネントには、単純にそれぞれのコンポーネントに対してフランス語とドイツ語のファイルを追加します。フランス語のコンポーネントの言語を フランス語と設定し、ドイツ語のコンポーネントの言語を ドイツ語と設定します。

最後に、使用する言語非依存関係を別のコンポーネントに追加します。このコンポーネントを言語非依存としてマークします。4 つのファイルグループのレイアウトを下記の表に示します。

テーブル 1・ファイル グループ レイアウトのサンプル

ファイルの言語	コンポーネント言語
英語固有ファイルすべて	英語 + サポートされない言語すべて
フランス語固有ファイルすべて	フランス語
ドイツ語固有ファイルすべて	ドイツ語
言語非依存関係すべて	言語非依存

言語固有機能を設計する

オブジェクトの言語固有データを含める場合、ターゲットする言語のそれぞれに対する機能を作成する必要があります。実際には機能の言語は定義できないことから、これは不必要な手順のように見えます。しかし、ディスクイメージ内のファイルの保管形態上、コンポーネントを 複数の言語から 1 つの機能に追加できなくなっています。リリース ウィザードを実行する際に、機能内のすべてのファイルをコピーし、リリース上の同じディレクトリに保管します。同じ名前が付いた 2 つのファイルが別々のコンポーネントに存在する場合、その 2 つのコンポーネントが同じ機能に属するとそのうちの 1 つが上書きされてしまいます。この問題を解決するには、ターゲットとする各言語の機能を別に作成する必要があります。次に、対応するコンポーネントを追加します。

実行時に、オブジェクトとインストールとの互換性を持たせる



***プロジェクト・** この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトは、オブジェクトを含むインストールによってサポートされている、すべてのオペレーティング システムと言語をサポートしています。オペレーティング システムが一致しないと、インストールが正しく実行されません。言語が一致しないと、インストールのエンドユーザーが混乱します。そのような不一致を避けるために、以下の 1 つを実行します:

- InstallShield でサポートされているすべてのオペレーティング システムと言語のオブジェクトにサポートを追加する。オブジェクト プロジェクトにオペレーティング システム サポートを追加するには、[一般情報] ビューに移動して、"プラットフォーム フィルター"設定を構成します。オブジェクト プロジェクトに言語サポートを追加するには、「インターナショナル化のためのオブジェクトの作成」を参照してください。
- インストール作成者にサポートされているオペレーティング システムと言語を通知する文書をオブジェクトに含める。これを行うには、必要な情報を含む .htm ファイルをまず作成し、[プロジェクト] メニューで [設定] をクリックします。[言語非依存オブジェクト プロパティ] タブをクリックして、"HTML ヘルプ" フィールドで .htm ファイルを指定します。

プロパティとメソッド



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

定義として、プロパティはデータ構造に関連付けられている属性を記述するのに使用され、メソッドは、オブジェクトがメッセージを受け取ったときに実行されるプロシージャのことを指します。

インストールの異なる側面に関連付けられたプロパティは、InstallShield のどの箇所でも、追加、作成および設定することができます。プロパティは、オブジェクト スクリプトで役に立つことがあります。

メソッドを使用することにより、プロパティと同様、オブジェクトがそれを含むインストールとインタラクトできるようになります。実際的な目的に対するメソッドとは関数のことです。メソッドと関数の違いは、メソッドはメソッド キーワードで宣言されるということだけです。メソッドを使用すると、オブジェクトの関数を他のオブジェクトやインストールで使用できるようにすることができます。

プロパティの作成



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

プロパティは、オブジェクト スクリプトで役に立つことがあります。オブジェクト スクリプトでは、プロパティを使うと、オブジェクトのユーザーは、InstallShield オブジェクト ウィザード によるデザインを行っている最中、またはスクリプトによってパラメーターを渡して実行している最中に特定のオブジェクトの設定を変更することができます。共通プロパティには、実行時やデザイン 時にオブジェクトからインストールへのエラー メッセージの転送、ユーザー名やオブジェクトに必要な情報の取得などがあります。プロパティの使用例については、[オブジェクト] ビューにある InstallShield オブジェクトのインライン ヘルプを参照してください。

プロパティについて以下の点にご注意ください。

- Boolean プロパティの値がオブジェクトスクリプトまたはそのオブジェクトを含むインストールのスクリプトによって TRUE に設定されている場合、後に続くプロパティの値の以降のチェックには 1 ではなく -1 の値が表示されます。
- オブジェクトとインストール スクリプトの間にある LIST 変数を渡すことはできませんが、文字列配列をパスすることは可能です。

プロパティの作成



タスク ***プロパティを作成するには、以下の手順に従ってください。***

1. **InstallScript** ビューを開きます。
2. **InstallScript** エクスプローラーで、**[プロパティ]** を右クリックして、**[新しいプロパティの追加]** をクリックします。**[新しいプロパティの追加]** ダイアログ ボックスが開きます。
3. プロパティの設定を定義します。
4. **[OK]** をクリックします。

[新しいプロパティの追加] ダイアログ ボックスにより、以下のスクリプトが追加されます。

- プロパティを宣言する構文:

  property(<プロシージャ>) <プロパティ データ型> <プロパティ名> ( );

  <プロシージャ> は読み取り専用プロパティの場合 **get**、書き込み専用プロパティの場合 **put** で、読み取り/書き込みプロパティの場合 **get,put** です。例:

  property(get,put) STRING MyProperty ( );

  プロパティ名の後のかっこ内に引数のデータ型を追加して、プロパティの引数を指定することができます。
  例:

  property(get,put) STRING MyProperty (NUMBER);

  プロパティの宣言に引数のデータ型を追加した場合、プロパティのプロシージャ関数の定義に対応する引数の変数を追加する必要があります。

- スクリプト内にプロパティの値を保存する変数の宣言例:

  STRING m_strMyProperty;

  [新しいプロパティの追加] ダイアログ ボックスで ([配列] チェック ボックスを選択して) 配列プロパティを指定する場合、プロパティの値を保存する VARIANT 型の変数と、プロパティの値を初期化するために **InitProperties** 関数ブロックで使用するサイズ指定のない配列変数の 2 つの変数を宣言します。例:

  STRING arrayMyArrayProperty();
  VARIANT m_arrayMyArrayProperty;

- プロパティの値を保存する変数を初期化する、**InitProperties** 関数ブロック内のステートメント。例:

  m_strMyProperty = "デフォルト値";

  [新しいプロパティの追加] ダイアログ ボックスで ([配列] チェック ボックスを選択して) 配列プロパティを指定した場合、プロパティの値を保存する変数は、配列変数の値に割り当てられます。例:

  m_arrayMyArrayProperty = arrayMyArrayProperty;

  配列プロパティの要素をヌルやゼロ以外の値に初期化するには、上記ステートメントの前に配列変数の要素の割り当てステートメントを追加します。例:

  /* サイズなしと宣言された配列変数のサイズを変更する。*/
  Resize (arrayMyArrayProperty , 3);

  /* 配列変数の要素に値を割り当てます。*/
  arrayMyArrayProperty(0) = "zero";
  arrayMyArrayProperty(1) = "one";
  arrayMyArrayProperty(2) = "two";

  m_arrayMyArrayProperty = arrayMyArrayProperty;

- プロパティ バッグ オブジェクトに保存されているプロパティの値を取得する、**ReadProperties** 関数ブロック内での関数呼び出し。この呼び出しの構文は次のとおりです。

  <Read*xxxx*Property> (PropertyBag, "<プロパティ名>", <プロパティ値変数>);

  <Read*xxxx*Property> は、プロパティの値のデータ型に適した Read*xxxx*Property 関数です。例:

  ReadStringProperty(PropertyBag, "MyProperty", m_strMyProperty);

- プロパティ バッグ オブジェクトにプロパティの値を保存する、**WriteProperties** 関数ブロック内での関数呼び出しこの呼び出しの構文は次のとおりです。

  <Write*xxxx*Property> (PropertyBag, "<プロパティ名>", <プロパティ値変数>);

  <Write*xxxx*Property> は、プロパティの値のデータ型に適した Write*xxxx*Property 関数です。例:

  WriteStringProperty(PropertyBag, "MyProperty", m_strMyProperty);

- 関数ブロックは、プロパティのプロシージャを定義します。つまり、オブジェクトを含むプロジェクト、またはオブジェクトのデザイン時のウィザード (該当する場合) によってプロパティの値が読み取られたときに実行されるアクションを定義します。

  get_<プロパティ名> 関数は読み取り専用または読み書き可能なプロパティに対して定義し、**put_<プロパティ名>** 関数は書き込み専用または読み書き可能なプロパティに対して定義します。例:

  ```
  function STRING get_MyProperty()
  begin
     return m_strMyProperty;
  end;

  function void put_MyProperty(newVal)
  begin
     m_strMyProperty = newVal;
  end;
  ```

  (引数のデータ型をプロパティ名の後のカッコ内のプロパティ宣言に追加することによって)プロパティの引数を指定する場合、プロパティのプロシージャ関数の定義に対応する引数変数を追加する必要があります。例:

  ```
  function STRING get_MyProperty( szAddedArgument )
  function void put_MyProperty( szAddedArgument, newVal )
  ```

  put_ 関数に引数を追加する場合、newVal 引数の前に配置してください。newVal 引数は、プロジェクトの呼び出しの割り当てステートメント、またはオブジェクトのデザインタイム ウィザード (該当する場合) によりプロパティに書き込まれた値を取ります。



メモ- *get_ および put_ 関数の名前を変更しないでください。変更した場合、スクリプトが正常に機能しません。*

構造評価プロパティの作成



プロジェクト- *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

値がデータ構造のプロパティを作成するには、オブジェクト スクリプトに次の内容を追加します。

- データ構造のグローバル宣言。以下に例を示します。

  ```
  typedef STRUCT
  begin
     NUMBER Mem1;
     NUMBER Mem2;
  end;
  ```

- データ型が OBJECT である読み取り専用プロパティの宣言。(プロパティに値を書き込むには、メソッドを定義します。)例:

  ```
  property(get) OBJECT StructProp();
  ```

  この宣言をオブジェクト スクリプトの Properties Section ブロックに配置します。

- オブジェクト スクリプト内にプロパティの値を格納する変数の宣言例:

  ```
  STRUCT m_structStructProp;
  ```

  この宣言をオブジェクトスクリプトの "ローカル変数セクション" ブロックに配置します。

- データ構造の各メンバーに対応する適切なデータ型の引数を持つメソッドの宣言。上記で定義した STRUCT 構造について、適切なメソッド宣言の例を以下に示します。

  ```
  method NUMBER SetStructProp(NUMBER, NUMBER);
  ```

  この宣言をオブジェクト スクリプトの "メソッド セクション" ブロックに配置します。

- プロパティの値を保存する変数のメンバーを初期化する、**InitProperties** 関数ブロック内のステートメント。例:

  ```
  m_structStructProp.Mem1 = 3;
  m_structStructProp.Mem2 = 5;
  ```

- プロパティ バッグ オブジェクトに保存されているプロパティの値を取得する、**ReadProperties** 関数ブロック内での関数呼び出しとステートメント関数呼び出しは次のような構文を持っています:

  <Read*xxxx*Property> (PropertyBag, "<メンバー値のストレージ名>", <メンバー値変数>);

  <Read*xxxx*Property> は、プロパティ メンバーの値のデータ型に適した Read*xxxx*Property 関数です。例:

  ```
  ReadNumberProperty(PropertyBag, "StructProp_Mem1", nStructPropMem1);
  ```

  このような各関数呼び出しについては、関数が取得する値と同等のプロパティ変数のメンバーを設定するステートメントを下に追加します。例:

  ```
  m_structStructProp.Mem1 = nStructPropMem1;
  ```

- プロパティ バッグ オブジェクトにプロパティメンバーの値を保存する、**WriteProperties** 関数ブロック内での関数呼び出し関数呼び出しは次のような構文を持っています:

  <Write*xxxx*Property> (PropertyBag, "<メンバー値のストレージ名>", <プロパティ変数メンバー>);

  <Write*xxxx*Property> は、プロパティ メンバーの値のデータ型に適した Write*xxxx*Property 関数です。例:

  ```
  WriteNumberProperty(PropertyBag, "StructProp_Mem1", m_structStructProp.Mem1);
  ```

- get_<プロパティ名> 関数ブロックは、プロパティの get プロシージャを定義します。つまり、オブジェクトを含むプロジェクト、またはオブジェクトのデザインタイム ウィザード (該当する場合) により、プロパティの値が読み取りまたは書き込まれた際に実行されるアクションを定義します。例:

  ```
  function OBJECT get_StructProp()
  begin
     return m_structStructProp;
  end;
  ```

- 宣言されたメソッドを定義する関数ブロック。このメソッドはメソッドに渡された引数と等しいプロパティ変数のメンバーを設定します。例:

```
function NUMBER SetStructProp( nMem1, nMem2 )
begin
    m_structStructProp.Mem1 = nMem1;
    m_structStructProp.Mem2 = nMem2;
end;
```

オブジェクトとインストールの間で文字列配列を渡す



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトとインストール間で文字列配列を渡すには、以下の手順を実行します。

- 以下のオブジェクト スクリプトを追加します。
  - データ型が 異なる読み書きのプロパティの宣言。例:

    ```
    property(get,put) variant Test();
    ```

    この宣言をオブジェクト スクリプトの Properties Section ブロックに配置します。
  - オブジェクト スクリプト内にプロパティの値を保存する変数に必要な宣言。

    ```
    STRING arrayTest();
    variant m_arrayTest;
    ```

    この宣言をオブジェクト スクリプトの Local Variables Section ブロックに配置します。
  - プロパティの値を保存する変数のメンバーを初期化する、InitProperties 関数ブロック内のステートメント。例:

    ```
    m_arrayTest = arrayTest;
    ```
  - プロパティバッグオブジェクトに保存されている値を取得する、ReadProperties 関数ブロック内での関数呼び出し。例:

    ```
    ReadArrayProperty(PropertyBag, "Test", m_arrayTest);
    ```
  - プロパティ バッグ オブジェクトに値を保存する、WriteProperties 関数ブロック内での関数呼び出し。例:

    ```
    WriteArrayProperty(PropertyBag, "Test", m_arrayTest);
    ```
  - 関数ブロックは、プロパティの get プロシージャを定義します。つまり、オブジェクトを含むプロジェクト、またはオブジェクトのデザインタイム ウィザード (該当する場合) により、プロパティの値が読み込まれた時に実行されるアクションを定義します。例:

    ```
    function variant get_Test()
    begin
        return m_arrayTest;
    end;
    ```
  - 関数ブロックは、プロパティの put プロシージャを定義します。つまり、オブジェクトを含むプロジェクト、またはオブジェクトのデザインタイム ウィザード (該当する場合) により、プロパティの値が書き込まれた時に実行されるアクションを定義します。例:

    ```
    function void put_Test(newVal)
    begin
        m_arrayTest = newVal;
    ```

```
end;
```

- 以下のインストールスクリプトを追加します。
  - オブジェクトからの文字列配列の読み取りやオブジェクトへの書き込みをする各関数ブロックでは、オブジェクト変数の宣言や文字列配列。例:

    ```
    OBJECT oObject;
    string szTest(3);
    ```

  - オブジェクトへのリファレンスを取得するための GetObject への呼び出し。例:

    ```
    set oObject = GetObject("Object");
    if (!IsObject(oObject)) then
        MessageBox( "オブジェクトのリファレンスを取得できませんでした。", INFORMATION );
        abort;
    endif;
    ```

  - "オブジェクトの文字列配列" プロパティの値を設定するには、以下のコードを使用します:

    ```
    oObject.Test = szTest;
    ```

    "オブジェクトの文字列配列" プロパティの値を取得するには、以下のコードを使用します:

    ```
    szTest = oObject.Test;
    ```

オブジェクトのプロパティにアクセスする



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトをコンパイルした後にプロパティを変更する方法は 2 つあります。最初の方法では、オブジェクトがインストール プロジェクトに追加された時に変更を行います。オブジェクトのウィザードインターフェイスを提供する場合、インストールのデザイン タイム中に読み書きのプロパティが変更されます。2 番目の方法では、実行時にスクリプトを使用してプロパティを変更します。オブジェクトのユーザーが実行時にプロパティにアクセスできるようにするには、オブジェクトのヘルプトピックにこれらのプロパティをドキュメント化する必要があります。これにより、ユーザーは必要に応じてプロパティを変更できます。

InstallShield インターフェイス からプロパティへアクセスする

インストール プロジェクトでオブジェクトのプロパティにアクセスすると、そのオブジェクトのスクリプトの一部を InstallShield インターフェイスで実行することになります。たとえば、オブジェクトをインストール プロジェクトと関連付ける場合、そのオブジェクトのプロパティは **get_PropertyName** 関数によって読み込まれます。この関数は、プロパティを含むすべてのオブジェクトに組み込まれています。オブジェクトウィザードでオブジェクトのプロパティを変更する場合は、**put_***PropertyName* 関数が呼び出されます。ここで、*PropertyName* は変更するプロパティの名前になります。

スクリプトを使用してプロパティにアクセスする

オブジェクトのプロパティにアクセスするには、まずそのオブジェクトへの参照を取得します。その後、アクセスするオブジェクトのプロパティを指定します。以下のコードは、オブジェクトのステータス プロパティを返す方法を示します:

```
function OnBegin()
  /*オブジェクトへの参照を保持するオブジェクト変数を宣言します。*/
```

```
    OBJECT oObject;
begin
    /* "カスタム オブジェクト" という名前のカスタム オブジェクトの参照を取得します。*/
      set oObject = GetObject("カスタム オブジェクト");

    /*Status.Number プロパティを確認。*/
    if oObject.Status.Number != OBJ_STATUS_SUCCESS then
        /*セットアップに合わせてエラーに応答。
        たとえば Status.description テキストを表示して
        セットアップを中止できます。*/
        MessageBox ( oObject.Status.Description, SEVERE );
        abort;
    endif;
end;
```

実行時にプロパティを変更する場合、その情報は保存されず、今後のインストールでは使用されません。たとえば、後でインストールをメンテナンス モードで実行すると、最初のインストールで与えられた値はプロパティに保持されません。

Boolean プロパティの値がオブジェクトスクリプトまたはそのオブジェクトを含むインストールのスクリプトによって TRUE に設定されている場合、後に続くプロパティの値の以降のチェックには 1 ではなく −1 の値が表示されます。



ヒント: *InstallScript Object で、単にオブジェクトの現在のステータス (Status.Number の現在の値) を取得するだけのときは、***GetStatus** *関数を使用できます。詳細については、「GetStatus」を参照してください。*

メソッドの作成



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

メソッドは、プロパティと同様に、オブジェクトとそれを含むインストールが相互に対応するようにすることができます。実際的な目的に対するメソッドとは関数のことです。メソッドと関数の違いは、メソッドはメソッドキーワードで宣言されるということだけです。メソッドを使うと、オブジェクトの関数を他のオブジェクトやセットアップで使用することができます。

メソッドの作成



タスク

メソッドを作成するには、以下の手順を実行します。

1. **InstallScript** ビューを開きます。
2. **InstallScript** エクスプローラーで、**[メソッド]** を右クリックして、**[新しいメソッドの追加]** をクリックします。**[新しいメソッドの追加]** ダイアログ ボックスが開きます。
3. メソッドの設定を定義します。
4. **[OK]** をクリックします。

[新しいメソッドの追加] ダイアログ ボックスにより、以下のオブジェクト スクリプトが追加されます。

- メソッドを宣言する構文

  method <メソッドデータタイプ> <メソッド名> (<引数データタイプ>);

  例:

  method STRING MyMethod(NUMBER, BOOL);

- メソッドを定義する関数ブロック例:

```
function STRING MyMethod( /*NUMBER*/ arg1, /*BOOL*/ arg2 )
begin
end;
```

メソッドを使用する



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトのメソッドには実行時に限りアクセスできます。オブジェクトのメソッドをインストール プロジェクト内から、または別のオブジェクトから呼び出すことができます。オブジェクトのメソッドにアクセスするには、まずそのオブジェクトの参照を取得します。次のコードはオブジェクトの現在の状態についての説明を表示します:

```
Set oObj = GetObject("ATL 3.0")
MessageBox(oObj.Status.Description, INFORMATION);
```

オブジェクトステータスのプロパティ



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

あらかじめ定義されているものも、ユーザーによって定義されたものも含め、すべての InstallScript オブジェクトは以下のような読み取り専用プロパティを持っています。

- Status
- StatusBase
- StatusStruct

StatusBase と StatusStruct は、高度な知識をもつユーザー向けです。これらのプロパティに関する詳細については、次のセクションを参照してください。

Status

Status プロパティは、次のプロパティをメンバーに持つ構造です。

テーブル 2 · Status プロパティのメンバー

プロパティ	説明
Status.Number	数値のオブジェクト ステータス。

テーブル 2 ・ Status プロパティのメンバー (続き.)

プロパティ	説明
Status.Description	オブジェクトのステータスを説明する文字列。
Status.szSource	エラーの場合、エラーのソース。通常、オブジェクトの文字列エントリから読み込まれたオブジェクトの名前。
Status.szScriptFile	エラーの場合、エラー発生時に実行されていたスクリプト ファイルの名前。
Status.nScriptLine	エラーの場合、エラー発生時に実行されていたスクリプト行。
Status.nScriptError	エラーの場合、エラーのリターン コード。

オブジェクト スクリプトで、**SetStatusEx** を呼んでこれらの値を設定することができます。オブジェクトのステータスをクエリすると、デフォルトでは最初サブオブジェクトのステータス (あれば) をクエリします。1 つ以上のオブジェクトのサブオブジェクトがエラーを報告 (OBJ_STATUS_SUCCESS より少ない Status.Number プロパティ) すると、最初に失敗したオブジェクトのステータスプロパティが親オブジェクトのステータスとして返されます。ユーザーによって定義されたオブジェクトのスクリプトは、(引数なしで) get_Status 関数を明示的に定義することによってデフォルトの動作をオーバーライドすることができます。

オブジェクトの Status.Number プロパティの値を確認して、オブジェクトを正常にインストールすることが可能かどうか、可能ではない場合、その原因を、判断し、エンドユーザーに Status.Description の値を表示することができます。この確認は、セットアップのどの時点でも行うことができますが、最低、以下の例にあるように、**OnBegin** イベント ハンドラーで成功または失敗したかを確認することをお勧めします。

```
function OnBegin()
    /*オブジェクトへの参照を保持するオブジェクト変数を宣言します。*/
    OBJECT oObject;
begin
    /* "カスタム オブジェクト" という名前のカスタム オブジェクトの参照を取得します。*/
    set oObject = GetObject("カスタム オブジェクト");

    /*Status.Number プロパティを確認。*/
    if oObject.Status.Number != OBJ_STATUS_SUCCESS then
        /*セットアップに合わせてエラーに応答。
        たとえば Status.description テキストを表示して
        セットアップを中止できます。*/
        MessageBox ( oObject.Status.Description, SEVERE );
        abort;
    endif;
end;
```

以下のテーブルには、すべてのオブジェクトに共通な Status.Number および Status.Description の可能な値がリストされています。一部のオブジェクトには、固有のプロパティに可能な値があります。可能な値のリストについては、個別のオブジェクトのヘルプ ページを参照してください(**[オブジェクト]** ビューまたは **[セットアップのデザイン]** または関連機能の **[機能]** ビューでオブジェクトを選択すると表示できます)。

テーブル 3 ・ すべてのオブジェクトに共通な Status.Number および Status.Description の可能な値

oObject.Status.Number	oObject.Status.Description
OBJ_STATUS_SUCCESS	操作は成功しました。
OBJ_STATUS_OSINVALID	ターゲット OS が無効です。
OBJ_STATUS_DISKSPACE	十分なディスク容量がありません。
OBJ_STATUS_LANGUAGESUPPORT	サポートされていない言語です。
OBJ_STATUS_NOTADMINISTRATOR	管理者特権が必要です。
OBJ_STATUS_SETKEY	レジストリキーを設定できません。
OBJ_STATUS_GETKEY	レジストリキーを取得できません。
OBJ_STATUS_SETVALUE	レジストリ値を設定できません。
OBJ_STATUS_GETVALUE	レジストリ値を取得できません。
OBJ_STATUS_SETTARGET	スクリプトで定義されたフォルダー値を設定できません。
OBJ_STATUS_LOADDLL	DLL をロードすることができません。
OBJ_STATUS_FILELOCKED	必要なファイルがロックされました。
OBJ_STATUS_FILENOTFOUND	ファイルが見つかりません。
OBJ_STATUS_FILECOPY	ファイルのコピーに失敗しました。
OBJ_STATUS_FILELAUNCH	ファイルを起動できませんでした。
OBJ_STATUS_SETLOGDATA	セットアップ ログにデータを設定できません。
OBJ_STATUS_GETLOGDATA	セットアップ ログからデータを取得できません。
OBJ_STATUS_INITVARS	変数の初期化ができません。

StatusBase

StatusBase プロパティは、Status プロパティと同じメンバーを持つ構造です。このプロパティは、通常、セットアップがこのプロパティを直接クエリする必要がない特殊なケース（セットアップはオブジェクトから提供されるプロパティを処理してすべてのカスタムステータスをバイパスすることができるため）を除き、カスタマイズされた get_Status メソッドを持つオブジェクトによってのみ使用されます。

StatusBase プロパティを使って、オブジェクトのサブオブジェクトのステータスのチェックなど、オブジェクトのデフォルトのステータス ハンドリングを利用することができます。オブジェクトに カスタム get_Status 関数を作成すると、以下の例にあるように、get_Status 関数内の StatusBaseプロパティを使用して、デフォルトの ステータス ハンドリングを利用することができます。

```
function object get_Status()
    object oThis;
begin
    //ここでカスタム処理を行います。

    //オブジェクトのリファレンスを取得します。
    set oThis = GetObject( "" );

    //これがオブジェクトでない場合は失敗します。
    if( !IsObject( oThis ) ) then
        return NULL;
    endif;

    //標準のステータス情報を返します。
    return oThis.StatusBase;
end;
```

StatusStruct

StatusStruct プロパティは、Status プロパティと同じメンバーを持つ構造です。このプロパティは、通常、インストールがこのプロパティを直接クエリする必要がない特殊なケース（セットアップはオブジェクトから提供されるプロパティを処理してすべてのカスタムステータスをバイパスすることができるため）を除き、カスタマイズされた get_Status メソッドを持つオブジェクトによってのみ使用されます。

StatusStruct プロパティを使って、デフォルトのオブジェクト ステータス ハンドリングをバイパスすることができます。オブジェクトにサブオブジェクト ステータス値を無視させたい場合、次の例にあるように、このプロパティを使用することができます。

```
function object get_Status()
    object oThis;
begin
    //オブジェクトのリファレンスを取得します。
    set oThis = GetObject( "" );

    //これがオブジェクトでない場合は失敗します。
    if( !IsObject( oThis ) ) then
        return NULL;
    endif;

    //ステータス情報を直接返します。
    return oThis.StatusProp;
end;
```

オブジェクトの実行時のユーザー インターフェイスを作成する



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

作成したオブジェクトで、このオブジェクトを含むインストールのエンドユーザーにダイアログ ボックスのシーケンスを表示できます。このダイアログのシーケンスを定義するには、オブジェクト スクリプトの UI イベント ハンドラー (OnFirstUIBefore、 OnFirstUIAfter、OnMaintUIBefore、および OnMaintUIAfter) を使用してください。(デフォルトでは、これらのハンドラーはダイアログをオブジェクトのプロジェクトで表示しません。)

ダイアログ シーケンスを使ったオブジェクトを含むプロジェクトでは、オブジェクトのダイアログを表示するには 2 つの方法があります。

- **ShowObjWizardPages** 関数を呼び出して、プロジェクトのある時点ですべてのオブジェクトのダイアログ シーケンスを表示します。
- オブジェクトの Show*xxxxx*UI*yyyyy* メソッド (つまり、 ShowFirstUIBefore、ShowFirstUIAfter、ShowMaintUIBefore、および ShowMaintUIAfter) をいずれか、またはすべてを選択して、各オブジェクトのダイアログ シーケンスを表示します。

ShowObjWizardPages 関数

ShowObjWizardPages 関数はデフォルトで InstallShield X、InstallShield DevStudio、または InstallShield Professional 6.1 以降を使って作成した InstallScript インストール プロジェクトの各 UI イベント ハンドラーの中で呼び出されます。デフォルトのイベント ハンドラーでのこの関数の利用方法をガイドラインとして参考にしてください。**ShowObjWizardPages** をプロジェクトの UI イベント ハンドラーの中で呼び出すと、インクルードした各オブジェクトの対応 UI コードが表示されます。たとえば、**ShowObjWizardPages** をプロジェクトの OnFirstUIBefore イベント ハンドラーの中で呼び出すと、インクルードした各オブジェクトの対応 OnFirstUIBefore コードが表示されます。オブジェクトがプロジェクトの [コンポーネント] ビューで表示される順番で、インクルードしたオブジェクトの UI イベント ハンドラーが実行されます。

エンドユーザーは、[次に] ボタンと [戻る] ボタンを使用して、インストール ダイアログ シーケンス全体を移動できます。それには、UI イベント ハンドラーコードのデフォルトどおりに各ダイアログ ボックス関数の戻り値 (NEXT または BACK) を変数 nResult に割り当てて、**ShowObjWizardPages** を次の要領で呼び出します。

```
nResult = ShowObjWizardPages( nResult );
```

ShowxxxxxUIyyyyy メソッド

ユーザー作成のすべてのオブジェクトは内部で、ShowFirstUIBefore、 ShowFirstUIAfter、 ShowMaintUIBefore、および ShowMaintUIAfter のメソッドをサポートしています。オブジェクトの Show*xxxxx*UI*yyyyy* メソッドを呼び出すと、オブジェクトの対応 UI コードが実行されます。たとえば、次の呼び出しは、オブジェクトの OnFirstUIBefore コードを実行します :

```
nResult = oObject.ShowFirstUIBefore(nResult);
```

ShowObjWizardPages 関数と同様に、エンドユーザーのナビゲーションを可能にするには、各ダイアログ関数の戻り値を nResult に割り当て、前述の例のように how*xxxxx*UI*yyyyy* メソッドを呼び出します。

オブジェクト UI イベント ハンドラーのコード

デフォルトのコードを変更して、オブジェクト UI イベント ハンドラーのコードを作成します。このデフォルトのコードは、ダイアログを表示しませんが、エンドユーザーのナビゲーションを可能にするために必要なコード構造を持っています。

- ダイアログを 1 つ表示するには、FIRST_DIALOG か LAST_DIALOG ラベルのすぐ後にダイアログ関数呼び出しを配置します。
- ダイアログを 2 つ表示するには、FIRST_DIALOG ラベルの後に 1 つめのダイアログ関数呼び出しを配置して、LAST_DIALOG ラベルのすぐ後に 2 つめのダイアログ関数呼び出しを配置します。
- ダイアログを 2 つ以上表示するには、FIRST_DIALOG ラベルの後に 1 つめのダイアログ関数呼び出しを配置して、LAST_DIALOG ラベルのすぐ後に 2 つめのダイアログ関数呼び出しを配置します。中間にある各ダイアログに対しては、FIRST_DIALOG と LAST_DIALOG ブロックのようなコードブロック構造で、ダイアログ関数呼び出しを配置します。

上記のそれぞれのケースで、ダイアログ関数呼び出しの戻り値を変数からの戻り値を変数 *nDirection* に割り当てます。

オブジェクトでサポートされていない機能



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトの目的と実装のために、InstallShield インターフェイス内でオブジェクトを作成する際には特定の機能がサポートされていません。これらのサポートされていない機能を以下の一覧に示します。オブジェクトとインストール プロジェクトとの詳しい設計概念の違いについては、「オブジェクトの設計」を参照してください。

テーブル 4 ・ オブジェクトでサポートされていない機能

サポートされていない機能	説明
セットアップの種類	セットアップの種類はオブジェクトではサポートされていません。既存のインストール プロジェクトの外にオブジェクトを作成する場合、セットアップの種類は無視されます。
メディアの種類	オブジェクトを作成する際には、ビルドするメディアの種類を選択することはありません。すべてのオブジェクトは CD-ROM オプションで圧縮されずにビルドされます。
ショートカットとフォルダー	[ショートカット] ビューはオブジェクトではサポートされていません。オブジェクト内にショートカットまたはフォルダーを作成するには、**AddFolderIcon**、**CreateProgramFolder**、または **CreateDir** 関数を使ってスクリプトから作成する必要があります。
サポート ファイル	標準インストールで、[サポート ファイル] ビューでのスプラッシュ クリーン、ビルボード、言語依存、言語非依存、およびアドバンスファイルを指定するための場所。オブジェクトは言語依存と言語非依存のファイルのみを許可します。
製品のレジストリ キー	 ***Windows ロゴ :*** *InstallShield で作成されたインストールが実行される際には、製品のレジストリ キーが自動的にターゲットマシン上に作成されます。製品のレジストリ キーは、HKLM¥SOFTWARE¥<会社名>¥<製品名>¥<バージョン> に作成され、マイクロソフトのロゴに準拠するために必要です。したがって、作成されたこれらのレジストリ キーが必要な場合は、自分で追加する必要があります。*

オブジェクトのビルド



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトのビルドは、本質的には、標準インストールのビルドと同じです。InstallShield ユーザー インターフェイスからオブジェクトをビルドするには、リリース ィザードを使用します。リリース ウィザードでビルドに必要な情報すべてが、プロンプトによって聞かれます。

ユーザー インターフェイスからオブジェクトをビルドする



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*



メモ: *オブジェクトは、ライブラリ ファイル* Ifxobject.obl *と* Isrt.obl *を使用してコンパイルする必要があります。[新規プロジェクト] ダイアログ ボックスの InstallScript オブジェクト プロジェクトタイプを使って、または [ファイル] メニューから [名前を付けて保存] コマンドを選択してプロジェクトが作成された場合、これらのファイルは、オブジェクトプロジェクトによって自動的に参照されます。オブジェクト プロジェクトがその他の方法で作成されている場合は、[ビルド] メニューから [設定] を選択します。次に、[コンパイル/リンク] タブのライブラリ (.obl) ボックスに、*Ifxobject.obl *と* Isrt.obl *がリストされていることを確認します。*

オブジェクトのビルドは、本質的には、標準インストールのビルドと同じです。オブジェクトをビルドするには、リリース ウィザードを使用します。リリース ウィザードでビルドに必要な情報すべてが、プロンプトによって聞かれます。リリース ウィザードではオブジェクトプロジェクト用に次のパネルが表示されます。

テーブル 5 ・ リリース ウィザードのオブジェクト固有のパネル

リリース ウィザード パネル	説明
[リリースの指定] パネル	新しいリリースを作成したり、既存のリリースを選択できます。
[一般オプション] パネル	プリプロセッサ変数定義とリリースの詳細プロパティを指定できます。
[プロパティの変更] ダイアログ ボックス / リリース ウィザード – [プラットフォーム] パネル	オブジェクトがサポートするオペレーティング システムを選択できます。
[セットアップ言語] パネル	[プロジェクトの設定] プロパティシートの [言語] ページで、複数の言語が選択されている場合に表示されます。オブジェクトでサポートする言語、デフォルトのオブジェクト言語、およびエンドユーザーがオブジェクトを実行する言語を選択できるようにするかどうかを選択します。
[デジタル署名] パネル	オブジェクトにデジタル署名することができます。オブジェクトにデジタル署名をすることで、オブジェクト内のコードが発表以来変更または破損していないことをエンドユーザーに対して保証します。

テーブル 5 ・ リリース ウィザードのオブジェクト固有のパネル (続き.)

リリース ウィザード パネル	説明
[ポストビルドのオプション] パネル	次回リリースをビルドした後に、ディスク イメージ フォルダーをフォルダーまたは FTP サイトにコピーするか、またはバッチ ファイルを実行します。
[リリース設定の概要] パネル	リリース情報の概要を表示します。 ***メモ・*** *オブジェクトをビルドすると、そのオブジェクトは [オブジェクト] ビューに自動追加され、それ以降に作成するすべてのインストールで使用できます。オブジェクトの以前のバージョンがギャラリーに存在する場合、新しくビルドされたオブジェクトで上書きされます。複数のメディアをビルドする場合、最新のメディアが [オブジェクト] ビューにコピーされます。*

コマンドラインからオブジェクトのコンパイルとビルドを行う

プロジェクト・ *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

コマンドラインからインストールをコンパイルおよびビルドするのと同じ形式で、コマンドラインからオブジェクトをコンパイルおよびビルドします。コマンドラインからオブジェクトをコンパイルおよびビルドする方法については、「ISCmdBld.exe を使用して、コマンドラインからリリースをビルドする」を参照してください。

オブジェクトのテストとデバッグ

プロジェクト・ *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトのテストは標準インストールのテストに類似しています。この 2 つのテストの違いは、オブジェクトは、テストするには標準インストールに含めなくてはならないということです。このため、まずオブジェクトをビルドし、次にそれをインストール プロジェクトに含めてから、インストール プロジェクトをビルドして実行する必要があります。次のデバッグ方法は、オブジェクトを開発する際に適用できます。

メモ・ *オブジェクトプロジェクトがあり、関連したインストール プロジェクトを InstallShieldの別のインスタンスで開いて、オブジェクトを変更および再ビルドした場合、それらの変更を統合するには関連したインストール プロジェクトを閉じてから再度開く必要があります。*

スクリプト デバッガー

スクリプト デバッガーて、実行するインストールのコードで手順をすすめることができます。スクリプト デバッガーでオブジェクトをデバッグするには、標準インストール プロジェクトにそのオブジェクトを含め、コンパイルし、デバッグする必要があります。もう 1 つの方法として、オブジェクトを標準インストールとしてデザイン、テスト、デバッグし、そして次の手順に従ってオブジェクトに変換することもできます。



タスク ***オブジェクトを変換するには、以下の手順を実行します。***

1. **[ファイル]** メニューで、**[名前を付けて保存]** をクリックします。**[名前を付けて保存]** ダイアログ ボックスが開きます。
2. プロジェクトのコピーの作成先となるフォルダーを選択します。新しいオブジェクトを元のプロジェクトと同じフォルダーに保存することはできません。
3. **[ファイル名]** ボックスで、新規プロジェクトの名前を入力します。
4. **[ファイルの種類]** リストで、**[InstallScript オブジェクト プロジェクト (*.ism)]** を選択します。
5. **[保存]** をクリックします。

指定した名前を持つオブジェクト プロジェクト ファイル (..ism ファイル) とすべてのプロジェクトのサブフォルダーが指定の場所に作成されます。プロジェクトが InstallShield で開きます。

詳細については、「InstallScript デバッガー」を参照してください。



メモ: *後でオブジェクトに変換することを考慮に入れてオブジェクトを標準インストールとしてデザインしている場合、次の制限があることに注意します。*

- *オブジェクトはセットアップの種類を含みません。さらに、定義するセットアップの種類やプロジェクトで呼び出すセットアップの種類関数は、オブジェクトがオブジェクトプロジェクトに変換される際に無視されます。*
- *オブジェクトでサポートされない関数やイベントが数多くあります。*
- *プロジェクトが変換される際に、ビルドされたメディアに関する情報はすべて失われるため、プロジェクトを再ビルドする必要があります。*

別のマシンでのオブジェクトのデバッグ

[**インライン デバッグ情報を生成する**] は、[ビルドの設定] ダイアログ ボックスの [コンパイル/リンク] タブにあります。オブジェクトが作成されたマシン以外のマシン上でオブジェクトをデバッグすることができます。このオプションを使用しないと、デバッグ情報は Setup.dbg という名前のファイルに配置され、その場所は Setup.inx のヘッダーに格納されます。このスキーマは、コンパイルを行ったのと同じマシン上でスクリプトをデバッグする場合、問題はありません。しかし、別のマシンを使用する場合、.dbg ファイルは同じ場所に保管されない可能性があります。この問題を緩和するために、[**インライン デバッグ情報の生成**] オプションはデザインされました。このオプションを使用する場合、デバッグ情報はすべて Setup.inx に保管されます。このため、インストールがデバッグ情報を検索する必要はありません。

下記の 2 つの理由により、最後の製品と共に出荷する Setup.inx をコンパイルする前に **[インライン デバッグ情報の生成]** オプションをクリアすることが推奨されます。1 つ目の理由は、デバッグ情報は Setup.inx のサイズを増やし、インストール実行にかかる時間をさらに遅くします。2 つめは、インラインデバッグ情報によって、他人がコードを容易にリバースエンジニアリングしてしまう可能性があります。

InstallShield キャビネット & ログ ファイル ビューアーを使用する

InstallShield キャビネット & ログ ファイル ビューアーを使って、以下を行います:

- InstallScript キャビネット ファイル (.cab) または InstallScript ヘッダー ファイル (.hdr)、およびそれらの圧縮ファイル、レジストリ エントリ、コンポーネント、機能、その他のデータ を参照する。このツールを使って、.cab ファイルからファイルを抽出することもできます。
- InstallScript インストールが作成した InstallScript ログ ファイル (.ilg) を参照する。このツールを使って、インストールがログ ファイルに記録した内容を参照できます。ログ ファイルには、重要なアンインストール情報がバイナリ形式で保存されます。

詳細については、「InstallShield キャビネット & ログ ファイル ビューアー」を参照してください。

カスタム デザインタイム ウィザードのデバッグ

オブジェクトに対してカスタム デザインタイム ウィザードを作成した場合、そのウィザードをデバッグする必要があります。

タスク ***ウィザードをデバッグするには、次の操作を行います。***

1. 開発環境からウィザードのデバッグバージョンを起動し、ブレークポイントを希望の場所に設定します。
2. InstallShield インターフェイスで、別のプロジェクトの機能にオブジェクトを挿入してオブジェクトのウィザードを起動します。
3. 開発環境が Microsoft Visual Basic の場合、次の操作を行う必要があります。
   a. インターフェイスを最小化します。**[サーバー使用中]** メッセージ ボックスが開きます。(メッセージ ボックスが開く前に、タイトルバーの最小化ボタンを何度かクリックしなければならないこともあります。)
   b. **[サーバー使用中]** メッセージ ボックスで、**[切り替え]** ボタンをクリックするか Enter キーを押します。

オブジェクトの配布

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトの主な利点は、オブジェクトがマシン上にインストールされて登録されている限り、InstallShield インストールを作成している人なら誰でも最低限の作業でインストールにオブジェクトを含めることができるという点です。オブジェクトを登録する最も簡単な方法は、InstallShield オブジェクト インストーラーを使用することです。このオブジェクトを使うと、InstallShield X 以降、InstallShield DevStudio、または、InstallShield Professional 6.1 以降があれば配布するオブジェクトをパッケージにしてターゲットマシンに登録することができます。

タスク ***インストールに InstallShield オブジェクト インストーラーを追加するには、以下の手順を実行します。***

1. **[オブジェクト]** ビューを開きます。**[InstallShield オブジェクト/マージ モジュール]** ペインには、使用可能なすべてのオブジェクトが一覧表示されます。
2. **[機能]** ペインから InstallShield オブジェクト インストーラーを追加する機能を選択します。
3. **[InstallShield オブジェクト インストーラー]** を右クリックして、**[選択された機能に追加]** を選択します。

関連したウィザードが表示されるので、これに従ってカスタマイズします。



メモ: *配布するオブジェクトが InstallShield オブジェクトギャラリーに存在する必要があります。これが存在しない場合、そのオブジェクトを登録する必要があります。詳しくは、「InstallScript プロジェクトでオブジェクトを登録する」をご覧ください。*

オブジェクトのスクリプト



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

InstallShieldでは、追加の関数を定義することでオブジェクトのサポートをInstallScript ビューにスクリプトを追加することでオブジェクト プロジェクトでサポートされていない定数のサポートを拡張することができます。基本的に、InstallShield インターフェイスでサポートされている関数を強化するスクリプトを書くことができです。

オブジェクト プロジェクトでサポートされていない関数



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

以下に、オブジェクト プロジェクトでサポートされていない InstallScript 関数の一覧を示します。

- OnFileReadOnly
- OnRemovingSharedFile
- OnFileLocked
- OnNextDisk
- OnMD5Error
- OnFileError
- OnComponentError
- SdSetupType
- SdSetupTypeEx
- SetupType
- ComponentSetupTypeSet
- ComponentSetupTypeEnum
- ComponentSetupTypeGetData

オブジェクト プロジェクトでサポートされていない定数



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

以下に、オブジェクト プロジェクトでサポートされていない InstallScript 定数の一覧を示します。

- IFX_ONNEXTDISK
- IFX_ONNEXTDISK_MSG
- IFX_MAINTUI_MSG
- IFX_ONMAINTUI_CAPTION
- IFX_SDFINISH_MSG1
- IFX_SDFINISH_MAINT_MSG1
- IFX_SDFINISH_MAINT_TITLE

オブジェクト プロジェクトでのみサポートされている関数



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

以下に、オブジェクト プロジェクトでのみサポートされている InstallScript 関数の一覧を示します。

- SetStatus
- WizardDirection

ScriptDefinedVar プロパティをオブジェクトに追加する



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

InstallShield のオブジェクトには、ScriptDefinedVar プロパティを含むものがあります。このプロパティで、デザイン時にパス (例、**<MYSERVICEFOLDER>¥MyService.exe**) を定義するのに使用するスクリプト定義の変数値を実行時に設定することができます。たとえば、以下のようなメイン スクリプトのコードを使うことにより、エンド ユーザーは*<MYSERVICEFOLDER>* の値を指定することができます:

```
/* 名前を付けたオブジェクトへの参照を取得します。
オブジェクト名は [コンポーネント] ペイン の「新しい カスタムオブジェクト 1」 です。*/
set oObj = GetObject("New Custom Object 1");

/* エンドユーザーからフォルダーの選択を取得します。*/
svDir = TARGETDIR;
AskDestPath ( "" , "サービスファイルの場所を指定してください:" ,
    svDir , 0 );

/* 希望する <MYSERVICEFOLDER> の値をオブジェクトに伝えます。*/
oObj.ScriptDefinedVar("<MYSERVICEFOLDER>") = svDir;
```



タスク ***オブジェクト スクリプトに ScriptDefinedVar プロパティを含めるには、下記の手順を実行します。***

1. **InstallScript** ビューを開きます。
2. **InstallScript** エクスプローラーで、**[プロパティ]** を右クリックして、**[新しいプロパティの追加]** をクリックします。**[新しいプロパティの追加]** ダイアログ ボックスが開きます。
3. 以下をエントリを行ってください。
   c. **[プロパティ名]** ボックスで、次のように入力します。

      ScriptDefinedVar

   d. **[データ型]** リストで、**[文字列]** を選択します (プロパティへ文字列データを書き込むため)。
   e. **[アクセス メソッド]** リストで、**[読み取り/書き込み]** を選択します。(オブジェクトのユーザーがメイン スクリプトから現在のプロパティ値を確認しないことが明確な場合は、代わりに **書き込み専用** を選択できます。)
4. **[OK]** をクリックしてダイアログ ボックスを閉じ、必要なコードをすべてオブジェクト スクリプトに自動追加します。
5. 自動追加されたプロパティのスクリプト コードに対して、次の変更を行います。
   a. **InstallScript** ビューを開きます。
   b. **InstallScript** エクスプローラーの **[プロパティ]** の下にある **ScriptDefinedVar** をダブル クリックして、スクリプト エディター ウィンドウのキャレットを ScriptDefinedVar 宣言に移動します。宣言を以下のように変更して、宣言に文字列パラメーターを追加します:

      ```
      property(get,put) STRING ScriptDefinedVar();
      ```

      変更後、

      ```
      property(get,put) STRING ScriptDefinedVar( STRING ) へ変更。
      ```

   c. **InstallScript** エクスプローラーの **[関数]** の下にある get_ScriptDefinedVar をダブル クリックして、スクリプト エディター ウィンドウのキャレットを **get_ScriptDefinedVar** 関数ブロックに移動します。次の関数ブロックを検索します。

      ```
      function STRING get_ScriptDefinedVar()
      begin
         return m_strScriptDefinedVar;
      end;
      ```

      次のように変更します。

      ```
      function STRING get_ScriptDefinedVar( szScriptVar ) /* 文字列引数を追加します。*/
      begin
         return TextSub.Value( szScriptVar ); /*  szScriptVar に関連した値を取得します。*/
      end;
      ```

   d. **InstallScript** エクスプローラーの **[関数]** の下にある **put_ScriptDefinedVar** をダブル クリックして、スクリプト エディター ペインのキャレットを **put_ScriptDefinedVar** 関数ブロックに移動します。次の関数ブロックを検索します。

      ```
      function void put_ScriptDefinedVar(newVal)
      begin
         m_strScriptDefinedVar = newVal;
      ```

```
end;
```

次のように変更します。

```
function void put_ScriptDefinedVar( szScriptVar, newVal ) /* 文字列引数を追加します。*/
begin
    TextSub.Value( szScriptVar ) = newVal; /* newVal を szScriptVar に関連付けます。*/
end;
```

オブジェクトでシステム変数を使用する



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

システム変数は、ソースパス、ターゲットパス、Windows フォルダー、および Windows システムフォルダーのような情報を含む、あらかじめ定義された変数です。スクリプト中でこれらの変数を定義することはできません。InstallShield では、インストール プロセスの開始時に自動的にシステム変数が初期化されます。

オブジェクト内のシステム変数を変更したり初期化したりすることは可能ですが、お勧めしません。オブジェクト内のシステム変数を変更すると、その影響はインストール全体に及びます。たとえば、オブジェクトによって TARGETDIR に特定の値が初期値として設定された場合、その値は、インストール プロジェクト内ですでに設定された値を上書きしてしまいます。従って、インストール全体にわたって、ファイルはインストール開発者によって指定されたディレクトリではなくオブジェクトで指定されたディレクトリへインストールされることになります。

スクリプトを使用してオブジェクトのインストール先を設定する



プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

スクリプト定義の変数を使用して、オブジェクトコンポーネントのインストール先のプロパティの一つを定義できます。そのためには、オブジェクトのユーザーは、スクリプト中で **ComponentCompareSizeRequired** が呼び出される前に、スクリプト定義の変数値を設定する必要があります。そうでない場合は、関数は不充分なディスク容量を報告します。次のうちのいずれかまたは両方の方法で、スクリプト定義の変数値を設定することができます。

- 書き込み専用、または読み取り/書き込みオブジェクトのプロパティの put_ function が **ComponentSetTarget** を呼び出すようにする。以下に例を示します。

  ```
  function VOID put_SetTargetDest ( szScriptVar , szValue )
  begin
      ComponentSetTarget ( MEDIA , szScriptVar , szValue );
  end;
  ```

- オブジェクトの OnFirstUIBefore イベント ハンドラーによって、エンドユーザーのダイアログを表示させる。

7

開発プロセスを再利用/分担するためのインストール プロジェクトのモジュール化



プロジェクト: *この情報は、DIM プロジェクトに適用します。*

InstallShield のデベロッパー インストール マニフェスト (DIM) は、コラボレーションを促進したいエンジニア チームのために提供されています。DIM プロジェクトを使うと、組織で、インストール開発を分担して、効率化を図ることができます。DIM は、機能サイズのプロジェクトで、インストール パッケージの論理的に別個に分かれている部分を構成する製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、および要素など、関連するアイテムを集めたものです。以下は、DIM を利用した時のいくつかの利点です:

- DIM を利用することにより、複数のチーム メンバーが、インストールの開発を同時に携わることができます。各ソフトウェア開発者またはチームメンバーは、異なる DIM について個別で作業することができ、リリース エンジニアは、1 つまたは複数のインストール プロジェクトでそれらを参照することができます。
- リリース エンジニアは、DIM を複数のインストール プロジェクトで繰り返し使用できるため、非常に効率的です。
- DIM では、基本の MSI プロジェクトで提供されている機能とほぼ同じ機能がサポートされています。このため、DIM の作成者には、インストールの部分開発を行うために必要な柔軟性がすべて提供されています。

この章では、InstallShield における DIM プロジェクトの使い方について説明し、次のトピックが含まれています:

- DIM のインストール情報を指定する
- DIM でパス変数を使用する
- DIM のファイルを編成する
- DIM のターゲット システムの構成
- DIM のインストール動作をカスタマイズする
- DIM のサーバーを構成する

DIM のインストール情報を指定する



プロジェクト: *この情報は、DIM プロジェクトに適用します。*

InstallShield は、プロジェクトの設定を、単一インストール プロジェクト ファイル (.dim) に保管します。このファイルには、DIM プロジェクトについてのすべての情報が含まれています。[一般情報] ビューでは、作成者名、プロジェクトについての内部メモ、リリース エンジニアが DIM をインストール プロジェクトに含める時に留意すべき手順の説明など、DIM についての基本的な情報を編集できます。

このセクションには、次のトピックが含まれています:

- プロジェクト ファイルを XML またはバイナリ形式で保存する
- DIM のデフォルトのインストール先フォルダーを指定する

プロジェクト ファイルを XML またはバイナリ形式で保存する



プロジェクト: *この情報は、DIM プロジェクトに適用します。*

InstallShield では、.ism プロジェクトファイルを XML またはバイナリ形式で保存することができます。



タスク

プロジェクト ファイルの形式を指定するには、以下の手順に従います:

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. ***"プロジェクト ファイルの形式"*** 設定で、適切なオプションを選択します: 選択可能なオプションは以下のとおりです:
   - **バイナリ** – プロジェクト ファイルをデータベース ファイルとして保存するには、このオプションを選択します。プロジェクトを開いたり保存したりする際のスピードは、この形式が最も早いです。
   - **XML** – プロジェクト ファイルを階層構造を持つテキスト ベースの形式で保存するには、このオプションを選択します。このプロジェクト ファイル形式は、ソース コード管理システムでの使用に最適です。



メモ: *DIM プロジェクト ファイル (.dim) を XML またはバイナリ形式で保存すると、プロジェクト ファイルは .ism 拡張子を保持します。*

DIM のデフォルトのインストール先フォルダーを指定する



プロジェクト: *この情報は、DIM プロジェクトに適用します。*

[一般情報] ビューの "INSTALLDIR" 設定に入力された値は、DIM のすべてのファイルのデフォルト フォルダーとなります。したがってその値は、デフォルトのインストール先フォルダーである、Windows Installer フォルダー プロパティ INSTALLDIR に割り当てられます。インストール プロジェクトでは、通常これは次のように評価されます:

[ProgramFilesFolder]会社名¥*製品名*

この DIM のユーザーが、デフォルトのインストール先を上書きできるようにするには、"INSTALLDIR" 設定に **[TARGETDIR]** を入力します。詳細については、「DIM リファレンスのインストール先のオーバーライド」を参照してください。

DIM でパス変数を使用する

プロジェクト: *この情報は、DIM プロジェクトに適用します。*

DIM プロジェクトに製品ファイルを追加すると、デフォルトで、製品ファイルのソースとして使用されるパス変数が作成されます。パス変数は、プロジェクトのソース ファイルを含むディレクトリを表すために使われる変数です。パス変数はデザイン時とビルド時の両方で使用されますが、インストールがエンドユーザーのマシンで実行されるインストールの実行時には使用できません。

ハードコード化されたパスの代わりにパス変数を使う利点として、たとえばソール ファイルが開発システム上にある別のディレクトリに移された場合、パス変数の値を変更するだけで済みます。つまり、ハードコード化されたすべてのソース ファイルのパスをそれぞれ更新する必要はありません。

DIM プロジェクトでパス変数を使用すると、ソース ファイルの管理を簡略化できます。特に、DIM プロジェクトが DIM プロジェクトを含める基本の MSI プロジェクトとは異なる場所にある場合、これは有用です。

また、すべてのコードと InstallShield プロジェクト ファイルを格納するためにソース コード管理を使っている場合や、ソース コード管理から適切なファイルを取得し、コードをコンパイルし、インストールをビルドするビルド マシンを使用している場合も、パス変数は有用です。このようなシナリオでは、異なるリリース エンジニアや開発者が、ソース管理から最新のファイルを取得するために別のローカル作業ディレクトリを使うことがあります。パス変数を使って場所を定義すると、ソースの場所が異なるために発生する問題を最小限に抑えることができます。

DIM プロジェクトにおける定義済みパス変数

以下は、DIM プロジェクトにおける定義済みパス変数、および通常それらに設定される値です。

テーブル 1 ・ DIM プロジェクトにおける定義済みパス変数

定義済みパス変数	通常値
CommonFilesFolder	C:¥Program Files¥Common Files¥
ISProductFolder	C:¥Program Files¥InstallShield¥2022¥
ISProjectDataFolder.DIM_GUID	<ISProjectFolder>¥ProjectName¥
ISProjectFolder.DIM_GUID	C:¥InstallShield 2022 Projects¥

テーブル 1・DIM プロジェクトにおける定義済みパス変数 (続き.)

定義済みパス変数	通常値
ISRedistPlatformDependentExpressFolder	C:¥Program Files¥InstallShield¥2022¥Redist¥Language Independent¥i386 Express
ISRedistPlatformDependentFolder	C:¥Program Files¥InstallShield¥2022¥Redist¥Language Independent¥i386
ProgramFilesFolder	C:¥Program Files¥
SystemFolder	C:¥Windows¥System32¥
WindowsFolder	C:¥Windows¥

パス変数に *DIM_GUID* が含まれる両方のケースでは、*DIM_GUID* は、DIM プロジェクトの [一般情報] ビューの "DIM GUID" 設定で定義される識別子になります。次は、DIM プロジェクトにおけるサンプル パス変数です:

- **ISProjectDataFolder.8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC**
- **ISProjectFolder.8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC**
- **MyNewPathVariable.8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC**

GUID は、DIM プロジェクトからの適用可能なパス変数を判別するのに役立ちます。また、同じパス変数が、ある DIM プロジェクトとその DIM を含む基本の MSI プロジェクトで使用されていて、異なる値がこれらの 2 つのパス変数に割り当てられている場合に発生する競合を最小化または削除する必要がある時にも役立ちます。

InstallShield*SystemFolder* および *WindowsFolder* など、GUID を含まないパス変数名は、で開いているプロジェクト専用ではないため、すべてのマシンで使用可能です。これらの種類のパス変数の値は、オペレーティング システムに応じて異なる場合があります。たとえば、32 ビット マシンでは、**ProgramFilesFolder** の値は C:¥Program Files¥ の可能性があります。一方、64 ビット マシンでは、C:¥Program Files (x86)¥ の可能性があります。

パス変数を参照するための構文

パス変数が InstallShield で使用されている場合、パス変数は、山かっこで囲まれます。例:

<ISProjectDataFolder.8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC>

場合によっては、別のパス変数の相対的になるようにパス変数を定義することできます。たとえば、**MySourceFiles** という名前の標準パス変数を作成したと仮定します。そしてこの時、**BinFiles** という名前の別のパス変数を使って、先の **MySourceFiles** ディレクトリ内にある **Bin** フォルダーへのパスも定義する必要もあると仮定します。この場合、**BinFiles** 変数の Defined Value 列に相対パス変数を入力する時、 でベースのパス変数を囲む必要があります。したがって、**BinFiles** 変数の Defined Value 列は、次のようになります:

<MySourceFiles>¥Bin

MySourceFiles の値が C:¥CheckoutRoot¥ だとすると、BinFiles の値は、C:¥CheckoutRoot¥Bin¥ になります。

サンプル パス変数のシナリオ

1 台またはそれ以上のマシンを使って InstallShield プロジェクトの作成/編集を行っているが、インストールのビルドには別のマシンを使ってるというような環境では、プロジェクトでパス変数をどうしようするか前もって考えておく必要があります。そうしなかった場合、InstallShield でインストールをビルドした時、ファイルが不足しているためにビルド エラーが発生したり、間違ったファイルがビルドに取り込まれたりします。

次の例では、InstallShield プロジェクトでパス変数を使う時の、いくつかの異なる方法が説明されています。最も容易な方法は、例 3 で説明されている方法です。

例 1: DIM を含む基本の MSI プロジェクトで DIM のパス変数の値を上書きする

次のテーブルでは、DIM プロジェクトで、1 つの定義済みパス変数と、いくつかの標準パス変数がどう使用されているかが説明されています。

テーブル 2 ・ 例 1 DIM プロジェクトにおけるパス変数

名前	定義された値	現在の値
ISProjectFolder.4E846FD3_5307_4CF5_9FF1_C476E5505666	(これは、InstallShield プロジェクト ファイルを含むフォルダーの定義済みパス変数です。この値は、手動で定義することはできません。)	C:¥Source¥Units¥Utility¥
CHECKOUT_ROOT.4E846FD3_5307_4CF5_9FF1_C476E5505666	C:¥Source	C:¥Source
PATH_TO_RELEASE_FILES.4E846FD3_5307_4CF5_9FF1_C476E5505666	<CHECKOUT_ROOT.4E846FD3_5307_4CF5_9FF1_C476E5505666>¥SpellLib¥Bin¥Release	C:¥Source¥SpellLib¥Bin¥Release
PATH_TO_DICTIONARIES.4E846FD3_5307_4CF5_9FF1_C476E5505666	<CHECKOUT_ROOT.4E846FD3_5307_4CF5_9FF1_C476E5505666>¥Dictionaries	C:¥Source¥Dictionaries

この例では、DIM プロジェクトの作成者は、パス変数 **CHECKOUT_ROOT.4E846FD3_5307_4CF5_9FF1_C476E5505666** を、自分のソース ルート フォルダー **C:¥Source** (ソース管理からファイルをチェックアウトしているマシン上の作業ディレクトリ) に割り当てています。ソール ファイルを含む他のフォルダーのパス変数は、このソース ルートパス変数に相対的に定義されています。

リリース エンジニアが、ビルド マシン上の異なる作業ディレクトリを使っている場合、場合によって、DIM を含むプロジェクトで基本の MSI インストールをビルドする時、このパス変数の値を上書きする必要があります。リリース エンジニアは、コマンドラインからインストールをビルドする時、-I パラメーターを ISCmdBld.exe に渡すことで、ソース ルート パス変数の新しい値を設定することができます。他のパス変数の値は、ソース ルート パス変数を使って定義されるため、これらの他のパス変数の値は、それに沿って更新されます。

この方法は、上書きを必要とするパス変数をそれぞれ含む複数の DIM プロジェクトがある基本の MSI プロジェクトのインストールをビルドする場合など、場合によって、手間がかりエラーが発生しやすいという欠点があるで注意が必要です。

例 2: ビルド環境にしたがって、DIM パス変数の値を変更する

DIM 開発マシンのソール ルート パスが、基本の MSI をビルドしているマシン上の対応するパスと一致しない場合、DIM を直接開いて、パス変数の値を必要に応じて更新することができます。DIM プロジェクトを基本の MSI プロジェクト内から開く方法については、「インストール プロジェクト内から参照された DIM プロジェクトを開く」を参照してください。

この方法は、テスト目的で毎日ビルド処理を実行している環境で、DIM プロジェクトにまだ変更を行っている場合、理想的ではありません。

例 3: ISProjectFolder.DIM_GUID のような定義済みパス変数に相対するパス変数を使用する

DIM プロジェクト内のパス変数を管理する最も簡単な方法は、**ISProjectFolder** などの定義済みパス変数に相対するパスにソール ファイルを常に保存しておくことです。**DIM_GUID**実際の手順を以下で説明します。

テーブル 3 · 例 3 DIM プロジェクトにおけるパス変数

名前	定義された値	現在の値
ISProjectFolder.4E846FD3_5307_4CF5_9FF1_C476E5505666	(これは、InstallShield プロジェクト ファイルを含むフォルダーの定義済みパス変数です。この値は、手動で定義することはできません。)	次は、考えられる開発マシンにおける現時点での値です: C:¥Source¥Units¥Utility¥ 次は、考えられるビルド マシンにおける現時点での値です: C:¥BuildWorkspace¥Source¥Units¥Utility¥
CHECKOUT_ROOT.4E846FD3_5307_4CF5_9FF1_C476E5505666	<ISProjectFolder.4E846FD3_5307_4CF5_9FF1_C476E5505666>¥..¥..  ***メモ:*** *このパスの値に定義済みのプロジェクト フォルダー パス変数を使用して、それから他のパス変数の値にチェックアウト ルート パス変数を使用すると、InstallShield プロジェクトおよびその他すべてのソース ファイルが、マシンからマシンに移動できるポータブルになります。*	次は、考えられる開発マシンにおける現時点での値です: C:¥Source 次は、考えられるビルド マシンにおける現時点での値です: C:¥BuildWorkspace¥Source
PATH_TO_RELEASE_FILES.4E846FD3_5307_4CF5_9FF1_C476E5505666	<CHECKOUT_ROOT.4E846FD3_5307_4CF5_9FF1_C476E5505666>¥SpellLib¥Bin¥Release	次は、考えられる開発マシンにおける現時点での値です: C:¥Source¥SpellLib¥Bin¥Release 次は、考えられるビルド マシンにおける現時点での値です: C:¥BuildWorkspace¥Source¥SpellLib¥Bin¥Release

テーブル 3 - 例 3 DIM プロジェクトにおけるパス変数 (続き.)

名前	定義された値	現在の値
PATH_TO_DICTIONARIES.4E846FD3_5307_4CF5_9FF1_C476E5505666	<CHECKOUT_ROOT.4E846FD3_5307_4CF5_9FF1_C476E5505666>¥Dictionaries	次は、考えられる開発マシンにおける現時点での値です: C:¥Source¥Dictionaries 次は、考えられるビルド マシンにおける現時点での値です: C:¥BuildWorkspace¥Source¥Dictionaries

チェックアウト ルート パス変数の値に定義済みのプロジェクト フォルダー パス変数を使用して、それから他のパス変数の値にチェックアウト ルート パス変数を使用すると、InstallShield プロジェクトおよびその他すべてのソース ファイルが、マシンからマシンに移動できるポータブルになります。

DIM のファイルを編成する



プロジェクト: *この情報は、DIM プロジェクトに適用します。*

DIM プロジェクトでは、ファイルは階層構造で編成されます。インストールの自分の担当の部分のファイルをコンポーネントに追加します。DIM を基本の MSI プロジェクトに含めると、DIM のコンポーネントは、基本の MSI プロジェクトの 1 つまたは複数の機能に追加されます。エンドユーザーがインストールする機能および除外する機能を選択できるようにした場合、基本の MSI インストールが実行された時、。エンドユーザーはインストールする機能を選択できるようになります。

DIM プロジェクトでコンポーネントにファイルを追加する場合、セットアップ ベスト プラクティスに留意する必要があります。デフォルトでは、DIM ベストプラクティス ウィザードがセットアップのデザインを監視し、ベストプラクティスに反した場合には警告を表示します。

ファイルの編成についての詳細は、ドキュメントの次のセクションを参照してください:

- ファイルとフォルダーを含める
- コンポーネントを使用する
- セットアップ ベスト プラクティス

DIM のターゲット システムの構成



プロジェクト: *この情報は、DIM プロジェクトに適用します。*

インストールでは、ターゲット システムの変更が必ず発生します。簡単なインストールでは、ファイルをコピーするだけのものもあります。より複雑なインストールでは、レジストリの変更、.ini ファイルの編集、ショートカットの作成、ODBC リソースの構成、環境変数の使用、XML ファイルの変更、およびテキスト ファイルの変更が行われます。DIM プロジェクトでこれらの種類の構成に関する詳細は、ドキュメントの次のセクションを参照してください:

- ショートカットおよびプログラム フォルダーの作成
- レジストリの編集
- .ini ファイル データの変更
- ODBC リソースの構成
- 環境変数を使用する
- XML ファイルの変更
- テキスト ファイルの変更

DIM のインストール動作をカスタマイズする



プロジェクト: *この情報は、DIM プロジェクトに適用します。*

インストール作成の重要な要素は、それをエンドユーザーのニーズに合わせてカスタマイズすることです。たとえば、Windows Installer が直接サポートしていない機能を追加するカスタム アクションを作成するのに役立ちます。また、検索の結果によって、インストールが続行できるかどうかが判別される場合など、ターゲット システムでインストール済みデータを検索する必要が出てくる場合もあります。Windows Installer のプロパティを使用すると、プロジェクトに対してプロジェクト全体の値を使用できます。DIM プロジェクトのインストール動作のカスタマイズに関する詳細は、ドキュメントの次のセクションを参照してください:

- カスタム アクションを使用
- インストールされたデータの検索
- Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方

DIM のサーバーを構成する



プロジェクト: *この情報は、DIM プロジェクトに適用します。*

DIM プロジェクトを作成しているとき、ターゲット システムにインストールされるテクノロジに対してサーバー側のサポートを提供する必要があることに気がつくことがあります。InstallShield では、インターネット インフォメーション サービス (IIS) Web サイトの作成と管理、COM+ アプリケーションおよびコンポーネントの管理、およびサーバー接続および設定による SQL スクリプトの管理および編成がサポートされています。DIM プロジェクトのサーバーの構成に関する詳細は、ドキュメントの次のセクションを参照してください:

- SQL サポートの構成
- COM+ アプリケーションとコンポーネントの管理
- インターネット インフォメーション サービス

8

アプリケーションのアップデート



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*
- *QuickPatch*

アプリケーションのアップグレードをインストールすることは、アプリケーションのオリジナルリリースのインストールよりもはるかに一般的な操作です。この意味で、効率的で、信頼のおけるアップグレードを作成することは、とても重要なタスクです。

[アップデートの更新] は、異なるタイプのアップグレードについての確かなバックグラウンド情報が提供されており、一般的なパッチについての誤解が分かりやすく説明されています。"アプリケーションの更新" では、また、製品に最適なアップグレード ソリューションを判断するのに役に立ち、アップグレード、パッチ、QuickPatch プロジェクト、差分リリース、および、完全リリースの作り方を手順を追って案内します。またこの章には、FlexNet Connect を利用してエンドユーザーに製品の新しいバージョンのリリースついて通知する方法について説明する次のトピックが含まれています:

- アップグレードの概要
- 最適なアップグレード ソリューションの決め方
- アップグレード、パッチ、および QuickPatch プロジェクトを使用する

アップグレードの概要



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*
- *QuickPatch*

ソフトウェア アプリケーションの保守は、開発にかかったコストよりも高くかかる場合があります。そのため、効率的で信頼のおけるアップグレードを作成することは重要なタスクです。アプリケーションに対する堅牢なアップグレードの配布が可能かどうかは、オリジナル インストール パッケージがどのように構成され、配布されたかによって左右されます。ヘルプトピックには、堅牢なアップグレード パッケージ作成の基盤となる情報が掲載されています。

基本の MSI プロジェクトと InstallScript MSI プロジェクトのアップグレード方法

Windows Installer は 3 種類 (スモール アップデート、マイナー アップグレード、およびメジャー アップグレード) の製品アップグレードをサポートしています。アップグレードは、完全インストールまたはパッチとしてパッケージすることができます。パッチはアップグレードを実装するための 1 つの仕組みに過ぎません。ただし完全リリースとは違って、パッチはインストール済みのファイルを最新版に変更するために必要な部分のみをユーザーに配布します。

MSI データベース、およびトランスフォーム プロジェクトのアップグレード方法

MSI データベースおよびトランスフォーム プロジェクトは、Windows Installer のメジャー アップグレードをサポートします。

InstallScript プロジェクトのアップグレード方法

InstallScript インストール用には、完全リリース パッケージ、または差分リリースのどちらかを使ってアプリケーションを更新することができます。最適なアップグレード方法を決定するには、ファイルサイズ、以前のリリースからのファイルの有無など様々な要素を考慮する必要があります。

メモ: *InstallShield Professional 6.0 よりも古いバージョンを利用して InstallScript プロジェクト用のメディアを作成した場合、メディアをアップデートすることはできません。*

このセクションには、次のトピックが含まれています:

- メジャーアップグレード
- マイナーアップグレード
- スモール アップデート
- パッチの適用
- QuickPatch プロジェクト
- 差分リリース
- 完全リリース

メジャーアップグレード



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

メジャーアップグレードでは製品の変更規模が大きいため、パッケージコードだけでなく製品バージョン番号および製品コードの両方を変更する価値があります。たとえばバージョン 1.2 製品の 2.0 へのアップデートです。メジャーアップグレードは、以前のバージョンが存在しない場合は初回インストールと同様に作動します。以前のバージョンが存在する場合、メジャーアップグレードは通常それをアンインストールしてから新しいバージョンをインストールします。また、メジャーアップグレードにまず以前のバージョンを上書きインストールさせてから、使われないファイルを削除することも可能です。方法としては、この方がより効率的ですが、コンポーネント作成規則の厳密な順守が必要です。

メジャーアップグレードの動作の仕方

Windows Installer のヘルプで示唆されているように、メジャー アップグレードを行うにはインストールの最新バージョンの製品コードを変更する必要があります。



メモ- *インストールの製品コードは、[一般情報] ビューで設定できます。新しい製品コードは、一意である限り、特に制約はありません。*

一意の製品コードを使用することで、エンド ユーザーのマシン上で、最新のインストールを Windows Installer サービスに独自に登録することができます。言い換えるとこれは、何かしらの手段が執られないかぎり、Windows Installer は、以前のバージョンとは無関係に製品の最新のバージョンをインストールするということを意味します。これは、2 つの異なるバージョンの製品が同じマシン上に共存できる場合は問題はありませんが、アップグレードされたことにはなりません。したがって、便宜上、ここでは 2 つのバージョンが同じマシン上で共存できないと仮定します。

最新インストールの製品コードを更新した場合、[アップグレード] ビューを使って、アップグレードを行う以前のすべてのバージョンについての情報を指定します。

メージャー アップグレードの作成に関する詳細については、"メジャー アップグレードを作成する" ヘルプを参照してください。

実行時のメージャーアップグレード

製品の以前のバージョンがターゲット マシン上に存在しない場合にエンド ユーザーがメジャー アップグレードを実行すると、初回インストールとしてインストールが行われます。

製品の以前のバージョンがターゲット マシン上に存在している状態でエンド ユーザーがメージャー アップグレードを実行すると、エンドユーザーはあたかも以前のインストールが存在しないマシン上に最新のアプリケーションをインストールするかのような印象を受けます。唯一の異なる点は、新しいリソースがインストールされる前に、インストールはまずターゲット マシンから以前のバージョンをアンインストールするということです。

この削除過程は、[セットアップ進行状況] ダイアログの進行状況バーで表示され、エンドユーザーはアンインストールの処理状況を直接見ることができます。以前のインストールの削除が完了すると、最新のインストールからのリソースがターゲット マシンにインストールされます。

このタイプのアップグレードは、つまり、完全アンインストールとそれに続くアプリケーションに関連付けられたリソースすべての再インストールからなります。したがって、エンドユーザーによって構成されたアプリケーションのためのすべてのデータはエンドユーザーのマシーンから完全に削除される可能性があります。エンドユーザー データの一部を残しておく必要がある場合、このデータのバックアップをとってから、新しいデータのインストールが完了した後、それを置き換えるカスタム アクションを作成してください。

マイナーアップグレード



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

製品データベースおよびファイルへの変更規模が ProductCode プロパティを変更するほどではないけれども、ProductVersion プロパティを変更をするだけの価値があるとき、これをマイナー アップグレードといいます。つまり、マイナーアップグレードの場合、パッケージコードおよび製品バージョン番号両方とも以前のインストールパッケージと異なりますが、製品コードは変わりません。バージョン 1.1 の製品からバージョン 1.2 へアップデートはひとつの例です。マイナーアップグレードでは、通常、異なるバージョンの間でインストールの構成において重要な変更は行われません。フル インストールとしてパッケージされたマイナー アップグレード は、以前のバージョンが存在しない場合は初回インストールと同様に動作しますが、製品が既にインストールされている場合、その上からインストールを行います。既存のインストールをアップグレードする場合、マイナーアップグレードは基本的にバージョン 1.2 と 1.1 のアプリケーションの差分のみをインストールします。



メモ: *InstallScript MSI マイナー アップグレード中、製品のベース バージョンがデフォルト以外の場所にインストールされていて、新しいコンポーネントが INSTALLDIR のサブフォルダ—に追加された場合、新しいコンポーネント ファイルは、ユーザーが指定したサブフォルダ—の場所ではなく、デフォルトの場所にインストールされます。*

回避策については、「InstallScript MSI Installs to Default Location During Minor Upgrades (マイナー アップグレード中に IInstallScript MSI を既定の場所にインストールする)」を参照してください。



ヒント: *アップグレードの製品バージョンは、[一般情報] ビューで変更できます。*

スモール アップデート



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

本質的にスモール アップデートは、インストール済みのアプリケーションのいくつかのファイルを変更するのに使用されるアップデート タイプで、一般的に小さいバグ修正の配布に使用されます。スモール アップデートには、ホットフィックスなどの製品バージョンの変更が必要なほど大きくはない変更が含まれます。スモールアップデートでは、パッケージコードの変更が必要です。

3.0 以前のバージョンの Windows Installer と共にインストールされるスモールアップデートの短所は、製品のより新しいバージョンのためのインストーラーをはじめとする、外部プログラムは、オリジナルバージョンとアップデートされたバージョンを区別することができないということです。また、3.0 以前のバージョンの Windows Installer は、スモールアップデートを正しい順序で適用することができないこともあります。

パッチの適用



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

パッチは、以前のバージョンの Windows Installer インストールパッケージをアップデートしてアプリケーションをアップデートするための効率的な仕組みです。パッチはインストール済みのファイルを新しいファイルへ変更するのに必要な部分のみをカスタマーに配布します。パッチが優れている理由のひとつは、アップグレードを配布するために必要なアップグレードパッケージのサイズを、完全インストール パッケージのそれと比べて格段に小さく収めることが可能な点です。アップグレードパッケージをできるだけ小さくすることで、インターネットを通してアップグレードを簡単に配布することが可能になります。

但し、パッチが必ずしも最善のソリューションとは限らないので注意してください。たとえば、インストールを圧縮形式から非圧縮形式に変更する場合、またはその逆を行なう場合、アップグレードはパッチではなく完全インストールとしてパッケージしなくてはなりません。アップグレードの最適なパッケージ オプションの選び方については、「アップグレードのパッケージ オプション」を参照してください。

パッチはパッチパッケージ (.msp) ファイルの形式で配布され、これをエンドユーザーがインストール済みの製品に適用します。パッチパッケージは、インストールの旧バージョンをその数に関わらずアップデートすることができます。パッチパッケージには、指定した以前のバージョンをアップデートするための個別のトランスフォームおよび指示が含まれています。

パッチ作成の重要な側面は、パッチパッケージが作成されるパラメーターを定義するパッチ作成プロパティ (.pcp) ファイルの生成です。.pcp ファイルは特定のスキーマを持つデータベースですが、InstallShield または Orca を使って直接開くことができます。

[パッチのデザイン] ビューは、パッチ作成処理を簡素化する、使いやすいインターフェイスを提供します。このビューは、パッチ作成で必要な処理すべてを論理的にグループにまとめます。

QuickPatch プロジェクト



プロジェクト: *以前のバージョンが Windows Installer を使用してインストールされている場合のみ、QuickPatch を作成して製品をアップデートすることができます。初期のバージョンの製品のインストールは、次のプロジェクト タイプのいずれかに作成されている必要があります:*

- *基本の MSI*

- *InstallScript MSI*
- *QuickPatch*

QuickPatch プロジェクトは、規模の小さいシングル アップグレードをユーザーへ配布したいインストール作成者へお勧めするプロジェクトの種類です。カスタム アクションの追加、.ini データの変更などのより広範囲におよび変更には通常、標準パッチが必要です。

QuickPatch はカスタマイズ可能な範囲が限られてはいますが、[パッチのデザイン] ビューを使わないシンプルなパッチ構成方法として利用できます。基本的にどちらのパッチ作成方法も同じ配布タイプ (.msp と .exe ファイル) を作成します。

QuickPatch では、次のすべてを実行することができます。

- 元のインストールまたは以前の QuickPatch へ新しいファイルを追加する。
- 元のインストールのファイルを削除する。
- 以前の QuickPatch と共に追加されたファイルを削除する。
- 上記と同じ操作をレジストリ エントリで実行する。
- 元のインストールに含まれていたが、現在の QuickPatch プロジェクトには適用しないカスタム アクションを削除する。

QuickPatch プロジェクトの作成は常に、[新規 QuickPatch 作成] ウィザードから始めます。ウィザードを完了すると InstallShield で QuickPatch プロジェクトが開き、プロジェクトの設定を構成できます。

差分リリース



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript プロジェクトでは、このタイプのリリースには指定された 1 つ以上の既存リリースのセットのに存在していなかった (および/または、それらのバージョンと日付、時間、サイズおよび属性が違う) ファイルを含む差分リリースを定義およびビルドすることができます。差分リリースは、既存のリリースによってインストールした製品のバージョンをアップデートするのに使用します。このタイプのアップグレードには、すべてのメンテナンス/アンインストール機能のファイルも含まれます。Data1.hdr、Data1.cab および Layout.bin の新しいバージョンは既存のバージョンを上書きせず、新規のフォルダーに配置されます。



メモ: *差分リリースは、指定された既存のメディアと新しいメディアの間で変更があったファイルからのみ構成されているため、指定された以前のバージョンの製品を持たないシステムに差分リリースをインストールすると、有効なアプリケーションは、ほとんどの場合、作成されません。以前のバージョンの製品がインストールされていないシステムをアップデートするには、フル リリースを作成します。*

差分リリースに含まれるファイル

差分リリースを定義する場合、新しい差分リリース作成時に現在のプロジェクトと比較する複数の既存リリースを指定します。これらの既存のリリースは、リリース ウィザードの [アップデート] パネルで指定します。現在のプロジェクトのファイルは、次のいずれかが該当する場合、差分リリースに含まれます。

- ファイルは、指定された比較リリースの少なくとも 1 つで欠けているか、そのコピーと異なる日時、サイズまたは属性を持つ場合。

- ファイルが、"差分"プロパティが "常に含める" に設定されているコンポーネントにある場合。
- ファイルが、指定の比較リリースの少なくとも 1 つと比べて欠けているコンポーネントにある場合。
- ファイルが、指定の比較リリースの少なくとも 1 つと比べて異なる名前を持つコンポーネントにある場合。
- 指定した比較リリースの少なくとも 1 つとインストール先、言語、またはオペレーティング システムのプロパティが異なるコンポーネントにファイルがある場合。
- ファイルが、指定の比較リリースの少なくとも 1 つと異なる、コンポーネント階層内の名前またはパスを持つコンポーネントと関連付けられたコンポーネントにある場合。

差分リリースのインストール

インストールが初期化されたときに ADDREMOVE システム変数が非ゼロ値になっていると、インストールは自動的に同じ製品コードを持つ以前にインストールされた差分リリースを初期化しようとします。インストールはすべての DISK1TARGET フォルダーのサブフォルダーを列挙し、差分リリースのサブフォルダーを検索します。次の条件がすべて満たされていると、差分リリースが見つかった場合、それがロードされます。

- サブフォルダーには、差分リリースの有効な InstallShield X 以降、InstallShield DevStudio 9 または InstallShield Professional 6 以降の Data1.hdr ファイルである Data1.hdr が含まれます。
- ヘッダーファイルに保存されている製品コードはインストールの製品コードに一致します。
- 差分リリースの製品バージョンは、インストールの製品バージョンより新しくなっています。

完全リリース



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript プロジェクトでは、アップグレードのための完全リリースを定義、ビルドすることができます。完全リリースには、インストール プロジェクトにリンクするすべてのファイルが含まれます。製品の以前のバージョンがターゲット システムに既に存在していない場合、このタイプのリリースは、デフォルトで、初回インストールとして動作します。また、以前のバージョンを完全に上書きインストールすることも可能です。

完全リリースを作成して 1 つ以上の以前のバージョンをアップデートするとき、そのリリースが特定のバージョンのみ対象にするかどうかを指定する必要があります。

- **非バージョン固有** – 既定のオプションです。非バージョン固有の完全リリースは、以前のバージョンの製品がインストールされていないシステムにインストールすることができます。また、以前のバージョンの製品はどれもアップデートすることができます。
- **バージョン固有** – バージョン固有の完全リリースは、更新するように指定したバージョンのみ更新します。完全リリースをバージョン固有と設定して、バージョン番号を指定しないでおくと、アップデートは以前のバージョンすべてに適用されます。

最適なアップグレード ソリューションの決め方



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*
- *QuickPatch*

製品のアップグレードの作成に使用される方法は、オリジナルのインストール パッケージがどのように開発されたかによって変わってきます。オリジナルのインストールが Windows Installer ベースのプロジェクトで作成された場合、基本的に製品のアップデートにはメジャー、マイナーまたはスモールアップデートを作成します。元のインストールが InstallScript プロジェクトで作成された場合は、差分または完全リリースを作成して製品を更新します。これらの両方の種類についての詳細については以下をお読みください。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクト

アップグレードのインストール作成は、そのタイプに関わらず、製品の以前のバージョンを持っていないターゲット システムを対象にするかどうかを決めるところから始まります。それが決まると、アップグレードのパッケージにどの種類の方法を使用するかを決めることができます。以下のテーブルは最適な方法を選ぶときに目安となる一般的な概要です。アップグレードのパッケージに関するテクニックについてより掘り下げた内容については、「アップグレードのパッケージ オプション」を参照してください。

テーブル 1 ・ Windows Installer ベースのプロジェクトの可能なアップグレード ソリューション

ターゲット システムの状況	必要なインストール の種類	アップデート パッケージの技法
ターゲット システムの中には以前のバージョンの製品がインストールされているものもあり、製品がまったくインストールされていないものもある。	ファイルのサイズが問題ではない場合、以下の両方を行うインストールを作成することも可能です。 ・ ターゲット システムに以前のバージョンが存在していない場合、初回インストールとして作動する。 ・ 既にターゲット システムにインストールされている場合、既存の製品をアップデートする。	メジャーまたはマイナーアップグレードを作成して、それを完全インストールとしてパッケージします。

テーブル 1 ・ Windows Installer ベースのプロジェクトの可能なアップグレード ソリューション (続き.)

ターゲット システムの状況	必要なインストール の種類	アップデート パッケージの技法
ターゲット システムの中には以前のバージョンの製品がインストールされているものもあり、製品がまったくインストールされていないものもある。	以前のバージョンの製品をアップグレードを必要とするエンドユーザーのためのスモール インストールが必要な場合、2 つのインストールを分けて作成することができます。 ・ 初回インストールとして作動する完全インストール。 ・ 既にインストールされている製品の 1 つまたは複数の以前のバージョンをアップデートする小規模のインストール。このインストールにはバージョンとバージョンの間で変更があったデータのみが含まれているので、完全インストール パッケージの配布よりも狭い帯域幅を利用したアップグレードの配布が可能となります。	初回インストールの場合、メジャーまたはマイナーアップグレードを作成して、それを完全インストールとしてパッケージします。製品の以前のバージョンを使用しているエンドユーザーのためには、メジャーまたはマイナーアップグレードを作成して、それをパッチとしてパッケージします。 ***ヒント:*** *場合によって、パッチが適当ではないことがあります。パッチがアップグレードに適切かどうかを判断するための詳しいガイドラインは、「アップグレードのパッケージ オプション」を参照してください。* ***メモ:*** *パッチ パッケージはアップグレードとは違いますので注意してください。パッチは簡単にいうと、省スペースでアップグレードを配布するためのメカニズムです。*
すべてのターゲット システムが製品の以前のバージョンを持っている。	既にインストールされている製品の 1 つまたは複数の以前のバージョンをアップデートする小規模のインストールを作成することができます。このインストールにはバージョンとバージョンの間で変更があったデータのみが含まれているので、完全インストール パッケージの配布よりも狭い帯域幅を利用したアップグレードの配布が可能となります。	以下のいずれかを選択します : ・ メジャーまたはマイナーアップグレードを作成して、それを標準パッチとしてパッケージ。 ・ QuickPatch プロジェクトの作成。 InstallShield の QuickPatch 技術を利用して以前にインストールされた製品のためのシンプルなパッチを作成することができます。QuickPatch プロジェクトを使ってパッチを作成する場合、カスタマイズの範囲に制限がありますが、標準パッチに較べると QuickPatch プロジェクトのほうが簡単に作成することができます。詳細については、「パッチと QuickPatch プロジェクトの違い」を参照してください。 ***メモ:*** *標準のパッチ同様、QuickPatch は、省スペースでアップグレードを配布するためのメカニズムです。*

InstallScript プロジェクト

元のインストールが InstallScript プロジェクトで作成された場合は、2 種類のアップデート リリースの中の 1 つを作成することができます。

- **差分リリース** – この種類のリリースには、1 つ以上の指定された既存リリースのセットに存在していなかった (および/または、それらのバージョンと日付、時間、サイズおよび属性が違う) ファイルが含まれています。差分リリースは、既存のリリースによってインストールした製品のバージョンをアップデートするのに使用します。
- **完全 (非差分) リリース** – ターゲット システムに以前のバージョンの製品が存在していない場合、このタイプのリリースは初回インストールとして動作します。また、以前のバージョンを完全に上書きインストールすることも可能です。

差分インストールはバージョンとバージョンの間で変更があったデータのみが含まれているので、完全インストール パッケージの配布よりも狭い帯域幅を利用したアップグレードの配布が可能となります。但し、完全リリースのみ、製品が全くインストールされていないシステムへのインストールが可能です。詳細については、「差分リリース」および「完全リリース」を参照してください。

このセクションには、次のトピックが含まれています:

- メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い
- 自動アップグレード
- パッチと QuickPatch プロジェクトの違い
- アップグレードのパッケージ オプション
- 差分リリースと完全リリース

メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

Windows Installer は 3 種類 (メジャーアップグレード、マイナーアップグレード、スモールアップデート) の製品アップグレードをサポートしています次のテーブルは、ユーザーのニーズに最も適したアップグレードの種類を決定するお手伝いをします。アップグレードの要件がマイナーアップグレードまたはスモールアップデートにひ

とつでも適さない場合、メジャーアップグレードを作成します。Windows Installer ベースのプロジェクトにどのタイプのアップグレードを利用すべきかが分からない場合、または特に希望するタイプがない場合は、自動アップグレードを作成することができます。

テーブル 2・メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い

アップグレードの要件	メジャー アップグレードの使用	マイナー アップグレードの使用	スモール アップグレードの使用	メモ
.msi パッケージ名の変更	はい	いいえ	いいえ	.msi ファイルが Setup.exe インストールのランチャ に圧縮されていなければ、デフォルトのファイル名は ″製品名″ プロパティから取得されます。
エンドユーザーが以前のバージョンと最新のバージョンを同一マシンにインストールできるようにする	はい	いいえ	いいえ	
新しいサブ機能を追加する	はい	場合による	場合による	新規のサブ機能が新規のコンポーネントのみで構成されている場合、スモールアップデート、マイナーアップグレード、またはメジャーアップグレードを利用することができます。新規のサブ機能が既存のコンポーネントで構成されている場合、メジャーアップグレードを利用しなければなりません。
製品ツリーで機能を移動または削除する	はい	いいえ	いいえ	
新規の機能に新規のコンポーネントを追加する	はい	はい	はい	
既存の機能に新規のコンポーネントを追加する	はい	可。ただし Windows Installer が 2.0 以降の場合	可。ただし Windows Installer が 2.0 以降の場合	Windows Installer 1.x には、アップグレード パッケージで、マイナーアップグレードおよびスモールアップデートの新規の機能に配置する新規のコンポーネントが必要です。また、特殊コマンドライン処理も同様に必要です。
製品ツリーでコンポーネントを移動または削除する	はい	いいえ	いいえ	

テーブル 2・メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い (続き.)

アップグレードの要件	メジャー アップグレードの使用	マイナー アップグレードの使用	スモール アップグレードの使用	メモ
既存のコンポーネントのコンポーネントコードを変更する	はい	いいえ	いいえ	
コンポーネントのキーファイルを変更する	はい	いいえ	いいえ	
ファイル、レジストリキー、またはショートカットのいずれかを追加、削除または変更する	はい	はい	はい	ファイル、レジストリキーまたはショートカットが複数のコンポーネントに存在して、そのコンポーネントが 2 つ以上の機能で共有されている場合、メジャーアップグレードを利用する必要があります。
MSI データベースまたはトランスフォーム プロジェクトで始める	はい	いいえ	いいえ	

異なるタイプのアップグレードに関連するコード

いくつかの Windows Installer コードは製品を識別するのに役立ちます。

- **パッケージ コード** – 概要情報ストリームの一部。特定のデータベースを認識します。パッケージコードは、Windows Installer プロパティではありません。同じパッケージコードを持つ 2 つの .msi データベースの内容は同じでなければなりません。したがって、ビルドごとにパッケージコードを変更する必要があります。
- **ProductVersion** – これは、製品バージョンを含む Windows Installer プロパティです。Windows Installer は、バージョンを比較する際、ProductVersion プロパティの最初の 3 つのフィールドのみを利用することに注意してください。たとえば、1.2.3.4 の製品バージョンの場合、4 は無視されます。(これは、ProductVersion 値の比較にのみ当てはまり、ファイル バージョンの比較には当てはまらないので注意してください。)
- **ProductCode** – これは、製品の GUID を含む Windows Installer プロパティです。たとえ ProductName プロパティの値が同じでも、Windows Installer は ProductCode GUID が異なる 2 つの製品は相互に関連が無いものとして扱います。
- **UpgradeCode** – これは、関連製品を示す GUID を識別する Windows Installer プロパティです。インストール済みの製品の関連バージョンを検索できるように、UpgradeCode は、関連製品ファミリの異なるバージョンおよび言語にわたって統一されている必要があります。[アップグレード] ビューでアップグレードの UpgradeCode を設定することができます。

どの種類の再配布可能ファイルを表示するのかを指定するには、[表示するオブジェクトの種類] リストで適切なオプションを選択します。次のテーブルで、異なるタイプのアップグレードに各コードをいつ変更すべきかが分かります。

テーブル 3 ・ 異なる種類のアップグレードごとに変更する必要があるコード

	パッケージ コード	製品バージョン	製品コード	アップグレード コード
スモール アップデート	X			
マイナーアップグレード	X	X		
メジャー アップグレード	X	X	X	

自動アップグレード



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

アップグレード作成過程を簡素化するため、InstallShield には自動アップグレードの作成機能が搭載されています。自動アップグレードは、どの種類のアップグレードの種類を利用すれば良いのか分からない場合、または特にアップグレードの種類にこだわらない場合に利用できる、特殊な種類のアップグレードです。自動アップグレードを利用すると、指定したパッケージの新旧を比較して InstallShield がメジャーアップグレードまたはマイナーアップグレードのどちらが最適かを判断します。

パッチと QuickPatch プロジェクトの違い



***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

完全に更新されたインストール パッケージを配布せずに製品を更新する場合は、Windows Installer パッチ パッケージ (.msp) を作成することができます。InstallShield では、異なるシナリオに対応する異なる 2 つのパッチ作成方法 ([パッチのデザイン] ビューを使用する方法、および QuickPatch プロジェクト タイプを使用する方法) が提供されています。次の両方の選択肢を読んで、製品に適切なソリューションを見極めます。



メモ: メジャーアップグレードは、通常パッチとしてパッケージされません。従って、下のテーブルは [パッチのデザイン] ビューで作成されたパッチはマイナーアップグレードまたはスモールアップデート用と仮定します。QuickPatch プロジェクトは、マイナーアップグレードとして機能する QuickPatch パッケージを作成します。

テーブル 4 ・ パッチと QuickPatch プロジェクトの違い

パッチの要件	パッチの使用	QuickPatch の使用	メモ
ベース パッケージをアップデートする多くの累積パッチを適用できる機能	はい	可 (QuickPatch の簡素化を使用する場合)	QuickPatch の簡素化を使用しない場合、15 回以上パッチを適用することはできません。詳細については、「QuickPatch パッケージを簡素化するかどうかを指定する」を参照してください。
新しいサブ機能を追加する	はい	いいえ	
新規のサブ機能に新規のコンポーネントを追加する	はい	いいえ	
既存の機能に新規のコンポーネントを追加する	可。ただし Windows Installer が 2.0 以降の場合	いいえ	Windows Installer 1.x には、アップグレード パッケージで、マイナーアップグレードおよびスモールアップデートの新規の機能に配置する新規のコンポーネントが必要です。また、特殊コマンドライン処理も同様に必要です。
ファイルの追加、変更、または削除	はい	はい	QuickPatch で、新しいファイルに新しいターゲットの場所を使用することはできません。新しいファイルには、元のインストールで既に定義されている場所のみを使用できます。[パッチのデザイン] ビューを使って作成したパッチには、この制限はありません。

テーブル 4 ・ パッチと QuickPatch プロジェクトの違い (続き.)

パッチの要件	パッチの使用	QuickPatch の使用	メモ
レジストリデータの追加、変更、または削除	はい	はい	QuickPatch で追加する新規レジストリ データはすべて、既に元のインストールに存在する機能に関連付ける必要があります。
ショートカットの追加、変更、または削除	はい	いいえ	
カスタム アクションの追加、変更、または削除	はい	オリジナルのベースインストールに含められたカスタム アクションのみ削除することができます。	
再配布可能ファイルの追加または削除	はい	いいえ	
ODBC リソースの追加、変更、または削除	はい	いいえ	
.ini ファイルの編集	はい	いいえ	
.xml ファイルの編集	はい	いいえ	
IIS Web サイト、コンポーネントサービス、および SQL スクリプト等のサーバー設定を構成する	はい	いいえ	
パッチ パッケージ内のファイル (例、アプリケーションの実行可能ファイル) にデジタル署名する。	はい	自動ではない	QuickPatch プロジェクトの場合、個々のファイルを手動で署名してから、プロジェクトに追加する必要があります。

[セットアップデザイン] ビューの使用

[パッチのデザイン] ビューではビジュアルに統合された方法でパッチを作成すると共に、各パッチ構成に関連する適切な設定を選択することができます。[パッチのデザイン] ビューでは複数パッチ構成を作成することも可能です。各パッチ構成にはパッチをビルドするために必要な設定とデータが含まれます。

ほとんどの場合、[パッチのデザイン] ビューからパッチ作成を開始します。簡単なパッチソリューションが必要な場合は QuickPatch プロジェクトを作成することができます。

QuickPatch プロジェクトを利用する

QuickPatch プロジェクトを作成したときに起動される新規 QuickPatch 作成ウィザードは、規模の小さいアップデートをエンドユーザーに向けに作成するインストール作成者を対象としています。これは [パッチのデザイン] ビューでのパッチ作成よりも簡単です。

QuickPatch プロジェクトを使ってパッチを作成する場合、カスタマイズの範囲に制限がありますが [パッチのデザイン] ビューと同じ種類のパッチ (.msp と .exe) が作成されます。

アップグレードのパッケージ オプション



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

エンド ユーザーのマシン上にインストール済みの製品のバージョンを更新するインストールを作成する場合、メジャー アップグレード、マイナー アップグレード、またはスモール アップデートをパッケージする 2 つの選択肢があります。

- 以前のバージョンがインストールされている場合は既存製品をアップデートし、以前のバージョンが存在しない場合は初回インストールとして作動する完全インストールとして、アップグレードをパッケージすることが可能です。
- アップデートするバージョン間で変更されているデータ (.msi データおよびバイトレベルのファイル差分) のみを含むパッチとしてアップグレードをパッケージすることが可能です。

フルインストール パッケージ

マイナーアップグレードとスモールアップデートは共に、以前のバージョンが存在しない場合は初回インストールと同様に作動し、製品が既にインストールされている場合、その上からインストールを行います。メジャーアップグレードも、以前のバージョンが存在する場合は初回インストールと同様に作動しますが、メジャーアップグレードは一般的に以前のバージョンすべてをアンインストールしてから新しいバージョンをインストールします。

パッチパッケージ

パッチを利用すると、アプリケーションのファイルを特定バージョンにアップデートするのに必要な、データベースの一部のみを配布することができます。このため、完全インストールとしてパッケージされたアップグレードに比べて規模の小さいパッケージを作成することが可能です。つまり、完全インストールパッケージの配布よりも狭い帯域幅を利用したアップグレードの配布が可能となります。



メモ- *パッチはアップグレードの一種ではありません。パッチは簡単にいうと、省スペースでメジャーアップグレード、マイナーアップグレード、またはスモールアップデートを配布するためのメカニズムです。実際、パッチを作成するためにはまず、アップグレードを設計してからそれをパッチとしてパッケージします。パッチを作成する前に、アップグレードをフル インストールパッケージとしてテストすることをお勧めします。*

アップグレードの最適なパッケージオプションの決定方法

「最適なアップグレード ソリューションの決め方」トピックには、以前のバージョンの更新のために最も適切なパッケージの種類を判断するときに参考となるテーブルが掲載されています。標準パッチまたは QuickPatch がアップグレードのパッケージ方法として最も適切なメカニズムの場合もあります。しかし状況によって、パッチの代わりに完全インストールとしてアップグレードをパッケージする方法が相応しい場合もあります。例:

- ターゲットイメージが Windows Installer 1,2 以前で作成され、アップグレードイメージが Windows Installer 2.0 以降で作成された場合、アップグレードはパッチではなく完全インストールとしてパッケージする必要があります。このスキーマの違いを超えたパッケージにパッチを作成すると、問題が発生します。詳しい情報については、アップグレードおよびパッチ検証用の関連検証ツールついて説明されている Val0011 をご覧ください。
- アップグレードがターゲット システム上で 1 つまたは複数のファイルを別の場所に移動する場合、パッチではなく完全インストールとしてアップグレードをパッケージしなくてはなりません。エンドユーザーがターゲット システム上にあるファイルを移動するアップグレード用のパッチをインストールすると、問題が発生する可能性があります。たとえば、パッチが作動しない、システム修復が作動しない、後続のパッチが作動しない、またエンドユーザーが製品をアンインストールできない等。ワークアラウンドとして、古いロケーションにあるファイルを削除して新しいロケーションにファイルを追加するパッチを作成することができます。
- インストールを圧縮形式から非圧縮形式に変更する場合、またはその逆を行なう場合、アップグレードはパッチではなく完全インストールとしてパッケージしなくてはなりません。このシナリオでパッチを利用すると、ターゲット システムの修復が作動しない、後続のパッチが作動しない、またはエンドユーザーが製品をアンインストールできない場合があります。
- ファイルを .cab ファイル間で移動させる必要がある場合、または .cab ファイル内でファイル順を変更する必要がある場合、アップグレードをパッチではなく完全インストールとしてパッケージしなくてはなりません。
- オリジナルインストールのファイル数が 32,767 より大きいときに、最新版のインストールに含まれるファイルが 32,767 未満の場合、パッチは失敗します。同様に、オリジナルインストールのファイル数が 32,767 未満のときに、最新版のインストールに含まれるファイルが 32,767 より大きい場合、パッチは失敗します。どちらの場合も、アップグレードをフルインストールとしてパッケージする必要があります。

  オリジナルインストールと、最新版インストールの両方のファイル数が 32,767 より大きい (または両方のファイル数が 32,767 未満) 場合、アップデートをパッチとしてパッケージすることが可能です。
- パッチは、InstallScript MSI プロジェクトのメージャーアップグレードには作成できません。従って、InstallScript MSI プロジェクトのメジャーアップグレードを配布する必要がある場合、完全インストールとしてアップグレードをパッケージします。

差分リリースと完全リリース



プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

製品の元のインストールが InstallScript プロジェクトで作成された場合、製品のアップデートに差分または完全リリースを作成する必要があります。

以前のバージョンを持つターゲット システムや、製品がまったくインストールされていないターゲット システムがある場合、アップグレードを完全リリースとしてパッケージすることができます。以前のバージョンがターゲット システムに既に存在していない場合、この種類のリリースは、デフォルトで初回インストールとして動作します。また、以前のバージョンを完全に上書きインストールすることも可能です。

すべてのターゲット システムが以前のバージョンを持っている場合、アップグレードを差分リリースとしてパッケージできます。この種類のリリースには、1 つ以上の指定された既存リリースのセットに存在していなかった (および/または、それらのバージョンと日付、時間、サイズおよび属性が違う) ファイルが含まれています。差分リリースは、既存のリリースによってインストールした製品のバージョンをアップデートするのに使用します。



メモ: *InstallShield Professional 6.0 よりも古いバージョンを利用して InstallScript プロジェクト用のメディアを作成した場合、メディアをアップデートすることはできません。*

アップグレード、パッチ、および QuickPatch プロジェクトを使用する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

MSI データベースおよびトランスフォーム プロジェクトはメジャー アップグレードをサポートしますが、マイナー アップグレード、スモール アップデート、パッチ、または QuickPatch パッケージはサポートしません。

アップグレードをフルインストール、または標準パッチとしてパッケージする場合、まずプロジェクトの最新版を開いて、必要に応じてファイルおよびレジストリ エントリを追加するなどの変更を加えます。

QuickPatch としてアップグレードをパッケージする場合、まず新しい QuickPatch プロジェクトを作成します。QuickPatch プロジェクトでは、QuickPatch を使ってどの以前のリリースをパッチするかを指定します。

メジャーアップグレード、マイナーアップグレード、スモールアップデート、自動アップグレード、パッチ、および QuickPatch プロジェクトの作成方法についての詳細は、このセクションのトピックをご覧ください。このセクションではまた、でき上がったパッケージが適切に役割を果たすかどうか、アップグレードまたはパッチを検証する方法を説明します。

このセクションには、次のトピックが含まれています:

- ファイルの上書き規則について
- アップグレード関する考慮事項
- InstallShield を構成してアップグレードの種類を自動判別する
- メジャー アップグレードを作成する
- マイナー アップグレードを作成する
- スモール アップデートを作成する
- パッチ時の考慮事項
- パッチの作成と適用
- QuickPatch プロジェクトを作成する

- アップグレード、パッチ、および QuickPatch パッケージを検証する
- グローバル アセンブリ キャッシュのアセンブリをパッチする
- 差分リリースと完全リリース

ファイルの上書き規則について



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

Windows Installer サービスは、デフォルトで、アップグレードが含むファイルがターゲット システムに既に存在するファイルを上書きするかどうかを判断するときいくつかのファイルの上書き規則を使用します。これらの規則は、REINSTALLMODE プロパティが o 設定を使用してターゲット システムの古いファイルを上書きインストールするときに適用されます。この動作を変更するには、o オプションを次の値の中から 1 つを使って置き換えます。

- p ー ターゲット システム上に対応するファイルがない場合のみ再インストールします。
- e ー ファイルが見つからないか、または、バージョンが古いか同じなとき再インストールします。
- d ー ファイルが見つからないか異なるとき、再インストールします。
- a ー バージョンに関わらず、すべてのファイルを再インストールします。

REINSTALLMODE の設定は、インストール時にすべての機能に適用されるので、個別に設定することはできません。また、REINSTALLMODE に a を含めるように設定すると、パッチの適用中にオリジナルのインストールソースを求めるプロンプトを引き起こす可能性があります。

File Versioning Rules (Windows Installer ヘルプ ライブラリ)
REINSTALLMODE Property (Windows Installer ヘルプ ライブラリ)

アップグレード関する考慮事項



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

MSI データベースおよびトランスフォーム プロジェクトはメジャー アップグレードをサポートしますが、マイナー アップグレード、スモール アップデート、パッチ、または QuickPatch パッケージはサポートしません。

以下は、アップグレード作成のためのいくつかのガイドラインです。

パッケージ コード、製品バージョン、および製品コードのアップデート

どの種類の再配布可能ファイルを表示するのかを指定するには、[表示するオブジェクトの種類] リストで適切なオプションを選択します。詳細については、「メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い」を参照してください。

アップグレードとパッチの最適化

マイナー アップグレードまたはスモール アップデートを構成するとき、ビルドの設定にあるパッチの最適化機能を利用します。

パッチの最適化を行うと、コンポーネントの名前、コンポーネント GUID、File テーブルのキー、および Directory テーブルのキーを、アップグレードされたイメージとターゲットのイメージとの間で同期することができます。これにより、パッチのサイズを最も小さくすることができます。パッチの最適化を行わなかった場合、パッチとターゲットの .msi パッケージのサイズが同じになってしまったり、Windows Installer のコンポーネントの規則に違反したりします。

パッチ作成プロセスは File テーブル キーを使って、2 つのファイルが同じファイルかどうかを判断します。(実際のファイル名は使用することができません。これは、パッケージが異なる条件下でインストールされた同じ名前のファイルを複数含んでいる可能性があるからです。)パッチの最適化で以前のパッケージを指定すると、同一のファイルに対して同一の File テーブル キーが使用されます。



ヒント: *パッケージがダイナミック ファイル リンクを使用する場合、アップグレードをビルドするときにパッチの最適化を使用することが推奨されます。これによって、リリース間でキーファイルの一貫性が保たれます。*

パッチの最適化機能を利用するには、リリース ウィザードの [詳細設定] パネルで Windows Installer パッケージを指定します。[リリース] ビューでそのリリースについての [ビルド] パネルにある "以前のパッケージ" 設定でも以前の Windows Installer パッケージを指定することができます。

マイナーアップグレードでシグネチャを追加する

一般的に、マイナーアップグレードには新規の最上位機能を含めることはできません。但し、既存の機能の新規サブ機能は許可されます。マイナー アップグレードに新規のサブ機能を追加するとき、それらがマイナーアップグレード中に適切にインストールされるように次のようにサブ機能の設定を 2 つ設定します。

- リモート インストール― **[親を優先]** を選択します。
- 必須―**[はい]** を選択します。

マイナーアップグレードのユーザー インターフェイスは、通常機能ツリーを表示しませんが、アップデートされたインストールのメンテナンス モードは、通常、通常機能ツリを表示します。エンドユーザーが新規のサブ機能を選択解除できないようにそれを機能ツリーから取り除くには、サブ機能の "表示" 設定を [非表示] に設定します。

インストール済みデータの削除

マイナーアップグレードがファイル、レジストリ設定、または、.ini ファイル等のデータを削除する場合、以前の製品バージョンに存在していたデータをプロジェクトから単純に削除しても、ファイルはアップグレード適用時に削除されません。このようなデータの削除方法については、「マイナー アップグレードを構成してインストール済みのデータを削除する」をご覧ください。

InstallShield を構成してアップグレードの種類を自動判別する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Windows Installer ベースのプロジェクトにどのタイプのアップグレードを利用すべきかが分からない場合、または特に希望する種類がない場合は、自動アップグレードを作成することができます。自動アップグレードを利用すると、指定したパッケージの新旧を比較して InstallShield がメジャーアップグレードまたはマイナーアップグレードのどちらが最適かを判断します。このセクションのトピックでは、自動アップグレードの作成および処理方法を説明します。

自動アップグレード アイテムの追加



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*



タスク ***自動アップグレード アイテムを追加するには、以下の手順に従います:***

1. **[アップグレード]** ビューを開きます。
2. **[アップグレード シナリオのセットアップ作成]** エクスプローラーで **[Windows Installer インストールのアップグレード]** を右クリックして、**[自動アップグレード アイテムの追加]** をクリックします。自動アップグレード項目がエクスプローラーへ追加されます。
3. インストールの最新版に対応するようにアイテムの名前を変更します。

自動アップグレード アイテムの設定を構成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

自動アップグレード アイテムをプロジェクトに追加する場合、自動アップグレード アイテムの設定を構成する必要があります。設定の構成には、[アップグレード] ビューの [アップグレード シナリオのセットアップ準備] エクスプローラーで新規の自動アップグレード アイテムをクリックします。

また、InstallShield が自動アップデートアイテムにメジャーまたはマイナーアップグレードが必要と判断するかどうかによって、InstallShield がアップグレードの作成に使用する一般アップグレードの設定も構成する必要があります。[アップグレード] ビューのエクスプローラーの [アップグレードシナリオのセットアップ準備] エクスプローラーの [Windows Installer セットアップのアップグレード] をクリックしたときに表示されるタブでこれらの全般設定を構成することができます。

- [共通] タブ
- [詳細] タブ

自動アップグレードからメジャーまたはマイナーアップグレードへ変換する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*



タスク ***自動アップグレード アイテムをメジャーまたはマイナーアップグレードへ変換するには、以下の手順に従います:***

1. **[アップグレード]** ビューを開きます。
2. 変換する自動アップグレード アイテムを右クリックしてから **[マイナーアップグレード アイテムに変換]** または、**[メジャーアップグレード アイテムに変換]** をクリックします。

自動アップグレード アイテムを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*



タスク ***自動アップグレード アイテムを追加するには、以下の手順に従います:***

1. **[アップグレード]** ビューを開きます。
2. 削除する自動アップグレード アイテムを右クリックしてから、**[削除]** をクリックします。

メジャー アップグレードを作成する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

メジャー アップグレードが状況に適した最も有効なアップグレード ソリューションであると判断したならば、[アップグレード] ビューでメジャー アップグレード アイテムを作成開始します。メジャーアップグレードは機能の重要な変更を意味し、[一般情報] ビューでアップデートする ProductCode プロパティの変更を必要とします。

メモ *製品コードを変更すると、*ProductName *値が同じでも、Windows Installer は製品の最新版と以前のバージョンが関連しないもとのして処理します。製品の両方のバージョンを同じシステム上にインストール可能にするには、製品コードおよびメインとなるインストールディレクトリ (多くの場合、INSTALLDIR) の両方を変更するだけです。*

基本的に、メジャー アップグレードでは 2 つの処理が 1 つのインストール パッケージに組み込まれています。メジャー アップグレードは、製品の新しいバージョンをインストールしてから古いバージョンをサイレントでアンインストールか、古いバージョンをサイレントでアンインストールしてから新しいバージョンをインストールします。これら 2 つの個別の処理のシーケンスは、製品の新しいバージョンのインストール パッケージ構成方法によって異なります。

InstallShield を使用してのメジャー アップグレードの作成に関する詳しい情報ついては、InstallShield ヘルプライブラリ内の該当セクションを参照してください。

メジャー アップグレード アイテムを追加する

プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

タスク ***メジャー アップグレード アイテムを追加するには、以下の手順に従います:***

1. **[アップグレード]** ビューを開きます。
2. **[アップグレード シナリオのセットアップ準備]** エクスプローラーで **[Windows Installer セットアップのアップグレード]** を右クリックして、**[メジャー アップグレード アイテムの追加]** をクリックします。

新しいメジャー アップグレード アイテムが追加されます。

メジャー アップグレードの設定を構成する

プロジェクト *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[アップグレード] ビューでメジャーアップグレード アイテムを作成する場合、変更可能な一般設定は、RemoveExistingProducts アクションのスケジュール場所です。この設定は、"Windows Installer セットアップのアップグレード" を選択してから [共通] タブをクリックし、[アップグレード] ビューで変更することが可能です。

メジャーアップグレード アイテムの構成可能な設定は、ダイレクト エディターの Upgrade テーブルよりもユーザーフレンドリーなフォーマットで表示されます。

アップグレードアイテムの [共通] タブでは、アップデートする製品バージョンと共に、アップデートする製品のアップグレードコード値を指定します。[詳細]タブで、ISSetAllUsers カスタム アクションはデフォルトで含まれます。このアクションは、メジャー アップグレード中に ALLUSERS を適切に設定します。(さらに、メジャーアップグレードが実行される場合、ISSetAllUsers はカスタムプロパティ IS_MAJOR_UPGRADE を設定します。)ISSetAllUsers アクションを含めるかどうかを指定するには、[オプション] ダイアログ ボックスの [一般] タブを使用します。[オプション] ダイアログ ボックスにアクセスするには、[ツール] メニューから [オプション] を選択します。そのタブで "**ISSetAllUsers アクションを自動的に作成**" チェック ボックスを選択または非選択します。

メジャー アップグレード アイテムを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

タスク ***[アップグレード] ビューからメジャー アップグレード アイテムを削除するには、以下の手順を実行します。***

1. **[アップグレード]** ビューを開きます。
2. **[アップグレード シナリオのセットアップ作成]** エクスプローラーで メジャー アップグレード アイテムを右クリックして、**[削除]** をクリックします。

マイナー アップグレードを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

マイナー アップグレードが状況に適した最も有効なアップグレードソリューションであると判断すれば、[アップグレード] ビューで マイナー アップグレード アイテムを追加して作業を開始することができます。マイナー アップグレード アイテムでインストールをビルドすると、ビルドエンジンは一連のテストを実行し、最新版のインストールが以前のバージョンのインストールを確実にアップグレードするかどうかを判別します。ただし、マイナー アップグレード アイテムの追加は、マイナー アップグレードが適切に機能するために不可欠という訳ではありません。詳細については、「マイナー アップグレードの実行時の動作」を参照してください。

インストールの最新バージョンに追加する新しい機能は、既に存在する機能のサブ機能として追加しなくてはなりません。これらの新しい機能を使用する場合、[機能] ビューで "リモート インストール" 設定に [親を優先]、"必須" 設定に [はい] を選択する必要があります。

マイナー アップグレード アイテムを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク

[アップグレード] ビューでマイナー アップグレード アイテムを追加するには、以下の手順を実行します。

1. **[アップグレード]** ビューを開きます。
2. **[アップグレード シナリオのセットアップ作成]** エクスプローラーで **[Windows Installer インストールのアップグレード]** を右クリックして、**[マイナー アップグレード アイテムの追加]** をクリックします。

新しいメジャー アップグレード アイテムが追加されます。

ヒント: *複数の以前のバージョンをアップデートするアップグレードを作成する場合、追加のマイナーアップグレード アイテムを作成することができます。*

マイナー アップグレードの設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[アップグレード] ビューでマイナーアップグレード アイテムの追加を行なう場合に、変更可能な一般設定は、製品の以前のバージョンが存在する場合に (ビルド構成がセットアップランチャーを含む場合) セットアップランチャーがどのように作動するかです。この設定は、"Windows Installer セットアップのアップグレード" を選択してから [共通] タブをクリックし、[アップグレード] ビューで変更することが可能です。

マイナーアップグレード アイテムを [アップグレード] ビューに追加してからそれが選択されていることを確認した後、そのマイナーアップグレード アイテムの "アップグレードするセットアップ" 設定でターゲットとする製品の以前のバージョンの .msi データベースを参照することができます。

"リリースフラグ" 設定のリリースフラグを指定し、ビルドからマイナーアップグレードアイテムを除外することもできます。

マイナー アップグレードを構成してインストール済みのデータを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

InstallShield は、インストール済みデータを削除するマイナーアップグレードの構成について、2 つの異なる方法をサポートします。次のセクションで、2 つの方法について説明します。

方法 1

以前のリリースでインストールされたデータを削除するマイナーアップグレードを構成するには、アップグレードを構成する InstallShield プロジェクトを開いて、[ファイルとフォルダー] ビューでそのファイルを削除します。そしてダイレクト エディター内の Remove テーブルに対応するレコードを作成します。ダイレクト エディターの Remove テーブルには、RemoveFile、RemoveIniFile、および RemoveRegistry テーブルがあります。

マイナーアップグレードで、コンポーネントからレジストリデータを削除するには、RemoveRegistry テーブルにレコードを作成する必要があります。RemoveRegistry テーブルのレコードは、関連するコンポーネントがインストールされた時に削除するレジストリキーおよび値について説明します。RemoveFile テーブルとは異なり、RemoveRegistry テーブルは関連するコンポーネントがアンインストールされた時に指定されたレジストリデータを削除するオプションを受け付けません。その代わり、"**すべてのキーをアンインストール**" フラグを使ってレジストリ値を作成することが可能です。コンポーネントが、[名前] フィールドにハイフンを用いたレジストリ値を含み、[値] フィールドが空白の場合、指定したレジストリキーおよびその内容は、コンポーネントが削除される時に削除されます。

その他のデータ型では通常、Windows Installer テーブルでアンインストールフラグの利用が可能か、対応するアンインストールテーブルを利用することができます。たとえば .ini データを削除する場合、RemoveIniFile テーブルを利用することができます。環境変変数データには、対応するアンインストールフラグがあります。



メモ: *コンポーネントが別の製品と共有されている場合、リソースを削除する時にコンポーネントコードを変更する必要があります。さらに、既存するコンポーネントのコンポーネントコードを変更するには、メジャーアップグレードが必要です。メジャーアップグレードの場合、完全なアンインストールが必要なため、[Remove] テーブルにデータを入力する必要はありません。*

方法 2

コンポーネントを 移行可能と設定し、マイナーアップグレードが適用された時に false 評価されるコンポーネントの条件を設定します。

RemoveFile Table (Windows Installer ヘルプ ライブラリ)
RemoveIniFile Table (Windows Installer ヘルプ ライブラリ)
RemoveRegistry Table (Windows Installer ヘルプ ライブラリ)

マイナー アップグレード アイテムを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*



タスク ***[アップグレード] ビューでマイナー アップグレード アイテムを削除するには、以下の手順を実行します。***

1. **[アップグレード]** ビューを開きます。
2. **[アップグレード シナリオのセットアップ作成]** エクスプローラーで、マイナー アップグレード アイテムを右クリックして、**[削除]** をクリックします。

マイナー アップグレードの実行時の動作



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Setup.exe を使ってマイナー アップグレードを実行する

Setup.exe 起動プログラムを含んだリリースをビルドすると、最新のインストールはマイナー アップグレードが可能になります。Setup.exe はアプリケーションの古いバージョンが存在するときにそれを検出することができます。Setup.exe が以前のバージョンを検出した場合、インストールの続きはマイナーアップグレードモードで実行します。

製品の以前のバージョンがインストールされているターゲット システム上でエンド ユーザーがマイナー アップグレードを実行した場合、アップグレードが行われることを通知するメッセージ ボックスが表示されます。このメッセージ ボックスは以前の製品名を指定し、エンドユーザーに対してどの既存アプリケーションをアップデートするのかを警告します。この時点で、エンドユーザーは必要に応じてインストール処理を中止することができます。このプロンプトの後には [ようこそ] ダイアログが続き、エンドユーザーが [ようこそ] ダイアログの [次へ] をクリックすると、セットアップが新しいリソースをインストールします。リソースのインストール中、インストールは [セットアップの進行状況] ダイアログの進行状況バーを更新して、ユーザーにアップグレードの処理状況を追跡できるようにします。



ヒント: *[アップグレード] ビューでは、エクスプローラーの Windows Installer セットアップのアップグレードをクリックして、スモール/マイナーアップグレード用の [共通] タブの設定を構成し、最初のアップグレード プロンプトを無効にすることができます。*



メモ: *マイナー アップグレードの実行時に、コンポーネントのキー ファイルが処理中にダウングレードされる場合、Windows Installer はそのコンポーネントの再インストールを行いません。このため、アップグレードのキー ファイルは、古いパッケージのキー ファイルのバージョン番号と等しいか、それよりも大きい番号でなくてはな*

りません。例外は、エンド ユーザーが REINSTALLMODE プロパティに ***a*** *を設定した場合です。この設定は、バージョンに関係なくすべてのファイルを強制的にインストールします。ファイルの上書き規則に関する情報は、「ターゲット システム上でファイルとコンポーネントを上書きする」を参照してください。*

マイナーアップグレードは、ターゲットマシンに既に存在するアプリケーションの上に新しいリソースをインストールします。アプリケーションを完全にアンインストールしてから再インストールするには、メジャーアップグレードの利用を検討してください。

Setup.exe を使わないマイナー アップグレードを実行する

インストールを Setup.exe ランチャーなしで配布する場合、エンド ユーザーはコマンドラインを使用してインストールを開始する必要があります。このため、Setup.exe を使わなくても同じような結果が得られるものの、Setup.exe の利用をお勧めします。

インストーラープロパティ、REINSTALLや REINSTALLMODE はアップグレードモードでインストールをスタートさせるためにコマンドラインから設定する必要があります。最も複雑なシナリオ以外、通常は REINSTALLMODE プロパティは、**vomus** に、REINSTALL プロパティは **ALL** に設定しなくてはなりません。標準的なコマンド次のとおりです:

```
msiexec.exe /i ¥product.msi REINSTALLMODE=vomus REINSTALL=ALL
```

アップデートを行なわない機能が含まれる場合、次のコマンドの通り REINSTALL をコンマ区切りのアップデートする機能リストに設定します。

```
msiexec /i ¥product.msi REINSTALLMODE=vomus REINSTALL=F1,F3,F5
```

REINSTALL プロパティで利用する機能名は大文字と小文字を区別します。

メモ: *初回インストールで* REINSTALL = **ALL** *と設定しないで下さい。*

ここで重要なのは、インストールの以前のバージョンがターゲットマシンに存在しない限りこれらのプロパティはコマンドラインで設定しない方が良いということです。コマンドラインで設定した場合、インストールはマイナー アップグレード モードで実行しているかのように見えますが、アプリケーション ファイルがインストールされない可能性があります。エンドユーザーがコマンドラインを使用すべきかそうでないかを識別できるように配慮してください。

メモ: *マイナーアップグレードを実行する場合、*REINSTALLMODE *属性 v は非常に重要です。このパラメーターはインストーラーに対して、新しいインストール パッケージを使って製品の内部セットアップキャッシュを更新するよう指示します。このパラメーター無しでは、インストールは最新のインストール パッケージに施された変更に関する情報を全く持ちません。*

スモール アップデートを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

スモール アップデートの作成手順は、重要な例外をひとつ除いては、マイナー アップグレードの作成手順と同じです。例外とは、スモールアップ デートとは違って、マイナーアップグレードでは製品バージョンを変更するということです。

Windows Installer 3.0 より以前のバージョンを利用して製品に適用することができるアップグレードを作成する必要がある場合、スモールアップデート以外の方法を利用することが推奨されます。スモールアップデートは製品バージョンを変更しないので、製品の最新版に利用されるインストーラーをはじめとする外部プログラムは、スモールアップデートが適用された製品とそうではない製品の違いを認識できません。

スモールアップデートは、通常、完全インストールとしてではなくパッチとしてパッケージされます。作成するアップグレードが、Windows Installer 3.0 以降を利用して製品に適用されることが分かっている場合、パッチ シーケンス サポートを利用しスモールアップデート パッチを使用して、以前のバージョンの製品を更新することができます。

パッチ時の考慮事項



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

以下は、パッチと QuickPatch プロジェクトを作成するときのいくつかのガイドラインです。

パッチ作成前にアップグレードをテストする

パッチ作成前にフル インストールとしてアップグレードのテストを行うことが推奨されます。フルインストール パッケージがアプリケーションの以前のバージョンを確実にアップデートすると、パッチを作成することができます。

アップデート起動ツール (Update.exe) を作成する

InstallShield では、アップデートに Update.exe アップデート ランチャーを含めるかどうかを指定できます。Update.exe アップデート ランチャーは次のような場合に必要です。

- 必要に応じて、自動的にターゲット システムの Windows Installer エンジンを更新またはインストールする。
- InstallScript MSI プロジェクトのパッチをビルドしている。
- 必要に応じて、.NET Framework をターゲット システムに自動的にインストールする。
- 製品構成に製品のインスタンスを複数インストールできるサポートが含まれているパッチをビルドし、かつインスタンスの選択ダイアログを必要に応じて表示する。

  複数インスタンスのサポートに関する詳しい情報は、「製品の複数のインスタンスをインストールする」および「製品の複数インスタンスをインストールするためのサポート」をご覧ください。

Update.exe アップデート ランチャーは、上記のシナリオを制御するブートストラップ アプリケーションです。

InstallShield で Update.exe アップデートランチャーを作成しないように構成した場合、.msp ファイルが作成されます。

詳細については、次を参照してください。

- パッチに Update.exe アップデート ランチャをビルドをするかどうかを指定する
- QuickPatch パッケージに Update.exe アップデート起動ツールをビルドをするかどうかを指定する

アップデート ランチャー名のカスタマイズ

InstallShield では、現在のパッチの指定された名前を持つ (.exe) 用にアップデート ランチャーを作成するかどうかを指定できます。デフォルトで、InstallShield はアップデート ランチャーの名前として Update.exe を使用します。

大きなファイル用のパッチの最適化

[パッチのデザイン] ビューで**[大きなファイル用のパッチの最適化]** チェック ボックスを選択すると、ファイル サイズが 4MB 以上のアプリケーションに対してより小さいビットレベルのパッチが作成されます。

QuickPatch プロジェクトでは、ビルドされた QuickPatch はそれぞれ自動的に大きいファイル用に最適化されます。

圧縮されたインストールのパッチ

パッチの作成プロセスには、以前のインストールと最新のインストールの非圧縮リリースが必要です。インストールが圧縮されているインストールの場合は、リリースの管理用イメージを利用できます。

InstallShield は、[パッチのデザイン] ビューでパッチ構成に圧縮されたインストールを指定すると、自動的に管理用イメージを作成します。InstallShield はまた、QuickPatch プロジェクトで非圧縮のイメージを指定すると、自動的にユーザーのために管理イメージを作成します。

パッチと REINSTALL プロパティ

パッチのテスト中にオリジナルインストールのソースがある場所を問い合わせるプロンプトが表示される場合、REINSTALL プロパティで変更した機能に明示的に設定することで問題を回避できる場合があります。このプロパティはデフォルトで **"ALL"** に設定されます。

パッチまたは QuickPatch パッケージにおける COM サーバーの自己登録

次の条件のすべてが当てはまる場合、QuickPatch プロジェクトに含まれる自己登録とマークされている新しい COM サーバーには、ISSelfReg テーブルを使った InstallShield 自己登録が使用されます。

- ISSelfReg テーブルは、QuickPatch の作成時に使用された、アップグレード済みの .msi ファイルにあります。
- ISSelfRegisterFiles カスタム アクションが、[実行] シーケンスにある。
- InstallShield の [オプション] ダイアログにある [プリファレンス] タブで、ISSelfReg オプションが選択されている。

前述の条件のうち、1 つでも当てはまらない場合、InstallShield は Windows Installer 自己登録 (SelfReg テーブル) を使用します。

前述の条件の 1 つでも当てはまらない場合に、パッチで InstallShield 自己登録を使用するには、QuickPatch プロジェクトの代わりに [パッチのデザイン] ビューを使用します。

Windows Installer 3.0 (およびそれ以上) とパッチ

Windows Installer 3.0 以上には、多数のパッチ関連の強化点があります。パッチまたは QuickPatch が Windows Installer 3.0 以上を用いてターゲットマシンで適用される場合、これらの強化点を利用することができます。詳細は、次を参照してください:

- パッチ シーケンス
- パッチのアンインストール
- 非管理者パッチ

Windows Installer 3.0 以上は、シーケンスデータを含まないメジャーアップグレードパッチをサポートします。ただし、インストーラーは MsiPatchSequence テーブルに格納されるシーケンスデータを含むメジャーアップグレードパッチはサポートしません。

Windows Installer 1.2 とパッチ

Windows Installer 1.2 エンジンをターゲットとするパッチを作成する場合、[パッチのデザイン] ビューで適切なパッチ構成を選択します。そして [詳細] タブで "最小 Windows Installer バージョン" プロパティを **120** に設定します。

製品の複数インスタンスをパッチするためのサポート

同じコンテキスト (環境) および同じマシン上で製品のインスタンスを複数インストールできるサポートを含む基本の MSI プロジェクトにパッチを作成したとき、InstallShield では、エンドユーザーがそのパッチを 1 つまたはすべてのインスタンスに適用することができるように構成されます。詳細については、「製品の複数インスタンスをインストールするためのサポート」を参照してください。

アップデート起動ツールのファイルのプロパティをカスタマイズする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

InstallShield では、Update.exe アップデート ランチャーのバージョン リソースにカスタム情報を使用できます。この情報は、アップデート ランチャーの [プロパティ] ダイアログ ボックスに表示されます。[プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックしたときに表示されます。

メモ: *[プロパティ] ダイアログ ボックスは、Windows のバージョンによって異なります。たとえば、Windows 7 システムでは、バージョン リソース情報は [プロパティ] ダイアログ ボックスの [詳細] タブに表示されます。一方、Windows XP システムでは、バージョン リソース情報は、同じダイアログボックスの [バージョン] タブに表示されます。*

また、Windows の一部のバージョンは、[プロパティ] ダイアログ ボックスの一部の設定を表示しません。

Update.exe *ファイルの [プロパティ] ダイアログ ボックスと同じ概観を持つ* Setup.exe *ファイルの [プロパティ] ダイアログ ボックスのスクリーンショットを見るには、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。*



タスク ***Update.exe 設定のデフォルトの InstallShield 値を独自のカスタム値でオーバーライドするには、以下の手順に従います:***

1. この手順は、作成中のパッケージが標準のパッチか QuickPatch パッケージかによって異なります。

   基本の MSI または InstallScript MSI プロジェクトで標準のパッチを作成している場合:

   a. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。

   b. **[パッチのデザイン]** エクスプローラーで、構成するパッチ構成を選択します。

   QuickPatch パッケージを作成している場合:

   a. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。

   b. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。

2. **[詳細]** タブをクリックしてから、**[アップデート ランチャーの設定]** 領域を探します。

3. 以下の設定に、Update.exe ファイルのプロパティに使用する値を入力します。

   - 会社名
   - 製品名
   - 製品バージョン
   - 説明
   - 著作権情報



ヒント: *"製品バージョン" 設定は、Update.exe のファイル バージョンと製品バージョンを更新します。ファイル バージョンは、常に 4 つのフィールドで構成されます。4 フィールドよりも少ないフィールドを指定すると、残りのフィールドには 0 が挿入されます。たとえば、製品バージョンとして* **1.1** *を指定すると、Update.exe のバージョン リソースで使用されるファイル バージョンは* **1.1.0.0** *となります。*

アップデート起動ツールの設定を空白のままに残すと、InstallShield はデフォルトの InstallShield 値を使用します。

アップデート起動ツールのアイコンを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

InstallShield では、Update.exe アップデート ランチャーに使用するアイコンを指定できます。アイコンには、.exe、.dll、または .ico ファイルを使用できます。

エンド ユーザーが Windows Explorer で Update.exe ファイルを参照したときにも、このアイコンが表示されます。このアイコンは、Update.exe の [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックすると表示されます。

アイコンを指定しなかった場合、InstallShield は Update.exe ファイルのデフォルト アイコンを使用します。



タスク ***Update.exe ファイルのアイコンを指定するには、以下の手順に従います:***

1. この手順は、作成中のパッケージが標準のパッチか QuickPatch パッケージかによって異なります。

   基本の MSI または InstallScript MSI プロジェクトで標準のパッチを作成している場合:

   a. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。

   b. **[パッチのデザイン]** エクスプローラーで、構成するパッチ構成を選択します。

   QuickPatch パッケージを作成している場合:

   a. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。

   b. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。

2. **[詳細]** タブをクリックしてから、**[アップデート ランチャーの設定]** 領域を探します。

3. "**アイコン**" 設定には、InstallShield がビルド時に Update.exe ファイルを作成するときに使用するアイコンを含むファイルの完全修飾名を指定します。

   ファイルを指定するには、明示的パスまたはパス変数を入力するか、省略記号ボタン (...) をクリックして **[アイコンの変更]** ダイアログ ボックスを開いて **[参照]** ボタンをクリックするとファイルを選択できます。

   デフォルトではインデックス 0 のアイコンが使われます。別のアイコンを指定するには、**[アイコンの変更]** ダイアログ ボックスでアイコンを選択するか、アイコンのインデックスまたはリソース ID (それぞれ前にマイナス (-) 記号がついている) をファイル名に付け足します。たとえば、**C:¥Temp¥MyLibrary.dll,2** は 2 というインデックスを持つアイコンを、**C:¥Temp¥MyLibrary.dll,-100** は 100 と言うリソース ID を持つアイコンを示します。

リリースを次回ビルドするとき、InstallShield が Update.exe ファイルに指定されたアイコンを使用します。

パッチ シーケンス



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

InstallShield では、ターゲット マシンにパッチが提供された順番に関係なく、Windows Installer バージョン 3.0 以降がインストール済みの製品にスモール アップデート パッチを適用する順番を指定することができます。パッチ シーケンス データを使って、Windows Installer が同じパッチ ファミリー内でパッケージされた各アップグレードの適切な関係を認識できるようにします。従って、製品にパッチ 2 を適用した後にパッチ 1 を適用すると、パッ

チ 2 のファイルを上書きせずにパッチ 1 が登録されます。Windows Installer バージョン 3.0 以前では、パッチ シーケンスは無視され、すべてのスモール アップデート パッチは、ターゲット マシンに提供された順番で製品に適用されます。

Windows Installer 3.0 以降で利用可能なパッチシーケンス機能は、パッチ作成処理を簡素化します。次のセクションでは、その方法を説明します。

Windows Installer 3.0 以前のバージョンを使って適用されるパッチを作成する

Windows Installer 3.0 より以前のバージョンを利用して製品に適用することができるパッチを作成する必要がある場合、スモールアップデート以外の方法を利用することが推奨されます。スモール アップデートは製品バージョンを変更することはありません。したがって製品の最新版に利用されるインストーラーを含み、外部プログラムは、スモール アップデートを伴う製品およびスモール アップデートを伴わない製品の違いを認識しません。Windows Installer 3.0 以前のバージョンに限ったシナリオでは、こういったインストーラーの制限事項について考慮する必要があるため、以前の製品状態について様々な可能性をターゲットにしなくてはなりません。次のテーブルに図式化されたサンプル アプリケーションのライフサイクルで、それによって生じる複雑さを説明します。

テーブル 5 ・ Windows Installer 3.0 以前のバージョンを使って適用されるパッチのサンプル アプリケーション ライフサイクル

アプリケーション パッケージ	製品バージョン	パッケージがターゲットにする以前のセットアップ
1.基本のインストール	1.0	—
2.マイナー アップグレード	1.1	1.0
3.マイナー アップグレード	1.2	1.0, 1.1
4.マイナー アップグレード	1.3	1.0、1.1、1.2、4
5.マイナー アップグレード	1.4	1.0、1.1、1.2、1.3
6.メジャー アップグレード	2.0	1.0, 1.1, 1.2, 1.3, 1.4

Windows Installer 3.0 を使って適用されるパッチを作成する

Windows Installer 3.0 以降で利用可能なパッチ シーケンス機能を使うと、スモール アップデートは製品バージョンを変更することはありませんが、製品の複数の異なるバージョン用に連続性を持たない 1 つのアップグレードを配布するスモール アップデートを安心して利用することができます。スモール アップデートとは違って、マイナー アップグレードは製品バージョンを変更します。マイナー アップグレードはまた、スモール アップデート パッチのシーケンス用フレームワークを形成します。製品のバージョン 1.1 用のスモール アップデートがバージョン 1.2 に適用された場合、インストーラーはスモール アップデートをターゲット システムに登録して、1.2 マイナー アップグレードが適用される前の段階と同じ要領でそれを適用します。

スモール アップデートパッチはまた、Windows Installer 3.0 以降を有効にし、製品に別のパッチが個別に適用または削除されても製品の有効な状態を保ちます。さらに、パッチ シーケンスを利用すると、ターゲット マシン上に存在する可能性のあるパッチの組み合わせすべてを考慮する必要なく、以前の比較的小さな製品状態の集まりからアップグレード パッケージを生成することができます。次のテーブルに図式化されたサンプル アプリケーション ライフサイクルは、この利点を説明します。

テーブル 6 ・ Windows Installer 3.0 を使って適用されるパッチのサンプル アプリケーション ライフサイクル

アプリケーション パッケージ	パッチ シーケンス番号	製品バージョン	パッケージがターゲットにする以前のセットアップ
1.基本のインストール	—	1.0	—
2.スモール アップデート	1	1.0	1.0
3.スモール アップデート	2	1.0	1.0
4.スモール アップデート	3	1.0	1.0
5.スモール アップデート	4	1.0	1.0
6.マイナー アップグレード	—	1.1	1.0

上のテーブル内のスモール アップデートのすべては、同じパッチ ファミリーに所属します。Windows Installer 3.0 以降はパッチ ファミリーを利用して、同じファミリー内でスモール アップデート パッチとその他すべてのパッチを比較し、各パッチをターゲット マシンに適用する順番を決定します。パッチ シーケンスは、パッチ パッケージ データベースの MsiPatchSequence テーブル に追加されます。このテーブルは、同じパッチ ファミリーをターゲットにするパッチの関係を定義します。

Windows Installer 3.1 を使って適用されるパッチを作成する

Windows Installer 3.1 で利用可能なパッチ シーケンス機能は、マイナー アップグレードに関してパッチ作成処理をさらに簡素化します。[パッチのデザイン] ビューの [詳細] タブにある **"RTM バージョンをターゲットにするマイナー アップグレード (MSI 3.1 が必要)"** プロパティを利用すると、マイナー アップグレード パッチで製品の RTM (Release to Manufacturing) バージョン(インストールされている場合、製品の最新メジャー アップグレード)をターゲットするかどうかを指定することができます。このプロパティには次のような 2 つのオプションがあります。

- 現在のマイナーアップグレード パッチを適用する前に、マイナー アップグレード パッチで製品の RTM バージョン (または、インストールされている場合、製品の最新メジャー アップグレード) までのすべてのパッチを実質的に削除する場合、このプロパティで [はい] を選択します。このオプションを選択すると、すべてのパッチが、シーケンシング データがあるないにかかわらず、削除されます。追加のベースライン バージョンをターゲットにする必要はないので、結果的にパッチのペイロードを増やすことになります。すべてのエンドユーザーは、中間のパッチを適用することなく、正常にこのパッチを適用することができます。

- このオプションを選択すると、場合により、RTM (または、存在する場合、製品の最新メジャー アップグレード) の後に作成された以前の各マイナー アップグレードをターゲットするのに必要な情報をパッチに含める必要があります。

  たとえば、SP 2 のマイナー アップグレード パッチを作成していて、このプロパティで [いいえ] を選択した場合、パッチは SP 1 のマイナー アップグレードをターゲットにする必要があります。またオプションで、他のベースライン (RTM など) をターゲットにすることもできます。これにより、ペイロードが増加します。RTM バージョンをターゲットにしない場合、RTM バージョンを持っているが SP 1 のマイナー アップパッチがないエンドユーザーは、SP 2 の前に SP 1 をインストールしなければなりません。

Windows Installer 4.5 を使って適用されるパッチを作成する

優先されるパッチが製品のコンポーネントをインストールするが、後で優先するパッチがそのコンポーネントを削除する場合、コンポーネントの機能の状態がアドバタイズに変更され、再インストールされない可能性があります。さらに、その機能に関連付けられた残りのコンポーネントは一切保持されません。これは、Windows Installer 4.0 以前を持つターゲット システム上ではトラブルの原因となります。Windows Installer 4.5 で利用できるパッチのシーケンス機能は、この問題を回避するための強化された堅牢な優先機能を提供します。

現在のパッチに含まれるコンポーネントをアンインストール用にフラグして、優先するパッチが適用された後にターゲット システム上でこのコンポーネントが孤立しないように指定できます。後続のパッチがインストールされ、またそれが最初のパッチよりも優先することがフラグされている場合、適切な場合に Windows Installer 4.5 はこのコンポーネントを登録解除およびアンインストールします。

Windows Installer 4.5 は、置き換えられて不必要になったコンポーネントをアンインストール用にフラグするための 2 つの方法をサポートします:

- 個別のコンポーネントの "置き換えられたコンポーネントのアンインストール" 設定に [はい] を選択して、適切な場合にこれをアンインストールすることを示します。[はい] を選択すると、InstallShield はコンポーネントの Component テーブルに msidbComponentAttributesUninstallOnSupersedence 属性を追加します。

  "置き換えられたコンポーネントのアンインストール" 設定にアクセスするには、[コンポーネント] ビューを開いてから、構成するコンポーネントを選択します。右側のグリッドに設定が表示されます。この設定のデフォルト値は [いいえ] です。

- 優先されたパッチに含まれる不必要なコンポーネントすべてをアンインストールすることを示すには、Windows Installer プロパティ MSIUNINSTALLSUPERSEDEDCOMPONENTS の値を 1 に設定します。[プロパティ マネージャー] ビューを使ってプロジェクトにこのプロパティを追加すると、プロパティの値をコマンドラインから設定できます。

Windows Installer 4.0 以前は、msidbComponentAttributesUninstallOnSupersedence 属性と MSIUNINSTALLSUPERSEDEDCOMPONENTS プロパティを無視します。

MsiPatchSequence Table (Windows Installer ヘルプ ライブラリ)

パッチのアンインストール



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI – InstallScript ユーザー インターフェイス (UI) のスタイルが、埋め込み UI ハンドラーとして InstallScript エンジンを使用する新しいスタイルの場合*
- *QuickPatch*

この情報は、InstallScript UI に従来型のスタイル (InstallScript エンジンを外部 UI ハンドラーとして使用するスタイル) が使用されている InstallScript MSI プロジェクトには適用しません。詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

Windows Installer 3.0 以降はスモールアップデートまたはマイナーアップグレードのパッチアンインストールを サポートします。アンインストールするパッチは Windows Installer 3.0 以降を利用してインストールされたものでなくてはなりません。パッチがアンインストールされた時、製品はパッチがインストールされる前の状態に戻ります。以前のバージョンの Windows Installer では、パッチの削除を希望するエンドユーザーはパッチされた製品をアンインストールしてから、パッチを適用せずに製品を再インストールする必要がありました。この場合、削除しないパッチについては再び適用する必要があります。

Windows XP SP2 以降を実行中のシステムでは エンドユーザーは [プログラムの追加と削除]を使ってパッチをアンインストールすることができます。以前のバージョンの Windows で Windows Installer 3.0 以降を実行中のシステムでは、パッチはコマンドラインからアンインストールします。さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「Uninstalling Patches」を参照してください。

すべてのパッチがアンインストール可能ではないため、パッチのアンインストールはデフォルトでは無効に設定されています。パッチのアンインストールが不可能である、または特定の機能がパッチのアンインストール中に動作しない具体例は次のとおりです。

- パッチとしてパッケージされたメジャーアップグレードはアンインストールすることができません。
- パッチが特定の行を Windows Installer テーブルに追加する場合、パッチをアンインストールすることはできません。パッチがこれらのテーブルのいずれかに行を追加する場合、パッチ検証の 1 つ (Val0015) によって、パッチがアンインストール不可能であることが警告されます。該当するテーブルの一覧は、「Val0015」を参照してください。
- RemoveFile テーブルまたは RemoveRegistry テーブルに内容を追加するパッチをエンド ユーザーが 削除できない場合があります。オリジナルパッケージに含まれていないファイルまたはレジストリ エントリを削除するようにパッチが設計されている場合、そのパッチをアンインストールしてもファイルまたはレジストリ エントリは修復されません。
- [SQL スクリプト] ビューを通して追加された SQL スクリプト サポートを含むアンインストール可能なパッチを作成した場合、その SQL スクリプトは、パッチのアンインストール中に実行されません。

エンドユーザーへ配布する前にパッチのアンインストールを充分テストすることが推奨されます。アンインストール不可能なパッチについてさらに詳しい情報は、Windows Installer ヘルプ ライブラリの「Uninstallable Patches」を参照してください。

非管理者パッチ



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

Windows Installer 3.0 以上では、管理者以外によるインストールが可能なパッチを作成することができます。非管理者によるパッチは次のすべての条件が満たされた場合に利用することができます。

- Microsoft Windows XP 以降のクライアント プラットフォームの場合、ターゲットマシンが Windows Installer 3.0 以降を実行中。サーバー プラットフォームはサポートしていない。
- アプリケーションは CD-ROM または DVD などのリムーバブル メディアからインストールされている。
- アプリケーションはマシンごとにインストールされている。



メモ- *ALLUSERS プロパティがコマンドラインで上書きされた場合、非管理者パッチは失敗します。*

- ベース インストールには、後に続くすべてのパッチの署名に使用される証明書を含まれていなくてはなりません。
- ベース インストールには、MsiPatchCertificate テーブルを含まれていなくてはなりません。このテーブルは非管理者が後に続くパッチを適用した際、そのデジタル署名を照合するために使用される署名者証明書を提供します。このテーブルには必要に応じて複数の証明書を含むことが可能で、あとに続くパッチは最低 1 つの証明書と照合することが可能でなくてはなりません。詳細については、「非管理者パッチのインストールを準備する」を参照してください。
- 非管理者パッチには、MsiDigitalCertificate テーブルが含まれていなくてはなりません。このテーブルには、署名済みパッチの署名証明書が含まれています。詳細については、「パッチ パッケージに署名する」または「QuickPatch パッケージに署名する」を参照してください。

前述の基準のどれにも当てはまらない場合、エンドユーザーはロックダウンされた環境でデジタル署名済みパッチをインストールすることができません。

非管理者パッチの一般的なシナリオは、コンピューター ゲーム業界で見られます。コンピューター ゲーム ユーザーの一部は子供で、自分のユーザープロファイルおよび HKEY_CURRENT_USER の下にあるレジストリ キー内のフォルダー以外のシステム領域にアクセスが不可能な場合があります。子供がマシンにインストールする内容およびアクセスできる範囲を、彼等の両親が管理者アクセスを利用してコントロールする例が考えられます。両親がすべてのアプリケーションをインストールした場合で、インストール済みのソフトウェアに対するパッチが利用可能なときに、前述の基準がすべて満たされた場合、子供は両親が付き添わなくても自分で非管理者パッチをダウンロード並びにインストールすることができます。



メモ- *ベース インストールが期限切れのデジタル証明書で署名してあり、MsiDigitalCertificate および MsiPatchCertificate テーブルに以前の証明書と新しい証明書を含む中間パッチが適用されなかった場合、非管理者パッチは、ユーザー アカウント管理 (UAC) を表示します。*

User Account Control (UAC) Patching (Windows Installer ヘルプ ライブラリ)

パッチの作成と適用



プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

ヘルプのこのセクションでは、インストールのパッチの作成およびその適用方法について説明します。このセクションは、パッチのアンインストールを有効にする方法、パッチのシーケンスの仕方、およびパッチパッケージの署名方法についても網羅しています。

新しいパッチ構成アイテムを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***新しいパッチ構成アイテムを追加するには、次の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** を右クリックして、**[新しいパッチ構成の追加]** をクリックします。

InstallShieldは、新しいパッチ構成アイテムを追加します。

新しい最新セットアップ アイテムを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***最新のセットアップ アイテムを追加するには、次の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、適切なパッチ構成アイテムを右クリックして、**[最新のセットアップの新規追加]** をクリックします。

InstallShieldが、最新のセットアップを新規追加します。

新しい以前のセットアップ アイテムを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***既存のセットアップ アイテムを追加するには、次の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、適切なパッチ構成アイテムを右クリックして、**[既存セットアップの新規追加]** をクリックします。

InstallShieldが、以前のセットアップを新規追加します。

外部ファイルを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***外部ファイルを追加するには、以下の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、**追加の外部ファイル**を右クリックして、**[外部ファイルの新規追加]** をクリックします。

InstallShieldが、外部ファイルを新規追加します。右側にあるペインで、このファイルを構成することができます。

パッチの設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

パッチの作成および構成には、[パッチのデザイン] ビューを利用するのが最善の方法です。[パッチのデザイン] ビューで作業中に、製品の以前のバージョン、および、より最新のバージョンのためのプロジェクトが既にビルドされてい必要があります。

[パッチのデザイン] ビューでデザインされた各パッチは、パッチ構成として開始します。パッチ構成は、少なくとも、最新セットアップ アイテム 1 つと 1 つまたは複数の以前のセットアップ アイテムからなります。パッチ構成段階では、次のプロパティタブでプロパティを構成することができます。

- 共通
- 識別
- デジタル署名
- Sequence

- アドバンスト UI

パッチ構成プロパティを設定した後、エクスプローラーで順に階層を展開し、最新および以前のセットアップ アイテムを構成します。

パッチの識別情報を指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Windows Installer 3.0 以降は、パッチがアンインストール不可能な場合も、適用された各パッチについての [プログラムの追加と削除] エントリをターゲット システムに追加します。



ヒント: *[パッチのデザイン] ビューでパッチ構成の最新セットアップを変更するたびに、その最新セットアップからの [プログラムの追加と削除] 情報が [識別] タブにある設定用の値として使用されます。必要に応じて、[識別] タブの値をオーバーライドすることもできます。*



タスク ***パッチの識別情報を指定するには、以下の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. パッチ構成がまだ作成されていない場合、それを作成します。
3. **[パッチのデザイン]** エクスプローラーで、パッチ構成をクリックします。
4. **[識別]** タブをクリックします。
5. 個々の設定を構成します。

パッチのアンインストールを有効にする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI — InstallScript ユーザー インターフェイス (UI) のスタイルが、埋め込み UI ハンドラーとして InstallScript エンジンを使用する新しいスタイルの場合*

この情報は、InstallScript UI に従来型のスタイル (InstallScript エンジンを外部 UI ハンドラーとして使用するスタイル) が使用されている InstallScript MSI プロジェクトには適用しません。詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

Windows Installer 3.0 以降は、スモール アップデートおよびマイナー アップグレード用のパッチのアンインストールをサポートします。ただし、すべての パッチがアンインストール可能という訳ではありません。パッチのアンインストールについての詳細および、関連する制限事項については、「パッチのアンインストール」を参照してください。



タスク ***パッチのアンインストールを有効にするには、以下の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、構成するパッチ構成をクリックします。
3. **[共通]** タブをクリックします。
4. **[パッチのアンインストールを許可する (Windows Installer 3.0 が必要)]** チェック ボックスを選択します。

パッチのシーケンス



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

パッチ シーケンスを定義する前に、[アップグレード] ビューで必要なアップグレード アイテムを作成する必要があります。その後、[パッチのデザイン] ビューでパッチの構成アイテムを追加します。これらのタスク完了後はじめて、パッチのシーケンスを指定することができます。



タスク ***パッチ シーケンスを定義するには、以下を手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、構成するパッチ構成をクリックします。
3. **[シーケンス]** タブをクリックします。
4. 以下のいずれかを実行します。
   - InstallShield が生成するデフォルトのパッチ シーケンスを使用する場合、[デフォルトのシーケンス エントリの使用] チェック ボックスを選択します。
   - パッチ シーケンスを使用しない場合、[デフォルトのパッチシーケンスを使用する] チェック ボックスをクリアします。
   - カスタム シーケンスを指定する場合、次の手順に従います:
     a. **"デフォルトのパッチ シーケンスの使用"** チェック ボックスをクリアします。
     b. **[追加]** をクリックします。**[パッチ シーケンス]** ダイアログ ボックスが開きます。
     c. 必要に応じてシーケンス情報を定義します。

パッチ パッケージに署名する

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Windows Installer 3.0 以上では、管理者以外によるインストールが可能なパッチを作成することができます。非管理者によるパッチは厳しい条件が満たされたときのみ利用することができます。たとえば、パッチが更新するベース インストールは、パッチ パッケージの署名に使用される証明書を含まなくてはなりません。満たされなければならない他の基準については、「非管理者パッチ」を参照してください。

メモ- *パッチのファイル (アプリケーションの実行可能ファイルなど) にデジタル署名を行う場合、[リリース] ビューの [署名] タブで署名するファイルを指定することができます。*

タスク ***パッチパッケージを署名するには、以下を手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、構成するパッチ構成をクリックします。
3. **[デジタル署名]** タブをクリックします。
4. **[パッチ パッケージの署名]** チェック ボックスを選択します。
5. デジタル署名 設定を構成します。

パッチ パッケージをパスワードで保護する

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

セキュリティをさらに向上させるため、パッチをパスワードで保護することができます。パッチをパスワードで保護すると、パッチを適用するエンド ユーザーは、アップデートを起動するために大文字小文字の区別があるパスワードを入力する必要があります。

タスク ***パッチパッケージをパスワードで保護するには、以下の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、構成するパッチ構成をクリックします。
3. **[詳細]** タブをクリックします。
4. **"ランチャーをパスワードで保護"** 設定で、**[はい]** を選択します。

5. **"ランチャ パスワード"** 設定に、パッチで使用するパスワードを指定します。



メモ: *パスワードは大文字と小文字を区別します。*

パッチに Update.exe アップデート ランチャをビルドをするかどうかを指定する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

InstallShield では、InstallShield で、作成中のパッチ パッケージに Update.exe アップデート ランチャを作成するかどうかを指定できます。

必要に応じて自動的にターゲット システムで Windows Installer サービスを更新またはインストールする場合、Update.exe アップデート ランチャーが必要です。Update.exe ランチャーが必要になる場合についての詳細は、「パッチ時の考慮事項」を参照してください。

InstallShield で Update.exe アップデートランチャーを作成しないように構成した場合、.msp ファイルが作成されます。



タスク ***パッチに Update.exe アップデート ランチャを含めるかどうかを指定するには、以下の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、構成するパッチ構成をクリックします。
3. **[共通]** タブをクリックします。
4. Update.exe を含める場合、**[Update.exe を作成する]** チェック ボックスを選択します。

   Update.exe を含めない場合、**[Update.exe を作成する]** チェック ボックスをクリアします。



ヒント: *[詳細] タブを利用しても、Update.exe を含めるかどうかを指定することができます。*

最新セットアップ アイテムを別のパッチ構成にコピーする



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

複数のパッチ構成アイテムを [パッチのデザイン] ビューに追加した場合、ひとつのパッチ構成内の最新セットアップ アイテムを別のパッチ構成にコピーすることができます。



タスク ***ひとつのパッチ構成内の最新セットアップ アイテムを別のパッチ構成にコピーするには、以下の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、コピーする最新セットアップアイテムを、それを持っていないパッチ構成アイテムまでドラッグします。

InstallShield は、前回のセットアップ アイテムおよび追加の外部ファイルをすべて含む最新のセットアップ アイテムをパッチ構成に追加します。

パッチ構成アイテムを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*



タスク ***パッチ構成アイテムを削除するには、次の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、削除するパッチの構成を右クリックしてから、**[削除]** をクリックします。

最新セットアップ アイテムを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*



タスク ***最新のセットアップ アイテムを削除するには、次の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、削除する最新のセットアップ アイテムを右クリックしてから、**[削除]** をクリックします。

以前のセットアップ アイテムを削除する



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*



タスク ***既存のセットアップ アイテムを削除するには、次の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、削除する既存のセットアップ アイテムを右クリックしてから、**[削除]** をクリックします。

パッチのビルド



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[パッチのデザイン] ビューでパッチの構成を設定し、[リリース] ビューでパッチの新しいリリースを作成すると、アップグレードのためのパッチをビルドする準備ができたことになります。



タスク ***パッチをビルドするには、以下の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、パッチをビルドするパッチ構成アイテムを右クリックしてから、**[パッチのビルド]** をクリックします。

InstallShield はパッチ構成アイテムに指定した設定に従ってパッチをビルドします。パッチは、パッチ構成の [共通] タブで指定した場所でビルドされます。InstallShield は次のようなサブディレクトリを指定した出力場所へ追加します。

¥*PatchConfigName*¥Patch

InstallShield は、また、デフォルトでパッチをビルドするごとにアップグレードとパッチの検証を実行します。詳細については、「アップグレード、パッチ、および QuickPatch パッケージを検証する」を参照してください。

最新のセットアップ バージョンが圧縮でビルドされると、ビルドはエラーを返して途中終了します。

パッチの適用



プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

パッチパッケージ (.msp) ファイルには、インストールされた 1 つ以上の製品バージョンをアップグレードするのに必要なトランスフォームおよび指示が含まれています。

パッチパッケージを適用するには、Windows Installer v. 1.1 以降がシステムにインストールされている必要があります。またアップデートするパッケージは、管理イメージとして、またはローカルにインストールされている必要もあります。さらに別の要件として、既存のセットアップパッケージが、パッチパッケージと同じ権限を使って同じユーザー向けにインストールされている必要があります。たとえば製品がすべてのユーザー用としてインストールされている場合は、アップデートもすべてのユーザー用にインストールする必要があります。

パッチを適用する場合、Windows エクスプローラーで .msp ファイルをダブルクリックするか、またはファイルを右クリックして [開く] を選択するのが最も簡単です。マイナーアップグレードのパッチを適用すると、最初に開くダイアログは PatchWelcome ダイアログです。メジャー アップグレードのパッチを適用すると、インストールをスタンドアロンで実行したときに表示されるような完全ダイアログ シーケンスが表示されます。

MsiExec.exe /p オプションを使用して、コマンドラインからパッチを適用できます。次のステートメントを入力すると、X:¥Product Updates¥Build 36¥PatchForV1.msp にあるパッチパッケージが適用されます。

```
msiexec /p "X:¥Product Updates¥Build 36¥PatchForV1.msp"Update.exe
```

パッチ構成に [パッチのデザイン] ビューで [Update.exe の作成] チェック ボックスを選択すると、.msp ファイルが実行可能ファイルにラップされます。次のコマンド ライン指定すると、Update.exe によってパッチ パッケージが起動されます。

```
msiexec /p <path to .msp file> REINSTALL=ALL REINSTALLMODE=omus
```

サイレントモードでのパッチの適用

サイレント モードでパッチを適用する場合、2 種類の方法があります。/qn コマンド ライン パラメーターを使って MsiExec.exe を起動するか、/s を Update.exe へ渡します。

サイレントモードでパッチを適用する場合、考慮を要する重要な点があります。正しく動作させるには、パッチを適用する際に、Windows Installer のプロパティ REINSTALL が **ALL** に、REINSTALLMODE が **omus** に設定されている必要があります。Update.exe は常にこれらのプロパティをコマンドラインで設定するため、パッチパッケージが Update.exe とともに適用されていた場合は、ユーザーが特別な操作を行う必要はありません。

完全なユーザー インターフェイスを使ってパッチパッケージを適用すると、インストールのデフォルトのダイアログの 1 つ、PatchWelcome が表示されます。ここには、正しいオプションを使って REINSTALL および REINSTALLMODE を設定するためのコントロール イベントが含まれています。ただしこのダイアログは、エンドユーザー インターフェイスが抑制されていると表示されないため、次のようにコマンドラインでプロパティを設定する必要があります。

```
msiexec /p <.msp ファイルへのパス> /qn REINSTALL=ALL REINSTALLMODE=omus
```

パッチは既存する .msi データベースのキャッシュを変更しないため、REINSTALLMODE に **v** 設定を含む必要があります。

QuickPatch プロジェクトを作成する



プロジェクト: *以前のバージョンが Windows Installer を使用してインストールされている場合のみ、QuickPatch を作成して製品をアップデートすることができます。初期のバージョンの製品のインストールは、次のプロジェクト タイプのいずれかに作成されている必要があります:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

QuickPatch プロジェクトは、規模の小さいシングル アップグレードをユーザーへ配布したいインストール作成者へお勧めします。QuickPatch はカスタマイズ可能な範囲が限られてはいますが、[パッチのデザイン] ビューを使わないシンプルなパッチ作成方法として利用できます。基本的にどちらのパッチ作成方法も同じ配布タイプ (.msp と .exe ファイル) を作成します。

既存の .msi ファイルまたは既存の QuickPatch をパッチする QuickPatch プロジェクトを作成するには、新規 QuickPatch 作成ウィザードを利用します。

既存の QuickPatch をパッチする QuickPatch プロジェクトを作成する



***プロジェクト・** この情報は、QuickPatch プロジェクトに適用します。*

既存の QuickPatch プロジェクトをベースにして、新規 QuickPatch プロジェクトを作成することができます。これは、オリジナルアプリケーションまたは特定のパッチ済みバージョンに対してパッチをあてる必要があるエンドユーザすべてに配布することができるパッチを作成します。



***注意 ・** [一般情報] ビューの [履歴] 領域に表示されるリリースを開いて変更を加えた場合、最新プロジェクトは動作不可能となります。これは [履歴] にあるリリース間で共有されている中間データが不足または改定された可能性があるためです。つまり、既存する QuickPatch プロジェクトをパッチする QuickPatch プロジェクトを作成するたびに InstallShield は既存の QuickPatch プロジェクト (.ism ファイル) のモードを読み取り専用に変更します。*



タスク ***既存の QuickPatch をパッチする QuickPatch プロジェクトを作成するには、以下の手順を実行します。***

1. InstallShield で、パッチをあてる QuickPatch プロジェクト (.ism ファイル) を開きます。
2. **[一般情報]** ビューを開きます。
3. **[一般情報]** エクスプローラーで、**[履歴]** を選択します。
4. **[次のバージョンの作成]** をクリックします。

新規 QuickPatch プロジェクトが InstallShield で開きます。

新規 QuickPatch 作成ウィザードを使用して、既存の QuickPatch に QuickPatch を作成することもできます。

QuickPatch パッケージを簡素化するかどうかを指定する



***プロジェクト・** この情報は、QuickPatch プロジェクトに適用します。*

QuickPatch プロジェクトの構成を行うとき、InstallShield で QuickPatch パッケージの作成を簡素化して、最もシンプルなパッケージをビルドするかどうか指定することができます。QuickPatch の簡素化を行うと、通常の QuickPatch パッケージに比べて新しいサブ機能とカスタム アクションの数が少ない QuickPatch パッケージを生成できます。

たとえば、QuickPatch プロジェクトに新しいファイルとレジストリ エントリが含まれていて、が QuickPatch の簡素化を行わなかった場合、そのファイルとレジストリ エントリ用に新しいサブ機能が作成されます。さらに、特定の Windows Installer パッチ要件に対応するため、1 つまたは複数のビルド済み カスタム アクションが追加されます。これに対し、InstallShield が QuickPatch の簡素化を行う場合、ファイルまたはレジストリ エントリは既存する機能に追加されるため、特別なビルド済み InstallShield カスタム アクションは必要ありません。



メモ: *次のシナリオにおいて、InstallShield が QuickPatch パッケージの作成処理を簡素化することはできません。*

- *QuickPatch パッケージがインストール済みのファイルを削除する。*
- *QuickPatch パッケージがレジストリ キーを削除する、またはその名前を変更する。*
- *QuickPatch パッケージが、簡素化されていない通常の QuickPatch イメージをターゲットとする。つまり、[一般情報] ビューの [履歴] 領域で、簡素化を行わなかった QuickPatch のチェック ボックスを選択した場合、QuickPatch の簡素化はできないということです。1 つまたは複数の簡素化されていない QuickPatch イメージをターゲットとする簡素化された QuickPatch のビルドを試みると、ビルド警告が表示され、簡素化は行われません。*

また、15 未満の簡素化されていない累積 QuickPatch パッケージをベース .msi パッケージまたはメジャー アップグレード パッケージに適用できる点にご注意ください。制限数を超えると、パッチを適用中に エラー 2701 が発生します。正確な制限は、パッケージの機能ツリーおよび簡素化されていない各 QuickPatch がそのツリーに追加するサブ機能の階層数によって決まります。Windows Installer の機能ツリー階層数の制限は 16 レベルです。

簡素化された QuickPatch パッケージには、新しいサブ機能が含まれないため、この制限がありません。



タスク ***QuickPatch パッチを簡素化するかどうかを指定するには、以下の手順に従います:***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[詳細]** タブをクリックします。
4. **"QuickPatch の簡素化"** 設定で、適切なオプションを選択します:
   - QuickPatch パッケージを簡素化する場合、**[はい]** を選択します。この値は、新しい QuickPatch プロジェクトのデフォルト値です。
   - QuickPatch パッケージを簡素化を選択しない場合、**[いいえ]** を選択します。

パッチのターゲット リリースを指定する



プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*



タスク ***QuickPatch プロジェクトでどのリリースをパッチするのかを指定するには、次の手順を実行します。***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[履歴]** を選択します。

3. 中間リリース（ベース QuickPatch イメージと現在の QuickPatch イメージの中間となるリリース）の隣にあるチェック ボックスを選択またはクリアして、現在のプロジェクトでパッチするのかどうかを指定します。



メモ- *ベース QuickPatch イメージまたは現在のプロジェクトのチェック ボックスをクリアすることはできません。中間 QuickPatch プロジェクトのチェック ボックスをクリアすると、現在の QuickPatch プロジェクトを使って、アプリケーションの最新イメージとして中間イメージを持つマシンをアップグレードすることはできません。*

たとえば、インストールとしてバージョン 1.0 を配布した場合で、その後オリジナルインストールをバージョン 1.1 にアップグレードするパッチをリリースした場合。バージョン 1.2 QuickPatch を作成して、履歴で 1.1 QuickPatch のチェック ボックスが選択されていない場合、 1.2 QuickPatch は 1.0 リリースのアップグレードに利用できますが、1.1 リリースには利用できません。

QuickPatch を実行するカスタム アクションを指定する



プロジェクト- *この情報は、QuickPatch プロジェクトに適用します。*



タスク ***QuickPatch の適用時に実行するカスタム アクションを指定するには、以下の手順を実行します。***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで **[カスタム アクション]** をクリックします。
3. カスタム アクションの隣にあるチェック ボックスを選択またはクリアして、QuickPatch のインストール中にそれらを実行するかどうかを指定します。

QuickPatch へファイルを追加する



プロジェクト- *この情報は、QuickPatch プロジェクトに適用します。*



タスク ***QuickPatch にファイルを追加するには、以下の手順を実行します。***



メモ- *QuickPatch プロジェクトは、追加する新しいファイルに新しいターゲットの場所を定義できるサポートを含みません。そのため、QuickPatch プロジェクトに新しいファイルを追加する場合、そのインストール場所は、元のインストールで定義されたフォルダーでなくてはなりません。*

1. **[パッチ設定の定義]** の下にあるビュー リストで、**[ファイル]** をクリックします。
2. **[パッチするファイル]** エクスプローラーを右クリックし、**[新規ファイルの挿入]** をクリックします。**[開く]** ダイアログ ボックスが開きます。

3. 追加するファイルをクリックして **[開く]** をクリックします。**[パッチするファイル]** エクスプローラーにファイルが追加されます。
4. 新しいファイルをクリックしてから、その設定を構成します。

QuickPatch の識別情報を指定する



プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

Windows Installer 3.0 以降は、QuickPatch がアンインストール不可能な場合も、適用された各 QuickPatch についての [プログラムの追加と削除] エントリをターゲット システムに追加します。



タスク ***QuickPatch の識別情報を指定するには、以下の手順に従います:***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[識別]** タブをクリックします。
4. 個々の設定を構成します。

QuickPatch のアンインストールを有効にする



プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

Windows Installer 3.0 以降は、スモール アップデートおよびマイナー アップグレード用の QuickPatch パッケージのアンインストールをサポートします。ただし、すべての QuickPatch パッケージがアンインストール可能という訳ではありません。標準パッチおよび QuickPatch パッケージのアンインストールについての詳細および、関連する制限事項については、「パッチのアンインストール」を参照してください。



タスク ***QuickPatch パッケージのアンインストールを有効にするには、以下の手順を実行します。***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[共通]** タブをクリックします。
4. **[パッチのアンインストールを許可する (Windows Installer 3.0 が必要)]** チェック ボックスを選択します。

QuickPatch パッケージをシーケンスする



プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*



タスク ***QuickPatch パッケージのシーケンスを定義するには、以下の手順を実行します。***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[詳細]** タブをクリックします。
4. **"パッチ シーケンスのエントリ作成"** 設定で、適切なオプションを選択します:
   - InstallShield が生成するデフォルトのパッチ シーケンスを使用する場合、**[はい]** を選択します。
   - パッチ シーケンスを使用しない場合、**[いいえ]** を選択します。

QuickPatch パッケージに署名する



プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

Windows Installer 3.0 以上では、管理者以外によるインストールが可能なパッチを作成することができます。非管理者による QuickPatch パッケージは厳しい条件が満たされたときのみ利用することができます。たとえば、パッチが更新するベース インストールは、パッチ パッケージの署名に使用される証明書を含まなくてはなりません。満たされなければならない他の基準については、「非管理者パッチ」を参照してください。



メモ: *QuickPatch プロジェクトでは、パッチ パッケージと* Update.exe *ファイルにデジタル署名することができます。QuickPatch パッケージの各ファイル (アプリケーションの実行可能ファイルなど) にデジタル署名を行う場合、それらを手動で署名してから、プロジェクトに追加しなければなりません。Windows SDK に含まれている* SignTool.exe *を使って、手動でファイルに署名を行うことができます。*



タスク ***パッチパッケージを署名するには、以下を手順に従います:***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[デジタル署名]** タブをクリックします。
4. **[パッチ パッケージの署名]** チェック ボックスを選択します。
5. デジタル署名設定を構成します。

QuickPatch パッケージをパスワードで保護する



プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

セキュリティをさらに向上させるため、QuickPatch パッケージをパスワードで保護することができます。QuickPatch パッチをパスワードで保護すると、QuickPatch パッチを適用するエンド ユーザーは、アップデートを起動するために大文字小文字の区別があるパスワードを入力する必要があります。



タスク ***QuickPatch パッケージをパスワードで保護するには、以下の手順に従います:***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[詳細]** タブをクリックします。
4. **"ランチャーをパスワードで保護"** 設定で、**[はい]** を選択します。
5. **"ランチャーのパスワード"** 設定に、パッチで使用するパスワードを指定します。



メモ: *パスワードは大文字と小文字を区別します。*

QuickPatch パッケージに Update.exe アップデート起動ツールをビルドをするかどうかを指定する



プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

InstallShield では、InstallShield で、作成中の QuickPatch パッケージに Update.exe アップデート ランチャを作成するかどうかを指定できます。

必要に応じて自動的にターゲット システムで Windows Installer サービスを更新またはインストールする場合、Update.exe アップデート ランチャーが必要です。Update.exe ランチャーが必要になる場合についての詳細は、「パッチ時の考慮事項」を参照してください。

InstallShield で Update.exe アップデートランチャーを作成しないように構成した場合、.msp ファイルが作成されます。



タスク ***QuickPatch に Update.exe アップデート起動ツールを含めるかどうかを指定するには、以下の手順に従います:***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[共通]** タブをクリックします。
4. Update.exe を含める場合、**[Update.exe を作成する]** チェック ボックスを選択します。

   Update.exe を含めない場合、**[Update.exe を作成する]** チェック ボックスをクリアします。



ヒント: *[ビルドの設定] 領域にある [詳細] タブを利用しても、Update.exe を含めるかどうかを指定することができます。*