デバイスドライバーのインストール

注)このドキュメントは、InstallShield 2011 Premier Edition を基に作成しています。InstallShield 2011 以外のバージョンでは設 定名などが異なる場合もあります。

概要

InstallShield のインストーラは DIFX(Microsoft Windows Driver Install Framework) に準拠したデバイスドライバー のインストールをサポートしています。この記事では、基本のMSI 形式インストーラにてデバイスドライバーのインス トールを行う場合の手順についてご説明いたします。

A. デバイスドライバーウィザードによるドライバーの追加

デバイスドライバーウィザードを使用してプロジェクトへデバイスドライバーファイルを追加します。

1. [編成]-[セットアップのデザイン]ビューにて、[セットアップのデザイン]を右クリックして、[新しい機能]を選択しま す。追加された機能を右クリックして、[デバイスドライバーウィザード]を選びます



 デバイスドライバーウィザードが起動します。[デバイスドライバーパッケージ]ダイアログで、インストーラに含め る対象となるデバイスドライバーパッケージ(inf ファイル)を指定します。

デバイス ドライバー ウィザード	×
デバイス ドライバー パッケージ デバイスドライバーパッケージのロケーションを指定します。	©°°°
デバイス ドライバー パッケージ (.inf) の完全パスを指定します。また、パッケージが複数または特定のシス キテクチャをサポートする場合、[システム アーキテクチャ] リストでターゲットにするアイテムを選択します。	የድም እር
デバイス ドライバー パッケージ(D):	
システム アーキテクチャ(S): 汎用	•
< 戻る(B) 次へ(N) > キャンセル	

3. [デバイスドライバーファイル]ダイアログでは、指定したデバイスドライバーパッケージファイルに基づいてインス トーラに含められるファイルが表示されます。

ドイス ドライバー ウィザード					×
テバイス ドライバー ファイ ノ デバイス ドライバー ファイ	▶ 「ルを検証します。				
次のファイルがデバイス ドラ-	イバー パッケージに	含まれています:			
名前	サイズ	タイプ	更新日時	属性	
🗃 rdt158lm.cat	10.4 KB	セキュリティ カタログ	2008/03/17	A	
🎒 rdt 158 lm. inf	1.38 KB	セットアップ情報	2008/03/17	A	
🔺 rdt 158lm.icm	532 バイト	ICC プロファイル	2008/03/17	А	
•					Þ
	< 戻る	(B) 次へ(N):	> キャンセル	,	ヘルプ

4. [デバイスドライバー情報]ダイアログにて、デバイスドライバーの種類の確認、および適宜指定を行うオプション が設定可能です。

デバイス ドライバー ウィザード
デバイスドライバー情報 デバイスドライバーパッケージの追加情報を指定します。
 デパイスドライバータイプはデパイスドライバーパッケージから読み取られます。情報が正しいかどうか確認してください。 デパイスドライバーの種類 フラグアンドブレイドライバー DIFx 2.01 を使用したランタイムオブション 既存のデパイスドライバーを若に上書きする(Δ) [デパイスモンビューターへ相続]ダイアログを非表示にする(D) デパイスドライバーの「プログラムの追加と削除計」エンドリを作成しない(R) 署名がないドライバー ファイルと、ファイルが定りないドライバーをインストールする(U) アンインストール時(にドライバーに関連付けられているパイナリファイルを削除する(このフラグの使用は 注意が必要です。詳細はヘルブファイルを参照してください)(E)
< 戻る(B) 次へ(N) > キャンセル ヘルプ

● デバイス ドライバーの種類

デバイス ドライバーの種類は、パッケージファイル(inf ファイル)の [Version] セクション

DriverPackageType の記載により判別されます。

DriverPackageType 值	対応するドライバー タイプ
ClassFilter	クラスフィルター ドライバー
FileSystem	ファイルシステム ドライバー
FileSystemFilter	ファイルシステム フィルタードライバー
KernelModule	エクスポート ドライバー
KernelService	カーネルサービス ドライバー
Network	ネットワーク ドライバー
PlugAndPlay	プラグアンドプレイ ドライバー

※ DriverPackageType に記載が無い場合、種類は常に「Plug and Play ドライバー」と判別されます。

5. [プロジェクト全体のデバイス ドライバーの情報]ダイアログでは、デバイスドライバーのターゲットとなるマシンの アーキテクチャ(32BitCPU、64BitCPU 等)の指定を行います。

デバイスドライバー ウィザード	×
フロジェクト全体のデバイス ドライバーの情報 デバイス ドライバー パッケージの追加情報を指定します。	° ••
これらの設定はプロジェクト内のすべてのデバイスに適用します。	
🗖 ローカライズ済みインストール ランタイム ダイアログをすべて含める(L)	
「デバイス ドライバー マシン アーキテクチャー	
デバイス ドライバーは 32 ビット マシンをターゲットする(3)	
○ デバイス ドライバーは Itanium 64 ビット マシンをターゲットする(6)	
○ デバイス ドライバーは x64 ビット マシンをターゲットする(<u>X</u>)	
< 戻る(B) 次へ(N) > キャンセル ヘルコ	۶

メニューバー[ビルド]ー[ビルド(B)] または、F7 ボタンよりプロジェクトのビルドを行ないます。
 生成された MSI インストーラの動作を確認します。

B. 32bit 用 MSI ファイルと 64Bit 用 MSI ファイルを同じプロジェクトを使用して生成する ※手順 B は InstallShield 2011 以降の環境でお使いいただけます

MSI 形式インストーラーでは Windows Installer の仕様により 32Bit 環境と 64Bit 環境の両方に対応したパッケージ を作成することはできません。32Bit 用・64Bit 用 のデバイスドライバーを含んだインストーラーに関しても、それぞれ 別々のパッケージとして MSI インストーラを作成する必要があります。

InstallShield では、[リリースフラグ]の機能を使用することで、同一のプロジェクトを使用して 32Bit 用・64Bit 用のイン ストーラをビルドすることが可能です。

(上記内容の詳細につきましては *別資料「32bit 用インストーラと 64Bit 用インストーラを同一のプロジェクトで作成す*る」をご参照ください。)

1. [編成]-[セットアップのデザイン]にて、二つの機能を作成します。それぞれの機能が 32Bit 用・64Bit 用機能と なりますので、分かりやすい名称に変更します。



 それぞれの機能に対してリリースフラグを設定します。機能を選択して右のビュー[全般]-[リリースフラグ]に 任意の文字列を設定します。以下の図では、32Bit 用機能に「X86」というリリースフラグを設定しています。64Bit 用機能には「X64」というリリースフラグを設定します

田 🗀 インストール情報	🗆 🌏 セットアップのデザイン	×86Bins プログラム機能	
🖻 🧰 編成	🗈 🕗 x86Bins	□ 全般	
	🗄 🕙 x64Bins	表示名	{ID_STRING1} NewFeature1
一一之根能			
		リモート インストール	ローカルを優先する
■ ● アゴリケーション・データ		インストール先	[INSTALLDIR]
国 🔂 システム構成		インストール レベル	1
三一〇 サーバー構成		表示	閉じて表示する
□ 🛅 動作とロジック		アドバタイズ	アドバタイズを許可する
😐 🚞 ユーザー インターフェイス		必須	いいえ
車 🗀 メディア		リリースフラグ	X86
🖻 🚞 追加ツール		条件	0 条件
	1		

3. 作成した機能を右クリックして、[デバイスドライバーウィザード]を選択します。前述の手順Aを元に、それぞれの 機能に 32Bit 用デバイスドライバー・64Bit 用デバイスドライバーを含めます。

※注意※

ウィザードの最終ダイアログ [プロジェクト全体のデバイス ドライバーの情報]ダイアログの設定は、64Bit用デバ イスドライバーを追加する場合も、[デバイス ドライバーは32ビットマシンをターゲットする]を選択してください。

バイスドライバー ウィザード プロジェクト全体のデバイス ドライバーの情報	×
デバイス ドライバー パッケージの追加情報を指定します。	٦_
これらの設定はプロジェクト内のすべてのデバイスに適用します。	
□ □-カライズ済みインストール ランタイム ダイアログをすべて含める(L)	
「デバイス ドライバー マシン アーキテクチャー	
デバイス ドライバーは 32 ビット マシンをターゲットする(3)	
○ デバイス ドライバーは Itanium 64 ビット マシンをターゲットする(<u>6</u>)	
○ デバイス ドライバーは x64 ビット マシンをターゲットする(X)	
< 戻る(B) 次へ(N) > キャンセル	ヘルプ
< 床の(B))X^(N) > キャンセル	ヘルブ

このダイアログにて、[デバイス ドライバーはx64 ビット マシンをターゲットにする]を選んだ場合、自動的に64Bit 環境のみでの実行を許可するインストール条件が追加されてしまうため、32Bit環境で実行が行えなくなります。 インストール条件が追加されてしまった場合は、[インストール情報]-[一般情報]-[全般]-[インストール条件] にて、条件の削除を行ってください。(下図のダイアログで赤い丸印のボタンを押します)

条件	メッセージ	
IS_DIFX_DRIVER_OVERRIDE OR Msix64	{IDS_DIFX_AMD64} [ProductName]は、X64 プロセッサが必要です。
トの今年にゴロバテノン家族子を追加する。		
LOOKETTE2 D1 () 1 C/ALE 1 - C/E/11 / 0. /ロバティ(P):		
_IsMaintenance	▼ 追加(△)	
寅貸子(<u>O</u>):		
	」 注意tm(D)	

 32Bit 用デバイスドライバーのインストールを 64Bit 環境で許可しない場合、エラーのカスタムアクションを追加し ます。 [動作とロジック]ー[カスタム アクションとシーケンス]ビューにて、[カスタム アクションを]右クリックして、 [新しいエラー]を選びます。



5. 新規追加されたエラーカスタムアクションを選択して、右のウィンドウにて以下の設定を行います。

設定項目名	設定內容	説明
エラーメッセージ	[ProductName] は、X86 プロセッサが必要で す。「OK」をクリックするとウィザードが閉じま す。	表示するメッセージを指定します
インストール UI シーケンス	〈最初のアクション〉	
インストール UI 条件	(Not Installed) And (ISReleaseFlags><″X86″) And VersionNT64	ISReleaseFlags プロパティにリリースフ ラグ X86 が含まれている場合かつ 64Bit 環境で実行されている
インストール 実行シーケンス	〈最初のアクション〉	
インストール 実行条件	(Not Installed) And (ISReleaseFlags><″X86″) And VersionNT64	ISReleaseFlags プロパティにリリースフ ラグ X86 が含まれている場合かつ 64Bit 環境で実行されている

※ 64Bit 用 MSI インストーラは、32Bit 環境で動作しませんので、上記のカスタムアクションを別途追加する必要 はございません。

6. カスタムアクションが使用する DLL ファイルをリリースごとに切り替えるため、カスタムのパス変数を定義します。

[メディア]-[パス変数]ビューにて、[新規作成]選択します。以下のパス変数を追加します。

名前:PATH_TO_DIFX 定義された値:<ISProductFolder>¥redist¥0409¥i386

田 🛅 インストール情報	🔢 新規作成 🖌 🗙 😽	📰 🗉 🛛 検索グリッド 👘 🕖	
田 🛅 アプリケーション データ	日 名前 マ	定義された値	現在の値
🖻 🧰 システム構成	O WindowsFolder		C:¥WINDOWS¥
□ □ □ サーバー構成	SystemFolder		C:¥WINDOWS¥system32¥
□ □ 動作とロシック	ProgramFilesFolder		C:¥Program Files¥
日 〇 メディア	DATH_TO_V1.0_FILES	E:¥My Documents¥My host¥ ! Sample¥SAMP	E:¥My Documents¥My host¥ ! Sample¥SAMPLE_FI
	DATH_TO_RDT158L	E:¥My Documents¥My host¥ ! Sample¥rdt15	E:¥My Documents¥My host¥ ! Sample¥rdt158lm_vi
🅐 アップグレード	DATH_TO_DIFX	<isproductfolder>¥redist¥0409¥i386</isproductfolder>	C:¥Program Files¥InstallShield¥2011¥redist¥0409¥i3
	DATH_TO_ALARM.6	E:¥My Documents¥My host¥ ! Sample¥Alarm	E:¥My Documents¥My host¥ ! Sample¥Alarm.6.00
Mar 11 11 11 11 11 11 11 11 11 11 11 11 11	ISProjectFolder		C:¥InstallShield 2011 プロジェクト¥Driver_3264combi
	ISPROJECTDIR		C:¥InstallShield 2011 プロジェクト¥Driver_3264combi

 カスタムアクションが使用する DLL ファイルの参照先を手順6 で作成したパス変数を使用する記述に変更します。[動作とロジック]-[カスタム アクションとシーケンス]ビューにて、以下のカスタムアクションの[DLL ファイル 名]を修正します。

カスタムアクション名	変更前	変更後
MsiCleanupOnSuccess	<isproductfolder>¥redist¥0409¥i386¥Difxapp.dll</isproductfolder>	<path_to_difx>¥Difxapp.dll</path_to_difx>
MsiInstallDrivers	<isproductfolder>¥redist¥0409¥i386¥Difxappa.dll</isproductfolder>	<path_to_difx>¥Difxappa.dll</path_to_difx>
MsiProcessDrivers	<isproductfolder>¥redist¥0409¥i386¥Difxapp.dll</isproductfolder>	<path_to_difx>¥Difxapp.dll</path_to_difx>
MsiRollbackInstall	<isproductfolder>¥redist¥0409¥i386¥Difxappa.dll</isproductfolder>	<path_to_difx>¥Difxappa.dll</path_to_difx>
MsiUninstallDrivers	<isproductfolder>¥redist¥0409¥i386¥Difxappa.dll</isproductfolder>	<path_to_difx>¥Difxappa.dll</path_to_difx>

※ 指定する DLL ファイル名は2種類(Difxapp.dll と Difxappa.dll)ある点にお気をつけください

	共通	
ISPrevenceowingrade	MeiFleanupOnSuccess 17.51.	<u>7/19-27</u>
 ISSelfRegisterCosting 	DLL ファイル名	<path_to_difx>¥Difxapp.dll</path_to_difx>
SSelfRegisterFiles	関設治	CleanupOnSuccess
S ISSelfRegisterFinalize	戻り値の処理	同期(終了コードを確認)
🔤 🛐 ISUnSelfRegisterFiles	スクリプト内実行	即時実行
	実行スケジュール	常に実行
	64 ビット スクリプトの使用	いいえ
- 🔊 MsiProcessDrivers	インストール UI シーケンス	<シーケンスになし>
- 💁 MsiRollbackInstall	インストール UI 条件	
🛐 MsiUninstallDrivers	インストール実行シーケンス	次の後:InstallFinalize

カスタムアクションアイコンの下に該当のカスタムアクションが見えない場合には、[すべてのカスタムアクション を表示]をクリックしてください

 [メディア]-[リリース]ビューにて、製品構成を二つ作成します。それぞれの製品構成に各環境に対応した[製品構成フラグ][テンプレート概要]を設定します。(以下の画像では、64Bit 環境用の製品構成に製品構成フラグ [X64]、テンプレートの概要に[x64;1041]を指定しています)

 ●・●・リリース ●・● binx64 ●・● リリース 1 ●・● binx86 ●・● binx86 ●・● binx86 ● ● binx86 	全般 複数インスタンス		
	binx64 製品構成		
	製品名		
	製品バージョン		
	パッケージコード		
	パッケージコードの生成	(tt)	
	製品構成フラグ	X64	
	サブジェクト		
	タイトル		
	テンプレートの概要	x64;1041	
	コメント		
	スキーマ		
	製品コード	×	

※ 製品構成フラグは手順2で指定したリリースフラグと同等の文字列を使用します。例えばリリースフラグとして 「X86」が指定された機能に関しては、製品構成フラグを[X64]としてビルドを行った場合、インストーラに含められ ません。 9. 設定を行った製品構成の配下にリリースを追加します。作成したリリースを選択して、右のビュー[ビルド]タブー [パス変数のオーバーライド]をダブルクリックします。表示されるダイアログにて、手順6で追加したパス変数 PATH_TO_DIFX にチェックをつけます。

◎ パス変数のオーバーライド	×
オーバーライドするパス変数を選択:	
PATH_TO_DIFX	
ヘルプ	OK キャンセル

10. [パス変数のオーバーライド]項目の下に追加される[PATH_TO_DIFX]パス変数のエントリに対して、32bit 用リリ ース・64Bit 用リリースに対してそれぞれ以下の階層を指定します。

32Bit 用リリース: <ISProductFolder>¥redist¥0409¥i386

リリースフラグ	
パス変数のテスト値を使用する	いいえ
曰 パス変数のオーバーライド	1 パス変数
PATH_TO_DIFX	<isproductfolder>¥redist¥0409¥i386</isproductfolder>
未使用のディレクトリを保持する	いいえ
Trialware ビルドの無効化	いいえ
DIM の検証	いいえ

64Bit 用リリース: <ISProductFolder>¥redist¥0409¥x64

リリースフラグ		
パス変数のテスト値を使用する	いいえ	
曰 パス変数のオーバーライド	1 パス変数	
PATH_TO_DIFX	<isproductfolder>¥redist¥0409¥x64</isproductfolder>	
未使用のディレクトリを保持する	いいえ	
Trialware ビルドの無効化	いいえ	
DIMの検証	いいえ	•

11. リリースを右クリックして[ビルド]を選択します。作成したリリースをそれぞれビルドします。

生成されたインストーラを、32Bit、64Bit 環境で実行して動作を確認します。

更新日: 2012/12/7