FLEXERA SOFTWARE

FLEXERA SOFTWARE®
InstallShield®

InstallShield 2016 InstallScript
リファレンス ガイド

法的情報

文書名： InstallShield 2016 InstallScript リファレンス ガイド

部品番号： ISP-2300-RG00

製品のリリース日： 2016 年 8 月

著作権情報

知的財産

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

(米国内向け) 制限付権利に関する表示

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

目次

InstallScript 言語リファレンス

InstallShield を利用すると、シンプルかつパワフルなプログラム言語 InstallScript を利用して簡単にインストールを作成することが可能です。InstallScript は C 言語に似ています。InstallScript は定義済みフォーマットと統制された構文を持ちます。特定のプロパティを持つ決まったデータ型を利用します。また、カスタム関数を作成することもできます。

しかし、InstallScript が C 言語が持つプログラム機能のすべてを備えているわけではありません。InstallScript の唯一の目的はインストールを作成することです。また、インストレーションにおいて世界中で InstallScript 以上に優れたプログラム言語はありません。開発者のプログラミング経験には関係なく、InstallScript を使ってインストレーションのビルドを素早く習得することができます。

プロジェクト・*InstallScript 関数、イベント、変数の中には、特定のプロジェクト タイプに使用が限定されているものがあります。*

テーブル 1・InstallScript 言語リファレンス

Section	説明
統合コンパイラ	InstallScript 統合コンパイラについての一般情報です。
コマンドライン コンパイラ	DOS プロンプトまたは DOS バッチファイルから起動できる InstallScript コマンドライン コンパイラについての詳細が含まれています。
セットアップ スクリプト	InstallScript 言語およびスクリプトの構造を紹介します。
言語キーワード	InstallScript がコマンドとして利用する言語である言語キーワードのバックグラウンドを提示します。言語キーワードはアクションを操作するために InstallScript コンパイラによって解釈されるか、またはステートメントの一部として認識されます。

テーブル 1 ・ InstallScript 言語リファレンス (続き)

Section	説明
定義済み定数	InstallScript で予約されている各定義済み定数の識別と説明です。これらの定数は、ビルトイン関数に渡されてビルトイン関数によって戻される特定のリテラル値を表します。
定義済みのスクリプト変数	InstallScript で使用できるスクリプト変数についての情報です。
データ型および定義済み構造	InstallScript でサポートされているデータ タイプと定義済み構造についてのコンテンツです。
プリプロセッサ ディレクティブ	スクリプトがコンパイルされる時に実行される InstallScript コンパイラに対する指示である、プリプロセッサ命令について説明します。プリプロセッサ命令はコンパイラに対して、コンパイル内の別のソースファイルを含む、定数を定義する、コンパイルタイム条件に基づいてステートメントを含む、または除く、そしてユーザー定義のエラーメッセージを表示するといった指示を出します。
フロー制御	スクリプト内での実行のフロー制御方法についての情報を提供します。
イベント ハンドラー	InstallScript プロジェクトインストレーションプログラムは InstallScript エンジンによって制御され、決められた順序で一連のイベントが生成されます。
関数	関数インストールスクリプトで使用できる異なるタイプの関数の説明です。InstallScript で利用できるビルトイン関数についての詳細およびその例も、ここで参照することができます。
演算子	InstallScript でサポートされている演算子についての情報を提供します。
オブジェクトおよびオブジェクト ハンドラー	InstallScript でサポートされているオブジェクトと、スクリプトコードの残りからエラー処理を分離する方法について説明します。

メモ・*関数の中には、InstallShield Professional で利用可能でも InstallShield では利用できないものもあります。関数のリストは、サポートされていない関数 を参照してください。*

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

統合コンパイラ

リリースをビルドしないでコンパイルする場合、InstallShield に統合されている InstallScript コンパイラを使用します。

タスク ***スクリプトをコンパイルするには、以下の手順に従います：***

[ビルド] メニューで **[コンパイル]** をクリックします。

[出力] ウィンドウにコンパイラ メッセージが表示されます。

コマンドライン コンパイラ

InstallShield 内部から起動できる統合コンパイラの他にも、InstallShield には DOS プロンプトまたは DOS バッチ ファイルから起動できるコマンドライン コンパイラが含まれています。このプログラムは **Compile.exe** と呼ばれ、次のフォルダーに保存されています。

*InstallShield Program Files フォルダー***System**

インストール プロジェクトのデザインが完了すると、**Compile.exe** を使用して、InstallShield 内部からスクリプトをコンパイルする場合とは異なるオプションを使ってインストール スクリプトをコンパイルすることができます。

Compile.exe で使用できる構文やコマンドライン パラメーターおよびスイッチについては、「Compile.exe」を参照してください。

***メモ・ISCmdBld.exe** を使用して、コマンドラインからリリースをビルドすると、ビルド エンジンは自動的にスクリプトをコンパイルします。したがって、InstallShield のプロジェクトに指定されているコンパイラ オプション以外のオプションを使用する必要がない場合、**Compile.exe** を直接使用する必要はありません。詳細については、「ISCmdBld.exe」を参照してください。*

セットアップ スクリプト

セットアップスクリプトとは、イベントハンドラー、またこれらのイベントハンドラーが呼び出す関数、そしてイベントハンドラーと関数が利用するデータの集合です。これらの要素はシンプル且つパワフルなプログラム言語、InstallScript 言語で表記されます。InstallScript は C 言語に似ています。InstallScript は定義済みフォーマットと統制された構文を持ちます。特定のプロパティを持つ決まったデータ型を利用します。また、カスタム関数を作成することもできます。

しかし、InstallScript が C 言語が持つプログラム機能のすべてを備えているわけではありません。InstallScript の唯一の目的はセットアップを作成することです。そして効率的に、効果的に実行します。

開発者のプログラミング経験には関係なく、InstallScript を使ってセットアップのビルドを素早く習得することができます。

InstallScript の制限事項

コンパイルされたスクリプト ファイル (**setup.inx**) の制限事項は以下のとおりです :

- ステートメントの最大数 : 約 4,294,967,295 (この制限を越えると、セットアップの初期化中にエラー ー 5009 が発生する場合があります。)
- グローバル変数の最大数 : 約 196,605 (数値 65,535、バリアント 65,535、文字列 65,535)
- typedefs の最大数 : 約 65,535
- prototypes の最大数 : 約 65,535
- 関数の最大数 : 約 65,535
- 関数ごとのステートメントの最大数 : 約 65,535
- 関数ごとのローカル変数の最大数 : 約 196,605 (数値 65,535、バリアント 65,535、文字列 65,535)

スクリプト ファイル (.rul) の制限は次の通りです :

- 最大行幅 : 1,024 文字
- ネストされたインクルード ファイルの最大数 : 80
- インクルード ファイルの最大数 : 2,048
- 識別子の長さ制限 : 63 文字
- マクロ展開の最大数 : 100
- マクロ展開テキストの長さ制限 : 256 文字
- ファイル名の長さ制限 : 256 文字
- ネストされた #if ステートメントの最大数 : 10
- 関数ごとのパラメーターの最大数 : 16

これらの .rul 制限が 1 つ以上超えると、コンパイラ エラーが発生します。

ヒント・前述の制限のいずれかに到達した場合は、コードを削除するか、インストール スクリプトを複数のプロジェクトに分割し、個別のインストールを作成してメイン インストール（親インストール）から子インストールを呼び出す方法で、インストール スクリプトのサイズを削減してください。

スクリプトの構造

各スクリプトには宣言と関数ブロックが含まれます。宣言は関数宣言の前、または関数ステートメントとその関数の begin ステートメントの間に配置することができます。

スクリプトの大まかな概要を以下に示します。

```
// 定数定義、グローバルデータ宣言、そして関数宣言

// 関数ブロック
```

宣言

各スクリプトはグローバルデータ宣言で始まります。ここでは、定数を定義し、利用する各グローバル変数とユーザー定義関数を宣言します。宣言は InstallScript コンパイラに対して、スクリプトが一覧にある項目を後で利用することを指示します。宣言はまた、関数とその属性または値との関連もビルドします。ビルトイン関数に関しては、InstallScript コンパイラが既に関数名を認識しているので宣言の必要はありません。

次に示すのは定数定義、データ宣言、そして関数宣言の一例です。

```
// 定数定義
#define PRODUCT "InstallShield"
#define LIMIT   100

   // 変数宣言
   CHAR  cVal;
   NUMBER nVal;
   STRING szName;

   // 関数宣言
   prototype DisplayMsg (NUMBER, STRING);
   prototype GetName (BYREF STRING);
```

プログラム ブロック

プログラムブロックは InstallShield Professional 5.5 またはそれ以前を使って書かれたスクリプトで利用されます。プログラムブロックは InstallScript カスタムアクション、またはイベントドリブン型スクリプトでは利用することができません。実行されるコードはイベントハンドラーとエントリポイント関数のみです。

```
program

   // イベントドリブン型スクリプトでは、プログラムブロックはオプションで空白の状態です。

endprogram
```

関数ブロック

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

プロトタイプ ステートメントを使って宣言されたすべての関数は、セットアップ スクリプトで endprogram キーワードに続く関数ブロック内で定義する必要があります。追加グローバルデータ宣言は、関数ブロック内の endproram ステートメントと最初の関数宣言の間または関数宣言の間で行うことができます。しかし関数ブロック内で宣言されたデータは、データ宣言の後に定義された関数のみが利用することができます。

識別子

演算子とは、スクリプト内の定数、変数、そして関数を象徴するのに作成する名前です。識別子を作成する際、次の構文ルールを遵守してください :

- 識別子の長さに決まりはありませんが、最初の 63 文字のみが重要です。
- 識別子の最初の文字はアルファベット (a-z、A-Z) またはアンダースコアでなくてはなりません。
- 残りの文字はアルファベット (a-z, A-Z) 、数字 (0-9)、あるいはアンダースコアの何れも可能です。
- 各識別子は固有である必要があります。InstallScript で予約語となっている識別子を作成しないように注意してください。

構文の区切り規則

その他のプログラム言語と同様に、InstallScript にもその利用法を統一する構文規則があります。InstallScript の基本の構文は C プログラム言語のそれに似ています。

次の区切り規則は、スクリプトのすべてのセクションに適用されます :

- ほとんどのステートメントはセミコロン (;) で終了します。これには end;、exit;、および return; など多くの 1 語ステートメントが含まれます。
- #define や #include といったプリプロセッサ ステートメントはセミコロンで終了してはなりません。
- キーワード program、endprogram、および begin は単独で別々の行に配置され、区切りを受け付けません。各関数ブロックで始まる関数行は区切りを受け付けません。
- start: または starthere: の様に、コロン (:) を使ってラベルを終了します。
- パラメーターリストを括弧で囲みます。複数のパラメーターはカンマで区切ります。

コメントの書き込み

InstallScript は、スクリプトでのコメント作成方法を 2 種類提供します。どちらかの方法を使って説明テキストをスクリプトへ追加または除外する、またはテストやデバッグの目的でスクリプトの特定部分を 「コメントアウト」することができます。

注意 *・ひとつの例外を除いて、コメントはスクリプトの任意の場所で始めることができます : #ifdef ステートメントや #ifndef ステートメントと同じ行にコメントを配置することはできません。必要な場合、コメントをこれらのステートメントの前または後に書かなくてはなりません。そうでない場合、コンパイラはエラーを戻します。*

テキストのブロック

コメント作成方法の 1 つはテキストのブロックを一対の文字 /* と */ で括る方法です。この方法はコメントを複数行にわたって書き込むのに有効です :

```
/* これは InstallScript 関数 PlaceBitmap の利用法を
＊示すサンプルコードの行です。*/
```

一行ごと

2 つめは、行に文字 // を挿入する方法です。コンパイラはその行についてのみ 2 つのスラッシュの右側部分をすべて無視します。

```
// これは InstallScript 関数 PlaceBitmap を
// 説明するサンプルコードのラインです。
```

空白の使用

C 言語や、その他のプログラム言語と同様に、InstallScript は文字列リテラル内以外は空白スペース (スペースやタブ、改行) を認識しません。スクリプトを読みやすくするために空白スペースの利用をお勧めします。

空白スペースを利用しないコード

例えば、次のコード セクションは詰まっていて判読しづらいです :

```
#define DISK_DRIVE "C:¥¥"
    STRING szDrive, svString;
    NUMBER nSpace, nResult;
szDrive = DISK_DRIVE;
nSpace = GetDiskSpace(szDrive);
nResult = NumToStr (svString, nSpace);
if (nResult < 0) then
MessageBox ("NumToStr が失敗しました。", SEVERE);
abort;
endif;
SprintfBox(INFORMATION, " 情報 ", " ディスク容量 : %s", svString);
```

空白スペースを利用したコード

字下がりと共に空白スペースを追加すると、同じコードが大変読みやすくなります :

```
    #define DISK_DRIVE "C:¥¥"

    STRING szDrive, svString;
    NUMBER  nSpace, nResult;

    szDrive = DISK_DRIVE;
    nSpace  = GetDiskSpace(szDrive);

    nResult = NumToStr (svString, nSpace);
```

```
if (nResult < 0) then
    MessageBox ("NumToStr が失敗しました。", SEVERE);
    abort;
endif;

SprintfBox(INFORMATION, " 情報 ",
        " ディスク容量 : %s", svString);
```

ハンガリー表記

InstallShield ヘルプトピックはハンガリー表記の拡張された形式を採用しています。これは、短く、データ型を示す小文字のプレフィックスを含む命名規則です。例えば、iPointSize は整数変数を象徴し、szFileName は文字列変数を示します。

ハンガリー表記は、スクリプト例ですべての変数のデータタイプを示すのに利用されています。関数構文説明では、パラメーターで渡される可能性のあるデータ型を示すパラメーター名にハンガリー表記が利用されています。たとえば、BatchDeleteEx の構文説明では 2 つのパラメーターを受け付けることを示します :

```
BatchDeleteEx ( szKey, nOptions );
```

szKey と認識される最初のパラメーターは文字列変数または定数が可能です。nOptions と認識される 2 番目のパラメーターは数値変数または定数が可能です。

変数パラメーター

この様に変数パラメーターが必要な場合、言語リファレンスは 2 文字セットとなった特定のプレフィックスを採用します。

- 1 番目の文字はデータの種類を示します。
- 2 番目の文字、v は変数です。

GetDir の構文説明では、最初のパラメーターは文字列変数または定数が可能ですが、2 番目のパラメーターは変数でなくてはなりません。

```
GetDir ( szPath, svDir );
```

変数パラメーターを必要とする関数は、一般的にこれらのパラメーターで呼び出し元へデータを戻します。

プレフィックス テーブル

ハンガリー表記は変数の種類を認識するのに有効です。スクリプトで変数名を作成する場合には、是非ハンガリー表記をご利用下さい。下のテーブルは InstallShield で利用される各プレフィックスについて説明します。

テーブル 1 ・ プレフィックス テーブル

前に追加する	データ型	関数構文で使用される状況
b	ブール型 (BOOL)	ブール型定数、リテラル、または変数。
bv	ブール型 (BOOL)	ブール型変数のみ。定数、リテラルは無効です。
c	文字 (CHAR)	文字定数、リテラル、または変数。

テーブル 1 - プレフィックス テーブル (続き)

前に追加する	データ型	関数構文で使用される状況
const	定数	定数またはリテラル。変数は無効です。
h	ハンドル (HWND)	ハンドル変数。
i	整数 (INT)	整数定数、リテラル、または変数
l	長い整数 (LONG)	長い整数定数、リテラル、または変数
lv	長い整数 (LONG)	長い整数変数のみ。定数、リテラルは無効です。
list	リスト (LIST)	リスト変数。
n	数値 (NUMBER)	数値定数、リテラル、または変数
nv	数値 (NUMBER)	数値変数のみ。定数、リテラルは無効です。
p	ポインター (POINTER)	ポインター変数。
pstruct	定義済み構造タイプへのポインター	利用されません。
s	短い整数 (SHORT)	長い整数定数、リテラル、または変数
sz	文字列 (STRING)	数値定数、リテラル、または変数
sv	文字列 (STRING)	文字列変数のみ。定数、リテラルは無効です。
struct	定義済み構造タイプ	利用されません。

エスケープシーケンス

エスケープ シーケンスは文字列にタブや改行、および引用符など特定の特殊文字を挿入するのに使用する一連の文字です。InstallScript のエスケープ シーケンスは C で使うそれとよく似ています。エスケープ文字と呼ばれる円記号で始め、特別な意味を持つ 1 つ以上の文字が後に続きます。円記号の後にエスケープ シーケンスで使用する以外の文字が続く場合、円記号は無視されます。

文字列に改行文字を挿入する

最もよく使用されるエスケープ シーケンスは ¥n で、これは文字列に改行文字を挿入します。「これは第 1 行目で、これは第 2 行目です」という文字列は、一行で表示または印刷されます。ただし、「これは第 1 行目で、¥n これは第 2 行目です」という文字列は、下のように表示または印刷されます。

これは第 1 行目で、
これは第 2 行目です。

メモ・¥n エスケープ シーケンスが使えるのは、複数行の静的テキスト フィールドのみです。たとえば AskText の szQuestion 引数に ¥n を挿入して、文字列を手動でフォーマットできます。¥n は、MessageBox と SprintfBox でも使用できます。

改行のエスケープ シーケンスは大文字と小文字を区別します。つまり ¥N と入力しても改行文字は 挿入されません。

パーセント記号 (%) も InstallScript で特別の機能があります。これは変数に保管される値の画面での表示方法を示す Sprintf や SprintfBox などの関数と一緒に使われる一連の文字、書式指定子 の最初の文字として使われます。

サポートされているエスケープシーケンス

以下の表に、InstallScript でサポートされているエスケープ シーケンス一覧を示します。

テーブル 2・サポートされているエスケープシーケンス

エスケープシーケンス	実行されるアクション
¥n	ラインフィードを挿入します。
¥'	文字列に単一の引用符を挿入します。
¥"	文字列に二重引用符を挿入します。
¥r	改行のみを挿入します。ラインフィードは挿入されません。
¥t	タブ文字を挿入します。
¥ooo	ASCII 文字 (整数ではない) を、8 進表記で示します。
¥¥	円記号を挿入します。

UNC (Universal Naming Convention) パス

InstallScript 文字列で UNC (Universal Naming Conversion) パスを指定するには、パス名の最初に *2* つの円記号 エスケープ シーケンス (つまり、4 つの円記号 – ¥¥¥¥) を使用して、ダブル円記号を作成する必要があります。たとえば、¥¥MyServer¥Public¥Readme.txt というパスを指定するには次のように入力します :

"¥¥¥¥MyServer¥¥Public¥¥Readme.txt"

二重引用符の埋め込み

文字列リテラルの一部として二重引用符を挿入するには 2 種類の方法があります。二重引用符から文字列リテラルを始める場合、¥" エスケープ文字を使って二重引用符を埋め込みます。単一引用符のあとに二重引用符を入力してリテラルを始めることもできます。

```
// これら 2 つのステートメントは両方、埋め込み二重引用符をもたらします
szQuote1 = " 誰が、¥" 三日坊主は負け犬だ ¥" と言いましたか ?";
szQuote2 = ' " やめた。" と言った人です。'; ";
```

単一引用符を埋め込むには、¥' エスケープシーケンスを利用するか、文字列リテラルを二重引用符で始めます :

```
// これら 2 つのステートメントは両方、埋め込み単一引用符をもたらします
szQuote1 = ' 誰が、¥' 良い人は最後に完了します ¥' と言いましたか ?';
szQuote2 = "' 私の勝ち。' と言った人です。";
```

メモ・セットアップスクリプトには標準 U.S. キーボードで、セミコロン (;) キーの右側にある標準の引用符 (" と ') を利用しなくてはなりません。たとえば、このヘルプファイルのスクリプト例以外の部分で使われている活版印刷で使う引用符 (“”、は利用できません。

書式指定子

書式指定子は、Sprintf や SprintfBox の関数と一緒に使用して、変数に保管される値の表示をコントロールします。書式指定子はパーセント記号 (%) で始め、少なくとも 1 つか 2 つの文字を後に続けます。書式指定子は下の形式に従います。

% [–] [#] [0] [width] [.precision] type

形式指定の各フィールドには特定の形式オプションを表す 1 つの文字または数字が入ります。たとえば タイプフィールドは、Sprintf または SprintfBox が関連付けられた引数を文字、文字列または数字として解釈するかどうかを判断します。最初の文字の % とタイプフィールドは両方とも必須です。カッコで括られた項目はオプションです。最も簡単な形式指定には、パーセント記号と %s などの種類文字が入ります。

次の例では、svString の値がメッセージボックスに表示されます。svFormat に割り当てられる形式指定子の %s は、svString の値を文字列として表すべきであることを SprintfBox に示します。

```
STRING szTitle, szFormat, szString;

szTitle  = " フォーマット指定のデモンストレーションを行います。";
szFormat = "%s";
szString = " これが文字列です。";

SprintfBox(INFORMATION, szTitle, szFormat, szString);
```

svFormat に割り当てられる値には、変数値と一緒に表示されるリテラル文字 (エスケープ シーケンス を含む) が含まれます。次の例では、ラベルの ID が数値変数の左に表示されます。nNumber = 100;

```
STRING szTitle, szFormat;
NUMBER nNumber;

szTitle  = " フォーマット指定のデモンストレーションを行います。";
szFormat = "nNumber = %d.";
nNumber = 100;

SprintfBox(INFORMATION, szTitle, szFormat, nNumber);
```

メモ・パーセント記号を印刷するには、svFormat に割り当てられた文字列にパーセント記号を 2 つ挿入する必要があります。印刷する数字が 100 と想定すると、次の形式特定文字列に “nNumber = 100%” と表示されます :

svFormat = "nNumber = %d%%."

テーブル 3 ・書式指定子フィールド

フィールド	意味
-	パーセント文字の後にハイフンを含めると、出力値は左に配置されてブランクまたはゼロのフィールドの幅の右にパディングされます。このフィールドを省略すると、出力値は右揃えになり、左にパディングされます。
#	この記号は、16 新数値の前に 0x (小文字) または 0X (大文字) を付けます。
0	ゼロの出力値をパディングしてフィールド幅を埋めます。このフィールドを省略すると、出力値はブランクでパディングされます。
width	このフィールドに配置する最低文字数を入力します。width (幅) フィールドを負ではない整数で入力します。幅指定を入力すると、値が切り捨てられることはありません。出力値の文字数が指定された幅より大きいか、このフィールドを省略すると、precision (精度) フィールドの値によって、値の各文字が表示されます。
precision	このフィールドに配置する最低桁数を入力します。引数の桁数が入力する精度値より少ない場合、左の出力値はゼロでパディングされます。桁数が精度値を越えた場合、値は切り捨てられません。この値にゼロを入力するか完全に省略するか、あるいはピリオド (.) が後に続く数字がない状態で表示されている場合、値は 1 に設定されます。文字列の場合、最大文字数が変換されます。
種類	対応する引数を文字か文字列か数字でフォーマットします。2 つの書式指定子文字の組み合わせが表示されると、そのうちの 1 つまたはもう 1 つを使用できますが、同時に両方を使用することはできません。これは必須フィールドです。このフィールドには次の文字の 1 つを入力する必要があります。 • c ー CHAR 型の単一文字をフォーマットします。Sprintf 関数はゼロの数値を持つ文字を無視します。 • d, i ー INT または NUMBER 型の単一整数をフォーマットします。 • ld, li ー LONG 型の単一の符号なし 16 進数整数をフォーマットします。 • lx, lX ー LONG 型の単一の符号なし 16 進数整数をフォーマットします。 • s ー 文字列 (STRING 型) をフォーマットします。

各書式指定子には一致する変数があります。変数は文字列の後に左から右にリストされます。最初の変数は文字列の最初の書式指定子と一致し、2 番目の変数は文字列の 2 番目の形式指定子と一致します。ランタイム時に InstallShield は各変数の内容を、一致する形式指定子の場所にある文字列に挿入します。

予約語

予約語予約語と予約文字は InstallScript では特殊な意味を持ち、これらは意図された目的以外で利用することができません。InstallScript には次のようなクラスの予約語があります :

- 関数

- 言語キーワード
- 定義済み定数
- システム変数
- イベント ハンドラー
- 定義済みのスクリプト変数

言語キーワード

言語キーワードはスクリプト内で InstallScript がコマンドとして利用する言語です。言語キーワードはアクションを操作するために InstallScript コンパイラによって解釈されるか、またはステートメントの一部として認識されます。既に定義されている目的以外には次のキーワードを利用することはできません (たとえば、これらのキーワードを変数名として利用することはできません):

abort

スクリプトが abort ステートメントを検出した場合、セットアップが終了します。abort ステートメントは、エンド ユーザーが Ｅｓｃ キーまたは InstallScript ダイアログの [キャンセル] ボタンを押してインストールを中止した場合に、InstallShield デフォルト exit ハンドラー (OnCanceling) でも検出されます。

メモ・*abort ステートメントはインストレーションを終了し、サイレントモードにてアンインストーラーを実行することで中止されたインストレーションを削除します。exit ステートメントはインストレーションを中止しますが、ターゲットシステムからは何も削除しません。*

abort ステートメントが OnFirstUIAfter イベントの後に開始した場合、ロールバックは呼び出されません。

BOOL

ブール値データ、TRUE (1) または FALSE (0) のどちらか。この種類の変数は、指定値以外の値を格納するために使用できません。C++ と同様に、InstallScript ではゼロ以外の値を TRUE と評価します。値がゼロの場合のみ、FALSE と評価されます。通常は、1 の値で TRUE を示します。

cdecl

プロジェクト・*この情報は、InstallScript プロジェクトに適用します。*

cdecl キーワードは cdecl 呼び出し規則を使う外部 DLL 関数を宣言するときに利用します。例 :

```
prototype cdecl POINTER Msvcrt.memcpy( byref string, pointer, long );
```

InstallShield Professional の以前のバージョンではセットアップエンジンは常に stdcall 規則を使用していましたが、一貫性のない DLL 規則を無視することがありました。

ほとんどの Windows API 関数は stdcall (WINAPI) 呼び出し規則を利用します。呼び出し規則についてのより詳しい情報は、Microsoft マニュアルをご覧下さい。

exit

セットアッププログラムが実行しているスクリプトで exit ステートメントを検出すると、そのプロセスは終了します。各セットアップ スクリプトは、最大で 1 つの exit ステートメントを含みます。インストールが完了する前に終了する原因となり得る条件式をスクリプトに含む場合、exit の代わりに abort を利用しなくてはなりません。

export

セットアップエンジンが直接呼び出す関数のプロトタイプは何れも export としてマークしなくてはなりません。次はその一例です :

```
export prototype NewFeature1_Installing();
```

external

キーワード external は予約されており、利用できない場合があります。

for...endfor

for ステートメントは、単数または複数のステートメントを決められた回数実行するために利用します。これはキーワード for と共に、for 構造にあるステートメントを実行する回数を指定する式で始まります。for 構造はキーワード endfor で終了します。

***メモ・** for ステートメント自体はセミコロンで終了されることはなく、セミコロンは endfor ステートメントの後に必要です。*

for...endfor の利用法

次の例では、関数 MessageBox が 10 回呼び出されます。最初に渡すとき、iCount が 1 に設定されます。1 は指定された範囲内 (1 から 10) にあるので、メッセージボックスが表示されます。そして iCount は 1 ごとに増やされ、for ステートが再び解決されます。このとき、iCount = 2 (指定された範囲内) で、メッセージボックスが 2 度目に表示されます。

iCount が 10 回渡された後、その値は 11 となります。この値は指定された範囲を超えるので、for ステートメントは終了します。

```
for iCount = 1 to 10
    MessageBox (" これは 10 回表示されます。", INFORMATION);
endfor;
```

増分を調整する

デフォルト増分値は 1 ですが、キーワード step を利用して増分を調整することができます。下の例では、ループが実行されるたびに step が iCount の値を 10 増加させます。最初に渡すとき、iCount = 10 で、2 回目は iCount = 20、3 回目は iCount = 30、といった要領です。

```
for iCount = 10 to 100 step 10
    MessageBox (" これは 10 回表示されます。", INFORMATION);
endfor;
```

高い番号から低い番号へのカウントダウン

キーワード to の代わりに downto を利用して、高い番号から低い番号へカウントダウンすることはできません。次の例では、メッセージボックスが 3 回表示されます。最初にループが始まったとき、 j は 20 に設定されています。

downto は制御する変数が減少するよう、また step 5 がループごとに 5 減少と指定しているので、ループが 2 回目に入ったとき j は 15 となります。3 回目には、j は 10 となります。

```
for j = 20 downto 10 step 5
    MessageBox (" これは 3 回表示されます。", INFORMATION);
endfor;
```

***メモ・** for ステートメント内でラベルを定義することはできません。*

goto

goto キーワードは指定したラベルの直後にあるステートメントに直接ブランチするのに利用されます。次の部分コードでは、goto ステートメントによって AskText ステートメントを使ってスクリプトが続行されます。

```
名前 :
AskText(" 会社名 :", "", szSrc);

if (szSrc = "") then
    MessageBox(" 会社名を入力してください。", SEVERE);
    goto Name;
endif;
```

メインプログラムの goto ステートメントでは、メインプログラムで宣言されたラベルを指定しなくてはなりません。関数内の goto ステートメントはその関数内で宣言されたラベルを指定しなくてはなりません。

***メモ・** try...catch...endcatch ステートメント内で goto ステートメントを使用することはできません。*

if

スクリプトが 複数のオプションから選択できるようにするには、if ステートメントを利用します。if ステートメントは下の例の通り、キーワード if、評価する条件、キーワード then と endif そして後にはセミコロンが続きます。

```
if (condition) then
    // 条件が true の場合、ステートメントが実行されます。
endif;
```

条件は次のうちいづれかが可能です :

- ブール型または 整数定数、変数、またはリテラル。
- ブール型または整数結果をもたらす式。
- 整数結果を戻す関数。

条件の周りの括弧はオプションですが、読み易くするために利用することをお勧めします。

***ヒント・** 多くの InstallScript 関数はそれが失敗した際に負の値を戻します。InstallScript 関数の結果を if ステートメントの条件として利用するとき、下に示したようなステートメントを利用して問題点をテストして下さい。*

```
if (FunctionA (ParameterOne) < 0) then
    // 問題点を処理するステートメント
else
    // 関数が成功した場合のステートメント
```

```
endif;
```

InstallScript は次の if ステートメント構造を提供します :

- goto を含む if 構造
- if–then 構造
- if–then–else 構造
- ネストされた if–then–else 構造
- elseif 構造

goto を含む if 構造

InstallScript は goto ステートメントと共にだけ利用できる if ステートメントの特別の形式をサポートします。

```
if condition goto labelname;
```

この特殊構造は次の特徴を持ちます :

- 条件の後には必ず goto ステートメントを続けます。
- キーワード then は利用されません。
- キーワード endif は利用されません。

次の例では、szSrc がヌル文字列 ("") である限り会社名の入力をユーザーへ要求します。

```
名前 :
 AskText(" 会社名 :", "", szSrc);
 if (szSrc = "") goto Name;
```

if–then 構造

最もシンプルな if ステートメントは、式を評価して式が true の場合に指定されたアクションを実行するものです。式が true でない場合、InstallShield はステートメント全体を無視します。例 :

```
if (szStringA = " 終了 ") then
   AskYesNo (" 終了してもよろしいですか ?", NO );
endif;
```

szStringA が "exit" に等しい場合、テストは TRUE (1) 評価し、AskYesNo 関数が呼び出されます。szStringA がその他を含む場合、結果は FALSE (0) となり、ステート全体が無視されます。

下のサンプルコードは、変数 nDialog と定数 DLG_ER を比較します。これが等しい場合、InstallShield は MessageBox 関数を実行します。

```
if (nDialog = DLG_ERR) then
   MessageBox (" エラーが発生しました ", WARNING);
endif;
```

ヒント・評価する式を括弧に入れることで、if ステートメントが読み易くなりまが、InstallScript では括弧はオプションです。

if-then-else 構造

if ステートメントは、条件が失敗した場合に実行する単数また複数のステートメントを指定することもできます。このオプションは、次の例の様にキーワード else で示されます。

```
if (condition) then
    // 条件が true の場合、ステートメントが実行されます。
else
    // 条件が false の場合、ステートメントが実行されます。
endif;
```

下の例では、szStringA が "exit" に等しい場合、テストは TRUE (1) 評価し、AskYesNo 関数が呼び出されます。szStringA が "exit" と等しくない場合、結果は FALSE (0) となり、else ステートメントに続いて MessageBox 関数が呼び出されます。

```
if szStringA = " 終了 " then
    AskYesNo (" 終了してもよろしいですね ?", NO );
else
    MessageBox (" お待ちください ...", INFORMATION );
endif;
```

ネストされた if-then-else 構造

ひとつの if ステートメントが別のステートメントに組み込まれている、ネストされた if ステートメントを作成することができます。

```
if (first condition) then
    if (second condition) then
        // 最初と 2 番目の条件が true の場合に実行する
        // ステートメント
    else
        // 最初条件が True で、2 番目の条件が False の場合に実行する
        // ステートメント
    endif;
else
    if (third condition) then
        // 最初の条件が false で、3 番目の条件が false の場合に実行する
        // ステートメント
    else
        // 最初の条件が false で、3 番目の条件が false の場合に実行する
        // ステートメント
    endif;
endif;
```

次の例では、szStringA の値が " 終了 " の場合、AskYesNo が呼び出されます。szStringA の値が " 終了 " の場合、プログラムはメッセージボックスを表示します。szStringA がこれらの値のいづれとも異なる場合、ラベル UserErrorHandler へと続きます。

```
if szStringA = " 終了 " then
    AskYesNo (" 終了してもよろしいですか ?", NO );
else
    if szStringA = " 続行 " then
        MessageBox (" お待ちください ...", INFORMATION );
    else
        UserErrorHandler;
    endif;
endif;
```

elseif 構造

InstallScript では elseif ステートメントを利用して if 構造体を作成することができます。if ステートメントの 1 つの else ブランチにはまた別の if ステートメントがあります :

```
if (first condition) then
   // 最初の条件が false で、2 番目の条件が true の場合に
   // ステートメント
elseif (second condition) then
   // 最初の条件が false で、2 番目の条件が true の場合に
   // 実行されるステートメント
elseif (third condition) then
   // 最初と 2 番目の条件が false で、3 番目の条件が
   // True の場合に実行されるステートメント
   //
endif;
```

次の例では、szStringA が “exit” と等しい場合、AskYesNo が呼び出されます。szStringA が “exit” と等しくない場合、プログラムは elseif ステートメントを続行して szStringA が “continue” であるかをテストします。szStringA が “continue” と等しい場合、結果は TRUE となり、MessageBox が呼び出されます。szStringA が “continue” と等しくない場合、プログラムは次の elseif に移動するといった要領で続きます。

```
if szStringA = ″ 終了 ″ then
   AskYesNo (″ 終了してもよろしいですね ?″, NO );
elseif szStringA = ″ 続行 ″ then
   MessageBox (″ お待ちください ...″, INFORMATION );
elseif szStringA = ″ 再起動 ″ then
   goto StartHere;
endif;
```

メモ・*if ステートメントの中のラベルを定義することはできません。*

method

プロジェクト・*この情報は、InstallScript プロジェクトに適用します。*

method キーワードは、次の構文において、オブジェクトスクリプトのメソッドを宣言するために使用されます。

method < 戻り変数タイプ > < メソッド名 > (< 引数変数タイプ >);

例 :

method STRING MyMethod (STRING, NUMBER);

[新しいメソッドの追加] ダイアログ ボックスを使用してオブジェクトプロジェクトにメソッドを追加すると、メソッドの宣言がオブジェクトスクリプトに自動的に配置されます。

property()

プロジェクト・*この情報は、InstallScript プロジェクトに適用します。*

property() キーワードは、次の構文において、オブジェクトスクリプトのプロパティおよび get や put プロシージャを宣言するために使用されます。

テーブル 1 ・ Property() キーワード宣言

アクセス権	宣言
読み取り専用	property(get) < 戻り変数タイプ > < プロパティ名 > (< 引数変数タイプ >);
書き込み専用	property(put) < 戻り変数タイプ > < プロパティ名 > (< 引数変数タイプ >);
読み取り / 書き込み	property(get,put) < 戻り変数タイプ > < プロパティ名 > (< 引数変数タイプ >);

例 :

property(get,put) STRING MyProperty (NUMBER);

[新しいプロパティの追加] ダイアログ ボックスを使用して、オブジェクトプロジェクトにプロパティを追加すると、プロパティの宣言がオブジェクトスクリプトに自動的に配置されます。

prototype

prototype キーワードは、コードの行に関数定義が含まれていることを InstallScript コンパイラに通知します。このキーワードの使用方法については、「関数の宣言」を参照してください。

repeat...until

repeat ステートメントは、C 言語の do...while ループに類似しています。これは InstallScript while ステートメントにも大変よく似ています。

repeat と InstallScript の while には 2 つの大きな違いがあります :

- repeat ステートメントは少なくとも 1 回はループしなくてはなりません。while ステートメントは全くループしない場合もあります。
- while ステートメントは、式が false と評価した場合に終了します。repeat ステートメントは、式が true と評価した場合に終了します。

タスク ***repeat ループを作成するには :***

1. while ループと同じ要領で、条件テストに利用する変数を設定します。
2. 区切りを利用せずに、同じ行内に repeat と入力します。
3. 繰り返す演算をビルドします。
4. テスト変数を変更する演算を追加します (例えば、nCount = nCount +1、または nCount = SomeVariable)。
5. カッコ内に条件テストを含む until ステートメントでループを終了します。

次の例は repeat ループ構文のデモンストレーションを行います :

```
    nCount = 1;
    repeat
        MessageBox (" カウントは 5 未満です ", INFORMATION);
        nCount = nCount + 1;
    until (nCount = 5);
```

***メモ・** repeat ステートメントの中のラベルを定義することはできません。*

return

return ステートメントを利用して、ユーザー定義の関数から値を戻すことができます (関数プロトタイプが void リターンタイプを指定しない場合)。return ステートメントが検出されると、プログラム フローは関数が呼び出された位置へ戻します。呼び出しからユーザー定義関数へ戻るのに利用したとき、return ステートメント は指定された値を呼び出し側に戻すことができます。

ほとんどのビルトイン関数の戻り値は関数が成功したことを示す 0 (ゼロ)、または失敗したことを示すゼロ以下 (< 0) を示します。次に示した様に、関数ブロックの end ステートメントの上にある return ステートメントを利用して戻り値に番号を割り当てることも可能です :

```
    return -1;
end;
```

この属性では、ローカル変数そのものが破棄された場合でもローカル変数の値を呼び出し側へ戻すことが可能です。

```
function MyFunction(ParamOne, ParamTwo)
    NUMBER nNumber;
begin
    nNumber = (ParamOne + ParamTwo);
    // を参照してください。を参照してください。
    return nNumber;
end;
```

set

set キーワードは、CreateObject 関数が戻したリファレンスへの OBJECT 変数の割り当ての前に配置しなくてはなりません。例 :

```
function OnBegin()
    OBJECT oMSI;
begin
    // create the object
    set oMSI = CreateObject("WindowsInstaller.Installer");

    // オブジェクト ( ユーザーのシステム上の MSI バージョンを表示する ) を利用します
    MessageBox(" ご利用の MSI バージョンは : " + oMSI.Version, INFORMATION);

    // オブジェクトを開放します
    set oMSI = NOTHING;
end;
```

メモ・COM オブジェクトの例外処理をさらに制御するために、try-catch-endcatch キーワードを使用することができます。

stdcall

プロジェクト・この情報は、InstallScript プロジェクトに適用されます。stdcall キーワードは stdcall 呼び出し規則を使う外部 DLL 関数を宣言するときに利用します。例:

```
prototype stdcall POINTER kernel32.lstrcpy( byref string, byref string);
```

呼び出し規則が指定されない場合、stdcall と見なされます。

ほとんどの Windows API 関数は stdcall (WINAPI) 呼び出し規則を利用します。呼び出し規則についてのより詳しい情報は、Microsoft マニュアルをご覧下さい。

switch...endswitch

switch ステートメントは elseif 構造 ステートメントと類似しています。switch ステートメントは、式の値に従ってコードの異なるセクションの 1 つを実行するのに利用します。switch ステートメントは式を評価し、そして定数値が式の結果と一致する case ステートメントにブランチします。case ステートメントに一致するものがなかった時、デフォルトステートメントが指定してある場合はコントロールはデフォルトへ渡します。

Switch ステートメントの作成

タスク ***switch ステートメントを作成するには:***

1. キーワード *switch* と、続けて評価する式をタイプします。式には定数、変数、算術式、論理式、または関数結果が利用でき、それらはかっこで囲む必要があります。この行を区切らないで下さい。
2. 各オプションには、キーワード *case* および 1 つまたは複数の定数を入力して、後にコロンをつけます。複数の定数が指定された場合、コンマで区切ります。ここでは定数のみを指定することができます。変数名、文字列 ID、関数結果、またはその他の種類の式をキーワード case の後に指定するとエラーが発生します。
3. 各 case の後にはコロンと、そのオプションで実行されるひとつまたは複数のステートメントを続けます。各ステートメントをセミコロンで終了します。
4. すべてのステートメントが指定された後、キーワード *default* とコロン (:) を続けて、式がどのケースにも一致しない場合にプログラムをコントロールします。
5. ブロックを キーワード *endswitch* とセミコロン (;) で閉じます。

スクリプト例

次の部分スクリプトは、実行されるコンピューターの現在のビデオ解像度を表示します。

```
STRING szMsg, svResult;
NUMBER nvResult;
```

```
GetSystemInfo (VIDEO, nvResult, svResult);

switch (nvResult)
   case IS_UNKNOWN:
      szMsg = " ユーザーのビデオは不明です。";
   case IS_EGA:
      szMsg = "EGA 解像度。";
   case IS_VGA:
      szMsg = "VGA 解像度。";
   case IS_SVGA:
      szMsg = "Super VGA (800 x 600) 解像度。";
   case IS_XVGA:
      szMsg = "XVGA (1024 x 768) 解像度。";
   case IS_UVGA:
      szMsg = "1024 x 768 以上の解像度。";
   デフォルト :
      szMsg = " エラー ";
endswitch;

MessageBox (szMsg, INFORMATION);
```

メモ・*switch ステートメントが実行される度に、ひとつの case ブロックのみが実行されます。InstallShield が case ブロックを実行した後、endswitch の後に次のステートメントを実行します。switch ブロックは while ループ内で利用すると非常に便利です。case ステートメントをフラグとして利用することで、オプション終了位置を作成することができます。*

try、catch　および endcatch

try、catch および endcatch は例外処理に使用するキーワードです。例外処理についての詳細は、「例外処理」を参照してください。

メモ・*try...catch...endcatch ステートメント内で goto ステートメントを使用することはできません。また、try...catch...endcatch ステートメント内でラベルを定義することはできません。*

void

void 編集を void と言うタイプで宣言できない点で、本当のデータ タイプとは言えません。void は関数プロトタイプでのみ利用され、次に示す例の様に関数が値を戻さないことを示します。

```
prototype void Subroutine(int);

function void Subroutine(int);
begin
   // 処理を行う、しかし
   // 値を戻さない
end;
```

while...endwhile

単一または複数のステートメントを特定の条件が true である限り繰り返し実行する場合は while ステートメントを利用します。ステートメントが最初に実行されたときに条件が true でない場合、ループは実行されません。

タスク ***while ループを作成するには：***

1. 初期状態への条件として利用する変数を設定します。
2. キーワード while を入力し、後に括弧で括った条件テストを続けます。この行を区切らないで下さい。
3. 繰り返す演算をビルドします。
4. テスト変数を変更する演算を追加します (例えば、nCount = nCount +1、または nCount = SomeVariable)。
5. endwhile とセミコロンを続けてループを終了します。

次の例では、メッセージボックスが 4 回表示されます。

```
nCount = 1;

while (nCount < 5)
   MessageBox (" これはまだ True です。", INFORMATION);
   nCount = nCount + 1;
endwhile;
```

nCount には 1 の初期値が割り当てられているため、while ステートメントが TRUE の初回実行を評価すると、メッセージ ボックスが表示されて nCount は 1 増分されます。4 番目がループを通過すると、nCount は 5 に等しくなります。while ステートメントで FALSE が評価されると、プログラムは endwhile の後のステートメントと一緒に続行されます。

メモ・*hwile ブロック内のラベルを定義することはできません。しかし、InstallScript で while ステートメントをネストすることができます。各 while ブロックは endwhile で終了しなくてはなりません。*

Nested while の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
 /* このスクリプトはネストされた while ループを説明します。
* 特定タイプのファイルを検索し、
* 各ファイルの行数を表示します。*/

#define SOURCEDIR "c:¥¥example";

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_Nested while(HWND);
```

```
function ExFn_Nested while(hMSI)
    LIST    listID;
    STRING  svTarget, svResult, filename, svLine, szPath, szFileName;
    NUMBER  nResult, nOp,nFileHandle,count;
begin

    count = 0;
    nOp = RESET;
    svTarget = SOURCEDIR;
    listID = ListCreate (STRINGLIST);

    while FindAllFiles (svTarget, "*.txt", svResult, nOp) = 0;

        // ファイル名を完全指定パスで取得します
        StrGetTokens(listID,svResult,"¥¥");
        ListCurrentString(listID,filename);

        // ファイル モードを通常に設定します。
        OpenFileMode(FILE_MODE_NORMAL);
        szFileName = filename;
        szPath     = svTarget;

        // 次のスクリプトは編集のためファイルを開きます。
        OpenFile(nFileHandle, szPath, szFileName);

/*----------------------------------------------------------------------*¥
 *
 * 次のスクリプトは開いたファイルからテキストの各行を呼び出し、
 * 行数を検出するカウントを増やします。
 *
¥*----------------------------------------------------------------------*/

        while (GetLine (nFileHandle, svLine) = 0)
            count = count + 1;
        endwhile;

        SprintfBox(INFORMATION," ファイルの行総数は ",
                " ファイル %s の行数は %d です。 ",filename,count);

        count = 0;

        // 次のスクリプトはファイルを閉じます
        CloseFile(nFileHandle);

        // 最後のファイルの場所でファイルの検索を続けます。
        nOp = CONTINUE;

        if (FindAllFiles (svTarget, "*.txt", svResult, nOp) < 0) then
            abort;
        endif;

    endwhile;

end;
```

フロー制御

ほとんどのプログラム言語と同様に、InstallScript はステートメントを最初のステートメントから始まって最後のステートメントで終わる関数ブロック内で順に処理します。関数ブロック内の線的な実行の流れは、ブランチや反復を行う条件ステートメントで制御することができます。

一般的にブランチは 1 つのパスまたは別のパスへと実行を導く if ステートメントで利用されます。反復は、単数または複数のステートメントを設定した回数、または指定した条件に一致する限り繰り返し実行するループステートメントで実行されます。

スクリプト内で実行フローを制御するため、InstallShield は次のキーワードを提供します :

- abort
- exit
- for...endfor
- goto
- if...then...else...endif
- repeat...until
- return
- switch...endswitch
- while...endwhile

定義済み定数

定義済みの定数とは、InstallScript によって予約されている識別子で、特定のリテラル値を表します。InstallScript では定義済みの定数を使用して、ビルトイン関数に渡されてビルトイン関数によって返される一定のデータ値を表します。リテラル値でなくこれらの定義済みの定数を使用すると、セットアップスクリプトが読みやすくなります。

InstallShield によって定義済みの定数に割り当てられている値は変更できません。ただし、定義済みの定数の値は、SprintfBox を呼び出すことにより表示することができます。次の例では、定義済みの定数 FEATURE_FIELD_SELECTED の値を表示します：

```
SprintfBox (INFORMATION, "", "%d", FEATURE_FIELD_SELECTED);
```

定義済みの定数の代わりにリテラル値を使用することもできますが、関数を示す場合は常に定義済みの定数を使用することが強く推奨されます。

InstallScript で使用される定義済みの定数を次のリストに示します。

-
-
-
-

AFTER

AFTER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- EzBatchAddString
- EzBatchAddPath
- ConfigAdd
- ConfigMove
- ListAddItem
- ListAddString
- PathAdd
- PathMove
- BatchAdd
- PathMove
- BatchMoveEx
- FileInsertLine
- EzConfigAddDriver

- EzConfigAddString

ALLCONTENTS

ALLCONTENTS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DeleteDir

ALLCONTROLS

ALLCONTROLS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CtrlSetFont

APPEND

APPEND は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FileInsertLine

ASKDESTPATH

ASKDESTPATH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- AskDestPath

ASKOPTIONS

ASKOPTIONS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- AskOptions
- PlaceWindow

ASKPATH

ASKPATH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- AskPath
- PlaceWindow

ASKTEXT

ASKTEXT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- AskText
- PlaceWindow

BACK

BACK は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- AskDestPath
- AskOptions
- AskPath
- AskText
- FeatureDialog
- SdAskDestPath
- SdAskOptions
- SdAskOptionsList
- SdBitmap
- SdDisplayTopics
- SdFeatureDialog
- SdFeatureDialog2
- SdFeatureDialogAdv
- SdFeatureMult

- SdLicense
- SdOptionsButtons
- SdRegisterUser
- SdRegisterUserEx
- SdSelectFolder
- SdShowAnyDialog
- SdShowDlgEdit1
- SdShowDlgEdit2
- SdShowDlgEdit3
- SdShowFileMods
- SdShowInfoList
- SdStartCopy
- SdWelcome
- SelectFolder
- Welcome

BACKBUTTON

BACKBUTTON は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Disable
- Enable
- Is

BACKGROUND

BACKGROUND は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceWindow
- SetColor
- Enable
- SizeWindow

- Disable

BACKGROUNDCAPTION

BACKGROUNDCAPTION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetTitle

BASEMEMORY

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

BASEMEMORY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

BEFORE

BEFORE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PathMove
- FileInsertLine
- EzBatchAddPath
- EzBatchAddString
- BatchAdd
- BatchMoveEx
- EzConfigAddDriver
- EzConfigAddString
- ConfigAdd
- ConfigMove
- ListAddItem
- ListAddString

- PathAdd

BIF_BROWSEFORCOMPUTER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

BIF_BROWSEFORCOMPUTER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SelectDirEx

BIF_BROWSEFORPRINTER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

BIF_BROWSEFORPRINTER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SelectDirEx

BIF_DONTGOBELOWDOMAIN

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

BIF_DONTGOBELOWDOMAIN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SelectDirEx

BIF_EDITBOX

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

BIF_EDITBOX は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SelectDirEx

BIF_RETURNFSANCESTORS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

BIF_RETURNFSANCESTORS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SelectDirEx

BIF_RETURNONLYFSDIRS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

BIF_RETURNONLYFSDIRS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SelectDirEx

BIF_STATUSTEXT

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

BIF_STATUSTEXT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SelectDirEx

BILLBOARD

BILLBOARD は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Disable
- PlaceWindow

BITMAPICON

BITMAPICON は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceBitmap

BK_BLUE

BK_BLUE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_GREEN

BK_GREEN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_MAGENTA

BK_MAGENTA は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_ORANGE

BK_ORANGE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_PINK

BK_PINK は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_RED

BK_RED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_SMOOTH

BK_SMOOTH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_SOLIDBLACK

BK_SOLIDBLACK は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_SOLIDBLUE

BK_SOLIDBLUE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_SOLIDGREEN

BK_SOLIDGREEN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_SOLIDMAGENTA

BK_SOLIDMAGENTA は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_SOLIDORANGE

BK_SOLIDORANGE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_SOLIDPINK

BK_SOLIDPINK は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_SOLIDRED

BK_SOLIDRED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_SOLIDWHITE

BK_SOLIDWHITE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_SOLIDYELLOW

BK_SOLIDYELLOW は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BK_YELLOW

BK_YELLOW は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

BLACK

BLACK は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetTitle

BLUE

BLUE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor
- SetTitle

BOOTUPDRIVE

BOOTUPDRIVE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

BUTTON_CHECKED

BUTTON_CHECKED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CtrlSetState
- CtrlGetState

BUTTON_UNCHECKED

BUTTON_UNCHECKED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CtrlSetState
- CtrlGetState

BYTES

BYTES は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ConvertSizeToUnits
- StrConvertSizeUnit

CANCEL

CANCEL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SelectDir

CANCELBUTTON

CANCELBUTTON は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Disable
- Enable
- Is

CDROM

CDROM は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

CDROM_DRIVE

CDROM_DRIVE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetValidDrivesList

CENTERED

CENTERED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceWindow

- PlaceBitmap

CHECKBOX

CHECKBOX は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DialogSetInfo

CHECKBOX95

CHECKBOX95 は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DialogSetInfo

CHECKLINE

CHECKLINE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DialogSetInfo

CHECKMARK

CHECKMARK は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DialogSetInfo

CLEAR_FILE_ATTR

CLEAR_FILE_ATTR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- XCopyFile

COLORS

COLORS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

COMMAND

COMMAND は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- BatchMoveEx
- ConfigFind
- EzBatchAddString
- BatchAdd
- BatchDeleteEx

COMMON

COMMON は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ProgDefGroupType

COMPACT

COMPACT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetupType
- SdSetupType

COMPARE_DATE

COMPARE_DATE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FileCompare

COMPARE_MD5_SIGNATURE

COMPARE_MD5_SIGNATURE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FileCompare

COMPARE_SIZE

COMPARE_SIZE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FileCompare

COMPARE_VERSION

COMPARE_VERSION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FileCompare

COMP_NORMAL

COMP_NORMAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- XCopyFile

COMP_UPDATE_DATE

COMP_UPDATE_DATE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- XCopyFile

COMP_UPDATE_SAME

COMP_UPDATE_SAME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- XCopyFile

COMP_UPDATE_VERSION

COMP_UPDATE_VERSION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- XCopyFile

CONTINUE

CONTINUE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FileGrep
- BatchFind
- FindFile
- ConfigFind
- PathFind

COPY_ERR_CREATEDIR

COPY_ERR_CREATEDIR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CopyFile
- XCopyFile

COPY_ERR_MEMORY

COPY_ERR_MEMORY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CopyFile
- XCopyFile

COPY_ERR_NODISKSPACE

COPY_ERR_NODISKSPACE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CopyFile
- XCopyFile

COPY_ERR_OPENINPUT

COPY_ERR_OPENINPUT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CopyFile
- XCopyFile

COPY_ERR_OPENOUTPUT

COPY_ERR_OPENOUTPUT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CopyFile
- XCopyFile

COPY_ERR_TARGETREADONLY

COPY_ERR_TARGETREADONLY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CopyFile
- XCopyFile

CPU

CPU は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

CS_OPTION_FLAG_NO_NEW_INSTALL_HIGHLIGHT

CS_OPTION_FLAG_NO_NEW_INSTALL_HIGHLIGHT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CreateShortcut
- ReplaceShortcut

CS_OPTION_FLAG_NO_STARTSCREEN_PIN

CS_OPTION_FLAG_NO_STARTSCREEN_PIN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CreateShortcut
- ReplaceShortcut

CS_OPTION_FLAG_PREVENT_PINNING

CS_OPTION_FLAG_PREVENT_PINNING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CreateShortcut
- ReplaceShortcut

CS_OPTION_FLAG_REPLACE_EXISTING

CS_OPTION_FLAG_REPLACE_EXISTING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CreateShortcut
- ReplaceShortcut

CS_OPTION_FLAG_RUN_MAXIMIZED

CS_OPTION_FLAG_RUN_MAXIMIZED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CreateShortcut
- ReplaceShortcut

CS_OPTION_FLAG_RUN_MINIMIZED

CS_OPTION_FLAG_RUN_MINIMIZED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CreateShortcut
- ReplaceShortcut

CURRENTROOTKEY

CURRENTROOTKEY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VarRestore
- VarSave

CUSTOM

CUSTOM は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetupType
- SdSetupType

DATA_COMPONENT

DATA_COMPONENT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SilentReadData
- SilentWriteData

DATA_LIST

DATA_LIST は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SilentWriteData

DATA_NUMBER

DATA_NUMBER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SilentReadData
- SilentWriteData

DATA_STRING

DATA_STRING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SilentReadData
- SilentWriteData

DATE

DATE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

DEFAULT

DEFAULT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- switch...endswitch

DEFWINDOWMODE

DEFWINDOWMODE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Enable

DELETE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

DEFAULT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions
- SERVICE_IS_PARAMS

DELETE_EOF

DELETE_EOF は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FileDeleteLine

DIALOGCACHE

DIALOGCACHE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Disable
- Enable

DIFXAPI_ERROR

DIFXAPI_ERROR は定義済みの定数で、1 つまたは複数のイベント ハンドラーと使用するために提供されている値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- OnDIFxLogCallback

DIFXAPI_INFO

DIFXAPI_INFO は定義済みの定数で、1 つまたは複数のイベント ハンドラーと使用するために提供されている値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- OnDIFxLogCallback

DIFXAPI_SUCCESS

DIFXAPI_SUCCESS は定義済みの定数で、1 つまたは複数のイベント ハンドラーと使用するために提供されている値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- OnDIFxLogCallback

DIFXAPI_WARNING

DIFXAPI_WARNING は定義済みの定数で、1 つまたは複数のイベント ハンドラーと使用するために提供されている値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- OnDIFxLogCallback

DIRECTORY

DIRECTORY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ParsePath

DIR_WRITEABLE

DIR_WRITEABLE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

DISABLE_ALLUSERBTN

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

DISABLE_ALLUSERBTN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

DISABLE_ALLUSERBTN 定数は、[すべてのユーザー] オプションを、通常は有効であるところを無効 (または非表示) にすることを示します。この変数のデフォルト値は、FALSE です。インストールが管理者権限またはパワーユーザー権限なしで実行されている場合、この変数の値に関わらず、[すべてのユーザー] オプションは常に非表示となります。

次の関数と共に利用します

- SdCustomerInformation
- SdCustomerInformationEx

DISABLE_PERUSERBTN

DISABLE_PERUSERBTN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

DISABLE_PERUSERBTN 定数は、[ユーザーごと] オプションを、通常は有効であるところを無効 (または HIDE_DISABLED_BTNS が TRUE の場合は非表示) にすることを示します。この変数のデフォルト値は、FALSE です。Windows 9x プラットフォーム上では、この変数の値に関わらず、[ユーザーごと] オプションは常に非表示となります。

次の関数と共に利用します

- SdCustomerInformation
- SdCustomerInformationEx

DISK

DISK は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ParsePath

DISK1FEATURE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

DISK1FEATURE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

DISK1FEATURE は、メンテナンスセットアップやアンインストレーションに必要なファイルを含むコンポーネントを指定します。(この機能はメディア ビルダーによって .cab ファイルに自動的に配置され、IDE には表示されません。)

次の関数と共に利用します

- FeatureSelectItem
- FeatureIsItemSelected

DISK_INFO_QUERY_ALL

DISK_INFO_QUERY_ALL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetDiskInfo

DISK_INFO_QUERY_BYTES_PER_CLUSTER

DISK_INFO_QUERY_BYTES_PER_CLUSTER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetDiskInfo

DISK_INFO_QUERY_DISK_FREE_SPACE

DISK_INFO_QUERY_DISK_FREE_SPACE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetDiskInfo

DISK_INFO_QUERY_DISK_TOTAL_SPACE

DISK_INFO_QUERY_DISK_TOTAL_SPACE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetDiskInfo

DISK_INFO_QUERY_DRIVE_TYPE

DISK_INFO_QUERY_DRIVE_TYPE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetDiskInfo

DISK_TOTALSPACE

DISK_TOTALSPACE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

DISK_TOTALSPACE_EX

DISK_TOTALSPACE_EX は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

DLG_ASK_OPTIONS

DLG_ASK_OPTIONS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDialogTitle

DLG_ASK_PATH

DLG_ASK_PATH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDialogTitle

DLG_ASK_TEXT

DLG_ASK_TEXT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDialogTitle

DLG_ASK_YESNO

DLG_ASK_YESNO は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDialogTitle

DLG_CENTERED

DLG_CENTERED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DefineDialog

DLG_CLOSE

DLG_CLOSE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- WaitOnDialog

DLG_DIR_DIRECTORY

DLG_DIR_DIRECTORY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CtrlDir

DLG_DIR_DRIVE

DLG_DIR_DRIVE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CtrlDir

DLG_DIR_FILE

DLG_DIR_FILE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CtrlDir

DLG_ENTER_DISK

DLG_ENTER_DISK は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDialogTitle

DLG_ERR

DLG_ERR は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- WaitOnDialog
- EzDefineDialog
- ReleaseDialog
- CtrlGetState
- DefineDialog

DLG_ERR_ALREADY_EXISTS

DLG_ERR_ALREADY_EXISTS は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DefineDialog
- EzDefineDialog

DLG_ERR_ENDDLG

DLG_ERR_ENDDLG は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ReleaseDialog

DLG_INFO_ALTIMAGE

DLG_INFO_ALTIMAGE は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DialogSetInfo

DLG_INFO_ALTIMAGE_HIDPI

DLG_INFO_ALTIMAGE_HIDPI は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DialogSetInfo

DLG_INFO_ALTIMAGE_REVERT_IMAGE

DLG_INFO_ALTIMAGE_REVERT_IMAGE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DialogSetInfo

DLG_INFO_ALTIMAGE_VERIFY_BMP

DLG_INFO_ALTIMAGE_VERIFY_BMP は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DialogSetInfo

DLG_INFO_CHECKSELECTION

この定数は現在使用されていません。InstallScript のチェックボックス コントロールは、現在の Windows テーマを使って自動的に作成され、高 DPI ディスプレイと互換性を持つ最新で統一性のある外観を持ちます。

DLG_INFO_KUNITS

DLG_INFO_KUNITS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DialogSetInfo

DLG_INFO_USEDECIMAL

DLG_INFO_USEDECIMAL は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DialogSetInfo

DLG_INIT

DLG_INIT は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- WaitOnDialog

DLG_MSG_ALL

DLG_MSG_ALL は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DefineDialog

DLG_MSG_INFORMATION

DLG_MSG_INFORMATION は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDialogTitle

DLG_MSG_SEVERE

DLG_MSG_SEVERE は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDialogTitle

DLG_MSG_STANDARD

DLG_MSG_STANDARD は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DefineDialog

DLG_MSG_WARNING

DLG_MSG_WARNING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDialogTitle

DLG_STATUS

DLG_STATUS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDialogTitle

DLG_USER_CAPTION

DLG_USER_CAPTION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDialogTitle

DOINSTALL_OPTION_NOHIDEPROGRESS

DOINSTALL_OPTION_NOHIDEPROGRESS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DoInstall

DOINSTALL_OPTION_NOHIDESPLASH

DOINSTALL_OPTION_NOHIDESPLASH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DoInstall

DOINSTALL_OPTION_NOLANGSWITCH

DOINSTALL_OPTION_NOLANGSWITCH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DoInstall

DOINSTALL_OPTION_NOSETBATCHINSTALL

DOINSTALL_OPTION_NOSETBATCHINSTALL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DoInstall

DOTNETFRAMEWORKINSTALLED

DOTNETFRAMEWORKINSTALLED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

DOTNETSERVICEPACKINSTALLED

DOTNETSERVICEPACKINSTALLED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

DRIVE

DRIVE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

DRIVE_CDROM

DRIVE_CDROM は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetDiskInfo

DRIVE_FIXED

DRIVE_FIXED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetDiskInfo

DRIVE_NO_ROOT_DIR

DRIVE_NO_ROOT_DIR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetDiskInfo

DRIVE_RAMDISK

DRIVE_RAMDISK は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetDiskInfo

DRIVE_REMOTE

DRIVE_REMOTE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetDiskInfo

DRIVE_REMOVABLE

DRIVE_REMOVABLE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetDiskInfo

DRIVE_UNKNOWN

DRIVE_UNKNOWN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetDiskInfo

DRIVER_PACKAGE_DELETE_FILES

DRIVER_PACKAGE_DELETE_FILES は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DIFxDriverPackageUninstall

DRIVER_PACKAGE_FORCE

DRIVER_PACKAGE_FORCE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DIFxDriverPackageInstall
- DIFxDriverPackagePreinstall
- DIFxDriverPackageUninstall

DRIVER_PACKAGE_LEGACY_MODE

DRIVER_PACKAGE_LEGACY_MODE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DIFxDriverPackageInstall
- DIFxDriverPackagePreinstall

DRIVER_PACKAGE_ONLY_IF_DEVICE_PRESENT

DRIVER_PACKAGE_ONLY_IF_DEVICE_PRESENT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DIFxDriverPackageInstall
- DIFxDriverPackagePreinstall

DRIVER_PACKAGE_REPAIR

DRIVER_PACKAGE_REPAIR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DIFxDriverPackageInstall
- DIFxDriverPackagePreinstall
- DIFxDriverPackageUninstall

DRIVER_PACKAGE_SILENT

DRIVER_PACKAGE_SILENT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DIFxDriverPackageInstall
- DIFxDriverPackagePreinstall

EDITBOX_CHANGE

EDITBOX_CHANGE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CtrlGetSubCommand

EFF_BOXSTRIPE

EFF_BOXSTRIPE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDisplayEffect

EFF_FADE

EFF_FADE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDisplayEffect

EFF_HORZREVEAL

EFF_HORZREVEAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDisplayEffect

EFF_HORZSTRIPE

EFF_HORZSTRIPE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDisplayEffect

EFF_NONE

EFF_NONE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDisplayEffect

EFF_REVEAL

EFF_REVEAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDisplayEffect

EFF_VERTSTRIPE

EFF_VERTSTRIPE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetDisplayEffect

END_OF_FILE

END_OF_FILE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FileGrep

END_OF_LIST

END_OF_LIST は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ListCurrentItem
- ListCurrentString
- ListGetFirstItem
- ListSetIndex
- ListDeleteItem
- ListDeleteString
- ListFindItem

- ListFindString
- ListCurrentString
- ListGetNextItem
- ListGetNextString
- ListSetCurrentItem
- ListSetCurrentString

ENTERDISK

ENTERDISK は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- EnterDisk
- PlaceWindow

EQUALS

EQUALS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerCompare
- FileCompare

ERROR_ACCESS_DENIED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_ACCESS_DENIED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_CIRCULAR_DEPENDENCY

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_CIRCULAR_DEPENDENCY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_DATABASE_DOES_NOT_EXIST

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_DATABASE_DOES_NOT_EXIST は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_DEPENDENT_SERVICES_RUNNING

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_DEPENDENT_SERVICES_RUNNING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_DUP_NAME

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_DUP_NAME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_FILE_NOT_FOUND

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_FILE_NOT_FOUND は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_INVALID_HANDLE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_INVALID_HANDLE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_INVALID_PARAMETER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_INVALID_PARAMETER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_INVALID_SERVICE_ACCOUNT

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_INVALID_SERVICE_ACCOUNT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_INVALID_SERVICE_CONTROL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_INVALID_SERVICE_CONTROL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_PATH_NOT_FOUND

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_PATH_NOT_FOUND は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_SERVICE_ALREADY_RUNNING

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_SERVICE_ALREADY_RUNNING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_SERVICE_CANNOT_ACCEPT_CTRL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_SERVICE_CANNOT_ACCEPT_CTRL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_SERVICE_DATABASE_LOCKED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_SERVICE_DATABASE_LOCKED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_SERVICE_DEPENDENCY_DELETED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_SERVICE_DEPENDENCY_DELETED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_SERVICE_DEPENDENCY_FAIL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_SERVICE_DEPENDENCY_FAIL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_SERVICE_DISABLED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_SERVICE_DISABLED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_SERVICE_DOES_NOT_EXIST

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_SERVICE_DOES_NOT_EXIST は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_SERVICE_EXISTS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_SERVICE_EXISTS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_SERVICE_LOGON_FAILED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_SERVICE_LOGON_FAILED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_SERVICE_NOT_ACTIVE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_SERVICE_NOT_ACTIVE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_SERVICE_NO_THREAD

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_SERVICE_NO_THREAD は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_SERVICE_REQUEST_TIMEOUT

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_SERVICE_REQUEST_TIMEOUT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERROR_TIMEOUT

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ERROR_TIMEOUT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetExtendedErrInfo

ERR_ABORT

ERR_ABORT は、定義済みの定数で、1 つまたは複数のイベント ハンドラーによって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- OnNextDisk

ERR_BOX_BADPATH

ERR_BOX_BADPATH は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetErrorMsg
- SetErrorTitle

ERR_BOX_BADTAGFILE

ERR_BOX_BADTAGFILE は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetErrorMsg
- SetErrorTitle

ERR_BOX_DISKID

ERR_BOX_DISKID は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetErrorTitle
- SetErrorMsg

ERR_BOX_DRIVEOPEN

ERR_BOX_DRIVEOPEN は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetErrorTitle
- SetErrorMsg

ERR_IGNORE

ERR_IGNORE は定義済みの定数で、1 つまたは複数のビルト イン関数またはイベント ハンドラーによって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SdExceptions

ERR_NO

ERR_NO は定義済みの定数で、1 つまたは複数のビルトイン関数またはイベント ハンドラーによって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SdExceptions

ERR_PERFORM_AFTER_REBOOT

ERR_PERFORM_AFTER_REBOOT は定義済みの定数で、1 つまたは複数のビルトイン関数またはイベント ハンドラーによって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SdExceptions

ERR_RETRY

ERR_RETRY は定義済みの定数で、1 つまたは複数のビルトイン関数またはイベント ハンドラーによって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- OnNextDisk
- SdExceptions

ERR_YES

ERR_YES は定義済みの定数で、1 つまたは複数のビルトイン関数またはイベント ハンドラーによって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SdExceptions

EXCLUDE_SUBDIR

EXCLUDE_SUBDIR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- XCopyFile
- FindAllDirs

EXCLUSIVE

EXCLUSIVE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SdAskOptionsList
- AskOptions
- SdAskOptions

EXISTS

EXISTS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ExistsDir
- ExistsDisk

EXIT

EXIT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Do
- HandlerEx

EXTENDEDMEMORY

EXTENDEDMEMORY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

EXTENSION_ONLY

EXTENSION_ONLY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ParsePath

FALSE

FALSE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- AskOptions
- CtrlSetMultCurSel
- DialogSetInfo
- FeatureAddItem
- FeatureGetData
- FeatureIsItemSelected
- FeatureSelectItem
- FeatureTotalSize
- LongPathToQuote
- SdDiskSpace2
- SelectDir
- SdShowMsg
- SQLDatabaseBrowse
- SQLRTConnect
- SQLRTConnect2
- SQLRTConnectDB
- SQLRTGetDatabases
- SQLRTGetServers
- SQLRTGetServers2
- SQLRTPutConnectionAuthentication
- SQLRTTestConnection
- SQLRTTestConnection2

- SQLServerSelectLogin
- SQLServerSelectLogin2

FEATURE_FIELD_CDROM_FOLDER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_FIELD_CDROM_FOLDER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData

FEATURE_FIELD_DESCRIPTION

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

FEATURE_FIELD_DESCRIPTION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData
- FeatureSetData

FEATURE_FIELD_DISPLAYNAME

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

FEATURE_FIELD_DISPLAYNAME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData
- FeatureSetData

FEATURE_FIELD_ENCRYPT

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_FIELD_ENCRYPT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData
- FeatureSetData

FEATURE_FIELD_FILENEED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_FIELD_FILENEED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData

FEATURE_FIELD_FLAGS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_FIELD_FLAGS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData

FEATURE_FIELD_FTPLOCATION

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

FEATURE_FIELD_FTPLOCATION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData

FEATURE_FIELD_GUID

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_FIELD_GUID は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData

FEATURE_FIELD_HANDLER_ONINSTALLED

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_FIELD_HANDLER_ONINSTALLED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData

FEATURE_FIELD_HANDLER_ONINSTALLING

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_FIELD_HANDLER_ONINSTALLING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData

FEATURE_FIELD_HANDLER_ONUNINSTALLED

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_FIELD_HANDLER_ONUNINSTALLED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData

FEATURE_FIELD_HANDLER_ONUNINSTALLING

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_FIELD_HANDLER_ONUNINSTALLING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData

FEATURE_FIELD_HTTPLOCATION

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_FIELD_HTTPLOCATION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData

FEATURE_FIELD_IMAGE

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_FIELD_IMAGE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureSetData

FEATURE_FIELD_MISC

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_FIELD_MISC は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData
- FeatureSetData

FEATURE_FIELD_PASSWORD

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_FIELD_PASSWORD は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData

FEATURE_FIELD_SELECTED

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_FIELD_SELECTED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData
- FeatureSetData

FEATURE_FIELD_SIZE

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_FIELD_SIZE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData
- FeatureSetData

FEATURE_FIELD_STATUS

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_FIELD_STATUS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData
- FeatureSetData

FEATURE_FIELD_VISIBLE

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_FIELD_VISIBLE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData
- FeatureSetData

FEATURE_INFO_ATTRIBUTE

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

FEATURE_INFO_ATTRIBUTE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_COMPONENT_FLAGS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_INFO_COMPONENT_FLAGS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_COMPSIZE_HIGH

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_INFO_COMPSIZE_HIGH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_COMPSIZE_LOW

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_INFO_COMPSIZE_LOW は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_DATE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_INFO_DATE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_DATE_EX

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_INFO_DATE_EX は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_DESTINATION

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_INFO_DESTINATION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_FTPLOCATION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_INFO_FTPLOCATION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_HTTPLOCATION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_INFO_HTTPLOCATION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_LANGUAGE

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_INFO_LANGUAGE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_MD5_SIGNATURE

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_INFO_MD5_SIGNATURE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_MISC

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_INFO_MISC は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_ORIGSIZE_HIGH

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_INFO_ORIGSIZE_HIGH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_ORIGSIZE_LOW

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_INFO_ORIGSIZE_LOW は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_OS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_INFO_OS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_OVERWRITE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_INFO_OVERWRITE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_PLATFORM_SUITE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_INFO_PLATFORM_SUITE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_TIME

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_INFO_TIME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_VERSIONLS

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_INFO_VERSIONLS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_VERSIONMS

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_INFO_VERSIONMS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_INFO_VERSIONSTR

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FEATURE_INFO_VERSIONSTR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileInfo

FEATURE_OPCOST_UNINSTALL_FILE

FEATURE_OPCOST_UNINSTALL_FILE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureAddUninstallCost
- FeatureSpendUninstallCost

FEATURE_OPCOST_UNINSTALL_REGORINI

FEATURE_OPCOST_UNINSTALL_REGORINI は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureAddUninstallCost
- FeatureSpendUninstallCost

FEATURE_OPCOST_UNINSTALL_UNREGFILE

FEATURE_OPCOST_UNINSTALL_UNREGFILE は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureAddUninstallCost
- FeatureSpendUninstallCost

FEATURE_VALUE_CRITICAL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_VALUE_CRITICAL は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData

FEATURE_VALUE_HIGHLYRECOMMENDED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_VALUE_HIGHLYRECOMMENDED は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData

FEATURE_VALUE_STANDARD

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FEATURE_VALUE_STANDARD は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureGetData

ファイル属性

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

テーブル 1・ファイルの属性

属性	説明
FILE_ATTR_NORMAL	ファイルは標準ファイルです。
FILE_ATTR_ARCHIVED	ファイルはアーカイブされています。
FILE_ATTR_DIRECTORY	ファイルはディレクトリです。
FILE_ATTR_HIDDEN	ファイルは隠しファイルです。
FILE_ATTR_READONLY	ファイルは読み取り専用です。
FILE_ATTR_SYSTEM	ファイルシステムファイルです。

FILE_ADD_FILE

FILE_ADD_FILE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

FILE_ADD_SUBDIRECTORY

FILE_ADD_SUBDIRECTORY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

FILE_ALL_ACCESS

FILE_ALL_ACCESS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

FILE_APPEND_DATA

FILE_APPEND_DATA は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

FILE_ATTR_ARCHIVED

FILE_ATTR_ARCHIVED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetFileInfo

FILE_ATTR_HIDDEN

FILE_ATTR_HIDDEN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetFileInfo

FILE_ATTR_NORMAL

FILE_ATTR_NORMAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetFileInfo

FILE_ATTR_READONLY

FILE_ATTR_READONLY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetFileInfo

FILE_ATTR_SYSTEM

FILE_ATTR_SYSTEM は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetFileInfo

FILE_ATTRIBUTE

FILE_ATTRIBUTE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetFileInfo
- SetFileInfo

FILE_BIN_CUR

FILE_BIN_CUR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SeekBytes

FILE_BIN_END

FILE_BIN_END は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SeekBytes

FILE_BIN_START

FILE_BIN_START は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SeekBytes

FILE_DATE

FILE_DATE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetFileInfo
- SetFileInfo

FILE_DELETE_CHILD

FILE_DELETE_CHILD は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

FILE_EXECUTE

FILE_EXECUTE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

FILE_EXISTS

FILE_EXISTS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

FILE_INSTALLED

FILE_INSTALLED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerSearchAndUpdateFile

FILE_IS_LOCKED

FILE_IS_LOCKED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerUpdateFile
- VerSearchAndUpdateFile

FILE_LINE_LENGTH

FILE_LINE_LENGTH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FileInsertLine
- FileGrep

FILE_LIST_DIRECTORY

FILE_LIST_DIRECTORY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

FILE_LOCKED

FILE_LOCKED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

FILE_MD5_SIGNATURE

FILE_MD5_SIGNATURE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetFileInfo

FILE_MODE_APPEND

FILE_MODE_APPEND は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- OpenFileMode
- CreateFile

FILE_MODE_APPEND_UNICODE

FILE_MODE_APPEND_UNICODE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- OpenFileMode

FILE_MODE_BINARY

FILE_MODE_BINARY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- WriteBytes
- OpenFileMode

FILE_MODE_BINARYREADONLY

FILE_MODE_BINARYREADONLY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- OpenFileMode

FILE_MODE_NORMAL

FILE_MODE_NORMAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- OpenFileMode

FILE_NOT_FOUND

FILE_NOT_FOUND は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerGetFileVersion
- FileGrep
- FileInsertLine
- FileCompare
- FileDeleteLine
- VerFindFileVersion

FILE_NO_VERSION

FILE_NO_VERSION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerGetFileVersion
- VerSearchAndUpdateFile
- VerFindFileVersion
- VerUpdateFile

FILE_RD_ONLY

FILE_RD_ONLY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerUpdateFile
- FileDeleteLine
- FileInsertLine
- VerSearchAndUpdateFile

FILE_READ_ATTRIBUTES

FILE_READ_ATTRIBUTES は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

FILE_READ_DATA

FILE_READ_DATA は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

FILE_READ_EA

FILE_LIST_DIRECTORY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

FILE_SHARED_COUNT

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FILE_SHARED_COUNT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetFileInfo

FILE_SIZE

FILE_SIZE (FILE_SIZE_LOW と同じです) は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetFileInfo

FILE_SIZE_HIGH

FILE_SIZE_HIGH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetFileInfo

FILE_SIZE_LOW

FILE_SIZE_LOW は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetFileInfo

FILE_SRC_OLD

FILE_SRC_OLD は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerSearchAndUpdateFile
- VerUpdateFile

FILE_TIME

FILE_TIME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetFileInfo
- SetFileInfo

FILE_TRAVERSE

FILE_TRAVERSE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

FILE_WRITE_ATTRIBUTES

FILE_WRITE_ATTRIBUTES は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

FILE_WRITE_DATA

FILE_WRITE_DATA は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

FILE_WRITE_EA

FILE_WRITE_EA は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

FILE_WRITEABLE

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *Installscript*
- *InstallScript オブジェクト*

FILE_WRITEABLE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

FILENAME

FILENAME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ParsePath

FILENAME_ONLY

FILENAME_ONLY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ParsePath

FIXED_DRIVE

FIXED_DRIVE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetValidDrivesList

FONT_AVAILABLE

FONT_AVAILABLE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

FULL

FULL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PathAdd
- PathFind
- PathMove
- PathDelete

FULLSCREEN

FULLSCREEN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceBitmap

FULLSCREENSIZE

FULLSCREENSIZE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceBitmap

FULLWINDOWMODE

FULLWINDOWMODE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Enable

FUNCTION_EXPORTED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FUNCTION_EXPORTED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

GBYTES

GBYTES は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ConvertSizeToUnits
- StrConvertSizeUnit

GENERIC_ALL

GENERIC_ALL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

GENERIC_EXECUTE

GENERIC_EXECUTE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

GENERIC_READ

GENERIC_READ は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

GENERIC_WRITE

GENERIC_WRITE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

GREATER_THAN

GREATER_THAN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FileCompare
- VerCompare

GREEN

GREEN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor
- SetTitle

GTFIS_OPTION_DELETE_TEMP_FILE

GTFIS_OPTION_DELETE_TEMP_FILE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetTempFileNameIS

GTFIS_OPTION_DONT_CREATE_DIR

GTFIS_OPTION_DONT_CREATE_DIR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetTempFileNameIS

GTFIS_OPTION_DONT_RESOLVE_TEXTSUBS

GTFIS_OPTION_DONT_RESOLVE_TEXTSUBS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetTempFileNameIS

GTFIS_OPTION_NONE

GTFIS_OPTION_NONE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetTempFileNameIS

HELP

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

HELP は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Do
- HandlerEx

HIDE_DISABLED_BTNS

HIDE_DISABLED_BTNS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

HIDE_DISABLED_BTNS 定数は、[ユーザーごと] および [すべてのユーザー] オプションを無効にするのではなく、非表示にすることを示します。この変数のデフォルト値は、TRUE です。この変数が TRUE に設定されると、オプションのどちらかが無効である場合、両方のオプションが非表示となります。

次の関数と共に利用します

- SdCustomerInformation
- SdCustomerInformationEx

HKEY_CLASSES_ROOT

HKEY_CLASSES_ROOT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBSetDefaultRoot

- RegDBSetKeyValueEx
- RegDBDeleteKey
- RegDBDeleteValue
- RegDBGetKeyValueEx
- RegDBKeyExist
- RegDBCreateKeyEx

HKEY_CURRENT_USER

HKEY_CURRENT_USER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBSetDefaultRoot

HKEY_LOCAL_MACHINE

HKEY_LOCAL_MACHINE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

メモ・Windows NT 4.0 では、HKEY_LOCAL_MACHINE の下に直接キーを作成することはできません。

次の関数と共に利用します

- RegDBConnectRegistry
- InstallationInfo
- RegDBSetDefaultRoot

HKEY_USERS

HKEY_USERS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

メモ・Windows NT 4.0 では、HKEY_USERS の下に直接キーを作成することはできません。

次の関数と共に利用します

- RegDBSetDefaultRoot
- RegDBConnectRegistry

HKEY_USER_SELECTABLE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

HKEY_USER_SELECTABLE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBSetDefaultRoot

HOURGLASS

HOURGLASS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Disable
- Enable

HWND_DESKTOP

HWND_DESKTOP は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetWindowHandle

HWND_INSTALL

HWND_INSTALL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetWindowHandle

IDCANCEL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

IDCANCEL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SelectDir
- SelectDirEx

IDOK

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

IDOK は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SelectDir
- SelectDirEx

IDS_IFX_ERROR_INVALID_MEDIA_PASSWORD

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

IDS_IFX_ERROR_INVALID_MEDIA_PASSWORD は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SdLoadString

IFX_ONNEXTDISK_PACKAGE_CAPTION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

IFX_ONNEXTDISK_PACKAGE_CAPTION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SdLoadString

IFX_ONNEXTDISK_PACKAGE_MSG

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

IFX_ONNEXTDISK_PACKAGE_MSG は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SdLoadString

INCLUDE_SUBDIR

INCLUDE_SUBDIR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FindAllDirs
- XCopyFile

INDVFILESTATUS

INDVFILESTATUS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Disable
- SetStatusWindow
- Enable

INFORMATION

INFORMATION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MessageBox
- SprintfBox

IS_PERMISSIONS_OPTION_64BIT_OBJECT

IS_PERMISSIONS_OPTION_64BIT_OBJECT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

IS_PERMISSIONS_OPTION_ALLOW_ACCESS

IS_PERMISSIONS_OPTION_ALLOW_ACCESS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

IS_PERMISSIONS_OPTION_DENY_ACCESS

IS_PERMISSIONS_OPTION_DENY_ACCESS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

IS_PERMISSIONS_OPTION_NO_APPLYDOWN

IS_PERMISSIONS_OPTION_NO_APPLYDOWN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

IS_PERMISSIONS_TYPE_FILE

IS_PERMISSIONS_TYPE_FILE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

IS_PERMISSIONS_TYPE_FOLDER

IS_PERMISSIONS_TYPE_FOLDER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

IS_PERMISSIONS_TYPE_REGISTRY

IS_PERMISSIONS_TYPE_REGISTRY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

ISDIFX_OPTION_DONT_ASSOCIATE

ISDIFX_OPTION_DONT_ASSOCIATE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DIFxDriverPackageInstall
- DIFxDriverPackageUninstall

ISDIFX_OPTION_DONT_RESOLVE_TEXTSUBS

ISDIFX_OPTION_DONT_RESOVE_TEXTSUBS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DIFxDriverPackageGetPath
- DIFxDriverPackageInstall
- DIFxDriverPackagePreinstall
- DIFxDriverPackageUninstall

ISDIFX_OPTION_LOG_IN_DRIVER_PACKAGE_PATH

ISDIFX_OPTION_LOG_IN_DRIVER_PACKAGE_PATH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DIFxDriverPackageInstall
- DIFxDriverPackagePreinstall

ISDIFX_OPTION_NO_REPAIR

ISDIFX_OPTION_NO_REPAIR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DIFxDriverPackageInstall
- DIFxDriverPackagePreinstall

ISERR_GEN_FAILURE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISERR_GEN_FAILURE は定義済みの定数で、ビルトイン関数が失敗し、失敗の原因についてより具体的な説明が提供されないときに、そのビルトイン関数によって返される値を表わすために使用されます。定義済みの定数の値を変更することはできません。

ISERR_SUCCESS

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

ISERR_SUCCESS は定義済みの定数で、ビルトイン関数が成功したときに、ビルトイン関数によって返される値を表わすために使用されます。定義済みの定数の値を変更することはできません。

ISLANG_AFRIKAANS

ISLANG_AFRIKAANS は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_AFRIKAANS_STANDARD

ISLANG_AFRIKAANS は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ALBANIAN

ISLANG_ALBANIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ALBANIAN_STANDARD

ISLANG_ALBANIAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ALL

ISLANG_ALL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterLanguage

ISLANG_ARABIC

ISLANG_ARABIC は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_ALGERIA

ISLANG_ARABIC_ALGERIA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_BAHRAIN

ISLANG_ARABIC_BAHRAIN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_EGYPT

ISLANG_ARABIC_EGYPT は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_IRAQ

ISLANG_ARABIC_IRAQ は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_JORDAN

ISLANG_ARABIC_JORDAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_KUWAIT

ISLANG_ARABIC_KUWAIT は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_LEBANON

ISLANG_ARABIC_LEBANON は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_LIBYA

ISLANG_ARABIC_LIBYA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_MOROCCO

ISLANG_ARABIC_MOROCCO は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_OMAN

ISLANG_ARABIC_OMAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_QATAR

ISLANG_ARABIC_QATAR は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_SAUDIARABIA

ISLANG_ARABIC_SAUDIARABIA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_SYRIA

ISLANG_ARABIC_SYRIA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_TUNISIA

ISLANG_ARABIC_TUNISIA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_UAE

ISLANG_ARABIC_UAE は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ARABIC_YEMEN

ISLANG_ARABIC_YEMEN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_BASQUE

ISLANG_BASQUE は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_BASQUE_STANDARD

ISLANG_BASQUE_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_BELARUSIAN

ISLANG_BELARUSIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_BELARUSIAN_STANDARD

ISLANG_BELARUSIAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_BULGARIAN

ISLANG_BULGARIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_BULGARIAN_STANDARD

ISLANG_BULGARIAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_CATALAN

ISLANG_CATALAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_CATALAN_STANDARD

ISLANG_CATALAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_CHINESE

ISLANG_CHINESE は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_CHINESE_HONGKONG

ISLANG_CHINESE_HONGKONG は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_CHINESE_PRC

ISLANG_CHINESE_PRC は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_CHINESE_SINGAPORE

ISLANG_CHINESE_SINGAPORE は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_CHINESE_TAIWAN

SLANG_CHINESE_TAIWAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_CROATIAN

ISLANG_CROATIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_CROATIAN_STANDARD

ISLANG_CROATIAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_CZECH

ISLANG_CZECH は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_CZECH_STANDARD

ISLANG_CZECH_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_DANISH

ISLANG_DANISH は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_DANISH_STANDARD

ISLANG_DANISH_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_DUTCH

ISLANG_DUTCH は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_DUTCH_BELGIAN

ISLANG_DUTCH_BELGIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_DUTCH_STANDARD

ISLANG_DUTCH_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ENGLISH

ISLANG_ENGLISH は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ENGLISH_AUSTRALIAN

ISLANG_ENGLISH_AUSTRALIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ENGLISH_BELIZE

ISLANG_ENGLISH_BELIZE は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ENGLISH_CANADIAN

ISLANG_ENGLISH_CANADIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ENGLISH_CARIBBEAN

ISLANG_ENGLISH_CARIBBEAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ENGLISH_IRELAND

ISLANG_ENGLISH_IRELAND は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ENGLISH_JAMAICA

ISLANG_ENGLISH_JAMAICA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ENGLISH_NEWZEALAND

ISLANG_ENGLISH_NEWZEALAND は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ENGLISH_SOUTHAFRICA

ISLANG_ENGLISH_SOUTHAFRICA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ENGLISH_TRINIDAD

ISLANG_ENGLISH_TRINIDAD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ENGLISH_UNITEDKINGDOM

ISLANG_ENGLISH_UNITEDKINGDOM は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ENGLISH_UNITEDSTATES

ISLANG_ENGLISH_UNITEDSTATES は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ESTONIAN

ISLANG_ESTONIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ESTONIAN_STANDARD

ISLANG_ESTONIAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_FAEROESE

ISLANG_FAEROESE は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_FAEROESE_STANDARD

ISLANG_FAEROESE_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_FARSI

ISLANG_FARSI は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_FARSI_STANDARD

ISLANG_FARSI_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_FINNISH

ISLANG_FINNISH は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_FINNISH_STANDARD

ISLANG_FINNISH_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_FRENCH

ISLANG_FRENCH は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_FRENCH_BELGIAN

ISLANG_FRENCH_BELGIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_FRENCH_CANADIAN

ISLANG_FRENCH_CANADIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_FRENCH_LUXEMBOURG

ISLANG_FRENCH_LUXEMBOURG は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_FRENCH_STANDARD

ISLANG_FRENCH_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_FRENCH_SWISS

ISLANG_FRENCH_SWISS は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_GERMAN

ISLANG_GERMAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_GERMAN_AUSTRIAN

ISLANG_GERMAN_AUSTRIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_GERMAN_LIECHTENSTEIN

ISLANG_GERMAN_LIECHTENSTEIN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_GERMAN_LUXEMBOURG

ISLANG_GERMAN_LUXEMBOURG は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_GERMAN_STANDARD

ISLANG_GERMAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_GERMAN_SWISS

ISLANG_GERMAN_SWISS は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_GREEK

ISLANG_GREEK は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_GREEK_STANDARD

ISLANG_GREEK_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_HEBREW

ISLANG_HEBREW は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_HEBREW_STANDARD

ISLANG_HEBREW_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_HUNGARIAN

ISLANG_HUNGARIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_HUNGARIAN_STANDARD

ISLANG_HUNGARIAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ICELANDIC

ISLANG_ICELANDIC は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ICELANDIC_STANDARD

ISLANG_ICELANDIC_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_INDONESIAN

ISLANG_INDONESIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_INDONESIAN_STANDARD

ISLANG_INDONESIAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ITALIAN

ISLANG_ITALIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ITALIAN_STANDARD

ISLANG_ITALIAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ITALIAN_SWISS

ISLANG_ITALIAN_SWISS は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_JAPANESE

ISLANG_JAPANESE は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_JAPANESE_STANDARD

ISLANG_JAPANESE_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_KOREAN

ISLANG_KOREAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_KOREAN_JOHAB

ISLANG_KOREAN_JOHAB は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_KOREAN_STANDARD

ISLANG_KOREAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_LATVIAN

ISLANG_LATVIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_LATVIAN_STANDARD

ISLANG_LATVIAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_LITHUANIAN

ISLANG_LITHUANIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_LITHUANIAN_STANDARD

ISLANG_LITHUANIAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_NORWEGIAN

ISLANG_NORWEGIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_NORWEGIAN_BOKMAL

ISLANG_NORWEGIAN_BOKMAL は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_NORWEGIAN_NYNORSK

ISLANG_NORWEGIAN_NYNORSK は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_POLISH

ISLANG_POLISH は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_POLISH_STANDARD

ISLANG_POLISH_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_PORTUGUESE

ISLANG_PORTUGUESE は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_PORTUGUESE_BRAZILIAN

ISLANG_PORTUGUESE_BRAZILIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_PORTUGUESE_STANDARD

ISLANG_PORTUGUESE_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ROMANIAN

ISLANG_ROMANIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_ROMANIAN_STANDARD

ISLANG_ROMANIAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_RUSSIAN

ISLANG_RUSSIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_RUSSIAN_STANDARD

ISLANG_RUSSIAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SERBIAN_CYRILLIC

ISLANG_SERBIAN_CYRILLIC は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SERBIAN_LATIN

ISLANG_SERBIAN_LATIN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SLOVAK

ISLANG_SLOVAK は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SLOVAK_STANDARD

ISLANG_SLOVAK_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SLOVENIAN

ISLANG_SLOVENIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SLOVENIAN_STANDARD

ISLANG_SLOVENIAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH

ISLANG_SPANISH は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_ARGENTINA

ISLANG_SPANISH_ARGENTINA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_BOLIVIA

ISLANG_SPANISH_BOLIVIA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_CHILE

ISLANG_SPANISH_CHILE は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_COLOMBIA

ISLANG_SPANISH_COLOMBIA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_COSTARICA

ISLANG_SPANISH_COSTARICA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_DOMINICANREPUBLIC

ISLANG_SPANISH_DOMINICANREPUBLIC は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_ECUADOR

ISLANG_SPANISH_ECUADOR は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_ELSALVADOR

ISLANG_SPANISH_ELSALVADOR は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_GUATEMALA

ISLANG_SPANISH_GUATEMALA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_HONDURAS

ISLANG_SPANISH_HONDURAS は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_MEXICAN

ISLANG_SPANISH_MEXICAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_MODERNSORT

ISLANG_SPANISH_MODERNSORT は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_NICARAGUA

ISLANG_SPANISH_NICARAGUA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_PANAMA

ISLANG_SPANISH_PANAMA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_PARAGUAY

ISLANG_SPANISH_PARAGUAY は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_PERU

ISLANG_SPANISH_PERU は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_PUERTORICO

ISLANG_SPANISH_PUERTORICO は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_TRADITIONALSORT

ISLANG_SPANISH_TRADITIONALSORT は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_URUGUAY

ISLANG_SPANISH_TRADITIONALSORT は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SPANISH_VENEZUELA

ISLANG_SPANISH_VENEZUELA は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SWEDISH

ISLANG_SWEDISH は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SWEDISH_FINLAND

ISLANG_SWEDISH_FINLAND は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_SWEDISH_STANDARD

ISLANG_SWEDISH_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_THAI

ISLANG_THAI は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_THAI_STANDARD

ISLANG_THAI_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_TURKISH

ISLANG_TURKISH は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_TURKISH_STANDARD

ISLANG_TURKISH_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_UKRAINIAN

ISLANG_UKRAINIAN は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_UKRAINIAN_STANDARD

ISLANG_UKRAINIAN_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_VIETNAMESE

ISLANG_VIETNAMESE は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISLANG_VIETNAMESE_STANDARD

ISLANG_VIETNAMESE_STANDARD は Windows language ID に対応する定義済みの定数です。この定数の使用方法については、「言語識別子」をご覧ください。

ISOSL_ALL

ISOSL_ALL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOSL_SUPPORTED

ISOSL_SUPPORTED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOSL_WIN7_SERVER2008R2

ISOSL_WIN7_SERVER2008R2 は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOSL_WIN8

ISOSL_WIN8 は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOSL_WIN81

ISOSL_WIN81 は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOSL_WIN10

ISOSL_WIN10 は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOSL_WINSERVER2003

ISOSL_WINSERVER2003 は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOSL_WINVISTA

メモ・*ISOSL_WINVISTA_SERVER2008 は ISOSL_WINVISTA に優先します。*

ISOSL_WINME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOSL_WINVISTA_SERVER2008

ISOSL_WINVISTA_SERVER2008 は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOSL_WINXP

ISOSL_WINXP は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_ALL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_ALL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_BACKOFFICE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_BACKOFFICE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_DATACENTER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_DATACENTER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_ENTERPRISE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_ENTERPRISE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_PROC_ARCH_32

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_PROC_ARCH_32 は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_PROC_ARCH_AMD64

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_PROC_ARCH_AMD64 は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_PROC_ARCH_IA64

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_PROC_ARCH_IA64 は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_SERVER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_SERVER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_SERVER2003_R2

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_SERVER2003_R2 は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_SMALLBUSINESS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_SMALLBUSINESS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_SMALLBUSINESS_RESTRICTED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_SMALLBUSINESS_RESTRICTED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_TERMINAL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_TERMINAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_WORKSTATION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_WORKSTATION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_XP_HOME

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_XP_HOME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISOS_ST_XP_PRO

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ISOS_ST_XP_PRO は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFilterOS

ISUS_AGENT_FEATURE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この定数は現在使用されていません。FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

ISUS_MAIN_FEATURE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この定数は現在使用されていません。FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

ISUS_TEXTSUB_HOST

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この定数は現在使用されていません。FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

ISUS_TEXTSUB_INTERVAL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この定数は現在使用されていません。FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

ISUS_TEXTSUB_LANGUAGE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この定数は現在使用されていません。FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

ISUS_TEXTSUB_LOGO

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この定数は現在使用されていません。FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

ISUS_TEXTSUB_MANAGER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この定数は現在使用されていません。FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

ISUS_TEXTSUB_VERSION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この定数は現在使用されていません。FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

ISUS_UPDATEMANAGER_FEATURE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この定数は現在使用されていません。FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

IS_386

IS_386 は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_486

IS_486 は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_ALPHA

IS_ALPHA は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_CDROM

IS_CDROM は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_EGA

IS_EGA は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_FIXED

IS_FIXED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_FOLDER

IS_FOLDER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- QueryProgItem
- ReplaceFolderIcon

IS_ITEM

IS_ITEM は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- QueryProgItem
- ReplaceFolderIcon

IS_PENTIUM

IS_PENTIUM は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_REMOTE

IS_REMOTE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_REMOVABLE

IS_REMOVABLE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_SVGA

IS_SVGA は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_UNKNOWN

IS_UNKNOWN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_UVGA

IS_UVGA は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_VGA

IS_VGA は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_WINDOWS

IS_WINDOWS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_WINDOWS9X

IS_WINDOWS9X は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_WINDOWSNT

IS_WINDOWSNT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

IS_XVGA

IS_XVGA は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

KBYTES

KBYTES は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ConvertSizeToUnits
- StrConvertSizeUnit

KEY_CREATE_LINK

KEY_CREATE_LINK は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

KEY_CREATE_SUB_KEY

KEY_CREATE_SUB_KEY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

KEY_ENUMERATE_SUB_KEYS

KEY_ENUMERATE_SUB_KEYS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

KEY_NOTIFY

GENERIC_WRITE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

KEY_QUERY_VALUE

KEY_QUERY_VALUE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

KEY_SET_VALUE

KEY_SET_VALUE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

LAAW_OPTION_CHANGEDIRECTORY

LAAW_OPTION_CHANGEDIRECTORY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- LaunchApplication

LAAW_OPTION_FIXUP_PROGRAM

LAAW_OPTION_CHANGEDIRECTORY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- LaunchApplication

LAAW_OPTION_HIDDEN

LAAW_OPTION_HIDDEN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DoInstall
- LaunchAppAndWait
- LaunchApplication
- WaitForApplication

LAAW_OPTION_MAXIMIZED

LAAW_OPTION_MAXIMIZED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DoInstall
- LaunchAppAndWait
- LaunchApplication
- WaitForApplication

LAAW_OPTION_MINIMIZED

LAAW_OPTION_MINIMIZED は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DoInstall
- LaunchAppAndWait
- LaunchApplication
- WaitForApplication

LAAW_OPTION_NO_CHANGEDIRECTORY

LAAW_OPTION_NO_CHANGEDIRECTORY は旧式の定義済み定数です。

LAAW_OPTION_NOWAIT

LAAW_OPTION_NOWAIT は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DoInstall
- LaunchAppAndWait
- LaunchApplication
- WaitForApplication

LAAW_OPTION_SET_BATCH_INSTALL

プロジェクト・*この定数は、InstallScript プロジェクトのイベント ドリブン型コードおよび InstallScript MSI プロジェクトで使用できます。基本の MSI、InstallScript MSI、またはスイート / アドバンスト UI プロジェクトの InstallScript カスタム アクションでは使用できません。*

LAAW_OPTION_SET_BATCH_INSTALL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DoInstall
- LaunchAppAndWait
- LaunchApplication
- WaitForApplication

LAAW_OPTION_SHOW_HOURGLASS

LAAW_OPTION_SHOW_HOURGLASS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DoInstall
- LaunchAppAndWait

LAAW_OPTION_USE_CALLBACK

LAAW_OPTION_USE_CALLBACK は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DoInstall
- LaunchAppAndWait
- LaunchApplication
- WaitForApplication

LAAW_OPTION_USE_SHELLEXECUTE

LAAW_OPTION_USE_SHELLEXECUTE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- LaunchApplication

LAAW_OPTION_WAIT

LAAW_OPTION_WAIT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DoInstall
- LaunchAppAndWait
- LaunchApplication
- WaitForApplication

LAAW_OPTION_WAIT_INCL_CHILD

LAAW_OPTION_WAIT_INCL_CHILD は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DoInstall
- LaunchAppAndWait
- LaunchApplication
- WaitForApplication

LANGUAGE_SUPPORTED

LANGUAGE_SUPPORTED は、インストーラーでサポートされる言語を指定するのに使用される定義済み定数です。言語は 0x で始まる 4 桁の 16 進数言語コードです。たとえば、英語の値は 0x0409 です。
STANDARD_SELECTED_LANGUAGE を使ってこの形式で文字列を作成するには、次のようなステートメントを使用します :

```
Sprintf (szLang, "0x%.04lx", STANDARD_SELECTED_LANGUAGE);
```

定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

LANGUAGE

LANGUAGE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

LESS_THAN

LESS_THAN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FileCompare
- VerCompare

LINE_NUMBER

LINE_NUMBER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FileDeleteLine
- FileInsertLine

LISTBOX_ENTER

LISTBOX_ENTER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CtrlGetSubCommand

LISTBOX_SELECT

LISTBOX_SELECT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- CtrlGetSubCommand

LISTFIRST

LISTFIRST は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ListSetIndex

LISTLAST

LISTLAST は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ListSetIndex

LISTNEXT

LISTNEXT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ListSetIndex

LISTPREV

LISTPREV は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ListSetIndex

LIST_NULL

LIST_NULL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ListCreate

LOCKEDFILE

LOCKEDFILE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- XCopyFile
- InstallationInfo
- VerUpdateFile
- DeinstallStart

LOGGING

LOGGING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DeinstallStart
- Disable
- Enable
- InstallationInfo

LOWER_LEFT

LOWER_LEFT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceBitmap
- PlaceWindow

LOWER_RIGHT

LOWER_RIGHT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceBitmap
- PlaceWindow

LWTF_OPTION_APPEND_TO_FILE

LWTF_OPTION_APPEND_TO_FILE は定義済みの定数で、1 つまたは複数のビルトイン関数との間でやり取りされる値を表すのに利用します。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ListWriteToFileEx

LWTF_OPTION_WRITE_AS_ANSI

LWTF_OPTION_WRITE_AS_ANSI は定義済みの定数で、1 つまたは複数のビルトイン関数との間でやり取りされる値を表すのに利用します。定義済みの定数の値を変更することはできません。

メモ・InstallShield の初期のバージョンでは、この定数は LWFT_OPTION_WRITE_AS_ANSI と呼ばれていました (LWFT が、LWTF の代わりに使用されていました)。後方互換性を維持するために、これらの定数は現在も使用することができます。これらの定数は同じ方法で定義されています。

次の関数と共に利用します

- ListWriteToFileEx

LWTF_OPTION_WRITE_AS_UNICODE

LWTF_OPTION_WRITE_AS_UNICODE は定義済みの定数で、1 つまたは複数のビルトイン関数との間でやり取りされる値を表すのに利用します。定義済みの定数の値を変更することはできません。

メモ・InstallShield の初期のバージョンでは、この定数は LWFT_OPTION_WRITE_AS_UNICODE と呼ばれていました (LWFT が、LWTF の代わりに使用されていました)。後方互換性を維持するために、これらの定数は現在も使用することができます。これらの定数は同じ方法で定義されています。

次の関数と共に利用します

- ListWriteToFileEx

MAGENTA

MAGENTA は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor
- SetTitle

MATH_COPROCESSOR

MATH_COPROCESSOR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

MBYTES

MBYTES は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ConvertSizeToUnits
- StrConvertSizeUnit

MEDIA_FIELD_ADDREMOVE_NOMODIFY

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_ADDREMOVE_NOMODIFY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData
- MediaGetDataEx

MEDIA_FIELD_ADDREMOVE_NOREMOVE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_ADDREMOVE_NOREMOVE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData
- MediaGetDataEx

MEDIA_FIELD_COMPANY_NAME

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_COMPANY_NAME は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData

MEDIA_FIELD_MEDIA_FLAGS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_MEDIA_FLAGS は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData

MEDIA_FIELD_PREVIOUS_VERSIONS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_PREVIOUS_VERSIONS は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData

MEDIA_FIELD_PRODUCT_COMMENTS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_PRODUCT_COMMENTS は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData
- MediaGetDataEx

MEDIA_FIELD_PRODUCT_EXE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_PRODUCT_EXE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData

MEDIA_FIELD_PRODUCT_ICON

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_PRODUCT_ICON は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData
- MediaGetDataEx

MEDIA_FIELD_PRODUCT_NAME

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_PRODUCT_NAME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData

MEDIA_FIELD_PRODUCT_README

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_PRODUCT_README は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData
- MediaGetDataEx

MEDIA_FIELD_PRODUCT_SUPPORT_CONTACT

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_PRODUCT_SUPPORT_CONTACT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData
- MediaGetDataEx

MEDIA_FIELD_PRODUCT_SUPPORT_PHONE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_PRODUCT_SUPPORT_PHONE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData
- MediaGetDataEx

MEDIA_FIELD_PRODUCT_SUPPORT_URL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_PRODUCT_SUPPORT_URL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData
- MediaGetDataEx

MEDIA_FIELD_PRODUCT_UPDATE_URL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_PRODUCT_UPDATE_URL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData
- MediaGetDataEx

MEDIA_FIELD_PRODUCT_URL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_PRODUCT_URL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData
- MediaGetDataEx

MEDIA_FIELD_PRODUCT_VERSION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_PRODUCT_VERSION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData

MEDIA_FIELD_TARGETDIR

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FIELD_TARGETDIR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData

MEDIA_FLAG_FORMAT_DIFFERENTIAL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FLAG_FORMAT_DIFFERENTIAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData

MEDIA_FLAG_FORMAT_PATCH

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FLAG_FORMAT_PATCH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MediaGetData

MEDIA_FLAG_UPDATEMODE_SUPPORTED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_FLAG_UPDATEMODE_SUPPORTED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

MEDIA_FLAG_UPDATEMODE_SUPPORTED フラグは常に設定されています。

次の関数と共に利用します

- MediaGetData
- MediaGetDataEx

MEDIA_PASSWORD_KEY

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MEDIA_PASSWORD_KEY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- LogReadCustomString
- LogWriteCustomString

METAFILE

METAFILE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SizeWindow

MMEDIA_AVI

MMEDIA_AVI は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlayMMedia
- PlaceWindow
- SizeWindow

MMEDIA_MIDI

MMEDIA_MIDI は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlayMMedia

MMEDIA_PLAYASYNCH

MMEDIA_PLAYASYNCH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlayMMedia

MMEDIA_PLAYCONTINUOUS

MMEDIA_PLAYCONTINUOUS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlayMMedia

MMEDIA_PLAYSYNCH

MMEDIA_PLAYSYNCH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlayMMedia

MMEDIA_STOP

MMEDIA_STOP は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlayMMedia

MMEDIA_SWF

MMEDIA_SWF は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlayMMedia
- PlaceWindow
- SizeWindow

MMEDIA_WAVE

MMEDIA_WAVE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlayMMedia

MODIFY

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MODIFY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SdWelcomeMaint

NEXT

NEXT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- AskDestPath
- AskOptions
- AskPath
- AskText
- FeatureDialog
- SdAskDestPath
- SdAskOptions
- SdAskOptionsList
- SdBitmap
- SdDisplayTopics
- SdFeatureDialog
- SdFeatureDialog2
- SdFeatureDialogAdv

- SdFeatureMult
- SdLicense
- SdOptionsButtons
- SdRegisterUser
- SdRegisterUserEx
- SdSelectFolder
- SdShowAnyDialog
- SdShowDlgEdit1
- SdShowDlgEdit2
- SdShowDlgEdit3
- SdShowFileMods
- SdShowInfoList
- SdStartCopy
- SdWelcome
- SelectFolder
- Welcome

NEXTBUTTON

NEXTBUTTON は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Disable
- Enable
- Is

NO

NO は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SdConfirmNewDir
- SdConfirmRegistration
- AskYesNo

NONEXCLUSIVE

NONEXCLUSIVE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SdAskOptionsList
- AskOptions
- SdAskOptions

NORMALMODE

NORMALMODE は、定義済みの定数で、セットアップがサイレント モードで実行されているかどうかをテストするのに利用することができます。詳細については、InstallShield システム変数 MODE を参照してください。

NORMAL_PRIORITY_CLASS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

NORMAL_PRIORITY_CLASS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

NOSET

NOSET は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- EzBatchAddString

NOTEXISTS

NOTEXISTS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ExistsDir

- ExistsDisk

NO_SUBDIR

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

NO_SUBDIR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureFileEnum

NULL

NULL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- AddFolderIcon
- CreateShortcut
- GetShortcutInfo
- FindWindow
- QueryProgItem
- ReplaceFolderIcon
- ReplaceShortcut

NUMBERLIST

NUMBERLIST は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ListCreate

OFF

OFF は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- StatusUpdate

OK

OK は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- EnterDisk

ON

ON は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- StatusUpdate

ONLYDIR

ONLYDIR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DeleteDir

OTHER_FAILURE

OTHER_FAILURE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FileDeleteLine
- VerUpdateFile
- FileCompare
- FileGrep
- FileInsertLine

OUT_OF_DISK_SPACE

OUT_OF_DISK_SPACE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerUpdateFile
- FileInsertLine
- FileDeleteLine
- VerSearchAndUpdateFile

PARALLEL

PARALLEL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

PARTIAL

PARTIAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PathAdd
- PathFind
- PathMove
- PathDelete

PATH

PATH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ParsePath

PATH_EXISTS

≫ InstallScript 言語リファレンス

PATH_EXISTS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

PCRESTORE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

PCRESTORE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Disable
- Enable

PERSONAL

PERSONAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ProgDefGroupType

READ_CONTROL

READ_CONTROL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS
- SetObjectPermissions

REBOOTED

REBOOTED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

RECORDMODE

RECORDMODE 定義済みの定数で、セットアップが Windows フォルダー内で自動的にサイレントセットアップファイル (.iss ファイル) を生成するかどうかをテストするのに利用します。これはセットアップインプットの記録です。詳細については、InstallScript システム変数 MODE を参照してください。

RED

RED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor
- SetTitle

REGDBREMOTEREGCONNECTED

REGDBREMOTEREGCONNECTED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

REGDB_APPPATH

REGDB_APPPATH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_APPPATH_DEFAULT

REGDB_APPPATH_DEFAULT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_BINARY

REGDB_BINARY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBSetAppInfo
- RegDBGetKeyValueEx
- RegDBSetKeyValueEx
- RegDBGetAppInfo
- RegDBSetKeyValueEx

REGDB_ERR_CONNECTIONEXISTS

REGDB_ERR_CONNECTIONEXISTS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBConnectRegistry

REGDB_ERR_CORRUPTEDREGISTRY

REGDB_ERR_CORRUPTEDREGISTRY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBConnectRegistry

REGDB_ERR_INITIALIZATION

REGDB_ERR_INITIALIZATION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBConnectRegistry

REGDB_ERR_INVALIDHANDLE

REGDB_ERR_INVALIDHANDLE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBConnectRegistry

REGDB_ERR_INVALIDNAME

REGDB_ERR_INVALIDNAME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBConnectRegistry

REGDB_KEYPATH_APPPATHS

REGDB_KEYPATH_APPPATHS は定義済みの定数で、その値は 一般アプリケーションパスキーのレジストリの場所です (ルートキーを除く)。これは **Software¥Microsoft¥Windows¥CurrentVersion¥App Paths¥** です。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。定義済みの定数の値を変更することはできません。

REGDB_KEYPATH_DOTNET_10

REGDB_KEYPATH_DOTNET_10 は定義済みの定数で、その値は .NET Framework のバージョン 1.0 のレジストリ キーの場所 (ルート キーを含まない) です。この定数は、次のように定義されています。

Software¥Microsoft¥NET Framework Setup¥Full¥v1.0.3705¥1033¥Microsoft .NET Framework Full v1.0.3705 (1033)¥

定義済みの定数の値を変更することはできません。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。この定義済み定数は **Is** 関数を使用した場合にもサポートされます。

次の関数と共に利用します

- Is

REGDB_KEYPATH_DOTNET_11

REGDB_KEYPATH_DOTNET_11 は定義済みの定数で、その値は .NET Framework のバージョン 1.1 のレジストリ キーの場所（ルート キーを含まない）です。この定数は、次のように定義されています。

Software¥Microsoft¥NET Framework Setup¥NDP¥v1.1.4322¥

定義済みの定数の値を変更することはできません。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。この定義済み定数は **Is** 関数を使用した場合にもサポートされます。

次の関数と共に利用します

- Is

REGDB_KEYPATH_DOTNET_20

REGDB_KEYPATH_DOTNET_20 は定義済みの定数で、その値は .NET Framework のバージョン 2.0 のレジストリ キーの場所（ルート キーを含まない）です。この定数は、次のように定義されています。

Software¥Microsoft¥NET Framework Setup¥NDP¥v2.0.50215¥

定義済みの定数の値を変更することはできません。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。この定義済み定数は **Is** 関数を使用した場合にもサポートされます。

次の関数と共に利用します

- Is

REGDB_KEYPATH_DOTNET_30

REGDB_KEYPATH_DOTNET_30 は定義済みの定数で、その値は .NET Framework 3.0 の RTM バージョンのレジストリ キーの場所（ルート キーを含まない）です。この定数は、次のように定義されています。

Software¥Microsoft¥NET Framework Setup¥NDP¥v3.0¥Setup¥

ヒント・*.NET Framework 3.0 の RTM バージョンがインストールされているかどうかを検出するには、REGDB_KEYPATH_DOTNET_30 変数を使用します。.NET Framework 3.0 の SP1、またはそれ以降のサービス パックがインストールされているかどうかを検出するには、REGDB_KEYPATH_DOTNET_30_SP を使用します。*

定義済みの定数の値を変更することはできません。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。この定義済み定数は **Is** 関数を使用した場合にもサポートされます。

次の関数と共に利用します

- Is

REGDB_KEYPATH_DOTNET_30_SP

REGDB_KEYPATH_DOTNET_30_SP は定義済みの定数で、その値は .NET Framework 3.0 SP1 またはそれ以降のサービス パックのレジストリ キーの場所 (ルート キーは含みません) です。この定数は、次のように定義されています。

Software¥Microsoft¥NET Framework Setup¥NDP¥v3.0¥

***ヒント・**.NET Framework 3.0 の SP1 またはそれ以降のサービス パックがインストールされているかどうかをクエリするとき、REGDB_KEYPATH_DOTNET_30_SP 変数を使用できます。.NET Framework 3.0 の RTM バージョンがインストールされているかどうかを検出するには、REGDB_KEYPATH_DOTNET_30 を使用します。*

定義済みの定数の値を変更することはできません。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。この定義済み定数は **Is** 関数を使用した場合にもサポートされます。

次の関数と共に利用します

- Is

REGDB_KEYPATH_DOTNET_35

REGDB_KEYPATH_DOTNET_35 は定義済みの定数で、その値は .NET Framework のバージョン 3.5 のレジストリ キーの場所 (ルート キーを含まない) です。この定数は、次のように定義されています。

Software¥Microsoft¥NET Framework Setup¥NDP¥v3.5¥

定義済みの定数の値を変更することはできません。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。この定義済み定数は **Is** 関数を使用した場合にもサポートされます。

次の関数と共に利用します

- Is

REGDB_KEYPATH_DOTNET_40_CLIENT

REGDB_KEYPATH_DOTNET_40_CLIENT は定義済みの定数で、その値は .NET Framework Client Profile のバージョン 4.0 のレジストリ キーの場所 (ルート キーを含まない) です。この定数は、次のように定義されています。

Software¥Microsoft¥NET Framework Setup¥NDP¥v4¥Client

定義済みの定数の値を変更することはできません。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。この定義済み定数は **Is** 関数を使用した場合にもサポートされます。

次の関数と共に利用します

- Is

REGDB_KEYPATH_DOTNET_40_FULL

REGDB_KEYPATH_DOTNET_40_FULL は定義済みの定数で、その値は .NET Framework のバージョン 4.0 のレジストリ キーの場所 (ルート キーを含まない) です。この定数は、次のように定義されています。

Software¥Microsoft¥NET Framework Setup¥NDP¥v4¥Full

定義済みの定数の値を変更することはできません。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。この定義済み定数は **Is** 関数を使用した場合にもサポートされます。

次の関数と共に利用します

- Is

REGDB_KEYPATH_ISUNINSTINFO

ISUNINSTINFO は定義済みの定数で、その値は InstallShield Uninstall Information キーのレジストリの場所 (ルート キーは含みません) です。この定数は、次のように定義されています。

Software¥Microsoft¥Windows¥CurrentVersion¥Uninstall¥InstallShield Uninstall Information

定義済みの定数の値を変更することはできません。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。この定義済み定数は **Is** 関数を使用した場合にもサポートされます。

次の関数と共に利用します

- Is

REGDB_KEYPATH_RUN

REGDB_KEYPATH_RUN は定義済みの定数で、その値は 一般アプリケーションパスキーのレジストリの場所です (ルートキーを除く)。これは **Software¥Microsoft¥Windows¥CurrentVersion¥Run¥** です。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。定義済みの定数の値を変更することはできません。

REGDB_KEYPATH_RUNONCE

REGDB_KEYPATH_RUNONCE は定義済みの定数で、その値は 一般アプリケーションパスキーのレジストリの場所です (ルートキーを除く)。これは **Software¥Microsoft¥Windows¥CurrentVersion¥RunOnce¥** です。. この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。定義済みの定数の値を変更することはできません。

REGDB_KEYPATH_RUNONCEEX

REGDB_KEYPATH_RUNONCEEX は定義済みの定数で、その値は 一般アプリケーションパスキーのレジストリの場所です (ルートキーを除く)。これは **Software¥Microsoft¥Windows¥CurrentVersion¥RunOnceEx¥** です。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。定義済みの定数の値を変更することはできません。

REGDB_KEYPATH_SHAREDDLLS

REGDB_KEYPATH_SHAREDDLLS は定義済みの定数で、その値は 一般アプリケーションパスキーのレジストリの場所です (ルートキーを除く)。これは **Software¥Microsoft¥Windows¥CurrentVersion¥SharedDLLs¥** です。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。定義済みの定数の値を変更することはできません。

REGDB_KEYPATH_UNINSTALL

REGDB_KEYPATH_UNINSTALL は定義済みの定数で、その値はアプリケーションの一般アンインストールキーのレジストリの場所です (ルートキーを除く)。これは **Software¥Microsoft¥Windows¥CurrentVersion¥Uninstall¥** です。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。定義済みの定数の値を変更することはできません。

REGDB_KEYPATH_WINCURRVER

REGDB_KEYPATH_WINCURRVER は定義済みの定数で、その値は Windows NT の現在のバージョンキーのレジストリの場所です (ルートキーを除く)。これは **Software¥Microsoft¥Windows¥CurrentVersion¥** です。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。定義済みの定数の値を変更することはできません。

REGDB_KEYPATH_WINCURRVER_AUTO

このシステム変数の値は Windows 95、Windows 98、および Windows Me では REGDB_KEYPATH_WINCURRVER であり、Windows NT、Windows 2000、および Windows XP 以降では REGDB_KEYPATH_WINNTCURRVER です。

REGDB_KEYPATH_WINNTCURRVER

REGDB_KEYPATH_WINNTCURRVER は定義済みの定数で、その値は Windows NT の現在のバージョンキーのレジストリの場所です (ルートキーを除く)。これは **Software¥Microsoft¥Windows NT¥CurrentVersion¥** です。この定数を使って、レジストリ関連の一般関数を呼び出すときにキーを指定することができます。定義済みの定数の値を変更することはできません。

REGDB_KEYS

REGDB_KEYS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBQueryKey

REGDB_NAMES

REGDB_NAMES は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBQueryKey

REGDB_NUMBER

REGDB_NUMBER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBGetAppInfo
- RegDBGetKeyValueEx
- RegDBSetKeyValueEx
- RegDBSetAppInfo

REGDB_STRING

REGDB_STRING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBGetAppInfo
- RegDBGetKeyValueEx
- RegDBSetKeyValueEx
- RegDBSetAppInfo

REGDB_STRING_EXPAND

REGDB_STRING_EXPAND は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBSetAppInfo
- RegDBGetKeyValueEx
- RegDBSetKeyValueEx

- RegDBGetAppInfo

REGDB_STRING_MULTI

REGDB_STRING_MULTI は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBGetAppInfo
- RegDBGetKeyValueEx
- RegDBSetKeyValueEx
- RegDBSetAppInfo

REGDB_UNINSTALL_COMMENTS

REGDB_UNINSTALL_COMMENTS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_CONTACT

REGDB_UNINSTALL_CONTACT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_DISPLAYICON

REGDB_UNINSTALL_DISPLAYICON は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem

- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_DISPLAY_VERSION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

REGDB_UNINSTALL_DISPLAY_VERSION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_HELPLINK

REGDB_UNINSTALL_HELPLINK は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_HELPTELEPHONE

REGDB_UNINSTALL_HELPTELEPHONE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_INSTALLDATE

REGDB_UNINSTALL_INSTALLDATE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_INSTALLLOC

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

REGDB_UNINSTALL_INSTALLLOC は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_INSTALLSOURCE

REGDB_UNINSTALL_INSTALLSOURCE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_LANGUAGE

REGDB_UNINSTALL_LANGUAGE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_LOGFILE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

REGDB_UNINSTALL_LOGFILE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_MAINT_OPTION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

REGDB_UNINSTALL_MAINT_OPTION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_MAJOR_VERSION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

REGDB_UNINSTALL_MAJOR_VERSION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_MAJOR_VERSION_OLD

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

REGDB_UNINSTALL_MAJOR_VERSION_OLD は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_MINOR_VERSION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

REGDB_UNINSTALL_MINOR_VERSION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_MINOR_VERSION_OLD

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

REGDB_UNINSTALL_MINOR_VERSION_OLD は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_MODIFYPATH

REGDB_UNINSTALL_MODIFYPATH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_NAME

REGDB_UNINSTALL_NAME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_NOMODIFY

REGDB_UNINSTALL_NOMODIFY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_NOREMOVE

REGDB_UNINSTALL_NOREMOVE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_NOREPAIR

REGDB_UNINSTALL_NOREPAIR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_PRODUCTGUID

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

REGDB_UNINSTALL_PRODUCTGUID は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_PRODUCTID

REGDB_UNINSTALL_PRODUCTID は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_PUBLISHER

REGDB_UNINSTALL_PUBLISHER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem

- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_README

REGDB_UNINSTALL_README は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_REGCOMPANY

REGDB_UNINSTALL_REGCOMPANY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_REGOWNER

REGDB_UNINSTALL_REGOWNER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_STRING

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

REGDB_UNINSTALL_STRING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_SYSTEMCOMPONENT

REGDB_UNINSTALL_SYSTEMCOMPONENT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_URLINFOABOUT

REGDB_UNINSTALL_URLINFOABOUT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_URLUPDATEINFO

REGDB_UNINSTALL_URLUPDATEINFO は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_UNINSTALL_VERSION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

REGDB_UNINSTALL_VERSION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBDeleteItem
- RegDBGetItem
- RegDBSetItem

REGDB_VALUENAME_APPPATH

REGDB_VALUENAME_APPPATH は定義済みの定数で、その値はアプリケーションパスキーの下にあるパス値名、つまり、*Path* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_APPPATHDEFAULT

REGDB_VALUENAME_APPPATHDEFAULT は定義済みの定数で、その値はアプリケーションパスキーの下にあるデフォルト値名、ヌル文字列 (“”) です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_INSTALL

REGDB_VALUENAME_INSTALL は、値が *Install* の定義済み定数です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_INSTALLSUCCESS

REGDB_VALUENAME_INSTALLSUCCESS は、値が *InstallSuccess* の定義済み定数です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_SP

REGDB_VALUENAME_INSTALL は、値が *SP* の定義済み定数です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_COMMENTS

REGDB_VALUENAME_UNINSTALL_COMMENTS は定義済みの定数で、その値はアプリケーション アンインストール キーの下にあるコメント値名 *Comments* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_CONTACT

REGDB_VALUENAME_UNINSTALL_CONTACT は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある連絡先の値名 *Contact* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_DISPLAYICON

REGDB_VALUENAME_UNINSTALL_DISPLAYICON は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある表示アイコン値名 *DisplayIcon* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_DISPLAYNAME

REGDB_VALUENAME_UNINSTALL_DISPLAYNAME は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある表示名の値名 *DisplayName* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_DISPLAYVERSION

REGDB_VALUENAME_UNINSTALL_DISPLAYVERSION は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある表示バージョン値名 *DisplayVersion* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_HELPLINK

REGDB_VALUENAME_UNINSTALL_HELPLINK は定義済みの定数で、その値はアプリケーション アンインストール キーの下にあるヘルプ リンク値名 *HelpLink* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_HELPTELEPHONE

REGDB_VALUENAME_UNINSTALL_HELPTELEPHONE は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある電話番号値名 *HelpTelephone* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_INSTALLDATE

REGDB_VALUENAME_UNINSTALL_INSTALLDATE は定義済みの定数で、その値はアプリケーション アンインストール キーの下にあるインストール日の値名 *InstallDate* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_INSTALLLOCATION

REGDB_VALUENAME_UNINSTALL_INSTALLLOCATION は定義済みの定数で、その値はアプリケーション アンインストール キーの下にあるインストール場所の値名 *InstallLocation* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_INSTALLSOURCE

REGDB_VALUENAME_UNINSTALL_INSTALLSOURCE は定義済みの定数で、その値はアプリケーション アンインストール キーの下にあるインストール ソースの値名 *InstallSource* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_LANGUAGE

REGDB_VALUENAME_UNINSTALL_LANGUAGE は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある言語の値名 *Language* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_LOGFILE

REGDB_VALUENAME_UNINSTALL_LOGFILE は定義済みの定数で、その値はアプリケーション アンインストール キーの下にあるログ ファイル値名 *LogFile* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_LOGMODE

REGDB_VALUENAME_UNINSTALL_LOGMODE は定義済みの定数で、その値はアプリケーション アンインストール キーの下にあるログ モード値名 *LogMode* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_MAJORVERSION

REGDB_VALUENAME_UNINSTALL_MAJORVERSION は定義済みの定数で、その値はアプリケーション アンインストール キーの下にあるメジャー バージョン値名 *VersionMajor* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_MAJORVERSION_OLD

REGDB_VALUENAME_UNINSTALL_MAJORVERSION_OLD は定義済みの定数で、その値はアプリケーション アンインストール キーの下にあるメジャー バージョン値名 *MajorVersion* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_MINORVERSION

REGDB_VALUENAME_UNINSTALL_MINORVERSION は定義済みの定数で、その値はアプリケーション アンインストール キーの下にあるマイナー バージョン値名 *VersionMinor* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_MINORVERSION_OLD

REGDB_VALUENAME_UNINSTALL_MINORVERSION_OLD は定義済みの定数で、その値はアプリケーション アンインストール キーの下にあるマイナー バージョン値名 *MinorVersion* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_MODIFYPATH

REGDB_VALUENAME_UNINSTALL_MODIFYPATH は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある変更パス値名 *ModifyPath* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_NOMODIFY

REGDB_VALUENAME_UNINSTALL_NOMODIFY は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある変更なし値名 *NoModify* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_NOREMOVE

REGDB_VALUENAME_UNINSTALL_NOREMOVE は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある削除なし値名 *NoRemove* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_NOREPAIR

REGDB_VALUENAME_UNINSTALL_NOREPAIR は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある修復なし値名 *NoRepair* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_PRODUCTGUID

REGDB_VALUENAME_UNINSTALL_PRODUCTGUID は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある製品 GUID 値名 *ProductGuid* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_PRODUCTID

REGDB_VALUENAME_UNINSTALL_PRODUCTID は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある製品 ID 値名 *ProductId* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_PUBLISHER

REGDB_VALUENAME_UNINSTALL_PUBLISHER は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある発行者値名 *Publisher* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_README

REGDB_VALUENAME_UNINSTALL_README は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある Readme 値名 *Readme* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_REGCOMPANY

REGDB_VALUENAME_UNINSTALL_REGCOMPANY は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある会社の値名 *RegCompany* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_REGOWNER

REGDB_VALUENAME_UNINSTALL_REGOWNER は定義済みの定数で、その値はアプリケーション アンインストール キーの下にある所有者の値名 *RegOwner* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_SYSTEMCOMPONENT

REGDB_VALUENAME_UNINSTALL_SYSTEMCOMPONENT は定義済みの定数で、その値はアプリケーション アンインストール キーの下にあるシステム コンポーネント値名 *SystemComponent* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_UNINSTALLSTRING

REGDB_VALUENAME_UNINSTALL_UNINSTALLSTRING は定義済みの定数で、その値はアプリケーションアンインストールキーの下にあるアンインストール文字列値名 *UninstallString* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_URLINFOABOUT

REGDB_VALUENAME_UNINSTALL_URLINFOABOUT は定義済みの定数で、その値はアプリケーションアンインストールキーの下にある URL 情報の値名 *URLInfoAbout* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_URLUPDATEINFO

REGDB_VALUENAME_UNINSTALL_URLUPDATEINFO は定義済みの定数で、その値はアプリケーションアンインストールキーの下にある URL 日付情報の値名 *URLDateInfo* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALL_VERSION

REGDB_VALUENAME_UNINSTALL_VERSION は定義済みの定数で、その値はアプリケーションアンインストールキーの下にあるアプリケーション バージョン値名 *Version* です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_UNINSTALLKEY

REGDB_VALUENAME_UNINSTALLKEY は、値が UninstallKey として定義済みの定数です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_WINCURRVER_REGORGANIZATION

REGDB_VALUENAME_WINCURRVER_REGORGANIZATION は定義済みの定数で、その値は Windows (Windows 95、Windows 98、および Windows Me) の登録済み組織値名、または Windows NT (Windows 95、Windows 98、および Windows Me) の現在のバージョンキー "RegisteredOrganization" です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_VALUENAME_WINCURRVER_REGOWNER

REGDB_VALUENAME_WINCURRVER_REGOWNER は定義済みの定数で、その値は Windows (Windows 95、Windows 98、および Windows Me) 登録済みのオーナー値名、または Windows NT (Windows 95、Windows 98、および Windows Me) の現在のバージョンキー "RegisteredOwner" です。この定数を使って、レジストリ関連の一般関数を呼び出すときに値名を指定することができます。定義済みの定数の値を変更することはできません。

REGDB_WINCURRVER_REGORGANIZATION

REGDB_WINCURRVER_REGORGANIZATION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBGetItem

REGDB_WINCURRVER_REGOWNER

REGDB_WINCURRVER_REGOWNER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegDBGetItem

REGFONT_OPTION_DEFAULT

REGFONT_OPTION_DEFAULT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegisterFontResource

REGFONT_OPTION_DONTBROADCASTFONTCHANGEMSG

REGFONT_OPTION_DONTBROADCASTFONTCHANGEMSG は定義済みの定数で、1 つまたは複数のビルトイン関数との間でやり取りされる値を表すのに利用します。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegisterFontResource

REGFONT_OPTION_DONTUPDATEREGISTRY

REGFONT_OPTION_DONTUPDATEREGISTRY は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RegisterFontResource

REGISTRYFUNCTIONS_USETEXTSUBS

REGISTRYFUNCTIONS_USETEXTSUBS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Disable
- Enable

REMOTE_DRIVE

REMOTE_DRIVE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetValidDrivesList

REMOVE

REMOVE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceBitmap

REMOVEABLE_DRIVE

REMOVEABLE_DRIVE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetValidDrivesList

REMOVEALL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

REMOVEALL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SdWelcomeMaint

REPAIR

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

REPAIR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SdWelcomeMaint

REPLACE

REPLACE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ConfigAdd
- AddFolderIcon
- FileInsertLine
- BatchAdd
- ReplaceFolderIcon

RESET

RESET は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FindFile

RESTART

RESTART は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FileGrep
- PathFind
- ConfigFind
- BatchFind

ROOT

ROOT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- DeleteDir

RUN_MAXIMIZED

RUN_MAXIMIZED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- AddFolderIcon
- ReplaceFolderIcon

RUN_MINIMIZED

RUN_MINIMIZED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- QueryProgItem
- AddFolderIcon
- ReplaceFolderIcon

SELECTFOLDER

SELECTFOLDER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SelectFolder

SELFREGISTER

SELFREGISTER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerUpdateFile
- XCopyFile

SELFREGISTERBATCH

SELFREGISTERBATCH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Enable
- Disable

SELFREGISTRATIONPROCESS

SELFREGISTRATIONPROCESS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Do

SERIAL

SERIAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

SERVICE_ADAPTER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_ADAPTER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_ALL_ACCESS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_ALL_ACCESS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_AUTO_START

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_AUTO_START は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_BOOT_START

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_BOOT_START は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_CHANGE_CONFIG

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_CHANGE_CONFIG は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_CONTINUE_PENDING

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_CONTINUE_PENDING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_STATUS

SERVICE_DEMAND_START

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_DEMAND_START は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_DISABLED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_DISABLED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_ENUMERATE_DEPENDENTS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_ENUMERATE_DEPENDENTS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_ERROR_CRITICAL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_ERROR_CRITICAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_ERROR_IGNORE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_ERROR_IGNORE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_ERROR_NORMAL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_ERROR_NORMAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_ERROR_SEVERE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_ERROR_SEVERE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_FILE_SYSTEM_DRIVER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_FILE_SYSTEM_DRIVER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_FLAG_DIFX_32

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript オブジェクト*

SERVICE_FLAG_DIFX_32 は、システム変数 ENABLED_ISERVICES でビット フラグとして設定することができる値を表すために使用される定義済みの定数です。定義済みの定数の値を変更することはできません。

SERVICE_FLAG_DIFX_AMD64

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript オブジェクト*

SERVICE_FLAG_DIFX_AMD64 は、システム変数 ENABLED_ISERVICES でビット フラグとして設定することができる値を表すために使用される定義済みの定数です。定義済みの定数の値を変更することはできません。

SERVICE_FLAG_DIFX_IA64

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript オブジェクト*

SERVICE_FLAG_DIFX_IA64 は、システム変数 ENABLED_ISERVICES でビット フラグとして設定することができる値を表すために使用される定義済みの定数です。定義済みの定数の値を変更することはできません。

SERVICE_FLAG_ISFONTREG

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript オブジェクト*

SERVICE_FLAG_ISFONTREG は、システム変数 ENABLED_ISERVICES でビット フラグとして設定することができる値を表すために使用される定義済みの定数です。式 ENABLED_ISERVICES & SERVICE_FLAG_ISFONTREG がゼロ以外の値に等しい場合、グローバルフォント登録が現在有効になっています。定義済みの定数の値を変更することはできません。

SERVICE_INTERACTIVE_PROCESS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_INTERACTIVE_PROCESS は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_INTERROGATE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_INTERROGATE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_ISFONTREG

SERVICE_ISFONTREG は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Disable

SERVICE_ISUPDATE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この定数は現在使用されていません。FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

SERVICE_KERNEL_DRIVER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_KERNEL_DRIVER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_PAUSED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_PAUSED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_STATUS

SERVICE_PAUSE_CONTINUE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_PAUSE_CONTINUE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_PAUSE_PENDING

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_PAUSE_PENDING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_STATUS

SERVICE_QUERY_CONFIG

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_QUERY_CONFIG は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_QUERY_STATUS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_QUERY_STATUS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_RECOGNIZER_DRIVER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_RECOGNIZER_DRIVER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_RUNNING

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_RUNNING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_STATUS

SERVICE_START

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_START は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_START_PENDING

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_START_PENDING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_STATUS

SERVICE_STOP

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_STOP は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_STOPPED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_STOPPED は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_STATUS

SERVICE_STOP_PENDING

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_STOP_PENDING は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_STATUS

SERVICE_SYSTEM_START

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_SYSTEM_START は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_USER_DEFINED_CONTROL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_USER_DEFINED_CONTROL は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_WIN32_OWN_PROCESS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_WIN32_OWN_PROCESS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SERVICE_WIN32_SHARE_PROCESS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SERVICE_WIN32_SHARE_PROCESS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS

SETUPTYPE

SETUPTYPE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetupType

SETUPTYPE_INFO_DESCRIPTION

SETUPTYPE_INFO_DESCRIPTION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureSetupTypeGetData

SETUPTYPE_INFO_DISPLAYNAME

SETUPTYPE_INFO_DISPLAYNAME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureSetupTypeGetData

SETUPTYPE_STR_COMPACT

SETUPTYPE_STR_COMPACT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureStandardSetupTypeSet

SETUPTYPE_STR_COMPLETE

SETUPTYPE_STR_COMPLETE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureStandardSetupTypeSet

SETUPTYPE_STR_CUSTOM

SETUPTYPE_STR_CUSTOM は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureStandardSetupTypeSet

SETUPTYPE_STR_TYPICAL

SETUPTYPE_STR_TYPICAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- FeatureStandardSetupTypeSet

SETUP_PACKAGE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SETUP_PACKAG は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

SEVERE

SEVERE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MessageBox
- SprintfBox

SHAREDFILE

SHAREDFILE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- XCopyFile
- VerUpdateFile
- DeinstallStart
- InstallationInfo
- System

SILENTMODE

SILENTMODE は、定義済みの定数で、セットアプがサイレント モードで実行されているかどうかをテストするのに利用することができます。詳細については、InstallShield システム変数 MODE を参照してください。

SKIN_LOADED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SKIN_LOADED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

SQL_BATCH_INSTALL

SQL_BATCH_INSTALL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SQLRTGetBatchList

SQL_BATCH_UNINSTALL

SQL_BATCH_UNINSTALL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SQLRTGetBatchList

SQL_BROWSE_ALIAS

SQL_BROWSE_ALIAS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SQLRTGetBrowseOption
- SQLRTSetBrowseOption

SQL_BROWSE_ALL

SQL_BROWSE_ALL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SQLRTGetBrowseOption
- SQLRTSetBrowseOption

SQL_BROWSE_LOCAL

SQL_BROWSE_LOCAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SQLRTGetBrowseOption
- SQLRTSetBrowseOption

SQL_BROWSE_REMOTE

SQL_BROWSE_REMOTE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SQLRTGetBrowseOption
- SQLRTSetBrowseOption

SQL_ERROR_GET_SCHEMA_VERSION

SQL_ERROR_GET_SCHEMA_VERSION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SQLRTGetComponentScriptError
- SQLRTGetComponentScriptError2

SQL_ERROR_SCRIPT_COMMAND_ERROR

SQL_ERROR_SCRIPT_COMMAND_ERROR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SQLRTGetComponentScriptError
- SQLRTGetComponentScriptError2

SQL_ERROR_SCRIPT_CONNECTION_NOT_OPEN

SQL_ERROR_SCRIPT_CONNECTION_NOT_OPEN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SQLRTGetComponentScriptError
- SQLRTGetComponentScriptError2

SQL_ERROR_SCRIPT_UNABLE_OPEN_FILE

SQL_ERROR_SCRIPT_UNABLE_OPEN_FILE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SQLRTGetComponentScriptError
- SQLRTGetComponentScriptError2

SQL_ERROR_SET_SCHEMA_VERSION

SQL_ERROR_SET_SCHEMA_VERSION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SQLRTGetComponentScriptError
- SQLRTGetComponentScriptError2

SRCINSTALLDIR

メモ・SRCTARGETDIR は、SRCINSTALLDIR を置換します。

SRCINSTALLDIR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VarRestore
- VarSave

SRCTARGETDIR

メモ・SRCTARGETDIR は、SRCINSTALLDIR を置換します。

SRCTARGETDIR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VarRestore
- VarSave

SSP_PROPERTY_NO_NEW_INSTALL_HIGHLIGHT

SSP_PROPERTY_NO_NEW_INSTALL_HIGHLIGHT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetShortcutProperty

SSP_PROPERTY_NO_STARTSCREEN_PIN

SSP_PROPERTY_NO_STARTSCREEN_PIN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetShortcutProperty

SSP_PROPERTY_PREVENT_PINNING

SSP_PROPERTY_PREVENT_PINNING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetShortcutProperty

STANDARD_RIGHTS_ALL

STANDARD_RIGHTS_ALL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

STANDARD_RIGHTS_EXECUTE

STANDARD_RIGHTS_EXECUTE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

STANDARD_RIGHTS_READ

STANDARD_RIGHTS_READ は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

STANDARD_RIGHTS_REQUIRED

STANDARD_RIGHTS_REQUIRED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS
- SetObjectPermissions

STANDARD_RIGHTS_WRITE

STANDARD_RIGHTS_WRITE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetObjectPermissions

STATUS

STATUS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceWindow
- Enable
- Disable

STATUSBAR

STATUSBAR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor

STATUSBBRD

STATUSBBRD は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Enable
- Disable

STATUSDLG

STATUSDLG は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceWindow
- Enable
- Disable

STATUSEX

STATUSEX は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceWindow
- Enable
- Disable

STATUSOLD

STATUSOLD は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceWindow
- Enable
- Disable

STRINGLIST

STRINGLIST は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ListCreate

STYLE_BOLD

STYLE_BOLD は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetFont
- SetFont

STYLE_ITALIC

STYLE_ITALIC は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetFont
- SetFont

STYLE_NORMAL

STYLE_NORMAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetFont
- SetFont

STYLE_SHADOW

STYLE_SHADOW は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetFont

STYLE_UNDERLINE

STYLE_UNDERLINE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetFont
- SetFont

SW_MAXIMIZE

SW_MAXIMIZE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ShowProgramFolder

SW_MINIMIZE

SW_MINIMIZE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ShowProgramFolder

SW_RESTORE

SW_RESTORE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ShowProgramFolder

SW_SHOW

SW_SHOW は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ShowProgramFolder

SYNCHRONIZE

SYNCHRONIZE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS
- SetObjectPermissions

SYS_BOOTMACHINE

SYS_BOOTMACHINE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- System
- RebootDialog
- SdFinishReboot

SYSTEM_DPI

SYSTEM_DPI は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

SYSTEM_DPI_SCALING

SYSTEM_DPI_SCALING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

TBYTES

TBYTES は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- ConvertSizeToUnits
- StrConvertSizeUnit

TILED

TILED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceBitmap

TIME

TIME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

TRUE

TRUE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- AskOptions
- CtrlSetMultCurSel
- DialogSetInfo
- FeatureAddItem
- FeatureGetData
- FeatureIsItemSelected
- FeatureSelectItem
- FeatureTotalSize
- LongPathToQuote
- SdDiskSpace2
- SelectDir
- SdShowMsg
- SQLDatabaseBrowse
- SQLRTConnect
- SQLRTConnect2
- SQLRTConnectDB
- SQLRTGetDatabases
- SQLRTGetServers
- SQLRTGetServers2
- SQLRTPutConnectionAuthentication
- SQLRTTestConnection
- SQLRTTestConnection2
- SQLServerSelectLogin
- SQLServerSelectLogin2

TTFONTFILEINFO_FONTTITLE

TTFONTFILEINFO_FONTTITLE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetTrueTypeFontFileInfo

TYPICAL

TYPICAL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SdSetupType
- SetupType

UPDATE_SERVICE_INSTALL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この定数は現在使用されていません。FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

UPDATESERVICECOMPONENT

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この定数は現在使用されていません。FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

UPPER_LEFT

UPPER_LEFT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceBitmap
- PlaceWindow

UPPER_RIGHT

UPPER_RIGHT は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- PlaceBitmap
- PlaceWindow

URL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

URL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

USER_ADMINISTRATOR

USER_ADMINISTRATOR は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

USER_INADMINGROUP

USER_INADMINGROUP は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

USER_POWERUSER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

USER_POWERUSER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

USE_LOADED_SKIN

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

USE_LOADED_SKIN は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Disable
- Enable

VALID_PATH

VALID_PATH は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

VERSION_COMPARE_RESULT_NEWER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

VERSION_COMPARE_RESULT_NEWER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerProductCompareVersions

VERSION_COMPARE_RESULT_NEWER_NOT_SUPPORTED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

VERSION_COMPARE_RESULT_NEWER_NOT_SUPPORTED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerProductCompareVersions

VERSION_COMPARE_RESULT_NOT_INSTALLED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

VERSION_COMPARE_RESULT_NOT_INSTALLED は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerProductCompareVersions

VERSION_COMPARE_RESULT_OLDER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

VERSION_COMPARE_RESULT_OLDER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerProductCompareVersions

VERSION_COMPARE_RESULT_SAME

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

VERSION_COMPARE_RESULT_SAME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerProductCompareVersions

VERSION_PREVIOUS_VERSION_DELIMITER

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

VERSION_PREVIOUS_VERSION_DELIMITER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerProductCompareVersions

VER_DLL_NOT_FOUND

VER_DLL_NOT_FOUND は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerSearchAndUpdateFile

VER_UPDATE_ALWAYS

VER_UPDATE_ALWAYS は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerUpdateFile
- VerSearchAndUpdateFile

VER_UPDATE_COND

VER_UPDATE_COND は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- VerSearchAndUpdateFile

VIDEO

VIDEO は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

VIRTUAL_MACHINE_TYPE

VIRTUAL_MACHINE_TYPE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

VOLUMELABEL

VOLUMELABEL は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

WARNING

WARNING は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- MessageBox
- SprintfBox

WEB_BASED_SETUP

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

WEB_BASED_SETUP は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

WELCOME

WELCOME は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Welcome

WHITE

WHITE は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetTitle

WILL_REBOOT

WILL_REBOOT は定義済みの定数で、1 つまたは複数のビルト イン関数またはイベント ハンドラーによって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- RebootDialog
- SdFinishReboot

WINDOWS_SHARED

WINDOWS_SHARED は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Is

WINMAJOR

WINMAJOR は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

WINMINOR

WINMINOR は定義済みの定数で、1 つまたは複数のビルト イン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- GetSystemInfo

WOW64FSREDIRECTION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

WOW64FSREDIRECTION は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- Disable
- Enable

WRITE_DAC

WRITE_DAC は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS
- SetObjectPermissions

WRITE_OWNER

WRITE_OWNER は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたり、または 1 つまたは複数のシステム変数に割り当てられる値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SERVICE_IS_PARAMS
- SetObjectPermissions

YELLOW

YELLOW は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- SetColor
- SetTitle

YES

YES は定義済みの定数で、1 つまたは複数のビルトイン関数によって渡されたり、返されたりする値を表わすために使用されます。定義済みの定数の値を変更することはできません。

次の関数と共に利用します

- AskYesNo
- SdLicense
- SdConfirmNewDir
- SdConfirmRegistration

_MAX_PATH

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MAX_PATH は、Windows API 関数に渡されるパス変数の最大の長さを表すために使用される定義済み定数です。次のサンプルコード行は、_MAX_PATH の使用例を具体的に説明します :

```
string szPath[_MAX_PATH]; /* 変数宣言 */
    ...
    Kernel32.GetTempPathA( _MAX_PATH, szPath ); /* Windows API 関数呼び出し */
```

定義済みのスクリプト変数

このセクションでは、スクリプトのコンパイル中に予約されている定義済みのスクリプト変数の一覧を示します。

FILE

スクリプトのコンパイル中に、この予約識別子は _FILE_ が属するソースファイルの完全修飾名を含む文字列と置き換えられます。_FILE_ は文字列定数が可能な場所であれば、スクリプトの任意の位置で指定することができますが、簡単なデバッグで _LINE_ と共に利用すると便利です。例えば、下に示したステートメントを構築してテスト中にソースファイル内の有効な場所へコピーすることで、セットアップを実行している最中にセットアップの特定部分をスクリプトの特定セクションへ容易に関連付けることができます。

```
SprintfBox (INFORMATION, "", " ファイル : %s¥n 行 :%ld";,
        _FILE_, _LINE_);
```

パスはセットアップの実行開始位置ではなく、ファイルをコンパイルする場所であることにご注意ください。必要であれば、2 番目のパラメーターで _FILE_ を使って ParsePath 関数を呼び出し、完全修飾ファイル名の正確な部分を呼び出すこともできます。下の部分コードはファイル名を抽出して、それを表示します。

```
ParsePath (svReturnString, _FILE_, FILENAME);
MessageBox (svReturnString, INFORMATION);
```

より完全で強力なデバッグには、InstallShield の [ビルド] メニューから [デバッグ] をクリックして [InstallScript デバッガー] を利用してください。詳細については、InstallScript デバッガーヘルプを参照してください。

LINE

セットアップのコンパイル中、この予約された識別子は _LINE_ があるソースファイル行の番号によって置き換えられます。_LINE_ は数値定数が可能な場所であれば、スクリプトの任意の位置で指定することができますが、簡単なデバッグで _FILE_ と共に利用すると便利です。例えば、下に示したステートメントを構築してテスト中にソースファイル内の有効な場所へコピーすることで、セットアップを実行している最中にセットアップの特定部分をスクリプトの特定セクションへ容易に関連付けることができます。

```
SprintfBox (INFORMATION, "", " ファイル : %s¥n 行 :%ld", _FILE_, _LINE_);
```

より完全で強力なデバッグには、InstallShield の [ビルド] メニューから [デバッグ] をクリックして [InstallScript デバッガー] を利用してください。詳細については、InstallScript デバッガーヘルプを参照してください。

BASICMSI

BASICMSI スクリプト変数は基本の MSI プロジェクトで定義されますが、InstallScript MSI プロジェクトと InstallScript プロジェクトでは定義されていないため、ゼロと評価されます。

メモ・BASICMSI はプリプロセッサ スイッチではありません。そのためこのスクリプト変数を使用して、再コンパイルせずに別のプロジェクト タイプで別の動作をするスクリプト コードを作成することができます。

BASICMSI を使って、異なるプロジェクトの種類で別々の動作を行う単一のスクリプトを書くことができます。そのためには次のよなコードをスクリプトに含みます。

```
if( BASICMSI ) then
  // 基本の MSI プロジェクトのコード
else
  //InstallScript MSI プロジェクトまたは InstallScript プロジェクトのコード
endif;
```

INSTALLSCRIPTMSI

INSTALLSCRIPTMSI は InstallScript MSI や基本の MSI プロジェクトでは定義されていますが、InstallScript プロジェクトでは未定義となり、ゼロ評価します。

メモ・INSTALLSCRIPTMSI はプリプロセッサ スイッチではありません。そのためこのスクリプト変数を使用して、再コンパイルせずに 2 つのプロジェクト タイプで別の動作をするスクリプト コードを作成することができます。

INSTALLSCRIPTMSI を使って、異なるプロジェクトの種類で別々の動作を行う単一のスクリプトを書くことができます。そのためには次のようなコードをスクリプトに含みます。

```
if( INSTALLSCRIPTMSI ) then
  //InstallScript MSI プロジェクトまたは基本の MSI プロジェクトのコード
else
  //InstallScript プロジェクトのコード ...
endif;
```

INSTALLSCRIPTMSIEEUI

INSTALLSCRIPTMSIEEUI 変数は、InstallScript MSI インストールで InstallScript エンジンが埋め込みユーザー インターフェイス (UI) として使用されるかどうかをインストールの実行時に判別できるように設定されます。この実装は、新しいスタイルの InstallScript UI とも呼ばれます。

InstallScript MSI インストールで InstallScript エンジンが埋め込みユーザー インターフェイス (UI) として使用される場合、INSTALLSCRIPTMSIEEUI は TRUE に設定されます。埋め込みユーザー インターフェイス ハンドラーが使用されない場合、この変数は FALSE に設定されます。

ヒント・InstallScript MSI インストールで InstallScript エンジンを埋め込みユーザー インターフェイス (UI) として使用するときの情報については、「InstallScript MSI インストールで InstallScript エンジンを外部または埋め込み UI ハンドラーとして使用する」を参照してください。

INSTALLSCRIPTMSIEEUI を使って、異なるユーザー インターフェイスのスタイルで別々の動作を行う単一スクリプトを作成することができます。そのためには次のようなコードをスクリプトに含みます。

```
if( INSTALLSCRIPTMSIEEUI ) then
  // 新しいスタイルの InstallScript MSI インストールのコード
  //( 埋め込み UI ハンドラーとしての InstallScript エンジン )...
else
  // 従来スタイルの InstallScript MSI インストールのコード
  //( 外部 UI ハンドラーとしての InstallScript エンジン )...
endif;
```

ISUS_PRODUCT_CODE

ISUS_PRODUCT_CODE 変数は、初期化の時に PRODUCT_GUID が設定される読み取り専用のスクリプト変数です。したがって、カスタマイズした場合は、セットアップを実行するたびにこのスクリプト変数をカスタマイズする必要があります。メンテナンス モードの間も同様です。

SERVICE_IS_PARAMS

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

SERVICE_IS_PARAMS 変数は、インストールの初期化中に ServiceInitParams への呼び出しによって自動的に初期化されます。

***メモ・**InstallScript サービス関数は内部的に Windows API 関数 OpenSCManager、CreateService、または ChangeServiceConfig を呼び出します。次の構造化された変数 SERVICE_IS_PARAMS のメンバーは、これらの Windows API 関数に対応する引数を指定します。*

- *SERVICE_IS_PARAMS.lpMachineName*
- *SERVICE_IS_PARAMS.lpDatabaseName*
- *SERVICE_IS_PARAMS.dwDesiredAccess*
- *SERVICE_IS_PARAMS.dwServiceType*
- *SERVICE_IS_PARAMS.dwStartType*
- *SERVICE_IS_PARAMS.dwErrorControl*
- *SERVICE_IS_PARAMS.lpLoadOrderGroup*
- *SERVICE_IS_PARAMS.lpdwTagId*
- *SERVICE_IS_PARAMS.lpDependencies*
- *SERVICE_IS_PARAMS.lpServiceStartName*
- *SERVICE_IS_PARAMS.lpPassword*

次の SERVICE_IS_PARAMS のメンバーは、希望の状態に達するサービスを待っている間にインストーラーがどのように動作するかを制御します。詳細については、各メンバーの説明を参照してください。

- *SERVICE_IS_PARAMS.nWaitHintMin*
- *SERVICE_IS_PARAMS.nWaitHintMax*
- *SERVICE_IS_PARAMS.nStartServiceWaitCount*

- *SERVICE_IS_PARAMS.nStopServiceWaitCount*

SERVICE_IS_PARAMS スクリプト変数には次のメンバーがあります。

テーブル 1 - SERVICE_IS_PARMS のパラメーター

メンバー	説明
dwServiceType	このメンバーは、これらの定義済み定数に設定できます。 • SERVICE_WIN32_OWN_PROCESS • SERVICE_WIN32_SHARE_PROCESS • SERVICE_KERNEL_DRIVER • SERVICE_FILE_SYSTEM_DRIVER • SERVICE_ADAPTER • SERVICE_RECOGNIZER_DRIVER • SERVICE_INTERACTIVE_PROCESS
dwStartType	このメンバーは、これらの定義済み定数に設定できます。 • SERVICE_BOOT_START • SERVICE_SYSTEM_START • SERVICE_AUTO_START • SERVICE_DEMAND_START • SERVICE_DISABLED
dwErrorControl	このメンバーは、これらの定義済み定数に設定できます。 • SERVICE_ERROR_IGNORE • SERVICE_ERROR_NORMAL • SERVICE_ERROR_SEVERE • SERVICE_ERROR_CRITICAL
nWaitHintMin	最小 dwWaitHint 待機時間をミリ秒で指定します。サービスが dwWaitHint を nWaitHintMin より小さく指定すると、nWaitHintMin が代わりに待機時間として使用されます。これは、サービスの開始と停止の両方に適用されます。 このメンバー変数のデフォルト値は 1000 (1 秒) で、ServiceInitParams を呼び出すことによって設定されます。サービスがどのように dwWaitHint を設定するかは、MDSN のマニュアルを参照してください。

テーブル 1 ・ SERVICE_IS_PARMS のパラメーター (続き)

メンバー	説明
nWaitHintMax	最大 dwWaitHint 待機時間をミリ秒で指定します。サービスが dwWaitHint を nWaitHintMax より長く指定すると、nWaitHintMax が代わりに待機時間として使用されます。これは、サービスの開始と停止の両方に適用されます。 このメンバー変数のデフォルト値は 10000 (10 秒) で、ServiceInitParams を呼び出すことによって設定されます。サービスがどのように dwWaitHint を設定するかは、MDSN のマニュアルを参照してください。
nStartServiceWaitCount	サービスがタイムアウトを開始する時間を秒で指定します。この値を、サービスが意図した状態に達した、しないに関わらず、これらの値を特定の値に変更して、一定の間隔後強制的にインストーラーを停止するように設定することができます。 ***重要 ・*** *nWaitHintMax と異なり、サービスが長い dwWaitHint を指定した場合、インストーラーはこのパラメーターの値に関係なくこの待機を中断しません。したがって、この値を ServiceInitParams で設定された INFINITE のデフォルト値から変更されないことをお勧めします。代わりに nWaitHintMax を更新して、不必要な待機を防ぐようにします。*
nStopServiceWaitCount	サービスがタイムアウトを停止する時間を秒で指定します。この値を、サービスが意図した状態に達した、しないに関わらず、これらの値を特定の値に変更して、一定の間隔後強制的にセットアップを停止するように設定することができます。 ***重要 ・*** *nWaitHintMax と異なり、サービスが長い dwWaitHint を指定した場合、インストーラーはこのパラメーターの値に関係なくこの待機を中断しません。したがって、この値を ServiceInitParams で設定された INFINITE のデフォルト値から変更されないことをお勧めします。代わりに nWaitHintMax を更新して、不必要な待機を防ぐようにします。*

追加情報

Windows API 関数の OpenSCManager、CreateService、ChangeServiceConfig についての詳細は、Windows API マニュアルを参照してください。

SERVICE_IS_STATUS

プロジェクト ・ *この情報は、InstallScript プロジェクトに適用します。*

ServiceGetServiceState を呼び出したとき、この構造化された変数はサービスについての ID 情報を戻します。このシステム変数は SERVICE_IS_STATUS タイプで、以下のメンバーを含みます :

テーブル 2 · SERVICE_IS_STATUS のパラメーター

メンバー	意味
dwServiceType	サービスの種類。このメンバーには次の値のひとつが可能です。 · SERVICE_FILE_SYSTEM_DRIVER Ñ サービスはファイル システム ドライバーです。 · SERVICE_KERNEL_DRIVER Ñ サービスはデバイス ドライバーです。 · SERVICE_WIN32_OWN_PROCESS Ñ サービスは独自のプロセスで実行します。 · SERVICE_WIN32_SHARE_PROCESS Ñ サービスは他のサービスとプロセスを共有します。 サービスタイプが SERVICE_WIN32_OWN_PROCESS または SERVICE_WIN32_SHARE_PROCESS のどちらかの場合、次の種類も指定されます。 · SERVICE_INTERACTIVE_PROCESS– サービスはデスクトップと対話できます。
dwCurrentState	サービスの現在の状態。このメンバーには次の値のひとつが可能です。 · SERVICE_CONTINUE_PENDING Ñ サービス継続は保留です。 · SERVICE_PAUSE_PENDING Ñ サービス一時停止は保留です。 · SERVICE_PAUSED Ñ サービスは一時停止されています。 · SERVICE_RUNNING Ñ サービスは実行中です。 · SERVICE_START_PENDING Ñ サービスが開始します。 · SERVICE_STOP_PENDING Ñ サービスは停止中です。 · SERVICE_STOPPED Ñ サービスは実行していません。
dwWin32ExitCode	サービスが開始または停止したときに発生するエラーをレポートする為に利用する Win32 エラーコードです。サービス特有のエラーコードを戻すためには、サービスがこの値を ERROR_SERVICE_SPECIFIC_ERROR へ設定して dwServiceSpecificExitCode メンバーがエラーコードを含むことを示さなくてはなりません。サービスが実行中で、通常終了した場合はこの値を NO_ERROR に設定します。
dwServiceSpecificExitCode	サービスが開始または停止された時にエラーが発生した場合に戻す、サービス特有のエラーコード。この値は dwWin32ExitCode メンバーが ERROR_SERVICE_SPECIFIC_ERROR に設定されていない限り無視されます。

テーブル 2 - SERVICE_IS_STATUS のパラメーター (続き)

メンバー	意味
dwCheckPoint	サービスが時間のかかる開始、停止、ポーズ、または続行処理に間にその進行状況をレポートするため、定期的に増加させる値。例えば、スタートアップの際に初期化の各段階を完了するごとにサービスはこの値を増加させます。サービスでの処理を引き起こすユーザーインターフェイスプログラムはこの値を利用して、時間のかかる処理の最中にサービスの進行状況を追跡します。サービスが開始、停止、ポーズ、または続行処理保留を持たない場合、この値は無効およびゼロです。
dwWaitHint	増加された dwCheckPoint 値または dwCurrentState での変更点を使って、サービスが Windows API 関数 SetServiceStatus への次の呼出を行う前に保留開始、停止、ポーズ、または続行処理にかかる予測されるミリ秒単位の時間。dwWaitHint が渡した値と dwCheckPoint が指定した時間が増加されていない場合、または dwCurrentState が変更されなかった場合、サービスコントロールマネージャーまたはサービスコントロールプログラムはエラーが発生したものと見なします。
dwControlsAccepted	コントロールはサービスがそのハンドラー関数で受け取り並びに処理するようコードします。Windows API 関数 ControlServic でコントロールコマンドを指定することで、ユーザーインターフェイスはサービスを制御することができます。デフォルトで、すべてのサービスは SERVICE_CONTROL_INTERROGATE 値を受け付けます。Table 3 に、このメンバーで使用可能な制御コードのリストを示します。

dwControlAccepted の制御コード

このテーブルに、dwControlAccept スクリプト変数で使用可能な制御コードの一覧を示します。

テーブル 3 - dwControlsAccepted の制御コード

制御コード	説明
SERVICE_ACCEPT_NETBINDCHANGE	サービスは、停止や再開することなくバインドに変更を受け付けることができるネットワークコンポーネントです。 この制御コードは、サービスが SERVICE_CONTROL_NETBINDADD、SERVICE_CONTROL_NETBINDREMOVE、SERVICE_CONTROL_NETBINDENABLE、そして SERVICE_CONTROL_NETBINDDISABLE 通知を受け取ることを可能にします。

テーブル 3 · dwControlsAccepted の制御コード（続き）

制御コード	説明
SERVICE_ACCEPT_PARAMCHANGE	サービスは停止または再開することなくスタートアップパラメーターを再び読み込むことが可能です。 この制御コードは、サービスが SERVICE_CONTROL_PARAMCHANGE 通知を受け取ることを可能にします。
SERVICE_ACCEPT_PAUSE_CONTINUE	サービスはポーズや続行することが可能です。 この制御コードは、サービスが SERVICE_CONTROL_PAUSE および SERVICE_CONTROL_CONTINUE 通知を受け取ることを可能にします。
SERVICE_ACCEPT_SHUTDOWN	サービスはシステムがシャットダウンしたときに通知します。 この制御コードは、サービスが SERVICE_CONTROL_SHUTDOWN 通知を受け取ることを可能にします。Windows API 関数 ControlService はこの通知を送ることができないことに注意して下さい。システムのみがこれを送ることができます。
SERVICE_ACCEPT_STOP	サービスは停止することが可能です。 この制御コードは、サービスが SERVICE_CONTROL_STOP 通知を受け取ることを可能にします。

dwControlAccept の値は次の拡張制御コードを含むことも可能です。これは Windows API 関数 **RegisterServiceCtrlHandlerEx** と共に利用できるサービス ハンドラー関数によってのみサポートされています。

テーブル 4 · dwControlAccept の拡張制御コード

制御コード	説明
SERVICE_ACCEPT_HARDWAREPROFILECHANGE	サービスはコンピューターのハードウェア プロファイルが変更された時に通知されます。システムがサービスへ SERVICE_CONTROL_HARDWAREPROFILECHANGE 通知を送ることを可能にします。
SERVICE_ACCEPT_POWEREVENT	サービスはコンピューターのパワー状態が変更された時に通知されます。これはシステムがサービスへ SERVICE_CONTROL_POWEREVENT 通知を送ることを可能にします。
SERVICE_ACCEPT_SESSIONCHANGE	Whistler の場合 : サービスはコンピューターのセッション状態が変更された時に通知されます。これはシステムがサービスへ SERVICE_CONTROL_SESSIONCHANGE 通知を送ることを可能にします。

SUITE_HOSTED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SUITE_HOSTED 変数を使って、InstallScript インストールが、アドバンスト UI またはスイート / アドバンスト UI インストール内の InstallScript パッケージとして実行されているかどうかを判別できます。

InstallScript インストールが、アドバンスト UI またはスイート / アドバンスト UI インストール内の InstallScript パッケージとして実行されている場合、SUITE_HOSTED には、ゼロ以外の値が設定されます。SUITE_HOSTED 変数にゼロが設定された場合、InstallScript インストールは、アドバンスト UI またはスイート / アドバンスト UI インストール内にある InstallScript パッケージとして実行されていないことを意味します。例 :

```
if SUITE_HOSTED then
   // スイート / アドバンスト UI またはアドバンスト UI インストール内の
   // InstallScript パッケージとして実行されている
   // InstallScript インストールのコード
else
   // スイート / アドバンスト UI またはアドバンスト UI インストール内の
   // InstallScript パッケージとしてではなく、スタンドアロンで実行されている
   // InstallScript インストールのコード
endif;
```

データ型および定義済み構造

データ型

InstallScript では、次のデータ型をサポートしています。一部のデータ型は大文字と小文字に関わらず利用できます:

テーブル 1 - データ型

データ型	説明
binary **BINARY**	文字列変数で指定されたバイナリ データを外部 DLL 関数に渡す、または DLL 関数から読み取ることを示します。STRING または WSTRING データ型とは異なり、BINARY データ型が指定されると、InstallScript エンジンはデータを文字列の文字として解釈せず、データ型の変換または検証も行いません。このため、有効な文字列で構成されているかどうかに関わらずにバイナリ データを渡すときに、このデータ型が使用されます。 このデータ型は、外部 DLL 関数のプロトタイプでのみ使用できます。このデータ型を変数インスタンスとして、または非 DLL InstallScript 関数のパラメーターとして使用すると、コンパイル エラー C8116 エラーが発生します。 標準 InstallScript 文字列を BINARY データ型を通して渡すと、文字列内の文字が ASCII 文字として渡されます。つまり、有効な文字列の文字においてバイナリ型は STRING データ型に似ていますが、WSTRING データ型とは異なります。
BOOL	ブール値データ、TRUE (1) または FALSE (0) のどちらか。この種類の変数は、指定値以外の値を格納するために使用できません。C++ と同様に、InstallScript ではゼロ以外の値を TRUE と評価します。値がゼロの場合のみ、FALSE と評価されます。通常は、1 の値で真を示します。
char **CHAR**	文字データ、8 ビットの単一符号付き文字。スクリプトにリテラル文字があると、一重引用符または二重引用符で囲まなければなりません。文字列に、数字の ASCII 値を割り当てることができます。char 変数を文字として表示するには、書式指定子 “%c” を関数 SprintfBox と一緒に使用します。char 変数の数値を表示するには、指定子 “%d” を使用します。 InstallScript の文字変数タイプには符号が付きます。したがって、拡張 ASCII 文字は数値として解析されると負の数になります。この問題を回避するには、数値変数に値を割り当てます。その後、数値を解析する前に数値変数と値 255 を AND (&) で組み合わせます。
HWND	ウィンドウのハンドル。HWND 変数タイプには、Windows で使用可能な他のタイプのハンドルも格納できます。HWND 変数は、通常 CmdGetHwndDlg または GetWindowHandle 関数を使って初期化されます。内部では HWND 変数が、データ型である NUMBER と同一です。
int **INT**	数値型と同じです。便宜上、用意されているだけです。

テーブル 1 - データ型 (続き)

データ型	説明
LIST	InstallScript リストへのポインター。LIST 変数は、常に ListCreate および ListDestroy 関数を使用して初期化および初期化解除されます。内部では LIST 変数が、データ型である NUMBER と同一です。
long LONG	NUMBER 型と同等。便宜上、用意されているだけです。
LPSTR	POINTER 型と同等。便宜上、用意されているだけです。 詳細については、「ポインター」を参照してください。
LPWSTR	WPOINTER 型と同等。便宜上、用意されているだけです。 詳細については、「ポインター」を参照してください。
number NUMBER	署名済み 4 バイト整数。Number は、数値データを格納するのに推奨されるデータ型です。他のプログラミング言語で使用される LONG 変数タイプと類似しています。-2,147,483,648 から +2,147,483,647 までの間の値を格納できます。InstallScript のすべての数値データ型は、NUMBER 変数タイプに相当します。
object OBJECT	COM オブジェクトへのリファレンス。リファレンスは、CreateObject 関数によって戻され、set キーワードを使用してオブジェクト変数に割り当てられます。
pointer POINTER	データへのポインター。文字列変数へのポインターの場合、ポイント先のデータは ANSI 文字列です。ポインター変数は、通常 (&) 演算子のアドレスを使用して初期化され、変数のアドレスをポインターの変数に割り当てます。 詳細については、「ポインター」を参照してください。
short SHORT	NUMBER 型と同等。便宜上、用意されているだけです。

テーブル 1 - データ型 (続き)

データ型	説明
string **STRING**	Unicode 文字の配列 (1 文字につき 2 バイト)。C++ 言語内の文字配列に類似している文字列変数は NULL で終了します。ただし、InstallShield では、同一文字列変数内で複数の NULL で終了する文字列を使用できません。文字列変数は、最高 65,535 までの文字数を明示して宣言できます。サイズを明示しないで宣言された文字列変数は、InstallShield によって自動的にサイズが決定されます。セットアップ内での文字列の連結は、連結演算子であるプラス記号 (+) を使用して実行します。連結する文字列は、以下のステートメントのように、演算子の両側にオペランドとして配置されます。演算子は、szFirstName の値に szLastName の値を追加して、結果として作成される文字列を szFullName に割り当てます。 `szFullName = szFirstName + szLastName;` 文字列変数を表示するには、SprintfBox 関数を書式指定子 "%s" と一緒に使用するか、MessageBox 関数を使用します。 ***メモ -*** *InstallScript コード内で STRING として文字列変数を宣言できます。InstallScript コード内でこの方法で宣言された文字列変数は、文字列テーブルで Unicode 文字列として格納されます。ただし、InstallScript コード外部で渡される構造体で Unicode 文字列を格納する場合 (たとえば DLL 関数)、InstallScript コード内で構造メンバーとして文字列を宣言するときに、WSTRING 型を使用しなくてはならない場合があります。詳細については、「データ構造体」を参照してください。*
variant **VARIANT**	文字、文字列、数値、オブジェクト参照など、任意の種類のデータ。可能な限り他のデータタイプを使用することが推奨されます。VARIANT データタイプは、例の様にスクリプト定義関数の宣言で配列を引数として扱う場合にのみ使用します。 `prototype number AverageValue( variant );` `function OnBegin()` `  number nAverage , nArray(10);` `begin` `  /* ここへ要素を配列 . するための値を割り当てます。*/` `  /* 関数に配列を渡します。*/` `  nAverage = AverageValue( nArray );` `end;` VARIANT データ型をデータ構造内で定義することはできません。

テーブル 1 - データ型 (続き)

データ型	説明
void **VOID**	void 編集を void と言うタイプで宣言できない点で、本当のデータ タイプとは言えません。void は関数プロトタイプでのみ利用され、次に示す例の様に関数が値を戻さないことを示します。 `prototype void Subroutine(int);` `function void Subroutine(int);` `begin` `// 処理を行う、しかし` `// 値を戻さない` `end;`
wpointer **WPOINTER**	文字列データへのポインター。Unicode 文字列データへのポインターが必要なとき以外は、常に POINTER 型を使用する必要があります。ポインター変数は、通常 (&) 演算子のアドレスを使用して初期化され、変数のアドレスをポインターの変数に割り当てます。 詳細については、「ポインター」を参照してください。
wstring **WSTRING**	STRING と同じですが、STRING と違ってワイド文字列引数または Unicode 文字列引数が必要な DLL 関数呼び出しの宣言に使用できる点が異なります。例 : `prototype long Kernel32.GetWindowsDirectoryW(BYREF wstring, int);` 次の例のように、WSTRING 引数で文字列変数を渡すことができます。例 : `wstring svWinDir;` `...` `GetWindowsDirectoryW(svWinDir, 1024);` ***メモ -*** *InstallScript コード内で STRING として文字列変数を宣言できます。InstallScript コード内でこの方法で宣言された文字列変数は、文字列テーブルで Unicode 文字列として格納されます。ただし、InstallScript コード外部で渡される構造体で Unicode 文字列を格納する場合 (たとえば DLL 関数)、InstallScript コード内で構造メンバーとして文字列を宣言するときに、WSTRING 型を使用しなくてはならない場合があります。詳細については、「データ構造体」を参照してください。*

メモ - *InstallScript には、符号なしのデータ型や浮動ポイント データ型はありません。*

定義済み構造

InstallScript では、次の事前定義構造をサポートしています。

テーブル 2 - 定義済み構造

定義済み構造	説明
_FONTFILEINFO	これは、OnInstalledFontFile および OnUninstallingFontFile イベント ハンドラーに渡されるデータ構造です。次のメンバーを持ちます : • string szFileName[_MAX_PATH]— システム上にインストールされるフォント ファイルへの完全パス • string szFaceName[_MAX_PATH]— インストールされるフォントのフェース名 (InstallShield でフォント ファイルの名前を指定する場合) 構造内の情報は、イベント ハンドラーが使用できるようにイベント ハンドラーに渡されます。その後、構造の値がインストールで使用されます。このため、構造メンバーを変更してもインストールには影響しません。
_LAAW_PARAMETERS	このデータ構造のメンバーのリスト、その使用目的、並びに利用可能な値については、LAAW_PARAMETERS を参照してください。
_SERVICE_IS_PARAMS	このデータ構造のメンバーのリスト、その使用目的、並びに利用可能な値については、SERVICE_IS_PARAMS を参照してください。
_SERVICE_IS_STATUS	このデータ構造のメンバーのリスト、その使用目的、並びに利用可能な値については、SERVICE_IS_STATUS を参照してください。
ISURL_COMPONENTS	URL の構成要素となる部分を含むデータ構造です。この構造は、Windows API 構造 URL_COMPONENTS に類似します。この構造の使用例については、ParseUrl を参照してください。この構造には次のメンバーが含まれます。 • szScheme— スキーム名を含む文字列値。 • nInternetScheme— インターネットプロトコルスキームを示す数値。これは HTTPS には 4 と等しく、HTTP では 3 と等しくなります。 • szUserName— ユーザー名を含む文字列値。 • szPassword— パスワードを含む文字列値。 • szHostName— ホスト名を含む文字列値。 • nInternetPort— ポート番号。 • szUrlPath—URL パスを含む文字列値。 • szExtraInfo— その他の情報を含む文字列値 (例えば、?something または #something)。
PROCESS_INFORMATION	このデータ構造のメンバーのリスト、その使用目的、並びに利用可能な値については、LAAW_PROCESS_INFORMATION を参照してください。

テーブル 2 - 定義済み構造 (続き)

定義済み構造	説明
STARTUPINFO	このデータ構造のメンバーのリスト、その使用目的、並びに利用可能な値については、LAAW_STARTUPINFO を参照してください。

Arrays

InstallScript データ型はいずれも配列として宣言して使用できます。変数を配列として宣言するには、宣言の変数名をかっこを括り、かっこの中には配列サイズを指定することができます。次の例は nArray を 10 NUMBER 要素を含む配列として宣言しています。

```
NUMBER nArray(10);
```

配列サイズを宣言しない場合は、次のようになります。

```
NUMBER nArray();
```

配列サイズは 0 にデフォルトの設定されます。スクリプトの配列のサイズを変更するには、Resize 演算子を使用します。配列のサイズを取得するには、SizeOf オペレータを使用します。

次の構文を使って、配列要素に値を割り当てます。

構文

< 配列変数名 >(< 配列インデックス >) = < 値 >;

例 :

```
nArray(0) = 1; /* 配列インデックス作成は 0 で開始します。*/
nArray(5) = 17;
```

配列を引数とするスクリプト定義の関数を宣言するときは、次に示すように、配列をパラメーターデータ型として使用しないでください。

```
prototype NUMBER AverageValue ( NUMBER() ); /* これはコンパイルされません。*/
```

代わりに、以下のように VARIANT データ型を使用します。

```
prototype NUMBER AverageValue( VARIANT );

function OnBegin()
    NUMBER nAverage, nArray(10);
begin
    /* ここへ要素を配列 . するための値を割り当てます。*/

    /* 関数に配列を渡します。*/
    nAverage = AverageValue( nArray );
end;
```

定数データ

定数は、定義された値を持つデータ項目です。InstallShield では 2 つのタイプの定数をサポートします。

- TRUE や RESET などの 事前定義定数 は InstallScript の一部です。これらの定数はビルトイン関数の関数パラメーターや戻り値に使用され、スクリプトで定義し直すことはできません。定義済みの定数を再定義しようとするとコンパイラエラーが発生します。
- ユーザー定義定数は各スクリプトに対し、プログラマーが必要に応じて宣言します。ユーザー定義定数は最初に宣言した後で再定義できますが、通常はあまりいいプログラミング方法ではありません。

ユーザー定義定数は、#define プリプロセッサステートメントを使って宣言します。(InstallScript では、C++ 言語でサポートしているような const キーワードを変数宣言に使用できません。) 文字列定数は引用符で括ります。数値の定数は引用符を使わず、数字だけを含みます。宣言された文字列定数は、文字列リテラルが使用できる場所であればどこでも使用できます。同様に、数値定数も、数値リテラルが使用できる場所であればどこでも使用できます。

次の例では、文字列定数と数値定数を宣言します。

```
#define COMPANY_NAME "Example_Company"
#define MAXCOUNT      1000
```

定数名は、InstallScript 識別子規則に従う必要があります。規則では、定数識別子はすべて大文字で作成します。InstallScript の定義済み定数はこの規則に従います。

データ構造体

データ構造は、メンバーと呼ばれる論理的に関連付けられた変数で構成される名前付きのデータアイテムです。多くのプログラム言語では、データ構造体はレコードと呼ばれ、レコード内の変数はフィールドと呼ばれます。InstallScript のデータ構造体はその形式と機能の点で C に類似しています。これには様々なデータタイプのメンバーを含むことができ、データ構造体内のメンバーはメンバー演算子 (.) を利用して直接参照することができます。

データ構造体の定義

データ構造体を定義するには、キーワード typedef を使ってデータ構造の名前を使って追跡します。構造内のフィールドは begin...end ブロック内で定義しなくてはなりません。下の例では、EMPLOYEE と呼ばれる構造を定義しています。EMPLOYEE のデータ構造には 3 つの変数 (従業員の名前の文字列変数、従業員の部署の文字列、および従業員の内線番号の数値変数) が含まれます。

```
typedef EMPLOYEE
begin
    STRING szName[50];
    STRING szDepartment[50];
    NUMBER nExtension;
end;
```

データ構造を定義するときは、実際は新しいデータ型を定義しています。プログラム内のデータ構造を利用利するには、まずその種類の変数を宣言しなくてはなりません。そのためには、定義されたデータ構造の名前をデータ型として利用し、次に識別子を続けます。下の例では、EMPLOYEE 型の変数を作成します。

```
EMPLOYEE structEmployee;
```

構造変数のメンバーを参照するには、メンバー演算子 (.) を使用します。). 下の例では、リテラル値が structEmployee の各メンバーに割り当てられています。

```
structEmployee.szName = "I. S. Coder";
structEmployee.szDepartment = " 開発 ";
structEmployee.nExtension = 555;
```

制限

構造体には次の制限が適用されます :

- 代入演算子を使って newstruct = struct1 の様に、ある構造の内容を別の構造へ割り当てることはできません。その代わり、一要素づつ構造をコピーしなくてはなりません。
- InstallScript のオートサイズ機能は typedef ステートメントでは使用できないため、構造内ですべての STRING 宣言のサイズを指定してください。
- 関数の内部で構造を宣言することはできません。
- 構造内で BYREF 演算子を利用することも、また BYREF 演算子を使って宣言したパラメーターの構造番号を渡すこともできません。ユーザー定義関数のユーザー定義構造の番号を変更するには、ポインターを構造へパスしてから 構造ポインター演算子 (->) を使って関数内のデータへアクセスします。
- ポインターへデータ構造のアドレスが割り当てられる前にこれを参照すると、ランタイムエラーが発生します。

構造体における Unicode サポート

InstallScript の構造体には、文字列、ポインター、その他の構造体をはじめとする任意の基本データ タイプを含めることができます。構造体に Unicode 文字列を含む必要があり、その構造体が外部 DLL に渡される場合、InstallScript エンジンはその構造内の文字列メンバー タイプを区別して、構造体サイズとメンバー オフセットを正しく計算します。Unicode として保存して渡す必要がある文字列メンバーは、WSTRING タイプを使って宣言できます。

Unicode 文字列が STRING 型で宣言して、文字列を構造内で使用すると、InstallScript エンジンはその文字列を外部 DLL に渡すときに ANSI として処理します。その結果、構造体のサイズとメンバーのオフセットに誤りが生じることがあり、DLL がその構造体に関連するデータの読み取りまたは書き込みが正しく行われません。

構造体のポインター メンバーは、WPOINTER として宣言できます。これによって、構造体で Unicode 文字列へのポインターを格納できます。

例

C と同様に、InstallScript ではデータ構造のネストや埋め込みが可能です。例えば、長方形の左上と右下の座標を定義づけるのに利用できる構造を作成することにします。各座標は 2 つの座標 (x 値と y 値) を持ちます。4 つのメンバー (左上の角の位置 x と y 、そして右下の角の位置 x と y) から成る構造を定義することができます。

しかし、各 x と y ペアが論理単位であるため、縦横の位置を定義する 2 つのメンバーをもつ POINT と呼ばれる構造をまず定義づける場合もあります。その場合、POINT 型の 2 つのメンバーを含む RECT と呼ばれる構造を定義づけることができます。POINT のひとつは左上の座標、そしてもうひとつは右下の座標を定義づけます。これら 2 つの構造体は次の通りです :

```
// ポイント構造を定義します。
typedef POINT
begin
    SHORT nX;
    SHORT nY;
end;

// ネストされたポイント構造を使って長方形構造を定義づけます。
typedef RECT
begin
    POINT UpperLeft;
    POINT LowerRight;
```

```
end;
```

構造へのポインターによって構造が参照されるとき、構造ポインター演算子 (->) を使って構造メンバーを指定しなくてはなりません。下の例では、RECT 型の変数が宣言され構造へのポインターが宣言された後、RECT 変数のアドレスがポインターへ割り当てられます。最後に、構造ポインター演算子を使って各メンバーを 0 に初期化します。

```
RECT Rectangle;
RECT POINTER pRect;

pRect = &Rectangle;
pRect->UpperLeft.nX  = 0;
pRect->UpperLeft.nY  = 0;
pRect->LowerRight.nX = 0;
pRect->LowerRight.nY = 0;
```

次のスクリプトでは、構造ポインター、ネスト構造、および構造ポインター演算子を使って修飾参照する構造ポインターについて、より完成されたデモンストレーションを行います。

```
// 構造体を使ってポイントを定義します。
typedef POINT
begin
    SHORT nX;
    SHORT nY;
end;

// ネスト構造を使って長方形を定義づけます。
typedef RECT
begin
    POINT UpperLeft;
    POINT LowerRight;
end;

// 長方形構造変数を宣言します。
RECT Rectangle;

// RECT 構造へポインターを定義します。
RECT POINTER pRect;

// 構造内容を表示するよう関数を宣言します。
prototype ShiftRectBy2(RECT POINTER);

. を参照してください。を参照してください。

// 長方形構造へのポインターを取得します。
pRect = &Rectangle;

// 長方形を定義するポイントを定義します。
pRect->UpperLeft.nX = 100;
pRect->UpperLeft.nY = 400;
pRect->LowerRight.nX = 200;
pRect->LowerRight.nY = 100;

// ShiftRectBy2 を呼び出す前にポイント x 値と  y  値を表示します。
SprintfBox (INFORMATION,
        "BEFORE calling ShiftRectBy2",
        "pRect->UpperLeft.nX  = %d¥n" +
        "pRect->UpperLeft.nY  = %d¥n" +
        "pRect->LowerRight.nX = %d¥n" +
```

```
            "pRect->LowerRight.nY = %d¥n",
            pRect->UpperLeft.nX,
            pRect->UpperLeft.nY,
            pRect->LowerRight.nX,
            pRect->LowerRight.nY
            );

    // 長方形を上方向へ 2、右方向へ 2 シフトさせます。
    ShiftRectBy2(pRect);

    // ShiftRectBy2 を呼び出した後にポイント x 値と y 値を表示します。
    SprintfBox (INFORMATION,
            "AFTER calling ShiftRectBy2",
            "pRect->UpperLeft.nX  = %d¥n" +
            "pRect->UpperLeft.nY  = %d¥n" +
            "pRect->LowerRight.nX = %d¥n" +
            "pRect->LowerRight.nY = %d¥n",
            pRect->UpperLeft.nX,
            pRect->UpperLeft.nY,
            pRect->LowerRight.nX,
            pRect->LowerRight.nY
            );

    // 長方形シフト関数を定義します。
    function ShiftRectBy2(pR)
    begin
       pR->UpperLeft.nX = pR->UpperLeft.nX + 2;
       pR->UpperLeft.nY = pR->UpperLeft.nY + 2;
       pR->LowerRight.nX = pR->LowerRight.nX + 2;
       pR->LowerRight.nY = pR->LowerRight.nY + 2;
    end;
```

言語識別子

InstallShield は Windows がサポートするすべての言語に対応する言語定数を提供します。しかし、これらの定数のほとんどは言語特定のコンポーネントの指定や、言語フィルタリングでサポートされていません。

InstallScript 定数は次の状況で利用することが可能です。

FeatureFilterLanguage のパラメーター

InstallScript 言語定数は関数 FeatureFilterLanguage への 2 番目のパラメーターとして利用することができます。この状況では、言語定数はどのファイルをフィルターするのか、あるいはフィルターを解除するのかを指定します。これには、サポートされている言語テーブル で一覧となっているサポートされている言語定数のみを利用してください。サポートされていない言語用のコンポーネントはメディアのビルド中にフィルタ（除外）され インストールされていない為、サポートされていない言語定数をコンポーネントのフィルタリングに利用しても効果は得られません。

GetSystemInfo の戻り値

言語定数は、nItem パラメーターで定数 LANGUAGE と共に呼び出した場合、関数 GetSystemInfo が nvResult に戻す値として利用されます。Windows はすべての言語定数をサポートするので、この状況では **ISRTDefs.h** にリストされている言語定数はどれでも利用することができます。

メモ・これらの戻り値に基づいた言語のフィルタリングを含むインストレーションの場合、switch ステートメントを使って、この関数によって戻される定数を言語フィルタリングでサポートされている定数の 1 つに変換する必要があります。

言語定数リファレンス

InstallShield でサポートされている言語定数の完全なリスト、そして対応する数値については **ISRTDefs.h** ファイルを参照してください。このファイルは InstallShield Program Files フォルダー内、**Script¥Include** フォルダーにあります。

サポートされている言語と InstallScript の定義済み定数のリストは、InstallScript の言語サポート をご覧ください。

メモ・セットアップがメッセージやプロンプトを表示するのに利用する言語は、システム変数 SELECTED_LANGUAGE に格納されています。

セットアップの初期化中にエンド ユーザーがインストール言語を選択する言語選択ダイアログが複数言語インストールで利用されるとき、言語ダイアログは対応する Windows の名前を表示します。なぜなら、これらの名前は Windows によって生成され、インストールが実行される Windows のバージョンにローカライズされるためです。

ポインター

ポインターとは別の変数のアドレスを含む変数です。ポインターを宣言するには、以下の 2 つのサンプル コードで示されるように、キーワード POINTER または WPOINTER を使って、変数名を続けます :

```
POINTER pPointerName;
WPOINTER pWPointerName;
```

データ構造のメンバーへアクセスするのに利用するポインターを宣言するには、キーワード POINTER または WPOINTER の前に構造タイプを配置します :

```
typedef RECT
begin
  SHORT sX;
  SHORT sY;
end;

RECT Rectangle;

RECT POINTER pRect;
```

アドレス演算子 (&) を利用して変数のアドレスをポインター変数へ割り当てます :

```
pPointerName = &MyStructure;
pNum = &nvNumber;
pString = &svString;
```

構造へのポインターをパラメーターとして扱う関数を定義する際、以下に示した通り、関数プロトタイプで 構造名を POINTER または WPOINTER と共に利用して下さい : 構造へのポインターをそのパラメーターとして指定する関数プロトタイプは、いづれも構造宣言の後に宣言しなくてはならない点に注意してください。

```
  typedef RECT
begin
  SHORT sX;
```

```
  SHORT sY;
end;

RECT Rectangle;
RECT POINTER pRect;

prototype SizeRectangle(RECT POINTER);
```

. を参照してください。を参照してください。

```
pRect = &Rectangle;
SizeRectangle(pRect);
```

. を参照してください。を参照してください。

```
function SizeRectangle(pRectangle)
begin
  pRectangle->sX = 10;
  pRectangle->sY = 5;
end;
```

文字列へのポインタを InstallScript コード外部で実装される関数に渡す

InstallScript コンパイラーでは、Unicode または ANSI 文字列へのポインターをスクリプト外部で実装される関数に渡すことができます。たとえば、文字列へのポインターをパラメーターで受け付ける DLL 関数を呼び出す場合、C または C++ 言語で DLL 関数のプロトタイプは以下のようになります :

```
void _stdcall MyDllFunction(LPCSTR pszString);
```

InstallScript では、関数は以下のようにプロトタイプ化されます :

```
prototype DLL.MyDllFunction(POINTER);
```

アドレス演算子 (&) を使って関数を呼び出して、ポインターを文字列に渡すことができます :

```
DLL.MyDllFunction(&myString);
```

スクリプト エンジンがこの関数呼び出しを行うとき、文字列 **myString** 内のデータはポインター値を通して **MyDllFunction** に渡されます。**MyDllFunction** は、**myString** に含まれる文字列の ANSI 表記へのポインターを受け取ります。

ポインター型 WPOINTER (またはオプションで、wpointer または LPWSTR) を使って、Unicode 文字列へのポインターをスクリプト外部に渡すことができます。たとえば、DLL が Unicode 文字列を使用する場合、C または C++ 言語でそのプロトタイプを以下のように変更できます :

```
void _stdcall MyDllFunction(LPCWSTR pszString);
```

InstallScript では、Unicode 文字列を使用する DLL へ Unicode 文字列ポインターを渡すために必要な変更は、プロトタイプだけです。これには、以下のとおり WPOINTER 型が含まれます :

```
prototype DLL.MyDllFunction(WPOINTER);
```

実行中のスクリプトで DLL 関数が呼び出されると、エンジンは ANSI バージョンではなく、**myString** に格納されている文字列の Unicode のコピーへのポインターを渡します。

ポインターの代わりに STRING および WSTRING を使用する

ほとんどの場合、文字列を外部 DLL 関数に渡す場合にポインターは不要です。STRING および WSTRING 型を POINTER または WPOINTER の位置で使用することができます。DLL 関数が ANSI 文字列を受け付ける場合、STRING 型を使用します。DLL 関数が Unicode 文字列を受け付ける場合、WSTRING 型を使用します。BYREF および BYVAL を使って、外部 DLL 関数による変更が可能な文字列または変更が不可能な文字列を渡すことができます。

したがって、関数に以下のプロトタイプを使用すると、値またはリファレンス (必要に応じてプロトタイプを BYREF に変更) によって ANSI 文字列が渡されます。

```
prototype DLL.MyDllFunction(byval string);
```

パラメーター型を BYVAL WSTRING に変更すると、ANSI バージョンの代わりに文字列の Unicode バージョンを渡すことが可能となります。

変数データ

変数とは、プログラムの実行中にその値を変更することのできる名前が付けられたデータ項目です。

変数宣言

フォーマット

変数は次のフォーマットで宣言しなくてはなりません :

```
data_type  VariableName1[, VariableName2 [,...]];
```

規則

変数宣言は次の規則に従わなくてはなりません :

- 変数命は最大 32 文字が可能です。
- 複数の変数名を単一の宣言で指定する場合、名前はコンマで区切らなくてはなりません。
- 各変数宣言の終わりにはセミコロンを付けなくてはなりません。

注意 : *InstallScript 変数と関数の名前は大文字と小文字を区別します。例えば、svItemCounter は svITEMCOUNTER と同じです。*

変数宣言の例

次の例では、7 つの変数が宣言されます。最後の宣言は 3 つの数値変数を作成します。

```
BOOL bValidEntry;

LONG lPopulation;

// 文字列のサイズは明示されます
STRING szUserName[128];

// 文字列はオートサイズ化されます
STRING szMessage;

NUMBER nFileSize, nDirSize, nDiskSpace;
```

文字列変数の宣言

サイズを明示して、または明示せずに文字列変数を宣言することができます。サイズを明示せずに宣言された文字列変数は、割り当てられた値を受け入れるようにセットアップの最中に自動的にサイズが調整されます。文字列サイズを *明示的に宣言しなくてはならない* 外部関数 (DLL または Windows API) へ渡される文字列変数以外、すべての文字列変数にはオートサイズの利用を推奨します。文字列の最大文字制限は 65534 です。

グローバル変数とローカル変数の違い

変数はグローバル変数、またはローカル変数のいずれかです :

- メインプログラムブロック以外、また関数の外で宣言されていれば、それはグローバル変数です。グローバル変数は、その宣言に続くセットアップスクリプトのすべてのステートメントで利用することができます。
- 関数宣言とその関数内のキーワード *begin* の間で宣言されていれば、それはローカル変数です。ローカル変数は、それが宣言された関数の中でのみ利用することが可能です

プロジェクト・*InstallScript イベントは、基本の MSI プロジェクトとマージモジュールプロジェクトでは使用されていません。したがって、これらのプロジェクトタイプのすべての InstallScript コードを InstallScript カスタムアクションに書き込む必要があります。グローバル変数は、これらのカスタム アクションが起動されたときの状態を共有しません。*

メモ・*InstallScript システム変数はグローバル変数なので、メインプログラム及びスクリプトのすべての関数で有効です。*

次の例では、変数 nVisibleEverywhere はスクリプト中の任意のステートメントによって参照することができます。変数 nVisibleOnlyToFunctions は関数によってのみ参照することができます。nVisibleOnlyToSecondFunction はメインプログラム、あるいは FirstFunction によって参照することができません。変数 szString は FirstFunction のローカル変数です。

```
    prototype FirstFunction();
    prototype SecondFunction();

    NUMBER nVisibleEverywhere;

    . を参照してください。を参照してください。

    nVisibleEverywhere = 10;

    FirstFunction();
    SecondFunction();

    . を参照してください。を参照してください。

    NUMBER nVisibleOnlyToFunctions;

function FirstFunction()
  STRING szString;
begin
  szString = "FirstFunction のローカル";
  nVisibleOnlyToFunctions = 20;
end;

    NUMBER nVisibleOnlyToSecondFunction;

function SecondFunction()
begin
  nVisibleOnlyToSecondFunction = 30;
end;
```

スクリプト内の識別子は固有でなくてはなりませんが、ローカル変数とグローバル変数は同じ名前を共有すること、またある関数が別の関数でローカル変数として宣言されている同じ名前を使ってローカル変数を宣言することも可能です。これらの例外は、InstallShield が関連付けられる関数に基づいてローカル変数名を認識する為に許可されています。下の例では、グローバル変数 szVal は同じ名前のローカル変数を持つ AFunction の動作による影響を受けません。関数 MessageBox は文字列はグローバル変数 szVal へ割り当てられた値 “YES” を表示します。

```
    STRING szVal;

    prototype AFunction();

    . を参照してください。を参照してください。

    szVal = "YES";
    AFunction();
    MessageBox(szVal, INFORMATION);

    . を参照してください。を参照してください。

function AFunction()
  STRING szVal;
begin
  szVal = "NO";
end;
```

関数定義内のパラメーター名はローカル変数とみなされます。グローバル変数が、そのパラメーターが同じ名前のグローバル変数を持つ関数へ渡されたとき、そのグローバル変数の値が変更されることはありません (関数プロトタイプでパラメーターを BYREF 演算子を使って指定した場合を除く)。次の例では、AFunction はグローバル変数 szVal 上に影響を与えません。スクリプトは文字列 “YES” を表示します。

```
    STRING szVal;
    prototype AFunction(STRING);

    . を参照してください。を参照してください。

    szVal = "YES";
    AFunction(szVal);
    MessageBox(szVal, INFORMATION);


    . を参照してください。を参照してください。

function AFunction(szVal)
begin
  szVal = "NO";
end;
```

文字列変数

文字列変数に関する情報については、次のトピックを参照してください。

- 文字列索引作成
- 文字列サイズと Autosize

文字列索引作成

文字列変数はヌルターミネータを使った Unicode 文字配列です。文字列名と、その後に角かっこで囲まれたインデックス値を指定して、文字列内で各文字を参照することができます。文字列の最初の文字は 0 位置でであることに注意してください。

下の例では、関数 BlankLeadingZeros は文字列インデックス法を利用して、数字が象徴する文字列の先頭のゼロを空白文字に置換します。

```
    prototype BlankLeadingZeros(BYREF STRING);

function BlankLeadingZeros(szString)
  INT iVal, iLength;
begin
  iVal = 0;
  iLength = StrLength (szString);

  while (szString[iVal] = "0") && (iVal <= iLength)
    szString[iVal] = " ";
    iVal++;
  endwhile;
end;
```

文字列サイズと Autosize

InstallShield オートサイズ

サイズ仕様なしで文字列変数を宣言するとき、InstallShield はその変数用の文字列バッファーのサイズを自動的に設定します。バッファーの割り当ては、最初に文字列を変数へ割り当てたときに行われます。後でその変数により長い文字列を割り当てた場合、InstallShield は有効なメモリの限界まで長い文字列を許容できるように、メモリ割り当てを増やします。しかし、後でオートサイズされた変数よりも短い文字列を割り当てた場合、InstallShield はメモリ割り当てを減らしません。

注意・*InstallShield のオートサイズ機能は typedef ステートメントでは動作しないので、構造ですべての STRING 宣言のサイズを指定しなくてはなりません。*

文字列サイズを指定する

文字列サイズを指定するとき、ヌルターミネータ 1 文字の位置を宣言しなくてはなりません。例えば、文字列に 128 文字を含む場合は、ヌルターミネータのスペースを考慮して 129 文字で長さを宣言しなくてはなりません。この理由から、最小文字列サイズは 2 となります。

サイズを宣言した文字列を利用するとき、その文字列が別の文字列とどのように利用されるかを考慮に入れなくてはなりません。たとえば、次の関数呼び出しを参考にして下さい :

```
  STRING szQuestion[20], szDefault[20], svResult[50];

begin
  szQuestion = " 会社名を入力します ";
  szDefault  = "My Software Company";
  AskText (szQuestion, szDefault, svResult);
```

文字列 svResult のサイズは文字列 szDefault のサイズと同じかそれよりも大きくなくてはなりません。そうでない場合、szDefault が受け入れた場合、関数が戻す svResult 変数へ収まりません。最も簡単な競合の回避方法は、InstallShield が (typedef ステートメントの文字列以外) すべての文字列をオートサイズするよう設定することです。

注意・*リファレンスによって関数に渡されるオートサイズ文字列変数は呼び出された関数の中では自動サイズ調整されません。関数が現在のパラメーターのサイズより大きい長さの値を割り当てようとすると、ランタイム エラー 401 が発生します。*

システム変数

システム変数は、ソースパス、ターゲットパス、Windows フォルダー、および Windows システムフォルダーのような情報を含む、あらかじめ定義されたスクリプト変数です。インストールは、インストール プロセスが始まった時にこれらのシステム変数を自動的に初期化するので、スクリプトで宣言する必要はありません。

プロジェクト・*Windows Installer ディレクトリ プロパティの多く (**INSTALLDIR**、**AppDataFolder**、および **TempFolder** など) は、基本の MSI および InstallScript MSI プロジェクトの InstallScript コードで変数として直接使用できます。*

システム変数とテキスト置換

システム変数の中には、対応するテキスト置換を持つものがあります。インストールは内部的にテキスト置換を使用して、下のテーブルで表示されているように特定のシステム変数の値を設定します。定義済みのテキスト置換を使用するのと同じ方法で、これらのテキスト置換をスクリプトで使用することができます。

書き込み可能なシステム変数とテキスト置換

テーブル 1 - 書き込み可能なシステム変数とテキスト置換

スクリプト変数	対応するテキスト置換	Comments
ALLUSERS	<PERUSER_INSTALL>	
DISK1TARGET	<DISK1TARGET>	
IFX_COMPANY_NAME	<IFX_COMPANY_NAME>	このテキスト置換がオブジェクトスクリプトで定義されている場合、メインのインストーラやインストーラのその他のオブジェクトではなく、そのオブジェクトにだけテキスト置換が適用されます。 メインのインストーラで定義されている場合は、オブジェクトには適用されません。
IFX_INSTALLED_DISPLAY_VERSION	<IFX_INSTALLED_DISPLAY_VERSION>	このテキスト置換がオブジェクトスクリプトで定義されている場合、メインのインストーラやインストーラのその他のオブジェクトではなく、そのオブジェクトにだけテキスト置換が適用されます。 メインのインストーラで定義されている場合は、オブジェクトには適用されません。
IFX_INSTALLED_VERSION	<IFX_INSTALLED_VERSION>	このテキスト置換がオブジェクトスクリプトで定義されている場合、メインのインストーラやインストーラのその他のオブジェクトではなく、そのオブジェクトにだけテキスト置換が適用されます。 メインのインストーラで定義されている場合は、オブジェクトには適用されません。
IFX_MULTI_INSTANCE_SUFFIX	<IFX_MULTI_INSTANCE_SUFFIX>	

テーブル 1・書き込み可能なシステム変数とテキスト置換（続き）

スクリプト変数	対応するテキスト置換	Comments
IFX_PRODUCT_DISPLAY_NAME	<IFX_PRODUCT_DISPLAY_NAME>	このテキスト置換がオブジェクトスクリプトで定義されている場合、メインのインストーラやインストーラのその他のオブジェクトではなく、そのオブジェクトにだけテキスト置換が適用されます。 メインのインストーラで定義されている場合は、オブジェクトには適用されません。
IFX_PRODUCT_DISPLAY_VERSION	<IFX_PRODUCT_DISPLAY_VERSION>	このテキスト置換がオブジェクトスクリプトで定義されている場合、メインのインストーラやインストーラのその他のオブジェクトではなく、そのオブジェクトにだけテキスト置換が適用されます。 メインのインストーラで定義されている場合は、オブジェクトには適用されません。
IFX_PRODUCT_KEY	<IFX_PRODUCT_KEY>	このテキスト置換がオブジェクトスクリプトで定義されている場合、メインのインストーラやインストーラのその他のオブジェクトではなく、そのオブジェクトにだけテキスト置換が適用されます。 メインのインストーラで定義されている場合は、オブジェクトには適用されません。
IFX_PRODUCT_NAME	<IFX_PRODUCT_NAME>	このテキスト置換がオブジェクトスクリプトで定義されている場合、メインのインストーラやインストーラのその他のオブジェクトではなく、そのオブジェクトにだけテキスト置換が適用されます。 メインのインストーラで定義されている場合は、オブジェクトには適用されません。

テーブル 1・書き込み可能なシステム変数とテキスト置換（続き）

スクリプト変数	対応するテキスト置換	Comments
IFX_PRODUCT_VERSION	<IFX_PRODUCT_VERSION>	このテキスト置換がオブジェクトスクリプトで定義されている場合、メインのインストーラやインストーラのその他のオブジェクトではなく、そのオブジェクトにだけテキスト置換が適用されます。 メインのインストーラで定義されている場合は、オブジェクトには適用されません。
IFX_SETUP_TITLE	<IFX_SETUP_TITLE>	このテキスト置換がオブジェクトスクリプトで定義されている場合、メインのインストーラやインストーラのその他のオブジェクトではなく、そのオブジェクトにだけテキスト置換が適用されます。 メインのインストーラで定義されている場合は、オブジェクトには適用されません。
IFX_SUPPORTED_VERSIONS	<IFX_SUPPORTED_VERSIONS>	このテキスト置換がオブジェクトスクリプトで定義されている場合、メインのインストーラやインストーラのその他のオブジェクトではなく、そのオブジェクトにだけテキスト置換が適用されます。 メインのインストーラで定義されている場合は、オブジェクトには適用されません。
SHELL_OBJECT_FOLDER	<SHELL_OBJECT_FOLDER>	
SRCDIR（ローカルから読み出す）	<SRCDIR>	
SRCDISK（ローカルから読み出す）	<SRCDISK>	
TARGETDIR	<TARGETDIR>	
TARGETDISK	<TARGETDISK>	
UNINST	<UNINST>	
UNINSTALL_STRING	<UNINSTALL_STRING>	

読み取り専用システム変数とテキスト置換

テーブル 2 - 読み取り専用システム変数とテキスト置換

スクリプト変数	対応するテキスト置換	Comments
COMMONFILES	<COMMONFILES>	
DISK1SETUPEXENAME	<DISK1SETUPEXENAME>	
ENGINECOMMONDIR	<ENGINECOMMONDIR>	
ENGINEDIR	<ENGINEDIR>	
FOLDER_APPDATA	<FOLDER_APPDATA>	
FOLDER_DOTNET_10	<FOLDER_DOTNET_10>	
FOLDER_DOTNET_11	<FOLDER_DOTNET_11>	
FOLDER_DOTNET_20	<FOLDER_DOTNET_20>	
FOLDER_DOTNET_30	<FOLDER_DOTNET_30>	
FOLDER_DOTNET_35	<FOLDER_DOTNET_35>	
FOLDER_DOTNET_40	<FOLDER_DOTNET_40>	
FOLDER_PERSONAL	<PERSONALDIR>	
FOLDER_TEMP	<FOLDER_TEMP>	
ISRES	<ISRES>	このテキスト置換がオブジェクト スクリプトで定義されている場合、メインのインストーラやインストーラのその他のオブジェクトではなく、そのオブジェクトにだけテキスト置換が適用されます。 メインのインストーラで定義されている場合は、オブジェクトには適用されません。

テーブル 2 ・読み取り専用システム変数とテキスト置換 (続き)

スクリプト変数	対応するテキスト置換	Comments
ISUSER	<ISUSER>	このテキスト置換がオブジェクト スクリプトで定義されている場合、メインのインストーラやインストーラのその他のオブジェクトではなく、そのオブジェクトにだけテキスト置換が適用されます。 メインのインストーラで定義されている場合は、オブジェクトには適用されません。
MULTI_INSTANCE_COUNT	<MULTI_INSTANCE_COUNT>	
PACKAGE_LOCATION	<PACKAGE_LOCATION>	
PROGRAMFILES	<PROGRAMFILES>	
SELECTED_LANGUAGE	<SELECTED_LANGUAGE>	
SHAREDSUPPORTDIR	<SHOW_PASSWORD_DIALOG>	
SHOW_PASSWORD_DIALOG	<SHOW_PASSWORD_DIALOG>	
SUPPORTDIR	<SUPPORTDIR>	
WINDIR	<WINDIR>	
WINDISK	<WINDISK>	
WINSYSDIR	<WINSYSDIR>	
WINSYSDISK	<WINSYSDISK>	

ADDREMOVE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

コントロール パネルの [プログラムの追加と削除] ダイアログ ボックスから実行される時、ADDREMOVE システム変数はゼロ以外の値と等しく、それ以外の場合は FALSE と等しくなるよう設定されています。このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。

ADDREMOVE_COMBINEDBUTTON

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数の値はアプリケーション アンインストール レジストリ キーの ModifyPath および UninstallString 値が存在するかどうか、またはそのデータを指定するために MaintenanceStart 関数が利用します。こうして [プログラムの追加と削除] ダイアログ ボックスのエントリに [変更] および [削除] ボタンを別々に表示するか、または共通の [変更 / 削除] ボタンを表示するかを示します。この変数についての詳細は、「MaintenanceStart」を参照してください。

このシステム変数は FALSE に初期化されます。

ADDREMOVE_HIDECHANGEOPTION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数の値は、MaintenanceStart 関数で、アプリケーション アンインストール レジストリ キーの ModifyPath、NoModify、UninstallString の値を指定するために使用されます。NoModify レジストリ値はコントロール パネルの [プログラムの追加と削除] ダイアログ ボックスでアプリケーションの [変更] ボタンを表示するかどうかを指定します。ModifyPath および UninstallString レジストリ値は [変更] および [削除] ボタンの動作を指定します。

このシステム変数は [一般情報] ビューの " 変更ボタンを無効にする " 設定で指定した値に基づいて初期化されます。

ADDREMOVE_HIDEREMOVEOPTION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数の値は、MaintenanceStart 関数で、アプリケーション アンインストール レジストリ キーの ModifyPath、NoRemove、UninstallString の値を指定するために使用されます。NoRemove レジストリ値はコントロール パネルの [プログラムの追加と削除] ダイアログ ボックスでアプリケーションの [削除] ボタンを表示するかどうかを指定します。ModifyPath および UninstallString レジストリ値は [変更] および [削除] ボタンの動作を指定します。

このシステム変数は [一般情報] ビューの " 削除ボタンを無効にする " 設定で指定した値に基づいて初期化されます。

ADDREMOVE_STRING_REMOVEONLY

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数の値は アプリケーション アンインストール レジストリ キーの UninstallString 値にデータを指定するために MaintenanceStart 関数が利用します。UninstallString 値はアプリケーションの [プログラムの追加と削除] エントリに [削除] ボタン が存在する場合、その動作を指定します。この変数についての詳細は、「MaintenanceStart」を参照してください。

このシステム変数は文字列値 " -removeonly" に初期化されます。

ADDREMOVE_SYSTEMCOMPONENT

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数の値は アプリケーションアンインストール レジストリキー の SystemComponent 値にデータを指定するために MaintenanceStart 関数が利用します。SystemComponent 値はコントロール パネルの [プログラムの追加と削除] ダイアログ ボックスでアプリケーションのエントリを表示するかどうかを指定します。

このシステム変数はエントリが表示されることを意味する FALSE に初期化されます。

ALLUSERS

ALLUSERS システム変数は、ターゲット システム上の現在のユーザーまたはすべてのユーザーによるアプリケーションのインストールを許可するインストール作成の鍵となります。ALLUSERS の値は次を決定します :

- RegDBSetDefaultRoot(HKEY_USER_SELECTABLE) が呼び出された後に呼び出されるレジストリ関数が利用するルートキー
- アプリケーション アンインストール キーがその下に作成されるルート キー
- レジストリセットの HKEY_USER_SELECTABLE ルートキーで指定されたレジストリエントリの場所
- システム変数 DISK1TARGET の値 (メンテナンスインストールとアンインストールを実行するためのインストールファイルのいくつかをコピーしたものが配置されているフォルダーへのパスを指定します。)
- デフォルトの OnFirstUIBefore コードが設定するシステム変数 TARGETDIR のデフォルト値
- ショートカットを作成した時に、[タイプ] プロパティ を [自動] に設定して InstallShield で定義されたショートカットが、個人のショートカットまたは一般ショートカットのリストに表示するかどうか
- SdCustomerInformation と SdCustomerInformationEx ダイアログでのデフォルトのオプション選択

ALLUSERS は、COM DLL ファイルの登録に影響は与えません。

次のセクションは、異なる種類のプロジェクトにおける、ALLUSERS の値の判別および設定方法の説明です。

InstallScript インストール

インストールが初回インストールとして実行されるとき、InstallScript エンジンは、初期化のプロセスで、ALLUSERS 変数に使用する最も適切な値を判別し、それをその値で初期化します。

- ユーザーに管理者権限がない場合、ALLUSERS は 0 に設定されます。この設定により、ユーザーごとインストールが実行されます。
- それ以外の場合、ALLUSERS は 1 に設定されます。この設定により、マシンごとインストールが実行されます。

インストールがメンテナンス モードで実行された場合、InstallScript エンジンは、ALLUSERS 変数の値を、初回インストールがユーザーごとに実行されたか、または、マシンごとに実行されたかに基づいて (アンインストール情報がインストールされている場所に基づいて) 判別します。

ターゲット システム上の現在のユーザーまたはすべてのユーザーへアプリケーションをインストールすることができる InstallScript インストールの参考例は、ALLUSERS Sample Project フォルダーにあるサンプル プロジェクトをご覧ください。このフォルダは、InstallShield Program Files フォルダの Samples フォルダ内にあるサブフォルダです。デフォルト保存先は次の場所です :

C:¥Program Files¥InstallShield¥2016¥Samples¥InstallScript¥ALLUSERS Sample Project

基本の MSI と InstallScript MSI インストールの InstallScriptk カスタム アクション

ALLUSERS 変数の値を取得する

ALLUSERS InstallScript 変数は、Windows Installer プロパティ **ALLUSERS** を次のようにクエリして決定されます :

テーブル 3 · ALLUSERS InstallScript 変数の値を取得する

Windows Installer プロパティ	InstallScript 変数	注
""	0	カスタム アクション内にあるユーザーごと、または、マシンごとの依存スクリプト コードは、ユーザーごととして動作します。
1	1	カスタム アクション内にあるユーザーごと、または、マシンごとの依存スクリプト コードは、マシンごととして動作します。
その他の値	InstallScript エンジンは、適切な値を判別する試みを行います。	

InstallScript エンジンが遅延カスタム アクションが実行中であると判別したために Windows Installer プロパティ **ALLUSERS** が判別できなかった場合、予期しないプロパティの値が返されるか、**MSIGetProperty** によってエラーの値が返されます。この理由により、InstallScript エンジンは、変数に最も適切な値を判別するように試みます。

InstallScript エンジンは、**MsiGetMode** を MSIRUNMODE_SCHEDULED、MSIRUNMODE_ROLLBACK、および MSIRUNMODE_COMMIT フラグと共に使用して、遅延カスタム アクションが実行中であるかどうかを判別します。**MsiGetMode** は、上記の値について True を返し、カスタム アクションは遅延と仮定され、InstallScript メカニズムが使用されます。

基本の MSI インストールで、**Property** テーブルの **ALLUSERS** に値がない場合、インストールが **ALLUSERS** ダイアログ (**ALLUSERS** Windows Installer プロパティを設定する CustomerInformation ダイアログなど) を表示する前に実行される InstallScript カスタム アクションでは ALLUSERS InstallScript 変数が 0 に設定されています。このため、InstallScript カスタム アクションはユーザーごとの動作を示します。したがって、すべての基本の MSI インストールでは、**Property** テーブルの **ALLUSERS** にデフォルトの値を持たせることをお勧めします。

ALLUSERS 変数の値を設定する

ALLUSERS InstallScript 変数がスクリプトで設定されたとき、InstallScript エンジンはまず、プラットフォームおよび権限レベルを確認して ALLUSERS InstallScript 変数を変更することが許可されるかどうかを判別します。(エンドユーザーが管理者またはパワー ユーザーでないとき、ALLUSERS は変更できません。)

ALLUSERS InstallScript 変数が変更可能な場合、InstallScript エンジンは、次のように、**ALLUSERS** Windows Installer プロパティを適切に更新する試みを行います:

テーブル 4 · ALLUSERS InstallScript 変数の値を設定する

InstallScript 変数	Windows Installer プロパティ
1	1
0	""

InstallScript エンジンは、Windows Installer プロパティが設定不可能な場合も、ALLUSERS InstallScript 変数を設定します。これにより、Windows Installer プロパティと InstallScript 変数の同期で問題が発生する可能性があります。したがって、カスタム アクションで ALLUSERS InstallScript 変数を変更する場合、プロパティが正常に変更できるように、Windows Installer プロパティも手動で設定することをお勧めします。

以下は、ユーザー アカウント制御が有効になっている Windows Vista における様々なシナリオでの動作です:

テーブル 5 · UAC が有効になっている Windows Vista での ALLUSERS の値

カスタム アクションの種類	マニフェスト	Property テーブルの ALLUSERS の値	Windows Installer プロパティ	結果の InstallScript 変数	注
即時	最高権限	2	1	1	
即時	起動者	2	""	0	InstallScript 変数は変更できません。マシンごとの InstallScript アクションは不可能です。
即時	最高権限	1	1	1	
即時	起動者	1	1	1	InstallScript 変数は変更できません。マシンごとの InstallScript アクションは失敗します。

テーブル 5 ・ UAC が有効になっている Windows Vista での ALLUSERS の値 (続き)

カスタム アクションの種類	マニフェスト	Property テーブルの ALLUSERS の値	Windows Installer プロパティ	結果の InstallScript 変数	注
即時、CustomerInformation ダイアログの前 (またはエンドユーザーがユーザーごとのインストールを選択)	起動者	""	""	0	InstallScript 変数は変更できません。マシンごとの InstallScript アクションは不可能です。
即時、CustomerInformation ダイアログの後	最高権限	""	1	1	
遅延	起動者	任意	判別できません	0	InstallScript 変数は変更できません。ALLUSERS の判別に InstallScript メソッドが使用されます。
遅延	最高権限	任意	判別できません	1	ALLUSERS の判別に InstallScript メソッドが使用されます。

以下は、Windows Vista 以前のシステムおよびユーザー アカウント制御が無効になっている Windows Vista システムにおける様々なシナリオでの動作です :

テーブル 6 ・ Windows Vista 以前のシステムおよび UAC が無効になっている Windows Vista における ALLUSERS の値

カスタム アクションの種類	ユーザー権限	Property テーブルの ALLUSERS の値	Windows Installer プロパティ	結果の InstallScript 変数	注
即時	管理者	2	1	1	
即時	制限あり	2	""	0	InstallScript 変数は変更できません。
即時	管理者	1	1	1	

テーブル 6 · Windows Vista 以前のシステムおよび UAC が無効になっている Windows Vista における ALLUSERS の値 (続き)

カスタム アクションの種類	ユーザー権限	Property テーブルの ALLUSERS の値	Windows Installer プロパティ	結果の InstallScript 変数	注
即時	制限あり	1	1	1	InstallScript 変数は変更できません。インストールは UI シーケンスで失敗し、マシンごとの InstallScript カスタム アクションは失敗します。
即時、CustomerInformation ダイアログの前 (またはエンドユーザーがユーザーごとのインストールを選択)	管理者	""	""	0	InstallScript 変数は変更できません。マシンごとの InstallScript アクションは不可能です。
即時、CustomerInformation ダイアログの後	管理者	""	1	1	
遅延	管理者	任意	判別できません	1	ALLUSERS の判別に InstallScript メソッドが使用されます。
遅延	制限あり	任意	判別できません	0	InstallScript 変数は変更できません。ALLUSERS の判別に InstallScript メソッドが使用されます。
Windows 9x のすべて	なし	任意	任意	1	InstallScript 変数は変更できません。ALLUSERS の判別に InstallScript メソッドが使用されます。

結果の InstallScript 変数が 0 の場合、InstallScript カスタム アクションがユーザーごとの環境で使用されます。結果の InstallScript 変数が 1 の場合、InstallScript カスタム アクションがマシンごとの環境で使用されます。

InstallScript MSI インストールにおけるイベント ドリブン型 InstallScript コード

InstallScript MSI インストールの動作は、ALLUSERS InstallScript 変数が変更されたとき、インストールが InstallScript カスタム アクションの説明に従って Windows Installer プロパティ **ALLUSERS** を更新しようとしたときを除き、InstallScript インストールの動作とほぼ同じです。

InstallScript MSI インストールでは、Windows Installer プロパティ **ALLUSERS** は、ALLUSERS InstallScript 変数の適切な値を判別するためにクエリされません。InstallScript エンジンは常に、InstallScript インストールの説明に従って値の判別が試みられます。

ALLUSERS によって異なる InstallScript インストールのデフォルトの動作

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

次のテーブルは、ALLUSERS システム変数に基づいてインストールがどのようにインストールされるかについての情報です。

テーブル 7 · ALLUSERS

	管理者 & パワー ユーザー	ユーザー & ゲスト
ALLUSERS のデフォルト設定	True	False
自動設定で確認される ALLUSERS プロパティ	マシンごと	ユーザーごと
HKEY_USER_SELECTABLE_AUTO	HKEY_LOCAL_MACHINE	HKEY_CURRENT_USER
レジストリ セット データ	HKEY_LOCAL_MACHINE	HKEY_CURRENT_USER
DISK1TARGET	Program Files¥InstallShield Installation Information¥GUID	My Docs¥InstallShield Installation Information¥GUID
アンインストール レジストリ キー	HKEY_LOCAL_MACHINE¥...¥Uninstall¥GUID	HKEY_CURRENT_USER¥...¥Uninstall¥GUID
TARGETDIR	Program Files¥Company Name¥Product Name	My Docs¥Company Name¥Product Name
エンジンのインストール	Program Files¥Common Files	My Docs¥...
COM 情報の登録 (.dll、.ocx、.exe)	サポートする	サポートしない

***メモ·** My Docs は、ユーザーが権利を持つインストール先の場所を指します。この値は、オペレーティング システムによって異なります。*

ADMINUSER

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript カスタム アクションがある基本の MSI プロジェクト*
- *InstallScript カスタム アクションがある InstallScript MSI プロジェクト*

この情報は InstallShield プロジェクトまたは、InstallScript MSI プロジェクト内のイベント ドリブン型の InstallScript コードには適用しません。

ADMINUSER システム変数は、Windows Installer プロパティ AdminUser の値に設定されます。

BATCH_INSTALL

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI – InstallScript ユーザー インターフェイス (UI) のスタイルが、外部 UI ハンドラーとして InstallScript エンジンを使用する従来型のスタイルの場合*

この情報は、InstallScript UI に新しいスタイル (InstallScript エンジンを埋め込み UI ハンドラーとして使用するスタイル) が使用されている InstallScript MSI プロジェクトには適用しません。詳しくは、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラとして使用する方法の違い」を参照してください。

BATCH_INSTALL システム変数がゼロ以外の値に設定されているとき、それはターゲット システムが再起動した後に実行が必要な操作が 1 つ以上あることを意味します。BATCH_INSTALL は、次の理由においてゼロ以外の値に設定されている可能性があります :

- あるファイルが既にターゲット システムに存在し、ロックされているため、そのファイルをインストールできないと判断された。
- BATCH_INSTALL がスクリプトから手動で非ゼロに設定された。これは、埋め込みインストールを完了するために再起動が必要だとオブジェクトで判断された場合、一部のオブジェクトで発生することがあります。
- **LaunchApplication** を呼び出したとき LAAW_OPTION_SET_BATCH_INSTALL が使用され、起動されたインストールの完了に再起動が必要だと関数で判別された。
- (**ServiceAddService** および関連する関数を使用して) インストールが Windows サービスを更新しようと試みたが、既存のサービスを終了できなかった。
- DIFx ドライバ のインストールにより、DIFx の統合で、再起動が必要であることが示された。

BATCH_INSTALL が FALSE に設定されている場合は、ロックされたファイルは見つからなかったこと、および、インストール プロセスが正常に終了できることを意味します。

詳しい情報は、「インストールまたはアンインストールがターゲット システムを再起動するタイミングを理解する」をご覧ください。

CMDLINE

CMDLINE 変数は、プロジェクト タイプに応じて異なります。

InstallScript プロジェクトの CMDLINE

InstallScript プロジェクトでは、**Setup.exe** はユーザー定義のコマンドライン引数をすべて受け入れ、ランタイムでシステム文字列変数の CMDLINE に割り当てます。

メモ・InstallScript プロジェクトにおける CMDLINE について、以下の点にご注意ください:

- *CMDLINE は、ユーザー定義のコマンドライン引数のみを格納します。InstallShield コマンドライン引数 (定義済み引数) は、CMDLINE にコピーされません。*
- *インストールは、ユーザー定義のコマンドライン引数を CMDLINE にコピーする際にすべての文字を小文字に変換します。CMDLINE を処理する際には、大文字と小文字を区別しないロジックを使用してください。*

アドバンスト UI およびスイート / アドバンスト UI プロジェクトに InstallScript パッケージとして含まれている InstallScript インストールの CMDLINE

データは、InstallScript パッケージを含むアドバンスト UI またはスイート / アドバンスト UI プロジェクトのパッケージに対する " コマンドライン " 設定と " サイレント コマンドライン " 設定で InstallScript パッケージに渡すことができます。データは、アドバンスト UI またはスイート / アドバンスト UI のスクリプト イベントで CMDLINE 変数から解析されるようになります。詳細については、「OnSuiteShowUI」を参照してください。

InstallScript MSI プロジェクトの CMDLINE

InstallScript MSI プロジェクトでは、/z スイッチを使って **Setup.exe** へ渡されたコマンドライン データはすべてシステム文字列値 CMDLINE に格納されます。例えば、ユーザーが次のコマンドラインを実行したとき、CMDLINE は " **カスタム データ** " 文字列に設定されます。

Setup.exe /z" カスタム データ "

基本の MSI プロジェクトの CMDLINE

基本の MSI プロジェクトでは、/v コマンドライン引数 を使って **Setup.exe** を通して **Msiexec.exe** へパブリック プロパティを渡すことができます。

COMMONFILES

COMMONFILES システム変数には、システムにインストールされたアプリケーションが共有するファイルを保存する、Windows で定義されたフォルダーの完全修飾名が含まれています。英語版 Windows では、このフォルダーは Common Files という名前で、Program ファイル フォルダーにあります。(他の言語版の Windows では、Common Files フォルダー名はデフォルトでローカライズされた名前に設定されています)。Common Files フォルダーは、アプリケーションが共有するファイルとフォルダーのためのデフォルトの場所としてお勧めします。

64 ビット Windows システムで、このフォルダーが 32 ビットのアプリケーションの共通ファイルを格納し、64 ビット アプリケーションの共有ファイルは COMMONFILES64 フォルダーにインストールする必要があります。

プロジェクト・*InstallScript インストールでセットアップの初期化中、COMMONFILES 変数の値は、Windows API 関数* ***SHGetSpecialFolderPath*** *を CSIDL_COMMON_FILES パラメーターと共に呼び出して取得します。*

基本の MSI と InstallScript MSI インストールでは、COMMONFILES 変数の値は Windows Installer プロパティ ***CommonFilesFolder*** *に基づいて初期化されます。遅延、コミット、およびロールバック カスタム アクションは、このプロパティーにアクセスすることはできません。そのため、対応する COMMONFILES 変数は、遅延、コミット、およびロールバック カスタム アクションでは空白です。詳細は、「遅延／コミット／ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。*

COMMONFILES64

COMMONFILES64 ステム変数には、64 ビットアプリケーションで共有されているファイルを保存する、Windows で 定義されたフォルダーの完全修飾名が含まれています。英語版 Windows では、このフォルダーは Common Files という名前で、PROGRAMFILES64 フォルダー にあります。(他の言語版の Windows では、Common Files フォルダー名はデフォルトでローカライズされた名前に設定されています)。Common Files フォルダーは、アプリケーションが共有するファイルとフォルダーのためのデフォルトの場所としてお勧めします。

プロジェクト・*InstallScript インストールでセットアップの初期化中、COMMONFILES64 変数の値は、Windows API 関数* ***SHGetSpecialFolderPath*** *を CSIDL_COMMON_FILES パラメーターと共に呼び出して取得します。*

基本の MSI と InstallScript MSI インストールでは、COMMONFILES64 変数の値は Windows Installer プロパティ ***CommonFiles64Folder*** *に基づいて初期化されます。遅延、コミット、およびロールバック カスタム アクションは、このプロパティーにアクセスすることはできません。そのため、対応する COMMONFILES64 変数は、遅延、コミット、およびロールバック カスタム アクションでは空白です。詳細は、「遅延／コミット／ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。*

DISK1SETUPEXENAME

DISK1SETUPEXENAME は読み取り専用のシステム変数で、ファイル名とファイル名の拡張子が含まれていますが、セットアップランチャー、インストーラーの実行可能ファイルのパスは含まれていません。デフォルト値は *Setup.exe* です。

プロジェクト・*DISK1SETUPEXENAME は、InstallScript プロジェクトでの使用が目的とされています。この変数を別のプロジェクトの種類で使用した場合、変数が予定通り設定されず、異なる動作やシナリオの原因となります。*

DISK1TARGET

このシステム変数には、メンテナンスインストールとアンインストールを実行するための特定のインストレーションファイル (コンパイル済みスクリプトファイルなど) のコピーが配置されているフォルダーへのパスが含まれます。

ENABLED_ISERVICES

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

現在有効な InstallShield サービスを示すビットフラグのセットを含むシステム変数。たとえば、式 **ENABLED_ISERVICES & SERVICE_FLAG_ISFONTREG** がゼロ以外の値に等しい場合、グローバル フォント登録が現在有効になっています。

ENGINECOMMONDIR

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ENGINECOMMONDIR システム変数は、システムで実行中の 6.x、7.x、そして 9.x InstallScript セットアップすべて (InstallScript MSI 以外) が利用するランタイムファイルを含むフォルダーへの完全修飾パスを格納します。

このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。

ENGINEDIR

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ENGINEDIR システム変数はセットアップで利用されているエンジンのバージョンに特有のランタイムファイルを含むフォルダーへの完全修飾パスを格納します。これは InstallShield Professional 6.x セットアップによるものではなく、7.00、7.01、または 9.00 です。

このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。

ERRORFILENAME

このシステム変数は、エラーを含むファイル名を格納します。たとえば、ビルトイン関数で特定のファイルをコピーしている間にエラーが生じた場合、InstallShield では、ERRORFILENAME にエラーを生じたファイル名を設定します。すべてのファイル処理関数で ERRORFILENAME を使用するわけではありません。

FOLDER_APPDATA

FOLDER_APPDATA システム変数は、オペレーティングシステムで定義され、アプリケーション特有データの共有リポジトリとして利用されるフォルダーへの完全修飾名を格納します。

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。

プロジェクト・*InstallScript インストールでセットアップの初期化中、FOLDER_APPDATA 変数の値は、Windows API 関数* ***SHGetSpecialFolderPath*** *を LPITEMIDLIST に CSIDL_APPDATA 値を使って呼び出して取得します。*

基本の MSI と InstallScript MSI インストールでは、FOLDER_APPDATA 変数の値は Windows Installer プロパティ ***AppDataFolder*** *または* ***LocalAppDataFolder*** *に基づいて初期化されます。遅延、コミット、およびロールバック カスタム アクションは、これらのプロパティーにアクセスすることはできません。そのため、対応する FOLDER_APPDATA 変数は、遅延、コミット、およびロールバック カスタム アクションでは空白です。詳細は、「遅延／コミット／ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。*

FOLDER_APPLICATIONS

FOLDER_APPLICATIONS システム変数はアプリケーション フォルダーのルートへの完全修飾パスを格納します。システム変数 ALLUSERS の値がゼロ以外の場合、このシステム変数の値はシステム変数 PROGRAMFILES の値に等しくなります。ALLUSERS が FALSE の場合、このシステム変数の値はシステム変数 FOLDER_APPDATA の値に等しくなります。

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。

FOLDER_APPLICATIONS64

FOLDER_APPLICATIONS64 システム変数は、64 ビットシステムで、アプリケーション フォルダーのルートへの完全修飾パスを格納します。システム変数 ALLUSERS の値がゼロ以外の場合、このシステム変数の値はシステム変数 PROGRAMFILES64 の値に等しくなります。ALLUSERS が FALSE の場合、このシステム変数の値はシステム変数 FOLDER_APPDATA の値に等しくなります。

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。

FOLDER_COMMON_APPDATA

FOLDER_COMMON_APPDATA システム変数は、オペレーティング システムで定義され、アプリケーション特有データの共有リポジトリとして利用されるフォルダーへの完全修飾名を格納します。

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。

プロジェクト・*InstallScript インストールでセットアップの初期化中、FOLDER_COMMON_APPDATA 変数の値は、Windows API 関数* ***SHGetSpecialFolderPath*** *を LPITEMIDLIST に CSIDL_COMMON_APPDATA 値を使って呼び出して取得します。*

基本の MSI と InstallScript MSI インストールでは、FOLDER_COMMON_APPDATA 変数の値は Windows Installer プロパティ ***CommonAppDataFolder*** *に基づいて初期化されます。遅延、コミット、およびロールバック カスタム アクションは、このプロパティーにアクセスすることはできません。そのため、対応する FOLDER_COMMON_APPDATA 変数は、遅延、コミット、およびロールバック カスタム アクションでは空白です。詳細は、「遅延／コミット／ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。*

FOLDER_DESKTOP

FOLDER_DESKTOP システム変数は Desktop フォルダーへの完全修飾パスを格納します。Desktop フォルダーは、プログラム フォルダーとエンド ユーザーのデスクトップに表示される項目を保持しています。

グループとフォルダーが確実に適切な場所に作成されるように、システム変数 ALLUSERS が変更されるとき、デフォルト グループまたはフォルダーの種類が [共通] から [個人]、あるいは [個人] から [共通] に変更された場合に、FOLDER_DESKTOP がポイントする場所も変更されます。

FOLDER_DOTNET_10

FOLDER_DOTNET_10 システム変数は、Microsoft .NET Framework 1.0 再配布可能ファイルがある場所にフォルダーの完全修飾パスを格納します :

<WINDIR>¥Microsoft.NET¥Framework¥v1.0.3705¥

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラ エラーの原因となります。

FOLDER_DOTNET_11

The FOLDER_DOTNET_11 システム変数は、Microsoft .NET Framework 1.1 再配布可能ファイルがある場所にフォルダーの完全修飾パスを格納します :

<WINDIR>¥Microsoft.NET¥Framework¥v1.1.4322¥

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラ エラーの原因となります。

FOLDER_DOTNET_20

The FOLDER_DOTNET_20 システム変数は、Microsoft .NET Framework 2.0 再配布可能ファイルがある場所にフォルダーの完全修飾パスを格納します :

<WINDIR>¥Microsoft.NET¥Framework¥v2.0.50727¥

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラ エラーの原因となります。

FOLDER_DOTNET_30

The FOLDER_DOTNET_30 システム変数は、Microsoft .NET Framework 3.0 再配布可能ファイルがある場所にフォルダーの完全修飾パスを格納します :

<WINDIR>¥Microsoft.NET¥Framework¥v3.0

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラ エラーの原因となります。

FOLDER_DOTNET_35

FOLDER_DOTNET_35 システム変数は、Microsoft .NET Framework 3.5 再配布可能ファイルがある場所にフォルダーの完全修飾パスを格納します :

<WINDIR>¥Microsoft.NET¥Framework¥v3.5

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラ エラーの原因となります。

FOLDER_DOTNET_40

FOLDER_DOTNET_40 システム変数は、Microsoft .NET Framework 4.0 再配布可能ファイルがある場所にフォルダーの完全修飾パスを格納します :

<WINDIR>¥Microsoft.NET¥Framework¥v4.0.30319

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラ エラーの原因となります。

FOLDER_FONTS

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript オブジェクト*

FOLDER_FONTS システム変数は Windows フォントフォルダーの完全修飾パスを格納します。

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。

FOLDER_LOCAL_APPDATA

FOLDER_COMMON_APPDATA システム変数は、オペレーティング システムで定義され、アプリケーション特有データの共有リポジトリとして利用されるフォルダーへの完全修飾名を格納します。一般的は値は、**C:¥Users¥<User>¥Application Data** (Windows Visa 以降の場合)、および **C:¥Documents and Settings¥<User>¥Application Data** (Windows Vista 以前のシステムの場合) です。

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。

プロジェクト・*InstallScript インストールでセットアップの初期化中、FOLDER_COMMON_APPDATA 変数の値は、Windows API 関数* ***SHGetSpecialFolderPath*** *を LPITEMIDLIST に CSIDL_LOCAL_APPDATA 値を使って呼び出して取得します。*

基本の MSI と InstallScript MSI インストールでは、FOLDER_LOCAL_APPDATA 変数の値は Windows Installer プロパティ ***LocalAppDataFolder*** *に基づいて初期化されます。遅延、コミット、およびロールバック カスタム アクションは、このプロパティーにアクセスすることはできません。そのため、対応する FOLDER_LOCAL_APPDATA 変数は、遅延、コミット、およびロールバック カスタム アクションでは空白です。詳細は、「遅延／コミット／ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。*

FOLDER_PERSONAL

プロジェクト・*この情報は、InstallScript プロジェクトに適用します。*

FOLDER_PERSONAL システム変数は、オペレーティングシステムで定義され、アプリケーション特有データの共有リポジトリとして利用されるフォルダーへの完全修飾名を格納します。一般的は値は、**C:¥Users¥<User>¥Application Data** (Windows Visa 以降の場合)、および **C:¥Documents and Settings¥<User>¥Application Data** (Windows Vista 以前のシステムの場合) です。

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。

プロジェクト・*InstallScript インストールでセットアップの初期化中、FOLDER_PERSONAL 変数の値は、Windows API 関数* ***SHGetSpecialFolderPath*** *を LPITEMIDLIST に CSIDL_PERSONAL 値を使って呼び出して取得します。*

基本の MSI と InstallScript MSI インストールでは、FOLDER_PERSONAL 変数の値は Windows Installer プロパティ ***PersonalFolder*** *に基づいて初期化されます。遅延、コミット、およびロールバック カスタム アクションは、このプロパティーにアクセスすることはできません。そのため、対応する FOLDER_PERSONAL 変数は、遅延、コミット、およびロールバック カスタム アクションでは空白です。詳細は、「遅延／コミット／ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。*

FOLDER_PROGRAMS

FOLDER_PROGRAMS システム変数は、スタート メニューのプログラム フォルダーへの完全修飾パスを格納します。スタート メニューからプログラムを選択する際に表示されます。

グループとフォルダーが確実に適切な場所に作成されるように、システム変数 ALLUSERS が変更されるとき、デフォルト グループまたはフォルダーの種類が [共通] から [個人]、あるいは [個人] から [共通] に変更された場合に、FOLDER_PROGRAMS がポイントする場所も変更されます。

FOLDER_STARTMENU

FOLDER_STARTMENU システム変数は、スタート メニュー フォルダーへの完全修飾パスを格納します。スタート メニュー フォルダーは、Windows の [スタート] ボタンをクリックすると表示されます。

グループとフォルダーが確実に適切な場所に作成されるように、システム変数 ALLUSERS が変更されるとき、デフォルト グループまたはフォルダーの種類が [共通] から [個人]、あるいは [個人] から [共通] に変更された場合に、FOLDER_STARTMENU がポイントする場所も変更されます。

FOLDER_STARTUP

FOLDER_STARTUP システム変数は、スタートアップ フォルダーへの完全修飾パスを格納します。スタートアップ フォルダーには、Window と同時に起動されるプログラム フォルダーや各種項目が格納されています。

グループとフォルダーが確実に適切な場所に作成されるように、システム変数 ALLUSERS が変更されるとき、デフォルト グループまたはフォルダーの種類が [共通] から [個人]、あるいは [個人] から [共通] に変更された場合に、FOLDER_STARTUP がポイントする場所も変更されます。

FOLDER_TEMP

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FOLDER_TEMP システム変数は、一時ファイルとして利用されるフォルダーの完全修飾名を格納します。このフォルダーは Windows 並びにシステム上の殆どのアプリケーションが利用するので、インストールが作成または削除することはありません。(システム変数 SUPPORTDIR にパスが格納されているフォルダーは、インストールの特有ファイルを格納するためにインストールによって作成され、インストールが完了した後に削除されます。)

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。

プロジェクト・InstallScript インストールでセットアップの初期化中、FOLDER_TEMP 変数の値は、Windows API 関数 GetTempPath を呼び出して取得します。

*基本の MSI と InstallScript MSI インストールでは、FOLDER_TEMP 変数の値は Windows Installer プロパティ **TempFolder** に基づいて初期化されます。遅延、コミット、およびロールバック カスタム アクションは、このプロパティーにアクセスすることはできません。そのため、対応する FOLDER_TEMP 変数は、遅延、コミット、およびロールバック カスタム アクションでは空白です。詳細は、「遅延／コミット／ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。*

HKEYCURRENTROOTKEY

このシステム変数の値はレジストリ関連の一般関数が使用するルートキーです。システム変数として使われる値は次の通りです。

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_USER

- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_CONFIG
- HKEY_DYN_DATA

HKEYCURRENTROOTKEY を前回定義済みの定数の 1 つ、定義済み定数 HKEY_USER_SELECTABLE、またはシステム変数 HKEY_USER_SELECTABLE_AUTO に等しく設定することでデフォルトルートキーを設定することができます。

RegDBGetDefaultRoot とは異なり、HKEYCURRENTROOTKEY の値が HKEY_USER_SELECTABLE となることはありません。一番最後にデフォルトキーを設定したときに HKEY_USER_SELECTABLE を使用した場合、HKEYCURRENTROOTKEY の値は ALLUSERS システム変数がゼロ以外の時は HKEY_LOCAL_MACHINE に、また ALLUSERS が FALSE の時は HKEY_CURRENT_USER となります。

HKEY_USER_SELECTABLE_AUTO

このシステム変数の値は ALLUSERS システム変数がゼロ以外の場合は HKEY_LOCAL_MACHINE、また ALLUSERS が FALSE の場合は HKEY_CURRENT_USER です。

IFX_COMPANY_NAME

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数は、文字列エントリ COMPANY_NAME が存在する場合はその値へ自動的に初期化されます。このエントリが存在しない場合、IFX_COMPANY_NAME は [プロジェクト設定] プロパティ シートの [アプリケーション] ページ で指定した会社名に初期化されます。

IFX_DISK1INSTALLED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

システム変数はセットアップの始まりではゼロと等しく設定されており、セットアップがメンテナンスセットアップまたはアンインストールに必要なファイルと共に機能をインストール、または再インストールする場合にはゼロ以外の値にリセットされます。(この機能はメディア ビルダーによって .cab ファイルに自動的に配置され、IDE には表示されません。)

IFX_INITIALIZED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

セットアップがイベント型の場合、システム変数はゼロ以外の値に等しく設定され、セットアップが手続き型の場合 (program...endprogram ブロックを含む場合) は FALSE に設定されます。

IFX_INSTALLED_DISPLAY_VERSION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

IFX_INSTALLED_DISPLAY_VERSION システム変数は標準ダイアログのスタティック テキストフィールドのプレースホルダー %VI、および **SdSubstituteProductInfo** 関数へ渡す文字列を置換します。このシステム変数は IFX_INSTALLED_VERSION の値へ自動的に初期化されます。新しい値を IFX_INSTALLED_VERSION へ割り当てる場合、IFX_INSTALLED_DISPLAY_VERSION の値は自動的には変更はされません。

IFX_INSTALLED_VERSION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数のデータがパックされた DWORD の場合、自動的にアプリケーションのアンインストールレジストリキーのバージョン値のデータに対応する文字列へ初期化されます。キーまたは値が存在しない、あるいはデータがパックされていない　DWORD の場合、IFX_INSTALLED_VERSION はヌル文字列 (“”) へ初期化されます。

IFX_KEYPATH_PRODUCT_INFO

このシステム変数は CreateInstallationInfo が作成したアプリケーション情報キーのレジストリロケーションを指定し、その値は RegDBGetAppInfo によって読み取られ、RegDBSetAppInfo によって変更されます。このシステム変数は次の値に初期化されます。

Software¥<IFX_COMPANY_NAME>¥<IFX_PRODUCT_NAME>¥<IFX_PRODUCT_VERSION>¥.

IFX_MULTI_INSTANCE_SUFFIX

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

IFX_MULTI_INSTANCE_SUFFIX システム変数は、OnFirstUIBefore イベントハンドラー関数のデフォルトコードで設定されています。IFX_MULTI_INSTANCE_SUFFIX は、複数インスタンス インストールに一意のターゲットフォルダー名を構築するハンドラー関数で使用されます。複数インスタンス インストールに一意のアンインストール表示名を構築する OnCustomizeUninstInfo ハンドラー関数でも使用されます。

IFX_PRODUCT_COMMENTS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数の値がヌル文字列 (“”) ではない場合、MaintenanceStart 関数が この値を利用してアプリケーションアンインストール レジストリキーの Comments 値にデータを指定します。このレジストリ値は コントロール パネルの [プログラムの追加と削除] にアプリケーションについての情報を提供します。

このシステム変数は [一般情報] ビュー の ″APR コメント ″ 設定で指定した値に初期化されます。

IFX_PRODUCT_DISPLAY_NAME

このシステム変数は標準ダイアログ静的テキストフィールドのプレースホルダー %P、そして **SdSubstituteProductInfo** 関数へ渡す文字列を置換します。このシステム変数は IFX_PRODUCT_NAME の値へ自動的に初期化されます。新しい値を IFX_PRODUCT_NAME へ割り当てる場合、IFX_PRODUCT_DISPLAY_NAME の値は自動的には変更はされません。

***メモ・システム変数** IFX_SETUP_TITLE は、ビルトイン ダイアログのタイトル バーにあるテキストを指定します。*

IFX_PRODUCT_DISPLAY_VERSION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数は標準ダイアログ静的テキストフィールドのプレースホルダー %VS、そして **SdSubstituteProductInfo** 関数へ渡す文字列を置換します。このシステム変数は IFX_PRODUCT_VERSION の値へ自動的に初期化されます。新しい値を IFX_PRODUCT_VERSION へ割り当てる場合、IFX_PRODUCT_DISPLAY_VERSION の値は自動的には変更はされません。

IFX_PRODUCT_ICON

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数の値がヌル文字列 (″″) ではない場合、MaintenanceStart 関数が この値を利用してアプリケーションアンインストール レジストリキーの DisplayIcon 値にデータを指定します。このレジストリ値は、コントロール パネルの [プログラムの追加と削除] でアプリケーションについて表示されるアイコンを指定します。

このシステム変数は [一般情報] ビューの [プログラムの追加と削除] 領域の ″ アイコンの表示 ″ 設定で指定した値に初期化されます。

IFX_PRODUCT_KEY

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数は、文字列エントリ PRODUCT_KEY が存在する場合はその値へ自動的に初期化されます。このエントリが存在しない場合、IFX_PRODUCT_KEY は [プロジェクト設定] プロパティ シートの [アプリケーション] ページで指定した実行可能ファイル名に初期化されます。

IFX_PRODUCT_NAME

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数は、文字列エントリ PRODUCT_NAME が存在する場合はその値へ自動的に初期化されます。このエントリが存在しない場合、IFX_PRODUCT_NAME は [プロジェクト設定] プロパティ シートの [アプリケーション] ページ で指定した製品名に初期化されます。

IFX_PRODUCT_README

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数の値がヌル文字列 (″″) ではない場合、MaintenanceStart 関数が この値を利用してアプリケーションアンインストール レジストリキーの Readme 値にデータを指定します。このレジストリ値は コントロール パネルの [プログラムの追加と削除] にアプリケーションについての情報を提供します。

このシステム変数は [一般情報] ビュー の ″README″ 設定で指定した値に初期化されます。

IFX_PRODUCT_REGISTEREDCOMPANY

システム変数 *IFX_PRODUCT_REGISTEREDCOMPANY* の値がヌル文字列 (″″) ではない場合、MaintenanceStart 関数がこの値を利用してアプリケーション アンインストール レジストリ キーの RegCompany 値にデータを指定します。このレジストリ値は コントロール パネルの [プログラムの追加と削除] にアプリケーションについての情報を提供します。

このシステム変数はレジストリ値 HKEY_LOCAL_MACHINE¥Software¥Microsoft¥*Windows* キー ¥CurrentVersion¥RegisteredOrganization のデータに初期化されます。この *Windows* キー は、ターゲットオペレーティングシステムが Windows XP 以降の場合は Windows NT で、その他の Windows オペレーティングシステムの場合は Windows です。このシステム変数の値はエンド ユーザーが SdRegisterUser、SdRegisterUserEx、SdCustomerInformation および SdCustomerInformationEx ダイアログの [会社名] 編集ボックスで入力した値によって変更されます。

プロジェクト・InstallScript MSI インストールの場合、IFX_PRODUCT_REGISTEREDCOMPANY の値が設定されると、Windows Installer プロパティ COMPANYNAME が自動的に更新されます。

IFX_PRODUCT_REGISTEREDOWNER

システム変数 *IFX_PRODUCT_REGISTEREDOWNER* の値がヌル文字列 (″″) ではない場合、MaintenanceStart 関数がこの値を利用してアプリケーション アンインストール レジストリ キーの RegOwner 値にデータを指定します。このレジストリ値は コントロール パネルの [プログラムの追加と削除] にアプリケーションについての情報を提供します。

このシステム変数はレジストリ値 HKEY_LOCAL_MACHINE¥Software¥Microsoft¥*Windows* キー ¥CurrentVersion¥RegisteredOwner のデータに初期化されます。この *Windows* キー は、ターゲットオペレーティングシステムが Windows XP 以降の場合は Windows NT で、その他の Windows オペレーティングシステムの場合は

Windows です。このシステム変数の値はエンドユーザーが SdRegisterUser、SdRegisterUserEx、SdCustomerInformation および SdCustomerInformationEx ダイアログの [ユーザー名] 編集ボックスで入力した値によって変更されます。

プロジェクト・*InstallScript MSI インストールの場合、IFX_PRODUCT_REGISTEREDOWNER の値が設定されると、Windows Installer プロパティ* ***USERNAME*** *が自動的に更新されます。*

IFX_PRODUCT_REGISTEREDSERIALNUM

システム変数 *IFX_PRODUCT_REGISTEREDSERIALNUM* の値がヌル文字列 ("") ではない場合、MaintenanceStart 関数がこの値を利用してアプリケーション アンインストール レジストリ キーの ProductId 値にデータを指定します。このレジストリ値は コントロール パネルの [プログラムの追加と削除] にアプリケーションについての情報を提供します。

このシステム変数の値はエンドユーザーが SdCustomerInformation および SdCustomerInformationEx ダイアログの [シリアル番号] 編集ボックスに入力した値によって変更されます。

IFX_PRODUCT_SUPPORT_CONTACT

プロジェクト・*この情報は、InstallScript プロジェクトに適用します。*

このシステム変数の値がヌル文字列 ("") ではない場合、MaintenanceStart 関数が この値を利用してアプリケーションアンインストール レジストリキーの Contact 値にデータを指定します。このレジストリ値は コントロール パネルの [プログラムの追加と削除] にアプリケーションについての情報を提供します。

このシステム変数は [一般情報] ビュー の " サポート連絡先 " 設定で指定した値に初期化されます。

IFX_PRODUCT_SUPPORT_PHONE

プロジェクト・*この情報は、InstallScript プロジェクトに適用します。*

このシステム変数の値がヌル文字列 ("") ではない場合、MaintenanceStart 関数が この値を利用してアプリケーションアンインストール レジストリキーの HelpTelephone 値にデータを指定します。このレジストリ値は コントロール パネルの [プログラムの追加と削除] にアプリケーションについての情報を提供します。

このシステム変数は [一般情報] ビュー の " サポート電話番号 " 設定で指定した値に初期化されます。

IFX_PRODUCT_SUPPORT_URL

プロジェクト・*この情報は、InstallScript プロジェクトに適用します。*

このシステム変数の値がヌル文字列 ("") ではない場合、MaintenanceStart 関数が この値を利用してアプリケーションアンインストール レジストリキーの HelpLink 値にデータを指定します。このレジストリ値は コントロール パネルの [プログラムの追加と削除] にアプリケーションについての情報を提供します。

このシステム変数は [一般情報] ビュー の " サポート URL" 設定で指定した値に初期化されます。

IFX_PRODUCT_UPDATE_URL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数の値がヌル文字列 ("") ではない場合、MaintenanceStart 関数が この値を利用してアプリケーションアンインストール レジストリキーの URLUpdateInfo 値にデータを指定します。このレジストリ値は コントロール パネルの [プログラムの追加と削除] にアプリケーションについての情報を提供します。

このシステム変数は [一般情報] ビュー の " 製品アップデート URL" 設定で指定した値に初期化されます。

IFX_PRODUCT_URL

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数の値がヌル文字列 ("") ではない場合、MaintenanceStart 関数が この値を利用してアプリケーションアンインストール レジストリキーの URLInfoAbout 値にデータを指定します。このレジストリ値は コントロール パネルの [プログラムの追加と削除] にアプリケーションについての情報を提供します。

このシステム変数は [一般情報] ビュー の " 発行元 / 製品 URL" 設定で指定した値に初期化されます。

IFX_PRODUCT_VERSION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数は、文字列エントリ PRODUCT_VERSION が存在する場合はその値へ自動的に初期化されます。このエントリが存在しない場合、IFX_PRODUCT_VERSION は [プロジェクト設定] プロパティシートの [アプリケーション] ページ で指定した製品バージョンに初期化されます。

IFX_SETUP_TITLE

このシステム変数はビルトイン ダイアログ (Windows API 関数が直接生成するダイアログ以外)、そして MessageBox 関数が生成するすべてのメッセージ ボックスのタイトル バーにあるテキストを指定します。このシステム変数は文字列エントリ TITLE_CAPTIONBAR の値へ自動的に初期化されます。このエントリが存在しない場合、IFX_SETUP_TITLE は次の内部コードと共に初期化されます :

```
Sprintf( IFX_SETUP_TITLE, SdLoadString( IDS_IFX_FORMAT_SETUP_TITLE ), IFX_PRODUCT_DISPLAY_NAME );
```

IFX_SETUP_TITLE の値を変更すると、セットアップが表示するすべてのダイアログのタイトルが自動的にリセットされます。

IFX_SUPPORTED_VERSIONS

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数は、メディアプロパティシートの [アップデート] ページまたは [メディアウィザード] の [アップデート] パネルで指定した、アップデートを適用する製品バージョンの垂直線 (|) 区切りリストへ自動的に初期化されます。

INFOFILENAME

BatchFileSave を使用してバッチファイルを保存したり、ConfigFileSave を使用して Config.sys ファイルを保存する場合、更新を行う前の状態でファイルのバックアップを作成するよう指定できます。InstallShield は、このバックアップ ファイルの名前を、システム変数 INFOFILENAME に割り当てます。バックアップファイルがあることをユーザーに警告するには、MessageBox 関数を使用して INFOFILENAME の値を表示します。

INSTALLDIR

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *基本の MSI*
- *InstallScript MSI*

InstallScript プロジェクトでは、TARGETDIR を使用します。

セットアップの初期化中に、インストールはシステム変数 INSTALLDIR にハード ドライブ上のターゲット フォルダーへの完全修飾パスを割り当てます。INSTALLDIR パスは、[一般情報] ビューで INSTALLDIR 設定に指定されたインストール先に基づいて解決されます。デフォルトでは、INSTALLDIR パスは .msi パッケージの **Directory** テーブル内の [ProgramFilesFolder]ISYourCompanyDir¥ISYourProductDir のエントリに基づいて解決されます。

INSTANCE_GUID

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数にはセットアップの GUID (Globally Unique Identifier) を含みます。これはアプリケーションのアンインストールレジストリキーの名前として利用されます。この変数の値は、複数インスタンス インストール以外はランタイムに PRODUCT_GUID と等しく設定されます。

このシステム変数の値は、複数のオブジェクトスクリプト間およびオブジェクトスクリプトとメインのインストールスクリプトとの間で共有されます。新しい値をこのシステム変数に割り当てることはできません。

ISDIFXAPPID

この定義済みグローバルシステム変数は、デバイス ドライバーをインストールまたはアンインストールする場合に関連付けるアプリケーションを決定します。ISDIFXAPPID は初期化中はデフォルトで PRODUCT_GUID に設定されており、必要に応じて変更して代替アプリケーション ID を指定することができます。

メモ・アプリケーションの関連付けを指定する方法について詳細は、DIFxAPI マニュアルの INSTALLERINFO 構造を参照してください。

ISMSI_HANDLE

このシステム変数は現在実行中の .msi データベースのハンドルへ設定されており、現在実行中のデータベースへのハンドルを必要とする Windows Installer API 関数への引数としてイベントハンドラー関数で利用することが可能です。

たとえば、OnBegin イベント ハンドラー内の USERNAME プロパティの値を読み取るには、次のようなコードを利用することができます :

```
function OnBegin()
    STRING svUsername[256];
    NUMBER nBuffer;
begin
    nBuffer = 256;
    MsiGetProperty(ISMSI_HANDLE, "USERNAME", svUsername, nBuffer);
    MessageBox("USERNAME = " + svUsername, INFORMATION);
end;
```

プロジェクト・ISMSI_HANDLE は基本の MSI プロジェクト、また InstallScript カスタムアクションではサポートされていません。

IS_NULLSTR_PTR

IS_NULLSTR_PTR 変数を使って、ヌル ポインターを InstallScript 文字列としてプロトタイプされているパラメーターを通して 外部 DLL 関数または Windows API に渡すことができます。この機能は byval 文字列、byref 文字列、wstring、およびバイナリ データ タイプで使用できます。

この機能は、byref number パラメーターには適用しません。NULL ポインターを byref number パラメーターに指定する場合、そのパラメーターをポインター データ型にプロトタイプ化して、必要に応じて数値変数のアドレスまたは NULL を渡さなくてはなりません。

IS_NULLSTR_PTR は、IS_NULLSTR_PTR を値に持つグローバル文字列変数のインスタンスです。ただし、新しい値をこの変数に割り当てるためのステートメントはコンパイルしますが、割り当てはその効果を持ちません。変数の値は IS_NULLSTR_PTR のままです。

この変数を非 DLL 関数に渡すと、関数は文字列 <IS_NULLSTR_PTR> を受け取ります。

値 IS_NULLSTR_PTR を持つ文字列を外部 DLL 関数に渡すと、IS_NULLSTR_PTR を使用したときと同じ結果となります。

IS_NULLSTR_PTR 変数を使って Windows API にヌル ポインターを渡す

Windows 関数 **WritePrivateProfileString** を使い、最初の 3 つのパラメーターに NULL を指定して Windows 9x 上の INI ファイル バッファーをフラッシュできます。ただし、この関数は以下のようにプロトタイプ化されているため、この目的を達成することは不可能に思われます。

prototype number　KERNEL32.WritePrivateProfileString (byval string, byval string, byval string, byval string);

ポインター データ型を使って、NULL を指定できますが、有効な文字列を指定するときに問題が起こります。

InstallScript エンジンを使ってヌル ポインターを関数に渡す場合、以下のコードを使用します :

KERNEL32.WritePrivateProfileString (IS_NULLSTR_PTR, IS_NULLSTR_PTR, IS_NULLSTR_PTR, szFile);

IS_NULLSTR_PTR 変数を使って 外部 DLL 関数ににヌル ポインターを渡す

IS_NULLSTR_PTR は、文字列を受け付ける外部 DLL 関数と共に使用できます。この場合、DLL 関数は NULL ポインターを受け取ります。

ISRES

セットアップを初期化すると、インストールによって **_isres.dll** がセットアップから圧縮解除され、ターゲットシステムの一時フォルダーにコピーされます。ファイルには固有の名前が付けられるので、他の InstallShield インストールと競合することはありません。このファイルにはセットアップリソースが含まれており、その完全修飾名は ISRES システム変数に割り当てられます。

ISUSER

セットアップの初期化中、インストールは **_isuser.dll** が存在する場合にそれをセットアップから圧縮解除し、ターゲットシステムの一時フォルダー SUPPORTDIR にコピーします。ファイルには一意の名前が付けられるので、他の InstallShield インストールと競合することはありません。このファイルにはユーザー定義セットアップリソースが含まれており、その完全修飾名は ISUSER システム変数に割り当てられます。

ISVERSION

セットアップ スクリプトの実行が開始されると、インストールが実行中の **Setup.exe** のバージョを取得して、それをシステム変数 ISVERSION に割り当てます。バージョン番号は Setup プログラムの [バージョン情報] ダイアログ ボックスにも表示されます。

LAAW_PARAMETERS

LAAW_OPTION_USE_SHELLEXECUTE なしで **LaunchApplication** を呼び出すか、**LaunchAppAndWait** または **LaunchApp** を呼び出すと、これらの関数は内部的に Windows API 関数 **CreateProcess** を呼び出します。LAAW_PARAMETERS 構造化された変数は **CreateProcess** の特定の引数、そして起動されたアプリケーションの実行中にテキスト ウィンドウを表示するかどうかを指定します。**CreateProcess** に関する詳細は、Windows API マニュアル を参照してください。

LAAW_PARAMETERS システム変数はセットアップの初期化中に LaunchAppAndWaitInitStartupInfo への呼び出しによって自動的に初期化されます。

テーブル 8 · LAAW_PARAMETERS

メンバー	説明
bCallbackEndedWait	コールバック関数が LAAW_CALLBACK_RETURN_END_WAIT を返したため、**WaitForApp** が待機を終了したことを示します。
bInheritHandles	**CreateProcess** へ対応する引数を設定します。
dwCreationFlags	**CreateProcess** へ対応する引数を設定します。
lpCurrentDirectory	**CreateProcess** へ対応する引数を設定します。このメンバーは **LaunchApplication** または **LaunchAndAppAndWait** の szDirectory パラメーターに設定されます。 lpCurrentDirectory の値を手動で設定しても、**LaunchApplication** または **LaunchAndAppAndWait** には反映されません。
lpEnvironment	**CreateProcess** へ対応する引数を設定します。
lpProcessAttributes	**CreateProcess** へ対応する引数を設定します。
lpThreadAttributes	**CreateProcess** へ対応する引数を設定します。
nCallbackInterval	このメンバーは、ミリ秒単位の間隔でコールバックを定義します。**LaunchApplicationInit** または **LaunchAppAndWaitInitStartupInfo** 関数を呼び出すと、デフォルトで 1000 (1 秒) に設定されます。
nLaunchResult	アプリケーションが起動されない場合、nLaunchResult メンバーは **CreateProcess** の後に呼び出した **GetLastError** の呼び出し結果を含みます。**LaunchApp**、**LaunchAppAndWait**、または **LaunchApplication** が成功し、LAAW_OPTION_WAIT オプションが指定されたとき、nLaunchResult メンバーは起動されたアプリケーションのリターンコードを含みます。
nTimeOut	**WaitForApplication** が呼び出されたとき、タイムアウトの値が内部的に **LaunchApplication** または **LaunchAndAppAndWait** によって使用されたことを示します。デフォルト値は INFINITE です。この値をカスタマイズして、**LaunchApplication** または **LaunchAndAppAndWait** の待機タイムアウトを設定できます。

テーブル 8 • LAAW_PARAMETERS (続き)

メンバー	説明
nTimeOutCheckInterval	アプリケーションを待機中、インストールが **WaitForApplication** (または 内部的に **WaitForApplication** を呼び出す **LaunchApplication** または **LaunchAndAppAndWait**) でタイムアウトの間隔が過ぎたかどうかを確認する頻度の間隔を示します。この値は、nTimeOut が INFINITE に設定されていて、かつ LAAW_USE_CALLBACK が指定されていないとき使用されません。LAAW_USE_CALLBACK が指定されている場合、タイムアウト / コールバックの確認する間隔は、LAAW_PARAMETERS.nTimeOutCheckInterval と LAAW_PARAMETERS.nCallbackInterval の 2 つの値の小さい方が使用されます。デフォルト値は 1000 です。
nWaitForInputIdleMax	Windows API **WaitForInputIdle** で、アプリケーションが初期化を完了するのを待機するときの最長時間 (ミリ秒) を指定します。この構造メンバーのデフォルト値は 2000 です。この値を 0 に設定して、アプリケーションが完了するのを待機し始める前に、アプリケーションの初期化を待機しないことを示すことができます。 **LaunchApplication** と **LaunchAndAppAndWait** は LAAW_OPTION_WAIT が指定されている場合のみアプリケーションの初期化を待機するため、この値は LAAW_OPTION_WAIT が指定されている場合のみ使用されます。
nWaitResult	**WaitForApplication** の呼び出しによって発生した最後の待機についての追加情報を示します。詳細については、「WaitForApplication」を参照してください。
szCommandLineResult	**CreateProcess** の内部呼び出しで lpCommandLine パラメーターとして使用された結果のコマンドラインが含まれます。このメンバーは、**LaunchApplication** または **LaunchAndAppAndWait** が呼び出されたときに挿入されるので、この関数が呼び出される前後に直接設定された値は効果を持ちません。また、**LaunchApplicationInit** または **LaunchAppAndWaitInitStartupInfo** 関数を呼び出したとき、このメンバーはヌル値 ("") に設定されます。
szStatusText	 ***プロジェクト*** *• szStatusText メンバーを、スイート / アドバンスト UI インストールで呼び出される InstallScript アクションに使用することはできません。* この番号がヌル文字列 ("") 以外の場合、起動されたアプリケーションの実行中にインストールがその内容を (SdShowMsg を呼び出して) テキストウィンドウに表示します。szStatusText は 4 キロバイト以上のデータを受け付けませんので注意してください。

LAAW_PROCESS_INFORMATION

LaunchApplication、LaunchAndAppAndWait、または **LaunchApp** を呼び出すと、この構造化された変数は起動した処理についての ID 情報を戻します。PROCESS_INFORMATION システム変数には、次のメンバーがあります。

テーブル 9 · LAAW_PROCESS_INFORMATION

メンバー	説明
hProcess	新規に作成された処理へのハンドル。このハンドルはプロセスオブジェクト上で動作するすべての関数の過程を指定するのに利用します。
hThread	新規プロセスのプライマリスレッドへのハンドル。このハンドルはプロセスオブジェクト上で動作するすべての関数のスレッドを指定するのに利用します。
dwProcessId	プロセスを識別するのに利用されるグローバルプロセス ID。この値はプロセスが作成された時点から終了される時点まで有効です。
dwThreadId	スレッドを識別するのに利用されるグローバルスレッド ID。この値はスレッドが作成された時点から終了される時点まで有効です。

LAAW_SHELLEXECUTEINFO

LAAW_SHELLEXECUTEINFO スクリプト変数は、**ShellExecuteEx** が呼び出されたとき **LaunchApplication** 関数によって使用される SHELLEXECUTEINFO 構造のインスタンスです。この構造のメンバーをカスタマイズして、**LaunchApplication** を LAAW_OPTION_USE_SHELLEXECUTE パラメーターと共に使用したときの動作を変更することができます。

SHELLEXECUTEINFO 構造

```
typedef SHELLEXECUTEINFO
begin
    int     cbSize;
    int     fMask;
    HWND    hwnd;
    pointer lpVerb;
    pointer lpFile;
    pointer lpParameters;
    pointer lpDirectory;
    int     nShow;
    HWND    hInstApp;
    pointer lpIDList;
    pointer lpClass;
    HWND    hkeyClass;
    int     dwHotKey;
```

```
    HWND    hIconMonitor;
    HWND    hProcess;
end;
```

LAAW_SHELLEXECUTEVERB

LAAW_SHELLEXECUTEVERB スクリプト変数は、**ShellExecuteEx** が呼び出されたとき **LaunchApplication** 関数によって使用される動詞を示す文字列です。デフォルト値は **open** です。デフォルトで、LAAW_SHELLEXECUTEINFO の lpVerb メンバーは、この文字列をポイントします。

ヒント・*Windows Vista 以降を実行中のシステム上で LAAW_OPTION_USE_SHELLEXECUTE を使用する場合で、完全な管理者アカウント（実行する実行可能ファイルを右クリックして［管理者として実行］をクリックした場合と似ています）を使ってアプリケーションを起動するとき、スクリプトで **LaunchApplication** を使用する前に LAAW_SHELLEXECUTEVERB を* **runas** *に設定します。*

```
LAAW_SHELLEXECUTEVERB = "runas";
```

これによって、起動するアプリケーションが関連設定を含むアプリケーション マニフェストを持っているかどうかにかかわらず、アプリケーションは確実に完全な管理者権限を使って実行されます。これによって、同意または資格情報を要求するユーザー アカウント制御 (UAC) のプロンプトが表示されることがあります。

Windows Vista 以前のオペレーティング システムを実行するマシン上で **runas** *が使用された場合、［別のユーザーとして実行］ダイアログ ボックスが表示されます。この動作は、実行する実行可能ファイルを右クリックして［別のユーザーとして実行］をクリックしたときと似ています。このダイアログ ボックスを使って、エンド ユーザーはアプリケーションを実行するために使用するユーザー アカウントを選択できます。*

LAAW_STARTUPINFO

起動した処理のために新しいウィンドウが作成された場合、**LaunchApplication**、LaunchAndAppAndWait、または **LaunchApp** を呼び出すと、LAAW_STARTUPINFO 構造化変数がメイン ウィンドウのプロパティを指定します。このシステム変数はインストールの初期化中に LaunchAppAndWaitInitStartupInfo への呼び出しによって自動的に初期化されます。

STARTUPINFO システム変数には次のメンバーがあります。

テーブル 10 ・ LAAW_STARTUPINFO

メンバー	説明
cb	構造のサイズをバイト単位で示します。
lpReserved	予約されています。このメンバーを NULL に設定します。

テーブル 10 • LAAW_STARTUPINFO (続き)

メンバー	説明
lpDesktop	デスクトップの名前のみを指定する、またはデスクトップの名前及びこの処理用のウィンドウステーションの名前の両方を指定するヌルで終わる文字列へのポインター。lpDesktop がポイントする文字列に含まれる円記号は、文字列にデスクトップ名とウィンドウステーション名の両方が含まれることを示します。lpDesktop が NULL の場合、新しい処理はデスクトップ及びウィンドウステーションの親処理を継承します。lpDesktop が空白文字列の場合、処理はデスクトップ及びウィンドウステーションの親処理を継承しません。その代わり、システムは新しいデスクトップ及びウィンドウステーションを作成するべきか否かを判断します。匿名ユーザーが既にデスクトップを持っている場合、システムは既存デスクトップを利用します。
lpTitle	コンソール プロセスでは、新しいコンソール ウィンドウが作成された場合、これがタイトル バーに表示されるタイトルとなります。ヌルの場合、代わりに実行可能ファイルの名前がウィンドウタイトルとして利用されます。新しいコンソール ウィンドウを作成しない GUI またはコンソール プロセスでは、このパラメーターは NULL でなくてはなりません。
dwX	dwFlags が STARTF_USEPOSITION を指定しない限り無視されます。新規ウィンドウが作成される場合、ウィンドウの左上の角の x オフセットをピクセル単位で指定します。オフセットは画面の左上の角からです。GUI プロセスでは、**CreateWindow** の x パラメーターが CW_USEDEFAULT の場合にオーバーラップしたウィンドウを作成する為、新しいプロセスが Windows API 関数の **CreateWindow** を最初に呼び出すのに指定の位置が使用されます。
dwY	dwFlags が STARTF_USEPOSITION を指定しない限り無視されます。新規ウィンドウが作成される場合、ウィンドウの左上の角の y オフセットをピクセル単位で指定します。オフセットは画面の左上の角からです。GUI 処理では、**CreateWindow** の y パラメーターが CW_USEDEFAULT の場合にオーバーラップしたウィンドウを作成する為、新しい処理が Windows API 関数の **CreateWindow** を最初に呼び出すのに指定の位置が使用されます。
dwXSize	dwFlags が STARTF_USESIZE を指定しない限り無視されます。新規しいウィンドウが作成される場合、幅をピクセル単位で指定します。GUI プロセスでは、**CreateWindow** の nWidth パラメーターが CW_USEDEFAULT の場合にオーバーラップしたウィンドウを作成する為、新しいプロセスが Windows API 関数の **CreateWindow** を最初に呼び出す際にのみ使用されます。
dwYSize	dwFlags が STARTF_USESIZE を指定しない限り無視されます。新規しいウィンドウが作成される場合、高さをピクセル単位で指定します。GUI プロセスでは、**CreateWindow** の nHight パラメーターが CW_USEDEFAULT の場合にオーバーラップしたウィンドウを作成する為、新しいプロセスが **CreateWindow** を最初に呼び出す際にのみ使用されます。
dwXCountChars	dwFlags が STARTF_USECOUNTCHARS を指定しない限り無視されます。コンソール プロセスでは、新しいコンソール ウィンドウが作成される場合、dwXCountChars が文字の列にあるスクリーンバッファーの幅を指定します。この値は、GUI プロセスで無視されます。

テーブル 10 · LAAW_STARTUPINFO (続き)

メンバー	説明
dwYCountChars	dwFlags が STARTF_USECOUNTCHARS を指定しない限り無視されます。コンソール プロセスでは、新しいコンソール ウィンドウが作成される場合、dwYCountChars が文字の行にあるスクリーン バッファーの高さを指定します。この値は、GUI プロセスで無視されます。
dwFillAttribute	dwFlags が STARTF_USEFILLATTRIBUTE を指定しない限り無視されます。制御アプリケーションに新しい制御ウィンドウが作成される場合、初期テキストと背景色を指定します。これらの値は GUI アプリケーションでは無視されます。この値には、FOREGROUND_BLUE、FOREGROUND_GREEN、FOREGROUND_RED、FOREGROUND_INTENSITY、BACKGROUND_BLUE、BACKGROUND_GREEN、BACKGROUND_RED、および BACKGROUND_INTENSITY の任意の組み合わせが可能です。**ISRTWindows.h** で定義されていない、これらの Windows 定数の利用法については、「スクリプトで Windows 定数を使用する」を参照して下さい。たとえば次の値の組み合わせは、白背景に赤色テキストを生成します : FOREGROUND_RED| BACKGROUND_RED| BACKGROUND_GREEN| BACKGROUND_BLUE

テーブル 10 ・ LAAW_STARTUPINFO (続き)

メンバー	説明
dwFlags	これは特定の STARTUPINFO メンバーが利用されているか否かを判断するビットフィールドで、処理がウィンドウを作成する際に利用されます。次の値の組み合わせを指定することができます : ・ STARTF_FORCEONFEEDBACK Ñ **LaunchApplication** または **LaunchAndAppAndWait** が呼び出されたあとの 2 秒間、カーソルがフィードバック モードであることを示します。この 2 秒間の間に処理が初めの GUI 呼び出しを行う場合、システムは処理へにさらに 5 秒間追加します。この 5 秒間の間に処理がウィンドウを表示する場合、システムは処理へウィンドウの描画を終える為のさらなる 5 秒間を追加します。 システムは Windows API 関数 **GetMessage** への最初の呼び出しの後、処理が描画中でもフィードバック カーソルをオフにします。 ・ STARTF_FORCEOFFFEEDBACK Ñ 処理開始中にフィードバックカーソルが強制的にオフとなったことを示します。標準カーソルが表示されます。 ・ STARTF_RUNFULLSCREEN Ñ 処理はウィンドウ モードではなく、フル画面モードで実行する必要があることを示します。 このフラグは x86 コンピューター上で実行されるアプリケーションの制御にのみ有効です。 ・ STARTF_USECOUNTCHARS Ñ この値が指定されなかった場合、dwXCountChars と dwYCountChars メンバーは無視されます。 ・ STARTF_USEFILLATTRIBUTE Ñ この値が指定されなかった場合、dwFillAttribute メンバーは無視されます。 ・ STARTF_USEPOSITION Ñ この値が指定されなかった場合、dwX と dwY メンバーは無視されます。 ・ STARTF_USESHOWWINDOW Ñ この値が指定されなかった場合、wShowWindow メンバーは無視されます。 ・ STARTF_USESIZE Ñ この値が指定されなかった場合、dwXSize と dwYSize メンバーは無視されます。 ・ STARTF_USESTDHANDLES Ñ hStdInput、hStdOutput、および STARTUPINFO 構造の hStdError メンバーで指定されたハンドルへの処理用に、標準入力、標準出力、そして標準エラーハンドラーを設定します。適切に動作させるためには、LAAW_PARAMETERS.bInheritHandles は TRUE に設定しなくてはなりません。 この値が指定されなかった場合、STARTUPINFO の hStdInput、hStdOutput、そして hStdError 各メンバー構造は無視されます。
wShowWindow	dwFlags が STARTF_USESHOWWINDOW を指定しない限り無視されます。wShowWindow メンバーには、Winuser.h で定義された任意の SW_ 定数が可能です。GUI プロセスでは、wShowWindow は Windows API 関数 **ShowWindow** が最初に呼び出されたときにデフォルト値を指定します。ShowWindow の **ShowWindow** パラメーターは無視されます。ShowWindow の後に続く呼び出しでは、**ShowWindow** の **ShowWindow** パラメーターが SW_SHOWDEFAULT に設定されている場合、wShowWindow メンバーが利用されます。

テーブル 10 • LAAW_STARTUPINFO (続き)

メンバー	説明
cbReserved2	予約済み。ゼロでなくてはなりません。
lpReserved2	予約済み。NULL でなくてはなりません。
hStdInput	dwFlags が STARTF_USESTDHANDLES を指定しない限り無視されます。STARTF_USESTDHANDLES が指定された場合に処理への標準入力ハンドルとして利用されるハンドルを指定します。
hStdOutput	dwFlags が STARTF_USESTDHANDLES を指定しない限り無視されます。STARTF_USESTDHANDLES が指定された場合に処理への標準出力ハンドルとして利用されるハンドルを指定します。
hStdError	dwFlags が STARTF_USESTDHANDLES を指定しない限り無視されます。STARTF_USESTDHANDLES が指定された場合に処理への標準エラーハンドルとして利用されるハンドルを指定します。

例

LaunchAppAndWait を呼び出す前に、起動されたアプリケーションが (0,0) 座標で表示されることを指定します。構造は次の様にカスタマイズします :

```
LAAW_STARTUPINFO.dwFlags = LAAW_STARTUPINFO.dwFlags | STARTF_USEPOSITION;
LAAW_STARTUPINFO.dwX = 0;
LAAW_STARTUPINFO.dwY = 0;
```

MAINTENANCE

インストールプログラムがメンテナンスモードを実行している場合、このシステム変数は TRUE に設定され、初回インストールには FALSE に設定されます。

MAINT_OPTION

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MAINT_OPTION システム変数は、[一般情報] ビューの " メンテナンス エクスペリエンス " 設定で指定したメンテナンス オプションに対応して、次の値の 1 つに設定されます。

- MAINT_OPTION_STANDARD
- MAINT_OPTION_MULTI_INSTANCE
- MAINT_OPTION_NONE

MEDIA

このシステム変数は現在のファイルメディアライブラリ、またはスクリプトで作成した機能セットを格納します。セットアップの初期化中に、MEDIA に「DATA」の値が割り当てられます。この値は、メディアビルドで作成された DATAx.cab ファイルに対応するものです。このシステム変数の値を変更してスクリプト作成コンポーネントセットを参照するには、FeatureMoveData を呼び出す前に値を 'DATA' に戻す必要があります。

MODE

システム変数 MODE には次の定数値のひとつが含まれます (実行時に値を変更することができないことにご注意ください):

テーブル 11 ・ MODE

定数	意味
SILENTMODE	Setup.exe がサイレントモードで実行されていることを示します。(つまり、ユーザーが /s 引数を使った Setup.exe を実行しています。)
NORMALMODE	Setup.exe がノーマルモードで実行されていることを示します。
RECORDMODE	Setup.exe によって、サイレントセットアップファイル (.iss ファイル) が自動的に生成されることを示します。.iss ファイルは、セットアップの入力値を記録したもので、デフォルトでは Windows フォルダーに配置されます。(つまり、ユーザーが / r 引数を使った Setup.exe を実行した場合です。)

次に示すように、MODE システム変数を if ステートメントで使用し、モードベースのスクリプトのフローをコントロールできます。

```
if (MODE = SILENTMODE) then
    // サイレント セットアップ アクションとイベントを実行します。
else
    // 普通のセットアップアクションとイベントを実行します。
endif;
```

メモ・基本の MSI プロジェクトでは、ユーザーが Windows Installer 条件 "UILevel=2" を使ってインストールをサイレントモードで実行しているかどうか判断することができます。

MSI_TARGETDIR

MSI_TARGETDIR は InstallScript MSI プロジェクト用の管理インストール (ユーザーが /a 引数を使って Setup.exe を実行した場合) のインストール先を意味します。

基本の MSI プロジェクトでは、TARGETDIR *プロパティ* (InstallScript 変数ではない) が管理インストールのインストール先を含みます。

MULTI_INSTANCE_COUNT

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数は、ターゲット システムに既にインストールされている現在実行中の複数インスタンスセットアップのインスタンスの数と等しく設定されています。新しい値をこのシステム変数に割り当てることはできません。

PACKAGE_LOCATION

プロジェクト・PACKAGE_LOCATION システム変数は、InstallScript プロジェクトにのみ適用されます。

このシステム変数はインストールが InstallShield 内で作成された自己展開実行可能ファイルから実行される場合、インストールの自己展開実行可能ファイルの完全修飾ファイル名を、それ以外の場合はヌル文字列値 (""") を含みます。

PRODUCT_GUID

この読み取り専用システム変数は、セットアップの GUID を含みます。これはプロジェクトの ProductCode プロパティの値に初期化されます。PRODUCT_GUID は、デフォルトで UNINSTALLKEY 変数の一部、または DISK1TARGET ディレクトリと SUPPORTDIR ディレクトリ の一部として利用されます。

PRODUCT_INSTALLED

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

このシステム変数は、インストーラーの有効なログファイルが存在する場合はゼロ以外の値に設定されます。セットアップが標準メンテナンスオプションを使って実行している場合、この変数は MAINTENANCE システム変数と等しくなります。

PROGRAMFILES

PROGRAMFILES システム変数には、Windows でアプリケーションを保存するよう定義されたフォルダーの完全修飾名が含まれています。英語版 Windows では、このフォルダーは Program Files という名前で、Windows がインストールされているドライブのルートにあります。(他の言語版の Windows フォルダー名はデフォルトでローカライズされた名前に設定されています)。Program Files フォルダーは、アプリケーションフォルダーのデフォルトの場所としてお勧めします。

64 ビット Windows システムでは、このフォルダーは 32 ビットのアプリケーション用のみで、デフォルトで Program Files (x86) という名前を持ちます。64 ビット アプリケーションは、PROGRAMFILES64 フォルダにインストールしなくてはなりません。

__ヒント__・複数のアプリケーションを配布している場合は、Program Files 内に企業名のフォルダーを作り、その中にアプリケーションフォルダーを作成することもできます。

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。

__プロジェクト__・InstallScript インストールでセットアップの初期化中、PROGRAMFILES 変数の値は、Windows API 関数 __SHGetSpecialFolderPath__ を CSIDL_COMMON_FILES パラメーターと共に呼び出して取得します。

基本の MSI と InstallScript MSI インストールでは、PROGRAMFILES 変数の値は Windows Installer プロパティ __ProgramFilesFolder__ に基づいて初期化されます。遅延、コミット、およびロールバック カスタム アクションは、このプロパティーにアクセスすることはできません。そのため、対応する PROGRAMFILES 変数は、遅延、コミット、およびロールバック カスタム アクションでは空白です。詳細は、「遅延／コミット／ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。

PROGRAMFILES64

PROGRAMFILES64 システム変数には、Windows で 64 ビット システム上に 64 ビット アプリケーションを保存するよう定義されたフォルダーの完全修飾名が含まれています。(32 ビット アプリケーションは PROGRAMFILES フォルダーに格納されます。) 英語版 Windows では、このフォルダーは Program Files という名前で、Windows がインストールされているドライブのルートにあります。(他の言語版の Windows フォルダー名はデフォルトでローカライズされた名前に設定されています)。Program Files フォルダーは、アプリケーション フォルダーのデフォルトの場所としてお勧めします。

__ヒント__・複数のアプリケーションを配布している場合は、Program Files 内に企業名のフォルダーを作り、その中にアプリケーションフォルダーを作成することもできます。

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。

__プロジェクト__・InstallScript インストールでセットアップの初期化中、PROGRAMFILES64 変数の値は、Windows API 関数 __SHGetSpecialFolderPath__ を CSIDL_PROGRAM_FILES パラメーターと共に呼び出して取得します。

基本の MSI と InstallScript MSI インストールでは、PROGRAMFILES64 変数の値は Windows Installer プロパティ __ProgramFiles64Folder__ に基づいて初期化されます。遅延、コミット、およびロールバック カスタム アクションは、このプロパティーにアクセスすることはできません。そのため、対応する PROGRAMFILES64 変数は、遅延、コミット、およびロールバック カスタム アクションでは空白です。詳細は、「遅延／コミット／ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。

REGDB_OPTIONS

REGDB_OPTIONS システム変数を使って、一般的なレジストリ関数にいろいろなオプションを設定することができます。次のテーブルは、指定が可能なオプションを説明します :

テーブル 12 · REGDB_OPTIONS

オプション	意味
REGDB_OPTION_DISABLETEXTSUBS	レジストリ関数に渡された文字列のテキスト置換を無効にします。このオプションは、開き山かっこ (<) と閉じ山かっこ (>) を含むレジストリ関数文字列で作業しているときに使用しますが、テキスト置換と解釈すべきではありません。
REGDB_OPTION_NO_DELETE_OLD_MAJMIN_VERSION	**MaintenanceStart** 関数が、以下の定数のレガシ値が削除されないように防ぎます : • REGDB_VALUENAME_UNINSTALL_MAJORVERSION (アプリケーション アンインストール キーの下にあるメジャー バージョン値名)。InstallShield 2009 以前で作成されたインストールで、この値は MajorVersion。 • REGDB_VALUENAME_UNINSTALL_MINORVERSION (アプリケーション アンインストール キーの下にあるマイナー バージョン値名)。InstallShield 2009 以前で作成されたインストールで、この値は MinorVersion。 詳細については、「InstallShield 2009 以前のプロジェクトをアップグレードする」の「InstallScript インストールのアンインストール キーにおけるメジャーおよびマイナー バージョンのレジストリ エントリの変更」セクションを参照してください。
REGDB_OPTION_WOW64_64KEY	すべての将来のレジストリ操作が、(64 ビットシステム上の) レジストリの 32 ビット領域ではなく 64 ビット領域に影響することを指定します。３２ ビットシステム上で、このオプションを設定しても影響はありません。 64 ビットのレジストリの場所へのインストールに関する詳細は、「64 ビット オペレーティング システムを InstallScript インストールでターゲットにする」を参照してください。
REGDB_OPTION_USE_DEFAULT_OPTIONS	以前設定されたオプションをすべてリセット (クリア) します。

オプションを追加するには、１ つまたは複数のオプションをビット単位の OR (|) 演算子を使用して次のように組み合わせます。

REGDB_OPTIONS = REGDB_OPTIONS | REGDB_OPTION_WOW64_64KEY

オプションを削除するには、ビット単位の AND (&) 演算子とビット単位の NOT (~) 演算子を使用して削除するオプションを次のように指定します。

REGDB_OPTIONS = REGDB_OPTIONS & ~REGDB_OPTION_WOW64_64KEY

メモ・*REGDB_OPTION_WOW64_64KEY オプションを有効にすると、レジストリセットのレジストリエントリが作成される場所に影響します。例えば、CreateRegistrySet 関数を呼び出す時にこのオプションを有効にした場合、レジストリセットはレジストリの 64 ビット領域に作成されます。インストールする特定の 64 ビット レジストリ セットに対して、このオプションを有効にする場合、その他のレジストリ エントリまたはレジストリ セットがレジストリの 64 ビット領域に誤って作成されないように、このオプションを無効にすることが推奨されます。64 ビットのレジストリの場所へのインストールに関する詳細は、「64 ビット オペレーティング システムを InstallScript インストールでターゲットにする」を参照してください。*

*InstallScript エンジンは、レジストリの 64 ビット 部分への製品の [プログラムの追加と削除] 情報のインストールをサポートしないため、特定のレジストリ関数、たとえば **CreateInstallationInfo**、**MaintenanceStart**、**RegDBGetItem**、**RegDBSetItem**、**RegDBGetAppInfo**、**RegDBSetAppInfo**、および **RegGetUninstCmdLine** では EGDB_OPTION_WOW64_64KEY オプションがサポートされていません。*

REINSTALLMODE

このシステム変数は、再インストール関数の 1 つが InstallScript インストールで呼び出された場合、つまり FeatureReinstall、FeatureUpdate、FeaturePatch が現在のインストーラーのインスタンスで呼び出された場合、ゼロ以外の値が入ります。

プロジェクト・*InstallScript MSI インストールの場合、**FeatureReinstall** 関数が呼び出されると、このシステム変数にはゼロ以外の値が入ります。**FeatureUpdate** および **FeaturePatch** は、InstallScript MSI インストールでは定義されていないので、呼び出さないようにしてください。*

REMOVEALLMODE

プロジェクト・*この情報は、InstallScript プロジェクトに適用します。*

このシステム変数は、アプリケーションが完全にアンインストールされた場合、つまり FeatureRemoveAll、FeatureRemoveAllInMedia、FeatureRemoveAllInMediaAndLog が現在のセットアップのインスタンスで呼び出された場合、ゼロ以外の値になり、それ以外の場合は FALSE になります。このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。

メモ・*アプリケーションが完全にアンインストールされた時、スクリプト コードのみを実行するには、次の if-then ステートメント内にコードを配置します:*

```
if REMOVEALLMODE!=0 then
   /* このコードは、アンインストール中にのみ実行されます。*/
endif;
```

特定のコンポーネントがアンインストールされた時に特定のアンインストールアクションを実行するには、コンポーネントの <ComponentName>_Uninstalling イベントを上書きして、このイベントで実行します。

REMOVEONLY

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

– removeonly を利用して Setup.exe が実行された場合、REMOVEONLY システム変数はゼロ以外の値に等しく設定され、それ以外の場合は FALSE に等しく設定されます。OnMaintUIBefore イベント ハンドラー関数のデフォルトコードは、REMOVEONLY の値に従って条件付で SdWelcomeMaint ダイアログを表示します。

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。

SELECTED_LANGUAGE

プロジェクト・SELECTED_LANGUAGE をサポートするプロジェクト タイプは次のとおりです。

- *InstallScript*
- *InstallScript MSI*

この SELECTED_LANGUAGE システム変数の数値には、インストーラーがプロンプトやメッセージの表示に使用する言語の ID が含まれます。

このシステム変数には対応する <SELECTED_LANGUAGE> テキスト置換があり、4 桁の 16 進数としてフォーマットされた SELECTED_LANGUAGE の値が含まれます (0x プレフィックスを含む)。たとえば、SELECTED_LANGUAGE が ISLANG_ENGLISH_UNITEDSTATES の場合、テキスト置換の値は 0x0409 になります。

SHAREDSUPPORTDIR

プロジェクト・InstallScript プロジェクトは SHAREDSUPPORTDIR をサポートします。

SHAREDSUPPORTDIR は読み取り専用の変数で、InstallScript インストールとそのインストールに含まれるすべての InstallScript オブジェクトとの間で共有されるサポートファイルのすべてを含むディレクトリを識別します。

このシステム変数には、対応する <SHAREDSUPPORTDIR> テキスト置換があります。

SHELL_OBJECT_FOLDER

プロジェクト・次のプロジェクト タイプは、SHELL_OBJECT_FOLDER をサポートします:

- *InstallScript*
- *InstallScript MSI*

SHELL_OBJECT_FOLDER システム変数は、実行時にスクリプトを使って (通常は [スタート メニュー] フォルダーにある) シェル オブジェクト フォルダー の名前を指定するのに使用されます。

[ショートカット] ビューにあるフォルダーの " 表示名 " 設定で **SHELL_OBJECT_FOLDER** (InstallScript または InstallScript MSI プロジェクトの場合) または **<SHELL_OBJECT_FOLDER>** (InstallScript プロジェクトの場合) を指定できます。その後、ショートカットが作成される前にスクリプトで SHELL_OBJECT_FOLDER 変数を設定することにより、このフォルダーの表示名を実行時に定義することができます。通常、ショートカットはファイルの転送時に作成されます。

プロジェクト・*InstallScrip プロジェクトでは、" 表示名 " 設定で* **<SHELL_OBJECT_FOLDER>** *または* **SHELL_OBJECT_FOLDER** *のどちらかを指定できます。どちらの場合も、テキスト置換が使用されます。ただし、山かっこの使用をお勧めします (例、* **<SHELL_OBJECT_FOLDER>** *)。*

InstallScript MSI プロジェクトでは、山括弧なしの **SHELL_OBJECT_FOLDER** *を指定しなくてはなりません。*

InstallScript MSI インストールでこの機能を使用する場合、[ショートカット] ビューに表示されるフォルダーの " キー名 " 設定で指定される文字は、すべて大文字でなくてはなりません (例、NEWFOLDER1)。

インストールがメンテナンス モードでない場合、SHELL_OBJECT_FOLDER は InstallScript エンジンの初期化中に IFX_PRODUCT_NAME と同じ値に初期化されます。これらの変数は、一旦初期化が完了すると同期されません。したがって、片方の変数を変更してから、もう片方の変数も変更する必要がある場合、両方とも手動で変更しなければなりません。この種の手作業による変更はログ記録され、次にメンテナンス操作が行われるときにログから読み取られます。そのため、表示名に SHELL_OBJECT_FOLDER 変数を使用するショートカットは、アンインストール中に削除される場合があります。

SHOW_PASSWORD_DIALOG

プロジェクト・*この情報は、InstallScript プロジェクトに適用します。*

メディアウィザードの [一般オプション] パネルまたはメディア プロパティ シートの [一般] ページで [セットアップの初期化中にパスワード ダイアログを表示する] チェック ボックスを選択している場合、SHOW_PASSWORD_DIALOG システム変数は TRUE になります。そうでない場合は FALSE です。

SRCDIR

このシステム変数には、Windows Installer パッケージを含むソースフォルダーへの完全修飾パスが格納されます。

SRCDIR は シーケンスが始まった時に Windows Installer プロパティ **SourceDir** の値へ初期化され、InstallScript カスタム アクションの新しい値を割り当てることはできません。

SRCDISK

このシステム変数には、ソースディスクが存在するドライブの名前が格納されます。セットアップの初期化時に、InstallShield によって、セットアップ スクリプト ファイル Setup.inx を含むディスクが格納されているドライブの名前が SRCDISK に代入されます。たとえば、A ドライブのフロッピー ディスクから Setup.exe を起動し、このディスクにファイル Setup.inx が含まれている場合、InstallShield により値 “A:” が SRCDISK に代入されます。InstallShield ではドライブ名にコロン (:) が含まれる点に注意してください。

メモ・この変数で指定されているドライブのルート フォルダーを参照する場合、変数に円記号 (¥) を追加する必要があります (2 つの円記号で指定)。たとえば SRCDISK の値が A: の場合、ステートメント SRCDISK + "¥¥" は A: ドライブのルート フォルダーを参照します。

SUPPORTDIR

セットアップの初期化中に、インストールは一時ファイルおよびインストールに圧縮されているファイルをコピーすることが可能なターゲット システム上のフォルダーを検索します。インストールは SUPPORTDIR の値を、そのフォルダーの完全修飾パスに設定します。

さらに、InstallShield の [サポート ファイル／ビルボード] ビューの言語非依存 (または言語固有) ファイル リストに追加するファイルは、インストールが初期化される時に SUPPORTDIR に圧縮解除され、インストールが完了したときに削除されます。

InstallScript プロジェクトの特定のサポート ファイルにアクセスするには、SUPPORTDIR 変数を直接使用してから、ファイル名を SUPPORTDIR 値に付加して、ファイルの完全パスを取得します。以下は、InstallScript イベント コードの例です。

```
prototype STRING GetSupportFilePathIS(STRING);
function STRING GetSupportFilePathIS(szSupportFile)
begin
   return SUPPORTDIR ^ szSupportFile;
end;
```

メモ・InstallScript 変数 SUPPORTDIR は、InstallScript オブジェクト スクリプト間で、または InstallScript オブジェクト スクリプトとメインのインストール スクリプトとの間では共有されません。

プロジェクト・Windows Installer プロパティ SUPPORTDIR の値は、InstallScript システム変数 SUPPORTDIR の値とは同じではありませんので注意してください。

イベントドリブン型の InstallScript では、SUPPORTDIR システム変数はサポート ファイルを含むフォルダーをポイントします。

基本の MSI および InstallScript MSI プロジェクトの場合、InstallScript カスタム アクションは個々のエンジンを初期化します。各エンジンは、プライマリ SUPPORTDIR がどこにあるか分かっていません。また、各エンジンはサポート ファイルのそれ自身のコピーを抽出しません。カスタム アクションから抽出されたサポート ファイルを見つける方法は、「.msi データベースにファイルを配置し、実行時に抽出する」を参照してください。

SYSINFO

セットアップの初期設定中に、インストールによって SYSINFO 構造変数のメンバーが設定され、ターゲット コンピューターのオペレーティング プラットフォームが特定されます。この変数のメンバーに割り当てられている値を調べると、スクリプトで以下のような情報を判断できます :

- オペレーティング システム
- オペレーティング システムのメジャーおよびマイナー バージョン
- オペレーティング システムのサブバージョン
- Internet Explorer のバージョン
- インストール済みの最新サービス パック
- エンドユーザーの管理者権限の有無 (Windows NT の場合)
- エンドユーザーがパワーユーザーかどうか
- システムが 64 ビットかどうか
- システムが仮想マシンかどうか
- システム言語の言語 ID、ユーザー言語、およびオペレーティングシステム言語

以下の表に、SYSINFO の各メンバーの意味を示します :

テーブル 13 · SYSINFO メンバー

メンバー	意味
SYSINFO.bIntel	TRUE の場合、プロセッサは Intel です。
SYSINFO.bIsVirtualMachine	TRUE の場合、仮想マシンが検出されます。 詳細については、「インストールが仮想マシン上で実行されているかどうかを検出する」を参照してください。
SYSINFO.bIsWow64	インストールが 64 ビット プラットフォームで実行されている場合、この値は、ゼロ以外の値になります。
SYSINFO.bShellExplorer	TRUE の場合、シェルはエクスプローラーです。
SYSINFO.bWinServer2003R2	このメンバーが TRUE のとき、オペレーティング システムは Windows Server 2003 R2 です。

テーブル 13 ・ SYSINFO メンバー (続き)

メンバー	意味
SYSINFO.nISOSL	値はターゲットコンピューターのオペレーティングシステムを示します。使用できる値は次のとおりです。 ・ ISOSL_WINXP Ñ Windows XP Edition ・ ISOSL_WINSERVER2003 Ñ Windows Server 2003 ・ ISOSL_WINVISTA_SERVER2008 (または ISOSL_WINVISTA) Ñ Windows Vista と Windows Server 2008 は、同じメジャー バージョン番号とマイナー バージョン番号を持つ点に注意してください。したがって、InstallScript を使って Windows Server 2008 と Windows Vista を区別するには、SYSINFO.nOSProductType = VER_NT_WORKSTATION が、Windows Vista の場合は TRUE、Windows Server 2008 の場合は FALSE であるかどうかを確認してください。 ・ ISOSL_WIN7_SERVER2008R2 Ñ Windows 7 または Windows Server 2008 R2 ・ ISOSL_WIN8—Windows Vista または Windows Server 8 ・ ISOSL_WIN81 — Windows 8.1 または Windows Server 2012 R2 ・ ISOSL_WIN10—Windows 10 ***メモ・** Windows のいくつかのクライアントおよびサーバー バージョンでは、同じメジャー バージョン番号とマイナー バージョン番号が使用されています。* ・ *Windows 8.1 と Windows Server 2012 R2 では、同じメジャー バージョン番号とマイナー バージョン番号が使用されています。* ・ *Windows 8 と Windows Server 2012 では、同じメジャー バージョン番号とマイナー バージョン番号が使用されています。* ・ *Windows 7 と Windows Server 2008 R2 では、同じメジャー バージョン番号とマイナー バージョン番号が使用されています。* ・ *Windows Vista と Windows Server 2008 は、同じメジャー バージョン番号とマイナー バージョン番号を持ちます。* *このため、インストールの実行時、これらの OS バージョンでは、クライアント バージョンは、同等のサーバー バージョンと同じものと見なされます。したがって、クライアント バージョン向けとマークされているコンポーネントは、サーバー バージョンにもインストールされます。クライアント バージョンとサーバー バージョンを区別するには、SYSINFO.nOSProductType と VER_NT_WORKSTATION が等しいかどうか確認します。クライアント バージョンでは、これらは等しくなっています (True)。サーバー バージョンでは、False です。*
SYSINFO.nOSMajor	オペレーティングシステムのメジャーバージョン番号を示す値です。
SYSINFO.nOSMinor	オペレーティングシステムのメジャーバージョン番号を示す値です。

テーブル 13 • SYSINFO メンバー (続き)

メンバー	意味
SYSINFO.nOSProductType	値は、Windows OSVERSIONINFOEX 構造体の wProductType を現在のプラットフォームに定義されたとおりに示します。使用できる値は次のとおりです。 • VER_NT_WORKSTATION • VER_NT_DOMAIN_CONTROLLER • VER_NT_SERVER また、wProduct Type でサポートされているその他の任意の定数を #define してテストすることもできます。さらに詳しい情報は、MSDN Web サイトの「OSVERSIONINFOEX Structure」を参照してください。
SYSINFO.nOSSuiteMask	値は、Windows OSVERSIONINFOEX 構造体の wSuitesMask を現在のプラットフォームに定義されたとおりに示します。使用できる値は次のとおりです。 • VER_SUITE_BACKOFFICE • VER_SUITE_DATACENTER • VER_SUITE_ENTERPRISE • VER_SUITE_PERSONAL • VER_SUITE_SMALLBUSINESS • VER_SUITE_SMALLBUSINESS_RESTRICTED • VER_SUITE_TERMINAL また、wSuiteMask でサポートされているその他の任意の定数を #define してテストすることもできます。さらに詳しい情報は、MSDN Web サイトの「OSVERSIONINFOEX Structure」を参照してください。

テーブル 13 ・ SYSINFO メンバー (続き)

メンバー	意味
SYSINFO.nSuites	ターゲット コンピューター上のスイート (複数可) を示す 1 つまたは複数の ビットフラグの組み合わせ。利用可能なビットフラグは次のとおりです。 ・ **ISOS_ST_ALL** ・ **ISOS_ST_XP_PRO** ・ **ISOS_ST_XP_HOME** ・ **ISOS_ST_SERVER** ・ **ISOS_ST_SERVER2003_R2** ・ **ISOS_ST_WORKSTATION** ・ **ISOS_ST_BACKOFFICE** ・ **ISOS_ST_DATACENTER** ・ **ISOS_ST_ENTERPRISE** ・ **ISOS_ST_SERVER2003_R2** ・ **ISOS_ST_SMALLBUSINESS** ・ **ISOS_ST_SMALLBUSINESS_RESTRICTED** ・ **ISOS_ST_TERMINAL** ・ **0 (ゼロ) Ñ** ターゲット マシンにスイートが検出されなかったことを示します。 ビット フラグが設定されているかどうかを確認するには、次の例にしたがってビット ワイズ AND (&) 演算子を利用します。 `if (SYSINFO.nSuites & ISOS_ST_XP_HOME) then` `  /* Windows XP Home Edition に固有の` `  処理を実行します。*/` `endif;` ***メモ・ここにリストされるスイートは、Windows API の OSVERSIONINFOEX データ構造で指定することができるものです。***
SYSINFO.nSystemDefaultUILangID	値はインストールされているオペレーティングシステム言語の ID を示します。
SYSINFO.nSystemLangID	値はシステム言語の ID を示します。
SYSINFO.nUserLangID	値はユーザー言語 ID を示します。
SYSINFO.nWinMajor	Windows のメジャーバージョン番号を示す値です。
SYSINFO.nWinMinor	Windows のマイナーバージョン番号を示す値です。

テーブル 13 • SYSINFO メンバー (続き)

メンバー	意味
SYSINFO.szInstalledIEVersion	システムの Internet Explorer バージョンを示す値。このメンバーはバージョン 4 以降でサポートされています。インストールされているバージョンが 4 よりも古い場合、値はヌル ("") となります。 ***メモ • バージョンが 4 よりも古い場合にこの値がヌルになる事実に依存しないでください。今後はこのメンバー変数で 4 以前のバージョンの Internet Explorer の検出をサポートするようになる可能性があるため、あえてバージョン 4 以降をテストしてください。***
SYSINFO.WINNT.bAdmin_Logged_On	このメンバーが TRUE の場合、エンドユーザーは管理者権限を使って NT でログインしています。
SYSINFO.WINNT.bPowerUser_Logged_On	このメンバーが TRUE の場合、現在のユーザーはパワー ユーザー グループに属します。
SYSINFO.WINNT.bWin10	***メモ • このメンバーは、イベント ベースの InstallScript コードに適用し、InstallScript カスタム アクションには適用しません。*** このメンバーが TRUE のとき、オペレーティング システムは Windows 10 です。
SYSINFO.WINNT.bWin81	***メモ • このメンバーは、イベント ベースの InstallScript コードに適用し、InstallScript カスタム アクションには適用しません。*** このメンバーが TRUE のとき、オペレーティング システムは Windows 8.1 または Windows Server 2012 R2 です。
SYSINFO.WINNT.bWin8	***メモ • このメンバーは、イベント ベースの InstallScript コードに適用し、InstallScript カスタム アクションには適用しません。*** このメンバーが TRUE のとき、オペレーティング システムは Windows 8 または Windows Server 2012 です。
SYSINFO.WINNT.bWin7_Server2008R2	***メモ • このメンバーは、イベント ベースの InstallScript コードに適用し、InstallScript カスタム アクションには適用しません。*** このメンバーが TRUE のとき、オペレーティング システムは Windows または Windows Server 2008 R2 です。

テーブル 13 · SYSINFO メンバー (続き)

メンバー	意味
SYSINFO.WINNT.bWinNT	このメンバーは TRUE の場合は、オペレーティング システムは Windows NT (Windows XP を含む) です。
SYSINFO.WINNT.bWinVista_ Server2008 (SYSINFO.WINNT.bWinVista)	SYSINFO.WINNT.bWinVista_Server2008 または SYSINFO.WINNT.bWinVista が TRUE の場合、オペレーティング システムは Windows Vista または Windows Server 2008 です。 Windows Server 2008 と Windows Vista の区別は、SYSINFO.nOSProductType が VER_NT_WORKSTATION に等しいかどうかを確認します。TRUE のときは、Windows Vista で、FALSE のときは、Windows Server 2008 です。
SYSINFO.WINNT.bWinXP	このメンバーが TRUE のとき、オペレーティング システムは Windows XP です。
SYSINFO.WINNT.bWinServer2003	このメンバーが TRUE のとき、オペレーティング システムは Windows Server 2003 または Windows Server 2008 R2 です。
SYSINFO.WINNT.nServicePack	インストール済みのサービス パックの数。 インストールはこの情報を Windows API **GetVersionEx** を呼び出し、nServicePackMajor 値を読み取って取得します。

例

次の部分的なコードでは、ターゲットシステムのオペレーティング システムが Windows XP の場合、メッセージ ボックスが表示されます。

```
if (SYSINFO.WINNT.bWinXP) then
    MessageBox("Windows XP にインストール中 ",INFORMATION);
endif;
```

SYSPROCESSORINFO

インストールの初期設定中に、インストールによってこの構造体の変数のメンバーが設定され、ターゲットコンピューターのプロセッサについての情報が識別されます。この変数のメンバーに割り当てられた値を調べて、スクリプトは、システム上のプロセッサの数またはプロセッサの種類などの情報を判断できます。

以下のテーブルでは、SYSPROCESSORINFO の各メンバーの意味を示されています。

***メモ ·** これらメンバーはそれぞれ、Windows SYSINFO 構造体にあるメンバーに対応しています。これらのメンバーは、初期化中に、64 ビット Windows システム上で Windows API GetSystemInfo または GetNativeSystemInfo 関数を呼び出すことで設定されます。この構造体に関しては、MSDN ライブラリ にある文書を参照してください。また、*

Microsoft によって文書化されているように、nProcessorType の使用は推奨しません。代わりに、nProcessorLevel と nProcessorArchitecture を使用してください。

テーブル 14 · SYSPROCESSORINFO

メンバー	意味
SYSPROCESSORINFO.nProcessorArchitecture	プロセッサ アーキテクチャを示します。使用できる値は次のとおりです。 PROCESSOR_ARCHITECTURE_INTEL PROCESSOR_ARCHITECTURE_IA64 PROCESSOR_ARCHITECTURE_AMD64 PROCESSOR_ARCHITECTURE_UNKNOWN  ***メモ ·** InstallScript には、この構造体メンバーの最も一般的な値に対する定数が含まれています。ただし、異例なケースで、この構造体メンバーに Windows によって定義された他の値が含まれていることもあります。その他に追加された可能性のある値については、対応する SYSINFO メンバーに関する Windows 文書を参照してください。*
SYSPROCESSORINFO.nNumberOfProcessors	システム上のプロセッサの数を示します。
SYSPROCESSORINFO.nProcessorType	プロセッサの種類を示します。使用できる値は次のとおりです。 PROCESSOR_INTEL_386 PROCESSOR_INTEL_486 PROCESSOR_INTEL_PENTIUM PROCESSOR_INTEL_IA64 PROCESSOR_AMD_X8664  ***メモ ·** InstallScript には、この構造体メンバーの最も一般的な値に対する定数が含まれています。ただし、異例なケースで、この構造体メンバーに Windows によって定義された他の値が含まれていることもあります。その他に追加された可能性のある値については、対応する SYSINFO メンバーに関する Windows 文書を参照してください。*
SYSPROCESSORINFO.nProcessorLevel	システム上のアーキテクチャ依存のプロセッサレベルを示します。これはよく、ベンダーによって定義され、表示目的で使用します。これらのメンバーの意味に関する詳しい情報は、SYSINFO 構造体に関する Windows 文書を参照してください。

テーブル 14 · SYSPROCESSORINFO (続き)

メンバー	意味
SYSPROCESSORINFO.nProcessorRevision	システム上のアーキテクチャ依存のプロセッサ リビジョンを示します。これらのメンバーの意味に関する詳しい情報は、SYSINFO 構造体に関する Windows 文書を参照してください。

TARGETDIR

セットアップの初期化中に、インストールはシステム変数 TARGETDIR にハード ディスク上のターゲット フォルダーへの完全修飾パスを割り当てます。このフォルダーは Win.ini ファイルが含まれるフォルダーで、通常 Windows フォルダーです。InstallScript 関数の中には、ファイルの操作時にこの変数を使用するものがあります。このような関数を使用する場合、事前に TARGETDIR 変数にターゲットフォルダーを設定しておく必要があります。OnFirstUIBefore イベントハンドラー関数のデフォルトコードは、値を TARGETDIR へ代入します。

このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。つまり、任意のスクリプトでこのシステム変数に割り当てた値は、明示的にリセットされない限り、次に実行されるどのスクリプトのコードでも同じ値になります。

TARGETDISK

セットアップの初期化時に、インストールによって、ターゲットディスクドライブの名前がシステム変数 TARGETDISK に代入されます。このドライブは Win.ini ファイルが含まれるドライブで、通常 C: ドライブです。InstallShield ではドライブ名にコロン (:) が含まれる点に注意してください。

***メモ ·** この変数で指定されているドライブのルートフォルダーを参照する場合、変数に円記号 (¥) を追加する必要があります (2 つの円記号で指定)。たとえば TARGETDISK の値が C: の場合、ステートメント TARGETDISK + "¥¥" は C: ドライブのルートフォルダーを参照します。*

UNINST

***プロジェクト ·** UNINST をサポートするプロジェクト タイプは次のとおりです。*

- *InstallScript*
- *InstallScript MSI*

UNINST システム変数は、以前のバージョンの InstallShield ソフトウェアとの互換性を目的として提供されています。ターゲット システムに存在する **Setup.exe** のコピーを起動してアンインストールを実行するコマンドラインを含みます。デフォルトの値は次のとおりです。

<UNINSTALL_STRING> -uninst

このコマンド ラインを使用すると、インストールは、インストールが起動されたときに OnUninstall イベントを実行します。詳細は、**Setup.exe** の /uninst コマンドライン パラメーター の情報を参照してください。

このコマンドラインは、DeinstallStart 関数によって該当するレジストリ値に保存されます。DeinstallStart 関数は、以前のバージョンの InstallShield ソフトウェアとの互換性を目的として提供されています。

このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。

UNINST に独自のコマンドラインスイッチを追加して、スクリプトのアンインストールコードで処理することができます。スイッチを追加してシステム変数 DISK1TARGET の値を変更する場合、UNINST への追加を行う前に DISK1TARGET を変更してください。DISK1TARGET は UNINST に組み込まれているので、DISK1TARGET を変更すると UNINSTALL は自動的に変更されます。

UNINSTALLKEY

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

プロジェクトによって異なる情報がある場合は、その内容が説明されています。

UNINSTALLKEY システム変数には、アンインストール情報を格納するのに使用したレジストリ キーの名前が含まれています。レジストリ キーは、SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Uninstall にあります。

InstallScript インストールでは、このキーは、HKEY_USER_SELECTABLE_AUTO の下に配置され、ALLUSERS スクリプト変数の値で制御されます。InstallScript MSI インストールでは、インストールが管理者によって実行されている場合、このキーは、HKEY_LOCAL_MACHINE の下に配置され、それ以外の場合、HKEY_CURRENT_USER の下に配置されます。

InstallScript のインストーラーでは、デフォルト値は **INSTANCE_GUID** です。InstallScript MSI インストーラーでは、UNINSTALLKEY のデフォルト値は **InstallShield_{ProductCode}** です。

別のアンインストールキーを指定するには、次の様にスクリプトで UNINSTALLKEY へ新しい値を割り当てます :

```
UNINSTALLKEY = "Sample App";
```

他にインストールされているアプリケーションとの競合を避けるため、必ずアプリケーションに一意の値を使用してください。

UNINSTALLKEY 変数がデフォルト値から変更されている場合、インストールは製品の追加レジストリ キーを自動的に作成します。

- **InstallScript インストールの場合** – REGDB_KEYPATH_ISUNINSTINFO ^ INSTANCE_GUID
- **InstallScript MSI インストールの場合** – REGDB_KEYPATH_ISUNINSTINFO ^ [ProductGuid]

このキーには、**UninstallKey** (REGDB_VALUENAME_UNINSTALLKEY) と名づけられた単一の文字列値が含まれています。値データには、UNINSTALLKEY 変数によって決定された製品のアンインストール キーの名前が含まれています。この値を使って、その他の製品がこの製品のカスタム アンインストール キーを使わずに製品のアンインストール情報を検出および使用できるようにすることが可能です。

UNINSTALL_DISPLAYNAME

このシステム変数は [プログラムの追加と削除] パネルの表示製品名を含みます。この値は一般的に、[一般情報] ビューで指定された製品名です。

別のアンインストール表示名を指定するには、次の様にスクリプトでデータ転送前に UNINSTALL_DISPLAYNAME へ新規値を割り当ててください。

```
UNINSTALL_DISPLAYNAME = "Sample App";
```

プロジェクト・*UNINSTALL_DISPLAYNAME は、InstallScript および InstallScript MSI インストールプロジェクトのみで利用することができます。基本の MSI プロジェクトは* ***ProductName*** *プロパティの値を、[プログラムの追加と削除] パネルの製品表示名として利用します。*

UNINSTALL_STRING

プロジェクト・*UNINSTALL_STRING をサポートするプロジェクト タイプは次のとおりです。*

- *InstallScript*
- *InstallScript MSI*

UNINSTALL_STRING システム変数には、アンインストールを実行するためにターゲット システムに配置されているセットアップランチャー、インストールの実行可能ファイルを起動するコマンドラインが含まれます。デフォルトの値は次のとおりです。

<DISK1TARGET>¥<DISK1SETUPEXENAME> -runfromtemp -l<SELECTED_LANGUAGE>

セットアップランチャーは、([削除ボタンの無効化] プロパティを利用して) [プログラムの追加と削除] の [削除] ボタンを非表示にしない限り、自動的に UNINSTALL_STRING コマンドラインをレジストリへ書き込みます。

UNINSTALL_STRING に独自のコマンドラインスイッチを追加して、スクリプトのアンインストール コードで処理することができます。

UPDATEMODE

UPDATEMODE システム変数は、InstallScript プロジェクトにのみ適用されます。

このシステム変数は、OnSetUpdateMode イベントハンドラー関数によって設定され、OnShowUI イベントハンドラーが適切な UI イベントハンドラーを呼び出すのに使用します。

WINDIR

WINDIR システム変数は、メイン オペレーティング環境を含むフォルダーの完全修飾名を含みます (例、**C:¥Windows**)。

このシステム変数は読み取り専用です。この変数に値を割り当てようとすると、コンパイラー エラーの原因となります。このシステム変数の値は、複数のオブジェクト スクリプト間およびオブジェクト スクリプトとメインのセットアップ スクリプトとの間で共有されます。

プロジェクト・*InstallScript インストールでセットアップの初期化中、WINDIR 変数の値は、Windows API 関数* ***GetWindowsDirectory*** *を呼び出して取得します。*

*基本の MSI と InstallScript MSI インストールでは、WINDIR 変数の値は Windows Installer プロパティ **WindowsFolder** に基づいて初期化されます。遅延、コミット、およびロールバック カスタム アクションは、このプロパティーにアクセスすることはできません。そのため、対応する WINDIR 変数は、遅延、コミット、およびロールバック カスタム アクションでは空白です。詳細は、「遅延／コミット／ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。*

WINDISK

WINDISK システム変数は、メイン オペレーティング環境を含むディスク ドライブの ID を含みます。このドライブは、通常 Windows プログラムを含む C ドライブです。InstallScript エンジンは、ドライブ名にコロン (:) を含みますので、ご注意ください。

***メモ・**この変数で指定されているドライブのルート フォルダーを参照する場合、変数に円記号 (¥) を追加する必要があります (2 つの円記号で指定)。たとえば WINDISK の値が C: の場合、ステートメント WINDISK + "¥¥" は C: ドライブのルート フォルダーを参照します。*

***プロジェクト・**InstallScript インストールでセットアップの初期化中、WINDIR 変数の値は、Windows API 関数 **SHGetSpecialFolderPath** を CSIDL_WINDOWS パラメーターと共に呼び出して取得します。*

*基本の MSI および InstallScript MSI インストールの場合、WINDIR 変数の値は、InstallScript 関数 **GetDisk** を WINDIR と一緒に呼び出して取得します。値の取得に失敗した場合、この変数は Windows Installer プロパティ **WindowsVolume** に基づいて初期化されます。遅延、コミット、およびロールバック カスタム アクションは、このプロパティーにアクセスすることはできません。そのため、対応する WINDIR 変数は、遅延、コミット、およびロールバック カスタム アクションでは空白です。詳細は、「遅延／コミット／ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。*

WINSYSDIR

WINSYSDIR システム変数は、System32 システム フォルダーの名前を含みます。このフォルダーは、Windows のバージョンに合わせて、アプリケーション拡張機能 (DLL)、デバイスドライバー、および、他の Windows システム ファイルを格納するために使用されます。

64 ビット Windows システム上では、この変数は、32 ビットアプリケーションによって使用されている Windows システムファイルを格納するフォルダーをポイントします。このフォルダーの名前は SysWOW64 です。64 ビット アプリケーションが使用するシステム ファイルには、異なる Windows システム フォルダーがあります。システム変数 WINSYSDIR64 は、このフォルダーへのアクセスを提供します。

***プロジェクト・**32 ビット システムにおいて InstallScript インストールでセットアップの初期化中、WINSYSDIR 変数の値は、Windows API 関数 **GetSystemFolder** を呼び出して取得します。64 ビット システムにおいて InstallScript インストールでセットアップの初期化中、WINSYSDIR 変数の値は、64 ビット実行可能ファイルから Windows API 関数 **GetSystemWow64Directory** を呼び出して取得します。*

*32 ビット システムにおいて基本の MSI と InstallScript MSI インストールでは、WINSYSDIR 変数の値は Windows Installer プロパティ **SystemFolder** に基づいて初期化されます。64 ビット システムにおいて、この変数は Windows Installer プロパティ **System64Folder** に基づいて初期化されます。遅延、コミット、およびロールバック カスタム アクションは、このプロパティーにアクセスすることはできません。そのため、対応する WINSYSDIR 変数は、遅*

延、コミット、およびロールバック カスタム アクションでは空白です。詳細は、「遅延／コミット／ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。

WINSYSDIR64

WINSYSDIR64 システム変数は 64 ビット System32 システム フォルダーの名前を含みます。このフォルダーは、Windows のバージョンに合わせて、アプリケーション拡張機能 (DLL)、デバイスドライバー、および、他の Windows システムファイルを格納するために使用されます。

WINSYSDIR64 変数は 64 ビット System32 フォルダーに設定されていますが、64 ビット Windows には、自動的に 32 ビットアプリケーション (InstallScript エンジンなど) を 32 ビット SysWOW64 フォルダーにリダイレクトする機能が含まれています。したがって、InstallScript コードを使って、WINSYSDIR64 の読み取りまたは書き込みを行う場合、場合によって、まず最初に、定数 WOW64FSREDIRECTION を関数 Disable と Enable と一緒に使って、ファイル システムのリダイレクトを無効にする必要があります。無効化をしなかった場合、WINSYSDIR64 の読み取りおよび書き込みは、誤って 32 ビット SysWOW64 システムフォルダーにリダイレクトされます。インストールが利用する可能性のある Windows 機能にはファイル システム リダイレクトを有効にしておく必要があるため、Windows ドキュメンテーションではリダイレクトを無効にするのはそれが必要な場合のみにとどめることが推奨されています。WINSYSDIR64 の読み取りまたは書き込みが完了したら、すぐにシステム ファイルのリダイレクトを有効にすることをお勧めします。

以下のコードは、スクリプトを通してファイルを WINSYSDIR64 に転送する前後において、どのようにリダイレクトを有効化 / 無効化するかの例です。

```
Disable(WOW64FSREDIRECTION );
XCopyFile( SUPPORTDIR ^ "MyFile.dll", WINSYSDIR64, COMP_NORMAL );
Enable (WOW64FSREDIRECTION);
```

プロジェクト・*InstallScript プロジェクトでは、WOW64FSREDIRECTION を使ってスクリプトを変更する必要なく、ファイルを 64 ビットの System32 フォルダーにインストールすることができます。この場所にインストールする必要があるファイルがある場合、ファイルとレジストリ データをコンポーネントに追加して、コンポーネントの "64 ビット コンポーネント " 設定で [はい] を選択します。実行時に、ファイル システムのリダイレクトが、System32 ファイルに対して、自動的に無効にされます。詳しくは、「64 ビット オペレーティング システムを InstallScript インストールでターゲットにする」を参照してください。*

64 ビット システムにおいて InstallScript インストールでセットアップの初期化中、WINSYSDIR64 変数の値は、64 ビット実行可能ファイルから Windows API 関数 ***GetSystemFolder*** *を呼び出して取得します。*

64 ビット システムにおいて基本の MSI と InstallScript MSI インストールでは、WINSYSDIR64 変数の値は Windows Installer プロパティ ***System64Folder*** *に基づいて初期化されます。遅延、コミット、およびロールバック カスタム アクションは、このプロパティーにアクセスすることはできません。そのため、対応する WINSYSDIR 変数は、遅延、コミット、およびロールバック カスタム アクションでは空白です。詳細は、「遅延／コミット／ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。*

WINSYSDISK

WINSYSDISK システム変数は Windows システム フォルダを含むディスク ドライブの名前を含みます (通常は C: ドライブ)。このフォルダーは、Windows のバージョンに合わせて、アプリケーション拡張機能 (DLL)、デバイスドライバー、および、他の Windows システムファイルを格納するために使用されます。InstallScript エンジンではドライブ名にコロン (:) が含まれる点に注意してください。Windows システムフォルダーの詳細については、InstallScript システム変数 WINSYSDIR の説明を参照してください。

メモ・この変数で指定されているドライブのルート フォルダーを参照する場合、変数に円記号 (¥) を追加する必要があります (2 つの円記号で指定)。たとえば WINSYSDISK の値が C: の場合、ステートメント WINSYSDISK + "¥¥" は C: ドライブのルート フォルダーを参照します。

***プロジェクト・**InstallScript インストールでセットアップの初期化中、WINSYSDIR 変数の値は、Windows API 関数 **GetSystemFolder** を呼び出して取得します。*

*基本の MSI および InstallScript MSI インストールの場合、WINSYSDIR 変数の値は、InstallScript 関数 **GetDisk** を WINSYSDIR と一緒に呼び出して取得します。値の取得に失敗した場合、この変数は Windows Installer プロパティ **WindowsVolume** に基づいて初期化されます。遅延、コミット、およびロールバック カスタム アクションは、このプロパティーにアクセスすることはできません。そのため、対応する WINSYSDIR 変数は、遅延、コミット、およびロールバック カスタム アクションでは空白です。詳細は、「遅延 / コミット / ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。*

プリプロセッサ ディレクティブ

プリプロセッサ ディレクティブは、スクリプトがコンパイルされる時に実行される InstallScript コンパイラに対する指示です。プリプロセッサ命令はコンパイラに対して、コンパイル内の別のソースファイルを含む、定数を定義する、コンパイルタイム条件に基づいてステートメントを含む、または除く、そしてユーザー定義のエラーメッセージを表示するといった指示を出します。InstallScript ディレクティブは C 言語のそれに似ていますが、全く同じという訳ではありません。

プリプロセッサ ディレクティブはシャープ記号 (#) で始まり、スクリプト内の任意の場所に挿入できます。それぞれの ディレクティブは独立した行とし、*セミコロンで 終了してはなりません。*

プリプロセッサ ディレクティブの使用

スクリプトでプリプロセッサ ディレクティブを使用する際には、次のガイドラインに従ってください。プリプロセッサ ディレクティブは、次の基準に準拠します。

- セミコロンで終了しない。
- 行を折り返さない。
- 長さが 250 文字を超えない。

メモ・条件 ディレクティブで利用される式は #define ディレクティブを使って定義された定数を含むことができます。変数を含むことはできません。

ブール演算子をプリプロセッサ ディレクティブで使用する

次のブール演算子は #if、#ifdef、#ifndef、そして #elif ステートメントでサポートされています :

- 論理 OR (||)
- 論理 AND (&&)
- 関係 (=、!=、>、>=、<、<=)

InstallScript がサポートするプリプロセッサ ディレクティブ

InstallScript は 次のプリプロセッサ ディレクティブをサポートします :

テーブル 1 ・ プリプロセッサ命令

プリプロセッサ ディレクティブ	説明
#define	シンボル定数を作成します。
#elif	#else と #if を 1 つのステートメントに組み合わせます。
#else	テストが失敗した場合に代替手段を示します。
#endif	プリプロセッサ条件 ディレクティブ (#if、#ifdef、#ifndef) を終了します。
#error	ユーザー定義のエラーメッセージを表示します。

テーブル 1・プリプロセッサ命令

プリプロセッサ ディレクティブ	説明
#if	条件ステートメントが true の場合にコンパイルします。
#ifdef	数値定数が定義されている場合にコンパイルします。
#ifndef	数値定数が定義されていない場合にコンパイルします。
#include	別のファイルの定数を含みます。
#undef	#define を使って定義された定数の定義を解除します。
# 警告	コンパイラー警告とユーザー定義のエラー メッセージを表示します。

#define

数値定数、または文字列定数を定義するには、#define を利用します。定数を定義してそれに値を割り当てるとき、InstallShield はその位置に関わらず定数を置換します。例えば、次の #define ステートメントは MAX_SIZE の値を 145 に設定します :

```
#define  MAX_SIZE      145
```

次の例では文字列定数を #define ディレクティブで宣言します。

```
##define  STR_MESSAGE   ″これはメッセージです。″
```

STR_MESSAGE を一度定義すれば、スクリプト上のどこでもそれを利用することができます。SprintfBox または MessageBox で表示する文字列メッセージは 255 文字以下でなくてはなりません。255 文字以上を表示する場合、表示する前に文字列を複数に分割してください。255 文字制限にはスペース、エスケープシーケンス、およびその他の特殊文字が含まれます。

メモ・定数を定義する別の方法として、[セットアップの設定] ダイアログ ボックスの [コンパイル] タブにある ″プリプロセッサ定義 ″ フィールドを使用することもできます。[セットアップの設定] ダイアログ ボックスでプリプロセッサ定義を追加または変更した場合、再コンパイル後に有効になります。

制限

#define ディレクティブについて、いくつかの制限があります :

- InstallShield は、数値や文字列の簡単な語彙置換を含むマクロのみを定義する #define の利用をサポートします。複数用語や演算子を利用する式を含むマクロを定義することはできません。
- #define ステートメントで宣言する定数は数字で始めることはできません。
- 多くの InstallShield 関数は定義済みの定数を使用します。定義済み定数を定義しようとすると、InstallShield スクリプトコンパイラはエラーメッセージを生成します
- InstallShield はゼロ (0) の値を定義されていない定数に割り当てます。

#elif

#elif コンパイルタイム ステート メントは機能については elseif ランタイムステート メントと類似しています。#if ステート メントと #else ステート メントを組み合わせて、別の条件を指定することができます。例 :

```
#if (A = 1)
   // A が 1 と等しい場合はコンパイルします
   . を参照してください。を参照してください。
#elif (A = 2)
   // A が 2 と等しい場合はコンパイルします
   . を参照してください。を参照してください。
#elif (B = 3)
   // B が 3 と等しい場合はコンパイルします
   . を参照してください。を参照してください。
#else
   // #elif 条件に true がひとつもない場合、次の部分を
   // コンパイルします
   . を参照してください。を参照してください。
#endif
```

メモ・#elif を利用する場合、#endif ひとつだけでセクションを終了します。

#error

#error ディレクティブを使って、コンパイルを中止してユーザー定義のエラーメッセージを表示します。表示するメッセージは #error の直後に入力し、ディレクティブから少なくとも 1 スペース離さなくてはなりません。

下の例では、定数 PRODUCTID は 1 または 2 と同じでなくてはなりません。PRODUCTID の値がその範囲内でない場合、定数 PRODUCTNAME は定義されず、ユーザー定義のエラーメッセージが表示されます。

```
#define PRODUCTID 1

#if (PRODUCTID = 1)
  #define PRODUCTNAME "Lite"
#elif (PRODUCTID = 2)
  #define PRODUCTNAME "Professional"
#endif

#ifndef PRODUCTNAME
  #error PRODUCTID out of range.
#endif
```

#if...#else...#endif

#if ステート メントを利用してコンパイルする行を選択してください。インストレーションンの様々なセクションのスイッチをオンまたはオフにして、より柔軟なスクリプトを作成することができます。#if ステート メントはランタイム if ステート メントと同じ要領で動作します :

```
#if (A = 1)
   // A が 1 と等しい場合はコンパイルします
   . を参照してください。を参照してください。
#else
   // コンパイル
```

. を参照してください。を参照してください。
#endif

#if を利用する際には、以下の制限事項に注意して下さい :

- #if ステートメントのフォーマットは、ランタイム if ステートメントのそれと同じで、#if ステートメントはキーワード #endif で終了しなくてはなりません。
- #if または #elif ステートメントでは数値定数のみをテストすることができます。
- InstallScript では、#if ステートメントのネストは最大 10 階層まで可能です。

#ifdef と #ifndef

指定した式が #defin で定義されている場合のみ、セクションをコンパイルするのに #ifdef ステートメントを利用してください。指定した式が定義されていない場合のみ、セクションをコンパイルするのに #ifndef ステートメントを利用してください。

例

#ifdef A
// A が定義されていればコンパイルする。
. を参照してください。を参照してください。
#endif

#ifndef A
// A が定義されていればスキップする。その他の場合はコンパイルする。
. を参照してください。を参照してください。
#endif

#ifdef A // A が定義されていればコンパイルする。. を参照してください。.#endif#ifndef A //A が定義されていればスキップする。その他の場合はコンパイルする。. を参照してください。.#endif

#else と #elif と共に、#ifdef と #ifndef を利用することもできます。

#ifdef nFilePath
// ステートメント
#else
// ステートメント
#endif

メモ・プリプロセッサ定義は、[プロジェクトの設定] ダイアログ ボックスの [コンパイル / リンク] タブにある [プリプロセッサの定義] 編集ボックスで入力できます。[プロジェクトの設定] ダイアログ ボックスでプリプロセッサ定義を追加または変更した場合、再コンパイル後に有効になります。

制限

#ifdef と #ifndef ステートメントを利用する場合、次の制限事項に注意して下さい :

- #ifdef と #ifndef ディレクティブと同じ行にコメントを配置することはできません。
- #ifdef または #ifndef ステートメントを使って 0 (ゼロ) の値を持つ定数をテストしてはいけません。
- #ifdef または #ifndef ステートメントでは数値定数のみをテストすることができます。

#include

#include ステートメントを利用して別のスクリプトの内容をメインのインストールスクリプトへ含みます。#include を利用する際、コンパイラは追加ソーススクリプトをメインインストレーションスクリプトの一部と同様に扱います。追加スクリプトまたはインクルードファイルは、変数宣言、その他のコンパイラ ディレクティブ、そしてプログラムステートメントを含むことがあります。

例えば、ユーザー定義の定数定義すべてを含む新たなファイルを作成し、それを #include ステートメントを使ってスクリプトファイルへ挿入することができます。後程、定数の何れかを再定義する必要がある場合、そのすべては中央にひとつにまとめられています。

InstallShield からコンパイルする際、InstallShield は次に示す順番でインクルード スクリプト ファイルを検索します :

1. プロジェクトのスクリプト ファイル ディレクトリ
2. [設定] ダイアログ ボックスの [コンパイル / リンク] タブにある " インクルード パス " 設定で指定されたディレクトリ。
3. InstallShield インクルードディレクトリ

2 つのインクルード ファイルが同じ名前を持っているが、異なる場所に存在する場合、InstallShield は前述の順序に従って最初に検出されたファイルにリンクします。また、インクルード ファイルが相対パスで指定された場合、InstallShield は前述のディレクトリに相対するパスを順番に検索します。

インクルード ファイルがこれらの場所で検出されなかった場合、#include ステートメントで完全修飾ファイル名を指定します。#include ステートメントを利用するとき、ファイル名またはパスを二重引用符で囲んで ("filename") 指定します。

スクリプトで #include ディレクティブを使用する際、次の点にご注意ください :

- InstallShield では、260 文字 (ファイル名を含む) より長いパスを処理できません。
- InstallScript では、インクルードファイルのネストは最大 8 階層まで可能です。
- InstallShield プリプロセッサは #include ディレクティブで円記号を制御文字として認識しません。パスを指定する場合、フォルダー名を分けるには 2 つの円記号の変わりに 1 つの円記号を利用してください。
- スクリプトで C 言語ヘッダー ファイルを含まないで下さい。InstallShield コンパイラは C の構造の一部を認識しません。InstallScript のみを利用してヘッダーファイルを作成します。

次の例では、#include ステートメントを利用するインストールスクリプトのセクションが **Support.rul** またはその他のファイルの内容を含みます。#include ステートメントが参照する各ソーススクリプトは特定の目的で書かれたもので、スクリプトがコンパイルされる時に追加されます。

```
// 次のインクルードファイルは、インストレーションに特定のルーチンを含みます。
#include "SUPPORT.RUL"

// 変数とプロトタイプ宣言を含むローカルインクルードファイル。
#include "DECLARE.RUL"


// LIBRARY ディレクトリからスクリプトを含みます
#include "..¥LIBRARY¥SYSCHK.H"

// DIALOGS ディレクトリからスクリプトを含みます
#include "..¥DIALOGS¥WELCOME¥WELCOME.H"
#include "..¥DIALOGS¥REGINS¥REGINS.H"
```

```
#include "..¥DIALOGS¥ICONS¥ICONS.H"
```

#undef

#define で以前に定義された定数を未定義にするには #undef ディレクティブを利用します。以前に #define ディレクティブ以外を使って定義された定数をこの ディレクティブを使って指定した場合、スクリプトをコンパイルしたときにエラーが発生します。次の例では 2 つの定数が定義され、2 番目が定義された場合に最初の定数が未定義となります。

```
#define NORMSETUP
#define BONUSPAK
```

. を参照してください。を参照してください。

```
#ifdef NORMSETUP
   MessageBox(' 標準セットアップのコンパイル。',INFORMATION);
#else
   MessageBox(' スーパーセットアップのコンパイル。',INFORMATION);
#endif

#ifdef BONUSPAK
   #undef NORMSETUP
#endif

#ifdef NORMSETUP
   MessageBox(' 標準セットアップのコンパイル。',INFORMATION);
#else
   MessageBox(' スーパーセットアップのコンパイル。',INFORMATION);
#endif
```

警告

#warning ディレクティブを使って、コンパイラー警告およびユーザー定義のエラーメッセージを表示します。表示するメッセージは #warning の直後に入力し、ディレクティブから少なくとも 1 スペース離さなくてはなりません。

下の例では、定数 PRODUCTID は 1 または 2 と同じでなくてはなりません。PRODUCTID の値がその範囲内でない場合、定数 PRODUCTNAME は定義されず、ユーザー定義の警告メッセージが表示されます。

```
#define PRODUCTID 1
#if (PRODUCTID = 1)
   #define PRODUCTNAME "Lite"
#elif (PRODUCTID = 2)
   #define PRODUCTNAME "Professional"
#endif
#ifndef PRODUCTNAME
   #warning PRODUCTID が範囲外です。
#endif
```

イベント ハンドラー

プロジェクト・*InstallScript イベント ハンドラーは、次のプロジェクト タイプで使用できます:*

- *InstallScript*
- *InstallScript MSI*

アドバンスト UI またはスイート / アドバンスト UI インストールに含まれている InstallScript パッケージには、いくつかの例外があります。詳細は次のとおりです。

イベントハンドラーについて

InstallScript インストールは InstallScript エンジンによって制御され、決められた順序で一連のイベントが生成されます。これらのイベントは、インストールの支持を実行するソフトウェアハンドラーをトリガーします。たとえば、インストールがロードされた直後に Begin というイベントが生成されます。このイベントは、OnBegin というイベント ハンドラーの実行をトリガーします。このハンドラーは、[Begin] イベントが発生する際に、実行する操作を指定します。インストレーションの他のイベントは、別のハンドラーをトリガーします。いずれにしても、イベントハンドラーは、アプリケーションをインストールする作業を実行します。

InstallScript MSI インストールの一部は InstallScript エンジン、および別の一部は Windows Installer エンジンによって駆動します。InstallScript エンジンによって駆動する部分は、InstallScript インストールに類似する一連のイベント ハンドラーを使用します。

いくつかの種類のイベント ハンドラーを使用できます:

- グローバル イベント ハンドラー
- 機能イベント ハンドラー
- その他のイベントハンドラー
- 拡張イベントハンドラー

InstallScript または InstallScript MSI プロジェクトを作成した際、InstallShield はデフォルトのグローバル イベント ハンドラーを生成します。各イベントハンドラーは InstallScript 言語でスクリプトされた関数です。同様に、プロジェクトに機能を追加する際、InstallShield は、デフォルトのイベントハンドラーセットをその機能用に作成します。また、イベントハンドラーのいずれか、またはすべてを上書きしたり、カスタマイズすることができます。

ここで重要なことは、イベントドリブン型スクリプトでは、InstallScript ビューではっきり表示されていなくてもイベントハンドラー関数が呼び出されるという点です。

イベントハンドラーの呼び出し順序

グローバルイベントハンドラーと機能イベントハンドラーは常に決められた順序で呼び出されます。イベントハンドラーはインストレーションの種類 (通常インストール、メンテナンスインストール、管理インストール、またはパッチインストール) に従って呼び出されます。その他のイベントハンドラーはインストレーションの最中に発生しないイベントへも応答するので、それらが呼び出されるときは特定の順序はありません。

プロジェクト・*InstallScript MSI メジャー アップグレードが製品の以前のバージョンをアンインストールするとき、InstallScript イベント ハンドラーが呼び出されることはありません。*

初回インストール

- OnBegin
- OnCCPSearch
- OnAppSearch
- OnFirstUIBefore
- OnGeneratingMSIScript (InstallScript MSI のみ)
- OnMoving
- 機能インストール中イベント
- OnInstallFilesActionBefore
- OnGeneratedMSIScript (InstallScript MSI のみ)
- OnInstallFilesActionAfter
- 機能インストール済みイベント
- OnMoved
- OnFirstUIAfter
- OnEnd

再開されたインストール

- OnResumeUIAfter (InstallScript MSI のみ)
- OnResumeUIBefore (InstallScript MSI のみ)

メンテナンス インストール

- OnBegin
- OnMaintUIBefore
- OnGeneratingMSIScript (InstallScript MSI のみ)
- OnMoving
- 機能のインストール中またはアンインストール中のイベント
- OnInstallFilesActionBefore
- OnInstallFilesActionAfter
- 機能インストール済みまたはアンインストール済みイベント
- OnMoved
- OnGeneratedMSIScript (InstallScript MSI のみ)
- OnMaintUIAfter
- OnEnd

パッチ インストール

- OnPatchUIBefore (InstallScript MSI のみ)
- OnGeneratingMSIScript (InstallScript MSI のみ)
- OnMoving
- 機能のインストール中またはアンインストール中のイベント
- OnInstallFilesActionBefore
- OnInstallFilesActionAfter
- 機能インストール済みまたはアンインストール済みイベント
- OnMoved
- OnGeneratedMSIScript (InstallScript MSI のみ)
- OnPatchUIAfter (InstallScript MSI のみ)

アドバンスト UI またはスイート / アドバンスト UI インストールに含まれている InstallScript パッケージの除外

アドバンスト UI またはスイート / アドバンスト UI プロジェクトに InstallScript インストールを InstallScript パッケージとして含める場合、それ自身のユーザー インターフェイス (UI) が表示され、InstallScript パッケージの UI は自動的に抑制されます。これらの変更を可能にするために、アドバンスト UI またはスイート / アドバンスト UI インストールでは、デフォルトで、いくつかのアドバンスト ＵＩ またはスイート / アドバンスト ＵＩ 固有の InstallScript イベントおよび関数が使用され、一部の標準 InstallScript イベントおよび関数は無視されます。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート / アドバンスト UI プロジェクトに追加する」をご覧ください。

インストール状態 (初回インストール、メンテナンス、またはアップデート) に応じて、OnSuiteShowUI は、OnFirstUIBefore や OnFirstUIAfter などの UI イベントを無視し、次のイベントを呼び出します :

- **初回インストール** – OnSuiteInstallBefore、OnSuiteInstallAfter
- **メンテナンス** – OnSuiteMaintBefore、OnSuiteMaintAfter
- **アップデート** – OnSuiteUpdateBefore、OnSuiteUpdateAfter

アドバンスト UI またはスイート / アドバンスト UI インストールで起動された、その他すべてのイベントおよびイベントの呼び出しシーケンスは、アドバンスト UI またはスイート / アドバンスト UI インストールから個別に起動された InstallScript インストール、または、アドバンスト UI またはスイート / アドバンスト UI インストールから実行可能パッケージとして起動された InstallScript インストール内のそれらと同一に保持されます。

イベントハンドラー インデックス

イベントハンドラーは、セットアップ中に発生するイベントへ応答する形で呼び出される InstallScript 関数です。これらのハンドラー名は予約済みです。(スクリプトでこれらの関数をプロトタイプ化する必要はありません。)

プロジェクト・次のイベント ハンドラーの一部は、InstallScript または InstallScript MSI プロジェクトにのみ適用されます。両方に適用されるものもあります。サポートされているプロジェクトは、プロジェクトの種類列を参照してください。

テーブル 1・ イベントハンドラー インデックス

イベント ハンドラー	プロジェクトの種類
OnAbort	InstallScript、InstallScript MSI
OnAdminInstallUIAfter	InstallScript MSI
OnAdminInstallUIBefore	InstallScript MSI
OnAdminPatchUIAfter	InstallScript MSI
OnAdminPatchUIBefore	InstallScript MSI
OnAdvertisementAfter	InstallScript MSI
OnAdvertisementBefore	InstallScript MSI
OnAppSearch	InstallScript、InstallScript MSI
OnBegin	InstallScript、InstallScript MSI
OnCanceling	InstallScript、InstallScript MSI
OnCCPSearch	InstallScript、InstallScript MSI
OnCheckMediaPassword	InstallScript
OnComponentError	InstallScript、InstallScript MSI
OnCustomizeUninstInfo	InstallScript
OnDIFxLogCallback	InstallScript
OnEnd	InstallScript、InstallScript MSI
OnError	InstallScript MSI
OnException	InstallScript MSI
OnFileError	InstallScript
OnFileLocked	InstallScript
OnFileReadOnly	InstallScript

テーブル 1・ イベントハンドラー インデックス（続き）

イベント ハンドラー	プロジェクトの種類
OnFilesInUse	InstallScript MSI
OnFilterComponents	InstallScript、InstallScript MSI
OnFirstUIAfter	InstallScript、InstallScript MSI
OnFirstUIBefore	InstallScript、InstallScript MSI
OnGeneratedMSIScript	InstallScript MSI
OnGeneratingMSIScript	InstallScript MSI
OnHelp	InstallScript、InstallScript MSI
OnIISComponentInstalled	InstallScript、InstallScript MSI
OnIISInitialize	InstallScript、InstallScript MSI
OnIISUninitialize	InstallScript、InstallScript MSI
OnIISVRootUninstalling	InstallScript、InstallScript MSI
OnInstalled	InstallScript、InstallScript MSI
OnInstalledFile	InstallScript
OnInstalledFontFile	InstallScript
OnInstallFilesActionAfter	InstallScript、InstallScript MSI
OnInstallFilesActionBefore	InstallScript、InstallScript MSI
OnInstalling	InstallScript、InstallScript MSI
OnInstallingFile	InstallScript
OnInternetError	InstallScript
OnLaunchAppAndWaitCallback	InstallScript、InstallScript MSI
OnLogonUserSetMsiProperties	InstallScript MSI
OnMaintUIAfter	InstallScript、InstallScript MSI
OnMaintUIBefore	InstallScript、InstallScript MSI
OnMD5Error	InstallScript

テーブル 1・ イベントハンドラー インデックス（続き）

イベント ハンドラー	プロジェクトの種類
OnMoved	InstallScript、InstallScript MSI
OnMoveData	InstallScript
OnMoving	InstallScript、InstallScript MSI
OnMsiSilentInstall	InstallScript MSI
OnNetApiCreateUserAccount	InstallScript、InstallScript MSI
OnNextDisk	InstallScript
OnOutOfDiskSpace	InstallScript MSI
OnPatchUIAfter	InstallScript MSI
OnPatchUIBefore	InstallScript MSI
OnRebooted	InstallScript InstallScript MSI – InstallScript ユーザー インターフェイス (UI) のスタイルが、外部 UI ハンドラーとして InstallScript エンジンを使用する従来型のスタイルの場合
OnRemovingSharedFile	InstallScript
OnResumeUIAfter	InstallScript MSI
OnResumeUIBefore	InstallScript MSI
OnRMFilesInUse	InstallScript MSI
OnSelfRegistrationError	InstallScript
OnSetTARGETDIR	InstallScript
OnSetUpdateMode	InstallScript
OnShowUI	InstallScript
OnSuiteInstallAfter	InstallScript
OnSuiteInstallBefore	InstallScript
OnSuiteMaintAfter	InstallScript
OnSuiteMaintBefore	InstallScript

テーブル 1・ イベントハンドラー インデックス（続き）

イベント ハンドラー	プロジェクトの種類
OnSuiteUpdateAfter	InstallScript
OnSuiteUpdateBefore	InstallScript
OnSuiteShowUI	InstallScript
OnSQLBatchScripts	InstallScript
OnSQLComponentInstalled	InstallScript
OnSQLComponentUninstalled	InstallScript
OnSQLLogin	InstallScript
OnSQLServerInitialize	InstallScript
OnSQLServerInitializeMaint	InstallScript
OnUninstall	InstallScript、InstallScript MSI
OnUnInstalled	InstallScript、InstallScript MSI
OnUninstalledFile	InstallScript
OnUnInstalling	InstallScript、InstallScript MSI
OnUninstallingDIFxDriverFile	InstallScript
OnUninstallingFile	InstallScript
OnUninstallingFontFile	InstallScript
OnUpdateUIAfter	InstallScript
OnUpdateUIBefore	InstallScript
OnWarning	InstallScript MSI
OnXMLComponentInstalled	InstallScript
OnXMLComponentUninstalling	InstallScript
OnXMLInitialize	InstallScript
OnXMLUninitialize	InstallScript

コンポーネント イベント ハンドラー

Windows Installer インストールは、プロジェクトの最上位レベルの構造にコンポーネントではなく機能を使用します。したがってコンポーネント イベント ハンドラーが機能イベント ハンドラーと置換されています。さらに、サポートされているすべてのコンポーネント関数には、その代替となる機能があります。詳細については、「機能関数」および「機能イベント ハンドラー」を参照してください。

グローバル イベント ハンドラー

グローバルイベントハンドラーは機能のインストレーションまたはアンインストレーションの前後に必要な処理を行います。次のカテゴリに分類されるイベントハンドラーが含まれます :

テーブル 2 · グローバルイベントハンドラーのカテゴリ

分類	説明
初期化ハンドラー	インストールエンジンで直接呼び出されるイベントハンドラー。
Before Move Data ハンドラー	機能がターゲットコンピューターにインストール、またはアンインストールされる前にトリガーされるイベントハンドラー。
Move Data ハンドラー	機能がターゲットコンピューターにインストール、またはアンインストールされる直前、または直後にトリガーされるイベントハンドラー。
After Data Move ハンドラー	機能がターゲットコンピューターにインストール、またはアンインストールされた後にトリガーされるイベントハンドラー。

初期化ハンドラー

InstallScript プロジェクトで、以下のイベントハンドラーはインストールエンジンによって直接呼び出されます。

テーブル 3 · Initialization ハンドラー

イベント ハンドラー	プロジェクトの種類	説明
OnCheckMediaPassword	InstallScript	インストール ログ ファイルにインストール メディアのパスワードが既に記録されていない場合 (インストールがメンテナンスまたはアンインストール モードで実行される場合は記録済み) で、リリース ウィザードの [パスワード] パネルで [セットアップの初期化中にパスワード ダイアログ ボックスを表示する] チェック ボックスが選択されているか、[リリース] ビューの " パスワード ダイアログの表示 " プロパティが [はい] に設定されているとき、エンドユーザーにパスワードを問い合わせるために初期化中にフレームワークによって直接呼び出されます。

テーブル 3 ・ Initialization ハンドラー

イベント ハンドラー	プロジェクトの種類	説明
OnFilterComponents	InstallScript、InstallScript MSI	各機能のコンポーネントを言語およびプラットフォーム別にフィルターするため、フレームワークによって直接呼び出されます。カスタムフィルターを実行するには、このイベントを上書きします。
OnIISCheckRequirements	InstallScript	このイベント ハンドラーは現在使用されていません。
OnSetTARGETDIR	InstallScript	初期化中 TARGETDIR をデフォルト値に初期化するためにフレームワークによって直接呼び出されます。
OnSetUpdateMode	InstallScript	初期化中、どの UI イベントが OnShowUI によって呼び出されるのかを制御する UPDATEMODE システム変数を適切に設定するため、にフレームワークによって直接呼び出されます。

OnCheckMediaPassword

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

OnCheckMediaPassword イベント ハンドラー関数はセットアップ エンジンによって直接呼び出されます。ハンドラーのデフォルト コードは、セットアップ ログ ファイルにパスワードが既に記録されていない場合 (インストールがメンテナンスまたはアンインストール モードで実行される場合は記録済み) で、リリース ウィザードの [パスワード] パネルで [セットアップの初期化中にパスワード ダイアログ ボックスを表示する] チェック ボックスが選択されているか、[リリース] プロパティ シートの " パスワード ダイアログの表示 " プロパティが [はい] に設定されているとき、エンド ユーザーにセットアップのパスワードを問い合わせます。

このイベント ハンドラーは手続き型スクリプト (program...endprogram ブロックを含むスクリプト) を利用するセットアップの (適切な場所で) 呼び出されます。

構文

OnCheckMediaPassword ( );

パラメーター

なし。

戻り値

なし。

OnFilterComponents

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

OnFilterComponents イベント ハンドラーは、機能フィルターを処理するため InstallScript エンジンによって直接呼び出されます。機能フィルター処理とは、言語とオペレーティング システムの設定に基づいてファイル転送内の機能コンポーネントを含むまたは除外する処理です。

構文

OnFilterComponents ( );

パラメーター

なし。

戻り値

なし。

追加情報

デフォルトでは、OnFilterComponents は FeatureFilterLanguage を呼び出して、SELECTED_LANGUAGE システム変数で指定された言語以外の言語を持つすべてのコンポーネントを除外し、FeatureFilterOS を呼び出して SYSINFO 変数の nISOSL メンバーによって指定された以外のオペレーティング システムを持つすべてのコンポーネントを除外します。

このイベント ハンドラーは手続き型スクリプト（program...endprogram ブロックを含むスクリプト）を利用するインストールでは呼び出されません。

OnSetTARGETDIR

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

OnSetTARGETDIR イベント ハンドラー関数は、システム変数 TARGETDIR の値を設定するためにセットアップ エンジンが直接呼び出します。

このイベント ハンドラーは手続き型スクリプト（program...endprogram ブロックを含むスクリプト）を使用するセットアップをはじめとする、あらゆるセットアップで呼び出されます。

構文

OnSetTARGETDIR ( );

パラメーター

なし。

戻り値

なし。

追加情報

- デフォルトでは、初回インストール OnSetTARGETDIR は TARGETDIR を [一般情報] ビューの TARGETDIR 設定で指定した値に設定するか、または InstallShield で値を指定していない場合は **FOLDER_APPLICATIONS¥IFX_COMPANY_NAME¥IFX_PRODUCT_NAME** に設定します。<FOLDER_APPLICATIONS>、<IFX_COMPANY_NAME>、および <IFX_PRODUCT_NAME> は、TARGETDIR が参照された時に値が解決されるテキスト代替で、OnSetTARGETDIR が呼び出された後にシステム変数 IFX_COMPANY_NAME または IFX_PRODUCT_NAME の値を変更した場合、その変更は後に続く TARGETDIR への参照に反映されます。
- メンテナンスインストールまたはアンインストールは TARGETDIR をログファイルに格納されている値に初期化し、デフォルトの OnSetTARGETDIR が TARGETDIR の値を変更することはありません。

OnSetUpdateMode

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

OnSetUpdateMode イベント ハンドラー関数はセットアップ エンジンが直接呼び出し、セットアップが既存インストレーションのアップデートで、システム変数 UPDATEMODE の値を適切に設定するかどうかを判断します。

このイベント ハンドラーは手続き型スクリプト (program...endprogram ブロックを含むスクリプト) を利用するセットアップの (適切な場所で) 呼び出されます。

構文

```
OnSetUpdateMode ( );
```

パラメーター

なし。

戻り値

なし。

Before Move Data ハンドラー

次のイベントハンドラーは、ファイルがターゲットコンピューターに転送される前にトリガーされます。さらに、これらのイベントのほとんどは、メンテナンスモード中にもトリガーされます。

テーブル 4 · Before Move Data ハンドラー

イベント ハンドラー	プロジェクトの種類	説明
OnBegin	InstallScript InstallScript MSI	InstallScript プロジェクトの場合 : 初期化イベントの後にフレームワークによって直接呼び出されます。 InstallScript MSI プロジェクトの場合 : [Begin] イベントに応答します。セットアップ内であらかじめ定義できる最初のイベントです。
OnAppSearch	InstallScript InstallScript MSI	InstallScript プロジェクトの場合 : OnBegin の後にフレームワークによって直接呼び出されます。 InstallScript MSI プロジェクトの場合 : pplication Search イベントに応答します。インストールでターゲットコンピューター上の特定のアプリケーションを検索する必要がある場合、このハンドラーをセットアップのコードに追加します。
OnCCPSearch	InstallScript InstallScript MSI	InstallScript プロジェクトの場合 : AppSearch の後にフレームワークによって直接呼び出されます。 InstallScript MSI プロジェクトの場合 : Upgrade Compliance イベントに応答します。エンドユーザーがアプリケーションをインストールするのに必須となるアプリケーションを検索する必要があるインストールでこのハンドラーをコードにして下さい。

テーブル 4 ・ Before Move Data ハンドラー

イベント ハンドラー	プロジェクトの種類	説明
OnFirstUIBefore	InstallScript InstallScript MSI	***メモ ・** InstallScript インストールが、InstallScript パッケージとして、アドバンスト UI またはスイート／アドバンスト UI プロジェクトに含まれている場合、アドバンスト UI またはスイート／アドバンスト UI インストールで、このイベントハンドラーは呼び出されません。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート／アドバンスト UI プロジェクトに追加する」をご覧ください。* InstallScript プロジェクトの場合 : セットアップが初回インストールモードで実行中、フレームワークによって呼び出されます。デフォルトでこのイベントは、エンドユーザーがインストールパラメーターを指定できる UI を表示します。 InstallScript MSI プロジェクトの場合 : アプリケーションの初期インストールの際にエンドユーザーから情報を収集したダイアログを表示して、[First UI Before] イベントに応答します。
OnMaintUIBefore	InstallScript InstallScript MSI	***メモ ・** InstallScript インストールが、InstallScript パッケージとして、アドバンスト UI またはスイート／アドバンスト UI プロジェクトに含まれている場合、アドバンスト UI またはスイート／アドバンスト UI インストールで、このイベントハンドラーは呼び出されません。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート／アドバンスト UI プロジェクトに追加する」をご覧ください。* InstallScript プロジェクトの場合 : インストールがメンテナンス モードで実行中、OnShowUI によって呼び出されます。OnShowUI をカスタマイズして、このイベントが呼び出されるかどうかを制御することができます。 InstallScript MSI プロジェクトの場合 : アプリケーションのメンテナンス インストールの際に、エンドユーザーから情報を収集するためのダイアログを表示して、MaintenanceUIBefore イベントに応答します。

テーブル 4 ・ Before Move Data ハンドラー

イベント ハンドラー	プロジェクトの種類	説明
OnUpdateUIBefore	InstallScript	 ***メモ ・*** *InstallScript インストールが、InstallScript パッケージとして、アドバンスト UI またはスイート／アドバンスト UI プロジェクトに含まれている場合、アドバンスト UI またはスイート／アドバンスト UI インストールで、このイベントハンドラーは呼び出されません。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート／アドバンスト UI プロジェクトに追加する」をご覧ください。* セットアップがアップデートモードで実行中、OnShowUI によって呼び出されます。デフォルトでこのイベントは、エンドユーザーがアプリケーションをカレントバージョンにアップデートするための UI を表示します。
OnSuiteInstallBefore	InstallScript	OnSuiteShowUI イベントによって、OnSuiteInstallBefore イベントが呼び出されます。デフォルトで、OnSuiteInstallBefore は、ファイルの転送に必要な機能の選択など、情報の初期化のために呼び出されます。  ***メモ ・*** *program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*
OnSuiteMaintBefore	InstallScript	OnSuiteShowUI イベントによって、OnSuiteMaintBefore イベントが呼び出されます。デフォルトで、OnSuiteMaintBefore は、ファイルの転送に必要な機能の選択など、情報の初期化のために呼び出されます。  ***メモ ・*** *program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*
OnSuiteUpdateBefore	InstallScript	OnSuiteShowUI イベントによって、OnSuiteMaintBefore イベントが呼び出されます。デフォルトで、OnSuiteMaintBefore は、ファイルの転送に必要な機能の選択など、情報の初期化のために呼び出されます。  ***メモ ・*** *program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*

テーブル 4 ・ Before Move Data ハンドラー

イベント ハンドラー	プロジェクトの種類	説明
OnSQLLogin	InstallScript MSI	InstallScript MSI プロジェクトの場合 : First UI Before イベントに応答します。このイベントハンドラー関数は、SQL ログイン情報を指定するためのスクリプトで使われるダイアログを作成します。情報には、ログイン ID とパスワードが含まれます。
OnSQLServerInitialize	InstallScript	SQL Server サポートに必要な接続を確立するため、OnFirstUIBefore によって呼び出されます。この関数は SQL Server ランタイムを初期化し、各接続ごとにログイン ダイアログを表示しながら、必要な SQL Server への接続の確立を試みます。
OnSQLServerInitializeMaint	InstallScript	SQL Server サポートに必要な接続を確立するため、OnMaintUIBefore によって呼び出されます。
OnIISInitialize	InstallScript	OnIISInitialize イベントは、OnFirstUIBefore によって呼び出され、IIS ランタイムを初期化します。 ***メモ ・** program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*
OnXMLInitialize	InstallScript	OnXMLInitialize イベントは、OnFirstUIBefore によって呼び出され、XML ランタイムを初期化します。 ***メモ ・** program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*

OnAppSearch

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

- *InstallScript*
- *InstallScript MSI*

OnAppSearch イベントハンドラーは Application Search イベントに応答します。セットアップでターゲットコンピューター上の特定のアプリケーションを検索する必要がある場合、このハンドラーをセットアップのコードに追加します。たとえば、コードで FindFile を呼び出してキーファイルを検索したり、RegDBKeyExist を呼び出してレジストリ エントリを検索することができます。

次の OnAppSearch 関数は、Notepad.exe を呼び出したファイルがユーザーの Windows、WinNT フォルダー、あるいはそのサブディレクトリで無効な場合、インストールを中止します。

```
function OnAppSearch()
    NUMBER nResult;
    STRING svIgnore;
begin
    nResult =
        FindAllFiles(WindowsFolder, "Notepad.exe",
                     svIgnore, RESET);

    if (nResult < 0) then
        MessageBox(" 認定するプログラムを検出できません。" +
                   " セットアップを終了します。", SEVERE);
        abort;
    endif;
end;
```

メモ・このイベントハンドラーのコードは、メンテナンスセットアップ中またはアンインストール中には実行されません。

OnBegin

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

OnBegin イベントハンドラーは、セットアップで最初に再定義できる Begin イベントに応答します。スクリプトで他の要素をこのハンドラーに含む前に実行しなくてはならないコード。たとえば、ここでユーザーのマシンが製品のシステム要件と一致したことを確認することができます。

スクリプトに iswi.h または ifx.h を 含むときに OnBegin はプロトタイプ化されます。次の例に従って、スクリプトで OnBegin を定義してください :

```
#include "iswi.h"

function OnBegin()
    // ローカル変数
begin
    // 始まりのコード
end;
```

たとえば、インストールを続行する前に特定のレジストリキーが存在するか否かを確認する OnBegin 関数は次の様に示されます :

```
// HKLM¥Software¥InstallShield が存在しない場合はインストールを中止します。
function OnBegin()
    NUMBER nReturn;
begin

// ルートキーの設定
RegDBSetDefaultRoot(HKEY_LOCAL_MACHINE);

// サブキーの存在を確認します。
nReturn = RegDBKeyExist("Software¥¥InstallShield");
```

```
if (nReturn < 0) then
    MessageBox(" ご利用のシステムは、システム要件を満たしていません。" +
            " セットアップを終了します。", SEVERE);
    abort;
endif;

end;
```

このイベントハンドラーのコードは、メンテナンスセットアップやアンインストール中でも、次の if-then 構造に置かれていない場合には常に実行されます。

```
if (!MAINTENANCE) then
    // メンテナンス以外のコード
endif;
```

このセットアップにより、システム変数 MAINTENANCE は、セットアップの最初の実行時には FALSE になり、それ以後のセットアップの実行時には TRUE になります。

OnCCPSearch

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

OnCCPSearch イベントハンドラーは Upgrade Compliance イベントに応答します。エンドユーザーがアプリケーションをインストールするのに必須となるアプリケーションを検索する必要があるセットアップではこのハンドラーをコードにして下さい。コードは、たとえば特定のファイルを指定する FindFile の呼び出しを行います。このイベントハンドラーのコードは、メンテナンスセットアップ中またはアンインストール中には実行されないので注意してください。

OnFirstUIBefore

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

InstallScript インストールが、InstallScript パッケージとして、アドバンスト UI またはスイート / アドバンスト UI プロジェクトに含まれている場合、アドバンスト UI またはスイート / アドバンスト UI インストールで、このイベント ハンドラーは呼び出されません。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート / アドバンスト UI プロジェクトに追加する」をご覧ください。

OnFirstUIBefore イベントハンドラーは、First UI Before イベントに応答します。これは、アプリケーションの初回インストール用の機能をインストールする前に処理しなくてはならないタスクを実行します。

通常、このハンドラーは次の処理のために InstallScript 関数を呼び出します :

- 画面をセットアップする。
- [ようこそ] ダイアログ、ソフトウェアライセンス、およびインストールするソフトウェアについての情報をエンドユーザーに表示する。

- エンド ユーザーから、ユーザー登録、インストール先のパス (InstallScript プロジェクトでは TARGETDIR、InstallScript MSI プロジェクトでは INSTALLDIR)、およびセットアップの種類をはじめとする情報を取得する。

OnIISInitialize

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

OnIISInitialize は、OnMoving の前に呼び出されます。Web サイトと仮想ルートをインストールする前に、OnIISVRootUninstalling と似た目的で、これを上書きしたり、IIS のバージョンをチェックするコードを追加したり、IIS をカスタマイズしたりすることができます。

OnMaintUIBefore

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

InstallScript インストールが、InstallScript パッケージとして、アドバンスト UI またはスイート / アドバンスト UI プロジェクトに含まれている場合、アドバンスト UI またはスイート / アドバンスト UI インストールで、このイベント ハンドラーは呼び出されません。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート / アドバンスト UI プロジェクトに追加する」をご覧ください。

OnMaintUIBefore イベント ハンドラーは、Maintenance UI Before イベントに応答します。このイベント ハンドラーでは、アプリケーションのメンテナンス インストールによって機能が再インストールされる前に発生するタスクが実行されます。

OnSQLLogin

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

OnSQLLogin イベントは、First UI Before イベントに応答します。このイベントハンドラー関数は、SQL ログイン情報を指定するためのスクリプトで使われるダイアログを作成します。情報には、ログイン ID とパスワードが含まれます。

メモ・OnSQLLogin イベントが呼び出される前に SQL ビルトイン関数を呼び出す場合、まず SQLRTInitialize2 関数を呼び出します。これは、すべての SQL 関連の関数に適用します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。

OnSQLServerInitialize

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

OnSQLServerInitialize イベントは、OnFirstUIBefore よって、SQL Server サポートに必要な接続を確立するために呼び出されます。この関数は SQL Server ランタイムを初期化し、各接続ごとにログイン ダイアログを表示しながら、必要な SQL Server への接続の確立を試みます。パラメーター nBtn は NEXT または BACK がその前に表示されたダイアログの結果かどうかを示します。これは情報提供のみの目的です。

InstallShield のより新しいバージョンにアップグレードした時に OnFirstUIBefore をオーバーライドしてしまっているスクリプトを使って作業していて、そのスクリプトが OnSQLServerInitialize を呼び出さない場合、OnFirstUIBefore コードをスクリプトファイルに追加する必要があります。

***メモ・**OnSQLServerInitialize イベントが呼び出される前に SQL ビルトイン関数を呼び出す場合、まず SQLRTInitialize2 関数を呼び出します。これは、すべての SQL 関連の関数に適用します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

OnSQLServerInitializeMaint

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

OnSQLServerInitializeMaint イベントは、OnMaintUIBefore によって、SQL Server サポートに必要な接続を確立するために呼び出されます。この関数は SQL Server ランタイムを初期化し、ログファイルに格納されているログイン認証情報を使って、必要な SQL Server への接続の確立を試みます。

***メモ・**program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*

OnSuiteInstallBefore

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

OnSuiteShowUI イベントによって、OnSuiteInstallBefore イベントが呼び出されます。デフォルトで、OnSuiteInstallBefore は、ファイルの転送に必要な機能の選択など、情報の初期化のために呼び出されます。

***メモ・**program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*

OnSuiteMaintBefore

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

OnSuiteShowUI イベントによって、OnSuiteMaintBefore イベントが呼び出されます。デフォルトで、OnSuiteMaintBefore は、ファイルの転送に必要な機能の選択など、情報の初期化のために呼び出されます。

***メモ・**program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*

OnSuiteUpdateBefore

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

OnSuiteShowUI イベントによって、OnSuiteMaintBefore イベントが呼び出されます。デフォルトで、OnSuiteMaintBefore は、ファイルの転送に必要な機能の選択など、情報の初期化のために呼び出されます。

***メモ・**program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*

OnUpdateUIBefore

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

InstallScript インストールが、InstallScript パッケージとして、アドバンスト UI またはスイート／アドバンスト UI プロジェクトに含まれている場合、アドバンスト UI またはスイート／アドバンスト UI インストールで、このイベント ハンドラーは呼び出されません。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート／アドバンスト UI プロジェクトに追加する」をご覧ください。

OnUpdateUIBefore イベントハンドラー関数は、OnShowUI イベントハンドラーによって呼び出され、アップデートセットアップ用のファイル転送前のユーザーインターフェイスを表示します。

このイベント ハンドラーは手続き型スクリプト（program...endprogram ブロックを含むスクリプト）を利用するセットアップでは呼び出されません。

構文

OnUpdateUIBefore ( );

パラメーター

なし。

戻り値

なし。

OnXMLInitialize

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

OnXMLInitialize イベントは、OnFirstUIBefore によって呼び出され、XML ランタイムを初期化します。

***メモ・**program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*

Move Data ハンドラー

以下のイベントハンドラーは、ターゲットコンピューターにすべての機能をインストールまたはアンインストールする直前、その最中、あるいはその直後にトリガーされます。

テーブル 5 · Move Data ハンドラー

イベント ハンドラー	プロジェクトの種類	説明
OnMoveData	InstallScript	OnShowUI によって呼び出され、ファイル転送を行います。イベントハンドラーのデフォルトコードは FeatureTransferData を呼び出してファイルを転送します。
OnCustomizeUninstInfo	InstallScript	OnMoveData によって呼び出され、MaintenanceStart の呼び出しの後、アンインストール情報をカスタマイズします。
OnMoving	InstallScript InstallScript MSI	InstallScript プロジェクトの場合 : FeatureTransferData または Feature MoveData を呼び出すインストールの結果として呼び出されます。このイベントは、ファイル転送操作が実行されるごとに呼び出されます。 InstallScript MSI プロジェクトの場合 : GenerateMSIScript アクションが実行される直前に呼び出されます。
OnMoved	InstallScript InstallScript MSI	InstallScript プロジェクトの場合 : スクリプト内で FeatureTransferData が呼び出され、データ転送が行われた後、フレームワークによって直接呼び出されます。 InstallScript MSI プロジェクトの場合 : ターゲットコンピューターにすべての機能がインストールまたはアンインストールされる直後に生成される Moved イベントに応答します。
OnInstallingFile	InstallScript	FeatureTransferData または FeatureMoveData の結果としてファイルがインストールされる直前に呼び出されます。
OnUninstallingFile	InstallScript	FeatureTransferData または FeatureMoveData の結果としてファイルがアンインストールされる直前に呼び出されます。
OnUninstallingFontFile	InstallScript	RegisterFontResource がログ記録したフォントファイルがアンインストールされる時に呼び出されます。
OnUninstallingDIFxDriverFile	InstallScript	DIFxDriverPackageInstall または DIFxDriverPackagePreinstall 関数によってインストールまたはプレインストールされているドライバーが、アンインストール ログが有効になってアンインストールされた場合に呼び出されます。
OnInstalledFile	InstallScript	FeatureTransferData または FeatureMoveData の結果としてファイルがインストールされた後に呼び出されます。

テーブル 5 ・ Move Data ハンドラー（続き）

イベント ハンドラー	プロジェクトの種類	説明
OnInstalledFontFile	InstallScript	フォントファイルとしてメディアにリストされているファイルがインストールされた後に呼び出されます。
OnUninstalledFile	InstallScript	FeatureTransferData または FeatureMoveData の結果としてファイルがアンインストールされた後に呼び出されます。
OnSQLComponentInstalled	InstallScript	各コンポーネントがインストールされるごとに呼び出され、その結果、コンポーネントに付加されているすべての SQL スクリプトが実行可能になります。SQLComponentInstalled は、各コンポーネントがインストールされるごとに呼び出され、その結果、コンポーネントに付加されているすべての SQL スクリプトが実行可能になります。
OnSQLComponentUninstalled	InstallScript	各コンポーネントがアンインストールされるごとに呼び出され、その結果、コンポーネントに付加されているすべての SQL スクリプトが実行可能になります。SQLComponentUninstalled イベントは、各コンポーネントがアンインストールされるごとに呼び出され、その結果、そのコンポーネントに付加されているすべての SQL スクリプトが実行可能になります。
OnSQLBatchScripts	InstallScript	ファイル転送の後にフレームワークによって自動的に呼び出されます。
OnIISComponentInstalled	InstallScript	IISComponentInstalled は、各コンポーネントがインストールされるごとに呼び出され、その結果、コンポーネントに付加されているすべての IIS インフォメーションがインストール可能になります。
OnIISVRootUninstalling	InstallScript	各 IISVRoot が削除される前に呼び出されます。
OnXMLComponentInstalled	InstallScript	ISXMLComponentInstall は、各コンポーネントがインストールされるごとに呼び出され、その結果、コンポーネントに付加されているすべての XML 情報がインストール可能になります。
OnXMLComponentUninstalling	InstallScript	XMLComponentUninstalling は、各 .xml ファイルが削除されるごとに呼び出されます。
OnNetApiCreateUserAccount	InstallScript	OnMoving イベントの前に呼び出されます。このイベントハンドラー関数は、リンクされたライブラリのリストで **NetApiRT.obl** が **Isrt.obl** の前に追加されない限り、何も効果を持ちません。
OnGeneratedMSIScript	InstallScript MSI	実行されたあと、MSI 標準アクション LaunchConditions に応答します。

テーブル 5 ・ Move Data ハンドラー (続き)

イベント ハンドラー	プロジェクトの種類	説明
OnGeneratingMSIScript	InstallScript MSI	実行される前、MSI 標準アクション LaunchConditions に応答します。
OnInstallFilesActionBefore	InstallScript MSI	呼び出される前は、InstallFiles イベント ハンドラー関数に応答します。
OnInstallFilesActionAfter	InstallScript MSI	呼び出された後は、InstallFiles イベント ハンドラー関数に応答します。

OnCustomizeUninstInfo

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

OnCustomizeUninstInfo イベントハンドラー関数は、MaintenanceStart を呼び出した後、OnMoveData イベントハンドラーのデフォルトコードによって呼び出され、アンインストール情報をカスタマイズします。

構文

OnCustomizeUninstInfo ( );

パラメーター

なし。

戻り値

なし。

OnGeneratedMSIScript

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

OnGeneratingMSIScript イベント ハンドラーは、MSI 標準アクション LaunchConditions が実行された後に呼び出されます。このイベントハンドラーは、デフォルトでコードを含みません。これはレガシー インストールに含まれます。

OnGeneratingMSIScript

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

OnGeneratingMSIScript イベント ハンドラーは、実行される前に MSI の標準のアクション、LaunchConditions に応答します。このイベントハンドラーは、デフォルトでコードを含みません。これはレガシー インストールに含まれます。

OnIISComponentInstalled

OnIISComponentInstalled イベントハンドラー関数は、各コンポーネントがインストールされるごとに呼び出され、その結果、コンポーネントに付加されているすべての IIS 情報がインストール可能になります。

構文

OnIISComponentInstalled (szComponent);

パラメーター

テーブル 6 ・ OnIISComponentInstalled のパラメーター

パラメーター	説明
szComponent	このパラメーターには、インストールされたコンポーネントの名前が付きます。

戻り値

このイベントハンドラー関数は、現時点では常に ISERR_SUCCESS を戻します。

OnIISVRootUninstalling

OnIISVRootUninstalling イベントは、仮想ディレクトリが削除される前にディレクトリごとに呼び出されます。スクリプトで IIS に関連する条件をチェックし、abort() を呼び出してセットアプを停止することができます。

OnInstalledFile

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

OnInstalledFile イベントハンドラー関数は、FeatureTransferData または FeatureMoveData の結果としてファイルがインストールされた後に呼び出されます。

構文

OnInstalledFile (szFilename);

パラメーター

テーブル 7 - OnInstalledFile のパラメーター

パラメーター	説明
szFilename	転送されたファイルの完全修飾ファイル名を指定します。

戻り値

なし。

OnInstalledFontFile

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

OnInstalledFontFile イベントハンドラー関数は、フォントファイルとしてメディアにリストされているファイルがインストールされた後に呼び出されます。

構文

```
OnInstalledFontFile ( pFontFileInfo );
```

パラメーター

テーブル 8 - OnInstalledFontFile のパラメーター

パラメーター	説明
pFontFileInfo	インストール中のフォント ファイルについての情報を提供する _FONTFILEINFO 構造 へのポインター。

戻り値

なし。

OnInstallFilesActionAfter

OnInstallFilesActionAfter イベントハンドラーは、標準 Windows Installer InstallFiles アクションが実行される直後に呼び出されます。

OnInstallFilesActionBefore

OnInstallFilesActionBefore イベントハンドラーは、標準 Windows Installer InstallFiles アクションが実行される直前に呼び出されます。

OnInstallingFile

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

OnInstallingFile イベントハンドラー関数は、FeatureTransferData または FeatureMoveData の結果としてファイルがインストールまたはアンインストールされる直前に呼び出されます。

このイベントハンドラーのコードは、メンテナンスセットアップ中でも、次の if-then 構造に置かれていない場合には常に実行されます。

```
if !MAINTENANCE then
    ¥¥ メンテナンス以外のコード
endif;
```

構文

```
OnInstallingFile ( szFilename );
```

パラメーター

テーブル 9・OnInstallingFile のパラメーター

パラメーター	説明
szFilename	まもなく転送されるファイルの完全修飾ファイル名を指定します。

戻り値

なし。

OnMoved

InstallScript プロジェクトでは、OnMoved は、FeatureTransferData および Feature MoveData が呼び出すインストールの結果として呼び出されます。このイベントは、バッチ自己登録を除く、すべてのファイル転送操作が完了した際、呼び出されます。InstallScript MSI プロジェクトでは、OnMoved は、アクション GenerateMSIScript が実行される直前に、呼び出されます。

このイベントハンドラーのコードは、メンテナンスセットアップやアンインストール中でも、次の if-then 構造に置かれていない場合には常に実行されます。

```
if (!MAINTENANCE) then
  // メンテナンス以外のコード
endif;
```

OnMoveData

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

OnMoveData イベントハンドラーは、ファイル転送を処理する OnShowUI イベントハンドラーが呼び出します。デフォルトでは、ファイルを転送するのに OnMoveData は FeatureTransferData を呼び出します。

このイベント ハンドラーは手続き型スクリプト（program...endprogram ブロックを含むスクリプト）を利用するセットアップでは呼び出されません。

構文

OnMoveData ( );

パラメーター

なし。

戻り値

なし。

OnMoving

InstallScript プロジェクトでは、OnMoving は、FeatureTransferData および Feature MoveData が呼び出すインストールの結果として呼び出されます。このイベントは、ファイル転送操作が実行されるごとに呼び出されます。InstallScript MSI プロジェクトでは、OnMoving は、アクション GenerateMSIScript が実行される直前に、呼び出されます。

このイベントハンドラーのコードは、メンテナンスセットアップ中でも、次の if-then 構造に置かれていない場合には常に実行されます。

```
if !MAINTENANCE then
    ¥¥ メンテナンス以外のコード
endif;
```

OnNetApiCreateUserAccount

OnNetApiCreateUserAccount イベントハンドラー関数は、OnMoving イベントの前に呼び出されます。このイベント ハンドラー関数は、リンクされたライブラリのリストで **NetApiRT.obl** が **Isrt.obl** の前に追加されない限り、何も効果を持ちません。

構文

OnNetApiCreateUserAccount()

パラメーター

なし

戻り値

このイベント ハンドラー関数は常に ISERR_SUCCESS を返します。

追加情報

タスク ***リンクされたライブラリのリストに NetApiRT.obl を追加するには、以下の手順に従います：***

1. **[ビルド]** メニューで **[設定]** をクリックします。**[設定]** ダイアログ ボックスが開きます。
2. **Libraries (.obl)** ボックスで、**NetApiRT.obl** を入力します。これは、必ず **Isrt.obl** の前にリストします。

NetApiRT.obl へのパスは以下の通りです：

<ISProductFolder>¥Script¥ISRT¥Lib¥NetApiRT.obl

OnSQLBatchScripts

OnSQLBatchScripts イベントは、ファイル転送後フレームワークによって自動的に呼び出されます。

OnSQLComponentInstalled

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

OnSQLComponentInstalled イベントは、各コンポーネントがインストールされるごとに呼び出され、その結果、そのコンポーネントに付加されているすべての SQL スクリプトが実行可能になります。SQLComponentInstalled は、各コンポーネントがインストールされるごとに呼び出され、その結果、コンポーネントに付加されているすべての SQL スクリプトが実行可能になります。

OnSQLComponentUninstalled

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SQLComponentUninstalled イベントは、各コンポーネントがアンインストールされるごとに呼び出され、その結果、そのコンポーネントに付加されているすべての SQL スクリプトが実行可能になります。
SQLComponentUninstalled イベントは、各コンポーネントがアンインストールされるごとに呼び出され、その結果、そのコンポーネントに付加されているすべての SQL スクリプトが実行可能になります。

OnUninstalledFile

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

OnUninstalledFile イベントハンドラー関数は、FeatureTransferData または FeatureMoveData の結果としてファイルがアンインストールされた後に呼び出されます。

構文

```
OnUninstalledFile ( szFilename );
```

パラメーター

テーブル 10 - OnUninstalledFile のパラメーター

パラメーター	説明
szFilename	アンインストールされたファイルの完全修飾ファイル名を指定します。

戻り値

なし。

OnUninstallingFile

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

OnUninstallingFile イベントハンドラー関数は、FeatureTransferData または FeatureMoveData の結果としてファイルがアンインストールされる直前に呼び出されます。

このイベントハンドラーのコードは、メンテナンスセットアップ中でも、次の if-then 構造に置かれていない場合には常に実行されます。

```
if !MAINTENANCE then
    ¥¥ メンテナンス以外のコード
endif;
```

構文

```
OnUninstallingFile ( szFilename );
```

パラメーター

テーブル 11 - OnUninstallingFile

パラメーター	説明
szFilename	まもなくアンインストールされるファイルの完全修飾ファイル名を指定します。

戻り値

なし。

OnUninstallingDIFxDriverFile

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

OnUninstallingDIFxDriverFile イベント ハンドラー関数は、DIFxDriverPackageInstall または DIFxDriverPackagePreinstall 関数によってインストールまたはプレインストールされているドライバーが、アンインストール ログが有効になってアンインストールされた場合に呼び出されます。

デフォルトでは、イベントは DIFxDriverPackageUninstall 関数を使用してドライバーをアンインストールします。

構文

OnUninstallingDIFxDriverFile (byval string szDriver);

パラメーター

テーブル 12 ・ OnUninstallingDIFxDriverFile

パラメーター	説明
szDriver	インストールされたドライバー ファイルの完全パスおよびファイル。

戻り値

なし。

OnUninstallingFontFile

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

OnUninstallingFontFile イベントハンドラー関数は、RegisterFontResource によってログ記録されたフォントファイルがアンインストールされた時に呼び出されます。

構文

OnUninstallingFontFile (pFontFileInfo);

パラメーター

テーブル 13 ・ OnUninstallingFontFile

パラメーター	説明
pFontFileInfo	アンインストール中のフォント ファイルについての情報を提供する _FONTFILEINFO 構造 へのポインター。

戻り値

なし。

OnXMLComponentInstalled

OnXMLComponentInstalled は、ISXMLComponentInstall イベントに関連付けられているイベントハンドラー関数です。ISXMLComponentInstall イベントは、各コンポーネントがインストールされるごとに呼び出され、その結果、コンポーネントに付加されているすべての XML 情報がインストール可能になります。

構文

OnXMLComponentInstalled (szComponent);

パラメーター

テーブル 14 - OnXMLComponentInstalled

パラメーター	説明
szComponent	このパラメーターには、インストールされたコンポーネントの名前が付きます。

戻り値

このイベントハンドラー関数は、現時点では常に ISERR_SUCCESS を戻します。

OnXMLComponentUninstalling

OnXMLComponentUninstalling は、XMLRTComponentUninstall イベントに関連付けられているイベントハンドラー関数です。XMLRTComponentUninstall イベントは、各 .xml ファイルが削除されるごとに呼び出されます。

構文

OnXMLComponentUninstalling (szXmlComponent)

パラメーター

テーブル 15 - OnXMLComponentUninstalling

パラメーター	説明
szXmlComponent	このパラメーターには、インストールされたコンポーネントの名前が付きます。

戻り値

このイベントハンドラー関数は、現時点では常に ISERR_SUCCESS を戻します。

After Data Move ハンドラー

次のイベントハンドラーは、ファイルや他のデータがターゲットコンピューターに転送された後にトリガーされます。

テーブル 16 ・ After Data Move ハンドラー

イベント ハンドラー	プロジェクトの種類	説明
OnFirstUIAfter	InstallScript InstallScript MSI	InstallScript プロジェクトの場合 : OnFirstUIAfter イベントは、インストールを初回インストールモードで実行中、インストールのファイル転送後に OnShowUI によって呼び出されます。デフォルトでこのイベントは、インストールが無事に完了したことをエンドユーザーに報告する UI を表示します。  ***メモ・****program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。* InstallScript MSI プロジェクトの場合 : First UI After イベントに応答します。
OnMaintUIAfter	InstallScript InstallScript MSI	InstallScript プロジェクトの場合 : OnMaintUIAfter イベントは、インストールをメンテナンス モードで実行中、インストールのファイル転送後に OnShowUI によって呼び出されます。デフォルトでこのイベントは、メンテナンスインストールが無事に完了したことをエンドユーザーに報告する UI を表示します。  ***メモ・****program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。* InstallScript MSI プロジェクトの場合 : Maintenance UI After イベントに応答します。
OnUpdateUIAfter	InstallScript	OnUpdateUIAfter イベントは、インストールをアップデートモードで実行中、インストールのファイル転送後に OnShowUI によって呼び出されます。デフォルトでこのイベントは、メンテナンスインストールが無事に完了したことをエンドユーザーに報告する UI を表示します。このイベントはプログラムで自動的には呼び出されません。

テーブル 16 • After Data Move ハンドラー (続き)

イベント ハンドラー	プロジェクトの種類	説明
OnEnd	InstallScript InstallScript MSI	InstallScript プロジェクトの場合 : OnEnd イベントは、セットアップの最後に呼び出されます。このイベントはインストールが中止 (abort) された場合には呼び出されません。 InstallScript MSI プロジェクトの場合 : End イベントに応答します。セットアップ内であらかじめ定義できる最後のイベントです。
OnSuiteInstallAfter	InstallScript	インストールでファイルの転送が完了した後、OnSuiteShowUI イベントによって、OnSuiteInstallAfter イベントが呼び出されます。 ***メモ •*** *program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*
OnSuiteMaintAfter	InstallScript	メンテナンスでファイルの転送が完了した後、OnSuiteShowUI イベントによって、OnSuiteMaintAfter イベントが呼び出されます。 ***メモ •*** *program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*
OnSuiteUpdateAfter	InstallScript	アップグレードでファイルの転送が完了した後、OnSuiteShowUI イベントによって、OnSuiteUpdateAfter イベントが呼び出されます。 ***メモ •*** *program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*
OnIISUninitialize	InstallScript	OnIISUninitialize イベントは、OnMoveDataAfter によって呼び出され、IIS ランタイムを初期化します。 ***メモ •*** *program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*

テーブル 16 ・ After Data Move ハンドラー (続き)

イベント ハンドラー	プロジェクトの種類	説明
OnXMLUninitialize	InstallScript	OnXMLUninitialize イベントは、OnMoveDataAfter によって呼び出され、XML ランタイムを初期化します。 ***メモ ・*** *program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*

OnEnd

OnEnd イベントハンドラーは、セットアップで最後に再定義できる End イベントに応答します。必要な任意のクリーンアップコードを OnEnd 関数に配置することができます。

スクリプトに ifx.h または iswi.h を 含むときに OnEnd はプロトタイプ化されます。次の例に従って、スクリプトで OnEnd を定義してください :

```
#include "iswi.h"

function OnEnd()
   // ローカル変数
begin
   // クリーンアップ コード
end;
```

このイベントハンドラーのコードは、メンテナンスセットアップ中でも、次の if-then 構造に置かれていない場合には常に実行されます。

```
 if !MAINTENANCE then
     ¥¥ メンテナンス以外のコード
  endif;
```

OnFirstUIAfter

プロジェクト ・ *InstallScript インストールが、InstallScript パッケージとして、アドバンスト UI またはスイート / アドバンスト UI プロジェクトに含まれている場合、アドバンスト UI またはスイート / アドバンスト UI インストールで、このイベント ハンドラーは呼び出されません。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート / アドバンスト UI プロジェクトに追加する」をご覧ください。*

OnFirstUIAfter イベントハンドラーは、First UI After イベントに応答します。これは、アプリケーションの初回インストール用の機能をインストールした後に処理しなくてはならないタスクを実行します。

OnIISUninitialize

OnIISUninitialize イベントは、OnMoveDataAfter によって呼び出され、IIS ランタイムを初期化します。endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。

OnMaintUIAfter

***プロジェクト・**InstallScript インストールが、InstallScript パッケージとして、アドバンスト UI またはスイート／アドバンスト UI プロジェクトに含まれている場合、アドバンスト UI またはスイート／アドバンスト UI インストールで、このイベント ハンドラーは呼び出されません。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート／アドバンスト UI プロジェクトに追加する」をご覧ください。*

OnMaintUIAfter イベントハンドラーは、Maintenance UI After イベントに応答します。このイベント ハンドラーでは、アプリケーションのメンテナンス インストールによって機能が再インストールされた後に発生するタスクが実行されます。

OnSuiteInstallAfter

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

インストールでファイルの転送が完了した後、OnSuiteShowUI イベントによって、OnSuiteInstallAfter イベントが呼び出されます。

***メモ・**program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*

OnSuiteMaintAfter

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

メンテナンスでファイルの転送が完了した後、OnSuiteShowUI イベントによって、OnSuiteMaintAfter イベントが呼び出されます。

***メモ・**program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*

OnSuiteUpdateAfter

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

アップグレードでファイルの転送が完了した後、OnSuiteShowUI イベントによって、OnSuiteUpdateAfter イベントが呼び出されます。

***メモ・**program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*

OnUpdateUIAfter

プロジェクト・*InstallScript インストールが、InstallScript パッケージとして、アドバンスト UI またはスイート／アドバンスト UI プロジェクトに含まれている場合、アドバンスト UI またはスイート／アドバンスト UI インストールで、このイベント ハンドラーは呼び出されません。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート／アドバンスト UI プロジェクトに追加する」をご覧ください。*

OnUpdateUIAfter イベントハンドラー関数は、OnShowUI イベントハンドラーによって呼び出され、アップデートセットアップ用のファイル転送後のユーザーインターフェイスを表示します。

このイベント ハンドラーは手続き型スクリプト（program...endprogram ブロックを含むスクリプト）を利用するセットアップでは呼び出されません。

構文

OnUpdateUIAfter ( );

パラメーター

なし。

戻り値

なし。

OnXMLUninitialize

OnXMLUninitialize イベントは、OnMoveDataAfter によって呼び出され、XML ランタイムを初期化します。endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。

機能イベント ハンドラー

機能イベントハンドラーは、単一機能のインストールまたはアンインストールの直前に必要なプロセスを実行します。機能イベントとハンドラーの数は、プロジェクトに含まれる機能の数によって異なります。

機能にイベント ハンドラー関数を作成するには、左側のイベント カテゴリ リストから機能名を選択して、右側のイベント リストから必要なイベントを選択します。InstallShield は 2 番目の InstallScript ファイルを作成し、InstallScript ビューで **FeatureEvents.rul** を呼び出します。

FeatureEvents.rul でデフォルトの機能のイベント ハンドラーコードを変更した場合、**Setup.rul** に次のステートメントを入れてインストールに変更を含める必要があります。

```
#include "FeatureEvents.rul"
```

以下は機能イベント ハンドラーのリストです。

テーブル 17・機能イベントハンドラー

イベント ハンドラー	プロジェクトの種類	説明
OnInstalled	InstallScript InstallScript MSI	InstallScript プロジェクトの場合：ターゲット システムで該当する機能がインストールされた直後に生成される Installed イベントに応答して実行します。 InstallScript MSI プロジェクトの場合、Windows Installer がファイルをターゲット システムに転送した直後に実行します。
OnInstalling	InstallScript InstallScript MSI	InstallScript プロジェクトの場合：ターゲット システムで機能がインストールされる直前に生成される Installing イベントに応答して実行します。 InstallScript MSI プロジェクトの場合、Windows Installer がファイルをターゲット システムに転送する直前に実行します。
OnUnInstalled	InstallScript InstallScript MSI	InstallScript プロジェクトの場合：ターゲット システムから機能が削除された後に生成される UnInstalled イベントに応答して実行します。 InstallScript MSI プロジェクトの場合、Windows Installer がファイルをターゲット システムから削除した後に実行します。
OnUnInstalling	InstallScript InstallScript MSI	InstallScript プロジェクトの場合：ターゲット システムから機能が削除される直前に生成される UnInstalling イベントに応答して実行します。 InstallScript MSI プロジェクトの場合、Windows Installer がファイルをターゲット システムから削除する前に実行します。

プロジェクト・*Windows Installer が InstallScript MSI インストールで機能のインストールを制御するため、機能イベントハンドラー関数が呼び出される順序を指定することはできません。さらに、機能イベントはすべての機能がターゲットシステムへコピーされるまで起動されません。*

また、機能アンインストール イベント ハンドラー (OnUnInstalling と OnUnInstalled) は、ロールバック中に呼び出されません。

OnInstalled

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

InstallScript プロジェクトの場合 : OnInstalled 機能イベント ハンドラーはターゲット システムで該当する機能がインストールされた直後に生成される Installed イベントに応答して実行します。

InstallScript MSI プロジェクトの場合、OnInstalled 機能イベント ハンドラーは Windows Installer がファイルをターゲット システムに転送した直後に実行します。

OnInstalled 機能イベント ハンドラーは、次の形式で InstallScript コードにリストされています :

```
MyFeatureName_Installed()
```

プロジェクト・Windows Installer が InstallScript MSI インストールで機能のインストールを制御するため、機能イベントハンドラー関数が呼び出される順序を指定することはできません。さらに、機能イベントはすべての機能がターゲットシステムへコピーされるまで起動されません。

また、機能アンインストール イベント ハンドラー (OnUnInstalling と OnUnInstalled) は、ロールバック中に呼び出されません。

OnInstalling

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

InstallScript プロジェクトの場合 : OnInstalling 機能イベント ハンドラーはターゲット システムで機能がインストールされる直前に生成される Installin イベントに応答して実行します。

InstallScript MSI プロジェクトの場合、OnInstalling 機能イベント ハンドラーは Windows Installer がファイルをターゲット システムに転送する直前に実行します。

OnInstalled 機能イベント ハンドラーは、次の形式で InstallScript コードにリストされています :

```
MyFeatureName_Installing()
```

プロジェクト・Windows Installer が InstallScript MSI インストールで機能のインストールを制御するため、機能イベントハンドラー関数が呼び出される順序を指定することはできません。さらに、機能イベントはすべての機能がターゲットシステムへコピーされるまで起動されません。

また、機能アンインストール イベント ハンドラー (OnUnInstalling と OnUnInstalled) は、ロールバック中に呼び出されません。

OnUnInstalled

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

InstallScript プロジェクトの場合 : OnUnInstalled 機能イベント ハンドラーはターゲット システムから機能が削除される直前に生成される UnInstalling イベントに応答して実行します。

InstallScript MSI プロジェクトの場合、OnUnInstalled 機能イベント ハンドラーは Windows Installer がファイルをターゲット システムから削除する直前に実行します。

OnUnInstalled 機能イベント ハンドラーは、次の形式で InstallScript コードにリストされています :

```
MyFeatureName_UnInstalled()
```

プロジェクト・Windows Installer が InstallScript MSI インストールで機能のインストールを制御するため、機能イベントハンドラー関数が呼び出される順序を指定することはできません。さらに、機能イベントはすべての機能がターゲットシステムへコピーされるまで起動されません。

また、機能アンインストール イベント ハンドラー (OnUnInstalling と OnUnInstalled) は、ロールバック中に呼び出されません。

OnUnInstalling

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

InstallScript プロジェクトの場合 : OnUnInstalling 機能イベント ハンドラーはターゲット システムから機能が削除される直前に生成される UnInstalling イベントに応答して実行します。

InstallScript MSI プロジェクトの場合、OnUnInstalling 機能イベント ハンドラーは Windows Installer がファイルをターゲット システムから削除する直前に実行します。

OnUnInstalling 機能イベント ハンドラーは、次の形式で InstallScript コードにリストされています :

```
MyFeatureName_UnInstalling()
```

プロジェクト・Windows Installer が InstallScript MSI インストールで機能のインストールを制御するため、機能イベントハンドラー関数が呼び出される順序を指定することはできません。さらに、機能イベントはすべての機能がターゲットシステムへコピーされるまで起動されません。

また、機能アンインストール イベント ハンドラー (OnUnInstalling と OnUnInstalled) は、ロールバック中に呼び出されません。

その他のイベントハンドラー

Miscellaneous イベントハンドラーはセットアップ中にユーザーがインストールを終了するといった、予定されていないイベントによってトリガーされます。

テーブル 18 · その他のイベントハンドラー

イベント ハンドラー	プロジェクトの種類	説明
OnAbort	InstallScript、InstallScript MSI	InstallScript プロジェクトの場合 : abort キーワードによってセットアップが中止 (abort) されるとき、OnAbort イベントハンドラーが呼び出されます。 InstallScript MSI プロジェクトの場合 : InstallScript の abort コマンドによって生成される Abort イベントに応答します。
OnAdminInstallUIAfter	InstallScript MSI	Admin Install UI After イベントに応答します。
OnAdminInstallUIBefore	InstallScript MSI	アプリケーションの管理インストレーションの際にエンドユーザーから情報を収集するダイアログを表示して、[Admin Install UI Before] イベントに応答します。
OnAdminPatchUIAfter	InstallScript MSI	管理パッチのファイル転送後に呼び出されます。
OnAdminPatchUIBefore	InstallScript MSI	管理パッチのファイル転送前に呼び出されます。
OnAdvertisementAfter	InstallScript MSI	Advertisement After イベントに応答します。
OnAdvertisementBefore	InstallScript MSI	Advertisement After Before イベントに応答します。
OnCanceling	InstallScript、InstallScript MSI	InstallScript プロジェクトの場合 : OnCanceling イベントは、インストールがキャンセルされたときに送られます。キャンセルは通常、エンドユーザーがダイアログの キャンセル ボタンをクリック、または Esc キーを押した場合に起こります。Do(EXIT) の呼び出しもこのイベントをトリガーします。 InstallScript MSI プロジェクトの場合 : エンドユーザーがビルトイン ダイアログで [キャンセル] ボタンをクリックしたときに生成される Cancel イベントに応答します。
OnComponentError	InstallScript、InstallScript MSI	InstallScript プロジェクトの場合 : OnComponentError イベントは、FeatureTransferData または FeatureMoveData への呼び出しがエラーを戻した場合に、フレームワークによって呼び出されます。 InstallScript MSI プロジェクトの場合 : 一般ファイル転送エラーに応答します。
OnDIFxLogCallback	InstallScript	ビルトインの DIFx コールバック機能によってログ記録された DIFx 関連のイベントが発生したときに呼び出されます。

テーブル 18 - その他のイベントハンドラー (続き)

イベント ハンドラー	プロジェクトの種類	説明
OnError	InstallScript MSI	このイベントは MSI が INSTALLMESSAGE_ERROR メッセージを送ったときに呼び出されます。
OnException	InstallScript MSI	手続き型スクリプトによって生成された例外に応答します。
OnFileError	InstallScript	OnFileError イベントは、ファイルのインストールまたはアンインストール中に未知のエラーが発生したときに呼び出されます。
OnFileLocked	InstallScript	OnFileLocked イベントは、他のアプリケーションによって使用中のファイルが、インストールまたはアンインストールの必要があるとき呼び出されます。但し、ロックまたは共有とマークされている可能性があるファイルグループの中にファイルが存在する場合は除きます。この場合、ファイルは再起動の後にインストールまたはアンインストールされます。
OnFileReadOnly	InstallScript	OnFileReadOnly イベントは、読み取り専用ファイルがインストールまたはアンインストールされる必要があるとき呼び出されます。
OnFilesInUse	InstallScript MSI	OnFilesInUse イベント ハンドラーは、Windows Installer が INSTALLMESSAGE_FILESINUSE メッセージをインストールに送ったときに呼び出されます。 デフォルトで、OnFilesInUse イベント ハンドラーは **SdFilesInUse** ダイアログを表示します。
OnHelp	InstallScript、InstallScript MSI	InstallScript プロジェクトの場合 : OnHelp イベントは、エンドユーザーが F1 キーを押すか、Do(HELP) を呼び出した場合に呼び出されます。 InstallScript MSI プロジェクトの場合 : エンドユーザーが F1 キーを押した場合に発生する Help イベントに応答します。
OnInternetError	InstallScript	OnInternetError は、インターネットからファイルをインストール中にエラーが発生した際、呼び出されます。
OnLaunchAppAndWaitCallback	InstallScript、InstallScript MSI	OnLaunchAppAndWaitCallback は、LAAW_OPTION_USE_CALLBACK が LaunchAppAndWait 関数を呼び出したときに指定されると、インストールが起動するアプリケーションを待機中に呼び出されます。

テーブル 18・その他のイベントハンドラー（続き）

イベント ハンドラー	プロジェクトの種類	説明
OnLogonUserSetMsiProperties	InstallScript MSI	OnLogonUserSetMsiProperties イベント ハンドラーは、「既存のユーザー アカウントを作成または設定する機能を追加する」の InstallScript MSI プロジェクトで説明されているログオン ユーザー サポート用の Windows Installer プロパティを設定します。
OnMD5Error	InstallScript	OnMD5Error イベントは、インストール中のファイルの MD5 シグネチャが InstallShield CAB ファイルにある MD5 値と一致しない場合に呼び出されます。(MD5 はメディアがビルドされる時に計算されます。)
OnMsiSilentInstall	InstallScript MSI	ユーザーが InstallScript MSI プロジェクトの パッケージをサイレントモードで実行したときに発生する MSI Silent Install イベントに応答します。
OnNextDisk	InstallScript	OnNextDisk イベントは、ファイル転送中に、セットアップが必要なデータファイルを検出できなかったときに呼び出されます。例えば、複数フロッピーまたは CD のインストール中に次のディスクが必要な場合、このイベントが発生します。
OnOutOfDiskSpace	InstallScript MSI	空きディスク容量が不足しているターゲットシステムへ応答します。
OnPatchUIAfter	InstallScript MSI	Patch UI After イベントに応答します。
OnPatchUIBefore	InstallScript MSI	パッチインストールの際にダイアログを表示して [Patch UI Before] イベントに応答します。
OnRebooted	InstallScript MSI (InstallScript ユーザー インターフェイス (UI) のスタイルが、外部 UI ハンドラーとして InstallScript エンジンを使用する従来型のスタイルの場合)	InstallScript プロジェクトの場合 : OnRebooted イベントは、システム再起動の後でセットアップが自動的に実行する時に呼び出されます。これは、この場合に呼び出される唯一のイベントです。 InstallScript MSI プロジェクトの場合 : ターゲット システムが再起動された後で、インストーラーが再開する場合に発生する Rebooted イベントに応答します。  ***重要**・このイベントは、InstallScript UI スタイルが (埋め込み UI ハンドラーとして InstallScript エンジンを使用する) 新しいスタイルである InstallScript MSI インストールでは呼び出されません。詳しくは、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラとして使用する方法の違い」を参照してください。*

テーブル 18 • その他のイベントハンドラー (続き)

イベント ハンドラー	プロジェクトの種類	説明
OnRemovingSharedFile	InstallScript	OnRemovingSharedFile イベントは、共有ファイルがアンインストール中で、かつファイルの参照カウントがゼロに達したときファイルの転送中に呼び出されます。
OnResumeUIAfter	InstallScript MSI	Resume UI After イベントに応答します。
OnResumeUIBefore	InstallScript MSI	Resume UI Before イベントに応答します。
OnRMFilesInUse	InstallScript MSI	OnRMFilesInUse イベント ハンドラーは、再起動マネージャーが有効になっていて、Windows Installer 4.0 が INSTALLMESSAGE_RMFILESINUSE メッセージをインストールに送ったときに呼び出されます。 デフォルトで、OnRMFilesInUse イベント ハンドラーは SdRMFilesInUse ダイアログを表示します。
OnSelfRegistrationError	InstallScript	OnSelfRegistrationError イベント ハンドラーは、Do(SELFREGISTRATIONPROCESS) の呼び出しがファイルを正常に登録できなかったときに、フレームワークによって直接呼び出されます。
OnWarning	InstallScript MSI	Windows Installer サービスがインストールのユーザーインターフェイスに INSTALLMESSAGE_WARNING メッセージを送る際に発生する Warning イベントに応答します。

OnAbort

プロジェクト • この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

OnAbort イベントハンドラーは、InstallScript abort ステートメントが生成した Abort イベントに応答します。これはターゲットしすてむに加えられた変更をアンインストールして終了します。イベントハンドラーでは、例えばインストレーションが作成した一時ファイルの削除など、必要に応じた任意の追加クリーンアップコードを配置することができます。

OnAdminInstallUIAfter

プロジェクト • この情報は、InstallScript MSI プロジェクトに適用します。

OnAdminInstallUIAfter イベントハンドラーは Admin Install UI After イベントに応答します。アプリケーションの管理インストール用にデータを転送する後に処理しなくてはならないにタスクを実行します。管理インストールを実行するには、ユーザーは /a スイッチを使った **Setup.exe** を起動します。

一般的に、このハンドラーは SdFinishEx を呼び出してユーザーに対して管理インストールが完了した旨を報告します。

OnAdminInstallUIBefore

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

OnAdminInstallUIBefore イベントハンドラーは Admin Install UI Before イベントに応答します。アプリケーションの管理インストール用にデータを転送する前に処理しなくてはならないにタスクを実行します。管理インストールを実行するには、ユーザーは /a スイッチを使った **Setup.exe** を起動します。

一般的に、このハンドラーは SdWelcome または AdminAskPath 関数を呼び出し、管理ユーザーに対してインストール先ディレクトリの入力を要求します。

OnAdminPatchUIAfter

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

AdminPatchUIAfter イベントは、管理パッチセットアップ中、ファイル転送の後に呼び出されます。デフォルトでこのイベントは、インストールが無事に完了したことをエンドユーザーに報告する UI を表示します。

OnAdminPatchUIBefore

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

OnAdminInstallUIBefore イベントは、管理パッチセットアップ中、ファイルの転送前に呼び出されます。デフォルトでこのイベントは、エンドユーザーがインストールパラメーターを指定できる UI を表示します。

OnAdvertisementAfter

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

OnAdvertisementAfter イベントハンドラーは Advertisement After イベントに応答します。**ユーザーが /j 引数を使った Setup.exe** を実行したときに起こる、アドバタイズされたインストールを行った後にタスクを実行します。

OnAdvertisementBefore

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

OnAdvertisementBefore イベントハンドラーは Advertisement Before イベントに応答します。ユーザーが /j 引数を使った **Setup.exe** を実行したときに起こる、アドバタイズされたインストールを行う前にタスクを実行します。

OnCanceling

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

OnCanceling イベント ハンドラーは、エンド ユーザーがビルトイン InstallScript ダイアログで [キャンセル] ボタンをクリックしたときに発生する Cancel イベントに応答します。

```
function OnCanceling( )
begin

if (YES = AskYesNo(
            " セットアップをキャンセルしてもよろしいですか ?",
            YES))
        then abort;
endif;

end;
```

OnComponentError

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

OnComponentError イベントハンドラーは、インストールで一般的なファイル転送エラーがあった場合に生成される ComponentError イベントに応答します。

デフォルトの OnComponentError 実装は、OnComponentError で宣言され、次のステートメントによって値が割り当てられた ErrorInfo オブジェクトのプロパティを使用します :

```
set ErrorInfo = ComponentErrorInfo( );
```

テーブル 19 • OnComponentError のパラメーター

プロパティ	説明
ErrorInfo.Feature	エラーが発生したときに転送されていた機能に関する情報を提供するプロパティのオブジェクト。特定の機能にエラーが関連づけられていない場合、または機能が識別されない場合、このプロパティは設定されません。IsObject(ErrorInfo.Feature) 確認してテストを行ってください。
ErrorInfo.Feature.Description	ファイル転送エラーを説明する文字列。この文字列は、ヌル ("") である可能性があります。
ErrorInfo.Feature.DisplayName	エラーが発生したときに転送されていた機能の表示名。この機能に対して表示名を指定しなかった場合、この文字列はヌル ("") になります。
ErrorInfo.Feature.Name	エラーが発生したときに転送されていた機能の名前。
ErrorInfo.FileError.Description	ErrorInfo.LastError の文字列形式の説明。
ErrorInfo.FileError.File	ファイル関連のエラー : インストールが転送中にエラーが発生したファイルのパスと名前。
ErrorInfo.FileGroup	エラーが発生したときに転送されていたコンポーネントの名前。
ErrorInfo.LastError	ファイル転送エラーの数値コード。

OnDIFxLogCallback

プロジェクト • この情報は、InstallScript プロジェクトに適用します。

OnDIFxLogCallback イベント ハンドラーは、ビルトインの DIFx コールバック機能によってログ記録された DIFx 関連のイベントが発生したときに呼び出されます。詳細は、Windows マニュアルの DIFXAPISetLogCallback を参照してください。

メモ • このイベントは、64 ビットのドライバーではサポートされていないため、64 ビットのドライバーをインストールしても呼び出されません。

構文

```
OnDIFxLogCallback ( byval number nEventType, byval number nError, byval string szDescription );
```

パラメーター

テーブル 20 ・ OnDIFxLogCallback のパラメーター

パラメーター	説明
nEventType	DIFxAPI に記載されたイベント タイプ。次の値が使用できます。 ・ DIFXAPI_SUCCESS Ñ 操作が成功したことを示すメッセージをログ記録する成功イベント。 ・ DIFXAPI_INFO– 操作のコンテキストまたは進行状況について説明したメッセージをログ記録する情報イベント。 ・ DIFXAPI_WARNING– 致命的エラーではない問題の可能性についてメッセージをログ記録する警告イベント。 ・ DIFXAPI_ERROR– 致命的エラーについてのメッセージをログ記録するエラー イベント。
nError	エラーが存在すれば、イベントに関連付けられた Win32 エラー コードを指定します。存在しない場合はゼロを指定します。
szDescription	エラーについて説明する文字列。

戻り値

なし。

OnError

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

このイベントは Windows Installer が INSTALLMESSAGE_ERROR メッセージを送ったときに呼び出されます。

OnException

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

OnException イベント ハンドラーは、手続き型スクリプト (明示的な program...endprogram ブロックを使用するスクリプト) によって生成された例外に応答します。デフォルトの実装はエラー番号、ソース、そして Err オブジェクト に格納された説明を表示します。

OnException はイベントベースのスクリプトでは呼び出されません。イベントベースのスクリプトの場合、catch 例外に try...catch...endcatch ブロックを実装しなくてはなりません。

OnFileError

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

OnFileError イベントハンドラーは、セットアップに他のファイルエラーイベントを発生しないファイルエラーがあった場合に発生する ComponentError イベントに応答します (例 FileLocked または SelfRegistrationError)。このイベントはオブジェクトでトリガーされることはないので、InstallShield オブジェクトを作成する際は注意してください。

このイベント ハンドラーは手続き型スクリプト (program...endprogram ブロックを含むスクリプト) を利用するセットアップの (適切な場所で) 呼び出されます。

構文

```
OnFileError ( szFilename, nError );
```

パラメーター

テーブル 21 - OnFileError のパラメーター

パラメーター	説明
szFilename	エラーが発生したファイルの完全修飾ファイル名を指定します。
nError	エラーが発生したときに Windows API 関数 GetLastError が戻す値を指定します。

戻り値

テーブル 22 - OnFileError の戻り値

戻り値	説明
ERR_IGNORE	OnFileError イベント ハンドラーによってセットアップに戻されます。OnFileError の最初の引数で指定されたファイルのインストールやアンインストールのエラーを無視し、該当するファイルの操作を行わずに続行するようセットアップに要求します。
ERR_RETRY	OnFileError イベント ハンドラーによってセットアップに戻されます。OnFileError の最初の引数で指定されたファイルのインストールやアンインストールを再試行するようセットアップに要求します。
ERR_ABORT	OnFileError イベントハンドラーによってセットアップに戻され、セットアップを中止するよう要求します。

OnFileLocked

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

OnFileLocked イベントハンドラーは、FileLocked イベントに応答します。このイベントは、削除または上書きを必要とする、ロックされた (使用中の) ファイルが見つかった場合に生成されます。このイベントはオブジェクトでトリガーされることはないので、InstallShield オブジェクトを作成する際は注意してください。

このイベントハンドラーは、″ ファイルのロック ″ プロパティが [はい] に設定されているファイルについては呼び出されません。この場合、ファイル操作はシステムの再起動後に自動的に実行されます。

このイベント ハンドラーは手続き型スクリプト (program...endprogram ブロックを含むスクリプト) を利用するセットアップの (適切な場所で) 呼び出されます。

構文

```
OnFileLocked ( szFilename );
```

パラメーター

テーブル 23 - OnFileLocked のパラメーター

パラメーター	説明
szFilename	ロックされているファイルの完全修飾ファイル名を指定します。

戻り値

テーブル 24 - OnFileLocked の戻り値

戻り値	説明
ERR_IGNORE	OnFileLocked イベント ハンドラーによってセットアップに戻されます。OnFileLocked の最初の引数で指定されたファイルのインストールやアンインストールのエラーを無視し、該当するファイルの操作を行わずに続行するようセットアップに要求します。
ERR_RETRY	OnFileLocked イベント ハンドラーによってセットアップに戻されます。OnFileLocked の引数で指定されたファイルのインストールまたはアンインストールを再試行するようセットアップに要求します。
ERR_ABORT	OnFileLocked イベントハンドラーによってセットアップに戻され、セットアップを中止するよう要求します。
ERR_PERFORM_AFTER_REBOOT	OnFileLocked イベントハンドラーによってセットアップに戻され、セットアップでターゲット システムの再起動後にファイル操作を実行するよう要求します。

OnFileReadOnly

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

OnFileReadOnly は、削除または上書きするファイルが読み込み専用に設定されている場合に発生する ReadOnly イベントに応答します。このイベントはオブジェクトでトリガーされることはないので、InstallShield オブジェクトを作成する際は注意してください。

このイベント ハンドラーは手続き型スクリプト (program...endprogram ブロックを含むスクリプト) を利用するセットアップの (適切な場所で) 呼び出されます。

構文

```
OnFileReadOnly ( szFilename );
```

パラメーター

テーブル 25 - OnFileReadOnly のパラメーター

パラメーター	説明
szFilename	読み取り専用ファイルの完全修飾ファイル名を指定します。

戻り値

テーブル 26 - OnFileReadyOnly の戻り値

戻り値	説明
ERR_YES	OnRemovingSharedFile イベントハンドラーによってセットアップに戻され、セットアップでファイル操作を実行するよう要求します。
ERR_NO	OnRemovingSharedFile イベントハンドラーによってセットアップに戻され、セットアップでファイル操作を実行しないよう要求します。

OnFilesInUse

***プロジェクト -** この情報は、InstallScript MSI プロジェクトに適用します。*

InstallScript MSI インストールで OnFilesInUse イベント ハンドラーは、Windows Installer が INSTALLMESSAGE_FILESINUSE メッセージをインストールに送ったとき呼び出されます。

szMessage パラメーターには Windows Installer から提供された文字列が含まれています。このパラメーターは使用中のファイルを示します。**SdFilesInUse** 関数はこの文字列を適切に解析します。

デフォルトで、OnFilesInUse イベント ハンドラーは **SdFilesInUse** ダイアログを表示します。イベント ハンドラーがダイアログが返した値を返すと、その値は Windows Installer に再度渡され、メッセージがどのように処理されたか、および Windows Installer が実行するべき操作が示されます。

構文

OnFilesInUse (szMessage);

パラメーター

テーブル 27 - OnFilesInUse のパラメーター

パラメーター	説明
szMessage	使用中のファイルを示す Windows Installer から提供された文字列。**SdFilesInUse** 関数はこの文字列を解析します。

戻り値

テーブル 28 - OnFilesInUse の文字列

戻り値	説明
IDCANCEL	インストールをキャンセルする必要があることを示します。
IDRETRY	Windows Installer が使用中のファイルを再確認して、必要な場合、INSTALLMESSAGE_FILESINUSE メッセージを送ることを示します。
IDIGNORE	Windows Installer がファイルが使用中であるという事実を無視して、インストールを続行することを示します。

OnHelp

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

- *InstallScript*
- *InstallScript MSI*

OnHelp イベントハンドラーはエンドユーザーが F1 キーを押した場合に生成される Help イベントに応答します。

```
function OnHelp( )
begin
   /* MySetupHelp.chm が [ サポートファイル / ビルボード ] ビューにあると
     想定します。*/
   LaunchAppAndWait( WINDIR ^ "Hh.exe",
                     SUPPORTDIR ^ "MySetupHelp.chm",
                     NOWAIT );
end;
```

OnInternetError

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

OnInternetError イベント ハンドラーは、セットアップに他のファイル エラー イベントを発生しないファイル エラーがあった場合に発生する ComponentError イベントに応答します (例 FileLocked または SelfRegistrationError)。このイベントはオブジェクトでトリガーされることはないので、InstallShield オブジェクトを作成する際は注意してください。

このイベント ハンドラーは手続き型スクリプト (program...endprogram ブロックを含むスクリプト) を利用するセットアップの (適切な場所で) 呼び出されます。

構文

OnInternetError (hInternet, szFilename, nError);

パラメーター

テーブル 29 ・ OnInternetError のパラメーター

パラメーター	説明
hInternet	内部ハンドル - 無視する。
szFilename	エラーが発生したファイルの完全修飾ファイル名を指定します。
nError	エラーが発生したときに Windows API 関数 GetLastError が戻す値を指定します。

戻り値

テーブル 30 ・ OnInternetError の戻り値

戻り値	説明
ERR_IGNORE	OnInternetError イベント ハンドラーによってセットアップに戻されます。OnInternetError の 2 番目の引数で指定されたファイルのインストールやアンインストールのエラーを無視し、該当するファイルの操作を行わずに続行するようセットアップに要求します。
ERR_RETRY	OnInternetError イベント ハンドラーによってセットアップに戻されます。OnInternetError の 2 番目の引数で指定されたファイルのインストールまたはアンインストールを再試行するようセットアップに要求します。
ERR_ABORT	OnInternetError イベントハンドラーによってセットアップに戻され、セットアップを中止するよう要求します。

OnLaunchAppAndWaitCallback

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

OnLaunchAppAndWaitCallback イベント ハンドラーは、LAAW_OPTION_USE_CALLBACK が **LaunchApplication** 関数を呼び出したときに指定されると、インストールが起動するアプリケーションを待機中に呼び出されます。イベントは、LAAW_PARAMETERS.nCallbackInterval パラメーターで指定された時間の間隔で呼び出されます。

インストールが LAAW_OPTION_USE_CALLBACK が指定されている複数の **LaunchApplication** を含む場合、同じイベントが各待機中に呼び出されます。この場合、LAAW_PARAMETERS.szCommandLineResult パラメーターを使用して現在実行中の呼び出しを判断します。LAAW_CALLBACK_RETURN_CONTINUE_TO_WAIT を戻して待機を継続するか、または、LAAW_CALLBACK_RETURN_END_WAIT を戻して待機を即座に終了することができます。

```
function number OnLaunchAppAndWaitCallback( )

begin
   return LAAW_CALLBACK_RETURN_CONTINUE_TO_WAIT;
end;
```

メモ・このイベントハンドラーは、インストールおよびオブジェクトプロジェクトで発生します。このイベントに指定されたオーバーライドはすべて、上書きされたスクリプトに適用します。メインのインストールスクリプトのオーバーライドは、含まれているオブジェクトに影響しません。また、オブジェクトプロジェクトのオーバーライドも、メインのインストールスクリプトに影響しません。

OnLogonUserSetMsiProperties

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

OnLogonUserSetMsiProperties イベント ハンドラーは、「既存のユーザー アカウントを作成または設定する機能を追加する」の InstallScript MSI プロジェクトで説明されているログオン ユーザー サポート用の Windows Installer プロパティを設定します。具体的に、OnLogonUserSetMsiProperties は以下を設定します

- Windows Installer プロパティ **IS_NET_API_LOGON_USERNAME** は、InstallScript 変数 IFX_NETAPI_USER_ACCOUNT に設定されます。
- Windows Installer プロパティ **IS_NET_API_LOGON_PASSWORD** は、InstallScript 変数 IFX_NETAPI_PASSWORD に設定されます。
- Windows Installer プロパティ **IS_NET_API_LOGON_GROUP** は、InstallScript 変数 IFX_NETAPI_GROUP に設定されます。

IFX_NETAPI_* 変数は、**SdLogonUserInformation** および関連ダイアログの入力フィールドの値に設定されます。

OnMD5Error

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

OnMD5Error イベントハンドラーは、セットアップがセットアップヘッダーファイルに格納された値に対応しない MD5 ハッシュ値を持つファイルを抽出する時に MD5 の確認中に生成される MD5Error イベントに応答します。このイベントはオブジェクトでトリガーされることはないので、InstallShield オブジェクトを作成する際は注意してください。

このイベント ハンドラーは手続き型スクリプト (program...endprogram ブロックを含むスクリプト) を利用するセットアップの (適切な場所で) 呼び出されます。

ヒント・*MD5 確認は破損したファイルを検出します。これは インターネット でのインストール中に便利です。MD5 確認を行わなかった場合、ファイル転送処理速度は速くなります。[リリース ウィザード] の [一般オプション] パネルにある [詳細] ボタンを利用するか、Setup.ini ファイルの [スタートアップ] セクションの CheckMD5 キーを使って MD5 確認を有効または無効にすることができます。*

構文

OnMD5Error (szFilename);

パラメーター

テーブル 31 ・ OnMD5Error のパラメーター

パラメーター	説明
szFilename	MD5 エラーを生成したファイルの完全修飾ファイル名を指定します。

戻り値

テーブル 32 ・ OnMD5Error の戻り値

戻り値	説明
ERR_IGNORE	OnMD5Error イベント ハンドラーによってセットアップに戻されます。OnMD5Error の引数で指定されたファイルのインストールやアンインストールのエラーを無視し、該当するファイルの操作を行わずに続行するようセットアップに要求します。
ERR_RETRY	OnMD5Error イベント ハンドラーによってセットアップに戻されます。OnMD5Error の引数で指定されたファイルのインストールまたはアンインストールを再試行するようセットアップに要求します。
ERR_ABORT	OnMD5Error イベントハンドラーによってセットアップに戻され、セットアップを中止するよう要求します。

追加情報

メディアビルダーを実行後にディスクイメージフォルダーに非圧縮ファイルを格納する場合、MD5Error イベントはランタイム中に生成されます。デフォルトの OnMD5Error コードを変更して、ダイアログを表示するのではなく、この MD5Error イベントを自動処理することができます。たとえば、TARGETDIR にインストールされている ReplacedAfterBuild.txt に対して MD5Error イベントを自動的に無視するには、OnMD5Error コードの先頭に次の行を追加します。

```
if szFilename = TARGETDIR ^ "ReplacedAfterBuild.txt" then
    return ERR_IGNORE;
endif;
```

OnMsiSilentInstall

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

InstallScript プロジェクトで作成されたインストール プログラムは、エンド ユーザーが **Setup.exe** を実行する必要があります。OnMsiSilentInstall イベント ハンドラーは、エンド ユーザーが InstallScript MSI プロジェクトの .msi データベースを **MsiExec.exe** への /q オプションを使ってサイレント モードで実行しようとしたときに、これに応答します。デフォルトの実装では、エラー メッセージを表示してからインストールを中止します。

OnNextDisk

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

Windows Installer はディスク要求を処理するため、このイベントは InstallScript MSI プロジェクトではサポートされていません。

OnNextDisk イベントハンドラーは NextDisk イベントに応答します。NextDisk イベントは、複数のディスクを使用するセットアップで、セットアップを続行するため次のディスクが必要になったときに発生します。

戻り値

テーブル 33 ・ OnNextDisk の戻り値

戻り値	説明
ERR_RETRY	OnNextDisk イベントハンドラーによってセットアップに戻されます。OnNextDisk の最初の引数で指定されたファイルを再度検索するようセットアップに要求します。
ERR_ABORT	OnNextDisk イベントハンドラーによってセットアップに戻され、セットアップを中止するよう要求します。

OnOutOfDiskSpace

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

OnOutOfDiskSpace イベントハンドラーは Out Of Disk Space イベントに応答します。OnOutOfDiskSpace のデフォルト実装は、SdDiskSpace2 ダイアログを表示してからインストールを終了します。

OnPatchUIAfter

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

OnPatchUIAfter イベントハンドラーは、パッチインストール用のデータ転送後に呼び出されます。OnPatchUIAfter のデフォルト実装によって、SdFinishEx 関数が呼び出されます。

OnPatchUIBefore

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

OnPatchUIBefore イベントハンドラーは、ユーザーがパッチインストールを起動した時に呼び出されます。OnPatchUIBefore のデフォルト実装は、SdPatchWelcome ダイアログ関数を呼び出します。

OnRebooted

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI – InstallScript ユーザー インターフェイス (UI) のスタイルが、外部 UI ハンドラーとして InstallScript エンジンを使用する従来型のスタイルの場合*

この情報は、InstallScript UI に新しいスタイル (InstallScript エンジンを埋め込み UI ハンドラーとして使用するスタイル) が使用されている InstallScript MSI プロジェクトには適用しません。詳しくは、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラとして使用する方法の違い」を参照してください。

OnRebooted イベントハンドラーは、ターゲットシステムが再起動された後で、インストールを再開する時に生成される Rebooted イベントに応答します。セットアップが再起動した後に再開する際、インストールが呼び出すイベント ハンドラーは OnRebooted だけです。

OnRemovingSharedFile

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

OnRemovingSharedFile イベントは、RemovingSharedFile イベントに応答します。RemovingSharedFile イベントは、アンインストレーション中に、他のアプリケーションと共有されている可能性があるファイルを削除しようとすると発生します。このイベントはオブジェクトでトリガーされることはないので、InstallShield オブジェクトを作成する際は注意してください。

このイベント ハンドラーは手続き型スクリプト (program...endprogram ブロックを含むスクリプト) を利用するセットアップの (適切な場所で) 呼び出されます。

構文

```
OnRemovingSharedFile ( szFilename );
```

パラメーター

テーブル 34 - OnRemovingSharedFile のパラメーター

パラメーター	説明
szFilename	共有ファイルの完全修飾ファイル名を指定します。

戻り値

テーブル 35 - OnRemovingSharedFile の戻り値

戻り値	説明
ERR_YES	OnRemovingSharedFile イベントハンドラーによってセットアップに戻され、セットアップでファイルを削除するよう要求します。
ERR_NO	OnRemovingSharedFile イベントハンドラーによってセットアップに戻され、セットアップでファイルを削除しないよう要求します。

OnResumeUIAfter

***プロジェクト -** この情報は、InstallScript MSI プロジェクトに適用します。*

OnResumeUIAfter イベントハンドラーは、Resume UI After イベントに応答します。これはアプリケーションの再インストールまたはマイナーアップグレードの後に実行する必要のあるタスクを行います。

OnResumeUIAfter および OnResumeUIBefore イベント ハンドラーは次の状況すべてが発生した場合のみ呼び出されます。

- アプリケーションがターゲットマシンに既にインストールされている。
- パッチが *実行されていない*。
- 次のプロパティの一つがコマンドラインで、または Setup.ini の CmdLine プロパティで設定されている。
  - ADDDEFAULT
  - ADDLOCAL
  - ADDSOURCE
  - ADVERTISE
  - COMPADDLOCAL
  - COMPADDSOURCE
  - FILEADDDEFAULT
  - FILEADDLOCAL
  - FILEADDSOURCE

- REINSTALL
- REMOVE

OnResumeUIBefore

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

OnResumeUIBefore イベントハンドラーは、Resume UI Before イベントに応答します。これはアプリケーションの再インストールまたはマイナーアップグレードの前に実行する必要のあるタスクを行います。

OnResumeUIBefore および OnResumeUIAfter イベント ハンドラーは次の状況すべてが発生した場合のみ呼び出されます。

- アプリケーションがターゲットマシンに既にインストールされている。
- パッチが *実行されていない*。
- 次のプロパティの一つがコマンドラインで、または Setup.ini の CmdLine プロパティで設定されている。
  - ADDDEFAULT
  - ADDLOCAL
  - ADDSOURCE
  - ADVERTISE
  - COMPADDLOCAL
  - COMPADDSOURCE
  - FILEADDDEFAULT
  - FILEADDLOCAL
  - FILEADDSOURCE
  - REINSTALL
  - REMOVE

OnRMFilesInUse

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

OnRMFilesInUse イベント ハンドラーは、再起動マネージャーが有効になっていて、Windows Installer 4.0 が INSTALLMESSAGE_RMFILESINUSE メッセージをインストールに送ったときに InstallScript MSI インストールで呼び出されます。

再起動マネージャーが使用できない、または無効になっているとき、INSTALLMESSAGE_RMFILESINUSE メッセージは送られませんので注意してください。詳しい情報は、Windows Installe r ライブラリの「SecureCustomProperties Property」を参照してください。

再起動マネージャーが使用できない、または無効になっているとき、もしくはターゲット システムに Windows Installer 3.x 以前があるとき、Windows Installer は INSTALLMESSAGE_FILESINUSE メッセージをインストールに送ります。

デフォルトで、OnRMFilesInUse イベント ハンドラーは **SdRMFilesInUse** ダイアログを表示します。イベント ハンドラーがダイアログが返した値を返すと、その値は Windows Installer に再度渡され、メッセージがどのように処理されたか、および Windows Installer が実行するべき操作が示されます。

構文

OnRMFilesInUse (szMessage);

パラメーター

テーブル 36 - OnRMFilesInUse のパラメーター

パラメーター	説明
szMessage	使用中のファイルを示す Windows Installer から提供された文字列。**SdRMFilesInUse** 関数はこの文字列を解析します。

戻り値

テーブル 37 - OnRMFilesInUse の文字列

戻り値	説明
IDCANCEL	インストールをキャンセルする必要があることを示します。
IDRETRY	Windows Installer が使用中のファイルを再確認して、必要な場合、INSTALLMESSAGE_RMFILESINUSE メッセージを送ることを示します。 ***メモ -*** *SdFilesInUse とは違い、SdRMFilesInUse ダイアログはこの値を返しません。*
IDIGNORE	Windows Installer がファイルが使用中であるという事実を無視して、インストールを続行することを示します。
IDOK	Windows Installer が再起動マネージャーを使用して、ファイルをロックしている実行中のアプリケーションをシャットダウンする試みを行うことを示します。詳しい情報は、Windows Installe r ヘルプ ライブラリの「INSTALLMESSAGE_RMFILESINUSE」を参照してください。

OnSelfRegistrationError

プロジェクト - *この情報は、InstallScript プロジェクトに適用します。*

OnSelfRegistrationError イベント ハンドラーは、Do(SELFREGISTRATIONPROCESS) の呼び出しがファイルを正常に登録できなかったときに、フレームワークによって直接呼び出されます。

構文

OnSelfRegistrationError ( );

パラメーター

なし。

戻り値

なし。

追加情報

デフォルトの OnSelfRegistrationError コードは、グローバル FileRegistrar オブジェクトの次のプロパティを使用します。

テーブル 38 · Global FileRegistar オブジェクトのプロパティ

プロパティ	説明
FileRegistrar.Errors.Count	登録されなかった自己登録ファイルの数
FileRegistrar.Errors(*i*)	*i*-th の登録を解除されたファイルに関する情報を提供するプロパティのオブジェクト *i* は 1 から FileRegistrar.Errors Count の値までを示します。)
FileRegistrar.Errors(*i*).File	*i*-th の登録を解除されたファイルの名前
FileRegistrar.Errors(*i*).Description	セットアップが *i*-th の登録を解除されたファイルの登録を試みた際に発生するエラーを説明する文字列この文字列は、ヌル ("") である可能性があります。
FileRegistrar.Errors(*i*).LastError	セットアップが *i*-th の登録解除されたファイルの登録を試みた際に発生するエラーをの数値コード

OnWarning

***プロジェクト·** この情報は、InstallScript MSI プロジェクトに適用します。*

OnWarning イベント ハンドラーは、Windows Installer サービスが INSTALLMESSAGE_WARNING メッセージを送信したときに生成される Warning イベントに応答します。

拡張イベントハンドラー

拡張イベント ハンドラーは、特定の状況でトリガーされます。

テーブル 39 - 拡張イベントハンドラー

イベント ハンドラー	プロジェクトの種類	説明
OnShowUI	InstallScript	インストールエンジンによって直接呼び出され、ユーザーインターフェイスおよびファイル転送を開始します。
OnSuiteShowUI	InstallScript	アドバンスト UI またはスイート / アドバンスト UI インストールで、InstallScript パッケージが起動されたとき、OnShowUI イベントの代わりに、OnSuiteShowUI イベントが呼び出されます。デフォルトで、OnSuiteShowUI イベントは、アドバンスト UI またはスイート / アドバンスト UI インストールから InstallScript パッケージに渡される機能を初期化し、ファイルの転送を開始します。
OnUninstall	InstallScript、InstallScript MSI	OnUninstall イベント ハンドラーは、インストールが /uninst パラメーターを使って実行された場合に呼び出されます。これは、/uninst パラメーターが使われたときに呼び出される唯一のイベントです。OnUninstall イベントのデフォルト コードは、以前にインストールされた製品をアンインストールします。

OnShowUI

この関数は、セットアップの UI シーケンスおよびファイル転送を制御します。

OnShowUI イベントはフレームワークによって直接呼び出され、インストールの UI シーケンスおよびファイル転送を開始します。デフォルトでこのイベントは、メンテナンスが無事に完了したことをエンドユーザーに報告する UI を表示します。

メモ - *program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。*

構文

```
OnShowUI ( );
```

パラメーター

なし。

戻り値

なし。

追加情報

次の手順を実行することで、手続き型スクリプトをイベント指向スクリプトへ簡単に変換することができます。この場合、ユーザー インターフェイス イベント ハンドラー (OnFirstUIBefore、OnMaintUIBefore、OnUpdateUIBefore、OnFirstUIAfter、OnMaintUIAfter、OnUpdateUIAfter) も OnMoveData も呼び出されませんが、必要に応じて他のすべての機能のイベント ハンドラーが呼び出されます。

タスク ***手続き型スクリプトをイベントベースのスクリプトへ変換するには、次の手順に従います。***

1. InstallShield でプロジェクトを開き、変換についてプロンプトされた時に [はい] をクリックします。
2. スクリプトを開いて次の行を変更します。

   ```
   program
   ```

   変更後、

   ```
   function OnShowUI
   begin
   ```

3. また次の行も変更します。

   ```
   endprogram
   ```

   変更後、

   ```
   end
   ```

4. (InstallShield が提供するオブジェクトには含まれない) ユーザーインターフェイス を含む任意のカスタム オブジェクトを表示する場合は、オブジェクト UI を表示する場所にあるダイアログ シーケンスに ShowObjWizardPages への呼び出しを追加します :

   ```
   ShowObjWizardPages( nResult );
   ```

   この関数はファイル転送の前または後に呼び出すことができます。ファイル転送が行われたかどうか、そして MAINTENANCE システム変数の値に基づいて適切なオブジェクトイベントが呼び出されます。
5. InstallShield でスクリプトをコンパイルするのに必要な変更を行う。

希望であれば、OnBegin イベントや OnEnd イベントをカスタマイズすることもできます。

動作する理由

InstallScript では、殆どのセットアップイベントはメイン UI イベント、OnShowUI によって操作されます。このイベントのデフォルトのコードを program...endprogram ブロック内のコードと置換すると、基本的にカスタム UI シーケンスを提供していることになります。OnShowUI はインストールが実行されると常に呼び出されるため、インストールは program...endprogram スクリプトの場合と同様に動作します。

アドバンスト UI またはスイート / アドバンスト UI インストール内の InstallScript パッケージを起動する

アドバンスト UI またはスイート / アドバンスト UI インストールで、InstallScript パッケージが起動されたとき、OnShowUI イベントの代わりに、OnSuiteShowUI イベントが呼び出されますので注意してください。

OnSuiteShowUI

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

アドバンスト UI またはスイート / アドバンスト UI インストールで、InstallScript パッケージが起動されたとき、OnShowUI イベントの代わりに、OnSuiteShowUI イベントが呼び出されます。デフォルトで、OnSuiteShowUI イベントは、アドバンスト UI またはスイート / アドバンスト UI インストールから InstallScript パッケージに渡される機能を初期化し、ファイルの転送を開始します。アドバンスト UI またはスイート / アドバンスト UI インストールのセットアップ ランチャーは、通常、アドバンスト UI またはスイート / アドバンスト UI インストール全体で、ユーザー インターフェイスを操作するため、OnSuiteShowUI イベントによっては、通常、ユーザー インターフェイスは表示されません。

インストール状態 (初回インストール、メンテナンス、またはアップデート) に応じて、OnSuiteShowUI では、OnFirstUIBefore や OnFirstUIAfter などの UI イベントが無視され、次のイベントが呼び出されます :

- **初回インストール** —OnSuiteInstallBefore、OnSuiteInstallAfter
- **メンテナンス** —OnSuiteMaintBefore、OnSuiteMaintAfter
- **アップデート** —OnSuiteUpdateBefore、OnSuiteUpdateAfter

アドバンスト UI またはスイート / アドバンスト UI インストールで起動された、その他すべてのイベントおよびイベントの呼び出しシーケンスは、アドバンスト UI またはスイート / アドバンスト UI インストールから個別に起動された InstallScript インストール、または、アドバンスト UI またはスイート / アドバンスト UI インストールから実行可能パッケージとして起動された InstallScript インストール内のそれらと同一に保持されます。

メモ・program...endprogram スタイルのインストールでは、このイベントは自動的には呼び出されません。

OnUninstall

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

OnUninstall イベントは、インストールが /uninst パラメーターを使って実行された場合に呼び出されます。これは、/uninst パラメーターが使われたときに呼び出される唯一のイベントです。OnUninstall イベントのデフォルト コードは、以前にインストールされた製品をアンインストールします。

関数

関数とは、特定のタスクを実行するために同時に実行される一連の指示の名前です。

関数の特徴

各関数には次の特徴があります :

- 関数には名前が付いています。各関数は固有の名前を持ちます。関数の名前を呼び出すと、どの一連の指示が実行されるかがわかり、また一貫した結果が得られます。ある関数の内部から別の関数を呼び出すことも可能です。
- 関数は独立したものです。多くの場合、関数はプログラムの別の部分を参照すること無しにその指示を実行することができます。
- 関数は特定のタスクを実行します。タスクとは、例えばビットマップを表示する、ファイル圧縮する、またはフォルダーを作成するなど、スクリプトが実行するべき単一の動作です。
- 関数はスクリプトへ値を戻すことができます。スクリプトが実行されたとき、関数の指示を実行します。指示の結果に基づき、関数はスクリプトへ情報を返します。

関数の種類

InstallShield では、セットアップスクリプト内で 3 種類の関数を利用することができます :

テーブル 1 ・関数の種類

関数の種類	説明
ビルトイン関数	InstallShield が提供する関数または標準ダイアログ用に含まれいる関数です。
ユーザー定義関数	ユーザーが作成した関数です。
DLL 呼び出し関数	DLL で呼び出すことができる関数です。

メモ・C プログラム言語と同様に、InstallScript はネストされた関数ブロックをサポートしません。

ビルトイン関数を使う

InstallShield はセットアップ スクリプトで利用できる何百ものビルトイン関数を装備しており、プログラム グループやアイテムの作成、フォルダーの操作、リストでの作業、セットアップ ステータスの監視、ダイアログの作成やファイルの操作、その他を行うことができます。InstallShield スクリプトコンパイラはこれらの関数名を既に認識しているので、関数を利用する前に宣言する必要はありません。

関数名とフォーマットについて理解する

ビルトイン関数を呼び出すためには、名前とそのフォーマットについて理解している必要があります。

要件に一致する関数を見つけるには、利用可能関数が主要なカテゴリ別に説明されている「カテゴリー別ビルトイン関数」を参照してください。[カテゴリー] リンクをクリックすると、そのカテゴリーに関連する関数と説明の一覧が表示されます。

すべてのビルトイン関数は、[ビルトイン関数] セクションにアルファベット順にリストされています。一覧にある関数の説明をすべて表示するには、その名前をクリックします。その関数についてのヘルプトピックは関数のフォーマットを提供します。

たとえば、AskYesNo はビルトイン関数で、ダイアログに質問を表示してエンドユーザーが [はい] または [いいえ] の何れかのボタンをクリックして応答するまで待機します。AskYesNo は次のフォーマットを持ちます :

```
AskYesNo (szQuestion, nDefault);
```

フォーマットは関数名の正しいスペルを表示し、次に括弧で括られた関数のパラメーターリストが続きます。ビルトイン関数のヘルプトピックでは、各パラメーターが ハンガリー表記で表示されます。これは、その位置で渡さなくてはならないデータタイプを示します。AskYesNo には 2 つのパラメーター (最初のパラメーターは文字列、2 番目は数字) が必要です。

メモ・C と同様に、InstallScript も大文字と小文字を区別します。ビルトイン関数名を大文字にするときには充分な注意が必要です。

スクリプトでビルトイン関数を利用する

スクリプトでビルトイン関数を利用するには、必要な数のパラメーターを渡すこと、また各パラメーターで渡すデータがその位置に示されたタイプであることを確認してください。不適切な数のパラメーターを渡した場合、あるいはパラメーター位置に不適切なデータ型を渡した場合、スクリプトはコンパイルしません。

各ビルトイン関数の特定のマニュアルはそのパラメーターの説明を提供します。AskYesNo では、szQuestion はダイアログに表示される質問で、nDefault は [はい] と [いいえ] のどちらのボタンをデフォルトで選択するかを示します。2 つの定義済み定数 (YES または NO) の一方を n Default に渡すことができます。

[はい] ボタンがデフォルトで選択されているダイアログを参考にして下さい。このダイアログを表示するには、次のように AskYesNo を呼び出します :

```
AskYesNo (" インストレーションが完了しました。ReadMe ファイルをお読みになりますか ?", YES);
```

メモ・パラメーターとして渡された文字列リテラルは引用符、または二重引用符で囲まなくてはなりません。例えば、“ ファイルの転送中です。しばらくお待ちください ”、' これは文字列です '、または “C:¥¥Myfolder¥¥Myfile.ext”。

トラブルシューティング

次の事項に注意してください。

- InstallScript 関数では割り当てステートメントをパラメータとして渡すことができません。さらに && や || 演算子を関数への引数の中で使用することはできません。
- 参照によって関数に渡されるオートサイズ文字列変数は呼び出された関数の中ではオートサイズされません。関数が現在のパラメーターのサイズより大きい長さの値を割り当てようとすると、ランタイム エラー 401 が発生します。このエラーを回避するためには、リファレンスが関数に値を渡すときの特定の文字列サイズを宣言します。

カテゴリー別ビルトイン関数

InstallScript 言語では、次のカテゴリの関数が使用できます。

テーブル 2 - カテゴリ別ビルトイン関数

関数のカテゴリ	説明
バッチ ファイル関数	バッチファイルと共に機能します。
コンポーネント関数	InstallShield では、InstallShield Professional で使われていたコンポーネント関数にとってかわり、機能関数が用いられます。
構成ファイル関数	デフォルトのシステム構成ファイルを変更します。
デバイス ドライバー関数	DIFx を使用してデバイス ドライバーをインストールおよびアンインストールします。
ダイアログ関数	InstallScript ダイアログおよびメッセージ ボックスを作成します。
ダイアログのカスタマイズ関数	InstallScript ダイアログをカスタマイズして、InstallScript ダイアログのテキスト、コントロール、および動作を変更します。
拡張性関数	ダイナミック リンク ライブラリでの関数の呼び出し、Windows API の呼び出し、別のアプリケーションやセットアップ スクリプトの起動、メモリへの .dll ファイルのロードまたはアンロードを行います。
機能関数	ファイル メディアを制御します。また、スクリプトで作成された機能のセットの作成、処理も行います。
ファイル関数とフォルダー関数	テキストファイル、バイナリファイル、フォルダーと共に機能します。
FlexNet Connect の関数	FlexNet Connect を利用して、エンドユーザーに提供されているアップデートについて通知できるようにします。
情報関数	以下のような動作環境で利用可能なリソースに関するデータ (ディスクスペース、メモリ、オペレーティングモード) を提供します。
初期化ファイル関数	初期化とプロファイル ファイルから、または、それらへの情報を取得またはコピーを行います。
リスト操作関数	セットアップ スクリプトでリストを実装します。
ログファイル関数	ログファイルのカスタムログ記録セクションから情報を取得し、またそこへ情報をコピーします。
長いファイル名関数	長いファイル名を扱うオペレーティングシステムが認識できるよう、短いファイル名から長いファイル名を作成し、短いファイル名を長いファイル名に変換し、長いファイル名の周りに二重引用符を配置します。

テーブル 2 - カテゴリ別ビルトイン関数（続き）

関数のカテゴリ	説明
その他の関数	低位ハードウェア インターフェイス、機能の作成と操作、そしてユーザー出力など様々な目的に利用できます。
オブジェクト関数	オブジェクトを初期化します。また、オブジェクト ステータス情報の取得および設定も行います。
パスバッファー関数	検索パスを含む文字列と共に機能します。パス文字列関数は、パスバッファーとして知られる一意の一時文字列変数で機能します。
レジストリ関数	レジストリへアクセスし、レジストリキーの読み取り、作成、および削除を行って、アンインストール用のレジストリ関連パラメーターを設定することができます。
共有およびロック ファイル関数	共有またはロックファイルを処理します。
シェル関数	ショートカットを作成、既存のショートカットを削除、およびショートカットの構成を行います。
レジストリ関連の特殊関数	必要最小限のレジストリキーと値を設定します。レジストリ関連の特殊関数は、アプリケーションごとのパスキー、アプリケーションのアンインストールキー、またはアプリケーション情報キーを使ってのみ動作します。
SQL 関数	カタログへの接続、SQL 関連のダイアログの作成、および SQL ランタイム エラーの取得など、SQL タスクの実行を行います。
文字列関数	文字列変数およびリテラルを操作します。文字列関数は C 言語標準関数と同様に動作します。戻り値もまた C 言語の規則に従います。
スイート / アドバンスト UI およびアドバンスト UI の対話関数	InstallScript パッケージを含むアドバンスト UI またはスイート / アドバンスト UI インストールとインタラクトする、または InstallScript アクションを含むスイート / アドバンスト UI インストールとインタラクトします。
テキスト置換	文字列を別の文字列に関連付けて（例、"<MYTEXTSUB>" を「テキストサブ値」に関連付ける）、他の文字列内で前の文字列を後の文字列に置換します（例、「この文字列は <MYTEXTSUB> のテキスト置換をデモンストレーションします」を「この文字列は 『テキストサブ値』のテキスト置換をデモンストレーションします」に変更する）。
アンインストール関数	インストール済みのアプリケーションのアンインストールおよび / またはメンテナンスセットアップに必要なサービスを実行します。
ユーザー インターフェイス関数	特定のエラーメッセージや、エラーボックスのタイトルをカスタマイズできます。ただし、セットアップの開発時に遭遇することがある内部エラーメッセージの中には、ユーザーインターフェイス関数で変更できないものがあります。

テーブル 2 · カテゴリ別ビルトイン関数（続き）

関数のカテゴリ	説明
バージョンチェック関数	特定ファイルのバージョンの取得、ファイルの検出とそのバージョンの取得、または既存ファイルを検索してファイルのより新しいバージョンのインストールを行います。関数は圧縮済みファイルまたは非圧縮済みファイルのどちらにも利用できます。
Windows Installer 関数	Windows Installer エンジンによってエクスポートされます。これらの関数を利用して、実行中のインストールのプロパティをクエリしたり操作したりすることができます。

バッチ ファイル関数

バッチファイルおよび構成ファイルで作業をおこなう際、普通のテキストファイルとしてこれらを扱うこともできますが、バッチファイルおよび構成ファイルの変更を目的として設計された InstallScript 関数を使用することもできます。

Ez バッチ関数と拡張バッチ関数

InstallScript バッチファイル関数には、Ez と拡張の 2 種があります。Ez 関数には、事前にプログラミングされた機能が多数備わっているため、素早く手軽に利用できます。構成ファイルとバッチファイルへの変更を柔軟かつ的確に制御する必要がある場合は、拡張関数を使用してください。

戻り値

InstallShield は、バッチ関数が失敗したときにメッセージを表示しませんが、ゼロよりも小さい値 (< 0) を返します。戻り値によって、関数が正しく実行されたかどうかが分かります。戻り値をチェックして、結果に基づいてメッセージを表示することができます。

Ez バッチ ファイル関数

Ez バッチファイル関数はデフォルトのバッチファイルへ変更を加えます。BatchSetFileName への呼び出しで変更されない限り、デフォルトバッチファイルは Autoexec.bat ファイルで、起動シーケンス中にシステムによって実行されます。各 Ez バッチファイル関数は、デフォルトのバッチファイルを開いて指定された変更を行った後、自動的に保存を行う点に注意してください。Ez バッチファイル関数を利用する際は開いたり保存するための呼び出しを行いません。

テーブル 3 · Ez バッチファイル関数

関数	説明
EzBatchAddPath	PATH コマンドで検索パス、または環境変数に割り当てた値にディレクトリを追加して、デフォルトのバッチファイルを変更します。
EzBatchAddString	テキストの行をデフォルトのバッチファイルに追加します。
EzBatchReplace	デフォルトのバッチファイルのステートメントを置き換えます。

拡張バッチ ファイル関数

拡張バッチファイル関数は Ez バッチファイル関数とは異なり、バッチファイルについて幅広い柔軟性を提供し、より詳細なコントロールを可能にします。バッチファイルについてより幅広く複雑な変更を加える必要があるときにこれらの関数を利用します。

これらの拡張関数を使ってバッチファイルを編集するには、まず BatchFileLoad を呼び出してファイルをメモリーにロードします。バッチファイルへの変更が完了した時点で、BatchFileSave を呼び出してファイルを保存しなくてはなりません。

InstallScript カスタムアクションが初期化される際、InstallShield はターゲットシステムのスタートアップバッチファイル (Autoexec.bat) をデフォルトバッチファイルとして選択します。BatchSetFileName への呼び出しで変更されない限り、特に別のファイル名が指定されない場合は BatchFileLoad によってこのファイルがメモリに読み込まれます。デフォルトバッチファイルの完全修飾ファイル名を決定するには、BatchGetFileName を呼び出します。

テーブル 4 · 拡張バッチファイル関数

関数	説明
BatchAdd	環境変数をバッチファイルに追加します。
BatchDeleteEx	バッチファイルの行を削除します。
BatchFileLoad	拡張バッチ関数を使って編集するため、バッチファイルをメモリにロードします。
BatchFileSave	BatchFileLoad を使ってロードしたバッチファイルを保存します。
BatchFind	バッチファイルのアイテムを検索します。
BatchGetFileName	デフォルトのバッチファイルの完全修飾ファイル名を読み出します。
BatchMoveEx	バッチファイルの中のアイテムを移動します。
BatchSetFileName	デフォルトのバッチファイルとするバッチファイルを指定します。

関連する関数

テーブル 5 · 関連する関数

関数	説明
SdShowFileMods	ファイル変更の提案や手順についてのオプションを表示するダイアログを作成します。

コンポーネント関数

Windows Installer に基づくインストレーションでは、コンポーネントではなく機能がインストールプロジェクト構成の最上位レベルとして利用されるためです。InstallShield ではコンポーネントに関連する関数は旧式の呼び名となります。サポートされているすべてのコンポーネント関数については、スクリプトでは対応する 機能関数を利用してください。

例えば、ComponentAddItem 関数は FeatureAddItem となります。詳細および機能関連関数の一覧は、「機能関数」を参照してください。

構成ファイル関数

構成ファイル関数は、デフォルトのシステム構成ファイルを変更します。構成ファイル関数には 2 つのタイプがあります。

- Ez Config.sys ファイル関数
- 詳細構成ファイル関数

Ez Config.sys ファイル関数

Ez 構成ファイル関数は、デフォルトシステム構成ファイルを変更します。ConfigSetFileName への呼び出しで変更されない限り、デフォルトシステム構成ファイルは Config.sys ファイルで、起動シーケンス中にシステムによって実行されます。

テーブル 6 ・ Ez 構成ファイル関数

関数	説明
EzConfigAddDriver	デバイスドライバーのステートメントをデフォルトのシステム構成ファイルに追加します。
EzConfigAddString	ステートメントやテキストの行をデフォルトのシステム構成ファイルに加えます。
EzConfigGetValue	FILES や BUFFERS などのシステム構成ファイルパラメーターの値を取得します。
EzConfigSetValue	FILES や BUFFERS などのシステム構成ファイルパラメーターの値を設定します。

メモ・これらの各関数はデフォルトシステム構成ファイルを開いて、割り当てられたタスクを実行した後、ファイルを元のディスクへ保存します。拡張構成ファイル関数と同様に、構成ファイルのロードまたは保存は不要です。

詳細構成ファイル関数

詳細構成ファイル関数は上級開発者向けで、幅広い柔軟性を持ち、Ｅｚ 構成ファイル関数に比べてより詳細にシステム構成ファイルを制御することができます。これらの詳細関数を使ってシステム構成にアクセス、並びに編集するには、まず ConfigFileLoad を呼び出します。システム構成ファイルの編集を終了した時点で、ConfigFileSave を呼び出して変更を保存します。関数 ConfigGetFileName と ConfigSetFileName は詳細構成ファイル関数、及び Ez 構成ファイル関数の両方で利用できます。

テーブル 7 - 詳細構成ファイル関数

関数	説明
ConfigAdd	メモリにロードされたシステム構成ファイルにステートメントを追加します。
ConfigDelete	アイテムをシステム構成ファイルから削除します。
ConfigFileLoad	システム構成ファイルを編集用にメモリにロードします。
ConfigFileSave	ConfigFileLoad を使ってメモリにロードされたシステム構成ファイルを保存します。
ConfigFind	システム構成ファイル内のアイテムを検索します。
ConfigGetFileName	デフォルトシステム構成ファイルの完全修飾名を読み出します。
ConfigGetInt	システム構成ファイルの値を読み出します。
ConfigMove	システム設定ファイル内のアイテムを移動します。
ConfigSetFileName	システム設定ファイルの完全修飾ファイル名を指定します。
ConfigSetInt	システム設定ファイルの値を設定します。

関連する関数

テーブル 8 - 関連する関数

関数	説明
SdShowFileMods	ファイル変更の提案や手順についてのオプションを表示するダイアログを作成します。

デバイス ドライバー関数

以下の関数は、Windows Driver Install Frameworks (DIFx) を使用してデバイス ドライバーのインストールとアンインストールを処理します。DIFx と DIFxAPI についての詳細は、MSDN ライブラリを参照してください。

テーブル 9 - デバイス ドライバー関数

関数	説明
DIFxDriverPackageGetPath	ドライバー パッケージがドライバー ストアにプレインストールされた後で、ドライバー パッケージの .inf ファイルのパスを取得します。
DIFxDriverPackageInstall	ドライバー ストアにドライバー パッケージをプレインストールして、ドライバーをシステムにインストールします。
DIFxDriverPackagePreinstall	プラグ アンド プレイ (PnP) 関数ドライバーのドライバー パッケージをドライバー ストアにプレインストールし、ドライバー パッケージの .inf ファイルをシステムの .inf ファイル ディレクトリにインストールします。
DIFxDriverPackageUninstall	システムから指定のドライバー パッケージをアンインストールし、ドライバー ストアからドライバー パッケージを削除します。

ダイアログ関数

以下の一部の関数は、[はい / いいえ] ダイアログやメッセージ ボックスなどのシンプルなダイアログを作成します。いくつかの関数では、様々な種類の一般的なダイアログを簡単に表示することができます。その他の関数は、スクリプト ダイアログ (Sd) 関数です。Sd ダイアログは、カスタム入力を含むダイアログを作成する特別な InstallScript 定義関数を使用して作成されます。その後、選択したアクションに基づいてスクリプトに値を自動的に返します。

メモ・[キャンセル] ボタンがあるダイアログは、ボタンをクリックしても CANCEL 値を返しません。その代わり、OnCanceling イベントハンドラーが呼び出されます。

テーブル 10・ダイアログ関数

関数	プロジェクトの種類	説明
AdminAskPath	InstallScript MSI	管理インストール時 (エンドユーザーが InstallScript MSI プロジェクト /a 引数を使った Setup.exe を実行する場合) にエンドユーザーに対してインストール先パスの入力を求めるダイアログを表示します。
AskDestPath	InstallScript、InstallScript MSI	インストール先パス情報を要求するダイアログを表示します。
AskOptions	InstallScript、InstallScript MSI	エンドユーザーに、チェック ボックスまたはラジオボタンを使ってオプションを選択するように求めるダイアログを表示します。
AskPath	InstallScript、InstallScript MSI	エンドユーザーにパスを入力するよう要求するダイアログを表示します。
AskText	InstallScript、InstallScript MSI	エンドユーザーにテキストを入力するよう要求するダイアログを表示します。
AskYesNo	基本の MSI、InstallScript、InstallScript MSI	エンド ユーザーに [はい] または [いいえ] ボタンをクリックして質問に答えるよう要求するメッセージ ボックスを表示します。
EnterDisk	基本の MSI、InstallScript、InstallScript MSI	エンド ユーザーに特定のディスクを要求するメッセージ ボックスを表示します。
EnterDiskError	基本の MSI、InstallScript、InstallScript MSI	指定されたパスとファイルが存在するかどうかを確認します。指定されたパスにファイルが存在しない場合、関数は適切なエラー メッセージ ボックスを表示してから、指定されたファイルが存在するかどうかに従って success または failure を返します。
EnterLoginInfo	InstallScript、InstallScript MSI	エンド ユーザーがユーザー名とパスワードを指定できるダイアログを表示します。ダイアログは、指定された情報を検証または使用しません。また、ダイアログが指定された情報のエラーをチェックすることもありません。
EnterPassword	InstallScript、InstallScript MSI	エンドユーザーに対してパスワードを問い合わせるダイアログを表示します。エンドユーザーが編集ボックス内に入力する文字はアスタリスク (*) として表示されます。

テーブル 10 - ダイアログ関数 (続き)

関数	プロジェクトの種類	説明
FeatureDialog	InstallScript、InstallScript MSI	エンド ユーザーが機能を選択してインストール先を指定できるダイアログを表示します。
MessageBox	基本の MSI、InstallScript、InstallScript MSI	メッセージをメッセージ ボックスに表示します。
MessageBoxEx	基本の MSI、InstallScript、InstallScript MSI	メッセージをメッセージ ボックスに表示します。
RebootDialog	InstallScript、InstallScript MSI	エンド ユーザーがコンピューターの再起動を選択できるメッセージ ボックスを表示します。
SdAskDestPath	InstallScript、InstallScript MSI	エンド ユーザーが別のインストール先パスを選択できるダイアログを作成します。
SdAskDestPath2	InstallScript、InstallScript MSI	エンド ユーザーが別のインストール先パスを選択できるダイアログを作成します。
SdAskOptions	InstallScript、InstallScript MSI	標準 AskOptions 関数より柔軟性の高いダイアログを作成します。
SdAskOptionsList	InstallScript、InstallScript MSI	エンドユーザーがリストのアイテムを選択および選択解除できるダイアログを表示します。
SdBitmap	InstallScript、InstallScript MSI	ダイアログ上にビットマップを表示します。
SdConfirmNewDir	InstallScript、InstallScript MSI	エンド ユーザーが選択したフォルダーを確認するようにプロンプトするメッセージ ボックスを表示します。
SdConfirmRegistration	InstallScript、InstallScript MSI	エンド ユーザーに対して、**SdRegisterUser** または **SdRegisterUserEx** によって表示されたダイアログに入力した情報の確認をプロンプトするメッセージ ボックスを表示します。
SdCustomerInformation	InstallScript、InstallScript MSI	エンド ユーザーがインストール中の製品のユーザー名および会社名を指定できるダイアログを表示します。このダイアログには、エンド ユーザーが製品をすべてのユーザーにインストールするか、または現在のユーザーにのみインストールするかを指定できるラジオボタンを含めることもできます。

テーブル 10 - ダイアログ関数 (続き)

関数	プロジェクトの種類	説明
SdCustomerInformationEx	InstallScript、InstallScript MSI	エンド ユーザーがインストール中の製品のユーザー名、会社名、およびシリアル番号を指定できるダイアログを表示します。このダイアログには、エンド ユーザーが製品をすべてのユーザーにインストールするか、または現在のユーザーにのみインストールするかを指定できるラジオボタンを含めることもできます。
SdDiskSpace2	InstallScript、InstallScript MSI	次のいずれかを表示するダイアログを表示します : • ボリューム、必要なディスク容量、利用可能なディスク容量、および必要なディスク容量と使用可能なディスク容量との差異のリスト。 • ターゲット システムに、インストールで必要なディスク容量が不足していることを示す警告メッセージ。ダイアログには、ボリューム、必要な容量、使用可能容量および必要な容量と使用可能な容量の差についてのリストビューも表示されます。
SdDiskSpaceRequirements	InstallScript、InstallScript MSI	ボリュームのリスト、必要なディスク容量、利用可能なディスク容量、および必要なディスク容量と使用可能なディスク容量との差異を表示します。 SdDiskSpace2 は、この関数に優先します。
SdDisplayTopics	InstallScript、InstallScript MSI	トピックのリストを表示します。
SdExceptions	InstallScript、InstallScript MSI	共有ファイル、ロックされた（使用中）ファイル、または読み取り専用ファイルを検出したことをエンド ユーザーに知らせるメッセージ ボックスが表示されます。
SdFeatureDialog	InstallScript、InstallScript MSI	エンド ユーザーがインストールする機能とインストール先フォルダーを選択できるダイアログを表示します。
SdFeatureDialog2	InstallScript、InstallScript MSI	エンド ユーザーがインストールするフォルダー、機能、そしてサブ機能を選択できるダイアログを表示します。
SdFeatureDialogAdv	InstallScript、InstallScript MSI	エンド ユーザーがインストールする機能とインストール先フォルダーを選択できるダイアログを表示します。
SdFeatureMult	InstallScript、InstallScript MSI	エンド ユーザーがインストールする機能やサブ機能を選択できるダイアログを表示します。ディスク容量に関する補足データも備えているので、インストールに最適な場所を決定するのに役立ちます。

テーブル 10・ダイアログ関数（続き）

関数	プロジェクトの種類	説明
SdFeatureTree	InstallScript、InstallScript MSI	エンド ユーザーがインストールする機能やサブ機能を選択できる、ツリー コントロールのあるダイアログを表示します。ディスク容量に関する補足データも備えているので、インストールに最適な場所を決定するのに役立ちます。
SdFilesInUse	InstallScript MSI	開いた状態でファイルをロックしているアプリケーションの一覧があるリスト ボックスを含むダイアログを表示します。
SdFinish	InstallScript、InstallScript MSI	エンドユーザーにセットアップが完了したことを知らせて、情報ファイルの表示、またはアプリケーションの起動オプションを選択できるダイアログを表示します。
SdFinishEx	InstallScript、InstallScript MSI	インストールが完了したことをエンドユーザーに知らせるダイアログを表示します。
SdFinishReboot	InstallScript、InstallScript MSI	セットアップが完了したことをユーザーに知らせ、Windows やコンピューターの再起動のオプションを選択できるダイアログを表示します。
SdFinishUpdate	InstallScript、InstallScript MSI	インストールが完了したことを通知するダイアログを表示します。このダイアログには、アプリケーションのアップデートを確認するためのオプションが含まれます。 ***メモ・SdFinishUpdate** はアップデートの確認を行いません。アップデートの確認をするためには、InstallScript コードに FlexNet Connect API の呼び出しを追加します。詳細は、FlexNet Connect SDK ドキュメントを参照してください。*
SdFinishUpdateEx	InstallScript、InstallScript MSI	インストールが完了したことを通知するダイアログを表示します。このダイアログには、アプリケーションのアップデートを確認するためのオプションが含まれます。 ***メモ・SdFinishUpdateEx** はアップデートの確認を行いません。アップデートの確認をするためには、InstallScript コードに FlexNet Connect API の呼び出しを追加します。詳細は、FlexNet Connect SDK ドキュメントを参照してください。*

テーブル 10 - ダイアログ関数 (続き)

関数	プロジェクトの種類	説明
SdFinishUpdateReboot	InstallScript、InstallScript MSI	インストールが完了したことを通知するダイアログを表示します。ダイアログは、エンド ユーザーに対してシステムの再起動オプションを提供し、アプリケーションのアップデートも確認します。 ***メモ - SdFinishUpdateReboot** はアップデートの確認を行いません。アップデートの確認をするためには、InstallScript コードに FlexNet Connect API の呼び出しを追加します。詳細は、FlexNet Connect SDK ドキュメントを参照してください。*
SdLicense	InstallScript、InstallScript MSI	複数行編集フィールドに使用許諾契約書が記載されたダイアログを表示します。ライセンス同意書は、パラメーター szLicenseFile で指定されるテキストファイルに格納されます。 ダイアログは、質問をスタティック テキスト フィールドに表示します。エンド ユーザーは、[はい] または [いいえ] ボタンをクリックして質問に答えます。 SdLicenseEx 関数は、この関数に優先します。
SdLicense2	InstallScript、InstallScript MSI	複数行編集フィールドに使用許諾契約書が記載されたダイアログを表示します。ライセンス同意書は、パラメーター szLicenseFile で指定されるテキストファイルに格納されます。 このダイアログは 2 つのラジオボタンを表示します (使用許諾契約書の条件に同意するためのボタンと、同意しないためのボタン)。エンド ユーザーが使用許諾契約書の条件に同意するための適切なボタンをクリックすると、[次へ] ボタンが有効になります。 SdLicense2Ex 関数は、この関数に優先します。
SdLicense2Ex	InstallScript、InstallScript MSI	複数行編集フィールドに使用許諾契約書が記載されたダイアログを表示します。使用許諾契約書は、パラメーター szLicenseFile で指定されるテキスト ファイル (.txt) またはリッチテキスト ファイル (.rtf) 形式で格納されます。 このダイアログは 2 つのラジオボタンを表示します (使用許諾契約書の条件に同意するためのボタンと、同意しないためのボタン)。エンド ユーザーが使用許諾契約書の条件に同意するための適切なボタンをクリックすると、[次へ] ボタンが有効になります。

テーブル 10 - ダイアログ関数 (続き)

関数	プロジェクトの種類	説明
SdLicense2Rtf	InstallScript、InstallScript MSI	複数行編集フィールドに使用許諾契約書が記載されたダイアログを表示します。使用許諾契約書は、パラメーター szLicenseFile で指定されるテキスト形式またはリッチテキスト形式 (RTF) のファイルに格納されます。 このダイアログは 2 つのラジオボタンを表示します (使用許諾契約書の条件に同意するためのボタンと、同意しないためのボタン)。エンド ユーザーが使用許諾契約書の条件に同意するための適切なボタンをクリックすると、[次へ] ボタンが有効になります。 SdLicense2Ex 関数は、この関数に優先します。
SdLicenseEx	InstallScript、InstallScript MSI	複数行編集フィールドに使用許諾契約書が記載されたダイアログを表示します。使用許諾契約書は、テキスト ファイル (.txt) またはリッチ テキスト ファイル (.rtf) で保存されます。 ダイアログは、質問をスタティック テキスト フィールドに表示します。エンド ユーザーは、[はい] または [いいえ] ボタンをクリックして質問に答えます。
SdLicenseRtf	InstallScript、InstallScript MSI	複数行編集フィールドに使用許諾契約書が記載されたダイアログを表示します。使用許諾契約書は、パラメーター szLicenseFile で指定されるテキスト形式またはリッチテキスト形式 (RTF) のファイルに格納されます。 ダイアログは、質問をスタティック テキスト フィールドに表示します。エンド ユーザーは、[はい] または [いいえ] ボタンをクリックして質問に答えます。 SdLicenseEx 関数は、この関数に優先します。
SdLogonUserBrowse	InstallScript、InstallScript MSI	エンド ユーザーが指定のドメインまたはサーバー、およびユーザー名を選択できるメッセージ ボックスを表示します。
SdLogonUserCreateUser	InstallScript、InstallScript MSI	エンド ユーザーが **SdLogonUserInformation** ダイアログで [新規ユーザー情報] ボタンを クリックしして、新しいユーザー情報を入力できるダイアログを表示します。
SdLogonUserInformation	InstallScript、InstallScript MSI	アカウントがインストール中に作成される場合、エンド ユーザーに既存のユーザーアカウント情報または新しいユーザー情報を入力するようプロンプトするダイアログを表示します。

テーブル 10 - ダイアログ関数 (続き)

関数	プロジェクトの種類	説明
SdLogonUserListGroups	InstallScript、InstallScript MSI	エンド ユーザーが指定のサーバーからグループを選択して、SdLogonUserCreateUser ダイアログの ″ グループ ″ フィールドに挿入することができるダイアログを表示します。
SdLogonUserListServers	InstallScript、InstallScript MSI	エンド ユーザーがユーザーアカウントが関連付けられているドメインまたはサーバーを参照することができるダイアログを表示します。
SdLogonUserListUsers	InstallScript、InstallScript MSI	エンド ユーザーが指定のドメインまたはサーバーに既存のユーザーを参照して選択できるダイアログを表示します。
SdOptionsButtons	InstallScript、InstallScript MSI	エンドユーザーがさまざまなオプションを選択できるユーザー定義のボタンを使ったダイアログを表示します。
SdOutOfDiskSpace	InstallScript MSI	Windows Installer の INSTALLMESSAGE_OUTOFDISKSPACE メッセージによってトリガーされたとき、このダイアログはターゲットシステムのディスク容量が足りないことを示すメッセージを表示します。 SdDiskSpace2 は、この関数に優先します。
SdPatchWelcome	InstallScript MSI	パッチ インストール中にエンド ユーザーに「ようこそ」メッセージを表示するダイアログを作成します。
SdRegisterUser	InstallScript、InstallScript MSI	エンド ユーザーがインストール中の製品のユーザー名および会社名を指定できるダイアログを表示します。
SdRegisterUserEx	InstallScript、InstallScript MSI	エンド ユーザーがインストール中の製品のユーザー名、会社名、およびシリアル番号を指定できるダイアログを表示します。
SdRMFilesInUse	InstallScript MSI	開いた状態でファイルをロックしているアプリケーションの一覧があるリスト ボックスを含むダイアログを表示します。ダイアログにはまた、インストールが、1) 再起動マネージャーを使用して、ファイルをロックしているアプリケーションを閉じる試みをするか、または 2) ロックされているファイルを上書きする試みをするか (結果として、インストールの完了に再起動が必要になる可能性が高くなります) エンドユーザーが指定できる 2 つのラジオ ボタンがあります。
SdSelectFolder	InstallScript、InstallScript MSI	エンド ユーザーがプログラム フォルダーのリストからフォルダーを選択できるダイアログを表示します。

テーブル 10 - ダイアログ関数 (続き)

関数	プロジェクトの種類	説明
SdSetupCompleteError	InstallScript、InstallScript MSI	エンド ユーザーに対してインストレールが完了する前に中断されたことを通知するダイアログを表示します。
SdSetupType	InstallScript、InstallScript MSI	エンドユーザーが、3 種類の標準セットアップ ([標準]、[最小]、または [カスタム]) の中から 1 つを選択できるダイアログを表示します。
SdSetupType2	InstallScript、InstallScript MSI	エンド ユーザーが、2 種類の標準セットアップ タイプ ([標準] と [カスタム]) のどちらかを選択できるダイアログを表示します。
SdSetupTypeEx	InstallScript MSI	エンド ユーザーが標準またはカスタム セットアップの種類を選択するためのダイアログを表示します。
SdShowAnyDialog	InstallScript、InstallScript MSI	リソース DLL から一般目的のダイアログを表示します。**SdShowAnyDialog** を使ってダイアログを表示すると、エンドユーザーが入力した内容を取得できません。
SdShowDlgEdit1	InstallScript、InstallScript MSI	1 つの単一行編集フィールドと他の静的コントロールを含むダイアログを表示します。
SdShowDlgEdit2	InstallScript、InstallScript MSI	2 つの単一行編集フィールドと他の静的コントロールを含むダイアログを表示します。
SdShowDlgEdit3	InstallScript、InstallScript MSI	3 つの単一行編集フィールドと他の静的コントロールを含むダイアログを表示します。
SdShowFileMods	InstallScript、InstallScript MSI	ファイルの変更をプレビューするダイアログを表示し、ここでエンド ユーザーは変更を承認、拒否、または変更をファイルに書き出すよう要求できます。
SdShowInfoList	InstallScript、InstallScript MSI	ダイアログでスクロール可能なメッセージのリストを表示します。
SdShowMsg	InstallScript、InstallScript MSI	メッセージを小さなウィンドウに表示します。
SdStartCopy	InstallScript、InstallScript MSI	エンドユーザーが指定したオプションや設定を表示するダイアログを表示します。
SdStartCopy2	InstallScript、InstallScript MSI	エンド ユーザーに対して、間もなくファイル転送処理が始まる事を通知するダイアログを表示します。ユーザーは必要に応じて設定を変更するために、[戻る] ボタンをクリックして前のダイアログに戻ることができます。

テーブル 10 · ダイアログ関数 (続き)

関数	プロジェクトの種類	説明
SdWelcome	InstallScript、InstallScript MSI	一般目的に使用するあいさつを表示します。
SdWelcomeMaint	InstallScript、InstallScript MSI	メンテナンス セットアップの最初に使用するダイアログを表示します。
SelectDir	InstallScript、InstallScript MSI	エンド ユーザーがフォルダーを選択できるダイアログを表示します。SelectDir は、フォルダーがない場合にそれを作成します。
SelectDirEx	InstallScript、InstallScript MSI	エンド ユーザーがフォルダーを選択できるダイアログを表示します。
SelectFolder	InstallScript、InstallScript MSI	エンド ユーザーがプログラム フォルダーのリストからフォルダーを選択できるダイアログを表示します。
SetupType	InstallScript、InstallScript MSI	エンド ユーザーが [標準]、[最小]、または [カスタム] セットアップを選択できるダイアログを表示します。
SetupType2	InstallScript、InstallScript MSI	エンド ユーザーが、2 種類の標準セットアップ タイプ ([完全] と [カスタム]) のどちらかを選択できるダイアログを表示します。
SprintfBox	基本の MSI、InstallScript、InstallScript MSI	単数または複数の文字、数値または文字列値で構成されるフォーマットされた文字列を返します。
SQLBrowse	InstallScript、InstallScript MSI	ユーザーがネットワーク上で提供されているすべての SQL Server のリストを表示することができるダイアログを作成します。 **SQLBrowse2** はこの関数に優先します。
SQLBrowse2	InstallScript、InstallScript MSI	エンド ユーザーが、接続に指定されたデータベース テクノロジ用にネットワーク上で提供されているすべてのデータベース サーバーのリストを表示できるようにするダイアログを作成します。
SQLServerLogin	InstallScript、InstallScript MSI	SQL ログイン情報を指定するスクリプトで使われるダイアログを作成します。情報には、ログイン ID とパスワードが含まれます。
SQLServerSelect	InstallScript、InstallScript MSI	ターゲットにするサーバーを指定するダイアログを作成します。

テーブル 10 - ダイアログ関数 (続き)

関数	プロジェクトの種類	説明
SQLServerSelectLogin	InstallScript、InstallScript MSI	ターゲットされたエンド ユーザーが、現在の接続に使用する SQL Server および使用するログインの資格情報を指定できるログイン ダイアログを作成します。ダイアログでは、DSN を通してアクセスされた SQL Server のリストを含むコンボ ボックスが表示されます。エンドユーザーは、このコンボ ボックスでサーバー名を入力することもできますし、コンボ ボックスの隣にある [参照] ボタンをクリックすることもできます。このボタンをクリックすると、インターネットで提供されている SQL Servers がすべて表示されます。 **SQLServerSelectLogin2** はこの関数に優先します。
SQLServerSelectLogin2	InstallScript、InstallScript MSI	デフォルト スクリプトで使用されるログイン ダイアログを作成します。このダイアログで、ターゲットされたエンド ユーザーは、現在の接続に使用する SQL Server と使用するログインの認証情報を指定できます。ダイアログでは、DSN を通してアクセスされた SQL Server のリストを含むコンボ ボックスが表示されます。エンドユーザーは、このコンボ ボックスでサーバー名を入力することもできますし、[サーバー名] コンボ ボックスの隣にある [参照] ボタンをクリックすることもできます。このボタンをクリックすると、インターネットで提供されている SQL Servers がすべて表示されます。 オプションで、この関数は、接続情報に関連付けられた接続名も表示します。またオプションで、エンド ユーザーは現在の接続にどのデータベース カタログを使用するのかを指定することができます。
SQLServerSelectLoginEx	InstallScript、InstallScript MSI	デフォルト スクリプトで使用されるログイン ダイアログを作成します。このダイアログで、ターゲットされたエンド ユーザーは、現在の接続に使用する SQL Server と使用するログインの認証情報を指定できます。ダイアログでは、DSN を通してアクセスされた SQL Server のリストを含むコンボ ボックスが表示されます。エンドユーザーは、このコンボ ボックスでサーバー名を入力することもできますし、[サーバー名] コンボ ボックスの隣にある [参照] ボタンをクリックすることもできます。このボタンをクリックすると、インターネットで提供されている SQL Servers がすべて表示されます。 この関数は、接続情報に関連付けられた接続名も表示します。 **SQLServerSelectLogin2** はこの関数に優先します。

テーブル 10 - ダイアログ関数 (続き)

関数	プロジェクトの種類	説明
Welcome	InstallScript、InstallScript MSI	ようこそ情報を表示するダイアログを表示します。

ダイアログのカスタマイズ関数

次の関数を使って新しい InstallScript ダイアログを作成およびカスタマイズして、テキスト、コントロール、および InstallScript ダイアログの動作を変更できます。一部の関数は、カスタム ダイアログ プロセスを処理します。

作成する Windows ダイアログはいずれもセットアップ スクリプトで利用することができます。ダイアログには一行、または複数行編集ボックス、あるいは単数、または複数選択リストボックス、コンボボックス、ラジオボタン、チェック ボックス、およびプッシュボタンを標準コントロールとして含むことが可能です。より高度なコントロールには、**CmdGetHwndDlg**、**LOWORD**、および **HIWORD** 拡張関数を利用することができます。

テーブル 11 - ダイアログのカスタマイズ関数

関数	説明
CmdGetHwndDlg	ダイアログのハンドルを読み出します。
CtrlClear	編集、スタティック、リストボックス、またはコンボボックスコントロールの内容を削除します。
CtrlDir	リストボックスまたはコンボボックスにディレクトリ一覧やファイル一覧を記入します。
CtrlGetCurSel	選択されたアイテムをリストボックスまたはコンボボックスから返します。
CtrlGetDlgItem	カスタム ダイアログ内のコントロールのウィンドウ ハンドルを取得します。
CtrlGetMLEText	複数行編集フィールドまたはスタティックフィールドのテキストを取得します。
CtrlGetMultCurSel	選択されたアイテムを複数選択リストボックスから返します。
CtrlGetState	ダイアログのラジオ ボタン、チェック ボックス、またはプッシュ ボタン コントロールの状態を取得します。
CtrlGetSubCommand	**WaitOnDialog** 関数呼び出し後のコントロールで行われた操作を読み出します。
CtrlGetText	編集フィールド、スタティックフィールド、またはコンボボックスの編集フィールドのテキストを読み出します。
CtrlGetUrlForLinkClicked	エンド ユーザーがクリックしたリンクへの URL を取得します。
CtrlPGroups	ターゲットシステムに存在するプログラムグループ名のリストを読み出します。

テーブル 11・ダイアログのカスタマイズ関数（続き）

関数	説明
CtrlSelectText	編集フィールドに表示されるテキストを選択します。
CtrlSetCurSel	リストボックスやコンボボックスの現在の選択を検出し設定します。
CtrlSetFont	ダイアログのコントロール用フォントを指定します。
CtrlSetList	リストの内容をリストボックスまたはコンボボックスに配置します。
CtrlSetMLEText	テキストを複数行編集フィールドに設定します。
CtrlSetMultCurSel	現在の選択を複数選択リストボックスに設定します。
CtrlSetState	チェック ボックス、ラジオボタン、またはプッシュボタンコントロールの現在の状態を設定します。
CtrlSetText	編集フィールド、スタティックテキストフィールド、またはコンボボックスの編集フィールドのテキストを設定します。
DefineDialog	InstallShield にカスタム ダイアログを登録します。
DialogSetFont	セットアップ中に表示されるダイアログのフォントを設定します。
DialogSetInfo	一部のビルトイン ダイアログ関数によって表示されるダイアログの表示アイテムを変更します。
EndCurrentDialog	**EndDialog** を呼び出して、現在表示されているダイアログを閉じます。
EndDialog	カスタム ダイアログを閉じます。
EzDefineDialog	InstallShield にカスタム ダイアログを登録します。
GetCurrentDialogName	現在表示されているダイアログの名前を読み出します。これは、ダイアログが定義されたときに **EzDefineDialog** への呼び出しで指定された名前です。
GetFont	フォントのハンドルを取得します。
HIWORD	32 ビット整数から高位の単語を読み出します。
LOWORD	32 ビット整数から低位の単語を読み出します。
ReleaseDialog	ダイアログに関連付けられたメモリを解放します。
SdGeneralInit	[次へ]、[戻る]、および [キャンセル] ボタンの状態を有効または無効に設定する処理を含む、標準ダイアログの初期化を行います。また、この関数はコントロール ID 700 から 724、および 202 を含むスタティック コントロール上の %P、%VS、および %VI インスタンスすべてを置換します。

テーブル 11 - ダイアログのカスタマイズ関数 (続き)

関数	説明
SdInit	Sd ダイアログ関数を呼び出すセットアップを作成します。
SdLoadString	指定されたリソース ID に関連付けられた文字列値を返します。
SdMakeName	カスタムダイアログのセクション名を作成します。このセクション名は、InstallShield Silent が使用する .iss ファイルの書き込みや読み取りに使われます。
SdProductName	製品名をスクリプト ダイアログの特定の静的フィールドに挿入します。
SdSubstituteProductInfo	すべての %P、%VS、そして %VI プレースホルダーをシステム変数 IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、そして IFX_INSTALLED_DISPLAY_VERSION の値と置き換えます。この関数は、文字列を表示する前にこの置換を自動的に実行しない MessageBox などの関数を呼び出す前に利用することができます。
SilentReadData	InstallShield Silent に、カスタム ダイアログ用に .iss ファイル ダイアログ データを読み込むよう指示します。
SilentWriteData	InstallShield Silent に、カスタム ダイアログ用に .iss ファイル ダイアログ データへ書き込むよう指示します。
WaitOnDialog	カスタム ダイアログを示します。

拡張性関数

拡張性関数を使うとダイナミックリンクライブラリでの関数呼び出し、Windows API の呼び出し、または別のアプリケーションやセットアップスクリプトの起動が可能になります。UseDLL 関数と UnUseDLL 関数を利用すると、メモリへ DLL をロードまたはアンロードし、DLL を活用することができます。LaunchApp 関数と LaunchAppAndWait 関数を利用すると、スクリプトが実行中に別の Windows アプリケーションまたは DOS アプリケーションを起動することができます。

テーブル 12 - 拡張性関数

関数	説明
CallDLLFx	関数を外部 DLL から呼び出します。
Delay	セットアップスクリプトの実行を遅らせます。
LaunchApp	別のプログラムを起動します。**LaunchApplication** はこの関数に優先します。
LaunchAppAndWait	別のプログラムを起動して、そのプログラムが終了するのを待ちます。**LaunchApplication** はこの関数に優先します。
LaunchAppAndWaitInitStartupInfo	LAAW_STARTUPINFO システム変数と LAAW_PARAMETERS システム変数を適切なデフォルト値に初期化します。**LaunchApplicationInit** はこの関数に優先します。

テーブル 12 - 拡張性関数 (続き)

関数	説明
LaunchApplication	Windows API 関数 **CreateProcess** または Windows API 関数 **ShellExecuteEx** を使用して、指定されたアプリケーションを起動します。アプリケーションが起動されたあと、インストールで新しい WaitForApplication 関数を呼び出して (オプション)、アプリケーションが終了するのを待機することができます。
LaunchApplicationInit	LAAW_STARTUPINFO システム変数と LAAW_PARAMETERS システム変数を適切なデフォルト値に初期化します。この関数はインストール初期化中に自動的に呼び出されます。
UnUseDLL	メモリから DLL をアンロードします。
UseDLL	DLL をメモリにロードします。
WaitForApplication	戻される前に、実行中のアプリケーションが終了するのを待機します。

機能関数

下の関数を利用して、ファイルメディアを制御したり、スクリプト作成の機能セットを作成することができます。

テーブル 13 - 機能関数

関数	プロジェクトの種類	説明
FeatureAddCost	InstallScript	機能が追加のインストール操作を含むように指定します。これは、インストール中、進行状況バーを更新するときに必要です。 ***メモ -*** *この関数は、ファイルメディアにのみサポートされています。**FeatureGetData** または **FeatureSetData** を使用して、スクリプトで作成された機能のサイズを設定します。*
FeatureAddItem	InstallScript、InstallScript MSI	スクリプトで作成された機能セットに新しい機能を追加します。
FeatureAddUninstallCost	InstallScript	機能が追加のアンインストール操作を含むように指定します。これは、アンインストール中、進行状況バーを更新するときに必要です。
FeatureCompareSizeRequired	InstallScript、InstallScript MSI	選択された機能に対して充分なディスク空き容量があるかを判断します。
FeatureDialog	InstallScript、InstallScript MSI	エンド ユーザーが機能を選択してインストール先を指定できるダイアログを表示します。

テーブル 13 - 機能関数 (続き)

関数	プロジェクトの種類	説明
FeatureError	InstallScript、InstallScript MSI	機能関数が失敗したとき、追加のエラー情報を返します。
FeatureErrorInfo	InstallScript、InstallScript MSI	機能関数が失敗したとき、追加のエラー情報を返します。
FeatureFileEnum	InstallScript、InstallScript MSI	指定された機能に関連付けられたコンポーネントでファイルのリストをビルドします。
FeatureFileInfo	InstallScript、InstallScript MSI	関数内で参照されるファイルメディアに内のファイルについて情報を読み出します。
FeatureFilterLanguage	InstallScript、InstallScript MSI	言語に基づいたフィルタリングを有効または無効にします。
FeatureFilterOS	InstallScript、InstallScript MSI	オペレーティングシステムに基づいたフィルタリングを有効または無効にします。
FeatureGetCost	InstallScript、InstallScript MSI	指定した機能のためにターゲットドライブ上で必要な総容量をキロバイト (KB) で読み出します。 **FeatureGetCostEx** はこの関数に優先します。
FeatureGetCostEx	InstallScript、InstallScript MSI	nvCostHigh および nvCostLow パラメーターを使用して、指定された機能のコストをバイト単位で取得します。
FeatureGetData	InstallScript、InstallScript MSI	機能に関する情報を取得します。
FeatureGetItemSize	InstallScript、InstallScript MSI	指定された機能の大きさを決定します。
FeatureGetTotalCost	InstallScript、InstallScript MSI	指定された機能のインストールとアンインストールに必要な総容量を判断します。 **FeatureGetCostEx** はこの関数に優先します。
FeatureInitialize	InstallScript	InstallShield の以前のバージョンで作成されたスクリプトとの互換性の目的でのみサポートされています。InstallShield では、複数のファイル メディア ライブラリの使用を避けることが推奨されます。
FeatureIsItemSelected	InstallScript、InstallScript MSI	特定の機能をエンドユーザーが選択したかどうかを判断します。
FeatureListItems	InstallScript、InstallScript MSI	機能のリストをファイルメディアライブラリ、またはスクリプトで作成された機能セットに作成します。

テーブル 13 • 機能関数 (続き)

関数	プロジェクトの種類	説明
FeatureLoadTarget	InstallScript	有効なログ ファイルが存在する任意のインストールの初期化中に自動的に呼び出されます。
FeatureMoveData	InstallScript	ファイルメディアで選択した機能に関連付けたファイルを転送し、圧縮解除します。
FeaturePatch	InstallScript	差分メディアを使用するインストールでのみ呼び出されます。この関数は、**FeatureTransferData** や **FeatureMoveData** への次の呼び出しを引き起こし、**FeatureTransferData** が呼び出される時に既にインストールされている、(Data1.hdr、Data1.cab、および Layout.bin は除く) メンテナンス / アンインストール機能のファイルを含む、すべての機能を再インストールします。
FeatureReinstall	InstallScript、InstallScript MSI	セットアップを構成して、**FeatureTransferData** の次の呼び出しが、最後にセットアップを実行したときに指定されたファイル転送を実行するようにします。
FeatureRemoveAll	InstallScript、InstallScript MSI	セットアップを構成して、**FeatureTransferData** の次の呼び出が、セットアップをアンインストールするようにします。
FeatureRemoveAllInLogOnly	InstallScript	アップデート インストール中に呼び出され、セットアップ ログ ファイルに記録された通り、現在のメディアに無いが以前にインストールされているすべての機能を強制的に削除します。
FeatureRemoveAllInMedia	InstallScript	現在のメディアにある、以前にインストールされてたすべての機能を強制的に削除するために、メンテナンス インストール中に利用されます。この関数は通常、**SdWelcomeMaint** ダイアログでユーザーが [削除] を選択したときに呼び出されます。
FeatureRemoveAllInMediaAndLog	InstallScript	アップデートのインストール中に呼び出され、以前にインストールされた機能すべてを強制的に削除します。これには現在のメディアに含まれる機能と、現在のメディアには含まれていないが、セットアップ ログ ファイルに記録されている機能を含みます。
FeatureSaveTarget	InstallScript、InstallScript MSI	インストール プロジェクトが使用するすべてのテキスト置換の現在の値を取得して、それらをインストール ログ ファイルに格納します。
FeatureSelectItem	InstallScript、InstallScript MSI	機能を選択または選択解除します。

テーブル 13 - 機能関数 (続き)

関数	プロジェクトの種類	説明
FeatureSelectNew	InstallScript	すべての新しい機能の選択状態について、選択済みまたは未選択のどちらかに設定します。
FeatureSetData	InstallScript、InstallScript MSI	指定された機能のプロパティおよびデータを設定します。
FeatureSetTarget	InstallScript、InstallScript MSI	ファイル メディア ライブラリのユーザー定義変数を指定します。
FeatureSetupTypeEnum	InstallScript、InstallScript MSI	特定ファイルメディアライブラリに関連付けられセットアップの種類を列挙します。
FeatureSetupTypeGetData	InstallScript、InstallScript MSI	InstallShield インターフェイス で作成された指定されたセットアップに関連付けられたデータを読み出します。
FeatureSetupTypeSet	InstallScript、InstallScript MSI	指定されたセットアップの種類に関連付けられたすべての機能を選択します。
FeatureSpendCost	InstallScript	インストールの外部イベントによって使用された特定のコストについて、進行状況バーを更新します。
FeatureSpendUninstallCost	InstallScript	インストールの外部イベントによって使用された特定のアンインストール コストについて、進行状況バーを更新します。
FeatureStandardSetupTypeSet	InstallScript、InstallScript MSI	現在のセットアップ タイプを、nSetupType で指定される標準セットアップ タイプに設定します。
FeatureTotalSize	InstallScript、InstallScript MSI	指定された機能およびサブ機能のバイト単位での合計サイズを計算します。
FeatureTransferData	InstallScript MSI (InstallScript ユーザー インターフェイス (UI) のスタイルが、外部 UI ハンドラーとして InstallScript エンジンを使用する従来型のスタイルの場合)	***重要 -*** *この関数は、InstallScript UI スタイルが (埋め込み UI ハンドラーとして InstallScript エンジンを使用する) 新しいスタイルである InstallScript MSI プロジェクトには適用しません。詳しくは、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラとして使用する方法の違い」を参照してください。* イベント指向スクリプトでは、選択状態および現在インストールされているかどうかに基づいて、機能を適切にインストールまたはアンインストールします。

テーブル 13 - 機能関数 (続き)

関数	プロジェクトの種類	説明
FeatureUpdate	InstallScript	FeatureTransferData または FeatureMoveData への次の呼び出しで、インストール済みのすべての機能を保持するようにインストールを構成します。
FeatureValidate	InstallScript	ファイル メディア ライブラリまたは指定された機能のパスワードを検証します。
SdSetupType	InstallScript、InstallScript MSI	エンド ユーザーが、3 種類の標準セットアップ ([標準]、[最小]、または [カスタム]) の中から 1 つを選択できるダイアログを表示します。
SdSetupType2	InstallScript、InstallScript MSI	エンド ユーザーが、2 種類の標準セットアップ タイプ ([標準] と [カスタム]) のどちらかを選択できるダイアログを表示します。
SdSetupTypeEx	InstallScript MSI	[完全] および [カスタム] 以外のセットアップ タイプを指定する際に、エンドユーザーがセットアップ タイプを選択できるダイアログを表示します。

スクリプトを使用して作成した機能セットとファイルメディアライブラリ

セットアップスクリプトで FeatureAddItem 関数を呼び出して、実行時に機能を作成できます。これらのスクリプトを使用して作成した機能は、メモリ内にのみ存在し、ファイルメディアライブラリとは直接関係がありません。ファイルメディアライブラリに保存された情報と異なり、スクリプトを使用して作成した機能は、コンポーネントやセットアップの種類と直接関連を持たず、関連付けることもできません。

ただし、スクリプトを使用して作成した機能を、機能のようなオプションとして、エンドユーザーに表示することができます。エンドユーザーが機能のダイアログで機能を選択した後、スクリプトを使用して作成した機能の選択ステータスをテストし、結果に基づいて操作を実行することができます。たとえば、XCopyFile または VerUpdateFile でファイルをインストールし、ファイルメディアライブラリの機能を選択するか、ファイルを作成または編集することができます。

スクリプトを使用して作成した機能のビルド

スクリプトを使用して新しく機能をビルドするには、FeatureAddItem 関数を呼び出します。その後、ファイルメディアライブラリ内の機能と同様に、InstallScript 機能関数を使用して、スクリプトを使用して作成した機能のプロパティを設定し、プロパティにアクセスします (例外を以下に示します)。

スクリプトを使用して作成した機能セットを参照する

スクリプトを使用して作成された機能は、その全体を指して「スクリプトを使用して作成した機能セット」と呼ばれます。これは、ファイルメディアライブラリの機能と同様に、機能関数で処理されるためです。メディア名を機能関数に渡す場合は、すべての機能をセットとして扱います。

メモ・メディア名は FeatureAddItem の最初のパラメーターで作成します。同じ「スクリプトを使用して作成した機能セット」の一部として機能やサブ機能を作成する場合や、スクリプトで既存のスクリプトを使用して作成した機能を参照する場合は、この値を使用します。

機能関数を利用する

この 2 種類の機能は大きく異なるので、スクリプトを使用して作成した機能、またはファイルメディアライブラリ内の機能のどちらを処理するかによって、機能関数を異なる方法で呼び出します。

ファイルメディアライブラリとスクリプトを使用して作成した機能セット関数

これらの関数はファイルメディアライブラリまたはスクリプトを使用して作成した機能の両方で利用することができます。

- FeatureGetData
- FeatureSetData

ファイルメディアライブラリ関数

以下の関数は、ファイルメディアライブラリの機能のみに使用し、スクリプトを使用して作成した機能には使用できません。

- FeatureCompareSizeRequired
- FeatureFilterLanguage
- FeatureFilterOS
- FeatureSetTarget
- FeatureSetupTypeEnum
- FeatureSetupTypeGetData
- FeatureSetupTypeSet
- FeatureTransferData

スクリプトを使用して作成した機能セット関数

FeatureAddItem 関数は、スクリプトを使用して作成した機能セットでのみ利用することができます。

ファイルメディアライブラリ

ファイル メディア ライブラリには、製品のファイル、および InstallShield インターフェイス で入力したインストールのコンポーネント、機能およびセットアップの種類の設定情報がすべて含まれています。

ファイルメディアライブラリはプロジェクトのリリースを作成した際に生成されます。これは、InstallScript ヘッダー ファイル **Data1.hdr** で定義されます。ファイル メディア ライブラリには、MEDIA システム変数に含まれるデフォルト値であるメディア名もあります。

ファイルメディアライブラリの情報は、InstallScript 機能関数を使って設定およびアクセスできます。

メモ・一部の InstallScript 機能関数は、スクリプト作成機能で使用するために特別に予約されています。

ファイル関数とフォルダー関数

ファイルとフォルダー関数を利用すると、テキストファイル、バイナリファイル、そしてフォルダーを使った作業を包括的に進めることができます。関数の多くは、変数 TARGETDIR (InstallScript プロジェクトの場合)、INSTALLDIR (InstallScript MSI および基本の MSI プロジェクトの場合) および SRCDIR をパスとして使用し、ファイル名のみをパラメーターとして受け付けます。場所によってワイルドカード文字も利用することができます。

テーブル 14 ・ファイル関数とフォルダー関数

関数	説明
ChangeDirectory	指定されたディレクトリを現在のディレクトリにします。
CloseFile	開いているファイルを閉じます。
CopyFile	あるフォルダーから別のフォルダーへファイルをコピーします。
CreateDir	新規フォルダーを作成します
CreateFile	指定されたファイル名でファイルを作成します。
DeleteDir	フォルダーを削除します。
DeleteFile	ファイルを削除します。
ExistsDir	指定されたディレクトリが存在するかどうかを判断します。
ExistsDisk	指定されたディスクが存在するかどうかを判断します。
FileCompare	ファイルを別のファイルと比較します。
FileDeleteLine	テキストファイルの行を削除します。
FileGrep	テキストファイル内で指定されたテキストを検索します。
FileInsertLine	テキストファイルに行を挿入します。
FindAllDirs	指定されたフォルダーの下にあるサブフォルダーをすべて検出します。
FindAllFiles	指定されたフォルダーとサブフォルダーの中のファイル要件に一致するファイルをすべて検出します。
FindFile	指定されたフォルダー内でファイル要件に一致する最初のファイルを検出します。
GetFileInfo	ファイルの属性、日付、時間、サイズを取得します。
GetLine	開いたファイルのテキスト行を取得します。

テーブル 14 • ファイル関数とフォルダー関数（続き）

関数	説明
GetTempFileNameIS	Windows API **GetTempFileName** を呼び出して一時ファイルを作成し、その関連アクションを実行します。**GetTempFileName** とは異なり、存在しないとき、**GetTempFileNameIS** は szPathName で指定されたフォルダーを作成します。
Is	ファイルおよびパス チェック サービスの提供、数値演算コプロセッサの検索、管理者権限のテスト、ターゲット システム上に特定バージョンの .NET Framework または言語パックが存在するかどうかを判断、および Microsoft Windows がネットワーク上の共有コピーから実行しているかどうかなどを判断します。
OpenFile	既存のファイルを開きます。
OpenFileMode	OpenFile 関数を使ってファイルを開くときのモードを設定します。
ReadBytes	バイナリファイルから指定したバイト数を読み取ります。
RenameFile	ファイルの名前を変更します。
SeekBytes	ファイルポインターをバイナリファイルに置きます。
SetFileInfo	ファイルの属性、日付、並びに時間を設定します。
SetObjectPermissions	ファイル、フォルダー、またはレジストリ キーのアクセス許可を設定します。ファイル、フォルダー、またはレジストリ キーは、インストールの一部としてインストールでき、またターゲット システムに既に存在する場合もあります。
WriteBytes	指定したバイト数を、現在のファイルのポインターの場所にあるバイナリファイルに書き込みます。
WriteLine	文字列をテキストファイルに書き込みます。
XCopyFile	単数または複数のファイルをソースフォルダーからターゲットフォルダーにコピーします。サブフォルダーも指定できます。

関連する関数

テーブル 15 ・ 関連する関数

関数	説明
SelectDir	エンド ユーザーがフォルダーを選択できるダイアログを表示します。**SelectDir** は、フォルダーがない場合にそれを作成します。

FlexNet Connect の関数

FlexNet Connect (以前は Update Service と呼ばれていました) は、いくつかのビルト イン InstallScript 関数でサポート されています。

テーブル 16 ・ FlexNet Connect の関数

関数	プロジェクトの種類	説明
GetUpdateStatus	基本の MSI、InstallScript MSI、InstallScript	この関数は現在使用されていません。
GetUpdateStatusReboot	基本の MSI、InstallScript MSI、InstallScript	この関数は現在使用されていません。
SdFinishUpdate	基本の MSI、InstallScript MSI、InstallScript	インストールが完了したことを通知するダイアログを表示します。このダイアログには、アプリケーションのアップデートを確認するためのオプションが含まれます。 SdFinishUpdateEx はこの関数に優先します。 ***メモ・SdFinishUpdate** はアップデートの確認を行いません。アップデートの確認をするためには、InstallScript コードに FlexNet Connect API の呼び出しを追加します。詳細は、FlexNet Connect SDK ドキュメントを参照してください。*

テーブル 16 • FlexNet Connect の関数 (続き)

関数	プロジェクトの種類	説明
SdFinishUpdateEx	基本の MSI、InstallScript MSI、InstallScript	インストールが完了したことを通知するダイアログを表示します。このダイアログには、アプリケーションのアップデートを確認するためのオプションが含まれます。 ***メモ • SdFinishUpdateEx** はアップデートの確認を行いません。アップデートの確認をするためには、InstallScript コードに FlexNet Connect API の呼び出しを追加します。詳細は、FlexNet Connect SDK ドキュメントを参照してください。*
SdFinishUpdateReboot	基本の MSI、InstallScript MSI、InstallScript	インストールが完了したことを通知するダイアログを表示します。ダイアログは、エンド ユーザーに対してシステムの再起動オプションを提供し、アプリケーションのアップデートも確認します。 ***メモ • SdFinishUpdateReboot** はアップデートの確認を行いません。アップデートの確認をするためには、InstallScript コードに FlexNet Connect API の呼び出しを追加します。詳細は、FlexNet Connect SDK ドキュメントを参照してください。*
SetUpdateStatus	基本の MSI、InstallScript MSI、InstallScript	この関数は現在使用されていません。
SetUpdateStatusReboot	基本の MSI、InstallScript MSI、InstallScript	この関数は現在使用されていません。
UpdateServiceCheckForUpdates	InstallScript	この関数は現在使用されていません。 FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。
UpdateServiceCreateShortcut	InstallScript	この関数は現在使用されていません。 FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

テーブル 16 ・ FlexNet Connect の関数 (続き)

関数	プロジェクトの種類	説明
UpdateServiceEnableUpdateManagerInstall	InstallScript	この関数は現在使用されていません。 FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。
UpdateServiceGetAgentTarget	InstallScript	この関数は現在使用されていません。 FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。
UpdateServiceOnEnabledStateChange	InstallScript	この関数は現在使用されていません。 FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。
UpdateServiceRegisterProduct	InstallScript	この関数は現在使用されていません。 FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。
UpdateServiceRegisterProductEx	InstallScript	この関数は現在使用されていません。 FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。
UpdateServiceSetHost	InstallScript	この関数は現在使用されていません。 FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

テーブル 16 • FlexNet Connect の関数 (続き)

関数	プロジェクトの種類	説明
UpdateServiceSetLanguage	InstallScript	この関数は現在使用されていません。 FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

情報関数

以下の情報関数は、動作環境で利用可能なリソースに関するデータ (ディスク空き容量、メモリ、オペレーティング モード) を提供します。

テーブル 17 • 情報関数

関数	説明
GetDiskInfo	指定したディスク ドライブの情報を取得します。
GetDiskSpace	指定されたディスク (最大 2 ギガバイト) の使用可能な (未使用の) バイト数を返します。
GetDiskSpaceEx	ディスクの空いている容量をバイト、キロバイト、メガバイト、またはギガバイトで返します。
GetEnvVar	環境変数の現在値を返します。
GetExtendedErrInfo	**SetExtendedErrInfo** が設定したエラー情報を返します。
GetExtents	画面の寸法を返します。
GetMemFree	この関数は古い形式のため、使用できません。
GetSystemInfo	システム情報を取得します。
GetTrueTypeFontFileInfo	特定の TrueType フォント ファイルについての情報を返します。
GetValidDrivesList	ターゲットシステムの使用可能なすべてのドライブの一覧を返します。
GetWindowHandle	メイン インストールウィンドウのハンドルを返します。
Is	ファイルおよびパス チェック サービスの提供、数値演算コプロセッサの検索、管理者権限のテスト、ターゲット システム上に特定バージョンの .NET Framework または 言語パックが存在するかどうかを判断、および Microsoft Windows がネットワーク上の共有コピーから実行しているかどうかなどを判断します。
SetExtendedErrInfo	**GetExtendedErrInfo** を使って読み込むことが可能な、エラー情報を設定します。

テーブル 17 • 情報関数 (続き)

関数	説明
SetInstallationInfo	システム変数 IFX_COMPANY_NAME、IFX_PRODUCT_NAME、IFX_PRODUCT_VERSION、および IFX_PRODUCT_KEY の値を設定します。

初期化ファイル関数

初期化ファイル関数は初期化とプロファイルファイルから、または初期化とプロファイルファイルへ情報を取得、またコピーします。初期化ファイルは特別な ASCII ファイルで、キー名値のペアを含みます。キー名値ペアはアプリケーションのランタイムオプションを意味します。プライベート初期化ファイルとシステム初期化ファイルは、アクセスしてアップデートを行うことができます。次のリストは各初期化ファイル関数を簡単に説明します。

テーブル 18 • 初期化ファイル関数

関数	説明
AddProfString	非固有キーを .ini ファイルのセクションに追加します。
GetProfInt	.ini ファイルの整数を返します。
GetProfString	.ini ファイルの文字列を返します。
GetProfStringList	.ini ファイルのキー名や文字列値のリストを読み出します。
ReplaceProfString	プロファイル (.ini) ファイルの文字列を置き換えます。
WriteProfInt	整数値を持つ文字列を .ini ファイルに書き込みます。
WriteProfString	文字列を .ini ファイルに書き込みます。

関連する関数

テーブル 19 - 関連する関数

関数	説明
SdShowFileMods	ファイル変更の提案や手順についてのオプションを表示するダイアログを作成します。

リスト操作関数

関連情報のグループを保管するにはリストを使用します。InstallScript には文字列リストと数値リストの 2 種類のリストがあります。各リストの種類を処理する 2 組の関数があります。"Item" で終わるリスト関数は番号リストと一緒に使用します。"String" で終わるリスト関数は文字列リストと一緒に使用します。文字列リストで番号リスト関数を使用したり、番号リストで文字列リスト関数を使用することはできません。以下はセットアップスクリプトでリストを導入するための関数です。

テーブル 20 - リスト操作関数

関数	説明
ListAddItem	アイテムをリストに追加します。
ListAddString	文字列をリストへ追加します。
ListCount	文字列の番号や数値アイテムを指定されたリストに返します。
ListCreate	新しい文字列または番号リストを作成します。
ListCurrentItem	リスト内の現在のアイテムを返します。
ListCurrentString	リスト内の現在の文字列を返します。
ListDeleteItem	リスト内の現在のアイテムを削除します。
ListDeleteString	リスト内の現在の文字列を削除します。
ListDestroy	リストを破棄します。
ListFindItem	指定されたアイテムを数値リスト内の現在のアイテムにします。
ListFindKeyValueString	指定された値の文字列または数値リストを検索します。最初のリストで見つかった文字列の位置に対応する追加リストから値が返されます。
ListFindString	指定されたアイテムを文字列リスト内の現在のアイテムにします。
ListGetFirstItem	番号リストから 1 番目のアイテムを読み出します。
ListGetFirstString	文字列リストから 1 番目の文字列を読み出します。

テーブル 20 • リスト操作関数 (続き)

関数	説明
ListGetNextItem	番号リストから現在のアイテムの後のアイテム取得します。
ListGetNextString	文字列リストから現在のアイテムの後のアイテムを取得します。
ListReadFromFile	テキストファイルをリストへ読み込みます。
ListSetCurrentItem	番号リストの現在のアイテムを設定します。
ListSetCurrentString	文字列リストの現在のアイテムを設定します。
ListSetIndex	インデックスを使ってリストの現在のアイテムを設定します。
ListValid	指定されたリストが有効かどうかを示します。
ListValidType	指定されたリストが有効かどうか、また指定されたの種類であるかどうかを示します。
ListWriteToFile	文字列リストをファイルに書き込みます。
ListWriteToFileEx	nOptions パラメーターで提供した定数にしたがって、テキスト ファイルに文字列リストを Unicode または ANSI として書き込むか、または追加します。

ログファイル関数

プロジェクト *• この情報は、InstallScript プロジェクトに適用します。*

ログファイル関数はログライルのカスタムログ記録セクションから情報を取得し、またそこへ情報をコピーします。スクリプトへ、カスタム値を読み込んでこれらの値に基づいたアクションを実行するコードを追加しない限り、カスタムログファイルエントリはアプリケーションのメンテナンスまたはアンインストールには影響しません。このログファイル関数はログファイルのメンテナンス / アンインストールセクション (つまり、インストールされたファイルや作成されたレジストリ エントリといったデータをセットアップが自動的に書き込むセクションで、メンテナンスやアンインストール中にそこからデータを自動的に読み込みます) からデータを読み込んだり、データを書き込むことはできません。次のリストは各ログファイル関数を簡単に説明します。

テーブル 21 • ログファイル関数

関数	説明
LogReadCustomNumber	ログファイルのカスタムログ記録セクションから数値データを読み込みます。
LogReadCustomString	ログファイルのカスタムログ記録セクションから文字列データを読み込みます。

テーブル 21 - ログファイル関数 (続き)

関数	説明
LogWriteCustomNumber	ログファイルのカスタムログ記録セクションへ数値データを書き込みます。
LogWriteCustomString	ログファイルのカスタムログ記録セクションへ文字列データを書き込みます。

長いファイル名関数

次の関数は長いファイル名を扱うオペレーティングシステムが認識できるよう、短いファイル名から長いファイル名を作成し、短いファイル名を長いファイル名に変換し、長いファイル名の周りに二重引用符を配置します。

テーブル 22 - 長いファイル名関数

関数	説明
LongPathFromShortPath	ショートファイル名からロングファイル名を作成します。
LongPathToQuote	二重引用符をロングファイル名に挿入または削除します。
LongPathToShortPath	ロングファイル名からショートファイル名を作成します。

その他の関数

次の関数は低位ハードウェアインターフェイス、機能の作成と操作、そしてユーザー出力など様々な目的に利用できます。

テーブル 23 - その他の関数

関数	説明
Do	現在定義されている EXIT ハンドラーおよび HELP ハンドラーを実行します。
DoInstall	別の InstallShield インストーラーを起動します。
FormatMessage	大きな負のエラーコード用のテ文字列エラーメッセージを返します。
Handler	この関数は現在使用されていません。代わりに HandlerEx を使用してください。
HandlerEx	終了イベントおよびヘルプイベントに対応して、ブランチするラベルを指定します。
ISCompareServicePack	ターゲットの OS にインストールしたサービスパック番号を指定したサービスパック番号と比較します。
IsEmpty	VARIANT 型の変数が初期化されたかどうかを確認します。

テーブル 23 • その他の関数 (続き)

関数	説明
MessageBeep	標準警告音を鳴らします。
Resize	InstallScript 配列のサイズを変更します。
SendMessage	Windows メッセージを別のウィンドウやアプリケーションに送ります。
SetObjectPermissions	ファイル、フォルダー、またはレジストリ キーのアクセス許可を設定します。
SizeOf	InstallScript 配列のサイズを返します。
Sprintf	単数または複数の文字、数値または文字列値で構成されるフォーマットされた文字列を返します。
SprintfMsiLog	メッセージを Windows Installer ログファイルへ直接書き込みます。
StreamFileFromBinary	バイナリキーをファイルを一緒にストリームします。
System	コンピューターを再起動します。
VarInit	**VarSave** および **VarRestore** 関数で使用される内部リストを初期化、または再初期化します。この関数を呼び出すと、前回 **VarSave** の呼び出しで保存されているが、後に続く **VarRestore** 関数によってまだ使われていない情報が実質的にクリアされます。
VarRestore	**VarSave** への最後の呼び出しで保存されたシステム変数 SRCDIR、TARGETDIR (InstallScript プロジェクトの場合) および INSTALLDIR (基本の MSI および InstallScript プロジェクトの場合) の値を復元します。
VarSave	システム変数 SRCDIR、TARGETDIR (InstallScript プロジェクトの場合)、および INSTALLDIR (基本の MSI および InstallScript プロジェクトの場合) の現在の値を保存します。

オブジェクト関数

オブジェクト関数を使用して、オブジェクトの初期化、およびオブジェクト ステータス情報の取得と設定をより効率的に行うことができます。

テーブル 24 • オブジェクト関数

関数	説明
CoCreateObject	COM オブジェクトを初期化し、SET キーワードを使用して、型 OBJECT の変数に割り当てることができる参照を返します。

テーブル 24 • オブジェクト関数 (続き)

関数	説明
CoCreateObjectDotNet	**CoCreateObjectDotNet** 関数は現在使用されていません。この関数の呼び出しは、szAppDomain パラメーターにヌル文字列 ("") を使って **DotNetCoCreateObject** 関数を呼び出すのと同じです。 詳細については、「DotNetCoCreateObject」を参照してください。
CoGetObject	指定した COM オブジェクトへリファレンスを返します (Visual Basic の GetObject 関数と同様)。このリファレンスは、設定されたキーワード を利用して OBJECT 型の変数へ割り当てることができます。
DotNetCoCreateObject	アセンブリを COM 相互運用性を登録せずに .NET アセンブリの関数を呼び出します。この関数は、**CoCreateObjectDotNet** 関数と違い、.NET アセンブリがロードされ実行される .NET アプリケーションのドメインを指定することができます。
DotNetUnloadAppDomain	指定された .NET アプリケーション ドメインをアンロードし、現在ロードされているアセンブリをすべて指定されたアプリケーション ドメインにリリースします。
GetObject	オブジェクトを初期化し、設定されたキーワードを使用して、OBJECT 型の変数に割り当てることができる参照を返します。
GetObjectByIndex	nIndex が指定したインストールまたはオブジェクトのサブオブジェクトを検索し、設定されたキーワードを使用して OBJECT 型の変数に割り当てられるリファレンスを返します。
GetObjectCount	オブジェクトまたはインストールに含まれるサブオブジェクトの数を返します。
GetStatus	オブジェクトの現在のステータス (Status.Number の現在の値) を取得します。
IsObject	OBJECT 型の変数に有効なオブジェクトへのリファレンスを割り当てられたかどうかを、CreateObject 関数または GetObject 関数を使用して確認します。
SetStatus	オブジェクト スクリプトで呼び出され、オブジェクトの Status.Number や Status.Description プロパティを設定します。
SetStatusEx	オブジェクト スクリプトで呼び出され、オブジェクトのステータス プロパティを設定します。

パスバッファー関数

パスバッファー関数は、検索パスを含む文字列を使った作業を支援します。パス文字列関数は、パスバッファーとして知られる一意の一時文字列変数で機能します。パスバッファーは InstallShield の内部で定義されます。すべてのパス文字列関数はパスバッファーの内容に従って動作します。

パス関数はパス文字列のビルドと操作を支援します。パス文字列を作成したあと、適切なファイルへ保存することができます。

テーブル 25 - パスバッファー関数

関数	説明
PathAdd	パスをパスバッファーの検索パスに追加します。
PathDelete	パスバッファーからディレクトリを削除します。
PathFind	パスバッファー内の特定パスや指定された名前を含むすべてのパスを検出します。
PathGet	パスバッファーの現在の値を取得します。
PathMove	パスバッファーをソートします。
PathSet	パスバッファーに値を割り当てます。

レジストリ関数

次の関数を利用してレジストリへアクセスし、レジストリキーの読み取り、作成、そして削除を行って、アンインストール用のレジストリ関連パラメーターを設定することができます。

テーブル 26 - レジストリ関数

関数	説明
CreateInstallationInfo	インストールするプログラムのアプリケーション情報キーとアプリケーションごとのパスキーを作成します。
CreateRegistrySet	[リソース] ペインの [レジストリエントリ] フォルダーで指定したレジストリエントリの 1 つまたはすべてのセットを作成します。
DeinstallSetReference	この関数は現在使用されていません。 ファイルがアンインストール中にロックされているかどうかを確認するには、nIsFlag パラメーターに FILE_LOCKED 定数を使って Is 関数を呼び出す、および適切に応答するスクリプト コードを作成します。
DeinstallStart	この関数は現在使用されていません。
InstallationInfo	この関数は現在使用されていません。代わりに、CreateInstallationInfo 関数を使用してください。
MaintenanceStart	<PRODUCT_GUID> レジストリキーを作成してアンインストール機能を有効にします。
RegDBConnectRegistry	リモートレジストリへの接続を開きます。

テーブル 26 • レジストリ関数 (続き)

関数	説明
RegDBCopyKeys	szSourceKe が指定するキーの下にあるレジストリ キーおよび値を szTargetKey が指定するキーへコピーします。
RegDBCopyValues	szSourceKe が指定するキーの下にあるレジストリ値を szTargetKey が指定するキーへコピーします。
RegDBCreateKeyEx	レジストリでキーを作成します。クラスオブジェクトをレジストリキーと関連付けることもできます (詳しい知識のあるユーザーのみ)。
RegDBDeleteItem	nItem の値に従ってアプリケーションごとのパスキーまたはアプリケーション アンインストール キーの下にある値を削除します。
RegDBDeleteKey	レジストリから指定されたキーを削除します。
RegDBDeleteValue	指定されたレジストリキーから値を削除します。
RegDBDisConnectRegistry	リモートレジストリへの接続を終了します。
RegDBGetAppInfo	アプリケーション情報キーの下にある値を取得します。
RegDBGetDefaultRoot	レジストリ関連の一般関数が利用するルート キーを返します。
RegDBGetItem	アプリケーションごとのパスキーまたはアプリケーションのアンインストールキーにある値を読み出します。
RegDBGetKeyValueEx	レジストリのキーから値を取得します。
RegDBGetUninstCmdLine	szUninstallKey が指定したアンインストール用の登録済みコマンドラインを取得し、svUninstCmdLine のコマンドラインを返します。
RegDBKeyExist	レジストリキーの存在を確認します。
RegDBQueryKey	キーをクエリして、サブキーおよび値名を取得します。
RegDBQueryKeyCount	サブキーの数または指定されたキーの下にある値を返します。
RegDBQueryStringMultiStringCount	指定されたキーの下にある特定の値によって指定される複数文字列値に含まれる文字列の数を返します。
RegDBSetAppInfo	アプリケーション情報キーの下に値を設定します。
RegDBSetDefaultRoot	ルートキーを設定します。
RegDBSetItem	アプリケーションごとのパスキーまたはアプリケーションのアンインストールキーの下にある値を割り当てます。

テーブル 26 - レジストリ関数 (続き)

関数	説明
RegDBSetKeyValueEx	レジストリエントリを設定します。
SetInstallationInfo	CreateInstallationInfo が使用する会社や製品情報を指定します。
SetObjectPermissions	ファイル、フォルダー、またはレジストリ キーのアクセス許可を設定します。

サービス関数

次の関数を利用してレジストリへアクセスし、レジストリキーの読み取り、作成、そして削除を行って、アンインストール用のレジストリ関連パラメーターを設定することができます。

テーブル 27 - サービス関数

関数	説明
ServiceAddService	ターゲット システム上で登録されているサービスのリストにサービスを追加します。
ServiceExistsService	指定されたサービスが登録されているかどうかを判別します。
ServiceGetServiceState	指定されたサービスの状態を取得します。
ServiceInitParams	SERVICE_IS_PARAMS システム変数のメンバーを次のデフォルト値に初期化します。この関数はセットアップ初期化中に自動的に呼び出されます。
ServiceRemoveService	サービスをターゲットシステムから削除します。
ServiceStartService	サービスを開始します。関数が呼び出されたときにサービスが実行中の場合、インストールは停止してサービスを再開始します。
ServiceStopService	サービスを停止します。

共有およびロック ファイル関数

共有ファイルというのは、複数のアプリケーションで使用できる DLL、.vbx、ドライバーなどのファイルです。InstallShield はアンインストール中に共有ファイルが削除されないように保護します。

SHAREDFILE オプションを使った関数は、すべてのファイルを共有ファイルとして処理するため、関与するすべてのファイルのレジストリ参照カウントを増やします。ファイルがターゲットディレクトリに存在して、0 より大きい参照カウンターを持つ場合、InstallShield はレジストリ参照カウントを 1 増やします。共有ファイルがターゲットディレクトリに存在せず、参照カウンターがない場合、InstallShield はカウンターを作成して 1 に設定します。共有ファイルがターゲット ディレクトリに存在するが、参照カウンターがない場合、InstallShield はカウンターを作成して、アンインストール中に誤って削除されないように、これを 2 に初期化します。

共有ファイルはロックされている場合更新してはいけません。InstallShield ファイル転送関数の一部は、Windows またはシステムが再起動したときにファイル転送中にロックされている .dll ファイルと .exe ファイルが記録および更新できるように、SHAREDFILE オプションを使用します。

InstallShield では、アプリケーションまたはシステムによってファイルが使用されている場合、ファイルがロックされていると見なします。ロックされたファイルは必ずしも共有ファイルではありません。

次の関数が共有ファイルまたはロックされたファイルを処理します。

テーブル 28 - 共有およびロックファイル関数

関数	説明
Is	ファイルおよびパス チェック サービスの提供、数値演算コプロセッサの検索、管理者権限のテスト、ターゲット システム上に特定バージョンの .NET Framework または 言語パックが存在するかどうかを判断、および Microsoft Windows がネットワーク上の共有コピーから実行しているかどうかなどを判断します。
RebootDialog	エンドユーザーが Windows の再起動やコンピューターの再起動を選択できるダイアログを表示します。
SdFinishReboot	インストールが完了したことを知らせるダイアログを表示して、エンドユーザーが Windows の再起動やコンピューターの再起動を選択できるようにします。
SetObjectPermissions	ファイル、フォルダー、またはレジストリ キーのアクセス許可を設定します。
VerUpdateFile	バージョンリソース情報を使用してファイルをアップデートします。ロックされた .dll ファイルと .exe ファイルを更新して、関与するすべてのファイルのレジストリ参照カウンターを増分します。
XCopyFile	ファイルとサブディレクトリをソースディレクトリからターゲットディレクトリにコピーします。ロックされた共有ファイルの処理を結合します。XCopyFile ではすべてのファイルが共有ファイルとみなされ、Windows またはシステムが再起動されたときに、更新用のロックされた .dll および .exe ファイルが記録されます。

シェル関数

シェル関数は、プログラムフォルダーの作成、既存プログラムフォルダーの削除、そして既存プログラムフォルダーへアイテムの追加を行います。セットアップの終わりに、ユーザーがソフトウェアへ素早くアクセスできるよう、アプリケーションを適切なプログラムフォルダーへ追加して下さい。次の関数もまた様々なアイコンオプションをサポートします。

テーブル 29 ・シェル関数

関数	説明
AddFolderIcon	ショートカットまたはプログラム フォルダーを [スタート] メニュー、[プログラム] メニュー、またはデスクトップに追加します。 **SHELL_OBJECT_FOLDER** はこの関数に優先します。
CreateProgramFolder	プログラム フォルダーを作成します。 **SHELL_OBJECT_FOLDER** はこの関数に優先します。
CreateShortcut	ショートカットまたはプログラム フォルダーを [スタート] メニュー、[プログラム] メニュー、またはデスクトップに追加します。 オプションで、ショートカットの Windows シェル プロパティを設定して、ショートカットを [スタート] メニューにピン留めできる機能を無効化するなどの動作を構成します。
CreateShortcutFolder	プログラム フォルダーを作成します。
DeleteFolderIcon	ショートカットをフォルダーから削除します。 **DeleteShortcut** はこの関数に優先します。
DeleteProgramFolder	ショートカット フォルダー (つまり [スタート] メニューの [プログラム] フォルダーのサブフォルダー) と、すべてのショートカットやショートカット フォルダーのすべてのサブフォルダー、および内容を含むそのフォルダーの内容を削除します。 **DeleteShortcutFolder** はこの関数に優先します。
DeleteShortcut	ショートカットをフォルダーから削除します。
DeleteShortcutFolder	ショートカット フォルダー (つまり [スタート] メニューの [プログラム] フォルダーのサブフォルダー) と、すべてのショートカットやショートカット フォルダーのすべてのサブフォルダー、および内容を含むそのフォルダーの内容を削除します。
GetFolderNameList	指定されたフォルダー中のすべてのサブフォルダー名とショートカットを取得します。
GetShortcutInfo	指定したショートカットまたはサブフォルダーの情報を返します。

テーブル 29・シェル関数（続き）

関数	説明
ProgDefGroupType	システム変数 ALLUSERS の値を設定します。
QueryProgItem	指定したショートカットまたはサブフォルダーの情報を返します。 **GetShortcutInfo** はこの関数に優先します。
QueryShellMgr	現在のシェルマネージャーの名前を返します。
ReplaceFolderIcon	指定されたフォルダーのショートカットを置き換えます。 **ReplaceShortcut** はこの関数に優先します。
ReplaceShortcut	指定されたフォルダーのショートカットを置き換えます。
SelectFolder	エンド ユーザーがプログラム フォルダーのリストからフォルダーを選択できるダイアログを表示します。
SetShortcutProperty	インストール実行時に Windows シェルによる設定が必要な 1 つ以上のショートカット プロパティ設定します。
ShowProgramFolder	指定したプログラムフォルダーを表示します。

レジストリ関連の特殊関数

レジストリ関連の特殊関数は、スクリプトの作成者が最低必要なレジストリキー及び値の設定をより簡単に行えるようにするために作られました。レジストリ関連の特殊関数は、以下に示すアプリケーションごとのパスキー、アプリケーションのアンインストールキー、またはアプリケーション情報キーを使ってのみ機能します。詳細については、それぞれの関数の説明を参照してください。

アプリケーションごとのパスキー

<root key>¥Software¥Microsoft¥Windows¥CurrentVersion¥App Paths¥<per application paths key>

このキーは、アプリケーションごとのパスキー、または App Paths キーとして参照されます。アプリケーションごとのパスキーはパス情報を保管し、これによって Windows はアプリケーションの実行可能ファイルを見つけることができます。ALLUSERS システム変数が FALSE の場合、または ProgDefGroupType(PERSONAL) を呼び出した場合のルートキーは HKEY_CURRENT_USER で、その他の場合は HKEY_LOCAL_MACHINE です。

テーブル 30 ・アプリケーションごとのパスキー

関数	説明
CreateInstallationInfo	アプリケーション実行可能ファイルの名前を使用して、アプリケーションごとのパスキーの作成に備えます。キーは、RegDBSetItem が呼び出されるまで作成されません (以下を参照)。 イベント指向のスクリプトを使用すると、デフォルトの OnMoveData イベント ハンドラーコードによって CreateInstallationInfo 関数が呼び出されます。
RegDBDeleteItem	アプリケーションごとにパス キーと、このキーの下にある [Path] または [DefaultPath] の値を削除します。
RegDBGetItem	アプリケーションごとのパスキーの下にある [Path] の値、または [DefaultPath] の値を取出します。
RegDBSetItem	アプリケーションごとにパスキーを作成し、[Path] の値または [DefaultPath] の値をこのキーの下に設定します。

アプリケーション アンインストール キー

< ルート キー >¥Software¥Microsoft¥Windows¥CurrentVersion¥Uninstall¥<INSTANCE_GUID>

このキーは、アプリケーションのアンインストールキーとして参照されます。アプリケーションのアンインストールキーは、アンインストール機能を可能にする情報を保管します。ALLUSERS システム変数が FALSE の場合、または ProgDefGroupType(PERSONAL) を呼び出した場合のルートキーは HKEY_CURRENT_USER で、その他の場合は HKEY_LOCAL_MACHINE です。

テーブル 31 ・アプリケーション アンインストール キー

関数	説明
MaintenanceStart	アプリケーションのアンインストールキーを作成し、[UninstallString]、[DisplayName] ([プログラムの追加と削除] で表示される名前)、および [LogFile] 値をこのキーの下に設定します。 (イベントベースのスクリプトでは、MaintenanceStart がデフォルト OnMoveData イベント ハンドラーコードで呼び出されます。
RegDBDeleteItem	アプリケーションのアンインストール キーの下にある [DisplayName] ([プログラムの追加と削除] に表示される名前) の値を削除します。
RegDBGetItem	アプリケーションのアンインストールキーの下の [DisplayName] の値を取出します。

テーブル 31 • アプリケーション アンインストール キー（続き）

関数	説明
RegDBSetItem	[DisplayName]（[プログラムの追加と削除] に表示される名前）の値をアプリケーションのアンインストールキーの下に設定します。 この値は、MaintenanceStart（イベントベースのスクリプトを使用すると、MaintenanceStart はデフォルトの OnMoveData イベント ハンドラーコード内で呼び出されます）によっても設定されます。

アプリケーション情報キー

<ルート キー>¥Software¥<会社キー>¥<製品キー>¥<バージョン キー>

このキーは、アプリケーション情報キーとして参照されます。インストールは、アプリケーション情報キーをインストールする各アプリケーションに作成する必要があります。アプリケーション情報キーは、アプリケーションに関する情報を保管します。ALLUSERS システム変数が FALSE の場合、または ProgDefGroupType(PERSONAL) を呼び出した場合のルートキーは HKEY_CURRENT_USER で、その他の場合は HKEY_LOCAL_MACHINE です。

テーブル 32 • アプリケーション情報キー

関数	説明
CreateInstallationInfo	会社名、製品名、製品バージョン番号を使ってアプリケーション情報キーを作成します。RegDBSetAppInfo 関数（以下を参照）を呼び出すまで、このキーの下に値は設定されません。 イベント指向のスクリプトを使用すると、デフォルトの OnMoveData イベント ハンドラーコードによって CreateInstallationInfo 関数が呼び出されます。
RegDBGetAppInfo	アプリケーション情報キーの下にある値を取得します。
RegDBSetAppInfo	アプリケーション情報キーの下に値を設定します。

SQL 関数

SQL 関数を使用して、カタログへの接続、SQL 関連のダイアログの作成、および SQL ランタイム エラーの取得などのタスクの実行することができます。

ヒント・SQL サポートと SQL 関連の InstallScript 関数については、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。

テーブル 33・SQL 関数

関数	プロジェクトの種類	説明
SQLBrowse	InstallScript、InstallScript MSI	ユーザーがネットワーク上で提供されているすべての SQL Server のリストを表示することができるダイアログを作成します。 **SQLBrowse2** はこの関数に優先します。
SQLBrowse2	InstallScript、InstallScript MSI	エンド ユーザーが、接続に指定されたデータベース テクノロジ用にネットワーク上で提供されているすべてのデータベース サーバーのリストを表示できるようにするダイアログを作成します。
SQLDatabaseBrowse	InstallScript、InstallScript MSI	エンド ユーザーが、指定されたデータベース サーバー上で使用できるすべてのデータベース カタログを一覧表示することができるダイアログを作成します。この関数は **SQLRTGetDatabases** を呼び出して、InstallScript プロジェクトの場合は **SQLRT.dll** を、InstallScript MSI プロジェクトの場合は **ISSQLSRV.dll** を使用します。
SQLRTComponentInstall	InstallScript	スクリプトがインストール時に実行されるようにスケジュールされている場合、指定されたコンポーネントに関連付けられている SQL スクリプトを実行します。
SQLRTComponentUninstall	InstallScript	スクリプトがアンインストールル時に実行されるようにスケジュールされている場合、指定されたコンポーネントに関連付けられている SQL スクリプトを実行します。
SQLRTConnect	InstallScript	指定された認証情報を使用して接続を確立します。 **SQLRTConnect2** はこの関数に優先します。
SQLRTConnect2	InstallScript	接続を確立します。インストール中にスクリプトを実行するために接続が利用される場合、この関数をファイル転送の前に呼び出す必要があります。**SQLRTConnect2** は、接続の確立に失敗した場合、データベース サーバー名を戻します。この関数は **SQLRT.dll** を使用するため、**SQLRTInitialize2** が呼び出されたあとでのみ、呼び出すことができます。
SQLRTConnectDB	InstallScript	特定のカタログへの接続を確立します。

テーブル 33 · SQL 関数 (続き)

関数	プロジェクトの種類	説明
SQLRTDoRollbackAll	InstallScript	ロールバック中に実行されるようにスケジュールされた SQL スクリプトをすべて実行します。
SQLRTGetBatchList	InstallScript	SQLGetBatchList 関数は、バッチ モードが有効になっているとき実行が必要な SQL スクリプトに関連付けられているコンポーネントの一覧を返します。 バッチ モードに関する詳細については、「接続に関連付けられている複数 SQL スクリプトの実行順序を指定する」をご覧ください。
SQLRTGetBatchMode	InstallScript	バッチ モードが有効か無効かを返します。
SQLRTGetBrowseOption	InstallScript	SQL Server の参照コンボ ボックスとリスト ボックス コントロールの参照オプションの現在の値を返します。これらでは、ローカル サーバー、リモート サーバー、サーバー エイリアス、およびこれらのサーバーの組み合わせを表示できます。
SQLRTGetComponentScriptError	InstallScript	コンポーネントに関連付けられている SQL スクリプトの実行中に発生した最後のエラーを取得します。 **SQLRTGetComponentScriptError2** はこの関数に優先します。
SQLRTGetComponentScriptError2	InstallScript	コンポーネントに関連付けられている SQL スクリプトの実行中に発生した最後のエラーを取得します。 この関数では、**SQLRTGetComponentScriptError** 関数で使用できないパラメーター がいくつか使用できます (szScriptName、szTechnology、szServer および szDB)。
SQLRTGetConnectionAuthentication	InstallScript、InstallScript MSI	デフォルトの SQL Server 接続認証タイプを取得します。
SQLRTGetConnectionInfo	InstallScript、InstallScript MSI	接続情報 (デフォルト サーバー、データベース、デフォルト ユーザー名、デフォルト パスワード) が含まれた文字列を取得します。
SQLRTGetConnections	InstallScript、InstallScript MSI	設定ファイルに存在する接続の文字列リストを取得します。

テーブル 33 • SQL 関数 (続き)

関数	プロジェクトの種類	説明
SQLRTGetDatabases	InstallScript、InstallScript MSI	指定されたデータベース サーバーで提供されているデータベース カタログのリストを返します。
SQLRTGetErrorMessage	InstallScript	接続を開いているときに、SQL ランタイムで最後に発生したエラーについての説明を返します。
SQLRTGetLastError	InstallScript	SQL ランタイムで最後に発生したエラーのテキストを返します。 **SQLRTGetLastError2** はこの関数に優先します。
SQLRTGetLastError2	InstallScript	SQL ランタイムで最後に発生したエラーについて詳細な情報を返し、適切な SQL エラーメッセージをロードします。
SQLRTGetScriptErrorMessage	InstallScript	SQL スクリプトが実行しているときに、SQL ランタイムで最後に発生したエラーについての説明を返します。
SQLRTGetServers	InstallScript、InstallScript MSI	インストールに含まれているすべてのデータベース テクノロジについて、ネットワークで提供されているデータベース サーバーのリストを返します。 **SQLRTGetServers2** はこの関数に優先します。
SQLRTGetServers2	InstallScript、InstallScript MSI	接続に指定されたデータベース テクノロジ用のデータベース サーバーのリストを返します。szConnection が空のとき、この関数は **SQLRTGetServers** のように動作します。
SQLRTInitialize	InstallScript	**SQLRT.dll** をロードし、設定ファイルを使用してこれを初期化します。SQLRT の中で呼び出される最初の関数である必要があります。 **SQLRTInitialize2** はこの関数に優先します。
SQLRTInitialize2	InstallScript、InstallScript MSI	InstallScript プロジェクトでは **SQLRT.dll** ファイルをロードし、InstallScript MSI プロジェクトでは **ISSQLSRV.dll** ファイルをロードします。また、この関数は設定ファイルを使用して、.dll ファイルを初期化します。 この関数は、SQLRT または ISSQLSRV で呼び出される最初の関数として指定する必要があります。

テーブル 33 · SQL 関数 (続き)

関数	プロジェクトの種類	説明
SQLRTPutConnectionAuthentication	InstallScript、InstallScript MSI	デフォルトの SQL Server 接続認証タイプを設定します。
SQLRTPutConnectionInfo	InstallScript、InstallScript MSI	接続情報 (デフォルト サーバー、デフォルト ユーザー名、デフォルト パスワード) を設定して、今後使用できるようにします。これは、[戻る] ボタンの使用などユーザーが以前に何を入力したかを思い出す必要がある場合に便利です。 **SQLRTPutConnectionInfo2** はこの関数に優先します。
SQLRTPutConnectionInfo2	InstallScript、InstallScript MSI	接続情報 (デフォルト サーバー、デフォルト データベース カタログ、デフォルト ユーザー名、およびデフォルト パスワード) を設定して、将来使用できるようにします。これは、[戻る] ボタンのように、エンドユーザーが以前に入力した情報を再び再現する必要がある場合に便利です。
SQLRTServerValidate	InstallScript MSI	インストールで指定された接続をテストします。
SQLRTSetBrowseOption	InstallScript	SQL Server の参照コンボ ボックスとリスト ボックス コントロールで、ローカル サーバー、リモート サーバー、サーバーのエイリアス、またはこれらのサーバーの組み合わせを表示するかどうかを指定できます。
SQLRTTestConnection	InstallScript MSI	指定された認証情報を使ってインストールで指定された接続をすべてテストします。 **SQLRTTestConnection2** はこの関数に優先します。
SQLRTTestConnection2	InstallScript MSI	接続を確立します。
SQLServerLogin	InstallScript、InstallScript MSI	SQL ログイン情報を指定するスクリプトで使われるダイアログを作成します。情報には、ログイン ID とパスワードが含まれます。
SQLServerSelect	InstallScript、InstallScript MSI	ターゲットにするサーバーを指定するダイアログを作成します。

テーブル 33 · SQL 関数 (続き)

関数	プロジェクトの種類	説明
SQLServerSelectLogin	InstallScript、InstallScript MSI	ターゲットされたエンド ユーザーが、現在の接続に使用する SQL Server および使用するログインの資格情報を指定できるログイン ダイアログを作成します。ダイアログでは、DSN を通してアクセスされた SQL Server のリストを含むコンボ ボックスが表示されます。エンドユーザーは、このコンボ ボックスでサーバー名を入力することもできますし、コンボ ボックスの隣にある [参照] ボタンをクリックすることもできます。このボタンをクリックすると、インターネットで提供されている SQL Servers がすべて表示されます。 **SQLServerSelectLogin2** はこの関数に優先します。
SQLServerSelectLogin2	InstallScript、InstallScript MSI	デフォルト スクリプトで使用されるログイン ダイアログを作成します。このダイアログで、ターゲットされたエンド ユーザーは、現在の接続に使用する SQL Server と使用するログインの認証情報を指定できます。ダイアログでは、DSN を通してアクセスされた SQL Server のリストを含むコンボ ボックスが表示されます。エンドユーザーは、このコンボ ボックスでサーバー名を入力することもできますし、[サーバー名] コンボ ボックスの隣にある [参照] ボタンをクリックすることもできます。このボタンをクリックすると、インターネットで提供されている SQL Servers がすべて表示されます。 オプションで、この関数は、接続情報に関連付けられた接続名も表示します。またオプションで、エンド ユーザーは現在の接続にどのデータベース カタログを使用するのかを指定することができます。

テーブル 33 ・ SQL 関数 (続き)

関数	プロジェクトの種類	説明
SQLServerSelectLoginEx	InstallScript、InstallScript MSI	デフォルト スクリプトで使用されるログイン ダイアログを作成します。このダイアログで、ターゲットされたエンド ユーザーは、現在の接続に使用する SQL Server と使用するログインの認証情報を指定できます。ダイアログでは、DSN を通してアクセスされた SQL Server のリストを含むコンボ ボックスが表示されます。エンドユーザーは、このコンボ ボックスでサーバー名を入力することもできますし、[サーバー名] コンボ ボックスの隣にある [参照] ボタンをクリックすることもできます。このボタンをクリックすると、インターネットで提供されている SQL Servers がすべて表示されます。 この関数は、接続情報に関連付けられた接続名も表示します。 **SQLServerSelectLogin2** はこの関数に優先します。

文字列関数

文字列関数は、文字列変数とリテラルを操作する能力を提供します。文字列関数は C 言語標準関数と同様に動作します。戻り値もまた C 言語の規則に従います。

テーブル 34 ・ 文字列関数

関数	説明
CopyBytes	ある文字列から別の文字列へ、指定したバイト数をコピーします。
GetCArrayFromISArray	指定された配列の実際のデータをポイントするポインター配列へのポインターを返します。この関数は、追加メモリを割り当てませんが、既存の配列にあるデータにポインターを返します。
GetCHARArrayFromISStringArray	指定された配列に含まれる幅広い文字列に対応する ANSI 文字列へのポインター配列へポインターを返します。
GetDir	パスや完全修飾ファイル名からインストール先ドライブを削除します。
GetDisk	パスや完全修飾ファイル名からディスクドライブのインストール先を取得します。
NumToStr	数値を文字列に変換します。
ParsePath	パスからドライブ、パス、ファイル名、または拡張子を取得します。

テーブル 34 • 文字列関数 (続き)

関数	説明
StrAddLastSlash	パス指定の末尾に円記号がない場合、それを追加します。
StrCompare	ある文字列を別の文字列と比較します。
StrConvertSizeUnit	指定された InstallScript サイズ単位定数の適切な表示文字列を返します。
StrFind	文字列を、別の文字列から探します。
StrFindEx	パラメーター szFindMe で渡された文字列がパラメーター szString で渡された文字列内で検出されたかどうかを判断します。関数は nStart が指定した場所で検索を開始します。
StrGetTokens	指定した区切り文字に基づいて文字列からトークンを取得します。
StrLength	**StrLengthChars** と同様に、指定の文字列変数で最初のヌル文字までの文字数 (つまり、UTF-16 エンコード文字列のコード ユニット数) を返します。
StrLengthChars	**StrLength** と同様に、指定の文字列変数で最初のヌル文字までの文字数 (つまり、UTF-16 エンコード文字列のコード ユニット数) を返します。
StrPutTokens	指定された文字列リストからリスト アイテムを抽出して、それを svString で指定された文字列に配置します。
StrRemoveLastSlash	パス文字列の最後の円記号を削除します。
StrReplace	nStart が指定した場所から svResult を検索し、見つかった szFind のインスタンスすべてを szReplace で置換します。
StrSub	文字列のサブ文字列を返します。
STRTOCHAR	文字列の最初の文字を CHAR 型のデータとして返します。
StrToLower	文字列のすべての英文字を小文字に変換します。
StrToNum	文字列を数値に変換します。
StrToNumHex	文字列を数値に、例えば 0x1A を 26 に変換します。
StrToUpper	文字列のすべての英文字を大文字に変換します。
StrTrim	文字列から先頭と行末の空白およびタブを削除します。

スイート / アドバンスト UI およびアドバンスト UI の対話関数

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

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

***メモ・**これらすべての関数は、アドバンスト UI またはスイート / アドバンスト UI インストールに InstallScript パッケージとして含める可能性がある InstallScript インストールに使用できます。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート / アドバンスト UI プロジェクトに追加する」をご覧ください。*

*さらに、**FeatureConfigureFeaturesFromSuite** および **SuiteReportError** を除く、これらすべての関数は、スイート / アドバンスト UI インストールに含まれている InstallScript アクションで使用できます。詳細については、「スイート / アドバンスト UI インストールに含まれる InstallScript コードを実行するアクションでの作業について」を参照してください。*

***FeatureConfigureFeaturesFromSuite** は、次のシナリオでも使用できますが、その他のアドバンスト UI またはスイート / アドバンスト UI の関数では、エラーが返されます。*

- *関数が、直接起動された (つまり、アドバンスト UI またはスイート / アドバンスト UI インストールから起動されなかった) InstallScript インストールで呼び出されたとき。*
- *関数が、アドバンスト UI またはスイート / アドバンスト UI インストールに実行可能パッケージとして含められている InstallScript インストールで呼び出されたとき。*

InstallScript には、InstallScript パッケージを含むアドバンスト UI またはスイート / アドバンスト UI インストールと対話するための次の関数が含まれています。これらの関数 (**FeatureConfigureFeaturesFromSuite** を除く) を使って、InstallScript アクションを通して実行中のスイート / アドバンスト UI インストールと対話することができます。

テーブル 35 ・ スイート / アドバンスト UI およびアドバンスト UI の対話関数

関数	説明
FeatureConfigureFeaturesFromSuite	アドバンスト UI またはスイート / アドバンスト UI プロパティ **ISFeatureInstall** と **ISFeatureRemove** の値に基づいて、アドバンスト UI またはスイート / アドバンスト UI インストールで実行中の InstallScript パッケージに対して機能を設定します。関数は、OnSuiteInstallBefore event ([インストール] 操作の場合) と OnSuiteMaintBefore event ([変更] 操作の場合) のデフォルト コードによって呼び出されます。
SuiteFormatString	文字列内のアドバンスト UI またはスイート / アドバンスト UI プロパティを、アドバンスト UI またはスイート / アドバンスト UI インストールからの値で解決します。
SuiteGetProperty	アドバンスト UI またはスイート / アドバンスト UI インストールからアドバンスト UI またはスイート / アドバンスト UI プロパティの値を取得します。

テーブル 35 • スイート / アドバンスト UI およびアドバンスト UI の対話関数 (続き)

関数	説明
SuiteLogInfo	アドバンスト UI またはスイート / アドバンスト UI インストールで実行中の InstallScript パッケージまたはアクションについての情報を、アドバンスト UI またはスイート / アドバンスト UI デバッグ ログに記録します。
SuiteReportError	アドバンスト UI またはスイート / アドバンスト UI のユーザー インターフェイスで、InstallScript パッケージの実行中に発生したエラーを通知するメッセージ ボックスを表示します。
SuiteResolveString	アドバンスト UI またはスイート / アドバンスト UI の文字列 ID を、アドバンスト UI またはスイート / アドバンスト UI インストールで実行中の InstallScript パッケージまたはアクション内の対応する文字列値で置き換えます。
SuiteSetProperty	アドバンスト UI またはスイート / アドバンスト UI インストールのアドバンスト UI またはスイート / アドバンスト UI プロパティの値を設定します。

テキスト置換

***プロジェクト** • この情報は、InstallScript プロジェクトに適用します。*

テキスト置換は、文字列を別の文字列に関連付けて (例、"<MYTEXTSUB>" を「テキストサブ値」に関連付ける)、他の文字列内で前の文字列を後の文字列に置換します (例、「この文字列は <MYTEXTSUB> のテキスト置換をデモンストレーションします」を「この文字列は『テキストサブ値』のテキスト置換をデモンストレーションします」に変更する)。テキスト置換の関連付けは、メイン インストールと含まれているすべてのオブジェクトのスクリプトに適用すること (グローバル) も、発生するスクリプト ファイル、#include プリプロセッサ命令を使ってそのスクリプトに含まれているスクリプト ファイル、およびスクリプトが含まれるすべてのスクリプト ファイルに適用すること (ローカル) もできます。オブジェクト スクリプトで定義するローカル テキスト置換の関連付けは、メイン インストールやインストールに含まれるその他のオブジェクトではなくそのオブジェクトだけに適用されます。メイン インストール スクリプトで定義するローカル テキスト置換の関連付けは、インストールに含まれるオブジェクトではなくメイン インストールにのみ適用されます。

テキスト置換の関連付けは、別のテキスト置換の関連付けに埋め込むことができます。例えば、"<MYTEXTSUB1>" を "My Text Sub 1 Value" に、"<MYTEXTSUB2>" を "Text Sub <MYTEXTSUB1> Embedded" にというように関連付けることができます。

InstallScript はテキスト置換の使用用に次の関数を含めます。

テーブル 36 • テキスト置換関数

関数	説明
TextSubGetValue	指定した文字列に関連付けられたテキスト置換文字列を取り出します。
TextSubParseTextSub	指定した文字列で最初のテキスト置換を見つけます。
TextSubSetValue	指定した文字列間でテキスト置換の関連付けを作成します。

テーブル 36 • テキスト置換関数 (続き)

関数	説明
TextSubSubstitute	指定した文字列変数でテキスト置換を実行します。

アンインストール関数

次の関数は、インストール済みのアプリケーションのアンインストール及び / またはメンテナンスセットアップに必要なサービスを実行します。

テーブル 37 • アンインストール関数

関数	説明
FeatureGetTotalCost	指定された機能のインストールとアンインストールに必要な総容量を判断します。
FeatureTransferData	指定された機能のインストールやアンインストールを実行します。
InstallationInfo	会社名、製品名、および製品のバージョン番号に基づくレジストリキーを作成します。
RegDBGetItem	アプリケーションごとのパスキーまたはアプリケーション アンインストール キーにある値を取得します。
RegDBSetItem	アプリケーションごとのパスキーまたはアプリケーションのアンインストールキーの下にある値を割り当てます。

ユーザー インターフェイス関数

ユーザーインターフェイスを関数を使用して、特定のエラーメッセージや、エラーメッセージボックスのタイトルをカスタマイズできます。ただし、セットアップの開発時に発生することがある内部エラーメッセージの中には、ユーザーインターフェイス関数で変更できないものがあります。

テーブル 38 • ビジュアルインターフェイス関数

関数	説明
Disable	ユーザーインターフェイス オブジェクトの表示を無効にします。
Enable	ユーザーインターフェイス オブジェクトの表示を有効にします。
FindWindow	ハンドルをウィンドウへ読み出します。
PlaceBitmap	インストール ウィンドウに画像を挿入します。
PlaceWindow	ユーザーインターフェイスオブジェクトの位置を設定します。

テーブル 38 • ビジュアルインターフェイス関数 (続き)

関数	説明
PlayMMedia	Adobe Flash アプリケーション ファイル (.swf)、AVI ファイル、またはサウンド ファイル (MIDI または WAVE) を再生します。
RGB	指定されたレッド、グリーン、ブルーの値に基づいてカスタムカラー値を返します。
SetColor	様々なユーザーインターフェイスアイテムの色を変更します。
SetDialogTitle	カスタム ダイアログのタイトルを作成します。
SetDisplayEffect	ビットマップとメタファイルの画像の表示効果を設定します。
SetErrorMsg	ディスク エラーが発生したとき、この関数は EnterDiskError 関数によって表示される対応エラー メッセージを設定します。
SetErrorTitle	ディスク エラーが発生したとき、この関数は EnterDiskError 関数によって表示されるエラー メッセージのタイトルバーを設定します。
SetFont	フォントの種類とスタイルを設定します。
SetStatusWindow	ステータスウィンドウのテキストを設定し、進行状況インジケーターに表示する完了パーセンテージの初期値を設定します。
SetTitle	メインウィンドウのタイトルのテキストと色を設定します。
SizeWindow	ほとんどのユーザーインターフェイスオブジェクトの大きさを指定します。
StatusUpdate	次のファイル転送操作後に進行状況インジケーターに表示する完了パーセントを設定します。

バージョンチェック関数

次の関数を使って、Windows ベースのシステムに存在するバージョン情報にアクセスが可能です。関数を利用するためには、バージョンリソースについての背景知識が必要です。Microsoft Windows ドキュメントを参照し、バージョンリソースについてより理解を深めてください。

テーブル 39 • バージョンチェック関数

関数	説明
VerCompare	バージョン情報を含む 2 つの文字列を比較します。
VerFindFileVersion	指定されたファイルを検索して、ファイルのバージョンと場所を取得します。

テーブル 39・バージョンチェック関数（続き）

関数	説明
VerGetFileLanguages	指定されたファイルがサポートする言語のリストを取得します。
VerGetFileVersion	指定されたファイルのバージョンを読み出します。
VerProductCompareVersions	製品バージョンを比較します。
VerProductGetInstalledVersion	アプリケーション アンインストール レジストリキーの Version 値がパックされた DWORD の場合、そのデータに対応する文字列を返します。
VerProductIsVersionSupported	バージョン文字列がサポートされているバージョンかどうかをチェックします。
VerProductNumToStr	指定のパックされた DWORD バージョンの文字列を返します。
VerProductStrToNum	指定の文字列のパックされた DWORD バージョンを返します。
VerProductVerFromVerParts	nVersionMajor、nVersionMinor、および nVersionBuild が指定したバージョン部分に対応するパックされた DWORD を取得します。
VerProductVerPartsFromVer	指定のパックされた DWORD のバージョン部分を個別の数値として取得します。
VerSearchAndUpdateFile	既存のファイルを、それより新しいバージョンと置き換えます。指定したファイルが存在しない場合、それより新しいバージョンがインストールされます。
VerUpdateFile	既存のファイルを、それより新しいバージョンと置き換えます。指定したファイルが存在しない場合、それより新しいバージョンがインストールされます。

Windows Installer 関数

Windows Installer 関数または API は、Windows Installer エンジンによってエクスポートされる関数です。これらの関数を使用すると、クエリを送ったり、実行中のインストーラーのテーブルを一時的に操作することができます。

一時レコードを実行中のデータベースに追加する主な用法として、ユーザー インターフェイス要素に実行時まで使用できないデータを自動入力する方法があります。たとえば Windows InstallerAPI を使用して、マップされたネットワーク ドライブ、ユーザー アカウント、ディレクトリ名、またその他にインストーラーが特定のターゲット システムで実行された場合にのみ使用可能になるデータをダイアログの ListBox 制御に自動入力することができます。

Windows Installer 関数の詳細については、Windows Installer ヘルプライブラリ を参照してください。

Windows Installer API 関数

Windows Installer API 関数はメイン スクリプトから、また InstallScript カスタムアクション 内から呼び出すことができます。InstallScript では、これらの Windows Installer API 関数をサポートします。

***メモ・** Windows Installer API 関数は現在実行中のデータベースへのハンドルを引数とみなします。InstallScript カスタムアクションでは、データベースハンドルはカスタムアクションへ渡される HWND 引数です。イベントハンドラー関数では、実行中の .msi データベースへのハンドルを格納するグローバル変数 ISMSI_HANDLE を利用することができます。*

テーブル 40 ・ Windows Installer API 関数

MsiApplyPatch	MsiGetLanguage	MsiRecordSetInteger
MsiCloseHandle	MsiGetLastErrorRecord	MsiRecordSetStream
MsiCreateTransformSummaryInfo	MsiGetMode	MsiRecordSetString
MsiDatabaseApplyTransform	MsiGetProperty	MsiSequence
MsiDatabaseExport	MsiGetSourcePath	MsiSetComponentState
MsiDatabaseGenerateTransform	MsiGetSummaryInformation	MsiSetFeatureAttributes
MsiDatabaseGetPrimaryKeys	MsiGetTargetPath	MsiSetFeatureState
MsiDatabaseImport	MsiInstallProduct	MsiSetInstallLevel
MsiDatabaseIsTablePersistent	MsiOpenDatabase	MsiSetMode
MsiDatabaseMerge	MsiOpenPackage	MsiSetProperty
MsiDatabaseOpenView	MsiPreviewBillboard	MsiSetTargetPath
MsiDoAction	MsiPreviewDialog	MsiSummaryInfoGetProperty
MsiEnumComponentCosts	MsiProcessMessage	MsiSummaryInfoSetProperty
MsiEvaluateCondition	MsiRecordClearData	MsiVerifyDiskSpace
MsiFormatRecord	MsiRecordDataSize	MsiViewClose
MsiGetActiveDatabase	MsiRecordGetFieldCount	MsiViewExecute
MsiGetComponentState	MsiRecordGetInteger	MsiViewFetch
MsiGetFeatureCost	MsiRecordGetString	MsiViewGetColumnInfo
MsiGetFeatureState	MsiRecordIsNull	MsiViewGetError

テーブル 40 · Windows Installer API 関数 (続き)

MsiGetFeatureValidStates	MsiRecordReadStream

カテゴリ別 Windows InstallShield API 関数

このセクションには、InstallScript で使用可能な Windows Installer API 関数の関数シグネチャが含まれています。関数の利用法、パラメーター、戻り値、そしてシーケンス制限に関する情報については、Windows InstallShield ヘルプライブラリ をご覧下さい。

MSI プロパティ関数とモード関数

テーブル 41 ・ MSI プロパティ関数とモード関数

関数	説明
prototype INT MsiSetProperty(HWND, BYVAL STRING, BYVAL STRING);	Windows Installer プロパティの値を設定します。プロパティが存在しない場合は、作成します。(具体例については、プロパティの取得と設定を参照してください。)
prototype INT MsiGetProductInfo(BYVAL STRING, BYVAL STRING, BYVAL STRING, BYREF INT);	パブリッシュ、並びにインストール済み製品について製品情報を返します。
prototype INT MsiGetProperty(HWND, BYVAL STRING, BYREF STRING, BYREF INT);	Windows Installer プロパティの値を取得します。プロパティが存在しない場合、ヌル文字列 ("") を返します。
prototype INT MsiGetLanguage(HWND);	実行中のインストールの数値言語 ID を返します。
prototype BOOL MsiGetMode(HWND, INT);	内部ブール型インストーラーの状態を返します。
prototype INT MsiSetMode(HWND, INT, BOOL);	内部ブール型インストーラーの状態を設定します。

機能関数とコンポーネント関数

テーブル 42 ・ 機能関数とコンポーネント関数

関数	説明
prototype INT MsiGetFeatureState(HWND, BYVAL STRING, BYREF INT, BYREF INT);	機能のインストール状態とアクション状態を取得します。
prototype INT MsiSetFeatureState(HWND, BYVAL STRING, INT);	機能のインストール状態を設定します。
prototype INT MsiSetFeatureAttributes(HWND, BYVAL STRING, INT);	機能の属性を設定します。
prototype INT MsiGetFeatureValidStates(HWND, BYVAL STRING, BYREF INT);	機能の有効なインストール状態を示すビットフラグのセットを返します。
prototype INT MsiGetComponentState(HWND, BYVAL STRING, BYREF INT, BYREF INT);	コンポーネントのインストール状態とアクション状態を取得します。

テーブル 42 - 機能関数とコンポーネント関数 (続き)

関数	説明
prototype INT MsiSetComponentState(HWND, BYVAL STRING, INT);	コンポーネントのインストール状態を設定します。
prototype INT MsiGetFeatureCost(HWND, BYVAL STRING, INT, INT, BYREF INT);	機能のディスクコスト (512 バイト単位)、そしてオプションでその親機能、並びに子機能を返します。
prototype INT MsiSetInstallLevel(HWND, INT);	製品全体のインストールレベルを設定します。

ディレクトリ関数

テーブル 43 - ディレクトリ関数

関数	説明
prototype INT MsiGetSourcePath(HWND, BYVAL STRING, BYREF STRING, BYREF INT);	ディレクトリテーブルで一覧となっているディレクトリの完全ソースパスを返します。(ディレクトリテーブルは、ダイレクトエディターで表示されます。)
prototype INT MsiGetTargetPath(HWND, BYVAL STRING, BYREF STRING, BYREF INT);	ディレクトリテーブルで一覧となっているディレクトリの完全ターゲットパスを返します。
prototype INT MsiSetTargetPath(HWND, BYVAL STRING, BYVAL STRING);	ディレクトリテーブルで一覧となっているディレクトリの完全ターゲットパスを設定します。
prototype INT MsiVerifyDiskSpace(HWND);	現在のインストレーションに充分なディスク容量があるかどうかを確認します。

データベース関数

MsiGetActiveDatabase を除いて、これらの関数のほとんどについて最初の HWND 引数は特定のデータベースビューまたはレコードへのハンドルです。

テーブル 44 - データベース関数

関数	説明
prototype INT MsiEvaluateCondition(HWND);	プロパティ名と値を含む条件式を評価します。
prototype INT MsiGetActiveDatabase(HWND);	データベースビューを開くのに利用できる、実行中の .msi データベースへのハンドルを取得します。
prototype INT MsiDatabaseApplyTransform(HWND, BYVAL STRING, INT);	トランスフォームをデータベースに適用します。トランスフォームは、オリジナルデータベースを修正することなくデータベースへの変更を記録する手段です。

テーブル 44 - データベース関数 (続き)

関数	説明
prototype INT MsiDatabaseExport(HWND, BYVAL STRING, BYVAL STRING, BYVAL STRING);	テキストアーカイブファイルへのオープンデータベースからインストーラーテーブルをエクスポートします。
prototype INT MsiDatabaseGenerateTransform(HWND, BYVAL STRING, INT, INT);	2 つのデータベース間の差異を含むトランスフォームファイルを生成します。トランスフォームは、オリジナルデータベースを修正することなくデータベースへの変更を記録する手段です。
prototype INT MsiDatabaseGetPrimaryKeys(HWND, BYVAL STRING, BYREF HWND);	指定したテーブルのすべてのプライマリキー列の名前を含むレコードを返します。この関数は MsiCloseHandle を利用して閉じるハンドルを返します。
prototype INT MsiDatabaseImport(HWND, BYVAL STRING, BYVAL STRING);	インストーラーテキスト アーカイブテーブルをオープンデータベースにインポートします。
prototype INT MsiDatabaseIsTablePersistent(HWND, BYVAL STRING);	特定のテーブルの状態を説明する一覧表を返します。
prototype INT MsiDatabaseMerge(HWND, HWND, BYVAL STRING);	重複行を許可して 2 つのデータベースを合併します。
prototype INT MsiDatabaseOpenView(HWND, BYVAL STRING, BYREF INT);	データベースクエリを準備し、ビューオブジェクトを作成します。
prototype INT MsiFormatRecord(HWND, HWND, BYREF STRING, BYREF INT);	フォーマット文字列を利用してレコードフィールドデータとプロパティをフォーマットします。
prototype INT MsiViewModify(HWND, INT, HWND);	データベースレコードを修正します。インストールの実行には、一時データベースの変更のみが可能です。
prototype INT MsiOpenDatabase(BYVAL STRING, BYVAL STRING, BYREF HWND);	データアクセスのため、データベースファイルを開きます。この関数は MsiCloseHandle を利用して閉じるハンドルを返します。
prototype INT MsiViewClose(HWND);	実行済みデータベースビューを閉じます。
prototype INT MsiViewExecute(HWND, HWND);	SQL クエリを実行します。
prototype INT MsiViewFetch(HWND, BYREF HWND);	現在のデータベースビュー用のレコードを取り出します。
prototype INT MsiRecordGetString(HWND, INT, BYREF STRING, BYREF INT);	指定したレコードの指定したフィールドに格納されている文字列を返します。
prototype INT MsiRecordSetString(HWND, INT, BYVAL STRING);	指定したレコードの指定したフィールドに格納されている文字列を設定します。

テーブル 44 • データベース関数 (続き)

関数	説明
prototype INT MsiRecordReadStream(HWND, INT, CHAR, POINTER);	レコードフィールドの文字列値を返します。
prototype INT MsiRecordSetStream(HWND, INT, BYVAL BINARY);	ファイルからレコードストリームフィールドを設定します。ストリームデータは一時フィールドには挿入できません。
prototype INT MsiRecordGetInteger(HWND, INT);	指定したレコードの指定したフィールドに格納されている整数を返します。
prototype INT MsiRecordSetInteger(HWND, INT, INT);	指定したレコードの指定したフィールドに格納されている整数を設定します。
prototype INT MsiViewGetColumnInfo(HWND, INT, BYREF INT);	データベース列名または定義を含むレコードを返します。
prototype INT MsiRecordGetFieldCount(HWND);	レコードのフィールド (列) 番号を返します。
prototype INT MsiCloseHandle(HWND);	レコードハンドルまたはデータベースを閉じます。
prototype MsiCloseAllHandles( );	開いているすべてのハンドルを閉じます。診断目的であることが前提であり、一般的なクリーンアップの為に呼び出すことはできません。
prototype INT MsiViewGetError(HWND, BYREF STRING, BYREF INT);	MsiViewModify が生成するエラーのエラーコードを返します。

概要情報ストリーム管理関数

テーブル 45 ・ 概要情報ストリーム管理関数

関数	説明
prototype MsiGetSummaryInformation(HWND, BYVAL STRING, INT, BYREF HWND);	インストーラーデータベースの概要情報データへのハンドルを取得します。この関数は MsiCloseHandle を利用して閉じるハンドルを返します。
prototype INT MsiSummaryInfoGetProperty(HWND, INT, BYREF INT, BYREF INT, POINTER, BYREF STRING, BYREF INT);	概要上から単一のプロパティを取得します。
prototype INT MsiSummaryInfoSetProperty(HWND, INT, INT, INT, POINTER, BYREF STRING);	単一の概要情報プロパティを設定します。

その他の関数

テーブル 46 ・ その他の関数

関数	説明
prototype INT MsiApplyPatch(BYVAL STRING, BYVAL STRING, INT, BYVAL STRING);	パッチを受け取ることができるパッチパッケージが一覧にする各製品に対し、MsiApplyPatch 関数はインストレーションを呼び出し、PATCH プロパティをパッチパッケージのパスへ設定します。
prototype HWND MsiCreateRecord(INT);	特定した数のフィールドを含む新規レコードを作成します。この関数は MsiCloseHandle を利用して閉じるハンドルを返します。
prototype INT MsiDoAction(HWND, BYVAL STRING);	ビルトインアクション、カスタムアクション、またはユーザーインターフェイスウィザードアクションを実行します。
prototype INT MsiEvaluateCondition(HWND, BYVAL STRING);	プロパティ名と値を含む条件式を評価します。
prototype INT MsiInstallProduct(BYVAL STRING, BYVAL STRING);	製品をインストールまたはアンインストールします。
prototype INT MsiOpenPackage(BYVAL STRING, BYREF HWND);	製品データベースへアクセスする関数と共に利用するパッケージを開きます。ハンドルが必要でなくなった時点で、MsiCloseHandle 関数をハンドルと共に呼び出さなくてはなりません。
prototype INT MsiPreviewBillboard(HWND, BYVAL STRING, BYVAL STRING);	表示されたダイアログのホストコントロール内でビルボードを表示します。ビルボード名をヌルに指定すると、ビルボード表示はすべて削除されます。
prototype INT MsiPreviewDialog(HWND, BYVAL STRING);	ダイアログをモードレス、非アクティブ状態で表示します。

テーブル 46 • その他の関数 (続き)

関数	説明
prototype int MsiProcessMessage(HWND, int, HWND);	処理用にエラーレコードをインストーラーへ送ります。
prototype INT MsiSequence(HWND, BYVAL STRING, INT);	指定したテーブル内で説明される様に、別のアクションシーケンスを実行します。

Windows Installer API 関数の例

```
/*---------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * ランタイムに、.msi データベースのテーブルにアクセスして
 * テンポラリ レコードを追加するのに利用される
 * Windows API 関数のいくつかの例を示します。
 *
 * メモ : このスクリプトを実行する前に、ReadyToInstall ダイアログへ
 *       ListBox コントロールを追加し、プロパティ LISTBOXPROP に
 *      そのコントロールを関連付けます。
 *
 * このスクリプトは、ReadyToInstall ダイアログの ListBox コントロールに
 * プロパティテーブルにリストされているすべてのプロパティの現在の値を
 * 挿入します。ListBox にエンドユーザーが選択したものは、
 * LISTBOXPROP に格納されます。
 *
¥*---------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "ifx.h"

//  InstallScript カスタムアクションのエントリポイント
export prototype PropDisplay(HWND);

function PropDisplay(hInstall)
    HWND hDB, hViewlist, hRecordlist;
    HWND hViewprop, hRecordprop;
    NUMBER nBuffer, r;
    STRING svPropname, svPropvalue;
begin

    hDB = MsiGetActiveDatabase(hInstall);

    // ListBox テーブルにビューを開きます
    MsiDatabaseOpenView(hDB,
        "SELECT * FROM `ListBox` WHERE `Property`='LISTBOXPROP'",
        hViewlist);
    MsiViewExecute(hViewlist, NULL);

    // Property テーブルにビューを開きます
    MsiDatabaseOpenView(hDB,
        "SELECT * FROM `Property`", hViewprop);
    MsiViewExecute(hViewprop, NULL);
```

```
    r = 0;

    // 各 Property レコードに対して、ListBox テーブルに
    //   PROPNAME="value" レコードを追加します
    while (MsiViewFetch(hViewprop, hRecordprop) != ERROR_NO_MORE_ITEMS)

        nBuffer = 256; // サイズ バッファーを設定します
        MsiRecordGetString(hRecordprop, 1, svPropname, nBuffer);
        nBuffer = 256; // サイズ バッファーをリセットします
        MsiGetProperty(hInstall, svPropname, svPropvalue, nBuffer);

        r = r + 1;

        hRecordlist = MsiCreateRecord(4);

        MsiRecordSetString(hRecordlist, 1, "LISTBOXPROP");
        MsiRecordSetInteger(hRecordlist, 2, r);
        MsiRecordSetString(hRecordlist, 3, svPropname);
        MsiRecordSetString(hRecordlist, 4, svPropname + "=" + svPropvalue);

        // 開いている .msi データベースは一時的にのみ変更が可能です
        MsiViewModify(hViewlist, MSIMODIFY_INSERT_TEMPORARY, hRecordlist);

    endwhile;

    MsiViewClose(hViewlist);
    MsiViewClose(hViewprop);
end;
```

演算子

演算子は、加算などの単一または複数のオペランドを利用して行う基本的な処理を指定する記号です。オペランドには定数、変数、または関数呼び出しが利用できます。下の例では、プラス記号 (+) はその左側にある変数と右側にある値を加算することを示します。

```
nCounter + 1;
```

演算子の機能

InstallScript のほとんどの演算子は C 言語で対応する演算子と非常に類似した動作を行います。C の様に、+ や ^ といった演算子の関数の中には、オペランドのデータ型に依存するものもあります。例えば、加算記号が数値オペランドと共に利用された場合は、加算が行われます。加算記号が文字列オペランドと共に利用された場合は、連結が行われます。

式

演算子とオペランドの組み合わせは式と呼ばれます。上の例は演算子が 1 つだけ含まれたシンプルな式です。下のような複雑な式では、複数の演算子を指定します :

```
nPrincipal * nRate - nFee;
```

複雑な式の評価

複雑な式では、演算が行われる順番は演算子の優先順位に基づきます。InstallScript 演算子の優先順位は C 言語のそれと同じです。C と同じように、式を括弧で囲んで優先順位を変更することができます。

結果

ほとんどの式はオペランドの既存値に基づいて新しい値を生成します。その種類にかかわらず、Trialware のリリースをビルドするときに [リリース] ビューの "Trialware ビルドの無効化 " プロパティがデフォルトの [いいえ] に設定されている場合、TestTools と呼ばれるフォルダーが、選択されたリリース用に DiskImages フォルダーと同じフォルダーに作成されます。算術式は数値を生成し、ブール型の式は TRUE または FALSE を生成します。文字列式は文字列を生成します。

アドレス演算子 (&)

アドレス演算子は単一の演算子で、スクリプト中の変数すべてのメモリーアドレスを取得するのに利用することができます。演算子そのものは変数名の前に配置し、スペースは挿入しません。アドレス演算子を使ってポインター変数へ変数のアドレスを割り当てたり、関数の呼び出しで引数として変数のアドレスを渡すことができます。アドレスを C プログラムに送り、一般的な C ポインターと同じ要領で操作することができます。

次の例では、データ構造のアドレスがポインター変数へ割り当てられています :

```
typedef DIMENSIONS
begin
  SHORT sLength;
  SHORT sWidth;
end;

DIMENSIONS rectangle;
DIMENSIONS POINTER pointerObject;

begin
```

```
  pointerObject = &rectangle;

  // を参照してください。

end;
```

注意・ローカル変数と共にアドレス演算子を利用する場合、宣言を行った関数が有効な期間のみローカル変数が存在することに注意してください。関数が戻された後、ローカル変数のアドレスは無効となります。

パス追加演算子 (^)

パスを 2 つ組み合わせる場合、またはパスとファイル名を組み合わせる場合はパス追加演算子 (^) を利用します。ファイル名またはサブディレクトリを追加する際、パスに適切な数の円記号が追加されているかどうか演算子が自動的に確認します。

次の様に入力した場合 :

```
  szStringVar = "C:¥¥MyPath¥¥" ^ "YourPath¥¥FileName";
```

szStringVar からの出力は :

C:¥MyPath¥YourPath¥FileName

MYPATH の後に円記号を追加し忘れた場合、次のように表示されます :

```
  szStringVar = "C:¥¥MyPath" ^ "YourPath¥¥FileName";
```

結果は有効な状態のままです。InstallScript ^ 演算子は自動的に円記号を追加します。

メモ・レジストリキーなど、パス以外の文字列を組み合わせる際に ^ 演算子を利用しないで下さい。別の種類の文字列には、連結演算子 (+) を利用します。

算術演算子 (+、−、*、/)

算術演算子はオペランドと共に加算や減算といった算術演算を行います。算術演算子には、単項と二項の二種類あります。単項演算子は、単一のオペランドの処理を行います。二項演算子は、2 つのオペランドの処理を行います。複数の演算子を含んだ複雑な式では、評価の順番は演算子優先順位に従います。

算術演算子の優先順位

InstallScript コンパイラが複雑な式 (シンプルな式を複数含むもの) に遭遇したとき、これらの式を一度に評価します。

演算子の優先順位

式が評価される順番は演算子の優先順位によって決定付けられます。コンパイラーは、C 言語が利用するのと同じ優先順位で算術演算子を評価します。

1. 負 (-) 単項。
2. 乗算と除算。
3. 加算と減算。
4. 左から右への処理

優先順位が同じレベルの演算子が複数同時に存在する場合、左側にある演算子から先に処理されます。たとえば、15 / 3 * 7 式では、InstallScript コンパイラはまず除算 (15 / 3) を行ってからその結果に 7 を乗算します。

処理に影響する括弧の利用

優先順位が低い演算子を最初に実行する場合は、括弧で囲まなくてはなりません。例えば、式 30 / 3 +7 で乗算の前に加算を行う場合、3 + 7 の前後に括弧を配置します。式 30 / (3 + 7) の括弧によって、コンパイラは初めに 3 と 7 を加算して 10 とし、そして 30 を 10 で除算することで結果を 3 とします。

ネストされた括弧

式の中で括弧をネストすることができます。InstallShield はネストされた式の計算用に一時的に 20 箇所を確保します。したがって、括弧の中で 19 レベルの演算をネストすることが可能です。InstallShield スクリプトコンパイラは最初に一番奥の演算行い、順に外側を処理していきます。

式の実例

例えば、式 36 - (3 * (2 + 6 - 4)) では、InstallScript コンパイラはまず 2 + 6 を行って 8 とし、8 から 4 を減算、そして 4 に 3 を乗算して 12 とし、最後に 36 から 12 を減算して結果を 24 とします。

メモ・括弧の中では、InstallShield はまず加算を行います、なぜなら同じ優先順位をもつ 2 つの演算子のうち一番左側に位置しているからです。

バイナリ算術演算子

InstallScript コンパイラは、次のテーブルに一覧表示されたバイナリ算術演算子を認識します。

テーブル 1・バイナリ算術演算子

記号	演算子	例	説明
+	加算	x + y	2 つのオペランドを加算します。
-	減算	x - y	最初のオペランドから 2 番目のオペランドを減算します。
*	乗算	x * y	2 つのオペランドを乗算します。

テーブル 1・バイナリ算術演算子（続き）

記号	演算子	例	説明
/	除算	x / y	2 番目のオペランドで 1 番目のオペランドが除算されます。 InstallShield は浮動小数点データタイプを持たないので、回答の小数点以下は省略されます。例えば、5/2 = 2.5 --> 2 となります。
%	剰余	19 % 5	最初のオペランドを 2 番目のオペランドで除算した余りを戻します。例えば、nResult = 19 % 5; results in nResult = 4 となります。

ヒント・算術演算子の前後には空白を含み、スクリプトを読みやすく見た目の統一を図ってください。

単項算術演算子

単項演算子は、単一のオペランド上で動作する算術演算子です。InstallScript コンパイラは 2 つの単項演算子、負（-）と正（+）を認識します。

負

負の単項演算子は式の記号を正から負へ、または負から正へ入れ替えます。式の前に負の単項演算子を利用した場合の結果は、本質的には式を -1 で乗算するのと同じです。

正

正の単項演算子は、その式を 1 で乗算した式と同じ結果を持ちます。負の数字の記号が正に変わることはありません。

代入演算子

次に示したコードサンプルに見られるように、定数、リテラル、変数、式結果、関数結果を同じ種類の変数へコピーするには、代入演算子 (=) を利用します。

```
STRING szName;
LONG nValue;
BOOL  bDone;
HWND hInstance;
INT  iStyle;
LIST LISTINFO;

szName = "InstallShield";
nValue = 15;
bDone = FALSE;
hInstance = 0;
iStyle = DLG_MSG_STANDARD|DLG_CENTERED;
```

```
LISTINFO = ListCreate(STRINGLIST);
```

上の例のように、InstallShield は明らかな文字列の長さ無しで宣言された文字列変数のサイズを自動的に決定します。デフォルトでは InstallShield は文字列変数を 256 バイトに自動設定します。変数へ (ヌルターミネータ を含んで) 256 バイトよりも長い文字列を割り当てた場合、InstallShield はその文字列変数用に確保するメモリ容量を増やします。

長さを明確にして文字列変数を宣言する場合、それに割り当てる文字列を受け取るのに充分な長さに設定しなくてはなりません。次の例では、文字列リテラルは 51 文字含んでいます。したがって、szStringVarA と szStringVarB は両方とも 52 以上の長さが宣言されなくてはなりません。これは、文字列そのものと文字列の末尾に自動的に追加されるヌルターミネータを含んだ充分な長さです。

```
STRING szStringVarA[52], szStringVarB[52];
szStringVarA = " これは 51 文字から成るサンプル文字列です。.";
szStringVarB = szStringVarA;
```

注意・*C++ とは異なり、InstallScript は単一のステートメント内で複数の代入演算子をサポートしません。InstallScript のステートメント a = b = c は、C++ ステートメントで言う a = b == c にあたります。つまり、初めの演算子は代入演算子として、また 2 番目の演算子は関係演算子として割り込んでいます。b が c と等しい場合、値 1 (TRUE) が a に割り当てられ、b が c と等しくない場合、値 0 (FALSE) が a に割り当てられます。*

ビット演算子 (&, |, ^, ~, <<, >>)

ビット、またはビット単位演算子を使うと、数値変数の各ビットを操作することができます。効果的に演算子を利用できるよう、バイナリ表記を理解する必要があります。このトピックではバイナリ演算子の概要を説明しますが、バイナリ表記については触れていません。

ビット演算子は、1 つの点を除いて論理演算子と同じ働きを持ちます。論理演算子は式に利用しますが、ビット演算子はビットで利用します。InstallScript コンパイラは、次のテーブルに一覧表示されたビット単位の演算子を認識します。

テーブル 2・ビット演算子

記号	演算子	説明
&	BitAND	BitAND は、演算子両方のビットが 1 の場合のみ 1 (TRUE) の結果をだします。その他の場合、結果は 0 (FALSE) です。
|	BitOR	OR を含むビットは演算子のビットが 0 の場合のみ 0 の結果を出します。その他の場合は、結果は 1 です。
^	BitXOR	OR を含まないビットは、オペランド内の対応するビットが相違する場合 (ひとつは 1、他方は 0) に 1 の結果を出します。その他の場合は、結果は 0 です。

テーブル 2 - ビット演算子 (続き)

記号	演算子	説明
~	BitNOT	BitNOT は単項演算子で、オペランド内の各ビットを逆にします。次の例に示したとおり、1 は 0 に、0 は 1 となります。 ~00001100 結果は 11110011 です。
<<	Shift left	指定されたビット数でビットを左側に移動させます。たとえば、次に示した式ではビットを 3 スペース分左側に移動します。 00001100 << 3 結果は 01100000 です。
>>	Shift right	指定されたビット数でビットを右側に移動させます。たとえば、次に示した式ではビットを 2 スペース分右側に移動します。 00001100 >> 2 結果は 00000011 です。

InstallScript でのシフト処理は、C 言語のそれと同様に動作します。2 ビット右へシフト (>> 2) させるとき、一番右側にある 2 つのビット値は失われます。右の 2 つのビット位置へ別のビット値がシフトされ、記号ビットが空白ビットへシフトされます。

シフト処理を使って 2 で累乗、または累除することができます。n スペース分整数を左にシフトさせるのは、その数 を 2 の n 乗で掛けた場合と同じです。n スペース分整数を右にシフトさせるのは、その数を 2 の n 乗で割った場合と同じです。シフト処理を使って 2 で累乗、または累除することができます。

BYREF 演算子

デフォルトでは、ユーザー定義関数のパラメーターは値が渡します。つまり、各パラメーターが指定するデータのコピーが関数へ渡されます。これは、関数がコピー上で処理を行い、元のデータは関数によって変更されることが無いためです。

メモ - 1 つだけ例外があります : *デフォルトでは、DLL 関数へ渡される文字列変数はリファレンスによって渡され、変数の値は関数によって変更される場合があります。*

ユーザー定義関数を、変数のコピーではなくパラメーターで受け取った変数上で直接処理する場合、次に表示した関数プロトタイプでパラメーター型を宣言して BYREF 演算子を指定しなければなりません。

```
prototype StrInvert( BYREF STRING );
```

BYREF 演算子はパラメーターがリファレンスによって渡されることを示します。つまり、実際の変数は関数へ渡され、その変数に加えられた変更はいづれも、関数が戻された際に呼び出した関数によるアクセスが可能です。リファレンスによって渡されたパラメーターは変数を必要とするところから、しばしば変数パラメーターと呼ばれます。リファレンスが定数やリテラルを渡すことはありません。

複数パラメーターと共に BYREF を利用する

ユーザー定義関数が複数のパラメーターを含む場合、各変数パラメーターと共に BYREF 演算子を指定しなくてはなりません。下の例では、最初と 3 番目のパラメーターがリファレンスによって渡され、2 番目のパラメーターは値によって渡されます :

```
prototype StrChangeChar( BYREF STRING, CHAR, BYREF BOOL);
```

制限

リファレンスがユーザー定義構成メンバーを渡すことはできません。これはコンパイラエラーにつながります。その代わりリファレンスを使って構成全体へポインターをパスしてから、構成ポインター演算子 (->) を利用して構成データ要素へアクセスまたはその変更を行う必要があります。

参照によって関数に渡されるオートサイズ文字列変数は呼び出された関数の中ではオートサイズされません。関数が現在のパラメーターのサイズより大きい長さの値を割り当てようとすると、ランタイム エラー 401 が発生します。このエラーを回避するためには、リファレンスが関数に値を渡すときの特定の文字列サイズを宣言します。

BYVAL 演算子

キーワード BYVAL は関数プロトタイプのパラメーター要件で指定することができ、下の例の様に、リファレンスではなく値によって渡されるパラメーターを示すことができます :

```
prototype DisplayString( BYVAL STRING );
```

パラメーターが値によって渡されたとき関数はそのコピーを受け取り、その値に加えられた変更はすべてローカル関数となります。

デフォルトでは、ユーザー定義関数のパラメーターは値によって渡されるので、関数プロトタイプでこのキーワードを指定する必要はありません。(1 つだけ例外があります : デフォルトでは、DLL 関数へ渡される文字列変数はリファレンスによって渡され、変数の値は関数によって変更される場合があります。)

変数のコピー上ではなく、パラメーターで受け取った変数で直接ユーザー定義関数を操作する場合、BYREF オペレータを指定しなくてはなりません。

文字列連結演算子 (+)

文字列連結演算子 (+) を利用して、1 つの文字列を別の文字列の後につなげることができます。連結された 2 つの文字列は、新しい 3 番目の文字列となります。次の例では、2 つの文字列定数が連結され、その結果が文字列値 szThirdString に割り当てられます。ステートメントが実行されたあと、szThirdString の値は " 初めの文字列 2 番目の文字列 " となります。

```
szThirdString = " 最初の文字列 " + "2 番目の文字列 ";
```

文字連結演算子式のオペランドは文字列リテラル、文字列定数、あるいは文字列変数のいづれでも可能です。次の例では、文字列定数と文字列リテラルが連結されます。結果の文字列値は szThirdString に割り当てられます。

```
#define FIRST_STRING " これが最初の文字列です "

STRING szThirdString;

// を参照してください。を参照してください。

szThirdString = FIRST_STRING + "2 番目の文字列 ";
```

注意・文字列連結演算子式の結果を文字列変数へ割り当てるとき、割り当てる文字列変数に対して連結文字列が長すぎないように注意しなくてはなりません。文字列を不適切なサイズの変数へ割り当てるステートメントは、ランタイム エラー 401 を発生させます。

間接演算子 (*)

間接演算子は単項の演算子で、ポインター変数が参照するメモリ位置に保存される値を取得するのに利用することができます。間接演算子はポインター変数名の前に配置し、スペースは挿入しません。

次の例では、nvalue は数値で、pnumber は数値へのポインターです。割り当てステートメントは pnumber がポイントする数値を nvalue へコピーするのに利用されます。

```
nvalue = *pnumber;
```

間接演算子はまた、次の例の様に数値をパラメーターとして扱う関数へ値を渡すのにも利用することができます :

```
somefunction(*pnumber);
```

次の制限は間接演算子へ適用されます :

- 間接演算子は、数値ポインターとのみ共に利用できます。
- 間接演算子は、値をメモリ位置へ割り当てるのに利用することはできません。
- 間接演算子は、BYREF 演算子をと共に宣言されたパラメーターを持つ関数への引数として利用することはできません。
- 間接演算子はポインターを宣言するのに利用することはできません。
- 間接演算子は関数宣言で変数パラメーターを宣言するのに利用することはできません。

論理演算子 (&&, ||, !)

論理演算子を利用すると、複数の相関的な質問を同時に行うことができます。例えば、論理演算子を使って y が 7 よりも大きく、szFilePath が “C:¥¥Program Files¥¥Company Name” を含む場合を質問できます。論理演算子は TRUE (1) または FALSE (0) の何れかの値を戻します。関係演算子と同様に、if や while ステートメントで最も頻繁に利用されます。

InstallScript コンパイラは、下のテーブルに一覧表示された論理演算子を認識します。

テーブル 3・論理演算子

記号	演算子	例	説明
&&	AND	exp1 && exp2	exp1 と exp2 の両方が true の場合のみ True で、その他の場合は false です。
||	または	exp1 || exp2	exp1 または exp2 の何れかが true の場合は True で、両方が false の場合のみ false (0) です。

テーブル 3 - 論理演算子 (続き)

記号	演算子	例	説明
!	NOT	!exp1	exp1 が true の場合 False で、exp1 が false の場合は true です。

論理演算子は、算術演算子や関係演算子よりもの優先順位が低いです。論理演算子の中でも、AND 演算子は OR 演算子よりも高い優先順位を持ちます。

注意 - *C++ とは異なり、InstallShield は論理式の完全なブール型の評価を行います。次の if ステートメントを考慮してください :*

```
if (iVar = 10) && (MyFunction( ) = 0) then
    MessageBox("TRUE です。", INFORMATION);
endif;
```

論理演算子の左側にある式が false の場合でも MyFunction が呼び出されます。ショート サーキットの論理演算評価 (&& の右側にある式は、&& の左側の式が true の場合のみ解決される) の効果を得るには、下に示すようなネストされた if ステートメントを利用します :

```
if (iVar = 10) then
    if (MyFunction( ) = 0) then
        MessageBox("TRUE です。", INFORMATION);
    endif;
endif;
```

メンバー演算子 (.)

メンバー演算子を使って構造変数の各要素を参照して下さい。メンバー演算子は構造変数名と要素名の間に空白無しで配置されなくてはなりません。下の例では、リテラル値は構造変数の各要素に割り当てられます。

```
typedef SETTINGSREC
begin
  BOOL bSwitchOn;
  STRING szMssg[255];
  INT nVal;
end;

SETTINGSREC settings;

settings.bSwitchOn = FALSE;
settings.szMssg = "Off";
settings.nVal = 0;
```

関係演算子 (<、>、=、<=、>=、!=)

関係演算子は条件ステートメントの内部で、if ステートメントと while ステートメントなど、ある式を別の式と比較します。たとえば、次のステートメントは "x は 20 より大きいか ?" を問い合わせます。

```
if (x > 20) then
```

この質問への解答は、TRUE (1) または FALSE (0) の何れかです。InstallScript コンパイラは、下のテーブルに一覧表示された関係演算子を認識します。

テーブル 4 - 関係演算子

記号	演算子	例	説明
=	等値	x = y	x が y と等しい場合は True。
>	より大きい	x > y	x が y 以上の場合は True。
<	より小さい	x < y	x が y 未満の場合は True。
>=	以上または等しい	x >= y	x が y 以上、または等しい場合は True。
<=	未満または等しい	x <= y	x が y 未満、または等しい場合は True。
!=	非等値	x != y	x が y と等しくない場合は True。

関係演算子を if...else ステートメントで利用する場合、プログラムは次のいづれかのアクションに従います :

- 式が TRUE の場合、if に続くステートメントは *実行されます*。else に続くステートメントは*実行されません*。
- 式が FALSE の場合、if に続くステートメントは *実行されません*。else に続くステートメントは*実行されます*。

関係演算子は算術演算子よりも全体的に優先順位が低いです。関係演算子の間で、以下、以下または等しい、以上、そして以上または等しいは、等値や非等値よりも優先順位をもちます。

注意 - 代入演算子と関係演算子を同じ条件式内で利用することはできません。たとえば、次のスクリプトは失敗します :

```
if ((listID = ListCreate (NUMBERLIST)) = LIST_NULL) then       .. を参照してください。 endif;
```

ヒント - == を使って等値をテストする C とは違って、InstallScript の代入演算子と関係演算子は同じ記号 (=) を利用します。

関係演算子の優先順位

関係演算子は算術演算子よりも全体的に優先順位が低いです。つまり、InstallShield は論理演算子の評価を始める前にすべての算術演算子を処理します。論理演算子は、括弧を使って演算の順序が変更されていない限り左から右へ評価されます。関係演算子内の優先順位は次の通りです :

- 最初に、未満 (<)、未満または等しい (<=)、より大きい (>)、およびより大きいまたは等しい (>=)。
- そして、等値 (=) と非等値 (!=)。

従って、算術式と関係式を組み合わせる場合、InstallShield スクリプトコンパイラは次の順序で処理の順番を評価します :

1. 負 (マイナス) 単項。
2. 乗算と除算。
3. 加算と減算。
4. 未満 (<)、未満または等しい (<=)、より大きい (>)、およびより大きいまたは等しい (>=)。
5. 等値 (=) と非等値 (!=)。

6 +7 > y 式では、InstallScript コンパイラは 6 と 7 を加算してから、結果の 13 を y と比較します。優先順位を変更するには括弧を利用してください。とえば、6 + (7 > y) 式では、InstallScript コンパイラはまず 7 が y 以上かどうかを判断します。7 が y 以上の場合は 1 (TRUE の数値) が 6 に加算され、7 が y 以上でない場合は 0 (FALSE の数値) が 6 に加算されます。

ステートメントに算術演算子、関係演算子、そして論理演算子が含まれる場合、コンパイラは次の順序で優先順位を評価します :

1. 負 (マイナス) 単項が最初優先順位です。
2. 乗算と除算が 2 番目の優先順位です。
3. 加算と減算は 3 番目の優先順位です。
4. 否定 (!) は 4 番目の優先順位です。
5. 未満 (<)、未満または等しい (<=)、より大きい (>)、およびより大きいまたは等しい (>=) は 5 番目の優先順位です。
6. 等値 (=) と非等値 (!=) は 6 番目の優先順位です。
7. AND (&&) は 7 番目の優先順位です。
8. OR (||) は 8 SP 1 番目の優先順位です。

論理演算子は if ステートメントと while ステートメント内で、関係演算子と同様に利用してください。下の例では、bInstallExample と bInstallHelp が true の場合に nExampleSize と nHelpSize を加算します。

```
if (bInstallExample && bInstallHelp) then
    nTotalSize = nExampleSize + nHelpSize;
endif;
```

次の例では、bInstallProgram1 *または* bInstallProgram2 の *いづれか* が TRUE の場合、bPublicFile を TRUE に設定します。

```
if (bInstallProgram1 || bInstallProgram2) then
    bPublicFile = TRUE;
endif;
```

***メモ・** && 演算子や || 演算子を関数への引数の中で使用することはできません。その代わり前述の例に従い、論理式の値をブール型変数に割り当ててから引数として変数と共に関数を呼び出してください。*

文字列演算子 (^、+、%)

文字列演算子を使うと、関数を利用せずに文字列を直接操作することができます。文字列演算子は大文字と小文字を区別しない点に注意してください。InstallScript コンパイラは次の文字列演算子をサポートします :

テーブル 5 · 文字列演算子

演算子	説明
パス追加演算子 (^)	パスまたはファイル名に追加パスを追加します。
文字列連結演算子 (+)	1 つの文字列に別の文字列を付け足します。
文字列検索演算子 (%)	サブ文字列を、別の文字列内で探します。
文字列定数演算子 (@)	プロジェクトの文字列エントリの 1 つに使用される文字列定数にアクセスします。

***メモ** · 文字列演算子のどちらの端にも、式を括る括弧を使用しないで下さい。たとえば、次のようなステートメントは避けてください :*

```
szPath = szTestPath ^ (AUTOFILE + ".bat");
```

その代わりに次に示す様に、文字列演算子を使って括弧を使わない式を作成してください。

```
szFile = AUTOFILE + ".bat";    szPath = szTestPath ^ szFile;
```

文字列定数演算子 (@)

文字列エントリを使って InstallScript コード内から定められた言語のテキスト文字列にアクセスできます。つまり、インストールの InstallScript コードと、インストール中に表示する言語固有の文字列とを全く個別のものとして取り扱い、各文字列はその文字列 ID を使って参照できます。InstallScript で文字列識別子を使用するとき、その先頭にアットマーク (@) を付けなくてはなりません。

[文字列エディター] ビューは、プロジェクト内の言語非依存文字列 ID の一覧と対応する言語固有の値が表示されます。詳細については、「エンド ユーザー インターフェイスをローカライズする」を参照してください。

InstallScript ファイル (.rul) を含むプロジェクトをビルドするとき、InstallScript コードに @ 演算子を使用する文字列エントリへの参照を 1 つ以上含まれている場合、InstallShield はビルド時に文字列エントリを検証します。プロジェクトに含まれる InstallScript ファイルの文字列 ID が [文字列エディター] ビューで定義されていない場合、InstallShield はビルド警告 -7174 を表示します。

***メモ** · @ 演算子は、文字列 ID で大文字と小文字を区別しません。したがって、スクリプトで文字列 ID を使用する場合、[文字列エディター] ビューで指定された文字列 ID の大文字小文字の区別と一致させる必要はありません。ただし、大文字と小文字を混ぜて使用することで、ビルド時に、スクリプト内の文字列エントリが [文字列エディター] ビューの対応文字列エントリと一致しないように防ぐことができます。したがって、文字列 ID のすべてのインスタンスで、大文字を使用することをお勧めします。*

*ヒント・文字列が見つからなかったときのエラー処理方法を用意する場合、@ 演算子の代わりに、**LoadStringFromStringTable** 関数を使用できます。*

構造ポインター演算子 (->)

構造ポインター演算子は、ポインター変数を使って構造内の個別の要素を参照するのに利用します。構造ポインター演算子はポインター変数名と要素名の間に空白無しで配置しなくてはなりません。下の例では、リテラル値は構造の各要素に割り当てられます。

```
typedef  DIMENSIONS
begin
  SHORT sLength;
  SHORT sWidth;
end;
DIMENSIONS Table;
NUMBER   nvNumValue;
DIMENSIONS  POINTER    pointerObject;
begin
  pointerObject = &Table;
  pointerObject->sLength = 500;
  pointerObject->sWidth = 750;
```

***注意**・ひとつの式で構造ポインターは 1 つだけ利用することができます。構造 A が構造 B へのポインターであるメンバー (Bptr) を含み、構造 B が構造 C へのポインターであるメンバー (Cptr) を含む場合、C のメンバーを A から参照することはできません。式 A.Btptr-Cptr-Cmember は InstallScript では無効です。*

文字列検索演算子 (%)

ブール型の文字列検索演算子 (%) を利用して、ある文字列が別の文字列のサブ文字列であるかどうかを判断します。次の例は、szStringVarA をテストして、文字列 “sample” が含まれているかどうかを判別します。含まれている場合は、MessageBox が呼び出されてメッセージが表示されます。

```
szStringVarA = " これは、サンプル文字列です。";
   if (szStringVarA % "sample") then
   MessageBox(" 変数は 'string' を含みます。",INFORMATION);
endif;
```

文字比較では大文字と小文字を区別しません。次の例では、メッセージボックスが表示されます。

```
typedef  DIMENSIONS
begin
  SHORT sLength;
  SHORT sWidth;
end;

DIMENSIONS Table;
NUMBER   nvNumValue;
DIMENSIONS  POINTER    pointerObject;

begin
  pointerObject = &Table;
```

```
pointerObject->sLength = 500;
pointerObject->sWidth = 750;
```

メモ・*InstallScript 関数 StrFind はまた、サブ文字列が別の文字列に含まれているかどうかも判断します。サブ文字列が検出されると、StrFind は文字列内にその位置を戻します。*

オブジェクトおよびオブジェクト ハンドラー

このセクションでは、InstallScript がサポートするオブジェクトとオブジェクト ハンドラーについて説明します。

オブジェクト

InstallScript は、次のようなオブジェクトをサポートしています。

テーブル 1 ・ InstallScript オブジェクト

オブジェクト	説明
Err オブジェクト	例外処理に使用します。
Objects オブジェクト	インデックスまたは名前を使ってプロジェクトの InstallShield オブジェクトにアクセス権を取得します。
Reboot オブジェクト	再起動後、インストールにコマンドラインを渡します。この情報は、インストールが再起動後実行されるようにインストールの終わりに適切なレジストリキーに書き込まれます。
TextSub オブジェクト	テキスト置換を作成します。

Err オブジェクト

Err オブジェクトは例外処理 に使用します。これには、次のプロパティとメソッドがあります。

プロパティ

テーブル 2 - Err オブジェクトのプロパティ

プロパティ	説明
数値	エラーを識別する番号。
ソース	エラーのソースを識別する文字列。
説明	エラーについて説明する文字列。
HelpFile	エラーについての追加情報が含まれるヘルプ ファイルの完全修飾ファイル名を指定する文字列。
HelpContext	エラーについての追加情報が含まれるヘルプトピックの識別子を指定する番号。
LastDllError	このプロパティは Windows API 関数 **GetLastError** の戻り値を含みます。

メソッド

テーブル 3 - Err オブジェクトのメソッド

メソッド	説明
Clear	Err オブジェクトのプロパティ値をクリアします。
Raise	try キーワード後に実行すると、スクリプトプロセスは、次の例外ハンドラー (catch/endcatch ブロック内のコード) へパスします。それ以外の場合は、プロセスはセットアップエンジンの組込み例外ハンドラーへパスします。また、Err オブジェクトプロパティの値を次のようにリセットします。 • Err.Raise( ); — NErr オブジェクトプロパティ値をリセットしません。 • Err.Raise(nNumber); — NErr.Number の値を nNumber にリセットします。 • Err.Raise(nNumber, szSource); — NErr.Number の値を nNumber に、Err.Source を szSource にリセットします。 • Err.Raise(nNumber, szSource, szDesc); — NErr.Number の値を nNumber に、Err.Source を szSource に、Err.Description を szDesc にリセットします。 • Err.Raise(nNumber, szSource, szDesc, szHelpFile); — NErr.Number の値を nNumber に、Err.Source を szSource に、Err.Description を szDesc に、Err.HelpFile を szHelpFile にリセットします。 • Err.Raise(nNumber, szSource, szDesc, szHelpFile, nHelpContext); — NErr.Number の値を nNumber に、Err.Source を szSource に、Err.Description を szDesc に、Err.HelpFile を szHelpFile に、Err.HelpContext を nHelpContext にリセットします。

Objects オブジェクト

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

Objects オブジェクトは、インデックスまたは名前を使ってプロジェクトの InstallShield オブジェクトにアクセスするのに使用されます。例 :

```
set obj1 = Objects(1);
set obj2 = Objects("New MFC 6.2 Runtime 1");
```

プロパティ

テーブル 4 · Objects オブジェクトのプロパティ

プロパティ	説明
Count	プロジェクトに含まれる InstallShield オブジェクトの数。

Reboot オブジェクト

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

Reboot オブジェクトは、再起動後にコマンドライン引数を **Setup.exe** に渡すのに使用されます。この情報は、システム再起動後にインストールが実行するように、インストールの最後に適切なレジストリ キーに書き込まれます。例えば、次のステートメントは再起動の後にデバッグ モードでインストールを実行します :

```
Reboot.CommandLine = "-d";
```

指定されたひとつまたは複数の引数は既存のコマンドラインに追加される点にご注意ください。既存のコマンドラインは置換されません。また、現時点では既存のコマンドラインを変更したり、そこからテキストを削除したりすることはできません。

InstallScript MSI インストールで InstallScript 変数 CMDLINE へパラメーターを渡します。

その他の **Setup.exe** コマンド ラインと同様に、InstallScript MSI インストールでは、再起動後に **Setup.exe** ファイルが起動されるときに情報を InstallScript 変数 CMDLINE に渡す場合、-z オプションを指定した後に情報を含める必要があります。たとえば、次のコマンド ラインは **Setup.exe** ファイルがシステム再起動の後に起動するときに、reboot 文字列、つまり CMDLINE に **TEST1 TEST2** を追加します。

```
Reboot.CommandLine = -z"TEST1 TEST2"
```

複数の -z パラメーターの使用はサポートされていないため、InstallScript MSI インストールで CMDLINE 変数のためのすべての情報は、単一の Reboot.CommandLine 呼び出しで指定しなくてはなりません。

TextSub オブジェクト

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

TextSub オブジェクトは、テキスト置換を行うときに使用されます。これには、次のプロパティとメソッドがあります。

プロパティ

テーブル 5 ・ TextSub オブジェクトのプロパティ

プロパティ	説明
Value(szIdentifier)	識別子 szIdentifier と関連付けられた文字列値。

メソッド

テーブル 6 ・ TextSub オブジェクトのメソッド

メソッド	説明
Substitute(szString)	szString の中のすべてのかぎ括弧付き識別子を、それと関連付けられた文字列に置換します。

例

以下のコードは、

```
TextSub.Value( "SUBBED" ) = " 代替テキスト ";
szString = "123<SUBBED>456<UNSUBBED>789";
TextSub.Substitute( szString );
```

gives szString に値 "123substituted text456UNSUBBED789" を与えます。

オブジェクト ハンドラー

InstallScript は、次のようなオブジェクトハンドラーをサポートしています。

テーブル 7 ・ InstallScript オブジェクト ハンドラー

オブジェクト ハンドラー	説明
InitProperties	プロジェクトにオブジェクトが挿入されたときに実行されます。これにより、オブジェクトプロパティの値を読み取りまたは書き込むために使用されるオブジェクトスクリプトの変数の値が初期化されます。
ReadProperties	オブジェクトプロジェクトが開かれたときと、オブジェクトを含むインストールが起動された時に実行されます。これにより、適切な Read*xxxx*Property 関数が呼び出され、プロパティバッグオブジェクトに保存されているプロパティ値が取得されます

テーブル 7 · InstallScript オブジェクト ハンドラー (続き)

オブジェクト ハンドラー	説明
WriteProperties	オブジェクトプロジェクトが保存またはビルドされたときに実行されます。このハンドラーは、適切な Write*xxxx*Property 関数を呼び出し、プロパティの値をプロパティバッグオブジェクトに保存します

InitProperties

***プロジェクト ·** この情報は、InstallScript プロジェクトに適用します。*

InitProperties ハンドラーは、プロジェクトにオブジェクトが挿入される際に実行されます。これにより、オブジェクトプロパティの値を読み取りまたは書き込むために使用されるオブジェクトスクリプトの変数の値が初期化されます。

[新しいプロパティの追加] ダイアログ ボックスを使用して、オブジェクトプロジェクトにプロパティを追加すると、ダイアログの [既定値] 編集ボックスのエントリに基づいた適切なステートメントが、このハンドラーに自動的に配置されます。

ReadProperties

***プロジェクト ·** この情報は、InstallScript プロジェクトに適用します。*

ReadProperties ハンドラーは、オブジェクトプロジェクトが開かれる際と、オブジェクトを含むセットアップが起動される際に実行されます。これにより、適切な Read*xxxx*Property 関数が呼び出され、プロパティバッグオブジェクトに保存されているプロパティ値が取得されます (プロパティ値は、WriteProperties ハンドラー によって呼び出される Write*xxxx*Property 関数によって、プロパティバッグオブジェクトに保存されます)。

[新しいプロパティの追加] ダイアログを使用して、オブジェクトプロジェクトにプロパティを追加すると、適切な Read*xxxx*Property 関数がこのハンドラーに自動的に配置されます。

WriteProperties

***プロジェクト ·** この情報は、InstallScript プロジェクトに適用します。*

WriteProperties ハンドラーは、オブジェクトプロジェクトが保存またはビルドされた時に実行されます。このハンドラーは、適切な Write*xxxx*Property 関数を呼び出し、プロパティの値をプロパティバッグオブジェクトに保存します (プロパティ値は、ReadProperties ハンドラー によって呼び出される Read*xxxx*Property 関数によって、プロパティバッグオブジェクトから取得されます)。

[新しいプロパティの追加] ダイアログを使用して、オブジェクトプロジェクトにプロパティを追加すると、適切な Write*xxxx*Property 関数がこのハンドラーに自動的に配置されます。

例外処理

例外処理を行うことで、その他のスクリプトコードからエラー処理を分離します。InstallScript は Err オブジェクトとキーワード try、catch、並びに endcatch を利用して例外処理をサポートします。

キーワード try に続くコードを実行中に例外が発生した場合、スクリプト プロセスは次の例外ハンドラー (つまり、catch/endcatch ブロックの内部のコード) へパスします。例外ハンドラーが実行されると、プロセスはその endcatch キーワードの次の行へパスします。try キーワードに続くコードによって例外が発生しなかった場合、例外ハンドラーのコードは省略され、プロセスは endcatch キーワードの次の行から再開します。

例外は Err オブジェクトの Raise メソッドの呼び出しで発生させることも可能です。これには 0 から 5 の引数が含まれます。対応する Err オブジェクトプロパティの値をチェックすることで、例外ハンドラーにあるこれらの引数の値を読み出すこともできます。

メモ・発生したたエラーは必ず負の数字でなくてはなりません。そうでない場合、インストールエンジンがエラーを適切な COM エラーに変換します。ですから、投入されるエラーコードはすべて負の数字でなくてはなりません。

次のコードサンプルは例外処理の実例です。

```
askfile:
AskText( " ファイルへのパス ?", "", svPathName );

try
   if (!Is( FILE_EXISTS, svPathName )) then
      /* ファイルが存在しない場合、例外を発生する。
        (ERR_NOT_EXIST に #define ステートメントの値が
        提供されていなければなりません。エラー番号は
        負の数字でなくてはなりません。)*/
      Err.Raise( ERR_NOT_EXIST );
   endif;

   if GetFileInfo ( svPathName, FILE_SIZE,
      nvFileSize, svResult )<0 then
      /* ファイル情報が得られなかった場合、
        例外を発生する。(ERR_NO_INFO に #define ステートメントの値が
        提供されたはずです。エラー
        番号は負の数字でなくてはなりません。)*/
      Err.Raise ( ERR_NO_INFO );
   endif;

   SprintfBox ( INFORMATION, " ファイル サイズ ", "%s のサイズは %ld です。",
      svPathName, nvFileSize );
catch
   /* 例外ハンドラー。*/
   nTemp = Err.Number;
   /* その原因に基づいて例外を処理。*/
   switch (nTemp)
      case ERR_NOT_EXIST:
         if AskYesNo( svPathName +
            " が存在しません。別のパスを入力しますか ?", YES )=YES then
            bTryAgain = TRUE;
         endif;
      case ERR_NO_INFO:
         MessageBox ( " 次のサイズを取得できませんでした " +
```

```
            svPathName, INFORMATION );
        bTryAgain = FALSE;
    endswitch;
endcatch;

if bTryAgain then
    bTryAgain = FALSE;
    goto askfile;
endif;
```

Try/catch/endcatch ブロックは次の要領でネストすることができます。

```
try
    /* 通常プロセス、パート 1。*/
    try
        /* 通常プロセス、パート 2。*/
    catch
        /* パート 2 の例外処理。*/
    endcatch;
    /* 通常プロセス、パート 3。*/
catch
    /* パート 1、パート 3 の例外処理。*/
endcatch;
```

ビルトイン関数 (A–D)

カテゴリ別の関数一覧は、「カテゴリー別ビルトイン関数」を参照してください。

AddFolderIcon

CreateShortcut 関数は **AddFolderIcon** 関数に優先します。

AddFolderIcon 関数を使って、以下のようなタスクを処理することができます :

- [スタート] メニュー、[プログラム] メニュー、またはデスクトップ上にショートカットまたはフォルダーを作成します。szProgramFolder パラメーターを使って、ショートカットまたはフォルダーの適切な場所を指定します。
- [スタート] メニュー上に複数階層のサブメニューを作成して、そのサブメニューにショートカットを含みます。

***メモ・AddFolderIcon** を呼び出すためには、そのショートカット ターゲットが既にターゲット システム上に存在している必要があります。*

***AddFolderIcon** はインターネット ショートカットの作成をサポートしません。*

構文

AddFolderIcon (szProgramFolder, szItemName, szCommandLine, szWorkingDir, szIconPath, nIcon, szShortCutKey, nFlag);

パラメーター

テーブル 1 ・ AddFolderIcon のパラメーター

パラメーター	説明
szProgramFolder	ショートカットを含めるフォルダーの名前を指定するか、作成するプログラム フォルダーの名前を指定します。フォルダーが存在しない場合は、インストーラーによって作成されます。このパラメーターに、複数レベルの階層メニューでサブフォルダーを指定できます。サブフォルダーが存在しない場合、**AddFolderIcon** がサブフォルダーを作成し、必要に応じて親フォルダーも作成します。 ショートカットを特定のフォルダーに追加する場合、以下のような完全修飾パスを指定します : **C:¥¥ProgramData¥¥Microsoft¥¥Windows¥¥Start Menu¥¥Programs** [スタート] メニューにある [プログラム] メニューにショートカットアイコンを追加するには、このパラメーターにヌル文字列 ("") を渡します。 次の InstallScript システム変数の 1 つをこのパラメーターで渡すことができます : ・ FOLDER_DESKTOP – デスクトップにショートカットを追加します。 ・ FOLDER_STARTUP – スタートアップ メニューにショートカットを追加します。 ・ FOLDER_STARTMENU – スタート メニューにショートカットを追加します。 ・ FOLDER_PROGRAMS – スタート ¥ プログラム メニューにショートカットを追加します。 InstallScript システム変数によって識別されるフォルダーの相対パスを指定することもできます。例 : **FOLDER_PROGRAMS ^ "ACCESSORIES¥¥GAMES"**
szItemName	ショートカット名を指定します。**AddFolderIcon** を呼び出してショートカットをプログラム フォルダーに追加すると、szCommandLine で指定されたリンク ディレクトリにリンクファイルも作成されます。エクスプローラー シェルでは、項目名に「/、¥、:、?、<、>、または |」を使用できません。

テーブル 1 ・ AddFolderIcon のパラメーター (続き)

パラメーター	説明
szCommandLine	次のうちの 1 つを指定します : ・ コマンドライン パラメーターをすべて含む、ショートカットに関連付けられた実行可能ファイルの完全修飾名。これは、ショートカットの [プロパティ] ダイアログ ボックスにある " リンク先 " 値に追加されます。Start Programs メニューにショートカットを追加するには、リンク ディレクトリの完全修飾パスを入力します。ここにアプリケーションがそのアイコン リンク ファイルを格納します。 ・ szItemName がサブフォルダーの場合、完全修飾パス。  ***注意 ・ コマンドラインに長いファイル名が含まれる場合は、引用符で囲む必要があります。しかし、コマンドラインパラメーターは引用符で囲んではなりません。そのため、szCommandLine 文字列を 2 つの個別の文字列から構築することをお勧めします。***
szWorkingDir	このショートカット ターゲットの作業ディレクトリを入力します。 szItemName がサブフォルダーの場合、このパラメーターは適用しません。 **AddFolderIcon** は、ショートカットの [プロパティ] ダイアログ ボックスの [ショートカット] タブにある [作業フォルダー] ボックスにこのディレクトリを書き込みます。ヌル文字列 ("") をこのパラメーターに渡すと、関数はこの [作業フォルダー] ボックスを空白のままにして、[リンク先] ボックスのパスが使用されます。  ***注意 ・ LongPathToQuote を呼び出してこのパスを引用符で囲まないようにしてください。InstallShield はこれらのパスも自動的に引用符で囲みます。***
szIconPath	ショートカットに表示するアイコンを含むファイルへの完全修飾パスを指定します。 szItemName がサブフォルダーの場合、このパラメーターは適用しません。  ***注意 ・ LongPathToQuote を呼び出してこのパスを引用符で囲まないようにしてください。InstallShield はこれらのパスも自動的に引用符で囲みます。***
nIcon	szIconPath で指定された実行可能ファイルに含まれるアイコン インデックスを指定します。インデックス 0 はファイルの最初のアイコンを、インデックス 1 は 2 番目のアイコンを意味します。以降の番号も同じように続きます。アイコンを使用しない場合、このパラメーターに **0** を指定します。 szItemName がサブフォルダーの場合、このパラメーターは適用しません。

テーブル 1 · AddFolderIcon のパラメーター (続き)

パラメーター	説明
szShortCutKey	ショートカットに割り当てるショートカット キーを文字列の形式で指定します。ショートカットに szShortCutKey を設定すると、エンド ユーザーが適切なホット キーを押してショートカットを起動することができます。 たとえば、エンド ユーザーが Ctrl と Alt キーを押しながら数字の 1 を押すと製品が起動するように設定する場合は、このパラメーターに "Ctrl+Alt+1" を渡します。 szItemName がサブフォルダーの場合、このパラメーターは適用しません。
nFlag	このパラメーターには、次の定義済み定数のいずれかを指定します。このパラメーターに複数の定義済み定数を渡す場合、各定数をビット単位 OR 演算子 (|) で区切ってください。 · REPLACE – フォルダー内の現在のアイコンまたはショートカットを置換します。 · RUN_MAXIMIZED – プログラムの起動時に最大化されることを示します。 · RUN_MINIMIZED – プログラムの起動時に最小化されることを示します。 · NULL – オプションがないことを示します。

戻り値

テーブル 2 · AddFolderIcon の戻り値

戻り値	説明
0	関数が指定されたフォルダー内のショートカットを追加または置換し、実行可能ファイルに関連付けたことを示します。
< 0	関数がショートカットの追加または置換、また実行可能ファイルとの関連付けに失敗したことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

AddFolderIcon の例

次の例から 1 つ選択します。

- スタート メニューとスタート プログラム メニューにある実行可能ファイルへショートカットを配置する。(AddFolderIcon の例 1)
- Startup メニューに複数階層のサブ メニューを作成して、そのメニューにショートカットを追加する。(AddFolderIcon の例 2)
- デスクトップ上にサブフォルダー、並びに新規フォルダーの実行可能ファイルをポイントするショートカットを配置する。(AddFolderIcon の例 3)

AddFolderIcon の例 1

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* AddFolderIcon 関数のデモンストレーションを行います。
*
* この例ではスタートメニューとスタートプログラムメニューにある
* 実行可能ファイルへショートカットを配置します。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*      ターゲットシステム上の Windows Notepad 実行可能ファイルの
*      完全修飾名と有効なテキスト ファイルを
*      参照するように設定してください。
*
¥*---------------------------------------------------------*/

#define PROGRAM "C:¥¥Windows¥¥Notepad.exe"
#define PARAM   "C:¥¥Windows¥¥Readme.txt"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_AddFolderIcon(HWND);

function ExFn_AddFolderIcon(hMSI)
    STRING szProgramFolder, szItemName, szCommandLine, szWorkingDir;
    STRING szShortCutKey, szProgram, szParam, szIconPath;
    NUMBER nIcon;
begin

    // AddFolderIcon を呼び出すパラメーターをセットアップします。
    szProgramFolder = FOLDER_STARTMENU;
    szItemName= "Notepad の例 1";
    szProgram       = PROGRAM;
    szParam         = PARAM;

    LongPathToQuote (szProgram, TRUE);

    LongPathToShortPath (szParam);

    szCommandLine = szProgram + " " + szParam;
    szWorkingDir  = "";
    szIconPath    = "";
    nIcon         = 0;
    szShortCutKey = "";

    // スタート メニューへショートカットを追加します。
    if (AddFolderIcon (szProgramFolder, szItemName, szCommandLine, szWorkingDir,
                       szIconPath, nIcon, szShortCutKey, REPLACE) < 0) then
        MessageBox("AddFolderIcon が失敗しました。", SEVERE);
```

```
    else
        SprintfBox (INFORMATION, "AddFolderIcon", "%s 無事に作成されました。",
                szItemName);
    endif;

    szProgramFolder = "";
    szItemName= "Notepad の例 2";

    // プログラム メニューへショートカットを追加します。
    if (AddFolderIcon (szProgramFolder, szItemName, szCommandLine, szWorkingDir,
                    szIconPath, nIcon, szShortCutKey, REPLACE) < 0) then
        MessageBox("AddFolderIcon が失敗しました。", SEVERE);
    else
        SprintfBox (INFORMATION, "AddFolderIcon", "%s 無事に作成されました。",
                szItemName);
    endif;

end;
```

AddFolderIcon の例 2

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * AddFolderIcon 関数のデモンストレーションを行います。
 *
 * この例では、Startup メニューに複数階層のサブメニューを作成し、
 * そこに実行可能ファイルへのショートカットを追加します。
 *
 * メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
 *      ターゲットシステム上の Windows Notepad 実行可能ファイルの
 *      完全修飾名と有効なテキスト ファイルを適切に
 *      参照するように設定してください。
 *
¥*---------------------------------------------------------------*/

#define PROGRAM "C:¥¥Windows¥¥Notepad.exe"
#define PARAM   "C:¥¥Windows¥¥Readme.txt"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_AddFolderIcon(HWND);

function ExFn_AddFolderIcon(hMSI)
    STRING  szProgramFolder, szItemName, szCommandLine, szWorkingDir;
    STRING  szIconPath, szShortCutKey, szProgram, szParam;
    NUMBER  nIcon, nFlag, nResult;
begin

    // Startup サブメニューの完全修飾名を設定します。
```

```
    szProgramFolder = FOLDER_STARTUP ^ "SubMenu の例 ";

    // ショートカットのコマンドライン プロパティを構築します。
    szProgram= PROGRAM;
    szParam    = PARAM;

    LongPathToQuote (szProgram, TRUE);

    LongPathToShortPath (szParam);

    szCommandLine = szProgram + " " + szParam;

    // AddFolderIcon へ渡すショートカットの他のプロパティを設定します。
    szItemName = "Notepad の例 1";
    szWorkingDir  = "";
    szIconPath    = "";
    nIcon         = 0;
    szShortCutKey = "";
    nFlag         = REPLACE|RUN_MAXIMIZED;

    // サブメニューにショートカットを追加、および必要に応じてサブメニューを作成します。
    nResult = AddFolderIcon (szProgramFolder, szItemName, szCommandLine,
                             szWorkingDir, szIconPath, nIcon,
                             szShortCutKey, nFlag);

    // 結果をレポートします。
    if (nResult < 0) then
       MessageBox("AddFolderIcon が失敗しました。", SEVERE);
    else
       SprintfBox (INFORMATION, "AddFolderIcon", "%s 無事に作成されました。",
                szItemName);
    endif;

end;
```

AddFolderIcon の例 3

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* AddFolderIcon 関数のデモンストレーションを行います。
*
* この例ではデスクトップ上にサブフォルダー、並びに新規フォルダーの
* 実行可能ファイルを指定するアイコンを配置します。フォルダーは、
* 実際のディレクトリをポイントするショートカットです。このフォルダーから
* ユーザーはプログラムを実行するショートカットを利用できます。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*      ターゲットシステム上の Windows Notepad 実行可能ファイルの
*      完全修飾名と有効なテキスト ファイルを
*      参照するように設定してください。
```

```
 *
¥*---------------------------------------------------------------*/

#define FOLDER "C:¥¥Windows¥¥"
#define PROGRAM "C:¥¥Windows¥¥Notepad.exe"
#define PARAM "C:¥¥Windows¥¥Readme.txt"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_AddFolderIcon(HWND);

function ExFn_AddFolderIcon(hMSI)
  STRING  szProgramFolder, szItemName, szCommandLine, szWorkingDir;
   STRING  szIconPath, szShortCutKey;
   STRINGszProgram, szParam, szFolderDir;
   NUMBER  nIcon, nFlag, nResult;
begin

  // szProgramFolder はローカル システム上のデスクトップです。
  szProgramFolder = FOLDER_DESKTOP;
  szItemName= " フォルダー例 ";

  // フォルダー アイコンが指定するフォルダーを作成します。
  szFolderDir = FOLDER ^ szItemName;
  CreateDir(szFolderDir);

  // フォルダー アイコンのコマンドラインはフォルダー パスでなくてはなりません。
  // また、パスが 8 文字以上である場合、
  // 引用符で囲む必要があります。

  szCommandLine = szFolderDir;
  LongPathToQuote(szCommandLine, TRUE);

  szWorkingDir  = "";
  szIconPath    = "";
  nIcon         = 0;
  szShortCutKey = "";
  nFlag = REPLACE|RUN_MINIMIZED;

  // フォルダー アイコンを作成し、そのアイコンが指定する目的のフォルダーを表示します。
  nResult = AddFolderIcon (szProgramFolder, szItemName, szCommandLine,
                           szWorkingDir, szIconPath, nIcon, szShortCutKey,
                           nFlag);

  if (nResult < 0) then
    MessageBox("AddFolderIcon が失敗しました。", SEVERE);
  else
    SprintfBox (INFORMATION, "AddFolderIcon", "%s 無事に作成されました。",
                szItemName);
  endif;

  // 作成したフォルダーを表示する。
  ShowProgramFolder (szFolderDir, SW_SHOW);

  // Example アイコンを新しく作成したフォルダーに追加します。
  szProgramFolder = szFolderDir;
  szItemName      = " メモ帳の例 ";
```

```
  // 空白スペースが区切り文字として間違えられないよう注意してください。
  szProgram       = PROGRAM;
  LongPathToQuote (szProgram, TRUE);

  szParam= PARAM;
  LongPathToShortPath (szParam);

  szCommandLine = szProgram + " " + szParam;
  szWorkingDir  = "";
  szIconPath    = "";
  nResult = AddFolderIcon (szProgramFolder, szItemName, szCommandLine,
                           szWorkingDir, szIconPath, nIcon, szShortCutKey,
                           nFlag);

  if (nResult < 0) then
    MessageBox("AddFolderIcon が失敗しました。", SEVERE);
  else
    SprintfBox (INFORMATION, "AddFolderIcon", "%s 無事に作成されました。",
                szItemName);
  endif;

end;
```

AddProfString

AddProfString 関数は条件無しで .ini ファイルへプロファイル文字列を追加します。AddProfString は、System.ini ファイル (device = ...) の [386Enh] セクションにあるような非固有キー の追加のみに利用します。AddProfString は指定した .ini ファイルセクションの終わりに KEY=VALUE ラインを追加します。既存キーの置換またはアップデートは行いません。既存の非固有キーをアップデートするには、ReplaceProfString を呼び出します。.ini ファイルで固有キーを追加、または既存する固有キーの値をアップデートするには、WriteProfString を呼び出します。

構文

```
AddProfString ( szFileName, szSectionName, szKeyName, szValue );
```

パラメーター

テーブル 3 - AddProfString のパラメーター

パラメーター	説明
szFileName	プロファイル文字列を追加する .ini ファイル名を指定します。szFileName が完全修飾名ではない場合 (ドライブ指定とパスが含まれていない場合)、InstallShield は Windows フォルダーのファイルを検索します。ファイルが存在しないと、指定のフォルダーに作成されます。パスがファイル名に含まれていない場合、ファイルは Windows フォルダーに作成されます。ファイル名に存在しないパスが含まれている場合、AddProfString は失敗します。
szSectionName	.ini ファイルセクションのセクション名を指定します。プロファイル文字列はそのセクションの最後に挿入されます。セクションが存在しない場合は、InstallShield によって作成されます。ここで指定するセクション名は、角括弧 ([ ]) で*囲まないでください*。szKeyName で指定したキーがセクションに既に存在する場合でも、プロファイル文字列が挿入されることに注意してください。
szKeyName	置換するキーの名前を指定します。このパラメーターの値は、プロファイル文字列の等号の左側 (szKeyName = szValue) に表示されます。
szValue	キーに割り当てる値を指定します。このパラメーターの値は、プロファイル文字列の等号の右側 (szKeyName = szValue) に表示されます。

戻り値

テーブル 4 - AddProfString の戻り値

戻り値	説明
0	AddProfString は .ini ファイルへ指定されたプロファイル文字列を無事に追加しました。
< 0	AddProfString はプロファイル文字列を追加できませんでした。

追加情報

- AddProfString は .ini ファイルを変更するのに Windows API を利用しません。Windows API は AddProfString で実行可能な変更を行うことができません。

- .ini ファイルに加えた変更は、アンインストール用にログ記録することができます。ただし、いくつかの重要な制限事項があります。詳細は、「 初期設定 (.ini) ファイルエントリのアンインストール」を参照してください。

AddProfString の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * 関数 AddProfString と GetProfString をデモンストレーションします。
 *
 * このスクリプトは、ファイルにプロファイル文字列を追加します。
 * 追加された文字列を読み出して表示します。
 *
 * メモ : このスクリプトを初めて実行した時、ドライブ C のルートに
 *      ISExampl.sys と名づけられた構成ファイルを作成します。
 *      このファイルはスクリプト解析が終了した時点で
 *      削除することができます。
 *
¥*---------------------------------------------------------*/

#define EXAMPLE_INI "C:¥¥ISExampl.ini"

// ファイルに追加する新しいセクション、キー、および値。
#define NEW_SECTION " 新しいセクション "
#define NEW_KEY " 新しいキー "
#define NEW_VALUE " テスト "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_AddProfString(HWND);

function ExFn_AddProfString(hMSI)
   STRING svResult;
begin

   // ファイルへプロファイル文字列を追加します。
   if (AddProfString (EXAMPLE_INI, NEW_SECTION, NEW_KEY, NEW_VALUE) != 0) then
      // 文字列を追加できない場合は、エラー メッセージを表示します。
      MessageBox ("AddProfString が失敗しました。", SEVERE);
   else
      // ファイルからキーの値を読み出します。
      if (GetProfString (EXAMPLE_INI, NEW_SECTION, NEW_KEY, svResult) != 0) then
         // 文字列読み出せない場合は、エラー メッセージを表示します。
         MessageBox ("GetProfString が失敗しました。", SEVERE);
      else
         // キーとその現在の値を表示します。
         MessageBox (NEW_KEY + "=" + svResult, INFORMATION);
```

```
        endif;
    endif;

end;
```

AdminAskPath

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

AdminAskPath 関数は、管理インストール時 (エンドユーザーが InstallScript MSI プロジェクト /a 引数を使った **Setup.exe** を実行する場合) にエンドユーザーに対してインストール先パスの入力を求めるダイアログを表示します。

構文

```
AdminAskPath ( szMsg, szDefaultPath, svResultPath );
```

パラメーター

テーブル 5 - AdminAskPath のパラメーター

パラメーター	説明
szMsg	このダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szDefaultPath	編集フィールドに表示するデフォルトのパスを指定します。エンドユーザーはこの文字列を変更できます。OnAdminInstallUIBefore のデフォルト実装では、このパラメーターに INSTALLDIR を渡します。
svResultPath	ユーザーがデフォルトのパスを利用、修正、または [フォルダーの選択] ダイアログから代替パスを選択するか否かに関わらず、結果のパス名を戻します。OnAdminInstallUIBefore のデフォルト実装では、このパラメーターに INSTALLDIR を渡します。

戻り値

テーブル 6 - AdminAskPath の戻り値

戻り値	説明
NEXT (1)	ユーザーが [次へ] ボタンをクリックしたことを示します。
BACK (12)	ユーザーが、[戻る] ボタンをクリックしたことを示します。

追加情報

AdminAskPath は **AskPath** ダイアログを使用し、また **AskPath** と同じダイアログ リソースを使用します。つまり、[ダイアログ エディター] の **AskPath** のレイアウトで行った変更はすべて **AdminAskPath** に反映されます。

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

AdminAskPath の例

***プロジェクト -** この情報は、InstallScript MSI プロジェクトに適用します。*

```
//---------------------------------------------------------------------------
//
// InstallShield スクリプトの例
//
//  AdminAskPath 関数のデモンストレーションを行います。
//
//  InstallScript MSI プロジェクトの OnAdminInstallUIBefore イベントで
//  ファイルを圧縮およびコピーする先のターゲット ディレクトリを
```

```
// ユーザーにプロンプトします。
//
//---------------------------------------------------------------------------

function OnAdminInstallUIBefore( )
    int nResult;

begin
Dlg_SdWelcome:
    SdWelcome("", "");

Dlg_AdminAskPath:
    // ユーザーにターゲット パスをプロンプトして、それを INSTALLDIR に格納します
    nResult = AdminAskPath("", INSTALLDIR, INSTALLDIR);
    if (nResult = BACK) goto Dlg_SdWelcome;

    // ステータス ダイアログを準備します
    SetStatusExStaticText( SdLoadString( IDS_IFX_STATUSEX_STATICTEXT_FIRSTUI ) );
    Enable(STATUSEX);
end;
```

AskDestPath

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

AskDestPath 関数は、ダイアログを表示します。このダイアログで、エンドユーザーはインストールするファイルのインストール先フォルダーを指定することができます。また、このダイアログには [参照] ボタンがあり、エンド ユーザーはシステム上の既存フォルダーを選択することができます。

[インストール先の選択] ダイアログから [フォルダーの選択] ダイアログを開くには、エンドユーザーは、[参照] ボタンをクリックする必要があります。[フォルダーの選択] ダイアログは、使用可能なフォルダーをすべてリスト表示します。エンドユーザーは既存のフォルダーから選択するか、新しくフォルダー名を入力できます。エンドユーザーが存在しないフォルダーの名前を入力した場合、そのフォルダーを新規作成する為のメッセージ ボックスが開きます。

メモ・AskDestPath を呼び出す前に新規フォルダーが存在しない場合、サイレントモードで実行するインストールは新規フォルダーを作成します。これによって、確認ダイアログは表示されません。このステップを踏まない場合は、2 つの条件を処理するために 2 つの応答ファイルが必要です。

エンドユーザーによって選択されたフォルダーは、書き込み可能であることが必要です。書き込み不可フォルダーは選択できません。エンドユーザーが書き込み不可能なファイルを選択できるようにするには、AskPath 関数を代わりに呼び出します。

構文

```
AskDestPath ( szTitle, szMsg, svDir, nReserved );
```

パラメーター

テーブル 7 ・ AskDestPath のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「インストール先の選択」を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このパラメーターに複数行のスタティックテキストを入力するには、改行する際に ¥n のエスケープシーケンスを挿入します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
svDir	ダイアログが開かれた際に表示するデフォルトのパスを指定します。エンドユーザーが選択したフォルダーへのパスを戻します。 ***メモ ・** svDir によって指定されたデフォルトのフォルダーがエンドユーザーのシステムに存在しない場合、エンドユーザーが [参照] ボタンをクリックして、[フォルダーの選択] ダイアログのステップに従ってフォルダーを作成しない限り、フォルダーは作成されません。したがって、必要に応じてフォルダーを作成する FeatureMoveData を呼び出す前に使用する予定のデフォルトフォルダーを指定するときは何時でも、フォルダーが存在するかどうかを判別するために AskDestPath が戻されるとき ExistsDir を呼び出す必要があります。フォルダーが存在しない場合、CreateDir を呼び出して、エンドユーザーのシステムでそのフォルダーを作成します。イベント指向のスクリプトを実行するインストールでは FeatureTransferData が自動的に呼び出されることに注意してください。*
nReserved	このパラメーターの値は 0 (ゼロ) でなくてはいけません。

戻り値

テーブル 8 ・ AskDestPath の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

AskDestPath の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * AskDestPath 関数のデモンストレーションを行います。
 *
 *  このスクリプトは AskDestPath を呼び出して、インストールがファイルを
 *  インストールする場所へのパスを取得します。そして、そのパスが
 * メッセージボックスに表示されます。
 *
¥*---------------------------------------------------------------*/

#define TITLE_TEXT"AskDestPath の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_AskDestPath(HWND);

function ExFn_AskDestPath(hMSI)
   STRING  szTitle, szMsg, svDir;
   NUMBER  nReturn;
begin

   // インストール用のデフォルトのパスを設定します。
   svDir = INSTALLDIR;

   // インストール先のディレクトリを取得します。ヌル文字列を 2 番目のパラメーターで渡し、
   // デフォルトのメッセージを表示します。
   nReturn = AskDestPath (TITLE_TEXT, "", svDir, 0);

   if (nReturn < 0) then
      // エラーを報告します。
      MessageBox ("AskDestPath に失敗。", SEVERE);
   elseif (nReturn = NEXT) then
      // 選択されたインストール先ディレクトリ名を表示します。
      MessageBox (svDir + " を選択しました。", INFORMATION);
   endif;

end;
```

AskOptions

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*

- *InstallScript MSI*

AskOptions 関数は、エンド ユーザーが 1 つまたは複数のオプションを選択するためのダイアログをフォーマットして表示します。ダイアログは、nValue の値に応じて、チェック ボックスやオプションボタンなど、最大 9 種類のコントロール選択肢を表示します。

このダイアログのデフォルトのタイトルは、「機能の選択」です。タイトルバーの内容を変更するには、AskOptions の前に SetDialogTitle を呼び出してください。

***メモ・** AskOptions 関数と一緒に PlaceWindow 関数を使うことはできません。背景ウィンドウモードを有効にしない限り、デフォルトではデスクトップ中央にダイアログが表示されます。インストールがウィンドウ モードの場合、背景ウィンドウの中央にダイアログが表示されます。*

構文

AskOptions (nValue, szMsg, szText1, bvCheck1, szText2, bvCheck2[, szTextn, bvCheckn] [,..., ...]);

パラメーター

テーブル 9 ・ AskOptions のパラメーター

パラメーター	説明
nValue	表示する制御のタイプを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ EXCLUSIVE ― オプションボタンを指定します。これによってエンドユーザーは 1 種類のオプションだけを選択することができます。 ・ NONEXCLUSIVE ― チェックボックスを指定します。これによってエンドユーザーは複数のオプションを選択することができます。
szMsg	ダイアログに表示するメッセージを指定します。このメッセージを使用して、オプションを記述したり、ユーザーが 1 つまたは複数のオプションを選択するかどうかを尋ねたりすることができます。メッセージが長すぎて 1 行に収まらないときは、ニューラインエスケープシーケンス ¥n を使用して改行を入れてください。 ***メモ・** 最大メッセージ文字列長はオペレーティング システムによって決定されます。デフォルトで、szMsg テキストの表示は、_Isres.dll の AskOptions ダイアログ リソースによって 2 行に制限されています。szMsg テキストで 2 行を超える行を表示するには、AskOptions ダイアログからカスタム ダイアログを作成します*
szText1	最初のチェック ボックスまたはオプションボタンの横に表示するテキストラベルを指定します。表示できる最大文字数はフォントによって異なります。指定された文字列がダイアログの静的テキスト フィールドに合うようにしてください。文字列が合わない場合、短くするか、SdAskOptions を呼び出してください。 アクセラレーターキーを作成するには、アンパサンド (&) を該当する文字の前に挿入してください。この文字は下線付きで表示され、その機能を示します。たとえば、カスタムのアクセラレータキーを Alt + C に設定するには " カスタム (&C)"、カスタムのアクセラレータキーを Alt + S に設定するには " カスタム (&S) " を渡します。
bvCheck1	ダイアログが開かれている際の最初のチェック ボックスまたはオプション ボタンの初期ステータスを指定します。ダイアログが閉じている際は、最初のチェック ボックスまたはオプションボタンのステータスを戻します。このパラメーターでは、次の定数をやりとりします : ・ TRUE ― 最初のチェック ボックスまたはオプション ボタンが選択されました。 ・ FALSE ― 最初のチェック ボックスまたはオプション ボタンが選択されていません。
szText2	2 番目ののチェック ボックスまたはオプションボタンの横に表示するテキストラベルを指定します。表示できる最大文字数はフォントによって異なります。指定された文字列がダイアログの静的テキスト フィールドに合うようにしてください。文字列が合わない場合、短くするか、SdAskOptions を呼び出してください。 szText1 で実行した時と同じ手順で、アクセラレータを作成します。

テーブル 9 ・ AskOptions のパラメーター (続き)

パラメーター	説明
bvCheck2	ダイアログが開かれている際の 2 つ目のチェック ボックスまたはオプション ボタンの初期ステータスを指定します。ダイアログが閉じている際は、2 つ目のチェック ボックスまたはオプション ボタンのステータスを戻します。このパラメーターでは、次の定数をやりとりします : • TRUE – 最初のチェック ボックスまたはオプション ボタンが選択されました。 • FALSE – 最初のチェック ボックスまたはオプション ボタンが選択されていません。 7 種類までの追加オプションを定義できます。各追加オプションは一対のパラメーター (AskOptions が戻った時のオプションのステータスを示すラベルと数値変数を決定付ける文字列パラメーター) が示します。オプションの初期ステータスを設定するには、AskOptions を呼び出す前に、TRUE または FALSE を数値変数に割り当てます。 ***メモ ・** nValue が EXCLUSIVE で複数のオプションの初期ステータスが TRUE の場合、AskOptions は、パラメーターリスト内で TRUE にセットされている最初のオプションをあらかじめ選択します。*

戻り値

テーブル 10 ・ AskOptions の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。制御のステータスは、個々の bvCheck 変数で返されます。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。制御のステータスは、個々の bvCheck 変数で返されます。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

AskOptions の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * AskOptions 関数のデモンストレーションを行います。
 *
 * AskOptions ダイアログが 2 回表示されます。まず最初に
 * チェック ボックスと共に表示され、次にオプションボタンと
 * 共に表示されます。この例では使用できるオプションの最大数 (9 つ )
 * を表示します。
 *
¥*---------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_AskOptions(HWND);

function ExFn_AskOptions(hMSI)
    STRING szMsg, szText1, szText2, szText3, szText4, szText5;
    STRING szText6, szText7, szText8, szText9;
    NUMBER nReturn, nValue,nvCheck1, nvCheck2, nvCheck3, nvCheck4;
    NUMBER nvCheck5, nvCheck6, nvCheck7, nvCheck8, nvCheck9;
begin

    szMsg = " 次のオプションから選択してください。";

    szText1 = " オプション 1";
    szText2 = " オプション 2";
    szText3 = " オプション 3";
    szText4 = " オプション 4";
    szText5 = " オプション 5";
    szText6 = " オプション 6";
    szText7 = " オプション 7";
    szText8 = " オプション 8";
    szText9 = " オプション 9";

    nvCheck1 = TRUE;
    nvCheck2 = FALSE;
    nvCheck3 = FALSE;
    nvCheck4 = FALSE;
    nvCheck5 = FALSE;
    nvCheck6 = FALSE;
    nvCheck7 = FALSE;
    nvCheck8 = FALSE;
    nvCheck9 = FALSE;

    // チェック ボックス (NONEXCLUSIVE) ダイアログを表示します。
    nValue = NONEXCLUSIVE;

    AskOptions (nValue, szMsg,
            szText1, nvCheck1,
            szText2, nvCheck2,
            szText3, nvCheck3,
            szText4, nvCheck4,
            szText5, nvCheck5,
            szText6, nvCheck6,
            szText7, nvCheck7,
            szText8, nvCheck8,
```

```
            szText9, nvCheck9);

    // オプション ボタン (EXCLUSIVE) ダイアログを表示します。
    nValue = EXCLUSIVE;
    AskOptions (nValue, szMsg,
            szText1, nvCheck1,
            szText2, nvCheck2,
            szText3, nvCheck3,
            szText4, nvCheck4,
            szText5, nvCheck5,
            szText6, nvCheck6,
            szText7, nvCheck7,
            szText8, nvCheck8,
            szText9, nvCheck9);

end;
```

AskPath

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

AskPath 関数は、エンドユーザーがインストール先のパスを入力するためのダイアログを表示します。このダイアログには、1 行の編集フィールドがあり、ここにデフォルトのパスを表示できます。エンドユーザーは、3 つのオプションから選択できます。

- デフォルトのパスを使用する。
- デフォルトのパスを編集する。
- [フォルダーの選択] ダイアログを表示してフォルダーを選択する。

[フォルダーの選択] ダイアログを開くには、エンドユーザーは [参照] ボタンをクリックしなくてはなりません。[フォルダーの選択] ダイアログは、使用可能なフォルダーをすべてリスト表示します。エンドユーザーは既存のフォルダーから選択するか、新しくフォルダー名を入力できます。エンドユーザーが存在しないフォルダー名を入力した場合は、フォルダーが新たに作成されます。

注意・AskPath は、エンドユーザーが入力したパスが存在するか否かを確認しません。AskPath を呼び出した後、CreateDir を呼び出して、パスを作成します。

メモ・AskPath 関数と一緒に PlaceWindow 関数を使うことはできません。背景ウィンドウモードを有効にされていない場合、デフォルトでダイアログはデスクトップの中央に表示されます。インストールがウィンドウ モードの場合、背景ウィンドウの中央にダイアログが表示されます。

ダイアログのデフォルト タイトルは、[インストール先の選択] です。このタイトルを変更するには、AskPath の前に、SetDialogTitle を呼び出してください。

AskPath 関数は、存在するけれども書き込み不可能なフォルダー名を受け入れます。エンドユーザーの書き込みフォルダーに対する選択を制限するには、AskDestPath 関数を代わりに呼び出します。

構文

AskPath (szMsg, szDefPath, svResultPath);

パラメーター

テーブル 11 ・ AskPath のパラメーター

パラメーター	説明
szMsg	このダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szDefPath	編集フィールドに表示するデフォルトのパスを指定します。エンドユーザーはこの文字列を変更できます。
svResultPath	ユーザーがデフォルトのパスを利用、修正、または [フォルダーの選択] ダイアログから代替パスを選択するか否かに関わらず、結果のパス名を戻します。AskPath は、svResultPath に配置する前に、パスの末尾に円記号を追加します。必要に応じて、AskPath が値を戻した後に StrRemoveLastSlash を呼び出して、この円記号を削除することができます。ユーザーが、[戻る] ボタンをクリックした場合、svResultPath の値は予測不能になります。したがって、szDefPath と svResultPath の両方に対して同じ変数を使用している場合、AskPath からの戻り値が BACK となる時の変数を必ず再初期化して下さい。 ***メモ・ダイアログで表示される編集フィールドはスクロールできるので、長い文字列にも対応します。編集フィールドに入力できる文字数は制限されていないので、svResultPath で渡される変数は明白なサイズを設定せずに宣言します。文字列変数が、ユーザーが入力したテキストを保存するには短かすぎる場合、この文字列は切り捨てられ、エラーメッセージが表示されます。なぜならこの関数は円記号とヌルターミネーターを文字列の末尾に追加するため、文字列のサイズは、ユーザーが入力するパスよりも 2 文字以上長くする必要があるからです。***

戻り値

テーブル 12 ・ AskPath の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンを選択したことを示します。SvResultPath は、ヌル文字列 ("") に設定されます。

追加情報

AdminAskPath は **AskPath** ダイアログを使用し、また **AskPath** と同じダイアログ リソースを使用します。つまり、[ダイアログ エディター] の **AskPath** のレイアウトで行った変更はすべて **AdminAskPath** に反映されます。

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

AskPath の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * AskPath 関数のデモンストレーションを行います。
 *
 * このスクリプトはエンドユーザーのコンピューターにある
 * フォルダーへのパスを取得します。パスが存在しない場合は、作成されます。
 * エンドユーザーによる指示があった場合は、その場所に
 * フォルダーが作成されます。最後に、選択されたパスが表示されます。
 *
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_AskPath(HWND);

function ExFn_AskPath(hMSI)
   STRING szMsg, svResultPath[101];
   BOOL bTargetDirOk;
begin

   // インストールダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   // AskPath ダイアログに表示するメッセージを作成します。
   szMsg = " アプリケーションのフォルダーを指定します。";

   // 有効なパスインジケーターを初期化します。
   bTargetDirOk = FALSE;

   repeat
      // ユーザーからパスを取得します。デフォルトのパスは次のとおりです。
      // システム変数 INSTALLDIR の現在の値です。
      if (AskPath (szMsg, INSTALLDIR, svResultPath) = NEXT) then
         // ターゲット システム上に
         // ユーザーが入力したパスが存在するか？
         if (ExistsDir (svResultPath) = 0) then
            // 存在する場合、ループを終了するようインジケーターを設定します。
            bTargetDirOk = TRUE;
         else
            // パスが存在しない場合、それを作成するべきか問い合わせます。
            if (AskYesNo (" フォルダーが存在しません。作成しますか ?",YES) = YES) then
```

```
            // フォルダー ( ディレクトリ ) を作成します。
            if (CreateDir (svResultPath) = 0) then
               // フォルダーが作成された場合、ループを終了するようインジケーターを設定します。
               bTargetDirOk = TRUE;
            else
               // フォルダーが作成されなかったことをエンドユーザーへ報告します。
               MessageBox (" 作成することができませんでした "+ svResultPath, WARNING);
            endif;
         endif;
      endif;
   endif;
until bTargetDirOk;

// ターゲット フォルダーの名前を表示します。
MessageBox (" ターゲット フォルダーは " + svResultPath, INFORMATION);

// あとに続くダイアログ用に [ 戻る ] ボタンを有効にすることもできます。
Enable(BACKBUTTON);

end;
```

AskText

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

AskText 関数は、スタティック テキスト フィールドと編集ボックスをそれぞれ 1 つ含むダイアログを表示します。パラメーター szQuestion のスタティックテキストフィールド用デフォルトのテキストを指定します。パラメーター szDefault の編集ボックス用デフォルトのテキストを指定します。

メモ・AskText 関数と一緒に PlaceWindow 関数を使うことはできません。背景ウィンドウモードを有効にしない限り、デフォルトではデスクトップ中央にダイアログが表示されます。インストールがウィンドウ モードの場合、背景ウィンドウの中央にダイアログが表示されます。

このダイアログのデフォルトのタイトルは、[情報入力] です。タイトルバーの内容を変更するには、AskText の前に SetDialogTitle を呼び出してください。

構文

```
AskText ( szQuestion, szDefault, svResult );
```

パラメーター

テーブル 13 ・ AskText のパラメーター

パラメーター	説明
szQuestion	表示する質問またはステートメントを指定します。このパラメーターの文字列の長さがスタティック テキスト フィールドの幅を超える場合は、1 つまたは複数の改行が文字列に挿入され、ダイアログ内に複数行にわたって表示されます。希望に応じて、1 つまたは複数のニューラインエスケープシーケンス (¥n) を文字列に挿入して、文字列を手動で形式設定できます。このパラメーターには、デフォルト値がありません。
szDefault	編集ボックスのデフォルトのテキストを指定します。この編集フィールドはスクロールできるので、長い文字列にも対応します。
svResult	[次へ] ボタンをクリックしてダイアログを閉じる際に、エンドユーザーが入力したテキストが返されます。ユーザーが、[戻る] ボタンをクリックした場合、svResult の値は予測不能になります。したがって、szDefault と svResult の両方に対して同じ変数を使用している場合、AskText からの戻り値が BACK の時は、再初期化します。 ***メモ・** svResult で渡す文字列変数は編集ボックスに入力されるテキストに対応できる充分な長さが必要です。そのため、変数を宣言するには自動サイズ調整メソッドを利用します。*

戻り値

テーブル 14 ・ AskText の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。

追加情報

AskText 関数によって表示されるダイアログは、スキンを使って表示することはできません。スキンを指定したかしないにかかわらず、同じように表示されます。

AskText の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * AskText 関数のデモンストレーションを行います。
 *
 * このスクリプトはエンドユーザーから会社名を取得します。
 *
¥*---------------------------------------------------------*/

#define MSG_TEXT        "会社名を入力してください。"
#define DEFAULT_COMPANY "My Software Company"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_AskText(HWND);

function ExFn_AskText(hMSI)
   STRING svCompany, szTitle;
   NUMBER nResult;
begin

   // 会社名を取得します。
   nResult = AskText (MSG_TEXT, DEFAULT_COMPANY, svCompany);

   if nResult = NEXT then
      // ユーザーが入力した会社名を表示します。
      MessageBox ("会社 : " + svCompany, INFORMATION);
   endif;

end;
```

AskYesNo

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

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

AskYesNo 関数は、エンドユーザーが [はい] または [いいえ] のボタンをクリックして質問に答える形式のメッセージボックスを表示します。**AskYesNo** のメッセージには、 4 種類の項目が含まれます。

- クエスチョンマーク アイコン
- 質問のテキスト
- [はい] ボタン
- [いいえ] ボタン

メモ・デフォルトのタイトルは「質問」です。タイトルバーの内容を変更するには、AskYesNo の前に SetDialogTitle を呼び出してください。

AskYesNo メッセージボックスは、対応する Windows API 関数への直接呼び出しによって作成されます。この関数は、システム モーダル ダイアログを表示します。一度表示されたモーダル ダイアログは、エンドユーザーが閉じるまでフォーカスを保持します。

Windows はこのダイアログを表示するため、インストールがダイアログ上にあるボタンのテキストを変更することはできません。Windows が、ボタンテキストをオペレーティング システムの言語で表示 (英語システムでは "Yes" または "No") するため、このテキストを手動でローカライズする必要はありません。さらに高度な柔軟性があるダイアログが必要な場合は、Windows API 関数を直接呼び出すか、カスタム ダイアログを使用します。

構文

AskYesNo (szQuestion, nDefault);

パラメーター

テーブル 15・AskYesNo のパラメーター

パラメーター	説明
szQuestion	メッセージボックスに表示する質問を指定します。メッセージが長すぎて 1 行に収まらない場合は、ニューラインエスケープシーケンス ¥n をメッセージに埋め込んで改行を挿入します。
nDefault	デフォルトで選択したボタンを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ YES— ダイアログが開くと [はい] ボタンがハイライト表示されます。 ・ NO— ダイアログが開くと [いいえ] ボタンがハイライト表示されます。

戻り値

テーブル 16・AskYesNo の戻り値

戻り値	説明
YES (1)	ユーザーが [はい] ボタンをクリックしたことを示します。
NO (0)	ユーザーが [いいえ] ボタンをクリックしたことを示します。

追加情報

AskYesNo 関数によって表示されるダイアログは、スキンを使って表示することはできません。スキンを指定したかしないにかかわらず、同じように表示されます。

AskYesNo の例

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

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

***メモ・**基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* AskYesNo 関数のデモンストレーションを行います。
*
* このスクリプトでは、ユーザーに対して ReadMe ファイルを
* 表示するかどうかを質問します。[ はい ] の場合、スクリプトは
* Windows Notepad を起動して ReadMe ファイルを開きます。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*      ターゲットシステム上の Windows Notepad 実行可能ファイルの
*      完全修飾名と有効なテキスト ファイルを
*      参照するように設定してください。
*
¥*---------------------------------------------------------*/

#define PROGRAM "C:¥¥Windows¥¥Notepad.exe"
#define PARAM   "C:¥¥Windows¥¥Readme.txt"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_AskYesNo(HWND);

function ExFn_AskYesNo(hMSI)
begin

   // AskYesNo ダイアログを表示します。デフォルトは [ はい ] に設定されています。
   if (AskYesNo(" インストールが完了しました。Readme " +
            " を読みますか？ ", YES )=YES then
      LaunchApp(PROGRAM, PARAM);
   endif;

end;
```

BatchAdd

BatchAdd 関数は SET コマンドまたはその他の DOS コマンドを、BatchFileLoad と共にメモリにロードされたバッチ ファイルに挿入します。nOptions パラメーターを利用してファイルの最初または最後のステートメントとして新規コマンドを追加したり、既存のステートメントを新規コマンドと置換したり、あるいは新規コマンドが既存ステートメントの前 / 後に追加されるよう指定したりできます。

BatchAdd の前に BatchFileLoad を呼び出して、修正するファイルをメモリにロードします。ファイルを変更した後、BatchFileSave を呼び出してディスクに保存します。

簡易バッチファイル関数を拡張バッチファイル関数と同時に使用しないでください。BatchFileLoad を呼び出した後、BatchFileSave を呼び出してファイルを保存するまで簡易バッチファイル関数を使用することはできません。

構文

```
BatchAdd ( szKey, szValue, szRefKey, nOptions );
```

パラメーター

テーブル 17 ・ BatchAdd のパラメーター

パラメーター	説明
szKey	バッチファイルへ追加するキーワードを指定します。PATH、TEMP、そして MYENV はこのパラメーターで利用できる有効なキーの例です。
szValue	バッチファイルへ追加するキーの値を指定します。この文字列は 512 バイト以内でなくてはなりません。512 バイト以上の文字列を渡すとインストールエラーが発生します。長い文字列を追加するには、FileGrep 関数と FileInsertLine 関数を利用します。 ***注意 ・*** *バッチファイルは長いパスを全くサポートしません。この関数を使用して長いパスを持つ行を追加する場合は、LongPathToShortPath を呼び出して長いパスを短い同等のパスに変換してから、バッチファイルに配置する文字列に追加してください。長いパスと長いファイル名についての情報は、「長いファイル名フォーマット」を参照してください。*
szRefKey	バッチファイルで ｓ ｚ Key を追加するのに関連する参照キーを指定します。

テーブル 17 ・ BatchAdd のパラメーター (続き)

パラメーター	説明
nOptions	ファイルのどの位置に行を挿入するかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ **BEFORE** ― szRefKey を含む初めの行の前にステートメントが追加されます。szRefKey がヌル文字列 ("") の場合、ステートメントはファイルの最初の行として追加されます。 ・ **AFTER** ― s z RefKey を含む最後の行の後にステートメントが追加されます。szRefKey がヌル文字列 ("") の場合、ステートメントはファイルの最後の行として追加されます。 ・ **REPLACE** ― ステートメントはファイルの既存行を置換します。複数の行に同じキーがある場合、最後の行のみが置換されます。 s z Key がファイルにない場合、新しい行が szRefKey の後に追加されます。szRefKey がヌル文字列 ("") の場合、新しい行がファイルの最後の行として追加されます。 追加するステートメントが SET コマンドでない場合、 s z Key でヌル文字列 ("") を渡し、 s z Value で完全なコマンドを渡し、そして以下に示すようにその他のオプション定数のひとつと定数 COMMAND を組み合わせるには、OR 演算子を利用します。 BatchAdd("", "PAUSE", "", COMMAND | AFTER); ***メモ ・** 定数 COMMAND と nOptions でパスする値とを OR 演算子を利用して組み合わせない限り、BatchAdd は、挿入する DOS キーワード SET をステートメントの初めに自動的に追加します。nOptions で REPLACE をあえて指定しない限り、バッチファイルに同じ行が重複する場合でも指定されたステートメントが追加されます。*

戻り値

テーブル 18 · BatchAdd の戻り値

戻り値	説明
0	BatchAdd はバッチファイルへ SET ステートメントまたはその他のコマンドを無事に追加しました。
< 0	BatchAdd はバッチファイルへ SET ステートメントまたはその他のコマンドを追加することができませんでした。

追加情報

InstallScript 参照キーは環境変数、DOS コマンド、またはプログラムファイル名のどれかです。環境変数は PATH、COMSPEC、LIB、その他の定義済みまたはユーザー定義の識別子といったキーワードです。環境変数の値は DOS SET コマンドを使って設定されます。バッチファイルに使われるステートメントは DOS コマンド、プログラム名 (コマンドラインパラメーターを含む / 含まず)、あるいはコメントの何れかでなくてはなりません。コマンドと環境変数の詳しい定義については、ご使用のオペレーティングシステムマニュアルを参照してください。

BatchAdd の例

以下の例は、次のインストールに適用します。

InstallScript/InstallScript MSI インストール

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
*  BatchAdd 関数のデモンストレーションを行います。
*
* このサンプルスクリプトでは、バッチファイルへ 3 つのステートメントを追加します。
* まず、PATH ステートメントを追加します。次に、環境変数 EXENV を設定する
* コマンドを追加します。その後、Windows を開始する既存のコマンドの
* 前に SHARE.EXE を起動するコマンドを
* 追加します。最後に、元のファイルを
* 編集済みファイルを元の名前の下に保存します。
*
* BatchAdd への呼び出しに失敗した場合、セットアップは
* バッチファイルへの変更を保存せずに終了します。
*
* メモ : このスクリプトを実行する前に、ISExampl.bat と
*      名づけられたバッチ ファイルを、ドライブ C のルートに作成します。
*      最も効果的に行うためには、ファイルに次の行を
*      含めます :
*
*      PATH=C:¥Windows
*      Win
*
¥*---------------------------------------------------------------*/
```

```
#define EXAMPLE_BAT "C:¥¥ISEXAMPL.BAT"
#define EXAMPLE_BAK "ISEXAMPL.BAK"

    STRING szPath;

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"
      function OnBegin()
      begin

    // 編集するバッチ ファイルをロードします。
    if (BatchFileLoad (EXAMPLE_BAT) < 0) then
        MessageBox (EXAMPLE_BAT + " をロードできませんでした。", SEVERE);
        abort;
    endif;

    // C:¥EXAPP¥BIN への既存の検索パスの値を追加する
    // SET PATH コマンドを追加します。
    szPath = "C:¥¥EXAPP¥¥BIN;%PATH%";

    if (BatchAdd ("PATH", szPath, "PATH", AFTER) < 0) then
        MessageBox ("BatchAdd の最初の呼び出しに失敗しました。", WARNING);
        abort;
    endif;

    //SET EXENV = C:¥OTHERAPP¥BIN を追加します。インストールが
    // 環境変数 EXENV がバッチファイルに既に存在する場合、
    // 最後の SET EXENV ステートメントが置換されます。
    szPath = "C:¥¥OTHERAPP¥¥BIN";

    if (BatchAdd ("EXENV", szPath, "EXENV", REPLACE) < 0) then
        MessageBox ("BatchAdd の 2 回目の呼び出しに失敗しました。", WARNING);
        abort;
    endif;

    // コマンド WIN の前にコマンド SHARE.EXE を追加します。
    if (BatchAdd ("", "SHARE.EXE", "WIN", BEFORE | COMMAND) < 0) then
        MessageBox ("BatchAdd の 3 回目の呼び出しに失敗しました。", WARNING);
        abort;
    endif;

    // 更新されたファイルを保存し、元のファイルをバックアップします。
    if (BatchFileSave(EXAMPLE_BAK) < 0) then
        MessageBox (EXAMPLE_BAK + " を保存できませんでした。", SEVERE);
    else
        MessageBox (" バッチファイルが保存されました。バックアップが作成されました。",INFORMATION);
    endif;

end;
```

以下の例は、次のインストールに適用します。

基本の MSI インストール

ヒント・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
*  BatchAdd 関数のデモンストレーションを行います。
*
* このサンプルスクリプトでは、バッチファイルへ 3 つのステートメントを追加します。
* まず、PATH ステートメントを追加します。次に、環境変数 EXENV を設定する
* コマンドを追加します。その後、Windows を開始する既存のコマンドの
* 前に SHARE.EXE を起動するコマンドを
* 追加します。最後に、元のファイルを
* 編集済みファイルを元の名前の下に保存します。
*
* BatchAdd への呼び出しに失敗した場合、セットアップは
* バッチファイルへの変更を保存せずに終了します。
*
* メモ : このスクリプトを実行する前に、ISExampl.bat と
*      名づけられたバッチ ファイルを、ドライブ C のルートに作成します。
*      最も効果的に行うためには、ファイルに次の行を
*      含めます :
*
*      PATH=C:¥Windows
*      Win
*
¥*---------------------------------------------------------*/

#define EXAMPLE_BAT "C:¥¥ISEXAMPL.BAT"
#define EXAMPLE_BAK "ISEXAMPL.BAK"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_BatchAdd(HWND);

function ExFn_BatchAdd(hMSI)
   STRING szPath;
begin

   // 編集するバッチ ファイルをロードします。
   if (BatchFileLoad (EXAMPLE_BAT) < 0) then
      MessageBox (EXAMPLE_BAT + " をロードできませんでした。", SEVERE);
      abort;
   endif;

   // C:¥EXAPP¥BIN への既存の検索パスの値を追加する
   // SET PATH コマンドを追加します。
   szPath = "C:¥¥EXAPP¥¥BIN;%PATH%";

   if (BatchAdd ("PATH", szPath, "PATH", AFTER) < 0) then
      MessageBox ("BatchAdd の最初の呼び出しに失敗しました。", WARNING);
```

```
    abort;
endif;

//SET EXENV = C:¥OTHERAPP¥BIN を追加します。インストールが
// 環境変数 EXENV がバッチファイルに既に存在する場合、
// 最後の SET EXENV ステートメントが置換されます。
szPath = "C:¥¥OTHERAPP¥¥BIN";

if (BatchAdd ("EXENV", szPath, "EXENV", REPLACE) < 0) then
    MessageBox ("BatchAdd の 2 回目の呼び出しに失敗しました。", WARNING);
    abort;
endif;

// コマンド WIN の前にコマンド SHARE.EXE を追加します。
if (BatchAdd ("", "SHARE.EXE", "WIN", BEFORE | COMMAND) < 0) then
    MessageBox ("BatchAdd の 3 回目の呼び出しに失敗しました。", WARNING);
    abort;
endif;

// 更新されたファイルを保存し、元のファイルをバックアップします。
if (BatchFileSave(EXAMPLE_BAK) < 0) then
    MessageBox (EXAMPLE_BAK + " を保存できませんでした。", SEVERE);
else
    MessageBox (" バッチファイルが保存されました。バックアップが作成されました。",INFORMATION);
endif;

end;
```

BatchDeleteEx

BatchDeleteEx 関数は、 s z Key で指定された値を含むバッチファイル内の行を削除します。

メモ・*BatchDeleteEx* *の前に* *BatchFileLoad* *を呼び出して、修正するファイルをメモリにロードします。ファイルを変更した後、* *BatchFileSave* *を呼び出してディスクに保存します。*

簡易バッチファイル関数を拡張バッチファイル関数と同時に使用しないでください。BatchFileLoad を呼び出した後、BatchFileSave を呼び出してファイルを保存するまで簡易バッチファイル関数を使用することはできません。

構文

```
BatchDeleteEx ( szKey, nOptions );
```

パラメーター

テーブル 19 ・ BatchDeleteEx のパラメーター

パラメーター	説明
szKey	削除する単数または複数行を指定する参照キーワードを指定します。
nOptions	szKey が SET ステートメントまたはコマンドで環境変数を指定するか否かを示します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ **0 ー** szKey は SET ステートメントの環境変数であることを指定します。環境変数は定義済みの識別子 (PATH、COMSPEC、または LIB など)、あるいはユーザー定義の識別子の何れかです。たとえば、 s z Key の値が "LIBPATH" で nOption が 0 に設定されている場合、次のステートメントが削除されます : SET LIBPATH=C:¥Lang¥Lib ・ **COMMAND ー** szKey が DOS コマンドまたはプログラムファイル名のどちらかを指定します。

戻り値

テーブル 20 ・ BatchDeleteEx の戻り値

戻り値	説明
0	BatchDeleteEx は指定された値を含む行を無事に削除しました。
< 0	BatchDeleteEx は指定された値を含む行を削除できませんでした。

BatchDeleteEx の例

メモ ・ *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
```

```
*
* BatchDeleteEx 関数のデモンストレーションを行います。
*
* このスクリプト例では、バッチ ファイルからの行削除を行います。まず、
* BatchFileLoad を呼び出してファイルをロードします。次に、PATH コマンドを持つ
* 行をすべて削除します。そして MyApp.exe ( たとえば C:¥MyApps¥MyApp.exe) を
* 参照するすべての行を削除します。
* 最後に、オリジナルファイルをバックアップして
* 編集されたファイルを元の名前の下に保存します。
*
* メモ : このスクリプトを実行する前に、ISExampl.bat と
*      名づけられたバッチ ファイルを、ドライブ C のルートに作成します。
*      最も効果的に行うためには、ファイルに次の行を
*      含めます :
*
*      SET PATH=C:¥Windows
*      C:¥MyApps¥MyApp.exe
*
¥*---------------------------------------------------------------*/

#define EXAMPLE_BAT "C:¥¥ISExampl.bat"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_BatchDeleteEx(HWND);

function ExFn_BatchDeleteEx(hMSI)
  STRING szBackupFile;
begin

  // 編集するバッチ ファイルをロードまたは作成します。
  if (BatchFileLoad (EXAMPLE_BAT) < 0) then
    MessageBox (EXAMPLE_BAT + " をロードできませんでした。", SEVERE);
    abort;
  endif;

  // SET PATH= コマンドをすべて削除します。
  BatchDeleteEx ("PATH", 0);

  // MyApp.exe への参照を含むすべての行を削除します。
  BatchDeleteEx ("MyApp.exe", COMMAND);

  // 編集されたバッチ ファイルを保存します。
  if (BatchFileSave("Example.bak") < 0) then
    MessageBox (EXAMPLE_BAT+ " を保存できませんでした。", SEVERE);
  else
    MessageBox (" バッチ ファイルが保存されました。",INFORMATION);
  endif;

end;
```

BatchFileLoad

BatchFileLoad 関数は指定したバッチファイルのコピーをメモリにロードし、ファイル上の操作でその他の拡張ファイル関数を呼び出せるようにします。szBatchFile で編集するバッチファイルの名前を指定するか、szBatchFile でヌル文字列 (″″) を渡してデフォルトのバッチファイルを編集します。これはシステムが利用する Autoexec.bat ファイルを起動するためにもともと InstallShield が設定するものです。

新規バッチファイルを作成するのに BatchFileLoad の呼び出しが可能です。そのためには、szBatchFile で存在しないファイル名を渡します。そして新規ファイルを編集するのに他のバッチ関数を呼び出します。最後に、BatchFileSave を呼び出してディスクへ新規ファイルを保存します。

メモ・拡張バッチファイル関数を利用する前に、BatchFileLoad を呼び出して変更するファイルをメモリにロードしなくてはなりません。ファイルを変更した後、BatchFileSave を呼び出してディスクに保存します。デフォルトでインストールスクリプトが利用するバッチファイルの完全修飾ファイル名を取得するには、BatchGetFileName を呼び出します。デフォルトでインストールスクリプトが別のバッチファイルを利用するように指定するには、BatchSetFileName を呼び出します。

簡易バッチファイル関数を拡張バッチファイル関数と同時に使用しないでください。BatchFileLoad を呼び出した後、BatchFileSave を呼び出してファイルを保存するまで簡易バッチファイル関数を使用することはできません。

構文

```
BatchFileLoad ( szBatchFile );
```

パラメーター

テーブル 21 ・ BatchFileLoad のパラメーター

パラメーター	説明
szBatchFile	メモリにロードするバッチファイルの完全修飾名を指定します。現在のデフォルトのバッチファイルをロードするには、ヌル文字列 (″″) を渡します。このパラメーターでファイルを指定すると、そのファイルがデフォルトのバッチファイルとなります。この関数を呼び出した後、ファイルを操作するのに拡張バッチファイル関数をすべて利用することができます。 BatchFileLoad を使った新しいバッチファイルを作成するには、szBatchFile に存在しないファイル名を渡します。そして新規ファイルを編集するのに他のバッチ関数を呼び出します。

戻り値

テーブル 22 ・ BatchFileLoad の戻り値

戻り値	説明
0	BatchFileLoad がバッチファイルバッファーを初期化しました。szConfigFile が既存バッチファイルを指定する場合、ファイルはバッファーにロードされます。それ以外の場合は空のバッファーが作成されます。
< 0	BatchFileLoad はバッチファイルバッファーを初期化することができませんでした。

BatchFileLoad の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* BatchFileLoad と BatchFileSave 関数のデモンストレーションを行います。
*
```

```
* このスクリプト例では、編集するためにバッチファイルを開く方法、
* オリジナルファイルのバックアップ作成方法、
* そして編集したファイルの保存方法とその閉じ方を説明します。
*
* BatchFileSave のファイルバックアップ機能がどうやって既存ファイルの上書きを
* 防ぐのかをデモンストレーションします。このスクリプトは 2 つの
* 異なるバッチ ファイルをロードし保存します。最初のバッチファイルは
* 特定のファイル名と共にバックアップされます。2 番目のファイルは
* ワイルドカード拡張子を使ってバックアップされ、BatchFileSave が
* 3 桁の一意のファイル拡張子を生成します。
*
* メモ : このスクリプトを実行する前に、C ドライブ のルートへ
*      2 つのファイル (ISExamp1.bat と ISExamp2.bat) を作成します。
*      効果的に行うためには、ISExamp1.* または ISExamp2.* と
*      名づけられた他のファイルを削除または移動する必要があります。
*
¥*---------------------------------------------------------------*/

//  この例で使われているバッチファイルとバックアップファイルの名前。
#define EXAMPLE1 "ISExamp1"
#define EXAMPLE2 "ISExamp2"

// バッチファイルの完全修飾名。
#define EXAMPLE1_BAT "C:¥¥" + EXAMPLE1 + ".bat"
#define EXAMPLE2_BAT "C:¥¥" + EXAMPLE2 + ".bat"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_BatchFileLoad(HWND);

function ExFn_BatchFileLoad(hMSI)
begin

  // EXAMPLE1_BAT. をロードします。
  if (BatchFileLoad (EXAMPLE1_BAT) < 0) then
    MessageBox (EXAMPLE1_BAT + " をロードできませんでした。", SEVERE);
    abort;
  endif;

  // 最初のファイルを編集するには、ここに他のバッチファイル関数を使います。
  // 拡張子 "bak" を持つ元のファイルのバックアップを行い、
  // 編集されたファイルを元の名前の下に保存します。ISExamp1.bak が既に存在する場合、
  // BatchFileSave は数字の付いた拡張子を生成します。
  if (BatchFileSave (EXAMPLE1 + ".bak") < 0) then
    MessageBox (EXAMPLE1_BAT+ " を保存できませんでした。", SEVERE);
    abort;
  else
    MessageBox (EXAMPLE1_BAT + " が保存されました。",INFORMATION);
  endif;

  // EXAMPLE2_BAT. をロードします。
  if (BatchFileLoad (EXAMPLE2_BAT) < 0) then
    MessageBox (EXAMPLE2_BAT + " をロードできませんでした。", SEVERE);
    abort;
  endif;

  // 2 番目のファイルを編集するには、ここに他のバッチファイル関数を使います。
  // 数字の付いた拡張子を使って元のバッチファイルをバックアップします。
```

```
    // そして編集済みファイルを元の名前の下に保存します。
    if (BatchFileSave (EXAMPLE2 + ″.*″)< 0) then
        MessageBox (EXAMPLE2_BAT + ″ を保存できませんでした。″, SEVERE);
        abort;
    else
        MessageBox (EXAMPLE2_BAT + ″ が保存されました。″,INFORMATION);
    endif;

end;
```

BatchFileSave

BatchFileSave は BatchFileLoad 関数を使ってメモリにロードしたバッチ ファイルをディスクへ保存します。ファイルはオリジナル名の元に保存されます。szBackupFile でファイル名が指定された場合、編集されたファイルがディスクに書き込まれる前にオリジナルファイル名はそのファイル名に変更されます。szBackupFile にヌル文字列 (″″) が含まれる場合、オリジナルファイルは変更されたファイルに置換されます。拡張バッチファイル関数を利用してバッチファイルの変更を完了したときに BatchFileSave を呼び出さ無かった場合、すべての変更点は失われます。

メモ・簡易バッチファイル関数を拡張バッチファイル関数と同時に使用しないでください。BatchFileLoad を呼び出した後、BatchFileSave を呼び出してファイルを保存するまで簡易バッチファイル関数を使用することはできません。

構文

```
BatchFileSave ( szBackupFile );
```

パラメーター

テーブル 23 ・ BatchFileSave のパラメーター

パラメーター	説明
szBackupFile	編集する前の状態でオリジナルファイルのバックアップコピーを保存するかどうかを指定します。 ・ バックアップファイルを作成しない場合、このパラメーターでヌル文字列を指定します。 ・ オリジナルファイルを特定の名前でバックアップするには、そのファイル名をこのパラメーターで渡します。ファイル名は不完全修飾 (つまり、ドライブ及び / またはパスを指定しない) でなくてはなりません。特定名のファイルが既に存在する場合、BatchFileSave が次の箇条書きリストに述べられた固有のファイル拡張子を生成することに注意してください。 ・ オリジナルファイルをインストールが生成したファイル拡張子とともにバックアップする場合は、ワイルドカード文字 "" をファイル拡張子として指定します (例 "Batch.")。そしてインストールが 001 から始まる数値を拡張子として割り当てます。その拡張子を含むファイルが既に存在する場合、固有のファイル名が作成されるまで拡張子の値が 1 づつ追加されます。 バックアップファイルが作成された後、InstallShield はシステム変数 INFOFILENAME でバックアップファイル名を保存します。 ***メモ ・*** *BatchFileLoad への最後の呼び出しで指定されたバッチファイルが存在しなかった場合、バックアップファイルは BatchFileSave への呼び出しで作成されたバッチファイルと同じになります。szBackupFile が元のバッチ ファイルの名前を指定する場合、バックアップ ファイルは作成されません。*

戻り値

テーブル 24 - BatchFileSave の戻り値

戻り値	説明
0	BatchFileSave はメモリにあるバッチファイルを無事ディスクに保存しました。
< 0	BatchFileSave はバッチファイルをディスクに保存することができませんでした。

BatchFileSave の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* BatchFileLoad と BatchFileSave 関数のデモンストレーションを行います。
*
* このスクリプト例では、編集するためにバッチファイルを開く方法、
* オリジナルファイルのバックアップ作成方法、
* そして編集したファイルの保存方法とその閉じ方を説明します。
*
* BatchFileSave のファイルバックアップ機能がどうやって既存ファイルの上書きを
* 防ぐのかをデモンストレーションします。このスクリプトは 2 つの
* 異なるバッチ ファイルをロードし保存します。最初のバッチファイルは
* 特定のファイル名と共にバックアップされます。2 番目のファイルは
* ワイルドカード拡張子を使ってバックアップされ、BatchFileSave が
* 3 桁の一意のファイル拡張子を生成します。
*
* メモ : このスクリプトを実行する前に、C ドライブ のルートへ
*      2 つのファイル (ISExamp1.bat と ISExamp2.bat) を作成します。
*      効果的に行うためには、ISExamp1.* または ISExamp2.* と
*      名づけられた他のファイルを削除または移動する必要があります。
*
¥*---------------------------------------------------------*/

//  この例で使われているバッチファイルとバックアップファイルの名前。
#define EXAMPLE1 "ISExamp1"
#define EXAMPLE2 "ISExamp2"

// バッチファイルの完全修飾名。
#define EXAMPLE1_BAT "C:¥¥" + EXAMPLE1 + ".bat"
#define EXAMPLE2_BAT "C:¥¥" + EXAMPLE2 + ".bat"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"
```

```
    export prototype ExFn_BatchFileSave(HWND);

function ExFn_BatchFileSave(hMSI)
begin

   // EXAMPLE1_BAT. をロードします。
   if (BatchFileLoad (EXAMPLE1_BAT) < 0) then
      MessageBox (EXAMPLE1_BAT + " をロードできませんでした。", SEVERE);
      abort;
   endif;

   // 最初のファイルを編集するには、ここに他のバッチファイル関数を使います。
   // 拡張子 "bak" を持つ元のファイルのバックアップを行い、
   // 編集されたファイルを元の名前の下に保存します。ISExamp1.bak が既に存在する場合、
   // BatchFileSave は数字の付いた拡張子を生成します。
   if (BatchFileSave (EXAMPLE1 + ".bak") < 0) then
      MessageBox (EXAMPLE1_BAT+ " を保存できませんでした。", SEVERE);
      abort;
   else
      MessageBox (EXAMPLE1_BAT + " が保存されました。",INFORMATION);
   endif;

   // EXAMPLE2_BAT. をロードします。
   if (BatchFileLoad (EXAMPLE2_BAT) < 0) then
      MessageBox (EXAMPLE2_BAT + " をロードできませんでした。", SEVERE);
      abort;
   endif;

   // 2 番目のファイルを編集するには、ここに他のバッチファイル関数を使います。
   // 数字の付いた拡張子を使って元のバッチファイルをバックアップします。
   // そして編集済みファイルを元の名前の下に保存します。
   if (BatchFileSave (EXAMPLE2 + ".*")< 0) then
      MessageBox (EXAMPLE2_BAT + " を保存できませんでした。", SEVERE);
      abort;
   else
      MessageBox (EXAMPLE2_BAT + " が保存されました。",INFORMATION);
   endif;

end;
```

BatchFind

BatchFind 関数は、ｓｚRefKey で指定された参照キーすべてをバッチファイル内で検索します。nOptions に定数 RESTART を指定した場合、最初に現れる参照キーが返されます。次に現れる szRefKey を検出するには、この関数を nOptions で CONTINUE に設定して繰り返し呼び出します。

メモ・BatchFind の前に BatchFileLoad を呼び出して、修正するファイルをメモリにロードします。ファイルを変更した後、BatchFileSave を呼び出してディスクに保存します。

簡易バッチファイル関数を拡張バッチファイル関数と同時に使用しないでください。BatchFileLoad を呼び出した後、BatchFileSave を呼び出してファイルを保存するまで簡易バッチファイル関数を使用することはできません。

構文

BatchFind (szRefKey, svResult, nOptions);

パラメーター

テーブル 25 · BatchFind のパラメーター

パラメーター	説明
szRefKey	検索する参照キーを指定します。環境変数、DOS コマンド、またはプログラム名を参照キーとすることができます。参照キーがファイル名の場合でファイル拡張子を指定しなかったとき、関数はベースファイル名を持つ参照キーすべてを戻します。例えば、Win.com を指定したとき、検索は参照キーのみを捜します。Win を指定したとき、バッチファイルに存在する Win.exe、Win.dll、Win.sys 等の参照キーを戻します。
svResult	バッチファイルで検出された参照キーの値を指定します。
nOptions	検索開始位置を指定します。このパラメーターに、あらかじめ定義されている以下の定数のうちの 1 つを渡します。 • CONTINUE – バッチファイルでの現在の位置から検索をはじめます。 • RESTART – バッチファイルの最初から検索を開始します。 検索している参照キーが DOS コマンドまたはプログラム名 (環境変数ではない) の場合 、以下の例のように定数 COMMAND と CONTINUE または RESTART とを組み合わせるのに OR を使います。 BatchFind ("SCAN.EXE", svResult, COMMAND | RESTART);

戻り値

テーブル 26 · BatchFind の戻り値

戻り値	説明
0	BatchFind は szRefKey の値を検出して svResult で戻しました。
< 0	BatchFind は szRefKey の値を見つけて svResult で戻すことができませんでした。

BatchFind の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* BatchFind 関数のデモンストレーションを行います。
*
* このスクリプト例では、バッチファイルを検索してそのファイルが
* SHARE.EXE を参照するコマンドを含むか否かをレポートします。
* そして、すべての PATH と SET PATH ステートメントを探して表示します。
*
* メモ : このスクリプトを実行する前に、ISExampl.bat と
*      名づけられたバッチ ファイルを作成し、C ドライブのルートに
*      保存します。バッチ ファイルは Share.exe を起動するコマンドを含み、
*      PATH 、あるいは SET PATH= ステートメントを
*      1 つ以上持ちます。
*
¥*---------------------------------------------------------*/

#define TARGET_BATCH "C:¥¥ISExampl.bat"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_BatchFind(HWND);

function ExFn_BatchFind(hMSI)
  STRING svResult;
  NUMBER nResult;
begin

  // ターゲット バッチ ファイルをロードします。
  if (BatchFileLoad (TARGET_BATCH ) < 0) then
    MessageBox (TARGET_BATCH + " をロードできませんでした。", SEVERE);
    abort;
  endif;

  // SHARE.EXE コマンドをチェックします。
  nResult = BatchFind ("SHARE.EXE", svResult, COMMAND);

  if (nResult < 0) then
    MessageBox ("SHARE.EXE コマンドが見つかりませんでした。", WARNING);
  else
    MessageBox ("SHARE.EXE コマンドが見つかりました。", INFORMATION);
  endif;

  // 最初の PATH または SET PATH= ステートメントを検索します。ファイルの最初から
  // 検索を開始するよう、3 番目のパラメーターに RESTART を渡します。
  nResult = BatchFind ("PATH", svResult, RESTART);

  if (nResult < 0) then
```

```
        MessageBox ("PATH コマンドが見つかりませんでした。", WARNING);
    else

        // PATH コマンドが存在するときはループします。
        while (nResult = 0)
            MessageBox (svResult, INFORMATION);

            // 次の PATH コマンドを検索します。CONTINUE を 3 番目の
            // パラメーターに渡して、最後の検索結果に続けてステートメント
            // で検索を続行します。
            nResult = BatchFind ("PATH", svResult, CONTINUE);
        endwhile;

        MessageBox ("PATH コマンドはこれ以上見当たりません。", WARNING);

    endif;

end;
```

BatchGetFileName

BatchGetFileName 関数はデフォルト バッチ ファイルの完全修飾名を読み出します。これはもともと InstallShield によって システムが利用する起動ファイル Autoexec.bat に設定されています。デフォルトでスクリプトが別のバッチファイルを利用するように指定するには、BatchSetFileName を呼び出します。

メモ・簡易バッチファイル関数を拡張バッチファイル関数と同時に使用しないでください。BatchFileLoad を呼び出した後、BatchFileSave を呼び出してファイルを保存するまで簡易バッチファイル関数を使用することはできません。

構文

```
BatchGetFileName ( svFileName );
```

パラメーター

テーブル 27 ・ BatchGetFileName のパラメーター

パラメーター	説明
svFileName	svFileName でディフォルトバッチファイルの完全修飾名を戻します。

戻り値

テーブル 28 ・ BatchGetFileName の戻り値

戻り値	説明
0	BatchGetFileName デフォルトバッチファイルの完全修飾ファイル名を読み出しました。
< 0	BatchGetFileName デフォルトバッチファイルの完全修飾ファイル名の読み出しに失敗しました。

BatchGetFileName の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* BatchGetFileName 関数 と BatchSetFileName 関数の
* デモンストレーションを行います。
*
* このスクリプト例では、デフォルト構成ファイルの完全修飾名を読み出します。
* これはもともと起動ドライブ内のファイル
* Autoexec.bat です。そして C:¥ISExampl.bat をデフォルトのバッチファイル
* とします。最後に、もう一度デフォルトのバッチファイルの名前を
* 読み出して、変更されたことを確認表示します。
*
¥*---------------------------------------------------------*/

#define DEFAULT_BATCH_FILE "C:¥¥ISExampl.bat"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_BatchGetFileName(HWND);

function ExFn_BatchGetFileName(hMSI)
  STRING svFilename;
```

```
begin

   // デフォルトのバッチ ファイルの名前を取得します。
   if (BatchGetFileName (svFilename) < 0) then
      // エラーをレポートし、中止します。
      MessageBox ("BatchGetFileName の最初の呼び出しに失敗しました。", SEVERE);
      abort;
   else
      // デフォルトのバッチ ファイルの名前を表示します。
      MessageBox (" デフォルトのバッチ ファイルは " + svFilename + " です。",
                  INFORMATION);
   endif;

   // C:¥ISExampl.bat をデフォルトのバッチ ファイルとします。
   if (BatchSetFileName(DEFAULT_BATCH_FILE) < 0) then
      // エラーを報告します。
      MessageBox (" 新しいデフォルトのバッチ ファイルを設定できませんでした。", SEVERE);
   else

      // デフォルトのバッチ ファイルが変更されたことを確認します。
      if (BatchGetFileName(svFilename) < 0) then
         // エラーを処理します。
         MessageBox ("BatchGetFileName の 2 回目の呼び出しに失敗しました。", SEVERE);
      else
         // デフォルトのバッチ ファイルの名前を表示します。
         MessageBox (" 現在のデフォルトのバッチ ファイルは " + svFilename + " です。",
                     INFORMATION);
      endif;

   endif;

end;
```

BatchMoveEx

BatchMoveEx 関数は ｓ ｚ Move が指定した行をバッチファイル内で移動させます。パラメーター nOptions はバッチファイルの最初または最後、あるいは ｓ ｚ RefKey で指定された行の前後のいづれに配置するのかを指定します。

メモ・*BatchMoveEx を呼び出す前に BatchFileLoad を呼び出して、修正するファイルをメモリにロードします。ファイルを変更した後、BatchFileSave を呼び出してディスクに保存します。*

簡易バッチファイル関数を拡張バッチファイル関数と同時に使用しないでください。BatchFileLoad を呼び出した後、BatchFileSave を呼び出してファイルを保存するまで簡易バッチファイル関数を使用することはできません。

構文

```
BatchMoveEx ( szMove, szRefKey, nOptions, nMoveOption );
```

パラメーター

テーブル 29 ・ BatchMoveEx のパラメーター

パラメーター	説明
szMove	移動させる行を認識する参照キーを指定します。
szRefKey	移動させる行を配置するのに使う参照行を識別するキーを指定します。szRefKey がヌル文字列 (“”) の場合、ｓｚMove が指定する行は nOptions の値に従ってファイルの最初または終わりへ移動します。
nOptions	行の移動先を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ BEFORE ― ｓｚMove が指定した行は szRefKey の参照キーを含む行の前に移動します。szRefKey がヌル文字列 (“”) の場合、szMove が指定した行はファイルの最初に移動します。 ・ AFTER ― ｓｚMove が指定した行は szRefKey の参照キーを含む行の後に移動します。szRefKey がヌル文字列 (“”) の場合、szMove が指定した行はファイルの最後に移動します。 検索している参照キーが DOS コマンドまたはプログラム名 (環境変数ではない) の場合 、以下の例のように定数 COMMAND と BEFORE または AFTER とを組み合わせるのに OR を使います。 BatchMoveEx (OPATHO, OSCAN.EXEO, BEFORE | COMMAND, 0);
nMoveOption	szMove がコマンドまたは環境変数の何れかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ 0 ― szMove が環境変数であると指定します。 ・ COMMAND ― ｓｚMove がコマンドであると指定します。

戻り値

テーブル 30 - BatchMoveEx の戻り値

戻り値	説明
0	BatchMoveEx がバッチファイル内で指定された行の移動を完了しました。
< 0	BatchMoveEx がバッチファイル内で指定された行を移動することができませんでした。

BatchMoveEx の例

***メモ -** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* BatchMoveEx 関数のデモンストレーションを行います。
*
* このスクリプト例では、バッチファイル内で行を移動させます。まず、
* BatchFileLoad を呼び出してファイルをロードします。次に、初めの
* PATH コマンドをファイルの最後まで移動します。そして、
* Windows を起動するステートメントの前に Share.exe を
* 参照する最初のステートメントを移動します。
*
* メモ : このスクリプトを実行する前に、ISExampl.bat と
*      名づけられたバッチ ファイルを、ドライブ C のルートに作成します。
*      最も効果的に行うためには、ファイルの最初の行は PATH コマンドで、
*      次のステートメントで Windows を起動し、
*      最後のステートメントは Share.exe を実行します。
*
*
¥*---------------------------------------------------------*/

#define TARGET_BATCH "C:¥¥ISExampl.bat"
#define BACKUP_BATCH "ISExampl.bak"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_BatchMoveEx(HWND);

function ExFn_BatchMoveEx(hMSI)
begin

  // 編集するバッチ ファイルをロードします。
  if BatchFileLoad (TARGET_BATCH) < 0 then
     MessageBox (TARGET_BATCH + " をロードできませんでした。", SEVERE);
```

```
    abort;
endif;

// PATH ステートメントをファイルの最後へ移動します。
if (BatchMoveEx ("PATH", "", AFTER, 0) < 0) then
    MessageBox ("PATH ステートメントを移動できませんでした。", SEVERE);
else
    MessageBox ("PATH ステートメントはファイルの最後に移動しました。", INFORMATION);
endif;

// WIN ステートメントの前に SHARE.EXE コマンドを移動します。
if (BatchMoveEx ("SHARE.EXE", "WIN", BEFORE|COMMAND, COMMAND) < 0) then
    MessageBox ("SHARE.EXE ステートメントを移動できませんでした。", SEVERE);
else
    MessageBox ("SHARE.EXE ステートメントが WIN ステートメントの前に移動しました。",
            INFORMATION);
endif;

// 更新されたファイルを保存し、元のファイルをバックアップします。
if BatchFileSave (BACKUP_BATCH) < 0 then
    MessageBox (BACKUP_BATCH + " を保存することができませんでした。", SEVERE);
else
    MessageBox (" バッチファイルが保存されました。バックアップが作成されました。", INFORMATION);
endif;

end;
```

BatchSetFileName

BatchSetFileName 関数は、パラメーターとしてヌル文字列 ("") を使って呼び出したとき Ez バッチ ファイル関数や BatchFileLoad が使用するバッチファイルの名前を指定します。InstallScript では、このファイルはデフォルトのバッチファイルと呼ばれます。インストールの初期化中、デフォルトのバッチ ファイルはシステムが使用する起動ファイル Autoexec.bat ファイルに設定されています。

BatchSetFileName はデフォルトバッチファイルの名前を割り当てるだけです。指定されたファイルが存在するか否かを確認、あるいはファイルのメモリーへのロードを行いません。そのため、ファイル名が無効な場合や、指定されたファイルが存在しない場合でも関数は成功します。無効なファイル名は、後に続く簡易バッチファイル関数や詳細バッチファイル関数の失敗の原因となります。

メモ・簡易バッチファイル関数を拡張バッチファイル関数と同時に使用しないでください。BatchFileLoad を呼び出した後、BatchFileSave を呼び出してファイルを保存するまで簡易バッチファイル関数を使用することはできません。

構文

```
BatchSetFileName ( szBatchFile );
```

パラメーター

テーブル 31 ・ BatchSetFileName のパラメーター

パラメーター	説明
szBatchFile	インストールスクリプトでデフォルトとして使用されるバッチファイルの完全修飾名を指定します。

戻り値

テーブル 32 ・ BatchSetFileName の戻り値

戻り値	説明
0	BatchSetFileName 指定されたファイルをデフォルトバッチファイルとして設定しました。
< 0	BatchSetFileName 指定されたファイルをデフォルトバッチファイルとして設定することができませんでした。

BatchSetFileName の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* BatchGetFileName 関数 と BatchSetFileName 関数の
* デモンストレーションを行います。
*
* このスクリプト例では、デフォルト構成ファイルの完全修飾名を読み出します。
* これはもともと起動ドライブ内のファイル
* Autoexec.bat です。そして C:¥ISExampl.bat をデフォルトのバッチファイル
* とします。最後に、もう一度デフォルトのバッチファイルの名前を
* 読み出して、変更されたことを確認表示します。
*
¥*---------------------------------------------------------*/

#define DEFAULT_BATCH_FILE "C:¥¥ISExampl.bat"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_BatchSetFileName(HWND);
```

```
function ExFn_BatchSetFileName(hMSI)
    STRING svFilename;
begin

    // デフォルトのバッチ ファイルの名前を取得します。
    if (BatchGetFileName (svFilename) < 0) then
        // エラーをレポートし、中止します。
        MessageBox ("BatchGetFileName の最初の呼び出しに失敗しました。", SEVERE);
        abort;
    else
        // デフォルトのバッチ ファイルの名前を表示します。
        MessageBox (" デフォルトのバッチ ファイルは " + svFilename + " です。",
                INFORMATION);
    endif;

    // C:¥ISExampl.bat をデフォルトのバッチ ファイルとします。
    if (BatchSetFileName(DEFAULT_BATCH_FILE) < 0) then
        // エラーを報告します。
        MessageBox (" 新しいデフォルトのバッチ ファイルを設定できませんでした。", SEVERE);
    else

        // デフォルトのバッチ ファイルが変更されたことを確認します。
        if (BatchGetFileName(svFilename) < 0) then
            // エラーを処理します。
            MessageBox ("BatchGetFileName の 2 回目の呼び出しに失敗しました。", SEVERE);
        else
            // デフォルトのバッチ ファイルの名前を表示します。
            MessageBox (" 現在のデフォルトのバッチ ファイルは " + svFilename + " です。",
            INFORMATION);
        endif;

    endif;

end;
```

CalculateAndAddFileCost

CalculateAndAddFileCost 関数は、指定されたファイルのコストを判断し、それを nvCostHigh および / または nvCostLow の現在の値に追加します。これにより、ループの中で関数を複数回呼び出して、複数のファイルのコストを計算および加算することができます。関数を呼び出してシングルファイルのコストを判別する前に、nvCostHigh および nvCostLow を ゼロに設定します。この関数は通常、サイズが認識されているファイルのコストを判別する必要があるときに使用され、コストを FeatureAddCost に渡すことができるようにします。

メモ・この関数は実際、インストールによって直接使用される情報の設定は行わないので注意してください。この関数を呼び出した後、必要に応じて FeatureAddCost を呼び出して、追加のコストを既存の機能に追加する必要があります。

構文

CalculateAndAddFileCost (nFileSizeHigh, nFileSizeLow, szTargetDir, nClusterSize, nvCostHigh, nvCostLow);

パラメーター

テーブル 33 ・ CalculateAndAddFileCost のパラメーター

パラメーター	説明
nFileSizeHigh	ファイル サイズの上位 31 ビット (バイト数)。通常、GetFileInfo を使って取得されます。
nFileSizeLow	ファイル サイズの下位 31 ビット (バイト数)。通常、GetFileInfo を使って取得されます。
szTargetDir	nClusterSize が 0 の場合、これはファイルのターゲットフォルダーになります。このパスは、ターゲットドライブのクラスタサイズを判断するときに利用されます。nClusterSize がゼロ以外の場合、このパラメーターは無視されます。
nClusterSize	ターゲットドライブのクラスタサイズを指定します。このパラメーターが 0 の場合、関数は szTargetDir からこの情報を判断します。
nvCostHigh	このファイルのインストール コストの上位 31 ビット (バイト数) がこの変数の現在の値に追加されます。
nvCostLow	このファイルのインストール コストの下位 31 ビット (バイト数) がこの変数の現在の値に追加されます。

戻り値

テーブル 34 ・ CalculateAndAddFileCost の戻り値

戻り値	説明
ISERR_SUCCESS	関数が成功したことを示します。
< ISERR_SUCCESS	関数の実行に失敗したことを示します。

CallDLLFx

***ヒント ・** CallDLLFx 関数は、以前のバージョンの InstallShield で作成したスクリプトとの互換性を目的としてのみサポートされています。**CallDLLFx** 関数の代わりに、「.dll フィル関数の呼び出し」で説明されている、より柔軟性のある方法を考慮してください。*

CallDLLFx 関数は指定した .dll ファイル内で関数を呼び出します。

構文

CallDLLFx (szDLL, szFunction, lvValue, svValue);

呼び出された関数には次の決められた定義を使用する必要があります。この定義では、主要な InstallShield ウィンドウの主要なウィンドウハンドル は hwnd です。

LONG APIENTRY YourFunction (HWND hwnd, LPLONG lpIValue, LPSTR lpszValue);

パラメーター

テーブル 35 ・ CallDLLFx のパラメーター

パラメーター	説明
szDLL	実行するための関数を含んだ .dll ファイルの完全修飾名を指定します。
szFunction	ｓ ｚDLL で指定した .dll ファイル内の関数名を指定します。
lvValue	.dll 関数へのリファレンスが渡す長い整数変数を指定します。
svValue	.dll 関数へ渡す文字列変数を指定します。

戻り値

CallDLLFx 関数は、.dll の関数から長い整数を戻します。

CallDLLFx の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CallDLLFx 関数のデモンストレーションを行います。
*
* メモ : このスクリプトでは、定数 DLL_FILE が
*     フォーマットが下のプロトタイプ宣言に一致する Test という名前の
*     含む .dll ファイルの完全修飾名に設定しなくてはなりません。
**      その関数は 3 番目と 4 番目の
*     パラメーターで渡される値を変更してから、
*     同じパラメーターにその値を返します。
*
*
¥*---------------------------------------------------------*/
#define ID_NEXT 1 // ユーザーが [ 次へ ] ボタンをクリックした場合に値を返します
```

```
#define ID_CANCEL 2 // ユーザーが [ キャンセル ] ボタンをクリックしたときに値を返します
#define ID_BACK 4 // ユーザーが [ 戻る ] ボタンをクリックしたときに値を返します


// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_CallDLLFx(HWND);

function ExFn_CallDLLFx(hMSI)
    INT    nValue, nResult;
    STRING szString, szResult, szDLL, szValue, szReturn;
begin

// セットアップウィンドウのタイトルを設定します。
SetTitle ("CallDLLFx Example", 18, WHITE);

// .dll の場所を設定します。
szDLL = SUPPORTDIR ^ "MYDLL.DLL";

// .dll 関数を呼び出すパラメーターを設定します。
nValue   = 3000;
szString = " テスト文字列 ";

// ユーザーへ入力を表示します。
SprintfBox (INFORMATION,"", "Before - nValue: %i , szString: %s",
nValue, szString);

// .dll 関数の呼び出し。値で渡します。
nResult = CallDLLFx(szDLL, "Test", nValue, szString);

// .dll 関数が戻した値を表示します。
SprintfBox(INFORMATION,"", "Returned - nValue: %i , szString: %s",
nValue, szString);

end;
```

ChangeDirectory

ChangeDirectory 関数は現在のディレクトリを設定します。

メモ・指定したディレクトリを現在のディレクトリとする為に ChangeDirectory を呼び出した後、そのディレクトリを削除することはできません。ディレクトリを削除する前に、ChangeDirectory をもう一度呼び出して別の現在のディレクトリを設定しなくてはなりません。

構文

```
ChangeDirectory ( szPath );
```

パラメーター

テーブル 36 - ChangeDirectory のパラメーター

パラメーター	説明
szPath	現在のフォルダーとして設定するディレクトリの名前を指定します。その名前は完全修飾名、あるいは UNC パスのいづれでも構いませんが、末尾に円記号を含むことはできません。必要に応じて、ChangeDirectory の前に StrRemoveLastSlash を呼び出します。 ***メモ -*** *スクリプトでファイルを指定する場合、適切な値を持つ現在のフォルダーに頼らず、常に (適切な InstallShield システム変数、例えば SRCDIR を使って) 完全パスを指定してください。スクリプトは現在のフォルダーを変更することが可能なコードを内部で実行するため、その値は必ずしも予期したものとは限りません。*

戻り値

テーブル 37 - ChangeDirectory の戻り値

戻り値	説明
0	ChangeDirectory は指定したディレクトリを現在のディレクトリに設定しました。
< 0	ChangeDirectory は指定したディレクトリを現在のディレクトリに設定することができませんでした。

ChangeDirectory の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ChangeDirectory 関数のデモンストレーションを行います。
*
* このスクリプト例では、Windows フォルダーを現在の
```

```
＊ディレクトリとし、NotePad を起動して Readme.txt
＊ファイルを表示します。
＊
¥*------------------------------------------------------------*/
// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ChangeDirectory(HWND);

function ExFn_ChangeDirectory(hMSI)
begin

// Windows フォルダーをデフォルトのディレクトリとします。ここで
// InstallShield システム変数 WINDIR は
// Windows フォルダーをポイントしていることに注意して下さい。


ChangeDirectory (WINDIR);
// Notepad を起動して Windows Readme.txt ファイルを参照します。
   LaunchApp ("Notepad.exe", "Readme.txt" );

end;
```

CharReplace

CharReplace 関数は、cFind 文字の全インスタンスを svString 文字列の cReplace に置換します。文字列インデックスが nStart 以下の文字は例外となります。文字列の最初の文字の文字列インデックスは 0 です。

構文

```
CharReplace ( svString, cFind, cReplace, nStart );
```

パラメーター

テーブル 38 · CharReplace のパラメーター

パラメーター	説明
svString	文字を置換する文字列を指定し、変更済み文字列を戻します。
cFind	置換する文字を指定します。STRTOCHAR 関数を使って、char 型リテラルを cFind と指定します。次の例を参照してください。 CharReplace(svString, STRTOCHAR('a'), STRTOCHAR('e'), nStart);
cReplace	cFind を置換する文字を指定します。STRTOCHAR 関数を使って、char 型リテラルを cReplace と指定します。次の例を参照してください。 CharReplace(svString, STRTOCHAR('a'), STRTOCHAR('e'), nStart);
nStart	cFind の検索開始位置にある文字列インデックスを指定します。svString の最初の文字の文字列インデックスは 0 (ゼロ) です。svString の cFind のすべてのインスタンスを置換する場合、nStart に 0 を指定します。

戻り値

テーブル 39 · CharReplace の戻り値

戻り値	説明
X	cReplace による cFind の置換合計数。
< ISERR_SUCCESS	文字の置換に失敗した関数。

追加情報

cFind または cReplace にはヌル文字 ('¥0') が可能です。ヌル区切り文字列を処理するには、cFind または cReplace を StrToChar('¥0') と指定します。ここでは定数 NULL は '¥0' ではなく 0 であること、またヌル文字を指定することができない点に注意してください。cFind または cReplace が '¥0' の時、CharReplace は文字列バッファーを戻す前に自動的に最後の 2 文字を '¥0' に設定します。従って、文字列のサイズ (明確に設定しなくてはなりません) は少なくとも保存する文字数よりも 2 文字長くなくてはなりません。

CharReplace の例

***メモ·** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
//---------------------------------------------------------------------------
//
// InstallScript スクリプト例
//
```

```
CharReplace 関数のデモンストレーションを行います。
//
このサンプルは、各バックスラッシュ 文字をスラッシュに置換前と変換後の
サンプル パス文字列を表示します。
//
//---------------------------------------------------------------------------

function OnBegin()
    STRING path_to_convert;
begin

// 変換するパスの例
path_to_convert = FOLDER_COMMON_APPDATA;

MessageBox(" 変換前のパス : " + path_to_convert,
    INFORMATION);

// バックスラッシュをスラッシュに置換します
CharReplace(path_to_convert,
    STRTOCHAR('¥¥'), STRTOCHAR('/'), 0);

MessageBox(" 変換後のパス : " + path_to_convert,
    INFORMATION);

end;
```

CloseFile

CloseFile 関数は OpenFile への呼び出しで開いたファイルを閉じます。ファイルを閉じた後に読み書きはできません。

構文

```
CloseFile ( nvFileHandle );
```

パラメーター

テーブル 40 - CloseFile のパラメーター

パラメーター	説明
nvFileHandle	閉じるファイルのハンドルを指定します。

戻り値

テーブル 41 - CloseFile の戻り値

戻り値	説明
0	関数が問題なくファイルを閉じたことを示します。
< 0	関数がファイルを閉じることができなかったことを示します。

CloseFile の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* OpenFile 関数 と CloseFile 関数 のデモンストレーションを行います。
*
* OpenFile はファイルを開くために呼び出されて、リストへ
* 読み込まれます。リストが表示されます。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*      既存のディレクトリ内の既存ファイルを参照するように
*      設定します。
*
¥*---------------------------------------------------------*/

#define EXAMPLE_FILE "Readme.txt"
#define EXAMPLE_DIR "C:¥¥Windows"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CloseFile(HWND);

function ExFn_CloseFile(hMSI)
    STRING  svLine;
    NUMBER nvFileHandle;
```

```
    LIST    listID;
begin

    // ファイル モードを通常に設定します。
    OpenFileMode (FILE_MODE_NORMAL);

    // テキスト ファイルを開きます。
    if (OpenFile (nvFileHandle, EXAMPLE_DIR, EXAMPLE_FILE) < 0) then
        MessageBox ("OpenFile が失敗しました。", SEVERE);
        abort;
    endif;

    // 空白文字列リストを作成します。
    listID = ListCreate (STRINGLIST);

    // テキスト ファイルの行を文字列リストへ読み出します。
    while GetLine (nvFileHandle, svLine) = 0
        ListAddString (listID, svLine, AFTER);
    endwhile;

    // ファイルを閉じます。
    if (CloseFile (nvFileHandle) < 0) then
        MessageBox ("CloseFile が失敗しました。", SEVERE);
    endif;

    // ファイルから読み出されたテキストを表示します。
    SdShowInfoList ("","",listID);

end;
```

CmdGetHwndDlg

CmdGetHwndDlg 関数は ｓ ｚ DialogName が認識したダイアログのウィンドウ ハンドルを取得します。ダイアログは **EzDefineDialog** または **DefineDialog** を使って既に定義済みで、**WaitOnDialog** を呼び出して初期化されなくてはなりません。

CmdGetHwndDlg は一般的にカスタム ダイアログ用に DLG_INIT ルーチンで呼び出されます。ダイアログのハンドルは HWND 変数に割り当てられ、必要とする他の関数によって利用されます。

メモ・ダイアログが WaitOnDialog 関数を使って初期化され、ウィンドウハンドルがそれに割り当てられている場合、そのハンドルは EndDialog への呼び出しによって閉じられない限り、ダイアログのみに関連付けられます。WaitOnDialog を呼び出して以前スクリプトで開閉したダイアログを開いた時、新しいハンドルを取得するためには CmdGetHwndDlg をもう一度呼び出さなくてはなりません。古いハンドルは既に無効です。

構文

```
CmdGetHwndDlg ( szDialogName );
```

パラメーター

テーブル 42 - CmdGetHwdDlg のパラメーター

パラメーター	説明
szDialogName	**EzDefineDialog** または **DefineDialog** を使って定義されたダイアログを指定します。

戻り値

テーブル 43 - CmdGetHwdDlg の戻り値

戻り値	説明
> 0	szDialogName で指定されたダイアログのウィンドウハンドル。
< 0	**CmdGetHwndDlg** はハンドルを読み出せませんでした。szDialogName が適切に定義され初期化されたダイアログを参照するよう確認します。

CmdGetHwndDlg の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CmdGetHwndDlg 関数のデモンストレーションを行います。
*
*  このスクリプト例ではカスタム ダイアログを表示します。ダイアログ ボックス
* 初期化で、スクリプトは次の処理を実行するために
*  ダイアログのウィンドウ ハンドルを読み出す CmdGetHwndDlg
* を呼び出します。
*
* --- ウィンドウ タイトルバーのテキストを変更する。
* --- ダイアログ中のボタンを有効、あるいは無効にする。
* --- ダイアログ ウィンドウを最大化または元に戻すメッセージを送る。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* InstallShield Sd ダイアログで、
* ビルトイン関数の SdBitmap によって表示されます。このダイアログは
* インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
* カスタム ダイアログとしてスクリプトで利用することができるからです。
* この例の要件に合わせて、
* 静的テキストの [ 戻る ] および [ 次へ ] ボタンをスクリプトが変更することに注意して下さい。
*
¥*---------------------------------------------------------*/

// ダイアログのコントロール
```

```
#define RES_DIALOG_ID       12027   // カスタム ダイアログの ID
#define RES_PBUT_RESTORE1 // ダイアログの [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL 9 // ダイアログの [ キャンセル ] ボタンの ID
#define RES_PBUT_MAXIMIZE12 // ダイアログの [ 戻る ] ボタンの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CmdGetHwndDlg(HWND);

function ExFn_CmdGetHwndDlg(hMSI)
    STRING szDialogName;
    NUMBER nResult, nCmdValue, hwndDlg;
    BOOL   bDone;
    HWND hwndDlg;
begin

    // このインストールでカスタム ダイアログを認識するための名前を指定します。
    //
    szDialogName = "CustomDialog";

    // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
    // _ISUSER.DLL または _ISRES.DLL からダイアログを取得します。ヌル文字列を
    // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
    // 4 番目のパラメーターにある ID によって識別されるためです。
    nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

    if (nResult < 0) then
        // エラーを報告し、終了します。
        MessageBox ("ダイアログの定義エラー", SEVERE);
        abort;
    endif;

    // ループを制御するのに使われるインジケーターを初期化します。
    bDone = FALSE;

    // 完了するまでループします。
    repeat
        // ダイアログを表示して次のダイアログ イベントを戻します。
        nCmdValue = WaitOnDialog (szDialogName);

        // イベントに応答します。
        switch (nCmdValue)
            case DLG_CLOSE:
                // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
                Do (EXIT);
            case DLG_ERR:
                MessageBox ("ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
                abort;
            case DLG_INIT:
                // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を
                // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
                // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
                // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
                hwndDlg = CmdGetHwndDlg (szDialogName);
                SdGeneralInit(szDialogName, hwndDlg, 0, "");

                // ボタンのスタティック テキストを設定します。
                CtrlSetText (szDialogName, RES_PBUT_MAXIMIZE, "最大化 (&M)");
```

```
            CtrlSetText (szDialogName, RES_PBUT_RESTORE, " 元に戻す (&R)");

            // Winsub からの呼び出しを利用して [ 元に戻す ] ボタンを無効にします。
            _WinSubEnableControl (hwndDlg, RES_PBUT_RESTORE, 0);
        case RES_PBUT_RESTORE:
            // ウィンドウを標準サイズに戻します。
            SendMessage (hwndDlg, WM_SYSCOMMAND, SC_RESTORE, 0);

            // Winsub からの呼び出しを利用して [ 元に戻す ] ボタンを無効にします。
            _WinSubEnableControl (hwndDlg, RES_PBUT_RESTORE, 0);

            // Winsub からの呼び出しを利用して [ 最大化 ] ボタンを有効にします。
            _WinSubEnableControl (hwndDlg, RES_PBUT_MAXIMIZE, 1);
        case RES_PBUT_MAXIMIZE:
            // ダイアログのウィンドウを最大化します。
            SendMessage (hwndDlg, WM_SYSCOMMAND, SC_MAXIMIZE, 0);

            // Winsub からの呼び出しを利用して [ 最大化 ] ボタンを無効にします。
            _WinSubEnableControl (hwndDlg, RES_PBUT_MAXIMIZE, 0);

            // Winsub からの呼び出しを利用して [ 元に戻す ] ボタンを有効にします。
            _WinSubEnableControl (hwndDlg, RES_PBUT_RESTORE, 1);
        case RES_PBUT_CANCEL:
            // ユーザーが [ キャンセル ] ボタンをクリックしました。
            Do (EXIT);
    endswitch;

    until bDone;

    // ダイアログを閉じます
    EndDialog (szDialogName);

    // メモリからダイアログを解放します。
    ReleaseDialog (szDialogName);

end;
```

CoCreateObject

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

CoCreateObject 関数は、szProgID によって名付けられた COM オブジェクトを初期化し、設定されたキーワードを使用して OBJECT 変数タイプに割り当てることができるリファレンスを戻します。

構文

```
CoCreateObject ( szProgID );
```

パラメーター

テーブル 44 - CoCreateObject のパラメーター

パラメーター	説明
szProgID	COM オブジェクトのプログラム ID が初期化されるよう指定します。

戻り値

設定されたキーワードを使用して、OBJECT という変数タイプに割り当てられるリファレンス。

追加情報

- オブジェクトが正常に初期化されたかどうかを確認するには、IsObject 関数を呼び出します。
- オブジェクト変数を NOTHING の値に設定するか、**CoCreateObject**、**CoCreateObjectDotNet**、**CoGetObject** または **DotNetCoCreateObject** 関数を使用してオブジェクトを再割り当てすると、任意のオブジェクト変数を解放することができます。ただし、これによってオブジェクトが参照するライブラリが自動的にロード解除されるわけではありません。Windows API の CoFreeLibrary を手動で呼び出してライブラリを解放する必要があります。そうしないとライブラリは、インストールが終了するまでロードされたままになります。詳細は、「COM オブジェクトを使用してインストールを拡張する」を参照してください。

CoCreateObjectDotNet

***プロジェクト - CoCreateObjectDotNet** 関数は、次のプロジェクト タイプでサポートされています:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript カスタム アクションを含む基本の MSI*

CoCreateObjectDotNet 関数は現在使用されていません。この関数の呼び出しは、szAppDomain パラメーターにヌル文字列 ("") を使って **DotNetCoCreateObject** 関数を呼び出すのと同じです。

詳細については、「DotNetCoCreateObject」を参照してください。

CoGetObject

プロジェクト - この情報は、InstallScript プロジェクトに適用します。

CoGetObject 関数は指定した COM オブジェクトへリファレンスを戻します (Visual Basic の GetObject 関数と同様)。このリファレンスは、設定されたキーワード を利用して変数タイプ OBJECT へ割り当てることができます。

構文

```
CoGetObject ( szFilename, szProgID );
```

パラメーター

テーブル 45 · CoGetObject のパラメーター

パラメーター	説明
szFilename	COM オブジェクトの完全修飾名を指定します。szProgID がヌルではない場合、パラメーターにはヌル文字列 ("") が可能です。
szProgID	COM オブジェクトのプログラム ID を指定します。szFilename がヌルではない場合、パラメーターにはヌル文字列 ("") が可能です。

戻り値

- 設定されたキーワードを使用して、OBJECT 型の変数に割り当てられるリファレンス。
- オブジェクト変数を NOTHING の値に設定するか、**CoCreateObject**、**CoCreateObjectDotNet**、**CoGetObject** または **DotNetCoCreateObject** 関数を使用してオブジェクトを再割り当てすると、任意のオブジェクト変数を解放することができます。ただし、これによってオブジェクトが参照するライブラリが自動的にロード解除されるわけではありません。Windows API の CoFreeLibrary を手動で呼び出してライブラリを解放する必要があります。そうしないとライブラリは、インストールが終了するまでロードされたままになります。詳細は、「COM オブジェクトを使用してインストールを拡張する」を参照してください。

追加情報

オブジェクトが正常に初期化されたかどうかを確認するには、IsObject 関数を呼び出します。

CoGetObject の例

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CoGetObject 関数のデモンストレーションを行います。
*
* この例では、IIS サーバー上での仮想ディレクトリの作成方法を
* 説明します。
*
¥*---------------------------------------------------------*/

#include "ifx.h"

#define VIRTUALDIR "My Virtual Dir"
#define VIRTUALDIRPATH "c:¥inetpub¥wwwroot¥MyDir"

function OnBegin()
OBJECT objIIS_Root, objVirtDir;

begin

  set objIIS_Root = CoGetObject("IIS://localhost/W3SVC/1/Root", "");
  if (IsObject(objIIS_Root)) then
```

```
    try
      set objVirtDir = objIIS_Root.Create("IISWebVirtualDir", VIRTUALDIR);
      if (IsObject(objVirtDir)) then
        objVirtDir.Path = VIRTUALDIRPATH;
        objVirtDir.AccessRead = TRUE;
        objVirtDir.AccessScript = TRUE;
        objVirtDir.SetInfo();
        objVirtDir.AppCreate(TRUE);
        objVirtDir.SetInfo();
      endif;
    catch
      MessageBox(" 仮想ディレクトリを作成できませんでした。", INFORMATION);
    endcatch;
  endif;

end;
```

ConfigAdd

ConfigAdd 関数は、ConfigFileLoad を使ってメモリにロードされたシステム構成ファイルにステートメントを追加します。参照キーに関連するステートメントの位置を指定する、あるいはステートメントをファイルの最初の行または最終行として追加することもできます。ファイルに既存する行を置換することもできます。

***メモ・** ConfigAdd を呼び出す前に、まず ConfigFileLoad を呼び出してシステム構成ファイルをメモリへロードしなくてはなりません。ファイルを編集した後、ConfigFileSave を呼び出してファイルを保存します。*

Ez 構成ファイル関数を拡張構成ファイル関数と同時に使用しないでください。ConfigFileLoad を呼び出した後、ConfigFileSave を呼び出して変更を保存するまで、簡易構成ファイル関数を利用することはできません。

構文

```
ConfigAdd ( szKey, szValue, szRefKey, nOptions );
```

パラメーター

テーブル 46 · ConfigAdd のパラメーター

パラメーター	説明
szKey	システム構成ファイルに追加されたステートメントでキーワードを指定します。
szValue	システム構成ファイルに追加されたキーワードの値を指定します。
szRefKey	システム構成ファイルで s z Key を追加するのに関連する参照キーを指定します。パラメーターでヌル文字列 ("") を渡したとき、その文字列は nOptions で渡された定義済み定数に従ってファイルの最初または最後の行として追加されます。
nOptions	その行を参照キーを含む行の前後いづれに追加するのか、あるいは既存する行を置換するのかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • BEFORE ― s z RefKey を含む行の前にステートメントが追加されます。szRefKey がヌル文字列 ("") の場合、ステートメントはファイルの最初の行として追加されます。 • AFTER ― s z RefKey を含む行の後にステートメントが追加されます。szRefKey がヌル文字列 ("") の場合、ステートメントはファイルの最後の行として追加されます。 • REPLACE ― ステートメントはファイルの既存行を置換します。複数の行に同じキーがある場合、最後の行のみが置換されます。置換する行がファイルに存在しない場合、新しい行がファイルの最後の行として追加されます。

戻り値

テーブル 47 · ConfigAdd の戻り値

戻り値	説明
0	ConfigAdd がステートメントを指定されたシステム構成ファイルへ追加しました。
< 0	ConfigAdd がステートメントを指定されたシステム構成ファイルへ追加できませんでした。

追加情報

ConfigAdd 関数がシステム構成ファイルの行を置換する時、2 つの行の参照キーを比較します。参照キーは行を認

識するサブ文字列です。例えば、次のステートメントの参照キーは Kybrd.drv です。

```
DEVICE=C:¥Windows¥System¥Kybrd.drv /1024 /C:345
```

次のステートメントでは、参照キーは PATH です :

```
SET PATH=C:¥Windows;C:¥Windows¥System
```

ConfigAdd の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ConfigAdd 関数のデモンストレーションを行います。
*
* このスクリプト例では、構成ファイルへ 2 つのステートメントを追加し
* ます。まず、編集のためにファイルをロードする ConfigFileLoad を
* 呼び出します。次に、DEVICE ステートメントを追加します。その後、
* DEVICEHIGH ステートメントを追加します。最後に、元のファイルを
* バックアップして編集したファイルを保存します。
*
* メモ : このスクリプトを実行する前に、C ドライブのルートへ
*      ISExampl.sys と名づけられた構成ファイルを作成します。
*      ファイルは次の行を含みます :
*
*      DEVICE=C:¥Exapp¥Exapp.sys
*      DEVICE=C:¥Otherapp.exe
*
¥*---------------------------------------------------------*/

#define EXAMPLE_SYS "C:¥¥ISExampl.sys"
#define EXAMSYS_BAK "ISExampl.bak"

  // ConfigAdd へパラメーターとして渡す変数。

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ConfigAdd(HWND);

function ExFn_ConfigAdd(hMSI)
  STRING szKey, szValue, szRefKey;
begin

  // ターゲット構成ファイルをメモリへロードします。
  if ConfigFileLoad (EXAMPLE_SYS) < 0 then
     MessageBox (EXAMPLE_SYS + " をロードできませんでした。", SEVERE);
     abort;
  endif;

  // ConfigAdd の最初の呼び出し用のパラメーターをセットアップします。
```

```
    szKey    = "DEVICE";
    szValue  = "C:¥¥Exapp¥¥Exapp2.sys";
    szRefKey = "Exapp.sys";

    //  Exapp.sys を参照する初めのステートメントの前に
    // DEVICE=C:¥Exapp¥Exapp2.SYS 行を追加します。
    if (ConfigAdd (szKey, szValue, szRefKey, BEFORE) < 0) then
        MessageBox ("ConfigAdd の最初の呼び出しに失敗しました。", WARNING);
        abort;
    endif;

    // ConfigAdd の 2 番目の呼び出し用のパラメーターをセットアップします。
    szKey    = "DEVICEHIGH";
    szValue  = "C:¥¥Otherapp¥¥Otherapp.exe";
    szRefKey = "Otherapp.exe";

    // ステートメント DEVICEHIGH=C:¥Otherapp¥Otherapp.exe と共に
    // OtherApp.exe を参照する既存の最終行を置換します。
    if (ConfigAdd (szKey, szValue, szRefKey, REPLACE) < 0) then
        MessageBox ("ConfigAdd の 2 番目の呼び出しに失敗しました。", WARNING);
        abort;
    endif;

    // 元のファイルをバックアップして、編集したファイルを保存します。
    if ConfigFileSave (EXAMSYS_BAK) < 0 then
        MessageBox (EXAMPLE_SYS +" を保存できませんでした。", SEVERE);
    else
        MessageBox (EXAMPLE_SYS + " を更新し、保存しました。",INFORMATION);
    endif;

end;
```

ConfigDelete

ConfigDelete 関数は、ConfigFileLoad への呼び出しでメモリにロードされたシステム構成ファイルから行を削除します。パラメーター szKey は削除する行を特定する参照キーを指定します。システム構成ファイルを編集するために拡張構成関数を利用した後には、変更を保存するために ConfigFileSave を呼び出さなくてはなりません。

メモ・簡易バッチファイル関数を拡張バッチファイル関数と同時に使用しないでください。BatchFileLoad を呼び出した後、BatchFileSave を呼び出してファイルを保存するまで簡易バッチファイル関数を使用することはできません。

構文

```
ConfigDelete ( szKey );
```

パラメーター

テーブル 48 - ConfigDelete のパラメーター

パラメーター	説明
szKey	削除する 1 行または複数行を特定する参照キーを指定します。Himem.sys、FILES、および STACKS を含む共通参照キー。

戻り値

テーブル 49 - ConfigDelete の戻り値

戻り値	説明
0	ConfigDelete がシステム構成ファイルから参照キーを含む行を削除しました。
< 0	ConfigDelete は指定された行を削除できませんでした。

ConfigDelete の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ConfigDelete 関数のデモンストレーションを行います。
*
* このスクリプト例では、ステートメントを構成ファイルから削除し
* ます。まず、編集のためにファイルをロードする ConfigFileLoad を
* 呼び出します。次に、FILES ステートメントを
* 含む行を削除します。最後に、元のファイルを
* バックアップして編集したファイルを保存します。
*
* メモ : このスクリプトを実行する前に、C ドライブのルートへ
*      ISExampl.sys と名づけられた構成ファイルを作成します。
*      このファイルは少なくとも 1 つ以上 FILES ステートメント含まなくてはなりません。
*
¥*---------------------------------------------------------------*/

#define TARGET_CONFIG  "C:¥¥ISExampl.sys"
#define BACKUP_CONFIG  "ISExampl.bak"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"
```

```
    export prototype ExFn_ConfigDelete(HWND);

function ExFn_ConfigDelete(hMSI)
   STRING szMsg;
begin

   // 編集するターゲット構成ファイルをロードします。
   if (ConfigFileLoad (TARGET_CONFIG) < 0) then
      MessageBox (TARGET_CONFIG + " をロードできませんでした。", SEVERE);
      abort;
   endif;

   // キー "FILES" を含むファイルにある行をすべて削除します。
   if (ConfigDelete ("FILES") < 0) then
      MessageBox ("ConfigDelete が失敗しました。", SEVERE);
   else
      // 元のファイルをバックアップして、編集したファイルを保存します。
      if ConfigFileSave (BACKUP_CONFIG) < 0 then
         MessageBox (TARGET_CONFIG + " を保存できませんでした。", SEVERE);
      else
         MessageBox (TARGET_CONFIG + " を更新し、保存しました。",INFORMATION);
      endif;
   endif;

end;
```

ConfigFileLoad

ConfigFileLoad 関数は指定したシステム構成ファイルのコピーをメモリにロードし、ファイル上での操作のためにその他の拡張構成ファイル関数を呼び出せるようにします。szConfigFile で編集するシステム構成ファイルの名前を指定するか、szConfigFile でヌル文字列 ("") を渡してデフォルトのシステム構成ファイルを編集します。これはシステムが利用する **Config.sys** ファイルを起動するために最初にインストールが設定するものです。

メモ・拡張構成ファイル関数を利用する前に、ConfigFileLoad を呼び出してシステム構成ファイルをメモリにロードしなくてはなりません。ファイルを変更した後、ConfigFileSave を呼び出してディスクに保存します。デフォルトシステム構成ファイルの完全修飾名を取得するには、ConfigGetFileName を呼び出します。別のファイルをデフォルトシステム構成ファイルとするには、ConfigSetFileName を呼び出します。

*新規構成ファイルを作成するのに **ConfigFileLoad** を呼び出すことはできません。新しい構成ファイルを作成するには、CreateFile と CloseFile を使用します。これは、空白のファイルを作成します。その後、**ConfigFileLoad** およびその他の関数を使って、必要に応じてファイルをロードおよび変更します。*

*Ez 構成ファイル関数を拡張構成ファイル関数と同時に使用しないでください。**ConfigFileLoad** 関数を呼び出した後、**ConfigFileSave** を使って変更を保存するまで、Ez 構成ファイル関数を利用することはできません。*

構文

```
ConfigFileLoad ( szConfigFile );
```

パラメーター

テーブル 50 - ConfigFileLoad のパラメーター

パラメーター	説明
szConfigFile	メモリにロードするシステム構成ファイルの完全修飾名を指定します。デフォルトシステム構成ファイルをロードするには、このパラメーターにヌル文字列 ("") を渡します。

戻り値

テーブル 51 - ConfigFileLoad の戻り値

戻り値	説明
0	ConfigFileLoad が構成ファイルバッファーを初期化しました。szConfigFile が既存構成ファイルを指定する場合、ファイルはバッファーにロードされます。それ以外の場合は空のバッファーが作成されます。
< 0	ConfigFileLoad は構成ファイルバッファーを初期化することができませんでした。

ConfigFileLoad の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ConfigFileLoad 関数と ConfigFileSave 関数のデモンストレーションを行います。
*
* このスクリプト例では、編集するために構成ファイルを開く方法、
* オリジナルファイルのバックアップ作成方法、
* そして編集したファイルの保存方法とその閉じ方を説明します。
*
* ConfigFileSave のファイルバックアップ機能がどのように既存ファイルの上書きを
* 防ぐのかをデモンストレーションします。このスクリプトは 2 つの
* 異なる構成ファイルをロードし保存します。最初のファイルは
* 特定のファイル名と共にバックアップされます。2 番目のファイルは
* ワイルドカード拡張子を使ってバックアップされ、ConfigFileSave が
* 3 桁の一意のファイル拡張子を生成します。
*
* メモ : このスクリプトを実行する前に、C ドライブ のルートへ
*      2 つのファイル (ISExamp1.sys と ISExamp2.sys) を作成します。
*      効果的に行うためには、ISExamp1.* または ISExamp2.* と
*      名づけられた他のファイルを削除または移動する必要があります。
```

```
 *
¥*---------------------------------------------------------------*/

// この例で使われている構成ファイルとバックアップ ファイルの名前。
#define EXAMPLE1 "ISEXAMP1"
#define EXAMPLE2 "ISEXAMP2"

// 構成ファイルの完全修飾名。
#define EXAMPLE1_SYS "C:¥¥" + EXAMPLE1 + ".sys"
#define EXAMPLE2_SYS "C:¥¥" + EXAMPLE2 + ".sys"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ConfigFileLoad(HWND);

function ExFn_ConfigFileLoad(hMSI)
begin

  // EXAMPLE1_SYS をロードします。
  if (ConfigFileLoad (EXAMPLE1_SYS) < 0) then
    MessageBox (EXAMPLE1_SYS + " をロードできませんでした。", SEVERE);
    abort;
  endif;

  // 最初のファイルを編集するには、ここに他の構成関数を使います。

  // 拡張子 'bak' を持つ元のファイルのバックアップを行い、
  // 編集されたファイルを元の名前の下に保存します。ISExamp1.bak が既に存在する場合、
  // ConfigFileSave は数字の付いた拡張子を生成します。
  if (ConfigFileSave (EXAMPLE1 + ".bak") < 0) then
    MessageBox (EXAMPLE1_SYS + " を保存できませんでした。", SEVERE);
    abort;
  else
    MessageBox (EXAMPLE1_SYS + " が保存されました。",INFORMATION);
  endif;

  // EXAMPLE2_SYS をロードします。
  if (ConfigFileLoad (EXAMPLE2_SYS) < 0) then
    MessageBox (EXAMPLE2_SYS + " をロードできませんでした。", SEVERE);
    abort;
  endif;

  // 2 番目のファイルを編集するには、ここに他の構成関数を使います。

  // 数字の付いた拡張子を使って元の構成ファイルをバックアップします。
  // そして編集済みファイルを元の名前の下に保存します。
  if (ConfigFileSave (EXAMPLE2 + ".*")< 0) then
    MessageBox (EXAMPLE2_SYS + " を保存できませんでした。", SEVERE);
    abort;
  else
    MessageBox (EXAMPLE2_SYS + " が保存されました。",INFORMATION);
  endif;

end;
```

ConfigFileSave

ConfigFileSave は ConfigFileLoad 関数を使ってメモリにロードしたシステム構成ファイルをディスクへ保存します。ファイルはオリジナル名の元に保存されます。szBackupFile でファイル名が指定された場合、編集されたファイルがディスクに書き込まれる前にオリジナルファイル名はそのファイル名に変更されます。szBackupFile にヌル文字列 ("") が含まれる場合、オリジナルファイルは変更されたファイルに置換されます。拡張構成ファイル関数を利用してシステム構成ファイルの編集を完了したときに ConfigFileSave を呼び出さ無かった場合、すべての変更点は失われます。

メモ・Ez 構成ファイル関数を拡張構成ファイル関数と同時に使用しないでください。ConfigFileLoad 関数を呼び出した後、ConfigFileSave を使って変更を保存するまで、簡易構成ファイル関数を利用することはできません。

構文

```
ConfigFileSave ( szBackupFile );
```

パラメーター

テーブル 52 - ConfigFileSave のパラメーター

パラメーター	説明
szBackupFile	編集する前の状態でオリジナルファイルのバックアップコピーを保存するか否かを、次の基準に従って指定します : ・ バックアップファイルを作成しない場合、このパラメーターでヌル文字列を指定します。 ・ オリジナルファイルを特定の名前でバックアップするには、そのファイル名をこのパラメーターで渡します。ファイル名は不完全修飾 (つまり、ドライブ及び / またはパスを指定しない) でなくてはなりません。指定した名前のファイルが既に存在する場合、ConfigFileSave が次の箇条書きリストに述べられた固有のファイル拡張子を生成することに注意してください。 ・ オリジナルファイルをインストールが生成したファイル拡張子とともにバックアップする場合は、ワイルドカード文字 (*) をファイル拡張子として指定します (例えば、"Config.*") 。そしてインストールが 001 から始まる数値を拡張子として割り当てます。その拡張子を含むファイルが既に存在する場合、固有のファイル名が作成されるまで拡張子の値が 1 づつ追加されます。 バックアップファイルが作成された後、InstallShield はシステム変数 INFOFILENAME でバックアップファイル名を保存します。

戻り値

テーブル 53 - ConfigFileSave の戻り値

戻り値	説明
0	ConfigFileSave ファイルをメモリからディスクへ保存しました。
< 0	ConfigFileSave はファイルをディスクに保存することができませんでした。

ConfigFileSave の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
```

```
    * InstallShield スクリプトの例
    *
    * ConfigFileLoad 関数と ConfigFileSave 関数のデモンストレーションを行います。
    *
    * このスクリプト例では、編集するために構成ファイルを開く方法、
    * オリジナルファイルのバックアップ作成方法、
    * そして編集したファイルの保存方法とその閉じ方を説明します。
    *
    * ConfigFileSave のファイルバックアップ機能がどのように既存ファイルの上書きを
    * 防ぐのかをデモンストレーションします。このスクリプトは 2 つの
    * 異なる構成ファイルをロードし保存します。最初のファイルは
    * 特定のファイル名と共にバックアップされます。2 番目のファイルは
    * ワイルドカード拡張子を使ってバックアップされ、ConfigFileSave が
    * 3 桁の一意のファイル拡張子を生成します。
    *
    * メモ : このスクリプトを実行する前に、C ドライブ のルートへ
    *      2 つのファイル (ISExamp1.sys と ISExamp2.sys) を作成します。
    *      効果的に行うためには、ISExamp1.* または ISExamp2.* と
    *      名づけられた他のファイルを削除または移動する必要があります。
    *
    ¥*---------------------------------------------------------*/

    //  この例で使われている構成ファイルとバックアップ ファイルの名前。
    #define EXAMPLE1 "ISEXAMP1"
    #define EXAMPLE2 "ISEXAMP2"

    // 構成ファイルの完全修飾名。
    #define EXAMPLE1_SYS "C:¥¥" + EXAMPLE1 + ".sys"
    #define EXAMPLE2_SYS "C:¥¥" + EXAMPLE2 + ".sys"

    // ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
    #include "Ifx.h"

        export prototype ExFn_ConfigFileSave(HWND);

    function ExFn_ConfigFileSave(hMSI)
    begin

       // EXAMPLE1_SYS をロードします。
       if (ConfigFileLoad (EXAMPLE1_SYS) < 0) then
          MessageBox (EXAMPLE1_SYS + " をロードできませんでした。", SEVERE);
          abort;
       endif;

       // 最初のファイルを編集するには、ここに他の構成関数を使います。

       // 拡張子 'bak' を持つ元のファイルのバックアップを行い、
       // 編集されたファイルを元の名前の下に保存します。ISExamp1.bak が既に存在する場合、
       // ConfigFileSave は数字の付いた拡張子を生成します。
       if (ConfigFileSave (EXAMPLE1 + ".bak") < 0) then
          MessageBox (EXAMPLE1_SYS + " を保存できませんでした。", SEVERE);
          abort;
       else
          MessageBox (EXAMPLE1_SYS + " が保存されました。",INFORMATION);
       endif;

       // EXAMPLE2_SYS をロードします。
       if (ConfigFileLoad (EXAMPLE2_SYS) < 0) then
          MessageBox (EXAMPLE2_SYS + " をロードできませんでした。", SEVERE);
```

```
        abort;
    endif;

    // 2 番目のファイルを編集するには、ここに他の構成関数を使います。

    // 数字の付いた拡張子を使って元の構成ファイルをバックアップします。
    // そして編集済みファイルを元の名前の下に保存します。
    if (ConfigFileSave (EXAMPLE2 + ".*")< 0) then
        MessageBox (EXAMPLE2_SYS + " を保存できませんでした。", SEVERE);
        abort;
    else
        MessageBox (EXAMPLE2_SYS + " が保存されました。",INFORMATION);
    endif;

end;
```

ConfigFind

ConfigFind は ConfigFileLoad 関数を使ってメモリにロードしたシステム構成ファイルを検索します。szRefKey パラメーターは、そのファイルにある検索ターゲットを指定する参照キーです。参照キーが検出されると、その値が svResult で戻されます。すべての szRefKey を検出するには、この関数を nOptions で CONTINUE に設定して繰り返し呼び出します。ファイルの上部から検出を再開しするには、nOptions で 定数 RESTART を指定します。ファイルを編集した後、ConfigFileSave を呼び出して保存します。

メモ・Ez 構成ファイル関数を拡張構成ファイル関数と同時に使用しないでください。ConfigFileLoad 関数を呼び出した後、ConfigFileSave を使って変更を保存するまで、簡易構成ファイル関数を利用することはできません。

構文

```
ConfigFind (szRefKey, svResult, nOptions);
```

パラメーター

テーブル 54 ・ ConfigFind のパラメーター

パラメーター	説明
szRefKey	検索する参照キーを指定します。参照キーが、ファイル拡張子を持たないファイル名の場合、すべてのファイル拡張子が検索に含まれます。例えば、Win.com を指定したとき、検索はその参照キーのみを捜します。WIN を指定した場合、ファイル Win.exe、Win.dll、Win.sys、などすべてが戻されます。
svResult	システム構成ファイルで検出された参照キーの値を戻します。
nOptions	ファイルの最初から検索を始めるのか、前回の検索終了地点から検索を続行するのかを指示します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ RESTART— ファイルの最初から検索を開始します。 ・ CONTINUE— システム構成ファイルでの現在の位置から検索をはじめます。 ・ COMMAND—szRefKey の参照キーが環境変数ではなく、コマンドであることを示します。次の例の通り、ビット単位 OR 演算子 (|) を利用して定数 COMMAND を RESTART や CONTINUE と組み合わせることができます。 ConfigFind("Vga.drv", svResult, CONTINUE | COMMAND); ***メモ・システム構成ファイルは、環境変数とコマンドのどちらも含むことが可能です。同じ名前を持つ環境変数とコマンドをを区別するには、定数 COMMAND を使って実行コマンドを検出中であることを指定します。***

戻り値

テーブル 55 - ConfigFind の戻り値

戻り値	説明
0	ConfigFind は指定された参照キーを検出して svResult で戻しました。
< 0	ConfigFind は指定された参照キーを検出することができませんでした。

ConfigFind の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ConfigFind 関数のデモンストレーションを行います。
*
* このスクリプト例では、バッチファイルを検索してそのファイルが
* BUFFERS コマンドを含むかどうかをレポートします。そして
* Abc44.sys が参照するすべてのコマンドを表示します。
*
* メモ : このスクリプトを実行する前に、C ドライブのルートへ
*      ISExampl.sys と呼ばれる構成ファイルを作成します。
*      構成ファイルには、次の行を含みます :
*    含みます :
*
*      DEVICE=C:¥Abc44.sys /e:300
*      DEVICE=C:¥Abc44.sys /s:off
*      BUFFERS=50
*
¥*---------------------------------------------------------*/

#define EXAMPLE_SYS "C:¥¥ISExampl.sys"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ConfigFind(HWND);

function ExFn_ConfigFind(hMSI)
   STRING svResult;
   NUMBER nResult;
begin

   // EXAMPLE_SYS. をロードします。
   if (ConfigFileLoad (EXAMPLE_SYS) < 0) then
```

```
    MessageBox (EXAMPLE_SYS + " をロードできませんでした。", SEVERE);
    abort;
endif;

// BUFFERS コマンドをチェックします。ファイル最上部で
// 検索を開始するよう、3 番目のパラメーターに RESTART を渡します。
nResult = ConfigFind("BUFFERS", svResult, RESTART);

if (nResult < 0) then
    MessageBox ("BUFFERS コマンドが見つかりませんでした。.", WARNING);
else
    MessageBox (svResult, INFORMATION);
endif;

// Abc44.sys を参照する初めのコマンドを検出します。
nResult = ConfigFind ("Abc44.sys", svResult, COMMAND | RESTART);

if nResult < 0 then
    MessageBox (" ファイル Abc44.sys は参照されません。", WARNING);
else

    // 一致するステートメントが検出される限りループします。
    while nResult = 0
        // 一致するステートメントを表示します。
        MessageBox (svResult, INFORMATION);

        // Abc44.sys を参照する次のステートメントを検出します。
        nResult = ConfigFind ("Abc44.sys", svResult, CONTINUE);
    endwhile;

    MessageBox (" これ以上 Abc44.sys で一致するものはありません。", WARNING);

endif;

end;
```

ConfigGetFileName

ConfigGetFileName 関数はデフォルトシステム構成ファイルの完全修飾名を読み出します。このファイルはもともと InstallShield によって、ターゲットシステムが起動したときに実行される Config.sys ファイルへ設定されています。デフォルトでスクリプトが別のバッチファイルを利用するように指定するには、ConfigSetFileName を呼び出します。

メモ・Ez 構成ファイル関数を拡張構成ファイル関数と同時に使用しないでください。ConfigFileLoad 関数を呼び出した後、ConfigFileSave を使って変更を保存するまで、簡易構成ファイル関数を利用することはできません。

構文

```
ConfigGetFileName (svFileName);
```

パラメーター

テーブル 56 - ConfigGetFileName のパラメーター

パラメーター	説明
svFileName	デフォルトシステム構成ファイルの完全修飾名を戻します。 ***メモ -*** *ごくまれに InstallShield がデフォルトの構成ファイルを決定できない場合があります。その場合、svFileName はヌル文字列 (""") です。*

戻り値

テーブル 57 - ConfigGetFileName の戻り値

戻り値	説明
0	ConfigGetFileName デフォルトシステム構成ファイルの完全修飾名を読み出しました。
< 0	ConfigGetFileName デフォルトシステム構成ファイルの完全修飾ファイル名の読み出しに失敗しました。

ConfigGetFileName の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ConfigGetFileName 関数 と ConfigSetFileName 関数の
* デモンストレーションを行います。
*
* このスクリプト例では、デフォルト構成ファイルの完全修飾名を読み出します。
* デフォルト構成ファイルは最初、ブート ドライブ上の
* Config.sys ファイルです。次いで、C:¥ISExampl.sys を
* デフォルトの構成ファイルにします。最後に、もう一度
* デフォルトの構成ファイルの名前を読み出して変更されたことを
* 表示します。
*
¥*---------------------------------------------------------*/

#define DEFAULT_CONFIG_FILE "C:¥¥ISExampl.sys"
```

```
// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ConfigGetFileName(HWND);

function ExFn_ConfigGetFileName(hMSI)
   STRING svFilename;
begin

   // デフォルトの構成ファイルの名前を取得して表示します。
   if (ConfigGetFileName (svFilename) < 0) then
      // エラーを報告し、中止します。
      MessageBox ("ConfigGetFileName の最初の呼び出しに失敗しました。", SEVERE);
      abort;
   else
      // デフォルトの構成ファイルの名前を表示します。
      MessageBox (" デフォルトの構成ファイルは " + svFilename + " です。",
            INFORMATION);
   endif;

   // C:¥ISExampl.sys をデフォルトの構成ファイルとします。
   if (ConfigSetFileName (DEFAULT_CONFIG_FILE) < 0) then
      // エラーを報告します。
      MessageBox (" 新しいデフォルトの構成ファイルを設定できませんでした。", SEVERE);
   else

      // デフォルトの構成ファイルが変更されたことを確認します。
      if (ConfigGetFileName (svFilename) = 0) then
         // デフォルトの構成ファイルの名前を表示します。
         MessageBox (" 現在のデフォルトの構成ファイルは " + svFilename +
               " です。", INFORMATION);
      else
         // エラーを報告します。
         MessageBox ("ConfigGetFileName の 2 回目の呼び出しに失敗しました。", SEVERE);
      endif;

   endif;

end;
```

ConfigGetInt

ConfigGetInt 関数は ConfigFileLoad 関数を使ってメモリにロードしたシステム構成ファイルから参照キーの整数値を読み出します。ConfigGetInt は、等号 (=) の右に一つだけ値を持つコマンドから値を読み出します。

***メモ・** ConfigGetInt は複数の値を持つコマンドには作動しません。たとえば、ConfigGetInt はステートメント FILES=20 を認識し、数値 20 を戻します。しかしこれはステートメント STACKS=9,128 を認識しません。*

ConfigGetInt を呼び出す前に、まず ConfigFileLoad を呼び出してシステム構成ファイルをメモリへロードしなくてはなりません。ファイルを編集した後、ConfigFileSave を呼び出してファイルを保存します。

Ez 構成ファイル関数を拡張構成ファイル関数と同時に使用しないでください。ConfigFileLoad 関数を呼び出した後、ConfigFileSave を使って変更を保存するまで、簡易構成ファイル関数を利用することはできません。

構文

ConfigGetInt (szKey, nvValue);

パラメーター

テーブル 58 - ConfigGetInt のパラメーター

パラメーター	説明
szKey	整数値を読み出すステートメントの参照キーを指定します。
nvValue	参照キーの整数値を戻します。

戻り値

テーブル 59 - ConfigGetInt の戻り値

戻り値	説明
0	ConfigGetInt は整数値を読み出しました。
< 0	ConfigGetInt は整数値を読み出せませんでした。

ConfigGetInt の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ConfigGetInt 関数と ConfigSetInt 関数のデモンストレーションを行います。
*
* このスクリプト例では、構成ファイルから FILES
* 取得します。FILES コマンドが検出されなかった場合、
* コマンド FILES=40 がファイルに追加されます。FILES コマンドが
* 検出された場合、その値がテストされます。値が 40 以下の場合、
* コマンドが FILES=40 と置換されます。
*
* メモ : このスクリプトを実行する前に、C ドライブのルートへ
*      ISExampl.sys と呼ばれる構成ファイルを作成します。
*      ファイルは次の行を含みます :
*
*      FILES=20;
*
¥*---------------------------------------------------------*/
```

```
#define EXAMPLE_SYS "C:¥¥ISExampl.sys"
#define EXAMPLE_BAK "ISExampl.bak"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ConfigGetInt(HWND);

function ExFn_ConfigGetInt(hMSI)
  NUMBER nvValue;
  BOOL   bFileChanged;
begin

  // 構成ファイルをロードします。
  if (ConfigFileLoad (EXAMPLE_SYS) < 0) then
    MessageBox (EXAMPLE_SYS + " をロードできませんでした。", SEVERE);
    abort;
  endif;

  // インジケーターを初期化し、ファイルが更新されたかどうかを表示します。
  bFileChanged = FALSE;

  // 構成ファイル内でコマンド "FILES" を検出します。
  if (ConfigGetInt ("FILES", nvValue) < 0) then

    // FILES コマンドは検出されませんでした。FILES コマンドを追加します。
    if ConfigAdd ("FILES", "40", "", AFTER) = 0 then
      MessageBox ("FILES=40 が " + EXAMPLE_SYS + " へ追加されました。", INFORMATION);
      bFileChanged = TRUE;
    else
      MessageBox ("FILES コマンドが見つかりません。" +
            EXAMPLE_SYS + " を更新できません。", SEVERE);
    endif;

  else

    // FILES コマンドが検出されました。
    if (nvValue >= 40) then
      // FILES コマンドの設定は OK です。
      SprintfBox (INFORMATION, "ConfigGetInt の例 ",
            "FILES=%d、変更は必要ありません。.", nvValue);
    else

      // FILES コマンドを変更する必要があります。
      if (ConfigSetInt ("FILES", 40) < 0) then
        MessageBox (EXAMPLE_SYS + " を更新できませんでした。", SEVERE);
      else
        MessageBox ("FILES 設定は 40 に変更されました。", INFORMATION);
        bFileChanged = TRUE;
      endif;

    endif;

  endif;

  // ファイルが編集された場合はそれを保存します。
  if bFileChanged then
```

```
        // 拡張子 'bak' を持つ元のファイルのバックアップを行い、
        // 編集されたファイルを元の名前の下に保存します。ISExamp1.bak が既に存在する場合、
        // ConfigFileSave は数字の付いた拡張子を生成します。
        if (ConfigFileSave (EXAMPLE_BAK) < 0) then
            MessageBox (EXAMPLE_SYS +" を保存できませんでした。", SEVERE);
        else
            MessageBox (EXAMPLE_SYS + " が保存されました。",INFORMATION);
        endif;

    endif;

end;
```

ConfigMove

ConfigMove は ConfigFileLoad 関数を使ってメモリにロードしたシステム構成ファイル内の行を移動します。行はファイルの最初または最後の位置、あるいはファイル内の特定行の前または後ろに移動させることができます。

メモ・ConfigMove 関数を呼び出す前に、まず ConfigFileLoad を呼び出して Config.sys ファイルをメモリへロードしなくてはなりません。ファイルを編集した後、ConfigFileSave を呼び出してファイルを保存します。

Ez 構成ファイル関数を拡張構成ファイル関数と同時に使用しないでください。ConfigFileLoad 関数を呼び出した後、ConfigFileSave を使って変更を保存するまで、簡易構成ファイル関数を利用することはできません。

構文

```
ConfigMove ( szMove, szRefKey, nOptions );
```

パラメーター

テーブル 60 ・ ConfigMove のパラメーター

パラメーター	説明
szMove	行の移動先を指定します。
szRefKey	移動させる行を配置するのに使う参照行を識別するキーを指定します。移動する行の位置は nOption の値によって決定されます。
nOptions	szMove の行を szRefKey の参照キーを含む行の前後いづれに移動するかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • **BEFORE** ― szMove が指定する行を szRefKey を含む行の前に配置します。szMove がヌル文字列 (“”) の場合、その行はシステム構成ファイルの最初の行の前に配置されます。 • **AFTER** ― szMove が指定する行を szRefKey を含む行の後に配置します。szMove がヌル文字列 (“”) の場合、その行はシステム構成ファイルの最後の行の後に配置されます。

戻り値

テーブル 61 ・ ConfigMove の戻り値

戻り値	説明
0	ConfigMove がシステム構成ファイル内で指定された行の移動を完了しました。
< 0	ConfigMove は行を移動できませんでした。

ConfigMove の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
```

```
＊ConfigMove 関数のデモンストレーションを行います。
＊
＊このスクリプト例では、構成ファイル内で行を移動させます。
＊まず、ファイルをロードする ConfigFileLoad を呼び出します。次に、
＊FILES ステートメントをファイルの最後に移動させます。そして、
＊BUFFERS ステートメントをファイルの最後に移動させます。
＊次に、DOS ステートメントの前に Himem.sys を参照するステートメントを
＊移動します。最後に、元のファイルを
＊バックアップして編集したファイルを保存します。
＊
＊メモ : このスクリプトを実行する前に、C ドライブのルートへ
＊      ISExampl.sys と呼ばれる構成ファイルを作成します。
＊      ファイルは次の行を含みます :
＊
＊      FILES=50
＊      DOS=HIGH,UMB
＊      DEVICE=C:¥WINDOWS¥SETVER.EXE
＊      BUFFERS=50
＊      Device=C:¥Windows¥Himem.sys
＊
¥*---------------------------------------------------------------*/

#define TARGET_CONFIG  "C:¥¥ISExampl.sys"
#define BACKUP_CONFIG  "ISExampl.bak"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ConfigMove(HWND);

function ExFn_ConfigMove(hMSI)
begin

   // 編集する構成ファイルをロードします。
   if ConfigFileLoad (TARGET_CONFIG) < 0 then
      MessageBox (TARGET_CONFIG + " をロードできませんでした。", SEVERE);
      abort;
   endif;

   // FILES ステートメントをファイルの最後に移動します。
   if (ConfigMove ("FILES", "", AFTER) < 0) then
      MessageBox ("FILES ステートメントを移動できませんでした。", SEVERE);
   else
      MessageBox ("FILES ステートメントをファイルの終わりに移動します。",
            INFORMATION);
   endif;

   // BUFFERS ステートメントをファイルの最後に移動します。
   if (ConfigMove ("BUFFERS", "", AFTER) < 0) then
      MessageBox ("BUFFERS ステートメントを移動できませんでした。", SEVERE);
   else
      MessageBox ("BUFFERS ステートメントをファイルの終わりに移動します。",
            INFORMATION);
   endif;

   // Himem.sys ステートメントを DOS ステートメントの前に移動します。
   if (ConfigMove ("Himem.sys", "DOS", BEFORE) < 0) then
      MessageBox ("Himem.sys ステートメントを移動できませんでした。", SEVERE);
   else
```

```
    MessageBox ("Himem.sys ステートメントが DOS ステートメントの前に移動しました。",
        INFORMATION);
endif;

// 更新されたファイルを保存し、元のファイルをバックアップします。
if ConfigFileSave (BACKUP_CONFIG) < 0 then
    MessageBox (BACKUP_CONFIG + " を保存することができませんでした。", SEVERE);
else
    MessageBox (" 構成ファイルが保存されました。バックアップが作成されました。",INFORMATION);
endif;

end;
```

ConfigSetFileName

ConfigSetFileName 関数はデフォルトシステム構成ファイルとして利用するファイルの完全修飾名を指定します。インストール初期化中に、インストールはターゲットシステムが開始されたときに実行された Config.sys ファイルを認識し、それをデフォルトのシステム構成ファイルとします。これが、インストールが編集する唯一のシステム構成ファイルの場合、この関数を必ず呼び出さなくてはなりません。簡易構成ファイルはそのファイルを利用し、パラメーターがヌル文字列が ("") のとき拡張構成関数 ConfigFileLoad がそのファイルを開きます。

しかし、簡易構成ファイル関数を使って起動 Config.sys ファイル以外の構成ファイルを変更するとき、デフォルトシステム構成ファイルを変更するために ConfigSetFileName を呼び出さなくてはなりません。たとえば、起動時には利用されないターゲットシステム上の Config.sys ファイルを作成すると仮定します。アプリケーションディレクトリでファイル名を設定することができます。簡易構成ファイル関数はそのファイル上で動作します。ヌルパラメーターと共に ConfigFileLoad を呼び出した場合、そのファイルはメモリにロードされ、そこで拡張ファイル関数を使って編集することができます。

***注意・** ConfigSetFileName 関数はメモリへシステム構成ファイルをロードしません。ConfigFileLoad を使ってファイルをメモリへロードしなくてはなりません。*

ConfigSetFileName は指定したファイル名を有効にはしません。無効なファイル名を指定したとき、この後すべての構成ファイル関数が失敗します。

***メモ・** Ez 構成ファイル関数を拡張構成ファイル関数と同時に使用しないでください。ConfigFileLoad 関数を呼び出した後、ConfigFileSave を使って変更を保存するまで、簡易構成ファイル関数を利用することはできません。*

構文

```
ConfigSetFileName ( szConfigFile );
```

パラメーター

テーブル 62 - ConfigSetFileName のパラメーター

パラメーター	説明
szConfigFile	デフォルトのシステム構成ファイルとして設定するファイルの完全修飾名を指定します。

戻り値

テーブル 63 - ConfigSetFileName の戻り値

戻り値	説明
0	ConfigSetFileName が指定したシステム構成ファイルを読み出しました。
< 0	ConfigSetFileName は指定されたファイルを読み出せませんでした。

ConfigSetFileName の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ConfigGetFileName 関数 と ConfigSetFileName 関数の
* デモンストレーションを行います。
*
* このスクリプト例では、デフォルト構成ファイルの完全修飾名を読み出します。
* デフォルト構成ファイルは最初、ブート ドライブ上の
* Config.sys ファイルです。次いで、C:¥ISExampl.sys を
* デフォルトの構成ファイルにします。最後に、もう一度
* デフォルトの構成ファイルの名前を読み出して変更されたことを
* 表示します。
*
¥*---------------------------------------------------------*/

#define DEFAULT_CONFIG_FILE "C:¥¥ISExampl.sys"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ConfigSetFileName(HWND);

function ExFn_ConfigSetFileName(hMSI)
  STRING svFilename;
```

```
begin

    // デフォルトの構成ファイルの名前を取得して表示します。
    if (ConfigGetFileName (svFilename) < 0) then
        // エラーを報告し、中止します。
        MessageBox ("ConfigGetFileName の最初の呼び出しに失敗しました。", SEVERE);
        abort;
    else
        // デフォルトの構成ファイルの名前を表示します。
        MessageBox (" デフォルトの構成ファイルは " + svFilename + " です。",
                INFORMATION);
    endif;

    // C:¥ISExampl.sys をデフォルトの構成ファイルとします。
    if (ConfigSetFileName (DEFAULT_CONFIG_FILE) < 0) then
        // エラーを報告します。
        MessageBox (" 新しいデフォルトの構成ファイルを設定できませんでした。", SEVERE);
    else

        // デフォルトの構成ファイルが変更されたことを確認します。
        if (ConfigGetFileName (svFilename) = 0) then
            // デフォルトの構成ファイルの名前を表示します。
            MessageBox (" 現在のデフォルトの構成ファイルは " + svFilename +
                    " です。", INFORMATION);
        else
            // エラーを報告します。
            MessageBox ("ConfigGetFileName の 2 回目の呼び出しに失敗しました。", SEVERE);
        endif;

    endif;

end;
```

ConfigSetInt

ConfigSetInt は ConfigFileLoad 関数を使ってメモリにロードしたシステム構成ファイル内の指定された整数値を変更します。ConfigSetInt は、等号 (=) の右に一つだけ値を持つコマンドの値を設定します。

メモ・関数は複数の値を持つコマンドには動作しません。たとえば、ConfigSetInt はステートメント FILES=20 を認識し、20 を別の値に変更することができますが、ステートメント STACKS=9,128 を認識しません。

ConfigSetInt を呼び出す前に、まず ConfigFileLoad を呼び出してシステム構成ファイルをメモリへロードしなくてはなりません。ファイルを編集した後、ConfigFileSave を呼び出してファイルを保存します。

Ez 構成ファイル関数を拡張構成ファイル関数と同時に使用しないでください。ConfigFileLoad 関数を呼び出した後、ConfigFileSave を使って変更を保存するまで、簡易構成ファイル関数を利用することはできません。

構文

```
ConfigSetInt ( szKey, nValue );
```

パラメーター

テーブル 64 - ConfigSetInt のパラメーター

パラメーター	説明
szKey	整数値を設定するコマンドの参照キーワードを指定します。
nValue	szKey のコマンドに設定する整数値を指定します。

戻り値

テーブル 65 - ConfigSetInt の戻り値

戻り値	説明
0	ConfigSetInt がシステム構成ファイル内で指定された整数の設定を完了しました。
< 0	ConfigSetInt は指定された整数を設定できませんでした。

ConfigSetInt の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ConfigGetInt 関数と ConfigSetInt 関数のデモンストレーションを行います。
*
* このスクリプト例では、構成ファイルから FILES
* 取得します。FILES コマンドが検出されなかった場合、
* コマンド FILES=40 がファイルに追加されます。FILES コマンドが
* 検出された場合、その値がテストされます。値が 40 以下の場合、
* コマンドが FILES=40 と置換されます。
*
* メモ : このスクリプトを実行する前に、C ドライブのルートへ
*      ISExampl.sys と呼ばれる構成ファイルを作成します。
*      ファイルは次の行を含みます :
*
*      FILES=20;
*
¥*---------------------------------------------------------------------*/
```

```
#define EXAMPLE_SYS "C:¥¥ISExampl.sys"
#define EXAMPLE_BAK "ISExampl.bak"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ConfigSetInt(HWND);

function ExFn_ConfigSetInt(hMSI)
  NUMBER nvValue;
  BOOL   bFileChanged;
begin

  // 構成ファイルをロードします。
  if (ConfigFileLoad (EXAMPLE_SYS) < 0) then
    MessageBox (EXAMPLE_SYS + " をロードできませんでした。", SEVERE);
    abort;
  endif;

  // インジケーターを初期化し、ファイルが更新されたかどうかを表示します。
  bFileChanged = FALSE;

  // 構成ファイル内でコマンド "FILES" を検出します。
  if (ConfigGetInt ("FILES", nvValue) < 0) then

    // FILES コマンドは検出されませんでした。FILES コマンドを追加します。
    if ConfigAdd ("FILES", "40", "", AFTER) = 0 then
      MessageBox ("FILES=40 が " + EXAMPLE_SYS + " へ追加されました。", INFORMATION);
      bFileChanged = TRUE;
    else
      MessageBox ("FILES コマンドが見つかりません。" +
                  EXAMPLE_SYS + " を更新できません。", SEVERE);
    endif;

  else

    // FILES コマンドが検出されました。
    if (nvValue >= 40) then
      // FILES コマンドの設定は OK です。
      SprintfBox (INFORMATION, "ConfigGetInt の例 ",
                  "FILES=%d、変更は必要ありません。.", nvValue);
    else

      // FILES コマンドを変更する必要があります。
      if (ConfigSetInt ("FILES", 40) < 0) then
        MessageBox (EXAMPLE_SYS + " を更新できませんでした。", SEVERE);
      else
        MessageBox ("FILES 設定は 40 に変更されました。", INFORMATION);
        bFileChanged = TRUE;
      endif;

    endif;

  endif;

  // ファイルが編集された場合はそれを保存します。
  if bFileChanged then

    // 拡張子 'bak' を持つ元のファイルのバックアップを行い、
```

```
        // 編集されたファイルを元の名前の下に保存します。ISExamp1.bak が既に存在する場合、
        // ConfigFileSave は数字の付いた拡張子を生成します。
        if (ConfigFileSave (EXAMPLE_BAK) < 0) then
            MessageBox (EXAMPLE_SYS +" を保存できませんでした。", SEVERE);
        else
            MessageBox (EXAMPLE_SYS + " が保存されました。",INFORMATION);
        endif;

    endif;

end;
```

ConvertSizeToUnits

ConvertSizeToUnits は、nUnitsSrc の nSizeSrcHigh および nSizeSrcLow で指定されたサイズを nUnitsTarget の nvSizeTargetHigh および nvSizeTargetLow に変換します。この関数を使って、セットアップが実行されるまで不明な特定のサイズ値に使用する最適な単位を判断することもできます。

構文

```
ConvertSizeToUnits ( nSizeSrcHigh, nSizeSrcLow, nUnitsSrc, nvSizeTargetHigh, nvSizeTargetLow, nvUnitsTarget );
```

パラメーター

テーブル 66 · CovertSizeToUnits のパラメーター

パラメーター	説明
nSizeSrcHigh	変換されるサイズの上位 31 ビットを指定します。
nSizeSrcLow	変換されるサイズの下位 31 ビットを指定します。
nUnitsSrc	nSizeSrcHigh および nSizeSrcLow の単位。次の値のうち 1 つを指定します。 • BYTES – 値はバイトです。 • KBYTES – 値はキロバイトです。 • MBYTES – 値はメガバイトです。 • GBYTES – 値はギガバイトです。 • TBYTES – 値はテラバイトです。
nvSizeTargetHigh	変換されたサイズの上位 31 ビットを返します。
nvSizeTargetLow	変換されたサイズの下位 31 ビットを戻します。
nUnitsTarget	nvSizeTargetHigh および nvSizeTargetLow の単位を指定します (-1 が指定された場合、戻します)。次の値のうち 1 つを指定します。(-1 または負の値を指定することもできます。この場合、関数は nUnitsTarget を nvSizeTargethigh が確実に 0 になるように最小の単位に設定します。): • BYTES – 値はバイトです。 • KBYTES – 値はキロバイトです。 • MBYTES – 値はメガバイトです。 • GBYTES – 値はギガバイトです。 • TBYTES – 値はテラバイトです。

戻り値

テーブル 67 · ConvertSizeToUnits の戻り値

戻り値	説明
ISERR_SUCCESS	関数が成功したことを示します。

テーブル 67 • ConvertSizeToUnits の戻り値 (続き)

戻り値	説明
< ISERR_SUCCESS	関数の実行に失敗したことを示します。
ISERR_INVALID_ARG	nSizeSrcHigh または nSizeSrcLow が負であることを示します。

ConvertWinHighLowSizeToISHighLowSize

The **ConvertWinHighLowSizeToISHighLowSize** 関数は、nSizeWinHigh および nSizeWinLow で指定された署名なしの 64 ビット Windows サイズを 対応する 62 ビット InstallShield 高および低サイズ値に変換します。

構文

ConvertWinHighLowSizeToISHighLowSize (nSizeWinHigh, nSizeWinLow, nvSizeISHigh, nvSizeISLow);

パラメーター

テーブル 68 - ConvertWinHighLowSizeToISHighLowSize のパラメーター

パラメーター	説明
nSizeWinHigh	Windows 64 ビットサイズ値の上位 31 ビットを指定します。
nSizeWinLow	Windows 64 ビットサイズ値の下位 31 ビットを指定します。
nSizeISHigh	InstallShield 62 ビットサイズ値の上位 31 ビットを返します。
nSizeISLow	InstallShield 62 ビット サイズ値の下位 31 ビットを戻します。

戻り値

テーブル 69 - ConvertWinHighLowSizeToISHighLowSize の戻り値

戻り値	説明
ISERR_SUCCESS	関数が成功したことを示します。
< ISERR_SUCCESS	関数の実行に失敗したことを示します。

CopyBytes

CopyBytes 関数は、ある文字列から別の文字列へ指定したバイト数をコピーします。元の文字列と目的の文字列へのオフセットインデックスを指定することができます。CopyBytes は、バイナリファイルを扱う際に便利です。

構文

```
CopyBytes ( svDest, nIndexDest, svSrc, nIndexSrc, nCount );
```

パラメーター

テーブル 70 ・ CopyBytes のパラメーター

パラメーター	説明
svDest	目的の文字列を指定します。
nIndexDest	バイトを挿入する目的の文字列位置のオフセットインデックス（始まり点）を指定します。文字列の初めのバイト位置は 0。
svSrc	元の文字列を指定します。サイズが 256 文字を超える自動調整された文字列を渡さないで下さい。その代わり、明確なサイズと共に文字列を宣言してください。文字列サイズについては、「文字列サイズと Autosize」を参照してください。
nIndexSrc	バイトをコピーし始める元の文字列位置のオフセット インデックス（始まり点）を指定します。文字列の初めのバイト位置は 0。
nCount	svSrc から svDest へコピーするバイトの合計数を指定します。この値は svSrc – 1 サイズを超えてはいけません。例えば、svSrc をサイズ 512 で宣言した場合（最大文字列長さの 511 を使う）、nCount で渡される値は 511 またはそれ以下でなくてはなりません。

戻り値

テーブル 71 ・ CopyBytes の戻り値

戻り値	説明
0	CopyBytes がある文字列から別の文字列へ、指定したバイト数をコピーしました。
< 0	CopyBytes はバイトをコピーできませんでした。

CopyBytes の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * CopyBytes 関数のデモンストレーションを行います。
 *
 * このスクリプト例では、ターゲットシステムから現在の日付を
 * 設定します。そして、日付から年をコピーしてエンドユーザーへ
 * 年を表示します。
 *
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CopyBytes(HWND);

function ExFn_CopyBytes(hMSI)
    STRING svDate, svYear;
    NUMBER nvResult, nIndexDate, nIndexYear, nCount;
begin

    // ターゲットシステムから日付を取得します。
    GetSystemInfo (DATE, nvResult, svDate);

    // CopyBytes へ渡すパラメーターをセットアップします。年は
    // svDate の最後の 4 バイトにあります。
    nIndexYear  = 0;
    nCount      = 4;
    nIndexDate  = StrLength(svDate) - nCount;

    // 年を表す 4 バイトを svYear へコピーします。
    if (CopyBytes (svYear, nIndexYear, svDate, nIndexDate, nCount) < 0) then
        // エラーを報告します。
        MessageBox ("CopyBytes が失敗しました .", SEVERE);
    else
        // 年を表示します。
        MessageBox (" 年は " + svYear, INFORMATION);
    endif;

end;
```

CopyCHARArrayToISStringArray

説明

CopyCHARArrayToISStringArray 関数は ANSI 文字列の既存配列 (pCHARArray がポイントする) から vArray が指定する既存文字列配列へ文字列をコピーします。

構文

```
CopyCHARArrayToISStringArray ( vArray, pCHARArray );
```

パラメーター

テーブル 72 ・ CopyCHARArrayToISStringArray のパラメーター

パラメーター	説明
vArray	文字列をコピーする文字列配列を指定します。
pCHARArray	ANSI 文字列へのポインターの配列へのポインターを指定します。一般的に、このポインターは以前の GetCHARArrayFromISStringArray への呼び出しで戻されます。

戻り値

テーブル 73 ・ CopyCHARArrayToISStringArray の戻り値

戻り値	説明
ISERR_SUCCESS	この関数は常に ISERR_SUCCESS を戻します。

追加情報

GetCHARArrayFromISStringArray への以前の呼び出しで pCHARArray が戻された場合、配列に含まれる文字列の変更には注意が必要です。文字列配列に含まれる文字列の長さはセットアップが内部的に管理するため、文字列の長さを変更すると CopyCHARArrayToISStringArray を呼び出した際に、文字列全体がオリジナル配列へコピーされません。

CopyFile

CopyFile 関数は szSrcFile によって指定されたファイルのコピーを作成します。新規ファイルは szTargetFil で指定した名前がつけられます。関数を動作させるためには、ターゲット場所へコピーするファイル名をこの関数の szTargetFile パラメーターで指定しなくてはなりません。

この関数を使って、ファイルを WINSYSDIR64 に転送する場合、まず WOW64FSREDIRECTION を使用してファイルシステムのリダイレクトを無効にする必要があります。無効化をしない場合、WINSYSDIR64 に転送されるファイルは不適切に 32 ビット SysWOW64 システムフォルダーにリダイレクトされます。インストールが利用する可能性のある Windows 機能にはファイル システム リダイレクトを有効にしておく必要があるため、Windows ドキュメンテーションではリダイレクトを無効にするのはそれが必要な場合のみにとどめることが推奨されています。必要なファイルを WINSYSDIR64 へ転送し終わったら、直ちにシステム ファイルのリダイレクトを有効にすることをお勧めします。詳しくは、「64 ビット オペレーティング システムを InstallScript インストールでターゲットにする」を参照してください。

***ヒント・**コピーの最中にステータス ダイアログが表示される場合は、**CopyFile** 関数を呼び出す前に **Disable** 関数を利用して [キャンセル] ボタンを無効にすることが、強く推奨されます。[キャンセル] ボタンを無効にしなかった場合で、エンド ユーザーがファイルのコピー操作を途中でキャンセルした場合、OnCancelling イベントハンドラーは呼び出されません。その代わり、ファイルのコピー操作はエラー コードを戻します。この場合、スクリプトによって適切なイベントを呼び出してからファイルのコピー操作を再開する必要があります。Enable と Disable 関数を利用して [キャンセル] ボタンを有効および無効にすることができます。*

*メモ・ファイル転送の際、XCopyFile は **CopyFile** の代替となります。**XCopyFile** はバージョンチェックを行い、システム再開始の後にアップデートできるようにロックされた .dll や .exe ファイルをマークし、共有 .dll と .exe ファイルのレジストリ参照カウンターを増分します。サブディレクトリを含むために **XCopyFile** を利用することもできます。*

*完全修飾でないファイル名を使用して、XcopyFile を使用中に SRCDIR と TARGETDIR の値を設定する場合、**CopyFile** を呼び出す前に VarSave を使って現在の値を保存し、それから VarRestore を使って復元してください。*

*ターゲット ディレクトリが存在しない場合は、**CopyFile** が作成します。*

***CopyFile** と共にワイルドカードを使って一連のファイルの名前を変更することはできません。ただし、**CopyFile** を使って単一のファイルの名前を変更することは可能です。*

構文

CopyFile (szSrcFile, szTargetFile);

パラメーター

テーブル 74 • CopyFile のパラメーター

パラメーター	説明
szSrcFile	コピーするファイルの名前を指定します。指定のファイルが完全修飾名、つまりパスが含まれる場合、**CopyFile** はファイルを指定の場所からコピーします。szSrcFile に完全修飾名ではないファイル名が含まれ、パス情報がない場合、**CopyFile** は SRCDIR システム変数によって識別されたディレクトリからコピーします。グループ化されたファイルをコピーするには、このパラメーターでワイルドカード文字を使ってください。 このパラメーターでは、有効な URL を指定できます。CGI または ASP 要求 (たとえば、"http://www.mydomain.net/login.asp?name=Me&pwd=wow") を渡す場合、その応答は szTargetFile パラメーターで指定されたファイルへ送られます。URL を渡すとき、ワイルドカード文字を含まないで下さい。URL が有効かどうかを確認するには、次を呼び出します : Is (VALID_PATH, szURL);
szTargetFile	szSrcFile が識別したファイルのコピーへ渡す名前を指定します。ファイル名が完全修飾名で、パスが含まれる場合、**CopyFile** はファイルをパスが指定する場所にコピーします。szSrcFile のファイル名が修飾されていない、つまりパス情報が含まれていない場合、システム変数 TARGETDIR (InstallScript インストールの場合) または INSTALLDIR (基本の MSI および InstallScript MSI インストールの場合) が指定するディレクトリにコピーが作成されます。ターゲットディレクトリが存在しない場合は、それが作成されます。 このパラメーターで URL は指定できません。ここで指定しても、関数が失敗して ISERR_INVALID_ARG を戻します。 ***メモ •*** *szSrcFile が指定するファイル名にワイルドカード文字が含まれている場合、szTargetFile のファイル名部分が無視されることはなく、szTargetFile 値は各ソース ファイルがコピーされるターゲット パスとして、その既存名を使って処理されます。たとえば、次のコードは D ドライブの* **File.txt** *と名づけられたフォルダーにファイルをコピーします :* CopyFile ("C:¥¥*.*", "D:¥¥File.txt"); *szTargetFile が修飾されないファイル名を指定する場合、ファイルは TARGETDIR (InstallScript インストールの場合) または* ***INSTALLDIR*** *(基本の MSI および InstallScript MSI インストールの場合) が指定するディレクトリにコピーされます。このため、****CopyFile*** *をグループ化されたファイルのコピーや名前の変更に利用することができません。szSrcFile が単数または複数のワイルドカード文字を含む場合、元のディレクトリとターゲット ディレクトリは別々でなくてはなりません。*

戻り値

テーブル 75 - CopyFile の戻り値

戻り値	説明
0	関数がファイルを元のディレクトリからターゲットディレクトリへコピーしたことを示します。
ISERR_INVALID_ARG	無効な引数が関数へ渡されたことを示します。
その他すべての負の値	関数が要求されたファイルをコピーできなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

WriteProfString または WriteProfInt を使って .ini ファイルを変更後、**CopyFile** を利用する前にキャッシュのバッファーをフラッシュしなくてはなりません。すべての .ini ファイルがキャッシュされます。この動作によって、特定ファイルへの変更の書き込みが遅くなります。この遅延は、後に続くファイル処理の妨げとなる場合があります。この問題を回避するには、ヌル パラメーターで **WriteProfString** を呼び出して、以下の通り Windows が即座に .ini ファイルへデータの書き込みを行う様に強制します。

```
WriteProfString ("C:¥¥Test.ini", "Windows",
"KeyboardDelay", "100");

// 全 4 つのパラメーター用のヌル文字列 ("")
WriteProfString ("", "", "", "");

// CopyFile はこれで更新されたファイルにアクセスできます。
CopyFile ("C:¥¥Test.ini", "C:¥¥Temp¥¥Test.ini");
```

CopyFile の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CopyFile 関数のデモンストレーションを行います。
*
* このスクリプトは SOURCE_DIR で指定されたディレクトリのファイルを
* TARGET_DIR で指定されたディレクトリへコピーします。
*
* メモ : このスクリプトを実行する前に、プロセッサ定数を
*      ターゲットシステム上の既存パスに
*      設定する必要があります。
```

```
 *
¥*---------------------------------------------------------------*/

#define  SOURCE_DIR "C:¥¥Source"
#define  TARGET_DIR "C:¥¥Target"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CopyFile(HWND);

function ExFn_CopyFile(hMSI)
   NUMBER nResult;
begin

   // サブディレクトリのファイルを含んで、ソース ディレクトリにある
   // すべてのファイルをターゲット ディレクトリへコピーします。
   nResult = CopyFile(SOURCE_DIR ^ "*.*", TARGET_DIR ^ "*.*");

   // コピー処理の結果を報告します。
   switch (nResult)
      case 0:
         MessageBox ("ファイルが無事にコピーされました。", INFORMATION);
      case COPY_ERR_CREATEDIR:
         MessageBox ("ターゲットディレクトリを作成することができませんでした。", SEVERE);
      case COPY_ERR_MEMORY:
         MessageBox ("メモリ容量が足りません。", SEVERE);
      case COPY_ERR_NODISKSPACE:
         MessageBox ("ディスク容量が足りません。", SEVERE);
      case COPY_ERR_OPENINPUT:
         MessageBox (SOURCE_DIR + "の入力ファイルを開くことができませんでした。",
                SEVERE);
      case COPY_ERR_OPENOUTPUT:
         MessageBox ("ソースファイルをコピーすることができませんでした。", SEVERE);
      case COPY_ERR_TARGETREADONLY:
         MessageBox ("ターゲットファイルが既に存在し、また上書することもできません。",
                SEVERE);
      デフォルト :
         MessageBox ("原因不明のエラーが発生しました。", SEVERE);
   endswitch;

end;
```

CreateDir

CreateDir 関数を呼び出して、ターゲットドライブにサブディレクトリを作成します。例えば **C:¥Programs¥Winapps¥Myapp** の様に、複数レベルにわたるサブディレクトリを含むパスを利用することもできます。パスにサブディレクトリがない場合は、CreateDir が作成します。例えば、**C:¥Programs¥Winapps** と **C:¥Programs¥Winapps¥Myapp** のどちらも存在しない場合、CreateDir は両方のサブディレクトリを作成します。

注意・*CreateDir は次の条件によって失敗しました。*

- *不法なパスです。*
- *ドライブまたはパスにあるサブディレクトリは書き込み禁止となっています。*

- *ドライブ名が無効です。*
- *サブディレクトリを作成するためのネットワーク権限がありません。*

構文

CreateDir (szDirPath);

パラメーター

テーブル 76 ・ CreateDir のパラメーター

パラメーター	説明
szDirPath	作成するサブディレクトリの完全修飾パスを指定します。パス内の各レベルを円記号エスケープ文字 (¥¥) を使って分割します。

戻り値

テーブル 77 ・ CreateDir の戻り値

戻り値	説明
0	関数がターゲットドライブに指定したディレクトリを作成した、あるいは既存ディレクトリを指定したことを示します。
< 0	ディレクトリが存在しないこと、そして関数がそれを作成することができなかったことを示します。大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

CreateDir の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CreateDir 関数のデモンストレーションを行います。
*
* ユーザーは有効なディレクトリの入力を求められます。インストールが
* 指定されたディレクトリが存在しない場合、それを作成するため
* CreateDir が呼び出されます。次いで、CreateDir が再び呼び出され、
* 指定したディレクトリの下に多階層ディレクトリ構造を
```

```
＊作成します。
＊
¥*---------------------------------------------------------*/

#define DEFAULT_DIR "C:¥¥ISExampl"
#define SUBDIRS     "N_Dir¥¥A_Dir"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CreateDir(HWND);

function ExFn_CreateDir(hMSI)
  STRING svPath;
begin

  // セットアップダイアログで [ 戻る ] ボタンを無効にします。
  Disable(BACKBUTTON);

  // ユーザーへディレクトリの作成を要求します。
  AskPath ("有効なパスを入力してください。", DEFAULT_DIR, svPath);

  // そのディレクトリが既に存在するかどうかを確認します。
  if (ExistsDir (svPath) != EXISTS) then

    // ディレクトリが存在しません。作成します。
    if (CreateDir (svPath) < 0) then
      // エラーをレポートし、中止します。
      MessageBox ("ディレクトリを作成できませんでした", SEVERE);
      abort;
    else
      // 成功を報告します
      SprintfBox (INFORMATION, "CreateDir", "%s が作成されました。", , svPath);
    endif;

  endif;

  // 指定したディレクトリの下に
  // 多階層ディレクトリ構造すべてを作成します。
  if (CreateDir (svPath ^ SUBDIRS) < 0) then
    MessageBox (svPath + "の下のサブディレクトリの作成に失敗しました。",
          WARNING);
  else
    SprintfBox (INFORMATION, "CreateDir", "%s が %s の下に作成されました。", SUBDIRS,
          svPath);
  endif;

end;
```

CreateFile

CreateFile 関数は新しいファイルを作成します。同じ名前のファイルが存在すると、CreateFile はこれを上書きします。CreateFile を使ってファイルを作成する前に、OpenFileMode でファイルモードを設定する必要があります。

CreateFile は新しく作成されたファイルを (バイナリファイルでは) 読み取り / 書き込みモード、または (テキストファイルでは) 追加 モードで開いたままにするので、GetLine、ReadBytes、WriteLine、または WriteBytes のようなその他の関数を使ってファイルから読み取ったり、ファイルへ書き込んだりすることができます。既存のファイルに書き込むには、最初に OpenFileMode 関数や OpenFile 関数を使って FILE_MODE_APPEND モードでファイルを開く必要があります。

メモ・読み取り / 書き込み　または追加モードに加え、新しく作成されたすべてのファイルは自動的に OF_SHARE_DENY_NONE モードで開きます。これは、ファイルに対する他のプログラムの読み取りまたは書き込みアクセスを拒否せずにファイルが開かれるということです。このモードは Windows API OpenFile にルートがあります。

ファイルからの読み取りまたはファイルへの書き込みが終わったら、CloseFile 関数でファイルを閉じる必要があります。

構文

```
CreateFile (nvFileHandle, szPath, szFileName);
```

パラメーター

テーブル 78 • CreateFile のパラメーター

パラメーター	説明
nvFileHandle	新しいファイルのハンドルを戻します。
szPath	新しいファイルを作成するサブディレクトリへの完全修飾パスを指定します。 ***メモ •*** *CreateFile では、szPath で指定されたフォルダーが存在しない場合、エラーが発生します。フォルダーが存在するかどうか、ExistsDir を呼び出してテストし、CreateDir を呼び出してフォルダーを作成することができます。*
szFileName	作成するファイルの名前を指定します。

戻り値

テーブル 79 • 戻り値

戻り値	説明
0	関数が新しいファイルを正常に作成したことを示します。
< 0	関数が指定されたファイルを作成できなかったことを示します。

追加情報

ログ記録が有効となっている場合、CreateFile 関数のアクションはアンインストールで記録されません。CreateFile が作成したファイルをアンインストールに記録する場合は、ログ記録が有効となっている間に、XCopyFile を使って、希望のファイル名を持つ開始ファイルをターゲットシステムに転送します。ログ記録は、Enable または Disable 関数を使って有効または無効にします。ログ記録が有効となっている場合、XCopyFile アクションがログされるので、開始ファイルがアンインストールに記録されます。ログされた開始ファイルを転送した後で、CreateFile およびその他のファイル関連関数を使って、そのファイルに書き込んだり上書きすることができます。ファイル名を変更することはできません。変更すると、アンインストール時に検出されなくなります。

CreateFile の例

メモ • *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * CreateFile 関数と WriteLine 関数のデモンストレーションを行います。
 *
 * 文字列を保存するファイルを作成するため CreateFile が呼び出されます。その後
 * 文字列は WriteLine 関数によってファイルに書き込まれます。
 *
 * メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
 *      EXAMPLE_DIR を設定して、ターゲット システム上の既存の
 *      ディレクトリを参照するようにします。EXAMPLE_FILE で、
 *      指定したファイルが既に存在する場合
 *      それが上書きされます。
 *
¥*---------------------------------------------------------------------*/

#define EXAMPLE_DIR  "C:¥¥"
#define EXAMPLE_FILE "ISExampl.txt"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CreateFile(HWND);

function ExFn_CreateFile(hMSI)
   STRING  szTitle, szMsg;
   NUMBER nvFileHandle;
begin

   // ファイル モードを追加に設定します。
   OpenFileMode (FILE_MODE_APPEND);

   // 新規ファイルを作成して開いた状態にします。
   if (CreateFile (nvFileHandle, EXAMPLE_DIR, EXAMPLE_FILE)  0) then
      // エラーを報告します。
      MessageBox ("CreateFile が失敗しました。", SEVERE);
      abort;
   else
      // ファイルへ書き込むメッセージを設定します。
      szMsg = " この行は、サンプル InstallShield スクリプトによって追加されました。";

      // ファイルへメッセージを追加します。
      if (WriteLine(nvFileHandle, szMsg) < 0) then
         // エラーを報告します。
         MessageBox ("WriteLine が失敗しました。", SEVERE);
      else
         // 成功を報告します。
         szTitle = "CreateFile & WriteLine";
         szMsg = " 作成に成功し、%s へ書き込みました。";
         SprintfBox (INFORMATION, szTitle, szMsg, EXAMPLE_FILE);
      endif;

   endif;

   // ファイルを閉じます。
   CloseFile (nvFileHandle);

end;
```

CreateInstallationInfo

イベントベースのスクリプトでは、**CreateInstallationInfo** 関数が First UI Before イベントの後で自動的に呼び出されます。これはシステム変数 IFX_KEYPATH_PRODUCT_INFO および IFX_PRODUCT_KEY を使用して、インストールするプログラム用のアプリリケーション情報キー、およびアプリケーションごとのパスキーを作成します。アプリケーション情報キーは CreateInstallationInfo の呼び出しの結果として即座に作成されます。アプリケーションごとのパスキーは、続く RegDBSetItem の呼び出しによってそのキーの下に [Path] または [DefaultPath] の値が設定されるまで作成されません。

CreateInstallationInfo は MaintenanceStart がアンインストール ログ ファイルを初期化するために使用する会社名、製品名、バージョン番号と共に、[ようこそ] ダイアログに表示する製品名を作成します。MaintenanceStart は、スクリプト内で CreateInstallationInfo が先に呼び出されていないとエラーになります。

CreateInstallationInfo はセットアップ内で一度だけ呼び出します。DoInstall を使用して複数のインストレールを立ち上げる場合は、各インストールごとに個別に CreateInstallationInfo を呼び出すことができます。

CreateInstallationInfo はレジストリ関連の特殊関数で、特定の定義済みレジストリキーと一緒に動作するように設計されています。詳細については、「レジストリ関連の特殊関数」を参照してください。

メモ・*InstallScript エンジンは、現在、レジストリの 64 ビット部分にある [プログラムの追加と削除] 情報の読み書きをサポートしません。そのため、REGDB_OPTIONS システム変数を使った REGDB_OPTION_WOW64_64KEY オプションをこのレジストリ関数で使用することはできません。REGDB_OPTION_WOW64_64KEY オプションを有効にしても、レジストリエントリがこの関数によって作成される場所に影響しません。*

構文

```
CreateInstallationInfo ( );
```

パラメーター

なし

戻り値

テーブル 80 - CreateInstallationInfo の戻り値

戻り値	説明
0	関数が成功したことを示します。
< 0	関数がレジストリキーのうち 1 つまたはそれ以上を作成できなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

- RegDBSetItem または RegDBGetItem を呼び出す前に、CreateInstallationInfo を呼び出す必要があります。詳細については、これらの関数の説明を参照してください。
- アプリケーションごとのパスキーは、実際そのキーの下に値を設定するために RegDBSetItem が呼び出されるまでは作成されません。
- IFX_COMPANY_NAME、IFX_PRODUCT_NAME、IFX_PRODUCT_VERSION が空でも CreateInstallationInfo が失敗することはなくなりました。CreateInstallationInfo が失敗するのは、リモート レジストリ が接続されている場合、IFX_KEYPATH_PRODUCT_INFO が空の場合、またはアプリケーション情報のレジストリ キーを作成できない場合です。

CreateObject

プロジェクト - *InstallScript プロジェクトで* ***CreateObject*** *関数は CoCreateObject に置換されました。*

CreateObject 関数は、szProgID によって名付けられた登録済みの COM オブジェクトを初期化し、設定されたキーワードを使用して OBJECT 変数タイプに割り当てることができるリファレンスを戻します。

オブジェクトが無事に初期化されたかどうかを確認する場合、キーワード try-catch-endcatch を使うと COM オブジェクトの例外処理についてコントロールの幅が広がります。

メモ - *CreateObject を動作させるためには、COM オブジェクトをターゲット システム上に登録しなくてはなりません。*

構文

```
CreateObject ( szProgID );
```

パラメーター

テーブル 81 ・ CreateObject のパラメーター

パラメーター	説明
szProgID	COM オブジェクトの ProgID が初期化されるよう指定します。

戻り値

設定されたキーワードを使用して、OBJECT 型の変数に割り当てられるリファレンス。

CreateProgramFolder

CreateProgramFolder 関数はターゲットシステム上に新規フォルダーを作成します。フォルダーが [スタートプログラム] メニューに作成されました。

構文

```
CreateProgramFolder ( szFolderName );
```

パラメーター

テーブル 82 - CreateProgramFolder のパラメーター

パラメーター	説明
szFolderName	ターゲットシステムへ追加するフォルダーの名前を指定します。

戻り値

テーブル 83 - CreateProgramFolder の戻り値

戻り値	説明
0	関数によってターゲットシステムへフォルダーが追加されたこと、あるいはフォルダーが既に存在することを示します。
< 0	関数が指定されたプログラムフォルダーを追加できなかったことを示します。

CreateProgramFolder の例

***メモ -** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CreateProgramFolder 関数のデモンストレーションを行います。
*
* このスクリプトは、ターゲットシステム上に
* ExampleFolder と名づけられたプログラムフォルダーを作成します。
*
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CreateProgramFolder(HWND);

function ExFn_CreateProgramFolder(hMSI)
   STRING  szFolderName, szTitle, szMsg;
begin

   // CreateProgramFolder の呼び出し用パラメーターをセットアップします。
   szFolderName = "ExampleFolder";
   szTitle      = "CreateProgramFolder";
   szMsg        = "%s が無事に作成されました。";
```

```
    // プログラム フォルダーを作成します。
    if (CreateProgramFolder (szFolderName) < 0) then
        MessageBox (" プログラム フォルダーを作成できませんでした。", SEVERE);
    else
        SprintfBox (INFORMATION, szTitle, szMsg, szFolderName);
    endif;

end;
```

CreateRegistrySet

CreateRegistrySet 関数は、特定のコンポーネントに*関連付けられていない*、現在のメディア内の、1 つまたはすべてのレジストリセットによって指定されたレジストリエントリを作成します。(現在のメディアの名前は、システム変数の MEDIA に格納されます。)

1 つ以上のコンポーネントに関連付けられたレジストリセットに対して **CreateRegistrySet** を呼び出す必要はありません。この関数はこれらのレジストリセットには影響がないからです。コンポーネントに関連付けられたレジストリセットに保管されているレジストリエントリは、コンポーネントそのものがインストールされた時 (たとえば **FeatureTransferData** 呼び出しが行われた時) に作成されます。

***メモ・** REGDB_OPTION_WOW64_64KEY オプションを有効にすると、レジストリセットのレジストリエントリが作成される場所に影響します。例えば、**CreateRegistrySet** 関数を呼び出す時にこのオプションを有効にした場合、レジストリセットはレジストリの 64 ビット領域に作成されます。これは、**OnMoving** イベントが戻されたあと即座にファイル転送中に作成されるデフォルト レジストリ セットと、コンポーネントがインストールされたとき自動的に作成されるコンポーネントに関連付けられたレジストリのセットを含みます。したがって、これらのレジストリセットからのデータが誤ってレジストリの 64 ビットの部分で作成されるのを防ぐために、このオプションをスクリプト内の適切なレジストリ API の呼び出し中でのみ設定し、スクリプトの実行を続行する前にこのオプションを無効化することを推奨します。*

構文

```
CreateRegistrySet (szRegistrySet);
```

パラメーター

テーブル 84 - CreateRegistrySet のパラメーター

パラメーター	説明
szRegistrySet	現在のメディアで、特定のコンポーネントに*関連付けられていない*レジストリセット名を指定します。現在のメディアで定義されていて、特定のコンポーネントに関連付けられていないすべてのレジストリセットを作成する場合は、このパラメーターにヌル文字列 ("") を渡します。

戻り値

テーブル 85 - CreateRegistrySet の戻り値

戻り値	説明
0	関数が成功したことを示します。
< 0	特定できないエラーが発生したことを示します。

追加情報

- MEDIA システム変数の値は、セットアップ初期設定中に「DATA」に設定されます。この変数の値を変更してスクリプト作成の機能セットを参照するには、**CreateRegistrySet** を呼び出す前に値を 'DATA' に戻す必要があります。
- レジストリ セットがビルドされたメディアに含まれていないコンポーネントにのみ割り当てられている場合 (レジストリ セットがどの機能にも含まれていない場合、または " ビルドに含める " プロパティが [いいえ] に設定されている機能にのみ含まれる場合、またはリリース ウィザードの機能パネルを使ってビルドされたメディアから除外した機能にのみ含まれる場合)、レジストリセットのエントリはファイル転送中に自動的に作成されず、**CreateRegistrySet** の呼び出しによってインストールすることができます。

CreateRegistrySet の例

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CreateShellObjects 関数と CreateRegistrySet 関数の
* デモンストレーションを行います。
*
* メモ : このスクリプト例を実行するには、IDE の [ リソース ] ペインで
*      レジストリエントリおよびシェル オブジェクトを指定するプロジェクトを
*      作成しなくてはなりません。
*
¥*---------------------------------------------------------------------*/


 export prototype ExFn_CreateRegistrySet(HWND);

function ExFn_CreateRegistrySet(hMSI)
```

```
    NUMBER ndisk;
    STRING szPassword;
    STRING svDir;
begin

    // ファイルをコピーするためのデフォルト ターゲットを設定します。
    svDir = "C:¥¥temp";

    // ユーザーからターゲット場所を取得します。
    AskDestPath ("", "", svDir, 0);

    // 指定されたターゲット場所を対応する
    // システム変数に割り当てます。
    INSTALLDIR = svDir;

    // 進行状況インジケーターを有効にします。
    SetStatusWindow (0, "");
    Enable (STATUS);
    StatusUpdate (ON, 100);

    // ファイルを転送します。
    if (ComponentMoveData (MEDIA,ndisk,0) < 0 ) then
        MessageBox (" データの移動エラー ", SEVERE);
        abort;
    endif;

    // [ リソース ] ペインで定義されたレジストリ セットを作成します。
    if (CreateRegistrySet ("") < 0) then
        MessageBox (" レジストリセットを作成できませんでした。", SEVERE);
        abort;
    endif;

    // [ リソース ] ペインで定義されたシェル オブジェクトを作成します。
    if (CreateShellObjects ("") < 0) then
        MessageBox (" シェル オブジェクトを作成できませんでした。", SEVERE);
        abort;
    endif;

end;
```

CreateShellObjects

CreateShellObjects 関数は、現在のメディアに含まれていて、特定のコンポーネントに *関連付けられていない* ショートカットを作成します。この関数は、ターゲットシステム上にフォルダーが存在しない場合、作成したショートカットを含むフォルダーも作成します。(現在のメディアの名前は、システム変数の MEDIA に格納されます。)

1 つ以上のコンポーネントに関連付けられたショートカットに対して CreateShellObjects を呼び出す必要はありません。この関数はこれらのショートカットには影響しないからです。コンポーネントに関連付けられたショートカットは、コンポーネントそのものがインストールされた時 (たとえば FeatureTransferData 呼び出しを使った場合) に作成されます。ショートカットが作成されると、ショートカットを含むフォルダーも必要に応じて作成されます。

***注意・**この関数は、InstallShield オブジェクトのスクリプトから呼び出された場合は機能しません。シェルオブジェクトの作成は、InstallShield オブジェクトプロジェクトではサポートされていません。*

***メモ・**CreateShellObjects は、ターゲット システム上に空のフォルダーを作成しません。[ショートカット] ビューのフォルダーが空の場合、この関数はフォルダーを作成しません。この制限は、FeatureTransferData を呼び出した時に自動的に作成されるショートカットと対応するフォルダーにも適用されます。空のフォルダーを作成するには、CreateProgramFolder 関数をスクリプトで呼び出してください。*

構文

CreateShellObjects (szReserved);

パラメーター

テーブル 86・CreateShellObjects のパラメーター

パラメーター	説明
szReserved	このパラメーターにヌル文字列 ("") を渡します。他の値は使用できません。

戻り値

テーブル 87・CreateShellObjects の戻り値

戻り値	説明
0	関数が成功したことを示します。
< 0	特定できないエラーが発生したことを示します。

追加情報

MEDIA システム変数の値は、セットアップ初期設定中に「DATA」に設定されます。この変数の値を変更してスクリプト作成コンポーネントセットを参照するには、CreateShellObjects を呼び出す前に値を 'DATA' に戻す必要があります。

この関数は、FeatureTransferData が呼び出された後に呼び出す必要があります。

CreateShellObjects の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CreateShellObjects 関数と CreateRegistrySet 関数の
* デモンストレーションを行います。
```

```
 *
 * メモ : このスクリプト例を実行するには、IDE の [ リソース ] ペインで
 *      レジストリエントリおよびシェル オブジェクトを指定するプロジェクトを
 *      作成しなくてはなりません。
 *
¥*---------------------------------------------------------------*/


      export prototype ExFn_CreateShellObjects(HWND);

function ExFn_CreateShellObjects(hMSI)
    NUMBER ndisk;
    STRING szPassword;
    STRING svDir;
begin

    // ファイルをコピーするためのデフォルト ターゲットを設定します。
    svDir = "C:¥¥temp";

    // ユーザーからターゲット場所を取得します。
    AskDestPath ("","",svDir,0);

    // 指定されたターゲット場所を対応する
    // システム変数に割り当てます。
    INSTALLDIR = svDir;

    // 進行状況インジケーターを有効にします。
    SetStatusWindow (0, "");
    Enable (STATUS);
    StatusUpdate (ON, 100);

    // ファイルを転送します。
    if (ComponentMoveData (MEDIA,ndisk,0) < 0 ) then
        MessageBox (" データの移動エラー ",SEVERE);
        abort;
    endif;

    // [ リソース ] ペインで定義されたレジストリ セットを作成します。
    if (CreateRegistrySet ("") < 0) then
        MessageBox (" レジストリ セットを作成できませんでした。",SEVERE);
        abort;
    endif;

    // [ リソース ] ペインで定義されたシェル オブジェクトを作成します。
    if (CreateShellObjects ("") < 0) then
        MessageBox (" シェル オブジェクトを作成できませんでした。",SEVERE);
        abort;
    endif;

end;
```

CreateShortcut

CreateShortcut 関数を使って、以下のようなタスクを処理することができます :

- ショートカットまたはプログラム フォルダーをデスクトップまたは [スタート] の [プログラム] メニュー、またはデスクトップに追加します。szShortcutFolder パラメーターを使って、ショートカットの適切な場所を指定します。
- [スタート] メニュー上に複数階層のサブメニューを作成して、そのサブメニューにショートカットを含みます。
- ショートカットの Windows シェル プロパティを設定して、ショートカットを [スタート] メニューにピン留めできる機能を無効化するなどの動作を構成します。

*メモ・**CreateShortcut** を呼び出すためには、そのショートカット ターゲットが既にターゲット システム上に存在している必要があります。*

***CreateShortcut** はインターネット ショートカットの作成をサポートしません。*

構文

CreateShortcut (szShortcutFolder, szName, szCommandLine, szWorkingDir, szIconPath, nIcon, szShortCutKey, nFlag);

パラメーター

テーブル 88 · CreateShortcut のパラメーター

パラメーター	説明
szShortcutFolder	ショートカットを含めるフォルダーの名前を指定するか、作成するプログラム フォルダーの名前を指定します。フォルダーが存在しない場合は、インストーラーによって作成されます。このパラメーターに、複数レベルの階層メニューでサブフォルダーを指定できます。サブフォルダーが存在しない場合、**CreateShortcut** がサブフォルダーを作成し、必要に応じて親フォルダーも作成します。 ショートカットを特定のフォルダーに追加する場合、以下のような完全修飾パスを指定します : **C:¥¥ProgramData¥¥Microsoft¥¥Windows¥¥Start Menu¥¥Programs** [スタート] メニューにある [プログラム] メニューにショートカットアイコンを追加するには、このパラメーターにヌル文字列 ("") を渡します。 次の InstallScript システム変数の 1 つをこのパラメーターで渡すことができます : · FOLDER_DESKTOP – デスクトップにショートカットを追加します。 · FOLDER_STARTUP – スタートアップ メニューにショートカットを追加します。 · FOLDER_STARTMENU – スタート メニューにショートカットを追加します。 · FOLDER_PROGRAMS – スタート ¥ プログラム メニューにショートカットを追加します。 InstallScript システム変数によって識別されるフォルダーの相対パスを指定することもできます。例 : **FOLDER_PROGRAMS ^ "ACCESSORIES¥¥GAMES"**
szName	ショートカット名を指定します。**CreateShortcut** を呼び出してショートカットをプログラム フォルダーに追加すると、szCommandLine で指定されたリンク ディレクトリにリンクファイルも作成されます。Windows Shell では、項目名に「/、¥、:、?、<、>、または |」を使用できません。

テーブル 88 ・ CreateShortcut のパラメーター (続き)

パラメーター	説明
szCommandLine	次のうちの 1 つを指定します : ・ コマンドライン パラメーターをすべて含む、ショートカットに関連付けられた実行可能ファイルの完全修飾名。これは、ショートカットの [プロパティ] ダイアログ ボックスにある " リンク先 " 値に追加されます。Start Programs メニューにショートカットを追加するには、リンク ディレクトリの完全修飾パスを入力します。ここにアプリケーションがそのアイコン リンク ファイルを格納します。 ・ szName がサブフォルダーの場合、完全修飾パス。  ***注意 ・ コマンドラインに長いファイル名が含まれる場合は、引用符で囲む必要があります。しかし、コマンドラインパラメーターは引用符で囲んではなりません。そのため、szCommandLine 文字列を 2 つの個別の文字列から構築することをお勧めします。***
szWorkingDir	このショートカット ターゲットの作業ディレクトリを入力します。 szName がサブフォルダーの場合、このパラメーターは適用しません。 **CreateShortcut** は、ショートカットの [プロパティ] ダイアログ ボックスの [ショートカット] タブにある [作業フォルダー] ボックスにこのディレクトリを書き込みます。ヌル文字列 ("") をこのパラメーターに渡すと、関数はこの [作業フォルダー] ボックスを空白のままにして、[リンク先] ボックスのパスが使用されます。  ***注意 ・ LongPathToQuote を呼び出してこのパスを引用符で囲まないようにしてください。InstallShield はこれらのパスも自動的に引用符で囲みます。***
szIconPath	ショートカットに表示するアイコンを含むファイルへの完全修飾パスを指定します。 szItemName がサブフォルダーの場合、このパラメーターは適用しません。  ***注意 ・ LongPathToQuote を呼び出してこのパスを引用符で囲まないようにしてください。InstallShield はこれらのパスも自動的に引用符で囲みます。***
nIcon	szIconPath で指定された実行可能ファイルに含まれるアイコン インデックスを指定します。インデックス 0 はファイルの最初のアイコンを、インデックス 1 は 2 番目のアイコンを意味します。以降の番号も同じように続きます。アイコンを使用しない場合、このパラメーターに **0** を指定します。 szName がサブフォルダーの場合、このパラメーターは適用しません。

テーブル 88 • CreateShortcut のパラメーター (続き)

パラメーター	説明
szShortCutKey	ショートカットに割り当てるショートカット キーを文字列の形式で指定します。ショートカットに szShortCutKey を設定すると、エンド ユーザーが適切なホット キーを押してショートカットを起動することができます。 たとえば、エンド ユーザーが Ctrl と Alt キーを押しながら数字の 1 を押すと製品が起動するように設定する場合は、このパラメーターに "Ctrl+Alt+1" を渡します。 szName がサブフォルダーの場合、このパラメーターは適用しません。

テーブル 88 • CreateShortcut のパラメーター (続き)

パラメーター	説明
nFlag	このパラメーターには、次の定義済み定数のいずれかを指定します。このパラメーターに複数の定義済み定数を渡す場合、各定数をビット単位 OR 演算子 (|) で区切ってください。 • CS_OPTION_FLAG_REPLACE_EXISTING – 既存のショートカットを置換します。 • CS_OPTION_FLAG_RUN_MAXIMIZED – 起動時にショートカットのターゲットが最大化されます。 • CS_OPTION_FLAG_RUN_MINIMIZED – 起動時にショートカットのターゲットが最小化されます。 • CS_OPTION_FLAG_PREVENT_PINNING – Windows 7 以降のシステムで、[スタート] メニューまたはタスクバーにショートカットをピン留めすることを防ぎます。このオプションは、エンド ユーザーがタスクバーおよび [スタート] メニューにショートカットをピン留めするためのコンテキスト メニュー コマンドを隠します。 インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを防ぎたい場合があります。 • CS_OPTION_FLAG_NO_NEW_INSTALL_HIGHLIGHT – エンド ユーザーが製品を Windows 7 以降のシステム上にインストールした後、ショートカットを新しくインストールされたプログラムとして強調表示しません。これは、ターゲット システム上で個別のアイテムに対して [[スタート] メニューのカスタマイズ] ダイアログ ボックスで **[新しくインストールされたプログラムを強調表示する]** チェック ボックスをクリアした場合と同じ効果を持ちます。 インストールの一部であるツールまたは従属的な製品のショートカットのオプションを使用したい場合があります。 • CS_OPTION_FLAG_NO_STARTSCREEN_PIN – Windows 8 ターゲット システム上で、デフォルトで [スタ — ト] 画面にショートカットをピン留めしません。この定数を渡すと、インストールは Windows 8 で使用可能になった Windows シェル プロパティを設定します。 インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを防ぎたい場合があります。 • NULL – オプションがないことを示します。 CS_OPTION_FLAG_PREVENT_PINNING および CS_OPTION_FLAG_NO_STARTSCREEN_PIN についての詳細は、追加情報セクションを参照してください。

戻り値

テーブル 89 • CreateShortcut の戻り値

戻り値	説明
0	関数が指定されたフォルダー内のショートカットを追加または置換し、実行可能ファイルに関連付けたことを示します。
< 0	関数がショートカットの追加または置換、また実行可能ファイルとの関連付けに失敗したことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

Additional Information

2 つの nFlag 定数について、以下にご注意ください。

CS_OPTION_FLAG_PREVENT_PINNING

ピン留めを行わないようにショートカットを構成した場合、[スタート] メニューの最もよく使われている製品のリストに、ショートカットのターゲットを含められなくなります。

特定の文字列を含むショートカットは、タスクバーまたは [スタート] メニューにピン留めすることができません。また、それらを最もよく使う製品リストに表示することもできません。例 :

- Documentation
- ヘルプ
- Install
- 削除
- Setup
- Support

CS_OPTION_FLAG_NO_STARTSCREEN_PIN

Windows 8 は、アプリケーションのアンインストールによってショートカットが削除された後でも、ショートカットの [スタート] 画面へのピン留めに関する情報を保持します。そのため、ショートカットがインストール済みの場合、ターゲット システム上で CS_OPTION_FLAG_NO_STARTSCREEN_PIN 定数は効果を持ちません。この機能をテストする際、ショートカットとそのターゲットが既にインストールされていない、クリーン マシン上でテストするようにして下さい。

CreateShortcut の例

次の例では、**CreateShortcut** の使い方をデモンストレーションします :

- スタート メニューとスタートプログラムメニューにある実行可能ファイルへショートカットを配置する。 (**CreateShortcut** 例 1)

- Startup メニューに複数階層のサブメニューを作成して、そのメニューにショートカットを追加する。(**CreateShortcut** 例 2)
- デスクトップ上にサブフォルダー、並びに新規フォルダーの実行可能ファイルをポイントするショートカットを配置する。(**CreateShortcut** 例 3)

CreateShortcut 例 1

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CreateShortcut 関数のデモンストレーションを行います。
*
* この例ではスタート メニューとスタートプログラムメニューにある
* 実行可能ファイルへショートカットを配置します。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*      ターゲットシステム上の Windows Notepad 実行可能ファイルの
*      有効なテキスト ファイルの完全修飾名を
*      参照するように設定してください。
*
¥*---------------------------------------------------------------*/

#define PROGRAM "C:¥¥Windows¥¥Notepad.exe"
#define PARAM   "C:¥¥Windows¥¥Readme.txt"

function OnFirstUIAfter( )
    STRING szShortcutFolder, szName, szCommandLine, szWorkingDir;
    STRING szShortCutKey, szProgram, szParam, szIconPath;
    NUMBER nIcon;
begin

    // CreateShortcut の呼び出し用パラメーターをセットアップします。
    szShortcutFolder = FOLDER_STARTMENU;
    szName      = " メモ帳の例 1";
    szProgram       = PROGRAM;
    szParam         = PARAM;

    LongPathToQuote (szProgram, TRUE);

    LongPathToShortPath (szParam);

    szCommandLine = szProgram + " " + szParam;
    szWorkingDir  = "";
    szIconPath    = "";
    nIcon         = 0;
    szShortCutKey = "";

    // スタート メニューへショートカットを追加します。
    if (CreateShortcut (szShortcutFolder, szName, szCommandLine, szWorkingDir, szIconPath,
```

```
                nIcon, szShortCutKey, CS_OPTION_FLAG_REPLACE_EXISTING) < 0) then
        MessageBox ("CreateShortcut が失敗しました。", SEVERE);
    else
        SprintfBox (INFORMATION, "CreateShortcut", "%s 無事に作成されました。",
                szName);
    endif;

    szShortcutFolder = "";
    szName          = " 別のメモ帳の例 ";

    // プログラム メニューへショートカットを追加します。
    if (CreateShortcut (szShortcutFolder, szName, szCommandLine, szWorkingDir, szIconPath,
                nIcon, szShortCutKey, CS_OPTION_FLAG_REPLACE_EXISTING) < 0) then
        MessageBox ("CreateShortcut が失敗しました。", SEVERE);
    else
        SprintfBox (INFORMATION, "CreateShortcut", "%s 無事に作成されました。",
                szName);
    endif;

end;
```

CreateShortcut 例 2

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CreateShortcut 関数のデモンストレーションを行います。
*
* この例では、Startup メニューに複数階層のサブメニューを作成し、
* そこに実行可能ファイルへのショートカットを追加します。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*      ターゲットシステム上の Windows Notepad 実行可能ファイルの
*      完全修飾名と有効なテキスト ファイルを適切に
*      参照するように設定してください。
*
¥*---------------------------------------------------------*/

#define PROGRAM "C:¥¥Windows¥¥Notepad.exe"
#define PARAM   "C:¥¥Windows¥¥Readme.txt"

function OnFirstUIAfter( )
    STRING  szShortcutFolder, szName, szCommandLine, szWorkingDir;
    STRING  szIconPath, szShortCutKey, szProgram, szParam;
    NUMBER  nIcon, nFlag, nResult;
begin

    // Startup サブメニューの完全修飾名を設定します。
    szShortcutFolder = FOLDER_STARTUP ^ "SubMenu 例 ";

    // ショートカットのコマンドライン プロパティを構築します。
```

```
    szProgram= PROGRAM;
    szParam    = PARAM;

    LongPathToQuote (szProgram, TRUE);

    LongPathToShortPath (szParam);

    szCommandLine = szProgram + " " + szParam;

    // CreateShortcut へ渡すショートカットの他のプロパティを設定します。
    szName       = " メモ帳の例 2";
    szWorkingDir  = "";
    szIconPath    = "";
    nIcon         = 0;
    szShortCutKey = "";
    nFlag         = CS_OPTION_FLAG_REPLACE_EXISTING|CS_OPTION_FLAG_RUN_MAXIMIZED;

    // サブメニューにショートカットを追加、および必要に応じてサブメニューを作成します。
    nResult = CreateShortcut (szShortcutFolder, szName, szCommandLine,
                              szWorkingDir, szIconPath, nIcon,
                              szShortCutKey, nFlag);

    // 結果をレポートします。
    if (nResult < 0) then
       MessageBox ("CreateShortcut が失敗しました。", SEVERE);
    else
       SprintfBox (INFORMATION, "CreateShortcut", "%s 無事に作成されました。",
                   szName);
    endif;

end;
```

CreateShortcut 例 3

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CreateShortcut 関数のデモンストレーションを行います。
*
* この例ではデスクトップ上にサブフォルダー、並びに新規フォルダーの
* 実行可能ファイルをポイントするショートカットを配置します。フォルダーは、
* 実際のディレクトリをポイントするショートカットです。このフォルダーから
* エンド ユーザーはプログラムを実行するショートカットを起動できます。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*      ターゲットシステム上の Windows Notepad 実行可能ファイルの
*      有効なテキスト ファイルの完全修飾名を
*      参照するように設定してください。
*
¥*---------------------------------------------------------*/
```

```
#define FOLDER "C:¥¥Windows¥¥"
#define PROGRAM "C:¥¥Windows¥¥Notepad.exe"
#define PARAM "C:¥¥Windows¥¥Readme.txt"

function OnFirstUIAfter( )
    STRING  szShortcutFolder, szName, szCommandLine, szWorkingDir;
    STRING  szIconPath, szShortCutKey;
    STRINGszProgram, szParam, szFolderDir;
    NUMBER  nIcon, nFlag, nResult;
begin

   // szShortcutFolder はローカル システム上のデスクトップです。
   szShortcutFolder = FOLDER_DESKTOP;
   szName           = " フォルダー例 3";

   // フォルダー アイコンが指定するフォルダーを作成します。
   szFolderDir = FOLDER ^ szName;
   CreateDir(szFolderDir);

   // フォルダー アイコンのコマンドラインはフォルダー パスでなくてはなりません。
   // また、パスが 8 文字以上である場合、
   // 引用符で囲む必要があります。

   szCommandLine = szFolderDir;
   LongPathToQuote(szCommandLine, TRUE);

   szWorkingDir  = "";
   szIconPath    = "";
   nIcon         = 0;
   szShortCutKey = "";
   nFlag         = CS_OPTION_FLAG_REPLACE_EXISTING|CS_OPTION_FLAG_RUN_MINIMIZED;

   // フォルダー ショートカットを作成し、そのアイコンが指定する目的のフォルダーを表示します。
   nResult = CreateShortcut (szShortcutFolder, szName, szCommandLine,
                             szWorkingDir, szIconPath, nIcon, szShortCutKey,
                             nFlag);

   if (nResult < 0) then
     MessageBox ("CreateShortcut が失敗しました。", SEVERE);
   else
     SprintfBox (INFORMATION, "CreateShortcut", "%s 無事に作成されました。",
                 szName);
   endif;

   // 作成したフォルダーを表示する。
   ShowProgramFolder (szFolderDir, SW_SHOW);

   // Example ショートカットを新しく作成したフォルダーに追加します。
   szShortcutFolder = szFolderDir;
   szName           = " メモ帳の例 3";

   // 空白スペースが区切り文字として間違えられないよう注意してください。
   szProgram        = PROGRAM;
   LongPathToQuote (szProgram, TRUE);

   szParam= PARAM;
   LongPathToShortPath (szParam);

   szCommandLine = szProgram + " " + szParam;
```

```
    szWorkingDir  = "";
    szIconPath    = "";
    nResult = CreateShortcut (szShortcutFolder, szName, szCommandLine,
                              szWorkingDir, szIconPath, nIcon, szShortCutKey,
                              nFlag);

    if (nResult < 0) then
      MessageBox ("CreateShortcut が失敗しました。", SEVERE);
    else
      SprintfBox (INFORMATION, "CreateShortcut", "%s 無事に作成されました。",
                  szName);
    endif;

end;
```

CreateShortcutFolder

CreateShortcutFolder 関数はターゲットシステム上に新規フォルダーを作成します。フォルダーが [スタートプログラム] メニューに作成されます。フォルダーが既に存在する場合、ハイライト表示されます。

構文

```
CreateShortcutFolder (szShortcutFolder);
```

パラメーター

テーブル 90 - CreateShortcutFolder のパラメーター

パラメーター	説明
szShortcutFolder	ターゲットシステムへ追加するフォルダーの名前を指定します。

戻り値

テーブル 91 - CreateShortcutFolder の戻り値

戻り値	説明
0	関数によってターゲットシステムへフォルダーが追加されたこと、あるいはフォルダーが既に存在することを示します。
< 0	関数が指定されたプログラムフォルダーを追加できなかったことを示します。

CreateShortcutFolder の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * CreateShortcutFolder 関数のデモンストレーションを行います。
 *
 * このスクリプトは、ターゲットシステム上に
 * ExampleFolder と名づけられたプログラムフォルダーを作成します。
 *
¥*---------------------------------------------------------------------*/

function OnFirstUIAfter( )
    STRING  szShortcutFolder, szTitle, szMsg;
begin

    // CreateShortcutFolder の呼び出し用パラメーターをセットアップします。
    szShortcutFolder = "ExampleFolder";
    szTitle      = "CreateShortcutFolder";
    szMsg        = "%s が無事に作成されました。";

    // プログラム フォルダーを作成します。
    if (CreateShortcutFolder (szShortcutFolder) < 0) then
        MessageBox ("プログラム フォルダーを作成できませんでした。", SEVERE);
    else
        SprintfBox (INFORMATION, szTitle, szMsg, szShortcutFolder);
    endif;

end;
```

CtrlClear

CtrlClear 関数は各種コントロールの内容をクリアします。すなわち、一行、または複数行の編集フィールド、スタティック テキスト フィールド、単一の、または複数の選択リストボックス、あるいは、カスタム ダイアログ内コンボ ボックスの編集フィールドの内容を削除します。

構文

```
CtrlClear ( szDialogName, nControlID );
```

パラメーター

テーブル 92 · CtrlClear のパラメーター

パラメーター	説明
szDialogName	削除するコントロールを含むダイアログの名前を指定します。
nControlID	szDialogName が認識したダイアログのコントロール ID を指定します。

戻り値

テーブル 93 · CtrlClear の戻り値

戻り値	説明
0	CtrlClear が指定したコントロールの内容を削除しました。
< 0	CtrlClear はダイアログの内容を削除できませんでした。

CtrlClear の例

***メモ ·** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CtrlClear 関数、CtrlGetText 関数、および CtrlSetText 関数の
* デモンストレーションを行います。
*
* このスクリプト例では、4 つのチェック ボックスを含むカスタム ダイアログを
* カスタムダイアログボックスを表示します。
* また、3 つのボタン ([ すべてクリア ]、[ 完了 ]、および [ キャンセル ]) が含まれます。
* ダイアログの初期化時に、スクリプトは CtrlSetText を呼び出して
* 各編集コントロールへデフォルト テキストを配置します。
*
* ボタン操作
*
* すべてクリア     すべての編集ボックスをクリアする CtrlClear を呼び出します。
*
* 完了         編集ボックスの値を読み出す CtrlGetText を呼び出します。
*            すべてのフィールドがデータを含む場合、
*            ダイアログ ボックスが閉じ、編集ボックスの値が
*            メッセージボックスに表示されます。
```

```
*
*キャンセル        ダイアログを閉じます。編集ボックス値が読み出されず、
*             表示もされません。.
*
*このスクリプトで利用される [ カスタム ] ダイアログは、
*実際、ビルトイン関数 SdRegisterUserEx が表示する
*SdRegisterUserEx このダイアログは
*インストールで既に圧縮済みのファイル _isres.dll に保存されており、
*カスタム ダイアログとしてスクリプトで利用することが
*表示されます。
*
¥*-------------------------------------------------------------*/

// 編集ボックスに表示する初期値。
#define USER_NAME    " ユーザー名 "
#define COMPANY_NAME " 会社名 "
#define SERIAL_NUM   "123"

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID       12002   // カスタム ダイアログの ID
#define RES_EDIT_NAME       301   // ユーザー名編集ボックスの ID
ne RES_EDIT_COMPANY    302   // 会社名編集ボックスの ID
#define RES_EDIT_SERIAL     303   // シリアル番号編集ボックスの ID
#define RES_PBUT_DONE1 // ダイアログの [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL 9 // ダイアログの [ キャンセル ] ボタンの ID
#define RES_PBUT_CLEAR12 // ダイアログの [ 戻る ] ボタンの ID

  STRING szDialogName, svName, svCompany, svSerial;
  NUMBER nResult, nCmdValue;
  BOOL   bDone;
  HWND hwndDlg;

#include "ifx.h"

function OnBegin()
begin

  // このインストールでカスタム ボックスを認識するための名前を指定します。
  szDialogName = "CustomDialog";

  // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
  // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
  // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
  // 4 番目のパラメーターにある ID によって識別されるためです。
  nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

  if (nResult < 0) then
    // エラーを報告し、終了します。
    MessageBox (" ダイアログの定義エラー ", SEVERE);
    abort;
  endif;

  // ループを制御するのに使われるインジケーターを初期化します。
  bDone = FALSE;

  // 完了するまでループします。
  repeat

    // ダイアログを表示して次のダイアログ イベントを戻します。
```

```
    nCmdValue = WaitOnDialog (szDialogName);

    // イベントに応答します。
    switch (nCmdValue)

      case IDCANCEL:
        // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
        Do (EXIT);

      case DLG_ERR:
        MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
        abort;

      case DLG_INIT:
        // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
        // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
        // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
        // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
        hwndDlg = CmdGetHwndDlg (szDialogName);
        SdGeneralInit(szDialogName, hwndDlg, 0, "");

        // ボタンのスタティック テキストを設定します。
        CtrlSetText (szDialogName, RES_PBUT_CLEAR, " すべてクリア (&A)");
        CtrlSetText (szDialogName, RES_PBUT_DONE, " 完了 (&D)");

        // 編集コントロールを初期化します。
        CtrlSetText (szDialogName, RES_EDIT_NAME, USER_NAME);
        CtrlSetText (szDialogName, RES_EDIT_COMPANY, COMPANY_NAME);
        CtrlSetText (szDialogName, RES_EDIT_SERIAL, SERIAL_NUM);

      case RES_PBUT_CLEAR:
        // 編集コントロールをすべてクリアします。
        CtrlClear (szDialogName, RES_EDIT_NAME);
        CtrlClear (szDialogName, RES_EDIT_COMPANY);
        CtrlClear (szDialogName, RES_EDIT_SERIAL);

      case RES_PBUT_DONE:
        // 編集ボックスからテキストを読み出します。
        CtrlGetText (szDialogName, RES_EDIT_NAME, svName);
        CtrlGetText (szDialogName, RES_EDIT_COMPANY, svCompany);
        CtrlGetText (szDialogName, RES_EDIT_SERIAL, svSerial);

        // 3 つのすべてのボックスにデータが入力されていることを確認します。
        if (StrLength (svName) = 0) ||
          (StrLength (svCompany) = 0) ||
          (StrLength (svSerial) = 0) then
            MessageBox (" すべてのフィールドに入力する必要があります。", WARNING);
        else
          bDone = TRUE;
        endif;

      case RES_PBUT_CANCEL:
        // ユーザーが [ キャンセル ] ボタンをクリックしました。
        Do (EXIT);
    endswitch;

until bDone;

// ダイアログを閉じます
```

```
    EndDialog (szDialogName);

    // メモリからダイアログを解放します。
    ReleaseDialog (szDialogName);

    // [ 完了 ] ボタンを使ってダイアログを閉じた場合、
    // 編集コントロールからテキストを表示します。
    if (nCmdValue = RES_PBUT_DONE) then
        SprintfBox (INFORMATION, " ユーザー情報 ",
                    " 名前 : %s¥n¥n 会社 : %s¥n¥n シリアル番号 : %s",
                    svName, svCompany,svSerial);
    endif;

end;
```

CtrlDir

CtrlDir 関数は、指定したパスまたは szDir のファイル名に適合するファイルリストをリストボックスまたはコンボボックスコントロールに入力します。リストにはファイル名、サブディレクトリ、そしてディスクドライブを含むことができます。CtrlDir 関数はカスタム ダイアログでのみ利用できます。

構文

```
CtrlDir ( szDialogName, nControlID, szDir, nItems );
```

パラメーター

テーブル 94 ・ CtrlDir のパラメーター

パラメーター	説明
szDialogName	ダイアログの名前を指定します。
nControlID	リストボックスまたはコンボボックスコントロールのリソース ID を指定します。
szDir	完全修飾パスまたはファイル名を指定します。ワイルドカード文字を含むことも可能です。
nItems	コントロールに表示するリストの種類を指定します。このパラメーターには、次の定義済み定数のいずれかを指定します。異なるタイプの要素を含むには、ビット単位 OR 演算子 (|) を使ってこれらの定数を組み合わせます。 ・ DLG_DIR_FILE – ファイル形式指定 szDir に一致するファイルのリストを作成します。 ・ DLG_DIR_DIRECTORY – パス形式指定 szDir に存在するサブディレクトリのリストを作成します。 ・ DLG_DIR_DRIVE – ドライブのリストを作成します。

戻り値

テーブル 95 ・ CtrlDir の戻り値

戻り値	説明
0	CtrlDir はダイアログに特定のコントロールを配置しました。
< 0	CtrlDir は指定されたコントロールを配置できませんでした。

CtrlDir の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
```

```
＊InstallShield スクリプトの例
＊
＊CtrlDir 関数のデモンストレーションを行います。
＊
＊このスクリプト例では、カスタム ダイアログのファイルリストを表示
＊表示されます。ディレクトリのリストは CtrlDir への呼び出しで作成
＊されます。これはリストをビルドしてカスタム ダイアログの
＊リスト ボックス コントロールへ配置します。そしてユーザーがキーボード、
＊またはマウスを使ってリストからアイテムを選択することができます。インストールが
＊[ 次へ ] ボタンを使って閉じられた場合、選択されたアイテムは
＊メッセージボックスに表示されます。
＊
＊このスクリプトで利用される [ カスタム ] ダイアログは、
＊実際、ビルトイン関数 SdAskOptions が表示する
＊InstallShield 標準ダイアログです。このダイアログは
＊インストールで既に圧縮済みのファイル _isres.dll に保存されており、
＊カスタム ダイアログとしてスクリプトで利用することが
＊表示されます。
＊
¥*-------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID       12033   // カスタム ダイアログの ID
#define RES_PBUT_NEXT        1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL        9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK         12   // [ 戻る ] ボタンの ID
#define RES_DIALOG_LISTBOX    401   // リストボックスの ID
#define RES_STA_MSG_ABOVE     710   // スタティック メッセージ上のリストの ID
#define RES_STA_MSG_BELOW     711   // スタティック メッセージ下のリストの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlDir(HWND);

function ExFn_CtrlDir(hMSI)
   STRING  szDialogName, svSelection;
   NUMBER  nResult, nCmdValue;
   BOOL    bDone;
   HWND    hwndDlg;
begin

   // このインストールでカスタム ボックスを認識するための名前を指定します。
   szDialogName    = "CustomDialog";

   // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
   // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
   // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
   // 4 番目のパラメーターにある ID によって識別されるためです。
   nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

   // エラーを報告し、終了します。
   if (nResult < 0) then
      MessageBox ("ダイアログの定義エラー", SEVERE);
      abort;
   endif;

   // ループを制御するのに使われるインジケーターを初期化します。
   bDone = FALSE;
```

```
// 完了するまでループします。
repeat

   // ダイアログを表示して次のダイアログ イベントを戻します。
   nCmdValue = WaitOnDialog (szDialogName);

   // イベントに応答します。
   switch (nCmdValue)
      case DLG_CLOSE:
         // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
         Do (EXIT);
      case DLG_ERR:
         MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
         abort;
      case DLG_INIT:
         // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
         // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
         // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
         // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
         hwndDlg = CmdGetHwndDlg (szDialogName);
         SdGeneralInit(szDialogName, hwndDlg, 0, "");

         // リスト ボックスの上と下に表示されるメッセージを設定します。
         CtrlSetText (szDialogName, RES_STA_MSG_ABOVE,
                  "Windows ドライブのルート ディレクトリで検出されたファイル :");
         CtrlSetText (szDialogName, RES_STA_MSG_BELOW,
                  " ファイルを選択してから、[ 次へ ] ボタンをクリックします ");

         // Windows があるドライブのルートディレクトリ内のすべての、
         // ファイル リストを作成し、ダイアログのリスト ボックス コントロールへ
         // 配置します。
         if (CtrlDir (szDialogName, RES_DIALOG_LISTBOX, WINDISK^"*.*",
                  DLG_DIR_FILE) < 0) then
            MessageBox ("CtrlDir が失敗しました。", SEVERE);
            bDone = TRUE;
         endif;
      case RES_PBUT_CANCEL:
          // ユーザーが [ キャンセル ] ボタンをクリックしました。
          Do (EXIT);
      case RES_PBUT_NEXT:
           // 表示するために現在の選択を取得します。
           CtrlGetCurSel (szDialogName, RES_DIALOG_LISTBOX, svSelection);
           bDone = TRUE;
      case RES_PBUT_BACK:
           bDone = TRUE;
   endswitch;

until bDone;

// ダイアログを閉じます
EndDialog (szDialogName);

// メモリからダイアログを解放します。
ReleaseDialog (szDialogName);

// [ 次へ ] ボタンを使ってダイアログが閉じられた場合、
// リストボックスから選択されたアイテムを表示します。
if (nCmdValue = RES_PBUT_NEXT) then
```

```
        MessageBox ( " 選択したのは " + svSelection, INFORMATION);
    endif;

end;
```

CtrlGetCurSel

CtrlGetCurSel 関数は、現在選択しているアイテムをカスタム ダイアログ内の、選択リストボックスまたはコンボボックス コントロールから読み出します。複数選択のリストボックスからアイテムを読み出すときは CtrlGetMultCurSel 関数を利用してください。

構文

```
CtrlGetCurSel ( szDialogName, nControlID, svText );
```

パラメーター

テーブル 96 - CtrlGetCurSel のパラメーター

パラメーター	説明
szDialogName	読み出すアイテムを含むカスタム ダイアログの名前を指定します。
nControlID	単一の選択リストボックスまたはコンボボックスコントロールのリソース ID を指定します。
svText	nControlID で指定されたコントロールを使って現在選択されているアイテムを戻します。

戻り値

テーブル 97 - CtrlGetCurSel の戻り値 I

戻り値	説明
0	CtrlGetCurSel がダイアログから現在選択されている項目を読み出しました。
< 0	CtrlGetCurSel は選択された項目を読み出すことができませんでした。

CtrlGetCurSel の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * CtrlGetCurSel 関数と CtrlSetCurSel 関数のデモンストレーションを行います。
 *
 * このスクリプト例では、編集ボックスとリスト ボックスを持つ
 * カスタム ダイアログを表示します。ダイアログが初期化された後、
 * スクリプトは Windows ディスクのルートにある
 * フォルダー名をダイアログのリスト ボックスに配置します。そこで
 * そして CtrlSetCurSel を呼び出して "Windows" を選択フォルダーとします。
 *
 * ユーザーがリストボックスからフォルダー名を選択するたびに
 * スクリプトは CtrlGetCurSel を呼び出して選択したアイテムを取得し、
 * 編集ボックスに配置できるようにします。ダイアログが
 * [ 完了 ] ボタンを使って閉じられた場合、現在選択しているアイテムが
 * メッセージボックスに表示されます。
 *
 * このスクリプトで利用される [ カスタム ] ダイアログは、
 * 実際、ビルトイン関数 SdAskOptions が表示する
 * InstallShield 標準ダイアログです。このダイアログは
 * インストールで既に圧縮済みのファイル _isres.dll に保存されており、
 * カスタム ダイアログとしてスクリプトで利用することが
 * 表示されます。
 *
¥*---------------------------------------------------------------------*/

// リストボックスで選択済みとなるフォルダー。
#define PRESELECTED_FOLDER "windows"

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID       12008  // カスタム ダイアログの ID
#define RES_PBUT_NEXT        1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL      9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK        12  // [ 戻る ] ボタンの ID
#define RES_DIALOG_EDITBOX   301  // 編集ボックスの ID
#define RES_DIALOG_LISTBOX   401  // リストボックスの ID
#define RES_STA_DESC710// ダイアログ上部のテキスト ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlGetCurSel(HWND);

function ExFn_CtrlGetCurSel(hMSI)
   STRING  szDialogName, svSelection, szDesc;
   NUMBER  nResult, nCmdValue;
   BOOL    bDone;
   HWND    hwndDlg;
begin

   // このインストールでカスタム ボックスを認識するための名前を指定します。
   szDialogName = "CustomDialog";

   // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
   //  _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
   // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
   // 4 番目のパラメーターにある ID によって識別されるためです。
   nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);
```

```
if (nResult < 0) then
   // エラーを報告し、終了します。
   MessageBox ("ダイアログの定義エラー", SEVERE);
   abort;
endif;

// 完了するまでループします。
repeat

   // ダイアログを表示して次のダイアログ イベントを戻します。
   nCmdValue = WaitOnDialog (szDialogName);

   // イベントに応答します。
   switch (nCmdValue)
      case DLG_CLOSE:
         // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
         Do (EXIT);
      case DLG_ERR:
         MessageBox ("ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
         abort;
      case DLG_INIT:
         // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
         // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
         // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
         // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
         hwndDlg = CmdGetHwndDlg (szDialogName);
         SdGeneralInit(szDialogName, hwndDlg, 0, "");

         // ウィンドウのタイトルを設定します。
         SetWindowText (hwndDlg, "Select Folder");

         // ダイアログの上部に表示されるメッセージを設定します。
         szDesc = "ドライブのルートから既存フォルダーを指定します"
                + WINSYSDISK + "¥n そして [ 次へ ] を押して続行します。";
         CtrlSetText (szDialogName,  RES_STA_DESC, szDesc);

         // ダイアログのリスト ボックスへ Windows ドライブのルートにある
         // すべてのフォルダーの名前を入力します。
         CtrlDir (szDialogName, RES_DIALOG_LISTBOX,
                WINSYSDISK + "¥¥*.*", DLG_DIR_DIRECTORY);

         // 選択済みフォルダーを選びます。
         CtrlSetCurSel (szDialogName, RES_DIALOG_LISTBOX,
                        PRESELECTED_FOLDER);

         // 選択済みフォルダーの名前を編集ボックスへ配置します。
         CtrlSetText (szDialogName, RES_DIALOG_EDITBOX, PRESELECTED_FOLDER);
      case RES_DIALOG_LISTBOX:
         // 現在のリストボックスの選択部分を取得します。
         CtrlGetCurSel (szDialogName, RES_DIALOG_LISTBOX, svSelection);

         // 括弧を削除します。
         StrSub (svSelection, svSelection, 1, StrLength(svSelection) - 2);

         // 現在の選択部分を編集ボックスに配置します。
         CtrlSetText (szDialogName, RES_DIALOG_EDITBOX, svSelection);
      case RES_PBUT_BACK:
         bDone = TRUE;
```

```
            case RES_PBUT_NEXT:
                // 編集ボックスから選択部分を取得します。
                CtrlGetText (szDialogName, RES_DIALOG_EDITBOX, svSelection);

                // 編集ボックスに Windows ディスクのルートに存在する
                // フォルダーの名前が含まれていることを確認します。
                if Is (PATH_EXISTS, WINSYSDISK + "¥¥"+ svSelection) then
                    bDone = TRUE;
                else
                    MessageBox (" フォルダーが存在しません。", WARNING);
                endif;
            case RES_PBUT_CANCEL:
                // ユーザーが [ キャンセル ] ボタンをクリックしました。
                Do (EXIT);
        endswitch;

    until bDone;

    // カスタム ダイアログを閉じます
    EndDialog (szDialogName);

    // メモリからカスタム ダイアログを削除します。
    ReleaseDialog (szDialogName);

    // [ 完了 ] ボタンを使って編集ボックスを閉じた場合、
    // 選択した項目を表示します。
    if (nCmdValue = RES_PBUT_NEXT) then
        MessageBox (svSelection + " が選択されました。", INFORMATION);
    endif;

end;
```

CtrlGetDlgItem

CtrlGetDlgItem は、カスタム ダイアログ内のコントロールのウィンドウ ハンドルを取得します。**CtrlGetDlgItem** は Windows API **GetDlgItem** と似ていますが、**CtrlGetDlgItem** の場合、ダイアログのウィンドウ ハンドルの代わりに InstallScript ダイアログ名を指定できます。

構文

```
CtrlGetDlgItem (byval string szDialogName, byval HWND hDialog, byval number nCtrlId);
```

パラメーター

テーブル 98 - CtrlGetDlgItem のパラメーター

パラメーター	説明
szDialogName	読み出すハンドルを持つコントロールを含むダイアログの名前を指定します。関数が呼び出される時点で、そのダイアログが存在していなくてはなりません。 hDialog がヌル文字列 ("") 以外の場合、szDialogName にヌル文字列を指定できます。
hDialog	読み出すハンドルを持つコントロールを含むダイアログのウィンドウ ハンドルを指定します。関数が呼び出される時点で、そのダイアログが存在していなくてはなりません。 文字列 ("") を指定すると、**CtrlGetDlgItem** が szDialogName を使って **CmdGetHwndDlg** を呼び出すことで、ダイアログのハンドルを判別します。
nCtrlId	ウィンドウ ハンドルを読み出すコントロールの ID を指定します。

戻り値

CtrlGetDlgItem は、コントロールのウィンドウ ハンドルを返すか、コントロールが存在しない場合、またはエラーが 発生した場合は NULL を返します。**GetExtendedErrInfo** は、追加エラー情報を返す時があります。

追加情報

複数のコントロールを検索するために **CtrlGetDlgItem** を複数回にわたって呼び出す場合、**CmdGetHwndDlg** を呼び出してダイアログのハンドルを取得し、それを各関数の呼び出しに渡すことが推奨されます。単一コントロールのウィンドウ ハンドルを検索する場合、ダイアログ名を指定し、hDialog をヌル文字列 ("") として指定することが推奨されます。

CtrlGetMLEText

CtrlGetMLEText 関数は、カスタム ダイアログの複数行編集フィールド コントロールの内容を読み出します。InstallShield は複数行編集フィールドの各行を listID が識別した文字列リストに配置します。CtrlGetText を呼び出して一行編集フィールドコントロールの内容を読み出します。

構文

```
CtrlGetMLEText ( szDialogName, nControlID, listID );
```

パラメーター

テーブル 99 - CtrlGetMLEText のパラメーター

パラメーター	説明
szDialogName	読み出す内容を含む複数行の編集コントロールのカスタム ダイアログの名前を指定します。
nControlID	複数行編集コントロールのリソース ID を指定します。
listID	nControlID が認識する編集フィールド行の文字列リストを戻します。listID によって識別される文字列リストは、ListCreate への呼び出しによって既に初期化されている必要があります。

戻り値

テーブル 100 - CtrlGetMLEText の戻り値

戻り値	説明
0	CtrlGetMLEText は複数行フィールドの内容を読み出しました。
< 0	CtrlGetMLEText はコントロールの内容を読み出すことができませんでした。

CtrlGetMLEText の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CtrlSetMLEText 関数と CtrlGetMLEText 関数のデモンストレーションを行います。
*
* このスクリプト例では、4 つのチェック ボックスを含むカスタム ダイアログを
* 表示します。このスクリプトは
* ターゲット システム上にすべてのプログラム フォルダーのリストを
* 作成し、CtrlSetMLEText を呼び出し、そのリストをダイアログの複数行編集ボックスに
* 配置します。またダイアログは [ 保存 ] ボタンを含み、エンドユーザーがフォルダー名を
* テキストファイルに保存することができます。
* オプションが選択されたとき、スクリプトは CtrlGetMLEText を呼び出して、
* 複数行編集ボックスからフォルダー名を取得します。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SdAskOptions が表示する
* InstallShield 標準ダイアログです。このダイアログは
* インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
* カスタム ダイアログとしてスクリプトで利用することが
```

```
＊表示されます。
＊
＊メモ : 複数行編集ボックスはリソースでは読み取り専用と定義
＊　　されています。その内容を編集することはできません。
＊
＊　　スクリプトはダイアログ ボックスの [ 次へ ] ボタンのスタティック テキストを
＊　　変更し、[ 戻る ] ボタンを無効にしてダイアログが例の目的を
達成できるようにします。
＊
＊　　GetGroupNameList 関数は、ターゲットシステムが Explorer シェル以外の
＊　　シェルで実行されている場合にエラーを返す場合があります。
＊
＊
¥*---------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID       12007   // カスタム ダイアログの ID
#define RES_PBUT_BACK         12   // [ 次へ ] ボタンの ID
#define RES_PBUT_DONE          9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_SAVE          1   // [ 戻る ] ボタンの ID
#define RES_DIALOG_EDITBOX     301   // 編集ボックスの ID
#define RES_TEXT             711   // 編集ボックス上のテキスト ID

// 複数行編集ボックスの上に表示する説明。
#define DESC_TEXT "[ 保存 ] をクリックして、ディスクファイルへプログラムフォルダー名のリストを保存して下さい。"

// エンドユーザーが [ 保存 ] ボタンをクリックした際、プログラム名が
// 現在のドライブのルートに保存されます。
#define FOLDER_LIST_FILE "¥¥ISExampl.txt"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlGetMLEText(HWND);

function ExFn_CtrlGetMLEText(hMSI)
   STRING szDialogName;
   NUMBER nCmdValue, nResult;
   BOOL   bSave, bDone;
   LIST   listFolders;
   HWND   hwndDlg;
begin

   // このインストールでカスタム ボックスを認識するための名前を指定します。
   szDialogName = "CustomDialog";

   // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
   // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
   // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
   // 4 番目のパラメーターにある ID によって識別されるためです。
   nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

   if (nResult < 0) then
      // エラーを報告し、終了します。
      MessageBox ("ダイアログの定義エラー", SEVERE);
      abort;
   endif;

   // while ループを制御するのに使われるインジケーターを初期化します。
```

```
bDone = FALSE;

// 完了するまでループします。
repeat

   // ダイアログを表示して次のダイアログ イベントを戻します。
   nCmdValue = WaitOnDialog (szDialogName);

   // イベントに応答します。
   switch (nCmdValue)
      case DLG_CLOSE:
         // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
         bDone = TRUE;
      case DLG_ERR:
         MessageBox (" ダイアログが失敗しました ", SEVERE);
         bDone = TRUE;
      case DLG_INIT:
         // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
         // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
         // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
         // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
         hwndDlg = CmdGetHwndDlg (szDialogName);
         SdGeneralInit(szDialogName, hwndDlg, 0, "");

         // ウィンドウのタイトルを設定します。
         SetWindowText (hwndDlg, "View Program Folders");

         // Winsub からの呼び出しを利用して [ 戻る ] ボタンを無効にします。
         _WinSubEnableControl (hwndDlg, RES_PBUT_BACK, 0);

         // ダイアログのスタティック テキスト。
         CtrlSetText (szDialogName, RES_TEXT, DESC_TEXT);
         CtrlSetText (szDialogName, RES_PBUT_SAVE, " 保存 (&S)");
         CtrlSetText (szDialogName, RES_PBUT_DONE, " 完了 (&D)");

         // プログラム フォルダー名を保存する文字列リストを作成します。
         listFolders = ListCreate (STRINGLIST);

         if (listFolders = LIST_NULL) then
            MessageBox (" リストを作成できませんでした。", SEVERE);
            bDone = TRUE;
         else
            // リストへフォルダー名を取得します。
            nResult = GetGroupNameList (listFolders);

            if (nResult = 0) then
               // フォルダー名をダイアログ ボックスの
               // 複数行編集ボックスから取得します。
               nResult = CtrlSetMLEText (szDialogName, RES_DIALOG_EDITBOX,
                                         listFolders);
            elseif (nResult != 0) then
               // GetGroupNameList または CtrlSetMLEText からのハンドル エラー。
               MessageBox (" フォルダー名リストを作成できませんでした。", SEVERE);
               bDone = TRUE;
            endif;

            // ListID 文字列リストを破棄します。
            ListDestroy (listFolders);
         endif;
```

```
        case RES_PBUT_SAVE :
            // プログラムファイル名を保存するインジケーターを初期化します。
            bSave = FALSE;

            if (AskYesNo(" リストを " + FOLDER_LIST_FILE + " として保存しますか ?", YES)) then
                // 既存ファイルを確認します。
                if (Is (FILE_EXISTS, FOLDER_LIST_FILE) = 1) then
                    // エンドユーザーに対し、既存ファイルの上書きを問い合わせます。
                    if (AskYesNo (" 既存の " + FOLDER_LIST_FILE +
                                " を上書きしますか ?", YES)) then
                        bSave = TRUE;
                    endif;
                else
                    bSave = TRUE;
                endif;
            endif;

            if bSave = TRUE then
                // ダイアログからのリストを保存する文字列リストを作成します。
                listFolders = ListCreate (STRINGLIST);

                if (listFolders = LIST_NULL) then
                    MessageBox (" リストを作成できませんでした。", SEVERE);
                else
                    // フォルダー名をダイアログ ボックスの
                    // 複数行編集ボックスから取得します。
                    nResult = CtrlGetMLEText (szDialogName, RES_DIALOG_EDITBOX,
                                              listFolders);
                    // リストをテキスト ファイルへ保存します。
                    ListWriteToFile (listFolders, FOLDER_LIST_FILE);

                    // ListID 文字列リストを破棄します。
                    ListDestroy (listFolders);
                endif;
            endif;
        case RES_PBUT_DONE:
            bDone = TRUE;
    endswitch;

until bDone;

// カスタム ダイアログを閉じます
EndDialog (szDialogName);

// メモリからカスタム ダイアログを削除します。
ReleaseDialog (szDialogName);

end;
```

CtrlGetMultCurSel

CtrlGetMultCurSel 関数は、複数選択リストボックスコントロール (LBS_MULTIPLESEL スタイルのリストボックスコントロール) から現在選択されている行を読み出します。(この関数は拡張選択リストボックスコントロール、つまり LBS_EXTENDEDSEL スタイルのリストボックスコントロールをサポートしません。) 複数選択リストボック

スで選択した各行は listID によって認識された文字列リストに配置されます。単一選択リストボックスコントロールから選択したテキストを読み出すには、CtrlGetCurSel 関数を呼び出します。CtrlGetMultCurSel はカスタム ダイアログでのみ利用できます。

構文

CtrlGetMultCurSel (szDialogName, nControlID, listID);

パラメーター

テーブル 101 - CtrlGetMultCurSel のパラメーター

パラメーター	説明
szDialogName	その内容を読み出すリスト ボックス コントロールを含むカスタム ダイアログの名前を指定します。
nControlID	複数行編集コントロールのリソース ID を指定します。
listID	nControlID が識別したリストボックスの行を戻します。listID によって識別される文字列リストは、ListCreate への呼び出しによって既に初期化されている必要があります。

戻り値

テーブル 102 - CtrlGetMultCurSel の戻り値

戻り値	説明
0	CtrlGetMultCurSel は現在選択した項目を読み出しました。
< 0	CtrlGetMultCurSel は項目を読み出すことができませんでした。

CtrlGetMultCurSel の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CtrlSetMultCurSel  関数と CtrlGetMultCurSel 関数の
```

```
* デモンストレーションを行います。
*
* このスクリプトはターゲットシステム上のすべてのプログラムフォルダーの名前を
* 読み出して、リストへ配置します。ダイアログ ボックスが
* 初期化されるとき、CtrlSetList 関数はこのリストをリスト ボックスで
* 表示するように設定します。次いで、CtrlSetMultCurSel 関数が
* 呼び出されてユーザーが選択したフォルダーを
* ハイライト表示します。
*
* その後このリストは破棄されます。[ 次へ ] ボタンが押されたとき、
* 新しいリストが作成されます。CtrlGetMultCurSel はリストボックスの要素を
* 読み出し、この新しい文字列リストへそれらを
* 割り当てます。このリストは、標準ダイアログで表示されます。
*
* メモ : このスクリプトを適切に実行するため、
*      RES_DIALOG_ID 定数と RES_DIALOG_LISTBOX 定数を _isuser.dll で
作成されたダイアログとリスト ボックスに設定しなくてはなりません。
*
*      ターゲット システムが Explorer シェル以外のシェルで実行
*      されている場合、この例で使われている GetGroupNameList 関数は
*      エラーを返す場合があります。
*
¥*---------------------------------------------------------------*/

// ダイアログ コントロール ID。
#define RES_DIALOG_ID// ダイアログ自身の ID
#define RES_PBUT_NEXT        1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL        9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK         12   // [ 戻る ] ボタンの ID
#define RES_DIALOG_LISTBOX          // リストボックスの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlGetMultCurSel(HWND);

function ExFn_CtrlGetMultCurSel(hMSI)
    STRING szDialogName, szDLL, szTitle, szMsg;
    STRING szText, szDefFolder, svResultFolder;
    NUMBER nCmdValue, nResult, nControlID, nSelectFlag;
    BOOL   bDone;
    LIST listID, listFolders;
    HWND hwndDlg;
begin

    Disable(BACKBUTTON);

    szDialogName = "CtrlSetMultCurSel";
    szDLL        = "";

    // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
    // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
    // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
    // 4 番目のパラメーターにある ID によって識別されるためです。
    nResult = EzDefineDialog (szDialogName, szDLL, "", RES_DIALOG_ID);

    if (nResult < 0) then
        MessageBox ("ダイアログの定義エラー ", SEVERE);
        bDone = TRUE;
```

```
else
    bDone = FALSE;
endif;

// ListID 文字列リストを作成します。
listID = ListCreate (STRINGLIST);

if (listID = LIST_NULL) then
   MessageBox (" リストを作成できませんでした。", SEVERE);
else
   MessageBox ("listID が作成されました。", INFORMATION);
endif;

// リストへプログラムフォルダー名を読み出します。
GetGroupNameList (listID);

// ユーザーからフォルダー名を取得します。
szTitle = "CtrlGetMultCurSel & CtrlSetMultCurSel";
SelectFolder (szTitle, szDefFolder, svResultFolder);

// 完了するまでループします。
while (bDone = FALSE)

   // ダイアログを表示して次のダイアログ イベントを戻します。
   nCmdValue = WaitOnDialog (szDialogName);

   // イベントに応答します。
   switch (nCmdValue)
      case DLG_ERR:
         MessageBox (" ダイアログを表示できませんでした。セットアップはキャンセルされました。",SEVERE);
         abort;
      case DLG_INIT:
         // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
         // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
         // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
         // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
         hwndDlg = CmdGetHwndDlg (szDialogName);
         SdGeneralInit(szDialogName, hwndDlg, 0, "");

         // 次はリストボックスをプログラムフォルダーのリストへ設定します。
         nControlID = RES_DIALOG_LISTBOX;
         CtrlSetList (szDialogName, nControlID, listID);

         szText = svResultFolder;
         nSelectFlag = TRUE;

         // ユーザーが選択したフォルダーをハイライト表示にします。
         if (CtrlSetMultCurSel (szDialogName, nControlID, szText,
                                nSelectFlag) < 0) then
            MessageBox ("CtrlSetMultCurSel が失敗しました。", SEVERE);
         endif;

         // ListID 文字列リストを破棄します。
         ListDestroy(listID);
         MessageBox ("listID が破棄されました。", INFORMATION);
      case DLG_CLOSE:
         // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
         Do (EXIT);
      case RES_PBUT_NEXT:
```

```
            // listFolders 文字列リストを作成します。
            listFolders = ListCreate (STRINGLIST);

            if (listFolders = LIST_NULL) then
               MessageBox (" リストを作成できませんでした。", SEVERE);
            else
               MessageBox ("listFolders が作成されました。", INFORMATION);
            endif;

            // リストボックスでハイライト表示された要素を読み出し、
            // listFolders 文字列リストへ配置します。
            if (CtrlGetMultCurSel (szDialogName, nControlID,
                                   listFolders) < 0) then
               MessageBox ("CtrlGetMultCurSel が失敗しました。", SEVERE);
            else
               MessageBox ("CtrlGetMultCurSel が成功しました。", INFORMATION);
            endif;

            bDone = TRUE;
         case RES_PBUT_BACK:
            bDone = TRUE;
         case RES_PBUT_CANCEL:
            // ユーザーがウィンドウの [ キャンセル ] ボタンをクリックしました。
            Do (EXIT);
      endswitch;

   endwhile;

   szMsg   = " 次はリストボックスでハイライト表示された要素です :";

   // ハイライト表示された要素のリストを表示します。
   SdShowInfoList (szTitle, szMsg, listFolders);

   // メモリから listFolder 文字列リストを削除します。
   ListDestroy (listFolders);
   MessageBox ("listFolders が破棄されました。", INFORMATION);

   // ダイアログを閉じます
   EndDialog (szDialogName);

   // メモリからダイアログを削除します。
   ReleaseDialog (szDialogName);

end;
```

CtrlGetState

CtrlGetState 関数は、カスタム ダイアログからチェック ボックスまたはオプション ボタン コントロールの状態を取得します。

構文

```
CtrlGetState ( szDialogName, nControlID );
```

パラメーター

テーブル 103 ・ CtrlGetState のパラメーター

パラメーター	説明
szDialogName	コントロールを含むダイアログの名前を指定します。
nControlID	状態を読み出すチェック ボックス、またはオプションボタンコントロールのリソース ID を指定します。

戻り値

テーブル 104 ・ CtrlGetState の戻り値

戻り値	説明
BUTTON_CHECKED (-1001)	チェック ボックスまたはオプションオボタンが選択されました。
BUTTON_UNCHECKED (-1002)	チェック ボックスまたはオプションオボタンが選択されていません。
DLG_ERR (-1)	CtrlGetState はコントロールの状態を判断することができませんでした。

CtrlGetState の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CtrlGetState 関数と CtrlSetStat 関数のデモンストレーションを行います。
*
* このスクリプト例では、4 つのチェック ボックスを含むカスタム ダイアログを
* 表示します。スクリプトは CtrlSetState を呼び出して最初の 2 つの
* チェック ボックスをチェック済みに設定します。残りの 2 つはデフォルトで、
* チェック無しです。ユーザーが [ 次へ ] ボタンをクリックしたとき、スクリプトは
* CtrlGetState を呼び出し、各チェック ボックスの状態を
* 読み出します。そしてスクリプトは、どのボックスがチェックされているかを
* メッセージボックスに表示します。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SdAskOptions が表示する
* InstallShield 標準ダイアログです。このダイアログは
```

```
＊インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
＊カスタム ダイアログとしてスクリプトで利用することが
＊表示されます。
＊
¥*---------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID       12020   // カスタム ダイアログの ID
#define RES_PBUT_NEXT         1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL       9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK        12   // [ 戻る ] ボタンの ID
#define  ID_OP1_CHECK       501   // オプション 1 チェック ボックスの ID
#define  ID_OP2_CHECK       502   // オプション 2 チェック ボックスの ID
#define  ID_OP3_CHECK       503   // オプション 3 チェック ボックスの ID
#define  ID_OP4_CHECK       504   // オプション 4 チェック ボックスの ID
#define  ID_STA_DESC        711   // スタティック テキスト説明の ID


// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlGetState(HWND);

function ExFn_CtrlGetState(hMSI)
    STRING  szDialogName, szMsg;
    NUMBER  nResult, nCmdValue, hwndDlg;
    BOOL    bDone;
begin

    // このインストールでカスタム ボックスを認識するための名前を指定します。
    szDialogName = "ExDialog";

    // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
    // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
    // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
    // 4 番目のパラメーターにある ID によって識別されるためです。
    nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

    if (nResult < 0) then
        // エラーを報告し、終了します。
        MessageBox ("ダイアログの定義エラー", SEVERE);
        abort;
    endif;

    // ループを制御するのに使われるインジケーターを初期化します。
    bDone = FALSE;

    repeat

        // ダイアログを表示して次のダイアログ イベントを戻します。
        nCmdValue = WaitOnDialog (szDialogName);

        // イベントに応答します。
        switch (nCmdValue)
            case DLG_CLOSE:
                // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
                Do (EXIT);
            case DLG_ERR:
                MessageBox ("ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
```

```
            abort;
        case DLG_INIT:
            // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
            // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
            // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
            // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
            hwndDlg = CmdGetHwndDlg (szDialogName);
            SdGeneralInit(szDialogName, hwndDlg, 0, "");

            // ウィンドウのタイトルを設定します。
            SetWindowText (hwndDlg, " オプションを選択 ");

            // チェック ボックスの上に表示されるスタティックテキスト説明を設定します。
            CtrlSetText (szDialogName, ID_STA_DESC,
                    " オプションを選択および / またはクリアします。そして [ 次へ ] をクリックします。");

            // デフォルトでオプションはクリアされているので、オプション 1 と 2 を選択します。
            if (CtrlSetState (szDialogName, ID_OP1_CHECK, BUTTON_CHECKED) < 0) then
                MessageBox ("CtrlSetState の最初の呼び出しに失敗しました。", SEVERE);
                bDone = TRUE;
            elseif (CtrlSetState(szDialogName, ID_OP2_CHECK, BUTTON_CHECKED) <  0) then
                MessageBox ("CtrlSetState への 2 回目の呼び出しに失敗しました。", SEVERE);
                bDone = TRUE;
            endif;

        case RES_PBUT_NEXT:
            bDone = TRUE;
        case RES_PBUT_CANCEL:
            // ユーザーが [ キャンセル ] ボタンをクリックしました。
            Do (EXIT);
        case RES_PBUT_BACK:
            bDone = TRUE;
    endswitch;

until bDone;

// エンドユーザーが [ 次へ ] ボタンをクリックするとメッセージをビルドします。
if (nCmdValue = RES_PBUT_NEXT) then
    // エンドユーザーへ表示するメッセージのビルドを開始します。
    szMsg = " 次のアイテムを選択しました :¥n¥n";

    // 最初のオプションが選択された場合、メッセージに行を追加します。
    if (CtrlGetState (szDialogName, ID_OP1_CHECK) = BUTTON_CHECKED) then
        szMsg = szMsg + " オプション 1¥n";
    endif;

    // 2 番目のオプションが選択された場合、メッセージに行を追加します。
    if (CtrlGetState (szDialogName, ID_OP2_CHECK) = BUTTON_CHECKED) then
        szMsg = szMsg + " オプション 2¥n";
    endif;

    // 3 番目のオプションが選択された場合、メッセージに行を追加します。
    if (CtrlGetState (szDialogName, ID_OP3_CHECK) = BUTTON_CHECKED) then
        szMsg = szMsg + " オプション 3¥n";
    endif;

    // 4 番目のオプションが選択された場合、メッセージに行を追加します。
    if (CtrlGetState (szDialogName, ID_OP4_CHECK) = BUTTON_CHECKED) then
        szMsg = szMsg + " オプション 4¥n";
```

```
        endif;
    endif;

    // カスタム ダイアログを閉じます
    EndDialog (szDialogName);

    // メモリからカスタム ダイアログを削除します。
    ReleaseDialog (szDialogName);

    // [ 次へ ] ボタンを使ってダイアログが閉じられたときにメッセージを表示します。
    if (nCmdValue = RES_PBUT_NEXT) then
        MessageBox (szMsg, INFORMATION);
    endif;

end;
```

CtrlGetSubCommand

CtrlGetSubCommand 関数は、カスタム ダイアログのコントロールで実行されたアクションを読み出します。たとえば、CtrlGetSubCommand はユーザーがリストボックスまたはコンボボックスコントロールをワンクリックしたか、ダブルクリックしたかを伝えます。また、編集フィールドにいつ変更が加えられたのかも示します。

上級開発者向けに、追加情報を扱う CmdGetHwndDlg 関数があります。

構文

```
CtrlGetSubCommand (szDialogName);
```

パラメーター

テーブル 105 ・ CtrlGetSubCommand のパラメーター

パラメーター	説明
szDialogName	カスタム ダイアログの名前を指定します。

戻り値

テーブル 106 ・ CtrlGetSubCommand の戻り値

戻り値	説明
EDITBOX_CHANGE (-1007)	編集ボックスの内容が変更されました。
LISTBOX_ENTER (-1008)	ユーザーがリストボックスのアイテムをダブルクリックしました。
LISTBOX_SELECT (-1009)	ユーザーがリストボックスのアイテムをシングルクリックしました。

CtrlGetSubCommand の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CtrlGetSubCommand 関数のデモンストレーションを行います。
*
* このスクリプト例では、カスタムダイアログのリストボックスに
* プログラム グループのリストを表示します。次いで、編集ボックスとリストボックスからの
* イベントへ次のように応答します :
*
* リスト ボックスをシングル クリック : 選択されたアイテムが編集ボックスへ配置されます。
*
* リストボックスをダブルクリック : ダブルクリックされたアイテムは後で表示するために
保存され、ダイアログが閉じます。
*
* 編集ボックスの値の変更 : デフォルトのシステム サウンドが再生されます。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SdAskOptions が表示する
* InstallShield 標準ダイアログです。このダイアログは
* インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
* カスタム ダイアログとしてスクリプトで利用することが
* 表示されます。
*
```

```
¥*---------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID12008// カスタム ダイアログの ID
#define RES_PBUT_NEXT        1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL        9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK         12   // [ 戻る ] ボタンの ID
#define RES_DIALOG_EDITBOX    301   // 編集ボックスの ID
#define RES_DIALOG_LISTBOX    401   // リストボックスの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlGetSubCommand(HWND);

function ExFn_CtrlGetSubCommand(hMSI)
    STRING szDialogName, svSelection;
    NUMBER nResult, nCmdValue, nSubCommand;
    BOOL   bDone, bSelected;
    HWND   hwndDlg;
begin

    // このインストールでカスタム ボックスを認識するための名前を指定します。
    szDialogName = "CustomDialog";

    // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
    // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
    // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
    // 4 番目のパラメーターにある ID によって識別されるためです。
    nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

    if (nResult < 0) then
        // エラーを報告し、終了します。
        MessageBox (" ダイアログの定義エラー ", SEVERE);
        abort;
    endif;

    // while ループを制御するのに使われるインジケーターを初期化し、
    // アイテムが選択されたかどうかを示します。
    bDone = FALSE;
    bSelected = FALSE;

    // 完了するまでループします。
    repeat

        // ダイアログを表示して次のダイアログ イベントを戻します。
        nCmdValue = WaitOnDialog (szDialogName);

        // イベントに応答します。
        switch (nCmdValue)
            case DLG_CLOSE:
                // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
                Do (EXIT);
            case DLG_ERR:
                MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
                abort;
            case DLG_INIT:
                // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
                // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
```

```
            // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
            // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
            hwndDlg = CmdGetHwndDlg (szDialogName);
            SdGeneralInit(szDialogName, hwndDlg, 0, "");

            // ダイアログのリスト ボックスへフォルダー リストを配置します。
            if (CtrlPGroups (szDialogName, RES_DIALOG_LISTBOX) < 0) then
               MessageBox ("CtrlPGroups が失敗しました。", SEVERE);
            endif;
         case RES_DIALOG_LISTBOX:
            // イベントを取得します。
            nSubCommand = CtrlGetSubCommand (szDialogName);

            if (nSubCommand = LISTBOX_SELECT) then
               // シングルクリック : 選択されたアイテムを編集ボックスに配置します。
               CtrlGetCurSel (szDialogName, RES_DIALOG_LISTBOX, svSelection );
               CtrlSetText (szDialogName, RES_DIALOG_EDITBOX, svSelection );
            elseif (nSubCommand = LISTBOX_ENTER) then
               // ダブル クリック : 選択したアイテムを取得し、
               // インジケーターを終了します。
               CtrlGetCurSel (szDialogName, RES_DIALOG_LISTBOX, svSelection );
               bSelected = TRUE;
               bDone = TRUE;
            endif;
         case RES_DIALOG_EDITBOX:
            // イベントを取得します。
            nSubCommand = CtrlGetSubCommand (szDialogName);

            // 編集ボックスの内容が変更された場合、デフォルトのシステムサウンドを再生します。
            if (nSubCommand = EDITBOX_CHANGE) then
               MessageBeep (0);
            endif;
         case RES_PBUT_CANCEL:
            // ユーザーが [ キャンセル ] ボタンをクリックしました。
            Do (EXIT);
         case RES_PBUT_NEXT:
            // 編集ボックスから現在の選択部分を取得します。
            CtrlGetText (szDialogName, RES_DIALOG_EDITBOX, svSelection );
            bSelected = TRUE;
            bDone = TRUE;
         case RES_PBUT_BACK:
            bDone = TRUE;
      endswitch;

   until bDone;

   // ダイアログを閉じます
   EndDialog (szDialogName);

   // メモリからダイアログを解放します。
   ReleaseDialog (szDialogName);

   if bSelected then
      // 選択されたフォルダー名を表示します。
      MessageBox (svSelection +" が選択されました。",INFORMATION);
   endif;

end;
```

CtrlGetText

CtrlGetText 関数は、編集フィールド、スタティックテキストフィールド、またはカスタムダイアログのボタンコントロールからテキストを読み出します。複数行編集フィールドコントロールからテキストを読み出すには、CtrlGetMLEText を呼び出してください。

構文

CtrlGetText (szDialogName, nControlID, svText);

パラメーター

テーブル 107 - CtrlGetText のパラメーター

パラメーター	説明
szDialogName	そのテキストを読み出すフィールドまたはコントロールを含むダイアログの名前を指定します。
nControlID	編集フィールド、スタティックテキストフィールド、またはボタンコントロールのリソース ID を指定します。
svText	nControlID が識別したコントロールまたはフィールドからテキストを戻します。

戻り値

テーブル 108 - CtrlGetText の戻り値

戻り値	説明
0	CtrlGetText が指定したコントロールの内容を読み出しました。
< 0	CtrlGetText は内容を読み出せませんでした。

CtrlGetText の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CtrlSetText 関数、CtrlGetText 関数、および CtrlSelectText 関数の
```

```
* デモンストレーションを行います。
*
* このスクリプト例では、ユーザーの名前と会社名を取得する 2 つの編集ボックスを
* 持つカスタム ダイアログを表示します。その後
* スクリプトは CtrlSetText を呼び出して初期値を編集ボックスに配置し、
* CtrlSelectText を呼び出して最初の編集ボックスの
* 内容を選択します。ユーザーが [ 次へ ] ボタンを
* クリックすると、スクリプトは CtrlGetText を呼び出して
* 編集ボックスの内容を読み出して、カスタムダイアログボックスを
* 閉じた後にメッセージ ボックスで表示できるようにします。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SdAskOptions が表示する
* InstallShield 標準ダイアログです。このダイアログは
* インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
* カスタム ダイアログとしてスクリプトで利用することが
* 表示されます。
*
¥*---------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID12001// カスタム ダイアログの ID
#define RES_PBUT_NEXT        1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL      9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK        12   //[ 戻る ] ボタンの ID
#define RES_EDITNAME        301   // 編集ボックスの ID
#define RES_EDITCOMPANY     302   // 編集ボックスの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlGetText(HWND);

function ExFn_CtrlGetText(hMSI)
    STRING szDialogName, svName, svCompany;
    NUMBER nResult, nCmdValue;
    BOOL   bDone;
    HWND   hwndDlg;
begin

    // このインストールでカスタム ボックスを認識するための名前を指定します。
    szDialogName = "CustomDialog";

    // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
    // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
    // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
    // 4 番目のパラメーターにある ID によって識別されるためです。
    nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

    if (nResult < 0) then
        // エラーを報告し、終了します。
        MessageBox ("ダイアログの定義エラー ", SEVERE);
        abort;
    endif;

    // ループを制御するのに使われるインジケーターを初期化します。
    bDone = FALSE;

    // 完了するまでループします。
```

```
    repeat

        // ダイアログを表示して次のダイアログ イベントを戻します。
        nCmdValue = WaitOnDialog (szDialogName);

        // イベントに応答します。
        switch (nCmdValue)
            case DLG_CLOSE:
                // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
                Do (EXIT);
            case DLG_ERR:
                MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
                abort;
            case DLG_INIT:
                // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
                // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
                // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
                // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
                hwndDlg = CmdGetHwndDlg (szDialogName);
                SdGeneralInit(szDialogName, hwndDlg, 0, "");

                // 初期値を編集ボックスに配置します。
                CtrlSetText (szDialogName, RES_EDITNAME, " 名前 ");
                CtrlSetText (szDialogName, RES_EDITCOMPANY, " 会社名 ");

                // [ 名前 ] 編集ボックスを選択します。
                CtrlSelectText (szDialogName, RES_EDITNAME);
            case RES_PBUT_NEXT:
                // 編集ボックスの内容を取得します。
                CtrlGetText (szDialogName, RES_EDITNAME, svName);
                CtrlGetText (szDialogName, RES_EDITCOMPANY, svCompany);

                // 両方の編集ボックスにデータが入力されていることを確認します。
                if (StrLength(svName) = 0) || (StrLength(svCompany) = 0) then
                    MessageBox (" 両方のフィールドに入力する必要があります。", INFORMATION);
                else
                    bDone = TRUE;
                endif;
            case RES_PBUT_CANCEL:
                // ユーザーが [ キャンセル ] ボタンをクリックしました。
                Do (EXIT);
            case RES_PBUT_BACK:
                bDone = TRUE;
        endswitch;

    until bDone;

    // ダイアログを閉じます
    EndDialog (szDialogName);

    // メモリからダイアログを削除します。
    ReleaseDialog (szDialogName);

    // ダイアログを [ 次へ ] ボタンで閉じた場合、名前と会社名を表示します。
    if nCmdValue = RES_PBUT_NEXT then
        MessageBox (svName + "¥n" + svCompany, INFORMATION);
    endif;

end;
```

CtrlGetUrlForLinkClicked

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

CtrlGetUrlForLinkClicked 関数は、エンド ユーザーがクリックしたリンクへの URL を取得します。

ヒント・InstallScript プロジェクトおよび InstallScript MSI プロジェクトでダイアログにリンクを追加する方法については、「ダイアログで HTML コントロールを使用する」を参照してください。

構文

CtrlGetUrlForLinkClicked (byval string szDialogName, byval number nControlID, byref string svText);

パラメーター

テーブル 109・CtrlGetUrlForLinkClicked のパラメーター

パラメーター	説明
szDialogName	HTML コントロールを含むダイアログの名前を指定します。
nControlID	HTML コントロールの コントロール ID を指定します。この ID は、HTML コントロールに変換されたスタティック コントロールの ID と同じです。
svText	リンク URL テキストを返す文字列変数を指定します。

戻り値

テーブル 110・CtrlGetUrlForLinkClicked の戻り値

戻り値	説明
0	**CtrlGetUrlForLinkClicked** は、エンド ユーザーがクリックしたリンクを判別できました。
ISERR_GEN_FAILURE	**CtrlGetUrlForLinkClicked** は、エンド ユーザーがクリックしたリンクを判別できませんでした。この関数は、指定したコントロール ID が HTML コントロールの ID ではない場合にこの値を返します。

CtrlGetUrlForLinkClicked の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

```
//---------------------------------------------------------------------------
//
// InstallShield スクリプトの例
//
HTML コントロールを CtrlGetUrlForLinkClicked および CtrlSetText 関数と
共に使用する方法をデモンストレーションします
//
このサンプル スクリプトを使用するには、以下の手順を行います :
//  1. カスタム ダイアログをプロジェクトに追加します。
//  2. スタティック テキスト コントロールをダイアログに追加します。
//
//---------------------------------------------------------------------------


#define MY_HYPERLINK1 1401

function MyCustomDialog(szTitle, szMsg)
    STRING  szDlg, szTemp, szUrl;
    NUMBER  nId, nMessage, nTemp, nSdDialog;
    HWND    hwndDlg;
    BOOL    bDone;

begin

    // このインストールでカスタム ボックスを認識するための名前を指定します。
    szDlg = "CustomName";

    while (!bDone)

        nId = WaitOnDialog( szDlg );

        switch(nId)
        case DLG_INIT:
            // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
            // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
            // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
            // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
            hwndDlg = CmdGetHwndDlg(szDlg);
            SdGeneralInit(szDlg, hwndDlg, 0, "");

            // リスト フィールドに対応する情報を配置します
            if( szMsg != "" ) then
                SdSetStatic(szDlg, SD_STA_MSG, szMsg);
            endif;

            SdSetDlgTitle(szDlg, hwndDlg, szTitle);

            CtrlSetText(szDlg, MY_HYPERLINK1,
                "[html]<style type=¥"text/css¥">html,body {padding:0; margin:0;} *
                {font-size: 8pt; font-family: ¥"MS Sans Serif¥";}</style>
                <a href=¥"http://www.MyWebSite.com¥">
                Visit my Web site</a>");

        case MY_HYPERLINK1:
            CtrlGetUrlForLinkClicked(szDlg, MY_HYPERLINK1, szUrl);
            MessageBox(" ハイパーリンクがクリックされました : " + szUrl, 0);
```

```
            // 必要に応じて、追加 case ステートメントを追加します。

        デフォルト :
            // 標準の処理を確認します
            if (SdIsStdButton( nId ) && SdDoStdButton( nId )) then
                bDone = TRUE;
            endif;
        endswitch;

    endwhile;
end;
```

参照
InstallScript プロジェクトおよび InstallScript MSI プロジェクトで新しいカスタム ダイアログを作成する
ダイアログで HTML コントロールを使用する

CtrlPGroups

CtrlPGroups 関数は、既存のプログラムフォルダーをリストボックスまたはコンボボックスコントロールに配置します。この関数はカスタム ダイアログでのみ利用できます。

構文

```
CtrlPGroups ( szDialogName, nControlID );
```

パラメーター

テーブル 111 ・ CtrlPGroups のパラメーター

パラメーター	説明
szDialogName	使用するコントロールを含むカスタム ダイアログの名前を指定します。
nControlID	リストボックスまたはコンボボックスコントロールのリソース ID を指定します。

戻り値

テーブル 112 ・ CtrlPGroups の戻り値

戻り値	説明
0	CtrlPGroups は、コントロールにある指定されたプログラムフォルダーのリストを配置しました。
< 0	CtrlPGroups は、コントロールにある指定されたプログラムフォルダーのリストを配置しました。

CtrlPGroups の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CtrlPGroups 関数のデモンストレーションを行います。
*
* このスクリプト例では、編集ボックスとリスト ボックスを持つ
* カスタム ダイアログを表示します。ダイアログが初期化された後、
* スクリプトは CtrlPGroups を呼び出してプログラムフォルダー名のリストを作成し、
* それをダイアログのリストボックスへ配置します。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SdAskOptions が表示する
* InstallShield 標準ダイアログです。このダイアログは
* インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
* カスタム ダイアログとしてスクリプトで利用することが
* 表示されます。
*
¥*---------------------------------------------------------------------*/
```

```
// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID12008// カスタム ダイアログの ID
#define RES_PBUT_NEXT        1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL      9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK        12  // [ 戻る ] ボタンの ID
#define RES_DIALOG_EDITBOX   301  // 編集ボックスの ID
#define RES_DIALOG_LISTBOX   401  // リストボックスの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlPGroups(HWND);

function ExFn_CtrlPGroups(hMSI)
   STRING szDialogName, svSelection;
   NUMBER nResult, nCmdValue, nControlID;
   BOOL   bDone;
   HWND   hwndDlg;
begin

   // このインストールでカスタム ボックスを認識するための名前を指定します。
   szDialogName = "CustomDialog";

   // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
   // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
   // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
   // 4 番目のパラメーターにある ID によって識別されるためです。
   nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

   if (nResult < 0) then
      // エラーを報告し、終了します。
      MessageBox ("ダイアログの定義エラー", SEVERE);
      abort;
   endif;

   // while ループを制御するのに使われるインジケーターを初期化します。
   bDone = FALSE;

   // 完了するまでループします。
   repeat

      // ダイアログを表示して次のダイアログ イベントを戻します。
      nCmdValue = WaitOnDialog (szDialogName);

      // イベントに応答します。
      switch (nCmdValue)
         case DLG_CLOSE:
            // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
            Do (EXIT);
         case DLG_ERR:
            MessageBox ("ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
            abort;
         case DLG_INIT:
            // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
            // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
            // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
            // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
            hwndDlg = CmdGetHwndDlg (szDialogName);
```

```
            SdGeneralInit(szDialogName, hwndDlg, 0, "");

            // ダイアログのリスト ボックスへフォルダー リストを配置します。
            if (CtrlPGroups (szDialogName, RES_DIALOG_LISTBOX) < 0) then
                MessageBox ("CtrlPGroups が失敗しました。", SEVERE);
                bDone = TRUE;
            endif;
        case RES_DIALOG_LISTBOX:
            // 現在のリストボックスの選択部分を取得します。
            CtrlGetCurSel (szDialogName, RES_DIALOG_LISTBOX, svSelection);

            // 現在の選択部分を編集ボックスに配置します。
            CtrlSetText (szDialogName, RES_DIALOG_EDITBOX, svSelection);
        case RES_PBUT_CANCEL:
            // ユーザーが [ キャンセル ] ボタンをクリックしました。
            Do (EXIT);
        case RES_PBUT_NEXT:
            // 編集ボックスから現在の値を取得します。
            CtrlGetText (szDialogName, RES_DIALOG_EDITBOX, svSelection);

            // 編集ボックスに既存プログラムフォルダーの名前が
            // 含まれていることを確認します。
            if CtrlSetCurSel (szDialogName, RES_DIALOG_LISTBOX, svSelection) =
0 then
                bDone = TRUE;
            else
                MessageBox ("Program folder does not exist.", WARNING);
            endif;
        case RES_PBUT_BACK:
            bDone = TRUE;
    endswitch;

    until bDone;

    // ダイアログを閉じます
    EndDialog (szDialogName);

    // メモリからダイアログを解放します。
    ReleaseDialog (szDialogName);

    // [ 完了 ] ボタンを使って編集ボックスを閉じた場合、
    // 選択した項目を表示します。
    if (nCmdValue = RES_PBUT_NEXT) then
        MessageBox (svSelection + " が選択されました。", INFORMATION);
    endif;

end;
```

CtrlSelectText

CtrlSelectText 関数は編集フィールドまたはコンボボックスの編集フィールドですべてのテキストを選択します。コントロールが複数行編集フィールドの場合、この関数はすべての行の全テキストを選択します。この関数はカスタム ダイアログでのみ利用できます。

構文

```
CtrlSelectText ( szDialogName, nControlID );
```

パラメーター

テーブル 113 ・ CtrlSelectText のパラメーター

パラメーター	説明
szDialogName	選択する編集フィールドを含む有効なダイアログの名前を指定します。
nControlID	選択する編集フィールドまたはコンボボックスコントロールのリソース ID を指定します。

戻り値

テーブル 114 ・ CtrlSelectText の戻り値

戻り値	説明
0	CtrlSelectText がフィールド内のすべてのテキストを選択しました。
< 0	CtrlSelectText はテキストを選択することができませんでした。

CtrlSelectText の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CtrlSetText 関数、CtrlGetText 関数、および CtrlSelectText 関数の
* デモンストレーションを行います。
*
* このスクリプト例では、ユーザーの名前と会社名を取得する 2 つの編集ボックスを
* 持つカスタム ダイアログを表示します。その後
* スクリプトは CtrlSetText を呼び出して初期値を編集ボックスに配置し、
* CtrlSelectText を呼び出して最初の編集ボックスの
* 内容を選択します。ユーザーが [ 次へ ] ボタンを
* クリックすると、スクリプトは CtrlGetText を呼び出して
* 編集ボックスの内容を読み出して、カスタムダイアログボックスを
* 閉じた後にメッセージ ボックスで表示できるようにします。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SdAskOptions が表示する
* InstallShield 標準ダイアログです。このダイアログは
* インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
```

```
＊カスタム ダイアログとしてスクリプトで利用することが
＊表示されます。
＊
¥*-------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID12001// カスタム ダイアログの ID
#define RES_PBUT_NEXT        1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL      9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK       12   //[ 戻る ] ボタンの ID
#define RES_EDITNAME       301   // 編集ボックスの ID
#define RES_EDITCOMPANY    302   // 編集ボックスの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlSelectText(HWND);

function ExFn_CtrlSelectText(hMSI)
   STRING szDialogName, svName, svCompany;
   NUMBER nResult, nCmdValue;
   BOOL   bDone;
   HWND   hwndDlg;
begin

   // このインストールでカスタム ボックスを認識するための名前を指定します。
   szDialogName = "CustomDialog";

   // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
   // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
   // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
   // 4 番目のパラメーターにある ID によって識別されるためです。
   nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

   if (nResult < 0) then
      // エラーを報告し、終了します。
      MessageBox ("ダイアログの定義エラー", SEVERE);
      abort;
   endif;

   // ループを制御するのに使われるインジケーターを初期化します。
   bDone = FALSE;

   // 完了するまでループします。
   repeat

      // ダイアログを表示して次のダイアログ イベントを戻します。
      nCmdValue = WaitOnDialog (szDialogName);

      // イベントに応答します。
      switch (nCmdValue)
         case DLG_CLOSE:
            // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
            Do (EXIT);
         case DLG_ERR:
            MessageBox ("ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
            abort;
         case DLG_INIT:
            // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
```

```
            // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
            // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
            // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
            hwndDlg = CmdGetHwndDlg (szDialogName);
            SdGeneralInit(szDialogName, hwndDlg, 0, "");

            // 初期値を編集ボックスに配置します。
            CtrlSetText (szDialogName, RES_EDITNAME, " 名前 ");
            CtrlSetText (szDialogName, RES_EDITCOMPANY, " 会社名 ");

            // [ 名前 ] 編集ボックスを選択します。
            CtrlSelectText (szDialogName, RES_EDITNAME);
        case RES_PBUT_NEXT:
            // 編集ボックスの内容を取得します。
            CtrlGetText (szDialogName, RES_EDITNAME, svName);
            CtrlGetText (szDialogName, RES_EDITCOMPANY, svCompany);

            // 両方の編集ボックスにデータが入力されていることを確認します。
            if (StrLength(svName) = 0) || (StrLength(svCompany) = 0) then
                MessageBox (" 両方のフィールドに入力する必要があります。", INFORMATION);
            else
                bDone = TRUE;
            endif;
        case RES_PBUT_CANCEL:
            // ユーザーが [ キャンセル ] ボタンをクリックしました。
            Do (EXIT);
        case RES_PBUT_BACK:
            bDone = TRUE;
    endswitch;

until bDone;

// ダイアログを閉じます
EndDialog (szDialogName);

// メモリからダイアログを削除します。
ReleaseDialog (szDialogName);

// ダイアログを [ 次へ ] ボタンで閉じた場合、名前と会社名を表示します。
if nCmdValue = RES_PBUT_NEXT then
    MessageBox (svName + "¥n" + svCompany, INFORMATION);
endif;

end;
```

CtrlSetCurSel

CtrlSetCurSel 関数は指定したリスト、または文字列のコンボボックスコントロールを検索します。文字列が見つかった場合、CtrlSetCurSel は項目を選択 (ハイライト表示) します。複数選択リストボックスとコンボボックスコントロールには CtrlSetMultCurSel を呼び出します。CtrlSetCurSel 関数はカスタム ダイアログでのみ利用できます。

構文

```
CtrlSetCurSel ( szDialogName, nControlID, szText );
```

パラメーター

テーブル 115 ・ CtrlSetCurSel のパラメーター

パラメーター	説明
szDialogName	検索するコントロールを含む有効なカスタム ダイアログの名前を指定します。
nControlID	検索文字列を含むコントロールの リソース ID を指定します。
szText	検索文字列を指定します。文字列が検出された場合、選択 (ハイライト表示) します。

戻り値

テーブル 116 ・ CtrlSetCurSel の戻り値

戻り値	説明
0	CtrlSetCurSel は指定した文字列を検出し、選択しました。
< 0	CtrlSetCurSel は指定された文字列を検出できず、選択できませんでした。

CtrlSetCurSel の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CtrlGetCurSel 関数と CtrlSetCurSel 関数のデモンストレーションを行います。
*
* このスクリプト例では、編集ボックスとリスト ボックスを持つ
* カスタム ダイアログを表示します。ダイアログが初期化された後、
* スクリプトは Windows ディスクのルートにある
* フォルダー名をダイアログのリスト ボックスに配置します。そこで
* そして CtrlSetCurSel を呼び出して "Windows" を選択フォルダーとします。
*
* ユーザーがリストボックスからフォルダー名を選択するたびに
* スクリプトは CtrlGetCurSel を呼び出して選択したアイテムを取得し、
* 編集ボックスに配置できるようにします。ダイアログが
* [ 完了 ] ボタンを使って閉じられた場合、現在選択しているアイテムが
* メッセージボックスに表示されます。
*
```

```
＊このスクリプトで利用される [ カスタム ] ダイアログは、
＊実際、ビルトイン関数 SdAskOptions が表示する
＊InstallShield 標準ダイアログです。このダイアログは
＊インストールで既に圧縮済みのファイル _isres.dll に保存されており、
＊カスタム ダイアログとしてスクリプトで利用することが
＊表示されます。
＊
¥*---------------------------------------------------------------*/

// リストボックスで選択済みとなるフォルダー。
#define PRESELECTED_FOLDER "windows"

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID       12008   // カスタム ダイアログの ID
#define RES_PBUT_NEXT        1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL         9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK          12   // [ 戻る ] ボタンの ID
#define RES_DIALOG_EDITBOX    301  // 編集ボックスの ID
#define RES_DIALOG_LISTBOX    401  // リストボックスの ID
#define RES_STA_DESC710// ダイアログ上部のテキスト ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlSetCurSel(HWND);

function ExFn_CtrlSetCurSel(hMSI)
    STRING  szDialogName, svSelection, szDesc;
    NUMBER  nResult, nCmdValue;
    BOOL    bDone;
    HWND    hwndDlg;
begin

    // このインストールでカスタム ボックスを認識するための名前を指定します。
    szDialogName = "CustomDialog";

    // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
    //  _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
    // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
    // 4 番目のパラメーターにある ID によって識別されるためです。
    nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

    if (nResult < 0) then
        // エラーを報告し、終了します。
        MessageBox ("ダイアログの定義エラー ", SEVERE);
        abort;
    endif;

    // 完了するまでループします。
    repeat

        // ダイアログを表示して次のダイアログ イベントを戻します。
        nCmdValue = WaitOnDialog (szDialogName);

        // イベントに応答します。
        switch (nCmdValue)
            case DLG_CLOSE:
                // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
                Do (EXIT);
```

```
        case DLG_ERR:
            MessageBox (″ ダイアログを表示できませんでした。セットアップがキャンセルされました。″, SEVERE);
            abort;
        case DLG_INIT:
            // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
            // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
            // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
            // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
            hwndDlg = CmdGetHwndDlg (szDialogName);
            SdGeneralInit(szDialogName, hwndDlg, 0, ″″);

            // ウィンドウのタイトルを設定します。
            SetWindowText (hwndDlg, ″Select Folder″);

            // ダイアログの上部に表示されるメッセージを設定します。
            szDesc = ″ ドライブのルートから既存フォルダーを指定します ″
                    + WINSYSDISK + ″¥n そして [ 次へ ] を押して続行します。″;
            CtrlSetText (szDialogName,  RES_STA_DESC, szDesc);

            // ダイアログのリスト ボックスへ Windows ドライブのルートにある
            // すべてのフォルダーの名前を入力します。
            CtrlDir (szDialogName, RES_DIALOG_LISTBOX,
                    WINSYSDISK + ″¥¥*.*″, DLG_DIR_DIRECTORY);

            // 選択済みフォルダーを選びます。
            CtrlSetCurSel (szDialogName, RES_DIALOG_LISTBOX,
                        PRESELECTED_FOLDER);

            // 選択済みフォルダーの名前を編集ボックスへ配置します。
            CtrlSetText (szDialogName, RES_DIALOG_EDITBOX, PRESELECTED_FOLDER);
        case RES_DIALOG_LISTBOX:
            // 現在のリストボックスの選択部分を取得します。
            CtrlGetCurSel (szDialogName, RES_DIALOG_LISTBOX, svSelection);

            // 括弧を削除します。
            StrSub (svSelection, svSelection, 1, StrLength(svSelection) - 2);

            // 現在の選択部分を編集ボックスに配置します。
            CtrlSetText (szDialogName, RES_DIALOG_EDITBOX, svSelection);
        case RES_PBUT_BACK:
            bDone = TRUE;
        case RES_PBUT_NEXT:
            // 編集ボックスから選択部分を取得します。
            CtrlGetText (szDialogName, RES_DIALOG_EDITBOX, svSelection);

            // 編集ボックスに Windows ディスクのルートに存在する
            // フォルダーの名前が含まれていることを確認します。
            if Is (PATH_EXISTS, WINSYSDISK + ″¥¥″+ svSelection) then
                bDone = TRUE;
            else
                MessageBox (″ フォルダーが存在しません。″, WARNING);
            endif;
        case RES_PBUT_CANCEL:
            // ユーザーが [ キャンセル ] ボタンをクリックしました。
            Do (EXIT);
    endswitch;

until bDone;
```

```
    // カスタム ダイアログを閉じます
    EndDialog (szDialogName);

    // メモリからカスタム ダイアログを削除します。
    ReleaseDialog (szDialogName);

    // [ 完了 ] ボタンを使って編集ボックスを閉じた場合、
    // 選択した項目を表示します。
    if (nCmdValue = RES_PBUT_NEXT) then
        MessageBox (svSelection + " が選択されました。", INFORMATION);
    endif;

end;
```

CtrlSetFont

CtrlSetFont 関数 はカスタム ダイアログ内のコントロールのフォントを指定します。この関数は、ダイアログ メッセージ処理ループの DLG_INIT ルーチン内から呼び出します。

構文

```
CtrlSetFont ( szDialogName, hFont, nControlID );
```

パラメーター

テーブル 117 ・ CtrlSetFont のパラメーター

パラメーター	説明
szDialogName	有効なダイアログの名前を指定します。
hFont	GetFont への呼び出しで作成されたフォントのハンドルを指定します。
nControlID	設定するコントロールのリソース ID を指定します。ダイアログのすべてのコントロールについてフォントを設定するには、このパラメーターに定義済み定数 ALLCONTROLS を渡します。

戻り値

テーブル 118 ・ CtrlSetFont の戻り値

戻り値	説明
0	CtrlSetFont は、ダイアログに要求されたフォントを設定しました。
< 0	CtrlSetFont は、要求したダイアログにフォントを設定できませんでした。

CtrlSetFont の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* GetFont 関数と CtrlSetFon 関数のデモンストレーションを行います。
*
* このスクリプト例では、GetFont を呼び出して 4 つのフォントのハンドルを
* 読み出します。これらのハンドルは CtrlSetFont へ渡され、
* カスタムダイアログボックスの静的テキストフィールドのフォントが設定
* されます。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SetupType が表示する
* InstallShield ダイアログです。このダイアログは
* インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
```

```
＊カスタム ダイアログとしてスクリプトで利用することが
＊表示されます。
＊
¥*-------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID       10203   // カスタム ダイアログの ID
#define RES_PBUT_NEXT         1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL       9   // [ キャンセル ] ボタンの ID
#define RES_TEXT_1          202   // 最初のスタティック テキストボックスの ID

#define RES_TEXT_2          210   // 2 番目のスタティックテキストボックスの ID
#define RES_TEXT_3          220   // 3 番目のスタティックテキストボックスの ID
#define RES_TEXT_4          230   // 4 番目のスタティック テキストボックスの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlSetFont(HWND);

function ExFn_CtrlSetFont(hMSI)
   STRING szDialogName;
   NUMBER nResult, nCmdValue;
   HWND   hFont1, hFont2, hFont3, hFont4, hwndDlg;
   BOOL   bDone;
begin

   // カスタム ダイアログ ボックスが表示するスタティック テキストに利用する
   // フォントのハンドルを取得します。
   hFont1 = GetFont("Arial", 14, STYLE_BOLD);
   hFont2 = GetFont("Times New Roman", 11, STYLE_ITALIC);
   hFont3 = GetFont("Arial", 10, STYLE_BOLD);
   hFont4 = GetFont("Courier New", 9, STYLE_NORMAL);

   if (hFont1 = 0 || hFont2 = 0 || hFont3 = 0 || hFont4 = 0) then
      // エラーを報告し、終了します。
      MessageBox (" すべてのフォントを取得できませんでした。", SEVERE);
      abort;
   endif;

   // このインストールでカスタム ボックスを認識するための名前を指定します。
   szDialogName = "CustomDialog";

   // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
   // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
   // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
   // 4 番目のパラメーターにある ID によって識別されるためです。

   nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

   if (nResult < 0) then
      // エラーを報告し、終了します。
      MessageBox (" ダイアログの定義エラー ", SEVERE);
      abort;
   endif;

   // while ループを制御するのに使われるインジケーターを初期化します。
   bDone = FALSE;
```

```
// 完了するまでループします。
repeat

   // ダイアログを表示して次のダイアログ イベントを戻します。
   nCmdValue = WaitOnDialog (szDialogName);

   // イベントに応答します。
   switch (nCmdValue)
      case DLG_CLOSE:
         // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
         Do (EXIT);
      case DLG_ERR:
         MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
         abort;
      case DLG_INIT:
         // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
         // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
         // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
         // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
         hwndDlg = CmdGetHwndDlg (szDialogName);
         SdGeneralInit(szDialogName, hwndDlg, 0, "");

         // スタティック テキスト ボックス 1 のフォントとテキストを設定します。
         if (CtrlSetFont (szDialogName, hFont1, RES_TEXT_1) = 0) then
            CtrlSetText (szDialogName, RES_TEXT_1,
                  " このテキストは Arial bold  14 ポイントに設定されています。");
         else
            CtrlSetText (szDialogName, RES_TEXT_1,
               " 最初のスタティック テキストボックス用のフォントを設定できませんでした。");
         endif;

         // スタティック テキスト ボックス 2 のフォントとテキストを設定します。
         if (CtrlSetFont (szDialogName, hFont2, RES_TEXT_2) = 0) then
            CtrlSetText (szDialogName, RES_TEXT_2,
                  " このテキストは Times New Roman italic 11 ポイントに設定されています。");
         else
            CtrlSetText (szDialogName, RES_TEXT_2,
                  "2 番目のスタティックテキストボックス用のフォントを設定できませんでした。");
         endif;

         // スタティック テキスト ボックス 3 のフォントとテキストを設定します。
         if (CtrlSetFont (szDialogName, hFont3, RES_TEXT_3) = 0) then
            CtrlSetText (szDialogName, RES_TEXT_3,
                  " このテキストは Arial bold  10 ポイントに設定されています。");
         else
            CtrlSetText (szDialogName, RES_TEXT_3,
                  "3 番目のスタティックテキストボックス用のフォントを設定できませんでした。");
         endif;

         // スタティック テキスト ボックス 4 のフォントとテキストを設定します。
         if (CtrlSetFont (szDialogName, hFont4, RES_TEXT_4) = 0) then
            CtrlSetText (szDialogName, RES_TEXT_4,
                  " このテキストは Courier New  9 ポイントに設定されています。");
         else
            CtrlSetText (szDialogName, RES_TEXT_4,
                  "4 番目のスタティックテキストボックス用のフォントを設定できませんでした。");
          endif;
      case RES_PBUT_NEXT:
         bDone = TRUE;
```

```
            case RES_PBUT_CANCEL:
                // ユーザーが [ キャンセル ] ボタンをクリックしました。
                Do (EXIT);
        endswitch;

    until bDone;

    // ダイアログを閉じます
    EndDialog (szDialogName);

    // メモリからダイアログを解放します。
    ReleaseDialog (szDialogName);

end;
```

CtrlSetList

CtrlSetList 関数は、指定した単独または複数選択リストボックス、あるいはコンボボックスコントロールに文字列リストの内容を配置します。既存するコンテンツはすべて、listID に含まれるアイテムで置換されます。InstallShield は、リストボックスまたはコンボボックスコントロールの各要素に文字列リストの各要素を配置します。

構文

```
CtrlSetList (szDialogName, nControlID, listID);
```

パラメーター

テーブル 119 ・ CtrlSetList のパラメーター

パラメーター	説明
szDialogName	リストボックスまたはコンボボックスを含むダイアログの名前を指定します。
nControlID	リストボックスまたはコンボボックスのリソース ID を指定します。
listID	リストボックスまたはコンボボックスコントロールにコピーする要素を含む文字列リストの名前を指定します。

戻り値

テーブル 120 ・ CtrlSetList の戻り値

戻り値	説明
0	CtrlSetList はコントロールへ文字列リストの内容を配置しました。
< 0	CtrlSetList はコントロールへ文字列リストの内容を配置することができませんでした。

CtrlSetList の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CtrlSetList 関数のデモンストレーションを行います。
*
* このスクリプト例では、リスト ボックスを含むカスタム ダイアログを
* 表示します。ダイアログが初期化された後、
* された後、スクリプトは CtrlSetList を呼び出して
* InstallShield 背景色の定数リストをカスタムダイアログのリストボックスへ
* 配置します。
*
* ユーザーは定数をダブルクリックするか、定数を選択してから
* [ 設定 ] ボタンをクリックして色定数に対応する背景を
* 参照することができます。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
```

```
    * 実際、ビルトイン関数 SdAskOptions が表示する
    * InstallShield 標準ダイアログです。このダイアログは
    * インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
    * カスタム ダイアログとしてスクリプトで利用することが
    * 表示されます。スクリプトはダイアログの
    * スタティック テキストを変更し、例の要件を満たすように
    * [ 戻る ] ボタンを無効にします。
    *
    ¥*---------------------------------------------------------------*/

    // ダイアログのコントロール。
    #define RES_DIALOG_ID       12033   // カスタム ダイアログの ID
    #define RES_PBUTTON_SET         1   // [ 次へ ] ボタンの ID
    #define RES_PBUTTON_DONE        9   // [ キャンセル ] ボタンの ID
    #define RES_PBUTTON_BACK       12   // [ 戻る ] ボタンの ID
    #define RES_DIALOG_LISTBOX    401   // 編集ボックスの ID.
    #define RES_TEXT_ABOVE        710   // 編集ボックス上のテキスト ID
    #define RES_TEXT_BELOW        711   // 編集ボックス下のテキスト ID

    // 複数行編集ボックスの上と下に表示する説明。
    #define DESC_TEXT_ABOVE "InstallShield の定義済み定数を利用して作成することのできる背景色を参照します。"
    #define DESC_TEXT_BELOW " 背景色を変更するには、色を選択してから [ 選択 ] ボタンをクリックするか、または色の名前を
    ダブルクリックしてください。"

    // ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
    #include "Ifx.h"

        // 色リストを作成するスクリプト定義の関数。
        prototype CreateColorList ();

        // 背景色を変更するスクリプト定義の関数。
        prototype SetBackgroundColor (STRING);

          export prototype ExFn_CtrlSetList(HWND);

    function ExFn_CtrlSetList(hMSI)
        STRING szDialogName, svCurSel;
        NUMBER nCmdValue, nResult;
        BOOL   bDone;
        LIST   listBackgroundColors;
        HWND   hwndDlg;
    begin

        Enable ( BACKGROUND );

        // このインストールでカスタム ボックスを認識するための名前を指定します。
        szDialogName = "CustomDialog";

        // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
        // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
        // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
        // 4 番目のパラメーターにある ID によって識別されるためです。
        nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

        if (nResult < 0) then
            // エラーを報告し、終了します。
            MessageBox (" ダイアログの定義エラー ", SEVERE);
            abort;
        endif;
```

```
// 色リストを作成するスクリプト定義の関数を呼び出します。
listBackgroundColors = CreateColorList ();

if (listBackgroundColors = LIST_NULL) then
   MessageBox (" 背景色のリストを作成することができませんでした ", SEVERE);
   abort;
endif;

// ループを制御するのに使われるインジケーターを初期化します。
bDone = FALSE;

repeat

   // ダイアログを表示して次のダイアログ イベントを戻します。
   nCmdValue = WaitOnDialog (szDialogName);

   // イベントに応答します。
   switch (nCmdValue)
      case DLG_CLOSE:
         // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
         bDone = TRUE;
      case DLG_ERR:
         MessageBox (" ダイアログが失敗しました ", SEVERE);
         bDone = TRUE;
      case DLG_INIT:
         // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
         // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
         // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
         // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
         hwndDlg = CmdGetHwndDlg (szDialogName);
         SdGeneralInit(szDialogName, hwndDlg, 0, "");

         // ウィンドウのタイトルを設定します。
         SetWindowText (hwndDlg, "View Program Folders");

         // ダイアログのスタティック テキスト。
         CtrlSetText (szDialogName, RES_TEXT_ABOVE, DESC_TEXT_ABOVE);
         CtrlSetText (szDialogName, RES_TEXT_BELOW, DESC_TEXT_BELOW);
         CtrlSetText (szDialogName, RES_PBUTTON_SET, " 設定 (&S)");
         CtrlSetText (szDialogName, RES_PBUTTON_DONE, " 完了 (&D)");

          // Winsub からの呼び出しを利用して [ 戻る ] ボタンを無効にします。
         _WinSubEnableControl (hwndDlg, RES_PBUTTON_BACK, 0);

         // ダイアログのリストボックスへカラーリストを配置します。
         nResult = CtrlSetList (szDialogName, RES_DIALOG_LISTBOX,
                                listBackgroundColors);
         if (nResult != 0) then
            // CtrlSetList からのハンドルエラー。
            MessageBox (" フォルダー名リストを作成できませんでした。", SEVERE);
            bDone = TRUE;
         endif;

         // 色リストの破棄。
         ListDestroy (listBackgroundColors);
      case RES_DIALOG_LISTBOX:
         // エンドユーザーが色をダブルクリックしたとき、それを表示します。
         if (CtrlGetSubCommand (szDialogName) = LISTBOX_ENTER) then
```

```
                CtrlGetCurSel (szDialogName, RES_DIALOG_LISTBOX, svCurSel);
                SetBackgroundColor (svCurSel);
            endif;
        case RES_PBUTTON_DONE:
            bDone = TRUE;
        case RES_PBUTTON_SET :
            // 選択された色を表示します。
            CtrlGetCurSel (szDialogName, RES_DIALOG_LISTBOX, svCurSel);
            SetBackgroundColor (svCurSel);
    endswitch;

  until bDone;

end;

/*----------------------------------------------------------------*¥
 *
 * スクリプト定義の関数をここから始めます。
 *
¥*----------------------------------------------------------------*/

// CreateColorList は背景色定数のリストを戻します。
function CreateColorList ()
  LIST listBkColors;
begin
  // 色定数を保持するためのリストを作成します。
  listBkColors = ListCreate (STRINGLIST);

  // 色定数のリストをビルドします。
  if (listBkColors != LIST_NULL) then
    ListAddString (listBkColors, "BK_BLUE", AFTER);
    ListAddString (listBkColors, "BK_GREEN", AFTER);
    ListAddString (listBkColors, "BK_MAGENTA", AFTER);
    ListAddString (listBkColors, "BK_ORANGE", AFTER);
    ListAddString (listBkColors, "BK_RED", AFTER);
    ListAddString (listBkColors, "BK_YELLOW", AFTER);
    ListAddString (listBkColors, "BK_SOLIDBLACK", AFTER);
    ListAddString (listBkColors, "BK_SOLIDBLUE", AFTER);
    ListAddString (listBkColors, "BK_SOLIDGREEN", AFTER);
    ListAddString (listBkColors, "BK_SOLIDMAGENTA", AFTER);
    ListAddString (listBkColors, "BK_SOLIDORANGE", AFTER);
    ListAddString (listBkColors, "BK_SOLIDPINK", AFTER);
    ListAddString (listBkColors, "BK_SOLIDRED", AFTER);
    ListAddString (listBkColors, "BK_SOLIDWHITE", AFTER);
    ListAddString (listBkColors, "BK_SOLIDYELLOW", AFTER);
  endif;

  // リストへポインターを戻します。
  return listBkColors;
end;

// SetBackgroundColor が szColor によって指定された色へ
// 背景を設定します。
function SetBackgroundColor (szColor)
  NUMBER nColor;
begin
  // エンドユーザーがどの色を選択したかを判断します。
  if szColor = "BK_BLUE" then
    nColor = BK_BLUE;
```

```
  elseif szColor = "BK_GREEN" then
    nColor = BK_GREEN;
  elseif szColor = "BK_MAGENTA" then
    nColor = BK_MAGENTA;
  elseif szColor = "BK_ORANGE" then
    nColor = BK_ORANGE;
  elseif szColor = "BK_RED" then
    nColor = BK_RED;
  elseif szColor = "BK_YELLOW" then
    nColor = BK_YELLOW;
  elseif szColor = "BK_SOLIDBLACK" then
    nColor = BK_SOLIDBLACK;
  elseif szColor = "BK_SOLIDBLUE" then
    nColor = BK_SOLIDBLACK;
  elseif szColor = "BK_SOLIDGREEN" then
    nColor = BK_SOLIDGREEN;
  elseif szColor = "BK_SOLIDMAGENTA" then
    nColor = BK_SOLIDMAGENTA;
  elseif szColor = "BK_SOLIDORANGE" then
    nColor = BK_SOLIDORANGE;
  elseif szColor = "BK_SOLIDPINK" then
    nColor = BK_SOLIDPINK;
  elseif szColor = "BK_SOLIDRED" then
    nColor = BK_SOLIDRED;
  elseif szColor = "BK_SOLIDWHITE" then
    nColor = BK_SOLIDWHITE;
  elseif szColor = "BK_SOLIDYELLOW" then
    nColor = BK_SOLIDYELLOW;
  endif;

  // 選択した色へ背景を設定します。
  SetColor (BACKGROUND, nColor);
end;
```

CtrlSetMLEText

CtrlSetMLEText 関数は複数行編集ボックスコントロールのテキストを設定します。InstallShield は、listID の各文字列を複数編集ボックス コントロールに配置します。この関数はカスタム ダイアログでのみ利用できます。

構文

```
CtrlSetMLEText ( szDialogName, nControlID, listID );
```

パラメーター

テーブル 121 ・ CtrlSetMLEText のパラメーター

パラメーター	説明
szDialogName	ダイアログの名前を指定します。
nControlID	ダイアログの複数行編集ボックス コントロールのリソース ID を指定します。
listID	複数行編集コントロールにコピーする要素を含む有効な文字列リストの名前を指定します。

戻り値

テーブル 122 ・ CtrlSetMLEText の戻り値

戻り値	説明
0	CtrlSetMLEText テキストをコントロールへ設定します。
< 0	CtrlGetMLEText はテキストをコントロールへ設定することができませんでした。

CtrlSetMLEText の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CtrlSetMLEText 関数と CtrlGetMLEText 関数のデモンストレーションを行います。
*
* このスクリプト例では、4 つのチェック ボックスを含むカスタム ダイアログを
* 表示します。このスクリプトは
* ターゲット システム上にすべてのプログラム フォルダーのリストを
* 作成し、CtrlSetMLEText を呼び出し、そのリストをダイアログの複数行編集ボックスに
* 配置します。またダイアログは [ 保存 ] ボタンを含み、エンドユーザーがフォルダー名を
* テキストファイルに保存することができます。
* オプションが選択されたとき、スクリプトは CtrlGetMLEText を呼び出して、
* 複数行編集ボックスからフォルダー名を取得します。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SdAskOptions が表示する
* InstallShield 標準ダイアログです。このダイアログは
```

```
* インストールで既に圧縮済みのファイル _isres.dll に保存されており、
* カスタム ダイアログとしてスクリプトで利用することが
* 表示されます。
*
* メモ : 複数行編集ボックスはリソースでは読み取り専用と定義
*       されています。その内容を編集することはできません。
*
*       スクリプトはダイアログ ボックスの [ 次へ ] ボタンのスタティック テキストを
*       変更し、[ 戻る ] ボタンを無効にしてダイアログが例の目的を
達成できるようにします。
*
*       GetGroupNameList 関数は、ターゲットシステムが Explorer シェル以外の
*       シェルで実行されている場合にエラーを返す場合があります。
*
*
¥*-------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID       12007   // カスタム ダイアログの ID
#define RES_PBUT_BACK         12   // [ 次へ ] ボタンの ID
#define RES_PBUT_DONE          9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_SAVE          1   // [ 戻る ] ボタンの ID
#define RES_DIALOG_EDITBOX     301   // 編集ボックスの ID
#define RES_TEXT              711   // 編集ボックス上のテキスト ID

// 複数行編集ボックスの上に表示する説明。
#define DESC_TEXT "[ 保存 ] をクリックして、ディスクファイルへプログラムフォルダー名のリストを保存して下さい。"

// エンドユーザーが [ 保存 ] ボタンをクリックした際、プログラム名が
// 現在のドライブのルートに保存されます。
#define FOLDER_LIST_FILE "¥¥ISExampl.txt"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlSetMLEText(HWND);

function ExFn_CtrlSetMLEText(hMSI)
   STRING szDialogName;
   NUMBER nCmdValue, nResult;
   BOOL   bSave, bDone;
   LIST   listFolders;
   HWND   hwndDlg;
begin

   // このインストールでカスタム ボックスを認識するための名前を指定します。
   szDialogName = "CustomDialog";

   // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
   // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
   // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
   // 4 番目のパラメーターにある ID によって識別されるためです。
   nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

   if (nResult < 0) then
      // エラーを報告し、終了します。
      MessageBox ("ダイアログの定義エラー ", SEVERE);
      abort;
   endif;
```

```
// while ループを制御するのに使われるインジケーターを初期化します。
bDone = FALSE;

// 完了するまでループします。
repeat

   // ダイアログを表示して次のダイアログ イベントを戻します。
   nCmdValue = WaitOnDialog (szDialogName);

   // イベントに応答します。
   switch (nCmdValue)
      case DLG_CLOSE:
         // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
         bDone = TRUE;
      case DLG_ERR:
         MessageBox (" ダイアログが失敗しました ", SEVERE);
         bDone = TRUE;
      case DLG_INIT:
         // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
         // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
         // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
         // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
         hwndDlg = CmdGetHwndDlg (szDialogName);
         SdGeneralInit(szDialogName, hwndDlg, 0, "");

         // ウィンドウのタイトルを設定します。
         SetWindowText (hwndDlg, "View Program Folders");

         // Winsub からの呼び出しを利用して [ 戻る ] ボタンを無効にします。
         _WinSubEnableControl (hwndDlg, RES_PBUT_BACK, 0);

         // ダイアログのスタティック テキスト。
         CtrlSetText (szDialogName, RES_TEXT, DESC_TEXT);
         CtrlSetText (szDialogName, RES_PBUT_SAVE, " 保存 (&S)");
         CtrlSetText (szDialogName, RES_PBUT_DONE, " 完了 (&D)");

         // プログラム フォルダー名を保存する文字列リストを作成します。
         listFolders = ListCreate (STRINGLIST);

         if (listFolders = LIST_NULL) then
            MessageBox (" リストを作成できませんでした。", SEVERE);
            bDone = TRUE;
         else
            // リストへフォルダー名を取得します。
            nResult = GetGroupNameList (listFolders);

            if (nResult = 0) then
               // フォルダー名をダイアログ ボックスの
               // 複数行編集ボックスから取得します。
               nResult = CtrlSetMLEText (szDialogName, RES_DIALOG_EDITBOX,
                                         listFolders);
            elseif (nResult != 0) then
               // GetGroupNameList または CtrlSetMLEText からのハンドル エラー。
               MessageBox (" フォルダー名リストを作成できませんでした。", SEVERE);
               bDone = TRUE;
            endif;

            // ListID 文字列リストを破棄します。
```

```
            ListDestroy (listFolders);
        endif;
    case RES_PBUT_SAVE :
        // プログラムファイル名を保存するインジケーターを初期化します。
        bSave = FALSE;

        if (AskYesNo(" リストを " + FOLDER_LIST_FILE + " として保存しますか ?", YES)) then
            // 既存ファイルを確認します。
            if (Is (FILE_EXISTS, FOLDER_LIST_FILE) = 1) then
                // エンドユーザーに対し、既存ファイルの上書きを問い合わせます。
                if (AskYesNo (" 既存の " + FOLDER_LIST_FILE +
                            " を上書きしますか ?", YES)) then
                    bSave = TRUE;
                endif;
            else
                bSave = TRUE;
            endif;
        endif;

        if bSave = TRUE then
            // ダイアログからのリストを保存する文字列リストを作成します。
            listFolders = ListCreate (STRINGLIST);

            if (listFolders = LIST_NULL) then
                MessageBox (" リストを作成できませんでした。", SEVERE);
            else
                // フォルダー名をダイアログ ボックスの
                // 複数行編集ボックスから取得します。
                nResult = CtrlGetMLEText (szDialogName, RES_DIALOG_EDITBOX,
                                          listFolders);
                // リストをテキスト ファイルへ保存します。
                ListWriteToFile (listFolders, FOLDER_LIST_FILE);

                // ListID 文字列リストを破棄します。
                ListDestroy (listFolders);
            endif;
        endif;
    case RES_PBUT_DONE:
        bDone = TRUE;
endswitch;

until bDone;

// カスタム ダイアログを閉じます
EndDialog (szDialogName);

// メモリからカスタム ダイアログを削除します。
ReleaseDialog (szDialogName);

end;
```

CtrlSetMultCurSel

CtrlSetMultCurSel 関数は、指定した複数選択リストまたはコンボボックスコントロールを検索します。If nSelectFlag が TRUE に設定されている場合、CtrlSetMultCurSel は見つけた項目を選択 (ハイライト表示) します。この関数はカスタム ダイアログでのみ利用できます。

構文

CtrlSetMultCurSel (szDialogName, nControlID, szText, nSelectFlag);

パラメーター

テーブル 123 ・ CtrlSetMultCurSel のパラメーター

パラメーター	説明
szDialogName	カスタム ダイアログの名前を指定します。
nControlID	ダイアログの複数選択リストボックスコントロールのリソース ID を指定します。
szText	検索文字列を指定します。
nSelectFlag	CtrlSetMultCurSel が検出した場合、アイテムを選択するかどうかを示します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ TRUE – アイテムが選択されることを示します。 ・ FALSE – アイテムが選択されないことを示します。

戻り値

テーブル 124 ・ CtrlSetMultCurSel の戻り値

戻り値	説明
0	CtrlSetMultCurSel がコントロールでテキストを検出し、nSelectFla で示されたとおり選択または非選択を行いました。
< 0	CtrlSetMultCurSel はテキストをコントロールへ検出することができませんでした。

CtrlSetMultCurSel の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
```

```
*
* CtrlSetMultCurSel  関数と CtrlGetMultCurSel 関数の
* デモンストレーションを行います。
*
* このスクリプトはターゲットシステム上のすべてのプログラムフォルダーの名前を
* 読み出して、リストへ配置します。ダイアログ ボックスが
* 初期化されるとき、CtrlSetList 関数はこのリストをリスト ボックスで
* 表示するように設定します。次いで、CtrlSetMultCurSel 関数が
* 呼び出されてユーザーが選択したフォルダーを
* ハイライト表示します。
*
* その後このリストは破棄されます。[ 次へ ] ボタンが押されたとき、
* 新しいリストが作成されます。CtrlGetMultCurSel はリストボックスの要素を
* 読み出し、この新しい文字列リストへそれらを
* 割り当てます。このリストは、標準ダイアログで表示されます。
*
* メモ : このスクリプトを適切に実行するため、
*      RES_DIALOG_ID 定数と RES_DIALOG_LISTBOX 定数を _isuser.dll で
作成されたダイアログとリスト ボックスに設定しなくてはなりません。
*
*      ターゲット システムが Explorer シェル以外のシェルで実行
*      されている場合、この例で使われている GetGroupNameList 関数は
*    エラーを戻す場合があります。
*
¥*---------------------------------------------------------------*/

// ダイアログのコントロール。
#define RES_DIALOG_ID// ダイアログの ID
#define RES_PBUT_NEXT        1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL      9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK        12  // [ 戻る ] ボタンの ID
#define RES_DIALOG_LISTBOX       // リストボックスの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlSetMultCurSel(HWND);

function ExFn_CtrlSetMultCurSel(hMSI)
   STRING szDialogName, szDLL, szTitle, szMsg;
   STRING szText, szDefFolder, svResultFolder;
   NUMBER nCmdValue, nResult, nControlID, nSelectFlag;
   BOOL   bDone;
   LIST listID, listFolders;
   HWND hwndDlg;
begin

   Disable(BACKBUTTON);

   szDialogName = "CtrlSetMultCurSel";
   szDLL        = "";

   // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
   // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
   // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
   // 4 番目のパラメーターにある ID によって識別されるためです。
   nResult = EzDefineDialog (szDialogName, szDLL, "", RES_DIALOG_ID);

   if (nResult < 0) then
```

```
    MessageBox ("ダイアログの定義エラー", SEVERE);
    bDone = TRUE;
else
    bDone = FALSE;
endif;

// ListID 文字列リストを作成します。
listID = ListCreate (STRINGLIST);

if (listID = LIST_NULL) then
   MessageBox ("リストを作成できませんでした。", SEVERE);
else
   MessageBox ("listID が作成されました。", INFORMATION);
endif;

// リストへプログラムフォルダー名を読み出します。
GetGroupNameList (listID);

// ユーザーからフォルダー名を取得します。
szTitle = "CtrlGetMultCurSel & CtrlSetMultCurSel";
SelectFolder (szTitle, szDefFolder, svResultFolder);

// 完了するまでループします。
while (bDone = FALSE)

    // ダイアログを表示して次のダイアログ イベントを戻します。
    nCmdValue = WaitOnDialog (szDialogName);

    // イベントに応答します。
    switch (nCmdValue)
        case DLG_ERR:
            MessageBox ("ダイアログを表示できませんでした。セットアップはキャンセルされました。",SEVERE);
            abort;
        case DLG_INIT:
            // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
            // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
            // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
            // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
            hwndDlg = CmdGetHwndDlg (szDialogName);
            SdGeneralInit(szDialogName, hwndDlg, 0, "");

            // 次はリストボックスをプログラムフォルダーのリストへ設定します。
            nControlID = RES_DIALOG_LISTBOX;
            CtrlSetList (szDialogName, nControlID, listID);

            szText = svResultFolder;
            nSelectFlag = TRUE;

            // ユーザーが選択したフォルダーをハイライト表示にします。
            if (CtrlSetMultCurSel (szDialogName, nControlID, szText,
                                   nSelectFlag) < 0) then
                MessageBox ("CtrlSetMultCurSel が失敗しました。", SEVERE);
            endif;

            // ListID 文字列リストを破棄します。
            ListDestroy(listID);
            MessageBox ("listID が破棄されました。", INFORMATION);
        case DLG_CLOSE:
            // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
```

```
            Do (EXIT);
        case RES_PBUT_NEXT:
            // listFolders 文字列リストを作成します。
            listFolders = ListCreate (STRINGLIST);

            if (listFolders = LIST_NULL) then
                MessageBox (" リストを作成できませんでした。", SEVERE);
            else
                MessageBox ("listFolders が作成されました。", INFORMATION);
            endif;

            // リストボックスでハイライト表示された要素を読み出し、
            // listFolders 文字列リストへ配置します。
            if (CtrlGetMultCurSel (szDialogName, nControlID,
                                   listFolders) < 0) then
                MessageBox ("CtrlGetMultCurSel が失敗しました。", SEVERE);
            else
                MessageBox ("CtrlGetMultCurSel が成功しました。", INFORMATION);
            endif;

            bDone = TRUE;
        case RES_PBUT_BACK:
            bDone = TRUE;
        case RES_PBUT_CANCEL:
            // ユーザーがウィンドウの [ キャンセル ] ボタンをクリックしました。
            Do (EXIT);
    endswitch;

endwhile;

szMsg  = " 次はリストボックスでハイライト表示された要素です :";

// ハイライト表示された要素のリストを表示します。
SdShowInfoList (szTitle, szMsg, listFolders);

// メモリから listFolder 文字列リストを削除します。
ListDestroy (listFolders);
MessageBox ("listFolders が破棄されました。", INFORMATION);

// ダイアログを閉じます
EndDialog (szDialogName);

// メモリからダイアログを削除します。
ReleaseDialog (szDialogName);

end;
```

CtrlSetState

CtrlSetState 関数は、カスタムダイアログからチェックボックスまたはオプションボタンコントロールの状態を設定します。リソースエディターまたはダイアログエディターを使って作成した場合、オプションボタンとチェックボックスに特定の用途を設定することができます。ボタンコントロールの動作に不都合を感じた場合は、エディターでコントロールの特性を確認してください。

構文

CtrlSetState (szDialogName, nControlID, nState);

パラメーター

テーブル 125 ・ CtrlSetState のパラメーター

パラメーター	説明
szDialogName	チェック ボックスまたはオプションボタン コントロールを含むダイアログの名前を指定します。
nControlID	チェック ボックスまたはオプションボタン コントロールのリソース ID を指定します。
nState	ボタンコントロールの新しい状態を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ BUTTON_CHECKED – ボタンの状態を CHECKED に設定します。 ・ BUTTON_UNCHECKED – ボタンの状態を UNCHECKED に設定します。

戻り値

テーブル 126 ・ CtrlSetState の戻り値

戻り値	説明
0	CtrlSetState をチェック ボックスまたはオプションボタンコントロールの状態を設定しました。
< 0	CtrlSetState はコントロールの状態を設定することができませんでした。

CtrlSetState の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
```

```
＊CtrlGetState 関数と CtrlSetStat 関数のデモンストレーションを行います。
＊
＊このスクリプト例では、4 つのチェック ボックスを含むカスタム ダイアログを
＊表示します。スクリプトは CtrlSetState を呼び出して最初の 2 つの
＊チェック ボックスをチェック済みに設定します。残りの 2 つはデフォルトで、
＊チェック無しです。ユーザーが [ 次へ ] ボタンをクリックしたとき、スクリプトは
＊CtrlGetState を呼び出し、各チェック ボックスの状態を
＊読み出します。そしてスクリプトは、どのボックスがチェックされているかを
＊メッセージボックスに表示します。
＊
＊このスクリプトで利用される [ カスタム ] ダイアログは、
＊実際、ビルトイン関数 SdAskOptions が表示する
＊InstallShield 標準ダイアログです。このダイアログは
＊インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
＊カスタム ダイアログとしてスクリプトで利用することが
＊表示されます。
＊
¥*----------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID       12020   // カスタム ダイアログの ID
#define RES_PBUT_NEXT         1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL       9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK        12   // [ 戻る ] ボタンの ID
#define  ID_OP1_CHECK       501   // オプション 1 チェック ボックスの ID
#define  ID_OP2_CHECK       502   // オプション 2 チェック ボックスの ID
#define  ID_OP3_CHECK       503   // オプション 3 チェック ボックスの ID
#define  ID_OP4_CHECK       504   // オプション 4 チェック ボックスの ID
#define  ID_STA_DESC        711   // スタティック テキスト説明の ID


// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlSetState(HWND);

function ExFn_CtrlSetState(hMSI)
   STRING  szDialogName, szMsg;
   NUMBER  nResult, nCmdValue, hwndDlg;
   BOOL    bDone;
begin

   // このインストールでカスタム ボックスを認識するための名前を指定します。
   szDialogName = "ExDialog";

   // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
   // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
   // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
   // 4 番目のパラメーターにある ID によって識別されるためです。
   nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

   if (nResult < 0) then
      // エラーを報告し、終了します。
      MessageBox (" ダイアログの定義エラー ", SEVERE);
      abort;
   endif;

   // ループを制御するのに使われるインジケーターを初期化します。
   bDone = FALSE;
```

```
repeat

    // ダイアログを表示して次のダイアログ イベントを戻します。
    nCmdValue = WaitOnDialog (szDialogName);

    // イベントに応答します。
    switch (nCmdValue)
        case DLG_CLOSE:
            // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
            Do (EXIT);
        case DLG_ERR:
            MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
            abort;
        case DLG_INIT:
            // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
            // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
            // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
            // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
            hwndDlg = CmdGetHwndDlg (szDialogName);
            SdGeneralInit(szDialogName, hwndDlg, 0, "");

            // ウィンドウのタイトルを設定します。
            SetWindowText (hwndDlg, " オプションを選択 ");

            // チェック ボックスの上に表示されるスタティックテキスト説明を設定します。
            CtrlSetText (szDialogName, ID_STA_DESC,
                    " オプションを選択および / またはクリアします。そして [ 次へ ] をクリックします。");

            // デフォルトでオプションはクリアされているので、オプション 1 と 2 を選択します。
            if (CtrlSetState (szDialogName, ID_OP1_CHECK, BUTTON_CHECKED) < 0) then
                MessageBox ("CtrlSetState の最初の呼び出しに失敗しました。", SEVERE);
                bDone = TRUE;
            elseif (CtrlSetState(szDialogName, ID_OP2_CHECK, BUTTON_CHECKED) <  0) then
                MessageBox ("CtrlSetState への 2 回目の呼び出しに失敗しました。", SEVERE);
                bDone = TRUE;
            endif;

        case RES_PBUT_NEXT:
            bDone = TRUE;
        case RES_PBUT_CANCEL:
            // ユーザーが [ キャンセル ] ボタンをクリックしました。
            Do (EXIT);
        case RES_PBUT_BACK:
            bDone = TRUE;
    endswitch;

until bDone;

// エンドユーザーが [ 次へ ] ボタンをクリックするとメッセージをビルドします。
if (nCmdValue = RES_PBUT_NEXT) then
    // エンドユーザーへ表示するメッセージのビルドを開始します。
    szMsg = " 次のアイテムを選択しました :¥n¥n";

    // 最初のオプションが選択された場合、メッセージに行を追加します。
    if (CtrlGetState (szDialogName, ID_OP1_CHECK) = BUTTON_CHECKED) then
        szMsg = szMsg + " オプション 1¥n";
    endif;
```

```
    // 2 番目のオプションが選択された場合、メッセージに行を追加します。
    if (CtrlGetState (szDialogName, ID_OP2_CHECK) = BUTTON_CHECKED) then
        szMsg = szMsg + " オプション 2¥n";
    endif;

    // 3 番目のオプションが選択された場合、メッセージに行を追加します。
    if (CtrlGetState (szDialogName, ID_OP3_CHECK) = BUTTON_CHECKED) then
        szMsg = szMsg + " オプション 3¥n";
    endif;

    // 4 番目のオプションが選択された場合、メッセージに行を追加します。
    if (CtrlGetState (szDialogName, ID_OP4_CHECK) = BUTTON_CHECKED) then
        szMsg = szMsg + " オプション 4¥n";
    endif;
  endif;

  // カスタム ダイアログを閉じます
  EndDialog (szDialogName);

  // メモリからカスタム ダイアログを削除します。
  ReleaseDialog (szDialogName);

  // [ 次へ ] ボタンを使ってダイアログが閉じられたときにメッセージを表示します。
  if (nCmdValue = RES_PBUT_NEXT) then
    MessageBox (szMsg, INFORMATION);
  endif;

end;
```

CtrlSetText

CtrlSetText 関数は、単一行編集フィールド、スタティックテキストフィールド、またはカスタムダイアログのボタンコントロールのテキストを設定します。複数行編集フィールドのテキストを設定するには、CtrlSetMLEText を利用します。

構文

```
CtrlSetText ( szDialogName, nControlID, szText );
```

パラメーター

テーブル 127 ・ CtrlSetText のパラメーター

パラメーター	説明
szDialogName	変更するダイアログ名前を指定します。
nControlID	テキストを設定する単一行編集フィールド、スタティック テキスト フィールド、またはボタン コントロールのリソース ID を指定します。
szText	テキストをコントロールへ配置するよう指定します。 ***ヒント・****コントロールが HTML コントロールの場合、szText 値の始まりに* **[html]** *を指定します。詳細については、「ダイアログで HTML コントロールを使用する」を参照してください。*

戻り値

テーブル 128 ・ CtrlSetText の戻り値

戻り値	説明
0	CtrlSetText がテキストをコントロールへ設定しました。
ISERR_GEN_FAILURE	CtrlSetMLEText はテキストをコントロールへ設定することができませんでした。

CtrlSetText の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* CtrlSetText 関数、CtrlGetText 関数、および CtrlSelectText 関数の
* デモンストレーションを行います。
*
* このスクリプト例では、ユーザーの名前と会社名を取得する 2 つの編集ボックスを
* 持つカスタム ダイアログを表示します。その後
* スクリプトは CtrlSetText を呼び出して初期値を編集ボックスに配置し、
* CtrlSelectText を呼び出して最初の編集ボックスの
* 内容を選択します。ユーザーが [ 次へ ] ボタンを
* クリックすると、スクリプトは CtrlGetText を呼び出して
* 編集ボックスの内容を読み出して、カスタムダイアログボックスを
* 閉じた後にメッセージ ボックスで表示できるようにします。
*
```

```
＊このスクリプトで利用される [ カスタム ] ダイアログは、
＊実際、ビルトイン関数 SdAskOptions が表示する
＊InstallShield 標準ダイアログです。このダイアログは
＊インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
＊カスタム ダイアログとしてスクリプトで利用することが
＊表示されます。
＊
¥*---------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID12001// カスタム ダイアログの ID
#define RES_PBUT_NEXT        1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL      9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK       12   //[ 戻る ] ボタンの ID
#define RES_EDITNAME       301   // 編集ボックスの ID
#define RES_EDITCOMPANY    302   // 編集ボックスの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_CtrlSetText(HWND);

function ExFn_CtrlSetText(hMSI)
   STRING szDialogName, svName, svCompany;
   NUMBER nResult, nCmdValue;
   BOOL   bDone;
   HWND   hwndDlg;
begin

   // このインストールでカスタム ボックスを認識するための名前を指定します。
   szDialogName = "CustomDialog";

   // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
   // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
   // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
   // 4 番目のパラメーターにある ID によって識別されるためです。
   nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

   if (nResult < 0) then
      // エラーを報告し、終了します。
      MessageBox ("ダイアログの定義エラー", SEVERE);
      abort;
   endif;

   // ループを制御するのに使われるインジケーターを初期化します。
   bDone = FALSE;

   // 完了するまでループします。
   repeat

      // ダイアログを表示して次のダイアログ イベントを戻します。
      nCmdValue = WaitOnDialog (szDialogName);

      // イベントに応答します。
      switch (nCmdValue)
         case DLG_CLOSE:
            // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
            Do (EXIT);
         case DLG_ERR:
```

```
            MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
            abort;
        case DLG_INIT:
            // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
            // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
            // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
            // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
            hwndDlg = CmdGetHwndDlg (szDialogName);
            SdGeneralInit(szDialogName, hwndDlg, 0, "");

            // 初期値を編集ボックスに配置します。
            CtrlSetText (szDialogName, RES_EDITNAME, " 名前 ");
            CtrlSetText (szDialogName, RES_EDITCOMPANY, " 会社名 ");

            // [ 名前 ] 編集ボックスを選択します。
            CtrlSelectText (szDialogName, RES_EDITNAME);
        case RES_PBUT_NEXT:
            // 編集ボックスの内容を取得します。
            CtrlGetText (szDialogName, RES_EDITNAME, svName);
            CtrlGetText (szDialogName, RES_EDITCOMPANY, svCompany);

            // 両方の編集ボックスにデータが入力されていることを確認します。
            if (StrLength(svName) = 0) || (StrLength(svCompany) = 0) then
                MessageBox (" 両方のフィールドに入力する必要があります。", INFORMATION);
            else
                bDone = TRUE;
            endif;
        case RES_PBUT_CANCEL:
            // ユーザーが [ キャンセル ] ボタンをクリックしました。
            Do (EXIT);
        case RES_PBUT_BACK:
            bDone = TRUE;
    endswitch;

    until bDone;

    // ダイアログを閉じます
    EndDialog (szDialogName);

    // メモリからダイアログを削除します。
    ReleaseDialog (szDialogName);

    // ダイアログを [ 次へ ] ボタンで閉じた場合、名前と会社名を表示します。
    if nCmdValue = RES_PBUT_NEXT then
        MessageBox (svName + "¥n" + svCompany, INFORMATION);
    endif;

end;
```

DefineDialog

DefineDialog 関数はカスタムダイアログを定義します。EzDefineDialog で指定できないダイアログ属性を指定する場合、**EzDefineDialog** の代わりにこの関数を呼び出します。

***メモ・DefineDialog** はカスタム ダイアログを表示しません。カスタムダイアログを表示するには、WaitOnDialog を呼び出さなくてはなりません。*

構文

DefineDialog (szDialogName, hInstance, szDLLName, nDialogID, szDialogID, nReserved, hwndOwner, lMsgLevel);

パラメーター

テーブル 129 ・ DefineDialog のパラメーター

パラメーター	説明
szDialogName	szDialogID または nDialogID が認識したダイアログと関連付ける名前を指定します。このダイアログを処理するには、カスタムダイアログ関数に続く呼び出しにこの名前を使用します。 ダイアログの名前は大文字と小文字を区別するため、このパラメーターで指定した通りに正確に使用しなくてはなりません。
hInstance	ダイアログが存在する .dll ファイルのインスタンスハンドルを指定します。szDLLName で .dll の完全修飾名を指定した場合、このパラメーターには 0 を指定できます。.dll ファイルのインスタンスハンドルを取得するには、Microsoft Windows API **LoadLibrary** を呼び出します。
szDLLName	このダイアログリソースを含む .dll ファイルの名前を指定します。この名前が修飾されていない場合、つまりファイル名と共にドライブとパスを指定しなかった場合、InstallScript エンジンは Windows フォルダーで .dll ファイルを検索します。そこで検出されなかった場合、InstallScript エンジンは検索パスで指定したフォルダーを検索します。ダイアログが **_isuser.dll** にあるとき、このパラメーターにヌル文字列 ("") を指定することができます。InstallScript はこのパラメーターがヌル文字列 ("") として指定してある場合、**_isuser.dll** を自動的に確認します。 ***メモ・*** **_isres.dll** *または* **_isuser.dll** *以外の .dll ファイルを利用した場合、****EzDefineDialog*** *を呼び出す前に UseDLL を呼び出して .dll ファイルをロードしなくてはなりません。.dll ファイルをメモリからアンロードするには、ReleaseDialog の後に UnUseDLL を呼び出します。*
nDialogID	リソースを識別するのに (文字列ではなく) 数値を利用する場合のダイアログのリソース ID。このパラメーターは szDialogID がヌル文字列 ("") の場合のみ利用します。szDialogID ではなく、nDialogID を使ってダイアログ リソースを識別することが推奨されます。

テーブル 129 ・ DefineDialog のパラメーター (続き)

パラメーター	説明
szDialogID	リソースを識別するのに (数値ではなく) 文字列を利用する場合のダイアログのリソース ID。このパラメーターがヌル文字列 (″″) の場合、はダイアログリソースを識別する為に nDialog を利用します。szDialogID ではなく、nDialogID を使ってダイアログリソースを識別することが推奨されます。 ***メモ ・** [ダイアログ] ビューでダイアログを作成した場合、文字列 ID と共にダイアログが作成されます。つまり、この名前を szDialogID パラメーターとして指定してダイアログを定義づけなくてはなりません。* *[ダイアログ] ビューで既存ダイアログを上書きした場合、ダイアログは数値 ID を持つため、nDialogID パラメーターで ID を指定しなくてはなりません。* *すべての場合に数値 ID を使用する場合 (推奨)、[ダイアログ] ビューでダイアログを編集し、作成されたダイアログの ISResrouceID プロパティを 0 ではなく任意のダイアログ ID に変更する必要があります。この変更を行うと、ダイアログを作成したときにこのダイアログ名は使用されません。この名前は InstallShield が [ダイアログ] ビューでダイアログを識別する場合にのみ使用されます。ダイアログは適切な数値 ID を使ってビルドされます。*
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。
hwndOwner	オーナーウィンドウのウィンドウハンドルを指定します。このパラメーターに HWND_INSTALL を指定して、ダイアログのメインインストール ウィンドウをダイアログのオーナーにします。
lMsgLevel	このパラメーターではどの Windows メッセージをダイアログに送るかを指定します。定数 DLG_CENTERED と次の定数のひとつを OR 演算子 (|) を利用して組み合わせなくてはなりません。 ・ DLG_MSG_STANDARD – ダイアログの制御に直接関わりのあるメッセージのみが渡されるよう、Windows メッセージのほとんどをフィルターします。 ・ DLG_MSG_ALL – Windows メッセージのほとんどを渡します。

戻り値

テーブル 130 ・ DefineDialog の戻り値

戻り値	説明
0	**DefineDialog** がダイアログを定義しました。
DLG_ERR_ALREADY_EXISTS (-3)	インストールスクリプトで定義済みのダイアログを再び定義しようとしている旨を示します。同じ名前で 2 つのダイアログを定義することはできません。
DLG_ERR (-1)	不特定エラー条件を示します。

DefineDialog の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* DefineDialog 関数、EndDialog 関数、そして ReleaseDialog 関数の
* デモンストレーションを行います。
*
* このスクリプトは ビットマップを表示するシンプルなカスタムダイアログを
* 開きます。ダイアログは次の 3 つのボタンで閉じることが
* [ 戻る ]、[ 次へ ]、および [ キャンセル ]。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SdAskOptions が表示する
* InstallShield 標準ダイアログです。このダイアログは
* インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
* カスタム ダイアログとしてスクリプトで利用することが
* 表示されます。
*
* このダイアログをカスタムダイアログとして利用するためには、
* DefineDialog を呼び出してそれをスクリプトで定義します。その後
* WaitOnDialog を呼び出してダイアログを表示します。イベントが
* ダイアログの処理を終了するとき、それを閉じるために EndDialog が
* 表示されます。次いで、ReleaseDialog への呼び出しによって、
* メモリからダイアログがリリースされます。
*
¥*---------------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID    12027  // ダイアログ自身の ID
#define RES_PBUT_NEXT        1  // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL      9  // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK       12  //[ 戻る ] ボタンの ID
```

```
// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_DefineDialog(HWND);

function ExFn_DefineDialog(hMSI)
    STRING  szDialogName, szDLLName, szDialog;
    NUMBER  nDialog, nResult, nCmdValue;
    BOOL    bDone;
    HWND    hInstance, hwndParent, hwndDlg;
begin

    // DefineDialog への最初のパラメーターとして渡すダイアログの
    // 名前を定義します。
    szDialogName = "ExampleDialog";

    // DefineDialog の 2 番目のパラメーターは 0 となります。
    // これは .dll ファイルが _isres.dll の中にあるためです。
    hInstance = 0;

    // DefineDialog の 3 番目のパラメーターは ヌルです。インストールは
    // _isuser.dll と _isres.dll にあるダイアログを検索します。
    szDLLName = "";

    // DefineDialog の 5 番目のパラメーターは 0 となります。なぜなら、
    // 4 番目のパラメーターにある ID によってダイアログが認識されるためです。
    szDialog = "";

    // この値は保存され、0 でなくてはなりません。
    hwndParent = 0;

    // ダイアログを定義します。インストールのメインウィンドウがダイアログ ボックスを保有します
    // ( パラメーター 7 内の HWND_INSTALL で表示されます ) を保有します。
    nResult  = DefineDialog (szDialogName, hInstance, szDLLName,
                        RES_DIALOG_ID, szDialog, hwndParent,
                        HWND_INSTALL, DLG_MSG_STANDARD|DLG_CENTERED);

    // エラーをチェックします。
    if (nResult < 0) then
        MessageBox (" ダイアログを定義中にエラーが発生しました。", SEVERE);
        bDone = TRUE;
        abort;
    endif;

   // while ループを制御するのに使われるインジケーターを初期化します。
   bDone = FALSE;

   // 完了するまでループします。
   repeat

        // ダイアログを表示して次のダイアログ イベントを戻します。
        nCmdValue = WaitOnDialog(szDialogName);

        // イベントに応答します。
        switch (nCmdValue)
            case DLG_CLOSE:
                // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
                Do (EXIT);
            case DLG_ERR:
```

```
            MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
            abort;
        case DLG_INIT:
            // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
            // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
            // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
            // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
            hwndDlg = CmdGetHwndDlg (szDialogName);
            SdGeneralInit(szDialogName, hwndDlg, 0, "");

        case RES_PBUT_CANCEL:
            // ユーザーが [ キャンセル ] ボタンをクリックしました。
            Do (EXIT);
        case RES_PBUT_NEXT:
            bDone = TRUE;
        case RES_PBUT_BACK:
            bDone = TRUE;
        // 標準の処理を確認します
        if (SdIsStdButton( nCmdValue ) && SdDoStdButton( nCmdValue )) then
            bDone = TRUE;
        endif;
    endswitch;

    until bDone;

    // ダイアログを閉じます
    EndDialog (szDialogName);

    // メモリからダイアログを解放します。
    ReleaseDialog (szDialogName);

end;
```

DeinstallSetReference

DeinstallSetReference 関数は現在使用されていません。

ファイルがアンインストール中にロックされているかどうかを確認するには、nIsFlag パラメーターに FILE_LOCKED 定数を使って **Is** 関数を呼び出す、および適切に応答するスクリプト コードを作成します。

構文

```
DeinstallSetReference (szReferenceFile);
```

DeinstallStart

DeinstallStart 関数は現在使用されていません。インストールは、アンインストールを有効にするために必要なレジストリ キーを作成します。

構文

```
DeinstallStart (szObsolete, svObsolete, szObsolete, lReserved);
```

Delay

Delay 関数は、指定した秒数の間スクリプトの実行を遅延します。同時に InstallShield で実行している別のタスクは、InstallShield が遅延されてもそのまま続行されます。

構文

Delay (nSeconds);

Delay の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* Delay 関数のデモンストレーションを行います。
*
* まず初めに、SdShowMsg を呼び出してメッセージボックスを表示されてから
* Delay を呼び出し、スクリプトを 3 秒間停止させます。そして、
* メッセージボックスを削除して Delay を再び呼び出し、2 秒間
* 停止します。最後に、別のメッセージを 3 秒間表示
* します。
*
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_Delay(HWND);

function ExFn_Delay(hMSI)
begin

  SdShowMsg ("このメッセージは 3 秒間表示されます。.", TRUE);

  Delay (3);

  SdShowMsg("", FALSE);

  Delay (2);

  SdShowMsg ("これは別のメッセージで、さらに 3 秒間表示 " +
      "されます。", TRUE);

  Delay (3);

end;
```

DeleteCHARArray

説明

DeleteCHARArray 関数は pCHARArray がポイントするポインターの配列を削除します。

構文

DeleteCHARArray (pCHARArray);

パラメーター

テーブル 131 - DeleteCHARArray のパラメーター

パラメーター	説明
pCHARArray	ANSI 文字列へのポインターの配列へのポインターを指定します。一般的に、このポインターは以前の GetCHARArrayFromISStringArray への呼び出しで戻されます。

戻り値

テーブル 132 - DeleteCHARArray の戻り値

戻り値	説明
ISERR_SUCCESS	この関数は常に ISERR_SUCCESS を戻します。

DeleteDir

DeleteDir 関数はサブディレクトリを削除します。nFlag パラメーターの値によって、サブディレクトリが空の場合のみ消去、サブディレクトリがファイルを含んでいても削除、またはルートディレクトリ全体を削除することができます。nFlag 設定の際は充分に注意して下さい。

***メモ -** 次の制限事項に注意してください。*

- *DeleteDir を使って現在のディレクトリを消去することはできません。*
- *ネットワークシステム上にあるファイルについて適切な権利を持たない範囲ではそれらを削除することはできません。*
- *DeleteFile では読み取り専用ファイル、非表示ファイル、またシステムファイルは削除できません。*
- *DeleteDir が読み取り専用ファイルを検出した場合、サブディレクトリにあるファイルのいくつかを削除した後に関数が失敗することもあります。*

構文

DeleteDir(szDir, nFlag);

パラメーター

テーブル 133 ・ DeleteDir のパラメーター

パラメーター	説明
szDir	削除するディレクトリの完全修飾名を指定します。
nFlag	削除オプションを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • ALLCONTENTS ― すべてのサブディレクトリとその下にあるファイルを含んで、szDir にあるディレクトリを削除します。削除するディレクトリはドライブのルートディレクトリではなく、サブディレクトリでなくてはなりません。 • ONLYDIR ― 空白の場合、szDir にあるディレクトリを削除します。その他の場合、関数は失敗します。 • ROOT ― szDir にあるディレクトリを、それがルートディレクトリの場合でも削除します。ｓｚDir がルートディレクトリの場合、DeleteDir がディスク上のすべてを削除します。

戻り値

テーブル 134 ・ DeleteDir の戻り値

戻り値	説明
0	関数がサブディレクトリを削除したことを示します。
< 0	関数がアサブディレクトリを削除できなかったことを示します。

DeleteDir の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
```

```
＊InstallShield スクリプトの例
＊
＊DeleteDir 関数のデモンストレーションを行います。
＊
＊まず、CreateDir を呼び出してディレクトリを作成します。そして、
＊DeleteDir を呼び出してそれを削除します。
＊
¥*---------------------------------------------------------------*/

#define EXAMPLE_DIR "C:¥¥Newdir"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_DeleteDir(HWND);

function ExFn_DeleteDir(hMSI)
begin

  // ディレクトリを作成します。
  if (CreateDir (EXAMPLE_DIR) != 0) then
    // エラーを報告し、中止します。
    MessageBox ("ディレクトリを作成できませんでした。", SEVERE);
  else

    // 成功を報告します。
    MessageBox (EXAMPLE_DIR + " が作成されました。", INFORMATION);

    // ディレクトリを削除します。ディレクトリが空でない場合、
    // 削除しません。
    if (DeleteDir (EXAMPLE_DIR, ONLYDIR) = 0) then
      // 成功を報告します。
      MessageBox (EXAMPLE_DIR + " が削除されました。", INFORMATION);
    else
      MessageBox ("ディレクトリを削除できませんでした。", SEVERE);
    endif;

  endif;

end;
```

DeleteFile

DeleteFile 関数は単数または複数ファイルを削除します。

プロジェクト・基本の MSI または InstallScript MSI プロジェクトでは、この関数は Windows Installer に備わっている RemoveFiles アクションを利用した方が良い結果が得られる場合があります。詳細については、Windows Installer ヘルプを参照してください。

メモ・次の事項に注意してください。

- ***DeleteFile** を利用して、適切な権利を持たないネットワーク システム領域にあるファイルを削除することはできません。*

- ***DeleteFile** を使って読み取り専用ファイル、非表示ファイル、またシステム ファイルを削除することはできません。*
- *FindFile と共にワイルドカード文字を使用してファイルを検出し、**DeleteFile** を使って削除することができます。*

構文

DeleteFile (szFile);

パラメーター

テーブル 135 ・ DeleteFile のパラメーター

パラメーター	説明
szFile	削除するファイルの名前を指定します。szFile に完全修飾ファイル名が指定されている、つまりパスが含まれている場合、**DeleteFile** は指定のディレクトリからファイルを削除します。szFile のファイル名が修飾されていない、つまりパス情報が含まれていない場合、**DeleteFile** はシステム変数 TARGETDIR (InstallScript インストールの場合) または INSTALLDIR (基本の MSI および InstallScript MSI インストールの場合) が指定するディレクトリからファイルを削除します。szFile にワイルドカード文字を含めて 1 つ以上のファイルを削除することもできます。

戻り値

テーブル 136 ・ DeleteFile の戻り値

戻り値	説明
0	関数が指定したファイルを正常に削除したことを示します。
ISERR_PATH_NOT_FOUND (0x80070003)	指定されたパスが見つかりませんでした。
ISERR_FILE_NOT_FOUND (0x80070004)	指定されたファイルが見つからないか、指定のワイルドカードに一致するファイルがありませんでした。
その他すべての負の値	関数で指定された 1 つ以上のファイルを削除できませんでした。複数のファイルの場合、削除できなかった最後のファイルのエラーが返されます。システム変数、ERRORFILENAME には、削除できなかったファイルの一覧がセミコロンで区切られて表示されます。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

DeleteFile の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* DeleteFile 関数のデモンストレーションを行います。
*
```

```
＊まず DeleteFile を呼び出し、指定したファイルをディレクトリから削除
＊作成します。そして再びこの関数を呼び出して "ｓｙｓ" 識別子を持つファイルすべてを
＊同じディレクトリから削除します。
＊
＊メモ : このスクリプトを実行する前に、C ドライブのルートへ
＊     ISExampl と名づけられたディレクトリを作成します。その後、ディレクトリに
＊    ISExampl.txt という名のファイルを作成します。最後に、
＊     そのディレクトリに識別子 "sys" を持つファイルを 2 つ以上
＊作成します。これらのファイルはスクリプトが削除します。
＊
¥*---------------------------------------------------------------*/

#define DEL_DIR      "C:¥¥ISExampl"
#define DEL_FILE     "ISExampl.txt"
#define DEL_SYS_FILES "*.sys"
#define TITLE_TEXT   "DeleteFile の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_DeleteFile(HWND);

function ExFn_DeleteFile(hMSI)
  STRING szMsg;
begin

   // ターゲットディレクトリから DEL_FILE が指定したファイルを削除します。
   if (DeleteFile (DEL_DIR ^ DEL_FILE) < 0) then
      MessageBox ("DeleteFile への最初の呼び出しに失敗しました。", SEVERE);
   else
      SprintfBox (INFORMATION, TITLE_TEXT, "%s が %s から削除されました。",
                  DEL_FILE, DEL_DIR);
   endif;

   // DEL_SYS_FILES が指定したファイルを
   // ターゲットディレクトリから削除します。
   if (DeleteFile (DEL_SYS_FILES) < 0) then
      MessageBox ("DeleteFile への 2 回目の呼び出しに失敗しました。", SEVERE);
   else
      SprintfBox (INFORMATION, TITLE_TEXT,
            "%s に一致するすべてのファイルは %s から削除されました。",
            DEL_SYS_FILES, DEL_DIR);
   endif;

end;
```

DeleteFolderIcon

DeleteShortcut 関数は **DeleteFolderIcon** 関数に優先します。

DeleteFolderIcon 関数は、フォルダーからショートカットを削除します。

メモ・DeleteFolderIcon をインターネット ショートカットに使用することはできません。

構文

DeleteFolderIcon (szProgramFolder, szItemName);

パラメーター

テーブル 137 ・ DeleteFolderIcon のパラメーター

パラメーター	説明
szProgramFolder	削除するショートカットを含むフォルダーの名前を指定します。
szItemName	削除するショートカットの名前を指定します。

戻り値

テーブル 138 ・ DeleteFolderIcon のパラメーター

戻り値	説明
0	関数が指定したショートカットを正常に削除したことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。
< 0	関数がショートカットを削除できなかったことを示します。

DeleteFolderIcon の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* DeleteFolderIcon 関数と DeleteProgramFolder 関数の
* デモンストレーションを行います。
*
* このスクリプトは「フォルダー例」フォルダーから「メモ帳の例」アイコンを
* 削除します。そして、DeleteProgramFolder が再び呼び出され、
* このフォルダーを削除します。
*
* メモ : このスクリプトを適切に実行するため、
*      プリプロセッサ定数をターゲットシステムの有効なフォルダーとアイコンに
*      設定してください。この例に使用するショートカットを簡単に作成するには、
*      AddFolderIcon の例 #3 を実行してください。
*
¥*---------------------------------------------------------------------*/
```

```
#define FOLDER "C:¥¥Windows¥¥ フォルダー例 "
#define ICON   "Notepad の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_DeleteFolderIcon(HWND);

function ExFn_DeleteFolderIcon(hMSI)
begin

   // フォルダーを表示します。
   ShowProgramFolder (FOLDER, SW_SHOW);
   Delay (3);

   // 'Notepad の例 ' アイコンを削除します。
   if (DeleteFolderIcon (FOLDER, ICON) < 0) then
      MessageBox ("DeleteFolderIcon が失敗しました。", SEVERE);
   endif;

   // 「フォルダー例」アイコンを削除します。
   if (DeleteProgramFolder (FOLDER) < 0) then
      MessageBox ("DeleteProgramFolder が失敗しました。", SEVERE);
   endif;

end;
```

DeleteProgramFolder

DeleteShortcutFolder 関数は **DeleteFolderIcon** 関数に優先します。

DeleteProgramFolder 関数はプログラムフォルダー (つまり [スタート] メニューの [プログラム] フォルダーのサブフォルダー) と、すべてのショートカットやプログラムフォルダーのすべてのサブフォルダー、および内容を含むそのフォルダーの内容を削除します。DeleteProgramFolder を使って [プログラム] フォルダーを削除することはできません。

ヒント・*Windows Installer ベースまたは InstallScript MSI ベースのプロジェクトでは、Windows Installer が持つ RemoveFiles アクションを利用した方が良い結果が得られる場合があります。または、単にアンインストールするだけの場合、MSI エンジンは本来セットアップ中に作成されたファイルとフォルダーすべての削除処理を行います。RemoveFolders アクションについての詳細は、Windows Installer ヘルプを参照してください。*

構文

```
DeleteProgramFolder (szFolderName);
```

パラメーター

テーブル 139 ・ DeleteProgramFolder のパラメーター

パラメーター	説明
szFolderName	削除するフォルダーの名前を指定します。

戻り値

テーブル 140 ・ DeleteProgramFolder の戻り値

戻り値	説明
0	関数が指定したフォルダーを正常に削除したことを示します。
< 0	関数がフォルダーを削除できなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

DeleteProgramFolder の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* DeleteFolderIcon 関数と DeleteProgramFolder 関数の
* デモンストレーションを行います。
*
* このスクリプトは「フォルダー例」フォルダーから「メモ帳の例」アイコンを
* 削除します。そして、DeleteProgramFolder が再び呼び出され、
* このフォルダーを削除します。
*
* メモ : このスクリプトを適切に実行するため、
*      プリプロセッサ定数をターゲットシステムの有効なフォルダーとアイコンに
*      設定してください。この例に使用するショートカットを簡単に作成するには、
*      AddFolderIcon の例 #3 を実行してください。
*
¥*---------------------------------------------------------------------*/

#define FOLDER "C:¥¥Windows¥¥ フォルダー例 "
#define ICON   "Notepad の例 "
```

```
// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_DeleteProgramFolder(HWND);

function ExFn_DeleteProgramFolder(hMSI)
begin

   // フォルダーを表示します。
   ShowProgramFolder (FOLDER, SW_SHOW);
   Delay (3);

   // 'Notepad の例 ' アイコンを削除します。
   if (DeleteFolderIcon (FOLDER, ICON) < 0) then
      MessageBox ("DeleteFolderIcon が失敗しました。", SEVERE);
   endif;

   // 「フォルダー例」アイコンを削除します。
   if (DeleteProgramFolder (FOLDER) < 0) then
      MessageBox ("DeleteProgramFolder が失敗しました。", SEVERE);
   endif;

end;
```

DeleteShortcut

DeleteShortcut 関数は、特定のフォルダーからショートカットを削除します。

***メモ・DeleteShortcut** をインターネット ショートカットに使用することはできません。*

構文

```
DeleteShortcut (szShortcutFolder, szName);
```

パラメーター

テーブル 141 ・ DeleteShortcut のパラメーター

パラメーター	説明
szShortcutFolder	削除するショートカットを含むフォルダーの名前を指定します。
szName	変更するショートカットの名前を指定します。

戻り値

テーブル 142 ・ DeleteShortcut のパラメーター

戻り値	説明
0	関数が指定したショートカットを正常に削除したことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。
< 0	関数がショートカットを削除できなかったことを示します。

DeleteShortcut の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* DeleteShortcut 関数と DeleteShortcutFolder 関数の
* デモンストレーションを行います。
*
* このスクリプトは「フォルダー例 3」から 「メモ帳の例 3」ショートカットを
* 削除します。そして、DeleteShortcutFolder が呼び出され、
* このフォルダーを削除します。
*
* メモ : このスクリプトを適切に実行するため、
*      プリプロセッサ定数をターゲットシステムの有効なフォルダーとショートカットに
*      設定してください。この例に使用するショートカットを簡単に作成するには、
*      するには、CreateShortcut の例 #3 を実行してください。
*
¥*---------------------------------------------------------------------*/

#define FOLDER "C:¥¥Windows¥¥ フォルダー例 3"
#define ICON   " メモ帳の例 "

function OnFirstUIAfter( )
```

```
begin

    // フォルダーを表示します。
    ShowProgramFolder (FOLDER, SW_SHOW);
    Delay (3);

    // 「メモ帳の例 3」アイコンを削除します。
    if (DeleteShortcut (FOLDER, SHORTCUT) < 0) then
        MessageBox ("DeleteShortcut が失敗しました。", SEVERE);
    endif;

    // 「フォルダー例 3」ショートカットを削除します。
    if (DeleteShortcutFolder (FOLDER) < 0) then
        MessageBox ("DeleteShortcutFolder が失敗しました。", SEVERE);
    endif;

end;
```

DeleteShortcutFolder

DeleteShortcutFolder 関数はショートカット フォルダー (つまり [スタート] メニューの [プログラム] フォルダーのサブフォルダー) と、すべてのショートカットやショートカット フォルダーのすべてのサブフォルダー、および内容を含むそのフォルダーの内容を削除します。**DeleteShortcutFolder** を使って [プログラム] フォルダーを削除することはできません。

ヒント・*基本の MSI または InstallScript MSI プロジェクトでは、この関数は Windows Installer に備わっている DeleteShortcutFolder アクションを利用した方が良い結果が得られる場合があります。または、単にアンインストールするだけの場合、Windows Installer エンジンは本来インストール中に作成されたファイルとフォルダーすべての削除処理を行います。RemoveFolders アクションについての詳細は、Windows Installer ヘルプを参照してください。*

構文

```
DeleteShortcutFolder ( szFolderName );
```

パラメーター

テーブル 143 ・ DeleteShortcutFolder のパラメーター

パラメーター	説明
szFolderName	削除するフォルダーの名前を指定します。

戻り値

テーブル 144 ・ DeleteShortcutFolder の戻り値

戻り値	説明
0	関数が指定したフォルダーを正常に削除したことを示します。
< 0	関数がフォルダーを削除できなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

DeleteShortcutFolder の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* DeleteShortcut 関数と DeleteShortcutFolder 関数の
* デモンストレーションを行います。
*
* このスクリプトは「フォルダー例 3」から 「メモ帳の例 3」ショートカットを
* 削除します。そして、DeleteShortcutFolder が呼び出され、
* このフォルダーを削除します。
*
* メモ : このスクリプトを適切に実行するため、
*      プリプロセッサ定数をターゲットシステムの有効なフォルダーとショートカットに
*      設定してください。この例に使用するショートカットを簡単に作成するには、
*      するには、CreateShortcut の例 #3 を実行してください。
*
¥*---------------------------------------------------------------------*/

#define FOLDER "C:¥¥Windows¥¥ フォルダー例 3"
#define ICON   " メモ帳の例 "

function OnFirstUIAfter( )
begin
```

```
    // フォルダーを表示します。
    ShowProgramFolder (FOLDER, SW_SHOW);
    Delay (3);

    // 「メモ帳の例 3」アイコンを削除します。
    if (DeleteShortcut (FOLDER, SHORTCUT) < 0) then
        MessageBox ("DeleteShortcut が失敗しました。", SEVERE);
    endif;

    // 「フォルダー例 3」ショートカットを削除します。
    if (DeleteShortcutFolder (FOLDER) < 0) then
        MessageBox ("DeleteShortcutFolder が失敗しました。", SEVERE);
    endif;

end;
```

DeleteWCHARArray

説明

DeleteWCHARArray 関数は pCHARArray がポイントするポインターの配列を削除します。

構文

```
DeleteWCHARArray ( pCHARArray );
```

パラメーター

テーブル 145 · DeleteWCHARArray のパラメーター

パラメーター	説明
pCHARArray	Unicode 文字列へのポインターの配列へのポインターを指定します。一般的に、このポインターは以前の GetWCHARArrayFromISStringArray への呼び出しで戻されます。

戻り値

テーブル 146 · DeleteWCHARArray 戻り値

戻り値	説明
ISERR_SUCCESS	この関数は常に ISERR_SUCCESS を戻します。

DialogSetFont

DialogSetFont 関数は、実行時に表示される InstallScript ダイアログのフォントを設定します。この関数は、InstallScript ダイアログおよびカスタム InstallScript ダイアログ (EzDefineDialog または DefineDialog で定義されるダイアログ) に使用できます。この関数は、Windows API 関数の MessageBox を呼び出すことによって表示されるダ

イアログには影響を与えません。これらのダイアログはメッセージボックスでユーザーが (Windows コントロール パネルを使って) 指定したフォントで表示されます。この関数はダイアログのタイトルバー内のテキストには影響しません。ダイアログタイトルバーのフォントは Windows が設定します。

構文

DialogSetFont(szFontName, nFontSize, nReserved);

パラメーター

テーブル 147 ・ DialogSetFont のパラメーター

パラメーター	説明
szFontName	使用するフォントを指定します (例、Times New Roman)。
nFontSize	フォントサイズを「10」などのように指定します。
nReserved	このパラメーターでゼロ (0) を渡します。他の値は使用できません。

戻り値

DialogSetFont は常に 0 (ゼロ) を返します。関数でフォントを変更できない場合、システムフォントにダイアログのテキストが表示されます。

追加情報

InstallShield ダイアログのフォントを変更する場合、セットアップを実行するすべてのシステムで使用できることがわかっているフォントを使用するようにしてください。また、さまざまな画面解像度を持つセットアップをテストして、フォントが正しく使用できることを確認してください。

DialogSetInfo

エディション・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI (イベント ドリブン型の InstallScript、InstallScript カスタム アクションを除く)*

DialogSetInfo 関数は、ランタイム ダイアログで、以下の表示要素を変更します :

- 表示されるイメージ
- エンドユーザーの選択を取得するために使用されるチェック ボックスのスタイル
- 有効なハード ドライブ要領を示す値の精度

単一のダイアログの特定の部分を変更するたびに、**DialogSetInfo** を呼び出す必要があります。**DialogSetInfo** の呼び出しによる変更は、残りのインストール中、または後続の **DialogSetInfo** の呼び出しによって再び変更されるまで反映されます。

*メモ・Sd ダイアログ関数を呼び出す前にスクリプトが **DialogSetInfo** を呼び出す場合、SdInit の呼び出しの前に **DialogSetInfo** を呼び出す必要があります。そうでない場合、**DialogSetInfo** の呼び出しは有効になりません。*

構文

```
DialogSetInfo (nInfoType, szInfoString, nParameter);
```

パラメーター

テーブル 148 · DialogSetInfo のパラメーター

パラメーター	説明
nInfoType	変更する表示機能を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • DLG_INFO_USEDECIMAL– デフォルトでは、機能サイズ、利用可能なディスク容量、および必要なディスク容量を示すために表示される値は、キロバイトまたはメガバイトに四捨五入されます。この値をキロバイトまたはメガバイトの 10 分の 1 の単位で四捨五入して表示する場合、この定数を渡します。このパラメーターで影響を受けるダイアログは、**FeatureDialog**、**SdFeatureDialog**、**SdFeatureDialog2**、**SdFeatureDialogAdv**、および **SdFeatureMult** です。 • DLG_INFO_KUNITS– デフォルトでは、機能サイズ、利用可能なディスク容量、および必要なディスク容量を示すために表示される値は、メガバイトで表示されます。この定数を渡して、これらの容量をキロバイトで表示します。このパラメーターで影響を受けるダイアログは、**FeatureDialog**、**SdFeatureDialog**、**SdFeatureDialog2**、**SdFeatureDialogAdv**、および **SdFeatureMult** です。 • DLG_INFO_ALTIMAGE – ダイアログで表示される代替のビットマップを指定します。nParameter が DLG_INFO_ALTIMAGE_VERIFY_BMP または TRUE に設定されている場合、szInfoString は、ダイアログに表示されるイメージを指定する必要があります。このパラメーターは、ダイアログの左側にある標準インストール イメージを表示する ダイアログすべてに適用されます。詳細は「nParameter」の説明を参照してください。 SetDisplayEffect で設定されている表示効果は、通常は特殊効果なしで表示される代替イメージに対しては適用しません。 • DLG_INFO_ALTIMAGE_HIDPI– この定数は、ダイアログで高 DPI イメージを指定します。高 DPI イメージは BMP、GIF、JPEG、PNG、および TIFF などをサポートします。イメージを中央揃えにして、ヘッダーの右側に配置するには、縦横比が 2.5:1 のイメージが推奨されます。これは、InstallShield で使用されているイメージと似たサイズで、イメージ内の実際のロゴ部分が、イメージ全体の約半分の幅で、残りの半分は透明となっています。推奨イメージ サイズは次の通りです : ◦ 100% スケールのイメージの場合は 180x75 ◦ 200% スケールのイメージの場合は 360x150 ◦ 250% スケールのイメージの場合は 450x187 など 透明化が必要な場合、それをサポートする PNG などのイメージ タイプを使い、ダイアログ内で szInfoString が表示するイメージの名前 (オプションでパスを含むことが可能) を指定します。このパラメーターは、ダイアログの左側にある標準インストール イメージを表示する ダイアログすべてに適用されます。DLG_INFO_ALTIMAGE_HIDPI が nInfoType で渡される場合、パラメーター値 szInfoString および nParameter が必要です。詳細は「szInfoString」および「nParameter」の説明を参照してください。 SetDisplayEffect で設定されている表示効果は、通常は特殊効果なしで表示される代替イメージに対しては適用しません。

テーブル 148 ・ DialogSetInfo のパラメーター (続き)

パラメーター	説明
szInfoString	DLG_INFO_ALTIMAGE が nInfoType で渡された場合、このパラメーターは、表示する代替ビットマップのファイル名、およびオプションでビットマップ属性のセットを指定します。ビットマップ属性が含まれている場合、このパラメーターに渡された文字列は次のようにフォーマットする必要があります。 ″ビットマップファイル名 ; 透明フラグ ;3-D flag;< 未使用 >; 背景色 ″ ・ **ビットマップファイル名 ー** ビットマップファイルの名前を指定します。ファイル名が完全でない場合 (つまり、ファイル名にドライブインストール先およびパスが含まれていない場合)、インストールは SUPPORTDIR でビットマップを検索します。 ・ **透過フラグ ー** ビットマップを透過的に示すかどうかを示します。このフラグが 1 (true) の場合、マジェンタ (RGB 値 : 255,0,255) であるビットマップの全部分が透明に表示されます。このパラメーターのデフォルト値は 0 (非透明) です。 ・ **3-D フラグ ー** ビットマップが含まれるスタティック フィールドの縁に 3-D 輪郭線を追加するかどうかを示します。このパラメーターのデフォルト値は 0 (3D 輪郭線なし) です。 ・ **< 未使用 >ー** フォーマットされた文字列のこの部分は無視されますが、必ず含める必要があります。フォーマットされた文字列には 4 つのセミコロンを含め、そのうちの 2 つのセミコロンを 3-D フラグと背景色との間に配置する必要があります。 ・ **背景色 ー** スタティック テキスト フィールドの背景に使用する色を示します。この色は、ビットマップが表示される静的テキストフィールドより小さい場合、または透明フラグが 1 に設定されており、ビットマップに透明な領域がある場合にのみ表示されます。背景色は、コンマで区切られた 3 つの数値である RGB 値として表される必要があります。 次の例では、SUPPORTDIR フォルダーにある **MyBitmap.bmp** ファイルからのビットマップを表示します。ビットマップは黒背景に配置され、3 D 輪郭線はありません。マジェンタであるビットマップはいずれの部分も、黒い背景色で表示されます。 SUPPORTDIR ^ ″MyBitmap.bmp″ + ″;1;1;;0,0,0″ 標準のビットマップは、57 x 53 であることに注意してください。代替ビットマップもこのサイズにする必要があります。ビットマップがこのサイズよりも大きい場合、タイトル領域内で垂直方向の中央に配置されます。さらに、ビットマップも右端がダイアログの右端に位置合わせされます。([Welcome]、[SdWelcome]、および [SdFinish] ダイアログでは、ビットマップの右端がビットマップが表示される大きいイメージの右端に位置合わせされます。) ビットマップの左端は、ダイアログの左に必要なだけ拡張します。ダイアログのタイトル領域の下に拡張するビットマップの部分は、切り取られます。ビットマップが 57 x 53 よりも小さい場合には正しく表示されますが、サイズは自動的に調整されません。 DLG_INFO_ALTIMAGE_HIDPI が nInfoType で渡されるとき、このパラメーターは表示する高 DPI イメージの名前 (オプションでパスも含める) を指定します。ファイルが指定されなかった場合、SUPPORTDIR に存在するものと見なされます。ファイルが存在しない場合は、DialogSetInfo が ISERR_FILE_NOT_FOUND を返します。 デフォルトのイメージが復元されたり、nInfoType が DLG_INFO_ALTIMAGE または DLG_INFO_ALTIMAGE_HIDPI でない場合、このパラメーターは無視されます。

テーブル 148 · DialogSetInfo のパラメーター (続き)

パラメーター	説明
nParameter	nInfoType と組み合わせて、ダイアログ機能の指定に使用されます。 nInfoType が DLG_INFO_ALTIMAGE の場合、以下の定数のうちいずれかを渡し、表示するビットマップを指定します : · DLG_INFO_ALTIMAGE_VERIFY_BMP – szInfoString で示されたビットマップを、次に続くダイアログに使用することを指定します。このビットマップを使用する前に、インストールはそのビットマップの存在を確認します。 · DLG_INFO_ALTIMAGE_REVERT_IMAGE (-1) – 次に続くダイアログが、デフォルトのビットマップを表示することを指定します。インストールは、ビットマップの存在を確認しません。 · TRUE – szInfoString で示されたビットマップを、次に続くダイアログに使用することを指定します。インストールは、そのビットマップの存在を確認しません。 nInfoType が DLG_INFO_ALTIMAGE_HIDPI のとき、nParameter は DPI 拡大 / 縮小率を指定します。たとえば、200% 拡大する場合 200、150% の場合 150 など。サポートされている最小縮小値は 25 です。この値に 0 が渡された場合、イメージは何も表示されません。If DLG_INFO_ALTIMAGE_REVERT_IMAGE が渡されると、以前に使用されたイメージが表示されます。 nInfoType が DLG_INFO_KUNITS または DLG_INFO_USEDECIMAL である場合、以下の定数のうちいずれかを渡し、表示されるサイズを指定します。 · TRUE – nInfoType によって指定されるサイズを表示するよう指定します。 · FALSE – デフォルトのスタイルでサイズを表示するよう指定します。

戻り値

テーブル 149 · DialogSetInfo の戻り値

戻り値	説明
ISERR_SUCCESS (0)	関数が指定されたスタイルを正しく設定しました。 DLG_INFO_ALTIMAGE_VERIFY_BMP が nParameter で渡されると、この戻り値はビットマップが見つかったことも示します。
< ISERR_SUCCESS (< 0)	関数がダイアログ情報を設定しようとした時に、不明なエラーが発生しました。
ISERR_FILE_NOT_FOUND (0x80070004)	szInfoString が示すイメージが見つかりませんでした。

追加情報

InstallScript インストールで **DialogSetInfo** への呼び出し結果をプレビューするには、[ダイアログ サンプラー](ツール メニューの InstallScript サブメニューにあります) を実行してから、[属性] ボタンをクリックしてダイアログの属性を変更し、**SdFeatureMult** などダイアログ内の変更を確認します。

DialogSetInfo の例

エディション・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI (イベント ドリブン型の InstallScript、InstallScript カスタム アクションを除く)*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * DialogSetInfo 関数のデモンストレーションを行います。
 *
 * このスクリプトは AskText を 2 回呼び出します。初めの呼び出しでは、
 * AskText ダイアログがデフォルト ビットマップを表示します。Then
 * DialogSetInfo が呼び出され、代替ビットマップを指定します。
 * そのビットマップが AskText への 2 回目の呼び出しで表示されます。
 *
 * メモ : このスクリプトを実行する前に、定義済み定数
 FULL_BMP_PATH を設定して、[ サポートファイル / ビルボード ] ビューに含まれる
 *      ビットマップ ファイルを参照するように設定してください。
 *
¥*---------------------------------------------------------------------*/

#define FULL_BMP_PATH SUPPORTDIR ^ "MyBitmap.bmp"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

function OnBegin()
    STRING  szText, szMsg, szBmpPath;
    STRING  svReturnText;
    NUMBER  nReturn;
begin

start:

    // セットアップダイアログで [ 戻る ] ボタンを無効にします。
    Disable(BACKBUTTON);

    // AskText ダイアログを、そのデフォルト ビットマップと共に表示します。
    szText  = " デフォルト ビットマップ。";
    szMsg   = " 左のビットマップがデフォルトです。";
    nReturn = AskText (szMsg, szText, svReturnText);

    // [ 戻る ] ボタンを有効にします。
    Enable(BACKBUTTON);

    szBmpPath = FULL_BMP_PATH;

    // AskText ダイアログ用の代替ビットマップを設定します。
    DialogSetInfo (DLG_INFO_ALTIMAGE, szBmpPath, TRUE);

    // AskText ダイアログで表示するテキストを設定します。
    szText = " 代替ビットマップ。";
    szMsg  = " 左のビットマップがカスタム ビットマップです。この代替 " +
```

```
            "ビットマップは DialogSetInfo の DLG_INFO_ALTBITMAP オプションを利用して
            "表示されます。";

        // 代替ビットマップと共に AskText ダイアログを
        // 表示します。
        nReturn = AskText (szMsg, szText, svReturnText);

        // [ 戻る ] ボタンの処理。
        if (nReturn = BACK) then
            // デフォルト ビットマップ設定を復元します。
            DialogSetInfo (DLG_INFO_ALTIMAGE, "", DLG_INFO_ALTIMAGE_REVERT_IMAGE);
            goto start;
        endif;

end;
```

ダイアログ スタイル

4 種類のダイアログ ボックススタイルが利用できます :

- CHECKBOX ダイアログ スタイル
- CHECKBOX95 ダイアログ スタイル
- CHECKMARK ダイアログ スタイル
- CHECKLINE ダイアログ スタイル

CHECKBOX ダイアログ スタイル

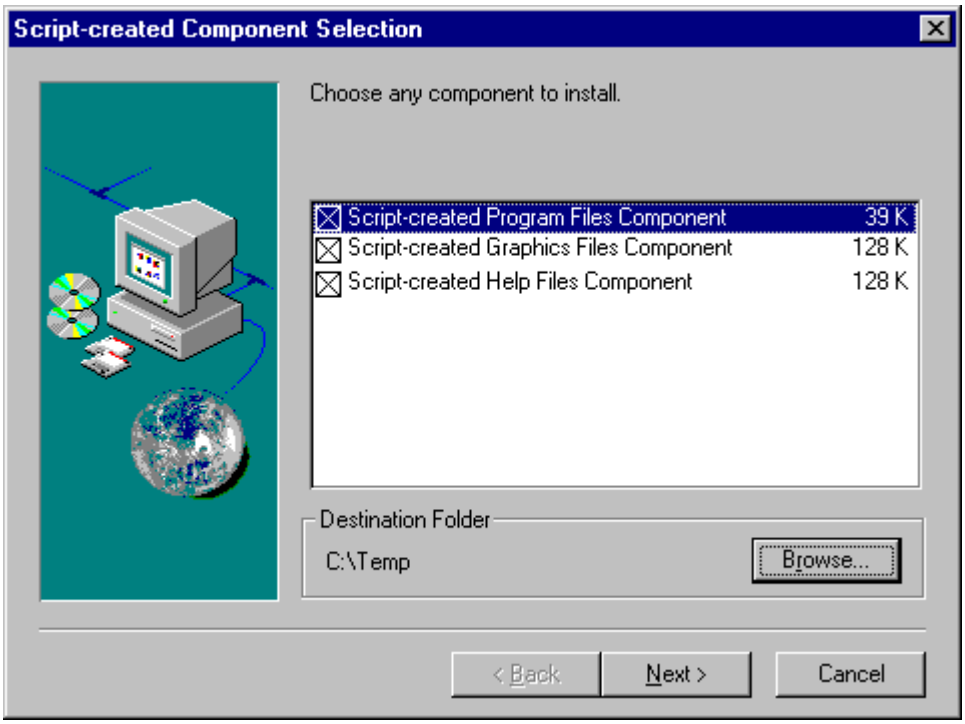

図 1: CHECKBOX ダイアログ スタイル

CHECKBOX95 ダイアログ スタイル

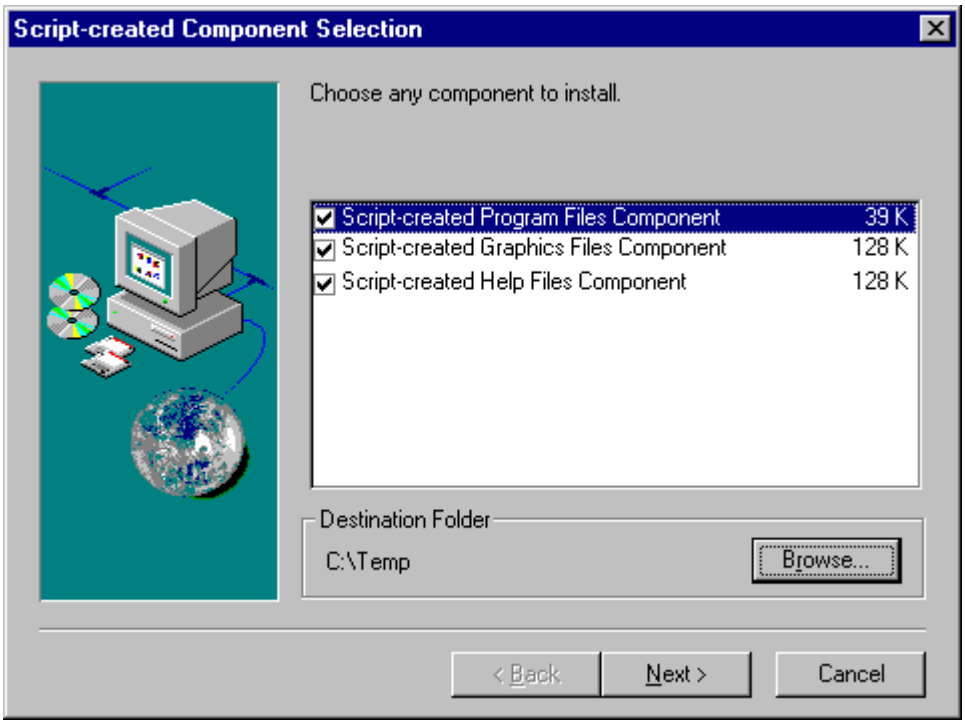

図 2: CHECKBOX95 ダイアログ スタイル

CHECKMARK ダイアログ スタイル

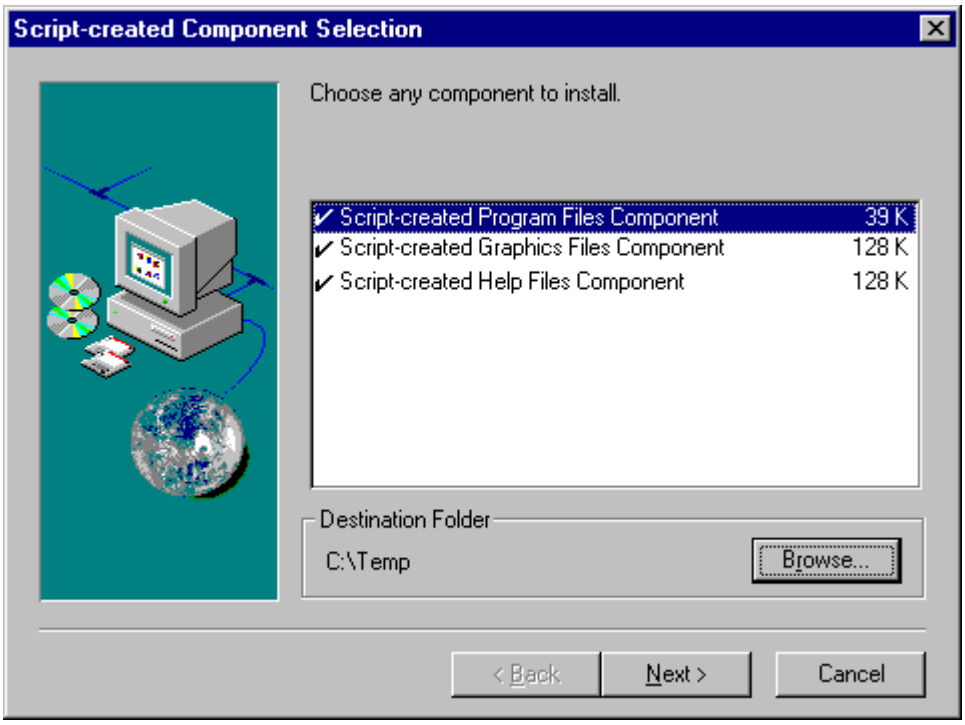

図 3: CHECKMARK ダイアログ スタイル

CHECKLINE ダイアログ スタイル

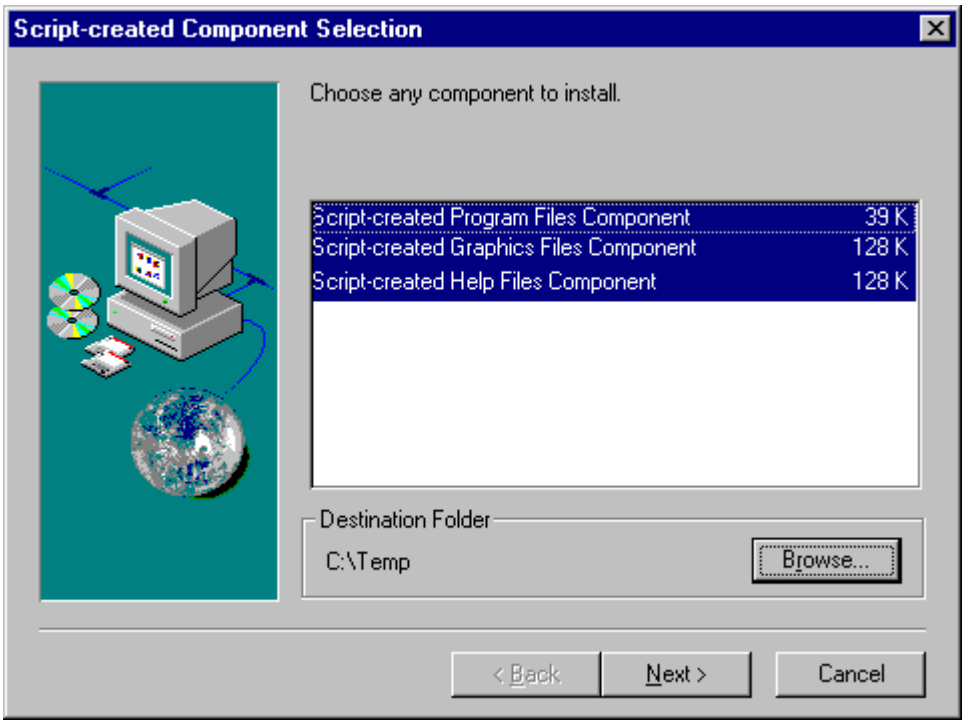

図 4: CHECKLINE ダイアログ スタイル

DIFxDriverPackageGetPath

***プロジェクト・**このトピックは、InstallScript プロジェクトのみに適用します。DIFx は Windows Installer で呼び出すことができるため、この関数は、InstallScript MSI プロジェクトでは必要ありません。*

DIFxDriverPackageGetPath 関数は、ドライバーストア .inf ファイルの完全修飾パスを取得します。.

***メモ・**この関数は、DIFxAPI 関数の DriverPackageGetPath を呼び出します。この関数とそのパラメーター、戻り値に関する詳細は、DIFxAPI マニュアルを参照してください。*

構文

DIFxDriverPackageGetPath(byval string szDriverPackageInfPath, byref string svDestInfPath, byval number nISFlags);

パラメーター

テーブル 150 - DIFxDriverPackageGetPath のパラメーター

パラメーター	説明
szDriverPackageInfPath	対応するドライバーストア .inf ファイルを取得するドライバー パッケージ .inf ファイルに対する完全修飾パスを指定します。
svDestInfPath	DriverPackageInfPath で提供されているドライバー パッケージ .inf ファイルに対応するドライバー ストア .inf ファイルの完全修飾パスを指定します。
nISFlags	InstallScript 特定のフラグを指定します。次のフラグが使用できます。 • 0– デフォルトの動作 • ISDIFX_OPTION_DONT_RESOLVE_TEXTSUBS – デフォルトで、szDriverPackageInfPath で見つかったテキスト置換が解決されます。ただし、このフラグを指定すると、テキスト置換は解決されません。

戻り値

テーブル 151 - DIFxDriverPackageGetPath の戻り値

戻り値	説明
ISERR_SUCCESS	関数は成功しました。
< ISERR_SUCCESS	関数が失敗しました。 **DriverPackageGetPath** からの戻り値が、Win32 エラー (正の戻り値) の場合、ISERR_WIN_BASE がエラーに追加され、エラーが ISERR_SUCCESS であることを明示します。 必要であれば次のコードを使用して元の Win32 エラーを取得できます。 if(nResult & ISERR_WIN_BASE) then nResult = nResult – ISERR_WIN_BASE; endif; 特定のエラーのリストについては、DIFx のエラー (InstallScript プロジェクト) を参照してください。

追加情報

DIFx と DIFxAPI についての詳細は、MSDN ライブラリを参照してください。

DIFxDriverPackageInstall

プロジェクト - *このトピックは、InstallScript プロジェクトのみに適用します。DIFx は Windows Installer で呼び出すことができるため、この関数は、InstallScript MSI プロジェクトでは必要ありません。*

DIFxDriverPackageInstall 関数はドライバー ストアにドライバー パッケージをインストールし、それからシステムにドライバーをインストールします。ドライバーに関連したすべてのファイルが 1 つのコンポーネントの中にあるか、このコンポーネントのインストール イベントが呼び出されたときにすべての適切なドライバー ファイルがインストールされることを保証できる場合は、この関数を呼び出す場所として、DIFx ドライバーのインストール イベントが入ったコンポーネントを使用することをお勧めします。上記の条件に状況が該当しない場合は、インストーラーの OnMoved イベントでこの関数を呼び出してください。

この関数を呼び出したときにアンインストール ログが有効になっていると、この関数によって呼び出されたドライバーのアンインストールはログ記録され、アプリケーションが削除されると自動的に OnUninstallingDIFxDriverFile イベントによって削除されます。

メモ・この関数は、DIFxAPI 関数の DriverPackageInstall を呼び出します。この関数とそのパラメーター、戻り値に関する詳細は、DIFxAPI マニュアルを参照してください。

構文

```
DIFxDriverPackageInstall( byval string szDriverPackageInfPath, byval number nFlags, byval number nISFlags  );
```

パラメーター

テーブル 152 · DIFxDriverPackageInstall のパラメーター

パラメーター	説明
szDriverPackageInfPath	インストールするドライバー パッケージのドライバー パッケージ .inf ファイルに対する完全修飾パスを提供する文字列。
nFlags	インストール操作を制御する 1 つまたは複数のフラグ。ほとんどの場合、インストーラーが自動的に適切なフラグを追加するため、0 を指定することができます。 次の追加タグを手動で指定できます。 ***メモ ·*** *これらのフラグは DIFxAPI 関数 DriverPackageInstall のフラグパラメーターで定義され、直接渡されます。これらのフラグに関する詳細は、DIFxAPI マニュアルを参照してください。* · DRIVER_PACKAGE_REPAIR – ドライバー パッケージが既にインストールされている場合でもドライバー ストアに指定のドライバー パッケージを再インストールします。このフラグは自動的に修復モードで指定されます。 · DRIVER_PACKAGE_ONLY_IF_DEVICE_PRESENT – (PnP 関数ドライバーにのみ適用) そのドライバー パッケージの方が、デバイス ツリーのデバイスより適している場合にのみドライバーをプレインストールおよびインストールします。 · DRIVER_PACKAGE_FORCE–(PnP 関数ドライバーにのみ適用) デフォルトで、新しいドライバーの方が現在デバイスにインストールされているドライバーより適している場合にのみ新しいドライバーをインストールします。このフラグを指定すると、関数は、現在デバイス用にインストールされているドライバー パッケージが指定のドライバー パッケージより適している場合でも指定のドライバー パッケージをプレインストールおよびインストールします。 · DRIVER_PACKAGE_SILENT– ユーザー ダイアログの表示を抑制します。ドライバー署名ダイアログに応答する場合など、インストールを続行するのにユーザー操作が必要になる場合は、ユーザー メッセージを表示せずにインストール操作は失敗します。この関数は、失敗の原因になったエラー コードを返します。 · DRIVER_PACKAGE_LEGACY_MODE– 未署名のドライバー パッケージや見つからないファイルがあるためにプレインストールを完了できないドライバー パッケージをプレインストールおよびインストールします。

テーブル 152 · DIFxDriverPackageInstall のパラメーター (続き)

パラメーター	説明
nISFlags	InstallScript 特定のフラグを指定します。次のフラグが使用できます。 • 0– デフォルトの動作 • ISDIFX_OPTION_DONT_ASSOCIATE – デフォルトで、インストールされたドライバーはインストール中のアプリケーションに関連付けられます。しかしこのフラグを指定すると、ドライバーはアプリケーションに関連付けられません。 • ISDIFX_OPTION_NO_REPAIR– デフォルトでは、REINSTALLMODE が TRUE のときに **DriverPackagePreinstall** 関数を呼び出すと、自動的に DRIVER_PACKAGE_REPAIR フラグが追加されます。しかしこのフラグを指定すると、DRIVER_PACKAGE_REPAIR は自動的に追加されません。ただし、nFlags で指定した場合、フラグは渡されます。 • ISDIFX_OPTION_LOG_IN_DRIVER_PACKAGE_PATH– デフォルトでは、インストールされたドライバーのアンインストールは、インストールされたドライバーのキャッシュにログ記録されます。しかしこのフラグを指定すると、ドライバーはパッケージ パスにログ記録されます。 • ISDIFX_OPTION_DONT_RESOLVE_TEXTSUBS – デフォルトで、szDriverPackageInfPath で見つかったテキスト置換が解決されます。ただし、このフラグを指定すると、テキスト置換は解決されません。

戻り値

テーブル 153 · DIFxDriverPackageInstall の戻り値

戻り値	説明
ISERR_SUCCESS	関数は成功しました。
< ISERR_SUCCESS	関数が失敗しました。 **DriverPackageInstall** からの戻り値が、Win32 エラー (正の戻り値) の場合、ISERR_WIN_BASE がエラーに追加され、エラーが ISERR_SUCCESS であることを明示します。 必要であれば次のコードを使用して元の Win32 エラーを取得できます。 `if( nResult & ISERR_WIN_BASE ) then` `nResult = nResult – ISERR_WIN_BASE;` `endif;` 特定のエラーのリストについては、DIFx のエラー (InstallScript プロジェクト) を参照してください。

追加情報

- DIFx と DIFxAPI についての詳細は、MSDN ライブラリを参照してください。

- **DIFxDriverPackageInstall** を使用してドライバーをインストールしたり **DIFxDriverPackageUninstall** を使用してドライバーをアンインストールすると、ドライバーはデフォルトでインストーラーがインストールしているアプリケーションに関連付けられます。この関連付けは、ISDIFX_OPTION_DONT_ASSOCIATE を指定することによって無効にできます。これらの関数は、次のスクリプト変数を使用して関連付けるアプリケーションを決定します。
  - ISDIFXAPPID
  - IFX_PRODUCT_DISPLAY_NAME
  - IFX_PRODUCT_NAME
  - IFX_COMPANY_NAME

DIFxDriverPackagePreinstall

***プロジェクト・**このトピックは、InstallScript プロジェクトのみに適用します。DIFx は Windows Installer で呼び出すことができるため、この関数は、InstallScript MSI プロジェクトでは必要ありません。*

DIFxDriverPackagePreinstall 関数は、プラグ アンド プレイ (PnP) 関数ドライバーのドライバー パッケージをドライバー ストアにプレインストールし、ドライバー パッケージの .inf ファイルをシステムの .inf ファイル ディレクトリにインストールします。ドライバーに関連したすべてのファイルが 1 つのコンポーネントの中にあるか、このコンポーネントのインストール イベントが呼び出されたときにすべての適切なドライバー ファイルがインストールされることを保証できる場合は、この関数を呼び出す場所として、DIFx ドライバーのインストール イベントが入ったコンポーネントを使用することをお勧めします。上記の条件に状況が該当しない場合は、インストーラーの OnMoved イベントでこの関数を呼び出してください。

この関数を呼び出したときにアンインストール ログが有効になっていると、この関数によってプレインストールされたドライバーのアンインストールはログ記録され、アプリケーションが削除されると自動的に OnUninstallingDIFxDriverFile イベントによって削除されます。

***メモ・**この関数は、DIFxAPI 関数の DriverPackagePreinstall を呼び出します。この関数とそのパラメーター、戻り値に関する詳細は、DIFxAPI マニュアルを参照してください。*

構文

DIFxDriverPackagePreinstall(byval string szDriverPackageInfPath, byval number nFlags, byval number nISFlags);

パラメーター

テーブル 154 ・ DIFxDriverPackagePreinstall のパラメーター

パラメーター	説明
szDriverPackageInfPath	プレインストールするドライバー パッケージのドライバー パッケージ .inf ファイルに対する完全修飾パスを提供する文字列。
nFlags	インストール操作を制御する 1 つまたは複数のフラグ。ほとんどの場合、0 を指定して、インストーラーで自動的に適切なフラグを追加します。 次の追加タグを手動で指定できます。 ***メモ・****これらのフラグは DIFxAPI 関数 DriverPackagePreinstall のフラグパラメーターで定義され、直接渡されます。これらのフラグに関する詳細は、DIFxAPI マニュアルを参照してください。* ・ DRIVER_PACKAGE_REPAIR – ドライバー パッケージが既にインストールされている場合でもドライバー ストアに指定のドライバー パッケージを再インストールします。このフラグは自動的に修復モードで指定されます。 ・ DRIVER_PACKAGE_ONLY_IF_DEVICE_PRESENT – (PnP 関数ドライバーにのみ適用) そのドライバー パッケージの方が、デバイス ツリーのデバイスより適している場合にのみドライバーをプレインストールおよびインストールします。 ・ DRIVER_PACKAGE_FORCE–(PnP 関数ドライバーにのみ適用) デフォルトで、新しいドライバーの方が現在デバイスにインストールされているドライバーより適している場合にのみ新しいドライバーをインストールします。このフラグを指定すると、関数は、現在デバイス用にインストールされているドライバー パッケージが指定のドライバー パッケージより適している場合でも指定のドライバー パッケージをプレインストールおよびインストールします。 ・ DRIVER_PACKAGE_SILENT– ユーザー ダイアログの表示を抑制します。ドライバー署名ダイアログに応答する場合など、インストールを続行するのにユーザー操作が必要になる場合は、ユーザー メッセージを表示せずにインストール操作は失敗します。この関数は、失敗の原因になったエラー コードを返します。 ・ DRIVER_PACKAGE_LEGACY_MODE– 未署名のドライバー パッケージや見つからないファイルがあるためにプレインストールを完了できないドライバー パッケージをプレインストールおよびインストールします。

テーブル 154 · DIFxDriverPackagePreinstall のパラメーター (続き)

パラメーター	説明
nISFlags	InstallScript 特定のフラグを指定します。次のフラグが使用できます。 • 0— デフォルトの動作 • ISDIFX_OPTION_NO_REPAIR— デフォルトでは、REINSTALLMODE が TRUE のときに **DriverPackagePreinstall** 関数を呼び出すと、自動的に DRIVER_PACKAGE_REPAIR フラグが追加されます。しかしこのフラグを指定すると、DRIVER_PACKAGE_REPAIR は自動的に追加されません。ただし、nFlags で指定した場合、フラグは渡されます。 • ISDIFX_OPTION_LOG_IN_DRIVER_PACKAGE_PATH— デフォルトでは、インストールされたドライバーのアンインストールは、インストールされたドライバーのキャッシュにログ記録されます。しかしこのフラグを指定すると、ドライバーはパッケージ ページにログ記録されます。 • ISDIFX_OPTION_DONT_RESOLVE_TEXTSUBS — デフォルトで、**szDriverPackageInfPath** で見つかったテキスト置換が解決されます。このフラグを指定すると、テキスト置換は解決されません。

戻り値

テーブル 155 · DIFxDriverPackagePreinstall の戻り値

戻り値	説明
ISERR_SUCCESS	関数は成功しました。
< ISERR_SUCCESS	関数が失敗しました。 **DriverPackagePreinstall** からの戻り値が、Win32 エラー (正の戻り値) の場合、ISERR_WIN_BASE がエラーに追加され、エラーが ISERR_SUCCESS であることを明示します。 必要であれば次のコードを使用して元の Win32 エラーを取得できます。 `if( nResult & ISERR_WIN_BASE ) then` `nResult = nResult – ISERR_WIN_BASE;` `endif;` 特定のエラーのリストについては、DIFx のエラー (InstallScript プロジェクト) を参照してください。

追加情報

DIFx と DIFxAPI についての詳細は、MSDN ライブラリを参照してください。

DIFxDriverPackageUninstall

***プロジェクト・**このトピックは、InstallScript プロジェクトのみに適用します。DIFx は Windows Installer で呼び出すことができるため、この関数は、InstallScript MSI プロジェクトでは必要ありません。*

DIFxDriverPackageUninstall 関数は、システムから指定のドライバー パッケージをアンインストールし、ドライバー ストアからドライバー パッケージを削除します。

DIFxDriverPackageInstall または **DIFxDriverPackagePreinstall** を使用してインストールしたドライバーの場合は、アンインストール ログが有効な間にこれらのドライバーが自動的に OnUninstallingDIFxDriverFile イベントによって削除されるため、この関数を明示的に呼び出す必要はありません。

***メモ・**この関数は、DIFxAPI 関数の DriverPackageUninstall を呼び出します。この関数とそのパラメーター、戻り値に関する詳細は、DIFxAPI マニュアルを参照してください。*

構文

```
DIFxDriverPackageUninstall( byval string szDriverPackageInfPath, byval number nFlags, byval number nISFlags );
```

パラメーター

テーブル 156 ・ DIFxDriverPackageUninstall のパラメーター

パラメーター	説明
szDriverPackageInfPath	プレインストールするドライバー パッケージのドライバー パッケージ .inf ファイルに対する完全修飾パスを提供する文字列。
nFlags	インストール操作を制御する 1 つまたは複数のフラグ。ほとんどの場合、0 を指定して、インストーラーで自動的に適切なフラグを追加します。 次の追加タグを手動で指定できます。 ***メモ・****これらのフラグは DIFxAPI 関数 DriverPackageUninstall のフラグパラメーターで定義され、直接渡されます。これらのフラグに関する詳細は、DIFxAPI マニュアルを参照してください。* ・ DRIVER_PACKAGE_FORCE—(PnP 関数ドライバーにのみ適用) デフォルトで、新しいドライバーの方が現在デバイスにインストールされているドライバーより適している場合にのみ新しいドライバーをインストールします。このフラグを指定すると、関数は、現在デバイス用にインストールされているドライバー パッケージが指定のドライバー パッケージより適している場合でも指定のドライバー パッケージをプレインストールおよびインストールします。 ・ DRIVER_PACKAGE_SILENT— ユーザー ダイアログの表示を抑制します。ドライバー署名ダイアログに応答する場合など、インストールを続行するのにユーザー操作が必要になる場合は、ユーザー メッセージを表示せずにインストール操作は失敗します。この関数は、失敗の原因になったエラーコードを返します。 ・ DRIVER_PACKAGE_DELETE_FILES— ドライバー パッケージがインストールされたときにシステムにコピーされたバイナリ ファイルをシステムから削除します。この関数は、バイナリ ファイルがドライバー ストア内の対応するバイナリ ファイルと同一の場合のみ、そのバイナリ ファイルをシステムから削除します。 ***注意・****このフラグを使用する際には注意が必要です。システムのバイナリ ファイルが他のドライバー パッケージまたはアプリケーションで必要ないことを確認できる場合にのみこのフラグを使用してください。*

テーブル 156 ・ DIFxDriverPackageUninstall のパラメーター (続き)

パラメーター	説明
nISFlags	InstallScript 特定のフラグを指定します。次のフラグが使用できます。 ・ 0— デフォルトの動作 ・ ISDIFX_OPTION_DONT_ASSOCIATE — デフォルトで、インストールされたドライバーはインストール中のアプリケーションに関連付けられます。しかしこのフラグを指定すると、ドライバーはアプリケーションに関連付けられません。 ・ ISDIFX_OPTION_DONT_RESOLVE_TEXTSUBS — デフォルトで、szDriverPackageInfPath で見つかったテキスト置換が解決されます。このフラグを指定すると、テキスト置換は解決されません。

戻り値

テーブル 157 ・ DIFxDriverPackageUninstall の戻り値

戻り値	説明
ISERR_SUCCESS	関数は成功しました。
< ISERR_SUCCESS	関数が失敗しました。 **DriverPackageUninstall** からの戻り値が、Win32 エラー (正の戻り値) の場合、ISERR_WIN_BASE がエラーに追加され、エラーが < ISERR_SUCCESS であることを明示します。 必要であれば次のコードを使用して元の Win32 エラーを取得できます。 `if( nResult & ISERR_WIN_BASE ) then` `nResult = nResult - ISERR_WIN_BASE;` `endif;` 特定のエラーのリストについては、DIFx のエラー (InstallScript プロジェクト) を参照してください。

追加情報

- DIFx と DIFxAPI についての詳細は、MSDN ライブラリを参照してください。
- **DIFxDriverPackageInstall** を使用してドライバーをインストールしたり **DIFxDriverPackageUninstall** を使用してドライバーをアンインストールすると、ドライバーはデフォルトでインストーラーがインストールしているアプリケーションに関連付けられます。この関連付けは、ISDIFX_OPTION_DONT_ASSOCIATE を指定することによって無効にできます。これらの関数は、次のスクリプト変数を使用して関連付けるアプリケーションを決定します。
  - ISDIFXAPPID
  - IFX_PRODUCT_DISPLAY_NAME
  - IFX_PRODUCT_NAME
  - IFX_COMPANY_NAME

Disable

Disable 関数は、nConstant パラメーターで指定したユーザーインターフェイスオブジェクトやセットアップ機能をアクティブにします。

スクリプトで **Disable** 関数を呼び出して [次へ] ボタンや [戻る] ボタンを無効にすると、関数呼び出しの後に表示されるすべてのダイアログ内のボタンが無効になります。[次へ] ボタンや [戻る] ボタン を再度有効にするには、対応する定数を使用して Enable 関数を呼び出す必要があります。

構文

```
Disable ( nConstant );
```

パラメーター

テーブル 158 ・ Disable のパラメーター

パラメーター	説明
nConstant	無効にするユーザーインターフェースオブジェクトやオペレーション機能を指定します。 このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ BACKBUTTON – いくつかのビルトイン ダイアログに表示される [戻る] ボタンを無効 (灰色表示) にします。[戻る] ボタンは、デフォルトで有効です。 ・ BACKGROUND – インストールのメイン背景ウィンドウを無効、または非表示にします。このパラメーターはインストールが全画面モードの場合は無効です。 ・ BILLBOARD – インストール中にビルボードの表示を抑制します。 ・ CANCELBUTTON – いくつかのビルトイン ダイアログに表示される [キャンセル] ボタンを無効 (灰色表示) にします。 ・ DIALOGCACHE – ダイアログ キャッシュのメカニズムを無効にします。ダイアログキャッシュについては、「Enable」を参照してください。 *__メモ ・__ DIALOGCACHE は、[次へ] ボタンや [戻る] ボタンがないダイアログでは無効です。* ・ HOURGLASS – マウス カーソルを ″ ビジー ″ カーソル (デフォルトでは砂時計) から普通のカーソル (デフォルトではポインター) へ変更します。

テーブル 158 ・ Disable のパラメーター (続き)

パラメーター	説明
nConstant (続き)	・ LOGGING – アンインストール ログ ファイルに情報が記録されないよう、アンインストール情報のログを無効にします。 デフォルトで、ログ記録は自動的に有効となります。ログ記録を無効にする必要がある場合、アンインストール向けにログ記録しない処理を行う直前に LOGGING 定数を使って **Disable** を呼び出す方法が推奨されます。その後、LOGGING 定数を使って **Enable** を呼び出し、ログ記録を再有効化します。 ログ記録に関する詳細については、「アンインストール用にログされた InstallScript 関数」を参照してください。  ***ヒント・****アンインストール中にコンポーネントが行った変更が元に戻らないように防ぐためには、InstallScript プロジェクトでコンポーネントの ″アンインストール″ 設定に [いいえ] を選択するか、InstallScript MSI プロジェクトでコンポーネントの ″パーマネント ″ 設定に [はい] を選択します。* ・ NEXTBUTTON – いくつかのビルトイン InstallScript ダイアログに表示される [次へ] ボタンを無効 (灰色表示) にします。デフォルトでは、ほとんどのダイアログでの [次へ] ボタンは有効になっています。  ***メモ・****NEXTBUTTON 定数を使って **Disable** を呼び出しても、Next ボタンを内部的に有効化 / 無効化する **SdCustomerInformation**、**SdCustomerInformationEx**、**SdRegisterUser**、または **SdRegisterUserEx** ダイアログでは効果がありません。* ・ PCRESTORE– デフォルトで有効になっている [システム回復互換性] を無効にします。 ・ REGISTRYFUNCTIONS_USETEXTSUBS– レジストリ関数に渡された文字列のテキスト置換を無効にします。これはデフォルトで有効になっています。このオプションは、開き山かっこ (<) と閉じ山かっこ (>) を含むレジストリ関数文字列で作業しているときに使用しますが、テキスト置換と解釈すべきではありません。 ・ SELFREGISTERBATCH– XCopyFile および SELFREGISTER 定数を使ってコピーしたファイルの登録の「バッチメソッド」を無効にします。バッチメソッドは、デフォルトでは有効です。 バッチ メソッドが無効な場合、**XCopyFile** と SELFREGISTER 定数を使ってコピーしたファイルはすぐに登録されます。batch メソッドが有効な場合、データ転送が行われるまで登録は遅延されます。

テーブル 158 ・ Disable のパラメーター (続き)

パラメーター	説明
nConstant (続き)	・ SERVICE_DIFX_32—32 ビット プラットフォームの DIFx サポートを無効にします。 ・ SERVICE_DIFX_AMD64—AMD 64 ビット プラットフォームの DIFx サポートを無効にします。 ・ SERVICE_DIFX_IA64—Itanium 64 ビット プラットフォームの DIFx サポートを無効にします。 ・ SERVICE_ISFONTREG— グローバル フォント登録を無効にします。詳しい情報は、「InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトでフォントをインストールする」を参照してください。デフォルトではグローバルフォント登録が有効になっています。一度無効に設定すると、スクリプトから再び有効にすることはできません。 ・ STATUS— 進行状況インジケーター (ステータス バー) を無効または非表示にします。 ・ STATUSBBRD— ビルボードを含む進行状況ダイアログを無効または非表示にします。 ・ STATUSDLG— ダイアログ スタイル進行状況インジケーター (ステータス バー) を無効または非表示にします。 ・ STATUSEX— デフォルト [セットアップ ステータス] ダイアログの表示を無効にします。 ・ STATUSOLD— 古いスタイルのプログレス インジケーター (ステータス バー) を無効または非表示にします。 ・ UPDATE_SERVICE_INSTALL— この定数は、古い形式です。 ・ USE_LOADED_SKIN— ダイアログ スキンと共に動作しないカスタム ダイアログ (例えば、標準サイズではないダイアログ) での利用を目的とします。ビルトイン ダイアログとの利用はお勧めしません。[リリース] ビューでリリースの [ビルド] タブにある " スキンの指定 " 設定で指定されたスキンの使用を無効にします。(この設定で **<スキンを利用しない>** オプションを選択した場合、この定数は効果を持ちません。) スキンを無効にすることができるのは、InstallScript コードで Disable(USE_LOADED_SKIN); を呼び出した後に表示されるダイアログのみです。カスタムダイアログでのスキンの利用を無効にするには、**WaitOnDialog** を呼び出す前に Disable(USE_LOADED_SKIN); を呼び出さなくてはなりません。再びスキンの利用を有効にするには、Enable(USE_LOADED_SKIN); を呼び出します。 ***プロジェクト ・** USE_LOADED_SKIN 定数は、InstallScript プロジェクトで使用できます。* ・ WOW64FSREDIRECTION—64 ビット Windows のファイル システム リダイレクトを無効化します。場合によって、これは、ファイルを WINSYSDIR64 にインストールする前に行う必要があります。詳しくは、「64 ビット オペレーティング システムを InstallScript インストールでターゲットにする」を参照してください。

戻り値

テーブル 159 ・ Disable の戻り値

戻り値	説明
0	関数が nConstant パラメーターで指定したユーザーインターフェイスオブジェクトやセットアップ機能を無効にしたことを示します。
< 0	関数が、nConstant のパラメーターで指定したユーザーインターフェイスオブジェクトやセットアップ機能を無効にできなかったことを示します。

Disable の例

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * Disable 関数と Enable 関数のデモンストレーションを行います。
 *
 * このスクリプトは 2 つのダイアログを表示します。最初のボックスでは、
 * [ 戻る ] ボタンが無効になっています。2 番目のボックスでは [ 次へ ] ボタンが
 * 無効で、[ 戻る ] ボタンが有効になっています。
 *
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_Disable(HWND);

function ExFn_Disable(hMSI)
begin

start:

  // セットアップダイアログで [ 戻る ] ボタンを無効にします。
  Disable(BACKBUTTON);

  // 次は [ 戻る ] ボタンが無効となったダイアログを表示します。
  SetupType ("", "", "", TYPICAL, 0);

  // [ 戻る ] ボタンを有効にします、
  Enable(BACKBUTTON);

  // [ 次へ ] ボタンが無効になっています。
  Disable (NEXTBUTTON);

  // 次は [ 戻る ] ボタンがのみが有効となったダイアログを表示します。
  if (SetupType ("", "", "", TYPICAL, 0) = BACK) then
    // [ 戻る ] ボタンが押された場合、[ 次へ ] ボタンが有効になります。
    Enable (NEXTBUTTON);
    goto start;
  endif;

end;
```

Do

Do 関数は、現在定義されている EXIT ハンドラーと HELP ハンドラーを実行します。この関数を使用すると、通常はユーザーが F1 キー (ヘルプ)、または [キャンセル] ボタン (終了) を押したときだけ起動するこれらのハンドラーに関するコントロールの幅が広がります。**Do** 関数を使用すると、カスタム ダイアログイベントやビルトイン ダイアログからのユーザー入力に対して、EXIT ハンドラーや HELP ハンドラーを実行することができます。さらに、スクリプトの開発中に、**Do** 関数を使用して EXIT ハンドラーおよび HELP ハンドラーの機能性をテストすることもできます。

Do 関数は、待機中の自己登録ファイルの登録にも使用できます。ファイルは自己登録ファイルをインストールするための「バッチメソッド」を使用して、登録のために待機状態にされます。Do(SELFREGISTRATIONPROCESS) を呼び出すと、インストールは待機中のすべてのファイルの自己登録を実行します。この処理は、いずれかのファイルが自己登録に失敗した場合も続行されます。(FeatureTransferData を呼び出すと、ファイルがインストールされた後、**FeatureTransferData** 呼び出しが戻る前に Do(SELFREGISTRATIONPROCESS) が自動的に呼び出されます。イベント指向のスクリプトを使用すると、**OnMoveData** イベント ハンドラー関数のデフォルトコードによって **FeatureTransferData** 関数が呼び出されます。

構文

```
Do (nOperation);
```

パラメーター

テーブル 160 - Do のパラメーター

パラメーター	説明
nOperation	実行するオペレーションの種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • EXIT—Exit 操作を開始します。EXIT ハンドラーを定義しなかった場合は、デフォルトの [Exit] ダイアログが表示されます。 • HELP —Help 操作を開始します。HELP ハンドラー を定義しなかった場合、関数は動作しません。 • SELFREGISTRATIONPROCESS— 登録待機中のすべての自己登録ファイルを登録します。

戻り値

テーブル 161 - Do の戻り値

戻り値	説明
0	**Do** 関数が指定された操作を無事に開始しました。
< 0	**Do** 関数は指定された操作を開始できませんでした。

追加情報

- **Do** 関数を使用すると、ユーザーが F1 キーを押さなくても、現在定義されている HELP ハンドラーや EXIT ハンドラーを実行できます。さらに、**Do** 関数は、HELP や EXIT ハンドラー ラベルの呼び出しに使用される

goto ステートメントよりもさらに多様な可能性をもたらします。つまり、goto ステートメントは使用できる状況が限られていますが、**Do** 関数は事実上、いつでも呼び出すことができます。デフォルトとカスタムの HELP および EXIT ハンドラーの詳細は、「HandlerEx」を参照してください。

- **Do** が何らかの原因で処理に失敗すると、-1 が返されます。自己登録に失敗したファイルの名前は、InstallScript システム変数 ERRORFILENAME に保管されます。この際、ファイルとファイルの間は、セミコロン (;) で区切られます。

Do の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* Do 関数のデモンストレーションを行います。
*
* このスクリプトは Do を呼び出して HELP と EXIT ハンドラーをテストします。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*      ターゲットシステムの有効なヘルプ ファイルを
*      参照するように設定します。
*
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

#define HELPFILE WINDIR^"Help¥¥Windows.chm"

    export prototype ExFn_Do(HWND);

function ExFn_Do(hMSI)
begin

    // exit ハンドラーをインストールします。
    HandlerEx (EXIT, Exit_Handler);

    // help ハンドラーをインストールします。
    HandlerEx (HELP, Help_Handler);

    // 継続ループ — またはユーザーが中止するまでループを実行します。
    while (TRUE)
        if (AskYesNo ("ヘルプを参照しますか ?", NO) = YES) then
            // help ハンドラーを実行します。
            Do (HELP);
        endif;

        // exit ハンドラーを実行します。
        Do (EXIT);
    endwhile;
```

```
// exit ハンドラー
Exit_Handler:

    // 終了の確認をとります。
    if (AskYesNo (" 終了してもよろしいですか ?", NO) = YES) then
        abort;
    else
        // 確実でない場合は続行します。
        return;
    endif;

// help ハンドラー
Help_Handler:

    // ヘルプを表示します。
    LaunchApplication (HELPFILE, "", "", SW_SHOW, INFINITE, LAAW_OPTION_WAIT);
    return;

end;
```

DoInstall

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

DoInstall 関数は、有効なセットアップ実行可能ファイル (.exe) を持つ別の InstallShield を起動します。2 番目のインストーラーは、この関数が呼び出されると直ちに実行されます。3 番目のパラメーターである nOptions は、起動したアプリケーションが終了するまでインストールを続行しないで、待機させるかどうかを含んだ様々なオプションを指定します。

***メモ・DoInstall** を直接使用して、.msi ファイルを起動することはできません。さらに、**DoInstall** では InstallShield のインストールに特有のコマンドライン パラメーターが追加されるため、**DoInstall** を使用して InstallShield 以外のインストールを起動することは避けてください。InstallShield ではないインストールを起動するには、**LaunchAppAndWait** 関数を利用します。*

構文

```
DoInstall ( byval string szSetupExe, byval string szCmdLine, byval number nOptions );
```

パラメーター

テーブル 162 ・ DoInstall のパラメーター

パラメーター	説明
szSetupExe	起動するセットアップ実行可能ファイルのドライブ指定や完全なパスなど、完全修飾名を指定します。 また、起動するインストールのコンパイルされたスクリプト ファイルを指定することもできます。InstallScript MSI インストールでは、インストールの .msi パッケージも指定できます。**DoInstall** を使用するには、インストーラーに **Setup.exe** という名前の有効なセットアップ実行可能ファイルが含まれているか、「Additional Information」に説明するような別名が含まれている必要があります。
szCmdLine	起動されたインストーラーのコマンドラインを指定します。このパラメーターには、任意の有効な InstallShield スタートアップコマンドラインを指定できます。**DoInstall** では、指定される nOptions に応じて一部のコマンドライン スイッチが自動的に追加されます。 ***メモ・*** *前のバージョンの InstallShield と異なり、コマンドライン パラメーターを子の InstallScript MSI インストーラーに渡す場合は、これらのオプションを /z パラメーターを通じて渡す必要があります (インストールを直接起動するときと同じ)。*

テーブル 162 • DoInstall のパラメーター (続き)

パラメーター	説明
nOptions	インストーラーを起動するオプションを指定します。次のような有効な LaunchApplication オプションを指定できます。 • LAAW_OPTION_NOWAIT • LAAW_OPTION_WAIT **DoInstall** にのみ適用される追加のオプションを指定することもできます。 • DOINSTALL_OPTION_NOHIDEPROGRESS— 初期化ユーザー インターフェイス全体 (初回の進行状況ダイアログとスプラッシュ画面を含む) を子のインストーラーに表示するよう指定します。 このオプションを指定しないと、**DoInstall** は /hide_progress オプションを使用して、子のインストーラーの初期化ユーザー インターフェイスを自動的に非表示にします。 • DOINSTALL_OPTION_NOHIDESPLASH— 起動している子のインストーラーに、スプラッシュ画面があればそれを表示するよう指定します。 • DOINSTALL_OPTION_NOLANGSWITCH — /l スイッチが指定されないことを指定します。 デフォルトでは、**DoInstall** は /l スイッチを szCmdLine に追加して、起動したインストーラーが起動インストーラーと同じ言語で実行されるようにします。子のインストーラーで親インストーラーの実行言語がサポートされていない場合 (**Is(LANGUAGE_SUPPORTED)** を呼び出して判断)、/l スイッチは追加されません。 • DOINSTALL_OPTION_NOSETBATCHINSTALL —**DoInstall** が、子のインストーラーが再起動が必要なアクションを実行するかどうかを判断するのに、LAAW_OPTION_SET_BATCH_INSTALL オプションを使用しないよう指定します。そのため、親インストーラーの BATCH_INSTALL は、子のインストーラーのアクションに関係なく、**DoInstall** によって変更されることはありません。 デフォルトでは、DoInstall は自動的に LAAW_OPTION_SET_BATCH_INSTALL オプションを使用します。 ビット単位 OR 演算子 (|) を使用して、これらの定数を組み合わせることができます。DOINSTALL_OPTION_NOSETBATCHINSTALL と LAAW_OPTION_SET_BATCH_INSTALL を組み合わせると、予期せぬ結果が生じることがあります。

戻り値

テーブル 163 ・ DoInstall の戻り値

戻り値	説明
0	**DoInstall** が LAAW_OPTION_WAIT を 3 番目の引数として呼び出された場合、**DoInstall** によって起動されたインストーラーは正常に終了されます。**DoInstall** が LAAW_OPTION_NOWAIT を 3 番目の引数として呼び出された場合は、常に 0 が戻されます。**DoInstall** 関数に従うステートメントで、呼び出しインストーラー内のコントロールが復帰します。
ISERR_SETUP_CANCELED (0x80042000)	LAAW_OPTION WAIT を 3 番目の引数として **DoInstall** によって起動された InstallScript インストーラーが、abort キーワードを出して終了しました。これは通常、ユーザーが、インストーラーをキャンセルしたことを示します。
-3	LAAW_OPTION WAIT を 3 番目の引数として **DoInstall** によって起動された InstallScript MSI インストーラーが、abort キーワードを出して終了しました。これは通常、ユーザーが、インストーラーをキャンセルしたことを示します。
その他すべての負の値	特定できないエラーが発生しました。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

Additional Information

- デフォルトでは、**DoInstall** 関数は、指定されたセットアップ実行可能ファイルを起動しようと試みます。実行可能ファイルではないファイル名が szSetupExe で指定されると、この関数は指定のフォルダーにある **Setup.exe** を起動しようと試みます。インストーラーの実行可能ファイルに、*Setup* 以外の名前が付いている場合、起動するインストーラーの **Setup.ini** ファイルは、LauncherName キーの [Startup] セクションで **Setup.exe** の新しい名前を持つ必要があります。

  プロジェクト・*InstallScript MSI プロジェクトの [リリース] ビューの製品構成で "* ***セットアップ ファイル名*** *" 設定に値を指定してから、リリースをビルドすると、InstallShield はファイル名の値を自動的に* **Setup.ini** *に追加します。InstallScript プロジェクトのセットアップランチャー実行可能ファイルの名前を変更した場合な、その名前を手動で* **Setup.ini** *ファイルに追加する必要があります。*

- インストールが、CD または DVD などのリムーバブル メディアから呼び出されたとき、Disk1 の **Setup.exe** ファイルは、インストール中、常に使用できる状態にあるとはかぎりません。(実行中に、**Setup.exe** が使用できなくなった場合、オペレーティング システムでプロンプトが表示され、エンドユーザーに正しいディスクを挿入するように求めてくることがあります。これにより、インストールが失敗することがあります。) したがって、この問題を回避するために、**Setup.exe** ファイルが Temp フォルダーにコピーされ、インストールがそこから再起動されます。元の **Setup.exe** は、ここで終了します。ただし、この処理が発生すると、**DoInstall** は、インストールが完了したと仮定して動作し、待機しません。

この問題を回避するため、子インストールを起動するときに、/clone_wait パラメーターを使用できます。その場合、起動されたインストールで、起動された元のプロセスが実行中のまま保たれ、親インストールは待機します。ただし、このワークアラウンドでは、**Setup.exe** を含む元の CD がインストールの途中で使用できない状態になる場合、問題が発生する場合があります。この例として、インストールの途中で一枚目の CD が使用できなくなる複数 CD のインストールがあげられます。

この問題を回避する唯一の方法は、起動されたプロセスの子プロセスの ID を判別するコードを追加して、子プロセスが完了するのを待機することです。

DoInstall の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* Doinstall 関数のデモンストレーションを行います。
*
* このスクリプト例では、DoInstall 関数を使用して、
* MessageBox のスクリプト例を実行します。
*
* メモ : この作業例を正しくすすめるため、
*      次の手順に従ってください :
*
*      1.2 番目のセットアップ プロジェクトを作成します。このセットアップは、
*          DoInstall 関数によって起動されます。このプロジェクトには
*          最新のビルド済みリリースが含まれます。
*
*      2. このセットアップのディスク 1 フォルダー中に
*          'Second' という名前の新しいフォルダーを作成します。
*
*      3.2 番目のセットアップから新しく作成した 'Second' フォルダーへ、
*          ディスク # フォルダーをコピーします。
*
*     これで、2 番目のセットアップが正常に起動されます。
*
¥*---------------------------------------------------------------------*/

#define SECOND_INSTALL_PATH SRCDIR ^ "Second¥¥Disk1"
#define SECOND_INSTALL_FILENAME "Setup.exe"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_DoInstall(HWND);

function ExFn_DoInstall(hMSI)
   NUMBER nReturn;
   STRING szTemp;
begin
```

```
    MessageBox ("2 番目のセットアップを起動しようとしています。", INFORMATION);

    //2 番目のセットアップを起動します。
    nReturn = DoInstall (SECOND_INSTALL_PATH ^ SECOND_INSTALL_FILENAME, "",
                         LAAW_OPTION_WAIT);

    if ( nReturn = 0) then
        //2 番目のセットアップの起動成功を報告します。
        MessageBox("2 番目のセットアップは正常に起動しました。", INFORMATION);
    else
        //2 番目のセットアップの起動失敗を報告します。
        SprintfBox(SEVERE, "", "DoInstall が %d のリターンコードで失敗しました。", nReturn);
    endif;

end;
```

DotNetCoCreateObject

***プロジェクト・DotNetCoCreateObject** 関数は次のプロジェクトの種類でサポートされています:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript カスタム アクションを含む基本の MSI*

DotNetCoCreateObject 関数は、アセンブリが COM 相互運用性のために登録されることなく .NET アセンブリの関数を呼び出します。この関数では、.NET アセンブリがロードされ実行される .NET アプリケーションのドメインを指定することができます。この関数を使用して作成された各オブジェクトは、単一の .NET アセンブリの単一クラスに関連付けられます。同じアセンブリの複数のクラスにアクセスする場合は、各クラスに別のオブジェクトを作成する必要があります。

***重要**・アセンブリを COM コンポーネントとして登録する必要はありませんが、アセンブリは COM 相互運用性と互換性を持つように構築する必要があります。Visual Studio .NET のバージョン 2003 以前を使用して作成されたアセンブリには、自動的にこの互換性が組み込まれています。ただし、Visual Studio 2005 を使用して作成したプロジェクトの場合、手動で該当するファイルに [assembly: ComVisible(true)] を指定する必要があります。*

構文

```
DotNetCoCreateObject ( byval string szAssemblyPathFile, byval string szAssemblyAndClassName, byval string szAppDomain );
```

パラメーター

テーブル 164 ・ DotNetCoCreateObject のパラメーター

パラメーター	説明
szAssemblyPathFile	適したクラスが含まれる NET アセンブリの完全パスとファイル名を指定します。
szAssemblyAndClassName	アセンブリとクラス名を指定します。 ***メモ・*** *アセンブリを、**CoCreateObject** 関数を使用して COM 相互運用性に登録した場合、この値は szProgId パラメーターと同じになります。*
szAppDomain	アセンブリをロードして実行する .NET アプリケーション ドメインを指定します。指定したアプリケーション ドメインがインストール プロセス中に存在しない場合、作成されます。存在する場合、その既存のアプリケーション ドメインが使用されます。**DotNetCoCreateObject** を同じ szAppDomain で複数回呼び出すと、これは True です。 このパラメーターにヌル文字列 ("") を指定するか、または **DotNetCoCreateObject** の代わりに **CoCreateObjectDotNet** を呼び出すと、アセンブリは、インストールが完了したあと、デフォルトのアプリケーション ドメインにロードされます。このため、.NET アセンブリ ファイルは、インストールが完了するまでロックされます。 .NET アプリケーションのドメインの詳細については MSDN ライブラリ の「NET Framework 開発者ガイド」をご覧ください。

戻り値

設定されたキーワードを使用して、OBJECT という変数タイプに割り当てられるリファレンス。

追加情報

- **DotNetCoCreateObject** 関数は、**CoCreateObjectDotNet** 関数に類似しています。唯一異なる点は、**DotNetCoCreateObject** と使用すると、ロードする .NET アプリケーション ドメインを指定できるという点です。このドメインで、アセンブリが実行されます。

  CoCreateObjectDotNet の場合、.NET アセンブリは、インストールが完了したあと、デフォルトのアプリケーションにロードされます。このため、.NET アセンブリ ファイルは、インストールが完了するまでロックされます。

- オブジェクト変数を NOTHING の値に設定するか、**CoCreateObject**、**CoCreateObjectDotNet**、**CoGetObject** または **DotNetCoCreateObject** 関数を使用してオブジェクトを再割り当てすると、任意のオブジェクト変数を解

放することができます。ただし、これによってオブジェクトが参照するライブラリが自動的にロード解除されるわけではありません。Windows API **CoFreeLibrary** を手動で呼び出してライブラリを解放する必要があります。そうしないとライブラリは、インストールが終了するまでロードされたままになります。詳細は、「COM オブジェクトを使用してインストールを拡張する」を参照してください。

- この関数は、オブジェクトを作成できない場合に例外をスローします。これは .NET Framework がシステムにインストールされていない、またはその他の理由で発生することがあります。この例外を処理するには、この関数の呼び出しを try...catch ブロックで括ってください。詳細については、「例外処理」を参照してください。

DotNetUnloadAppDomain

***プロジェクト・DotNetUnloadAppDomain** 関数は次のプロジェクトの種類でサポートされています:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript カスタム アクションを含む基本の MSI*

DotNetUnloadAppDomain 関数は、指定された .NET アプリケーション ドメインをアンロードし、現在ロードされているアセンブリをすべて指定されたアプリケーション ドメインにリリースします。

***メモ・アプリケーション ドメインがアンロードされると、DotNetCoCreateObject** で作成されたすべての .NET オブジェクトは無効になります。したがって、**DotNetUnloadAppDomain** を呼び出す前に、SET コマンドを使って、これらのオブジェクトを NOTHING に設定する必要があります。*

構文

```
DotNetUnloadAppDomain ( byval string szAppDomain );
```

パラメーター

テーブル 165 ・ DotNetUnloadAppDomain のパラメーター

パラメーター	説明
szAppDomain	アンロードされる .NET アプリケーション ドメインを指定します。 .NET アプリケーションのドメインの詳細については MSDN ライブラリ の「NET Framework 開発者ガイド」をご覧ください。

戻り値

テーブル 166 ・ DotNetUnloadAppDomain の戻り値

戻り値	説明
>= ISERR_SUCCESS	ドメインは正常にアンロードされました。
< ISERR_SUCESS	ドメインはアンロードされませんでした。

ビルトイン関数 (E-G)

カテゴリ別の関数一覧は、「カテゴリー別ビルトイン関数」を参照してください。

Enable

Enable 関数は、nConstant パラメーターで指定したユーザーインターフェイスオブジェクトやセットアップ機能をアクティブにします。

デフォルトで、インストールは背景なしで実行します。ウィンドウモードを有効にするには、BACKGROUND 定数と共に **Enable** を呼び出し、次に DEFWINDOWMODE または FULLWINDOWMODE と共に再び呼び出します。これらの定数は基本の MSI インストールでは使用できません。

メモ・スクリプトで Disable 関数を呼び出して [次へ] ボタンや [戻る] ボタンを無効にすると、関数呼び出しの後に表示されるすべてのダイアログ内のボタンが無効になります。[次へ] ボタンや [戻る] ボタン を再度有効にするには、対応する定数を使用して Enable 関数を呼び出す必要があります。

構文

```
Enable (nConstant);
```

パラメーター

テーブル 1 ・ Enable のパラメーター

パラメーター	説明
nConstant	有効にするユーザーインターフェースオブジェクトやオペレーション機能を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ BACKBUTTON – いくつかのビルトイン ダイアログに表示される [戻る] ボタンを有効にします。[戻る] ボタンは、デフォルトで有効になり、**Disable** 関数の呼び出しで無効になります。 ・ BACKGROUND – インストールがウィンドウ モードの際、メイン背景ウィンドウを表示しますインストールがデフォルト モードを表す全画面表示モードの場合、この定数は無効になります。ウィンドウモードを有効にするには、定数 DEFWINDOWMODE または FULLWINDOWMODE を使用して **Enable** を呼び出す必要があります。 ・ CANCELBUTTON – いくつかのビルトイン ダイアログとステータス ダイアログで表示される [キャンセル] ボタンを有効にします。 ・ DEFWINDOWMODE – メインの背景ウィンドウを構成して、タイトルバーのある通常のウィンドウにします。背景ウィンドウが有効な場合、外観が直ちに変更されます。背景ウィンドウが有効でない場合、定数の BACKGROUND を使用して **Enable** を呼び出すまでスクリーンの表示は変更されません。 ・ DIALOGCACHE – ダイアログのキャッシュ メカニズムを有効にします。このメカニズムは、表示が 1 つのダイアログから別のダイアログに変わるときに現れるスクリーン フラッシュを消去します。このスクリーン フラッシュは、インストールのタイトルバーがウィンドウ モードで実行されている場合に最もよく認識できます。ダイアログのキャッシュメカニズムは、[戻る] ボタンや [次へ] ボタンがあるダイアログ ボックスに対してのみ有効です。ダイアログのキャッシュはデフォルトでは有効です。 ***メモ・** DIALOGCACHE は、[次へ] ボタンや [戻る] ボタンがないダイアログでは無効です。* ・ FULLWINDOWMODE – メインの背景ウィンドウを構成して、タイトルバーのある最大のウィンドウにします。背景ウィンドウが有効の場合、表示がただちに変更されます。背景ウィンドウが有効でない場合、定数の BACKGROUND を使用して **Enable** を呼び出すまでスクリーンの表示は変更されません。 ・ HOURGLASS – マウスカーソルを、″ ビジー ″ 状態を表すカーソルである砂時計に、デフォルトで変わるようにします。 ・ INDVFILESTATUS – **FeatureMoveData**、**CopyFile**、または **XCopyFile** が呼び出され、プログレス インジケーターが有効になったときに、各ファイルの完全修飾名を転送されたとおりに (プログレス インジケーターの 2 行目に) 表示できるようにします。

テーブル 1 ・ Enable のパラメーター (続き)

パラメーター	説明
nConstant (続き)	・ LOGGING – アンインストール情報に関するログ機能を有効にします。ログ機能が有効な場合、InstallScript エンジンがアンインストールをログに記録した結果が、アンインストール ログ ファイルに残され、またアンインストール作業中に元に戻されます。 デフォルトで、ログ記録は自動的に有効となります。ログ記録を無効にする必要がある場合、アンインストール向けにログ記録しない処理を行う直前に LOGGING 定数を使って **Disable** を呼び出す方法が推奨されます。その後、LOGGING 定数を使って **Enable** を呼び出し、ログ記録を再有効化します。 ログ記録に関する詳細については、「アンインストール用にログされた InstallScript 関数」を参照してください。 ・ NEXTBUTTON – ビルトインダイアログに表示される [次へ] ボタンを有効にします。デフォルトでは、ほとんどのダイアログでの [次へ] ボタンは有効になっています。 ***メモ・** BACKBUTTON 定数を使って **Enable** を呼び出しても、Next ボタンを内部的に有効化 / 無効化する **SdCustomerInformation**、**SdCustomerInformationEx**、**SdRegisterUser**、または **SdRegisterUserEx** ダイアログでは効果がありません。* ・ PCRESTORE – デフォルトで有効になっている [システム回復互換性] を有効にします。 ・ REGISTRYFUNCTIONS_USETEXTSUBS – レジストリ関数に渡された文字列のテキスト置換を有効にします。これはデフォルトで有効になっています。 ・ SELFREGISTERBATCH – XCopyFile および SELFREGISTER 定数を使ってコピーしたファイルの登録の「バッチメソッド」を使用するかどうかを指定します。バッチメソッドは、デフォルトでは有効です。 バッチ メソッドが無効な場合、**XCopyFile** と SELFREGISTER 定数を使ってコピーしたファイルはすぐに登録されます。batch メソッドが有効な場合、データ転送が行われるまで登録は遅延されます。 ・ SERVICE_DIFX_32 – 32 ビット プラットフォームの DIFx サポートを有効にします。 ・ SERVICE_DIFX_AMD64 – AMD 64 ビット プラットフォームの DIFx サポートを有効にします。 ・ SERVICE_DIFX_IA64 – Itanium 64 ビット プラットフォームの DIFx サポートを有効にします。 ・ STATUS – 進行状況インジケーター (ステータスバー) の表示を有効にします。 ・ STATUSBBRD – ビルボードを含む進行状況ダイアログの表示を有効にします。 ・ STATUSDLG – ダイアログスタイルの進行状況インジケーター (ステータスバー) の表示を有効にします。 ・ STATUSEX – デフォルト [セットアップ ステータス] ダイアログの表示を無効にします。 ・ STATUSOLD – [キャンセル] ボタンがない古いスタイルの進行状況インジケーター (ステータスバー) の表示を有効にします。

テーブル 1 ・ Enable のパラメーター (続き)

パラメーター	説明
nConstant (続き)	・ UPDATE_SERVICE_INSTALL– この定数は、古い形式です。 ・ USE_LOADED_SKIN – ダイアログ スキンと共に動作しないカスタム ダイアログ (例えば、標準サイズではないダイアログ) での利用を目的とします。ビルトイン ダイアログとの利用はお勧めしません。[リリース] ビューでリリースの [ビルド] タブにある " スキンの指定 " 設定で指定されたスキンの使用を有効にします。(この設定で <**スキンを利用しない**> オプションを選択した場合、この定数は効果を持ちません。) スキンを指定した場合はデフォルトで表示されます。スキンを使って表示することができないダイアログを表示する関数によって、Disable(USE_LOADED_SKIN); および Enable(USE_LOADED_SKIN); が内部で呼び出されます。スキンを有効にすることができるのは、スクリプトで Enable(USE_LOADED_SKIN); を呼び出した後に表示されるダイアログのみです。(無効にした後) カスタム ダイアログでのスキンの利用を有効にするには、**WaitOnDialog** を呼び出す前に Enable(USE_LOADED_SKIN); を呼び出さなくてはなりません。 ***プロジェクト ・** USE_LOADED_SKIN 定数は、InstallScript プロジェクトで使用できます。* ・ WOW64FSREDIRECTION – 64 ビット Windows のファイル システム リダイレクトを有効化します。場合によって、これは、ファイルを WINSYSDIR64 にインストールした後に行う必要があります。詳しくは、「64 ビット オペレーティング システムを InstallScript インストールでターゲットにする」を参照してください。

戻り値

テーブル 2 ・ Enable の戻り値

戻り値	説明
0	0 関数が、nConstant のパラメーターで指定したユーザーインターフェイスオブジェクトやセットアップ機能を有効にしたことを示します。
< 0	0 関数が、nConstant のパラメーターで指定したユーザーインターフェイスオブジェクトやセットアップ機能を有効にできなかったことを示します。

Enable の例

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* Disable 関数と Enable 関数のデモンストレーションを行います。
*
* このスクリプトは 2 つのダイアログを表示します。最初のボックスでは、
*[ 戻る ] ボタンが無効になっています。2 番目のボックスでは [ 次へ ] ボタンが
* 無効で、[ 戻る ] ボタンが有効になっています。
*
¥*---------------------------------------------------------------------*/
```

```
// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_Enable(HWND);

function ExFn_Enable(hMSI)
begin

start:

   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   // 次は [ 戻る ] ボタンが無効となったダイアログを表示します。
   SetupType ("", "", "", TYPICAL, 0);

   // [ 戻る ] ボタンを有効にします、
   Enable(BACKBUTTON);

   // [ 次へ ] ボタンが無効になっています。
   Disable (NEXTBUTTON);

   // 次は [ 戻る ] ボタンがのみが有効となったダイアログを表示します。
   if (SetupType ("", "", "", TYPICAL, 0) = BACK) then
      // [ 戻る ] ボタンが押された場合、[ 次へ ] ボタンが有効になります。
      Enable (NEXTBUTTON);
      goto start;
   endif;

end;
```

EndCurrentDialog

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

EndCurrentDialog 関数は、EndDialog を呼び出して現在表示されているダイアログを閉じます。ダイアログを削除し、ダイアログを閉じる処理を開始します。

構文

```
EndCurrentDialog ( );
```

パラメーター

なし

戻り値

テーブル 3 - EndCurrentDialog の戻り値

戻り値	説明
>= ISERR_SUCCESS	EndCurrentDialog がダイアログを閉じました。
< ISERR_SUCCESS	現在表示中のダイアログはありません。

EndDialog

EndDialog 関数はカスタムダイアログを閉じます。ダイアログを削除し、ダイアログを閉じる処理を開始します。次の条件が存在する場合、EndDialog を使います :

- [次へ] またはそれに相当するボタンを押したとき。
- [キャンセル] またはそれに相当するボタンを押したとき。
- [閉じる] システムメニューオプションが選択されたとき。このアクションは DLG_CLOSE メッセージを送ります。
- その他、ユーザーがダイアログ操作を終了したとき。

EndDialog を呼び出してカスタムボックスを終了したあと、ReleaseDialog 関数を呼び出してカスタムダイアログボックに関連するメモリを開放します。

メモ - *EndDialog への呼び出しで閉じたカスタムダイアログは、WaitOnDialog を呼び出して再び表示することができます。ただし、ReleaseDialog を呼び出してメモリからダイアログを削除しなかったことを前提とします。WaitOnDialog を呼び出して以前スクリプトで開閉したダイアログを開いた時、新しいハンドルを取得するためには CmdGetHwndDlg をもう一度呼び出さなくてはなりません。古いハンドルは既に無効です。*

構文

```
EndDialog (szDialogName);
```

パラメーター

テーブル 4 - EndDialog のパラメーター

パラメーター	説明
szDialogName	閉じるダイアログの名前を指定します。

戻り値

テーブル 5 - EndDialog の戻り値

戻り値	説明
0	EndDialog がダイアログを閉じました。
< 0	EndDialog はダイアログを閉じることができませんでした。

EndDialog の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* DefineDialog 関数、EndDialog 関数、そして ReleaseDialog 関数の
* デモンストレーションを行います。
*
* このスクリプトは ビットマップを表示するシンプルなカスタムダイアログを
* 開きます。ダイアログは次の 3 つのボタンで閉じることが
* [ 戻る ]、[ 次へ ]、および [ キャンセル ]。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SdAskOptions が表示する
* InstallShield 標準ダイアログです。このダイアログは
* インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
* カスタム ダイアログとしてスクリプトで利用することが
* 表示されます。
*
* このダイアログをカスタムダイアログとして利用するためには、
* DefineDialog を呼び出してそれをスクリプトで定義します。その後
* WaitOnDialog を呼び出してダイアログを表示します。イベントが
* ダイアログの処理を終了するとき、それを閉じるために EndDialog が
* 表示されます。次いで、ReleaseDialog への呼び出しによって、
* メモリからダイアログがリリースされます。
*
¥*---------------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
```

```
#define RES_DIALOG_ID     12027   // ダイアログ自身の ID
#define RES_PBUT_NEXT       1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL     9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK      12   //[ 戻る ] ボタンの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_EndDialog(HWND);

function ExFn_EndDialog(hMSI)
    STRING  szDialogName, szDLLName, szDialog;
    NUMBER  nDialog, nResult, nCmdValue;
    BOOL    bDone;
    HWND    hInstance, hwndParent, hwndDlg;
begin

    // DefineDialog への最初のパラメーターとして渡すダイアログの
    // 名前を定義します。
    szDialogName = "ExampleDialog";

    // DefineDialog の 2 番目のパラメーターは 0 となります。
    // これは .dll ファイルが _isres.dll の中にあるためです。
    hInstance = 0;

    // DefineDialog の 3 番目のパラメーターは ヌルとなります。インストールは
    // _isuser.dll と _isres.dll にあるダイアログを検索します。
    szDLLName = "";

    // DefineDialog の 5 番目のパラメーターは 0 となります。なぜなら、
    // 4 番目のパラメーターにある ID によってダイアログが認識されるためです。
    szDialog = "";

    // この値は保存され、0 でなくてはなりません。
    hwndParent = 0;

    // ダイアログを定義します。インストールのメインウィンドウがダイアログ ボックスを保有します
    // ( パラメーター 7 内の HWND_INSTALL で表示されます ) を保有します。
    nResult  = DefineDialog (szDialogName, hInstance, szDLLName,
                        RES_DIALOG_ID, szDialog, hwndParent,
                        HWND_INSTALL, DLG_MSG_STANDARD|DLG_CENTERED);

    // エラーをチェックします。
    if (nResult < 0) then
        MessageBox (" ダイアログを定義中にエラーが発生しました。", SEVERE);
        bDone = TRUE;
        abort;
    endif;

   // while ループを制御するのに使われるインジケーターを初期化します。
   bDone = FALSE;

   // 完了するまでループします。
   repeat

        // ダイアログを表示して次のダイアログ イベントを戻します。
        nCmdValue = WaitOnDialog(szDialogName);

        // イベントに応答します。
```

```
    switch (nCmdValue)
        case DLG_CLOSE:
            // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
            Do (EXIT);
        case DLG_ERR:
            MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
            abort;
        case DLG_INIT:
            // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
            // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
            // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
            // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
            hwndDlg = CmdGetHwndDlg (szDialogName);
            SdGeneralInit(szDialogName, hwndDlg, 0, "");
        case RES_PBUT_CANCEL:
            // ユーザーが [ キャンセル ] ボタンをクリックしました。
            Do (EXIT);
        case RES_PBUT_NEXT:
            bDone = TRUE;
        case RES_PBUT_BACK:
            bDone = TRUE;
    endswitch;

    until bDone;

    // ダイアログを閉じます
    EndDialog (szDialogName);

    // メモリからダイアログを解放します。
    ReleaseDialog (szDialogName);

end;
```

EnterDisk

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

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

EnterDisk 関数は、エンドユーザーに対して次のディスクを挿入するよう促すメッセージ ボックスを表示します。システム変数 SRCDIR にはデフォルトのパスが含まれ、このパスはメッセージ ボックスに表示されます。エンドユーザーは、新しいパスを入力して [OK] をクリックすることでデフォルトパスを変更することができます。

EnterDisk は、szTagFile で指定したタグファイル用のディスクを検索して、正しいディスクを認識します。ディスクにタグ ファイルが含まれていない場合、関数は **EnterDiskError** 関数を呼び出して、エンド ユーザーに正しいディスクを挿入するようにプロンプトするエラー メッセージ ボックスを表示します。ビルド時に、InstallShield はディスク イメージ フォルダー内でタグ ファイルを自動作成しません。タグファイルを使用するには、ディスク イメージフォルダーを作成してから、そのフォルダーにタグファイルを追加してください。

メモ・EnterDisk 関数と一緒に PlaceWindow 関数を使うことはできません。背景ウィンドウモードを有効にしない限り、デフォルトではデスクトップ中央にダイアログ ボックスが表示されます。インストールがウィンドウ モードの場合、背景ウィンドウの中央にメッセージ ボックスが表示されます。

デフォルトのタイトルは、「セットアップには次のディスクが必要です」です。このタイトルを変更するには、EnterDisk の前に、SetDialogTitle を呼び出してください。

構文

EnterDisk (szMsg, szTagFile);

パラメーター

テーブル 6・EnterDisk のパラメーター

パラメーター	説明
szMsg	適切なディスクを挿入するようエンド ユーザーにプロンプトするメッセージを指定します。
szTagFile	タグファイルの名前を指定します。**EnterDisk** は挿入されたディスクでこのファイルを検索します。ファイルが検出されない場合、関数は **EnterDiskError** 関数を呼び出して、ユーザーに正しいディスクの挿入を求めるメッセージを表示します。このパラメーターにヌル文字列 ("") を入力すると、関数は正しいディスクが使用されたものと認識してファイルを検索しません。

戻り値

テーブル 7・EnterDisk の戻り値

戻り値	説明
OK (1)	ユーザーが、[OK] ボタンをクリックしたことを示します。
< 0	特定できないエラーが発生したことを示します。

追加情報

EnterDisk 関数によって表示されるダイアログは、スキンを使って表示することはできません。スキンの指定に関わらず、同じように表示されます。

EnterDisk の例

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します:*

- *基本の MSI*
- *InstallScript*

- *InstallScript MSI*

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* この例では EnterDisk 関数のデモンストレーションを行います。
*
* ディスクを挿入、またはパスを指定するようユーザーに問いあわせるため、EnterDisk
* が呼び出されます。そして、EnterDisk はその場所にあるタグファイルを
* 検索します。
*
¥*---------------------------------------------------------*/

#include "ifx.h"

    export prototype ExFn_EnterDisk(HWND);

function ExFn_EnterDisk(hMSI)
    STRING szMsg, szTagFile;
begin

    // EnterDisk を呼び出すパラメーターをセットアップします。
    szMsg     = " ディスク 2 を挿入してください ";
    szTagFile = "ISExampl.txt";

    // EnterDisk ダイアログを表示します。
    EnterDisk (szMsg, szTagFile);

end;
```

EnterDiskError

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

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

EnterDiskError 関数は、指定されたパスおよびファイルが存在するかどうかを確認します。指定されたパスにファイルが存在しない場合、関数は適切なエラーを表示してから、指定されたファイルが存在するかどうかに従って success または failure を返します。

ヒント・EnterDiskError は、特定のファイル無しで特定のパスの存在を確認することもできます。

構文

EnterDiskError (byval string szPath, byval string szFile);

パラメーター

テーブル 8 · EnterDiskError のパラメーター

パラメーター	説明
szPath	確認するパスを指定します。
szFile	確認するファイルの名前を指定します。 **EnterDisk** が特定のファイルではなく、パスのみを確認する場合、ヌル文字列 ("") を指定します。

戻り値

テーブル 9 · EnterDiskError の戻り値

戻り値	説明
>= ISERR_SUCCESS	指定されたパスとファイルが存在します。
< ISERR_SUCCESS	指定されたパスとファイルが存在しません。

追加情報

EnterDiskError 関数は一般的に **EnterDisk** 関数によって内部的に呼び出されるため、標準の **MessageBox** サイレント モード処理以外のサイレント モード処理は一切含まれていません。

デフォルトで、ダイアログは発生するエラーに関わらず、インストールと同じエラー メッセージとデフォルトのエラー ボックス タイトルを表示します。この動作は、nErrorId を以下のように設定して **SetErrorTitle** および **SetErrorMsg** 関数を呼び出して変更できます :

- ERR_BOX_DISKID – 指定されたディスクが存在しない時に表示されるタイトルまたはメッセージをカスタマイズします。
- ERR_BOX_BADPATH – 指定されたパスが存在しない時に表示されるタイトルまたはメッセージをカスタマイズします。
- ERR_BOX_BADTAGFILE – 指定されたファイルが存在しない時に表示されるタイトルまたはメッセージをカスタマイズします。(szFile にヌル文字列 ("") を指定すると、エラーが確認されず、メッセージも表示されません。)

EnterLoginInfo

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

- *InstallScript*

- *InstallScript MSI*

EnterLoginInfo 関数は、エンド ユーザーがユーザー名とパスワードを指定できるダイアログを表示します。ダイアログは、指定された情報を検証または使用しません。また、ダイアログが指定された情報のエラーをチェックすることもありません。

EnterLoginInfo ダイアログは、一般的にエンド ユーザーがネットワーク ユーザー名とパスワードを指定するために使用されます。**SdLogonUserInformation** ダイアログは **EnterLoginInfo** ダイアログと似ていますが、**SdLogonUserInformation** ダイアログは追加オプションを提供します。SQL Server にログインするために、インストールが情報を取得する必要がある場合、**SQLServerLogin** は追加機能を提供します。

構文

EnterLoginInfo (byval string szMsg, byref string svUser, byref string svPassword);

パラメーター

テーブル 10 · EnterLoginInfo のパラメーター

パラメーター	説明
szMsg	ダイアログに表示するスタティック テキストを指定します。デフォルトのテキストを表示するには、このパラメーターでヌル文字列 ("") を渡します。
svUser	関数が呼び出されたときに [ユーザー名] ボックスにあらかじめ表示されるデフォルト値を指定し、関数が戻るときにエンド ユーザーが指定した文字列を指定します。 ダイアログの [ユーザー名] ボックスに入力可能な最大文字数は 255 文字です。
svPassword	関数が呼び出されたときに [パスワード] ボックスにあらかじめ表示されるデフォルト値を指定し、関数が戻るときにエンド ユーザーが指定した文字列を指定します。 ダイアログの [パスワード] ボックスに入力可能な最大文字数は 255 文字です。

戻り値

テーブル 11 · EnterLoginInfo も戻り値

戻り値	説明
>= ISERR_SUCCESS	関数は成功しました。
< ISERR_SUCCESS	関数は成功しませんでした。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

EnterPassword

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

EnterPassword 関数はエンドユーザーに対してパスワードを問い合わせるダイアログを表示します。エンドユーザーが編集ボックス内に入力する文字はアスタリスク (*) として表示されます。

メモ・戻されたパスワードを確認するには、OnCheckMediaPassword イベントハンドラー関数のデフォルトコードの処理と同様に FeatureValidate を呼び出します。

構文

```
EnterPassword ( szMsg, szDefault, svResult );
```

パラメーター

テーブル 12・EnterPassword のパラメーター

パラメーター	説明
szMsg	ダイアログに表示するメッセージを指定します。このテキストはスタティック コントロールとみなされます。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szDefault	最初に編集ボックスに表示されるテキストを指定します。
svResult	エンドユーザーが編集ボックスに入力するテキストを返します。

戻り値

テーブル 13・EnterPassword の戻り値

戻り値	説明
NEXT	ユーザーが [次へ] ボタンをクリックしたことを示します。
BACK	ユーザーが、[戻る] ボタンをクリックしたことを示します。
< ISERR_SUCCESS	ダイアログが表示されなかったことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

ExistsDir

ExistsDir 関数はターゲットシステム上またはインターネットで指定されたディレクトリの存在を確認します。

構文

ExistsDir (szPath);

パラメーター

テーブル 14 ・ ExistsDir のパラメーター

パラメーター	説明
szPath	ターゲットシステム上で検索するディレクトリの完全修飾パスを指定する、またはインターネット上で検索するディレクトリの有効な URL (Uniform Resource Locator) を指定します。URL が有効かどうかを確認するには、Is(VALID_PATH, szURL) を呼び出します。

戻り値

テーブル 15 ・ ExistsDir の戻り値

戻り値	説明
EXISTS (0)	指定したディレクトリが存在することを示します。
NOTEXISTS (-1)	指定したディレクトリが存在しないことを示します。
その他すべての負の値	関数が指定されたプログラムフォルダーが存在するかどうかを判断できなかったことを示します。大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

ExistsDir の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ExistsDir 関数のデモンストレーションを行います。
*
* ユーザーからディレクトリ名を取得するため AskPath が呼び出されます。
* そしてディレクトリの存在確認のため ExistsDir が呼び出され
* ます。
*
```

```
¥*---------------------------------------------------------------*/

#define TITLE_TEXT "ExistsDir の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ExistsDir(HWND);

function ExFn_ExistsDir(hMSI)
   STRING svPath;
begin

   // 作成するパスを取得します。
   AskPath (" パスを入力してください :", "", svPath);

   // ディレクトリの存在を確認します。
   if (ExistsDir (svPath) = EXISTS) then
      SprintfBox (INFORMATION, TITLE_TEXT, "%s は既に存在します。.", svPath);
   else
      SprintfBox (INFORMATION, TITLE_TEXT, "%s は存在しません ", svPath);
   endif;

end;
```

ExistsDisk

ExistsDisk 関数はターゲットシステム上で指定されたディスクドライブの存在を確認します。

構文

```
ExistsDisk ( szDisk );
```

パラメーター

テーブル 16 ・ ExistsDisk のパラメーター

パラメーター	説明
szDisk	ディスクドライブの名前を指定します。

戻り値

テーブル 17 ・ ExistsDisk の戻り値

戻り値	説明
EXISTS (0)	ターゲットシステムに指定したドライブが存在することを示します。
NOTEXISTS (-1)	ターゲットシステムに指定したディレクトリが存在しないことを示します。

ExistsDisk の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ExistsDisk 関数のデモンストレーションを行います。
*
* ユーザーからディスクドライブ名を取得するため AskText が呼び出されます。
* そしてドライブの存在確認のため ExistsDir が呼び出され
* ます。
*
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "ExistsDisk の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ExistsDisk(HWND);

function ExFn_ExistsDisk(hMSI)
  STRING svDrive;
begin

  if (AskText (" ディスクドライブの名前を入力してください。", "C", svDrive) = NEXT) then
    // 指定されたドライブの存在を確認します。
    if (ExistsDisk (svDrive) = EXISTS) then
      SprintfBox (INFORMATION, TITLE_TEXT, " ドライブ %s がありました。", svDrive);
    else
```

```
            SprintfBox (INFORMATION, TITLE_TEXT, " ドライブ %s は存在しません。",
svDrive);
        endif;
    endif;

end;
```

EzBatchAddPath

EzBatchAddPath 関数は、PATH コマンドの検索パスまたは環境変数に割り当てる値にパスを追加して、デフォルトのバッチファイルを変更します。BatchSetFileName への呼び出しで変更されない限り、デフォルト バッチ ファイルは Autoexec.bat ファイルで、起動シーケンス中にシステムによって実行されます。デフォルトバッチファイルの完全修飾ファイル名を決定するには、BatchGetFileName を呼び出します。EzBatchAddPath が使用するバッチファイルの名前を変更するには、BatchSetFileName を呼び出します。

注意・EzBatchAddPath は変更するファイルのバックアップコピーを作成しません。

EzBatchAddPath は、デフォルトバッチファイルが非表示または読み取り専用の場合に失敗することがあります。

メモ・簡易バッチファイル関数を拡張バッチファイル関数と同時に使用しないでください。BatchFileLoad を呼び出してメモリへバッチファイルをロードした後、BatchFileSave を呼び出してファイルを保存するまで簡易バッチファイル関数を呼び出すことはできません。

構文

EzBatchAddPath (szKey, szPath, szRefDir, nPosition);

パラメーター

テーブル 18 · EzBatchAddPath のパラメーター

パラメーター	説明
szKey	変更する環境変数の名前を指定します。例えば、PATH ステートメントを変更するにはここへ "PATH" を指定します。指定した環境変数がデフォルトバッチファイルで検出されなかった場合、SET ステートメントすべてがその環境変数に作成され、ファイルに挿入されます。 ***注意 ·*** *EzBatchAddPath 関数は、長いファイル名をサポートしません。EzBatchAddPath に渡す前に、LongPathToShortPath を呼び出して長いパスをそれに相当する短いパスに変換します。長いファイル名の詳細については、「長いファイル名」を参照して下さい。*
szPath	環境変数の現在の値に追加するパスを指定します。セミコロン検索パス内の別のパスと分割するため、区切りとなるセミコロンが挿入されます。
szRefDir	szPath で指定した新しいパスの追加に関連する参照キー (パス) を指定します。これがヌル文字列 ("") の場合、nPosition の値に従って検索パスの始まりまたは終わりにディレクトリが追加されます。szRefDir が指定したパスが検索パスにない場合、szKey の値が最後に追加されます。
nPosition	検索パスで新しいパスを追加する場所を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · BEFORE— 新規パスは szRefDir が指定したパスの前に挿入されます。szRefDir にヌル文字列 ("") が含まれている場合、ディレクトリが検索パスの前に追加されます。 · AFTER— 新規パスは szRefDir が指定したパスの後に挿入されます。szRefDir にヌル文字列 ("") が含まれている場合、ディレクトリが検索パスの終わりに追加されます。

戻り値

テーブル 19 · EzBatchAddPath の戻り値

戻り値	説明
0	EzBatchAddPath は、バッチファイルへパスを追加しました。
< 0	EzBatchAddPath は、バッチファイルへパスを追加することができませんでした。

EzBatchAddPath の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* EzBatchAddPath 関数のデモンストレーションを行います。
*
* まず、BatchSetFileName を呼び出して、デフォルトのバッチファイルを
* EXAMPLE_BATCH へ変更します。そして EzBatchAddPath を呼び出して
* PATH ステートメントの初めに C:¥WINDOWS を追加します。A
* EzBatchAddPath への 2 回目の呼び出しは PATH ステートメントの
* WINDOWS キーワードの後に C:¥UTILS を追加します。
*
* メモ : このスクリプトを実行する前に、ISExampl.bat と
*      名づけられたバッチ ファイルを作成し、C ドライブのルートに
*    格納してください。
*
¥*---------------------------------------------------------------------*/

#define EXAMPLE_BATCH "C:¥¥ISExampl.bat"
#define TITLE "EzBatchAddPath の例 "
#define MSG   " 成功しました。¥n¥n%s は、%s ステートメントへ追加されました。"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_EzBatchAddPath(HWND);

function ExFn_EzBatchAddPath(hMSI)
   STRING szKey, szPath, szRefDir;
begin

   // デフォルト バッチファイルを設定します。
   BatchSetFileName (EXAMPLE_BATCH);

   // EzBatchAddPath への次の呼び出し用にパラメーターをセットアップします。
   szKey     = "PATH";
   szPath    = "C:¥¥WINDOWS";
   szRefDir  = "";

   if (EzBatchAddPath (szKey, szPath, szRefDir, BEFORE)  0) then
      // エラーを報告します。
      MessageBox ("EzBatchAddPath failed.", SEVERE);
   else

      // 成功を報告します。
      SprintfBox (INFORMATION, TITLE, MSG, szPath, szKey);

      // EzBatchAddPath への次の呼び出し用にパラメーターをセットアップします。
      szKey     = "PATH";
      szPath    = "C:¥¥UTILS";
      szRefDir  = "WINDOWS";
```

```
    if (EzBatchAddPath (szKey, szPath, szRefDir, AFTER) < 0) then
        // エラーを報告します。
        MessageBox ("EzBatchAddPath failed.", SEVERE);
    else
        // 成功を報告します。
        SprintfBox (INFORMATION, TITLE, MSG, szPath, szKey);
    endif;

  endif;

end;
```

EzBatchAddString

EzBatchAddString 関数は、BatchSetFileName への呼び出しで変更されない限り、デフォルト バッチ ファイルへテキスト行を追加します。デフォルト バッチ ファイルは起動シーケンスの最中にシステムが実行する Autoexec.bat ファイルです。デフォルトバッチファイルの完全修飾ファイル名を決定するには、BatchGetFileName を呼び出します。EzBatchAddPath が使用するバッチファイルの名前を変更するには、BatchSetFileName を呼び出します。

注意・*EzBatchAddString 関数は、デフォルトバッチファイルが非表示または読み取り専用の場合に失敗することがあります。*

EzBatchAddPath は変更するファイルのバックアップコピーを作成しません。

メモ・*簡易バッチファイル関数を拡張バッチファイル関数と同時に使用しないでください。BatchFileLoad を呼び出してメモリへバッチファイルをロードした後、BatchFileSave を呼び出してファイルを保存するまで簡易バッチファイル関数を呼び出すことはできません。*

構文

```
EzBatchAddString ( szLine, szRefKey, nOptions );
```

パラメーター

テーブル 20 ・ EzBatchAddString のパラメーター

パラメーター	説明
szLine	ファイルへ追加するテキストの行を指定します。nOptions で NO SET を指定しない限り、この関数は szLine が環境変数であるとみなします。テキスト "SET" は、デフォルトバッチファイルへ szLine が書き込まれる前に文字列の前に挿入されます。 ***注意 ・** EzBatchAddPath 関数は、長いファイル名をサポートしません。この関数を使用して長いパスを持つ行を追加する場合は、LongPathToShortPath を呼び出して長いパスを短い同等のパスに変換してから、バッチファイルに配置する文字列に追加してください。長いファイル名の詳細については、「長いファイル名」を参照して下さい。*
szRefKey	デフォルトのバッチファイル内で szLine の追加場所に関連する参照キーを指定します。EzBatchAddString は参照キーをデフォルトバッチファイルで検索し、nOptions の値に従って szLine の内容をそのキーの前または後ろへ配置します。 ***メモ ・** EzBatchAddString はパラメーター szRefKey 内で適切な参照キーワードを検索します。例えば、環境変数のキーワードは環境変数名そのものです。*
nOptions	利用するオプションを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ BEFORE ― szRefKey を含む行の前に szLine が追加されます。szRefKey がヌル文字列 (""), の場合、szLine はファイルの最初の行として追加されます。 ・ AFTER ― szRefKey を含む行の後に szLine が追加されます。szRefKey がヌル文字列 ("") の場合、szLine はファイルの最後の行として追加されます。 ・ REPLACE ― szLine はファイルの既存行を置換します。同じキーが複数行に存在する場合、EzBatchAddString はそのキーを含む最後の行のみを置換します。 ・ NOSET ― テキスト "SET" が szLine の文字列の前には挿入されないよう指定します。 ・ COMMAND ― 検索している参照キーが DOS コマンドまたはプログラム名 (環境変数ではない) ことを指定します。 NOSET と COMMAND 定数は、相互に組み合わせたり、論理 OR 演算子を使用して、BEFORE、AFTER、REPLACE 定数と組み合わせることができます。たとえば、検索している参照キーが DOS コマンドまたはプログラム名 (環境変数ではない) の場合 、以下の例のように定数 COMMAND と AFTER 定数とを組み合わせるのに OR を使います。 `EzBatchAddString (szLine, szRefKey, AFTER | COMMAND);`

戻り値

テーブル 21 ・ EzBatchAddString の戻り値

戻り値	説明
0	EzBatchAddString は、バッチファイルへテキスト文字列を追加しました。.
< 0	EzBatchAddString はテキストを文字列を追加できませんでした。

EzBatchAddString の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* EzBatchAddString 関数のデモンストレーションを行います。
*
* このスクリプトはバッチファイル上で 4 つの操作を行います :
*
* -- まず、ファイルの最初にコメント行を追加します。
*
* -- 次に、PATH ステートメントの後に "SET TEMP=C:¥EXAPP3" 行を
*    追加します。
*
* -- そして、"SET TEMP=C:¥EXAPP3" ステートメントの後に
*    コマンド "C:¥EXAPP3¥EXAPP.EXE" を追加します。
*
* -- 最後に、既存の PATH ステートメントを新しい PATH ステートメントと
*    置き換えます。
*
* メモ : このスクリプトを実行する前に、ISExampl.bat と
*       名づけられたバッチ ファイルを作成し、C ドライブのルートに
*    格納します。効果的にこの作業を行うため、ファイルに
*    PATH ステートメントを含みます。
*
¥*---------------------------------------------------------------------*/

#define BATCH_FILE "C:¥¥ISExampl.bat"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_EzBatchAddString(HWND);

function ExFn_EzBatchAddString(hMSI)
    STRING szLine, szRefKey;
    NUMBER nOptions;
begin
```

```
// デフォルト バッチファイルを設定します。
BatchSetFileName (BATCH_FILE);

// EzBatchAddString への次の呼び出し用にパラメーターをセットアップします。
szLine    = "rem これはファイルの最初の行です。.";
szRefKey  = "";
nOptions  = BEFORE|NOSET;

// バッチファイルの最初にコメント行を追加します。
if (EzBatchAddString (szLine, szRefKey, nOptions) < 0) then
   MessageBox ("EzBatchAddString への最初の呼び出しに失敗しました。", SEVERE);
else
   MessageBox ("EzBatchAddString への最初の呼び出しに成功しました。", INFORMATION);
endif;

// EzBatchAddString への次の呼び出し用にパラメーターをセットアップします。
szLine    = "TEMP=C:¥¥EXAPP3";
szRefKey  = "PATH";
nOptions  = AFTER;

// PATH ステートメントの後に行を追加します。
if (EzBatchAddString (szLine, szRefKey, nOptions) < 0) then
   MessageBox ("EzBatchAddString への 2 回目の呼び出しに失敗しました。", SEVERE);
else
   MessageBox ("EzBatchAddString への 2 回目の呼び出しに成功しました。", INFORMATION);
endif;

// EzBatchAddString への次の呼び出し用にパラメーターをセットアップします。
szLine    = "C:¥¥EXAPP3¥¥EXAPP.EXE";
szRefKey  = "TEMP";
nOptions  = AFTER|NOSET;

// SET TEMP ステートメントの後にコマンド行を追加します。
if (EzBatchAddString (szLine, szRefKey, nOptions) < 0) then
   MessageBox ("EzBatchAddString への 3 回目の呼び出しに失敗しました。", SEVERE);
else
   MessageBox ("EzBatchAddString への 3 回目の呼び出しに成功しました。", INFORMATION);
endif;

// EzBatchAddString への次の呼び出し用にパラメーターをセットアップします。
szLine    = "PATH=C:¥¥;C:¥¥Windows";
szRefKey  = "PATH";
nOptions  = AFTER|NOSET|REPLACE|COMMAND;

 // PATH ステートメントを置換します。
if (EzBatchAddString (szLine, szRefKey, nOptions) < 0) then
   MessageBox ("EzBatchAddString への 4 回目の呼び出しに失敗しました。", SEVERE);
else
   MessageBox ("EzBatchAddString への 4 回目の呼び出しに成功しました。", INFORMATION);
endif;

endprogram
```

EzBatchReplace

EzBatchReplace 関数は、BatchSetFileName への呼び出しで変更されない限り、デフォルト バッチ ファイルで既存テキスト行を置換します。デフォルトバッチファイルは起動シーケンスの最中にシステムが実行する Autoexec.bat ファイルです。デフォルトバッチファイルの完全修飾ファイル名を決定するには、BatchGetFileName を呼び出します。EzBatchAddPath が使用するバッチファイルの名前を変更するには、BatchSetFileName を呼び出します。

バッチファイルの一般的なキーは PATH、COMSPEC、TEMP、Smartdrv.exe、Win.com、および Share.exe です。

注意・*EzBatchReplace 関数は、デフォルトバッチファイルが非表示または読み取り専用の場合に失敗することがあります。*

EzBatchReplace は変更するファイルのバックアップコピーを作成しません。

メモ・*簡易バッチファイル関数を拡張バッチファイル関数と同時に使用しないでください。BatchFileLoad を呼び出してメモリへバッチファイルをロードした後、BatchFileSave を呼び出してファイルを保存するまで簡易バッチファイル関数を呼び出すことはできません。*

構文

```
EzBatchReplace ( szNewString );
```

パラメーター

テーブル 22 ・ EzBatchReplace のパラメーター

パラメーター	説明
szNewString	ファイルで既に行が存在する場所に挿入する新規文字列を指定します。EzBatchReplace は szNewString を解析し、文字列のキーを決定します。その後、同じキーを含む行をデフォルトのバッチファイル内で検索します。関数は最後に検出された行を同じキーで置換します。 ***注意 ・** EzBatchReplace 関数は、長いファイル名をサポートしません。この関数を使用して長いパスを持つ行を置換する場合は、LongPathToShortPath を呼び出して長いパスを短い同等のパスに変換してから、バッチファイルの文字列で置換してください。長いファイル名の詳細については、「長いファイル名」を参照して下さい。*

戻り値

テーブル 23 ・ EzBatchReplace の戻り値

戻り値	説明
0	EzBatchReplace は、テキストの行の置換に成功しました。
< 0	EzBatchReplace は、テキストの行を置換することができませんでした。

EzBatchReplace の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* EzBatchReplace 関数のデモンストレーションを行います。
*
* EzBatchReplace がバッチファイルの行を置換するために呼び出されます。
*
* このスクリプトは指定したバッチファイルで 3 行を置換します。
* まず初めに、SET COMSPEC コマンドを置き換えます。そして
* PATH ステートメントを置き換えます。最後に、SMARTDRV.EXE を
* 参照するコマンドラインを置き換えます。
*
* メモ : このスクリプトを実行する前に、ISExampl.bat と
*      名づけられたバッチ ファイルを作成し、C ドライブのルートに
*      格納します。効果的にこの作業を行うため、次の行を
```

```
*    含みます :
*
*      SET COMSPEC=C:¥COMMAND.COM
*      PATH C:¥
*      SMARTDRV.EXE
*
¥*---------------------------------------------------------------------*/

#define EXAMPLE_BAT "C:¥¥ISExampl.bat"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_EzBatchReplace(HWND);

function ExFn_EzBatchReplace(hMSI)
begin

   // デフォルト バッチファイルを設定します。
   BatchSetFileName (EXAMPLE_BAT);

   // SET COMSPEC ステートメントを置換します。
   if (EzBatchReplace ("SET COMSPEC=C:¥¥DOS¥¥COMMAND.COM") < 0) then
      MessageBox ("EzBatchReplace への最初の呼び出しに失敗しました。", SEVERE);
   else
      MessageBox ("EzBatchReplace への最初の呼び出しに成功しました。", INFORMATION);
   endif;


   // PATH コマンドを置換します。
   if (EzBatchReplace ("PATH C:¥¥DOS;C:¥¥MYAPP") < 0) then
      MessageBox ("EzBatchReplace への 2 回目の呼び出しに失敗しました。", SEVERE);
   else
      MessageBox ("EzBatchReplace への 2 回目の呼び出しに成功しました。", INFORMATION);
   endif;

   // SMARTDRIVE ステートメントを置換します。
   if (EzBatchReplace ("SMARTDRV.EXE /P 1024 /C 512") < 0) then
      MessageBox ("EzBatchReplace への 3 回目の呼び出しに失敗しました。", SEVERE);
   else
      MessageBox ("EzBatchReplace への 3 回目の呼び出しに成功しました。", INFORMATION);
   endif;

end;
```

EzConfigAddDriver

EzConfigAddDriver 関数は、デバイスドライバーステートメントをデフォルトのシステム構成ファイルに追加します。ドライブステートメント位置は、他のドライバーステートメントと関連付けて指定することができます。例えば、アプリケーションは Windows Himem.sys ドライバーの前または後にデバイスドライバーのロードを必要とする場合があります。

ConfigSetFileName への呼び出しで変更されない限り、デフォルトシステム構成ファイルは Config.sys ファイルで、起動シーケンス中にシステムによって実行されます。別のファイルをデフォルトシステム構成ファイルとするには、ConfigSetFileName を呼び出します。デフォルトシステム構成ファイルの完全修飾名を決定するには、ConfigGetFileName を呼び出します。

メモ・簡易構成ファイル関数を拡張構成ファイル関数と同時に使用しないでください。ConfigFileLoad を呼び出した後、ConfigFileSave を呼び出してファイルを保存するまで、簡易構成ファイル関数を呼び出すことはできません。

構文

EzConfigAddDriver (szDriver, szRefKey, nPosition);

パラメーター

テーブル 24 · EzConfigAddDriver のパラメーター

パラメーター	説明
szDriver	ファイルに追加するドライバーの完全修飾名を指定します。システム構成ファイルの一箇所、または複数箇所にドライバーが既に存在する場合、この関数はドライバーを含む最後の行のみを差し替えます。
szRefKey	szDriver の追加に関連するデバイスドライバーの名を指定します。
nPosition	szDriver を szRefKey が指定したドライバーの前または後のいづれに追加するかを示します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · BEFORE – s z RefKey を含む行の前にステートメントが追加されます。szRefKey がヌル文字列 ("") を含む場合、ドライバーはシステム構成ファイルの最初のドライバーとして挿入されます。 · AFTER – s z RefKey を含む行の後にステートメントが追加されます。szRefKey がヌル文字列 ("") を含む場合、ドライバーはシステム構成ファイルの最後のドライバーとして挿入されます。

戻り値

テーブル 25 · EzConfigAddDriver の戻り値

戻り値	説明
0	EzConfigAddDriver はファイルへデバイスドライバーステートメントを追加しました。
< 0	EzConfigAddString はドライバーステートメントを追加することができませんでした。

EzConfigAddDriver の例

***メモ ·** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* EzConfigAddDriver 関数のデモンストレーションを行います。
*
* この例では EzConfigAddDriver を呼び出して 4 つのステートメントを
* 構成ファイルに追加します。
```

```
*
* -- EzConfigAddDriver への最初の呼び出しは、最終行の後に新しいテキスト行を
*    追加します。
*
* -- 2 番目の呼び出しは、最初の行の前に新しいテキスト行を
* 追加します。
*
* -- 3 番目の呼び出しは、HIMEM.SYS キーのあとにドライバー C:¥DRDOS¥HIDOS.SYS を
*    追加します。
*
* -- 4 番目の呼び出しは、HIMEM.SYS キーの前にドライバー EXAPP.SYS を
*    追加します。
*
* メモ : このスクリプトを実行する前に、C ドライブのルートへ
*       ISExampl.sys と名づけられた構成ファイルを作成します。
*       効率よく行うため、ファイルに次の行を含みます :
*
*       DEVICE=C:¥¥Himem.sys
*
¥*---------------------------------------------------------------*/

#define EXAMPLE_SYS "C:¥¥ISEXampl.sys"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_EzConfigAddDriver(HWND);

function ExFn_EzConfigAddDriver(hMSI)
  STRING szDriver, szRefKey;
begin

  // デフォルト構成ファイルを設定します。
  ConfigSetFileName (EXAMPLE_SYS);

  // EzConfigAddDriver への次の呼び出し用にパラメーターをセットアップします。
  szDriver  = "C:¥¥NEW¥¥MYAPP.DRV";
  szRefKey  = "";

  // 構成ファイルの終わりにドライバーを追加します。
  if (EzConfigAddDriver (szDriver, szRefKey, AFTER) < 0) then
    MessageBox ("EzConfigAddDriver への最初の呼び出しに失敗しました。", SEVERE);
  else
    MessageBox ("EzConfigAddDriver への最初の呼び出しに成功しました。", INFORMATION);
  endif;

  // EzConfigAddDriver への次の呼び出し用にパラメーターをセットアップします。
  szDriver  = "C:¥¥SYSTEM¥¥DMDRVR.BIN";
  szRefKey  = "";

  // 構成ファイルの初めにドライバーを追加します。
  if (EzConfigAddDriver (szDriver, szRefKey, BEFORE) < 0) then
    MessageBox ("EzConfigAddDriver への 2 回目の呼び出しに失敗しました。", SEVERE);
  else
    MessageBox ("EzConfigAddDriver への 2 回目の呼び出しに成功しました。", INFORMATION);
  endif;

  // EzConfigAddDriver への次の呼び出し用にパラメーターをセットアップします。
  szDriver  = "C:¥¥DRDOS¥¥HIDOS.SYS";
```

```
    szRefKey  = "HIMEM.SYS";

    // "HIMEM.SYS" キーの後に "HIDOS.SYS" ドライバーを追加します。
    if (EzConfigAddDriver (szDriver, szRefKey, AFTER) < 0) then
        MessageBox ("EzConfigAddDriver への 3 回目の呼び出しに失敗しました。", SEVERE);
    else
        MessageBox ("EzConfigAddDriver への 3 回目の呼び出しに成功しました。.", INFORMATION);
    endif;

    // EzConfigAddDriver への次の呼び出し用にパラメーターをセットアップします。
    szDriver  = "EXAPP.SYS";
    szRefKey  = "HIMEM.SYS";

    // "HIMEM.SYS" キーの後に "EXAPP.SYS" ドライバーを追加します。
    if (EzConfigAddDriver (szDriver, szRefKey, BEFORE) < 0) then
        MessageBox ("EzConfigAddDriver への 4 回目の呼び出しに失敗しました。", SEVERE);
    else
        MessageBox ("EzConfigAddDriver への 4 回目の呼び出しに成功しました。", INFORMATION);
    endif;

end;
```

EzConfigAddString

EzConfigAddString 関数はテキストの行をデフォルトのシステム構成ファイルに追加します。行の位置は、ファイル内の別のステートメントと関連付けて指定することができます。

ConfigSetFileName への呼び出しで変更されない限り、デフォルトシステム構成ファイルは Config.sys ファイルで、起動シーケンス中にシステムによって実行されます。別のファイルをデフォルトシステム構成ファイルとするには、ConfigSetFileName を呼び出します。デフォルトシステム構成ファイルの完全修飾名を決定するには、ConfigGetFileName を呼び出します。

メモ・簡易構成ファイル関数を拡張構成ファイル関数と同時に使用しないでください。*ConfigFileLoad* *を呼び出した後、*ConfigFileSave* *を呼び出してファイルを保存するまで、簡易構成ファイル関数を呼び出すことはできません。*

構文

```
EzConfigAddString ( szLine, szRefKey, nOptions );
```

パラメーター

テーブル 26 - EzConfigAddString のパラメーター

パラメーター	説明
szLine	システム構成ファイルへ追加するテキストの行を指定します。
szRefKey	システム構成ファイル内で szLine の配置場所に関連する参照キーを指定します。EzConfigAddString はシステム構成ファイルで参照キーを検索して、n Option s で渡された定数に従って、パラメーター szLine の内容を参照キーを含む行の前 / 後に配置します。
nOptions	szLine で指定した行を szRefKey を含む行の前後いずれに追加するかを示します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • **BEFORE** – szLine が指定する行を szRefKey を含む行の前に追加します。szRefKey がヌル文字列 ("") を含む場合、szLine はシステム構成ファイルの最初の行として挿入されます。 • **AFTER** – szLine が指定する行を szRefKey を含む行の後に追加します。szRefKey がヌル文字列 ("") を含む場合、szLine はシステム構成ファイルの最後の行として挿入されます。

戻り値

テーブル 27 - EzConfigAddString の戻り値

戻り値	説明
0	EzConfigAddString が文字列をデフォルトシステム構成ファイルへ追加しました。
< 0	EzConfigAddString はテキストを文字列を追加できませんでした。

EzConfigAddString の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* EzConfigAddString 関数のデモンストレーションを行います。
*
* この例は EzConfigAddString を呼び出して、構成ファイルへ 4 行を
* 追加します。
*
* -- 最初の呼び出しは、ファイルの最後に新しいテキスト行を追加
```

```
＊します。
＊
＊―― 2 番目の呼び出しは、ファイルの最初に新しいテキスト行を追加
＊します。
＊
＊―― 3 番目の呼び出しは LASTDRIVE キーの後に FASTOPEN=512 行を
＊　追加します。
＊
＊―― 4 番目の呼び出しは、EXAPPHI.SYS キーの前にドライバー EXAPPHI=ON 行を
＊　追加します。
＊
＊メモ : このスクリプトを実行する前に、C ドライブのルートへ
＊　　ISExampl.sys と名づけられた構成ファイルを作成し、C ドライブのルートに
＊　　格納します。効果的にこの作業を行うため、次の行を
＊　 含みます :
＊
＊　　LASTRDRIVE=F
＊　　DEVICE=Exapphi.sys
＊
¥*----------------------------------------------------------------*/

#define EXAMPLE_SYS "C:¥¥ISExampl.sys"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_EzConfigAddString(HWND);

function ExFn_EzConfigAddString(hMSI)
  STRING szLine, szRefKey;
begin

  // デフォルト構成ファイルを設定します。
  ConfigSetFileName (EXAMPLE_SYS);

  // EzConfigAddString への次の呼び出し用にパラメーターをセットアップします。
  szLine   = "SHELL=C:¥¥COMMAND.COM /P /E:512";
  szRefKey = "";

  // ファイルの終わりに行を追加します。
  if (EzConfigAddString (szLine, szRefKey, AFTER) < 0) then
    MessageBox ("EzConfigAddString への最初の呼び出しに失敗しました。", SEVERE);
  else
    MessageBox ("EzConfigAddString への最初の呼び出しに成功しました。.", INFORMATION);
  endif;

  szLine   = "DEVICE=C:¥¥System¥¥Dmdrvr.bin";
  szRefKey = "";

  // ファイルの最初に行を追加します。
  if (EzConfigAddString (szLine, szRefKey, BEFORE) < 0) then
    MessageBox ("EzConfigAddString への 2 回目の呼び出しに失敗しました。", SEVERE);
  else
    MessageBox ("EzConfigAddString への 2 回目の呼び出しに成功しました。", INFORMATION);
   endif;

  // EzConfigAddString への次の呼び出し用にパラメーターをセットアップします。
  szLine   = "FASTOPEN=512";
  szRefKey = "LASTDRIVE";
```

```
    // "LASTDRIVE" キーの後のファイルに行を追加します。
    if (EzConfigAddString (szLine, szRefKey, AFTER) < 0) then
        MessageBox ("EzConfigAddString への 3 回目の呼び出しに失敗しました。", SEVERE);
    else
        MessageBox ("Third  call to EzConfigAddString succeeded.", INFORMATION);
     endif;

    // EzConfigAddString への次の呼び出し用にパラメーターをセットアップします。
    szLine    = "EXAPPHI=ON";
    szRefKey  = "EXAPPHI.SYS";

    // "Exapphi.sys" キーの前のファイルに行を追加します。
    if (EzConfigAddString (szLine, szRefKey, BEFORE) < 0) then
        MessageBox ("EzConfigAddString への 4 回目の呼び出しに失敗しました。", SEVERE);
    else
        MessageBox ("EzConfigAddString への 4 回目の呼び出しに成功しました。", INFORMATION);
    endif;

end;
```

EzConfigGetValue

EzConfigGetValue 関数は、デフォルトシステム構成ファイルから FILES や BUFFERS といったパラメーターの数値を読み出します。

ConfigSetFileName への呼び出しで変更されない限り、デフォルトシステム構成ファイルは Config.sys ファイルで、起動シーケンス中にシステムによって実行されます。別のファイルをデフォルトシステム構成ファイルとするには、ConfigSetFileName を呼び出します。デフォルトシステム構成ファイルの完全修飾名を決定するには、ConfigGetFileName を呼び出します。

メモ・簡易構成ファイル関数を拡張構成ファイル関数と同時に使用しないでください。ConfigFileLoad を呼び出した後、ConfigFileSave を呼び出してファイルを保存するまで、簡易構成ファイル関数を呼び出すことはできません。

構文

```
EzConfigGetValue ( szRefKey, nvValue );
```

パラメーター

テーブル 28 ・ EzConfigGetValue のパラメーター

パラメーター	説明
szRefKey	値を読み出すパラメーターの名前を指定します。
nvValue	szRefKey で指定したコマンドの数値を返します。

戻り値

テーブル 29 ・ EzConfigGetValue の戻り値

戻り値	説明
0	EzConfigGetValue が値を読み出しました。
< 0	EzConfigGetValue は値を読み出すことができませんでした。

EzConfigGetValue の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* EzConfigGetValue 関数のデモンストレーションを行います。
*
* EzConfigGetValue が呼び出されて、既定構成ファイルにあるキーから
* 値を読み出します。ConfigSetFileName の呼び出しで
* 変更されない限り、デフォルト構成ファイルは
* 起動ドライブのルートに位置する Config.sys ファイルです。
*
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_EzConfigGetValue(HWND);

function ExFn_EzConfigGetValue(hMSI)
   NUMBER nvValue;
begin

   // デフォルト構成ファイルから FILES キーの数値を
   // 読み出します。
   if (EzConfigGetValue ("FILES", nvValue) < 0) then
      // エラーを報告します。
```

```
        MessageBox ("EzConfigGetValue が失敗しました。", SEVERE);
    else
        // FILES キーの値を表示します。
        SprintfBox (INFORMATION, "EzConfigGetValue の例 ",
                "FILES = %d", nvValue);
    endif;

end;
```

EzConfigSetValue

EzConfigSetValue 関数は、デフォルトシステム構成ファイルでコマンドの値を設定します。

ConfigSetFileName への呼び出しで変更されない限り、デフォルトシステム構成ファイルは Config.sys ファイルで、起動シーケンス中にシステムによって実行されます。別のファイルをデフォルトシステム構成ファイルとするには、ConfigSetFileName を呼び出します。デフォルトシステム構成ファイルの完全修飾名を決定するには、ConfigGetFileName を呼び出します。

メモ・簡易構成ファイル関数を拡張構成ファイル関数と同時に使用しないでください。ConfigFileLoad を呼び出した後、ConfigFileSave を呼び出してファイルを保存するまで、簡易構成ファイル関数を呼び出すことはできません。

構文

```
EzConfigSetValue (szRefKey, nValue);
```

パラメーター

テーブル 30 ・ EzConfigSetValue のパラメーター

パラメーター	説明
szRefKey	値を変更またはデフォルトシステム構成ファイルへ追加するコマンドを指定します。そのファイルにキーが存在しない場合、追加されます。
nValue	szRefKey で指定したコマンドの新しい値を指定します。

戻り値

テーブル 31 ・ EzConfigSetValue の戻り値

戻り値	説明
0	EzConfigSetValue が値を設定しました。
< 0	EzConfigSetValue は値を設定することができませんでした。

EzConfigSetValue の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* EzConfigSetValue 関数のデモンストレーションを行います。
*
* EzConfigSetValue が呼び出されて、デフォルト構成ファイルの BUFFER　および
* FILES コマンドを追加または変更します。
*
* メモ : このスクリプトを初めて実行した時、ドライブ C のルートに
*　　ISExampl.sys と名づけられたファイルが作成されます。
*　　このファイルはスクリプト解析が終了した時点で
*　　削除することができます。
*
¥*---------------------------------------------------------------------*/

#define EXAMPLE_SYS "C:¥¥ISExampl.sys"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_EzConfigSetValue(HWND);

function ExFn_EzConfigSetValue(hMSI)
  STRING szRefKey, szMsg;
```

```
    NUMBER nValue;
begin

    // デフォルト構成ファイルを設定します。
    ConfigSetFileName (EXAMPLE_SYS);

    // BUFFERS コマンドを追加または変更します。
    if (EzConfigSetValue ("BUFFERS", 30) < 0) then
        MessageBox (EXAMPLE_SYS + " にある Buffers を設定できませんでした。", SEVERE);
    else
        MessageBox (" バッファーが " + EXAMPLE_SYS+ " で 30 に設定されました。", INFORMATION);
    endif;

    // FILES コマンドを追加または変更します。
    if (EzConfigSetValue ("FILES", 30) < 0) then
        MessageBox (EXAMPLE_SYS + " 内のファイルを設定できませんでした。", SEVERE);
    else
        MessageBox (" ファイルが " + EXAMPLE_SYS + " で 30 に設定されました。", INFORMATION);
    endif;

end;
```

EzDefineDialog

EzDefineDialog 関数はカスタム ダイアログを定義します。

メモ・関数はカスタムダイアログを表示しません。カスタムダイアログを表示するには、WaitOnDialog を呼び出します。

構文

```
EzDefineDialog ( szDialogName, szDLLName, szDialogID, nDialogID );
```

パラメーター

テーブル 32 ・ EzDefineDialog のパラメーター

パラメーター	説明
szDialogName	szDialogID または nDialogID が認識したダイアログと関連付ける名前を指定します。このダイアログを処理するには、カスタムダイアログ関数に続く呼び出しにこの名前を使用します。 ダイアログの名前は大文字と小文字を区別するため、このパラメーターで指定した通りに正確に使用しなくてはなりません。
szDLLName	このダイアログリソースを含む .dll ファイルの名前を指定します。この名前が修飾されていない場合、つまりファイル名と共にドライブとパスを指定しなかった場合、InstallScript エンジンは Windows フォルダーで .dll ファイルを検索します。そこで検出されなかった場合、InstallScript エンジンは検索パスで指定したフォルダーを検索します。ダイアログが **_isuser.dll** にあるとき、このパラメーターにヌル文字列 ("") を指定することができます。InstallScript はこのパラメーターがヌル文字列 ("") として指定してある場合、**_isuser.dll** を自動的に確認します。 ***メモ・*** ***_isres.dll*** *または* ***_isuser.dll*** *以外の .dll ファイルを利用した場合、****EzDefineDialog*** *を呼び出す前に UseDLL を呼び出して .dll ファイルをロードしなくてはなりません。.dll ファイルをメモリからアンロードするには、ReleaseDialog の後に UnUseDLL を呼び出します。*
szDialogID	リソースを識別するのに (数値ではなく) 文字列を利用する場合のダイアログのリソース ID。このパラメーターがヌル文字列 ("") の場合、はダイアログリソースを識別する為に nDialog を利用します。szDialogID ではなく、nDialogID を使ってダイアログ リソースを識別することが推奨されます。 ***メモ・****[ダイアログ] ビューでダイアログを作成した場合、文字列 ID と共にダイアログが作成されます。つまり、この名前を szDialogID パラメーターとして指定してダイアログを定義づけなくてはなりません。* *[ダイアログ] ビューで既存ダイアログを上書きした場合、ダイアログは数値 ID を持つため、nDialogID パラメーターで ID を指定しなくてはなりません。* *すべての場合に数値 ID を使用する場合 (推奨)、[ダイアログ] ビューでダイアログを編集し、作成されたダイアログの ISResrouceID プロパティを 0 ではなく任意のダイアログ ID に変更する必要があります。この変更を行うと、ダイアログを作成したときにこのダイアログ名は使用されません。この名前は InstallShield が [ダイアログ] ビューでダイアログを識別する場合にのみ使用されます。ダイアログは適切な数値 ID を使ってビルドされます。*

テーブル 32 ・ EzDefineDialog のパラメーター (続き)

パラメーター	説明
nDialogID	リソースを識別するのに (文字列ではなく) 数値を利用する場合のダイアログのリソース ID。このパラメーターは szDialogID がヌル文字列 (“”) の場合のみ利用します。szDialogID ではなく、nDialogID を使ってダイアログ リソースを識別することが推奨されます。

戻り値

テーブル 33 ・ EzDefineDialog の戻り値

戻り値	説明
0	**EzDefineDialog** がダイアログを定義しました。
DLG_ERR_ALREADY_EXISTS (-3)	インストールスクリプトで定義済みのダイアログを再び定義しようとしている旨を示します。同じ名前で 2 つのダイアログを定義することはできません。
DLG_ERR (-1)	不特定エラー条件を示します。

EzDefineDialog の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* EzDefineDialog 関数のデモンストレーションを行います。
*
* このスクリプトは ビットマップを表示するシンプルなカスタムダイアログを
* 開きます。ダイアログは次の 3 つのボタンで閉じることが
* [ 戻る ]、[ 次へ ]、および [ キャンセル ]。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SdAskOptions が表示する
* InstallShield 標準ダイアログです。このダイアログは
* インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
* カスタム ダイアログとしてスクリプトで利用することが
* 表示されます。
*
* このカスタムダイアログを利用するため、スクリプトはまず EzDefineDialog を
* 呼び出して定義します。そしてダイアログを表示して
* WaitOnDialog を呼び出してダイアログ イベントを取得します。その後、
* イベントがダイアログの処理を終了するとき、EndDialog が呼び出されて
* ダイアログを閉じます。次いで、ReleaseDialog への呼び出しによって、
* メモリからダイアログがリリースされます。
```

```
 *
¥*---------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID     12027   // ダイアログ自身の ID
#define RES_PBUT_NEXT         1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL       9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK        12   //[ 戻る ] ボタンの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_EzDefineDialog(HWND);

function ExFn_EzDefineDialog(hMSI)
   STRING  szDialogName, szDLLName, szDialog;
   NUMBER  nDialog, nResult, nCmdValue;
   BOOL    bDone;
   HWND    hInstance, hwndParent, hwndDlg;
begin

   // このインストールでカスタム ボックスを認識するための名前を指定します。
   szDialogName = "CustomDialog";

   // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
   // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
   // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
   // 4 番目のパラメーターにある ID によって識別されるためです。
   nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

   if (nResult < 0) then
      // エラーを報告し、終了します。
      MessageBox ("ダイアログの定義エラー", SEVERE);
      abort;
   endif;

   // while ループを制御するのに使われるインジケーターを初期化します。
   bDone = FALSE;

   // 完了するまでループします。
   repeat

      // ダイアログを表示して次のダイアログ イベントを戻します。
      nCmdValue = WaitOnDialog(szDialogName);

      // イベントに応答します。
      switch (nCmdValue)
         case DLG_CLOSE:
            // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
            Do (EXIT);
         case DLG_ERR:
            MessageBox ("ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
            abort;
         case DLG_INIT:
            // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
            // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
            // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
            // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
            hwndDlg = CmdGetHwndDlg (szDialogName);
```

```
            SdGeneralInit(szDialogName, hwndDlg, 0, "");
        case RES_PBUT_CANCEL:
            // ユーザーが [ キャンセル ] ボタンをクリックしました。
            Do (EXIT);
        case RES_PBUT_NEXT:
            bDone = TRUE;
        case RES_PBUT_BACK:
            bDone = TRUE;
        if (SdIsStdButton( nCmdValue ) && SdDoStdButton( nCmdValue )) then
            bDone = TRUE;
        endif;
    endswitch;

until bDone;

// ダイアログを閉じます
EndDialog (szDialogName);

// メモリからダイアログを解放します。
ReleaseDialog (szDialogName);

end;
```

FeatureAddCost

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FeatureAddCost 関数は、機能が追加のインストール操作を含むように指定します。これは、インストール中、進行状況バーを更新するときに必要です。具体的に **FeatureAddCost** 関数は、nCost の値を指定された機能に追加します。この関数は、**StatusUpdate** を呼び出す前にいつでも呼び出すことができますが、通常、インストールされる機能に意図されたサイズがダイアログで表示されるように機能ダイアログの前に使用します。

重要・FeatureAddCost 関数は、ファイル メディアにのみサポートされています。FeatureGetData または FeatureSetData を使用して、スクリプトで作成された機能のサイズを設定します。

メモ・この関数で指定された情報は、次に来るインストールで記憶されません。この関数は、修復およびメンテナンスモード中も含め、機能がインストールされるたびに呼び出す必要があります。また、セットアップエンジンは、指定されたコストがシングルファイルまたは複数ファイル用なのか判断することはできません。したがって、セットアップエンジンは、ターゲットドライブのクラスタサイズにしたがって指定されたサイズを調整しようとはしません。したがって、この関数を呼び出す前にクラスタ サイズを考慮して、GetAndAddFileCost、CalculateAndAddFileCost、または GetAndAddAllFilesCost 関数を使用してファイルのコストを判別する必要があります。

構文

```
FeatureAddCost ( szMediaSource, szFeature, szTarget, nCostHigh, nCostLow );
```

パラメーター

テーブル 34 - FeatureAddCost のパラメーター

パラメーター	説明
szMediaSource	メディアソース。通常、MEDIA。
szFeature	このコストを追加する機能を指定します。インストールコストが機能によって計算されるとき、メディア内にある特定の機能の指定が必要です。コストはメディア全体に指定することはできません。
szTarget	nCost が指定したコストのターゲットディレクトリを指定します。この値は、このコストが適用するボリュームまたはドライブを判断するとき、および 機能ダイアログで機能に必要なサイズを表示するときに使用されます。現在、カスタムコストのシングル szTarget 変数のみが各機能にサポートされています。この関数を同じ機能に複数回呼び出す場合、szTarget 変数は毎回同じでなければなりません。変数が異なる場合、最後の呼び出し中に指定された値が、機能のすべての追加コストのターゲットに使用されます。
nCostHigh	機能に追加される追加のコストの上位 31 ビットを指定します。
nCostLow	機能に追加される追加のコストの下位 31 ビットを指定します。

戻り値

テーブル 35 - FeatureAddCost の戻り値

戻り値	説明
ISERR_SUCCESS	関数がコストを正常に追加したことを示します。
< ISERR_SUCCESS	関数がコストを追加できなかったことを示します。

FeatureAddItem

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

- *InstallScript*
- *InstallScript MSI*

FeatureAddItem 関数は、スクリプト作成の機能セットに別の機能を追加します。szFeatureSet が指定した名前を持つスクリプト作成の機能セットが存在しない場合、この関数がそれを作成します。

作成されたスクリプト作成の機能セットへ追加する各機能へ **FeatureAddItem** を一回づつ呼び出します。それぞれが固有の名前 (szFeatureSet パラメーター) を持つ 複数のスクリプト作成機能セットを作成することができます。

***注意 -** この関数は、ファイルメディアライブラリで使用することはできません。*

構文

FeatureAddItem (szFeatureSet, szFeature, nDataSize, bSelected);

パラメーター

テーブル 36 · FeatureAddItem のパラメーター

パラメーター	説明
szFeatureSet	アイテムを追加するスクリプト作成機能セットの名前を指定します。スクリプト作成機能セットが存在しない場合、**FeatureAddItem** によって作成されます。
szFeature	追加する機能の名前を指定します。ヌル文字列 ("") は使用しないでください。トップ階層の機能やサブ機能の参照方法は、「関数呼び出しで機能やサブ機能を指定する」を参照してください。
nDataSize	機能が表すデータのサイズをバイトで指定します。機能が連続した複数のファイルである場合、すべてのファイルの圧縮を解除した合計サイズをこのパラメーターに指定します。
bSelected	機能のデフォルトの選択状態を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · TRUE ― 機能がデフォルトで選択されることを示します。親機能が選択されていなサブ機能を選択するのに TRUE が渡された場合、bSelected パラメーターで TRUE を渡してもサブ機能は選択されません。 · FALSE ― 機能がデフォルトで選択されないことを示します。 ***メモ ·** デフォルトの選択設定は、ダイアログに表示されている機能またはサブ機能をユーザーが選択する際にクリアされます。ユーザーが選択した機能をクリアした場合、すべてのサブ機能もクリアされます。ユーザーが機能のサブ機能すべてをクリアした場合、機能の選択設定はクリアされます。*

戻り値

テーブル 37 · FeatureAddItem の戻り値

戻り値	説明
0	**FeatureAddItem** が項目を機能へ、またはサブ機能をスクリプト作成の機能セットへ追加しました。
< 0	**FeatureAddItem** が項目を機能へ、またはサブ機能をスクリプト作成の機能セットへ追加することができませんでした。追加の情報については、「FeatureError」を参照してください。

追加情報

- エンドユーザーが選択できる機能を一階層表示するには、FeatureDialog、SdFeatureDialog、または SdFeatureDialogAdv を使用してください。
- 機能やそのサブ機能を表示するには、SdFeatureDialog2、SdFeatureMult、または SdFeatureTree を使用してください。

- **FeatureAddItem** とスクリプト作成機能セットを使用すると、ユーザーがファイルメディア機能オプション以外のオプションを選択できるようになります。

  1. **FeatureAddItem** を呼び出して、必要なオプションが含まれるスクリプト作成機能セットを作成します。
  2. パラメーター szFeature にスクリプト作成コンポーネントセット名を指定して **SdAskOptions** または **SdAskOptionsList** を呼び出し、これらのオプションを表示して選択できるようにします。
  3. どのオプションが選択されたか判断するため、**FeatureIsItemSelected** を呼び出します。

FeatureAddItem の例

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * FeatureAddItem 関数のデモンストレーションを行います。
 *
 * このスクリプトでは、FeatureAddItem を呼び出して、
 * ターゲットシステムに接続されたドライブで構成された
 * スクリプト作成の機能セットを作成します。これらの機能は、エンドユーザーが
 * ドライブから選択できるように、SdAskOptionsList ダイアログに表示
 * されます。
 *
¥*---------------------------------------------------------------*/

LIST listDrives, listFeatures;
STRING szSaveMEDIAValue, svDrive;
LONG nResult;
BOOL bSelected;

#include "ifx.h"

function OnBegin()
begin

    // ドライブ名を格納する 2 つのリストを作成します。
    listDrives = ListCreate(STRINGLIST);

    // リストへすべての固定ドライブを追加します。
    GetValidDrivesList(listDrives, FIXED_DRIVE, -1);

    // システム変数 MEDIA の値を保存し、
    // 後のデータ転送のための呼び出しで復元できるようにします。
    szSaveMEDIAValue = MEDIA;

    // スクリプト作成の機能セットの名前を指定します。
    MEDIA = " ドライブ ";

    // スクリプト作成の機能背ってオにドライブ名を入力します。
    nResult = ListGetFirstString(listDrives, svDrive);
    while ( nResult != END_OF_LIST )
        FeatureAddItem(MEDIA, svDrive, 0, FALSE);
        nResult = ListGetNextString(listDrives, svDrive);
    endwhile;

    // ドライブリストを表示します。ユーザがドライブの 1 つを選択できるようにします。
```

```
    SdAskOptionsList(" ドライブの選択 " ,
    " システムに接続しているドライブの 1 つを選択します。",
    "" , EXCLUSIVE);

    // リストから選択されたドライブを探します。
   bSelected = FALSE;
   nResult = ListGetFirstString(listDrives, svDrive);
   bSelected = ComponentIsItemSelected (MEDIA , svDrive);
   while ((nResult != END_OF_LIST) && !bSelected);
       nResult = ListGetNextString(listDrives, svDrive);
       bSelected = ComponentIsItemSelected (MEDIA , svDrive);
    endwhile;

    // メモリからリストをリリースします。
    ListDestroy(listDrives);
    if bSelected then
       MessageBox(svDrive + " ドライブが選択されています。", INFORMATION);
    endif;

    // MEDIA をファイル転送のために以前の値に復元します。
    MEDIA = szSaveMEDIAValue;

end;
```

FeatureAddUninstallCost

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FeatureAddUninstallCost 関数は、機能が追加のアンインストール操作を含むように指定します。これは、アンインストール中、進行状況バーを更新するときに必要です。アンインストール ログ ファイルに含まれている操作にこの関数を呼び出す必要はありません。これには、インストール中に **FeatureAddCost** 関数を呼び出して進行状況バーを更新する必要があった可能性がある **XCopyFile** 関数によって処理される操作も含まれます。この関数は、InstallScript エンジン以外の機能によって実行された操作にのみ必要です。この関数は、**StatusUpdate** を呼び出す前にいつでも呼び出すことができ、複数回呼び出すことも可能です。

構文

```
FeatureAddUninstallCost ( szMediaSource, szFeature, nOps, nOpType);
```

パラメーター

テーブル 38 · FeatureAddUninstallCost のパラメーター

パラメーター	説明
szMediaSource	メディアソース。通常、MEDIA。
szFeature	このコストを追加する機能を指定します。メディア内にある特定の機能も指定が必要ですので、ご注意ください。アンインストールコストは機能によって計算されます。コストはメディア全体に指定することはできません。
nOps	追加される操作の数。
nOpType	操作の種類を示します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · FEATURE_OPCOST_UNINSTALL_FILE – 操作をファイルのアンインストールとして指定します。値 4096 を使用して、この定数を指定することもできます。また、異なるサイズを試して、カスタム操作に使用するサイズを判断できます。 · FEATURE_OPCOST_UNINSTALL_REGORINI – 操作をレジストリファイルのアンインストールとして指定します。値 2048 を使用して、この定数を指定することもできます。また、異なるサイズを試して、カスタム操作に使用するサイズを判断できます。 · FEATURE_OPCOST_UNINSTALL_UNREGFILE – 操作をファイルの登録解除として指定します。値 16384 を使用して、この定数を指定することもできます。また、異なるサイズを試して、カスタム操作に使用するサイズを判断できます。 ***メモ ·** また、このパラメーターで数字の値を渡して、特定のコストの操作を示すこともできます。異なる値を試して、指定されたカスタム操作に適切な値を判別する必要がある場合もあります。追加されたアンインストールコストの合計は、nOps を nOpType で乗じた値です。*

戻り値

テーブル 39 · FeatureAddUninstallCost の戻り値

戻り値	説明
ISERR_SUCCESS	関数がコストを正常に追加したことを示します。
< ISERR_SUCCESS	関数がコストを追加できなかったことを示します。

FeatureCompareSizeRequired

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FeatureCompareSizeRequired 関数は、ターゲットフォルダーが szFeatureSource によって指定された機能に十分な空き容量があるか決定します。これはファイルメディアでなくてはなりません。ターゲットフォルダーに十分な空き容量がない場合、完全修飾フォルダー名が svTarget で返され、必要な空容量が nvSize で返されます。

セットアップが機能のインストール先フォルダーをランタイムでファイルライブラリ内に指定する場合、**FeatureCompareSizeRequired** を使って空き容量を調べる前に、FeatureSetTarget を呼び出して指定する必要があります。

メモ・この関数は、スクリプト作成機能セットでは使用できません。

構文

```
FeatureCompareSizeRequired ( szFeatureSource, svTarget, nvSize );
```

パラメーター

テーブル 40 ・ FeatureCompareSizeRequired のパラメーター

パラメーター	説明
szFeatureSource	システム変数の MEDIA をこのパラメーターで渡します。その他の値は使用できません。
svTarget	ターゲットドライブに十分な空き容量がある場合、ヌル文字列 ("") を返します。ターゲットドライブに十分な空き容量が *ない* 場合は、ターゲットパスを返します。 ***メモ・*** *パラメーター svTarget はフォルダー名を返すためのみに使用します。パラメーターを使ってインストール先フォルダーを指定することはできません。* ***FeatureCompareSizeRequired*** *は、[コンポーネント] ビューで各コンポーネントに指定されたインストール先フォルダーが示すドライブをチェックします。szFeatureSource が [一般アプリケーションのインストール先] にインストールする機能を含むファイル メディアの場合、* ***FeatureCompareSizeRequired*** *を呼び出す前にインストール先のパスを TARGETDIR (InstallScript インストールの場合) または INSTALLDIR (InstallScript MSI インストールの場合) に割り当てる必要があります。AskDestPath または機能のダイアログを呼び出して、エンドユーザーからインストール先のパスを取得することができます。*
nvSize	ターゲットドライブに十分な空き容量がある場合、0 を返します。ターゲットドライブに十分な空き容量が *ない* 場合、必要なサイズをバイトで返します。

戻り値

テーブル 41 ・ FeatureCompareSizeRequired の戻り値

戻り値	説明
0	FeatureCompareSizeRequired が成功しました。
< 0	FeatureCompareSizeRequired が失敗しました。追加情報については、FeatureError を呼び出してください。

FeatureCompareSizeRequired の例

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureAddItem 関数、FeatureSetData 関数、FeatureCompareSizeRequired 関数、
* そして SdFeatureMult 関数のデモンストレーションを行います。
*
```

```
＊メモ : DESTDIR はフロッピードライブなど、有効スペースが
＊　　殆どないドライブを参照しなくてはなりません。このスクリプトを
＊　　実行する前に、フロッピードライブに必ずディスクを挿入して
＊　　下さい。
＊
¥*---------------------------------------------------------------*/

#define FEAT1          "CAD プログラムファイル"
#define FEAT1SIZE      25000000
#define FEAT1DESC      "CAD プログラム EXE と DLL。"
#define FEAT2          "CAD テンプレート"
#define FEAT2SIZE      18000000
#define FEAT2DESC      "CAD テンプレートファイル。"
#define SUBFEAT1       "工業"
#define SUBFEAT1SIZE   7000000
#define SUBFEAT1DESC   "CAD 工業技術テンプレートファイル。"
#define SUBFEAT2       "民間"
#define SUBFEAT2SIZE   5000000
#define SUBFEAT2DESC   "CAD 民間技術テンプレートファイル。"
#define SUBFEAT3       "機械"
#define SUBFEAT3SIZE   6000000
#define SUBFEAT3DESC   "CAD 機械技術テンプレートファイル。"
#define DESTDIR        "a:¥¥"
#define SDFEATTITLE    "スクリプト作成の機能を表示します"
#define SDFEATMSG      "すべての機能を確実に選択してください。"
#define FEATSIZEERRMSG "ターゲットドライブへのアクセスが可能であることを確認してください。"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_FeatureCompareSizeRequired(HWND);

function ExFn_FeatureCompareSizeRequired(hMSI)
   STRING svDir, svTarget;
   NUMBER nvSize, nData;
begin

   // セットアップダイアログで [ 戻る ] ボタンを無効にします。Disable(BACKBUTTON);
   // サブ機能を含むスクリプト作成の機能セットを作成し、
   // それに、"ランタイム CAD" というメディア名をつけます。
   MEDIA = "ランタイム CAD";

   FeatureAddItem (MEDIA, FEAT1, FEAT1SIZE, TRUE);

   FeatureSetData (MEDIA, FEAT1, FEATURE_FIELD_DESCRIPTION,
                   nData, FEAT1DESC);

   FeatureAddItem (MEDIA, FEAT2, FEAT1SIZE, TRUE);

   FeatureSetData (MEDIA, FEAT2, FEATURE_FIELD_DESCRIPTION, nData, FEAT2DESC);

   FeatureAddItem (MEDIA, FEAT2 + "¥¥" + SUBFEAT1, SUBFEAT1SIZE, TRUE);

   FeatureSetData (MEDIA, FEAT2 + "¥¥" + SUBFEAT1, FEATURE_FIELD_DESCRIPTION,
                   nData, SUBFEAT1DESC);

   FeatureAddItem (MEDIA, FEAT2 + "¥¥" + SUBFEAT2, SUBFEAT2SIZE, TRUE);

   FeatureSetData (MEDIA, FEAT2 + "¥¥" + SUBFEAT2,  FEATURE_FIELD_DESCRIPTION,
```

```
                nData, SUBFEAT2DESC);

    FeatureAddItem (MEDIA, FEAT2 + "¥¥" + SUBFEAT3, SUBFEAT3SIZE, TRUE);

    FeatureSetData (MEDIA, FEAT2 + "¥¥" + SUBFEAT3, FEATURE_FIELD_DESCRIPTION,
                nData, SUBFEAT2DESC);

    // スクリプト作成の機能とサブ機能を表示します。
    SdFeatureMult (SDFEATTITLE, SDFEATMSG, svDir, "");

    // フロッピードライブとして定義する場合、INSTALLDIR を DESTDIR に設定します
    // （ディスクを挿入した状態の場合）、FeatureCompareSizeRequired が
    // ' スペースが足りません ' メッセージを表示する原因となります。
    nvSize = 0;

    INSTALLDIR = DESTDIR;

    if (FeatureCompareSizeRequired(MEDIA, svTarget, nvSize) < 0) then
        MessageBox (FEATSIZEERRMSG, SEVERE);
    endif;

end;
```

FeatureConfigureFeaturesFromSuite

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

メモ・この関数は、アドバンスト UI またはスイート / アドバンスト UI インストールに InstallScript パッケージとして含める可能性がある InstallScript インストールに使用できます。この関数は、アドバンスト UI またはスイート / アドバンスト UI インストールに実行可能パッケージとして含められている InstallScript インストールでも使用できます。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート / アドバンスト UI プロジェクトに追加する」をご覧ください。

また、直接起動された （つまり、アドバンスト UI またはスイート / アドバンスト UI インストールから起動されなかった）InstallScript インストールでも使用することができます。

FeatureConfigureFeaturesFromSuite 関数は、アドバンスト UI またはスイート / アドバンスト UI プロパティ **ISFeatureInstall** と **ISFeatureRemove** の値に基づいて、アドバンスト UI またはスイート / アドバンスト UI インストールで実行中の InstallScript パッケージに対して機能を設定します。関数は、OnSuiteInstallBefore イベント（[インストール] 操作の場合) と OnSuiteMaintBefore イベント（[変更] 操作の場合) のデフォルト コードによって呼び出されます。

FeatureConfigureFeaturesFromSuite 関数を使って、アドバンスト UI またはスイート / アドバンスト UI インストールで実行可能パッケージとして起動される InstallScript インストールに対してだけでなく、アドバンスト UI またはスイート / アドバンスト UI インストールの外で直接起動される InstallScript インストールに対しても機能を設定できます。これらのケースでは、関数は、**ISFeatureInstall** と **ISFeatureRemove** のキー値のペアに基づいて機能を設定します。

構文

```
FeatureConfigureFeaturesFromSuite (string szCommandLine);
```

パラメーター

テーブル 42 - FeatureConfigureFeaturesFromSuite のパラメーター

パラメーター	説明
szCommandLine	**ISFeatureInstall** と **ISFeatureRemove** のいずれか、または両方を設定するために使用するコマンドラインを指定します。次のように、機能の名前をカンマ区切りで設定します : ISFeatureInstall=Feature1,Feature2 ISFeatureRemove=Feature3 上記の例では、Feature1 と Feature2 は、インストールされるように選択されています。Feature3 は、存在していれば、削除されるように選択されています。 機能が、両方のプロパティの値で使用されている場合、**ISFeatureRemove** プロパティに設定された機能は、**ISFeatureInstall** プロパティに優先されます。 szCommandLine が、いずれかのプロパティを設定しない場合、または両方とも設定しない場合、**FeatureConfigureFeaturesFromSuite** への呼び出しは効果がありません。 **ISFeatureInstall** と **ISFeatureRemove** のトップ階層の機能やサブ機能の参照方法は、「関数呼び出しで機能やサブ機能を指定する」を参照してください。

戻り値

この関数に戻り値はありません。

FeatureDialog

プロジェクト - *この情報は、次のプロジェクトの種類に適用します :*

- *InstallScript*
- *InstallScript MSI*

FeatureDialog 関数は、エンド ユーザーがインストールの機能リストから 1 つまたは複数の項目を選択できるダイアログを表示します。また、エンド ユーザーはインストール先を選択することもできます。

エンドユーザーが [参照] ボタンをクリックすると [フォルダーの選択] ダイアログが開き、既存のフォルダーのリストが開きます。エンドユーザーはリストから既存フォルダーを選択するか、パスフィールドに新規フォルダー名を入力することができます。**FeatureDialog** は指定されたフォルダーの名前を svDir で返します。ユーザーが存在しないフォルダー名を入力すると、インストールはフォルダーを作成します。

注意 - *インストールでセットアップの種類ダイアログを使用しない場合は、**FeatureDialog** を呼び出す前に FeatureSetupTypeSet を必ず呼び出して、InstallShield の [セットアップの種類] ビューで定義されているセットアップの種類を指定してください。*

構文

FeatureDialog (szTitle, szMsg, svDir, szFeature);

パラメーター

テーブル 43 ・ FeatureDialog のパラメーター

パラメーター	説明
szTitle	ダイアログ タイトルを指定します。デフォルトのタイトル (" 機能の選択 ") を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
svDir	デフォルトのインストール先変数を指定します。エンドユーザーが選択したフォルダーを返します。svDir で戻された場所は、それを TARGETDIR (InstallScript installations) または INSTALLDIR (InstallScript MSI インストール) に割り当てない限り、ファイル転送には反映されません。 svDir によって指定されたデフォルトのフォルダーがエンド ユーザーのシステムに存在しない場合、エンドユーザーが [参照] ボタンをクリックして、[フォルダーの選択] ダイアログのステップに従いフォルダーを作成しない限り、フォルダーは作成されません。従って、デフォルトフォルダーを指定するときはいつでも、FeatureDialog が戻るときに ExistsDir を呼び出し、そのフォルダーの存在を確認しなくてはなりません。フォルダーが存在しない場合、CreateDir を呼び出して、エンド ユーザーのシステムでそのフォルダーを作成します。
szFeature	選択できるようにサブ機能を表示する機能を指定します。このパラメーターにヌル文字列 ("") を渡して、トップレベルの機能をすべて表示します。トップ階層の機能やサブ機能の参照方法は、「関数呼び出しで機能やサブ機能を指定する」を参照してください。 FeatureDialog はシステム変数 MEDIA が指定するスクリプト作成の機能セットで必要な機能を検索します。 ***メモ・最大の機能サイズを表示できるように、機能名は必要に応じて切り詰められます。このサイズを表示するために必要な領域は、最大機能サイズ (2 GB)、現在使用されている機能サイズオプション、およびダイアログ機能情報を表示するために使用されるフォントによって異なります。機能サイズのオプションは、DialogSetInfo 関数で設定します。*** ***最大サイズの表示に必要な領域が決定されると、すべての機能名は残りの領域に収まるよう、必要に応じて自動的に切り詰められます。これにより、機能名が機能サイズと重ならないようにします。*** ***この方法では、サイズの表示に必要な領域が小さい (またはサイズが選択されていない) 機能の名前も切り詰められるので注意してください。機能名がすべて確実に表示されてフル活用できるように、機能名または表示名をダイアログの有効スペースより短くしてください。***

戻り値

テーブル 44 - FeatureDialog のパラメーター

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。
< 0	FeatureDialog ダイアログを表示できないようにします。追加情報については、FeatureError を呼び出してください。

追加情報

[フォルダーの選択] ダイアログは、各機能が必要なディスク容量も表示します。機能のサイズは、選択されない場合 0 と表示されます。サイズが選択されると、実際のサイズが表示されます。

FeatureDialog の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * FeatureDialog 関数のデモンストレーションを行います。
 *
 * このスクリプト例は、ユーザーがインストールできるセットアップに含まれる
 * 機能のリストと、各機能に必要な容量を表示する
 * ダイアログ ボックスを表示します。
 *
 * コメント : このスクリプト例を実行するには、いくつかの機能および
 *          / またはファイルを持つコンポーネントを含むサブ機能を
 *          持つプロジェクトを作成 ( またはプロジェクトに挿入 ) します。
 *
 *
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

// Windows Installer API 関数プロトタイプと定数用に iswi.h をインクルードし、
// OnBegin イベントと OnEnd イベント用のコードを宣言します。
#include "iswi.h"

// キーワードのエクスポートは MyFunction () をエントリポイント関数として識別します。
// 使用する引数は、Installer データベースへのハンドルでなくてはなりません。
  export prototype MyFunction(HWND);
```

```
// 必要な処理 : グローバル変数の宣言、定数の定義、およびユーザー定義の関数並びに DLL 関数を
//         ここでプロトタイプ化します。

function MyFunction(hMSI)

    STRING  szTitle, szMsg, svDir;

begin

    svDir   = INSTALLDIR;
    szTitle = " 機能の選択 ";
    szMsg   = " コンピューターへインストールする機能を選択してください。";

    // 利用可能な機能を表示します。
    FeatureDialog (szTitle, szMsg, svDir, "");

end;
```

FeatureError

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

FeatureError はサポートされていない関数についてのエラー情報を返しません。例えば、ComponentInitialize、ComponentUpdate、および ComponentValidate は機能関数としてサポートされていません。FeatureError はこれらの関数には作動しません。

FeatureError 関数は、機能関数が 0 未満の値を返したときに追加のエラー情報を取得します。次の部分コードは、FeatureError の標準的な実行例を示しています。

```
nResult = FeatureGetData (MEDIA, svFeature, FEATURE_FIELD_SELECTED, nvResult,
        svResult);

if(nResult < 0) then
    FeatureError(svFeatureSource, svFeature, svComponent, svFile, nvError);

    SprintfBox (INFORMATION, "FeatureGetData エラー情報 ",
            "FeatureGetData に次のエラーがあります :¥n¥n" +
            " メディア名 : %s¥n 機能 : %s¥n コンポーネント : %s¥n" +
            " ファイル : %s¥n エラー番号 : %ld",
            svFeatureSource, svFeature, svComponent, svFile, nvError);
endif;
```

FeatureError 関数は、他の機能関数がゼロ未満の値を返した後にのみ呼び出します。FeatureError は、他の機能関数がゼロ未満の値を戻さなかった場合に呼び出すと、無効なエラーコードを戻すことがあります。

メモ・FeatureError はスクリプト作成機能セットにのみ すべてのパラメーター (メディア名、機能名、コンポーネント名、ファイル名、そしてエラーコード) の情報を返します。ファイルメディア (MEDIA) の機能 には、FeatureError はエラーコードのみを返します。

構文

FeatureError(svFeatureSource, svFeature, svComponent, svFile, nvError);

パラメーター

テーブル 45 ・ FeatureError のパラメーター

パラメーター	説明
svFeatureSource	nvError が示すエラーに関連する場合は、スクリプト作成機能セットのメディア名を返します。
svFeature	nvError が示すエラーに関連する場合は、機能名を返します。
svComponent	nvError によって示されるエラーに関連するコンポーネント名を返します。
svFile	nvError によって示されるエラーに関連するファイル名を返します。
nvError	以前に機能関連の関数を呼び出した際に発生したエラーの種類を識別するコードを返します。これらのエラーコードを以下に示します。

Error Codes

次の表は、FeatureError が戻すエラーコードについて説明します。

メモ・メディアに関連するエラーを訂正した後は、プロジェクトを再ビルドする必要があります。

テーブル 46 ・ FeatureError のエラーコード

コード	説明	原因
-101	機能を追加できません。	FeatureAddItem はスクリプト作成機能セットへ機能を追加できませんでした。
-102	指定した機能は既に存在します。	FeatureAddItem が同じメディア名と機能名で 2 回呼び出されました。
-103	指定した機能を選択、または選択解除できません。	FeatureSelectItem は現在選択した機能が必要とする機能を選択または選択解除するのに利用されました。必要な機能を選択または選択解除する前に (その他の機能を必要とする) 従属機能 を選択解除します。
-105	メディアに指定した機能が見つかりません。	存在しない機能へアクセスしようとしました。このエラーは、機能関数への呼び出しで機能名が間違って指定された場合に発生します。機能名は、[セットアップデザイン] ビューで、または FeatureAddItem への呼び出しで表示されるのと全く同様に指定しなくてはなりません。機能名では、大文字と小文字が区別されます。

テーブル 46 · FeatureError のエラーコード (続き)

コード	説明	原因
-108	ディスク容量が足りません。	ターゲットディスクまたはディレクトリに空き容量が足りないか、TARGETDIR (InstallScript インストール) または INSTALLDIR (InstallScript MSI インストール) が無効なためディスク容量を判断できないか、または機能のスクリプト定義フォルダーが設定されていません。
-118	実行しようとした操作は、スクリプト作成機能では利用できません。	スクリプト作成の機能セット名が、ファイルメディア操作専用の機能関数に渡されました。
-125	機能関数で指定したリストが無効です。	FeatureListItems を呼び出す際に、関数に渡すリストが有効であることを確認してください。
-126	実行しようとした操作は、ファイルメディア ライブラリでは行えません。	ファイルメディア名が、スクリプト作成の機能セットでのみ動作する機能関数 (例、FeatureAddItem) に渡されました。
-132	指定したメディアが見つかりません。	メディアが宣言されましたが、どの機能にも関連付けされていません。スクリプト作成の機能を確実にメディアと関連付けてください。
-136	メモリを割り当てられません。	セットアップに必要なメモリが不足しています。エンドユーザーに対して、他のすべてのアプリケーションを閉じるかセットアップをキャンセルし、システムを再起動させてセットアップを再スタートさせるようメッセージを表示します。
-137	指定したオプションは無効です。	コンポーネントとファイル名の両方が必要な場所でコンポーネントのみを指定するなど、機能関数に無効なオプションが指定されました。
-147	機能関連の関数に無効な値が渡されました。	機能関数に渡された値の 1 つが無効です。このエラーは、たとえば FeatureAddItem の 2 番目のパラメーターに空の文字列を渡したことなどが原因です。

戻り値

テーブル 47 ・ FeatureError の戻り値

戻り値	説明
0	FeatureError が成功しました。
< 0	FeatureError が失敗しました。

FeatureError の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureError 関数のデモンストレーションを行います。
*
* このスクリプト例は、スクリプト作成の機能セットへ機能を追加する
* 仕組みについてデモンストレーションを行います。
*
* コメント : このスクリプト例を実行するには、いくつかの機能および
*       / またはファイルを含むコンポーネントを持つサブ機能を含む
プロジェクトを作成 ( またはプロジェクトに挿入 ) します。
*
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

// Windows Installer API 関数プロトタイプと定数用に iswi.h をインクルードし、
// OnBegin イベントと OnEnd イベント用のコードを宣言します。
#include "iswi.h"

// キーワードのエクスポートは MyFunction () をエントリポイント関数として識別します。
// 使用する引数は、Installer データベースへのハンドルでなくてはなりません。
  export prototype MyFunction(HWND);
  prototype HandleMoveDataError(number);

// 必要な処理 : グローバル変数の宣言、定数の定義、およびユーザー定義の関数並びに DLL 関数を
//        ここでプロトタイプ化します。

function MyFunction(hMSI)

    STRING  szTitle, szMsg;
    NUMBER  listID, nResult;

begin

    szTitle = "MEDIA 機能をリストにする ";
    szMsg   = "MEDIA は次のトップレベル機能を含みます :";

    // 文字列リストを初期化します。
    listID = ListCreate (STRINGLIST);

    // 指定したメディアにトップレベル機能のリストを作成します。
```

```
    nResult = FeatureListItems (MEDIA, "", listID);

    // エラーハンドラー関数を呼びだします。
    HandleMoveDataError (nResult);

    // トップレベル機能のリストを表示します。
    SdShowInfoList (szTitle, szMsg, listID);

end;

function HandleMoveDataError(nResult)

    STRING szErrMsg, svFeature , svComponent , svFile;

begin

    svFeature = "";
    svComponent = "";
    svFile = "";

    // FeatureListItems が 0 以外の値を返した場合、エラーメッセージを
    // 表示します。
    switch (nResult)
        case 0:
            return 0;
        デフォルト :
            FeatureError (MEDIA , svFeature , svComponent , svFile , nResult);
            szErrMsg = " 処理中にエラーが発生しました : %d"+ "¥n¥n" +
                " 機能 : " + svFeature + "¥n" +
                " コンポーネント : " + svComponent + "¥n" +
                " ファイル : " + svFile;
            SprintfBox (SEVERE, "", szErrMsg, nResult);
            return nResult;
    endswitch;

end;
```

FeatureErrorInfo

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

OnComponentError イベントハンドラーのデフォルトコードで呼び出される **FeatureErrorInfo** 関数は、FeatureError 以外のエラーイベントを生成しないファイル転送エラー（FileLocked または SelfRegistrationError など ）に関する情報を返します。

メモ・この関数は、ファイル転送エラーに関する情報のみを返します。スクリプト作成機能からのエラーに関する情報については、FeatureError を呼び出してください。

構文

FeatureErrorInfo ( );

パラメーター

なし

戻り値

設定されたキーワードを使用して、OBJECT 型の変数に割り当てられるリファレンス。参照が割り当てられると、変数 (下の例の oErrorInfo) は以下のプロパティを持つようになります。

テーブル 48 · FeatureErrorInfo の戻り値

プロパティ	説明
oErrorInfo.Feature	エラーが発生したときに転送されていた機能に関する情報を提供するプロパティのオブジェクト。特定の機能にエラーが関連づけられていない場合、または機能が識別されない場合、このプロパティは設定されません。IsObject(oErrorInfo.Feature) 確認してテストを行ってください。
oErrorInfo.Feature.DisplayName	エラーが発生したときに転送されていた機能の表示名。この機能に対して表示名を指定しなかった場合、この文字列はヌル (″″) になります。
oErrorInfo.Feature.Name	エラーが発生したときに転送されていた機能の名前。
oErrorInfo.Feature.Description	ファイル転送エラーを説明する文字列。この文字列は、ヌル (″″) である可能性があります。
oErrorInfo.LastError	ファイル転送エラーの数値コード。

FeatureErrorInfo の例

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureError 関数のデモンストレーションを行います。
*
* このスクリプト例は、スクリプト作成の機能セットへ機能を追加する
* 仕組みについてデモンストレーションを行います。
*
*  コメント : このスクリプト例を実行するには、いくつかの機能および
*       / またはファイルを含むコンポーネントを持つサブ機能を含む
プロジェクトを作成 ( またはプロジェクトに挿入 ) します。
*
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include ″Ifx.h″

// Windows Installer API 関数プロトタイプと定数用に iswi.h をインクルードし、
// OnBegin イベントと OnEnd イベント用のコードを宣言します。
```

```
#include "iswi.h"

// キーワードのエクスポートは MyFunction () をエントリポイント関数として識別します。
// 使用する引数は、Installer データベースへのハンドルでなくてはなりません。
  export prototype MyFunction(HWND);
  prototype HandleMoveDataError(number);

// 必要な処理 : グローバル変数の宣言、定数の定義、およびユーザー定義の関数並びに DLL 関数を
//        ここでプロトタイプ化します。

function MyFunction(hMSI)

   STRING  szTitle, szMsg;
   NUMBER  listID, nResult;

begin

   szTitle = "MEDIA 機能をリストにする ";
   szMsg   = "MEDIA は次のトップレベル機能を含みます :";

   // 文字列リストを初期化します。
   listID = ListCreate (STRINGLIST);

   // 指定したメディアにトップレベル機能のリストを作成します。
   nResult = FeatureListItems (MEDIA, "", listID);

   // エラーハンドラー関数を呼びだします。
   HandleMoveDataError (nResult);

   // トップレベル機能のリストを表示します。
   SdShowInfoList (szTitle, szMsg, listID);

end;

function HandleMoveDataError(nResult)

   STRING szErrMsg, svFeature , svComponent , svFile;

begin

   svFeature = "";
   svComponent = "";
   svFile = "";

   // FeatureListItems が 0 以外の値を返した場合、エラーメッセージを
   // 表示します。
   switch (nResult)
      case 0:
         return 0;
      デフォルト :
         FeatureError (MEDIA , svFeature , svComponent , svFile , nResult);
         szErrMsg = " 処理中にエラーが発生しました : %d"+ "¥n¥n" +
            " 機能 : " + svFeature + "¥n" +
            " コンポーネント : " + svComponent + "¥n" +
            " ファイル : " + svFile;
         SprintfBox (SEVERE, "", szErrMsg, nResult);
         return nResult;
   endswitch;
```

```
end;
```

FeatureFileEnum

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

FeatureFileEnum 関数は、指定された機能に関連付けられたコンポーネントにあるファイルのリスト、または特定の機能に関連付けられているコンポーネントのリストをビルドします。

メモ・この関数は、スクリプト作成機能セットでは使用できません。

構文

```
FeatureFileEnum ( szFeatureSource, szFeature, szQuery, listFilesorComponents, nOption );
```

パラメーター

テーブル 49 ・ FeatureFileEnum のパラメーター

パラメーター	説明
szFeatureSource	このパラメーターで MEDIA を渡します。
szFeature	列挙する機能の名前を指定します。このパラメーターにヌル文字列 (""") を渡すことはできません。**FeatureFileEnum** は特定の機能名を必要とします。
szQuery	特定の機能にあるコンポーネントをクエリする場合、コンポーネントの仕様を指定します (ファイルの仕様と同じフォーマットです)。コンポーネントの仕様を 2 つの円記号で区切り、その式を 二重引用符で括ります。 コンポーネントの仕様にはワイルドカード文字を含むことが可能で、アスタリスク (*) は任意の数の文字 (ゼロ文字を含む) を、また疑問符 (?) は単一の文字の代替を示します。以下は、szFeature で指定された機能にあるすべてのコンポーネントを指定する例です : "*.*" 特定のコンポーネントにあるファイルをクエリする場合、コンポーネントの名前とファイルの仕様を指定します。コンポーネント名とファイル指定を 2 つの円記号で区切り、式を 2 つの引用符で括ります。ファイルの仕様のみを指定することもできます。ファイルの仕様のファイル名部分に、ワイルドカード文字を含めることができます。次の例ではコンポーネント Graphic_Examples にあるすべてのファイルを指定します : "Graphic_Examples¥¥*.*"  ***メモ・****InstallScript MSI インストールでこの関数を使用する場合 (コンポーネントおよびファイルのクエリ両方)、複雑なワイルドカード式はサポートされていないため、いくつかの制限があります。* *ファイルをクエリする場合、次の 1 つを szQuery として指定する必要があります。* • *< コンポーネント名 >¥*.** • *< コンポーネント名 >¥< コンポーネント ファイル名 >* • *< コンポーネント名 >¥*.< 拡張子 >* • *< コンポーネント名 >¥< ファイル名 >.** *コンポーネントをクエリする際、次の 1 つを szQuery として指定する必要があります。* • **.** • *< 完全コンポーネント >*
listFilesOrComponents	コンポーネントをクエリするとき、szQuery に一致するコンポーネントのリストを返します。ファイルをクエリするとき、szQuery と一致するすべてのファイル名のリストを返します。listFilesOrComponents によって識別される文字列リストは、ListCreate への呼び出しによって既に初期化されている必要があります。

テーブル 49 ・ FeatureFileEnum のパラメーター (続き)

パラメーター	説明
nOption	このパラメーターで NO_SUBDIR を渡します。コンポーネントのリストをクエリする際、nOptions は無視されるので注意が必要です。

戻り値

テーブル 50 ・ FeatureFileEnum の戻り値

戻り値	説明
0	FeatureFileEnum が成功しました。
<0	FeatureFileEnum が失敗しました。

FeatureFileEnum の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureFileEnum 関数のデモンストレーションを行います。
*
* メモ : このスクリプト例を実行するには、
*       次の機能 (f)、サブ機能 (sf)、および コンポーネント (c)
*       を使ったプロジェクトを作成します :
*
*       (f) Program_Files
*          (c) Program_DLLS
*          (c) Program_EXEs
*       (f) Example_Files
*          (sf) Graphics
*             (c) Graphic_Examples
*
*       必ずファイルをコンポーネントへ割り当てて、
*       列挙できるようにして下さい。
*
¥*---------------------------------------------------------------*/

#define FEAT1       "Program Files"
#define FEAT2       "Example Files"
#define SUBFEAT1    "Graphics"
#define EXECCOMP    " プログラムの実行可能ファイル "
#define GRAPHCOMP   "Graphic の例 "
#define SDSHOWTITLE "FeatureFileEnum の結果 "

#define SDSHOWMSG1  FEAT1 + " 列挙したファイル :"
#define SDSHOWMSG2  FEAT2 + " 列挙したファイル :"

prototype HandleFeatureError (NUMBER);

NUMBER nResult;
LIST   listList1, listList2;
```

```
program

  // ファイル名を格納する 2 つのリストを作成します。
  listList1 = ListCreate (STRINGLIST);
  listList2 = ListCreate (STRINGLIST);

  // プログラムファイルのリストをビルドします。
  nResult = FeatureFileEnum (MEDIA, FEAT1, EXECCOMP + "¥¥*.*",
                             listList1, INCLUDE_SUBDIR);

  HandleFeatureError (nResult);

  // グラフィックファイルのリストをビルドします。
  nResult = FeatureFileEnum (MEDIA, FEAT2 + "¥¥" + SUBFEAT1,
                             GRAPHCOMP+"¥¥*.*", listList2, INCLUDE_SUBDIR);

  HandleFeatureError (nResult);

  // プログラムファイルを表示します。
  SdShowInfoList (SDSHOWTITLE, SDSHOWMSG1, listList1);

  // グラフィックファイルを表示します。
  SdShowInfoList (SDSHOWTITLE, SDSHOWMSG2, listList2);

  // メモリからリストをリリースします。

  ListDestroy (listList1);
  ListDestroy (listList2);

endprogram

/*---------------------------------------------------------------------------*¥
 *
 * 関数 :  HandleFeatureError
 *
 * 目的 :  この関数は、機能関数が戻した値を評価し、
 *         関数によって返された値を評価し、その値が 0 以下だとエラー
 *         番号を返してセットアップを終了します。
 *
/*---------------------------------------------------------------------------*¥

function HandleFeatureError (nResult)

  NUMBER  nvError;
  STRING  svFeatureSource, svFeature, svComponent, svFile;

begin
  if (nResult < 0) then

    FeatureError (svFeatureSource, svFeature, svComponent, svFile, nvError);

    SprintfBox (INFORMATION, " データ転送エラー情報 ",
          "FeatureError が、次のデータ転送エラーを返しました。¥n" +
          " セットアップは終了します。¥n¥n" +
        " メディア名 : %s¥n 機能 : %s¥n コンポーネント : %s¥n" +

          " ファイル : %s¥n エラー番号 : %ld",
          svFeatureSource, svFeature, svComponent, svFile, nvError);
```

```
        abort;

    endif;
end;
```

FeatureFileInfo

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

この関数は InstallScript オブジェクト プロジェクト内のファイルと一緒には使用できません。

FeatureFileInfo は、szFeatureSource (MEDIA) が参照するファイルメディア内のファイルについての情報を取得します。

メモ・この関数は、スクリプト作成機能と一緒に使用できません。

構文

FeatureFileInfo (szFeatureSource, szFeature, szFile, nInfo, nvResult, svResult);

パラメーター

テーブル 51 ・ FeatureFileInfo のパラメーター

パラメーター	説明
szFeatureSource	このパラメーターで MEDIA を渡します。
szFeature	ファイルが見つかったコンポーネントを含む機能を指定します。
szFile	パスを含んで、あるいは含まずにコンポーネントとファイル名を指定します。szFile 式のトークンを 2 つの円記号で区切り、式を 2 つの引用符で括ります。 ・ szFile に 2 つのトークンがある場合、最初のトークンはコンポーネント名、2 つ目はファイル名です : **“ コンポーネント ¥¥ ファイル名 ”** ・ szFile に 2 つ以上のトークンがある場合、最初のトークンはフコンポーネント名を、最後のトークンはファイル名を、そして中間のトークンはパスを表します : **″ コンポーネント ¥¥ パス ¥¥ ファイル名 ″** 例 : ・ ″Shared_Files¥¥Is5.dll″ – Shared_Files はコンポーネントで、Is5.dll はファイル名です。 ・ ″Shared_Files¥¥dev¥¥myapp¥¥dlls¥¥Is5.dll″ – Shared_Files はコンポーネント、¥¥dev¥¥myapp¥¥dlls はパス、そして Is5.dll はファイル名です。 ***メモ ・** InstallShield では、ファイルおよびフォルダー構造全体を、[ファイル] ビューにドラッグアンドドロップして、ファイルグループに追加することができます。*

テーブル 51・FeatureFileInfo のパラメーター (続き)

パラメーター	説明
nInfo	読み出す情報のタイプを指定します。このパラメーターで定数を渡します。使用可能な定数は、指定のコンポーネント、またはコンポーネント内の指定のファイルに適用します。 使用可能なファイル固有の定数は、以下の通りです : ・ FEATURE_INFO_ATTRIBUTE ― 指定のファイルの属性値。戻された File テーブル属性についての詳細は、Windows Installer ヘルプを参照して下さい。この定数は、InstallScript MSI インストールでのみサポートされています。InstallScript プロジェクトでは、サポートされていません。 ・ FEATURE_INFO_DATE ― 指定のファイルの日付値 (mm-dd-yy 形式)。この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。InstallScript MSI インストールで、この定数が FeatureFileInfo と共に使用されると、関数は失敗を返します。 ・ FEATURE_INFO_DATE_EX ― 指定のファイルの日付値 (mm-dd-yyyy 2000 年対応形式)。この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。InstallScript MSI インストールで、この定数が FeatureFileInfo と共に使用されると、関数は失敗を返します。 ・ FEATURE_INFO_TIME ― 指定のファイルの時刻値 (hh:mm 24 時間形式)。この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。InstallScript MSI インストールで、この定数が FeatureFileInfo と共に使用されると、関数は失敗を返します。 ・ FEATURE_INFO_VERSIONLS ― マイナーバージョン番号を文字列形式で表示。 ・ FEATURE_INFO_VERSIONMS ― メジャー バージョン番号を文字列形式で表示。 ・ FEATURE_INFO_VERSIONSTR ― 全バージョン番号を文字列形式で表示。 ・ FEATURE_INFO_MD5_SIGNATURE ― s z File によって指定されたファイルの MD5 シグネチャを返します。この情報はメディアのビルド中に生成されます。したがって、このパラメーターを使ってこの関数を呼び出すと、MD5 シグネチャは生成されません。 *メモ・特定のファイルに対する加工前の MD5 シグネチャ データは、それぞれが 16 ビット (0x00 および 0xFF の間) で生成された 16 の数値で構成されます。これらの値は通常、unsigned char 型の文字列に格納されます。ただし、InstallScript 言語は unsigned char 型をサポートしないため、加工前の MD5 ファイルデータはそのまま戻らず、各 16 数値は対応する文字列に変換され結果文字列に配置されます。このため、32 文字からなる文字列は 2 文字ずつの各セットが数値 1 つを意味します。これは、MD5 16 進法文字列として呼ばれることもあります。*

テーブル 51・FeatureFileInfo のパラメーター (続き)

パラメーター	説明
nInfo (続く)	• FEATURE_INFO_ORIGSIZE_HIGH― 元のファイルのサイズの上位 31 ビットを返します。ファイルの元のサイズが 2GB より大きい場合、ファイルのサイズは、FEATURE_INFO_ORIGSIZE_HIGH に戻された値を 2GB で掛けて、FEATURE_INFO_ORIGSIZE_LOW に戻された値に加えた値に等しくなります。この定数は、InstallScript インストールでのみサポートされています。 • FEATURE_INFO_ORIGSIZE_LOW― 元のファイルのサイズの下位 31 ビットを返します。ファイルの元のサイズが 2GB より大きい場合、ファイルのサイズは、FEATURE_INFO_ORIGSIZE_HIGH に戻された値を 2GB で掛けて、FEATURE_INFO_ORIGSIZE_LOW に戻された値に加えた値に等しくなります。 • FEATURE_INFO_COMPSIZE_LOW ― 圧縮ファイルのサイズの下位 31 ビットを返します。ファイルの圧縮されたサイズが 2 GB より大きい場合、ファイルのサイズは、FEATURE_INFO_COMPSIZE_HIGH に戻された値を 2 GB で掛けて、FEATURE_INFO_COMPSIZE_LOW に戻された値に加えた値に等しくなります。 • FEATURE_INFO_COMPSIZE_HIGH― 圧縮ファイルのサイズの上位 31 ビットを返します。ファイルの元のサイズが 2GB より大きい場合、ファイルのサイズは、FEATURE_INFO_ORIGSIZE_HIGH に戻された値を 2GB で掛けて、FEATURE_INFO_ORIGSIZE_LOW に戻された値に加えた値に等しくなります。この定数は、InstallScript インストールでのみサポートされています。 ***メモ・ConvertSizeToUnits** 関数を使用して、これらの値を KBYTES、MBYTES、GBYTES、または TBYTES の単一の値に変換できます。*

テーブル 51 • FeatureFileInfo のパラメーター (続き)

パラメーター	説明
nInfo (続く)	使用可能なコンポーネント固有の定数は、以下の通りです : • FEATURE_INFO_COMPONENT_FLAGS – 指定のコンポーネントに設定される一般フラグこの定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。 • FEATURE_INFO_DESTINATION – コンポーネントの " インストール先 " 設定。インストール先に含まれているテキスト置換はいずれも、戻された文字列では解決されていません。解決済みのテキスト置換を使ってコンポーネントのインストール先を決定する場合、戻された文字列で TextSubSubstitute を呼び出します。 • FEATURE_INFO_DOTNET – 指定のコンポーネントに選択された .NET フラグこの定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。 • FEATURE_INFO_FTPLOCATION – 指定のコンポーネントの "FTP の場所 " 設定。この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。 • FEATURE_INFO_HTTPLOCATION – 指定のコンポーネントの "HTTP の場所 " 設定。この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。 • FEATURE_INFO_LANGUAGE – 指定のコンポーネントの " 言語 " 設定。これはコンポーネントのみに当てはまるので、szFile はコンポーネントを認識する単一トークン式でなくてはなりません。 • FEATURE_INFO_MISC – 指定のコンポーネントの " その他 " 設定この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。 • FEATURE_INFO_OS – 指定のコンポーネントの " オペレーティング システム " 設定これはコンポーネントのみに当てはまるので、szFile はコンポーネントを識別する単一トークン式でなくてはなりません。この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。 • FEATURE_INFO_OVERWRITE – 指定のコンポーネントに設定される上書きフラグこの定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。 • FEATURE_INFO_PLATFORM_SUITE – 指定のコンポーネントの " プラットフォーム スイート " 設定これはコンポーネントのみに当てはまるので、szFile はコンポーネントを識別する単一トークン式でなくてはなりません。この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。

テーブル 51 ・ FeatureFileInfo のパラメーター (続き)

パラメーター	説明
nvResult	nInfo に渡された定数に依存する情報を指定します。 nInfo が FEATURE_INFO_ORIGSIZE_HIGH、FEATURE_INFO_ORIGSIZE_LOW、または FEATURE_INFO_ATTRIBUTE の時、nvResult は数値を返します。 nInfo が FEATURE_INFO_ATTRIBUTE の時、nvResult は Windows Installer **File** テーブルからのファイル属性を示す数値を返します。ファイル属性設定を決定するには、ビット単位 OR 演算子の結果を nvResult と一緒に使用します。nvResult のテスト方法例は、FeatureFileInfo の例を参照してください。File テーブル属性についての詳細は、Windows Installer ヘルプをご覧下さい。 nInfo が FEATURE_INFO_LANGUAGE または FEATURE_INFO_OS の場合、nvResult が占めるパラメーター位置に LIST 変数を指定して、言語またはオペレーティングシステム ID を保存する必要があります。**FeatureFileInfo** を呼び出す前に、定数 NUMBERLIST で ListCreate を呼び出して、コンポーネントで指定される言語またはオペレーティング システムを決定する必要があります。 listID = ListCreate(NUMBERLIST); FeatureFileInfo(szFeatureSource, szFeature, szFile, FEATURE_INFO_LANGUAGE, listID, svResult); nInfo が FEATURE_INFO_LANGUAGE の時、**FeatureFileInfo** は言語 ID をリストに追加し、nInfo が FEATURE_INFO_OS の場合、**FeatureFileInfo** はオペレーティング システム ID をリストに追加します。 ninfo が FEATURE_INFO_COMPONENT_FLAGS の時、nvResult は以下の値の 1 つまたは複数を返します : ・ FEATURE_INFO_COMPONENT_FLAG_COMPRESSED – コンポーネントの " 圧縮 " 設定が [はい] です。 ・ FEATURE_INFO_COMPONENT_FLAG_DATAASFILES – コンポーネントのファイルは、リリースの CD-ROM タイプ用の特定のフォルダーに配置されます。(そのコンポーネントを含む機能の "CD-ROM フォルダー " 設定によって決まります。) この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。 ・ FEATURE_INFO_COMPONENT_FLAG_DONTUNINSTALL – コンポーネントの " 圧縮 " 設定が [いいえ] です。 ・ FEATURE_INFO_COMPONENT_FLAG_ENCRYPTED – コンポーネントが暗号化されています。(そのコンポーネントを含む機能の " 暗号化 " 設定で [はい] が選択されているかどうかによって決まります。) ・ FEATURE_INFO_COMPONENT_FLAG_POTENTIALLYLOCKED – コンポーネントの " ファイルのロック " 設定が [はい] です。 ・ FEATURE_INFO_COMPONENT_FLAG_SELFREGISTERING – コンポーネントの " 自己登録 " 設定が [はい] です。 ・ FEATURE_INFO_COMPONENT_FLAG_SHARED – コンポーネントの " 共有 " 設定が [はい] です。

テーブル 51 ・ FeatureFileInfo のパラメーター (続き)

パラメーター	説明
nvResult (続き)	ninfo が FEATURE_INFO_OVERWRITE の時、nvResult は以下の値の 1 つまたは複数を返します : ・ FEATURE_VALUE_ALWAYS – コンポーネントの " 上書き " 設定が [常に上書き] です。 ・ FEATURE_VALUE_DATE_NEWER – コンポーネントの " 上書き " 設定が [新しい日付] です。 ・ FEATURE_VALUE_DATE_OLDER – コンポーネントの " 上書き " 設定が [古い日付] です。 ・ FEATURE_VALUE_DATE_SAME – コンポーネントの " 上書き " 設定が [同一または新しい日付] です。 ・ FEATURE_VALUE_NEVER – コンポーネントの " 上書き " 設定が [上書きしない] です。 ・ FEATURE_VALUE_VERSION_NEWER – コンポーネントの " 上書き " 設定が [新しいバージョン] です。 ・ FEATURE_VALUE_VERSION_OLDER – コンポーネントの " 上書き " 設定が [古いバージョン] です。 ・ FEATURE_VALUE_VERSION_SAME – コンポーネントの " 上書き " 設定が [同一または新しいバージョン] です。 nvResult に日付とバージョン値の両方が含まれている可能性もあります。 ninfo が FEATURE_INFO_DOTNET の場合、nvResult は以下の値を返します : FEATURE_VALUE_LOCAL_ASSEMBLY – コンポーネントは、ローカル アセンブリとしてマークされています。
svResult	nInfo が対応する情報の種類を指定した日付、時間、またはバージョンを返します。

戻り値

テーブル 52 ・ FeatureFileInfo の戻り値

戻り値	説明
0	FeatureFileInfo が成功しました。
<0	FeatureFileInfo が失敗しました。

FeatureFileInfo の例

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
```

```
*
* FeatureFileInfo 関数のデモンストレーションを行います。
*
* このスクリプトは FeatureFileInfo を繰り返し呼び出し、
* ファイルメディアについての情報を読み出します。情報はリストに
* 格納されてから表示されます。
*
* メモ : このスクリプト例を実行するには、
*      次の機能 (f)、サブ機能 (sf)、および コンポーネント (c)
*      を使ったプロジェクトを作成します :
*
*     (f) Example_Files
*         (sf) Graphics
*           (c) Graphic_Examples
*
*     列挙ができるよう、Graphic_Examples コンポーネントに
*  1 つ以上のファイルを確実に割り当ててください。
*  また、ファイル名を #define FILE 行で必ず指定
*  して下さい。
*
¥*---------------------------------------------------------------*/

#define FEAT        "Example Files"
#define SUBFEAT     "Graphics"
#define COMP   "Graphic の例 "
#define FILE        "comdlg32.dll"
#define SDSHOWTITLE  "ComponentFileInfo 結果 "
#define SDSHOWMSG    FILE + " 情報 :"

prototype HandleFeatureError (NUMBER);

NUMBER nReturn, nvResult;
STRING svResult;
LIST   listID;

program

  // ファイルメディア情報を格納するためのリストを作成します。
  listID = ListCreate (STRINGLIST);

  // 指定したファイルのオリジナルサイズを取得します。
  nReturn = FeatureFileInfo (MEDIA, FEAT + "¥¥" + SUBFEAT,
                             COMP +  "¥¥" + FILE,
                             FEATURE_INFO_ORIGSIZE , nvResult, svResult);

  HandleFeatureError (nvResult);

  // オリジナルサイズを文字列に変換します。
  Sprintf (svResult ,"%d", nvResult);

  // リストへ文字列を配置します。
  ListAddString (listID, " ファイルの元のサイズ : " +
                 svResult, AFTER);

  // 指定したファイルの属性を取得します。
  nReturn = FeatureFileInfo (MEDIA, FEAT + "¥¥" + SUBFEAT,
                             COMP +  "¥¥" + FILE,
                             FEATURE_INFO_ATTRIBUTE , nvResult, svResult);
```

```
HandleFeatureError (nReturn);

// 属性が設定されなかった場合、通常の属性を示します。
if (nvResult = FILE_ATTR_NORMAL) then
    svResult = " 通常 ";

// 属性が設定された場合、表示用に連結します。

else
  if (FILE_ATTR_ARCHIVED & nvResult) then
     svResult = " アーカイブ済み、";
  endif;

  if (FILE_ATTR_HIDDEN & nvResult) then
      svResult = svResult + " 非表示、";
  endif;

  if (FILE_ATTR_READONLY & nvResult) then
      svResult = svResult + " 読み取り専用、";
  endif;

  if (FILE_ATTR_SYSTEM & nvResult) then
     svResult = svResult + " システム、";
  endif;

  if (FILE_ATTR_DIRECTORY & nvResult) then
     svResult = svResult + " ディレクトリ、";

  endif;

endif;

// リストへ属性の文字列を追加します。
ListAddString (listID, " ファイルの属性は         " +
               svResult, AFTER);

// 指定したファイルのメジャーファイルバージョンを取得します。
nReturn = FeatureFileInfo (MEDIA, FEAT + "¥¥" + SUBFEAT,
                           COMP +  "¥¥" + FILE,
                           FEATURE_INFO_VERSIONMS , nvResult, svResult);
HandleFeatureError (nReturn);

// リストへメジャーファイルバージョンを追加します。
ListAddString (listID, "32-bit 上位バージョン値         " +
               svResult, AFTER);

// 指定したファイルのメジャーファイルバージョンを取得します。
nReturn = FeatureFileInfo (MEDIA, FEAT + "¥¥" + SUBFEAT,
                           COMP +  "¥¥" + FILE,
                           FEATURE_INFO_VERSIONLS , nvResult, svResult);
HandleFeatureError (nReturn);

// リストへマイナーファイルバージョンを追加します。

ListAddString (listID, "32-bit 下位バージョン値は       " +
               svResult, AFTER);

// 指定したファイルの完全ファイルバージョンを取得します。
nReturn = FeatureFileInfo (MEDIA, FEAT + "¥¥" + SUBFEAT,
```

```
                    COMP + "¥¥" + FILE,
                    FEATURE_INFO_VERSIONSTR , nvResult, svResult);

    HandleFeatureError (nReturn);

    // リストへ完全ファイルバージョンを追加します。
    ListAddString (listID, " ファイルのバージョン : " +

            svResult, AFTER);

    // リストを表示します。
    SdShowInfoList (SDSHOWTITLE, SDSHOWMSG, listID);

    // メモリからリストをリリースします。
    ListDestroy(listID);

endprogram

/*---------------------------------------------------------------------------*¥
 *
 * 関数 :  HandleFeatureError
 *
 *  目的 :  この関数は、機能関数が戻した値を評価し、
 *          関数によって返された値を評価し、その値が 0 以下だとエラー
 *          番号を返してセットアップを終了します。
 *
/*---------------------------------------------------------------------------*¥
function HandleFeatureError (nvResult)

  NUMBER  nvError;
  STRING  svFeatureSource, svFeature, svComponent, svFile;

begin
  if (nvResult < 0) then
    FeatureError(svFeatureSource, svFeature, svComponent, svFile, nvError);
    SprintfBox (INFORMATION, " データ転送エラー情報 ",
            "FeatureError が、次のデータ転送エラーを返しました。¥n" +

            " セットアップは終了します。¥n¥n" +
            " メディア名 : %s¥n 機能 : %s¥n コンポーネント : %s¥n" +
            " ファイル : %s¥n エラー番号 : %ld",
            svFeatureSource, svFeature, svComponent, svFile, nvError);
    abort;
  endif;
end;
```

FeatureFilterLanguage

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

FeatureFilterLanguage 関数は、言語に基づいてファイル転送からファイルをフィルター (除外) します。デフォルトでは、メディアビルドに含まれるすべての言語がフィルター解除 (インクルード) されます。フィルターまたはフィルター解除する各言語に対し、FeatureFilterLanguage を呼び出す必要があります。

メモ・この関数は、スクリプト作成機能セットでは使用できません。

言語固有コンポーネントをフィルターする

タスク ***インストレーションの最中に言語固有コンポーネントをフィルターするには:***

1. FeatureFilterLanguage をパラメーター nLangID を ISLANG_ALL に、bFiltered を TRUE に設定して呼び出し、すべての言語をフィルター (除外) します。
2. インストールする各言語に対して、nLangID を該当する言語定数に、パラメーター bFiltered を FALSE に設定して FeatureFilterLanguage を呼び出します。各呼び出しは nLangID で指定した言語のコンポーネントのフィルターを解除します (含みます)。

メモ・OR 演算子 (|) を使用して、nLangID パラメーターに複数の言語定数を指定することはできません。複数の言語定数を指定すると、関数が適切に動作しません。

別の言語をサポートする

InstallShield では、Windows でサポートされている任意の言語または言語サブグループ用にコンポーネントを指定することができます。しかし、[リリースウィザード] で言語固有のコンポーネントをビルドするためは、そのコンポーネントの言語用のサポートを所有していなくてはなりません。また、セットアップがコンポーネントの言語をサポートしていることも必要です。

InstallShield またはセットアップでサポートされていない言語に指定されている言語固有のコンポーネントをセットアップが含む場合、コンポーネントは [リリースウィザード] でフィルター (除外) されていなくてはなりません。

GetSystemInfo と共に FeatureFilterLanguage を利用する

GetSystemInfo 関数と共に FeatureFilterLanguage を使用する場合、言語固有のコンポーネントを指定する際に使用できる言語定数は、GetSystemInfo で戻される言語定数のサブセットである点を考慮する必要があります。

これらの戻り値に基づいて言語フィルタリングをセットアップにインクルードする場合、switch ステートメントを使って、この関数によって戻される定数を言語フィルタリングでサポートされている定数の 1 つに変換する必要があります。

構文

FeatureFilterLanguage (szFeatureSource, nLangID, bFiltered);

パラメーター

テーブル 53 - FeatureFilterLanguage のパラメーター

パラメーター	説明
szFeatureSource	ファイルメディアのメディア名を指定します。
nLangID	フィルタリングまたはフィルタ解除する言語の ID を指定します。各関数呼び出しで指定できる言語定数は *1* つのみです。すべての言語をフィルタリングするには、ISLANG_ALL をこのパラメーターに指定します。詳細は、「言語識別子」を参照してください。
bFiltered	nLangID で指定した言語をフィルタリングするかフィルタ解除 (インクルード) するかを示します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • TRUE ― nLangID で指定した言語をフィルタリングします。つまり、ファイル転送に含みません。 • FALSE ― nLangID で指定した言語をフィルタリングしません。つまり、ファイル転送に含みます。

戻り値

テーブル 54 - FeatureFilterLanguage の戻り値

戻り値	説明
0	FeatureFilterLanguage が成功しました。
< 0	FeatureFilterLanguage 失敗しました。

FeatureFilterLanguage の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureFilterLanguage 関数のデモンストレーションを行います。
*
* まず FeatureFilterLanguage が呼び出され、すべての言語が除外
* されます。次に GetSystemInfo が呼び出され、ターゲットコンピューターの
* デフォルトの言語 / ロケール を決定します。
*
* そして FeatureFilterLanguage が再び呼び出され、ターゲットコンピューターに
* 適切な言語を含みます。言語サポートが
* ターゲットコンピューターに備わっていない場合は英語が使用
* されます。最後に FeatureMoveData が呼び出され、インストールを
* 作成します。
*
¥*---------------------------------------------------------------*/
```

```
// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_FeatureFilterLanguage(HWND);

function ExFn_FeatureFilterLanguage(hMSI)
   STRING  szResult;
   NUMBER  nResult, nDisk;
begin

   // すべての言語固有ファイルグループをフィルタリングする。
   FeatureFilterLanguage (MEDIA, ISLANG_ALL, TRUE);

   // ターゲットマシンのデフォルトの言語またはロケール設定を読み出します。
   GetSystemInfo (LANGUAGE, nResult, szResult);

   // ターゲットマシンのデフォルトの言語またはロケール設定に特定の
   // ファイルグループのフィルタリングを無効にします。

   switch (nResult)
      case ISLANG_FRENCH_CANADIAN:
         FeatureFilterLanguage (MEDIA, ISLANG_FRENCH_CANADIAN, FALSE);
      case ISLANG_FRENCH_STANDARD, ISLANG_FRENCH_BELGIAN, ISLANG_FRENCH_SWISS, ISLANG_FRENCH_LUXEMBOURG:
         FeatureFilterLanguage (MEDIA, ISLANG_FRENCH_STANDARD, FALSE );
      case ISLANG_GERMAN_STANDARD, ISLANG_GERMAN_SWISS, ISLANG_GERMAN_AUSTRIAN, ISLANG_GERMAN_LUXEMBOURG,
ISLANG_GERMAN_LIECHTENSTEIN:
         FeatureFilterLanguage (MEDIA, ISLANG_GERMAN, FALSE);
      // 英語をデフォルトとして利用します。.
      デフォルト :
         FeatureFilterLanguage (MEDIA, ISLANG_ENGLISH_UNITEDSTATES, FALSE);
   endswitch;

   // 選択された言語用のファイルを転送します。
   FeatureMoveData (MEDIA, nDisk, 0);

end;
```

FeatureFilterOS

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

FeatureFilterOS 関数は、指定のオペレーティングシステムおよびスイート用にフラグが付けられたコンポーネントをフィルタリングします。

メモ・この関数は、スクリプト作成機能セットでは使用できません。

初回のインストールで行われるフィルタリングは、メンテナンスモード中にも行う必要があります。初回およびメンテナンス セットアップ両方で実行されるコードで必ずこの関数を呼び出して下さい。この関数は、イベントハンドラー OnAppSearch、OnCCPSearch、OnFirstUIBefore、OnFirstUIAfter、OnMaintUIBefore、または OnMaintUIAfter からは呼び出さないでください。

プロジェクト・*InstallScript MSI インストールでは、デフォルトでコンポーネントはフィルタリングされません。*

*InstallScript インストールでは、****FeatureFilterOS*** *が、初回インストール中およびメンテナンス インストール中に、デフォルトの OnFilterComponents イベント ハンドラーで呼び出されます。このイベントハンドラーは、ターゲットシステムに存在しないオペレーティング システムおよびスイート固有のコンポーネントをデフォルトでフィルタリングします。メンテナンス モードで実行されるインストールは、初回インストール中に実行されるたフィルタリングについての情報を持ちません。*

構文

```
FeatureFilterOS ( szMediaLibrary, nSuites, nOS, bFiltered );
```

パラメーター

テーブル 55 ・ FeatureFilterOS のパラメーター

パラメーター	説明
szMediaLibrary	ファイル メディア ライブラリのメディア名を指定します。
nSuites	フィルタリングするオペレーティングシステム スイートを指定します。以下の値から選択します。ビット単位 OR 演算子 (|) を使用して、値を組み合わせることができます。 ***メモ ・ ここにリストされるスイートは、Windows API の OSVERSIONINFOEX データ構造で指定することができるものです。*** ・ ISOS_ST_ALL ― すべての Windows スイート ・ ISOS_ST_XP_PRO ― Windows XP Professional ・ ISOS_ST_XP_HOME ― Windows XP Home Edition ・ ISOS_ST_SERVER ― Windows Server ・ ISOS_ST_WORKSTATION ― Windows Workstation ・ ISOS_ST_BACKOFFICE ― Microsoft BackOffice ・ ISOS_ST_DATACENTER ― Windows Server Datacenter ・ ISOS_ST_ENTERPRISE ― Windows Server Enterprise ・ ISOS_ST_SERVER2003_R2 ― Microsoft Windows Server 2003 R2 ・ ISOS_ST_SMALLBUSINESS ― Microsoft Small Business Server ・ ISOS_ST_SMALLBUSINESS_RESTRICED ― 制限つきクライアント ライセンスのある Microsoft Small Business Server ・ ISOS_ST_TERMINAL ― Microsoft Terminal Services ・ ISOS_ST_PROC_ARCH_32 ― 32 ビットプロセッサ ・ ISOS_ST_PROC_ARCH_IA64 ― Intel Itanium 64 ビットプロセッサ ・ ISOS_ST_PROC_ARCH_AMD64 ― Intel Itanium 64 ビットプロセッサ ・ 0 (ゼロ) ― FeatureFilterOS がコンポーネントのプラットフォームスイートのプロパティを無視するように指定します。 ***プロジェクト ・ InstallScript MSI プロジェクトでは、プラットフォーム スイートがサポートされていません。そのため、InstallScript MSI プロジェクトでは、nSuites. に数値 0 を指定する必要があります。そうでない場合、関数が失敗して、nOS で指定された情報が無視されます。***

テーブル 55 ・ FeatureFilterOS のパラメーター (続き)

パラメーター	説明
nOS	フィルタリングするオペレーティング システムを指定します (複数可)。以下の値から選択します。ビット単位 OR 演算子 (|) を使用して、値を組み合わせることができます。 ・ ISOSL_ALL ー すべての Windows システム ・ ISOSL_WINXP Ñ Windows XP Edition ・ ISOSL_WINSERVER2003 Ñ Windows Server 2003 ・ ISOSL_WINVISTA_SERVER2008 (or ISOSL_WINVISTA) ー Windows Vista または Windows Server 2008 ・ ISOSL_WIN7_SERVER2008R2 Ñ Windows 7 または Windows Server 2008 R2 ・ ISOSL_WIN8—Windows Vista または Windows Server 8 ・ ISOSL_WIN81 ー Windows 8.1 または Windows Server 2012 R2 ・ ISOSL_WIN10—Windows 10 ***メモ ・*** *Windows のいくつかのクライアントおよびサーバー バージョンでは、同じメジャー バージョン番号とマイナー バージョン番号が使用されています。* ・ *Windows 8.1 と Windows Server 2012 R2 では、同じメジャー バージョン番号とマイナー バージョン番号が使用されています。* ・ *Windows 8 と Windows Server 2012 では、同じメジャー バージョン番号とマイナー バージョン番号が使用されています。* ・ *Windows 7 と Windows Server 2008 R2 では、同じメジャー バージョン番号とマイナー バージョン番号が使用されています。* ・ *Windows Vista と Windows Server 2008 は、同じメジャー バージョン番号とマイナー バージョン番号を持ちます。* *このため、インストールの実行時、これらの OS バージョンでは、クライアント バージョンは、同等のサーバー バージョンと同じものと見なされます。したがって、クライアント バージョン向けとマークされているコンポーネントは、サーバー バージョンにもインストールされます。クライアント バージョンとサーバー バージョンを区別するには、SYSINFO.nOSProductType と VER_NT_WORKSTATION が等しいかどうか確認します。クライアント バージョンでは、これらは等しくなっています (True)。サーバー バージョンでは、False です。*
bFiltered	nOS で指定したオペレーティングシステムを、フィルタリング (除外) するかフィルタ解除 (インクルード) するかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ TRUE ー 指定したオペレーティング システム (複数指定可) をフィルタリングします。つまり、それらをファイル転送には含みません。 ・ FALSE ー 指定したオペレーティングシステム (複数指定可) をフィルタリングしません。つまり、それらをファイル転送に含みます。

戻り値

テーブル 56 · FeatureFilterOS の戻り値

戻り値	説明
0	**FeatureFilterOS** が成功しました。
< 0	**FeatureFilterOS** が失敗しました。追加情報については、FeatureError を呼び出してください。

追加情報

ISOSL_ALL 及び ISOSL_SUPPORTED 両方がすべての Windows システムを指定する場合に、**FeatureFilterOS** へ 3 番目の引数として渡された場合、それらは異なる数値を持ちます。ISOSL_ALL がゼロ (0) で、ISOSL_SUPPORTED はビット単位の OR 演算子を利用して各オペレーティングシステム定数を組み合わせて取得した値です。つまり、ISOSL_WIN7_SERVER2008R2|ISOSL_WINVISTA_SERVER2008|... です。. 場合によって、ISOSL_SUPPORTED やその他のオペレーティング システム定数でビット単位の演算子を利用すると便利です。

FeatureFilterOS の例

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * FeatureFilterOS 関数のデモンストレーションを行います。
 *
 * このスクリプトを実行するには、空のセットアッププロジェクトを作成します。
 * [ コンポーネント ] ビューのオペレーティングシステム に、
 * Windows 95、Windows 98、そして Windows NT 4.0 (Intel) を指定します。
 * リリースフラグ を使ってこのプラットフォームをビルドに含むかを指定
 * します。Windows 95、Windows 98、および Windows NT 4.0 (Intel) を
 * 指定します。[ リリースウィザード ] を実行するとき、[ フィルター設定 ] パネルに
 * リリースフラグを入力します。
 *
¥*---------------------------------------------------------------*/

// 次の define ステートメントを文字列エントリに変換して
// セットアップをローカライズできます。
#define  COMPANY_NAME        "MultiLangOS Inc."
#define  PRODUCT_NAME        "MultiLangOS"
#define  PRODUCT_VERSION      "1.0"
#define  PROGRAMFOLDER        "MultiLangOS"
#define  PRODUCT_KEY         "Mlangos.exe"
#define  DEINST_KEY          "MultiLangOS"
#define  ASKDESTTITLE         " インストール先 "
#define  ASKDESTMSG           " インストール先を選択してください。"
#define  COMPERRTITLE         " データ転送エラー情報 "
#define  COMPERRMSG1          "FeatureError が次のエラーを返しました。"
#define  COMPERRMSG2          " セットアップは終了します。"
#define  COMPERRMSG3          " メディア名 :"
#define  COMPERRMSG4          " 機能 :"
#define  COMPERRMSG4          " コンポーネント :"
#define  COMPERRMSG6          " ファイル :"
#define  COMPERRMSG7          " エラー番号 :"
```

```
    prototype HandleFeatureError (NUMBER);

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

     export prototype ExFn_FeatureFilterOS(HWND);

function ExFn_FeatureFilterOS(hMSI)
    STRING  svResult, svDir, svLogFile;
    NUMBER  nvResult, nvDisk;
begin

    // セットアップダイアログで [ 戻る ] ボタンを無効にします。
    Disable(BACKBUTTON);

    // アンインストールをセットアップしてインストール先のを取得します。
    InstallationInfo(COMPANY_NAME, PRODUCT_NAME, PRODUCT_VERSION, PRODUCT_KEY);

    INSTALLDIR = PROGRAMFILES ^ PROGRAMFOLDER;

    AskDestPath (ASKDESTTITLE, ASKDESTMSG, INSTALLDIR , 0 );

    DeinstallStart (INSTALLDIR, svLogFile, DEINST_KEY, 0);

    RegDBSetItem (REGDB_UNINSTALL_NAME, DEINST_KEY);

    // オペレーティングシステムを取得して FeatureFilterOS を呼び出し、
    // 現在存在しないオペレーティングシステムをフィルタリングします。
    GetSystemInfo (OS, nvResult, svResult);

    switch (nvResult)
        case IS_WINDOWSNT:
            GetSystemInfo (WINMAJOR, nvResult, svResult);
            if (nvResult = 4) then
                // NT 4.0 なので、Windows 95 と Windows 98 をフィルタリングします。
                FeatureFilterOS (MEDIA, 0, ISOSL_WIN95, TRUE);
                FeatureFilterOS (MEDIA, 0, ISOSL_WIN98, TRUE);
            else
                MessageBox (" ターゲット システム OS がサポートされていません。", SEVERE);
                abort;
            endif;
        case IS_WINDOWS9X:
            // Windows 95 または 98 なので、Windows NT 4,0 をフィルタリングします。
            FeatureFilterOS (MEDIA, 0, ISOSL_NT40, TRUE);

            // Windows 95 か Windows 98 かを決定します。
            GetSystemInfo (WINMINOR, nvResult, svResult);

            if (nvResult < 10) then
                // Windows 95 なので、Windows 98 もフィルタリングします。
                FeatureFilterOS (MEDIA, 0, ISOSL_WIN98, TRUE);
            else
                // Windows 98 なので、Windows 95 もフィルタリングします。
                FeatureFilterOS (MEDIA, 0, ISOSL_WIN95, TRUE);
            endif;
        デフォルト :
            MessageBox (" ターゲット システム OS がサポートされていません。", SEVERE);
            abort;
```

```
    endswitch;

    // ファイルをターゲットシステムに転送します。
    nvResult = FeatureMoveData (MEDIA, nvDisk, 0);

    if (nvResult < 0) then
        HandleFeatureError (nvResult);
    endif;

end;

/*---------------------------------------------------------------------------*¥
 *
 * 関数 :  HandleFeatureError
 *
 * 目的 :  この関数は、機能関数が戻した値を評価し、
 *          関数によって返された値を評価し、その値が 0 以下だとエラー
 *          番号を返してセットアップを終了します。
 *
/*---------------------------------------------------------------------------*¥
function HandleFeatureError (nResult)
  NUMBER  nvError;
  STRING  svFeatureSource, svFeature, svComponent, svFile;
begin
  FeatureError (svFeatureSource, svFeature, svComponent, svFile, nvError);
  SprintfBox(INFORMATION, FEATERRTITLE,
            FEATERRMSG1 + "¥n" + FEATERRMSG2 + "¥n¥n" +
            FEATERRMSG3 + " %s¥n" + FEATERRMSG4 + " %s¥n" +
            FEATERRMSG5 + " %s¥n" + FEATERRMSG6 + " %s¥n" +
            FEATERRMSG7 + " %ld",
            svFeatureSource, svFeature, svComponent, svFile, nvError);
  abort;
end;
```

FeatureGetCost

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

FeatureGetCost 関数は古い形式です。代わりに、FeatureGetCostEx 関数を使用してください。

FeatureGetCost 関数は szFeature が指定した機能のためにターゲットドライブ上で必要な総容量をキロバイト (KB) 単位で判断します。必要ドライブスペースを判断する際、この関数は機能が現在選択されているかどうか、機能に関連付けられているコンポーネントが現在オペレーティングシステムまたは言語によってフィルターされているかどうか、およびターゲットドライブ上のクラスタサイズを考慮します。

構文

```
FeatureGetCost ( szFeatureSource, szFeature, szTargetDir, nvRequiredSpace );
```

パラメーター

テーブル 57 · FeatureGetCost のパラメーター

パラメーター	説明
szFeatureSource	インストールおよびアンインストールに対して機能が指定されているファイル メディアのメディア名を指定します。通常は、システム変数の MEDIA をこのパラメーターで渡します。
szFeature	ターゲットドライブ上でそれに必要な総容量を判断したい機能を指定します。
szTargetDir	必要なドライブの空き容量を判断する際に、使用されるドライブまたはそのパスを指定します。通常は、システム変数の TARGETDIR をこのパラメーターで渡します。
nvRequiredSpace	必要なドライブの空き容量を KB 単位で返します。

戻り値

テーブル 58 · FeatureGetCost の戻り値

戻り値	説明
0	必要なドライブ空き容量が正常に判断されたことを示します。
< 0	必要なドライブ空き容量が正常に判断できなかったことを示します。 大きな負の戻り値に関連付けられたエラー メッセージ テキストを取得できます。たとえば、FormatMessage を呼び出して -2147024891 (0x80070005)。

FeatureGetCost の例

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureGetCost 関数のデモンストレーションを行います。
*
* このスクリプト例は、ユーザーが選択した各トップレベル機能の
* サイズをユーザーに対して示すダイアログ ボックスを表示します。
*
* コメント : このスクリプト例を実行するには、いくつかの機能および
*          / またはファイルを持つコンポーネントを含むサブ機能を
*          持つプロジェクトを作成 ( またはプロジェクトに挿入 ) します。
*
*
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
    string svDir, svFeature, svFeatureInfo;
```

```
    LIST listFeatures, listFeatureInfo;
    number nListGetString, nvRequiredSpace;
begin
    svDir = TARGETDIR;
    SdFeatureTree ("", "", svDir, "", 1);

    listData = ListCreate (STRINGLIST);
    FeatureListItems ( MEDIA, "", listFeatures );
    nListGetString = ListGetFirstString ( listFeatures, svFeature );
    listFeatureInfo = ListCreate ( STRINGLIST );
    while nListGetString=0
        FeatureGetCost ( MEDIA, svFeature, svDir, nvRequiredSpace );
        Sprintf ( svFeatureInfo, " 選択した機能 %s のサイズは %ld KB です。¥n",
            svFeature, nvRequiredSpace );
        ListAddString ( listFeatureInfo, svFeatureInfo, AFTER );
        nListGetString = ListGetNextString ( listFeatures, svFeature );
    endwhile;
    ListDestroy ( listFeatures );

    SdShowInfoList ( "", "", listFeatureInfo );
    ListDestroy ( listFeatureInfo );
end;
```

FeatureGetCostEx

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

FeatureGetCostEx 関数は、nvCostHigh および nvCostLow パラメーターを使用して、指定された機能のコストをバイト単位で返します。szFeature が "" のとき、メディア全体のコストが戻されます。この関数は **FeatureGetCost** 関数と **FeatureGetTotalCost** 関数を置換します。

構文

```
FeatureGetCostEx ( szMediaSource, szFeature, szTarget, nvCostHigh, nvCostLow );
```

パラメーター

テーブル 59 - FeatureGetCostEx のパラメーター

パラメーター	説明
szMediaSource	メディアソース。通常、MEDIA。
szFeature	コストを取得する機能を指定します。この値が "" のとき、関数はメディア全体のコストを返します。
szTarget	szFeature が指定した機能のターゲットディレクトリを指定します。
nvCostHigh	機能のコストの上位 31 ビットを返します。戻された各コストの単位は、2 GB です。たとえば、この変数の値 4 は、コストが 8GB であることを表します。
nvCostLow	機能のコストの下位 31 ビットを返します。この変数の最大値は、1 または 2GB です。これより大きいコストはすべて、nvCostHigh 変数で戻されます。

戻り値

テーブル 60 - FeatureGetCostEx の戻り値

戻り値	説明
ISERR_SUCCESS	関数が成功したことを示します。
< ISERR_SUCCESS	関数の実行に失敗したことを示します。

FeatureGetData

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

- *InstallScript*
- *InstallScript MSI*

FeatureGetData 関数は、機能に関する情報を読み出します。

構文

```
FeatureGetData ( szFeatureSource, szFeature, nInfo, nvResult, svResult );
```

パラメーター

テーブル 61 ・ FeatureGetData のパラメーター

パラメーター	説明
szFeatureSource	スクリプト作成機能セットまたは (InstallScript プロジェクトの場合は) 情報の読み出し先となる機能を含むファイルメディアライブラリのメディア名を指定します。
szFeature	情報を読み出す機能の名前を指定します。トップ階層の機能やサブ機能の参照方法は、「関数呼び出しで機能やサブ機能を指定する」を参照してください。
nInfo	読み出す情報の種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ FEATURE_FIELD_CDROM_FOLDER – CD-ROM メディアタイプの場合、CD-ROM 上での機能データの場所。この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。この定数は、ファイル メディア ライブラリで使用できますが、スクリプト作成の機能セットでは使用できません。 ・ FEATURE_FIELD_DESCRIPTION – 機能選択ダイアログで機能が選択されたを時に表示される説明。これは機能の ″ 説明 ″ プロパティからのテキストです。 ・ FEATURE_FIELD_DISPLAYNAME– 選択するダイアログに表示する機能名。これは機能の ″ 表示名 ″ 設定の値です。 ・ FEATURE_FIELD_ENCRYPT – 機能が暗号化されているかどうかを指定します。この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。この定数は、ファイル メディア ライブラリで使用できますが、スクリプト作成の機能セットでは使用できません。 ・ FEATURE_FIELD_FILENEED– インストールにおける機能ファイルの重要度を定義します。この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。 ・ FEATURE_FIELD_FLAGS – 機能に設定されているフラグを示します。この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。この定数は、ファイル メディア ライブラリで使用できますが、スクリプト作成の機能セットでは使用できません。 ・ FEATURE_FIELD_FTPLOCATION – FTP ロケーションを指定します。この値は機能の ″FTP の場所 ″ 設定に格納されています。 ・ FEATURE_FIELD_GUID – 機能と関連付けられている GUID。この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。この定数は、ファイル メディア ライブラリで使用できますが、スクリプト作成の機能セットでは使用できません。

テーブル 61 ・ FeatureGetData のパラメーター (続き)

パラメーター	説明
nInfo (続く)	• FEATURE_FIELD_HANDLER_ONINSTALLED – 機能の OnInstalled イベントの名前。この定数は、ファイル メディア ライブラリで使用できますが、スクリプト作成の機能セットでは使用できません。 • FEATURE_FIELD_HANDLER_ONINSTALLING – 機能の OnInstalling イベントの名前。この定数は、ファイル メディア ライブラリで使用できますが、スクリプト作成の機能セットでは使用できません。 • FEATURE_FIELD_HANDLER_ONUNINSTALLED – 機能の OnUninstalled イベントの名前。この定数は、ファイル メディア ライブラリで使用できますが、スクリプト作成の機能セットでは使用できません。 • FEATURE_FIELD_HANDLER_ONUNINSTALLING – 機能の OnUninstalling イベントの名前。この定数は、ファイル メディア ライブラリで使用できますが、スクリプト作成の機能セットでは使用できません。 • FEATURE_FIELD_HTTPLOCATION – HTTP ロケーションを指定します。この値は機能の "HTTP の場所 " 設定に格納されています。 • FEATURE_FIELD_IMAGE – nvResult で表示されるアイコンのインデックスを指定します。機能にアイコンを表示しない場合、nvResult は -1 を返します。 • FEATURE_FIELD_MISC – その他のテキスト。このフィールドは、任意の情報を使用して機能にフラグを立てたり識別したりできるので、ランタイムに非常に有効です。 • FEATURE_FIELD_PASSWORD – 機能にパスワードが関連付けられているかどうかを指定します。この定数は、ファイル メディア ライブラリで使用できますが、スクリプト作成の機能セットでは使用できません。この定数は、InstallScript インストールでのみサポートされています。InstallScript MSI プロジェクトでは、サポートされていません。 • FEATURE_FIELD_SELECTED – szFeature で指定された機能が選択されているかどうかを示します。 • FEATURE_FIELD_SIZE–szFeature で指定した機能に対する元のファイルサイズの合計。この定数は、ファイル メディア ライブラリで使用できますが、スクリプト作成の機能セットでは使用できません。 • FEATURE_FIELD_STATUS – InstallScript インストールで、このテキストはファイル転送中に進行状況インジケーターに表示されます。この定数は、ファイル メディア ライブラリで使用できますが、スクリプト作成の機能セットでは使用できません。InstallScript MSI インストールでは、機能の表示名が戻されます。 • FEATURE_FIELD_VISIBLE–szFeature で指定された機能を選択ダイアログに表示するかどうかを決定します。この値は機能の " 表示 " 設定に格納されています。

テーブル 61 ・ FeatureGetData のパラメーター (続き)

パラメーター	説明
nvResult	nInfo が数値結果を生成した場合に数値を返します。 nInfo が FEATURE_FIELD_FILENEED の時、nvResult は以下の値の 1 つを返します : ・ FEATURE_VALUE_CRITICAL – この機能には重要なファイルが含まれています。 ・ FEATURE_VALUE_HIGHLYRECOMMENDED – この機能は強く推奨されます。 ・ FEATURE_VALUE_STANDARD – この機能は含まれる場合と含まれない場合があります。 nInfo が FEATURE_FIELD_FLAGS の時、nvResult は以下の値の 1 つを返します : ・ FEATURE_DATA_FLAG_PASSWORD – 機能はパスワードで保護されています。 ・ FEATURE_DATA_FLAG_PASSWORD_VALIDATED – 機能のパスワードが検証済みです。(インストールが **FeatureValidate** を使用しているか、機能がパスワードで保護されていません。) ・ FEATURE_DATA_FLAG_DATA_AS_FILES – 機能のファイルは、リリースの CD-ROM タイプ用の特定のフォルダーに配置されます。 ・ FEATURE_DATA_FLAG_SPLIT_AFTER – このフラグは、ビルドが生成した機能にのみ設定されます。 ・ FEATURE_DATA_FLAG_SPLIT_BEFORE – このフラグは、ビルドが生成した機能にのみ設定されます。 ・ FEATURE_DATA_FLAG_SPLIT_BEFORE_NOT_ALLOWED – このフラグは、ビルドが生成した機能にのみ設定されます。 ・ FEATURE_DATA_FLAG_SPLIT_NOT_ALLOWED – このフラグは、ビルドが生成した機能にのみ設定されます。 ・ FEATURE_DATA_FLAG_VISIBLE – 機能は表示されます。 ・ FEATURE_DATA_FLAG_VOLATILE – 機能は可変です。 ・ FEATURE_DATA_FLAG_ENCRYPTED – 機能は暗号化されています。

テーブル 61 ・ FeatureGetData のパラメーター (続き)

パラメーター	説明
	nInfo が FEATURE_FIELD_PASSWORD の時、nvResult は以下の値の 1 つを返します : ・ TRUE – 機能はパスワードで保護されています。機能がパスワードで保護されている場合、エンドユーザーから正しいパスワードを取得し、**FeatureTransferData** を呼び出してファイルメディアライブラリのファイルを転送する前に **FeatureValidate** でそれを検証する必要があります。 ・ FALSE – 機能はパスワードで保護されていません。 nInfo が FEATURE_FIELD_SELECTED の時、nvResult は以下の値の 1 つを返します : ・ TRUE – この機能は選択されます。 ・ FALSE – この機能は選択されません。 nInfo が FEATURE_FIELD_VISIBLE の時、nvResult は以下の値の 1 つを返します : ・ TRUE – この機能は表示されます。 ・ FALSE – この機能は表示されません。
svResult	nInfo が文字列結果を生成した場合に文字列値を返します。

戻り値

テーブル 62 ・ FeatureGetData の戻り値

戻り値	説明
0	FeatureGetData が成功しました。
< 0	FeatureGetData が失敗しました。追加情報については、FeatureError を呼び出してください。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

FeatureGetData の例

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureGetData 関数のデモンストレーションを行います。
*
* このスクリプト例は、指定した機能についての情報を読み出す
* 関数のデモンストレーションを行います。
*
*  コメント : このスクリプト例を実行するには、いくつかの機能および
*           またはファイルを含むコンポーネントを持つサブ機能を含む
プロジェクトを作成 ( またはプロジェクトに挿入 ) します。
```

```
 *
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

// Windows Installer API 関数プロトタイプと定数用に iswi.h をインクルードし、
// OnBegin イベントと OnEnd イベント用のコードを宣言します。
#include "iswi.h"

    // キーワードのエクスポートは MyFunction () をエントリポイント関数として識別します。
    // 使用する引数は、Installer データベースへのハンドルでなくてはなりません。
    export prototype MyFunction(HWND);

    // 必要な処理 : グローバル変数の宣言、定数の定義、およびユーザー定義の関数並びに DLL 関数を
    //        ここでプロトタイプ化します。

function MyFunction(hMSI)

    STRING  svDir, szTitle, szMsg, svResult;
    NUMBER  nvResult;

begin

    svDir   = INSTALLDIR;
    szTitle = " 機能の選択 ";
    szMsg   = " コンピューターへインストールする機能を選択してください。";

    // 利用可能な機能を表示します。
    SdFeatureTree (szTitle, szMsg, svDir, "", 2);

    // Feature1 の説明プロパティを取得します。
    FeatureGetData (MEDIA, "Feature1", feature_FIELD_DESCRIPTION, nvResult, svResult);

    // Feature1 の説明を取得します。
    MessageBox ("Feature1 の説明 : " + svResult, INFORMATION);

    // Feature 2 が選択されたかどうかを決定します。
    FeatureGetData (MEDIA, "Feature2", feature_FIELD_SELECTED, nvResult, svResult);

    // Feature 2 が選択されたかどうかを示すメッセージを表示します。
    if nvResult=0  then
      MessageBox ("Feature2 が選択されていません。", INFORMATION);
    else
      MessageBox ("Feature2 が選択されています。", INFORMATION);
    endif;

end;
```

FeatureGetItemSize

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

FeatureGetItemSize 関数は、指定の機能のサイズをバイトで取得します。ここにはサブ機能のサイズは含まれません。

メモ・この関数は、スクリプト作成機能セットでは使用できません。

構文

FeatureGetItemSize (szFeatureSource, szFeature, nvSize);

パラメーター

テーブル 63 ・ FeatureGetItemSize のパラメーター

パラメーター	説明
szFeatureSource	サイズを読み取る機能が含まれるファイルメディアの名前を指定します。
szFeature	サイズが読み出される機能の名前を指定します。
nvSize	指定された機能のサイズをバイトで返します。また、FeatureGetData を呼び出して、指定された機能のオリジナルファイルサイズの合計値を取得することもできます。選択したすべての機能とサブ機能の合計サイズを判断するには、FeatureTotalSize を呼び出してください。

戻り値

テーブル 64 ・ FeatureGetItemSize の戻り値

戻り値	説明
0	FeatureGetItemSize が成功しました。
< 0	FeatureGetItemSize が失敗しました。追加情報については、FeatureError を呼び出してください。

FeatureGetItemSize の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureGetItemSize 関数のデモンストレーションを行います。
*
* この例では FeatureGetItemSize を呼び出して機能とサブ機能の
* サイズを取得します。サイズはダイアログ ボックスに
* 表示されます。
*
* メモ : このスクリプト例を実行するには、ファイルを含むコンポーネントを持つ
*       いくつかの機能および / またはサブ機能を含む
*       プロジェクトを作成 ( またはプロジェクトに挿入 ) します。
*        この例の #define ステートメントで示されたとおり、
```

```
*       機能とサブ機能をひとつずつ指定しなくてはなりません。
*       または、機能名を参照するよう、#define ステートメントを
*       変更します。
*
¥*---------------------------------------------------------------*/

#define FEAT_NAME1 " プログラムファイル "
#define FEAT_NAME2 "Example Files¥¥Graphics"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_FeatureGetItemSize(HWND);

function ExFn_FeatureGetItemSize(hMSI)
   NUMBER nvSize;
   STRING szString;
   LIST listInfo;
begin

   // 機能サイズを格納するための文字列リストを作成します。
   listInfo = ListCreate (STRINGLIST);

   // FEAT_NAME1 のサイズを取得します。
   FeatureGetItemSize (MEDIA, FEAT_NAME1, nvSize);

   // 数値を文字列に変換します。
   NumToStr (szString, nvSize);

   // リストへ文字列を配置します。
   ListAddString (listInfo, "" + FEAT_NAME1 +
            ":" + szString, AFTER);

   // FEAT_NAME2 のサイズを取得します。
   FeatureGetItemSize (MEDIA, FEAT_NAME2, nvSize);

   // 数値を文字列に変換します。
   NumToStr (szString, nvSize);

   // リストへ文字列を配置します。
   ListAddString (listInfo, "" + FEAT_NAME2 +
            ":" + szString, AFTER);

   // 機能サイズのリストを表示します。
   SdShowInfoList ("FeatureGetItemSize への呼び出し結果 ",
            " 機能サイズは :", listInfo);

   // メモリからリストをリリースします。
   ListDestroy (listInfo);

end;
```

FeatureGetTotalCost

メモ・この関数は現在使用されていません。代わりに、FeatureGetCostEx 関数を使用してください。

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FeatureGetTotalCost 関数は、機能ダイアログまたはセットアップの種類ダイアログでエンドユーザーが選択するなどして指定された機能のインストールやアンインストールの際に、ターゲットドライブで必要な合計空き容量のキロバイト数 (KB) を判断します。必要ドライブスペースを判断する際、この関数はどの機能が現在選択されているのか、機能に関連付けられているコンポーネントが現在オペレーティングシステムまたは言語によってフィルターされているかどうか、およびターゲットドライブ上のクラスタサイズを考慮します。

構文

```
FeatureGetTotalCost ( szFeatureSource, szTargetDir, nvTotalRequiredSpace );
```

パラメーター

テーブル 65 - FeatureGetTotalCost のパラメーター

パラメーター	説明
szFeatureSource	インストールおよびアンインストールに対して機能が指定されているファイル メディアのメディア名を指定します。通常は、システム変数の MEDIA をこのパラメーターで渡します。
szTargetDir	必要なドライブの空き容量を判断する際に、使用されるドライブまたはそのパスを指定します。通常は、システム変数の TARGETDIR をこのパラメーターで渡します。
nvTotalRequiredSpace	必要なドライブの空き容量を KB 単位で返します。

戻り値

テーブル 66 - FeatureGetTotalCost の戻り値

戻り値	説明
0	必要なドライブ空き容量が正常に判断されたことを示します。
< 0	必要なドライブ空き容量が正常に判断できなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

FeatureInitialize

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

FeatureInitialize 関数は、以前のバージョンの InstallShield で作成したスクリプトとの互換性の目的でのみサポートされています。複数のファイル メディア ライブラリを InstallShield で使用することは不要となったため、また使用することによってインストーラーが破損する可能性もあるため、複数のファイル メディア ライブラリを使用しないことが推奨されます。

FeatureInitialize 関数はメディア名とファイルメディア ライブラリを関連付け、そのメディア ライブラリにアクセスできるようにします。

構文

```
FeatureInitialize ( szMediaLibrary, szMediaLibraryFile );
```

パラメーター

テーブル 67 · FeatureInitialize のパラメーター

パラメーター	説明
szMediaLibrary	FeatureMoveData の例 を使って転送されるファイルを持つファイルメディアライブラリに関連付けるメディア名を指定します。 Data というメディア名は、デフォルトのファイルメディアライブラリである Data1.cab 用に確保されています。パラメーター szMediaLibrary に Data を渡すことはできません。
szMediaLibraryFile	初期化するファイルメディアライブラリのファイル名を指定します。ファイル名のフォーマットは xxx1.cab で、second1.cab、wow1.cab などになります。パスは指定しないでください。このファイルは、インストーラーのソースフォルダー (SRCDIR) に配置する必要があります。

戻り値

テーブル 68 · FeatureInitialize の戻り値

戻り値	説明
0	FeatureInitialize が成功しました。
< 0	FeatureInitialize が失敗しました。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

この関数は、スクリプト作成機能セットでは使用できません。

FeatureInitialize で初期化されたファイルメディアライブラリは、**FeatureMoveData** を呼び出してインストールしなくてはなりません。この様なメディアライブラリをインストールするのに **FeatureTransferData** を呼び出すと失敗します。

インストーラーが **FeatureInitialize** を使って初期化されたファイルメディアライブラリを インストールする場合、アンインストールは **MaintenanceStart** ではなく、**DeinstallStart** を呼び出して有効にしなくてはなりません。(これはファイルが **FeatureMoveData** の呼び出しでインストールしなくてはならない場合で、メンテナンスセットアップはサポートしません。)

szMediaLibraryFile に格納されているメンテナンス / アンインストール機能は、**FeatureMoveData** を使ってインストールすることはできません。**FeatureInitialize** を呼び出す前の **FeatureMoveData** への最初の呼び出しのみファイル メディア ライブラリのメンテナンス / アンインストール機能をインストールすることができます。

デフォルトのメディア名 Data を使用してデフォルトのファイルメディアライブラリ (Data1.cab) にアクセスする前に、**FeatureInitialize** を呼び出す必要はありません。デフォルトのメディアは、インストーラーの初期化中に自動的に初期化されます。ファイル メディア ライブラリは、インストールのソース フォルダーに配置する必要があります。このフォルダーの名前は、インストーラーの初期化中にシステム変数 SRCDIR に割り当てられます。

FeatureInitialize の例

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureInitialize 関数のデモンストレーションを行います。
*
¥*---------------------------------------------------------------------*/

NUMBER nResult, n;

program

    // ターゲットディレクトリを設定します。
    TARGETDIR = "C:¥¥temp" ^ MEDIA;

    // 転送する機能を選択します。
    nResult = FeatureDialog ("", "", TARGETDIR,"");

    if (nResult < 0) then
      SprintfBox (SEVERE, "FeatureDialog ERROR", "%ld", nResult);
    endif;

    // data1.cab ( データメディア ) と関連付けるファイルを転送します。
    nResult = FeatureMoveData (MEDIA, n, 0);
    if (nResult < 0) then
      SprintfBox (SEVERE, "FeatureMoveData ERROR", "%ld", nResult);
    endif;

    // 2 番目のメディア用にセットアップします。
    MEDIA = "second";

    // ターゲットディレクトリを設定します。
    TARGETDIR = "C:¥¥temp" ^ MEDIA;

    // 新しいメディアを second1.cab  キャブファイルと関連付けます。
    nResult = FeatureInitialize (MEDIA, "second1.cab");
    if (nResult < 0) then
      SprintfBox (SEVERE, "FeatureInitialize ERROR", "%ld", nResult);
    endif;

    // 転送する機能を選択します。
    nResult = FeatureDialog ("", "", TARGETDIR,"");
    if (nResult < 0) then
      SprintfBox (SEVERE, "FeatureDialog ERROR", "%ld", nResult);
    endif;

    // FeatureMoveData 関数を再度初期化します。
    nResult = FeatureMoveData ("", n ,0);
    if (nResult < 0) then
      SprintfBox (SEVERE, "FeatureMoveData ERROR", "%ld", nResult);
    endif;

    // second1.cab (2 番目のメディア ) と関連付けるファイルを転送します。
    nResult = FeatureMoveData (MEDIA, n, 0);
    if (nResult < 0) then
      SprintfBox (SEVERE, "FeatureMoveData ERROR", "%ld", nResult);
    endif;
```

```
endprogram

// ソースファイル : Is5fn628.rul
```

FeatureIsItemSelected

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

InstallScript プロジェクトでは **FeatureIsItemSelected** が、指定した機能の現在の選択状態を判別します。InstallScript MSI プロジェクトの場合、**FeatureIsItemSelected** は、指定された機能の現在のインストール状態を判別します。FeatureGetData を使用すると、機能が選択されているか否かを判断することも可能です。

FeatureIsItemSelected は一般的にファイル転送の前または後に機能固有のタスクを処理するために呼び出されます。.InstallScript インストールでファイル転送中に機能固有のタスクを処理する場合、機能イベント ハンドラー関数にコードを配置することが推奨されます。

構文

```
FeatureIsItemSelected ( szFeatureSource, szFeature );
```

パラメーター

テーブル 69 - FeatureIsItemSelected のパラメーター

パラメーター	説明
szFeatureSource	InstallScript プロジェクトの場合、インストール状態あ m たは選択設定が決定されているメディア ライブラリを指定します。 InstallScript MSI プロジェクトの場合、スクリプト作成の機能セットのメディア名を指定します。
szFeature	インストール状態または選択設定を判断する機能の名前を指定します。トップ階層の機能やサブ機能の参照方法は、「関数呼び出しで機能やサブ機能を指定する」を参照してください。

戻り値

テーブル 70 - FeatureIsItemSelected の戻り値

戻り値	説明
TRUE (1)	InstallScript プロジェクトでは、szFeature が選択されています。 InstallScript MSI プロジェクトの場合、szFeature のインストール状態は INSTALLSTATE_LOCAL (機能がローカル ドライブにインストールされた) です。
FALSE (0)	InstallScript プロジェクトでは、szFeature は選択されていません。 InstallScript MSI プロジェクトの場合、szFeature のインストール状態は INSTALLSTATE_ABSENT (機能がアンインストールされた) です。
< 0	関数は機能がインストールされているか、または選択されているかどうか判断できませんでした。追加情報については、FeatureError を呼び出してください。

FeatureIsItemSelected の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureIsItemSelected 関数のデモンストレーションを行います。
*
* このスクリプト例は、ユーザーがインストールできるセットアップに含まれる
* 機能のリストと、各機能に必要な容量を表示する
* ダイアログ ボックスを表示します。ユーザーが機能を
* 選択したとき、機能のインストール状態を
* 表示します。
*
* コメント : このスクリプト例を実行するには、いくつかの機能および
*      / またはファイルを含むコンポーネントを持つサブ機能を含む
プロジェクトを作成 ( またはプロジェクトに挿入 ) します。
*
```

```
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

// Windows Installer API 関数プロトタイプと定数用に iswi.h をインクルードし、
// OnBegin イベントと OnEnd イベント用のコードを宣言します。
#include "iswi.h"

// キーワードのエクスポートは MyFunction () をエントリポイント関数として識別します。
// 使用する引数は、Installer データベースへのハンドルでなくてはなりません。
  export prototype MyFunction(HWND);

// 必要な処理 : グローバル変数の宣言、定数の定義、およびユーザー定義の関数並びに DLL 関数を
//        ここでプロトタイプ化します。

function MyFunction(hMSI)

   STRING  szTitle, szMsg, svDir;
   NUMBER  nResult;

begin

   svDir   = INSTALLDIR;
   szTitle = " 機能の選択 ";
   szMsg   = " コンピューターへインストールする機能を選択してください。";

   // 利用可能な機能を表示します。
   SdFeatureTree (szTitle, szMsg, svDir, "", 2);

   // Subfeature1 のインストール後の状態を決定します。
   nResult = FeatureIsItemSelected (MEDIA,  "Feature1¥¥Subfeature1");

   // Subfeature1 のインストール後の状態を示すメッセージを表示します。
   if nResult = 1  then
     MessageBox ("Subfeature1 はローカルにインストールされます。", INFORMATION);
   else
     MessageBox ("Subfeature1 がアンインストールされます。", INFORMATION);
   endif;

end;
```

FeatureListItems

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

FeatureListItems 関数は ｓ ｚ Feature で指定された機能の下にあるすべてのサブ機能をリストします。サブ機能の完全修飾名リストは listFeatures に格納されます。szFeature にサブ機能がある場合、listFeatures は空のリストを返します。

構文

FeatureListItems (szFeatureSource, szFeature, listFeatures);

パラメーター

テーブル 71 ・ FeatureListItems のパラメーター

パラメーター	説明
szFeatureSource	サブ機能をリストするスクリプト作成の機能セットのメディア名を指定します。
szFeature	サブ機能をリストする機能を指定します。このパラメーターにヌル文字列 (″″) を渡して、トップレベルの機能をすべてリストします。トップ階層の機能やサブ機能の参照方法は、「関数呼び出しで機能やサブ機能を指定する」を参照してください。
listFeatures	機能のリストを返します。FeatureListItems を呼び出す前に ListCreate を呼び出して listFeatures が認識する文字列リストを初期化しなくてはなりません。

戻り値

テーブル 72 ・ FeatureListItems の戻り値

戻り値	説明
0	FeatureListItems が機能をリストしました。
< 0	FeatureListItems 機能をリストすることができませんでした。追加情報については、FeatureError を呼び出してください。

FeatureListItems の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureListItems 関数のデモンストレーションを行います。
*
* このスクリプト例は、指定した機能の下にあるすべてのサブ機能を
* リストにする関数のデモンストレーションを行います。
*
* コメント : このスクリプト例を実行するには、いくつかの機能および
*       / またはファイルを含むコンポーネントを持つサブ機能を含む
プロジェクトを作成 ( またはプロジェクトに挿入 ) します。
*
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include ″Ifx.h″

// Windows Installer API 関数プロトタイプと定数用に iswi.h をインクルードし、
```

```
// OnBegin イベントと OnEnd イベント用のコードを宣言します。
#include "iswi.h"

// キーワードのエクスポートは MyFunction () をエントリポイント関数として識別します。
// 使用する引数は、Installer データベースへのハンドルでなくてはなりません。
  export prototype MyFunction(HWND);

// 必要な処理 : グローバル変数の宣言、定数の定義、およびユーザー定義の関数並びに DLL 関数を
//        ここでプロトタイプ化します。

function MyFunction(hMSI)

    STRING  szTitle, szMsg;
    NUMBERlistID;

begin

    szTitle = "MEDIA 機能をリストにする ";
    szMsg   = "MEDIA は次のトップレベル機能を含みます :";

    // 文字列リストを初期化します。
    listID = ListCreate (STRINGLIST);

    // 指定したメディアにトップレベル機能のリストを作成します。
    FeatureListItems (MEDIA, "", listID);

    // トップレベル機能のリストを表示します。
    SdShowInfoList (szTitle, szMsg, listID);

end;
```

FeatureLoadTarget

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FeatureLoadTarget 関数は、有効なログファイルが存在する任意のインストールの初期化中に自動的に呼び出されます。

構文

```
FeatureLoadTarget( szReserved );
```

パラメーター

テーブル 73 · FeatureLoadTarget のパラメーター

パラメーター	説明
szReserved	このパラメーターにヌル文字列 ("") を渡します。他の値は使用できません。

戻り値

テーブル 74 · FeatureLoadTarget の戻り値

戻り値	説明
0	この関数は常にゼロ (0) を返します。

追加情報

FeatureLoadTarget はログファイルを読み取ってセットアップのすべてのファイルグループのターゲットディレクトリを読み出します。FeatureLoadTarget はセットアップで使用されたすべてのテキスト置換の値を読み出します。これにはすべてのテキスト置換値の他にも、すべてのファイルグループやターゲットディレクトリ関連システム変数が含まれます。

FeatureMoveData

***プロジェクト ·** この情報は、InstallScript プロジェクトに適用されます。*

FeatureMoveData 関数は、szFeatureSource が参照するファイル メディアで選択した機能に関連付けたファイルを転送し、圧縮解除します。必要に応じて、関数はエンドユーザーに対して自動的に次のディスクを要求します。

この関数を使って、ファイルを WINSYSDIR64 に転送する場合、まず WOW64FSREDIRECTION を使用してファイル システムのリダイレクトを無効にする必要があります。無効化をしない場合、WINSYSDIR64 に転送されるファイルは不適切に 32 ビット SysWOW64 システムフォルダーにリダイレクトされます。インストールが利用する可能性のある Windows 機能にはファイル システム リダイレクトを有効にしておく必要があるため、Windows ドキュメンテーションではリダイレクトを無効にするのはそれが必要な場合のみにとどめることが推奨されています。必要なファイルを WINSYSDIR64 へ転送し終わったら、直ちにシステム ファイルのリダイレクトを有効にすることをお勧めします。詳しくは、「64 ビット オペレーティング システムを InstallScript インストールでターゲットにする」を参照してください。

FeatureMoveData を呼び出すと、ファイルがインストールされた後、呼び出しが戻る前に Do (SELFREGISTRATIONPROCESS) が自動的に呼び出されます。従って、インストールでファイル転送の後、自己登録の前に追加アクションの処理が必要な場合、それらのアクションを OnMoved イベントに配置します。OnMoved イベントはファイル転送のあと、バッチ自己登録が発生する前に呼び出されます。

FeatureMoveData は同じメディアに対して繰り返し呼び出すことができますが、2 番目以降の呼び出しを行う場合は、あらかじめ第 1 パラメーターの位置にヌル文字列 ("") を使って FeatureMoveData を呼び出し、内部構造体をリセットしておく必要があります。デフォルト メディアおよび内部構造体は、FeatureMoveData の最初の呼び出しを行う前に、InstallShield によって自動的に初期化されます。

メモ・この関数は、スクリプト作成機能セットでは使用できません。

構文

FeatureMoveData (szFeatureSource, nvReserved, nReserved);

パラメーター

テーブル 75・FeatureMoveData のパラメーター

パラメーター	説明
szFeatureSource	転送されるファイルが含まれるファイルメディアの名前を指定します。
nvReserved	この引数で数値変数を渡します。有用な情報は戻されません。
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。

戻り値

テーブル 76・FeatureMoveData の戻り値

戻り値	説明
0	FeatureMoveData が成功しました。
< 0	FeatureMoveData が失敗しました。追加情報については、FeatureError を呼び出してください。

FeatureMoveData の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdSetupTypeEx 関数、SdFeatureDialog 関数、FeatureIsItemSelected 関数、
* FeatureError 関数、そして PlaceWindow 関数のデモンストレーションを行います。
*
* メモ : このスクリプト例を実行するには、
*        次の機能 (f)、サブ機能 (sf)、および コンポーネント (c)
*        を利用します :
*
*       (f) Program_Files
*              (c) Program_DLLS
*              (c) Program_EXEs
*           (f) Example_Files
*              (sf) Small_Documents
*                  (c) Small_Document_Examples
*              (sf) Books
*                  (c) Book_Examples
*              (sf) Graphics
```

```
*                (c) Graphic_Examples
*          (f) Help_Files
*             (c) Help_Files
*          (f) Utilities
*             (sf) Grammar_Checker
*                (c) Grammar_Checker
*             (sf) Art Studio
*                (c) Art_Studio
*          (f) Evaluation_Copy
*             (c) Evaluation_Copy
*             (c) Help_Files
*
*″ダミー″ファイルをコンポーネントに挿入します。ここで、
*Program EXE のコンポーネントへ挿入するメイン EXE ( 下の MAIN_EXE)
*に正しいファイル名を定義するようにしてください。
*
*このスクリプト例はファイルをインストールし、アイコンを
*[ スタート プログラム ] メニューに追加して、アンインストール機能を
*提供します。
*
¥*---------------------------------------------------------------*/

// 文字列を定義します。実際のインストールでは [ 文字列エディター ] ビューでこれらを定義し、
// スクリプトの各文字列 ID の前に @ 記号を挿入します。
#define FEAT_SELECT_TITLE            ″機能の選択″
#define FEAT_SELECT_MSG              ″インストールする機能とサブ機能を選択します。″
#define FEAT_PROGRAMFILES_DISPLAYNAME  ″プログラムファイル″
#define PASSWORD_PROMPT              ″パスワードを入力してください。″
#define PASSWORD_ERRMSG ″パスワードが間違っています。再度入力してください。″
#define TITLE_MAIN                 ″ワードプロセッサ″
#define TITLE_CAPTIONBAR             ″ワードプロセッサのセットアップ″
#define APPBASE_PATH               ″会社名 ¥¥ ワードプロセッサ″
#define COMPANY_NAME ″会社名″
#define PRODUCT_NAME                 ″ワードプロセッサ″
#define  PRODUCT_VERSION     ″1.0″
#define PRODUCT_KEY                ″ワードプロセッサ″
#define DEINSTALL_KEY              ″ワードプロセッサ″
#define UNINSTALL_NAME             ″ワードプロセッサ″
#define ADDINGICON                 ″プログラムアイコンを [ スタート ] の [ プログラム ] メニューに追加しています …″
#define PROGRAMDIR                 ″プログラム″
#define DEFAULT_FOLDER_NAME          ″″
#define APP_NAME                 ″ワードプロセッサ″
#define COMPLETE_MSG               ″セットアップが完了しました。[ スタート ] プログラムメニューからワードプロセッサ
を実行できます。″
#define MAIN_EXE                 ″WRITE.EXE″
#define SETUPTYPE_TITLE            ″セットアップの種類の選択″
#define SETUPTYPE_MSG              ″セットアップの種類を選択してください。″
#define SETUPTYPE_CUSTOM             ″カスタム″

   // グローバル変数の宣言。

   // 関数宣言
   prototype SetUpFileTransfer ();
   prototype HandleFeatureError (NUMBER);
   prototype FinishSetup ();

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include ″Ifx.h″
```

```
    export prototype ExFn_FeatureMoveData(HWND);

function ExFn_FeatureMoveData(hMSI)
    STRING  svData, svLogFile,  szProgram, szFeature;
    STRING svResult, svSetupType, svDir;
    BOOL    bInitStepsDone, bPwdValid;
    NUMBER  nvData, nvDisk, nResult;
begin

    SetUpFileTransfer ();

    // インストールダイアログで [ 戻る ] ボタンを無効にします。
    Disable(BACKBUTTON);

    // セットアップの種類を取得します。
    svDir = INSTALLDIR;

    SdSetupTypeEx (SETUPTYPE_TITLE, SETUPTYPE_MSG, "", svSetupType, 0);

    // ユーザーが [ カスタム ] セットアップの種類を選択したら、機能選択ダイアログを表示します。
    if (svSetupType = SETUPTYPE_CUSTOM) then
        SdFeatureDialog (FEAT_SELECT_TITLE, FEAT_SELECT_MSG, svDir, "");
    endif;

    // インストールダイアログで [ 戻る ] ボタンを有効にします。
    Enable(BACKBUTTON);

    // 進行状況インジケーター、情報ゲージ、およびビルボードの場所を含む
    // 進行状況インジケーターをセットアップします。
    PlaceWindow (FEEDBACK, LOWER_LEFT, LOWER_LEFT, LOWER_LEFT);

    PlaceWindow (STATUSDLG, CENTERED, LOWER_RIGHT, LOWER_RIGHT);

    PlaceWindow (BILLBOARD, CENTERED, CENTERED, CENTERED);

    Enable (STATUSDLG);

    Enable (INDVFILESTATUS);

    // 次のファイル転送操作の完了を示す、進行状況バーの最後の
    // パーセントを表示します。
    StatusUpdate (ON, 95);

    // ファイルをターゲットシステムに転送します。FeatureMoveData prompts
    // フロッピーディスク インストール用の次のディスク
    nResult=FeatureMoveData (MEDIA, nvDisk, 0);

    // FeatureError 関数の動作を確認します。
    HandleFeatureError (nResult);

    FinishSetup();

end;

/*----------------------------------------------------------------------------*¥
 *
 * Function:  SetupFileTransfer()
 *
 *  目的 : この関数はファイル転送をセットアップします。ここで
```

```
*          この関数へのプロセスを抽出するのは、
*          このスクリプト例が示す関数呼び出しを分かり易くするためです。
*
/*---------------------------------------------------------------------------*¥
function SetUpFileTransfer()

begin
  // インストール画面のセットアップ。
  Enable (FULLWINDOWMODE);
  SdProductName ( PRODUCT_NAME );
  SetTitle (TITLE_MAIN, 24, WHITE);
  SetTitle (TITLE_CAPTIONBAR, 0, BACKGROUNDCAPTION);
  Enable (BACKGROUND);

  // ユーザーを歓迎し、システムが最小要件を満たしているか確認して
  // インストール先を確認します。
  bInitStepsDone = FALSE;

  while (!bInitStepsDone)
    Disable(BACKBUTTON);
    Welcome ("", 0);
    Enable(BACKBUTTON);
    INSTALLDIR = PROGRAMFILES ^ APPBASE_PATH;
    if (AskDestPath ("", "", INSTALLDIR, 0) != BACK) then
      bInitStepsDone = TRUE;
    endif;
  endwhile;

  // レジストリエントリと次の DeinstallStart への呼び出しに必要な
  // インストール情報を設定します。
  InstallationInfo (COMPANY_NAME, PRODUCT_NAME,
                    PRODUCT_VERSION, PRODUCT_KEY);

  // レジストリエントリを含むアンインストール ログ ファイルを初期化します。
  svLogFile = "Uninst.isu";

  DeinstallStart (INSTALLDIR, svLogFile, DEINSTALL_KEY, 0);

  RegDBSetItem (REGDB_UNINSTALL_NAME, UNINSTALL_NAME);
end;

/*---------------------------------------------------------------------------*¥
*
* 関数 :  HandleFeatureError
*
* 目的 :  この関数は、機能関数が戻した値を評価し、
*          関数によって返された値を評価し、その値が 0 以下だとエラー
*          番号を返してインストールを終了します。
*
/*---------------------------------------------------------------------------*¥
function HandleFeatureError (nResult)
  NUMBER  nvError;
  STRING  svFeatureSource, svFeature, svComponent, svFile;
begin
  if (nResult < 0) then
    FeatureError (svFeatureSource, svFeature, svComponent, svFile, nvError);
    SprintfBox (INFORMATION, " データ転送エラー情報 ",
                "FeatureError は " +
                " 次のデータ転送エラーを戻しました : ¥n" +
```

```
            " セットアップは終了します。¥n¥n" +
            " メディア名 : %s¥n 機能 : %s¥n コンポーネント : %s¥n" +
            " ファイル : %s¥n エラー番号 : %ld",
            svFeatureSource, svFeature, svComponent, svFile, nvError);
   abort;
  endif;
end;

/*---------------------------------------------------------------------------*¥
 *
 * 関数 :  FinishSetup()
 *
 * 目的 : この関数はインストールを終了します。ここで
 *          この関数へのプロセスを抽出するのは、
 *          このスクリプト例が示す関数呼び出しを分かり易くするためです。
 *
/*---------------------------------------------------------------------------*¥
function FinishSetup()
begin
  // 次のファイル転送操作の完了を示す、進行状況バーの最後の
  // パーセントを表示します。
  StatusUpdate (ON, 99);

  // スタートプログラムメニューアイコンの進行状況バーを 99% に増分します。
  SetStatusWindow (96 , ADDINGICON);

  // APP_NAME アイコンを DEFAULT_FOLDER_NAME フォルダーに追加します。
  szProgram = INSTALLDIR ^ PROGRAMDIR ^ MAIN_EXE;

  LongPathToQuote (szProgram, TRUE);
  AddFolderIcon (DEFAULT_FOLDER_NAME, APP_NAME, szProgram,
              INSTALLDIR ^ PROGRAMDIR, "", 0, "", REPLACE);

  Delay (1);

  // 進行状況インジケーターとその設定を無効にします。
  Disable  (INDVFILESTATUS);
  Disable (STATUSDLG);

  // インストール完了を通知し、Readme ファイルの表示を提案します。
  MessageBox (COMPLETE_MSG, INFORMATION);

end;
```

FeaturePatch

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FeaturePatch 関数は、差分メディアを利用するインストールでのみ呼び出されます。(MEDIA_FIELD_MEDIA_FLAGS を 2 番目のパラメーターとして MediaGetData を呼び出してメディアフォーマットを確認することができます。)

FeaturePatch は、FeatureTransferData や FeatureMoveData への次の呼び出しを引き起こし、**FeatureTransferData** が呼び出された時に、既にインストールされているすべてメンテナンス / アンインストール機能の ファイルを含む (Data1.hdr、Data1.cab、および Layout.bin は除く) すべての機能を再インストールします。(メンテナンス / ア

ンインストール機能はリリースビルダーによってディスクイメージに自動的に配置され、InstallShield では表示されません。) 実行中のファイルメディア ライブラリのコピーは、後に続くメンテナンス操作、つまり更新、修復またはアンインストールに利用できるようターゲットマシンに格納されます。

構文

FeaturePatch ( );

パラメーター

なし。

戻り値

テーブル 77 - FeaturePatch の戻り値

戻り値	説明
0	関数が成功したことを示します。
< 0	関数が失敗したことを示します。

追加情報

FeaturePatch は、OnUpdateUIBefore イベント ハンドラー関数のデフォルト コードで呼び出されます。

FeatureReinstall

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

- *InstallScript*
- *InstallScript MSI*

既に実行済みのセットアップで **FeatureReinstall** 関数を呼び出した後に FeatureTransferData の次の呼び出しを行うと、最後にセットアップを実行したときに指定したファイル転送を行います。

構文

FeatureReinstall ( );

パラメーター

なし

戻り値

テーブル 78 ・ FeatureReinstall の戻り値

戻り値	説明
0	セットアップの再インストレーションの準備が正常に行われたことを示します。
< 0	セットアップの再インストレーションの準備が、正常に行われなかったことを示します。

FeatureRemoveAll

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

FeatureRemoveAll はメンテナンスインストール中に前回インストール済みのすべての機能を強制的に削除するのに利用します。FeatureRemoveAll は通常、SdWelcomeMaint ダイアログでユーザーが [削除] を選択したときに呼び出されます。

FeatureRemoveAll を呼び出すとすべての機能の選択が解除されます。その結果、FeatureTransferData が呼び出されたとき、既にインストールされている機能 (セットアップログファイルがある場合、それを読んで判断される) が削除 (アンインストール) されます。

セットアップは、セットアップログファイルを読んで機能が既にインストールされているか否かを判断します。セットアップ初期設定中に有効なログファイルが検出されなかった場合、すべての機能が *インストールされていない*とみなします。この場合、**FeatureRemoveAll** の呼び出しはすべての機能選択をクリアしますが、**FeatureTransferData** が呼び出されたときには機能のアンインストールを強制しません。

***メモ・FeatureRemoveAll** はまた、DISK1TARGET の場所へ自動的にインストールされた Disk 1 セットアップファイルを含む機能を初めとする内部機能をすべて選択解除します。*

構文

```
FeatureRemoveAll ( );
```

パラメーター

なし

戻り値

テーブル 79 - FeatureRemoveAll の戻り値

戻り値	説明
0	関数がすべての機能選択を選択解除したことを示します。
< 0	関数がすべての機能選択を選択解除することができなかったこを示します。

FeatureRemoveAllInLogOnly

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

FeatureRemoveAllInLogOnly 関数はアップデートインストール中に呼び出され、セットアップログファイルに記録された通り、現在のメディアに無いが以前にインストールされているすべての機能を強制的に削除します。

構文

```
FeatureRemoveAllInLogOnly ( );
```

パラメーター

なし。

戻り値

テーブル 80 - FeatureRemoveAllInLogOnly の戻り値

戻り値	説明
0	関数が機能選択を選択解除したことを示します。
< 0	関数が機能選択を選択解除することができなかったこを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

FeatureRemoveAllInLogOnly を呼び出すと、現在のメディアに無いが以前にインストールされた機能すべてが選択解除されます。その結果、FeatureTransferData が呼び出されたときこれらの機能は削除 (アンインストール) されます

***注意 -** ″追加″ インストーラーで FeatureRemoveAllInLogOnly を呼び出さないで下さい。追加インストーラーとは、以前のインストーラーの機能の一部分を含むシステム上に既に存在するアプリケーションと同じ INSTANCE_GUID*

を含むインストーラーです。これは現在のインストーラーの機能以外、前回のインストーラーに含まれるすべての機能をアンインストール（削除）します。

インストーラーの初期化中に有効なログファイルが検出されなかった場合、FeatureRemoveAllInLogOnly を呼び出しても効果がありません。

FeatureRemoveAllInMedia

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FeatureRemoveAllInMedia は現在のメディアにある、以前にインストールされてたすべての機能を強制的に削除するために、メンテナンスインストール中に利用されます。この関数は通常、SdWelcomeMaint ダイアログでユーザーが [削除] を選択したときに呼び出されます。

FeatureRemoveAllInMedia を呼び出したとき、メディアヘッダーにリストされている機能のみが削除されます。但しアップデートされたアプリケーションで、機能が削除されているがアンインストールされていない場合（つまり、アップデートメディアに機能は存在しないがオリジナルメディアにそれらが存在した場合で、アップデートが FeatureRemoveAllInLogOnly を呼び出さなかった場合）、これらの機能がアンインストール中に削除されるかどうかはアプリケーションが差分メディアを使ってアップデートされたか、またはフルアップデートメディアを使ってアップデートされたかによって異なります。

- フルメディアを使ってアプリケーションがアップデートされた場合、フルメディアヘッダーはオリジナル メディアヘッダーと置換されます。従って、これらの機能はアンインストールの最中に削除されません。
- 差分メディアを使ってアプリケーションがアップデートされた場合、両方のヘッダーファイルが存在します。したがって、FeatureRemoveAllInMedia がこれらの機能を削除します。

インストールされた機能すべてが確実にアンインストールされるよう、インストールが FeatureRemoveAllInMediaAndLog を呼び出す必要があります。これによってすべての機能が削除されます。

構文

```
FeatureRemoveAllInMedia();
```

パラメーター

なし。

戻り値

テーブル 81 - FeatureRemoveAllInMedia の戻り値

戻り値	説明
0	関数がすべての機能選択を選択解除したことを示します。
< 0	関数がすべての機能選択を選択解除することができなかったこを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

FeatureRemoveAllInMedia を呼び出すと、現在のメディアにある機能すべてが選択解除されます。その結果、FeatureTransferData が呼び出されたとき、既にインストールされている機能 (インストールログファイルがある場合、それを読んで判断される) が削除 (アンインストール) されます。

インストールは、インストール ログファイルを読み取って機能が既にインストールされているかどうかを判断します。インストール初期化中 (MAINTENANCE は FALSE) に有効なログファイルが検出されなかった場合、すべての機能が *インストールされていない* と見なされます。この場合、FeatureRemoveAllInMedia を呼び出すとすべての機能が選択解除され、インストールは行われません。しかし、後に続く **FeatureTransferData** の呼び出しは (ログファイルにアンインストールする項目が含まれているため) 何もアンインストールしません。この場合の **FeatureRemoveAllInMedia** 呼び出しはお勧めできません。

メモ - *FeatureRemoveAll はメンテナンス / アンインストール機能を含むすべての内部機能も選択解除します。*

FeatureRemoveAllInMediaAndLog

プロジェクト - *この情報は、InstallScript プロジェクトに適用します。*

FeatureRemoveAllInMediaAndLog 関数はアップデートのインストール中に呼び出され、以前にインストールされた機能すべてを強制的に削除します。これには現在のメディアに含まれる機能と、現在のメディアには含まれていないがセットアップログファイルに記録されている機能を含みます。

構文

```
FeatureRemoveAllInMediaAndLog();
```

パラメーター

なし。

戻り値

テーブル 82 - FeatureRemoveAllInMediaAndLog の戻り値

戻り値	説明
0	関数がすべての機能選択を選択解除したことを示します。
< 0	関数がすべての機能選択を選択解除することができなかったこを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

FeatureRemoveAllInMediaAndLog を呼び出すことは、FeatureRemoveAllInMedia と FeatureRemoveAllInLogOnly を呼び出すのと同じ作用があります。

FeatureRemoveAllInMediaAndLog を呼び出すと以前にインストールされているすべての機能の選択が解除されます。その結果、FeatureTransferData が呼び出されたときこれらの機能は削除 (アンインストール) されます

インストールは、セットアップ ログファイルを読み取って機能が既にインストールされているかどうかを判断します。インストール初期化中 (MAINTENANCE は FALSE) に有効なログファイルが検出されなかった場合、すべての機能が*インストールされていない*と見なされます。この場合、**FeatureRemoveAllInMediaAndLog** を呼び出すとすべての機能が選択解除され、インストールは行われません。しかし、後に続く **FeatureTransferData** の呼び出しは (ログファイルにアンインストールする項目が含まれているため) 何もアンインストールしません。この場合の **FeatureRemoveAllInMediaAndLog** 呼び出しはお勧めできません。

メモ - *FeatureRemoveAllInMediaAndLog はメンテナンス / アンインストール機能を含むすべての内部機能も選択解除します。*

FeatureSaveTarget

プロジェクト - *この情報は、次のプロジェクトの種類に適用します :*

- *InstallScript*
- *InstallScript MSI*

FeatureSaveTarget はセットアッププロジェクトによって使用されたすべてのテキスト置換の現在値を取得して、インストールログファイルにこれを保管します。これにはすべてのコンポーネントの現在のターゲットディレクトリやすべてのターゲットディレクトリ関連システム変数、およびその他のすべてのテキスト置換値などが含まれます。

セットアップが初期化されて有効なログファイルが見つかると、この関数で以前に保管されていた値は自動的に変更されます。メンテナンスセットアップはこれによって以前にインストールされていたコンポーネントをアップデートし、適切な場所に新しいコンポーネントをインストールします。

構文

FeatureSaveTarget (szReserved);

パラメーター

テーブル 83 ・ FeatureSaveTarget のパラメーター

パラメーター	説明
szReserved	このパラメーターにヌル文字列 ("") を渡します。他の値は使用できません。

戻り値

この関数は常にゼロ (0) を返します。

FeatureSelectItem

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

- *InstallScript*
- *InstallScript MSI*

FeatureSelectItem 関数は、機能の選択ステータスを [選択] または [非選択] のいずれかに設定します。**FeatureSelectItem** を使って、機能ダイアログに機能が表示される前に選択ステータスを変更したり、インストールの必要に応じて、選択内容を後から変更または上書きしたりできます。

***メモ・FeatureSelectItem** を呼び出した後にセットアップの種類選択関数を呼び出す場合 (たとえば **FeatureSetupTypeSet**、**SetupType2**、**SdSetupType**、または **SdSetupTypeEx**)、**FeatureSelectItem** によって設定された機能の選択は、セットアップの種類での選択で上書きされます。(これは、インストール プロジェクトに追加する内部の機能に影響しない機能に適用します。)**SetupType2** は、OnBegin、OnCCPSearch および OnAppSearch イベント ハンドラー関数が呼び出される OnFirstUIBefore イベント ハンドラー関数のデフォルトコードに呼び出されます。OnFirstUIBefore イベント ハンドラーを上書きまたはカスタマイズできます。*

***FeatureSelectItem** を使って機能の選択を解除する場合、機能ダイアログを使って機能の選択解除する場合と同じ規則が適用されます。つまり、機能が現在選択されている機能で必要な場合、その選択を解除することはできません。したがって、別の機能が必要とする機能の選択を解除するには、その機能を必要とする機能の選択を解除してから、選択解除を行う必要があります。*

- *以前の機能選択が自動的に既存のログ ファイルからロードされるため、通常はメンテナンス モード中に **FeatureSelectItem** を呼び出す必要はありません。*

構文

FeatureSelectItem (szFeatureSource, szFeature, bSelect);

パラメーター

テーブル 84 - FeatureSelectItem のパラメーター

パラメーター	説明
szFeatureSource	スクリプト作成機能セットまたは (InstallScript プロジェクトの場合は) 選択状態を設定する機能を含むファイルメディアライブラリのメディア名を指定します。
szFeature	選択ステータスを設定する機能を指定します。 このパラメーターでは、ヌル文字列 ("") を指定できます。ヌル文字列を指定すると、szFeatureSource からの機能は bSelect の値に基づいて選択 / 非選択されます。 InstallScript で機能およびサブ機能を指定する方法についての詳細は、「関数呼び出しで機能やサブ機能を指定する」を参照してください。
bSelect	機能が選択されるべきかどうかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ TRUE ― 特定の機能を選択します。 ・ FALSE ― 特定の機能を選択しません。

戻り値

テーブル 85 - FeatureSelectItem の戻り値

戻り値	説明
0	**FeatureSelectItem** が機能の選択ステータスを正しく設定しました。
< 0	**FeatureSelectItem** が機能の選択ステータスを設定できませんでした。追加の情報については、「FeatureError」を参照してください。

FeatureSelectItem の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureSelectItem 関数のデモンストレーションを行います。
*
* このスクリプト例は、機能の状態を選択または非選択に設定する
* 関数のデモンストレーションを行います。
*
*  コメント : このスクリプト例を実行するには、いくつかの機能および
*       / またはファイルを含むコンポーネントを持つサブ機能を含む
プロジェクトを作成 ( またはプロジェクトに挿入 ) します。
*
¥*---------------------------------------------------------------*/
```

```
// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

// Windows Installer API 関数プロトタイプと定数用に iswi.h をインクルードし、
// OnBegin イベントと OnEnd イベント用のコードを宣言します。
#include "iswi.h"

// キーワードのエクスポートは MyFunction () をエントリポイント関数として識別します。
// 使用する引数は、Installer データベースへのハンドルでなくてはなりません。
  export prototype MyFunction(HWND);

// 必要な処理 : グローバル変数の宣言、定数の定義、およびユーザー定義の関数並びに DLL 関数を
//        ここでプロトタイプ化します。

function MyFunction(hMSI)

   STRING  svDir, szTitle, szMsg;

begin

   svDir   = INSTALLDIR;
   szTitle = " 機能の選択 ";
   szMsg   = " コンピューターへインストールする機能を選択してください。";

   // Subfeature2 の選択状態を非選択に設定します。
   FeatureSelectItem (MEDIA, "Feature1¥¥Subfeature2", FALSE);

   // 利用可能な機能を表示します。
   SdFeatureTree (szTitle, szMsg, svDir, "", 2);

end;
```

FeatureSelectNew

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FeatureSelectNew 関数は新しい機能の選択状態を選択または選択解除の何れかに設定します。

構文

```
FeatureSelectNew (szFeatureSource, bSelect);
```

パラメーター

テーブル 86 ・ FeatureSelectNew のパラメーター

パラメーター	説明
szFeatureSource	選択ステータスを設定する、新しい機能を含むファイルメディアライブラリのメディア名を指定します。一般的に、この引数はシステム変数 MEDIA へ同等に設定されます。
bSelected	機能を選択するか選択解除するかどうかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ TRUE ― 新しい機能を選択します。 ・ FALSE ― 新しい機能を選択解除します。

戻り値

テーブル 87 ・ FeatureSelectNew の戻り値

パラメーター	説明
0	FeatureSelectNew が機能の選択状態を設定しました。
< 0	FeatureSelectNew が機能の選択状態を設定できませんでした。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

新しい機能はインストーラーのファイルメディアライブラリに存在するが、既存のログファイルには存在しない機能です。FeatureSelectNew は一般的にアップデートインストールで呼び出され、機能ダイアログに機能を表示する前に新しい機能を選択します。FeatureSelectNew は、OnUpdateUIBefore イベントハンドラー関数のデフォルトコードで呼び出されます。

***メモ ・** FeatureSelectNew を呼び出した後でセットアップの種類選択関数を呼び出すと、FeatureSelectNew で設定された機能選択はセットアップの種類の選択で上書きされます。*

以前の機能選択が自動的に既存のログファイルからロードされるため、通常はメンテナンス モード中に FeatureSelectNew を呼び出す必要はありません。

FeatureSetData

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

- *InstallScript*

- *InstallScript MSI*

FeatureSetData 関数は、指定した機能のプロパティやデータを設定します。設定のほとんどは [機能] ビューのプロパティに対応します。

構文

FeatureSetData (szFeatureSource, szFeature, nInfo, nData, szData);

パラメーター

テーブル 88 · FeatureSetData のパラメーター

パラメーター	説明
szFeatureSource	スクリプト作成機能セットまたは (InstallScript プロジェクトの場合は) プロパティ及びデータを設定する機能を含むファイルメディアライブラリのメディア名を指定します。
szFeature	機能の名前を指定します。InstallScript での機能やサブ機能の指定についての詳細は、「 関数呼び出しで機能やサブ機能を指定する」を参照してください。
nInfo	設定する情報の種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · **FEATURE_FIELD_DESCRIPTION**— このテキストは、選択ダイアログの [説明] フィールドに表示されます。 · **FEATURE_FIELD_FTPLOCATION** — FTP ロケーション。 · **FEATURE_FIELD_HTTPLOCATION** — HTTP ロケーション。 · **FEATURE_FIELD_STATUS (スクリプト作成機能セット以外)** — このテキストは、ファイル転送中に進行状況インジケーターに表示されます。 · **FEATURE_FIELD_VISIBLE** — 機能が機能選択ダイアログに表示されるか否かを示します。パラメーターの nData は、以下のいずれかになります。 TRUE: 機能は表示されます。 FALSE: 機能は表示されません。 · **FEATURE_FIELD_SELECTED** — 機能の選択状態を設定します。この設定は、FeatureSelectItem と同様の影響を与えます。パラメーターの nData は、以下のいずれかになります。 TRUE: 機能を選択します。 FALSE: 機能を選択しません。 · **FEATURE_FIELD_SIZE (ファイルメディア以外)** — 機能に対する元のファイルのサイズの合計。 · **FEATURE_FIELD_MISC** — その他のテキスト。 · **FEATURE_FIELD_DISPLAYNAME (オブジェクトプロジェクト以外)** — szFeature で指定した昨日の機能選択ダイアログに表示する名前を示します。 · **FEATURE_FIELD_IMAGE** — 機能のデフォルトのアイコンの割り当てを上書きします。表示するアイコンのインデックスを nData で渡します。機能についてアイコンを表示しないように指定するには、nData で -1 を渡します。
nData	nInfo によって示された情報が数値の場合に、設定する数値を指定します。
szData	nInfo によって示された情報が文字列の場合に、設定する文字列値を指定します。

戻り値

テーブル 89 ・ FeatureSetData の戻り値

戻り値	説明
0	FeatureSetData が成功しました。
< 0	FeatureSetData が失敗しました。追加情報については、FeatureError を呼び出してください。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

FeatureSetData の例

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureSetData 関数のデモンストレーションを行います。
*
* このスクリプト例は、指定した機能のデータとプロパティを
* 設定する関数のデモンストレーションを行います。
*
* コメント : このスクリプト例を実行するには、いくつかの機能および
*      / またはファイルを含むコンポーネントを持つサブ機能を含む
*            プロジェクトを作成 ( またはプロジェクトに挿入 ) します。
*
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

// Windows Installer API 関数プロトタイプと定数用に iswi.h をインクルードし、
// OnBegin イベントと OnEnd イベント用のコードを宣言します。
#include "iswi.h"

// キーワードのエクスポートは MyFunction () をエントリポイント関数として識別します。
// 使用する引数は、Installer データベースへのハンドルでなくてはなりません。
  export prototype MyFunction(HWND);

// 必要な処理 : グローバル変数の宣言、定数の定義、およびユーザー定義の関数並びに DLL 関数を
//       ここでプロトタイプ化します。

function MyFunction(hMSI)

   STRING  svDir, szTitle, szMsg, szData;
   NUMBER  nData;

begin

   svDir   = INSTALLDIR;
   szTitle = " 機能の選択 ";
   szMsg   = " コンピューターへインストールする機能を選択してください。";
```

```
    szData  = " 必要な機能 ";

    // エンドユーザーから Feature1 を隠します。
    FeatureSetData (MEDIA, "Feature1", FEATURE_FIELD_VISIBLE, FALSE, szData);

    // Feature2 の表示名を設定します。
    FeatureSetData (MEDIA, "Feature2", FEATURE_FIELD_DISPLAYNAME, nData, szData);

    // 利用可能な機能を表示します。
    SdFeatureTree (szTitle, szMsg, svDir, "", 2);

end;
```

FeatureSetTarget

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

FeatureSetTarget 関数は szLocation の値を szPropertyVar が指定するパブリック プロパティ (InstallScript MSI プロジェクトの場合)、またはプロパティ変数 (InstallScript プロジェクトの場合) に割り当てます。プロパティ、またはプロパティ変数は [コンポーネント] ビューの [インストール先フィールド]、または [ショートカット] ビュー内のショートカットの [ターゲット] フィールドで利用するこができます。FeatureMoveData を呼び出す前に、**FeatureSetTarget** を呼び出してください。

メモ・この関数は、スクリプト作成機能セットでは使用できません。

この関数は、オブジェクトのプロパティ変数の値を設定できません。オブジェクトのプロパティ変数の値を設定するには、オブジェクトの ScriptDefinedVar プロパティを使用する必要があります。詳細については、オブジェクトのヘルプページを参照してください (オブジェクトのヘルプページは、[オブジェクト] ビューでオブジェクトを選択するか、オブジェクトがプロジェクトに含まれている場合は [機能] ビューまたは [セットアップデザイン] ビューから選択することにより表示することができます)。ScriptDefinedVar プロパティを作成したオブジェクトに追加する方法は、「ScriptDefinedVar プロパティをオブジェクトに追加する 」 を参照してください。

InstallScript MSI プロジェクトでは、この関数は **Directory** テーブルにあるパブリック プロパティにパスを設定するために利用されます。コンポーネントのインストール先、またはショートカットのターゲットをクリックして新規ディレクトリを作成することができます。新しく作成したディレクトリは、その Directory_Parent が TARGETDIR となります。スクリプトを通して、**FeatureSetTarget** を使ってパブリック プロパティ (ディレクトリ) を設定することができます。

構文

```
FeatureSetTarget ( szFeatureSource, szPropertyVar, szLocation );
```

パラメーター

テーブル 90 ・ FeatureSetTarget のパラメーター

パラメーター	説明
szFeatureSource	ユーザー定義の変数が設定されるファイルメディアライブラリのメディア名を指定します。
szPropertyVar	ユーザー定義の変数を指定します。InstallShield では、ユーザー定義の変数は 変数名 という形式になります。  ***プロジェクト ・** (InstallScript MSI のみ :) このパラメーターで利用されるプロパティはパブリックプロパティでなくてはなりません。このプロパティについて、[プロパティマネージャー] でエントリを作成する必要はありません。プロパティをインストール先として利用する場合、機能ではなくコンポーネントのインストール先として利用しなくてはなりません。*
szLocation	ユーザー定義変数を代替するパス式を指定します。この文字列には、長いパスを指定する場合でも余分な引用符を含めないでください。szLocation の値は、szPropertyVar が使用される方法に基づいて、完全パス (ドライブ文字およびコロンを含む) または部分的パスのどちらかになります。

戻り値

この関数は常に 0 を返します。

FeatureSetTarget の例

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureSetTarget 関数のデモンストレーションを行います。この関数を
* InstallScript プロジェクト用のスクリプトへ追加すると、
* SdAskDestPath ダイアログの INSTALLDIR 値が変更
* 表示されます。そのインストール先が OTHERLOC のコンポーネントを含む
* プロジェクトでは、それも変更されます。
*
¥*---------------------------------------------------------------------*/

function OnBegin()

begin
// 初回インストール用に INSTALLDIR または OTHERLOC ディレクトリプロパティの値を変更します。
//
if (!MAINTENANCE) then
    FeatureSetTarget(MEDIA, "<INSTALLDIR>", "C:¥¥RightHere");
    FeatureSetTarget(MEDIA, "<OTHERLOC>", "C:¥¥SomewhereElse");
endif;

end;
```

FeatureSetupTypeEnum

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

FeatureSetupTypeEnum 関数は、指定のメディアに関連するセットアップタイプをすべて列挙します。これらのセットアップの種類は IDE で設定し、ファイルメディアに格納されます。ListCreate 関数を使用して、listSetupTypes 文字列リストを作成する必要があります。

メモ・この関数は、スクリプト作成機能セットでは使用できません。

構文

FeatureSetupTypeEnum (szFeatureSource, listSetupTypes);

パラメーター

テーブル 91・FeatureSetupTypeEnum のパラメーター

パラメーター	説明
szFeatureSource	セットアップの種類が列挙されるファイルメディアのメディア名を指定します。
listSetupTypes	指定のメディアの全セットアップの種類の一覧を返します。listSetupTypes によって識別される文字列リストは、ListCreate への呼び出しによって既に初期化されている必要があります。

戻り値

テーブル 92・FeatureSetupTypeEnum の戻り値

戻り値	説明
0	FeatureSetupTypeEnum が成功しました。
< 0	FeatureSetupTypeEnum が失敗しました。

FeatureSetupTypeEnum の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureSetupTypeEnum 関数を示します。
*
```

```
＊このスクリプトはメディアないのセットアップの種類を列挙します。
＊
 メモ : このスクリプト例を実行するには、いくつかの機能および
＊      定義されたプロジェクトを作成
＊      ( またはプロジェクトに挿入 ) します。DEFTYPE の値はセットアップの種類のいずれかの
＊      名前でなくてはなりません。
＊
¥*---------------------------------------------------------------*/

#define SDSHOWTITLE ″セットアップの種類の列挙″
#define SDSHOWMSG   MEDIA + ″列挙されたメディアのセットアップの種類は :″
#define SETUPTITLE              ″セットアップの種類の選択″
#define SETUPMSG    ″セットアップの種類を選択します。″
#define DEFTYPE     ″標準″

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include ″Ifx.h″

      export prototype ExFn_FeatureSetupTypeEnum(HWND);

function ExFn_FeatureSetupTypeEnum(hMSI)
    LIST   listID;
    STRING svSetupType;
begin

    // セットアップダイアログで [ 戻る ] ボタンを無効にします。
    Disable(BACKBUTTON);

    // セットアップの種類を格納するためのリストを作成します。
    listID = ListCreate ( STRINGLIST );

    // セットアップの種類名をメディアからリストへ取得します。
    if (FeatureSetupTypeEnum( MEDIA, listID) < 0 ) then
        MessageBox (″FeatureSetupTypeEnum が失敗しました。″, WARNING);
    endif;

    // セットアップの種類を表示します。
    SdShowInfoList (SDSHOWTITLE, SDSHOWMSG, listID);

    // ここで選択ダイアログにセットアップの種類を表示します。
    svSetupType = DEFTYPE;
    SdSetupTypeEx (SETUPTITLE, SETUPMSG, ″″, svSetupType, 0);

    // メモリからリストをリリースします。
    ListDestroy(listID);

end;
```

FeatureSetupTypeGetData

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

FeatureSetupTypeGetData 関数は、指定したセットアップタイプに関連するデータを読み出します。取得後、このデータは、あらゆる目的に使用できます。

FeatureSetupTypeGetData の典型的なのアプリケーションは、カスタム セットアップに関連したダイアログにセットアップ情報を表示します。カスタムダイアログを表示する WaitOnDialog を呼び出した後、switch-case ステートメント内で FeatureSetupTypeGetData を呼び出します。

メモ・この関数は、スクリプト作成機能セットでは使用できません。

構文

FeatureSetupTypeGetData (szFeatureSource, szSetupType, nInfo, nvResult, svResult);

パラメーター

テーブル 93 · FeatureSetupTypeGetData のパラメーター

パラメーター	説明
szFeatureSource	セットアップの種類に関連したデータが取得されるファイルメディアのメディア名を指定します。
szSetupType	セットアップの種類名を指定します。この名前は、「標準」のように、InstallShield インターフェイス で表示されるとおりに指定する必要があります。
nInfo	読み出す情報を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • SETUPTYPE_INFO_DESCRIPTION – 指定のセットアップの種類の説明を取得します。説明は、svResult で戻されます。 • SETUPTYPE_INFO_DISPLAYNAME – セットアップの種類の表示名を取得します。名前は、svResult で戻されます。
nvResult	nInfo がその種類の情報を指定する際、NUMBER 値または LONG 値を返します。
svResult	nInfo がそのタイプの情報を指定する際、STRING 値を返します。

戻り値

テーブル 94 · FeatureSetupTypeGetData の戻り値

戻り値	説明
0	FeatureSetupTypeGetData が成功しました。
< 0	FeatureSetupTypeGetData が失敗しました。追加情報については、FeatureError を呼び出してください。

FeatureSetupTypeGetData の例

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureSetupTypeGetData 関数、FeatureGetData 関数、
* FeatureSetData 関数、SdFeatureDialog2 関数、そして FeatureSelectItem 関数の
* デモンストレーションを行います。
*
* メモ : このスクリプト例を実行するには、
*        次の機能 (f)、サブ機能 (sf)、および コンポーネント (c)
*        を利用します :
*
*       (f) Program_Files
```

```
*              (c) Program_DLLS
*              (c) Program_EXEs
*          (f) Example_Files
*              (sf) Small_Documents
*                  (c) Small_Document_Examples
*              (sf) Books
*                  (c) Book_Examples
*              (sf) Graphics
*                  (c) Graphic_Examples
*          (f) Help_Files
*              (c) Help_Files
*          (f) Utilities
*              (sf) Grammar_Checker
*                  (c) Grammar_Checker
*              (sf) Art_Studio
*                  (c) Art_Studio
*
* Program_Files と Example_Files 機能およびそのサブ機能について、
* 機能プロパティシートにある説明フィールドへ、必ず
* 詳細を記入するようにして下さい。
*
¥*---------------------------------------------------------------*/

// 文字列を定義します。実際のインストールでは [ 文字列エディター ] ビューでこれらを定義し、
// スクリプトの各文字列 ID の前に @ 記号を挿入します。
#define FEAT_SELECT_TITLE              " 機能選択 "
#define FEAT_SELECT_MSG1               " 重要 ! 様々な機能名、"
#define FEAT_SELECT_MSG2               " サブ機能名、説明、そして "
#define FEAT_SELECT_MSG3               " 選択設定に注意してください。"
#define FEAT_SELECT_MSG4               " 重要 ! 変更された機能名、"
#define FEAT_SELECT_MSG5               " サブ機能名、説明、そして "
#define FEAT_SELECT_MSG6               " 選択設定に注意してください。"
#define FEAT_PROGRAMFILES_DISPLAYNAME   " プログラムファイル "
#define FEAT_EXAMPLEFILES_DISPLAYNAME   "Example Files"
#define FEAT_SMALLDOCUMENTS_DISPLAYNAME "Small Documents"
#define FEAT_BOOKS_DISPLAYNAME          "Books"
#define FEAT_GRAPHICS_DISPLAYNAME       "Graphics"
#define SETUP_TYPE                      " 標準 "

// グローバル変数の宣言。
// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_FeatureSetupTypeGetData(HWND);

function ExFn_FeatureSetupTypeGetData(hMSI)
   STRING  svInfo, szInfo, szFeature;
   NUMBER  nvInfo, nInfo, nResult;
begin

   // SETUP_TYPE セットアップの種類の詳細フィールドデータを取得します。
   FeatureSetupTypeGetData (MEDIA, SETUP_TYPE, SETUPTYPE_INFO_DESCRIPTION,
                            nvInfo, svInfo );

   SprintfBox (INFORMATION, "FeatureSetupTypeGetData のデモンストレーション ",
           "FeatureSetupTypeGetData は " +
           "" + SETUP_TYPE + " 詳細フィールド :¥n¥n%s から次の値を取得します ",
           svInfo);
```

```
// FeatureGetData を使って FEAT_PROGRAMFILES_DISPLAYNAME 機能の
// 詳細フィールドデータを取得します。
szFeature = FEAT_PROGRAMFILES_DISPLAYNAME;

nResult = FeatureGetData (MEDIA, szFeature, FEATURE_FIELD_DESCRIPTION,
                          nvInfo, svInfo);

SprintfBox (INFORMATION, "FeatureGetData のデモンストレーション ",
          "FeatureGetData は " +
          "" + FEAT_PROGRAMFILES_DISPLAYNAME +
          " 説明フィールドから次の値を取得します :¥n¥n%s", svInfo);

// セットアップダイアログで [ 戻る ] ボタンを無効にします。
Disable(BACKBUTTON);

// 機能選択ダイアログの元の説明フィールド値
// を表示します。
SdFeatureDialog2 (FEAT_SELECT_TITLE, FEAT_SELECT_MSG1 + FEAT_SELECT_MSG2 +
                  FEAT_SELECT_MSG3, INSTALLDIR, "");

// Program_Files 機能と Example_Files 機能並びにサブ機能の
// 表示名を変更します。
szInfo = " 機能名を変更しました。";

nResult = FeatureSetData (MEDIA, szFeature, FEATURE_FIELD_DISPLAYNAME,
                          nInfo, szInfo);

szFeature = FEAT_EXAMPLEFILES_DISPLAYNAME;

nResult = FeatureSetData (MEDIA, szFeature, FEATURE_FIELD_DISPLAYNAME,
                          nInfo, szInfo);

szFeature = FEAT_EXAMPLEFILES_DISPLAYNAME + "¥¥" +
            FEAT_SMALLDOCUMENTS_DISPLAYNAME;

nResult = FeatureSetData (MEDIA, szFeature, FEATURE_FIELD_DISPLAYNAME,
                          nInfo, szInfo);

szFeature = FEAT_EXAMPLEFILES_DISPLAYNAME + "¥¥" +
            FEAT_BOOKS_DISPLAYNAME;

nResult = FeatureSetData (MEDIA, szFeature, FEATURE_FIELD_DISPLAYNAME,
                          nInfo, szInfo);

szFeature = FEAT_EXAMPLEFILES_DISPLAYNAME + "¥¥" + FEAT_GRAPHICS_DISPLAYNAME;

nResult = FeatureSetData (MEDIA, szFeature, FEATURE_FIELD_DISPLAYNAME,
                          nInfo, szInfo);

// Program_Files 機能と Example_Files 機能並びにサブ機能に表示される
// 説明を変更しました。
szFeature = FEAT_PROGRAMFILES_DISPLAYNAME;
szInfo = " 説明フィールド値を変更しました。";

nResult = FeatureSetData (MEDIA, szFeature, FEATURE_FIELD_DESCRIPTION,
                          nInfo, szInfo);

szFeature = FEAT_EXAMPLEFILES_DISPLAYNAME;
```

```
    nResult = FeatureSetData (MEDIA, szFeature, FEATURE_FIELD_DESCRIPTION,
                              nInfo, szInfo);

    szFeature = FEAT_EXAMPLEFILES_DISPLAYNAME + "¥¥" +
                FEAT_SMALLDOCUMENTS_DISPLAYNAME;

    nResult = FeatureSetData (MEDIA, szFeature, FEATURE_FIELD_DESCRIPTION,
                              nInfo, szInfo);

    szFeature = FEAT_EXAMPLEFILES_DISPLAYNAME + "¥¥" + FEAT_BOOKS_DISPLAYNAME;

    nResult = FeatureSetData (MEDIA, szFeature, FEATURE_FIELD_DESCRIPTION,
                              nInfo, szInfo);

    szFeature = FEAT_EXAMPLEFILES_DISPLAYNAME + "¥¥" + FEAT_GRAPHICS_DISPLAYNAME;

    nResult = FeatureSetData (MEDIA, szFeature, FEATURE_FIELD_DESCRIPTION,
                              nInfo, szInfo);

    // Program_Files 機能、Example_Files 機能 （並びに拡張子に従ったすべてのサブ機能）
    // の選択を解除します。
    FeatureSelectItem (MEDIA, FEAT_PROGRAMFILES_DISPLAYNAME, FALSE);
    FeatureSelectItem (MEDIA, FEAT_EXAMPLEFILES_DISPLAYNAME, FALSE);

    // 機能を再び表示します。名前、詳細、そして選択設定が
    // 変更されています。

    SdFeatureDialog2 (FEAT_SELECT_TITLE, FEAT_SELECT_MSG4 + FEAT_SELECT_MSG5 +
                      FEAT_SELECT_MSG3, INSTALLDIR, "");

end;
```

FeatureSetupTypeSet

プロジェクト・この情報は、次のプロジェクトの種類に適用します：

- *InstallScript*
- *InstallScript MSI*

FeatureSetupTypeSet 関数は、szFeatureSource によって参照されるファイルメディアライブラリの指定のセットアップタイプを設定します。FeatureSetupTypeSet を使用して、SdSetupTypeEx のようなセットアップタイプダイアログの設定を上書きできます。

メモ・この関数は、スクリプト作成機能セットでは使用できません。

構文

```
FeatureSetupTypeSet ( szFeatureSource, szSetupType );
```

パラメーター

テーブル 95 - FeatureSetupTypeSet のパラメーター

パラメーター	説明
szFeatureSource	セットアップの種類が設定されるファイルメディアのメディア名を指定します。
szSetupType	設定するセットアップの種類を指定します。

戻り値

テーブル 96 - FeatureSetupTypeSet の戻り値

戻り値	説明
0	FeatureSetupTypeSet が成功しました。
< 0	FeatureSetupTypeSet 失敗しました。

FeatureSetupTypeSet の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureSetupTypeSet 関数のデモンストレーションを行います。
*
* メモ : このスクリプト例を実行するには、
*      機能を含むプロジェクトを作成 ( またはプロジェクトに挿入 )
*      します。セットアップの種類のデフォルトの
*    機能選択を必ず指定してください。[ 最小 ] セットアップ タイプを
*      含まない場合、下の #define SETUP_TYPE 行で
*    セットアップの種類のひとつを SETUP_TYPE と
*    定義してください。
*
¥*---------------------------------------------------------------*/

#define SETUP_TYPE   " 最小 "
#define SETUPTYPE_TITLE              " セットアップの種類の選択 "
#define SDSETUPMSG   " セットアップの種類を " + SETUP_TYPE + " 以外から選択してください。"
#define SDFEATTITLE  " 機能の選択 "
#define SDFEATMSG1   "FeatureSetupTypeSet の前の機能選択。"
#define SDFEATMSG2   "FeatureSetupTypeSet の後の機能選択。"
#define MSG1         "FeatureSetupTypeSet はすべてを選択します ¥n"
#define MSG2         "" + SETUP_TYPE + " セットアップの種類にある機能。"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_FeatureSetupTypeSet(HWND);

function ExFn_FeatureSetupTypeSet(hMSI)
```

```
    STRING szSetupType, svSetup, svDir;
begin

  // セットアップダイアログで [ 戻る ] ボタンを無効にします。
  Disable(BACKBUTTON);

  // SETUP_TYPE 以外のセットアップの種類を選択して、デフォルトの選択設定を
  // 表示します。
  SdSetupTypeEx (SDSETUPTITLE, SDSETUPMSG, "", svSetup, 0);

  // 選択したセットアップの種類の機能選択を表示します。
  svDir = INSTALLDIR;
  SdFeatureMult (SDFEATTITLE, SDFEATMSG1, svDir, "");

  MessageBox (MSG1 + MSG2, INFORMATION);

  // ここで SETUP_TYPE の機能を選択して、前回のセットアップの種類選択を
  // 変更または上書きします。その他すべては選択解除されています。
  szSetupType = SETUP_TYPE;
  if (FeatureSetupTypeSet (MEDIA, szSetupType) < 0) then
     MessageBox ("FeatureSetupTypeSet failed.", SEVERE);
  endif;

  // 新しい機能選択を表示します。
  SdFeatureMult (SDFEATTITLE, SDFEATMSG2, svDir, "");

end;
```

FeatureSpendCost

プロジェクト・この情報は、InstallScript プロジェクトに適用されます。

FeatureSpendCost 関数は、セットアップの外部にあるイベントによって使用された特定のコストについて、進行状況バーを更新する必要があることをセットアップに伝えます。この関数は、セットアップ自身が相応のコストを使用したかのように適切に進行状況バーを更新する効果があります。この関数は、**FeatureTransferData** や **FeatureMoveData** の呼び出しによって発生するイベントなど、標準ファイル操作イベント中のみに呼び出すことができます。通常、この関数は、コストを使用する外部のイベントが発生した際に発生する OnInstalling または OnInstalled イベントの実行中に呼び出されます。

メモ・この関数を、XCopyFile または CopyFile などのセットアップエンジンが実行するファイル転送操作に呼び出す必要はありません。これらのメカニズムは、操作中自動的にステータスバーをアップデートします。ただし、FeatureAddCost は、この場合、インストールが追加のコストを認識するように (ファイル転送操作が始まる前に) 呼び出す必要があります。

構文

```
FeatureSpendCost ( nCostHigh, nCostLow );
```

パラメーター

テーブル 97 - FeatureSpendCost のパラメーター

パラメーター	説明
nCostHigh	使用するインストール コストの上位 31 ビット を指定します。
nCostLow	使用するインストール コストの下位 31 ビットを指定します。

戻り値

テーブル 98 - FeatureSpendCost の戻り値

戻り値	説明
ISERR_SUCCESS	関数が進行状況バーを正常にアップデートしたことを示します。
< ISERR_SUCCESS	関数が進行状況バーをアップデートできなかったことを示します。

FeatureSpendUninstallCost

***プロジェクト -** この関数は、InstallScript プロジェクトに適用します。*

FeatureSpendUninstallCost 関数は、セットアップの外部にあるイベントによって使用された一定のアンインストール‘コスト’の進行状況バーを更新する必要があることをセットアップに伝えます。この関数は、セットアップ自身が相応のコストを使用したかのように適切に進行状況バーを更新する効果があります。この関数は、**FeatureTransferData** や **FeatureMoveData** の呼び出しによって発生するイベントなど、標準ファイル操作イベント中のみに呼び出すことができます。通常、この関数は、コストを使用する外部のイベントが発生した際に発生する OnInstalling または OnInstalled イベントの実行中に呼び出されます。

***メモ -** この関数を、XCopyFile または CopyFile 操作 などアンインストール ログ ファイルにリストされるファイル転送操作に呼び出す必要はありません。ステータスバーは、ログファイルにリストされるアイテムに自動的に更新されます。また、セットアップはログファイルにリストされるすべてのアイテムを自動的に明らかにするので、ログファイルにリストされるアイテムに FeatureAddUninstallCost を呼び出す必要はありません。ただし、FeatureAddCost は、この場合、ファイル転送メカニズムが追加のコストを認識するように呼び出す必要があります。*

構文

```
FeatureSpendUninstallCost ( nOps, nOpType);
```

パラメーター

テーブル 99 - FeatureSpendUninstallCost のパラメーター

パラメーター	説明
nOps	使用する操作の数。
nOpType	操作の種類を示します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • FEATURE_OPCOST_UNINSTALL_FILE – 操作をファイルのアンインストールとして指定します。値 4096 を使用して、この定数を指定することもできます。また、異なるサイズを試して、カスタム操作に使用するサイズを判断できます。 • FEATURE_OPCOST_UNINSTALL_REGORINI – 操作をレジストリファイルのアンインストールとして指定します。値 2048 を使用して、この定数を指定することもできます。また、異なるサイズを試して、カスタム操作に使用するサイズを判断できます。 • FEATURE_OPCOST_UNINSTALL_UNREGFILE – 操作をファイルの登録解除として指定します。値 16384 を使用して、この定数を指定することもできます。また、異なるサイズを試して、カスタム操作に使用するサイズを判断できます。 ***メモ -*** *また、このパラメーターで数字の値を渡して、特定のコストの操作を示すこともできます。異なる値を試して、指定されたカスタム操作に適切な値を判別する必要がある場合もあります。追加されたアンインストールコストの合計は、nOps を nOpType で乗じた値です。*

戻り値

テーブル 100 - FeatureSpendUninstallCost のパラメーター

戻り値	説明
ISERR_SUCCESS	関数が進行状況バーを正常にアップデートしたことを示します。
< ISERR_SUCCESS	関数が進行状況バーをアップデートできなかったことを示します。

FeatureStandardSetupTypeSet

プロジェクト - *この情報は、次のプロジェクトの種類に適用します :*

- *InstallScript*
- *InstallScript MSI*

FeatureStandardSetupTypeSet 関数は、現在のセットアップタイプを nSetupType が指定した標準セットアップタイプに設定します。この関数は、FeatureSetupTypeSet 関数をセットアップタイプを設定する適切な文字列と共に呼び出してセットアップタイプを設定しようと試みます。

構文

FeatureStandardSetupTypeSet (szFeatureSource, nSetupType);

パラメーター

テーブル 101 ・ FeatureStandardSetupTypeSet のパラメーター

パラメーター	説明
szFeatureSource	セットアップの種類が設定されるファイルメディアのメディア名を指定します。
nSetupType	設定するセットアップの種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ TYPICAL ・ COMPLETE ・ COMPACT ・ CUSTOM 次にリストされている定義済みのグローバルスクリプトをカスタマイズすることで、各数値定数に関連付けられている文字列セットアップの種類の値をカスタマイズすることができます。関数は、適切なスクリプト変数を試みてセットアップまたはインストールの種類を設定した後、英語、および日本語 (以下参照) のセットアップの種類名を試みます。また、COMPLETE セットアップの種類が指定されると、関数は COMPLETE の値をチェックしたあと、すべての TYPICAL 値を試みます。セットアップの種類にこれらのどの値も設定することができなかった場合、関数は失敗を返します。次の定義済みグローバルスクリプト変数はカスタマイズ可能です。 ・ SETUPTYPE_STR_TYPICAL ― TYPICAL が指定された場合、デフォルトで設定されるセットアップの種類を定義します。 ・ SETUPTYPE_STR_COMPLETECOMPLETE― が指定された場合、デフォルトで設定されるセットアップの種類を定義します。 ・ SETUPTYPE_STR_COMPACT ― COMPACT が指定された場合、デフォルトで設定されるセットアップの種類を定義します。 ・ SETUPTYPE_STR_CUSTOM ― CUSTOM が指定された場合、デフォルトで設定されるセットアップの種類を定義します。

次のテーブルは、各定数が nSetupType パラメーターで指定された場合に FeatureStandardSetupTypeSet 関数が使用する値を示します。

テーブル 102 ・ nSetupType の定数

定数	使用文字列
TYPICAL	SETUPTYPE_STR_TYPICAL 標準

テーブル 102 • nSetupType の定数 (続き)

定数	使用文字列
COMPLETE	SETUPTYPE_STR_COMPLETE 完全
COMPACT	SETUPTYPE_STR_COMPACT Compact
CUSTOM	SETUPTYPE_STR_CUSTOM カスタム

戻り値

テーブル 103 • FeatureStandardSetupTypeSet の戻り値

戻り値	説明
>= ISERR_SUCCESS	FeatureStandardSetupTypeSet は指定されたセットアップタイプを正常に設定しました。
< ISERR_SUCCESS	FeatureStandardSetupTypeSet は指定されたセットアップタイプを正常に設定できませんでした。

FeatureTotalSize

プロジェクト • *この情報は、次のプロジェクトの種類に適用します :*

- *InstallScript*
- *InstallScript MSI*

FeatureTotalSize 関数は、szFeature で参照された機能の合計サイズをバイトで返します。

- サイズ計算の中にサブ機能を入れるには、bIncludeSubcomp を TRUE に設定してください。
- 特定のメディア中のすべての機能の合計サイズを得るには、set szFeature を null 文字列 ("") に、そして bIncludeSubcomp を TRUE に設定します。

メモ • *FeatureTotalSize によって戻された値と SdFeatureDialog、SdFeatureDialog2、SdFeatureDialogAdv、SdFeatureMult ダイアログで表示される必要空き容量値に誤差が出る場合もあります。この違いは、機能と関連するコンポーネントがオペレーティングシステムまたは言語によって現在フィルターされているかどうか、またはターゲットデバイス上のクラスタサイズをこの関数が考慮しないために発生します。こういった要因を計算に入れたドライブ容量を取得するには、FeatureGetCost を呼び出します。*

構文

FeatureTotalSize (szFeatureSource, szFeature, bIncludeSubcomp, bTargetSize);

パラメーター

テーブル 104 ・ FeatureTotalSize のパラメーター

パラメーター	説明
szFeatureSource	選択した機能の合計サイズが戻されるファイルメディアのファイル名を指定します。
szFeature	サイズが読み出される機能の名前を指定します。メディア全体のサイズを得るには、このパラメーターにヌル文字列 ("") を渡します。
bIncludeSubcomp	選択した s z Feature のサブ機能を含むか否かを示します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ TRUE – 選択したサブ機能をサイズ計算に含めます。 ・ FALSE – サブ機能をサイズ計算に含めません。
bTargetSize	オリジナルの非圧縮サイズ、またはメディアライブラリでのサイズの何れを読み出すのかを示します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ TRUE – オリジナル、非圧縮サイズを読み出します。 ・ FALSE – メディアライブラリでのサイズを読み出します。

戻り値

テーブル 105 ・ FeatureTotalSize の戻り値

戻り値	説明
XXXX	選択した機能の合計サイズをバイト数で表した値です。
< 0	FeatureTotalSize が失敗しました。追加情報については、FeatureError を呼び出してください。

FeatureTotalSize の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FeatureListItems 関数、SdFeatureMult 関数、そして FeatureTotalSize 関数の
* デモンストレーションを行います。
*
* メモ : このスクリプト例を実行するには、
次の機能 (f)、サブ機能 (sf)、および コンポーネント (c)
*       を利用します :
```

```
*
*       (f) Program_Files
*             (c) Program_DLLS
*             (c) Program_EXEs
*          (f) Example_Files
*             (sf) Small_Documents
*                (c) Small_Document_Examples
*             (sf) Books
*                (c) Book_Examples
*             (sf) Graphics
*                (c) Graphic_Examples
*          (f) Help_Files
*             (c) Help_Files
*          (f) Utilities
*             (sf) Grammar_Checker
*                (c) Grammar_Checker
*             (sf) Art_Studio
*                (c) Art_Studio
*
¥*---------------------------------------------------------------*/

#define FEAT_SELECT_TITLE            "機能選択"
#define FEAT_SELECT_MSG              "インストールする機能とサブ機能を選択します。"
#define FEATTOTSIZEMSG1     "機能選択肢を変更し、¥n"
#define FEATTOTSIZEMSG2     "FeatureTotalSize の呼び出しに反映されたサイズの変更を確認しますか ?"

   // グローバル変数の宣言。

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

     export prototype ExFn_FeatureTotalSize(HWND);

function ExFn_FeatureTotalSize(hMSI)
   STRING  szDir, svString;
   NUMBER  nResult, nDone;
   LIST    listCompList, listTemp;
begin

   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   // トップレベル機能すべての文字列リストを作成します。
   listCompList = ListCreate (STRINGLIST);
   FeatureListItems (MEDIA, "", listCompList);

   // トップレベル機能の文字列リストを表示します。
   SdShowInfoList ("MEDIA 機能をリストします ", "MEDIA は " +
            " 次のトップレベル機能を含みます :", listCompList);

   // listCompList の各トップレベル機能を順番に取得し、
   // そのサブ機能がある場合、すべてをリストにして表示します。
   nResult = ListGetFirstString ( listCompList, svString );
   while ( nResult != END_OF_LIST )
      listTemp = ListCreate (STRINGLIST);
      FeatureListItems (MEDIA, svString, listTemp);
      SdShowInfoList (" サブ機能リスト ", svString + " は " +
               " 次のサブ機能を含みます :", listTemp);
      ListDestroy (listTemp);
```

```
    nResult = ListGetNextString (listCompList, svString);
  endwhile;

  // 機能選択ダイアログと、選択されたすべての機能の合計サイズを
  // 表示します。選択肢を変更するためにループし、FeatureTotalSize への呼び出しに
  // 反映した合計サイズの変更を確認します。
  nDone = YES;
  while (nDone = YES)
    szDir = INSTALLDIR;
    SdFeatureMult (FEAT_SELECT_TITLE, FEAT_SELECT_MSG, szDir, "" );

    nResult = FeatureTotalSize(MEDIA, "", TRUE, TRUE);
    SprintfBox (INFORMATION, "", " 全ファイルの合計サイズ " +
            " 選択した機能 :¥n¥n%ld", nResult);
    nDone = AskYesNo (FEATTOTSIZEMSG1 + FEATTOTSIZEMSG2, YES);
  endwhile;

  ListDestroy (listCompList);

end;
```

FeatureTransferData

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI － InstallScript ユーザー インターフェイス (UI) のスタイルが、外部 UI ハンドラーとして InstallScript エンジンを使用する従来型のスタイルの場合*

この情報は、InstallScript UI に新しいスタイル (InstallScript エンジンを埋め込み UI ハンドラーとして使用するスタイル) が使用されている InstallScript MSI プロジェクトには適用しません。詳しくは、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラとして使用する方法の違い」を参照してください。

イベント指向スクリプトを使用する場合、OnFirstUIBefore イベントの後に自動的に **FeatureTransferData** 関数が呼び出され、OnMoving、OnMoved、および機能イベントと対話します。FeatureTransferData は、選択状態および現在のインストール状態に基づいて、機能を適切にインストールしたりアンインストールします。

FeatureTransferData は、以下を実行します。

- **FeatureTransferData** は、機能ダイアログまたはセットアップの種類 ダイアログでユーザーによって選択された機能で、現在インストールされていない機能をインストールします。
- **FeatureTransferData** は、選択解除されていて、現在インストールされている機能をアンインストールします。

インストール初期設定中に既存のログファイルを読み取ることによって、インストールは機能が現在インストールされているかどうかを判別します。インストール初期設定中に有効なログファイルが検出されなかった場合、すべての機能がインストールされていないとみなされます。

FeatureTransferData を呼び出すと、ファイルがインストールされた後、呼び出しが戻る前に Do(SELFREGISTRATIONPROCESS); が自動的に呼び出されます。従って、インストールでファイル転送の後、自己登録の前に追加アクションの処理が必要な場合、それらのアクションを OnMoved イベントに配置します。OnMoved イベントはファイル転送のあと、バッチ自己登録が発生する前に呼び出されます。

構文

FeatureTransferData (szFeatureSource);

パラメーター

テーブル 106 ・ FeatureTransferData のパラメーター

パラメーター	説明
szFeatureSource	インストールおよびアンインストールに対して機能が指定されているファイルメディアのメディア名を指定します。通常は、システム変数の MEDIA をこのパラメーターで渡します。

戻り値

テーブル 107 ・ FeatureTransferData の戻り値

戻り値	説明
0	関数が機能のインストールおよびアンインストールを正しく実行したことを示します。
< 0	関数が機能のインストールおよびアンインストールを実行できなかったことを示します。

追加情報

FeatureTransferData を呼び出す前に次の関数の一つを呼び出して、呼び出し結果に影響を与えることができます :

- **FeatureReinstall** – **FeatureTransferData** が呼び出されたときに、現在選択されている機能を再インストールします。
- **FeatureRemoveAll** – **FeatureTransferData** が呼び出されたときに、すべての機能を削除 (アンインストール) します。

FeatureUpdate

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

FeatureUpdate 関数は、FeatureTransferData (または FeatureMoveData) への次の呼び出しを引き起こし、FeatureTransferData を呼び出したときに既にインストールされている、メンテナンス / アンインストール機能を除くすべての機能を再インストールします。(この機能はメディアビルダーによって .cab ファイルに自動的に配置され、InstallShield には表示されません。)

構文

FeatureUpdate (szReserved);

パラメーター

テーブル 108 ・ FeatureUpdate のパラメーター

パラメーター	説明
szReserved	このパラメーターにヌル文字列 (“”) を渡します。他の値は使用できません。

戻り値

テーブル 109 ・ FeatureUpdate の戻り値

戻り値	説明
0	関数が成功したことを示します。
< 0	関数が失敗したことを示します。

追加情報

後に続くメンテナンス処理 (更新、修復、またはアンインストール) でメンテナンス / アンインストール機能を利用しないインストーラーでは FeatureUpdate を呼び出します。FeatureUpdate は一般的にアップデートまたは追加インストーラーで呼び出されます。これらのインストーラーでは、インストーラーメディアが以前のメディアの要素のすべてを含まないため、後に続くメンテナンス処理では利用できません。

FeatureUpdate は FeatureReinstall に似ていますが、FeatureReinstall ではメンテナンス / アンインストール機能も再インストールされます。

***メモ ・** FeatureUpdate と FeatureReinstall を両方呼び出すと、FeatureUpdate への呼び出しには影響がありません。したがって FeatureUpdate を呼び出す場合は、同じインストーラーに FeatureReinstall を呼び出さないでください。*

アップデートインストーラーとして InstallShield の同じバージョンを使ってインストールされたアプリケーションのみをアップデートするには FeatureUpdate を呼び出します。以前のバージョンの InstallShield を使ってインストールされたアプリケーションをアップデートするのに FeatureUpdate を呼び出した場合、メンテナンス / アンインストールファイルはアップデートされません。このため、後に続くメンテナンス処理では、以前のバージョンで作成されたメンテナンス / アンインストールファイルが利用されます。

次の情報はこの状況に適用します : アプリケーションのより新しいバージョンが存在する場合は、インストールを中断するインストーラーによって、アプリケーションの以前のバージョンがインストールされました。この場合、インストールされたアプリケーションのバージョン番号を変更するアップデートインストーラーでは FeatureUpdate を呼び出さないでください。これを行うと、アップデートインストーラーを実行した後に現在インストールされているアプリケーションをアンインストールすることができなくなります。

FeatureValidate

***プロジェクト ・** この情報は、InstallScript プロジェクトに適用します。*

メモ・この関数は、スクリプト作成機能セットでは使用できません。

FeatureValidate 関数は、ファイルメディアライブラリまたは指定された機能のパスワードを検証します。

構文

FeatureValidate (szMediaLibrary, szFeature, szPassword);

パラメーター

テーブル 110 ・ FeatureValidate のパラメーター

パラメーター	説明
szMediaLibrary	パスワードが検証されるファイルメディアライブラリの名前を指定します。
szFeature	機能の名前を指定します。このパラメーターがヌル文字列 ("") の場合は、メディアライブラリ全体が対象となります。
szPassword	検証されるパスワードを指定します。

戻り値

テーブル 111 ・ FeatureValidate の戻り値

戻り値	説明
0	FeatureValidate が成功しました。
< 0	FeatureValidate が失敗しました。追加情報については、FeatureError を呼び出してください。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

FeatureValidate の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdSetupTypeEx 関数、SdFeatureDialog 関数、FeatureIsItemSelected 関数、
* FeatureGetDat 関数、FeatureValidate 関数 のデモンストレーションを行います。
*
* メモ : このスクリプト例を実行するには、
*        次の機能 (f)、サブ機能 (sf)、および コンポーネント (c)
*        を利用します :
*
*           (f) Program Files
*              (c) Program DLLS
*              (c) Program EXEs
```

```
*           (f) Example Files
*              (sf) Small Documents
*                 (c) Small Document Examples
*              (sf) Books
*                 (c) Book Examples
*              (sf) Graphics
*                 (c) Graphic Examples
*           (f) Help Files
*              (c) Help Files
*           (f) Utilities
*              (sf) Grammar Checker
*                 (c) Grammar Checker
*              (sf) Art Studio
*                 (c) Art Studio
*           (f) Evaluation Copy
*           (c) Evaluation Copy
*              (c) Help Files
*
* "ダミー" ファイルをコンポーネントに挿入します。ここで、
* Program EXE のコンポーネントへ挿入するメイン EXE ( 下の MAIN_EXE)
* ( 下の MAIN_EXE ) の正しいファイル名を定義してください。
* セットアップを [ プログラムファイル ] 機能へ割り当てられたパスワードと一緒に、
* またはパスワード無しで実行します ( 毎回メディアを再ビルドするように
* してください。)
*
* またビルボード (Bbrd1.bmp、Bbrd2.bmp、および Bbrd3.bmp と名づける )
* を作成して、[ サポートファイル ] ビューの
* [ 言語非依存 ] フォルダー内のプロジェクトに追加することもできます。
*
* このスクリプト例はファイルをインストールし、アイコンを
* [ スタート プログラム ] メニューに追加して、アンインストール機能を
* 提供します。
*
¥*---------------------------------------------------------------*/

// 文字列を定義します。実際のインストールでは [ 文字列エディター ] ビューでこれらを定義し、
// スクリプトの各文字列 ID の前に @ 記号を挿入します。
#define FEATURE_SELECT_TITLE            "機能選択"
#define FEATURE_SELECT_MSG              "インストールする機能とサブ機能を選択します。"
#define FEATURE_PROGRAMFILES_DISPLAYNAME  "プログラムファイル"
#define PASSWORD_PROMPT                 "パスワードを入力してください。"
#define PASSWORD_ERRMSG "パスワードが間違っています。再度入力してください。"


// グローバル変数の宣言。
STRING  svData, svLogFile,  szProgram, szFeature, svResult, svSetupType,
svDir;
BOOL    bInitStepsDone, bPwdValid;
NUMBER  nvData, nResult;

#include "ifx.h"

function OnFirstUIBefore()
begin
   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   // セットアップの種類を取得します。
   svDir = TARGETDIR;
```

```
    SdSetupTypeEx (SETUPTYPE_TITLE, SETUPTYPE_MSG, "", svSetupType, 0);

    // ユーザーが [ カスタム ] セットアップの種類を選択したら、機能選択ダイアログを表示します。
    if (svSetupType = SETUPTYPE_CUSTOM) then
        SdFeatureDialog (FEATURE_SELECT_TITLE, FEATURE_SELECT_MSG, svDir, "");
    endif;

    // セットアップダイアログで [ 戻る ] ボタンを有効にします。
    Enable(BACKBUTTON);

    // [ プログラムファイル ] 機能が選択され、それにパスワードが
    // 割り当てられている場合、ユーザーはパスワードを入力して
    // それが認証される必要があります。
    nResult = FALSE;
    nvData = FALSE;
    nResult = FeatureIsItemSelected (MEDIA, FEATURE_PROGRAMFILES_DISPLAYNAME);
    FeatureGetData (MEDIA, FEATURE_PROGRAMFILES_DISPLAYNAME,
        FEATURE_FIELD_PASSWORD, nvData, svData);
    if (nResult && nvData) then
        bPwdValid = FALSE;
        Disable (BACKBUTTON);  // [ 戻る ] ボタンが不要かサポートされていない
        while  (!bPwdValid)
            AskText (PASSWORD_PROMPT, "", svResult);
            nResult = FeatureValidate (MEDIA, FEATURE_PROGRAMFILES_DISPLAYNAME,
                svResult);
            if  (nResult = 0) then
                bPwdValid = TRUE;
            else
                MessageBox (PASSWORD_ERRMSG, SEVERE);
            endif;
        endwhile;
        Enable (BACKBUTTON); // [ 戻る ] ボタンのデフォルトのステータスを復元します。
    endif;
end;
```

FileCompare

FileCompare 関数は、2 つのファイルのサイズ、変更日、コンテンツおよびバージョンを比較します。

構文

```
FileCompare (szFileName1, szFileName2, nCompareFlag);
```

パラメーター

テーブル 112 ・ FileCompare のパラメーター

パラメーター	説明
szFileName1	比較する最初のファイルの名前を指定します。インストールはシステム変数 SRCDIR をファイルのパスとして使用します。
szFileName2	比較する 2 番目のファイルの名前を指定します。インストールは以下のシステム変数をファイルのパスとして使用します : ・ TARGETDIR (InstallScript インストール) ・ INSTALLDIR (基本の MSI または InstallScript MSI インストール)
nCompareFlag	比較オプションを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ COMPARE_SIZE—2 つのファイルのサイズを比較します。 ・ COMPARE_DATE —2 種類のファイルの変更日を比較します。 ・ COMPARE_VERSION —2 つのファイルのバージョンを比較します。 ・ COMPARE_MD5_SIGNATURE—2 種類のファイルの MD5 シグネチャを比較します。MD5 シグネチャが一致する場合、ファイルの内容は同じです。

戻り値 (COMPARE_MD5_SIGNATURE の指定時)

テーブル 113 ・ FileCompare の戻り値 (COMPARE MD5_SIGNATURE)

戻り値	説明
>= ISERR_SUCCESS	関数はファイルの比較に成功しました。詳しい内容 : ・ EQUALS ー ファイル MD5 シグネチャは同じです (ファイルのコンテンツは同じです)。 ・ NOT_EQUALS ー ファイルの MD5 シグネチャが異なります。
<= ISERR_SUCCESS	関数は、ファイルの比較に失敗しました。

戻り値 (COMPARE_SIZE, COMPARE_DATE、または COMPARE_VERSION の指定時)

テーブル 114 · FileCompare の戻り値 (COMPARE_SIZE, COMPARE_DATE, or COMPARE_VERSION)

戻り値	説明
>= ISERR_SUCCESS	関数はファイルの比較に成功しました。詳しい内容 : · EQUALS— 最初のファイルと 2 番目のファイルは、日付、サイズ、またはバージョンが同じです。 · GREATER_THAN— 最初のファイルは 2 番目のファイルよりも新しい、または大きいことを示します。 · LESS_THAN— 最初のファイルは 2 番目のファイルよりも古い、または小さいことを示します。
<= ISERR_SUCCESS	関数は、ファイルの比較に失敗しました。

FileCompare の例

メモ · *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FileCompare 関数のデモンストレーションを行います。
*
* このスクリプト例は、FileCompare を 3 回呼び出します :
*
* -- 最初の呼び出しでは、SDIR 内の FILE_COMP1 のバージョンと
*   TDIR 内の FILE_COMP2 のバージョンを比較します。
*
* -- 次の呼び出しでは、FILE_COMP1 が FILE_COMP2 よりも先に
*   作成されたかを確認します。
*
* -- 3 回目の呼び出しでは、FILE_COMP1 が FILE COMP2 よりも
*   小さいかを確認します。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数を
*     ターゲットコンピューター上の既存のディレクトリ内の
*     既存ファイルを参照するよう設定します。
*
¥*---------------------------------------------------------------------*/

#define  SDIR       "C:¥¥"
#define  TDIR       "C:¥¥"
#define  FILE_COMP1  "Example1.dll"
#define  FILE_COMP2  "Example2.dll"
#define  MSG_TITLE   "FileCompare の例 "
```

```
// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_FileCompare(HWND);

function ExFn_FileCompare(hMSI)
   NUMBER nResult;
begin

   /*----------------------------------------------------------------*¥
    *
    * 2 つのファイルのバージョンを比較します。
    *
   ¥*----------------------------------------------------------------*/

   nResult = FileCompare (SDIR ^ FILE_COMP1, TDIR ^ FILE_COMP2,
                          COMPARE_VERSION);

   // エラーを報告して、どのファイルも検出されなかった場合に終了します。
   if (nResult = FILE_NOT_FOUND) then
      SprintfBox (INFORMATION, MSG_TITLE, "%s および / または %s が見つかりませんでした。",
                  FILE_COMP1, FILE_COMP2);
      abort;
   endif;

   // ファイルが存在するので、比較結果を報告します。
   switch (nResult)
      case EQUALS:
         SprintfBox (INFORMATION, MSG_TITLE, "%s は %s と同じバージョンです。",
                     FILE_COMP1, FILE_COMP2);
      case GREATER_THAN:
         SprintfBox (INFORMATION, MSG_TITLE, "%s は %s よりも新しいバージョンです。",
                     FILE_COMP1, FILE_COMP2);
      case LESS_THAN:
         SprintfBox (INFORMATION, MSG_TITLE, "%s は %s よりも古いバージョンです。",
                     FILE_COMP1, FILE_COMP2);
      case OTHER_FAILURE:
         SprintfBox (INFORMATION, MSG_TITLE,
                     "%s および / または %s にはバージョン情報がありません。",
                     FILE_COMP1, FILE_COMP2);
   endswitch;

   /*----------------------------------------------------------------*¥
    *
    * 作成日を比較します。
    *
   ¥*----------------------------------------------------------------*/

   nResult = FileCompare (SDIR ^ FILE_COMP1, TDIR ^ FILE_COMP2,
                          COMPARE_DATE);

   switch (nResult)
      case LESS_THAN:
         SprintfBox (INFORMATION, MSG_TITLE,
                     "%s は %s よりも前に作成されました。",
                     FILE_COMP1, FILE_COMP2);
      case GREATER_THAN:
         SprintfBox (INFORMATION, MSG_TITLE,
                     "%s は %s よりも前に作成されました。",
```

```
                FILE_COMP2, FILE_COMP1);
        case EQUALS:
            SprintfBox (INFORMATION, MSG_TITLE,
                "%s と %s の作成日時は同じです。",
                FILE_COMP1, FILE_COMP2);
        デフォルト :
            SprintfBox (INFORMATION, MSG_TITLE,
                "%s と %s の作成日時を比較することができませんでした。",
                FILE_COMP2, FILE_COMP1);
    endswitch;

    /*---------------------------------------------------------*¥
     *
     * ファイルサイズを比較します。
     *
    ¥*---------------------------------------------------------*/

    nResult = FileCompare (SDIR ^ FILE_COMP1, TDIR ^ FILE_COMP2,
        COMPARE_SIZE);

    switch (nResult)
        case LESS_THAN:
            SprintfBox (INFORMATION, MSG_TITLE,
                "%s は %s よりも小さいです。",
                FILE_COMP1, FILE_COMP2);
        case GREATER_THAN:
            SprintfBox (INFORMATION, MSG_TITLE,
                "%s は %s よりも小さいです。",
                FILE_COMP2, FILE_COMP1);
        case EQUALS:
            SprintfBox (INFORMATION, MSG_TITLE,
                "%s と %s のサイズは同じです。",
                FILE_COMP1, FILE_COMP2);
        デフォルト :
            SprintfBox(INFORMATION, MSG_TITLE,
                "%s と %s のサイズを比較することができませんでした。",
                FILE_COMP1, FILE_COMP2);
    endswitch;

end;
```

FileDeleteLine

FileDeleteLine 関数は、最初と最後の行番号を使って、テキストファイルから (最初と最後の行を含んだ) 範囲を削除します。この関数はバイナリファイルではなく、行型テキストファイル上で動作します。FileGrep と共に FileDeleteLine を利用してファイル内のテキスト行を検索して削除することができます。

注意 : *szFileName が指定するファイルは OpenFile への呼び出しで既に開いた状態でなくてはなりません。szFileName が既に開いている場合、FileDeleteLine の前に CloseFile を呼び出します。*

構文

```
FileDeleteLine ( szFileName, nStartLineNum, nEndLineNum );
```

パラメーター

テーブル 115 ・ FileDeleteLine のパラメーター

パラメーター	説明
szFileName	システム変数 SRCEIR が指定するディレクトリ内に配置されたファイルの完全修飾されていない名前、またはファイルへの完全修飾パスを指定します。nStartLineNum と nEndLineNum が指定した行が存在する場合は、そのファイルから削除されます。
nStartLineNum	ファイルから削除する最初の行の番号を指定します。行番号は 0 から始まることに注意してください。
nEndLineNum	ファイルから削除する最後の行の番号を指定します。行番号は 0 から始まることに注意してください。nStartLineNum が指定する行からファイルの終わりまで削除する場合、このパラメーターに定義済み定数 DELETE_EOF を渡します。

戻り値

テーブル 116 ・ FileDeleteLine の戻り値

戻り値	説明
0	FileDeleteLine がファイルから指定された行を削除しました。
< 0	FileDeleteLine は次の条件の 1 つが原因で失敗しました。 ・ FILE_NOT_FOUND (-2) – インストールは szFileName でファイルを検出できませんでした。 ・ FILE_RD_ONLY (-5) – ファイルは書き込み禁止となっています。 ・ LINE_NUMBER (-7) – 指定した行番号のひとつが 0 以下、または、行番号がファイルに存在しません。 ・ OUT_OF_DISK_SPACE (-6) – ディスクドライブには、指定した操作を完了するのに充分な空きスペースがありません。 ・ OTHER_FAILURE (-1) – その他の特定できないエラーが発生しました。

FileDeleteLine の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FileDeleteLine 関数のデモンストレーションを行います。
*
```

```
＊このスクリプトは、ファイル内で "PATH" 文字列を含む初めの行を
＊検索します。その文字列を含む行を検出した場合、
＊それを削除します。最後に、ファイル内の削除した行の位置に新しい行が
＊追加されます。"PATH" という単語を含む行が
＊検出されなかった場合には、ファイルのトップに
＊表示します。
＊
＊メモ : このスクリプトを実行する前に、ISExampl.bat と
＊　　名づけられたバッチ ファイルを、ドライブ C のルートに作成します。
＊　　最も効果的に行うためには、ファイルに PATH コマンド を含みます。
＊
¥*-------------------------------------------------------------*/

#define SDIR        "C:¥¥"
#define EXAMPLE_BAT  "ISExampl.bat"
#define TITLE       "FileDeleteLine の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_FileDeleteLine(HWND);

function ExFn_FileDeleteLine(hMSI)
  STRING szSearchStr, svReturnLine, szNewString, szMsg;
  NUMBER nvResult, nvLineNum;
begin

  // FileGrep の検索文字列パラメーターをセットアップします。
  szSearchStr = "PATH";

  // 指定したファイル内で検索文字列を探します。
  nvResult = FileGrep (SDIR ^ EXAMPLE_BAT, szSearchStr,
                svReturnLine, nvLineNum, RESTART);

  switch(nvResult)
    case FILE_NOT_FOUND:
      // エラーを報告し、中止します。
      MessageBox (EXAMPLE_BAT + " が見つかりませんでした。", WARNING);
      abort;
    case FILE_LINE_LENGTH:
      // エラーを報告し、中止します。
      MessageBox (EXAMPLE_BAT + " 行が長すぎます。", WARNING);
      abort;
    case OTHER_FAILURE:
      // エラーを報告し、中止します。
      MessageBox (EXAMPLE_BAT + "FileGrep への呼び出しで、不明エラーが発生しました。",
            WARNING);
      abort;
    case END_OF_FILE:
      // 検索文字列 が見つからなかったことを報告します。
      szMsg = "¥"%s¥" %s で見つかりませんでした。";
      SprintfBox (INFORMATION, TITLE, szMsg, szSearchStr, EXAMPLE_BAT);

      // FileInsertLine の行番号パラメーターを設定します。
      nvLineNum = 0;
    case 0:
      // 検索文字列を含む行を削除します。
      if (FileDeleteLine (EXAMPLE_BAT, nvLineNum, nvLineNum) < 0) then
        MessageBox ("FileDeleteLine への呼び出しに失敗しました。", SEVERE);
```

```
            abort;
        else
            // 削除されたことを報告します。
            szMsg = "¥"%s¥" %s の行 %d で見つかりました。:¥n¥n%s¥n¥n¥ 行が削除されました。";
            SprintfBox (INFORMATION, TITLE, szMsg, szSearchStr, nvLineNum,
                    EXAMPLE_BAT, svReturnLine);
        endif;
    endswitch;

    // FileInsertLine の新規文字列パラメーターをセットアップします。
    szNewString = "PATH=C:¥¥Windows¥¥Bin;C:¥¥Bin;C:¥¥Ishield;";

    // 新規文字列を挿入します。
    if (FileInsertLine (EXAMPLE_BAT, szNewString, nvLineNum, BEFORE) < 0) then
        // エラーを報告します。
        MessageBox ("FileInsertLine への呼び出しに失敗しました。", SEVERE);
    else
        // 成功を報告します。
        szMsg = " 次の文字列は行 %d / %s へ挿入されました :¥n¥n %s";
        SprintfBox (INFORMATION, TITLE, szMsg, nvLineNum, EXAMPLE_BAT,
                szNewString);
    endif;

end;
```

FileGrep

FileGrep 関数は、テキストファイル内で指定した文字列を検索します。文字列が検出された場合、その文字列を含む行は svReturnLine で戻され、nvLineNumber に行番号が戻されます。検索は大文字と小文字を区別しません。FileGrep はバイナリファイルではなく、行型テキストファイル上で動作します。

***メモ・**FileGrep を使って検索を始める前に、ファイルを開く必要はありません。また、FileGrep への呼び出しの後にファイルを閉じる必要もありません。ファイルの開閉は FileGrep が自動的に行います。FileGrep は OpenFile への前回の呼び出し結果として既に開かれているファイル上でも殆どの場合適切に動作しますが、ファイルが開かれている場合は、追加モードで FILE_NOT_FOUND を返します。*

構文

```
FileGrep ( szFileName, szSearchStr, svReturnLine, nvLineNumber, nFlag );
```

パラメーター

テーブル 117 ・ FileGrep のパラメーター

パラメーター	説明
szFileName	検索するファイルの完全修飾名を指定します。
szSearchStr	検索文字列を指定します。
svReturnLine	szSearchStr が検出された行を返します。
nvLineNumber	szSearchStr が検出された行番号を返します。行番号は 0 から始まります。
nFlag	検索オプションを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ CONTINUE—(それが存在する場合は) 次の検索文字列を読み出します。 ・ RESTART— 最初の検索文字列を読み出します。

戻り値

テーブル 118 ・ FileGrep の戻り値

戻り値	説明
0	FileGrep は指定した文字列を検出しました。
< 0	FileGrep は次の条件の 1 つが原因で失敗しました。 ・ END_OF_FILE (-4) — 検索文字列を検出せずにファイルの終わりに到達しました。 ・ FILE_NOT_FOUND (-2) — szFileName でファイルを検出できませんでした。 ・ FILE_LINE_LENGTH (-3) — 行が長さ制限を越えています。. ・ OTHER_FAILURE (-1) — 特定できないエラーが発生しました。

FileGrep の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FileGrep 関数のデモンストレーションを行います。
*
* FileGrep が呼び出され、ファイル内で "PATH" を含む最初の行を
* 検索します。結果はメッセージボックスに表示
```

```
＊されます。FileGrep 関数は大文字と小文字を区別しない点にご注意
＊下さい。
＊
＊メモ : このスクリプトを実行する前に、ISExampl.bat と
＊　　名づけられたバッチ ファイルを、ドライブ C のルートに作成します。
＊　　最も効果的に行うためには、ファイルに PATH コマンド を含みます。
＊
¥*------------------------------------------------------------*/

#define  SOURCE_DIR  "C:¥¥"
#define  SOURCE_FILE "ISExampl.bat"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_FileGrep(HWND);

function ExFn_FileGrep(hMSI)
   STRING  svLine, szNewString, svReturnLine, szMsg;
   NUMBER  nvLineNumber, nvResult;
begin

   // 指定したソースファイル内で検索文字列を探します。
   nvResult = FileGrep (SOURCE_DIR ^ SOURCE_FILE, "PATH", svReturnLine,
                        nvLineNumber, RESTART);

   switch(nvResult)
      case FILE_NOT_FOUND:
         // エラーを報告し、終了します。
         MessageBox( SOURCE_FILE + " が検出されませんでした。", WARNING);
         abort;
      case FILE_LINE_LENGTH:
         // エラーを報告し、終了します。
         MessageBox (SOURCE_FILE + " の行が長すぎます。.", WARNING);
         abort;
      case OTHER_FAILURE:
         // エラーを報告し、終了します。
         MessageBox (SOURCE_FILE + " FileGrep への呼び出しで不明なエラーが発生しました。",
                  WARNING);
         abort;
   endswitch;

   // ファイルの終わりまでループします。
   while (nvResult != END_OF_FILE)

      // SprintfBox 用のメッセージ文字列をセットアップします。
      szMsg = " 行 %d/%s で 'PATH' が検出されました :¥n¥n'%s'";

      // ファイルから一致する行を報告します。
      SprintfBox (INFORMATION, "FileGrep", szMsg, nvLineNumber, SOURCE_FILE,
               svReturnLine);

      // 再び検索します。
      nvResult = FileGrep (SOURCE_DIR ^ SOURCE_FILE, "PATH", svReturnLine,
                        nvLineNumber, CONTINUE);

   endwhile;

end;
```

FileInsertLine

FileInsertLine 関数は、行番号を利用して行を挿入または置換します。FileInsertLine と FileGrep を組み合わせて利用すると、行を検索して行番号を戻すことができます。

FileInsertLine は、1,024 バイト以下の行で構成される行型のテキストファイルで使用できます。InstallShield では、長さ制限を越えた行はバイナリファイルとして認識され、FileInsertLine が失敗する原因となります。

注意 *・szFileName が指定するファイルは OpenFile への呼び出しで既に開いた状態でなくてはなりません。szFileName が既に開いている場合、FileInsertLine の前に CloseFile を呼び出します。*

構文

FileInsertLine (szFileName, szInsertLine, nLineNumber, nInsertFlag);

パラメーター

テーブル 119 ・ FileInsertLine のパラメーター

パラメーター	説明
szFileName	システム変数 SRCEIR が指定するディレクトリ内に配置されたファイルの完全修飾されていない名前、またはファイルへの完全修飾パスを指定します。そのファイルが存在する場合、szInsertLine の値が挿入されます。
szInsertLine	ファイルへ挿入する文字列を指定します。
nLineNumber	szInsertLine を挿入する位置の行番号を指定します。最初の行番号は 0 です。
nInsertFlag	挿入場所オプションを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ BEFORE—nLineNumber の前に行を挿入します。 ・ AFTER — nLineNumber の後に行を挿入します。 ・ APPEND — nLineNumber が示した行へ szInsertLine を追加します。 ・ REPLACE — nLineNumber が示す行を szInsertLine に置換します。

戻り値

テーブル 120 ・ FileInsertLine の戻り値

戻り値	説明
0	FileInsertLine が指定したファイルへ行を挿入しました。
< 0	FileInsertLine は次の条件の 1 つが原因で失敗しました。 ・ FILE_LINE_LENGTH (-3) — 行の長さがテキストファイルで利用できる限界を超えることを示します。 ・ FILE_NOT_FOUND (-2) — FileInsertLine は、szFileName でファイルを検出できませんでした。 ・ FILE_RD_ONLY (-5) — ファイルが書き込み禁止であることを示します。 ・ LINE_NUMBER (-7) — 指定した行番号のひとつが 0 以下、または、行番号がファイルに存在しないことを示します。 ・ OUT_OF_DISK_SPACE (-6) — ディスクドライブには、指定した操作を完了するのに充分な空きスペースがないことを示します。 ・ OTHER_FAILURE (-1) — 特定できないエラーが発生しました。

FileInsertLine の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FileInsertLine 関数のデモンストレーションを行います。
*
* AskText ダイアログが表示され、ユーザーからテキスト行を
* 取得します。そして、このテキストは、TARGET_FILE が指定するテキストファイルに
* 最初の行として挿入されます。
*
* FileInsertLine が再び呼び出され、同じテキストを最初の行に追加
* します。ファイルの最初の行に同じテキストのコピー 2 つが
* 配置されます。
*
* メモ : このスクリプトを実行する前に、ISExampl.bat と名づけられた
*      バッチファイルを、ドライブ SRCDIR のルートに作成します。
*
¥*---------------------------------------------------------------*/

#define TARGET_FILE  "ISExampl.bat"
#define TITLE        "FileInsertLine の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_FileInsertLine(HWND);

function ExFn_FileInsertLine(hMSI)
   STRING szMsg, svText;
begin

   // AskText を呼び出すメッセージパラメーターをセットアップします。
   szMsg = " EXAMPLE.BAT へ挿入する行を入力してください。";

   // ファイルへ追加する行を取得します。
   AskText (szMsg, "", svText);

   // 指定したファイルの最初の行としてテキストを挿入します。
   if (FileInsertLine (TARGET_FILE, svText, 0, BEFORE) < 0) then
      MessageBox ("FileInsertLine が失敗しました。", SEVERE);
   else
      // SprintfBoxt を呼び出すメッセージパラメーターをセットアップします。
      szMsg   = "'%s' は %s の最初の行として挿入されました。";

      // メッセージを表示します。
      SprintfBox (INFORMATION, TITLE, szMsg, svText, TARGET_FILE);
   endif;

   // 同じ行へ同じ文字列を追加します。
   if (FileInsertLine(TARGET_FILE, svText, 0, APPEND) < 0) then
      MessageBox ("FileInsertLine が失敗しました。", SEVERE);
   else
```

```
        // SprintfBoxt を呼び出すメッセージパラメーターをセットアップします。
        szMsg   = "'%s' は %s の最初の行に正常に追加されました。";

        // メッセージを表示します。
        SprintfBox (INFORMATION, TITLE, szMsg, svText, TARGET_FILE);
    endif;

end;
```

FindAllDirs

FindAllDirs 関数は階層ディスクまたはディレクトリ構造全体を指定されたディレクトリから順に検索して、サブディレクトリ名の文字列リストを返します。FindAllDirs を利用して、特定のディレクトリのサブディレクトリを検索するか、あるいはディスク上のディレクトリ全体を検索することができます。

nOp が INCLUDE_SUBDIR の場合、検索は szDir で指定されたディレクトリから始まりサブディレクトリ構造へと続行されます。指定したディレクトリがルートディレクトリで、nOp に INCLUDE_SUBDIR が含まれる場合、ディスクすべてのディレクトリ名が戻されます。

構文

```
FindAllDirs ( szDir, nOp, listDirs );
```

パラメーター

テーブル 121 ・ FindAllDirs のパラメーター

パラメーター	説明
szDir	検索するディレクトリの名前を指定します。 ***メモ・** ディレクトリが引用符で括られていると、FindAllDirs は失敗します。フォルダー名が引用符で括られていないことを確認するには、FindAllDirs を呼び出す前に LongPathToQuote (szPath, FALSE) を呼び出します。*
nOp	szDir の下にあるサブディレクトリすべてを検索するか否か指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ EXCLUDE_SUBDIR ― サブディレクトリを除外します。 ・ INCLUDE_SUBDIR ― サブディレクトリを含めます。
listDirs	完全修飾ディレクトリ名のリストを返します。listDirs によって識別される文字列リストは、ListCreate への呼び出しによって既に初期化されている必要があります。

戻り値

テーブル 122 ・ FindAllDirs の戻り値

戻り値	説明
0	FindAllDirs が、サブディレクトリ名のリストを生成しました。
< 0	FindAllDirs 関数はリストを生成することができませんでした。

FindAllDirs の例

***メモ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* この例では FindAllDirs 関数のデモンストレーションを行います。
*
* FindAllDirs が呼び出し、指定したディレクトリに配置されたすべてのディレクトリを
* 読み出します。サブディレクトリはユーザーの判断で
* 含まれます。
*
¥*---------------------------------------------------------------------*/
```

```
#define TITLE "FindAllDirs の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_FindAllDirs(HWND);

function ExFn_FindAllDirs(hMSI)
   LIST   listDirs;
   STRING svSearchPath, szMsg;
   NUMBER nOp, nResult;
begin

   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   // ユーザーへパスを問い合わせます。
   AskPath (" 既存のパスを入力してください。.", "", svSearchPath);

   // サブディレクトリを含むかどうかを問い合わせます。
   if (AskYesNo (" サブディレクトリを含みますか ?", YES) = YES) then
      nOp    = INCLUDE_SUBDIR;
      szMsg = " ディレクトリとサブディレクトリ ";
   else
      nOp    = EXCLUDE_SUBDIR;
      szMsg = " ディレクトリのみ ";
   endif;

   // リストのビルド中にメッセージを表示します。
   SdShowMsg (" 検索中です .. お待ちください。", TRUE);

   // ディレクトリ名に STRING リストを作成します。
   listDirs = ListCreate (STRINGLIST);

   // 要求された要素を検出してリストに配置します。
   nResult = FindAllDirs (svSearchPath, nOp, listDirs);

   // メッセージボックスを閉じます。
   SdShowMsg("", FALSE);

   if ( nResult< 0) then
      // 一致しないことを報告します。
      SprintfBox (INFORMATION, TITLE, " %s にディレクトリはありません。",
                  svSearchPath);
   else
      // リストを表示します。
      SdShowInfoList (TITLE, szMsg, listDirs);
   endif;

end;
```

FindAllFiles

FindAllFiles 関数は、階層サブディレクトリ構造全体を指定したディレクトリから順に検索し、特定のファイル修飾名を持つ最初のファイルの名前を返します。指定したディレクトリがルートディレクトリの場合、InstallShield はディスク全体を検索します。関数は最初にファイル名が一致した場所で停止します。

メモ・*nOp に渡された引数が RESET の場合、InstallShield は szDir パラメーターで指定されたディレクトリを検索し、szFileName に合致するファイルを検出するまでサブディレクトリ構造の検索を続行します。nOp が CONTINUE に等しい場合、最後に関数が呼び出されたときに終了した場所から続行します。関数を繰り返し呼び出し、szFileName に一致するファイルすべてを検出します。*

新しい検索をする為にこの関数を初めて呼び出す場合は、nOp を RESET へ設定します。nOp CONTINUE に設定し、FindAllFiles 関数が失敗したときに終了するループに関数の呼び出しを配置することで、指定したファイル名の存在すべてを検索することができます。

ファイル修飾名と一致するファイルを検出する

タスク ***ファイル指定に一致するすべてのファイルを検出するには:***

1. 検索するフォルダー名を szDir に割り当てます。
2. 例えば S5*.txt など、ファイル修飾名を szFileName へ割り当てます。
3. nOp を RESET にして **FindAllFiles** を呼び出します。
4. **FindAllFiles** からの戻りコードが 0 のときに svResult でファイル名を保存してから、nOp を CONTINUE に設定して **FindAllFiles** を呼び出します。
5. nOp を CANCEL にして **FindAllFiles** を呼び出します。

注意・*FindAllFiles(..., RESET) や、FindAllFiles(..., CONTINUE) ループと共に XCopyFile 関数を利用することはできません。FindAllFiles ループの中で XCopyFile を呼び出す場合、FindAllFiles(..., CONTINUE) が戻すファイル名が間違っている場合もあります。*

構文

```
FindAllFiles ( szDir, szFileName, svResult, nOp );
```

パラメーター

テーブル 123 ・ FindAllFiles のパラメーター

パラメーター	説明
szDir	検索するディレクトリの名前を指定します。
szFileName	検索するファイル修飾名を指定します。このパラメーターではワイルドカード文字を利用することができます。
svResult	検出された場合、最初に一致したファイルの完全修飾名を返します。**FindAllFiles** が失敗した場合、svResult は変更されないままです。
nOp	検索オプションを示します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ CONTINUE ― 前回の検索が停止した場所から検索を再開します。 ・ RESET ― szDir が指定したディレクトリの最初から検索を開始します。 ・ CANCEL ― **FindAllFiles** への前回の呼び出しでアクセスしたファイルとフォルダーすべてを開放します。このパラメーターを使って **FindAllFiles** を呼び出して、**FindAllFiles** がアクセスを行っているファイルとフォルダー上で次のファイルとフォルダー操作が正しく行われるようにします。

戻り値

テーブル 124 ・ FindAllFiles の戻り値

戻り値	説明
0	関数が修飾名と一致するファイルが読み出し、返したことを示します。
< 0	関数が修飾名と一致するファイルを検出できなかったことを示します。

FindAllFiles の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FindAllFiles 関数のデモンストレーションを行います。
*
* このスクリプトは、ユーザーからディレクトリ名とファイルの修飾名を取得
*  します。そして FindAllFiles を繰り返し呼び出し、指定したディレクトリに配置され、
* 修飾名と一致するファイルのリストを
```

```
＊ビルドします。最後に、リストボックスに
＊一致したファイルの一覧が表示されます。
＊
¥*---------------------------------------------------------------*/

#define TITLE_TEXT "FindAllFiles の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_FindAllFiles(HWND);

function ExFn_FindAllFiles(hMSI)
   STRING szMsg, svDir, svFileSpec, svMatchingFileName, svNumFiles;
   NUMBER nResult, nNumFiles;
   LIST   listFiles;
begin

selectdir:

   // SelectDir を呼び出すパラメーターをセットアップします。
   szMsg = " 検索するディレクトリを選択します。";
   svDir = "";

   // 検索ディレクトリを選択します。
   SelectDir (TITLE_TEXT, szMsg, svDir, FALSE);

askfile:
   szMsg = "" + svDir + " で検索するファイル指定を入力します :";

   // ユーザーからファイル指定を取得します。
   if (AskText (szMsg , "*.*", svFileSpec) = BACK) then
      goto selectdir;
   endif;

   // ファイルリストの文字列リストを作成します。
   listFiles = ListCreate (STRINGLIST);

   if listFiles = LIST_NULL then
      MessageBox (" リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // ファイル カウントをゼロに設定します。
   nNumFiles = 0;

   // ファイルリストを作成中にメッセージを表示します。
   SdShowMsg (" 検索中です .. を参照してください。TRUE

   // ファイル指定と一致する最初のファイルを取得します。
   nResult = FindAllFiles (svDir, svFileSpec, svMatchingFileName, RESET);

   while(nResult = 0)
      // リストへファイルを追加します。
      if ListAddString (listFiles, svMatchingFileName, AFTER) < 0 then
         MessageBox (" 完全なファイルリストをビルドすることができませんでした ", WARNING);
         goto showmatches;
      endif;
```

```
        // ファイル カウンターを増加させます。
        nNumFiles = nNumFiles + 1;

        // 次の一致ファイル名を検索します。
        nResult = FindAllFiles(svDir, svFileSpec, svMatchingFileName, CONTINUE);
    endwhile;

showmatches:

    // FindAllFiles がアクセスするすべてのファイルとフォルダーを開放します。セットアップが
    // Windows NT プラットフォームをターゲットとしない場合、この手順は
    // 不要です。
    FindAllFiles(svDir, svFileSpec, svMatchingFileName, CANCEL);

    // ファイルカウントを表示用に文字列へ変換します。
    NumToStr(svNumFiles, nNumFiles);

    // ファイルリストがビルドされている最中に表示するメッセージをクリアします。
    SdShowMsg("", FALSE);

    // ファイル指定が一致するものを表示します。
    szMsg = " 一致するファイル数 : " + svNumFiles;
    if (SdShowInfoList(TITLE_TEXT, szMsg, listFiles) = BACK) then
        ListDestroy(listFiles);
        goto askfile;
    endif;

    // メモリからリストを削除します。
    ListDestroy(listFiles);

end;
```

FindFile

FindFile 関数は指定したファイルへのディレクトリを検索します。InstallShield は、パラメーター svResult 内の最初に一致するファイルを返します。

メモ・関数は指定したサブディレクトリのみを検索します。ディスクまたはディレクトリツリー全体の検索は行いません。

構文

```
FindFile ( szPath, szFileName, svResult );
```

パラメーター

テーブル 125 ・ FindFile のパラメーター

パラメーター	説明
szPath	検索するディレクトリの名前を指定します。このディレクトリの下にあるサブディレクトリは*検索されません*。 ***メモ ・** ディレクトリが引用符で括られていると、FindFile は失敗します。フォルダー名が引用符で括られていないことを確認するには、FindFile を呼び出す前に LongPathToQuote (szPath, FALSE) を呼び出します。*
szFileName	検索するするファイルの名前を指定します。このパラメーターではワイルドカード文字を利用することができます。
svResult	szFileName に一致する最初のファイルの名前を返します。このパラメーターは修飾されていないファイル名 (つまり、ドライブ指定とパスが含まれていない場合) を含みます。

戻り値

テーブル 126 ・ FindFile の戻り値

戻り値	説明
0	関数が指定したファイルを検出したことを示します。
< 0	関数がファイルを検出できなかったことを示します。

FindFile の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * FindFile 関数のデモンストレーションを行います。
 *
 * FindFile は C ドライブのルートにある Config.sys ファイルを検索するために
 * 呼び出されます。
 *
¥*---------------------------------------------------------------------*/

#define FILE_SPEC "Config.sys"
#define SEARCH_DIR "C:¥¥"
```

```
#define TITLE_TEXT "FindFile の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_FindFile(HWND);

function ExFn_FindFile(hMSI)
   STRING svResult;
begin

   if (FindFile (SEARCH_DIR, FILE_SPEC, svResult) < 0) then
      MessageBox ("FindFile が失敗しました。", SEVERE);
   else
      SprintfBox (INFORMATION, TITLE_TEXT, "%s が %s で見つかりました。", svResult,
                  SEARCH_DIR);
   endif;

end;
```

FindWindow

FindWindow 関数は上級開発者向けに、ウィンドウクラスとウィンドウ名を指定することでウィンドウへのハンドルを取得する方法を提供します。アプリケーションのクラスとウィンドウ名がわかれば、そのハンドルを取得することができます。そしてこのハンドルを使ってウィンドウへ直接メッセージを送ることができます。

ヒント・ウィンドウのクラスと名前を検出するには、Microsoft Spy.exe プログラムを実行してください。

構文

```
FindWindow ( szClassName, szWinName );
```

パラメーター

テーブル 127 ・ FindWindow のパラメーター

パラメーター	説明
szClassName	ウィンドウが属するクラスの名前を指定します。「任意のクラス」を指定するには、このパラメーターにヌル文字列を渡します。
szWinName	ウィンドウキャプション (タイトル) を指定します。指定されたクラスの最上位にあるウィンドウのハンドルを戻すには、このパラメーターでヌル文字列 (″″) を渡します。

戻り値

テーブル 128 ・ FindWindow の戻り値

戻り値	説明
XXXX	ウィンドウのハンドル
NULL (0)	FindWindow が指定した名前とクラス名を持つウィンドウを検出できませんでした。

FindWindow の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* FindWindow 関数と SendMessage 関数のデモンストレーションを行います。
*
* このスクリプトは Windows Notepad を起動し、
* FindWindow を呼び出して Notepad ウィンドウを検出します。次に、
* SendMessage を呼び出してウィンドウを最大化し、3 秒後に
* SendMessage を再び呼び出してウィンドウを最小化
* します。スクリプトが終了したとき、Windows NotePad は開いた状態ですが、
* 最小化されています。SendMessage へ渡されるパラメーターは
* Windows システム メッセージで、その値は
* スクリプトでは定数として定義されいることに注意してください。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*      定数 NOTEPAD を設定して、Windows Notepad 実行可能ファイルの完全
*    修飾名が参照されるようにしてください。
*
¥*---------------------------------------------------------------------*/
```

```
#define NOTEPAD "C:¥¥Windows¥¥Notepad.exe"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_FindWindow(HWND);

function ExFn_FindWindow(hMSI)
   NUMBER nMsg, nwParam, nlParam;
   HWND nHwnd;
begin

   // セットアップの背景ウィンドウを表示しません。
   Disable(BACKGROUND);

   // Windows Notepad を開きます。
   if (LaunchApp (NOTEPAD, "") < 0 ) then
      MessageBox ("Notepad を起動できませんでした。", SEVERE);
      abort;
   endif;

   // 3 秒間待機して、最大化される前にそのウィンドウを
   // 参照できるようにします。
   Delay (3);

   // Notepad ウィンドウのハンドルを読み出します。最初の
   // パラメーターはウィンドウクラスです。2 番目のパラメーターにある
   // ヌル文字列は一番最初の Notepad ウィンドウを指定します。
   nHwnd = FindWindow ("NotePAD", "");

   if (nHwnd = NULL) then
      MessageBox ("Notepad ウィンドウを検出できませんでした。", SEVERE);
   else
      // システム コマンドを送り、ウィンドウを最大化します。
      SendMessage (nHwnd, WM_SYSCOMMAND, SC_MAXIMIZE, 0);

      // 3 秒間待機して、最小化される前にそのウィンドウを
      // 参照できるようにします。
      Delay (3);

      // システム コマンドを送り、ウィンドウを最小化します。
      SendMessage (nHwnd, WM_SYSCOMMAND, SC_MINIMIZE, nlParam);
   endif;

end;
```

FormatMessage

FormatMessage 関数は、ビルトイン InstallShield 関数が戻す大きなエラーコードと関連付けられたエラーメッセージテキストを提供します。

構文

```
FormatMessage ( nErrorReturnCode );
```

パラメーター

テーブル 129 ・ FormatMessage のパラメーター

パラメーター	説明
nErrorReturnCode	大きなエラーコードを渡します。たとえば、2147024891 (0x80070005) は、ビルトイン InstallScript 関数によって戻された値です。エラーコード -1 を渡しても、役に立つ結果は得られません。

戻り値

エラーコード nErrorReturnCode に関連したエラーメッセージテキストを含む文字列。

FormatMessage の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
//---------------------------------------------------------------------------
//
// InstallShield スクリプトの例
//
//  FormatMessage 関数のデモンストレーションを行います。
//
//  デモンストレーションとして、意図的に存在しないソース ディレクトリで XCopyFile を
//  呼び出します。関数が失敗することが予想されます。エラーについての
//  システム情報を提供するには、FormatMessage を呼び出して
//  MessageBox にメッセージ文字列を表示します。
//
//---------------------------------------------------------------------------

function OnBegin()
    NUMBER nReturn;
begin

// 存在しないディレクトリ上の XCopyFile を呼び出します
nReturn = XCopyFile("C:¥¥no_such_directory", "C:¥¥destination", COMP_NORMAL);

// XCopyFile が失敗したときに、エラー メッセージを表示してインストーラーを終了します
if (nReturn < 0) then
    MessageBox(FormatMessage(nReturn), SEVERE);
    abort;
endif;

end;
```

GetAndAddAllFilesCost

GetAndAddAllFilesCost 関数は、szSrcDir に含まれていて szWildcard が識別するワイルドカードのパターンに一致するすべてのファイルのコストを判断し、このコストを nvCostHigh および / または nvCostLow の現在の値に追加します。その後このコストを **FeatureAddCost** に渡すことができます。

***メモ・GetAndAddAllFilesCost** は実際、インストールによって直接使用される情報の設定は行わないので注意してください。**FeatureAddCost** を（必要に応じて）この機能の呼び出しの後に呼び出して、既存する機能に追加コストを付け足す必要があります。*

構文

GetAndAddAllFilesCost (szSrcDir, szWildcard, szTargetDir, nClusterSize, nvInstallCostHigh, nvInstallCostLow, nvUninstallCost);

パラメーター

テーブル 130 ・ GetAndAddAllFilesCost のパラメーター

パラメーター	説明
szSrcDir	そのコストを判断するファイルを含むソースロケーションの完全修飾パス。
szWildcard	ファイルの検索に使用するワイルドカード。szWildcard と一致するファイルが見つからない場合、関数は失敗を返します。
szTargetDir	nClusterSize が 0 の場合、ファイルがインストールされる完全修飾パス。このパスは、ターゲットドライブのクラスタサイズを判断するときに利用されます。nClusterSize がゼロ以外の場合、このパラメーターは無視されます。
nClusterSize	ターゲットドライブのクラスタサイズを指定します。このパラメーターが 0 の場合、関数は szTargetDir からこの情報を判断します。
nvInstallCostHigh	このファイルのインストールコストの上位 ３１ ビット (バイト数) がこの変数の現在の値に追加されます。
nvInstallCostLow	このファイルのインストールコストの下位 ３１ ビット (バイト数) がこの変数の現在の値に追加されます。
nvUninstallCost	これらのファイルのアンインストールコストは、この変数の現在の値に追加されます。この値は、アンインストールされるファイルあたり 1 に等しくなります。したがって、アンインストールされる 100 のファイルは、値 100 を返します。

戻り値

テーブル 131 ・ GetAndAddAllFilesCost の戻り値

戻り値	説明
ISERR_SUCCESS	関数が成功したことを示します。
< ISERR_SUCCESS	関数の実行に失敗したことを示します。

GetAndAddFileCost

GetAndAddFileCost 関数は、指定されたファイルのコストを判断し、それを nvCostHigh および / または nvCostLow の現在の値に追加します。これにより、ループの中で関数を複数回呼び出して、複数のファイルのコストを計算および加算することができます。関数を呼び出してシングルファイルのコストを判別する前に、nvCostHigh および nvCostLow を ゼロに設定します。この関数は通常、ディスク上の既存のファイルを判別する必要があるときに使用され、コストを FeatureAddCost に渡すことができるようにします。

メモ・この関数は実際、インストールによって直接使用される情報の設定は行わないので注意してください。この関数を呼び出した後、必要に応じて FeatureAddCost を呼び出して、追加のコストを既存の機能に追加する必要があります。

構文

GetAndAddFileCost (szSrcFile, szTargetDir, nClusterSize, nvCostHigh, nvCostLow);

パラメーター

テーブル 132・GetAndAddFileCost のパラメーター

パラメーター	説明
szSrcFile	コストを判断する、ディスク上にある既存ファイルの完全修飾パスとファイル名。
szTargetDir	nClusterSize が 0 の場合、これはファイルのターゲットフォルダーになります。このパスは、ターゲットドライブのクラスタサイズを判断するときに利用されます。nClusterSize がゼロ以外の場合、このパラメーターは無視されます。
nClusterSize	ターゲットドライブのクラスタサイズを指定します。このパラメーターが 0 の場合、関数は szTargetDir からこの情報を判断します。
nvCostHigh	このファイルのインストール コストの上位 31 ビット (バイト数) がこの変数の現在の値に追加されます。
nvCostLow	このファイルのインストール コストの下位 31 ビット (バイト数) がこの変数の現在の値に追加されます。

戻り値

テーブル 133・GetAndAddFileCost の戻り値

戻り値	説明
ISERR_SUCCESS	関数が成功したことを示します。
< ISERR_SUCCESS	関数の実行に失敗したことを示します。

GetCArrayFromISArray

GetCArrayFromISArray 関数は、指定された配列の実際のデータをポイントするポインター配列にポインターを返します。この関数は、追加メモリを割り当てませんが、既存の配列にあるデータにポインターを返します。vArray が文字列配列の場合、戻されたポインターを LPCWSTR* または LPWSTR* 引数を受け取る関数に渡すことができます。vArray が数値配列の場合、戻されたポインターを LPCDWORD* または LPDWORD* 引数を受け取る関数に渡すことができます。

構文

GetCArrayFromISArray (vArray);

パラメーター

テーブル 134 ・ GetCArrayFromISArray のパラメーター

パラメーター	説明
vArray	ポインターが必要な配列を指定します。

戻り値

テーブル 135 ・ GetCArrayFromISArray の戻り値

戻り値	説明
pointer	指定された配列の実際のデータをポイントするポインター配列へのポインター。
< ISERR_SUCCESS	関数が失敗したことを示します。

追加情報

戻されたポインターを使って、その配列に含まれる文字列を変更する際には、注意が必要です。文字列配列に含まれる文字列の長さは、インストールによって内部的に管理されます。そのため、文字列の長さを変更した場合、インストールが文字列配列に含まれるデータを管理することはできません。

GetCHARArrayFromISStringArray

GetCHARArrayFromISStringArray 関数は、指定された配列に含まれる幅広い文字列に対応する ANSI 文字列へのポインター配列へポインターを返します。

構文

GetCHARArrayFromISStringArray (vArray);

パラメーター

テーブル 136 - GetCHARArrayFromISStringArray のパラメーター

パラメーター	説明
vArray	ポインターが必要な文字列配列を指定します。

戻り値

テーブル 137 - GetCHARArrayFromISStringArray の戻り値

戻り値	説明
pointer	ANSI 文字列へのポインターの配列へのポインター。
< ISERR_SUCCESS	関数が失敗したことを示します。

追加情報

GetCHARArrayFromISStringArray 関数はポインターの配列と ANSI 文字列用に追加メモリを割り当てます。このポインターは LPCSTR* または LPSTR* 引数を受け取る関数へ渡されます。新しく作成した配列での作業を完了した後、DeleteCHARArray を呼び出してメモリから配列を削除します。

CopyCHARArrayToISStringArray を呼び出してポインターの配列から元の文字列配列へデータを返した場合、配列に含まれる文字列を変更する際は注意してください。文字列配列に含まれる文字列の長さはインストールが内部的に管理するため、文字列の長さを変更すると **CopyCHARArrayToISStringArray** を呼び出した際に、文字列全体がオリジナル配列へコピーされません。

GetCurrentDialogName

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

GetCurrentDialogName 関数は、現在表示されているダイアログの名前を読み出します。これは、ダイアログが定義されたときに EzDefineDialog への呼び出しで指定された名前です。

この情報は EndDialog を呼び出してダイアログを閉じるのに利用できます。

構文

```
GetCurrentDialogName ( svDialogName );
```

パラメーター

テーブル 138 ・ GetCurrentDialogName のパラメーター

パラメーター	説明
svDialogName	現在表示されているダイアログの名前を、また現在ダイアログが表示されていない場合はヌル文字列 ("") を返します。

戻り値

テーブル 139 ・ GetCurrentDialogName の戻り値

戻り値	説明
>= ISERR_SUCCESS	GetCurrentDialogName は現在表示されているダイアログの名前を読み出します。
< ISERR_SUCCESS	現在表示中のダイアログはありません。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

GetCurrentDir

GetCurrentDir 関数は現在のディレクトリを返します。

メモ・スクリプトでファイルを指定する場合、適切な値を持つ現在のフォルダーに頼らず、常に (適切な InstallShield システム変数、例えば SRCDIR を使って) 完全パスを指定してください。スクリプトは現在のフォルダーを変更することが可能なコードを内部で実行するため、その値は必ずしも予期したものとは限りません。

構文

```
GetCurrentDir( svCurrentDir );
```

パラメーター

テーブル 140 ・ GetCurrentDir のパラメーター

パラメーター	説明
svCurrentDir	現在のディレクトリを返します。

戻り値

テーブル 141 ・ GetCurrentDir の戻り値

戻り値	説明
>= **ISERR_SUCCESS**	関数がカレントディレクトリを正常に取り出しだことを示します。
< **ISERR_SUCCESS**	関数がカレントディレクトリを読み出せなかったことを示します。

GetDir

GetDir の関数は、szPath で渡された完全修飾パスまたはファイル名からドライブ指定を削除し、残りのパス又はファイル名を svDir に返します。パスはドライブ指定を含まなくては成りません。UNC パスの場合もあります。

次の例では、完全修飾パス C:¥Windows が svDir で ¥Windows として戻されます。

```
GetDir("C:¥¥Windows", svDir);
```

次の例では、UNC パス ¥¥TheServer¥TheSharedDevice¥Programs が svDir で ¥Programs として戻されます。

```
GetDir("¥¥¥TheServer¥¥TheSharedDevice¥¥Programs", svDir);
```

構文

```
GetDir ( szPath, svDir );
```

パラメーター

テーブル 142 ・ GetDir のパラメーター

パラメーター	説明
szPath	ドライブ指定を含むパスを指定します。
svDir	ドライブ指定を除くパスを返します。szPath が UNC パスの場合、GetDir はサーバー名と共通デバイス名を除いたパスを返します。

戻り値

テーブル 143 ・ GetDir の戻り値

戻り値	説明
0	関数がドライブ指定を除くパスを返したことを示します。
< 0	関数がドライブ指定を除くパスを戻すことができなかったことを示します。

GetDir の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* GetDir 関数のデモンストレーションを行います。
*
* このスクリプトはユーザーから完全修飾ディレクトリ名を取得
*  します。次に、GetDir を呼び出してドライブ指定以外の
* 選択されたディレクトリ名を返します。そして、その結果のパスが表示されます。
* それを表示します。
*
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "GetDir example"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_GetDir(HWND);

function ExFn_GetDir(hMSI)
   STRING szMsg, svSelectedDir, svDirNameOnly;
begin

   // ユーザーから完全修飾ディレクトリ名を取得します。
```

```
    AskPath (" ディレクトリを選択します。", INSTALLDIR, svSelectedDir);

    // ドライブ指定を抜いたディレクトリ名を取得します。
    if (GetDir (svSelectedDir, svDirNameOnly) < 0) then
        // エラーを報告します。
        MessageBox ("GetDir が失敗しました。", SEVERE);
    else
        // ディレクトリ名を GetDir によって返されたとおりに表示します。
        SprintfBox (INFORMATION, TITLE_TEXT,
                    " ドライブ指定無しの選択したディレクトリ :: %s",
                    svDirNameOnly);
    endif;

end;
```

GetDisk

GetDisk 関数は、szPath が指定した完全修飾パスまたはファイル名からディスクドライブ指定を抽出し、svDisk へ返します。

構文

```
GetDisk ( szPath, svDisk );
```

パラメーター

テーブル 144 ・ GetDisk のパラメーター

パラメーター	説明
szPath	ドライブ指定を示す完全修飾パス、またはファイル名を指定します。ドライブ指定が含まれなかった場合、GetDisk は失敗します。szPath でパスされる値は UNC パスでなくてはなりません。
svDisk	(コロンを含む) ドライブ指定を返します。szPath が UNC パスの場合、GetDisk はサーバー名と共通デバイス名を **¥¥server¥sharedevice** の形式で返します。

戻り値

テーブル 145 ・ GetDisk の戻り値

戻り値	説明
0	関数がドライブ指定を返したことを示します。
< 0	関数がドライブ指定を戻さなかったことを示します。

GetDisk の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * GetDisk 関数のデモンストレーションを行います。
 *
 * このスクリプトは、ユーザーから完全修飾ディレクトリ名を
 *   取得します。次に、GetDisk を呼び出してディスクドライブ指定を
 * 返します。そして、ドライブ指定が表示されます。
 *
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "GetDisk  の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_GetDisk(HWND);

function ExFn_GetDisk(hMSI)
   STRING svSelectedDir, svDisk;
```

```
begin

  // ユーザーから完全修飾ディレクトリ名を取得します。
   AskPath (" ディレクトリを選択します。", INSTALLDIR, svSelectedDir);

   // 選択したディレクトリ名からドライブ指定を取得します。
   if (GetDisk (svSelectedDir, svDisk) < 0) then
      // エラーを報告します。
      MessageBox ("GetDir が失敗しました。", SEVERE);
   else
      // GetDisk が返したとおりにドライブ指定を表示します。
      SprintfBox (INFORMATION, TITLE_TEXT, " ディスク ドライブ : %s", svDisk);
   endif;

end;
```

GetDiskInfo

GetDiskInfo 関数は、指定されたディスクドライブについての情報を取得します。この変数のメンバーに割り当てられている値を調べると、スクリプトでディスクドライブについての情報を判断できます。

構文

```
GetDiskInfo ( _DISK_INFO& pdi );
```

パラメーター

テーブル 146 - GetDiskInfo のパラメーター

パラメーター	説明
_DISK_INFO& pdi	関数が戻った後、情報を指定して指定された情報を取得および格納する既存の _DISK_INFO (関数を呼び出す前にインストールによって割り当てられます) へのポインターを指定します。

以下のテーブルで、_DISK_INFO の各メンバーの意味が示されています。

テーブル 147 - _DISK_INFO メンバー

メンバー	意味
szDiskPath[_MAX_PATH]	情報を戻すドライブの完全修飾名を指定します。

テーブル 147 • _DISK_INFO メンバー (続き)

メンバー	意味
nInfoToQuery	クエリする情報を指定します。使用できる値は次のとおりです。 • DISK_INFO_QUERY_ALL – ディスクドライブについてのすべての情報をクエリします。 • DISK_INFO_QUERY_BYTES_PER_CLUSTER – ドライブのクラスタ (クラスタサイズ) あたりのバイト数をクエリします。 • DISK_INFO_QUERY_DISK_TOTAL_SPACE – ドライブ上の総空き容量をクエリします。 • DISK_INFO_QUERY_DISK_FREE_SPACE – ドライブ上の空き容量をクエリします。 • DISK_INFO_QUERY_DRIVE_TYPE – ドライブタイプをクエリします。.
nBytesPerCluster	DISK_INFO_QUERY_BYTES_PER_CLUSTER が c nInfoToQuery で指定されている場合、クラスタあたりのバイト数を返します。
nTotalSpaceHigh	ターゲットドライブの総領域の上位 31 ビットを返します。
nTotalSpaceLow	ターゲットドライブの総領域の下位 31 ビットを返します。
nFreeSpaceHigh	ターゲットドライブの空き領域の上位 31 ビットを返します。
nFreeSpaceLow	ターゲット ドライブの空き領域の下位 31 ビットを返します。
nDriveType	Windows API GetDriveType に戻された同じ定数を使用してドライブの種類を返します。便宜上、次の定数が予め定義されています。 • DRIVE_UNKNOWN – ドライブ タイプが不明です。 • DRIVE_NO_ROOT_DIR – ドライブ タイプがルート ディレクトリ以外として定義されています。 • DRIVE_REMOVABLE – ドライブ タイプがリムーバルとして定義されています。 • DRIVE_FIXED – ドライブ タイプが固定して定義されています。 • DRIVE_REMOTE – ドライブ タイプがリモートとして定義されています。 • DRIVE_CDROM – ドライブ タイプが CD-ROM として定義されています。 • DRIVE_RAMDISK – ドライブ タイプが RAM DISK として定義されています。

テーブル 147 ・ _DISK_INFO メンバー (続き)

メンバー	意味
nResultDiskSpace	次のフラグを指定すると、戻されたときにこのメンバーに有効の情報が含まれます。 ・ DISK_INFO_QUERY_BYTES_PER_CLUSTER ・ DISK_INFO_QUERY_DISK_TOTAL_SPACE ・ DISK_INFO_QUERY_DISK_FREE_SPACE このメンバーでは、情報をクエリした結果が戻されます。

戻り値

テーブル 148 ・ GetDiskInfo の戻り値

戻り値	説明
ISERR_SUCCESS	関数が成功したことを示します。
< ISERR_SUCCESS	関数が失敗したことを示します。この関数にヌル ポインターが渡されると、この値が戻されます。

nResultDiskSpace メンバーを確認して、クエリが成功したかどうかを判断できます。

GetDiskInfo の例

メモ・基本の MSI インストールでこの関数を呼び出すには、まずエントリー ポイント関数用のカスタム アクションを作成し、シーケンスで、またはダイアログのコントロール イベントの結果としてカスタム アクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
*  GetDiskInfo 関数のデモンストレーションを行います。
*
* このスクリプトは、Windows ドライブ上のディスク空き容量を取得し、
* メッセージ ボックスに表示します。
*
¥*---------------------------------------------------------------------*/

function OnBegin()
    _DISK_INFO di;
    NUMBER n; // 必須項目 : 関数の戻り値に関する事柄
    NUMBER nvSizeTargetHigh, nvSizeTargetLow, nUnitsTarget;
begin

// init._DISK_INFO members: what drive, what info
di.szDiskPath = WINDISK;
```

```
di.nInfoToQuery = DISK_INFO_QUERY_DISK_FREE_SPACE;

n = GetDiskInfo(&di);

// 空き容量について適切なユニットへ変更
nUnitsTarget = MBYTES;

n = ConvertSizeToUnits(
    di.nFreeSpaceHigh, di.nFreeSpaceLow, BYTES,
    nvSizeTargetHigh, nvSizeTargetLow, nUnitsTarget);

SprintfBox(INFORMATION, "Free Space",
    " 空き容量 : %d MB",
    nvSizeTargetLow);

end;
```

GetDiskSpace

この関数は現在使用されていません。代わりに、GetDiskInfo 関数を使用してください。

GetDiskSpace 関数は szPath で指定したドライブまたはパス上の空き容量をバイト単位で返します。

構文

```
GetDiskSpace(szDrive);
```

パラメーター

テーブル 149 ・ GetDiskSpace のパラメーター

パラメーター	説明
szPath	関数がその利用可能な空き容量を戻すパスを指定します。指定する値は UNC パスが可能です。

戻り値

テーブル 150 ・ GetDiskSpace の戻り値

戻り値	説明
> 0	指定したディレクトリの空き容量のバイト数。戻される最大値は 2 GB です。これより大きいディスク容量の場合も 2 GB が戻されます。セットアップが 2 GB よりも大きい空き容量を確認する必要がある場合、GetDiskSpaceEx を呼び出して下さい。
< 0	GetDiskSpace が空き容量を取得できなかったことを示します。

GetDiskSpace の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* GetDiskSpace 関数のデモンストレーションを行います。
*
* このスクリプトはエンド ユーザーから完全修飾名を取得します。
* そしてパスからドライブ指定を抽出し、
* そのドライブの空き容量を取得してメッセージボックスに
* 空き容量を表示します。
*
* 必要に応じて数値が表示される前に
* スクリプト定義の関数を利用してコンマを挿入します。
*
¥*---------------------------------------------------------------------*/


    // 文字列内の数値をフォーマットするユーザー定義関数。
    prototype FormatIntString (BYREF STRING);

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_GetDiskSpace(HWND);
```

```
function ExFn_GetDiskSpace(hMSI)
    STRING svResultPath, svDrive;
    LONG   lFreeSpace;
    STRING svFreeSpace;
begin

    // ターゲットパスを要求します。デフォルトでは C を利用します。
    AskPath ("ドライブの選択:", "C:¥¥", svResultPath);

    // パスからドライブ指定を取得します。
    GetDisk (svResultPath, svDrive);

    // そのドライブ上のディスク空き容量を取得します。
    lFreeSpace = GetDiskSpace (svDrive);

    if (lFreeSpace < 0) then
        // GetDiskSpace からのエラーを処理します。
        MessageBox ("GetDiskSpace が失敗しました。", SEVERE);
    else
        // ディスク空き容量を文字列へ変換します。
        NumToStr (svFreeSpace, lFreeSpace);

        // 必要に応じて数値にコンマを挿入します。
        FormatIntString (svFreeSpace) ;

        // 空き容量を報告します。
        MessageBox (svFreeSpace + " の空き容量が次のドライブにあります:" + svDrive + ".",
                INFORMATION);
    endif;

end;


function FormatIntString(svInteger)
// 必要であれば
// 文字列変数に保存されている正数にコンマを挿入します。
  INT nLen;
  STRING svSubStr, svTemp;
begin
  nLen = StrLength (svInteger);

  if nLen > 3 then
    nLen = nLen - 3;
    StrSub (svTemp , svInteger, nLen, 3);

    while nLen > 3
      nLen = nLen - 3;
      StrSub (svSubStr, svInteger, nLen, 3);
      svTemp = svSubStr + "," + svTemp;
    endwhile;

    StrSub (svSubStr, svInteger, 0,nLen);
    svInteger = svSubStr + "," + svTemp;
  endif;
end;
```

GetDiskSpaceEx

この関数は現在使用されていません。代わりに、GetDiskInfo 関数を使用してください。

GetDiskSpaceEx 関数は指定されたパス上の空き容量を返します。nUnits にパスされる値により GetDiskSpaceEx が返した値がバイト、キロバイト、メガバイト、またはギガバイト単位のどれであるかを判断します。

構文

GetDiskSpaceEx (szDrive, nUnits);

パラメーター

テーブル 151 ・ GetDiskSpaceEx のパラメーター

パラメーター	説明
szPath	関数がその利用可能な空き容量を戻すパスを指定します。指定する値は UNC パスが可能です。
nUnits	測定単位を示すために、以下のあらかじめ定義された定数のうち 1 つを渡します。 ・ BYTES—GetDiskSpaceEx が空きバイトの値を戻すことを示します。 ・ KBYTES—GetDiskSpaceEx が空きバイトの値をキロバイトで戻すことを示します。 ・ MBYTES—GetDiskSpaceEx が空きバイトの値をメガバイトで戻すことを示します。 ・ GBYTES — GetDiskSpaceEx が空きバイトの値をギガバイトで戻すことを示します。

戻り値

テーブル 152 ・ GetDiskSpaceEx の戻り値

戻り値	説明
> 0	指定したディレクトリの空きバイト、空きキロバイト、空きメガバイト、空きギガバイトの値は、nUnits の値によって異なります。
< 0	GetDiskSpaceEx が空き容量を取得できなかったことを示します。

追加情報

GetDiskSpaceEx の制限は、nUnits で渡す測定単位の 2 ギガ (2 31) 単位です。BYTES を指定する場合の制限は 2 GB、KBYTES を指定する場合の制限は 2 TB といった要領です。ほとんどのセットアップには KBYTES を指定してください。

GetDiskSpaceEx は、nUnits で指定した単一ユニットに最も近い値で有効スペースを返します。たとえば、GBYTES を指定する場合、利用できる 2.5 GB があるパスは 2 を返します。より正確な空き容量情報が必要なセットアップの場合、Windows API 関数 GetDiskFreeSpaceEx を直接呼び出します。

GetDiskSpaceEx の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* GetDiskSpaceEx 関数のデモンストレーションを行います。
*
* このスクリプトはエンド ユーザーから完全修飾名を取得します。
* そしてパスからドライブ指定を抽出し、
* そのドライブの空き容量を取得してメッセージボックスに
* 空き容量を表示します。
*
¥*---------------------------------------------------------------*/

STRING   szPath;

  INT     iResult;



#include "ifx.h"

program

  // ターゲットパスを要求します。

   szPath = PROGRAMFILES;
   iResult = SdAskDestPath
   (" フォルダーの選択 ¥n 有効なディスクの空き容量を確認するフォルダーを選択します。.", " ", szPath, 0);

   if (iResult < 0) then
      MessageBox(" ダイアログを表示できません。", SEVERE);
      endif;

  // 指定したパスのディスク空き容量を ( キロバイト単位で ) 取得します。

   iResult = GetDiskSpaceEx(szPath, KBYTES);

   if (iResult < 0) then
      MessageBox(" 有効なディスク容量を取得できません。", SEVERE);
      endif;

   // 有効な空き容量を表示します。

      SprintfBox(INFORMATION, " 有効な空き容量 ", "%s で %d KB 有効です。", iResult, szPath);

endprogram
```

GetEnvVar

GetEnvVar 関数は、環境変数の現在の値を読み出します。

構文

GetEnvVar (szParameter、svValue);

パラメーター

テーブル 153 - GetEnvVar のパラメーター

パラメーター	説明
szParameter	値を取得する環境変数名を指定します。
svValue	環境変数の現在の値を返します。

戻り値

テーブル 154 - GetEnvVar の戻り値

戻り値	説明
0	関数が環境変数の値を取得したことを示します。
< 0	関数が環境変数の値を取得できなかったことを示します。

GetEnvVar の例

このスクリプトは基本の MSI インストール用に構成されています。基本の MSI インストールでこの関数を呼び出すには、まずエントリー ポイント関数用のカスタム アクションを作成し、シーケンスで、またはダイアログのコントロール イベントの結果としてカスタム アクションを実行してから、リリースをビルドします。このコードを InstallScript または InstallScript MSI インストールで実行するには、begin ステートメント及び end ステートメントの間にあるコードのみをコピーして、スクリプトの OnBegin イベントハンドラー関数内に貼り付けます。

***メモ -** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* GetEnvVar 関数のデモンストレーションを行います。
*
* GetEnvVar は TEMP と呼ばれる環境変数の値を取得するために
* 呼び出されます。
*
¥*---------------------------------------------------------------*/
```

```
#define TITLE_TEXT "GetEnvVar の例 "

#define ENV_TEMP "TEMP"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_GetEnvVar(HWND);

function ExFn_GetEnvVar(hMSI)
   STRING svEnvVar;
begin

   // ENV_TEMP が指定した環境変数の値を取得します。
   if (GetEnvVar (ENV_TEMP, svEnvVar) < 0) then
      // エラーを報告します。
      MessageBox ("環境変数 " + ENV_TEMP + " を検出できませんでした。.", SEVERE);
   else
      // 環境変数の値を表示します。
      SprintfBox (INFORMATION, TITLE_TEXT, "%s = %s", ENV_TEMP, svEnvVar);
   endif;

end;
```

GetExtendedErrInfo

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

GetExtendedErrInfo 関数は、SetExtendedErrInfo が設定するエラー情報を返します。多くの InstallScript 関数は SetExtendedErrInfo(_FILE_、_LINE_、nError) を内部で呼び出します。

構文

```
GetExtendedErrInfo ( svScriptFile, nvLineNumber, nvError );
```

パラメーター

テーブル 155 ・ GetExtendedErrInfo のパラメーター

パラメーター	説明
svScriptFile	エラーが発生したスクリプトファイルを返します。
nvLineNumber	エラーが発生した行番号を返します。
nvError	エラーコードを返します。

戻り値

テーブル 156 ・ GetExtendedErrInfo の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がエラー情報を読み出しました。
< ISERR_SUCCESS	関数がエラー情報の読み出しに失敗しました。

GetExtents

GetExtents 関数は、画面の寸法をピクセル単位で読み出します。画面の幅は nvDx に、そして高さは nvDy に戻されます。例えば、標準 VGA モニターの場合は nvDx には 640 を、nvDy には 480 を返します。

***メモ・** ターゲットマシンの画面の寸法は、ScreenX と ScreenY プロパティを利用することで本来 Windows Installer サービスを通して取得することができます。*

構文

```
GetExtents ( nvDx, nvDy );
```

パラメーター

テーブル 157 ・ GetExtents のパラメーター

パラメーター	説明
nvDx	画面の幅をピクセル単位で返します。
nvDy	画面の高さをピクセル単位で返します。

戻り値

テーブル 158 ・ GetExtents の戻り値

戻り値	説明
0	関数が画面の寸法を読み出したことを示します。
< 0	関数が値を読み出せなかったことを示します。

GetExtents の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* GetExtents 関数のデモンストレーションを行います。
*
* このスクリプトは GetExtents を呼び出してターゲットシステムの
* 現在のビデオ解像度を取得します。そしてダイアログ ボックスに
* 情報を表示します。
*
¥*---------------------------------------------------------------*/

#define TITLE_TEXT "GetExtents の例 "
#define MSG_TEXT " ビデオ解像度 : %d x %d"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_GetExtents(HWND);

function ExFn_GetExtents(hMSI)
   NUMBER  nvDx, nvDy;
begin

   // ターゲットシステムのビデオ解像度を取得します。
   if (GetExtents (nvDx, nvDy) < 0) then
```

```
        // エラーを報告します。
        MessageBox ("GetExtents が失敗しました。", SEVERE);
    else
        // 情報を表示します
        SprintfBox (INFORMATION, TITLE_TEXT, MSG_TEXT, nvDx, nvDy);
    endif;

end;
```

GetFileInfo

GetFileInfo 関数を呼び出し、ファイルまたはディレクトリの属性、変更日、時刻、MD5 シグネチャ、または、サイズを判別します。各 GetFileInfo ステートメントで、データを 1 つのみ要求できます。たとえば、ファイルまたはディレクトリの日付情報と時刻情報を取得するには、GetFileInfo を日付を取得するために 1 度、そして時刻を取得するためにもう 1 度と、合計 2 度呼び出す必要があります。

構文

```
GetFileInfo (szPathName, nType, nvResult, svResult);
```

パラメーター

テーブル 159 ・ GetFileInfo のパラメーター

パラメーター	説明
szPathName	情報を取得するファイルまたはディレクトリの完全パスを指定します。nType パラメーター として FILE_ATTRIBUTE または FILE_SHARED_COUNT を渡さない限り、このパラメーターで有効な URL (Uniform Resource Locator) を指定します。この場合、関数が失敗して ISERR_INVALID_ARG が戻ります。URL が有効かどうかを確認するには、Is(VALID_PATH, szURL) を呼び出します。
nType	取得するファイルまたはディレクトリ情報の種類を指定します。情報が数値の場合、GetFileInfo によって nvResult に配置されます。情報が文字列の場合、GetFileInfo が svResult に配置します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ FILE_ATTRIBUTE— ファイルまたはディレクトリの属性が nvResult に戻されます。 ・ FILE_DATE— ファイルまたはディレクトリの変更日は、YYYY¥MM¥DD (年 ¥ 月 ¥ 日) 形式で svResult に戻されます。 ・ FILE_MD5_SIGNATURE— s z File で指定されたファイルの MD5 シグネチャを返します。MD5 シグネチャを生成および戻す操作は、ファイル全体の中身を読む必要があるので高いコストがかかります。このパラメーターは、絶対に必要な場合のみ、使用してください。FILE_MD5_SIGNATURE は、URL にはサポートされていません。  ***メモ・** 特定のファイルに対する加工前の MD5 シグネチャ データは、それぞれが 16 ビット (0x00 および 0xFF の間) で生成された 16 の数値で構成されます。これらの値は通常、unsigned char 型の文字列に格納されます。ただし、InstallScript 言語は unsigned char 型をサポートしないため、加工前の MD5 ファイルデータはそのまま戻らず、各 16 数値は対応する文字列に変換され結果文字列に配置されます。このため、32 文字からなる文字列は 2 文字ずつの各セットが数値 1 つを意味します。これは、MD5 16 進法文字列として呼ばれることもあります。* ・ FILE_SHARED_COUNT — ファイルの参照カウント。 ・ FILE_SIZE — サイズは、バイト数で nvResult に戻されます (FILE_SIZE_LOW と同様)。 ・ FILE_SIZE_LOW— ファイルのサイズの下位 31 ビット (バイト数) が nvResult に戻されます。ファイルのサイズが 2GB より大きい場合、ファイルのサイズは、FILE_SIZE_HIGH に戻された値を 2GB で掛けて、FILE_SIZE_HIGH に戻された値に加えた値に等しくなります。 ・ FILE_SIZE_HIGH— ファイルのサイズの上位 31 ビット (バイト数) が nvResult に戻されます。ファイルが 2 GB より小さい場合、この値は 0 (ゼロ) です。そうでない場合、ファイルのサイズは、FILE_SIZE_HIGH に戻された値を 2GB で掛けて、FILE_SIZE_HIGH に戻された値に加えた値に等しくなります。 ・ FILE_TIME— ファイルまたはディレクトリの変更時刻は、HH:MM:SS (時 : 分 : 秒) 形式で svResult に戻されます。

テーブル 159 ・ GetFileInfo のパラメーター (続き)

パラメーター	説明
	 ***メモ ・** FILE_ATTRIBUTE を 2 番目のパラメーター (nType) として使用して GetFileInfo を呼び出した後、if-then-else 論理を使用してファイルまたはディレクトリの属性を判別します。nvResult = FILE_ATTR_NORMAL の場合、属性は設定されていません。nvResult != FILE_ATTR_NORMAL の場合、nvResult でのビットワイズ AND (&) 演算子の結果、および以下の 1 つ以上のファイル属性定数をテストし、設定されている属性を判別します。* • *FILE_ATTR_NORMAL: ファイルは標準ファイルです。* • *FILE_ATTR_ARCHIVED: ファイルはアーカイブされています。* • *FILE_ATTR_DIRECTORY: ファイルはディレクトリです。* • *FILE_ATTR_HIDDEN: ファイルは隠しファイルです。* • *FILE_ATTR_READONLY: ファイルは読み取り専用です。* • *FILE_ATTR_SYSTEM: ファイルはシステム ファイルです。.* `if (nvResult = FILE_ATTR_NORMAL) then` `// ファイルは、NORMAL です。` `else` `if (FILE_ATTR_HIDDEN & nvResult) then` `// ファイルは、HIDDEN です。` `endif;` `if (FILE_ATTR_READONLY & nvResult) then` `// ファイルは、READ-ONLY です。` `endif;` `endif;`
nvResult	数値を返します。
svResult	文字列情報を返します。

戻り値

テーブル 160 ・ GetFileInfo の戻り値

戻り値	説明
0	関数を使用して要求したファイル情報が取得されたことを示します。
ISERR_INVALID_ARG (-2)	無効な引数が関数へ渡されたことを示します。
その他すべての負の値	関数を使用して要求した情報が取得できなかったことを示します。

GetFileInfo の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* GetFileInfo 関数のデモンストレーションを行います。
*
* GetFileInfo が呼び出されて、ファイルの日付、時刻、および
* 属性を読み出します。
*
* メモ : このスクリプトを適切に実行するため、
*    定数 EXAMPLE_TXT をターゲットシステム上に既存する
*     ファイル名に設定しなくてはなりません。
*
¥*---------------------------------------------------------------*/

#define EXAMPLE_FILE "C:¥¥IO.sys"
#define TITLE_TEXT   "GetFileInfo の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_GetFileInfo(HWND);

function ExFn_GetFileInfo(hMSI)
    STRING svResult, szAttributes;
    NUMBER nvResult;
    LIST   listID;
begin

    // ファイルについての情報を格納するリストを作成します。
    listID = ListCreate (STRINGLIST);

    // ファイルが作成された日付、または最後に更新された日付を svResult へ取得します。
    if (GetFileInfo (EXAMPLE_FILE, FILE_DATE, nvResult, svResult) = 0) then
        // リストへファイルの日付を追加します。
```

```
        ListAddString (listID, " ファイルの日付 : " + svResult, AFTER);
    endif;

    // ファイルが作成された時刻、または最後に更新された時刻を svResult へ取得します。
    if (GetFileInfo (EXAMPLE_FILE, FILE_TIME, nvResult, svResult) = 0) then
        // リストへファイルの時刻を追加します。
        ListAddString (listID, " ファイルの時刻 : " + svResult, AFTER);
    endif;

    // ファイルの属性を nvResult へ取得します。
    if (GetFileInfo (EXAMPLE_FILE, FILE_ATTRIBUTE, nvResult, svResult) = 0) then
      // 属性無しテスト。
      if (nvResult = FILE_ATTR_NORMAL) then
          // 属性は設定されていません。この情報をリストへ追加します。
          ListAddString(listID, " ファイル属性 : 標準 ", AFTER);
      else
          // この文字列へ属性を追加します。
          szAttributes = " ファイル属性 : ";

          // アーカイブされていますか？
          if (FILE_ATTR_ARCHIVED & nvResult) then
              szAttributes = szAttributes + " アーカイブ済み ";
          endif;

          // 隠しファイルですか？
          if (FILE_ATTR_HIDDEN & nvResult) then
              szAttributes = szAttributes + " 隠しファイル ";
          endif;

          // 読み取り専用ですか？
          if (FILE_ATTR_READONLY & nvResult) then
              szAttributes = szAttributes + " 読み取り専用 ";
          endif;

          // システムファイルですか？
          if (FILE_ATTR_SYSTEM & nvResult) then
              szAttributes = szAttributes + " システムファイル ";
          endif;

          // ディレクトリですか？
          if (FILE_ATTR_DIRECTORY & nvResult) then
              szAttributes = szAttributes + " ディレクトリ ";
          endif;

        endif;

        // リストへファイル属性を追加します。
        ListAddString (listID, szAttributes, AFTER);

    endif;

    // リストを表示します。
    SdShowInfoList (TITLE_TEXT, EXAMPLE_FILE, listID);

    // メモリからリストを削除します。
    ListDestroy(listID);

end;
```

GetFolderNameList

GetFolderNameList 関数は、指定したフォルダー内のすべてのプログラム項目ショートカットおよびサブフォルダーを列挙するのに利用します。この関数は、ルートフォルダー内のプログラム項目ショートカットおよびサブフォルダーの列挙にも利用できます。

構文

GetFolderNameList (szFolderName, listItemsID, listSubFoldersID);

パラメーター

テーブル 161 ・ GetFolderNameList のパラメーター

パラメーター	説明
szFolderName	照会するフォルダーの名前を指定します。szFolderName の完全修飾パスを次の様に指定することができます : **"C:¥¥Windows¥¥Start Menu¥¥Programs¥¥Accessories¥¥Games"** szFolderName がヌルの場合、**GetFolderNameList** はデフォルトのプログラム ディレクトリを検索します。szFolderName に絶対パス (ドライブ名を含むパス。例、"C:¥¥Program Files¥¥AppName") を指定しなかった場合、**GetFolderNameList** はデフォルトの Program ディレクトリの下にあるサブフォルダを検索します。この場所は InstallScript 変数 ALLUSERS の値、およびターゲット システム上の Windows バージョンによって異なります。 InstallScript システム変数を使用することもできます。 ・ FOLDER_DESKTOP ― [デスクトップ] フォルダーを検索します。 ・ FOLDER_STARTUP ― [スタートアップ] メニューを検索します。 ・ FOLDER_STARTMENU ― [スタート] メニューを検索します。 ・ FOLDER_PROGRAMS― スタート メニューの [プログラム] フォルダーを検索します。 または、次のような関連パスを使用できます。 FOLDER_PROGRAMS ^ "ACCESSORIES¥¥GAMES"
listItemsID	szFolderName 内のプログラム項目のショートカットの名前を表示したリストを返します。szFolderName が個人および共有プログラム項目の両方を含んでいる場合、このパラメーターで戻されたリストには共通または個人プログラム項目のショートカットのどちらかが含まれますが、両方は含まれません。listItemsID によって識別されるリストは、**ListCreate** への呼び出しによって既に初期化されている必要があります。
listSubFoldersID	szFolderName 内のサブフォルダーの名前を表示したリストを返します。szFolderName が個人および共用フォルダーの両方を含んでいる場合、このパラメーターで戻されたリストには共通または個人フォルダーのどちらかが含まれますが、両方は含まれません。listSubFoldersID によって識別されるリストは、**ListCreate** への呼び出しによって既に初期化されている必要があります。

戻り値

テーブル 162 ・ GetFolderNameList の戻り値

戻り値	説明
0	**GetFolderNameList** はすべてのプログラム項目およびサブフォルダー名を正常に取得しました。

テーブル 162 ・ GetFolderNameList の戻り値 (続き)

戻り値	説明
< 0	**GetFolderNameList** はプログラム項目およびサブフォルダー名を取得できませんでした。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

GetFolderNameList の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* GetFolderNameList 関数のデモンストレーションを行います。
*
* GetFolderNameList が呼び出され、デスクトップ上のフォルダーとプログラム項目すべてが
* 読み出されます。
*
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_GetFolderNameList(HWND);

function ExFn_GetFolderNameList(hMSI)
   NUMBER  nResult;
   LIST    listItemsID, listFoldersID;
begin

   // フォルダーとポログラム名のリストを作成します。
   listItemsID = ListCreate (STRINGLIST);
   listFoldersID = ListCreate (STRINGLIST);

   if (listItemsID = LIST_NULL) || (listFoldersID = LIST_NULL) then
      MessageBox ("リストを作成できませんでした。", SEVERE);
   else
      // リストのビルド中にメッセージボックスを表示します。
      SdShowMsg ("検索中です .. お待ちください。", TRUE);

      // フォルダーとプログラム名をリストへ配置します。
      nResult = GetFolderNameList (FOLDER_DESKTOP, listItemsID, listFoldersID);

      // メッセージボックスを閉じます。
      SdShowMsg("", FALSE);
```

```
    if (nResult < 0) then
        MessageBox (" デスクトップ フォルダーとプログラム名を読み出すことができません。",
                SEVERE);
    else

        // リストを表示します。
        repeat

            // [ 戻る ] ボタンを無効にします。
            Disable(BACKBUTTON);

            // 項目リストを表示します。
            nResult = SdShowInfoList ("", " 項目リスト :", listItemsID);

            // [ 戻る ] ボタンを有効にします。
            Enable(BACKBUTTON);

            // フォルダーリストを表示します。
            nResult = SdShowInfoList ("", " フォルダーリスト :", listFoldersID);

        until (nResult = NEXT);

    endif;
  endif;

end;
```

GetFont

GetFont 関数は、フォントをビルドしてそのハンドルを読み出します。フォントハンドルを利用してカスタムダイアログのコントロールで使用するフォントを指定できます。

***メモ・** InstallShield はインストールが終了したとき、この関数を使って作成されたすべてのフォントを削除します。さらに、InstallShield は終了と同時にすべてのシステムリソースをリリースします。*

構文

```
GetFont ( szFontName, nPointSize, nAttributes );
```

パラメーター

テーブル 163 · GetFont のパラメーター

パラメーター	説明
szFontName	ビルドするフォントの名前を指定します。
nPointSize	ビルドするフォントのポイント サイズを指定します。
nAttributes	フォントスタイルを指定します。このパラメーターには、次の定義済み定数のいずれかを指定します。定数とビット単位 OR 演算子 (|) を組み合わせて複数のスタイルを指定します。 · STYLE_BOLD ― 太字スタイルフォントを指定します。 · STYLE_ITALIC ― イタリック体にするフォントを指定します。 · STYLE_NORMAL ― 通常スタイルフォントを指定します。 · STYLE_UNDERLINE ― 下線を引く文字を指定します。

戻り値

テーブル 164 · GetFont の戻り値

戻り値	説明
XXXX	フォントへのハンドル。

GetFont が szFontName で名づけられたフォントを検出できなかった場合、関数は Arial フォントをビルドし、フォントへハンドルを返します。

GetFont の例

***メモ ·** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* GetFont 関数と CtrlSetFon 関数のデモンストレーションを行います。
*
* このスクリプト例では、GetFont を呼び出して 4 つのフォントのハンドルを
* 読み出します。これらのハンドルは CtrlSetFont へ渡され、
* カスタムダイアログボックスの静的テキストフィールドのフォントが設定
* されます。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SetupType が表示する
```

```
    * InstallShield ダイアログです。このダイアログは
    * インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
    * カスタム ダイアログとしてスクリプトで利用することが
    * 表示されます。
    *
    ¥*---------------------------------------------------------*/

    // ダイアログ ID とコントロール ID。
    #define RES_DIALOG_ID       10203   // カスタム ダイアログの ID
    #define RES_PBUT_NEXT         1   // [ 次へ ] ボタンの ID
    #define RES_PBUT_CANCEL       9   // [ キャンセル ] ボタンの ID
    #define RES_TEXT_1          202   // 最初のスタティック テキストボックスの ID

    #define RES_TEXT_2          210   // 2 番目のスタティックテキストボックスの ID
    #define RES_TEXT_3          220   // 3 番目のスタティックテキストボックスの ID
    #define RES_TEXT_4          230   // 4 番目のスタティック テキストボックスの ID

    // ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
    #include "Ifx.h"

        export prototype ExFn_GetFont(HWND);

    function ExFn_GetFont(hMSI)
        STRING szDialogName;
        NUMBER nResult, nCmdValue;
        HWND   hFont1, hFont2, hFont3, hFont4, hwndDlg;
        BOOL   bDone;
    begin

        // カスタム ダイアログ ボックスが表示するスタティック テキストに利用する
        // フォントのハンドルを取得します。
        hFont1 = GetFont("Arial", 14, STYLE_BOLD);
        hFont2 = GetFont("Times New Roman", 11, STYLE_ITALIC);
        hFont3 = GetFont("Arial", 10, STYLE_BOLD);
        hFont4 = GetFont("Courier New", 9, STYLE_NORMAL);

        if (hFont1 = 0 || hFont2 = 0 || hFont3 = 0 || hFont4 = 0) then
            // エラーを報告し、終了します。
            MessageBox (" すべてのフォントを取得できませんでした。", SEVERE);
            abort;
        endif;

        // このインストールでカスタム ボックスを認識するための名前を指定します。
        szDialogName = "CustomDialog";

        // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
        // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
        // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
        // 4 番目のパラメーターにある ID によって識別されるためです。

        nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

        if (nResult < 0) then
            // エラーを報告し、終了します。
            MessageBox (" ダイアログの定義エラー ", SEVERE);
            abort;
        endif;

        // while ループを制御するのに使われるインジケーターを初期化します。
```

```
bDone = FALSE;

// 完了するまでループします。
repeat

   // ダイアログを表示して次のダイアログ イベントを戻します。
   nCmdValue = WaitOnDialog (szDialogName);

   // イベントに応答します。
   switch (nCmdValue)
      case DLG_CLOSE:
         // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
         Do (EXIT);
      case DLG_ERR:
         MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
         abort;
      case DLG_INIT:
         // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
         // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
         // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
         // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
         hwndDlg = CmdGetHwndDlg (szDialogName);
         SdGeneralInit(szDialogName, hwndDlg, 0, "");

         // スタティック テキスト ボックス 1 のフォントとテキストを設定します。
         if (CtrlSetFont (szDialogName, hFont1, RES_TEXT_1) = 0) then
            CtrlSetText (szDialogName, RES_TEXT_1,
                  " このテキストは Arial bold  14 ポイントに設定されています。");
         else
            CtrlSetText (szDialogName, RES_TEXT_1,
               " 最初のスタティック テキストボックス用のフォントを設定できませんでした。");
         endif;

         // スタティック テキスト ボックス 2 のフォントとテキストを設定します。
         if (CtrlSetFont (szDialogName, hFont2, RES_TEXT_2) = 0) then
            CtrlSetText (szDialogName, RES_TEXT_2,
                  " このテキストは Times New Roman italic 11 ポイントに設定されています。");
         else
            CtrlSetText (szDialogName, RES_TEXT_2,
                  "2 番目のスタティックテキストボックス用のフォントを設定できませんでした。");
         endif;

         // スタティック テキスト ボックス 3 のフォントとテキストを設定します。
         if (CtrlSetFont (szDialogName, hFont3, RES_TEXT_3) = 0) then
            CtrlSetText (szDialogName, RES_TEXT_3,
                  " このテキストは Arial bold  10 ポイントに設定されています。");
         else
            CtrlSetText (szDialogName, RES_TEXT_3,
                  "3 番目のスタティックテキストボックス用のフォントを設定できませんでした。");
         endif;

         // スタティック テキスト ボックス 4 のフォントとテキストを設定します。
         if (CtrlSetFont (szDialogName, hFont4, RES_TEXT_4) = 0) then
            CtrlSetText (szDialogName, RES_TEXT_4,
                  " このテキストは Courier New  9 ポイントに設定されています。");
         else
            CtrlSetText (szDialogName, RES_TEXT_4,
                  "4 番目のスタティックテキストボックス用のフォントを設定できませんでした。");
          endif;
```

```
            case RES_PBUT_NEXT:
                bDone = TRUE;
            case RES_PBUT_CANCEL:
                // ユーザーが [ キャンセル ] ボタンをクリックしました。
                Do (EXIT);
        endswitch;

    until bDone;

    // ダイアログを閉じます
    EndDialog (szDialogName);

    // メモリからダイアログを解放します。
    ReleaseDialog (szDialogName);

end;
```

GetLine

GetLine 関数は、読み取り専用モードで開かれたテキストファイルからテキスト行を読み込みます。GetLine を呼び出す前に、まず OpenFileMode を呼び出してファイルを読み取り専用モードに設定してから、OpenFile を呼び出してファイル (インターネット上のファイルも可能) を開く必要があります。GetLine への最初の呼び出しは、ファイルから最初のテキスト行を読み取ります。行を読み終わると、GetLine は次の行にファイルポインターを再配置します。GetLine への 2 回目の呼び出しでは 2 行目を読み取り、次の行へと続きます。GetLine は戻す行の末尾にある改行文字や復帰改行文字を削除します。

GetLine がファイル内のすべての行を読み終わると、end-of-file エラーを返します。追加モードでファイルを開いた場合、GetLine 関数への呼び出しは失敗します。これは、ファイルポインターがファイルの終わりにある為です。関数はまた、nvFileHandle が指定したファイルがバイナリモードで開かれた場合にも失敗します。

行の最大文字制限は 4,096 です。ファイルから複数行を読み取るには、各行に別々の GetLine を利用するか、GetLine ステートメントをループに配置してください。

ヒント・テキストファイルを書き込むには、WriteLine 関数を利用してください。WriteLine は常に末尾に改行文字や復帰改行文字の組み合わせをもつ行を作成します。

構文

```
GetLine ( nvFileHandle, svLine );
```

パラメーター

テーブル 165 ・ GetLine のパラメーター

パラメーター	説明
nvFileHandle	OpenFile への呼び出しで開かれたファイルのハンドルを指定します。
svLine	nvFileHandle が指定したファイルからテキスト行を返します。

戻り値

テーブル 166 ・ GetLine の戻り値

戻り値	説明
0	開いているテキストファイルから関数がテキスト行を読み出したことを示します。
< 0	ファイルの end-of-file エラーまたはその他の条件により、関数が失敗したことを示します。この条件は GetLine がファイルのすべての行を読みとったことも示します。

GetLine の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* GetLine 関数のデモンストレーションを行います。
*
* このスクリプトでは GetLine が呼び出されてテキストファイルの各行を
* 読みます。
*
* メモ : このスクリプトを適切に実行するため、
*      ターゲットシステム上の既存テキストファイルを参照するよう
*      プリプロセッサ定数を設定します。
*
¥*---------------------------------------------------------------------*/

#define EXAMPLE_FILE "Readme.txt"
#define EXAMPLE_DIR "C:¥¥Windows"
#define TITLE_TEXT   "GetLine の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"
```

```
    export prototype ExFn_GetLine(HWND);

function ExFn_GetLine(hMSI)
    STRING  szFileName, szPath, szText, svLine;
    NUMBER  nFlag, nFileHandle;
    LIST    listID;
begin

    // ファイルからの行を格納するためのリストを作成します。
    listID = ListCreate (STRINGLIST);

    // ファイル モードを通常に設定します。
    OpenFileMode (FILE_MODE_NORMAL);

    // 編集のためファイルを開きます。
    OpenFile (nFileHandle, EXAMPLE_DIR, EXAMPLE_FILE);

    // ファイルの行をリストへ取得します。
    while (GetLine (nFileHandle, svLine) = 0)
        ListAddString (listID, svLine, AFTER);
    endwhile;

    // ファイルを閉じます。
    CloseFile (nFileHandle);

    // リストを表示します。
    SdShowInfoList (TITLE_TEXT, EXAMPLE_FILE, listID);

    // メモリからリストを削除します。
    ListDestroy(listID);

end;
```

GetMemFree

GetMemFree 関数は古い形式のため、使用できません。この関数は常に 1048576 を返します。

構文

```
GetMemFree();
```

ヒント・ターゲットシステム上の有効物理メモリ容量を判断するには、GetSystemInfo を呼び出します。

GetObject

GetObject 関数は、szObjectName によって名付けられた オブジェクトを初期化し、設定されたキーワードを使用して OBJECT 型の変数に割り当てられるリファレンスを返します。新規または既存の COM オブジェクトへのリファレンス (Visual Basic の GetObject 関数の動作と同じ) を取得するには、CoGetObject を呼び出します。

ヒント・オブジェクトが正常に初期化されたかどうかを確認するには、IsObject 関数を呼び出します。

構文

GetObject (szObjectName);

パラメーター

テーブル 167 ・ GetObject のパラメーター

パラメーター	説明
szObjectName	オブジェクトの名前が、IDE の [機能] ビューに表示される名前として初期化されるよう、例のように指定します : " 新しい ATL 3.0" ***メモ・** 正しい InstallShield オブジェクトが初期化されたことを確認するには、プロジェクトの各 InstallShield オブジェクトに固有な名前を付けます。* オブジェクトプロジェクトで szObjectName がヌル文字列 ("") に設定されている場合、GetObject はリファレンスを呼び出し元のオブジェクトに返します (セットアッププロジェクトでは、GetObject は無効なリファレンスを返します)。 szObjectName をヌル文字列に設定すると、オブジェクト内でオブジェクトプロパティを読み書きするのに便利です。例えば、次のコードはオブジェクトの現在の状態についての説明を表示します。 OBJECT oThis; set oThis = GetObject(""); if(IsObject(oThis)) then MessageBox(oThis.Status.Description, INFORMATION); endif;

戻り値

設定されたキーワードを使用して、OBJECT 型の変数に割り当てられるリファレンス。

GetObjectByIndex

***プロジェクト・** この情報は、InstallScript プロジェクトに適用します。*

GetObjectByIndex 関数は、nIndex が指定したセットアップまたはオブジェクトのサブオブジェクトを検出し、設定されたキーワードを使用して OBJECT 変数タイプに割り当てられるリファレンスを返します。

構文

GetObjectByIndex (nIndex);

パラメーター

テーブル 168 ・ GetObjectByIndex のパラメーター

パラメーター	説明
nIndex	サブオブジェクトのインデックスを指定します。インデックス番号は 1 から始まります。オブジェクトが引数としてゼロ (0) を GetObjectByIndex へ渡す場合、関数はサブオブジェクトではなく関数を呼び出すオブジェクトまたはセットアップへのリファレンスを返します。(これは、GetObject("") を呼び出すのと同じことです。) セットアップが GetObjectByIndex(0) を呼び出す場合、戻されたレファレンスは特に重要な情報を含みません。

戻り値

設定されたキーワードを使用して、OBJECT 型の変数に割り当てられるリファレンス。

追加情報

オブジェクトが正常に初期化されたかどうかを確認するには、IsObject 関数を呼び出します。

オブジェクトまたはセットアップが含むサブオブジェクトの数を取得するには、GetObjectCount を呼び出します。

GetObjectCount

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

GetObjectCount 関数は、オブジェクトまたはセットアップが含むサブオブジェクトの数を返します。

構文

```
GetObjectCount ( );
```

パラメーター

なし

戻り値

テーブル 169 ・ GetObjectCount の戻り値

戻り値	説明
>= 0	サブオブジェクトの数。
< ISERR_SUCCESS	GetObjectCount はサブオブジェクトの数を判断できませんでした。

追加情報

セットアップまたはオブジェクト内のインデックスが指定するサブオブジェクトへのリファレンスを取得するには、GetObjectByIndex を呼び出します。

GetProfInt

GetProfInt 関数は .ini ファイルの整数を読み出します。GetProfInt は、nDefault のパラメーターが 0 に指定された Windows API GetPrivateProfileInt と同様に動作します。

構文

```
GetProfInt ( szFileName, szSectionName, szKeyName, nvValue );
```

パラメーター

テーブル 170 ・ GetProfInt パラメーター

パラメーター	説明
szFileName	キーの整数値を取得する .ini ファイルの名前を指定します。szFileName が完全修飾名ではない (つまり、ドライブの指定およびパスが*含まれていない*) 場合、InstallShield は Windows フォルダーでファイルを検索します。
szSectionName	szKeyName を検索する、.ini ファイルセクション名を指定します。ここで指定するセクション名は、各括弧 ([ ]) で囲まないでください。この名前の検索は大文字と小文字を区別しません。
szKeyName	nvValue に返される整数値を持つキーを指定します。このキーの検索は大文字と小文字を区別しません。
nvValue	szKeyName に現在割り当てられている整数値を返します。GetPrivateProfileInt 関数の制限のため、この関数はプロファイルから 16 ビット値のみを戻すことができます。従って最大戻り値は 65,535 で、大きな値は不正確な可能性があります。大きな値を戻す必要がある場合、FileGrep や FileInsertLine といった一般ファイル処理関数を利用し、StrToNum を呼び出して戻された文字列を整数に変換してください。

戻り値

GetProfInt は常に 0 を返します。

追加情報

- Windows API 関数 **GetPrivateProfileInt** と同様に、ファイル、セクション、キー名が検出されなかった場合にエラーは戻されません。その代わり、nvValue は 0 を含みます。このため、エラーと戻り値 0 との判別ができません。0 とエラーを区別するため、**GetPrivateProfileInt** を直接呼出して、代替となるデフォルト値を指定してください。
- **GetPrivateProfileInt** の呼び出しの一部 (つまり **GetProfInt** の呼び出し) は、プロファイルではなく Windows レジストリに自動的にマップされます。

GetProfInt の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* GetProfInt 関数のデモンストレーションを行います。
*
* GetProfInt が呼び出されて、EXAMPLE_INI が指定したファイルから
* キーの値を読み出します。
*
* メモ : このスクリプトを実行する前に、C ドライブのルートへ
*      ISExample.ini と呼ばれるファイルを作成します。ファイルには
*      次の行を含みます。
*
*      [ISExample]
*      ISKey=100
*
¥*---------------------------------------------------------------------*/

#define  EXAMPLE_INI  "C:¥¥ISExampl.ini"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_GetProfInt(HWND);

function ExFn_GetProfInt(hMSI)
   STRING szSectionName, szKeyName;
   NUMBER nvValue;
begin

   szSectionName = "ISExample";
   szKeyName     = "ISKey";

   // szSectionName セクションにある szKeyName の値を取得します。
   GetProfInt (EXAMPLE_INI, szSectionName, szKeyName, nvValue);

   SprintfBox (INFORMATION, "GetProfInt の例 ",
            "%s の値は %d です。", szKeyName, nvValue);
```

```
end;
```

GetProfSectionKeyCount

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

GetProSectionKeyCount 関数は、szFilename で指定した初期設定ファイルにある szSectionName が指定するセクション内のキーの数を返します。

構文

```
GetProfSectionKeyCount ( szFilename, szSectionName );
```

パラメーター

テーブル 171 ・ GetProfSectionKeyCount のパラメーター

パラメーター	説明
szFilename	キーを数える .ini ファイルの完全修飾名を指定します。
szSectionName	キーを数える .ini ファイルセクションの名前を指定します。ここで指定するセクション名は、各括弧 ([ ]) で囲まないでください。この名前の検索は大文字と小文字を区別しません。

戻り値

テーブル 172 ・ GetProfSectionKeyCount の戻り値

戻り値	説明
X	指定したファイルの指定したセクションにあるキーの数。
< ISERR_SUCCESS	関数がキーの数を判断できなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

GetProfString

GetProfString 関数は、指定の .ini ファイルからプロファイル文字列を読み出しします。GetProfString は Windows API の GetPrivateProfileString と同じ働きをします。

メモ・GetProfString はオペレーティング環境の API で提供された関数を使用して .ini ファイルにアクセスします。このため、InstallShield の機能はオペレーティング環境によって制限される可能性があります。

構文

GetProfString (szFileName, szSectionName, szKeyName, svResult);

パラメーター

テーブル 173 ・ GetProfString のパラメーター

パラメーター	説明
szFileName	キーの現在値を取得する .ini ファイルの名前を指定します。szFileName が完全修飾名ではない (つまり、ドライブの指定およびパスが *含まれていない*) 場合、InstallShield は Windows フォルダーでファイルを検索します。
szSectionName	szKeyName を検索する、.ini ファイルセクション名を指定します。ここで指定するセクション名は、各括弧 ([ ]) で囲まないでください。この名前の検索は大文字と小文字を区別しません。初期化ファイルのすべてのセクション名一覧を取得するには、このパラメーターにヌル文字列 ("") を渡します。 ***ヒント・** 初期化ファイルのすべてのセクション名一覧を取得するには、szSectionName にヌル文字列 ("") を渡します。ヌル文字列で区切られたセクション名が svResult パラメーターに返されます。SvResult にはすべてのセクション名を受理できる長さが必要です。この文字列から個別のセクション名を抽出するには、StrGetTokens 関数を使用します。*
szKeyName	svResult に返される値を持つキーを指定します。このキーの検索は大文字と小文字を区別しません。セクションのすべてのキー名一覧を取得するには、このパラメーターにヌル文字列 ("") を渡します。 ***ヒント・** szSectionName で指定したセクションのすべてのキー名一覧を取得するには、szKeyName にヌル文字列 ("") を渡します。ヌル文字列で区切られたキー名が svResult パラメーターに返されます。SvResult にはすべてのキー名を受理できる長さが必要です。この文字列から個別のキー名を抽出するには、StrGetTokens 関数を使用します。*
svResult	szSectionName でセクション名を指定し、szKeyName でキー名を指定すると、そのキーの値がこのパラメーターに返されます。szSectionName でヌル文字列 ("") を指定すると、すべてのセクション名が svResult に返されます。szKeyName でヌル文字列 ("") を指定すると、szSectionName で指定したセクションのすべてのキー名が svResult に戻されます。

戻り値

テーブル 174 ・ GetProfString の戻り値

戻り値	説明
0	GetProfString はプロファイル文字列値を返しました。
< 0	GetProfString は値を戻すことができませんでした。

テーブル 174 ・ GetProfString の戻り値 (続き)

戻り値	説明
-2	キー値の長さが GetProfString が svResult に戻すことのできる最大文字数、2048 文字を越えました。

GetProfString の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* 関数 AddProfString と GetProfString をデモンストレーションします。
*
* このスクリプトは、ファイルにプロファイル文字列を追加します。
* 追加された文字列を読み出して表示します。
*
* メモ : このスクリプトを初めて実行した時、ドライブ C のルートに
*      ISExampl.ini と名づけられたファイルを作成します。
*      このファイルはスクリプト解析が終了した時点で
*      削除することができます。
*
¥*---------------------------------------------------------------*/

#define EXAMPLE_INI "C:¥¥ISExampl.ini"

// ファイルに追加する新しいセクション、キー、および値。
#define NEW_SECTION " 新しいセクション "
#define NEW_KEY " 新しいキー "
#define NEW_VALUE " テスト "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_GetProfString(HWND);

function ExFn_GetProfString(hMSI)
   STRING svResult;
begin

   // ファイルへプロファイル文字列を追加します。
   if (AddProfString (EXAMPLE_INI, NEW_SECTION, NEW_KEY, NEW_VALUE) != 0) then
      // 文字列を追加できない場合は、エラー メッセージを表示します。
      MessageBox ("AddProfString が失敗しました。", SEVERE);
   else
      // ファイルからキーの値を読み出します。
      if (GetProfString (EXAMPLE_INI, NEW_SECTION, NEW_KEY, svResult) != 0) then
         // 文字列読み出せない場合は、エラー メッセージを表示します。
         MessageBox ("GetProfString が失敗しました。", SEVERE);
```

```
        else
            // キーとその現在の値を表示します。
            MessageBox (NEW_KEY + "=" + svResult, INFORMATION);
        endif;
    endif;

end;
```

GetProfStringList

GetProfStringList 関数は、指定した初期化ファイル内の指定したセクションからのキー名と文字列値のリストを読み出します。

構文

```
GetProfStringList ( szFileName, szSectionName, listKeyNames, listValues );
```

パラメーター

テーブル 175 • GetProfStringList のパラメーター

パラメーター	説明
szFileName	キー名と文字列値を取得する元の .ini ファイルの名前を指定します。szFileName が完全修飾名ではない場合 (ドライブ指定とパスが含まれていない場合)、InstallShield は Windows フォルダーのファイルを検索します。
szSectionName	キー名と文字列値を検索する元の .ini ファイルのセクションを指定します。ここで指定するセクション名は、各括弧 ([ ]) で囲まないでください。この名前の検索は大文字と小文字を区別しません。
listKeyNames	キー名のリストを返します。listKeyNames によって識別される文字列リストは、ListCreate への呼び出しによって既に初期化されている必要があります。
listValues	文字列値のリストを返します。listValues によって識別される文字列リストは、ListCreate への呼び出しによって既に初期化されている必要があります。

戻り値

テーブル 176 • GetProfStringList の戻り値

戻り値	説明
>= ISERR_SUCCESS (0)	関数がセクションを読み込み、指定したリストへキー名と文字列値を挿入したことを示します。
< ISERR_SUCCESS (0)	関数がセクションを読み込むことができなかった、または指定したリストへキー名と文字列値を挿入できなかったことを示します。

追加情報

GetProfStringList は Windows API **GetPrivateProfileSection** を呼び出して、そのデータに 32 KB バッファーを指定します。つまり、関数はセクションの最初の 32 KB のデータのみを返します。32 KB データ以上のセクションを処理する必要があるインストールでは、Windows API **GetPrivateProfileSection** を直接呼び出して大きいバッファー サイズを指定 (および戻された情報を手動で解析) する必要があります。

GetProfStringList の例

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* 関数 GetProfStringList リストのデモンストレーションを行います。
*
* スクリプトは初期化ファイルからキーと値を
* 読み出します。
¥*---------------------------------------------------------------------*/

#define EXAMPLE_INI "C:¥¥ISExampl.ini"
```

```
// ファイルに追加する新しいセクション、キー、および値。
#define SECTION "InstallShield"
#define KEY1    "Key1"
#define KEY2    "Key2"
#define KEY3    "Key3"
#define KEY4    "Key4"
#define KEY5    "Key5"
#define VALUE1  1
#define VALUE2  2
#define VALUE3  3
#define VALUE4  4
#define VALUE5  5

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

  export prototype ExFn_GetProfString(HWND);

function ExFn_GetProfString(hMSI)
    STRING svResult, svKeyName, svKeyVal;
    LIST listKeyNames, listKeyValues;
    NUMBER nVal;
begin

    // ファイルへプロファイル文字列を追加します。
    WriteProfInt (EXAMPLE_INI, SECTION, KEY1, VALUE1);
    WriteProfInt (EXAMPLE_INI, SECTION, KEY2, VALUE2);
    WriteProfInt (EXAMPLE_INI, SECTION, KEY3, VALUE3);
    WriteProfInt (EXAMPLE_INI, SECTION, KEY4, VALUE4);
    WriteProfInt (EXAMPLE_INI, SECTION, KEY5, VALUE5);

    // キー名を保持するためのリストを作成します。
    listKeyNames = ListCreate(STRINGLIST);

    // エラーが発生した場合にそれを報告し、終了します。
    if (listKeyNames = LIST_NULL) then
        MessageBox ("リストを作成できませんでした。", SEVERE);
        // カスタム エラー処理コードをここに追加します。
        abort;
    endif;

    // キー値を保持するためのリストを作成します。
    listKeyValues = ListCreate(STRINGLIST);

    // エラーが発生した場合にそれを報告し、終了します。
    if (listKeyValues = LIST_NULL) then
        MessageBox ("リストを作成できませんでした。", SEVERE);
        // カスタム エラー処理コードをここに追加します。
        abort;
    endif;

    // 指定したファイルのセクションからキーを読み出します。
    nVal = GetProfStringList (EXAMPLE_INI, SECTION,
                              listKeyNames, listKeyValues);

    if (nVal = 0) then
        nVal = ListGetFirstString (listKeyNames, svKeyName);
```

```
        if (nVal = END_OF_LIST) then
            MessageBox(" [" + SECTION + "] にはキーがありません ", WARNING);
        else
            ListGetFirstString (listKeyValues, svKeyVal);

            repeat
                // キーとその値を表示します。
                MessageBox(svKeyName + "=" + svKeyVal, INFORMATION);

                nVal = ListGetNextString (listKeyNames, svKeyName);
                if !(nVal = END_OF_LIST) then
                    ListGetNextString (listKeyValues, svKeyVal);
                endif;

            until nVal = END_OF_LIST;

        endif;
    endif;

end;
```

GetShortcutInfo

GetShortcutInfo 関数は特定のプログラムアイテム、またはサブフォルダー名の存在を確認します。InstallScript エンジンがショートカットまたはサブフォルダーを検出すると **GetShortcutInfo** がその属性を返します。属性は製品のコマンドライン、作業ディレクトリ、アイコンパス、ショートカット キー、および最小化フラグを含みます。

GetShortcutInfo を利用するには、パラメーター szFolderName と szName で情報を入力してください。InstallScript エンジン ファイルは残りのパラメーターにショートカットまたはサブフォルダーの属性を入力します。

構文

```
GetShortcutInfo (szShortcutFolder, szName, svCmdLine, svWrkDir, svIconPath, nvIconIndex, svShortCutKey, nvMinimizeFlag);
```

パラメーター

テーブル 177 • GetShortcutInfo のパラメーター

パラメーター	説明
szShortcutFolder	ショートカットまたはサブフォルダーを含むフォルダーの名前を指定します。szShortcutFolder の完全修飾パスを次の様に指定することができます : "C:¥¥ProgramData¥¥Microsoft¥¥Windows¥¥Start Menu¥¥Programs¥¥Games" szShortcutFolder がヌルの場合、**GetShortcutInfo** はデフォルトのプログラム ディレクトリを検索します。szShortcutFolder に絶対パス (ドライブ名を含むパス。例、"C:¥¥Program Files¥¥AppName") を指定しなかった場合、**GetShortcutInfo** はデフォルトの Program ディレクトリの下にあるサブフォルダを検索します。この場所は InstallScript 変数 ALLUSERS の値、およびターゲット システム上の Windows バージョンによって異なります。 InstallScript システム変数を使用することもできます。 • FOLDER_DESKTOP – [デスクトップ] フォルダーでアイテムをクエリします。 • FOLDER_STARTUP – Startup メニューでアイテムをクエリします。 • FOLDER_STARTMENU – Start メニューでアイテムをクエリします。 • FOLDER_PROGRAMS – Start¥Programs メニューでアイテムをクエリします。 または、次のような関連パスを使用できます。 **FOLDER_PROGRAMS ^ "ACCESSORIES¥¥GAMES"**
szName	探しているショートカットまたはサブフォルダーの名前を指定します。
svCmdLine	関数は、アイテムの実行可能ファイルのコマンドラインまたはサブフォルダーへの完全パスを返します。
svWrkDir	関数は、プログラムアイテムの作業ディレクトリの完全パスを返します。(szName がサブフォルダーの場合を除きます。)
svIconPath	関数は、.ico ファイルまたは .exe ファイルのファイル名と完全パスを返します。(szName がサブフォルダーの場合を除きます。)
nvIconIndex	関数は、ショートカットに使用されているアイコンのインデックスを返します。(szName がサブフォルダーの場合を除きます。)
svShortCutKey	関数は、アイテムのショートカット キーを返します。(szName がサブフォルダーの場合を除きます。)

テーブル 177 ・ GetShortcutInfo のパラメーター (続き)

パラメーター	説明
nvMinimizeFlag	アプリケーション ウィンドウが最初に表示されたときに最小化するかどうかを示す、次の定数のを返します。 ・ NULL – アプリケーションのウィンドウは、スタートアップ時には最小化されないことを示します。 ・ RUN_MINIMIZED – アプリケーションのウィンドウは、スタートアップ時に最小化されることを示します。 (szName がサブフォルダーの場合を除きます。)

戻り値

テーブル 178 ・ GetShortcutInfo の戻り値

戻り値	説明
IS_ITEM (0)	szName は、szShortcutFolder のショートカットであることを示します。
IS_FOLDER (1)	szName は、szShortcutFolder のサブフォルダーであることを示します。
< 0	関数がショートカットまたはサブフォルダー名を検出できなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

[スタート] メニューの配置は各言語別によって異なります。InstallScript エンジンが自動的に正しいパスを選択します。

GetShortcutInfo の例

***メモ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* GetShortcutInfo 関数のデモンストレーションを行います。
*
* GetShortcutInfo が呼び出されて、ターゲットファイルまたはサブフォルダーの
* 属性を検出します。
*
```

```
＊メモ : このスクリプトを実行する前に、
＊      FOLDER_NAME および SHORTCUT が、
＊      既存のフォルダー名とショートカットを参照するように定義します。
＊
¥*---------------------------------------------------------------*/
// ショートカットまたはフォルダー名を参照するように定数を定義します。
#define FOLDER_NAME"C:¥¥Windows¥¥Start Menu¥¥Programs"
#define SHORTCUT    "InstallShield"

function OnFirstUIAfter( )
    STRING svCmdLine, svWrkDir, svIconPath;
    STRING svShortCutKey, svGroupPath, szTitle, szMsg, szInfo, svMinFlag;
    STRING svMinimizeFlag;
    NUMBER nvIconIndex, nvMinimizeFlag, nResult, nvMinFlag;
    LIST listInfo, listID;
begin

    // FOLDER_NAME フォルダーでアイテムを検索します。
    nResult = GetShortcutInfo (FOLDER_NAME, SHORTCUT, svCmdLine, svWrkDir,
                               svIconPath, nvIconIndex, svShortCutKey,
                               nvMinimizeFlag);

    // 文字列リストを作成します。
    listInfo = ListCreate (STRINGLIST);

    // GetShortcutInfo のエラーをチェックします。
    if (nResult < 0) then
        // エラーをレポートし、中止します。
        MessageBox ("GetShortcutInfo が失敗しました。", SEVERE);
        abort;
    // アイテムがアプリケーションか否かを確認します。
    elseif (nResult = IS_ITEM) then
        // コマンドラインを文字列リストへ追加します。
        Sprintf (szInfo, "%s のコマンドライン : %s", SHORTCUT, svCmdLine);
        ListAddString (listInfo, szInfo, AFTER);

        // 作業ディレクトリを文字列リストへ追加します。
        Sprintf (szInfo, "%s の作業ディレクトリ : %s", SHORTCUT, svWrkDir);
        ListAddString (listInfo, szInfo, AFTER);

        // アイコンパスを文字列リストへ追加します。
        Sprintf (szInfo, "%s のアイコンパス : %s", SHORTCUT, svIconPath);
        ListAddString (listInfo, szInfo, AFTER);

        // アイコンインデックスを文字列リストへ追加します。
        Sprintf (szInfo, " アイコンのインデックス : %d", nvIconIndex);
        ListAddString (listInfo, szInfo, AFTER);

        // 文字列リストへショートカットキーを追加します。
        Sprintf (szInfo, "%s のショートカット キー : %s", SHORTCUT,
              svShortCutKey);
        ListAddString (listInfo, szInfo, AFTER);

        // アイテムがフォルダーか否かを確認します。
    elseif (nResult = IS_FOLDER) then
        // メッセージを文字列リストへ追加します。
        Sprintf (szInfo, " アイテムはサブフォルダーです GetShortcutInfo は、" +
              " サブフォルダーに関する情報を充分に読み出しません。");
        ListAddString (listInfo, szInfo, AFTER);
```

```
    endif;

    // 文字列リストを表示します。
    szTitle = "GetShortcutInfo の例 ";
    szMsg = " アイテムの属性は次の通りです :";
    SdShowInfoList (szTitle, szMsg, listInfo);

    // リストを破棄します。
    ListDestroy(listID);

end;
```

GetStatus

***プロジェクト・GetStatus** 関数は、InstallScript オブジェクト プロジェクトに適用します。*

GetStatus 関数は、オブジェクトの現在のステータス (Status.Number の現在の値) を取得します。

***ヒント・**追加の情報、または、オブジェクトを含むインストールからオブジェクトのステータスを取得する場合、オブジェクトの Status オブジェクトを使用します。*

構文

```
number GetStatus();
```

パラメーター

GetStatus にはパラメーターはありません。

戻り値

GetStatus は、オブジェクトのステータスを返します。.

GetSystemInfo

GetSystemInfo 関数はターゲットシステムについての情報を読み出します。**GetSystemInfo** は Windows API を利用して戻す情報を収集します。

構文

```
GetSystemInfo (nItem, nvResult, svResult);
```

パラメーター

テーブル 179 • GetSystemInfo のパラメーター

パラメーター	説明
nItem	読み出す情報の種類を指定します。
nvResult	数値データ形式でシステム情報を返します。
svResult	文字列データ形式でシステム情報を返します。

以下の表に、システム情報を読み出すために nItem パラメーターに渡すことが可能な定数の一覧を示します。特定の定数 (DISK_TOTALSPACE_EX など) を使用する場合、関数を呼び出す前に nvResult または svResult パラメーターで追加情報を指定する必要があります。

テーブル 180 • nItem オプション

nItem オプション	nvResult 戻り値	svResult 戻り値
BOOTUPDRIVE	起動ドライブの ID。1 = A、2 = B、3 = C。値に 64 (10 進数) を追加して、この値への文字列値を設定することによって、この数値を適当なドライブ文字に変換することができます。次の構文を利用して変換します : svResult[0] = 64 + nvResult;	起動ドライブのドライブインストール先 (ドライブ名の後にコロン) を返します。
CDROM	CD-ROM が使用できるかどうかを TRUE か FALSE で示します。	なし
COLORS	ユーザーのシステムで使用できる色の数を返します。結果はモニターカードではなく、ターゲットシステムのビデオドライバーから読み出されます。カードが 256 色をサポートできてもドライバーが 16 色しかサポートできなければ、返される色の数は 16 になります。	なし

テーブル 180 • nItem オプション (続き)

nItem オプション	nvResult 戻り値	svResult 戻り値
CPU	 *メモ • このパラメーターは重要ではありません。SYSPROCESSORINFO の 構造メンバーを使用して、プロセッサタイプを判別します。* 次の定数の 1 つが戻されます : • IS_UNKNOWN ― ユーザーの CPU は不明です。 • IS_386 ― ユーザーは 386 プロセッサを使用しています。 • IS_486 ― ユーザーは 486 プロセッサを使用しています。 • IS_PENTIUM― ユーザーは PENTIUM プロセッサを使用しています。 • IS_ALPHA ― ユーザーは ALPHA プロセッサを使用しています。	なし
DATE	なし	現在のシステムの日付を MM-DD-YYYY の形式で表します。月と日のフィールドでは最初のゼロは抑制されます。
DISK_TOTALSPACE	 *メモ • このパラメーターは現在使用されていません。代わりに、GetDiskInfo 関数と共に利用可能はパラメーターを使用してください。* svResult で指定されたディスクドライブの総容量を返します。戻される最大値は 2 GB です。これより大きい総ディスク容量の場合も 2 GB が返されます。	ドライブ文字。このパラメーターは*関数*へ 渡されます。つまり GetSystemInfo を*呼び出す前に*、svResult に値を割り当てる必要があります。また、ドライブ文字の後にコロン (:) を入れることを忘れないでください。忘れると関数は失敗します。このパラメーターで UNC パスを指定することもできます。
DISK_TOTALSPACE_EX	 *メモ • このパラメーターは現在使用されていません。代わりに、GetDiskInfo 関数と共に利用可能はパラメーターを使用してください。* 測定単位を指定します。このパラメーターに、あらかじめ定義されている定数 (BYTES、KBYTES、MBYTES、または GBYTES) のうちの 1 つを渡します。svResult で指定されたディスクドライブの総容量を返します。	ドライブ文字。このパラメーターは*関数*へ 渡されます。つまり GetSystemInfo を*呼び出す前に*、svResult に値を割り当てる必要があります。また、ドライブ文字の後にコロン (:) を入れることを忘れないでください。忘れると関数は失敗します。このパラメーターで UNC パスを指定することもできます。

テーブル 180 • nItem オプション (続き)

nItem オプション	nvResult 戻り値	svResult 戻り値
DRIVE	*メモ • このパラメーターは現在使用されていません。代わりに、GetDiskInfo 関数と共に利用可能はパラメーターを使用してください。* svResult で指定されたドライブの種類を返します。次の定数の 1 つが戻されます : • IS_UNKNOWN ― ターゲットドライブが不明です。 • IS_REMOVABLE ― ターゲットドライブはフロッピードライブです。 • IS_FIXED ― ターゲット ドライブは固定ドライブです。 • IS_CDROM ― ターゲットドライブは CD-ROM ドライブです。 • IS_REMOTE ― ターゲットドライブはネットワークドライブです。	コロン (:) が後に続くドライブの文字。このパラメーターは*関数*へ渡されます。つまり GetSystemInfo を*呼び出す前に*、svResult に値を割り当てる必要があります。 オペレーティング システムの制限により、UNC パスは svResult ではサポートされていません。svResult で UNC パスを渡す場合、関数は IS_UNKNOWN を返します。
EXTENDEDMEMORY	マシンにインストールされているメモリ *総容量* を返します。オペレーティングシステムの制限により、返される値がシステムにインストールされている実際のメモリとやや異なる場合があります。この値は実際の値の 100K (0.1 MB) 以内の差です。返される値はキロバイト単位です。 GetSystemInfo (EXTENDEDMEMORY, ...) は最大 2 テラバイト (TB) までの拡張メモリを正確に返します。システムに 2 TB 以上の拡張メモリがある場合、2 TB が戻されます。	なし
LANGUAGE	このパラメーターにはターゲット システムの InstallScript 言語定数が戻されます。詳しい情報は、「複数言語インストールの作成」を参照してください。	nvResult で返された言語定数に等しい言語名文字列が、このパラメーターに返されます。

テーブル 180 • nItem オプション (続き)

nItem オプション	nvResult 戻り値	svResult 戻り値
OS	 *メモ • このパラメーターは重要ではありません。SYSINFO 構造変数の WIN9X.bWin9X または WINNT.bWinNT メンバーを利用して、オペレーティングシステム情報を取得します。* コンポーネント、機能、およびカスタム アクションに条件付きロジックを使用して、オペレーティング システムに基づいて製品のすべて、またはその一部を条件付きでインストールすることができます。詳細については、「条件ステートメントのビルド」を参照してください。	なし
OSMAJOR	このパラメーターは重要ではありません。SYSINFO 構造変数の nOSMajor メンバーを利用して、オペレーティングシステム情報を取得します。	なし
OSMINOR	このパラメーターは重要ではありません。SYSINFO 構造変数の nOSMinor メンバーを利用して、オペレーティングシステム情報を取得します。	なし
PARALLEL	使用できる物理パラレルポート数を返します。	なし
SERIAL	使用できる物理シリアルポート数を返します。	なし
SYSTEM_DPI	システム DPI 値を返します。	なし
SYSTEM_DPI_SCALING	システム DPI スケーリング値を返します。	なし
TIME	なし	現在のシステム時間を HH:MM:SS 形式で表します。

テーブル 180 - nItem オプション (続き)

nItem オプション	nvResult 戻り値	svResult 戻り値
VIDEO	インストールされているビデオアダプタの種類を返します。(InstallShield は CGA またはモノクロームのビデオドライバーを検出できません。) 次の定数の 1 つが戻されます : • IS_UNKNOWN ー ユーザーのビデオは不明です。 • IS_EGA ー EGA 解像度。 • IS_VGA ー VGA 解像度。 • IS_SVGA ー Super VGA (800 x 600) 解像度。 • IS_XVGA ー XVGA (1024 x 768) 解像度。 • IS_UVGA ー 1024 x 768 以上の解像度。	なし
VIRTUAL_MACHINE_TYPE	なし この定数は、関数の戻り値を通して、その値を戻します。 詳細については、「インストールが仮想マシン上で実行されているかどうかを検出する」を参照してください。	なし
VOLUMELABEL	なし	読み出すボリュームラベルを持つディスクのドライブのインストール先 (ドライブ名の後にコロン) を渡します。指定のドライブのボリュームラベルがこのパラメーターに返されます。ドライバーにボリュームラベルがない場合、ヌル文字列 ("") が返されます。
WINMAJOR	このパラメーターは重要ではありません。SYSINFO 構造変数の nWinMajor メンバーを利用して、オペレーティングシステム情報を取得します。	なし
WINMINOR	このパラメーターは重要ではありません。SYSINFO 構造変数の nWinMinor メンバーを利用して、オペレーティングシステム情報を取得します。	なし

戻り値

テーブル 181 ・ GetSystemInfo の戻り値

戻り値	説明
IS_VM_TYPE_HYPERV	インストールは Microsoft Hyper-V マシン上で実行中です。 **GetSystemInfo** は、VIRTUAL_MACHINE_TYPE が nItem に渡された場合に、この値を戻す場合があります。詳細については、「インストールが仮想マシン上で実行されているかどうかを検出する」を参照してください。
IS_VM_TYPE_NONE	仮想マシンが検出されませんでした。 **GetSystemInfo** は、VIRTUAL_MACHINE_TYPE が nItem に渡された場合に、この値を戻す場合があります。詳細については、「インストールが仮想マシン上で実行されているかどうかを検出する」を参照してください。
IS_VM_TYPE_VMWARE	インストールは VMware Player、VMware Workstation、または VMware Server などの VMware 製品上で実行されています。 **GetSystemInfo** は、VIRTUAL_MACHINE_TYPE が nItem に渡された場合に、この値を戻す場合があります。詳細については、「インストールが仮想マシン上で実行されているかどうかを検出する」を参照してください。
IS_VM_TYPE_VIRTUALPC	インストールは Microsoft 仮想 PC マシン上で実行中です。 **GetSystemInfo** は、VIRTUAL_MACHINE_TYPE が nItem に渡された場合に、この値を戻す場合があります。詳細については、「インストールが仮想マシン上で実行されているかどうかを検出する」を参照してください。
IS_VM_TYPE_UNKNOWN	仮想マシンの種類が不明です。 **GetSystemInfo** は、VIRTUAL_MACHINE_TYPE が nItem に渡された場合に、この値を戻す場合があります。詳細については、「インストールが仮想マシン上で実行されているかどうかを検出する」を参照してください。
0	関数が指定した情報を返したことを示します。
< 0	関数を使用して要求した情報が返せなかったことを示します。

GetSystemInfo の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* GetSystemInfo 関数のデモンストレーションを行います。
```

```
*
* このスクリプトは GetSystemInfo で有効な多くの定数を利用し、
* すべての可能な戻り値をテストします。
* 結果はダイアログに表示されます。
*
¥*-----------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_GetSystemInfo(HWND);

function ExFn_GetSystemInfo(hMSI)
    STRING   szTitle, szMsg, svResult, szInfo;
    NUMBER   nvResult;
    LIST     listInfo;
begin

    // システム情報のリストを作成します。
    listInfo = ListCreate (STRINGLIST);

    // 拡張メモリの容量を取得します。
    if (GetSystemInfo (EXTENDEDMEMORY, nvResult, svResult) < 0) then
        szInfo = "EXTENDEDMEMORY 情報を取得することができませんでした。";
    else
        Sprintf(szInfo, " 拡張メモリ : %d K", nvResult);
    endif;

    // リストへ情報を追加します。
    ListAddString(listInfo, szInfo, AFTER);

    // 起動ドライブを取得します。
    if (GetSystemInfo (BOOTUPDRIVE, nvResult, svResult) < 0) then
        szInfo = "BOOTUPDRIVE 情報を取得できませんでした。";
    else
        Sprintf(szInfo, " 起動ドライブ : %s", svResult);
    endif;

    // リストへ情報を追加します。
    ListAddString(listInfo, szInfo, AFTER);

    // CD-ROM についての情報を取得します。
    if (GetSystemInfo (CDROM, nvResult, svResult) < 0) then
        szInfo = "CD-ROM 情報を取得できませんでした。";
    else
        if (nvResult = 0) then
            svResult = " いいえ ";
        else
            svResult = " はい ";
        endif;

        Sprintf(szInfo, "CDROM: %s", svResult);
    endif;

    // リストへ情報を追加します。
    ListAddString(listInfo, szInfo, AFTER);

    // ビデオ アダプタを取得します。
    if (GetSystemInfo (VIDEO, nvResult, svResult) < 0) then
```

```
    szInfo = "VIDEO 情報を取得できませんでした。";
else
    switch (nvResult)
        case IS_UNKNOWN:
            szInfo = "VIDEO: UNKNOWN";
        case IS_SVGA:
            szInfo = "VIDEO: SVGA";
        case IS_XVGA:
            szInfo = "VIDEO: XVGA";
        case IS_UVGA:
            szInfo = "VIDEO: UVGA";
    endswitch;
endif;

// リストへ情報を追加します。
ListAddString(listInfo, szInfo, AFTER);

// 利用可能な色の数を取得します。
if (GetSystemInfo (COLORS, nvResult, svResult) < 0) then
    szInfo = "COLORS 情報を取得できませんでした。";
else
    Sprintf(szInfo, " 色の数 : %d", nvResult);
endif;

// リストへ情報を追加します。
ListAddString(listInfo, szInfo, AFTER);

// 現在の日付を取得します。
if (GetSystemInfo (DATE, nvResult, svResult) < 0) then
    szInfo = "DATE 情報を取得できませんでした。";
else
    Sprintf(szInfo, " 日付 : %s", svResult);
endif;

// リストへ情報を追加します。
ListAddString(listInfo, szInfo, AFTER);

// 現在の時間を取得します。
if (GetSystemInfo (TIME, nvResult, svResult) < 0) then
    szInfo = "TIME 情報を取得できませんでした。";
else
    Sprintf(szInfo, " 時刻 : %s", svResult);
endif;

// リストへ情報を追加します。
ListAddString(listInfo, szInfo, AFTER);

// オペレーティングシステムを取得します。
if (GetSystemInfo (OS, nvResult, svResult) < 0) then
    szInfo = " オペレーティングシステム情報を取得できませんでした。";
else
    switch (nvResult)
        case IS_WINDOWSNT:
            szInfo = "OS: Windows NT";
        case IS_WINDOWS9X:
            GetSystemInfo (WINMINOR, nvResult, svResult);

            if (nvResult < 10) then
                szInfo = "OS: Windows 95";
```

```
            else
                szInfo = "OS: Windows 98";
            endif;
        endswitch;
    endif;

    // リストへ情報を追加します。
    ListAddString(listInfo, szInfo, AFTER);

    // 情報を表示します。
    szTitle   = " システム情報 ";
    szMsg     = " 次に示すのは、ご利用中のシステムに関連する情報です :¥n";

    SdShowInfoList (szTitle, szMsg, listInfo);

    ListDestroy(listInfo);

end;
```

GetTempFileNameIS

GetTempFileNameIS 関数は Windows API **GetTempFileName** を呼び出して一時ファイルを作成し、その関連アクションを実行します。Windows API **GetTempFileName** とは異なり、存在しないとき、**GetTempFileNameIS** は szPathName で指定されたフォルダーを作成します。**CreateDir** 関数を使用するときと同様、新規作成されたフォルダーはアンインストールではログ記録されません。

構文

```
GetTempFileNameIS( byval string szPathName, byval string szPrefixString, byval number nUnique, byref string svTempFileName, byval
    number nOptions );
```

パラメーター

テーブル 182 ・ GetTempFileNameIS のパラメーター

パラメーター	説明
szPathName	Windows API GetTempFileName の lpPathName パラメーターのパスを指定します。 このパラメーターにテキスト置換を使用することができます。
szPrefixString	Windows API GetTempFileName の lpPrefixString パラメーターの文字列を指定します。 このパラメーターにテキスト置換を使用することができます。
nUnique	Windows API GetTempFileName の nUnique パラメーターの整数を指定します。
svTempFileName	Windows API GetTempFileName の lpTempFileName パラメーターの値を指定します。 ***メモ ・** InstallScript 文字列の長さは自動的に _MAX_PATH 以上になります。したがって、この文字列を手動でサイズ指定する必要はありません。*
nOptions	InstallScript 特定のオプションを指定します。以下から、このパラメーターで渡す定義済み定数すべて選択します : ・ GTFIS_OPTION_NONE ― InstallScript 固有のオプションではありません。デフォルトでは、これが設定されています。 ・ GTFIS_OPTION_DONT_RESOLVE_TEXTSUBS ― szPathName と szPrefixString でテキスト置換解決する **TextSubSubstitute** を呼び出しません。 ・ GTFIS_OPTION_DONT_CREATE_DIR ― 存在しないとき、szPathName によって指定されたディレクトリを作成しません。このオプションが指定された状態で、szPathName が存在しないとき、関数は失敗しますので注意してください。 ・ GTFIS_OPTION_DELETE_TEMP_FILE ― 戻される前に、作成された一時ファイルを削除します。(このオプションは、関数で一時ファイルが作成されるように nUnique を指定した場合のみ有益です。詳細については、「GetTempFileName 関数」を参照してください。) 関数は、一時ファイルが削除できなくても失敗は返しません。 ・

戻り値

テーブル 183 ・ GetTempFileNameIS の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数が成功したことを示します。
< ISERR_SUCCESS	関数の実行に失敗したことを示します。

GetTrueTypeFontFileInfo

GetTrueTypeFontFileInfo 関数は szTrueTypeFontFile が指定する TrueType フォント ファイルについての情報を svResult に返します。

構文

GetTrueTypeFontFileInfo (szTrueTypeFontFile, nInfo, nLanguage, svResult);

パラメーター

テーブル 184 ・ GetTrueTypeFontFileInfo のパラメーター

パラメーター	説明
szTrueTypeFontFile	その情報を取得する TrueType ファイルの完全修飾ファイル名を指定します。この関数は .fon および .fot といった TrueType 以外のファイル、あるいは True Type コレクションファイル (.ttc ファイル) をサポートしない点にご注意下さい。
nInfo	ｓｖResult に戻される情報を指定します。このパラメーターに、以下の定数のうちの 1 つを渡します。 ・ TTFONTFILEINFO_FONTTITLE— フォントのタイトルを指定します。 ・ name ID—name ID を使用して、フォント情報を取得します。 ・
nLanguage	svResult に戻される文字列の言語を指定します。定義済み言語定数、数値言語 ID、システム変数 SELECTED_LANGUAGE、または SELECTED_LANGUAGE を渡した場合と同じ結果が得られる 0 (ゼロ) をこのパラメーターで渡します。すべての言語で特定のフォントについての情報すべてが利用できるとは限りませんので、ご注意ください。
svResult	ｎInfo で指定したフォント情報を返します。ファイルのデータが Unicode フォーマットで格納されているとしても、ANSI (ユニコードではない) 文字列が戻されます。

戻り値

テーブル 185 ・ GetTrueTypeFontFileInfo の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数が要求した情報を読み出したことを示します。
< ISERR_SUCCESS	関数を使用して要求した情報が取得できなかったことを示します。

GetUpdateStatus

GetUpdateStatus 関数は現在使用されていません。この関数が呼び出されたとき、それは FALSE を返します。

構文

```
BOOL GetUpdateStatus();
```

GetUpdateStatusReboot

GetUpdateStatusReboot 関数は現在使用されていません。この関数が呼び出されたとき、それは FALSE を返します。

構文

BOOL GetUpdateStatusReboot();

GetValidDrivesList

GetValidDrivesList 関数は特定の条件を満たすターゲットシステムに接続されているすべてのドライブリストを読み出します。この条件にはドライブの種類、ドライブ容量の最小値が含まれます。ドライブの扉が開いていると、ドライブ名はリストに挿入されたままです。

構文

GetValidDrivesList (listID, nDriveType, nMinDriveSpace);

パラメーター

テーブル 186 · GetValidDrivesList のパラメーター

パラメーター	説明
listID	有効なドライブ名のリストを返します。listID によって識別される文字列リストは、ListCreate への呼び出しによって既に初期化されている必要があります。
nDriveType	検索するドライブの種類を指定します。' このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · –1– すべてドライブの種類を検索します。 · FIXED_DRIVE– 固定ドライブのみを検索します。 · REMOTE_DRIVE– リモート ドライブのみを検索します。リモートドライブは一般的にネットワーク上にあります。 · REMOVEABLE_DRIVE– リムーバブル ドライブのみを検索します。フロッピードライブはリムーバブルドライブです。 · CDROM_DRIVE–CD-ROM ドライブのみを検索します。
nMinDriveSpace	ドライブを戻り値のリストに入れる場合の、空き容量をバイト単位で示した最小値。nMinDriveSpace がゼロ未満の場合、GetValidDrivesList はドライブの最小空き容量をチェックしません。これはフロッピードライブに便利です。

戻り値

テーブル 187 · GetValidDrivesList の戻り値

戻り値	説明
0	GetValidDrivesList は要求されたリストを読み出すことができました。
< 0	GetValidDrivesList はリストを読み出せませんでした。

追加情報

- ドライブがリストされる前に、検索するドライブの種類や最小有効ディスクの容量を指定することができます。
- ネットワークマッピングドライブはリモートドライブとして戻されることもあります。GetValidDrivesList はネットワーク上のすべてのドライブを戻さない場合もあります。マッピングドライブとしてのみ指定されたドライブのみが戻されます。

GetValidDrivesList の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * GetValidDrivesList 関数のデモンストレーションを行います。
 *
 * GetValidDrivesList は 2 回呼び出されます。1 回目は、120,000 バイト以上の
 * 空き容量を持つリムーバブルドライブのリストを戻すため、
 * そして 2 回目は 1,000,000 バイト以上の空き容量を持つ
 * 固定ドライブのリストを戻すためです。
 *
¥*---------------------------------------------------------------*/

#define TITLE        "GetValidDrivesList の例 "
#define MSG_REMOVABLE "120,000 バイトの空き容量を持つリムーバブルドライブ :"
#define MSG_FIXED     "1,000,000 バイトの空き容量を持つ固定ドライブ。"
#define MSG_ERR       "GetValidDrivesList が失敗しました。"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_GetValidDrivesList(HWND);

function ExFn_GetValidDrivesList(hMSI)
   LIST   listID;
begin

   // リムーバブル ドライブの名前を保持するリストを作成します。
   listID = ListCreate (STRINGLIST);

   // 120,000 バイト以上の空き容量を持つリムーバブル ドライブを取得します。
   if (GetValidDrivesList (listID, REMOVEABLE_DRIVE, 120000) < 0) then
      // エラーを報告し、終了します。
      MessageBox (MSG_ERR, SEVERE);
      abort;
   else
      // リムーバブル ドライブのリストを表示します。
      SdShowInfoList (TITLE, MSG_REMOVABLE, listID);
   endif;

   // リムーバブル ドライブのリストを破棄します。
   ListDestroy(listID);

   // 固定ドライブの名前を保持するリストを作成します。
   listID = ListCreate (STRINGLIST);

   // 1,000,000 バイト以上の空き容量を持つリ固定ドライブを取得します。
   if (GetValidDrivesList (listID, FIXED_DRIVE, 1000000) < 0) then
      // エラーを報告し、終了します。
      MessageBox (MSG_ERR, SEVERE);
```

```
        abort;
    else
        // 固定ドライブのリストを表示します。
        SdShowInfoList (TITLE, MSG_FIXED, listID);
    endif;

end;
```

GetWCHARArrayFromISStringArray

GetWCHARArrayFromISStringArray 関数は、指定された配列に含まれる Unicode 文字へのポインター配列へポインターを返します。

構文

```
GetWCHARArrayFromISStringArray ( vArray );
```

パラメーター

テーブル 188 - GetWCHARArrayFromISStringArray のパラメーター

パラメーター	説明
vArray	ポインターが必要な文字列配列を指定します。

戻り値

テーブル 189 - GetWCHARArrayFromISStringArray の戻り値

戻り値	説明
pointer	Unicode 文字列へのポインターの配列へのポインター。
< ISERR_SUCCESS	関数が失敗したことを示します。

追加情報

GetWCHARArrayFromISStringArray 関数はポインターの配列と Unicode 文字列用に追加メモリを割り当てます。このポインターは LPWSTR* 引数を受け取る関数へ渡されます。新しく作成した配列での作業を完了した後、DeleteWCHARArray を呼び出してメモリから配列を削除します。

CopyCHARArrayToISStringArray を呼び出してポインターの配列から元の文字列配列へデータを返した場合、配列に含まれる文字列を変更する際は注意してください。文字列配列に含まれる文字列の長さはインストールが内部的に管理するため、文字列の長さを変更すると **CopyCHARArrayToISStringArray** を呼び出した際に、文字列全体がオリジナル配列へコピーされません。

GetWindowHandle

GetWindowHandle 関数はインストレーションのメインウィンドウのハンドルを取得します。

構文

GetWindowHandle (nHwndFlag);

パラメーター

テーブル 190 - GetWindowHandle のパラメーター

パラメーター	説明
nHwndFlag	InstallShield のメインウィンドウのウィンドウハンドルを指定します。このパラメーターで、定義済み定数 HWND_INSTALL を渡します。

戻り値

テーブル 191 - GetWindowHandle のパラメーター

戻り値	説明
X	X はウィンドウのハンドルです。
< 0	GetWindowHandle はハンドルを読み出すことができませんでした。

GetWindowHandle の例

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * GetWindowHandle 関数のデモンストレーションを行います。
 *
 * このスクリプトは標準ウィンドウでセットアップを開始します。3 秒後に
 * ウィンドウは最小化されます。その後、もう一度停止し、
 * ウィンドウは最大化され、メッセージボックスが表示されます。
 *
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_GetWindowHandle(HWND);

function ExFn_GetWindowHandle(hMSI)
    NUMBER nHwnd;
    HWND hInstallHwnd;
begin

    // このセットアップ用の標準ウィンドウを指定します。
    Enable (DEFWINDOWMODE);

    // 背景ウィンドウを表示します。
    Enable (BACKGROUND);

    // セットアプのウィンドウハンドルを取得します。
```

```
    nHwnd = GetWindowHandle (HWND_INSTALL);

    // 3 秒間待機します。
    Delay (3);

    // システム コマンドを送り、ウィンドウを最小化します。
    SendMessage (nHwnd, WM_SYSCOMMAND, SC_MINIMIZE, 0);

    // 3 秒間待機します。
    Delay (3);

    // システム コマンドを送り、ウィンドウを最大化します。
    SendMessage (nHwnd, WM_SYSCOMMAND, SC_MAXIMIZE, 0);

    // メッセージを表示します。
    MessageBox (" デモンストレーションが完了しました。", INFORMATION);

end;
```

ビルトイン関数 (H-P)

カテゴリ別の関数一覧は、「カテゴリー別ビルトイン関数」を参照してください。

Handler

Handler 関数は現在使用されていません。代わりに HandlerEx を使用してください。

HandlerEx

HandlerEx 関数は [ヘルプ] アクセラレータキー (F1) や [キャンセル] キーといったイベントのカスタム ハンドラーを作成します。

エンドユーザーが F1 キーを押すと、現在定義されている HELP ハンドラーが実行されます。エンドユーザーが [キャンセル] キーを押すと、現在定義されている EXIT ハンドラーが実行されます。カスタム HELP または EXIT ハンドラーが **HandlerEx** 関数を使って定義されていない場合、デフォルトのハンドラーが実行されます。デフォルトの EXIT ハンドラーでは、[終了] ダイアログが表示されます。デフォルトの HELP ハンドラーでは何も実行されません。

HandlerEx を使って定義されたカスタム ハンドラーを実行する場合、nObject イベントが発生したときに、パラメーター Label で指定されている固有のラベルが InstallScript エンジン によって呼び出されます。InstallScript エンジンがハンドラー コードの return ステートメント (ラベルの下) に達すると、コントロールは、ハンドラー ラベルが呼び出されなかった場合に続いて実行される予定のステートメントに戻ります。

HandlerEx を使って EXIT または HELP のカスタム処理を指定できます。Exit ハンドラー内では **MessageBox**、**SprintfBox**、および **AskYesNo** ダイアログが表示できますが、標準ダイアログは表示できません。

構文

```
HandlerEx (nObject, Label);
```

パラメーター

テーブル 1 · HandlerEx のパラメーター

パラメーター	説明
nObject	トラップするイベントを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · **EXIT** –[キャンセル] ボタンが押されたときにカスタムハンドラーが呼び出されるよう指定します。ハンドラーが定義されていない場合、[キャンセル] ボタンが押されると、デフォルトの [終了] ダイアログが表示されます。 · **HELP**–F1 アクセラレータキーが押されたときにカスタムハンドラーが呼び出されるよう指定します。ハンドラーが定義されていない場合、F1 アクセラレータキーが押されても何も実行されません。
Label	指定したボタンまたはアクセラレータキーが押されたときにプログラムがジャンプする先のラベル名を指定します。このラベルは、数値または文字列変数として定義しないでください。 現在定義されているハンドラーをキャンセルしてデフォルトハンドラーを再インストールするには、このパラメーターで -1 を渡します。この方法は、特定の処理を行う場合にのみカスタムハンドラーをインストールし、その後デフォルトのハンドラーに戻るようなスクリプトで役に立ちます。 ***注意 ·** このラベルは、数値または文字列変数として定義しないでください。*

戻り値

テーブル 2 · HandlerEx の戻り値

戻り値	説明
0	**HandlerEx** により正常にハンドルが作成されました。
< 0	**HandlerEx** はハンドルを作成できませんでした。

追加情報

· 利用可能な唯一のアクセラレータは F1 ファンクションキー (ヘルプ) のみです。

· Help (F1) を使って、ヘルプ エンジンを起動したり、その他の適切なヘルプを表示することができます。エンド ユーザーが F1 キーを押すと、現在定義されている Help ハンドラーが InstallScript エンジンによって呼び出されます。Help ハンドラーではヘルプのあらゆる機能を利用できます。コンテキスト ヘルプを作成する場合は、スクリプトのコンテキストを追跡する必要があります。

これには、たとえば、現在のコンテキスト文字列を含む文字列変数を利用する方法があります。HELP ハンドラー内の Switch-case ステートメントにこの文字列変数を組み込むと、コンテキスト文字列の値に基づいて適切なヘルプイベントが実行されます。また、Help 処理ルーチンで nSdDialog グローバル変数の値をテストすることもできます。nSdDialog は、現在実行中の Sd ダイアログのダイアログ ID (*InstallShield Program Files フォル*

ダー¥Script¥Isrt¥Include フォルダーの _isres.h に記述されている ID) に設定されます (標準ダイアログが実行されていない場合、nSdDialog は定義されません)。これによりユーザーは、Sd ダイアログの実行中でも Sd ダイアログ固有のヘルプにアクセスできるようになります。

- HELP ハンドラーと同様に EXIT ハンドラーの場合も、カスタムハンドラーおよび Sd ダイアログ固有の EXIT ハンドラーを定義して、実行することができます。

HandlerEx の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* HandlerEx 関数のデモンストレーションを行います。
*
* このスクリプトは、セットアップでのエラーとヘルプハンドラーのインストール方法を
* 説明します。
*
* まず初めに、ハンドラーがインストールされます。そして、ユーザーに Sd ダイアログ
* ボックスが表示されます。ダイアログが表示されている間に、ユーザーが
* F1 ( ヘルプ ) または [ キャンセル ] ボタンを押すと、
* ハンドラーコードが起動します。
*
* デモンストレーションの目的で、スクリプトは InstallScript 言語リファレンスと
* ヘルプ ライブラリのヘルプ ファイルを利用します。
* これらのファイルのコピー (InstallShield Program Files フォルダー ¥Program フォルダーから Langref.chm と HelpLib.chm) を
* [ サポート ファイル / ビルボード ] ビューの [ 言語非依存 ] 領域に
* 挿入します。
¥*---------------------------------------------------------------*/

// 含まれているヘッダー ファイル ---------------------------------------------------
#include "ifx.h"

export prototype void ExFn_Handler();
prototype OnHelp_Handler();
prototype OnExit_Handler();

// ヘルプトピック ID を定義します。
#define HELP_WELCOME      101
#define HELP_REGISTERUSER 102

INT    nHelpID;

function void ExFn_Handler()
    STRING svName, svCompany, szMsg;
begin
    // help ハンドラーをインストールします。
    HandlerEx (HELP, OnHelp_Handler);

    // exit ハンドラーをインストールします。
```

```
    HandlerEx (EXIT, OnExit_Handler);

    // ダイアログで表示するメッセージを設定します。
    szMsg = " F1 を押して InstallShield の " +
        " このダイアログについてのヘルプを表示してください。¥n" +
        "Cancel を押して、カスタム exit ハンドラーを起動してください。";

WelcomeDialog:
    // ヘルプトピック ID を設定します。
    nHelpID = HELP_WELCOME;

    // ようこそダイアログを表示します。
    SdWelcome ("SdWelcome ダイアログ ", szMsg);

    // ヘルプトピック ID を設定します。
    nHelpID = HELP_REGISTERUSER;

    if SdRegisterUser ("Register", szMsg, svName, svCompany) = BACK then
        // ようこそダイアログへ戻ります。
        goto WelcomeDialog;
    endif;

end;

function OnHelp_Handler()
    STRING szHelpTopic;
begin
    // ヘルプから表示するトピックを設定します。
    switch (nHelpID)
        case HELP_WELCOME      : szHelpTopic = "5223 ";
        case HELP_REGISTERUSER : szHelpTopic = "5211 ";
    endswitch;

    //InstallScript の言語リファレンスを起動し、
    // 選択したヘルプをトピックを表示します。

    LaunchApplication (WINDIR ^ "hh.exe", "-mapid " + szHelpTopic +
                       SUPPORTDIR ^ "Help_Lib.chm",
                       "", SW_SHOW, INFINITE, LAAW_OPTION_WAIT);
end;

function OnExit_Handler()
begin
    // 終了の確認をとります。
    if AskYesNo (" 終了しますか ?", FALSE) = YES then
        // セットアップを終了します。
        abort;
    else
        // セットアップへ戻ります。
        return 0;
    endif;
end;
```

HIBYTE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

HIBYTE 関数は shValue が指定した 16 ビット整数値から上位バイトを高い順に抽出します。

構文

HIBYTE (shValue);

パラメーター

テーブル 3 - HIBYTE のパラメーター

パラメーター	説明
shValue	上位バイトを抽出する 16 ビット整数を指定します。

戻り値

この関数は整数の高位バイトを戻します。

HIWORD

HIWORD 関数は、lValue が指定した 32 ビット正数値から高位の単語 (上位 2 バイト) を抽出し、戻します。

InstallShield の HIWORD は記号拡張子を利用する点で対応する C マクロとは異なります。その結果、HIWORD が戻す値の高位バイトは、lValue が負の場合は 1 で埋められます。必要に応じて、次に示すようにビットワイズ AND 演算子 (&) を使って結果を 0xFFFF と組み合わせることで値を正数にすることができます :

```
lValue = HIWORD(lValue);
lValue = lValue & 0xFFFF;
```

構文

HIWORD (lValue);

パラメーター

テーブル 4 - HIWORD のパラメーター

パラメーター	説明
lValue	上位 2 バイトを抽出する 32 ビット正数を指定します。

戻り値

HIWORD は lValue の高位の単語 (上位 2 バイト) を戻します。

HIWORD の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * HIWORD と LOWORD のデモンストレーションを行います。
 *
 * このスクリプト例は、値から低位の単語と高位の単語を取得する
 * HIWORD と LOWORD の利用法を説明します。
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "LOWORD/HIWORD の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_HIWORD(HWND);

function ExFn_HIWORD(hMSI)
   STRING  szMsg;
   NUMBER  nData, nLOWORD, nHIWORD;
begin

   nData = 305419896; // 16 進値 : 12345678
   // 低位の単語、22136 (16 進法 : 5678) を取得します。

   nLOWORD = LOWORD (nData);

   // 高位の単語、4660 (16 進法 : 1234) を取得します。
   nHIWORD  = HIWORD (nData);

   // 結果を表示します。
   szMsg  = "LOWORD: %ld¥nHIWORD: %ld";
   SprintfBox (INFORMATION, TITLE_TEXT, szMsg, nLOWORD, nHIWORD);

end;
```

InstallationInfo

InstallationInfo 関数は現在使用されていません。代わりに、CreateInstallationInfo 関数を使用してください。

構文

InstallationInfo (szCompany、 szProduct、szVersion、szProductKey) ;

Is

Is 関数は、スクリプトで一般的に必要とされる情報を取得します。

構文

Is (nIsFlag、 szIsData);

パラメーター

テーブル 5 ・ Is のパラメーター

パラメーター	説明
nIsFlag	読み出す情報の種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ BACKBUTTON Ñ いくつかのビルトインダイアログに表示される [戻る] ボタンが有効かどうか。 ・ CANCELBUTTON Ñ いくつかのビルトインダイアログに表示される [キャンセル] ボタンが有効かどうか。 ・ DOTNETFRAMEWORKINSTALLED Ñ 特定のバージョンの .NET Framework または言語パックをインストールするかどうか。再起動が必要なセットアップ前提条件の詳細は、「追加情報」を参照してください。 ・ DOTNETSERVICEPACKINSTALLED Ñ .NET Framework の特定のサービス パック (または、より新しいバージョンのサービス パック) をインストールするかどうか。再起動が必要なセットアップ前提条件の詳細は、「追加情報」を参照してください。 ・ DIR_WRITEABLE Ñ インストールが szIsData で指定されたディレクトリに書き込めるかどうか。 ・ FILE_EXISTS Ñ szIsData で指定されたファイルが存在するかどうか。 ・ FILE_LOCKED Ñ ファイルがロックされているかどうか。十分な権限がないためにファイルへのアクセスが不可能な場合、**Is** は TRUE を返します。 ・ FILE_WRITEABLE Ñ インストールが szIsData で指定されたファイルに書き込めるかどうか。 ・ FONT_AVAILABLE Ñ フォントはインストールされた szIsData で指定されたタイトルと同じかどうか。関数はすべての文字セットにあるフォントを検索することに注意してください。単一の文字セットを検索するには、Windows API 関数 **EnumFontFamiliesEx** を呼び出します。この関数についての詳細は Microsoft Windows API マニュアルを参照してください。 ・ FUNCTION_EXPORTED Ñ szIsData で指定された DLL が szIsData で指定された関数をエクスポートするかどうか。DLL が存在してロード可能で、関数がエクスポートされた場合、**Is** は TRUE を返します。それ以外の場合、**Is** は False を返します。 ・ LANGUAGE_SUPPORTED Ñ (InstallScript プロジェクトおよび InstallScript MSI プロジェクト) インストーラーで、szIsData で指定されている言語をサポートするかどうか。 ・ LOGGING Ñ (InstallScript プロジェクトのみ) アンインストールのログ記録が有効かどうか。 ・ MATH_COPROCESSOR Ñ 数値演算コプロセッサがターゲットシステムに存在するかどうか。 ・ NEXTBUTTON Ñ いくつかのビルトインダイアログに表示される [次へ] ボタンが有効かどうか。デフォルトでは、ほとんどのダイアログでの [次へ] ボタンは有効になっています。 ・ PATH_EXISTS Ñ szIsData で指定されたパスが存在するかどうか。 ・ REGDBREMOTEREGCONNECTED Ñ リモート レジストリが現在接続中かどうか。 ・ REBOOTED Ñ インストールが再起動後に実行中かどうか。

テーブル 5 ・ Is のパラメーター (続き)

パラメーター	説明
nIsFlag (続く)	・ SETUP_PACKAGE Ñ (InstallScript プロジェクトのみ) セットアップが 自己展開型実行可能ファイルから実行しているかどうか。 メンテナンス モードまたはアンインストール中、オリジナルセットアップが自己展開型実行可能ファイルの場合 (正確には、メンテナンスまたはアンインストールは自己展開型実行可能ファイルからではなく DISK1TARGET フォルダーにある **Setup.exe** のコピーから実行されています)、Is(SETUP_PACKAGE、szIsData) は TRUE を戻します。 ・ SKIN_LOADED Ñ ダイアログ スキンがロードされているかどうか。 ・ URL Ñ szIsData で指定された文字列が URL であるかどうか。(つまり "http://"、"https://"、"file://"、または "ftp://" で文字列が始まっているかどうか。) ・ USER_ADMINISTRATOR Ñ 現在のユーザーが管理者特権を持っているかどうか。Windows Vista 以降のシステムにおける一部のケースを除くすべてのケースで USER_ADMINISTRATOR が nFlag に渡されたとき、**Is** は TRUE を返します。Windows Vista 以降のシステムでは、USER_ADMINISTRATOR が nFlag に渡され、SE_GROUP_USE_FOR_DENY_ONLY セキュリティ識別子 (SID) 属性がこのグループに設定されていないとき、**Is** は TRUE を返します。つまり、現在のユーザーが管理者グループに属しているにもかかわらず、Windows Vista 上でインストールを標準アクセス トークンを使ってインストールを実行している場合、**Is** は FALSE を返します。 ・ USER_INADMINGROUP – 現在のユーザーが管理者グループに属するかどうか。SE_GROUP_USE_FOR_DENY_ONLY SID 属性がこのグループに設定されているいないにかかわらず (つまり、管理者権限で実行しているユーザーが標準アクセス トークンを使ってインストールを実行しているいないにかかわらず)、**Is** はこの定数に TRUE を返します。 ・ USER_POWERUSER – 現在のユーザーがパワー ユーザー グループに属するかどうか。 ・ VALID_PATH – szIsData で指定されたパスが有効なパスかどうか。これによってパスの存在は確認できません。構文がチェックされるだけです。ユーザーからパス情報を取得した時にこの定数を使用することができます。するとこの関数は、パス情報が正しく入力されたかどうかを確認します。 有効な URL 文字列は次の基準に当てはまります : (1) **http://**、**https://**、または **file://** から始まる。(2) 数値、文字、および記号 ! $ % & ' ( ) * + – . / ¥ _ のみを含む。$ % & ' ( ) * + – ./ ¥ _

テーブル 5 ・ Is のパラメーター (続き)

パラメーター	説明
nIsFlag (続く)	・ WEB_BASED_SETUP Ñ (InstallScript プロジェクトのみ) インストールがインターネットから実行されているかどうか。 ***メモ ・*** *Is(WEB_BASED_SETUP,szIsData) は、インストールの現在のインスタンスが Web から実行されているかどうかを確認します。そのため、インストールが [プログラムの追加と削除] ダイアログから実行されるとき、Is(WEB_BASED_SETUP,szIsData) は常に FALSE を戻します。これは元のインストールが Web から実行されている場合でも、インストールは常にローカル ファイルから実行されるためです。* ・ WINDOWS_SHARED Ñ Microsoft Windows がネットワークからの共有コピーを実行しているかどうか。 Microsoft Windows の共有コピーはネットワーク上にインストールされ、多くのユーザーが共有する共通ファイルを持っています。

テーブル 5 ・ Is のパラメーター (続き)

パラメーター	説明
szIsData	下に示すように、nIsFlag に渡された定数に依存する情報を指定します。パスまたはファイル名が引用符で括られていると、**Is** は失敗します。パスまたはファイル名が引用符で括られていないか確認するには、**Is** を呼び出す前に LongPathToQuote(szIsData, FALSE) を呼び出してください。次のリストは、各 nIsFlag オプションを指定したときに szIsData に含まれる内容を説明したものです。 ・ DIR_WRITEABLE Ñ szIsData は確認する完全修飾パスを指定します。 ・ DOTNETFRAMEWORKINSTALLED Ñ 特定のバージョンの .NET Framework または言語パックがインストールされているかどうかを判断します。再起動が必要なセットアップ前提条件の詳細は、「追加情報」を参照してください。 ・ DOTNETSERVICEPACKINSTALLED Ñ szIsData は、確認する .NET Framework の最小サービス パックおよびバージョンを、次のフォーマットで指定します。 *サービス パック番号*|*.NET バージョンのレジストリ定数またはパス* *サービス パック番号* は、最小サービス パックの数値を示します。.NET Framework 1.0 の場合、0 から 3 までの値を指定できます。.NET Framework 1.1 以降の場合、すべての数値がサポートされています。サービス パック番号と垂直線文字 (|) 区切り記号は両方とも含める必要があります。これらがないとき、関数の失敗を返します。 *.NET バージョンのレジストリ定数またはパス*は、確認する .NET Framework のバージョンを示すレジストリ定数またはレジストリ パスを示します。サポートされている定義済み値に関する情報、および **Is** 関数が確認を行うときの動作に関する詳細は、「追加情報」を参照してください。 ・ FILE_EXISTS Ñ szIsData は完全修飾ファイル名を指定します。このパラメーターでは、有効な URL を指定できます。URL が有効かどうかを確認するには、Is(VALID_PATH, szIsData) を呼び出します。 ・ FILE_LOCKED Ñ szIsData は完全修飾ファイル名を指定します。 ・ FILE_WRITEABLE Ñ szIsData は完全修飾ファイル名を指定します。 ・ FONT_AVAILABLE Ñ szIsData はフォントタイトルを指定します。 ・ FUNCTION_EXPORTED Ñ szIsData は DLL の完全修飾ファイルを指定し、その後に垂直線 (|) 区切りと関数の名前が続きます : 例、 *C:¥¥MyDLLFolder¥¥MyDLL.dll*|*MyFunction*。 ・ LANGUAGE_SUPPORTED – szIsData は、**Setup.ini** ファイルと言語 ID へのパスを次の形式で指定します : *Setup.ini のパス*	*言語 ID* **Setup.ini** のパスが指定されていない場合、現在のインストーラーが使用されます。言語 ID が指定されていない場合は、SELECTED_LANGUAGE が使用されます。 言語 ID は 0x で始まる 4 桁の 16 進数言語コードです。たとえば、英語の値は 0x0409 です。STANDARD_SELECTED_LANGUAGE を使ってこの形式で文字列を作成するには、次のようなステートメントを使用します : Sprintf (szLang, "0x%.04lx", STANDARD_SELECTED_LANGUAGE);

テーブル 5 - Is のパラメーター (続き)

パラメーター	説明
szIsData (続く)	• LOGGING Ñ szIsData は無視されます。 • MATH_COPROCESSOR Ñ szIsData を指定する szIsData は無視されます。 • PATH_EXISTS Ñ szIsData は完全修飾パスを指定します。このパラメーターでは、有効な URL を指定できます。URL が有効かどうかを確認するには、Is(VALID_PATH, szIsData) を呼び出します。 • REGDBREMOTEREGCONNECTED Ñ szIsData は無視されます。 • SETUP_PACKAGE Ñ szIsData は無視されます。 • SKIN_LOADED Ñ szIsData は無視されます。 • USER_ADMINISTRATOR Ñ szIsData は無視されます。 • USER_POWERUSER Ñ szIsData は無視されます。 • VALID_PATH Ñ szIsData は完全修飾パスを指定します。 • WEB_BASED_SETUP Ñ szIsData は無視されます。 • WINDOWS_SHARED Ñ szIsData は無視されます。

戻り値

テーブル 6 - Is の戻り値

戻り値	説明
TRUE (1)	答えが TRUE であることを示します。
FALSE (0)	答えが FALSE であることを示します。
< 0	Is 関数は質問に答えることができませんでした。

追加情報

.NET Framework のバージョンとサービスの詳細情報

次の定義済み定数は、DOTNETFRAMEWORKINSTALLED または DOTNETSERVICEPACKINSTALLED を使用して、.NET Framework の特定のバージョンを指定するためにサポートされています。

- REGDB_KEYPATH_DOTNET_40_CLIENT
- REGDB_KEYPATH_DOTNET_40_FULL
- REGDB_KEYPATH_DOTNET_35
- REGDB_KEYPATH_DOTNET_30_SP Ñ .NET Framework 3.0 の SP1 (またはそれ以降のサービス パック) がインストールされているかどうかを検出するには、この変数を使用します。
- REGDB_KEYPATH_DOTNET_30 Ñ .NET Framework 3.0 の RTM バージョンがインストールされているかどうかを検出するには、この変数を使用します。

- REGDB_KEYPATH_DOTNET_20
- REGDB_KEYPATH_DOTNET_11
- REGDB_KEYPATH_DOTNET_10

__ヒント__・これらの各定義済み定数は HKEY_LOCAL_MACHINE の下の適切なレジストリ パスに対応しているため、DOTNETFRAMEWORKINSTALLED または DOTNETSERVICEPACKINSTALLED の szIsData で直接レジストリ パスを指定することもできます。

.NET Framework 3.0 のインストールは、次の場所にレジストリ値 InstallSuccess とその値のデータ 1 を書き込みます :

HKEY_LOCAL_MACHINE¥Software¥Microsoft¥NET Framework Setup¥NDP¥v3.0¥Setup¥

したがって、REGDB_KEYPATH_DOTNET_30 がその場所に設定されます。

.NET Framework のその他すべてのバージョンのインストールは、次の場所にレジストリ値 Install とその値のデータ 1 を書き込みます :

HKEY_LOCAL_MACHINE¥Software¥Microsoft¥NET Framework Setup¥NDP¥ *バージョン番号*__¥__

REGDB_KEYPATH_DOTNET_35、REGDB_KEYPATH_DOTNET_20、REGDB_KEYPATH_DOTNET_11、および REGDB_KEYPATH_DOTNET_10 定数は、そのパスに基づいて適切な値に設定されます。

.NET Framework 3.0 SP1 のインストールは、次の場所にもレジストリ値 Install とその値のデータ 1 を書き込みます :

HKEY_LOCAL_MACHINE¥Software¥Microsoft¥NET Framework Setup¥NDP¥v3.0¥

したがって、REGDB_KEYPATH_DOTNET_30_SP がその場所に設定されます。

DOTNETFRAMEWORKINSTALLED 定数と一緒に **Is** 関数を使用すると、関数は自動的に Install と InstallSuccess の両方の値を確認します。Install または InstallSuccess 値が存在し、その値のデータが 1 に設定されている場合、**Is** は TRUE を返します。それ以外の場合、**Is** は False を返します。

.NET Framework 1.0 サービス パックのインストールは、サービス パック番号を示すレジストリ値を作成しません。したがって、.NET Framework 1.0 のテストに DOTNETSERVICEPACKINSTALLED を使うと、Is 関数はサービス パックを判別するために FOLDER_DOTNET_10 の **mscorlib.dll** ファイルのバージョンと、既知のバージョン番号とを比較します。

- **mscorlib.dll** のバージョンが 1.0.3705.6018 以降の場合、SP3 が存在します。
- **mscorlib.dll** のバージョンが 1.0.3705.288 以降で 1.0.3705.6018 未満の場合、SP2 が存在します。
- **mscorlib.dll** のバージョンが 1.0.3705.209 以降で 1.0.3705.288 未満の場合、SP1 が存在します。
- **mscorlib.dll** のバージョンが 1.0.3705.0 以降で 1.0.3705.209 未満の場合、元の RTM が存在します。

.NET Framework の他のバージョンのテストに DOTNETSERVICEPACKINSTALLED を使用すると、関数は、szIsData で指定されている [.NET バージョンのレジストリ定数またはパス] の下にある REGDB_VALUENAME_SP 値を szIsData で指定されているサービス パック番号と比較します。この比較は、バージョン番号が等しいか、またはより新しいかという基準で行われます。したがって、szIsData でサービス パック 2 を指定した場合、関数は、インストールされたサービス パックが 2 以降のとき、True を返します。

.NET Framework Language Pack の詳細

.NET Framework のバージョン 1.1 以降には、言語パックサポートが含まれますが、バージョン 1.0 には含まれません。.NET Framework のバージョン 1.1 以降では、特定の .NET 言語パックがインストールされているかどうかを、DOTNETFRAMEWORKINSTALLED 定数を使ってテストできます。そのためには、適切な .NET バージョン定数と、文字列に変換された言語のロケール識別子 (LCID) を指定します。.NET バージョン定数と LCID をキャレット演算子で区切ります。たとえば、次の構文を使用して、.NET Framework 1.1 のドイツ語 Language Pack がインストールされているかどうかをテストできます :

```
NumToStr( szLang, ISLANG_GERMAN_STANDARD );

REGDB_KEYPATH_DOTNET_11 ^ szLang;
```

.NET Framework 1.1 は、Microsoft のドキュメントに記載されているとおり、次の LCID をサポートします。

テーブル 7 - サポートされている .NET LCID

言語	LCID	対応する Is 定数
中国語 (簡体字)	2052 (0x0804)	ISLANG_CHINESE_SIMPLIFIED
中国語 (繁体)	1028 (0x0404)	ISLANG_CHINESE_TRADITIONAL
チェコ語	1029 (0x0405)	ISLANG_CZECH_STANDARD
デンマーク語	1030 (0x0406)	ISLANG_DANISH_STANDARD
オランダ語	1043 (0x0413)	ISLANG_DUTCH_STANDARD
フィンランド語	1035 (0x040B)	ISLANG_FINNISH_STANDARD
フランス語	1036 (0x040C)	ISLANG_FRENCH_STANDARD
ドイツ語	1031 (0x0407)	ISLANG_GERMAN_STANDARD
ギリシャ語	1032 (0x0408)	ISLANG_GREEK_STANDARD
ハンガリー語	1038 (0x040E)	ISLANG_HUNGARIAN_STANDARD
イタリア語	1040 (0x0410)	ISLANG_ITALIAN_STANDARD
日本語	1041 (0x0411)	ISLANG_JAPANESE_STANDARD
韓国語	1042 (0x0412)	ISLANG_KOREAN_STANDARD
ノルウェー語	1044 (0x0414)	ISLANG_NORWEGIAN_BOKMAL
ポーランド語	1045 (0x0415)	ISLANG_POLISH_STANDARD
ポルトガル語 (ブラジル)	1046 (0x0416)	ISLANG_PORTUGUESE_BRAZILIAN

テーブル 7・サポートされている .NET LCID (続き)

言語	LCID	対応する Is 定数
ポルトガル語 (ポルトガル)	2070 (0x0816)	ISLANG_PORTUGUESE_STANDARD
ロシア語	1049 (0x0419)	ISLANG_RUSSIAN_STANDARD
スペイン語	3082 (0x0C0A)	ISLANG_SPANISH_MODERNSORT
スウェーデン語	1053 (0x041D)	ISLANG_SWEDISH_STANDARD
トルコ語	1055 (0x041F)	ISLANG_TURKISH_STANDARD

Is の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* Is 関数のデモンストレーションを行います。
*
* Is 関数は、最初の呼び出しでファイルが書き込み禁止になっているかを
* 確認します。2 回目の呼び出しでディレクトリが書き込み禁止になっているかを
* 確認します。これはネットワーク上のディレクトリの
* 書き込み制限を確認するのに便利です。RenameFile への
* 呼び出されたとき、関数は Windows がネットワークサーバーにインストールされたか、
* ローカルシステムにインストールされたのかを確認
* します。
*
* メモ : このスクリプトを実行する前に、
*      ターゲットシステム上の既存ファイルをポイントするように
*      設定してください。
*
¥*---------------------------------------------------------------*/

#define EXAMPLE_DIR   "C:¥¥WINDOWS"
#define EXAMPLE_FILE  EXAMPLE_DIR^"Win.com"
#define TITLE         "IIs の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_Is(HWND);

function ExFn_Is(hMSI)
    NUMBER nResult;
begin
```

```
    // EXAMPLE_FILE ファイルが書込み禁止になっているかどうか確認します。
    nResult  = Is (FILE_WRITEABLE, EXAMPLE_FILE);

    // 結果をレポートします。
    if (nResult = TRUE) then
        SprintfBox (INFORMATION, TITLE, "%s は書き込み可能です。", EXAMPLE_FILE);
    elseif (nResult = FALSE) then
        SprintfBox (INFORMATION, TITLE, "%s 書き込み禁止です。", EXAMPLE_FILE);
    else
        SprintfBox (INFORMATION, TITLE,
                "Unable to determine if %s 書き込み可能です。", EXAMPLE_FILE);
    endif;

    // ディレクトリが書込み禁止になっているかどうか確認します。
    nResult  = Is (DIR_WRITEABLE, EXAMPLE_DIR);

    // 結果をレポートします。
    if (nResult = TRUE) then
        SprintfBox (INFORMATION, TITLE, "%s は書き込み可能です。.", EXAMPLE_DIR);
    elseif (nResult = FALSE) then
        SprintfBox (INFORMATION, TITLE, "%s は書き込み不可能です。", EXAMPLE_DIR);
    else
        SprintfBox (INFORMATION, TITLE,
                "%s が書き込み可能かどうかを判断できません。", EXAMPLE_DIR);
    endif;

    // Windows がネットワーク上で共有されているか否かを確認します。
    nResult  = Is (WINDOWS_SHARED, "");

    // 結果をレポートします。
    if (nResult = TRUE) then
        MessageBox ("Windows は共有されています。", INFORMATION);
    elseif (nResult = FALSE) then
        MessageBox ("Windows は共有されていません。", INFORMATION);
    else
        MessageBox ("Windows の共有されているかを判断できませんでした。", SEVERE);
    endif;

end;
```

ISCompareServicePack

ISCompareServicePack 関数は、InstallShield Professional で作成したスクリプトとの互換性の目的でのみサポートされています。Windows サービス パック番号を調べるには、SYSINFO.WINNT.nServicePack の値を調べることをお勧めします。

ISCompareServicePack 関数は、Windows NT システムにインストールされているサービス パック番号と szServicePack で指定されているサービス パック番号を比較します。

構文

```
ISCompareServicePack (szServicePack);
```

パラメーター

テーブル 8 ・ ISCompareServicePack のパラメーター

パラメーター	説明
szServicePack	ターゲットコンピューター上の サービス パック番号と比較するサービス パック番号を指定します。この文字列は Service Pack n の形式を使用する必要があります。n はサービス パック番号を示します。比較は大文字と小文字を区別します。

戻り値

テーブル 9 ・ ISCompareServicePack の戻り値

戻り値	説明
LESS_THAN (1)	サービス パックがインストールされていないか、ターゲットシステムの サービス パックが szServicePack に渡される値より小さいです。
EQUALS (2)	サービス パック番号が一致しています。
GREATER_THAN (0)	ターゲットシステムのサービス パック番号は szServicePack に渡された番号より大きいです。
-1	ISCompareServicePack はサービス パック番号を比較できませんでした。

ISCompareServicePack の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ISCompareServicePack 関数のデモンストレーションを行います。
*
* メモ : ISCompareServicePack は Sdint.rul で定義されます。
*
¥*---------------------------------------------------------------*/
#define SERVICE_PACK "サービスパック 3"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"
```

```
    export prototype ExFn_ISCompareServicePack(HWND);

function ExFn_ISCompareServicePack(hMSI)
    BOOL bWinNT;
    NUMBER nvResult;
    STRING svResult;
begin

    // どのオペレ－ティングシステムを実行しているかを判断します。
    GetSystemInfo (OS, nvResult, svResult);

    if (nvResult =  IS_WINDOWSNT) then
        // Windows NT を実行中です。
        bWinNT = TRUE;

        // サービスパック 3 がインストールされているか確認します。
        nvResult = ISCompareServicePack (SERVICE_PACK);

        if (nvResult < 0) then
            MessageBox (" エラー : ISCompareServicePack が失敗しました。", SEVERE);
        elseif (nvResult = LESS_THAN) then
            MessageBox (" サービスパックがないか次の値を下回っています : "
                    + SERVICE_PACK, INFORMATION);
        elseif (nvResult = EQUALS) then
            MessageBox (" サービスパックは " + SERVICE_PACK, INFORMATION);
        elseif (nvResult = GREATER_THAN) then
            MessageBox (" 次のサービスパック以上です "
                    + SERVICE_PACK, INFORMATION);
        endif;
    else
        MessageBox (" ターゲットシステムは  Windows NT を実行中ではありません。", SEVERE);
    endif;

end;
```

ISDeterminePlatform

ISDeterminePlatform 関数はシステム変数 SYSINFO を設定します。この変数は、メンバーがターゲット コンピューターのオペレーティング プラットフォームについての情報を指定するのに利用される構造化された変数です。ISDeterminePlatform は、セットアップ初期化中にセットアップエンジンが直接呼び出します。

パラメーター

なし

戻り値

なし

IsEmpty

IsEmpty 関数は、VARIANT 変数タイプが初期化されたかどうかを確認します。

OBJECT 変数タイプが有効なオブジェクトに参照を割り当てたかどうか確認する場合は、IsObject を呼び出してください。

構文

IsEmpty (vVariant);

パラメーター

テーブル 10 - IsEmpty のパラメーター

パラメーター	説明
vVariant	確認する変数を指定します。

戻り値

テーブル 11 - IsEmpty の戻り値

戻り値	説明
TRUE (1)	vVariant が初期化されました。
FALSE (0)	vVariant が初期化されませんでした。

IsEmpty の例

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * IsEmpty 関数のデモンストレーションを行います。
 *
¥*---------------------------------------------------------------*/

function OnBegin()
    VARIANT vVariant;
    BOOL bEmpty;
    STRING szString;
begin
    bEmpty = IsEmpty ( vVariant );
    /* "vVariant is が空です。" メッセージが表示されます。*/
    if bEmpty then
        MessageBox ( "vVariant が空です。", INFORMATION );
    else
        MessageBox ( "vVariant が空ではありません。", INFORMATION );
    endif;

    /* vVariant を初期化します。*/
    vVariant = szString;

    bEmpty = IsEmpty ( vVariant );
    /* "vVariant is が空ではありません。" メッセージが表示されます。*/
    if bEmpty then
```

```
        MessageBox ( "vVariant が空です。", INFORMATION );
    else
        MessageBox ( "vVariant が空ではありません。", INFORMATION );
    endif;

    /* サンプルセットアップスクリプトを終了します。*/
    abort;
end;
```

IsObject

IsObject 関数は、OBJECT タイプの変数が有効なオブジェクトにリファレンスを割り当てたかどうかを CreateObject 関数または GetObject 関数を使用して確認します。

構文

```
IsObject ( oObject );
```

パラメーター

テーブル 12 - IsObject のパラメーター

パラメーター	説明
oObject	確認する変数を指定します。

戻り値

テーブル 13 - IsObject の戻り値

戻り値	説明
TRUE	oObject によって、有効なオブジェクトに参照が割り当てられています。
FALSE	oObject によって、有効なオブジェクトに参照が割り当てられていません。

LaunchApp

LaunchApp 関数を使うと、スクリプト内から別のアプリケーションを起動できます。**LaunchApp** は LaunchAppAndWait(szCommand, szCmdLine, LAAW_OPTION_NOWAIT) を呼び出します。**LaunchApp** パラメーターと戻り値については、「LaunchAppAndWait」を参照してください。

LaunchApp の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* LaunchApp 関数のデモンストレーションを行います。
*
* LaunchApp が呼び出されて、アプリケーションが実行されます。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*      ターゲットシステム上の Windows Notepad 実行可能ファイルの
*      完全修飾名と有効なテキスト ファイルを
*      参照するように設定してください。
*
¥*---------------------------------------------------------------------*/

#define APPLICATION WINDIR^"Notepad.exe"
#define CMD_LINE    WINDIR^"Readme.txt"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_LaunchApp(HWND);

function ExFn_LaunchApp(hMSI)
begin

   // Windows Notepad アプリケーションを起動して
   // Windows Readme.txt ファイルを編集します。
   if (LaunchApp (APPLICATION, CMD_LINE) < 0) then
      MessageBox (APPLICATION+" を起動できませんでした。", SEVERE);
   endif;

end;
```

LaunchAppAndWait

LaunchAppAndWait 関数を使うと、スクリプト内から別のアプリケーションを起動できます。**LaunchAppAndWait** は次を呼び出します :

```
LaunchApplication( szProgram, szCmdLine, "", LAAW_STARTUPINFO.wShowWindow, LAAW_PARAMETERS.nTimeOut, nOptions | LAAW_OPTION_CHANGEDIRECTORY | LAAW_OPTION_FIXUP_PROGRAM );
```

LaunchAppAndWait のパラメーターと戻り値については、「LaunchApplication」を参照してください。

構文

```
LaunchAppAndWait ( szProgram, szCmdLine, nOptions );
```

LaunchAppAndWait の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* LaunchAppAndWait 関数のデモンストレーションを行います。
*
* このスクリプトはユーザーに対して 3 つのオプションを提供します :
*
* -- Notepad の起動。Notepad が閉じた後、セットアップを続行
* -- Notepad を起動後、すぐセットアップを続行
* -- インストールの中止
*
* ユーザーが最初のオプションを選択した場合、インストールは Notepad を起動し、
* それが閉じられるまで待機してから続行します。
* ユーザーが 2 番目のオプションを選択した場合、インストールは Notepad を起動し、
* 直後にスクリプトを実行します。
* ユーザーが 3 番目のオプションを選択した場合、
* インストールが終了します。
*
¥*---------------------------------------------------------------------*/

#define PROGRAM          WINDIR^"NotePAD.EXE"
#define LAUNCH_WAIT_TEXT "Notepad の起動。Notepad を閉じた後、セットアップを続行 "
#define LAUNCH_GO_TEXT   "Notepad の起動。Notepad を閉じた後、すぐセットアップを続行 "
#define EXIT_TEXT        " インストールの中止 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_LaunchAppAndWait(HWND);

function ExFn_LaunchAppAndWait(hMSI)
   STRING szProgram, szCmdLine, szMsg;
   BOOL bLaunchAndGo, bLaunchAndWait, bExit;
   NUMBER nWait;
begin

   // 通常ウィンドウでインストールを実行します。
   Enable (BACKGROUND);
   Enable (DEFWINDOWMODE);

   // インストールダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   // ユーザーからオプションを取得します。
   AskOptions (EXCLUSIVE, " テスト ",
           LAUNCH_WAIT_TEXT, bLaunchAndWait,
           LAUNCH_GO_TEXT, bLaunchAndGo,
           EXIT_TEXT, bExit);
```

```
    if !bExit then

        // 変数を LaunchAppAndWait に渡して、
        // 待機するか否かを示します。
        if bLaunchAndWait then
            nWait = WAIT;
        else
            nWait = NOWAIT;
        endif;

        // Notepad を起動します。nWait の値が
        // いつインストールを続行するのかを決定します。
        if (LaunchAppAndWait (PROGRAM, "", nWait) < 0) then
            MessageBox (PROGRAM +" を起動することができませんでした。",SEVERE);
        endif;

        MessageBox (" セットアップを終了します。", INFORMATION);

    endif;

end;
```

LaunchAppAndWaitInitStartupInfo

LaunchAppAndWaitInitStartupInfo 関数は LAAW_STARTUPINFO および LAAW_PARAMETERS システム変数を適切なデフォルト値に初期化します。この関数はインストール初期化中に自動的に呼び出されます。

構文

```
LaunchAppAndWaitInitStartupInfo ( );
```

システム変数

テーブル 14 ・ LaunchAppAndWaitInitStartupInfo のシステム変数

システム変数	デフォルト値
LAAW_STARTUPINFO.cb	SizeOf(LAAW_STARTUPINFO)
LAAW_STARTUPINFO.lpReserved	NULL
LAAW_STARTUPINFO.lpDesktop	NULL
LAAW_STARTUPINFO.lpTitle	NULL
LAAW_STARTUPINFO.wShowWindow	SW_SHOWDEFAULT
LAAW_STARTUPINFO.lpReserved2	NULL
LAAW_STARTUPINFO.cbReserved2	0
LAAW_STARTUPINFO.dwFlags	STARTF_USESHOWWINDOW
LAAW_PARAMETERS.szStatusText	""
LAAW_PARAMETERS.szCommandLineResult	""
LAAW_PARAMETERS.lpProcessAttributes	NULL
LAAW_PARAMETERS.lpThreadAttributes	NULL
LAAW_PARAMETERS.bInheritHandles	FALSE
LAAW_PARAMETERS.dwCreationFlags	NORMAL_PRIORITY_CLASS
LAAW_PARAMETERS.lpEnvironment	NULL
LAAW_PARAMETERS.lpCurrentDirectory	NULL
LAAW_PARAMETERS.nCallbackInterval	1000
LAAW_PARAMETERS.nLaunchResult	0

パラメーター

なし。

戻り値

テーブル 15 - LaunchAppAndWaitInitStartupInfo の戻り値

戻り値	説明
ISERR_SUCCESS	この関数は常に ISERR_SUCCESS を戻します。

LaunchApplication

UninstallApplication 関数は指定されたアプリケーションを起動し、オプションでそれを待機します。

LaunchApplication は、Windows API 関数 CreateProcess または Windows API 関数 ShellExecuteEx (nOptions が LAAW_OPTION_USE_SHELLEXECUTE を含む場合) のどちらかを使用して指定されたアプリケーションを起動します。アプリケーションが起動されたあと、LAAW_OPTION_WAIT または LAAW_OPTION_WAIT_INCL_CHILD オプションが指定されている場合、インストールは WaitForApplication を呼び出してアプリケーションが終了するのを待機します。処理の完了、または指定されたタイムアウトの値が経過した時点で、インストールは続行されます。

構文

```
LaunchApplication( byval string szProgram, byval string szCmdLine, byval string szDirectory, byval number nShowWindow, byval number
    nTimeOut, byval number nOptions );
```

パラメーター

テーブル 16 · LaunchApplication のパラメーター

パラメーター	説明
szProgram	起動されるアプリケーションの完全パスとファイル名を指定します。 nOptions に LAAW_OPTION_USE_SHELLEXECUTE が含まれていない場合、代わりに szCmdLine パラメーターで、起動するアプリケーションのファイル名を指定することができます。その場合、szProgram パラメーターにヌル文字列 ("") を渡します。 szProgram パラメーターは、LAAW_OPTION_USE_SHELLEXECUTE オプションが使用されるときのみ URL をサポートします。
szCmdLine	起動したアプリケーションに渡すコマンドラインのパラメーターを指定します。コマンドラインのパラメーターを使わずにアプリケーションを起動するには、にヌル文字列 ("") を渡します。 nOptions に LAAW_OPTION_USE_SHELLEXECUTE が含まれていない場合、szCmdLine パラメーターで起動するアプリケーションのファイル名を指定することもできます。その場合、次の操作を必ず行ってください。 · アプリケーションへのパスとコマンドラインの間をスペースで区切ります。 · アプリケーションの完全修飾パスに長いフォルダー名や長いファイル名が含まれる場合、szCmdLine へ追加する前に LongPathToQuote に渡します。例 : szApplicationPath = WINDIR ^ "Notepad.exe"; szApplicationCmdLine = SRCDIR ^ "Readme.txt"; LongPathToQuote(szApplicationPath, TRUE); szCmdLine = szApplicationPath + " " + szApplicationCmdLine;
szDirectory	アプリケーションの作業ディレクトリを指定します。空の文字列 ("") が指定されている場合 (および LAAW_OPTION_USE_SHELLEXECUTE が指定されていない場合)、関数は空文字列の代わりに NULL を **CreateProcess** に渡します。

テーブル 16 · LaunchApplication のパラメーター (続き)

パラメーター	説明
nShowWindow	アプリケーションの初期ウィンドウの状態を STARTUPINFO 構造の wShowWindow パラメーターで **CreateProcess** に渡される状態で、または SHELLEXECUTEINFO 構造 の nShow パラメーターで **ShellExecuteEx** に渡される状態 (LAAW_OPTION_USE_SHELLEXECUTE が指定されている場合) で指定します。前期の関数について、任意の文書化されている値を指定することができます。いくつかの値は **ISRTWindows.h** であらかじめ定義されています : · SW_FORCEMINIMIZE · SW_HIDE · SW_MAX · SW_MAXIMIZE · SW_MINIMIZE · SW_NORMAL · SW_RESTORE · SW_SHOW · SW_SHOWDEFAULT · SW_SHOWMAXIMIZED · SW_SHOWMINIMIZED · SW_SHOWMINNOACTIVE · SW_SHOWNA · SW_SHOWNOACTIVATE · SW_SHOWNORMAL
nTimeOut	**WaitForApplication** 関数に渡される nTimeOut 値 (ミリ秒) を指定します。LAAW_OPTION_WAIT を指定しないと、指定された値は無視されますので注意してください。

テーブル 16 • LaunchApplication のパラメーター (続き)

パラメーター	説明
nOptions	インストールが続行する前に、起動されたアプリケーションが終了するまで待機するかどうかを含む様々なオプションを指定します。 このパラメーターに、あらかじめ定義されている以下の定数のうちの 1 つを渡します。次の例外を除き、ビット単位 OR 演算子 (|) を利用してこれらの定数を組み合わせることができます : LAAW_OPTION_NOWAIT を LAAW_OPTION_WAIT と組み合わせることはできません。また LAAW_OPTION_HIDDEN、LAAW_OPTION_MINIMIZED、および LAAW_OPTION_MAXIMIZED を組み合わせることはできません。 • LAAW_OPTION_USE_SHELLEXECUTE Ñ 関数が **CreateProcess** ではなく Windows API 関数 **ShellExecuteEx** を呼び出して、アプリケーションを起動することを示します。 • LAAW_OPTION_NOWAIT Ñ 関数で **WaitForApplication** に渡されます。このパラメーターを使用することは、関数 LaunchApp の呼び出しと同じ効果を持ちます。 • LAAW_OPTION_WAIT Ñ 関数で **WaitForApplication** に渡されます。 • LAAW_OPTION_USE_CALLBACK Ñ 関数で **WaitForApplication** に渡されます。 • LAAW_OPTION_SET_BATCH_INSTALL Ñ たとえば RunOnce キーの更新など、起動されたアプリケーションによってシステムの再起動を必要とする変更が発生する場合、関数が内部的に BATCH_INSTALL をゼロ以外の値に設定することを指定します。関数は、指定されたアプリケーションを起動する前にシステムの状態を確認してこれを判断し、そしてアプリケーションを起動した後 (そして LAAW_OPTION_WAIT が指定されている場合はアプリケーションが完了するのを待って) 現在のシステムの状態を比較します。このフラグはシステム再起動の必要性を通知するステータス情報を戻さないサードパーティ インストールを起動したときに利用することができます。 LAAW_OPTION_SET_BATCH_INSTALL は InstallScript プロジェクトのイベント ドリブン型コードおよび InstallScript MSI プロジェクトで使用できます。InstallScript カスタム アクションには使用できません。 • LAAW_OPTION_SHOW_HOURGLASS– 起動されたアプリケーションが実行中はカーソルを砂時計に変更することを指定します。 • LAAW_OPTION_WAIT_INCL_CHILD– 関数で **WaitForApplication** に渡されます。

テーブル 16 ・ LaunchApplication のパラメーター (続き)

パラメーター	説明
nOptions (続き)	 ***メモ ・** LAAW_OPTION_WAIT_INCL_CHILD が使用されたとき、関数は、起動されたプロセスの直接の子プロセスのみ検出し待機します。初回で起動されたプロセスの子プロセスによって起動されたその他の子プロセスは検出 / 待機されません。* *LAAW_OPTION_USE_SHELLEXECUTE を LAAW_OPTION_WAIT_INCL_CHILD と共に使用する方法については、「追加情報」をご覧ください。* このパラメーターで次のオプションもサポートされていますが、これらは推奨されていません : ・ LAAW_OPTION_HIDDEN— 起動されたアプリケーションのメインウィンドウが初めは非表示であることを指定します。 ・ LAAW_OPTION_MINIMIZED— 起動されたアプリケーションのメイン ウィンドウが初めは最小化されていることを指定します。 ・ LAAW_OPTION_MAXIMIZED— 起動されたアプリケーションのメイン ウィンドウが初めは最大化されていることを指定します。 ・ LAAW_OPTION_NO_CHANGEDIRECTORY— 現在使用されていません。このパラメーターは無視されます。 ・ LAAW_OPTION_CHANGEDIRECTORY— LaunchApplication がインストールの現在のディレクトリを、この後すぐ起動されるアプリケーションのディレクトリに一時的に更新することを指定します。デフォルトでは、LaunchApplication はインストールの現在のディレクトリを変更しません。LaunchApplication は、LaunchApplication が戻る前に、インストールの現在のディレクトリを元の値にリセットします。  ***メモ ・** LaunchApplication の代わりに LaunchAppAndWait を呼び出すと、このオプションが自動的に含められます。* ・ LAAW_OPTION_FIXUP_PROGRAM— CreateProcess または ShellExecuteEx の呼び出しが適切に動作するために、LaunchApplication が szProgram で LongPathFromShortPath を呼び出すことを指定します。ほとんどの場合、これは必要ありません。  ***メモ ・** LaunchApplication の代わりに LaunchAppAndWait を呼び出すと、このオプションが自動的に含められます。*

戻り値

テーブル 17 ・ LaunchApplication の戻り値

戻り値	説明
ISERR_SUCCESS	アプリケーションが正常に起動したことを示します。
< ISERR_SUCCESS	アプリケーションが正常に起動しなかったことを示します。 ***メモ ・*** *アプリケーションが起動されない場合、* ***CreateProcess*** *または* ***ShellExecuteEx*** *の呼び出しのあと、LAAW_PARAMETER システム変数の nLaunchResult メンバーに Windows API 関数* ***GetLastError*** *の呼び出し結果が含まれます。関数が成功し、LAAW_OPTION_WAIT オプションが指定された場合、LAAW_PARAMETERS システム変数の nLaunchResult メンバーは起動されたアプリケーションが戻すコードを含みます。*

追加情報

- LAAW_OPTION_USE_SHELLEXECUTE を使用しない場合、**LaunchApplication** を呼び出す前に LAAW_STARTUPINFO システム変数を更新して、アプリケーションの動作をカスタマイズします。

  LAAW_OPTION_USE_SHELLEXECUTE を使用する場合、LAAW_SHELLEXECUTEINFO 構造をカスタマイズして、アプリケーションの動作をカスタマイズします。

  Windows Vista 以降を実行中のシステム上で LAAW_OPTION_USE_SHELLEXECUTE を使用する場合で、完全な管理者アカウント (実行する実行可能ファイルを右クリックして [管理者として実行] をクリックした場合と似ています) を使ってアプリケーションを起動するとき、スクリプトで **LaunchApplication** を使用する前に LAAW_SHELLEXECUTEVERB を **runas** に設定します :

  ```
  LAAW_SHELLEXECUTEVERB = "runas";
  ```

  これによって、起動するアプリケーションが関連設定を含むアプリケーション マニフェストを持っているかどうかにかかわらず、アプリケーションは確実に完全な管理者権限を使って実行されます。これによって、同意または資格情報を要求するユーザー アカウント制御 (UAC) のプロンプトが表示されることがあります。

  Windows Vista 以前のオペレーティング システムを実行するマシン上で **runas** が使用された場合、[別のユーザーとして実行] ダイアログ ボックスが表示されます。この動作は、実行する実行可能ファイルを右クリックして [別のユーザーとして実行] をクリックしたときと似ています。このダイアログ ボックスを使って、エンド ユーザーはアプリケーションを実行するために使用するユーザー アカウントを選択できます。

- 起動された処理についての ID 情報を取得するには、LAAW_PROCESS_INFORMATION システム変数または LAAW_SHELLEXECUTEINFO システム変数 (LAAW_OPTION_USE_SHELLEXECUTE を使用する場合) を使用します。

- LAAW_OPTION_USE_SHELLEXECUTE を使用する場合、hProcess メンバーを除き LAAW_PROCESS_INFORMATION は使用されません。hProcess メンバーは、(LAAW_SHELLEXECUTEINFO.hProcess メンバーから) 起動されたプロセスのプロセス ハンドルに更新されます。

- **ShellExecuteEx** は、起動されたプロセスのプロセス ID を返しません。このため、LAAW_OPTION_USE_SHELLEXECUTE が使用された場合、LAAW_OPTION_WAIT_INCL_CHILD オプションは、プロ

セス ID を判別するための Windows API **GetProcessId** が使用可能な場合のみ有効です。Windows API のドキュメントによると、**GetProcessId** は Windows XP SP1 以降および Windows Server 2003 以降で使用可能です。この API が使用できない場合、LAAW_OPTION_WAIT_INCL_CHILD は LAAW_OPTION_WAIT と同じ動作をします。

- インストールが、CD または DVD などのリムーバブル メディアから呼び出されたとき、Disk1 の **Setup.exe** ファイルは、インストール中、常に使用できる状態にあるとはかぎりません。(実行中に、**Setup.exe** が使用できなくなった場合、オペレーティング システムでプロンプトが表示され、エンドユーザーに正しいディスクを挿入するように求めてくることがあります。これにより、インストールが失敗することがあります。) したがって、この問題を回避するために、**Setup.exe** ファイルが Temp フォルダーにコピーされ、インストールがそこから再起動されます。元の **Setup.exe** は、ここで終了します。ただし、この処理が発生すると、**LaunchApplication** は、インストールが完了したと仮定して動作し、待機しません。

  この問題を回避するため、子インストールを起動するときに、/clone_wait パラメーターを使用できます。その場合、起動されたインストールで、起動された元のプロセスが実行中のまま保たれ、親インストールは待機します。ただし、このワークアラウンドでは、**Setup.exe** を含む元の CD がインストールの途中で使用できない状態になる場合、問題が発生する場合があります。この例として、インストールの途中で一枚目の CD が使用できなくなる複数 CD のインストールがあげられます。

  この問題を回避する唯一の方法は、起動されたプロセスの子プロセスの ID を判別するコードを追加して、子プロセスが完了するのを待機することです。

LaunchApplicationInit

LaunchApplicationInit 関数は LAAW_STARTUPINFO および LAAW_PARAMETERS システム変数を適切なデフォルト値に初期化します。この関数はインストール初期化中に自動的に呼び出されます。この関数は LaunchAppAndWaitInitStartupInfo 関数に優先します。

構文

```
LaunchApplicationInit( );
```

システム変数

テーブル 18 ・ LaunchApplicationInit のシステム変数

システム変数	デフォルト値
LAAW_PARAMETERS.bCallbackEndedWait	FALSE
LAAW_PARAMETERS.bInheritHandles	FALSE
LAAW_PARAMETERS.dwCreationFlags	NORMAL_PRIORITY_CLASS
LAAW_PARAMETERS.lpCurrentDirectory	NULL
LAAW_PARAMETERS.lpEnvironment	NULL
LAAW_PARAMETERS.lpProcessAttributes	NULL
LAAW_PARAMETERS.lpThreadAttributes	NULL
LAAW_PARAMETERS.nCallbackInterval	1000
LAAW_PARAMETERS.nLaunchResult	0
LAAW_PARAMETERS.nTimeOut	INFINITE
LAAW_PARAMETERS.nTimeOutCheckInterval	1000
LAAW_PARAMETERS.nWaitForInputIdleMax	2000
LAAW_PARAMETERS.nWaitResult	WAIT_OBJECT_0
LAAW_PARAMETERS.szCommandLineResult	""
LAAW_PARAMETERS.szStatusText	""
LAAW_SHELLEXECUTEINFO.cbSize	SizeOf(LAAW_SHELLEXECUTEINFO)
LAAW_SHELLEXECUTEINFO.dwHotKey	0
LAAW_SHELLEXECUTEINFO.fMask	SEE_MASK_NOCLOSEPROCESS | SEE_MASK_FLAG_NO_UI
LAAW_SHELLEXECUTEINFO.hIconMonitor	NULL
LAAW_SHELLEXECUTEINFO.hInstApp	NULL
LAAW_SHELLEXECUTEINFO.hkeyClass	NULL
LAAW_SHELLEXECUTEINFO.hProcess	NULL

テーブル 18 • LaunchApplicationInit のシステム変数 (続き)

システム変数	デフォルト値
LAAW_SHELLEXECUTEINFO.hwnd	GetWindowHandle(HWND_INSTALL)
LAAW_SHELLEXECUTEINFO.lpClass	NULL
LAAW_SHELLEXECUTEINFO.lpDirectory	NULL
LAAW_SHELLEXECUTEINFO.lpFile	NULL
LAAW_SHELLEXECUTEINFO.lpIDList	NULL
LAAW_SHELLEXECUTEINFO.lpParameters	NULL
LAAW_SHELLEXECUTEINFO.lpVerb	&LAAW_SHELLEXECUTEVERB
LAAW_SHELLEXECUTEINFO.nShow	SW_SHOWDEFAULT
LAAW_SHELLEXECUTEVERB	"open"
LAAW_STARTUPINFO.cb	SizeOf(LAAW_STARTUPINFO)
LAAW_STARTUPINFO.cbReserved2	0
LAAW_STARTUPINFO.dwFlags	STARTF_USESHOWWINDOW
LAAW_STARTUPINFO.lpDesktop	NULL
LAAW_STARTUPINFO.lpReserved	NULL
LAAW_STARTUPINFO.lpReserved2	NULL
LAAW_STARTUPINFO.lpTitle	NULL
LAAW_STARTUPINFO.wShowWindow	SW_SHOWDEFAULT

パラメーター

なし。

戻り値

テーブル 19 - LaunchApplicationInit の戻り値

戻り値	説明
ISERR_SUCCESS	この関数は常に ISERR_SUCCESS を戻します。

ListAddItem

ListAddItem 関数は現在の要素の前後に番号の要素を追加します。

メモ - *ListAddItem は番号リストでのみ使用できます。*

タスク ***リストをスキャンするには、以下の手順を実行します。***

1. ListGetFirstItem を呼び出してリストの最初の要素を取得します。
2. リストの終わりに到達するまで ListGetNextItem を繰り返し呼び出します。

リストの特定の要素を現在の要素にするには、ListSetIndex を呼び出します。

構文

```
ListAddItem (listID, nItem, nPlacementFlag);
```

パラメーター

テーブル 20 ・ ListAddItem のパラメーター

パラメーター	説明
listID	番号リスト名を指定します。listID によって識別されるリストは、ListCreate への呼び出しによって既に初期化されている必要があります。
nItem	リストに追加する数値の要素を指定します。
nPlacementFlag	現在の要素と相対した nItem の配置場所を指定します。新しい要素は現在の要素の前か後に置かれます。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ AFTERÑ リストの現在の要素の後に新しい要素を追加します。 ・ BEFOREÑ リストの現在の要素の前に新しい要素を追加します。

戻り値

テーブル 21 ・ ListAddItem の戻り値

戻り値	説明
> = ISERR_SUCCESS (0)	関数は成功しました。
< ISERR_SUCCESS (0)	関数は成功しませんでした。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

ListAddItem の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListAddItem 関数のデモンストレーションを行います。
*
* まず、ListCreate への呼び出しで空白リストが作成されます。
* そして ListAddItem が 3 回呼び出され、リストへ番号 1、2、そして 3 が
* 追加されます。番号は昇順に追加されませんが、
* ListAddItem への各呼び出しの
* 3 番目のパラメーターで渡される
* 配置フラグによって、順に並べられます。リストが
* ビルドされた後、それが表示されます。最後に、現在の要素が表示
```

```
＊されます。
＊
¥*---------------------------------------------------------------------*/

#define TITLE "ListAddItem の例 "
#define MSSG  " 次は追加される項目のリストです :¥n¥n"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListAddItem(HWND);

function ExFn_ListAddItem(hMSI)
   NUMBER nvItem, nvResult;
   LIST   listID;
   STRING szMsg, svItem;
begin

   // 空白数値リストを作成します。
   listID = ListCreate(NUMBERLIST);

   // エラーが発生した場合、それをレポートして終了させます。
   if (listID = LIST_NULL) then
      MessageBox (" リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // 番号 1 をリストへ追加します。
   if (ListAddItem (listID, 1, AFTER) < 0) then
      MessageBox ("ListAddItem への最初の呼び出しに失敗しました。", INFORMATION);
      abort;
   endif;

   // 整数 1、位置 1、が現在の要素です。
   // 現在の要素の後に番号 3 を追加します。
   if (ListAddItem (listID, 3, AFTER) < 0) then
      MessageBox ("ListAddItem への 2 回目の呼び出しに失敗しました。", INFORMATION);
      abort;
   endif;

   // 整数 3、位置 2、が現在の要素です。
   // 番号 2 をリストへ追加します。
   if (ListAddItem (listID, 2, BEFORE) < 0) then
      MessageBox ("ListAddItem への 3 回目の呼び出しに失敗しました。", INFORMATION);
      abort;
   endif;

   // 現在の要素を読み出して表示します。
   ListCurrentItem (listID, nvItem);

   SprintfBox (INFORMATION, TITLE, " 現在の項目 : %d", nvItem);

      // 番号を報告するメッセージのビルドを開始します。
      szMsg = MSSG;

      // 番号を取得してメッセージへそれらを追加します。
      nvResult = ListGetFirstItem (listID, nvItem);
      while (nvResult != END_OF_LIST)
         NumToStr (svItem, nvItem);
```

```
            szMsg = szMsg + svItem + " ";
            nvResult = ListGetNextItem (listID, nvItem);
        endwhile;

        // 番号を表示します。
        MessageBox (szMsg, INFORMATION);

    // メモリからリストを削除します。
    ListDestroy(listID);

end;
```

ListAddList

ListAddList 関数は、1 つめのリスト (listAdd) から目的のリスト (listDest) に要素を追加します。この関数は、文字列リストと数値リストの両方で使用できますが、listDest および listAdd で一致しなくてはなりません。

構文

```
ListAddList (listDest, listAdd, nPlacementFlag);
```

パラメーター

テーブル 22 - ListAddList のパラメーター

パラメーター	説明
listDest	目的のリストを指定します。**ListAddList** は、このリストの現在の要素の直前または直後に新しい要素を追加します。
listAdd	目的のリストに追加する要素を含むリストを指定します。**ListAddList** は、すべての要素を ListDest に (現在の要素から、このリストの最後の要素に) 追加します。
nPlacementFlag	**ListAddList** がリスト要素を listDest の現在の要素の前または後のどちらに追加するのかを指定します。次の値が可能です : • BEFORE • AFTER

戻り値

テーブル 23 - ListAddList の戻り値

戻り値	説明
ISERR_SUCCESS	関数が成功したことを示します。
< ISERR_SUCCESS	関数の実行に失敗したことを示します。

ListAddString

ListAddString 関数は文字列リストの現在の要素の前後に文字列の要素を追加します。

メモ - *ListAddString は文字列リスト でのみ使用できます。*

タスク ***リストをスキャンするには、以下の手順を実行します。***

1. ListGetFirstItem を呼び出してリストの最初の要素を取得します。
2. リストの終わりに到達するまで ListGetNextItem を繰り返し呼び出します。

リストの特定の要素を現在の要素にするには、ListSetIndex を呼び出します。

構文

ListAddString (listID, szString, nPlacementFlag);

パラメーター

テーブル 24 · ListAddString のパラメーター

パラメーター	説明
listID	文字列リスト名を指定します。listID によって識別されるリストは、ListCreate への呼び出しによって既に初期化されている必要があります。
szString	リストに追加する文字列の要素を指定します。
nPlacementFlag	現在の要素と相対した szString の配置場所を指定します。新しい文字列は現在の要素の前か後に置かれます。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · AFTER Ñ リストの現在の要素の後に新しい文字列を追加します。 · BEFORE Ñ リストの現在の要素の前に新しい文字列を追加します。

戻り値

テーブル 25 · ListAddString の戻り値

戻り値	説明
> = ISERR_SUCCESS (0)	関数は成功しました。
< ISERR_SUCCESS (0)	関数は成功しませんでした。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

ListAddString の例

***メモ·** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListAddString 関数のデモンストレーションを行います。
*
* ListAddString を呼び出し、文字列リストへ文字列を追加します。
* そして、再び呼び出して最初の文字列のあとに別の文字列を追加
* します。この文字列はまた、現在の要素として設定されます。
* ListAddString の 3 回目の呼び出しでは、現在の要素の前に文字列を
* 追加します。
*
¥*---------------------------------------------------------------*/
```

```
#define  TITLE_TEXT "ListAddString の例 "
#define  MSG_TEXT   " ハードウェア機器 :"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListAddString(HWND);

function ExFn_ListAddString(hMSI)
  STRING szString, svString;
   LIST   listID;
begin

   // 空白文字列リストを作成します。
   listID = ListCreate (STRINGLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox (" リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // リストへ文字列を追加する。この文字列が現在の要素となります。
   szString = " キーボード ";
   if (ListAddString (listID, szString, AFTER) < 0) then
      MessageBox ("ListAddString が失敗しました。", INFORMATION);
   endif;

   // 2 番目の文字列を追加します。現在の要素の後に挿入します。
   // そしてこの文字列が現在の要素となります。
   szString = " マウス ";
   if (ListAddString (listID, szString, AFTER) < 0) then
      MessageBox ("ListAddString が失敗しました。", INFORMATION);
   endif;

   // 3 番目の文字列を追加します。現在の要素の前に挿入します。
   szString = " モニター ";

   if (ListAddString (listID, szString, BEFORE) < 0) then
      MessageBox ("ListAddString が失敗しました。", INFORMATION);
   endif;

   // 文字列のリストを表示します。
   SdShowInfoList (TITLE_TEXT, MSG_TEXT, listID);

   // 現在の要素を読み出して表示します。
   ListCurrentString (listID, svString);

   MessageBox (svString, INFORMATION);

   // メモリからリストを削除します。
   ListDestroy(listID);

end;
```

ListAppendFromArray

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ListAppendFromArray 関数は、varSource によって指定された配列内の要素を listResult によって指定されたリストに追加します。

構文

ListAppendFromArray (listResult, varSource, bString);

パラメーター

テーブル 26 ・ ListAppendFromArray のパラメーター

パラメーター	説明
listResult	リスト名を指定します。listResult によって識別されるリストは、ListCreate への呼び出しによって既に初期化されている必要があります。
varSource	リストに追加する配列を指定します。
bString	varSource が文字配列で listResult が文字列リストの場合は、このパラメーターを TRUE に設定してください。VarSource が数値配列で listResult が番号リストの場合は、このパラメーターを FALSE に設定してください。

戻り値

テーブル 27 ・ ListAppendFromArray の戻り値

戻り値	説明
>= 0	リストの新しい要素数。
< ISERR_SUCCESS	この関数が、リストに配列を追加できなかったことを示します。

Comments

配列とリストタイプは一致する必要があります (つまり両方が文字列または両方が数値)。一致しないとこの関数は失敗します。必要であれば、ListAppendToArray を呼び出す前に、ListConvertNumToStr か ListConvertStrToNum を呼び出して、データを正しく変換してください。

ListAppendToArray

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ListAppendToArray 関数は、listSource によって指定されたリスト内の要素を、varResult によって指定された配列に追加し、新しい要素を保存するように配列のサイズを正しく変更します。

構文

ListAppendToArray (varResult, listSource, bString);

パラメーター

テーブル 28 - ListAppendToArray のパラメーター

パラメーター	説明
varResult	配列を指定します。
listSource	配列に追加するリストを指定します。listSource によって識別されるリストは、ListCreate への呼び出しによって既に初期化されている必要があります。
bString	varResult が文字配列で listSource が文字列リストの場合は、このパラメーターを TRUE に設定してください。varResult が数値配列で listSource が番号リストの場合は、このパラメーターを FALSE に設定してください。

戻り値

テーブル 29 - ListAppendToArray の戻り値

戻り値	説明
>= 0	配列の新しい要素数。
< ISERR_SUCCESS	この関数が、配列にリストを追加できなかったことを示します。

Comments

配列とリストタイプは一致する必要があります (つまり両方が文字列または両方が数値)。一致しないとこの関数は失敗します。必要であれば、ListAppendToArray を呼び出す前に、ListConvertNumToStr か ListConvertStrToNum を呼び出して、データを正しく変換してください。

現在の配列サイズによって、配列の最後と新しい要素の配置場所が決定されます。たとえば、配列サイズが 10 で、1 つの要素だけに文字列が含まれている場合でも、新しいデータは 11 番目以降の要素として追加されます。オートサイズされた配列を使用した場合、新しい要素は配列の最後に追加されます。

ListConvertNumToStr

プロジェクト - この情報は、InstallScript プロジェクトに適用します。

ListConvertNumToStr 関数は、listNumber の各数値要素を同等の文字列値に変換して listString に追加します。

構文

ListConvertNumToStr (listString, listNumber);

パラメーター

テーブル 30 ・ ListConvertNumToStr のパラメーター

パラメーター	説明
listString	変換された数値を追加する文字列リストを指定します。listString によって識別されるリストは、ListCreate への呼び出しによって既に初期化されている必要があります。
listNumber	番号リストを指定します。listNumber によって識別されるリストは、**ListCreate** への呼び出しによって既に初期化されている必要があります。

戻り値

テーブル 31 ・ ListConvertNumToStr の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数が数値リストを文字列リストに変換したことを示します。
< ISERR_SUCCESS	関数が数値リストを文字列リストに変換できなかったことを示します。

Comments

関数は NumToStr を使用して、数値の要素を文字列に変換します。listNumber の要素の変換が失敗すると、ヌル文字列 ("") がその要素の listString に追加されます。

ListConvertStrToNum

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ListConvertStrToNum 関数は、listString の各文字列要素を同等の数値に変換して listNumber に追加します。

構文

ListConvertStrToNum (listNumber, listString);

パラメーター

テーブル 32 - ListConvertStrToNum のパラメーター

パラメーター	説明
listNumber	変換された文字列を追加する文字列数値リストを指定します。listNumber によって識別されるリストは、ListCreate への呼び出しによって既に初期化されている必要があります。
listString	文字列リストを指定します。listString によって識別されるリストは、ListCreate への呼び出しによって既に初期化されている必要があります。

戻り値

テーブル 33 - ListConvertStrToNum の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数が文字列リストを数値リストに変換したことを示します。
< ISERR_SUCCESS	関数が文字列リストを数値リストに変換できなかったことを示します。

Comments

関数は StrToNum を使用して、文字列を数値に変換します。listString の要素の変換が失敗すると、0 がその要素の listNumber に追加されます。

ListCount

ListCount 関数はリストの要素数を戻します。

メモ - *この関数は文字列リストでも数値リストでも使用できます。*

構文

```
ListCount (listID);
```

パラメーター

テーブル 34 - ListCount のパラメーター

パラメーター	説明
listID	文字列または番号のリスト名を指定します。

戻り値

テーブル 35 - ListCount の戻り値

戻り値	説明
>= 0	リストの項目数。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。
< 0	ListCount はリストの要素数を決定できませんでした。

ListCount の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListCount 関数のデモンストレーションを行います。
*
* 次のスクリプトは、プログラムフォルダーの名前を
* 文字列リストへ追加し、リストへ文字列の数を表示
* します。
*
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListCount(HWND);

function ExFn_ListCount(hMSI)
   STRING svString;
   LIST   listID;
   NUMBER nCount;
begin

   // 文字列リストを作成します。
   listID = ListCreate (STRINGLIST);
```

```
    // エラーが発生した場合にそれを報告し、終了します。
    if (listID = LIST_NULL) then
        MessageBox ("リストを作成できませんでした。", SEVERE);
        abort;
    endif;

    // リストへプログラムフォルダー名を取得します。
    GetGroupNameList (listID);

    // リスト内のプログラムフォルダーの数を数えます。
    nCount = ListCount (listID);

    // エラーをレポートする、またはフォルダーカウントを表示します。
    if (nCount < 0) then
        MessageBox ("ListCount が失敗しました。", SEVERE);
    else
        SprintfBox (INFORMATION, "ListCount",
                    "%i のプログラムフォルダーが存在します。", nCount);
    endif;

    // メモリからリストを削除します。
    ListDestroy(listID);

end;
```

ListCreate

ListCreate 関数は空白文字列または数値リストを作成します。スクリプト内で任意の数のリストを作成することができます。リストには任意の数の要素を含むことが可能です。唯一の制約は利用可能なメモリの空き容量です。

リスト関数を呼び出すとき、この関数が戻すリストの有効な ID を渡さなくてはなりません。この関数がリスト作成に成功したか否かを確認します。さもなければ、無効なリスト上ですべての関数が失敗することになります。

リストが必要でなくなったとき、ListDestroy 関数を使ってリストを破棄することができます。

各リストには、要素をリストの「現在」の要素として認識するポインターが含まれます。様々なリスト関数がリストの現在の要素を再配置します。

メモ・リストに数値と文字列の両方の要素を含むことはできません。InstallScript は文字列リストと数値リストの処理を行う個別の関数を提供します。数値リストの ID を文字列リスト関数と共に利用すること、または文字列リストの ID を数値リスト関数を共に利用することはできません。数値リストには終わりに“Item”が付くリスト関数を、そして文字列リストには終わりに“String”が付くリスト関数を利用します。

構文

```
ListCreate ( nListType );
```

パラメーター

テーブル 36 ・ ListCreate のパラメーター

パラメーター	説明
nListType	作成するリストの種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ NUMBERLIST Ñ 番号リストを指定します。 ・ STRINGLIST Ñ 文字列リストを指定します。

戻り値

テーブル 37 ・ ListCreate の戻り値

戻り値	説明
ListID	新しく作成された ID、空白リスト。別の InstallScript リスト関数でこのリストを利用する場合は、常にこの ID を利用しなくてはなりません。この変数を確認し、関数が LIST_NULL を戻さないようにします。
LIST_NULL (-1)	InstallShield がリストを作成できなかったことを示します。これはめったに見られない条件で、メモリに関する重大な問題が原因です。このようなメモリ問題がある場合、セットアップを続行するのに困難な状況が予測されます。

追加情報

有効なリスト ID をリストを必要とする関数へ渡す前に、ListCreate を使ってリストをビルドしなくてはなりません。

ListCreate の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListCreate と ListDestroy 関数のデモンストレーションを行います。
*
* 番号付きリストを作成するために ListCreate が呼び出されます。ListDestroy
* が呼び出されて、それを破棄します。
*
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"
```

```
    export prototype ExFn_ListCreate(HWND);

function ExFn_ListCreate(hMSI)
   LIST   listID;
   NUMBER nvItem;
begin

   // 空白数値リストを作成します。
   listID = ListCreate(NUMBERLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox (" リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // 番号 1078 をリストへ追加します。
   ListAddItem (listID, 1078, AFTER);

   // 番号 304 をリストへ追加します。
   ListAddItem (listID, 304, AFTER);

   // リストの現在の項目 （304）を読み出します。
   ListCurrentItem (listID, nvItem);

   // 現在の項目を表示します。
   SprintfBox (INFORMATION, "ListCreate",
         " リスト内の現在の項目 : %d", nvItem);

   // リストの最初の項目 （1078）を読み出します。
   ListGetFirstItem (listID, nvItem);

   // 最初の項目を表示します。
   SprintfBox (INFORMATION, "ListCreate",
         " リスト内の最初の項目 : %d", nvItem);

   // メモリからリストを削除します。
   ListDestroy(listID);

end;
```

ListCurrentItem

ListCurrentItem 関数は、listID で指定された番号リストから現在の要素を取得します。

また、ListGetFirstItem と ListGetNextItem 関数を使用してリストをトラバースし、任意の要素を現在の要素にすることができます。

***メモ・** ListCurrentItem は番号リストでのみ使用できます。*

構文

```
ListCurrentItem (listID, nvItem);
```

パラメーター

テーブル 38 ・ ListCurrentItem のパラメーター

パラメーター	説明
listID	番号リストを指定します。
nvItem	リストの現在の要素の値を返します。

戻り値

テーブル 39 ・ ListCurrentItem の戻り値

戻り値	説明
0	この関数によって、番号リストの現在の要素が正常に取得されたことを示します。
< 0	この関数によって、番号リストの現在の要素が取得できなかったことを示します。
END_OF_LIST (1)	リストが空で現在の要素がないことを示します。
ISERR_LIST_NOSUCHLIST	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

ListCurrentItem の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListCurrentItem 関数のデモンストレーションを行います。
*
* このスクリプトは、3 つの数値をリストへ追加し、現在のリスト項目の値を
* 読み出して表示します。
*
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "ListCurrentItem の例 "


// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListCurrentItem(HWND);
```

```
function ExFn_ListCurrentItem(hMSI)
    STRING szMsg;
    LIST   listID;
    NUMBER nItem;
begin

    // 数値リストを作成します。
    listID = ListCreate(NUMBERLIST);

    // エラーが発生した場合にそれを報告し、終了します。
    if (listID = LIST_NULL) then
        MessageBox (" リストを作成できませんでした。", SEVERE);
        abort;
    endif;

    // 数値 100、200、そして 300 をリストに追加します。
    for nItem = 100 to 300 step 100
        ListAddItem (listID, nItem, AFTER);
    endfor;

    // 数値リストから現在の要素を取得します。
    if (ListCurrentItem (listID, nItem) < 0) then
        MessageBox ("ListCurrentItem が失敗しました。.", SEVERE);
    else
        // 現在の項目の値を報告します。
        szMsg = " リストの現在の要素の値 : %d";
        SprintfBox (INFORMATION, TITLE_TEXT, szMsg, nItem);
    endif;

end;
```

ListCurrentString

ListCurrentString 関数は、listID で指定された文字列リストから現在の要素を取得します。

また、ListGetFirstString と ListGetNextString 関数を使用してリストをトラバースし、任意の要素を現在の要素にすることができます。

メモ・ListCurrentString は文字列リストでのみ使用できます。

構文

ListCurrentString (listID, svString);

パラメーター

テーブル 40 - ListCurrentString のパラメーター

パラメーター	説明
listID	文字列リストを指定します。
svString	リストの現在の要素の値を返します。

戻り値

テーブル 41 - ListCurrentString の戻り値

戻り値	説明
0	この関数によって、文字列リストの現在の要素が正常に取得されたことを示します。
< 0	この関数によって、文字列リストの現在の要素が取得できなかったことを示します。
END_OF_LIST (1)	リストが空で現在の要素がないことを示します。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

ListCurrentString の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListCurrentString 関数のデモンストレーションを行います。
*
* ListAddString は文字列リストから現在の文字列要素を
* 読みだします。
*
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "ListCurrentString の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListCurrentString(HWND);
```

```
function ExFn_ListCurrentString(hMSI)
    STRING szString, svString, szMsg;
    LIST   listID;
begin

    // 文字列リストを作成します。
    listID = ListCreate (STRINGLIST);

    // エラーが発生した場合にそれを報告し、終了します。
    if (listID = LIST_NULL) then
        MessageBox (" リストを作成できませんでした。", SEVERE);
        abort;
    endif;

    // リストへ 3 つの文字列を追加します。
    ListAddString (listID, " 最初の文字列 ", AFTER);
    ListAddString (listID, "2 番目の文字列 ", AFTER);
    ListAddString (listID, "3 番目の文字列 ", AFTER);

    // 文字列リストの現在の要素を取得します。
    if (ListCurrentString (listID, svString) < 0) then
        MessageBox ("DeleteProgramFolder が失敗しました。", SEVERE);
    else
        // 現在の項目の値を報告します。
        szMsg = " リスト内の現在の要素 : '%s'";
        SprintfBox (INFORMATION, TITLE_TEXT, szMsg, svString);
    endif;

end;
```

ListDeleteAll

ListDeleteAll 関数は、指定されたリスト内のすべての要素を削除します。この関数は文字列リストでも番号リストでも使用できます。

構文

```
ListDeleteAll ( list);
```

パラメーター

テーブル 42 - ListDeleteAll のパラメーター

パラメーター	説明
list	すべての要素を削除するように指定されたリスト。

戻り値

テーブル 43 - ListDeleteAll の戻り値

戻り値	説明
ISERR_SUCCESS	関数が成功したことを示します。
< ISERR_SUCCESS	関数の実行に失敗したことを示します。

ListDeleteItem

ListDeleteItem 関数は、listID で指定された番号リストから現在の要素を削除します。

また、ListGetFirstItem と ListGetNextItem 関数を使用してリストをトラバースし、任意の要素を現在の要素にすることができます。

メモ - *ListDeleteItem は番号リストでのみ使用できます。*

構文

```
ListDeleteItem (listID);
```

パラメーター

テーブル 44 • ListDeleteItem のパラメーター

パラメーター	説明
listID	現在の要素を削除する番号リストを指定します。

戻り値

テーブル 45 • ListDeleteItem の戻り値

戻り値	説明
0	この関数によって、番号リストの現在の要素が正常に削除されたことを示します。
< 0	リストが空で現在の要素がないことを示します。
END_OF_LIST (1)	リストが空で現在の要素がないことを示します。
ISERR_LIST_NOSUCHLIST	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

ListDeleteItem の例

メモ • *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListDeleteItem 関数のデモンストレーションを行います。
*
* このスクリプトは最初に番号リストを作成し、ユーザーに対して
* 削除する番号を選択するようプロンプトします。そして ListFindItem が呼び出され、
* 選択した番号を現在の項目とします。次に、
* ListDeleteItem が呼び出され、リストから選択された番号を削除
* します。最後に、リストに残っている番号が表示
* 表示されます。
*
¥*---------------------------------------------------------------*/

#define TITLE_TEXT "ListDeleteItem の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListDeleteItem(HWND);
```

```
function ExFn_ListDeleteItem(hMSI)
    STRING   svItem, szMsg;
    LIST     listID;
    NUMBER   nvItem, nvResult;
begin

    // 数値リストを作成します。
    listID = ListCreate(NUMBERLIST);

    // エラーが発生した場合にそれを報告し、終了します。
    if (listID = LIST_NULL) then
        MessageBox (" リストを作成できませんでした。", SEVERE);
        abort;
    endif;

    // 番号 1、2、3 そして 4 をリストに追加します。
    for nvItem = 1 to 4
        ListAddItem (listID, nvItem, AFTER);
    endfor;

    repeat
        // [ 戻る ] ボタンを無効にします。
        Disable(BACKBUTTON);

        // 削除する番号を選択するようユーザーへ要求します。
        if AskText (" 番号 1、2、3、そして 4 が次のリストへ追加されました。¥n"+
                    " 削除する 3 つの番号のうち 1 つを選択します。", "", svItem)then
        endif;

        // 文字列データから番号データへ値を変換します。
        nvResult = StrToNum (nvItem, svItem);

        if  (nvItem < 1) || (nvItem > 4) then
            MessageBox ("1 と 4 の間の番号を入力しなくてはなりません。", WARNING);
        endif;
    until (nvItem > 0) && (nvItem <5);

    // ListFindItem がリストの最初から検索を始めるように、
    // 最初のリスト要素を現在の要素にします。
    ListSetIndex (listID, 0);

    // 選択した番号を検索します。
    nvResult = ListFindItem(listID, nvItem);

    // リストから選択した番号を削除します。
    if (ListDeleteItem (listID) < 0) then
        MessageBox (" 選択した番号を削除できませんでした。", SEVERE);
    else
        // 残りの番号を報告するメッセージのビルドを開始します。
        szMsg = " リストには現在次の番号が含まれます :¥n¥n";

        // 残りの番号を取得してメッセージへそれらを追加します。
        nvResult = ListGetFirstItem (listID, nvItem);
        while (nvResult != END_OF_LIST)
            NumToStr (svItem, nvItem);
            szMsg = szMsg + svItem + "  ";
            nvResult = ListGetNextItem (listID, nvItem);
        endwhile;
```

```
        // 残りの番号を表示します。
        MessageBox (szMsg, INFORMATION);
    endif;

    // メモリからリストを削除します。
    ListDestroy(listID);

end;
```

ListDeleteString

ListDeleteString 関数は、listID で指定された文字列リストから現在の要素を削除します。

また、ListGetFirstString と ListGetNextString 関数を使用してリストをトラバースし、任意の要素を現在の要素にすることができます。

メモ・*ListCurrentString は文字列リストでのみ使用できます。*

構文

```
ListDeleteString (listID);
```

パラメーター

テーブル 46 - ListDeleteString のパラメーター

パラメーター	説明
listID	現在の要素を削除する文字列リストを指定します。

戻り値

テーブル 47 - ListDeleteString の戻り値

戻り値	説明
0	この関数によって、文字列リストの現在の要素が正常に削除されたことを示します。
< 0	この関数によって、文字列リストの現在の要素が削除できなかったことを示します。
END_OF_LIST (1)	リストが空で現在の要素がないことを示します。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

ListDeleteString の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListDeleteString 関数のデモンストレーションを行います。
*
* このスクリプトは最初に文字列リストを作成し、ダイアログに
* 現在の文字列を表示します。次日、ListDeleteString が
* 2 回呼び出され、リストの最後の 2 つの文字列を削除します。
* そして現在の文字列が再び表示されます。
*
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "ListDeleteString & ListCurrentString"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListDeleteString(HWND);
```

```
function ExFn_ListDeleteString(hMSI)
   STRING   szString, svString, szMsg;
    LIST     listID;
begin

   // 文字列リストを作成します。
   listID = ListCreate (STRINGLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox (" リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // リストへ 3 つの文字列を追加します。
   ListAddString (listID, " 最初の文字列 ", AFTER);
   ListAddString (listID, "2 番目の文字列 ", AFTER);
   ListAddString (listID, "3 番目の文字列 ", AFTER);

   // リストにある現在の文字列を表示します。
   if (ListCurrentString (listID, svString) < 0) then
      MessageBox ("ListCurrentString への最初の呼び出しに失敗しました。", SEVERE);
   else

      szMsg = " リストの現在の文字列 : %s";
      SprintfBox (INFORMATION, TITLE_TEXT, szMsg, svString);

   endif;

   // 現在の文字列を削除します。 ("Value three")
    if (ListDeleteString (listID) < 0) then
      MessageBox ("ListDeleteString への最初の呼び出しに失敗しました。", SEVERE);
   endif;

   // 現在の文字列を削除します。 ("Value two")
   if (ListDeleteString (listID) < 0) then
      MessageBox ("ListDeleteString への 2 回目の呼び出しに失敗しました。", SEVERE);
   endif;

   // リストにある現在の文字列を表示します。
   if (ListCurrentString(listID, svString) < 0) then
      MessageBox ("ListCurrentString への 2 回目の呼び出しに失敗しました。", SEVERE);

   else
      szMsg = " リストの現在の文字列 : %s";
      SprintfBox (INFORMATION, TITLE_TEXT, szMsg, svString);

   endif;

   // メモリからリストを削除します。
   ListDestroy(listID);

end;
```

ListDestroy

ListDestroy 関数は、リストの内容とリストそのものを破棄します。この関数は、listID で識別された文字列または番号リストを削除するのに使用します。

リストが必要なくなったときや、セットアップスクリプトの最後で作成したすべてのリストを破棄してください。リストを破棄すると、リストに関連したすべてのメモリが解放されます。

メモ・この関数は文字列リストでも番号リストでも使用できます。リストを破棄した後、リスト関数でその listID を使用しないでください。

構文

```
ListDestroy(listID);
```

パラメーター

テーブル 48・ListDestroy のパラメーター

パラメーター	説明
listID	破棄する文字列または番号リストを指定します。

戻り値

テーブル 49・ListDestroy の戻り値

戻り値	説明
0	関数が正常にリストを破棄し、メモリからリストが削除されたことを示します。
< 0	関数がリストを破棄できなかったことを示します。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

ListDestroy の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListCreate と ListDestroy 関数のデモンストレーションを行います。
```

```
*
*番号リストを作成するために ListCreate が呼び出されます。ListDestroy
*が呼び出されて、それを破棄します。
*
¥*------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListDestroy(HWND);

function ExFn_ListDestroy(hMSI)
   LIST   listID;
   NUMBER nvItem;
begin

   // 空白数値リストを作成します。
   listID = ListCreate(NUMBERLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox ("リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // 番号 1078 をリストへ追加します。
   ListAddItem (listID, 1078, AFTER);

   // 番号 304 をリストへ追加します。
   ListAddItem (listID, 304, AFTER);

   // リストの現在の項目 (304) を読み出します。
   ListCurrentItem (listID, nvItem);

   // 現在の項目を表示します。
   SprintfBox (INFORMATION, "ListCreate",
         "リスト内の現在の項目 : %d", nvItem);

   // リストの最初の項目 (1078) を読み出します。
   ListGetFirstItem (listID, nvItem);

   // 最初の項目を表示します。
   SprintfBox (INFORMATION, "ListCreate",
         "リスト内の最初の項目 : %d", nvItem);

   // メモリからリストを削除します。
   ListDestroy(listID);

end;
```

ListFindItem

ListFindItem 関数は、現在の要素から始めてその位置から続けて番号リストの指定した要素を検索します。リストの最初から検索する場合は、ListGetFirstItem 関数を使用します。ListFindItem で要素が見つかると、その要素がリストの現在の要素になります。

メモ・ListFindItem 関数は番号リストでのみ使用できます。

構文

ListFindItem (listID, nItem);

パラメーター

テーブル 50・ListFindItem のパラメーター

パラメーター	説明
listID	検索する番号リストを指定します。
nItem	リストで検索する項目を指定します。

戻り値

テーブル 51・ListFindItem の戻り値

戻り値	説明
0	関数が要求された要素を正しく検出しました。
< 0	エラーのため、関数は指定のリストを検索できませんでした。このエラーは、listID によって指定されたリストが存在しない場合などに発生します。
END_OF_LIST (1)	関数はリストの終わりまで検索しましたが、要求された要素が見つかりませんでした。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しません。有効なリストの ID は **ListCreate** 関数からの戻り値です。

ListFindItem の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListFindItem 関数のデモンストレーションを行います。
*
* このスクリプトは番号リストを作成してそこに 3 つの番号を
* 追加します。そしてユーザーは番号のひとつを推測するよう要求されます。
```

```
＊ListFindItem が呼び出され、ユーザーが入力した番号をリスト内で
＊検索します。ユーザーに対し、推測した番号が正解か、不正解かを
＊伝える メッセージボックスが表示されます。
＊
¥*---------------------------------------------------------------*/

#define TITLE_TEXT "ListFindItem の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListFindItem(HWND);

function ExFn_ListFindItem(hMSI)
   STRING svItem;
   NUMBER nItem, nResult;
   LIST   listID;
begin

   // 番号リストを作成します。
   listID = ListCreate(NUMBERLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox (" リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // リストへ 3 つの番号を追加します。
   ListAddItem (listID, 1, AFTER);
   ListAddItem (listID, 5, AFTER);
   ListAddItem (listID, 9, AFTER);

   // ユーザーへ番号を要求します。
   if AskText ("1 から 10 までの 3 つの番号がリストへ追加されました。¥n"+
            " これらの番号のうち 1 つを推測します。", "", svItem) = NEXT then
      // 文字列データから番号データへ値を変換します。
      if StrToNum (nItem, svItem) < 0 then
         MessageBox (" 番号をが入力されませんでした : 処理がキャンセルされました。", SEVERE);
      else
         // ListFindItem がリストの最初から検索を始めるように、
         // 最初のリスト要素を現在の要素にします。
         ListSetIndex (listID, 0);

         // 番号リストを検索します。
         nResult = ListFindItem (listID, nItem);

         // 検索結果を表示されます。
         if (nResult < 0) then
            MessageBox ("ListFindItem が失敗しました。", SEVERE);
         elseif (nResult = END_OF_LIST) then
            SprintfBox (WARNING, TITLE_TEXT, " 残念ながら、%d はリストに存在しません。", nItem);
         elseif (nResult = 0) then
            SprintfBox (INFORMATION, TITLE_TEXT, "%d がリストに存在します。", nItem);
         endif;
      endif;
   endif;

   // メモリからリストを削除します。
```

```
    ListDestroy(listID);

end;
```

ListFindKeyValueString

ListFindKeyValueString 関数は、指定された値の文字列または数値リストを検索します。最初のリストで見つかった文字列の位置に対応する追加リストから値が返されます。これによって、キーと値のペアー覧で特定のキーを検索して、対応する値を取得できます。

構文

```
ListFindKeyValueString (byval LIST listKeys, byval LIST listValues, byval string szKey, byref string svValue);
```

パラメーター

テーブル 52 - ListFindKeyValueString のパラメーター

パラメーター	説明
listKeys	検索するキーの一覧を示します。文字列または数値リストを指定できます。リストは **ListFindKeyValueString** の前に **ListCreate** を呼び出して、初期化しなくてはなりません。 検索の開始ポイントは、リストの現在の位置にかかわらず、常にリストの始まりです。関数が返るとき、リストの現在の位置が検出された要素となります。
listValues	キーのリストに対応する値のリストを示します。文字列または数値リストを指定できます。リストは **ListFindKeyValueString** の前に **ListCreate** を呼び出して、初期化しなくてはなりません。 listKeys と listValues のリストの種類は、必ずしも同じである必要はありません。
szKey	検索する文字列を指定します。 listKeys が数値リストの場合、szKey は **StrToNum** 関数を使って数値に変換されてから、リストの検索に使用されます。 **ListFindKeyValueString** は **ListFindString** (リストが文字列リストの場合) または **ListFindItem** (リストが数値リストの場合) を呼び出してリストを検索します。**ListFindString** が大文字と小文字を区別して検索するため、文字列リストの検索では大文字と小文字が区別されます。
svValue	2 番目のリストからの値がこのパラメーターで返されます。listValues が数値リストの場合、**NumToStr** を使って数値が文字列に変換されてから、文字列として返されます。

戻り値

テーブル 53 -

戻り値	説明
0	関数が要求された要素を正しく検出しました。
< 0	エラーのため、関数は指定のリストを検索できませんでした。このエラーは、listID によって指定されたリストが存在しない場合などに発生します。
END_OF_LIST (1)	関数はリストの終わりまで検索しましたが、要求された要素が見つかりませんでした。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しません。有効なリストの ID は **ListCreate** 関数からの戻り値です。

ListFindString

ListFindString 関数は、現在の要素から始めてその位置から続けて文字列リストの指定した要素を検索します。文字列リストの最初から検索する場合は、ListGetFirstString 関数を使用します。ListFindString で要素が見つかると、その要素がリストの現在の要素になります。

メモ・ListFindString 関数は大文字と小文字を区別して文字列の比較を行い、文字列リストでのみ動作します。

構文

```
ListFindString (listID, szString);
```

パラメーター

テーブル 54・ListFindString のパラメーター

パラメーター	説明
listID	検索する文字列リストを指定します。
szString	リストで検索する項目を指定します。InstallShield ではこの文字列の検索時に大文字と小文字を区別する比較を実行します。

戻り値

テーブル 55・ListFindString の戻り値

戻り値	説明
0	関数が要求された要素を正常に見つけたことを示します。
< 0	エラーのため、指定のリストが検索できなかったことを示します。このエラーは、listID によって指定されたリストが存在しない場合などに発生します。
END_OF_LIST (1)	InstallShield はリストの終わりまで検索しましたが、要求された要素を見つけられなかったことを示します。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

ListFindString の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * ListFindString 関数のデモンストレーションを行います。
 *
 * このスクリプトはユーザーに対してターゲットシステムにあるプログラムフォルダー名の
 * 入力を要求します。そして ListFindString が呼び出され、
 * ターゲットシステムにあるすべてのフォルダー名を含むリスト内で
 * このフォルダー名を検索します。検索結果を
 * レポートするメッセージが表示されます。
 *
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "ListFindString の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListFindString(HWND);

function ExFn_ListFindString(hMSI)
   STRING szString;
   LIST   listID, listSubfoldersID;
   NUMBER nResult;
begin

   // 文字列リストを作成します。
   listID = ListCreate (STRINGLIST);
   listSubfoldersID = ListCreate (STRINGLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) || (listSubfoldersID = LIST_NULL) then
      MessageBox (" リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // リストにプログラムフォルダーの名前を入力します。
   GetFolderNameList (FOLDER_PROGRAMS, listID, listSubfoldersID);

   // ユーザーにフォルダー名の入力を要求します。
   AskText (" 既存プログラムフォルダーの名前を入力してください。",
         "", szString);

   // ListFindItem がリストの最初から検索を始めるように、
   // 最初のリスト要素を現在の要素にします。
   ListSetIndex (listSubfoldersID, 0);

   // リスト内でユーザーが入力したフォルダー名を検索します。
   // 注意 : 文字列は大文字と小文字を区別します。
   nResult = ListFindString (listSubfoldersID, szString);

   // 検索結果を報告します。
   if (nResult < 0) then
      MessageBox ("ListFindString が失敗しました。", SEVERE);
   elseif (nResult = END_OF_LIST) then
      SprintfBox (WARNING, TITLE_TEXT, "%s が見つかりませんでした。", szString);
   elseif (nResult = 0) then
      SprintfBox (INFORMATION, TITLE_TEXT, "ListFindString を検出しました : %s。",
```

```
                szString);
    endif;

    // メモリからリストを削除します。
    ListDestroy(listID);
    ListDestroy (listSubfoldersID);

end;
```

ListGetFirstItem

ListGetFirstItem 関数は、番号リストの 1 番目の要素を読み出します。最初の項目が、リストの現在の要素になります。

***メモ・** ListGetFirstItem 関数は番号リストでのみ使用できます。*

構文

```
ListGetFirstItem (listID, nvItem);
```

パラメーター

テーブル 56 ・ ListGetFirstItem のパラメーター

パラメーター	説明
listID	最初の要素を取得する番号リストを指定します。
nvItem	番号リストの最初の要素を返します。

戻り値

テーブル 57 ・ ListGetFirstItem の戻り値

戻り値	説明
0	この関数によって、番号リストの最初の要素が正常に取得されたことを示します。
-1	エラーのため、番号リストの最初の要素が取得できなかったことを示します。
END_OF_LIST (1)	リストが空であることを示します。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

ListGetFirstItem の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListGetFirstItem 関数と ListGetNextItem 関数のデモンストレーションを
* デモンストレーションを行います。
*
* このスクリプトは番号リストの作成からはじまり、そこに 3 つの番号を
* 追加します。ListGetFirstItem が呼び出され、リストから
* 最初の番号が取得されます。この呼び出しで最初の番号が
* 現在の要素となります。リストから読み出された番号を
* 表示するループ処理が続き、そして
* ListGetNextItem を呼び出して次の番号を取得します。ループは
* ListGetFirstItem または ListGetNextItem がリストの終わりに到達するまで実行
* されます。
*
```

```
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "ListGetFirstItem & ListGetNextItem"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListGetFirstItem(HWND);

function ExFn_ListGetFirstItem(hMSI)
   STRING szMsg;
   LIST   listID;
   NUMBER nvItem, nResult;
begin

   // 番号リストを作成します。
   listID = ListCreate(NUMBERLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox ("リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // リストへ 3 つの番号を追加します。
   ListAddItem (listID, 1024, AFTER);
   ListAddItem (listID, 360, AFTER);
   ListAddItem (listID, 777, AFTER);

   // リストから最初の番号を取得します。
   nResult = ListGetFirstItem (listID, nvItem);

   // リストの終わりでない場合はループし続けます。
   while ( nResult != END_OF_LIST )
      // リストから読み出した番号を表示します。
      SprintfBox (INFORMATION, TITLE_TEXT, "%i", nvItem);

      // リストから次の番号を取得します。
      nResult = ListGetNextItem (listID, nvItem);
   endwhile;

   // メモリからリストを削除します。
   ListDestroy(listID);

end;
```

ListGetFirstString

ListGetFirstString 関数は、文字列リストの 1 番目の要素を読み出します。最初の文字列が、リストの現在の要素になります。

ListGetFirstString 関数は文字列リストでのみ使用できます。

構文

```
ListGetFirstString (listID, svString);
```

パラメーター

テーブル 58 ・ ListGetFirstString のパラメーター

パラメーター	説明
listID	最初の要素を取得する文字列リストを指定します。
svString	文字列リストの最初の要素を返します。

戻り値

テーブル 59 ・ ListGetFirstString の戻り値

戻り値	説明
0	この関数によって、文字列リストの最初の要素が正常に取得されたことを示します。
-1	エラーのため、文字列リストの最初の要素が取得できなかったことを示します。
END_OF_LIST (1)	リストが空であることを示します。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

ListGetFirstString の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListGetFirstString 関数と ListGetNextString 関数のデモンストレーションを
* デモンストレーションを行います。
*
* このスクリプトは文字列リストの作成からはじまり、プログラムフォルダー名を
* 追加します。ListGetFirstString が呼び出されて、
* リストから最初の文字列が取得されます。この呼び出しはまた、
* 最初の文字列を現在の要素とします。リストから
* 読み出した文字列を表示してループ処理が続き、
* ListGetNextString を呼び出して次の文字列を取得します。その後
* ループは ListGetFirstString または ListGetNextString がリストの最後に到達するまで
* 続きます。
*
```

```
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListGetFirstString(HWND);

function ExFn_ListGetFirstString(hMSI)
   STRING svString;
   LIST   listID;
   NUMBER nResult;
begin

   // 文字列リストを作成します。
   listID = ListCreate (STRINGLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox ("リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // リストへプログラムフォルダー名を取得します。
   if GetGroupNameList(listID) < 0 then ;
    // ListCreate が失敗したことを報告します。
      MessageBox ("プログラムフォルダー名を読み出すことができませんでした。", SEVERE);
   else
      // リストから最初の文字列を取得します。
      nResult = ListGetFirstString (listID, svString);

      // リスト項目が引き続き読み出されている間ループします。
      while ( nResult != END_OF_LIST )
         // 現在の要素を表示します。
         MessageBox (svString, INFORMATION);

         // リストの次の文字列を取得します。
         nResult = ListGetNextString (listID, svString);
      endwhile;
   endif;

   // メモリからリストを削除します。
   ListDestroy(listID);

end;
```

ListGetIndex

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ListGetIndex 関数は、指定されたリストの現在の要素のインデックスを取得します。インデックス番号はゼロ (0) から始まります。

構文

```
ListGetIndex ( listID, nIndex );
```

パラメーター

テーブル 60 - ListGetIndex のパラメーター

パラメーター	説明
listID	現在の要素のインデックスを読み出す ID、文字列、または番号リストを入力してください。
nIndex	リストの現在のアイテムのインデックスを戻します。

戻り値

テーブル 61 - ListGetIndex の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がリストの現在の要素のインデックスを正常に取得したことを示します。
< ISERR_SUCCESS	関数がリストの現在の要素のインデックスを正常に取得できなかったことを示します。
ISERR_LIST_NOSUCHLIST	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

追加情報

- ListCurrentItem と ListCurrentString を使用して、現在の要素の値を取得します。
- この関数は文字列リストでも番号リストでも使用できます。

ListGetNextItem

ListGetNextItem 関数は、番号リストの現在の要素の後の項目を読み出します。読み出されたアイテムが、リストの現在の要素になります。

メモ - *ListGetNextItem 関数は番号リストでのみ使用できます。*

構文

ListGetNextItem (listID, nvItem);

パラメーター

テーブル 62 ・ ListGetNextItem のパラメーター

パラメーター	説明
listID	次の要素を取得する番号リストを指定します。
nvItem	番号リストで現在の要素に続く項目を返します。その項目が、リストの現在の要素になります。

戻り値

テーブル 63 ・ ListGetNextItem の戻り値

戻り値	説明
0	この関数によって、番号リストの現在の要素の後の要素が正常に取得されたことを示します。
-1	エラーのため、番号リストの指定の要素が取得できなかったことを示します。
END_OF_LIST (1)	現在の項目がリストの最後の要素であることを示します。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

ListGetNextItem の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListGetFirstItem 関数と ListGetNextItem 関数のデモンストレーションを
* デモンストレーションを行います。
*
* このスクリプトは番号リストの作成からはじまり、そこに 3 つの番号を
* 追加します。ListGetFirstItem が呼び出され、リストから
* 最初の番号が取得されます。この呼び出しで最初の番号が
* 現在の要素となります。リストから読み出された番号を
* 表示するループ処理が続き、そして
```

```
＊ListGetNextItem を呼び出して次の番号を取得します。ループは
＊ListGetFirstItem または ListGetNextItem がリストの終わりに到達するまで実行
＊されます。
＊
¥*--------------------------------------------------------------*/

#define TITLE_TEXT "ListGetFirstItem & ListGetNextItem"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListGetNextItem(HWND);

function ExFn_ListGetNextItem(hMSI)
   STRING szMsg;
   LIST   listID;
   NUMBER nvItem, nResult;
begin

   // 番号リストを作成します。
   listID = ListCreate(NUMBERLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox ("リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // リストへ 3 つの番号を追加します。
   ListAddItem (listID, 1024, AFTER);
   ListAddItem (listID, 360, AFTER);
   ListAddItem (listID, 777, AFTER);

   // リストから最初の番号を取得します。
   nResult = ListGetFirstItem (listID, nvItem);

   // リストの終わりでない場合はループし続けます。
   while ( nResult != END_OF_LIST )
      // リストから読み出した番号を表示します。
      SprintfBox (INFORMATION, TITLE_TEXT, "%i", nvItem);

      // リストから次の番号を取得します。
      nResult = ListGetNextItem (listID, nvItem);
   endwhile;

   // メモリからリストを削除します。
   ListDestroy(listID);

end;
```

ListGetNextString

ListGetNextString 関数は、文字列リストの、現在の要素の次の要素を読み出します。取得された要素が、リストの現在の要素になります。

メモ・ListGetNextString 関数は文字列リストでのみ使用できます。

構文

ListGetNextString (listID, svString);

パラメーター

テーブル 64・ListGetNextString のパラメーター

パラメーター	説明
listID	次の要素を取得する文字列リストを指定します。
svString	文字列リストで現在の要素に続く文字列を返します。その文字列が、リストの現在の要素になります。

戻り値

テーブル 65・ListGetNextString の戻り値

戻り値	説明
0	この関数によって、文字列リストの現在の要素の後の要素が正常に取得されたことを示します。
-1	エラーのため、文字列リストの指定の要素が取得できなかったことを示します。
END_OF_LIST (1)	現在の項目がリストの最後の要素であることを示します。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

ListGetNextString の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
```

```
＊InstallShield スクリプトの例
＊
＊ListGetFirstString 関数と ListGetNextString 関数のデモンストレーションを
＊デモンストレーションを行います。
＊
＊このスクリプトは文字列リストの作成からはじまり、プログラムフォルダー名を
＊追加します。ListGetFirstString が呼び出されて、
＊リストから最初の文字列が取得されます。この呼び出しはまた、
＊最初の文字列を現在の要素とします。リストから
＊読み出した文字列を表示してループ処理が続き、
＊ListGetNextString を呼び出して次の文字列を取得します。その後
＊ループは ListGetFirstString または ListGetNextString がリストの最後に到達するまで
＊続きます。
＊
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListGetNextString(HWND);

function ExFn_ListGetNextString(hMSI)
   STRING svString;
   LIST   listID;
   NUMBER nResult;
begin

   // 文字列リストを作成します。
   listID = ListCreate (STRINGLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox ("リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // リストへプログラムフォルダー名を取得します。
   if GetGroupNameList(listID) < 0 then ;
      // ListCreate が失敗したことを報告します。
      MessageBox ("プログラムフォルダー名を読み出すことができませんでした。", SEVERE);
   else
      // リストから最初の文字列を取得します。
      nResult = ListGetFirstString (listID, svString);

      // リスト項目が引き続き読み出されている間ループします。
      while ( nResult != END_OF_LIST )
         // 現在の要素を表示します。
         MessageBox (svString, INFORMATION);

         // リストの次の文字列を取得します。
         nResult = ListGetNextString (listID, svString);
      endwhile;
   endif;

   // メモリからリストを削除します。
   ListDestroy(listID);

end;
```

ListGetType

ListGetType 関数は、リストが STRINGLIST と NUMBERLIST のいづれであるかを判断します。

構文

ListGetType (listID);

パラメーター

テーブル 66 - ListGetType のパラメーター

パラメーター	説明
listID	文字列リスト名を指定します。listID によって識別されるリストは、ListCreate への呼び出しによって既に初期化されている必要があります。

戻り値

テーブル 67 - ListGetType の戻り値

戻り値	説明
STRINGLIST	リストが文字列リストであることを示します。
NUMBERLIST	リストが番号リストであることを示します。
<0	関数がリストの種類を判断できなかったことを示します。

ListGetType の例

メモ - *listID によって識別されるリストは、ListCreate<LangrefListCreate.htm> への呼び出しによって既に初期化されている必要があります。*

```
// サンプルコード
LIST   listID;
int    nType;

program
   listID = ListCreate(STRINGLIST);
   if (LIST_NULL != listID) then
      nType = ListGetType(listID);
      if (STRINGLIST = nType) then
         MessageBox("It is a STRINGLIST", INFORMATION);
      else
         MessageBox("It is a NUMBERLIST", INFORMATION);
      endif;
```

```
    endif;
    ListDestroy(listID);

    listID = ListCreate(NUMBERLIST);
    if (LIST_NULL != listID) then
        nType = ListGetType(listID);
        if (STRINGLIST = nType) then
            MessageBox("It is a STRINGLIST", INFORMATION);
        else
            MessageBox("It is a NUMBERLIST", INFORMATION);
        endif;
    endif;
    ListDestroy(listID);
endprogram
```

ListReadFromFile

ListReadFromFile 関数は、テキストファイルをリストへ読み込みます。テキストファイルをリストに読み込むと、セットアップの最後に README ファイルを表示したり、ListWriteToFile を使ってディスクに文字列リストを書き込むなど、セットアップでさまざまな関数にこのリストを使用できます。

この関数は各文字列の最後にある改行文字を検出し、その文字をリストの各要素の区切り文字に使用します。

***メモ・** ListReadFromFile 関数は文字列リストとテキストファイルでのみ使用できます。*

構文

```
ListReadFromFile (listID, szFile);
```

パラメーター

テーブル 68 - ListReadFromFile のパラメーター

パラメーター	説明
listID	szFile で指定されたファイルから読み取った行のリストを返します。listID によって識別されるリストは、ListCreate への呼び出しによって既に初期化されている必要があります。
szFile	リストに読み込まれるファイルの完全修飾名を指定します。

戻り値

テーブル 69 - ListReadFromFile の戻り値

戻り値	説明
0	ファイルのテキスト行を正常にリストに読み込んだことを示します。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。
< 0	ファイルのテキスト行をリストに読み込めなかったことを示します。

ListReadFromFile の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListReadFromFile 関数と ListWriteToFile 関数のデモンストレーションを
* デモンストレーションを行います。
*
* ListReadFromFile が呼び出されて Autoexec.bat ファイルを読み込み、
* 表示します。リストが表示されたあと、
* リストの終わりに備考が付け加えられ、新規ファイルへ
* リストが書き込まれます。オリジナルファイルは変更されないままの
* 状態です。
*
* メモ : このスクリプトを適切に実行するため、
```

```
＊ C ドライブのルートディレクトリに Autoexec.bat と名づけられた
＊バッチファイルが必要です。
＊
¥*-------------------------------------------------------------*/

#define OLD_FILE "C:¥¥Autoexec.bat"
#define NEW_FILE "C:¥¥Autoexec.lst"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListReadFromFile(HWND);

function ExFn_ListReadFromFile(hMSI)
   LIST   listID;
begin

   // 文字列リストを作成します。
   listID = ListCreate (STRINGLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox (" リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // ファイルを文字列リストへ読み込みます。
   if (ListReadFromFile (listID, OLD_FILE) < 0) then
      // ListReadFromFile が失敗したことを報告します。
      MessageBox (OLD_FILE + " を読み込むことができませんでした。", SEVERE);
   else
      // リストを表示します。
      SdShowInfoList ("Demo", " リスト例ファイル :", listID);

      // リストへ新規文字列を追加します。
      ListAddString (listID, "IS セットアップが REM を追加しました。", AFTER);

      // ファイルへのリストを書き込みます。
      if (ListWriteToFile (listID, NEW_FILE) < 0) then
         MessageBox (""+ NEW_FILE + " へリストを書き込むことができませんでした。", SEVERE);
      else
         MessageBox (" リストを無事に書き込みました "+ NEW_FILE + "。", INFORMATION);
      endif;
   endif;

   // メモリからリストを削除します。
   ListDestroy(listID);

end;
```

ListSetCurrentItem

ListSetCurrentItem 関数は、番号リストの現在の要素に nItem の値を割り当てます。

メモ・ListSetCurrentItem 関数は番号リストでのみ使用できます。

構文

ListSetCurrentItem (listID, nItem);

パラメーター

テーブル 70 ・ ListSetCurrentItem のパラメーター

パラメーター	説明
listID	現在の要素を更新する番号リスト名を指定します。listID によって識別されるリストは、ListCreate への呼び出しによって既に初期化されている必要があります。
nItem	現在の要素と置換する数値を指定します。

戻り値

テーブル 71 ・ ListSetCurrentItem の戻り値

戻り値	説明
0	この関数によって、番号リストの現在の要素が正常に更新されたことを示します。
< 0	この関数によって、番号リストの現在の要素が更新できなかったことを示します。
END_OF_LIST (1)	リストが空であることを示します。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

ListSetCurrentItem の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListSetCurrentItem 関数のデモンストレーションを行います。
*
* このスクリプトは最初に番号リストを作成して、そこに
* 2 つの番号追加します。* そしてリストがダイアログに表示されます。
* ListSetCurrentItem が呼び出されてリストの現在の要素の値を
```

```
＊更新します。最後に、この新しいリストが表示
＊表示されます。
＊
¥*---------------------------------------------------------------*/

#define  MSG_TEXT   "listID の要素 :¥n¥n"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListSetCurrentItem(HWND);

function ExFn_ListSetCurrentItem(hMSI)
   LIST   listID;
   NUMBER nItem, nvResult, nvItem;
   STRING szMsg, svItem;
begin

   // 空白数値リストを作成します。
   listID = ListCreate(NUMBERLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox ("リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // リストへ 2 つの番号を追加します。
   ListAddItem (listID, 1024, AFTER);
   ListAddItem (listID, 360, AFTER);

   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

      // 番号を報告するメッセージのビルドを開始します。
      szMsg = MSG_TEXT;

      // 番号を取得してメッセージへそれらを追加します。
      nvResult = ListGetFirstItem (listID, nvItem);
      while (nvResult != END_OF_LIST)
         NumToStr (svItem, nvItem);
         szMsg = szMsg + svItem + "  ";
         nvResult = ListGetNextItem (listID, nvItem);
      endwhile;

      // 番号を表示します。
      MessageBox (szMsg, INFORMATION);

   // 2 番目の項目の値を新しい値と置き換えます。
   if (ListSetCurrentItem (listID, 777) < 0) then
      MessageBox ("ListSetCurrentItem が失敗しました。", SEVERE);
   else
      // 番号を報告するメッセージのビルドを開始します。
      szMsg = MSG_TEXT;

      // 番号を取得してメッセージへそれらを追加します。
      nvResult = ListGetFirstItem (listID, nvItem);

      while (nvResult != END_OF_LIST)
```

```
            NumToStr (svItem, nvItem);
            szMsg = szMsg + svItem + " ";
            nvResult = ListGetNextItem (listID, nvItem);
        endwhile;

        // 番号を表示します。
        MessageBox (szMsg, INFORMATION);
    endif;

    // メモリからリストを削除します。
    ListDestroy(listID);

end;
```

ListSetCurrentString

ListSetCurrentString 関数は、文字列リストの現在の要素に szString の値を割り当てます。

***メモ・** ListSetCurrentString 関数は文字列リストでのみ使用できます。*

構文

```
ListSetCurrentString (listID, szString);
```

パラメーター

テーブル 72 ・ ListSetCurrentString のパラメーター

パラメーター	説明
listID	現在の要素を更新する文字列リスト名を指定します。listID によって識別されるリストは、ListCreate への呼び出しによって既に初期化されている必要があります。
szString	現在の要素と置換する文字列値を指定します。

戻り値

テーブル 73 ・ ListSetCurrentString の戻り値

戻り値	説明
0	この関数によって、番号リストの現在の要素が正常に更新されたことを示します。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。
< 0	この関数によって、番号リストの現在の要素が更新できなかったことを示します。このエラーが起きる主な理由は、索引が使用できるリスト要素の範囲外にあるためです。

ListSetCurrentString の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListSetCurrentString 関数のデモンストレーションを行います。
*
* このスクリプトでは、2 つの文字列を空白リストへ追加します。
次いでリストがダイアログに表示されます。次に、
* ListSetCurrentString が呼び出され、現在の要素を置換
* します。そしてリストが再びダイアログに
* 表示されます。
*
¥*---------------------------------------------------------------*/
```

```
#define  TITLE_TEXT "ListSetCurrentString の例 "
#define  MSG_TEXT   "listID の要素 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListSetCurrentString(HWND);

function ExFn_ListSetCurrentString(hMSI)
   LIST   listID;
begin

   // 空白文字列リストを作成します。
   listID = ListCreate (STRINGLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox (" リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // リストへ 2 つの文字列を追加します。
   ListAddString (listID, "Keyboard", AFTER);
   ListAddString (listID, " モニター ", AFTER);

   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   // リストを表示します。
   SdShowInfoList (TITLE_TEXT, MSG_TEXT, listID);

   // 2 番目の項目の値を新しい値と置き換えます。
   if (ListSetCurrentString(listID, "Mouse") < 0) then
      MessageBox ("ListSetCurrentString が失敗しました。", SEVERE);
   endif;

   // 新しい代替リストを表示します。
   SdShowInfoList (TITLE_TEXT, MSG_TEXT, listID);

   // メモリからリストを削除します。
   ListDestroy(listID);

end;
```

ListSetIndex

ListSetIndex 関数は、インデックスを使って文字列または番号リストの特定の要素を現在の要素にします。また、定数を使ってリスト要素を 1 つずつスキャンすることも、リストの最初か最後に移動してリストをスキャンすることもできます。インデックスを使ってリスト項目にアクセスすることによって、番号リストと文字列リストを配列処理できます。

インデックス化した要素を現在の要素に設定した後で、ListCurrentItem か ListCurrentString 関数をスクリプトで使用して、インデックス化された (現在の) 項目の値を取得できます。

***メモ・** ListSetIndex 関数は文字列リストでも番号リストでも使用できます。*

構文

ListSetIndex (listID, nIndex);

パラメーター

テーブル 74 • ListSetIndex のパラメーター

パラメーター	説明
listID	インデックスを設定する文字列または番号リスト名を指定します。
nIndex	現在の要素に設定する要素の番号を指定します。リスト要素の番号はゼロ (0) から始まります。たとえば nIndex パラメーターに 5 と入力すると、リストで 6 番目の場所にある項目が現在の要素になります。 このパラメーターに、数値か、あらかじめ定義されている以下の定数のうちの 1 つを渡します。 • LISTFIRST Ñ リストの最初の要素に移動します。 • LISTLAST Ñ リストの最後の要素に移動します。 • LISTNEXT Ñ リストの次の要素に移動します。 • LISTPREV Ñ リストの前の要素に移動します。

戻り値

テーブル 75 • ListSetIndex の戻り値

戻り値	説明
0	この関数によって、リストの現在の要素が正常に更新されたことを示します。
< 0	この関数によって、リストの現在の要素が更新できなかったことを示します。
END_OF_LIST (1)	インデックスが使用できるリスト要素の範囲外にあることを示します。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。

ListSetIndex の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListSetIndex 関数のデモンストレーションを行います。
*
* このスクリプト例はプログラムフォルダーのリストから最後の 3 項目を
* 表示します。
*
* メモ : ターゲットシステムに Explorer 以外のシェルがある場合、
*    GetGroupNameList 関数は、エラーを戻す可能性があります。
*
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "ListSetIndex の例 "
#define MSG_TEXT   " このリスト内の最後の 3 項目に注意してください。"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListSetIndex(HWND);

function ExFn_ListSetIndex(hMSI)
   STRING   svString;
   LIST     listID;
   NUMBER   nResult, nIndex;
begin

   // 文字列リストを作成します。
   listID  = ListCreate (STRINGLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox (" リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // リストへプログラムフォルダー名を取得します。
   if GetGroupNameList (listID) < 0 then ;
      // ListCreate が失敗したことを報告します。
      MessageBox (" プログラムフォルダー名を読み出すことができませんでした。", SEVERE);
   else
      // リストを表示します。
      SdShowInfoList( TITLE_TEXT, MSG_TEXT, listID);

      // リストの最後から 3 番目の項目を現在の要素に
      // します。3 項目よりも少ない場合は、
      // すべての項目を表示します。リストのインデックスは 0 で始まります。
      nIndex = ListCount (listID);

      if nIndex > 3 then
         nIndex = nIndex - 3;
```

```
        endif;

        nResult = ListSetIndex (listID, nIndex);

        // 表示する項目がある限りループし続けます。
        while ( nResult != END_OF_LIST )
            // 現在のリスト項目を取得します。
            ListCurrentString (listID, svString);

            // 現在のリスト項目を表示します。
            MessageBox (svString, INFORMATION);

            // 次の項目を現在の項目にします。
            nResult = ListSetIndex (listID, LISTNEXT);
        endwhile;
    endif;

    // メモリからリストを削除します。
    ListDestroy(listID);

end;
```

ListValid

ListValid 関数は listID が指定したリストが有効かどうかを示します。つまり、ListCreate の呼び出しによって初期化済みで、ListDestroy の呼び出して無効とされていないかどうかを示します。この関数は文字列リストでも番号リストでも使用できます。

構文

```
ListValid ( listID );
```

パラメーター

テーブル 76 • ListValid のパラメーター

パラメーター	説明
listID	チェックする文字列または番号リストを指定します。

戻り値

テーブル 77 • ListValid の戻り値

戻り値	説明
>= ISERR_SUCCESS	リストが有効であることを示します。
< ISERR_SUCCESS	リストが無効であることを示します。
ISERR_LIST_NOSUCHLIST	リストが初期化されていないことを示します。

ListValid の例

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * ListValid 関数のデモンストレーションを行います。
 *
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

function OnBegin()
    LIST   listID;
    number nListValid;
begin
    // 空白数値リストを作成します。
    listID = ListCreate(NUMBERLIST);

    // エラーが発生した場合にそれを報告し、終了します。
    if (listID = LIST_NULL) then
        MessageBox ("リストを作成できませんでした。", SEVERE);
        abort;
    endif;

    // リスト ID が有効かどうかをチェックします。
    nListValid = ListValid (listID);
    if (nListValid >= ISERR_SUCCESS) then
        MessageBox ("リスト ID が有効です。", INFORMATION);
    else
        MessageBox ("リスト ID が無効です。", INFORMATION);
    endif;
```

```
    // メモリからリストを削除します。
    ListDestroy(listID);

    // リスト ID が有効かどうかをチェックします。
    nListValid = ListValid (listID);
    if (nListValid >= ISERR_SUCCESS) then
        MessageBox (" リスト ID が有効です。", INFORMATION);
    else
        MessageBox (" リスト ID が無効です。", INFORMATION);
    endif;

    // このサンプル スクリプトをプロジェクトにカット アンド ペーストして実行すると、
    // 次の行によって、スクリプトの実行が中止されます。
    abort;
end;
```

ListValidType

ListValidType 関数は listID が指定したリストが有効かどうかを示します。つまり、ListCreate の呼び出しによって初期化済みで、ListDestroy の呼び出して無効とされていないかどうかを示します。

構文

```
ListValidType ( listID, nType );
```

パラメーター

テーブル 78 ・ ListValidType のパラメーター

パラメーター	説明
listID	チェックする文字列または番号リストを指定します。
nType	指定されたリストが比較される対象となるリストの種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ NUMBERLIST Ñ 番号リストを指定します。 ・ STRINGLIST Ñ 文字列リストを指定します。

戻り値

テーブル 79 ・ ListValidType の戻り値

戻り値	説明
>= ISERR_SUCCESS	リストが有効であり、また指定されたの種類であることを示します。
< ISERR_SUCCESS	リストが無効であり、また指定された種類ではないことを示します。
ISERR_LIST_NOSUCHLIST	リストが初期化されていないことを示します。
ISERR_LIST_TYPEMISMATCH	リストが指定された種類ではないことを示します。

ListValidType の例

```
/*---------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * ListValidType 関数のデモンストレーションを行います。
 *
¥*---------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

function OnBegin()
    LIST   listID;
    number nListValidType;
begin
```

```
    // 空白文字列リストを作成します。
    listID = ListCreate (STRINGLIST);

    // エラーが発生した場合にそれを報告し、終了します。
    if (listID = LIST_NULL) then
        MessageBox (" リストを作成できませんでした。", SEVERE);
        abort;
    endif;

    // リスト ID が有効かどうか、またどのタイプなのかをチェックします。
    nListValidType = ListValidType (listID, NUMBERLIST);
    if (nListValidType >= ISERR_SUCCESS) then
        MessageBox (" リスト ID は有効で、数値リストです。",
            INFORMATION);
    else
        nListValidType = ListValidType (listID, STRINGLIST);
        if (nListValidType >= ISERR_SUCCESS) then
            MessageBox (" リスト ID は有効で、文字列リストです。",
                INFORMATION);
        else
            MessageBox (" リスト ID が無効です。", INFORMATION);
        endif;
    endif;

   // メモリからリストを削除します。
    ListDestroy(listID);

    // リスト ID が有効かどうか、またどのタイプなのかをチェックします。
    nListValidType = ListValidType (listID, NUMBERLIST);
    if (nListValidType >= ISERR_SUCCESS) then
        MessageBox (" リスト ID は有効で、数値リストです。",
            INFORMATION);
    else
        nListValidType = ListValidType (listID, STRINGLIST);
        if (nListValidType >= ISERR_SUCCESS) then
            MessageBox (" リスト ID は有効で、文字列リストです。",
                INFORMATION);
        else
            MessageBox (" リスト ID が無効です。", INFORMATION);
        endif;
    endif;

    // このサンプル スクリプトをプロジェクトにカット アンド ペーストして実行すると、
    // 次の行によって、スクリプトの実行が中止されます。
    abort;
end;
```

ListWriteToFile

ListWriteToFile 関数は、文字列リストをテキストファイルに書き込みます。各文字列は、テキストファイルの個別の行に表示されます。

メモ・ファイルが既に存在し、既存のファイルが Unicode である場合、ファイルを Unicode として書き出します。それ以外の場合は、ファイルを ANSI として書き出します。

構文

ListWriteToFile (listID, szFileName);

パラメーター

テーブル 80 ・ ListWriteToFile のパラメーター

パラメーター	説明
listID	テキストファイルに書き込む文字列リスト名を指定します。
szFileName	文字列リストを書き込むファイルの完全修飾名を指定します。ファイルが存在しない場合は、作成されます。ファイルが既に存在する場合、上書きされます。

戻り値

テーブル 81 ・ ListWriteToFile の戻り値

戻り値	説明
0	関数は成功しました。
ISERR_LIST_NOSUCHLIST (-501)	指定された ID を持つリストが存在しないことを示します。有効なリストの ID が ListCreate 関数からの戻り値になります。
< 0	関数が失敗しました。

ListWriteToFile の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ListReadFromFile 関数と ListWriteToFile 関数のデモンストレーションを
* デモンストレーションを行います。
*
* ListReadFromFile が呼び出されて Autoexec.bat ファイルを読み込み、
* 表示します。リストが表示されたあと、
* リストの終わりに備考が付け加えられ、新規ファイルへ
* リストが書き込まれます。オリジナルファイルは変更されないままの
* 状態です。
```

```
 *
 * メモ : このスクリプトを適切に実行するため、
 * C ドライブのルートディレクトリに Autoexec.bat と名づけられた
 * バッチファイルが必要です。
 *
¥*---------------------------------------------------------------*/

#define OLD_FILE "C:¥¥Autoexec.bat"
#define NEW_FILE "C:¥¥Autoexec.lst"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ListWriteToFile(HWND);

function ExFn_ListWriteToFile(hMSI)
   LIST   listID;
begin

   // 文字列リストを作成します。
   listID = ListCreate (STRINGLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox (" リストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // ファイルを文字列リストへ読み込みます。
   if (ListReadFromFile (listID, OLD_FILE) < 0) then
      // ListReadFromFile が失敗したことを報告します。
      MessageBox (OLD_FILE + " を読み込むことができませんでした。", SEVERE);
   else
      // リストを表示します。
      SdShowInfoList ("Demo", " リスト例ファイル :", listID);

      // リストへ新規文字列を追加します。
      ListAddString (listID, "IS セットアップが REM を追加しました。", AFTER);

      // ファイルへのリストを書き込みます。
      if (ListWriteToFile (listID, NEW_FILE) < 0) then
         MessageBox (""+ NEW_FILE + " へリストを書き込むことができませんでした。", SEVERE);
      else
         MessageBox (" リストを無事に書き込みました "+ NEW_FILE + "。", INFORMATION);
      endif;
   endif;

   // メモリからリストを削除します。
   ListDestroy(listID);

end;
```

ListWriteToFileEx

ListWriteToFileEx 関数は、文字列リストをテキストファイルに書き込むか、または追加します。各文字列は、テキストファイルの個別の行に表示されます。

構文

ListWriteToFileEx (listID, szFileName, nOptions);

パラメーター

テーブル 82 - ListWriteToFileEx のパラメーター

パラメーター	説明
listID	テキストファイルに書き込む文字列リスト名を指定します。
szFileName	文字列リストを書き込むファイルの完全修飾名を指定します。ファイルが存在しない場合は、作成されます。ファイルが既に存在し、LWTF_OPTION_APPEND_TO_FILE が nOptions に指定されていない場合、ファイルは上書きされます。
nOptions	ファイルのエンコードを指定します。また、文字列リストが存在する場合、それをファイルに追加することができるかどうかを指定します。次の定義済み定数から定数を選択します (複数可)。 • 0 Ñ ファイルが既に存在し、既存のファイルが Unicode である場合、ファイルを Unicode として書き出します。それ以外の場合は、ファイルを ANSI として書き出します。 • LWTF_OPTION_WRITE_AS_UNICODE Ñ ファイルを Unicode として書き出します。 • LWTF_OPTION_WRITE_AS_ANSI Ñ ファイルを ANSI として書き出します。 • LWTF_OPTION_APPEND_TO_FILE Ñ リストの定数を既存のファイルに追加します。ファイルが存在しない場合、関数によりファイルが作成されます (これは、このオプションが指定されなかった場合と同一の動作です)。 ビット単位 OR 演算子 (|) を使用して、これらの 2 つの定数を組み合わせることができます。たとえば、この文字リストの定数を ANSI としてファイルに追加する場合、LWTF_OPTION_WRITE_AS_ANSI 定数と LWTF_OPTION_APPEND_TO_FILE 定数を組み合わせます。

戻り値

テーブル 83 - ListWriteToFileEx の戻り値

戻り値	説明
ISERR_SUCCESS	関数が成功したことを示します。
< ISERR_SUCCESS	関数の実行に失敗したことを示します。

LoadStringFromStringTable

LoadStringFromStringTable 関数は、szID が指定した文字列エントリの値を svString にロードします。svString バッ

ファーは、必要に応じて自動的にサイズ調整して文字列値を格納します。

構文

LoadStringFromStringTable (szID, svString);

パラメーター

テーブル 84 - LoadStringFromStringTable のパラメーター

パラメーター	説明
szID	文字列エントリの文字列 ID を指定します。ID の前にアットマーク (@) をつけないで下さい。
svString	szID によって指定された ID と関連付けられた文字列値を戻します。

戻り値

テーブル 85 - LoadStringFromStringTable の戻り値

戻り値	説明
>= ISERR_SUCCESS	文字列値が s v String へロードされたことを示します。
< ISERR_SUCCESS	文字列 ID が [文字列エディター] ビューで見つからなかったことを示します。

追加情報

LoadStringFromStringTable 関数は、文字列 ID で大文字と小文字を区別しません。したがって、スクリプトで文字列 ID を使用する場合、[文字列エディター] ビューで指定された文字列 ID の大文字小文字の区別と一致させる必要はありません。ただし、大文字と小文字を混ぜて使用することで、ビルド時に、スクリプト内の文字列エントリが [文字列エディター] ビューの対応文字列エントリと一致しないように防ぐことができます。したがって、文字列 ID のすべてのインスタンスで、大文字を使用することをお勧めします。

LoadStringFromStringTable 関数は @ 演算子に相当しますが、次はいくつかの例外です :

- InstallScript ファイル (.rul) を含むプロジェクトをビルドするとき、InstallScript コードに @ 演算子を使用する文字列エントリへの参照を 1 つ以上含まれている場合、InstallShield はビルド時に文字列エントリを検証します。プロジェクトに含まれる InstallScript ファイルの文字列 ID がプロジェクトの文字列エントリの 1 つに定義されていない場合、InstallShield はビルド警告 -7174 を表示します。**LoadStringFromStringTable** を使用する場合、文字列テーブルのエントリはビルド時に検証されません。**LoadStringFromStringTable** を使用するスクリプトでは、文字列が見つからなかったときのエラー処理が既に用意されていると仮定されます。
- 指定された ID が [文字列エディター] ビューに存在しない場合、@ 演算子によって実行時にメッセージ ボックスが表示されます。**LoadStringFromStringTable** は、単に失敗 (ISERR_SUCCESS) を返します。

LOBYTE

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

LOBYTE 関数は、shValue によって指定された 16 ビット整数値から下位バイトを低い順に返します。

構文

LOBYTE (shValue);

パラメーター

テーブル 86 · LOBYTE のパラメーター

パラメーター	説明
shValue	下位バイトを抽出する 16-bit 整数を指定します。

戻り値

この関数は整数の下位バイトを戻します。

LogReadCustomNumber

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

LogReadCustomNumber は szKey が指定したキー名の下にあるログファイルのカスタム ログ記録セクションに格納されている番号を読み込み、nvValue に番号データを戻します。

構文

LogReadCustomNumber(szKey, nvValue);

パラメーター

テーブル 87 ・ LogReadCustomNumber のパラメーター

パラメーター	説明
szKey	ログファイルから読み込まれる数値を認識するキー名を指定します。
nvValue	ログファイルから読み込まれた番号データを戻します。

戻り値

テーブル 88 ・ LogReadCustomNumber の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がログファイルから番号を読み込んだことを示します。
< ISERR_SUCCESS	関数がログファイルから番号を読み込むことができなかったことを示します。

追加情報

LogReadCustomNumber はログ記録が有効か無効かによって影響されることはありません。**LogReadCustomNumber** は LogWriteCustomNumber を使って入力した値と、LogWriteCustomString を使って文字列として入力した数値 (例えば、**"123"**) を読み取ることができます。**LogReadCustomNumber** を数値以外のデータ (たとえば、**"123abc"**) を読み込むのに利用した場合には失敗します。

スクリプトへ、カスタム値を読み込んでこれらの値に基づいたアクションを実行するコードを追加しない限り、カスタム ログ ファイル エントリは製品のメンテナンスまたはアンインストールには影響しません。**LogReadCustomString** はログファイルのメンテナンス / アンインストールセクション (つまり、インストールされたファイルや作成されたレジストリエントリといったデータをインストールが自動的に書き込むセクションで、メンテナンスやアンインストール中にそこからデータを自動的に読み込みます) からデータを読み込むことはできません。

製品が完全にアンインストールされる場合のみスクリプト コードを実行するには、if-then ステートメントを使用します :

```
if REMOVEALLMODE!=0 then
   /* このコードは、アンインストール中にのみ実行されます。*/
endif;
```

特定の機能がアンインストールされるときに、特定のアンインストール アクションを実行するには、機能の UnInstalling イベントを適切なコードでオーバーライドします。

LogReadCustomNumber の例

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

```
//---------------------------------------------------------------------------
//
```

```
// InstallShield スクリプトの例
//
//  Demonstrates the LogWriteCustomString, LogWriteCustomNumber,
LogReadCustomString、および LogReadCustomNumber functions のデモンストレーションを行います。
//
//--------------------------------------------------------------------------

// インストール中に、ログ ファイルにカスタム データを書き込みます
function OnMoved()
    LIST listDrives;
    NUMBER nDriveCount, nvIgnore;
    STRING svDate;
begin

if (!MAINTENANCE) then
    // 現在の日付を取得します
    GetSystemInfo(DATE, nvIgnore, svDate);

    // 利用可能なドライブ名の現在のカウントを取得します
    listDrives = ListCreate(STRINGLIST);
    GetValidDrivesList(listDrives, -1, -1);
    nDriveCount = ListCount(listDrives);
    ListDestroy(listDrives);

    // カスタム データを .ilg ログ ファイルに書き込みます
    LogWriteCustomString("InstallDate", svDate);
    LogWriteCustomNumber("DriveCount", nDriveCount);
endif;

end;


// 完全アンインストール中に、ログ ファイルからカスタム データを読み出します
function OnMoving( )
    NUMBER nvDriveCount;
    STRING svInstallDate;
begin

if (REMOVEALLMODE) then
    LogReadCustomNumber("DriveCount", nvDriveCount);
    LogReadCustomString("InstallDate", svInstallDate);
    SprintfBox(INFORMATION, " カスタム ログ データ ",
        " インストール中のドライブカウントは " +
        " %d、日付は %s でした。",
        nvDriveCount, svInstallDate);
endif;

end;
```

LogReadCustomString

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

LogReadCustomString は szKey が指定したキー名の下にあるログファイルのカスタムログ記録セクションに格納されている文字列を読み込み、svValue に文字列データを戻します。

構文

LogReadCustomString(szKey, svValue);

パラメーター

テーブル 89 · LogReadCustomString のパラメーター

パラメーター	説明
szKey	ログファイルから読み込まれる文字列を認識するキー名を指定します。
svValue	ログファイルから読み込まれた文字列データを戻します。

戻り値

テーブル 90 · LogReadCustomString の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がログファイルから文字列を読み込んだことを示します。
< ISERR_SUCCESS	関数がログファイルから文字列を読み込むことができなかったことを示します。

追加情報

LogReadCustomString はログ記録が有効か無効かによって影響されることはありません。**LogReadCustomString** は、LogWriteCustomString または LogWriteCustomNumber のどちらかを使って入力した値を読み取ることができます。

スクリプトへ、カスタム値を読み込んでこれらの値に基づいたアクションを実行するコードを追加しない限り、カスタム ログ ファイル エントリは製品のメンテナンスまたはアンインストールには影響しません。**LogReadCustomString** はログファイルのメンテナンス / アンインストールセクション (つまり、インストールされたファイルや作成されたレジストリエントリといったデータをインストールが自動的に書き込むセクションで、メンテナンスやアンインストール中にそこからデータを自動的に読み込みます) からデータを読み込むことはできません。

***メモ ·** リリース ウィザードの [パスワードと著作権] パネルの **[セットアップ初期化中にパスワード ダイアログ ボックスを表示する]** チェック ボックスを選択した場合、または [リリース] ビューで ″パスワード ダイアログの表示 ″ プロパティを [はい] に設定した場合に、OnCheckMediaPassword イベント ハンドラー関数のデフォルト コードは szKey を MEDIA_PASSWORD_KEY に設定した状態で **LogWriteCustomString** を呼び出し、パスワード保護されているインストールのパスワード (エンドユーザーが入力) を保存して、メンテナンスやアップグレード操作の際に同じパスワードを再入力しなくてもよいようにします。そのデフォルトコードはまた、szKey を MEDIA_PASSWORD_KEY に設定して **LogWriteCustomString** を呼び出し、エンド ユーザーに対してパスワードを問い合わせる前にそのパスワードが既に保存済みかどうかを確認します。*

製品が完全にアンインストールされる場合のみスクリプト コードを実行するには、if-then ステートメントを使用します :

```
if REMOVEALLMODE!=0 then
   /* このコードは、アンインストール中にのみ実行されます。*/
```

```
endif;
```

特定の機能がアンインストールされるときに、特定のアンインストール アクションを実行するには、機能の UnInstalling イベントを適切なコードでオーバーライドします。

LogReadCustomString の例

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

```
//---------------------------------------------------------------------------
//
// InstallShield スクリプトの例
//
//  Demonstrates the LogWriteCustomString, LogWriteCustomNumber,
LogReadCustomString、および LogReadCustomNumber functions のデモンストレーションを行います。
//
//---------------------------------------------------------------------------

// インストール中に、ログ ファイルにカスタム データを書き込みます
function OnMoved()
    LIST listDrives;
    NUMBER nDriveCount, nvIgnore;
    STRING svDate;
begin

if (!MAINTENANCE) then
    // 現在の日付を取得します
    GetSystemInfo(DATE, nvIgnore, svDate);

    // 利用可能なドライブ名の現在のカウントを取得します
    listDrives = ListCreate(STRINGLIST);
    GetValidDrivesList(listDrives, -1, -1);
    nDriveCount = ListCount(listDrives);
    ListDestroy(listDrives);

    // カスタム データを .ilg ログ ファイルに書き込みます
    LogWriteCustomString("InstallDate", svDate);
    LogWriteCustomNumber("DriveCount", nDriveCount);
endif;

end;


// 完全アンインストール中に、ログ ファイルからカスタム データを読み出します
function OnMoving( )
    NUMBER nvDriveCount;
    STRING svInstallDate;
begin

if (REMOVEALLMODE) then
    LogReadCustomNumber("DriveCount", nvDriveCount);
    LogReadCustomString("InstallDate", svInstallDate);
    SprintfBox(INFORMATION, " カスタム ログ データ ",
        " インストール中のドライブカウントは " +
        " %d、日付は %s でした。 ",
```

```
        nvDriveCount, svInstallDate);
endif;

end;
```

LogWriteCustomNumber

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

LogWriteCustomNumber は szValue が指定した数値を nValue が指定したキー名の下にあるカスタムログ記録セクションのログファイルへ書き込みます。

構文

```
LogWriteCustomNumber( szKey, nValue );
```

パラメーター

テーブル 91 ・ LogWriteCustomNumber のパラメーター

パラメーター	説明
szKey	ログファイルへ書き込まれる数値を認識するキー名を指定します。LogWriteCustomString または LogWriteCustomNumber のどちらを使って書き込まれたかに関わらず、特定のログファイルへ書き込まれたすべてのキー名は固有でなくてはなりません。カスタムログ記録セクションに既に存在するキー名を指定した場合、その値が上書きされます。
nValue	ログファイルへ書き込まれる番号を指定します。

戻り値

テーブル 92 ・ LogWriteCustomNumber の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数が数値をログファイルへ書き込んだこと示します。
< ISERR_SUCCESS	関数が数値をログファイルへ書き込むことができなかったことを示します。

追加情報

LogWriteCustomNumber への呼び出しは、ログ記録が無効になっている場合失敗します。

スクリプトへ、カスタム値を読み込んでこれらの値に基づいたアクションを実行するコードを追加しない限り、カスタム ログ ファイル エントリは製品のメンテナンスまたはアンインストールには影響しません。**LogWriteCustomNumber** はログ ファイルのメンテナンス / アンインストールセクション (つまり、インストールされたファイルや作成されたレジストリ エントリといったデータをインストールが自動的に書き込むセクションで、メンテナンスやアンインストール中にはそこからデータを自動的に読み込みます) にデータを書き込むむことはできません。

製品が完全にアンインストールされる場合のみスクリプト コードを実行するには、if-then ステートメントを使用します :

```
if REMOVEALLMODE!=0 then
   /* このコードは、アンインストール中にのみ実行されます。*/
endif;
```

特定の機能がアンインストールされるときに、特定のアンインストール アクションを実行するには、機能の UnInstalling イベントを適切なコードでオーバーライドします。

LogWriteCustomNumber の例

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

```
//---------------------------------------------------------------------------
//
// InstallShield スクリプトの例
//
//  Demonstrates the LogWriteCustomString, LogWriteCustomNumber,
LogReadCustomString、および LogReadCustomNumber functions のデモンストレーションを行います。
//
//---------------------------------------------------------------------------

// インストール中に、ログ ファイルにカスタム データを書き込みます
function OnMoved()
   LIST listDrives;
   NUMBER nDriveCount, nvIgnore;
   STRING svDate;
begin

if (!MAINTENANCE) then
   // 現在の日付を取得します
   GetSystemInfo(DATE, nvIgnore, svDate);

   // 利用可能なドライブ名の現在のカウントを取得します
   listDrives = ListCreate(STRINGLIST);
   GetValidDrivesList(listDrives, -1, -1);
   nDriveCount = ListCount(listDrives);
   ListDestroy(listDrives);

   // カスタム データを .ilg ログ ファイルに書き込みます
   LogWriteCustomString("InstallDate", svDate);
   LogWriteCustomNumber("DriveCount", nDriveCount);
endif;

end;


// 完全アンインストール中に、ログ ファイルからカスタム データを読み出します
function OnMoving( )
   NUMBER nvDriveCount;
   STRING svInstallDate;
begin

if (REMOVEALLMODE) then
```

```
    LogReadCustomNumber("DriveCount", nvDriveCount);
    LogReadCustomString("InstallDate", svInstallDate);
    SprintfBox(INFORMATION, " カスタム ログ データ ",
      " インストール中のドライブカウントは " +
      " %d、日付は %s でした。",
      nvDriveCount, svInstallDate);
endif;

end;
```

LogWriteCustomString

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

LogWriteCustomString は szValue が指定した文字列を szKey が指定したキー名の下にあるカスタムログ記録セクションのログファイルへ書き込みます。

構文

```
LogWriteCustomString( szKey, szValue );
```

パラメーター

テーブル 93 · LogWriteCustomString のパラメーター

パラメーター	説明
szKey	ログファイルへ書き込まれる文字列を認識するキー名を指定します。LogWriteCustomString または LogWriteCustomNumber のどちらを使って書き込まれたかに関わらず、特定のログファイルへ書き込まれたすべてのキー名は固有でなくてはなりません。カスタムログ記録セクションに既に存在するキー名を指定した場合、その値が上書きされます。
szValue	ログファイルへ書き込む数値を指定します。

戻り値

テーブル 94 · LogWriteCustomString の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数が文字列をログファイルへ書き込んだこと示します。
< ISERR_SUCCESS	関数が文字列をログファイルへ書き込むことができなかったことを示します。

追加情報

LogWriteCustomString への呼び出しは、ログ記録が無効になっている場合失敗します。

スクリプトへ、カスタム値を読み込んでこれらの値に基づいたアクションを実行するコードを追加しない限り、カスタム ログ ファイル エントリは製品のメンテナンスまたはアンインストールには影響しません。**LogWriteCustomString** はログ ファイルのメンテナンス / アンインストールセクション (つまり、インストールされたファイルや作成されたレジストリ エントリといったデータをインストールが自動的に書き込むセクションで、メンテナンスやアンインストール中にはそこからデータを自動的に読み込みます) にデータを書き込むむことはできません。

メモ・リリース ウィザードの [パスワードと著作権] パネルの [セットアップ初期化中にパスワード ダイアログ ボックスを表示する] チェック ボックスを選択した場合、または [リリース] ビューで " パスワード ダイアログの表示 " プロパティを [はい] に設定した場合に、OnCheckMediaPassword イベント ハンドラー関数のデフォルト コードは szKey を MEDIA_PASSWORD_KEY に設定した状態で LogWriteCustomString を呼び出し、パスワード保護されているインストールのパスワード (エンドユーザーが入力) を保存して、メンテナンスやアップグレード操作の際に同じパスワードを再入力しなくてもよいようにします。そのデフォルトコードはまた、szKey を MEDIA_PASSWORD_KEY に設定して LogWriteCustomString を呼び出し、エンド ユーザーに対してパスワードを問い合わせる前にそのパスワードが既に保存済みかどうかを確認します。

製品が完全にアンインストールされる場合のみスクリプト コードを実行するには、if-then ステートメントを使用します :

```
if REMOVEALLMODE!=0 then
   /* このコードは、アンインストール中にのみ実行されます。*/
endif;
```

特定の機能がアンインストールされるときに、特定のアンインストール アクションを実行するには、機能の UnInstalling イベントを適切なコードでオーバーライドします。

LogWriteCustomString の例

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

```
//---------------------------------------------------------------------------
//
// InstallShield スクリプトの例
//
//  Demonstrates the LogWriteCustomString, LogWriteCustomNumber,
LogReadCustomString、および LogReadCustomNumber functions のデモンストレーションを行います。
//
//---------------------------------------------------------------------------

// インストール中に、ログ ファイルにカスタム データを書き込みます
function OnMoved()
   LIST listDrives;
   NUMBER nDriveCount, nvIgnore;
   STRING svDate;
begin

if (!MAINTENANCE) then
   // 現在の日付を取得します
   GetSystemInfo(DATE, nvIgnore, svDate);

   // 利用可能なドライブ名の現在のカウントを取得します
```

```
    listDrives = ListCreate(STRINGLIST);
    GetValidDrivesList(listDrives, -1, -1);
    nDriveCount = ListCount(listDrives);
    ListDestroy(listDrives);

    // カスタム データを .ilg ログ ファイルに書き込みます
    LogWriteCustomString("InstallDate", svDate);
    LogWriteCustomNumber("DriveCount", nDriveCount);
endif;

end;


// 完全アンインストール中に、ログ ファイルからカスタム データを読み出します
function OnMoving( )
    NUMBER nvDriveCount;
    STRING svInstallDate;
begin

if (REMOVEALLMODE) then
    LogReadCustomNumber("DriveCount", nvDriveCount);
    LogReadCustomString("InstallDate", svInstallDate);
    SprintfBox(INFORMATION, " カスタム ログ データ ",
        " インストール中のドライブカウントは " +
        " %d、日付は %s でした。",
        nvDriveCount, svInstallDate);
endif;

end;
```

LongPathFromShortPath

LongPathFromShortPath 関数を使って短いファイル名を対応する長いファイル名に変換します。

長いファイル名の詳細については、「長いファイル名」を参照して下さい。

構文

```
LongPathFromShortPath ( svPath );
```

パラメーター

テーブル 95 - LongPathFromShortPath のパラメーター

パラメーター	説明
svPath	短いファイル名を指定して、関連付けられた長いファイル名を戻します。

戻り値

テーブル 96 - LongPathFromShortPath の戻り値

戻り値	説明
0	関数が成功したことを示します。
< 0	関数の実行に失敗したことを示します。

LongPathFromShortPath の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* LongPathToShortPath 関数 と LongPathFromShortPath 関数の
* デモンストレーションを行います。
*
* まず、LongPathToShortPath を呼び出して長いパスを短いパスへ
* 変換します。そして LongPathFromShortPath を呼び出し、
* 短いパスを長いパスへ変換します。それぞれの結果は
* メッセージボックスに表示されます。
*
¥*---------------------------------------------------------------*/

#define LONG_PATH "C:¥¥Program files"
#define TITLE "LongPathToShortPath & LongPathFromShortPath"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_LongPathFromShortPath(HWND);

function ExFn_LongPathFromShortPath(hMSI)
   STRING svPath, szTitle, szMsg;
begin

   // ユーザーに長いパスの入力を要求します。
```

```
    szMsg    = " 既存する長いパスを選択してください :";
    AskPath (szMsg, LONG_PATH, svPath);

    // 長いパスを表示します。
    szMsg = " 長いパスは次のように表示されます : ¥n¥n%s";
    SprintfBox (INFORMATION, TITLE, szMsg, svPath);

    // 長いパスを短いパスへ変換します。
    if (LongPathToShortPath (svPath) < 0) then;
        MessageBox ("LongPathToShortPath が失敗しました。", SEVERE);
        abort;
    else
        // 短いパスを表示します。
        szMsg = " 短いパスは次のように表示されます : ¥n¥n%s";
        SprintfBox (INFORMATION, TITLE, szMsg, svPath);
    endif;

    // 長いパスを短いパスへ戻します。
    if (LongPathFromShortPath (svPath) < 0) then
        MessageBox ("LongPathFromShortPath が失敗しました。.", SEVERE);
    else
        // 元に戻された長いパスを表示します。
        szMsg = " 元に戻された長いパスは次のように表示されます : ¥n¥n%s";
        SprintfBox (INFORMATION, TITLE, szMsg, svPath);
    endif;

end;
```

LongPathToQuote

LongPathToQuote 関数は長いファイル名の周りに二重引用符を配置、または削除します。

長いファイル名の詳細については、「長いファイル名」を参照して下さい。

空白を含む長いファイル名はコマンドラインに渡す前に二重引用符で囲んでください。LongPathToShortPath 関数を使用して短いファイル名に変換する場合、長いファイル名から二重引用符を削除する必要があります。これを怠った場合、長いファイル名が残ったままになります。

メモ・この関数は、ファイル名に空白文字があった場合のみ引用符を追加します。たとえば、C:¥¥ThisismyApp は空白文字を含まない長いファイル名なので、引用符は追加されません。

構文

```
LongPathToQuote ( svPath, nParameter );
```

パラメーター

テーブル 97 ・ LongPathToQuote のパラメーター

パラメーター	説明
svPath	長いファイル名を指定し、nParameter で渡した値に従ってその名前と引用符または引用符無しで戻します。
nParameter	引用符を長いパスに追加するか削除するかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ TRUE Ñ 引用符が長いパスに追加されます。 ・ FALSE Ñ 引用符が長いパスから削除されます。

戻り値

テーブル 98 ・ LongPathToQuote の戻り値

戻り値	説明
0	関数が成功したことを示します。
< 0	関数の実行に失敗したことを示します。

LongPathToQuote の例

メモ ・ *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* LongPathToQuote 関数のデモンストレーションを行います。
*
* このスクリプトは LongPathToQuote を呼び出して長いファイル名の周りに
* 二重引用符を配置します。この結果はダイアログ ボックスに
* 表示されます。そして、LongPathToQuote が再び呼び出され、
* 引用符が削除された結果がダイアログに
* 表示されます。
*
¥*---------------------------------------------------------------------*/

// ベースとなるパス ( 長いファイル名 ) の定数を定義します。
```

```
#define BASE_PATH "C:¥¥Program Files"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_LongPathToQuote(HWND);

function ExFn_LongPathToQuote(hMSI)
   STRING  svPath, szMainDirectory, szMsg;
begin

   // LongPathToQuote を呼び出すパラメーターをセットアップします。
   svPath = BASE_PATH;

   // svPath の長いファイル名の周りに二重引用符を配置します。
   if (LongPathToQuote (svPath, TRUE) < 0) then
      MessageBox ("LongPathToQuote への最初の呼び出しに失敗しました。", SEVERE);
      abort;
   endif;

   // svPath の引用された長いファイル名を表示します。
   szMsg = " 引用された長いファイル名 :¥n¥n" + svPath;
   MessageBox (szMsg, INFORMATION);

   // svPath の長いファイル名から二重引用符を削除します。
   if (LongPathToQuote (svPath, FALSE) < 0) then
      MessageBox ("LongPathToQuote への 2 回目の呼び出しに失敗しました。", SEVERE);
      abort;
   endif;

   // 引用符を削除した長いファイル名を表示します。
   szMsg = " 引用符無しの長いファイル名は下の通りです : ¥n¥n" + svPath;

   MessageBox (szMsg, INFORMATION);

end;
```

LongPathToShortPath

LongPathToShortPath 関数を使って長いファイル名を対応する短いファイル名に変換します。パラメーター svPath は完全パスと相対パスのどちらも可能で、ファイル名を含むこともできます。ただし、そのパスが指定するフォルダーまたはファイルはターゲットシステム上に存在しなくてはなりません。

長いファイル名の詳細については、「長いファイル名」を参照して下さい。

構文

```
LongPathToShortPath ( svPath );
```

パラメーター

テーブル 99 - LongPathToShortPath のパラメーター

パラメーター	説明
svPath	長いファイル名を指定して、関連付けられた短いファイル名を戻します。 ***メモ -*** *LongPathToShortPath は長いファイル名から末尾の円記号を削除します。*

戻り値

テーブル 100 - LongPathToShortPath の戻り値

戻り値	説明
0	関数が成功したことを示します。
< 0	関数の実行に失敗したことを示します。

追加情報

LongPathToShortPath は指定したフォルダーまたはファイルがターゲットシステムで検出された場合のみ成功するので、相対パスを設定する前に現在のフォルダーを設定する必要があります。たとえば、svPath に相対パス **InstallShield** が含まれ、これが **C:¥Program Files** フォルダーに存在する場合、現在のフォルダーが **C:¥Program Files** でない限り、セットアップはこれを見つけることができません。**LongPathToShortPath** を呼び出す前に必要に応じて ChangeDirectory 関数を使い、ターゲットフォルダーまたはパスが検出されるよう現在のフォルダーを変更してください。

LongPathToShortPath の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* LongPathToShortPath 関数 と LongPathFromShortPath 関数の
* デモンストレーションを行います。
*
* まず、LongPathToShortPath を呼び出して長いパスを短いパスへ
* 変換します。そして LongPathFromShortPath を呼び出し、
* 短いパスを長いパスへ変換します。それぞれの結果は
* メッセージボックスに表示されます。
*
```

```
¥*---------------------------------------------------------------*/

#define LONG_PATH "C:¥¥ プログラムファイル "
#define TITLE "LongPathToShortPath & LongPathFromShortPath"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_LongPathToShortPath(HWND);

function ExFn_LongPathToShortPath(hMSI)
  STRING svPath, szTitle, szMsg;
begin

   // ユーザーに長いパスの入力を要求します。
   szMsg   = " 既存する長いパスを選択してください :";
   AskPath (szMsg, LONG_PATH, svPath);

   // 長いパスを表示します。
   szMsg = " 長いパスは次のように表示されます : ¥n¥n%s";
   SprintfBox (INFORMATION, TITLE, szMsg, svPath);

   // 長いパスを短いパスへ変換します。
   if (LongPathToShortPath (svPath) < 0) then;
      MessageBox ("LongPathToShortPath が失敗しました。", SEVERE);
      abort;
   else
      // 短いパスを表示します。
      szMsg = " 短いパスは次のように表示されます : ¥n¥n%s";
      SprintfBox (INFORMATION, TITLE, szMsg, svPath);
   endif;

   // 長いパスを短いパスへ戻します。
   if (LongPathFromShortPath (svPath) < 0) then
      MessageBox ("LongPathFromShortPath が失敗しました。.", SEVERE);
   else
      // 元に戻された長いパスを表示します。
      szMsg = " 元に戻された長いパスは次のように表示されます : ¥n¥n%s";
      SprintfBox (INFORMATION, TITLE, szMsg, svPath);
   endif;

end;
```

LOWORD

LOWORD 関数は、lValue が指定した 32-bit 整数値から低位単語 (2 バイト) を抽出します。

構文

```
LOWORD ( lValue );
```

パラメーター

テーブル 101 ・ LOWORD のパラメーター

パラメーター	説明
lValue	下位 2 バイトを抽出する元の 32-bit 整数を指定します。

戻り値

この関数は整数の低位単語 (低位 2 バイト) を戻します。

LOWORD の例

基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * HIWORD と LOWORD のデモンストレーションを行います。
 *
 * このスクリプト例は、値から低位の単語と高位の単語を取得する
 * HIWORD と LOWORD の利用法を説明します。
¥*---------------------------------------------------------------*/

#define TITLE_TEXT "LOWORD/HIWORD の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_LOWORD(HWND);

function ExFn_LOWORD(hMSI)
  STRING  szMsg;
  NUMBER  nData, nLOWORD, nHIWORD;
begin

  nData = 305419896; // 16 進値 : 12345678
  // 低位の単語、22136 (16 進法 : 5678) を取得します。

  nLOWORD = LOWORD (nData);

  // 高位の単語、4660 (16 進法 : 1234) を取得します。
  nHIWORD  = HIWORD (nData);

  // 結果を表示します。
  szMsg  = "LOWORD: %ld¥nHIWORD: %ld";
  SprintfBox (INFORMATION, TITLE_TEXT, szMsg, nLOWORD, nHIWORD);

end;
```

MaintenanceStart

MaintenanceStart 関数は、メンテナンスやアンインストールの初期化の際に使用されるレジストリ キーとそれに関連付けられている値を作成し、[プログラムの追加と削除] にアプリケーションの情報を提供します。この関数は OnMoveData イベントハンドラー関数のデフォルト コードによって呼び出されます。

メモ・*InstallScript エンジンは、現在、レジストリの 64 ビット部分にある [プログラムの追加と削除] 情報の読み書きをサポートしません。そのため、REGDB_OPTIONS システム変数を使った REGDB_OPTION_WOW64_64KEY オプションをこのレジストリ関数で使用することはできません。REGDB_OPTION_WOW64_64KEY オプションを有効にしても、レジストリエントリがこの関数によって作成される場所に影響しません。*

MaintenanceStart は、以下の値をアプリケーションアンインストール レジストリキーの下に作成します。

テーブル 102・アプリケーションアンインストールレジストリキーの値

値名	値データ
Comments	ヌル以外の場合、システム変数 IFX_PRODUCT_COMMENTS の値。ヌルの場合は関数はエントリを作成しません。
Contact	ヌル以外の場合、システム変数 IFX_PRODUCT_SUPPORT_CONTACT の値。ヌルの場合は関数はエントリを作成しません。
DisplayIcon	ヌル以外の場合、システム変数 IFX_PRODUCT_ICON の値。ヌルの場合は関数はエントリを作成しません。
DisplayName	ヌル以外の場合、システム変数 IFX_PRODUCT_NAME の値。ヌルの場合は関数はエントリを作成しません。
DisplayVersion	システム変数 IFX_PRODUCT_VERSION の値。
HelpLink	ヌル以外の場合、システム変数 IFX_PRODUCT_SUPPORT_URL の値。ヌルの場合は関数はエントリを作成しません。
HelpTelephone	ヌル以外の場合、システム変数 IFX_PRODUCT_SUPPORT_PHONE の値。ヌルの場合は関数はエントリを作成しません。
InstallDate	*年 月 日* 形式のインストールが実行される日付。
InstallLocation	システム変数 TARGETDIR の値。
InstallSource	システム変数 SRCDIR の値。
言語	システム変数 SELECTED_LANGUAGE の値。
LogFile	<DISK1TARGET>¥Setup.ilg
LogMode	システム変数 MAINT_OPTION の値。

テーブル 102 - アプリケーションアンインストールレジストリキーの値 (続き)

値名	値データ
ModifyPath	ADDREMOVE_HIDECHANGEOPTION が FALSE で、次の 2 つの条件のいずれかが TRUE の場合、システム変数 UNINSTALL_STRING の値。 • ADDREMOVE_HIDEREMOVEOPTION がゼロではない場合 • ADDREMOVE_HIDEREMOVEOPTION および ADDREMOVE_COMBINEDBUTTON の両方が FALSE の場合 その他の場合、関数はエントリを作成しません。
NoModify	システム変数 ADDREMOVE_HIDEREMOVEOPTION の値。
NoRemove	システム変数 ADDREMOVE_HIDEREMOVEOPTION の値。
NoRepair	ADDREMOVE_HIDECHANGEOPTION または ADDREMOVE_HIDEREMOVEOPTION がゼロ以外の場合は 1、ゼロの場合は関数はエントリを作成しません。
ProductGuid	システム変数 PRODUCT_GUID の値。
ProductId	ヌル以外の場合、システム変数 IFX_PRODUCT_REGISTEREDSERIALNUM の値。ヌルの場合は関数はエントリを作成しません。
発行者	ヌル以外の場合、システム変数 IFX_COMPANY_NAME の値。ヌルの場合は関数はエントリを作成しません。
Readme	ヌル以外の場合、システム変数 IFX_PRODUCT_README の値。ヌルの場合は関数はエントリを作成しません。
RegCompany	ヌル以外の場合、システム変数 IFX_PRODUCT_REGISTEREDCOMPANY の値。ヌルの場合は関数はエントリを作成しません。
RegOwner	ヌル以外の場合、システム変数 IFX_PRODUCT_REGISTEREDOWNER の値。ヌルの場合は関数はエントリを作成しません。
SystemComponent	システム変数 ADDREMOVE_SYSTEMCOMPONENT がゼロ以外の場合は 1、ゼロの場合は関数はエントリを作成しません。
UninstallString	システム変数 UNINSTALL_STRING の値。さらに、ADDREMOVE_HIDEREMOVEOPTION が FALSE で次の 2 つの条件のどちらかが true の場合、システム変数 ADDREMOVE_STRING_REMOVEONLY の値がこのレジストリ データに追加されます。 • ADDREMOVE_HIDECHANGEOPTION がゼロではない場合 • ADDREMOVE_HIDECHANGEOPTION および ADDREMOVE_COMBINEDBUTTON の両方が FALSE の場合
URLInfoAbout	ヌル以外の場合、システム変数 IFX_PRODUCT_URL の値。ヌルの場合は関数はエントリを作成しません。

テーブル 102 ・ アプリケーションアンインストールレジストリキーの値 (続き)

値名	値データ
URLUpdateInfo	ヌル以外の場合、システム変数 IFX_PRODUCT_UPDATE_URL の値。ヌルの場合は関数はエントリを作成しません。
Version	DisplayVersion 値にあるデータに対応するパックされた DWORD。
VersionMajor	Version 値のデータの最初のバイト。
VersionMinor	Version 値のデータの 2 番目のバイト。

構文

MaintenanceStart ( );

パラメーター

なし

戻り値

テーブル 103 ・ MaintenanceStart の戻り値

戻り値	説明
0	この関数によりレジストリキーとその関連値が正常に作成されたことを示します。
< 0	この関数がレジストリキーとその関連値の作成に失敗したことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

InstallScript インストールは常に、Uninstall キーに VersionMajor および VersionMinor レジストリ値を作成します。これは、InstallShield 2016 で作成された新しいインストール、および InstallShield 2009 以前からアップグレードされたインストールに適用します。以前、InstallShield 2009 以前では、InstallScript インストールが作成する値の名前は MajorVersion および MinorVersion でしたが、今後は作成しません。

MaintenanceStart 関数が呼び出されると、レジストリに VersionMajor および VersionMinor 値名が作成されます。MajorVersion および MinorVersion 値名が既存する場合、デフォルトでこれらは削除されます。ターゲット システムから MajorVersion および MinorVersion 値名を削除したくない場合、REGDB_OPTION_NO_DELETE_OLD_MAJMIN_VERSION という名前の新しい REGDB_OPTIONS オプションを使用します。古い値名のみを使用し続けたい場合は、**MaintenanceStart** が返された後に新しいバージョンを削除しなくてはなりません。

MediaGetData

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MediaGetData 関数はファイルメディアライブラリ についての情報を読み出します。

MediaGetData calls MediaGetDataEx(szMediaSource, nInfo, nvResult, svResult, FALSE).**MediaGetData** のパラメーターと戻り値については、「MediaGetDataEx」を参照してください。

構文

MediaGetData (szMediaSource, nInfo, nvResult, svResult);

MediaGetDataEx

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

MediaGetDataEx 関数は文字列エントリに格納されているメディア情報を含めた、ファイルメディアライブラリについての情報を読み出します。

構文

MediaGetDataEx (szMediaSource, nInfo, nvResult, svResult, bCheckStringTable);

パラメーター

テーブル 104 ・ MediaGetDataEx のパラメーター

パラメーター	説明
szMediaSource	情報を読み出すファイルメディアライブラリの名前を指定します。一般的に、この引数の値はシステム変数 MEDIA です。
nInfo	読み出す情報の種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ MEDIA_FIELD_ADDREMOVE_NOMODIFY Ñ [一般情報] ビューの ″ 変更ボタンを無効にする ″ 設定で指定した設定を読み出します。値は nvResult に戻されます。0 は [いいえ] に対応し、1 は [はい] に対応します。 ・ MEDIA_FIELD_ADDREMOVE_NOREMOVE Ñ [一般情報] ビューの ″ 削除ボタンを無効にする ″ 設定で指定した設定を読み出します。値は nvResult に戻されます。0 は [いいえ] に対応し、1 は [はい] に対応します。 ・ MEDIA_FIELD_COMPANY_NAME Ñ [一般情報] ビューで指定した会社名を読み出します。会社名は、svResult で戻されます。 ・ MEDIA_FIELD_MEDIA_FLAGS– 指定されたファイル メディア ライブラリの形式を読み出します。nvResult で戻された数値には、次のビットフラグの中から該当する値が含まれます。 MEDIA_FLAG_FORMAT_DIFFERENTIAL: 差分ファイルメディアライブラリを示します。 MEDIA_FLAG_UPDATEMODE_SUPPORTED: アップデート対応のファイルメディアライブラリを示します。このフラグは常に設定されています。 ・ MEDIA_FIELD_PREVIOUS_VERSIONS–[リリース] ビューの ″ サポートされているバージョン ″ プロパティまたはリリース ウィザードの [アップデート] パネルで指定したバージョン情報を読み出します。バージョン情報は svResult で戻されます。 ・ MEDIA_FIELD_PRODUCT_COMMENTS– [一般情報] ビューの ″ARP コメント ″ 設定で指定したコメントを読み出します。値は、svResult で戻されます。 ・ MEDIA_FIELD_PRODUCT_EXE –[一般情報] ビューで指定した実行可能ファイル名を読み出します。実行可能名は、svResult で戻されます。

テーブル 104 ・ MediaGetDataEx のパラメーター (続き)

パラメーター	説明
nInfo (続く)	• MEDIA_FIELD_PRODUCT_ICON—[一般情報] ビューの " 表示アイコン " 設定で指定した表示アイコンを読み出します。値は、svResult で戻されます。 • MEDIA_FIELD_PRODUCT_EXE —[一般情報] ビューで指定した実行可能ファイル名を読み出します。製品名は、svResult に戻ります。 • MEDIA_FIELD_PRODUCT_README—[一般情報] ビューの "README" 設定で指定した Readme ファイルを読み出します。値は、svResult で戻されます。 • MEDIA_FIELD_PRODUCT_SUPPORT_CONTACT—[一般情報] ビューの " サポート連絡先 " 設定で指定したサポート連絡先を読み出します。値は、svResult で戻されます。 • MEDIA_FIELD_PRODUCT_SUPPORT_PHONE— [一般情報] ビューの " サポート電話番号 " 設定で指定したサポート電話番号を読み出します。値は、svResult で戻されます。 • MEDIA_FIELD_PRODUCT_SUPPORT_URL Ñ [一般情報] ビューの " サポート URL" 設定で指定したサポート URL を読み出します。値は、svResult で戻されます。 • MEDIA_FIELD_PRODUCT_UPDATE_URL Ñ [一般情報] ビューの " 製品アップデート URL" 設定で指定した製品アップデート URL を読み出します。値は、svResult で戻されます。 • MEDIA_FIELD_PRODUCT_URL Ñ [一般情報] ビューの " 発行元 / 製品 URL" 設定で指定した発行元 / 製品 URL を読み出します。値は、svResult で戻されます。 • MEDIA_FIELD_PRODUCT_VERSION Ñ [一般情報] ビューで指定したバージョン情報を読み出します。バージョン情報は svResult で戻されます。 • MEDIA_FIELD_TARGETDIR Ñ [一般情報] ビューで指定した TARGETDIR の値を読み出します。
bCheckStringTable	svResult が InstallShield に入力した文字列を戻す (FALSE) のか、または InstallShield へ入力した文字列が 文字列 ID の場合はその ID に関連付けられた文字列を戻す (TRUE) のかを指定します。bCheckStringTable が TRUE になっているが、InstallShield へ入力した文字列が文字列 ID ではない場合は、svResult は InstallShield に入力した文字列を戻します。

戻り値

テーブル 105 ・ MediaGetDataEx の戻り値

戻り値	説明
0	MediaGetData は要求された情報を読み出しました。
< 0	MediaGetData は要求された情報の読み出しに失敗しました。

MessageBeep

MessageBeep 関数はデフォルトシステムサウンドを再生します。

ヒント・オーディオファイルを再生するのに PlayMMedia を呼び出して頭出しすることも可能です。

構文

MessageBeep (nReserved);

パラメーター

テーブル 106 ・ MessageBeep のパラメーター

パラメーター	説明
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。

戻り値

この関数に戻り値はありません。

MessageBeep の例

基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * MessageBeep 関数のデモンストレーションを行います。
 *
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_MessageBeep(HWND);

function ExFn_MessageBeep(hMSI)
begin

   // デフォルト システムサウンドを再生します。
   MessageBeep (0);

   // メッセージボックスを表示します。
   MessageBox ("サウンドをお待ちください。", INFORMATION);

   // デフォルト システムサウンドを再生します。
   MessageBeep (0);

end;
```

MessageBox

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

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

MessageBox 関数は、メッセージ、そのメッセージの性質を示すアイコン (情報、警告、または深刻な問題)、および [OK] ボタンが含まれたダイアログを表示します。デフォルトのタイトルは nType の値によって変わり、この値はアイコンの種類も示します。たとえば、nType に INFORMATION を渡すと、「**情報**」というタイトルがタイトルバーに表示されます。**MessageBox** の前に SetDialogTitle を呼び出して設定しない限り、デフォルトのタイトルは空白です。タイトルが空白の場合、メッセージ ボックスのタイトル バー テキストとして (IFX_SETUP_TITLE からの) 製品名が表示されます。

この関数は Microsoft Windows API **MessageBox** を使用します。メッセージ ボックスのサイズと位置を決定するのは、インストールではなく、操作環境です。また、動作環境によって [OK] ボタンのテキストがローカル言語 (実行する OS の言語) で生成されます。このボタンのテキストを変更することはできません。さまざまな **MessageBox** タイプの使用方法の詳細については、該当する Windows SDK で **MessageBox** Windows API 関数の説明を参照してください。

Windows メッセージボックス定数を使用する場合は、以下の点に注意してください。

- 一部の Windows **MessageBox** タイプ定数は、*InstallShield Program Files フォルダー*¥**Script**¥**Isrt**¥**Include** フォルダーにある **ISRTWindows.h** ファイルで既に定義されています。このファイルは、スクリプトに **Ifx.h** を含めると、自動的にインストールに含まれます。**ISRTWindows.h** に定義されている定数を再定義する必要はありません。再定義すると、コンパイラ警告が発生します。どの定数が定義済みかを判別するには、**ISRTWindows.h** ファイルを参照します。
- **ISRTWindows.h** に定義されていない定数を使用する場合は、インストールスクリプトの宣言ブロックに必ずそれらを定義しなければなりません (#define を使用)。通常 C++ プログラムの一部である **Windows.h** ファイルを、単純挿入することはできません。未定義定数に割り当てる必要がある値は、通常は、該当する Windows SDK や開発ツール付属のインクルードファイルに含まれています。(Microsoft Visual C++ の場合、ほとんどの定数は *InstallShield Program Files フォルダー*¥**Script**¥**Resource** フォルダーにある **Winuser.h** ファイルに含まれています。)
- 単一のインストールで Windows のメッセージボックス定数と InstallShield のメッセージボックス定数を併用することはできません。OR 演算子 (|) を使用して InstallShield メッセージボックス定数と Windows メッセージボックス定数と組み合わせても、Windows メッセージボックス定数は無視されます。
- Windows プラットフォームの中には、特定の Windows メッセージボックススタイルをサポートしていないものがあります。インストール先のオペレーティングシステムで特定のスタイルがサポートされているかどうかを判断するには、該当する Windows SDK を参照してください。
- **MessageBox** 関数で Windows のメッセージボックススタイルが使用されている場合、そのメッセージボックスのキャプション (タイトル) は「**インストール**」になります。別のキャプションを表示する必要がある場合は、MessageBoxEx 関数を使用してください。

構文

```
MessageBox (szMsg, nType);
```

パラメーター

テーブル 107 ・ MessageBox のパラメーター

パラメーター	説明
szMsg	表示するタイトルを指定します。InstallShield では、メッセージのテキストは、メッセージボックスに収まるよう自動改行されません。メッセージが長すぎて 1 行に収まらないときは、文字列の適切な箇所に改行エスケープ文字 (¥n) を埋め込んで改行してください。
nType	作成するメッセージボックスの種類や、メッセージボックスに表示するアイコンの種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ INFORMATION ・ WARNING ・ SEVERE このパラメーターではあらゆる種類の Windows API MessageBox が指定できます。OR 演算子を使って複数のスタイルを論理的に組み合わせることで、必要なタイプの MessageBox を作成することができます。

戻り値

戻り値は、Microsoft Windows 標準のメッセージボックスのスタイルを使用しない限り、重要ではありません。これらのスタイルを使用する場合、戻り値は MessageBox API 関数の戻り値と同じになります。

追加情報

MessageBox 関数によって表示されるダイアログは、スキンを使って表示することはできません。スキンの指定に関わらず、同じように表示されます。

MessageBox の例

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

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

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
```

```
*
* MessageBox 関数のデモンストレーションを行います。
*
* このスクリプトは 3 つのメッセージボックスにそれぞれ別々の
* メッセージとアイコンを表示します。
*
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_MessageBox(HWND);

function ExFn_MessageBox(hMSI)
   STRING szMsg;
begin

   // 情報アイコンを示すメッセージボックスを表示します。
   szMsg = " プログラム例をインストールします。";
   MessageBox (szMsg, INFORMATION);

   // 警告アイコンを示すメッセージボックスを表示します。
   szMsg = " このバージョンをインストールすると以前のバージョンを置換します。";
   MessageBox (szMsg, WARNING);

   // 問題アイコンを示すメッセージボックスを表示します。
   szMsg = " このアプリケーションをフロッピードライブにインストールすることはできません。";
   MessageBox (szMsg, SEVERE);

end;
```

MessageBoxEx

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

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

MessageBoxEx 関数は、メッセージ、そのメッセージの性質を示すアイコン (情報、警告、または深刻な問題)、および [OK] ボタンが含まれたダイアログを表示します。タイトルは s z Caption が設定します。

この関数では Microsoft Windows API MessageBox が使用されます。メッセージボックスのサイズと位置を決定するのは、InstallShield ではなく、操作環境です。また、動作環境によって [OK] ボタンのテキストがローカル言語 (実行する OS の言語) で生成されます。このボタンのテキストを変更することはできません。さまざまな Windows MessageBox タイプの使用方法の詳細については、該当する Windows SDK で MessageBox Windows API 関数の説明を参照してください。

Windows メッセージボックス定数を使用する場合は、以下の点に注意してください。

- 一部の Windows **MessageBox** タイプ定数は、*InstallShield Program Files フォルダー*¥**Script**¥**Isrt**¥**Include** フォルダーにある **ISRTWindows.h** ファイルで既に定義されています。このファイルは、スクリプトに **Ifx.h** を含めると、自動的にインストールに含まれます。**ISRTWindows.h** に定義されている定数を再定義する必要はありません。

再定義すると、コンパイラ警告が発生します。どの定数が定義済みかを判別するには、**ISRTWindows.h** ファイルを参照します。

- **ISRTWindows.h** に定義されていない定数を使用する場合は、インストールスクリプトの宣言ブロックに必ずそれらを定義しなければなりません (#define を使用)。通常 C++ プログラムの一部である **Windows.h** ファイルを、単純挿入することはできません。未定義定数に割り当てる必要がある値は、通常は、該当する Windows SDK や開発ツール付属のインクルードファイルに含まれています。(Microsoft Visual C++ の場合、ほとんどの定数は *InstallShield Program Files フォルダー*¥Script¥Resource フォルダーにある **Winuser.h** ファイルに含まれています。)
- 単一のインストールで Windows のメッセージボックス定数と InstallShield のメッセージボックス定数を併用することはできません。OR 演算子を使用して InstallShield メッセージボックス定数と Windows メッセージボックス定数と組み合わせても、Windows メッセージボックス定数は無視されます。
- Windows プラットフォームの中には、特定の Windows メッセージボックススタイルをサポートしていないものがあります。インストール先のオペレーティングシステムで特定のスタイルがサポートされているかどうかを判断するには、該当する Windows SDK を参照してください。

構文

```
MessageBoxEx( szMsg, szCaption, nType );
```

パラメーター

テーブル 108 ・ MessageBoxEx のパラメーター

パラメーター	説明
szMsg	表示するタイトルを指定します。InstallShield では、メッセージのテキストは、メッセージボックスに収まるよう自動改行されません。メッセージが長すぎて 1 行に収まらないときは、文字列の適切な箇所に改行エスケープ文字 (¥n) を埋め込んで改行してください。
szCaption	表示するタイトルを指定します。このパラメーターでヌル文字列 ("") を渡した場合、システム変数 IFX_SETUP_TITLE の値がダイアログのタイトルとして利用されます。
nType	作成するメッセージボックスの種類や、メッセージボックスに表示するアイコンの種類を指定します。このパラメーターで、下記に示す定義済み定数のうちの一つを渡します (エクスプローラーシェルのアイコンを表示します)。 ・ INFORMATION ・ WARNING ・ SEVERE このパラメーターではあらゆる種類の Windows API MessageBox が指定できます。OR 演算子 (|) を使って複数のスタイルを論理的に組み合わせることで、必要な種類の MessageBox を作成することができます。

戻り値

戻り値は、Microsoft Windows 標準のメッセージボックスのスタイルを使用しない限り、重要ではありません。これらのスタイルを使用する場合、戻り値は MessageBox API 関数の戻り値と同じになります。

追加情報

MessageBoxEx 関数によって表示されるメッセージは、スキンを使って表示することはできません。スキンの指定に関わらず、同じように表示されます。

NumToStr

NumToStr 関数は、数値を文字列に変換します。

構文

```
NumToStr ( svString, nValue );
```

パラメーター

テーブル 109 ・ NumToStr のパラメーター

パラメーター	説明
svString	nValue に相当する文字列を戻します。
nValue	文字列へ変換する数値を指定します。

戻り値

テーブル 110 ・ NumToStr

戻り値	説明
0	関数が数値を文字列に変換したことを示します。
< 0	関数が数値を文字列に変換できなかったことを示します。

NumToStr の例

基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * NumToStr 関数のデモンストレーションを行います。
 *
 * このスクリプトは NumToStr を呼び出して、MessageBox 関数で
 * 表示できるよう、システム上で有効な空き容量の数値を
 * 文字列に変換します。
 *
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_NumToStr(HWND);

function ExFn_NumToStr(hMSI)
   STRING  svString;
   NUMBER  nSpace, nResult;
begin

   // C ドライブ上のディスク空き容量を取得します。
   nSpace  = GetDiskSpace ("C:");

   // 数値を文字列に変換します。
```

```
    nResult = NumToStr (svString, nSpace);

    if (nResult < 0) then
        MessageBox ("NumToStr が失敗しました。", SEVERE);
    else
        // C ドライブ上のディスク空き容量を表示します。
        MessageBox (svString + " バイトが C ドライブ上で空きです。", INFORMATION);
    endif;

end;
```

OpenFile

OpenFile 関数は既存のテキストファイルまたはバイナリファイルを開きます。ファイルを開く前に、OpenFileMode を呼び出してファイルモードを設定しなくてはなりません。

ヒント・次の事項に注意してください。

- *テキストファイルを開いた後、GetLine と WriteLine を呼び出してファイルから読み取りまたはファイルへの書き込みを行います。GetLine または WriteLine を使ってファイルからの読み取りまたはファイルへの書き込みが終わったら、CloseFile 関数でファイルを閉じる必要があります。*
- *バイナリファイルを開いた後、ReadBytes と WriteBytes を呼び出してファイルから読み取りまたはファイルへの書き込みを行います。*
- *SeekBytes を利用してバイナリファイルへ書き込む前にファイルポインターを配置することができます。*
- *FileGrep 関数、FileInsertLine 関数、そして FileDeleteLine 関数を利用して、テキストの検索、読み取り、そして書き込みを行うこともできます。しかしながら、これらの関数ではファイルを開いたり閉じたりする必要がありません (内部処理されます)。*
- *CreateFile を使ってファイルを作成してください。CreateFile は新規ファイルを、テキストファイルでは追加モードで開いた状態に、そしてバイナリファイルでは読み取り / 書き込みモードで開いた状態にします。*

構文

OpenFile (nvFileHandle, szPath, szFileName);

パラメーター

テーブル 111 ・ OpenFile のパラメーター

パラメーター	説明
nvFileHandle	開かれたファイルのファイルハンドルを戻します。その他のファイル関連の InstallScript 関数を呼び出す場合、ファイルを識別するためにこのハンドルを使って下さい。
szPath	開くファイルのパス (ドライブ指定を含むことも可能) を指定します。OpenFileMode を呼び出してファイルモードを FILE_MODE_NORMAL または FILE_MODE_BINARYREADONLY に設定したあと、このパラメーターで有効な URL (Uniform Resource Locator) を指定することもできます。CGI または ASP 要求 (たとえば、"http://www.mydomain.net/login.asp?name=Me&pwd=wow") を渡す場合、その応答はメモリへ送られ、ReadBytes が読み取ります。URL が有効かどうかを確認するには、Is(VALID_PATH, szURL) を呼び出します。このパラメーターでヌル文字列 ("") を渡すと、関数が失敗します。
szFileName	開くファイルへの、ドライブ指定またはパス無しの非完全修飾名を指定します。このパラメーターでヌル文字列 ("") を渡すと、関数が失敗します。

戻り値

テーブル 112 ・ OpenFile の戻り値

戻り値	説明
0	関数がファイルを開いたことを示します。
< 0	関数がファイルを開くことができなかったことを示します。

OpenFile の例

基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* OpenFile 関数 と CloseFile 関数 のデモンストレーションを行います。
*
* OpenFile はファイルを開くために呼び出されて、リストへ
* 読み込まれます。リストが表示されます。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*      既存のディレクトリ内の既存ファイルを参照するように
*      設定します。
*
¥*---------------------------------------------------------------------*/
```

```
#define EXAMPLE_FILE "Readme.txt"
#define EXAMPLE_DIR "C:¥¥Windows"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_OpenFile(HWND);

function ExFn_OpenFile(hMSI)
   STRING  svLine;
   NUMBER nvFileHandle;
   LIST    listID;
begin

   // ファイル モードを通常に設定します。
   OpenFileMode (FILE_MODE_NORMAL);

   // テキスト ファイルを開きます。
   if (OpenFile (nvFileHandle, EXAMPLE_DIR, EXAMPLE_FILE) < 0) then
      MessageBox ("OpenFile が失敗しました。", SEVERE);
      abort;
   endif;

   // 空白文字列リストを作成します。
   listID = ListCreate (STRINGLIST);

   // テキスト ファイルの行を文字列リストへ読み出します。
   while GetLine (nvFileHandle, svLine) = 0
      ListAddString (listID, svLine, AFTER);
   endwhile;

   // ファイルを閉じます。
   if (CloseFile (nvFileHandle) < 0) then
      MessageBox ("CloseFile が失敗しました。", SEVERE);
   endif;

   // ファイルから読み出されたテキストを表示します。
   SdShowInfoList ("", "", listID);

end;
```

OpenFileMode

OpenFileMode 関数は作成または開くファイルのモードを設定します。パラメーター nMode として渡す式は、次の中から 1 つのファイルモードを設定します :

- 追加モードの ANSI または Unicode テキストファイル。
- 読み取り専用モードの ANSI または Unicode テキストファイル。
- 読み取り / 書き込みモードのバイナリファイル。
- 読み取り専用モードのバイナリファイル。

ファイルモードを設定した後、OpenFile を呼び出して既存のファイルを開く、または CreateFile を呼び出して新規ファイルを作成して開きます。

構文

OpenFileMode (nMode);

パラメーター

テーブル 113 ・ OpenFileMode のパラメーター

パラメーター	説明
nMode	ファイルを開くのに利用するモードを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ FILE_MODE_APPEND Ñ この定数を利用すると、追加モードでテキストファイルを開くまたは作成することが可能です。**OpenFile** を利用してファイルが追加モードで開かれた場合、ファイルポインターはファイルの終わりにあります。WriteLine 関数を利用してファイルの終わりに行を追加することができます。**CreateFile** を利用して作成されたファイルは新規 (空白) なので、追加される行はファイルの始めに書き込まれます。追加モードでファイルを開いた場合、**GetLine** 関数への呼び出しは失敗しますのでご注意ください。これは、ファイル ポインターがファイルの終わりにあるためです。 ・ FILE_MODE_APPEND_UNICODE Ñ **CreateFile** が新規のファイルを、デフォルトの ANSI ではなく、Unicode ファイルで作成する必要があることを示します。このオプションは、既存ファイルが開かれるときには影響しません。既存ファイルは常に既存のフォーマットで開かれ、保存されます。ファイルをひとつのフォーマットから別のフォーマットへ変換するには、適切な nOptions を使って **ListWriteToFileEx** と **ListWriteToFile** を使用します。 ・ FILE_MODE_NORMAL Ñ この定数を利用すると、読み取り専用モードでテキストファイルを開くことが可能です。**OpenFile** を使ってファイルが読み取り専用モードで開かれた場合、ファイル ポインターはファイルの始めにあります。**GetLine** 関数を利用してファイルから読み取ることができます。FILE_MODE_NORMAL が有効な状態で **CreateFile** を使って作成したファイルは、実際 FILE_MODE_APPEND モードで作成されます。 ・ FILE_MODE_BINARY Ñ この定数を利用すると、読み取り / 書き込みモードでバイナリ ファイルを開くまたは作成することが可能です。**OpenFile** または **CreateFile** と共にファイルをバイナリ モードで作成または開いた場合、**ReadBytes** を呼び出してファイルから読み取り、**WriteBytes** を呼び出してファイルへ書き込むことが可能です。バイナリファイルへの書き込みは現在のファイルポインター位置で始まります。新規に作成または開かれたファイルでは、その始まり部分の位置 0 となります。**OpenFile** を利用して開かれた既存バイナリファイルへ追加する場合、書き込む前に **SeekBytes** を使ってファイルポインター位置を決めなくてはなりません。CD-ROM または読み取り専用ドライブ上のファイルを開く場合、**OpenFileMode** を呼び出してファイルモードを読み取り専用 (FILE_MODE_BINARYREADONLY) に設定します。 ・ FILE_MODE_BINARYREADONLY Ñ この定数はバイナリファイルを読み取り専用モードで開く点以外は FILE_MODE_BINARY 定数と同じです。CD-ROM または読み取り専用ドライブ上でバイナリファイルを開くとき、この定数を使ってバイナリファイルを開きます。CD-ROM、または読み取り専用ドライブ上でバイナリファイルを開くと FILE_MODE_BINARY が失敗します。

戻り値

テーブル 114 ・ OpenFileMode の戻り値

戻り値	説明
0	関数がファイルモードを設定したことを示します。
< 0	関数がファイルモードを設定できなかったことを示します。

OpenFileMode の例

基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* OpenFileMode 関数のデモンストレーションを行います。
*
* このスクリプトはテキストファイルを読み取り専用モード (FILE_MODE_NORMAL) で
* します。そしてファイルの最初の行を読み出して
* 表示します。
*
* 次に、ファイルをバイナリモードで開き、ファイルポインターを 15 番目のバイトに
* 配置し、ファイルから 28 バイトを読み取ります。
*
* メモ : このスクリプトを正しく実行するため、
*    既存ディレクトリ内の既存ファイルが参照されるよう、
*    プリプロセッサ定数を設定しなくてはなりません。
*
¥*---------------------------------------------------------------------*/

#define EXAMPLE_DIR  "C:¥¥"
#define EXAMPLE_TEXT_FILE "ISExampl.txt"
#define EXAMPLE_BIN_FILE  "ISExampl.bin"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_OpenFileMode(HWND);

function ExFn_OpenFileMode(hMSI)
   STRING svLine, svString;
   NUMBER nvFileHandle;
begin

   // ファイル モードを通常に設定します。
   OpenFileMode (FILE_MODE_NORMAL);

   // ファイルを開きます。
   if (OpenFile (nvFileHandle, EXAMPLE_DIR, EXAMPLE_TEXT_FILE) < 0) then
      MessageBox ("OpenFile が失敗しました。", SEVERE);
      abort;
```

```
    endif;

    // テキストファイルの最初の行を取得します。
    GetLine (nvFileHandle, svLine);

    // 行を表示します。
    MessageBox (svLine, INFORMATION);

    // ファイルを閉じます。
    CloseFile (nvFileHandle);

    // ファイルモードをバイナリ読み取り / 書き込みモードに設定します。
    OpenFileMode (FILE_MODE_BINARY);

    // ファイルを開きます。
    if (OpenFile (nvFileHandle, EXAMPLE_DIR, EXAMPLE_BIN_FILE) < 0) then
        MessageBox ("OpenFile が失敗しました。", SEVERE);
    else
        // ファイルポインターをバイト 15 へ移動します。
        SeekBytes (nvFileHandle, 15, FILE_BIN_START);

        // バイナリファイルから 28 バイトを svString へ読み取ります。
        if (ReadBytes (nvFileHandle, svString, 0, 28) < 0) then
            MessageBox ("ReadBytes が失敗しました。", SEVERE);
        else
            // 文字列を表示します。
            MessageBox (svString, INFORMATION);
        endif;

        // バイナリファイルを閉じます。
        CloseFile (nvFileHandle);
    endif;

end;
```

ParsePath

ParsePath 関数は、既存パスの指定した部分を読み取ります。関数は短いパス、長いパス、そして特定のファイル名を含む UNC パスあるいは特定のファイル名を含まない UNC といった、任意の有効なパスで利用できます。これらは、この関数を使って解析できるパス例の一部です。

- ¥Path1¥Path2¥Filename.exe
- FileName
- Filename.exe
- ¥Path1¥Path2¥Filename
- D:
- D:¥
- ¥¥Server Name¥Share Name¥Share Directory
- Any other legal DOS path

構文

ParsePath (svReturnString, szPath, nOperation);

パラメーター

テーブル 115 ・ ParsePath のパラメーター

パラメーター	説明
svReturnString	nOperation が指定する nOperation にあるパスの一部を戻します。
szPath	解析するパスを指定します。ファイル名が含まれないパスを指定する場合、ParsePath に渡す前にパスの最後に円記号を追加する必要があります。追加しないとパスの最後の部分がファイル名と解釈されてしまいます。
nOperation	パスのどの要素を戻すかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ DIRECTORY Ñ ディスクドライブ名を省いたパスとファイル名が svReturnString に戻されることを示します。UNC パスと共にこのオプションを利用した場合、ParsePath はサーバーや共有デバイス名を省いたパス、そしてファイル名が指定されている場合はそれも省いて戻します。たとえば、UNC パス ¥¥TheServer¥TheSharedDevice¥TheApp¥TheFile.exe は svReturnString に ¥TheApp¥ と戻されます。 ・ DISK Ñ ディスクドライブ指定 (ドライブ名の後にコロン) が svReturnString へ戻されることを示します。このオプションを UNC パスと共に利用した場合、ParsePath はサーバーと共有デバイス名を戻します。たとえば、UNC パス ¥¥TheServer¥TheSharedDevice¥TheApp¥TheFile.exe は svReturnString に ¥¥TheServer¥TheSharedDevice と戻されます。 ・ EXTENSION_ONLY Ñ ファイル拡張子が svReturnString に戻されることを示します。ここにピリオドは含まれません。 ・ FILENAME Ñ 完全ファイル名 (ファイル拡張子を含む) が svReturnString に戻されることを示します。 ・ FILENAME_ONLY Ñ ファイル名のみ (ファイル拡張子を含まない) が svReturnString に戻されることを示します。 ・ PATH Ñ ファイル名を省いたパスが svReturnString に戻されることを示します。このオプションは、ドライブ指定 (szPath で指定された場合) が戻されたパスに含まれる点で DIRECTORY とは異なります。szPath が UNC パスを指定する際、サーバーや共有デバイス名が戻されたパスに含まれます。たとえば、UNC パス ¥¥TheServer¥TheSharedDevice¥TheApp¥TheFile.exe は svReturnString に ¥¥TheServer¥TheSharedDevice¥TheApp¥ と戻されます。

戻り値

テーブル 116 ・ ParsePath の戻り値

戻り値	説明
0	関数がパス文字列を解析したことを示します。
< 0	関数がパス文字列を解析できなかったことを示します。

ParsePath の例

基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ParsePath 関数のデモンストレーションを行います。
*
* ParsePath は完全修飾ファイル名から様々な情報を読み取るために
* 6 回呼び出されます。
*
¥*---------------------------------------------------------------*/

#define EXAMPLE_PATH "C:¥¥Windows¥¥Readme.txt"
#define TITLE_TEXT   "ParsePath の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ParsePath(HWND);

function ExFn_ParsePath(hMSI)
   STRING szMsg, svReturnString;
begin

   // パスからドライブ名を取得します。
   if (ParsePath (svReturnString, EXAMPLE_PATH, DISK) < 0) then
      MessageBox ("ParsePath が失敗しました ", SEVERE);
   else
      szMsg = "nOperation = DISK¥n¥nParsed Path: %s";
      SprintfBox (INFORMATION, TITLE_TEXT , szMsg, svReturnString);
   endif;

   // 完全パスを取得します。
   szMsg = "nOperation = PATH¥n¥nParsed Path: %s";

   if (ParsePath (svReturnString, EXAMPLE_PATH, PATH) < 0) then
      MessageBox ("ParsePath が失敗しました ", SEVERE);
   else
      SprintfBox (INFORMATION, TITLE_TEXT , szMsg, svReturnString);
   endif;

   // ディレクトリ名を取得します。
   if (ParsePath (svReturnString, EXAMPLE_PATH, DIRECTORY) < 0) then
      MessageBox ("ParsePath が失敗しました ", SEVERE);
   else
      szMsg = "nOperation = DIRECTORY¥n¥nParsed Path: %s";
      SprintfBox (INFORMATION, TITLE_TEXT , szMsg, svReturnString);
   endif;

   // ファイル名と拡張子を取得します。
   if (ParsePath (svReturnString, EXAMPLE_PATH, FILENAME) < 0) then
      MessageBox ("ParsePath が失敗しました ", SEVERE);
   else
```

```
    szMsg = "nOperation = FILENAME¥n¥nParsed Path: %s";
    SprintfBox (INFORMATION, TITLE_TEXT , szMsg, svReturnString);
endif;

// 拡張子なしのファイル名を取得します。
if (ParsePath (svReturnString, EXAMPLE_PATH, FILENAME_ONLY) < 0) then
    MessageBox ("ParsePath が失敗しました ", SEVERE);
else
    szMsg= "nOperation = FILE_NAME_ONLY¥n¥nParsed Path: %s";
    SprintfBox (INFORMATION, TITLE_TEXT , szMsg, svReturnString);
endif;

// ファイル拡張子を取得します。
if (ParsePath (svReturnString, EXAMPLE_PATH, EXTENSION_ONLY) < 0) then
    MessageBox ("ParsePath が失敗しました ", SEVERE);
else
    szMsg    = "nOperation = EXTENSION_ONLY¥n¥n 解析済みのパス : %s";
    SprintfBox (INFORMATION, TITLE_TEXT , szMsg, svReturnString);
endif;

end;
```

ParseUrl

ParseUrl 関数は指定された URL の一部を読み出します。関数は任意の有効な URL に利用できます。

構文

```
ParseUrl ( szUrl, pISUrlComponents );
```

パラメーター

テーブル 117 ・ ParseUrl のパラメーター

パラメーター	説明
szUrl	解析する URL を指定します。
pISUrlComponents	指定された URL の一部を含む ISURL_COMPONENTS データ構造へのポインターを戻します。

戻り値

テーブル 118 ・ ParseUrl の戻り値

戻り値	説明
0	関数がパス文字列を解析したことを示します。
< 0	関数がパス文字列を解析できなかったことを示します。

ParseUrl の例

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
*  ParseUrl 関数のデモンストレーションを行います。
*
¥*---------------------------------------------------------------------*/

function OnBegin()
    ISURL_COMPONENTS ISUrlComponents;
    STRING szUrl;
begin
    szUrl =
        "http://myusername:mypassword@www.mydomain.com:8080/" +
        "myfolder/mypage.asp?mykey=myvalue";
    ParseUrl ( szUrl, &ISUrlComponents );
    SprintfBox ( INFORMATION , "ParseUrl" ,
        "scheme = %s¥nscheme number = %ld¥nusername = %s¥npassword = %s" +
        "¥nhostname = %s¥nport = %ld¥nurlpath = %s¥nextrainfo = %s",
        ISUrlComponents.szScheme, ISUrlComponents.nInternetScheme,
        /* nInternetScheme は HTTP では 3 、そして HTTPS では 4 となります。*/
        ISUrlComponents.szUserName, ISUrlComponents.szPassword,
        ISUrlComponents.szHostName, ISUrlComponents.nInternetPort,
        ISUrlComponents.szUrlPath, ISUrlComponents.szExtraInfo );
    abort;
end;
```

そのスクリプトは、次の出力を生成します :

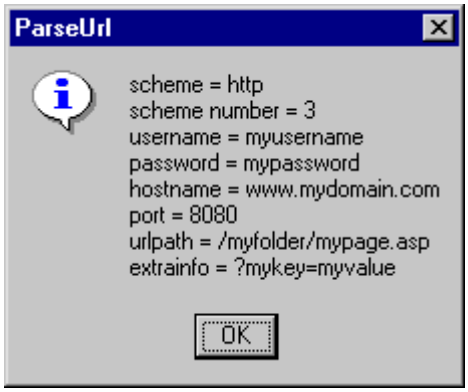

図 1: ParseUrl の出力

PathAdd

PathAdd 関数は、パスをパスバッファーの検索パスに追加します。この関数を使うと、ディレクトリの場所をパスバッファーの既存のディレクトリに相対して指定できます。さらに、ディレクトリをパスバッファーの最初または最後のディレクトリとして追加できます。

この関数は Autoexec.bat ファイルまたはパス環境変数のパス ステートメントとは一切関係ありません。この関数はパスバッファー上でのみ動作します。これによって検索パスを構築、変更および操作しやすくなります。さまざまなバッチファイル関数を使って、変更したパス文字列を Autoexe.bat ファイルに追加できます。

構文

PathAdd (szDir, szRefDir, bRefDir, bPosition);

パラメーター

テーブル 119 ・ PathAdd のパラメーター

パラメーター	説明
szDir	パスバッファーに追加するパスを指定します。 ***メモ ・** 現在パスバッファーにある szDir のディレクトリを指定すると、InstallShield はこれを重複せず、デフォルトのディレクトリの場所も変更されません。InstallShield はディレクトリ名の最後にあるバックスラッシュを無視します。*
szRefDir	追加する新しいパスに相対した、現在のパスバッファーのパスを指定します。
bRefDir	szRefDir が完全修飾パスかどうかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ FULL Ñ szRefDir は完全修飾パスで、ドライブ指定と完全なパスを含みます。 ・ PARTIAL Ñ szRefDir はディレクトリ名のみで、ドライブまたはパス情報を含みません。
bPosition	szRefDir に相対する szDir の挿入場所を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ AFTER Ñ szDir を szRefDir の後に挿入するよう指定します。szRefDir でヌル文字列 (″″) が指定されている場合、szDir はパス バッファーのパスの最後に挿入されます。 ・ BEFORE Ñ szDir を szRefDir の前に挿入するよう指定します。szRefDir でヌル文字列 (″″) が指定されている場合、szDir はパス バッファーのパスの前に挿入されます。

戻り値

テーブル 120 ・ PathAdd の戻り値

戻り値	説明
0	ディレクトリがパスバッファーに正常に追加されたことを示します。
< 0	ディレクトリがパスバッファーに追加できなかったことを示します。

PathAdd の例

基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* PathAdd 関数のデモンストレーションを行います。
*
```

```
＊このスクリプト例は、パスバッファーの検索パスへパスを追加する方法を
＊説明します。初めに、検索パスを使ったパスバッファーの初期化を
＊行います。そして、検索パスの前に新しいパスを
＊ビルドします。次に、検索バッファーから修正済み検索パスを取得し、
＊それを表示します。
＊
＊そして修正された検索パスを使ってパスバッファーを初期化し、
＊パスバッファーの最後のパスの前に新しいパスを追加
＊します。最後に、検索バッファーから修正済み検索パスを取得し、
＊それを表示します。
＊
＊メモ : 最初の PathAdd ステートメントの後に PathGet  が呼び出されます。
＊    これは、デモンストレーションの目的でパスバッファーの値が
＊    表示されるようにするためです。実際のスクリプトでは、
＊    PathGet を呼び出す前にバッファーの検索パスのビルドで
＊    終了します。
＊
¥*---------------------------------------------------------------*/

#define TITLE " パスバッファーの例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_PathAdd(HWND);

function ExFn_PathAdd(hMSI)
   STRING svSearchPath;
begin

   // 検索パスをセットアップし、PathSet へのパラメーターとして渡します。
   svSearchPath = "C:¥¥DOS;C:¥¥Windows;C:¥¥Temp";

   // パスバッファーを初期化します。
   PathSet (svSearchPath);

   // 初期検索パスを表示します。
   SprintfBox (INFORMATION,TITLE,
           " 開始検索パスは %s.",svSearchPath);

   // C:¥MSOffice を検索パスの最初のパスとして追加します。
   if (PathAdd("C:¥¥MSOffice", "", FULL, BEFORE) < 0) then
      MessageBox ("C:¥¥MsOffice をパスバッファーへ追加することができませんでした。", SEVERE);
      abort;
   endif;

   // パスバッファーから検索パスを取得します。この呼び出しで
   // 割り当てられたメモリを開放します。
   PathGet (svSearchPath);

   // 検索パスを表示します。
   // svSearchPath は C:¥MSOffice、C:¥DOS、C:¥Windows、そして C:¥Temp を含みます。
   SprintfBox (INFORMATION,TITLE,
           "C:¥¥MSOffice が最初のパスの前に追加されました。¥n¥n 検索パスは %s です。",
           svSearchPath);

   // 検索パスを格納するパスバッファーを初期化します。
   PathSet (svSearchPath);
```

```
    // パスバッファーで C:¥Temp の前に C:¥APP2 を追加します。
    if (PathAdd ("C:¥¥APP2", "Temp", PARTIAL, BEFORE) < 0) then
        MessageBox ("C:¥¥APP2 をパスバッファーへ追加することができませんでした。", SEVERE);
        abort;
    endif;

    // パスバッファーから検索パスを取得します。この呼び出しで
    // 割り当てられたメモリを開放します。
    PathGet (svSearchPath);

    // 修正済み検索パスを表示します。
    // svSearchPath は C:¥MSOffice、C:¥DOS、C:¥Windows、C:¥App2、そして C:¥Temp を含みます。
    SprintfBox (INFORMATION,TITLE,
            "C:¥¥APP2 が C:¥¥Temp の前に追加されました。¥n¥n 検索パスは %s です。",
            svSearchPath);

end;
```

PathDelete

PathDelete 関数はパスバッファーの特定のディレクトリを削除します。ディレクトリ名は指定するか、完全修飾パスを入力します。

この関数は Autoexec.bat ファイルまたはパス環境変数のパス ステートメントとは一切関係ありません。この関数はパスバッファー上でのみ動作します。これによって検索パスを構築、変更および操作しやすくなります。

ヒント・パスバッファーの内容を取得するには PathGet を呼び出してください。パスバッファーの内容を設定するには PathSet を呼び出してください。

構文

```
PathDelete (szDir, bDir);
```

パラメーター

テーブル 121 ・ PathDelete のパラメーター

パラメーター	説明
szDir	パスバッファーから削除するパスを指定します。
bDir	szRefDir が完全修飾パスかどうかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ FULL Ñ szRefDir は完全修飾パスで、ドライブ指定と完全なパスを含みます。 ・ PARTIAL Ñ szRefDir はディレクトリ名のみで、ドライブまたはパス情報を含みません。

戻り値

テーブル 122 ・ PathDelete の戻り値

戻り値	説明
0	ディレクトリがパスバッファーから正常に削除されたことを示します。
< 0	ディレクトリがパスバッファーから削除できなかったことを示します。

PathDelete の例

基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*

* InstallShield スクリプトの例
*
* PathDelete 関数のデモンストレーションを行います。
*
* このスクリプト例は、パスバッファーからパスを削除する方法を説明
* します。初めに、検索パスを使ったパスバッファーの初期化を
* 行います。そして、"Temp" へのリファレンスを含むすべてのパスを
* 削除します。次に、検索バッファーから修正済み検索パスを取得し、
* それを表示します。
*
* そして修正された検索パスを使ってパスバッファーを初期化し、
* 特定のパスを削除します。最後に、検索バッファーから
* 検索パスを取得してそれを表示します。
*
* メモ : 最初の PathDelete ステートメントの後に PathGet が呼び出されます。
*    これは、デモンストレーションの目的でパスバッファーの値が
*    表示されるようにするためです。実際のスクリプトでは、
*    PathGet を呼び出す前にバッファーの検索パスのビルドで
*    終了します。
```

```
 *
¥*---------------------------------------------------------------------*/

#define TITLE " パスバッファーの例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_PathDelete(HWND);

function ExFn_PathDelete(hMSI)
   STRING svSearchPath;
begin

   // 検索パスをセットアップし、PathSet へのパラメーターとして渡します。
   svSearchPath = "C:¥¥DOS;C:¥¥WINDOWS;C:¥¥TEMP;" +
                  "C:¥¥EXAMPLE¥¥SOURCE;D:¥¥WORK¥¥TEMP";

   // パスバッファーを初期化します。
   PathSet (svSearchPath);

   // 初期検索パスを表示します。
   SprintfBox (INFORMATION,TITLE,
           " 開始検索パスは %s.",svSearchPath);

   // パスバッファーから C:¥Temp を削除します。
   if (PathDelete ("TEMP", PARTIAL) < 0) then
      MessageBox ("PathDelete への最初の呼び出しに失敗しました。", SEVERE);
   endif;

   // パスバッファーから検索パスを取得します。この呼び出しで
   // 割り当てられたメモリを開放します。
   PathGet (svSearchPath);

   // 検索パスを表示します。
   // svSearchPath は C:¥DOS;C:¥WINDOWS;C:¥EXAMPLE¥SOURCE を含みます。
   SprintfBox (INFORMATION, TITLE,
           "'Temp' を参照するすべてのパスが削除されました。¥n¥n 検索パスは %s です。",
           svSearchPath);

   // パスバッファーを再びセットアプします。
   PathSet (svSearchPath);

   // パスバッファーから C:¥¥EXAMPLE¥¥SOURCE を削除します。
   if (PathDelete ("C:¥¥EXAMPLE¥¥SOURCE", FULL) < 0) then
      MessageBox ("PathDelete への 2 回目の呼び出しに失敗しました。",SEVERE);
   endif;

   // パスバッファーから検索パスを取得します。この呼び出しで
   // 割り当てられたメモリを開放します。
   PathGet (svSearchPath);

   // 検索パスを表示します。
   // svSearchPath は C:¥DOS;C:¥WINDOWS を含みます。
   SprintfBox (INFORMATION, TITLE,
           "C:¥¥EXAMPLE¥¥SOURCE が削除されました。¥n¥nPath は %s です。",
           svSearchPath);

end;
```

PathFind

PathFind 関数は、特定のディレクトリのパスバッファーを検索します。ディレクトリは、完全パスかディレクトリ名だけで指定できます。

この関数は Autoexec.bat ファイルまたはパス環境変数のパス ステートメントとは一切関係ありません。この関数はパスバッファー上でのみ動作します。これによって検索パスを構築、変更および操作しやすくなります。さまざまなバッチファイル関数を使って、一時パス文字列を Autoexe.bat ファイルに追加できます。

構文

PathFind (szDir, svResult, bDir, bSearch);

パラメーター

テーブル 123 ・ PathFind のパラメーター

パラメーター	説明
szDir	パスバッファーで検索するパスを指定します。
svResult	関数によって返されたパスバッファーに見つかった完全ディレクトリとパスを返します。
bDir	szDir に完全修飾または完全でないディレクトリ名を含むかどうか指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ FULL Ñ szRefDir は完全修飾パスで、ディレクトリへのドライブ指定と完全なパスを含みます。 ・ PARTIAL Ñ szRefDir はディレクトリ名のみで、ドライブまたはパス情報を含みません。
bSearch	検索開始場所を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ CONTINUE Ñ 以前の検索を終了した場所からパス バッファーの検索を続行します。 ・ RESTART Ñ パス バッファーの最初から検索を開始します。

戻り値

テーブル 124 ・ PathFind の戻り値

戻り値	説明
0	ディレクトリがパスバッファーから正常に検索されたことを示します。
< 0	ディレクトリがパスバッファーから検索できなかったことを示します。

PathFind の例

基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* PathSet 関数、PathFind 関数、そして PathGet 関数のデモンストレーションを行います。
*
* まず、PathSet が呼び出され、パスバッファーへ検索パスへ配置
* します。そして PathFind が呼び出され、特定パスのインスタンスについて
* パスバッファーをを検索します。最後に、PathGet が呼び出され、
* パスバッファーの内容を戻します。
```

```
 *
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_PathFind(HWND);

function ExFn_PathFind(hMSI)
   STRING szString, szMsg, svResult, svString, szDir;
   NUMBER nResult;
   BOOL   bDir, bSearch;
begin

   // 検索パスをセットアップし、PathSet へのパラメーターとして渡します。
   szString = "C:¥¥DOS;C:¥¥USERS¥¥BIN;C:¥¥MSC¥¥BIN;";

   // パスバッファーへ検索パスを配置します。
   if (PathSet (szString) < 0) then
      // エラーを報告し、終了します。
      MessageBox ("PathSet が失敗しました。", SEVERE);
      abort;
   else
      szMsg = "PathSet がパスバッファーを次の通り設定します : %s";
      SprintfBox (INFORMATION, "PathSet の例 ", szMsg, szString);
   endif;

   // PathFind 変数を設定します。
   szDir = "BIN";

   // パスバッファーで "BIN" と名づけられたフォルダーを含むパスを検索します。
   nResult = PathFind(szDir, svResult, PARTIAL, RESTART);

   // PathFind でエラーを確認します。
   if (nResult < 0) then
      MessageBox("PathFind が失敗しました。", SEVERE);
      abort;
   endif;

   // szDir 文字列のすべてを検出するため文字列内をループします。

   while (nResult = 0)
      SprintfBox(INFORMATION, "PathFind の例 ",
            "%s を検索。¥n¥n 見つかったパス :  %s", szDir, svResult);
      nResult = PathFind(szDir, svResult, PARTIAL, CONTINUE);
   endwhile;

   // パスバッファーの内容を取得します。
   if (PathGet (svString) < 0) then
      MessageBox ("PathGet が失敗しました。", SEVERE);
   else
      // パス文字列を表示します。
      SprintfBox (INFORMATION, "PathGet の例 " ," パス : %s", svString);
   endif;

end;
```

PathGet

PathGet 関数は、PathSet を呼び出すことによって一時的に作成されたパスバッファーに現在保管されている検索パスを読み出します。パスバッファーを使うと、検索パスを構築および編集できます。編集しているパスが完成したら、PathGet を呼び出して検索パスを文字列変数に配置し、これをセットアップの他の関数に渡せるようにします。

この関数は Autoexec.bat ファイルまたはパス環境変数のパス ステートメントとは一切関係ありません。この関数はパスバッファー上でのみ動作します。これによって検索パスを構築、変更および操作しやすくなります。適切なバッチファイル関数を使って、一時パス文字列を Autoexe.bat ファイルに追加できます。

ヒント・*PathGet はパスバッファーから検索パスを取得して、パスバッファーに割り当てられたメモリを解放します。PathSet を呼び出してパスバッファーを再初期化しないかぎり、次に PathGet を呼び出しても失敗します。*

構文

PathGet (svString);

パラメーター

テーブル 125・PathGet のパラメーター

パラメーター	説明
svString	パスバッファーの内容を返します。

戻り値

テーブル 126・PathGet の戻り値

戻り値	説明
0	パスバッファーに現在保管されているパスが正常に取得されたことを示します。
< 0	一時パス文字列バッファーに現在保管されているパス文字列を取得できなかったことを示します。

PathGet の例

基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* PathSet 関数、PathFind 関数、そして PathGet 関数のデモンストレーションを行います。
*
```

```
＊まず、PathSet が呼び出され、パスバッファーへ検索パスへ配置
＊します。そして PathFind が呼び出され、特定パスのインスタンスについて
＊パスバッファーをを検索します。最後に、PathGet が呼び出され、
＊パスバッファーの内容を戻します。
＊
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_PathGet(HWND);

function ExFn_PathGet(hMSI)
   STRING szString, szMsg, svResult, svString, szDir;
   NUMBER nResult;
   BOOL   bDir, bSearch;
begin

   // 検索パスをセットアップし、PathSet へのパラメーターとして渡します。
   szString = "C:¥¥DOS;C:¥¥USERS¥¥BIN;C:¥¥MSC¥¥BIN;";

   // パスバッファーへ検索パスを配置します。
   if (PathSet (szString) < 0) then
      // エラーを報告し、終了します。
      MessageBox ("PathSet が失敗しました。", SEVERE);
      abort;
   else
      szMsg = "PathSet がパスバッファーを次の通り設定します : %s";
      SprintfBox (INFORMATION, "PathSet の例 ", szMsg, szString);
   endif;

   // PathFind 変数を設定します。
   szDir = "BIN";

   // パスバッファーで "BIN" と名づけられたフォルダーを含むパスを検索します。
   nResult = PathFind(szDir, svResult, PARTIAL, RESTART);

   // PathFind でエラーを確認します。
   if (nResult < 0) then
      MessageBox("PathFind が失敗しました。", SEVERE);
      abort;
   endif;

   // szDir 文字列のすべてを検出するため文字列内をループします。
   while (nResult = 0)
      SprintfBox(INFORMATION, "PathFind の例 ",
            "%s を検索。¥n¥n 見つかったパス :  %s", szDir, svResult);
      nResult = PathFind(szDir, svResult, PARTIAL, CONTINUE);
   endwhile;

   // パスバッファーの内容を取得します。
   if (PathGet (svString) < 0) then
      MessageBox ("PathGet が失敗しました。", SEVERE);
   else
      // パス文字列を表示します。
      SprintfBox (INFORMATION, "PathGet の例 " ," パス : %s", svString);
   endif;

end;
```

PathMove

PathMove 関数は、パスバッファーのディレクトリを別の場所に配置します。この関数を使用して別のディレクトリに相対してディレクトリを配置したり、パス文字列の最初または最後のアイテムとしてディレクトリを配置できます。

この関数は Autoexec.bat ファイルまたは PATH 環境変数の PATH ステートメントとは一切関係ありません。この関数はパスバッファー上でのみ動作します。これによって検索パスを構築、変更および操作しやすくなります。さまざまなバッチファイル関数を使って、一時パス文字列を Autoexec.bat に追加できます。

ヒント・パスバッファーの内容を取得するには PathGet を呼び出してください。パスバッファーの内容を設定するには PathSet を呼び出してください。

構文

PathMove (szDir, szRefDir, bDir, bRefDir, bPosition);

パラメーター

テーブル 127 ・ PathMove のパラメーター

パラメーター	説明
szDir	再配置するパスバッファーの完全パスか部分パスを指定します。
szRefDir	szDir のパスの移動先に相対した、パスバッファーのパスを指定します。szDir のパスをパスバッファーのパスの最初または最後に移動するには、このパラメーターにヌル文字列 (“”) を渡します。
bDir	szDir に完全修飾または完全でないディレクトリ名を含むか指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ FULL Ñ szDir に完全修飾ディレクトリ名が含まれるように指定します。 ・ PARTIAL Ñ szDir にディレクトリ名のみが含まれるように指定します。
bRefDir	szRefDir に完全修飾または完全でないディレクトリ名を含むか指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ FULL Ñ szRefDir に完全修飾ディレクトリ名が含まれるように指定します。 ・ PARTIAL Ñ szRefDir にディレクトリ名のみが含まれるように指定します。
bPosition	szRefDir に相対する szDir の移動場所を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ AFTER Ñ szDir を szRefDir の後に配置するよう指定します。szRefDir でヌル文字列 (“”) が指定されている場合、szDir はパス バッファーのパスの最後に配置されます。 ・ BEFORE Ñ szDir を szRefDir の前に配置するよう指定します。szRefDir でヌル文字列 (“”) が指定されている場合、szDir はパス バッファーのパスの前に配置されます。

戻り値

テーブル 128 ・ PathMove の戻り値

戻り値	説明
0	パスバッファーのディレクトリが正常に再配置されたことを示します。
< 0	パスバッファーのディレクトリを再配置できなかったことを示します。

PathMove の例

基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * PathMove 関数のデモンストレーションを行います。
 *
 * このスクリプト例は、パスバッファーのパスを再配置する方法を説明
 * します。初めに、検索パスを使ったパスバッファーの初期化を
 * 行います。そして、ひとつのパスを別のパスの先に移動します。
 * 次に、検索バッファーから修正済み検索パスを取得し、
 * それを表示します。
 *
 * そして修正された検索パスを使ってパスバッファーを初期化し、
 * ひとつのパスを別のパスの後に移動します。最後に、検索バッファーから
 * 修正済み検索パスを取得してそれを
 * 表示します。
 *
 * メモ : 最初の PathMove ステートメントの後に PathGet が呼び出されます。
 *    これは、デモンストレーションの目的でパスバッファーの値が
 *    表示されるようにするためです。実際のスクリプトでは、
 *    PathGet を呼び出す前にバッファーの検索パスのビルドで
 *    終了します。
 *
¥*---------------------------------------------------------------------*/

#define TITLE " パスバッファーの例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_PathMove(HWND);

function ExFn_PathMove(hMSI)
   STRING svSearchPath;
begin

   // 検索パスをセットアップし、PathSet へのパラメーターとして渡します。
   svSearchPath = "C:¥¥MsOffice;C:¥¥DOS;C:¥¥Windows;C:¥¥App2;C:¥¥Temp";

   // パスバッファーを初期化します。
   PathSet (svSearchPath);

   // 初期検索パスを表示します。
   SprintfBox (INFORMATION, TITLE,
         " 開始検索パスは %s.",svSearchPath);

   // C:¥App2 を C:¥DOS の前に移動します。
   if (PathMove("C:¥¥App2", "C:¥¥DOS", FULL, FULL, BEFORE) < 0) then
      MessageBox ("C:¥¥App2 を移動できませんでした。", SEVERE);
      abort;
   endif;

   // パスバッファーから検索パスを取得し、パスバッファー用に
   // 割り当てられたメモリを開放します。
   PathGet (svSearchPath);

   // 検索パスを表示します。
   // svSearchPath は、C:¥MSOFFICE、C:¥APP2、C:¥DOS、C:¥WINDOWS、そして C:¥TEMP を含みます。
   SprintfBox (INFORMATION, TITLE,
```

```
            "C:¥¥APP2 が C:¥¥DOS の前に移動しました。¥n¥n 検索パスは %s です。",
            svSearchPath);

    // パスバッファーを検索パスと共に初期化します。
    PathSet (svSearchPath);

    // C:¥DOS を C:¥Temp の後に移動します。
    if (PathMove ("C:¥¥DOS", "TEMP", FULL, PARTIAL, AFTER) < 0) then
        MessageBox ("C:¥¥DOS を移動できませんでした。", SEVERE);
    endif;

    // パスバッファーから検索パスを取得し、パスバッファー用に
    // 割り当てられたメモリを開放します。
    PathGet (svSearchPath);

    // 検索パスを表示します。
    // svSearchPath は、C:¥MSOFFICE、C:¥APP2、C:¥WINDOWS、C:¥TEMP、そして C:¥DOS を含みます。
    SprintfBox (INFORMATION, TITLE,
            "C:¥¥DOS が C:¥¥Temp の後に移動しました。n¥n 検索パスは %s です。",
            svSearchPath);

end;
```

PathSet

PathSet 関数は、パスバッファーに検索パス文字列を格納します。他のパス関数を使用してこのバッファーを操作できます。szString の値は絶対パス (ドライブ指定を含むパス。例えば、"C:¥¥Program Files¥¥AppName") でなくてはなりません。

この関数は Autoexec.bat ファイルまたは PATH 環境変数の PATH ステートメントとは一切関係ありません。この関数はパスバッファー上でのみ動作します。これによって検索パスを構築、変更および操作しやすくなります。これにより、このテンポラリパス文字列を Autoexec.bat ファイルまたはパス環境変数に追加することができます。

構文

```
PathSet ( szString );
```

パラメーター

テーブル 129 ・ PathSet のパラメーター

パラメーター	説明
szString	パスバッファーに格納する検索パスを指定します。検索パスはは完全修飾パスです。つまり、ドライブのインストール先とディレクトリへの完全パスを含みます。

戻り値

テーブル 130 ・ PathSet の戻り値

戻り値	説明
0	関数が検索パス文字列をパスバッファーに格納したことを示します。
< 0	関数が検索パス文字列をパスバッファーに格納できなかったことを示します。

PathSet の例

基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* PathSet 関数、PathFind 関数、そして PathGet 関数のデモンストレーションを行います。
*
* まず、PathSet が呼び出され、パスバッファーへ検索パスへ配置
* します。そして PathFind が呼び出され、特定パスのインスタンスについて
* パスバッファーをを検索します。最後に、PathGet が呼び出され、
* パスバッファーの内容を戻します。
*
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_PathSet(HWND);

function ExFn_PathSet(hMSI)
   STRING szString, szMsg, svResult, svString, szDir;
   NUMBER nResult;
   BOOL   bDir, bSearch;
begin

   // 検索パスをセットアップし、PathSet へのパラメーターとして渡します。
   szString = "C:¥¥DOS;C:¥¥USERS¥¥BIN;C:¥¥MSC¥¥BIN;";

   // パスバッファーへ検索パスを配置します。
```

```
    if (PathSet (szString) < 0) then
        // エラーを報告し、終了します。
        MessageBox ("PathSet が失敗しました。", SEVERE);
        abort;
    else
        szMsg = "PathSet がパスバッファーを次の通り設定します : %s";
        SprintfBox (INFORMATION, "PathSet の例 ", szMsg, szString);
    endif;

    // PathFind 変数を設定します。
    szDir = "BIN";

    // パスバッファーで "BIN" と名づけられたフォルダーを含むパスを検索します。
    nResult = PathFind(szDir, svResult, PARTIAL, RESTART);

    // PathFind でエラーを確認します。
    if (nResult < 0) then
        MessageBox("PathFind が失敗しました。", SEVERE);
        abort;
    endif;

    // szDir 文字列のすべてを検出するため文字列内をループします。
    while (nResult = 0)
        SprintfBox(INFORMATION, "PathFind の例 ",
                "%s を検索。¥n¥n 見つかったパス :  %s", szDir, svResult);
        nResult = PathFind(szDir, svResult, PARTIAL, CONTINUE);
    endwhile;

    // パスバッファーの内容を取得します。
    if (PathGet (svString) < 0) then
        MessageBox ("PathGet が失敗しました。", SEVERE);
    else
        // パス文字列を表示します。
        SprintfBox (INFORMATION, "PathGet の例 " ," パス : %s", svString);
    endif;

end;
```

PlaceBitmap

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

PlaceBitmap 関数は、インストールウィンドウに画像を挿入します。画像ソースは szName で指定します。ソースはビットマップファイル (.bmp)、メタファイル (.wmf ファイル) またはダイナミックリンクライブラリ (.dll) を使用できます。

構文

```
PlaceBitmap (szName, nID_BITMAP, nDx, nDy, nDrawOp);
```

パラメーター

テーブル 131 ・ PlaceBitmap のパラメーター

パラメーター	説明
szName	表示されるビットマップファイル (.bmp)、メタファイル (.wm ファイル) またはダイナミックリンクライブラリ (.dll) の完全修飾名を指定します。InstallShield は、ビットマップファイルとメタファイルをファイル拡張子で認識します。ビットマップファイルは .bmp の拡張子を持つ必要があります。メタファイルは .wmf の拡張子です。ダイナミックリンクライブラリは .dll の拡張子を持っています。ファイル名に拡張子がついていないと、InstallShield は拡張子を .dll とみなします。 代替透明色を指定するには、ファイル名の後にセミコロンを付けて、その後に RGB カラー値を入力します。(RGB カラーは、3 つの数値で表し、それぞれをカンマで区切ります。) その色は、szName で指定されたビットマップの透明色として使用されます。これが既に表示されているビットマップには影響を *与えない* ことに注意してください。また、これが、PlaceBitmap を呼び出すと表示されるビットマップのデフォルトの透明色になることもありません。 下のパラメーターは、白を透明色に指定します。 SUPPORTDIR ^ "Bitmap.bmp;255,255,255" nDrawOptions を REMOVE に設定すると、このパラメーターは無視されます。
nID_BITMAP	ビットマップが .dll 内にある場合、そのビットマップのリソース ID を指定します。ビットマップソースがメタファイルまたはビットマップファイルの場合、現在表示されていない画像の値を指定します。同時に表示されている画像は、固有の ID 番号を持つ必要があります。nDrawOptions が REMOVE に設定されている場合、このパラメーターは表示画像の ID を持つ必要があります。
nDx	数値または CENTERED 定数をこのパラメーターに渡します。 ・ nDrawOp が LOWER_LEFT、LOWER_RIGHT、UPPER_LEFT または UPPER_RIGHT に設定されている場合、インストールウィンドウの端と画像の端の左右の距離をピクセルで指定する数値を渡します。 ・ nDrawOp が LOWER_LEFT、LOWER_RIGHT、UPPER_LEFT または UPPER_RIGHT に設定されている場合、CENTERED 定数を渡すと、画像が横軸の中央に配置されます。画像は、nDy で指定されたピクセル分、インストールウィンドウの上または下から移動します。CENTERED 定数は、nDx に渡されると、画像を横に中央配置するため、nDrawOp 引数は画像の縦の配置分だけ影響を与えます。

テーブル 131 ・ PlaceBitmap のパラメーター (続き)

パラメーター	説明
nDy	数値または CENTERED 定数をこのパラメーターに渡します。 ・ nDrawOp が LOWER_LEFT、LOWER_RIGHT、UPPER_LEFT または UPPER_RIGHT に設定されている場合、インストールウィンドウの端と画像の端の縦の距離をピクセルで指定する数値を渡します。 ・ nDrawOp が LOWER_LEFT、LOWER_RIGHT、UPPER_LEFT または UPPER_RIGHT に設定されている場合、CENTERED 定数を渡すと、画像が縦軸の中央に配置されます。画像は、nDx で指定されたピクセル分、インストールウィンドウの左または右から移動します。CENTERED 定数は、nDy に渡されると、画像を縦に中央配置するため、nDrawOp 引数は画像の横の配置分だけ影響を与えます。  ***メモ ・** CENTERED 定数を nDx と nDy の両方のパラメーターに渡して、画像をインストールウィンドウの中央に配置することができます。これは、CENTERED 定数を nDrawOp パラメーターに渡すのと同じです。*

テーブル 131 • PlaceBitmap のパラメーター (続き)

パラメーター	説明
nDrawOp	ビットマップの配置場所を指定し、配置オプションを設定するか、または以前に配置されたビットマップを削除します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • BITMAPICON Ñ ビットマップに透明部分があることを示します。ビット単位 OR 演算子 (|) を使ってこの定数を TILED、FULLSCREEN または FULLSCREENSIZE を除くその他の定数と組み合わせることができます。BITMAPICON が Ored で他の定数と一緒に使われている場合、ビット単位演算子は無視されて BITMAPICON が使用されます。 BITMAPICON は、szName がメタファイルまたは 24-bit のビットマップを指定した場合は無効です。BITMAPICON を指定すると、SetDisplayEffect を使って特殊効果が有効になっていても、ビットマップは通常に表示されることに注意してください。 • TILED Ñ ビットマップがメイン インストール ウィンドウに並べて表示されます。この定数は通常、インストールウィンドウの背景の作成に使用されます。この定数が指定されると、場所指定のオプションは無視されて、SetDisplayEffect を使って特殊効果が有効になっていても、ビットマップは通常に表示されます。 • FULLSCREEN Ñ 画像をインストール ウィンドウ全体に表示します。画像は表示時にサイズ変更されません。ビットマップ画像が InstallShield メインウィンドウより小さい場合、その画像はウィンドウの中央に配置され、背景は現在の背景色で表示されます。デフォルト値は青緑です。これは SetColor 関数を使って変更することができます。この定数が指定されると、場所指定のオプションは無視されて、SetDisplayEffect を使って特殊効果が有効になっていても、ビットマップは通常に表示されます。 • FULLSCREENSIZE Ñ 画像をインストール ウィンドウ全体に広がるように表示します。この定数が指定されると、場所指定のオプションは無視されて、SetDisplayEffect を使って特殊効果が有効になっていても、ビットマップは通常に表示されます。
nDrawOp (続き)	• CENTERED Ñ ビットマップをインストールウィンドウの中央に配置します。 • LOWER_LEFT Ñ ビットマップを InstallShield インストール ウィンドウの左下に配置します。 • LOWER_RIGHT Ñ ビットマップを InstallShield インストール ウィンドウの右下に配置します。 • UPPER_LEFT Ñ ビットマップを InstallShield インストール ウィンドウの左上に配置します。 • UPPER_RIGHT Ñ ビットマップを InstallShield インストール ウィンドウの右上に配置します。 • REMOVE Ñ 以前に配置されたビットマップまたはメタファイルを削除します。SetDisplayEffect で有効になった特殊効果はすべて無視されます。

戻り値

テーブル 132 ・ PlaceBitmap の戻り値

戻り値	説明
0	関数が正常に画像を見つけて配置したことを示します。
< 0	関数が画像を見つけたり、配置できなかったことを示します。

Comments

InstallShield は、2 色、16 色、または 256 色およびトゥルーカラー (24 ビット) のビットマップをサポートしています。2 色、16 色、または 256 色のビットマップは透明部分が使用できます。

透明ビットマップは、背景ウィンドウに統合させて表示する画像の表示に便利です。指定の透明色と一致するビットマップ内のピクセルは表示されません。そのため、その場所の背景のピクセルが表示されたままになります。セットアップでは、会社名とロゴを上手に組み込んだ透明ビットマップは、多くの場合、インストールウィンドウでタイトルとして使用されます。

透明ビットマップを指定するには、定数 BITMAPICON を nDrawOp パラメーターに渡す必要があります。また、ビットマップで透明に指定する色を考慮する必要があります。デフォルトの透明色は紫 (RGB(255,0,255)) です。別の透明色を指定するには、szName パラメーターを下のように使用します。

メタファイルは配置されず作成されるため、本来は透明です。BITMAPICON がメタファイル用に指定されると、そのパラメーターは無視されます。

メモ・*非透明のビットマップの多くの特殊表示効果は、SetDisplayEffect 関数を使って使用できます。この関数は、メタファイルに対しても制限付きの表示効果を提供します。*

ウィンドウ内のビットマップの位置は、次の 2 つの方法のどちらかを使って指定します。

- 場所を表す定数のひとつを nDrawOp パラメーターに渡す。
- インストールウィンドウの端から、垂直および水平のオフセットを nDx and nDy に渡す。
- nDx か nDy に、垂直または水平オフセットを組み合わせて CENTERED 定数を渡す。

必要なくなったビットマップまたはメタファイルは、PlaceBitmap を REMOVE 定数と一緒に、nDrawOp パラメーターとして呼び出して削除してください。不要なビットマップは、別のビットマップがそのビットマップをカバーしていたとしても削除することをお勧めします。これは、最初のビットマップのパレットエントリが、ビットマップが削除されるまでリリースされないからです。

ヒント・*16 色または 256 色モードで動作するシステムに表示されているトゥルーカラービットマップは、カラーパレットで使用できる色だけを使用します。追加のカラーパレットエントリが使用できても、その他の色がビットマップに割り当てられることはありません。24-bit のビットマップを持つセットアップが 16 色 または 256 色のシステムで動作すると予想される場合、16 色または 256 色バージョンのビットマップを含めてください。それから GetSystemInfo を COLORS パラメーターと共に呼び出して、ビットマップ表示色を選択する前に現在の色モードを決定してください。*

SetDisplayEffect を呼び出すと、並べて表示しない、全画面の透明ビットマップの特殊効果を設定することができます。メタファイルについて限定的に特殊効果を設定することもできます。

InstallShield では 24 ビットの透明ビットマップをサポートしていません。透明色を 24-bit のビットマップに含めて BITMAPICON 定数を指定すると、色は普通に表示されます。

256 色モードで動作するシステムに 256 色のビットマップを配置すると、InstallShield はビットマップのカラーパレットをシステムカラーパレットに割り当てようとします。複数の 256 色のビットマップが配置されると、InstallShield はすべての可視ビットマップのカラーパレットをシステムカラーパレットと合併して、一番最後に配置されたビットマップを優先させようとします。この動作により、追加のビットマップが表示されると、以前に配置されたビットマップの色が変わることがあります。

256 色モードで、256 色でディザリングされた背景のシステムでは、多くの色を持つビットマップの場合、背景に使用されているカラーパレットエントリが再割り当てされる場合があります。これにより、背景にグラデーション効果が表れることがあります。多くの色を使うビットマップを持つセットアップを、256 色のシステムで動作する場合、256 色で階調された背景を使用しないことをお勧めします。

システムカラーパレットは、256 色モードで動作されるシステムにのみ存在します。High Color (16 ビット) または True Color (24 ビット) モードで動作するシステムや、65535 (16 ビット) カラー モードで動作するシステムにはシステム カラー パレットがありません。これらのシステムでは、カラーパレット処理の問題がないため、色は RGB カラー値を使って直接表示されます。詳細は、「色化けの防止」を参照してください。

メタファイルはレンダリングされるため、カスタムカラーパレットがありません。メタファイルが 256 色のシステムで表示されると、カラーパレット処理の問題は起こりません。メタファイルは、現在カラーパレットで使用できる色で表示されます。そのため、256 色のシステムで動作するセットアップでは、標準の 16 色以外で色を表示するメタファイルを使用しないでください。

PlaceBitmap の例

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * PlaceBitmap 関数のデモンストレーションを行います。
 *
 * PlaceBitmap はスクリーン上のビットマップを表示及び削除するために
 * 呼び出されます。SetDisplayEffect 関数はビットマップの表示効果を
 * 設定します。
 *
 * メモ : このスクリプトを実行する前に、定数 BMP_PATH が
 *      ターゲットシステム上の既存のビットマップファイルを参照するように
 *      設定します。
 *
¥*---------------------------------------------------------------------*/

#define BMP_PATH  "C:¥¥Windows¥¥Bubbles.bmp"
#define BITMAP_ID_1 12
#define BITMAP_ID_2 13
#define BITMAP_ID_3 14

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_Placebitmap(HWND);

function ExFn_Placebitmap(hMSI)
begin
```

```
    Enable ( BACKGROUND );

    // 左上角にビットマップを表示します。
    PlaceBitmap (BMP_PATH, BITMAP_ID_1, 10, 12, UPPER_LEFT);

    // ビットマップリビール効果を設定します。
    SetDisplayEffect (EFF_REVEAL);

    // 右下角にビットマップを表示します。
    PlaceBitmap (BMP_PATH, BITMAP_ID_2, 10, 10, LOWER_RIGHT);

    Delay(3);

    // 左上角にあるビットマップを削除します。
    PlaceBitmap ("", BITMAP_ID_2, 0, 0, REMOVE);

    // 右下角にあるビットマップを表削除します。
    PlaceBitmap ("", BITMAP_ID_1, 0, 0, REMOVE);

    // ビットマップの効果フェードを設定します。
    SetDisplayEffect (EFF_FADE);

    // ビットマップを画面中央に表示します。
    PlaceBitmap (BMP_PATH, BITMAP_ID_3, CENTERED, CENTERED, 0);
    Delay (3);

    // 画面中央にあるビットマップを削除します。
    PlaceBitmap ("", BITMAP_ID_3, 0, 0, REMOVE);
    Delay (1);

end;
```

PlaceWindow

PlaceWindow 関数は、ユーザー インターフェイス オブジェクトの位置を変更する場合に使用します。これには、**PlayMMedia** を通して実行時に表示されるビルボード、Adobe Flash アプリケーション ファイル、および AVI ファイルが含まれます。nDx と nDy で、オブジェクトの辺と画面の端部との距離を指定します。

この関数を使用すると、インストールがさまざまな画面解像度で実行することに注意してください。オブジェクトの位置を決める前に、画面の大きさを決定することもできます。距離はピクセル単位で測定され、オブジェクトの端部と指定した画面角の端との長さになります。

***メモ・PlaceWindow** 関数は、進行状況ダイアログで表示される種類のビルボードには反映されません。異なる種類のビルボードについての詳細は、「InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルとファイルの種類」を参照してください。*

制限

この関数はメッセージ ボックスやカスタム ダイアログを配置するのに使用することはできません。

- メッセージボックスはネイティブの Windows API で作成されるため、この関数で位置を変更することはできません。メッセージボックスの位置は Windows API で決定されており、インストールからは制御できません。

- カスタム ダイアログは、この関数を使って配置することはできません。**PlaceWindow** を **AskOptions**、**AskPath**、**AskText**、または **EnterDisk** 関数と共に使用することはできません。背景ウィンドウモードを有効にしない限り、デフォルトではデスクトップ中央にダイアログが表示されます。インストールがウィンドウモードの場合、背景ウィンドウの中央にダイアログが表示されます。

構文

PlaceWindow (nObject, nDx, nDy, nCorner);

パラメーター

テーブル 133 ・ PlaceWindow のパラメーター

パラメーター	説明
nObject	位置を変更するオブジェクトを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ **ASKOPTIONS** Ñ AskOptions ダイアログを移動します。 ・ **ASKPATH** Ñ AskPath ダイアログを移動します。 ・ **ASKTEXT** Ñ AskText ダイアログを移動します。 ・ **BACKGROUND** Ñ 背景ウィンドウを移動します。 ・ **BILLBOARD** Ñ ファイル転送処理で使用されるビルボードの位置を設定します。 ・ **ENTERDISK** Ñ EnterDisk ダイアログを移動します。 ・ **MMEDIA_AVI** Ñ 次に再生される .avi ファイルのウィンドウ位置を設定します。デフォルトで、.avi ファイルは実行時に画面の左隅 (左から 10 ピクセル、上から 10 ピクセルの位置) のウィンドウ内で再生されます。 ・ **MMEDIA_SWF** Ñ 次に再生される Adobe Flash アプリケーション ファイル (.swf) のウィンドウ位置を設定します。 ・ **STATUS** Ñ 進行状況インジケーターを移動します。 ・ **STATUSDLG** Ñ ダイアログスタイルの進行状況インジケーターを移動します。 ・ **STATUSEX** – [セットアップ ステータス] ダイアログを移動します。 ・ **STATUSOLD** Ñ 旧式の進行状況インジケーターを移動します。  ***ヒント ・** PlaceWindow を呼び出して進行状況インジケーターまたはステータス ダイアログを移動する場合、セットアップで有効にしたフィードバック オブジェクトに正確な定数を渡してください。たとえば、Enable (STATUSOLD) を呼び出した場合、PlaceWindow には STATUSOLD を渡さなくてはなりません。*
nDx	オブジェクトの該当する端部と水平軸上の画面端部との距離を、ピクセル単位で指定します。
nDy	オブジェクトの該当する端部と垂直軸上の画面端部との距離を、ピクセル単位で指定します。

テーブル 133 ・ PlaceWindow のパラメーター (続き)

パラメーター	説明
nCorner	nDx と nDy で表される距離を測定するときに基準とする角を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ LOWER_LEFT Ñ nDx を InstallShield メインウィンドウの左から、nDy を下から測ります。 ・ LOWER_RIGHT Ñ nDx を InstallShield メインウィンドウの右から、nDy を下から測ります。 ・ UPPER_LEFT Ñ nDx を InstallShield メインウィンドウの左から、nDy を上から測ります。 ・ UPPER_RIGHT—nDx を InstallShield メインウィンドウの右から、nDy を上から測ります。 ・ CENTERED Ñ ユーザー インターフェイス オブジェクトをウィンドウ中央に配置します。  ***メモ・** CENTERED は nDx または nDy に配置して、オブジェクトを水平のみまたは垂直のみ中央に配置することも可能です。*

戻り値

テーブル 134 ・ PlaceWindow の戻り値

戻り値	説明
0	オブジェクトの位置が正常に変更されたことを示します。
< 0	オブジェクトの位置が正常に変更されなかったことを示します。

追加情報

インストールが Adobe Flash アプリケーション ファイル (.swf) または AVI ファイルを再生する場合は、PlayMMedia 関数を使用します。

PlaceWindow の例

***メモ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
```

```
＊PlaceWindow 関数のデモンストレーションを行います。
＊
＊PlaceWindow が呼び出され、背景ウィンドウを
＊50 ピクセル右、画面の左上角の下に
＊配置します。
＊
¥*--------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_PlaceWindow(HWND);

function ExFn_PlaceWindow(hMSI)
begin

   Enable ( BACKGROUND );

   Enable ( DEFWINDOWMODE );

   MessageBox ("これは背景ウィンドウのデフォルト位置です。",
           INFORMATION);

   // PlaceWindow の呼び出しを 3 秒間遅延させます。
   Delay(3);

   // 背景ウィンドウの位置を変更します。
   if (PlaceWindow (BACKGROUND, 50, 50, UPPER_LEFT) < 0) then
      MessageBox ("PlaceWindow が失敗しました。", SEVERE);
   else
      MessageBox ("これが背景ウィンドウの新しい位置です。",
              INFORMATION);
      // 既存のスクリプトを 3 秒間遅延させます。
      Delay(3);
   endif;

end;
```

PlayMMedia

PlayMMedia 関数は Adobe Flash アプリケーション ファイル (.swf)、AVI ファイル、またはサウンド ファイル (MIDI または WAVE) を再生します。

***ヒント・PlayMMedia** を使用して Flash ファイルまたは AVI ファイルを表示する場合、インストールで背景ウィンドウを表示する必要があります。詳細については、「InstallScript と InstallScript MSI インストールで背景ウィンドウを表示する」を参照してください。*

InstallShield を使って、背景ウィンドウを表示しないで、インストールでビルボードとして Flash ファイルを表示することができます。詳細については、「InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルとファイルの種類」を参照してください。

構文

```
PlayMMedia (nType, szFileName, nOperation, nReserved);
```

パラメーター

テーブル 135 · PlayMMedia のパラメーター

パラメーター	説明
nType	インストールが再生するファイルの種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · MMEDIA_AVI Ñ ファイルは AVI ファイルです。 · MMEDIA_MIDI Ñ ファイルは MIDI サウンド フォーマットです。 · MMEDIA_SWF Ñ ファイルは Adobe Flash アプリケーション ファイル (.swf) です。 · MMEDIA_WAVE Ñ ファイルは WAVE サウンド フォーマットです。
szFileName	検索する再生するファイルの完全修飾名を指定します。
nOperation	再生モードを指定します。以下から、このパラメーターで渡す定義済み定数すべて選択します : · MMEDIA_PLAYSYNCH Ñ 同期で再生します。 · MMEDIA_PLAYASYNCH Ñ 非同期で再生します。この定数は OR 演算子 (|) を使って MMEDIA_PLAYCONTINUOUS と組み合わせることができます。 · MMEDIA_PLAYCONTINUOUS Ñ 継続ループで再生します。この値はサウンドファイル /AVI ファイルを同期モードで再生中は利用することができません。これはファイルを非同期モードで再生中のみ利用することができます。OR 演算子 (|) を使って MMEDIA_PLAYASYNCH と組み合わせます。 · MMEDIA_STOP Ñ 再生を停止します。 ***メモ ·** Flash ファイルは、MMEDIA_PLAY* 定数が nOperation で渡されているかどうかに関わらず、非同期で 1 度だけ再生します。*
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。

戻り値

テーブル 136 · PlayMMedia の戻り値

戻り値	説明
0	関数が正しくファイルを再生しました。
< 0	関数がファイルを再生することができませんでした。 これが発生する一例として、**PlayMMedia** に Flash ファイルが指定されているが、ターゲット システム上に Flash Player が存在しない場合があります。

追加情報

Flash ファイルまたは AVI ファイルを使用する場合、**SizeWindow** と **PlaceWindow** を使って、Flash ファイルまたは AVI ファイルを表示する背景ウィンドウのサイズと配置を制御できます。

PlayMMedia の例

基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* PlayMMedia 関数のデモンストレーションを行います。
*
* このスクリプトはセットアップ中に AVI ファイルを再生します。
*
* メモ : このスクリプト例を実行するには、
*      / またはファイルが入ったコンポーネントを持つサブ機能を含む
*      プロジェクトを作成 ( またはプロジェクトに挿入 ) します。Then
*    IDE の [ サポート ファイル ] ビューにあるディスク 1 へ
*    AVI ファイルを追加します。AVI ファイルを指定するには、下の #define SOURCE で
*    ファイル名前を変更します。
*
* 警告 : この例ではアンインストール機能を含まないため、
*      重要なファイルを上書きしない、共有ファイルをインストールしない、
*      またはレジストリをアップデートしないプロジェクトでのみ
*      利用してください。
*
¥*---------------------------------------------------------------*/

#define SOURCE  SRCDIR + "windy7(1).avi"
#define TITLE1  "AVI を同期で再生しています ..."
#define TITLE1  "AVI を同期で継続的に再生しています ..."

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_PlayMMedia(HWND);

function ExFn_PlayMMedia(hMSI)
   NUMBER nvDisk;
begin

   Enable ( BACKGROUND );

   // 最初に AVI を同期で再生して、別のイベントなしで
   // どのように再生へ導くのかをデモンストレーションします。
   SetTitle (TITLE1, 16, YELLOW);
   PlaceWindow (MMEDIA_AVI, 10, 10, UPPER_RIGHT);

   if (PlayMMedia (MMEDIA_AVI, SOURCE, MMEDIA_PLAYSYNCH, 0) < 0) then
      MessageBox ("AVI ファイルを再生することができません。", WARNING);
   endif;

   // ここで AVI を非同期で再生します。AVI は
   // ファイル転送が始まっても続行します。
```

```
    SetTitle (TITLE2, 16, YELLOW);

    PlaceWindow (MMEDIA_AVI, 10, 10, LOWER_RIGHT);

    if (PlayMMedia (MMEDIA_AVI, SOURCE,
                MMEDIA_PLAYASYNCH | MMEDIA_PLAYCONTINUOUS, 0) < 0) then
       MessageBox ("AVI ファイルを再生することができません。", WARNING);
    endif;

    Enable (STATUSDLG);
    Enable (INDVFILESTATUS);

    StatusUpdate (ON, 99);

    // ファイルを転送します。
    ComponentMoveData (MEDIA, nvDisk, 0);

    Disable (INDVFILESTATUS);
    Disable (STATUSDLG);

    // AVI はセットアップが終了したときに停止します。しかし、
    // 次の用に明示的に停止することもできます :
    PlayMMedia (MMEDIA_AVI, SOURCE, MMEDIA_STOP, 0);

end;
```

PostShowComponentDlg

注意 - *この関数は不要なので、InstallShield ではサポートされていません。InstallShield Professional 2.03 で PreShowComponentDlg を必要とした関数は、基本の MSI プロジェクト用にはサポートされていません。セットアッププロジェクトでこれらの関数を利用する場合、基本の MSI プロジェクトの変換 を行い、InstallScript MSI プロジェクトタイプに変換して関数をスクリプトに追加しなくてはなりません。*

PostShowComponentDlg 関数は、InstallShield Professional コンポーネントを変換して InstallShield-Windows Installer Edition の機能に戻します。スクリプトでコンポーネントダイアログまたは関数を呼び出した後、PostShowComponentDlg を呼び出さなくてはなりません。この関数は、コンポーネントの選択に基づいて Windows Installer 機能を選択します。

構文

```
PostShowComponentDlg (hMSI);
```

パラメーター

テーブル 137 - PostShowComponentDlg のパラメーター

パラメーター	説明
hMSI	エントリポイント関数 へ渡される Windows Installer (MSI) データベースへのハンドル。

戻り値

テーブル 138 - PostShowComponentDlg の戻り値

戻り値	説明
0	関数が InstallShield Professional コンポーネント を変換して Windows Installer 機能に戻したことを示します。
< 0	関数が InstallShield Professional コンポーネントを変換して Windows Installer 機能へ戻すことができなかったことを示します。

PreShowComponentDlg

注意 - *この関数は不要なので、InstallShield ではサポートされていません。InstallShield Professional 2.03 で PreShowComponentDlg を必要とした関数は、基本の MSI プロジェクト用にはサポートされていません。セットアッププロジェクトでこれらの関数を利用する場合、基本の MSI プロジェクトの変換 を行い、InstallScript MSI プロジェクトタイプに変換して関数をスクリプトに追加しなくてはなりません。*

PreShowComponentDlg 関数は、InstallShield-Windows Installer Edition の機能を InstallShield Professional コンポーネントに変換します。機能コスティングの機能も初期化します。

構文

PreShowComponentDlg (hMSI);

パラメーター

テーブル 139 ・ PreShowComponentDlg のパラメーター

パラメーター	説明
hMSI	エントリポイント関数 へ渡される Windows Installer (MSI) データベースへのハンドル。

戻り値

テーブル 140 ・ PreShowComponentDlg のパラメーター

戻り値	説明
0	関数が Windows Installer 機能を InstallShield Professional コンポーネントに変換したことを示します。
< 0	関数が Windows Installer 機能を InstallShield Professional コンポーネントに変換できなかったことを示します。

ProgDefGroupType

ProgDefGroupType 関数は、システム変数 ALLUSERS の値を設定します。詳細については、「ALLUSERS」を参照してください。

構文

```
ProgDefGroupType ( nType );
```

パラメーター

テーブル 141 ・ ProgDefGroupType のパラメーター

パラメーター	説明
nType	InstallScript 変数 ALLUSERS に使用する値を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ PERSONAL Ñ ALLUSERS を FALSE に設定します。 ・ COMMON Ñ ALLUSERS を TRUE に設定します。

戻り値

テーブル 142 ・ ProgDefGroupType の戻り値

戻り値	説明
0	この関数は常にゼロを戻します。

ビルトイン関数 (Q-R)

カテゴリ別の関数一覧は、「カテゴリー別ビルトイン関数」を参照してください。

QueryProgItem

GetShortcutInfo 関数は **QueryProgItem** 関数に優先します。

QueryProgItem 関数は特定のプログラムアイテム、またはサブフォルダー名の存在を確認します。InstallScript がアイテムまたはサブフォルダーを検出すると、**QueryProgItem** がその属性を戻します。属性は製品のコマンドライン、作業ディレクトリ、アイコンパス、ショートカット キー、および最小化フラグを含みます。

QueryProgItem を利用するには、パラメーター szFolderName と szItemName で情報を入力してください。InstallScript エンジン ファイルは残りのパラメーターにプログラム アイテムの属性を入力します。

構文

```
QueryProgItem ( szFolderName, szItemName, svCmdLine, svWrkDir, svIconPath, nvIconIndex, svShortCutKey, nvMinimizeFlag );
```

パラメーター

テーブル 1 ・ QueryProgItem のパラメーター

パラメーター	説明
szFolderName	アイテムまたはサブフォルダーを含むフォルダーの名前を指定します。szFolderName の完全修飾パスを次の様に指定することができます : "C:¥¥ProgramData¥¥Microsoft¥¥Windows¥¥Start Menu¥¥Programs¥¥Games" szFolderName がヌルの場合、**QueryProgItem** はデフォルトの Programs ディレクトリを検索します。szFolderName に絶対パス (ドライブ名を含むパス。例、"C:¥¥Program Files¥¥AppName") を指定しなかった場合、**QueryProgItem** はデフォルトの Program ディレクトリの下にあるサブフォルダを検索します。この場所は InstallScript 変数 ALLUSERS の値、およびターゲット システム上の Windows バージョンによって異なります。 InstallScript システム変数を使用することもできます。 ・ FOLDER_DESKTOP ― [デスクトップ] フォルダーでアイテムをクエリします。 ・ FOLDER_STARTUP ― Startup メニューでアイテムをクエリします。 ・ FOLDER_STARTMENU ― Start メニューでアイテムをクエリします。 ・ FOLDER_PROGRAMS ― Start¥Programs メニューでアイテムをクエリします。 または、次のような関連パスを使用できます。 FOLDER_PROGRAMS ^ "ACCESSORIES¥¥GAMES"
szItemName	検索するプログラムアイテムまたはサブフォルダーの名前を指定します。
svCmdLine	アイテムの実行可能ファイルのコマンドラインまたはサブフォルダーへの完全パスを戻します。
svWrkDir	プログラムアイテムの作業ディレクトリの完全パスを戻します。(szItemName がサブフォルダーの場合を除きます)。
svIconPath	.ico ファイルまたは .exe ファイルの完全修飾ファイル名を戻します。(szItemName がサブフォルダーの場合を除きます)。
nvIconIndex	プログラムアイテムに使うアイコンのインデックスを戻します。(szItemName がサブフォルダーの場合を除きます)。
svShortCutKey	アイテムのショートカットキーを戻します。(szItemName がサブフォルダーの場合を除きます)。

テーブル 1 ・ QueryProgItem のパラメーター (続き)

パラメーター	説明
nvMinimizeFlag	(szItemName がサブフォルダーの場合を除きます)。アプリケーションウィンドウが最初に表示されたときに最小化するか否かを示す、次の定数のを戻します。 ・ NULL ― アプリケーションのウィンドウは、スタートアップ時には最小化されないことを示します。 ・ RUN_MINIMIZED ― アプリケーションのウィンドウは、スタートアップ時に最小化されることを示します。

戻り値

テーブル 2 ・ QueryProgItem の戻り値

戻り値	説明
IS_ITEM (0)	szItemName がプログラムアイテムまたは szFolderName のショートカットであることを示します。
IS_FOLDER (1)	szItemName が szFolderName のサブフォルダーであることを示します。
< 0	関数がプログラムアイテムまたはサブフォルダー名を検出できなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

[スタート] メニューの配置は各言語別によって異なります。InstallScript エンジンが自動的に正しいパスを選択します。

QueryProgItem の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* QueryProgItem 関数のデモンストレーションを行います。
*
* QueryProgItem が呼び出されて、ターゲットファイルまたはフォルダーの
* 属性を検出します。
```

```
    *
    * メモ : このスクリプトを実行する前に、
    *      FOLDER_NAME と ITEM_NAME が
    *      既存のフォルダー名やフォルダーアイテムを参照するように設定します。
    *
    ¥*---------------------------------------------------------------*/
    // ファイル名またはフォルダー名を参照するように定数を定義します。
    #define FOLDER_NAME"C:¥¥Windows¥¥Start Menu¥¥Programs"
    #define ITEM_NAME"InstallShield"

    // ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
    #include "Ifx.h"

        export prototype ExFn_QueryProgItem(HWND);

    function ExFn_QueryProgItem(hMSI)
        STRING svCmdLine, svWrkDir, svIconPath;
        STRING svShortCutKey, svGroupPath, szTitle, szMsg, szInfo, svMinFlag;
        STRING svMinimizeFlag;
        NUMBER nvIconIndex, nvMinimizeFlag, nResult, nvMinFlag;
        LIST listInfo, listID;
    begin

        // FOLDER_NAME フォルダーでアイテムを検索します。
        nResult = QueryProgItem (FOLDER_NAME, ITEM_NAME, svCmdLine, svWrkDir,
                                 svIconPath, nvIconIndex, svShortCutKey,
                                 nvMinimizeFlag);

        // 文字列リストを作成します。
        listInfo = ListCreate (STRINGLIST);

        // QueryProgItem エラーをチェックします。
        if (nResult < 0) then
            // エラーをレポートし、中止します。
            MessageBox ("QueryProgItem が失敗しました。", SEVERE);
            abort;
        // アイテムがアプリケーションか否かを確認します。
        elseif (nResult = IS_ITEM) then
            // コマンドラインを文字列リストへ追加します。
            Sprintf(szInfo, "%s のコマンドライン : %s", ITEM_NAME, svCmdLine);
            ListAddString(listInfo, szInfo, AFTER);

            // 作業ディレクトリを文字列リストへ追加します。
            Sprintf(szInfo, "%s の作業ディレクトリ : %s", ITEM_NAME, svWrkDir);
            ListAddString(listInfo, szInfo, AFTER);

            // アイコンパスを文字列リストへ追加します。
            Sprintf(szInfo, "%s のアイコン パス : %s", ITEM_NAME, svIconPath);
            ListAddString(listInfo, szInfo, AFTER);

            // アイコンインデックスを文字列リストへ追加します。
            Sprintf(szInfo, " アイコンのインデックス : %d", nvIconIndex);
            ListAddString(listInfo, szInfo, AFTER);

            // 文字列リストへショートカットキーを追加します。
            Sprintf (szInfo, "%s のショートカットキー : %s", ITEM_NAME,
                    svShortCutKey);
            ListAddString(listInfo, szInfo, AFTER);
```

```
    // アイテムがフォルダーか否かを確認します。
elseif (nResult = IS_FOLDER) then
    // メッセージを文字列リストへ追加します。
    Sprintf (szInfo, " アイテムはサブフォルダーです。QueryProgItem は " +
          " サブフォルダーに関する情報を充分に読み出しません。");
    ListAddString(listInfo, szInfo, AFTER);
endif;

// 文字列リストを表示します。
szTitle = "QueryProgItem の例 ";
szMsg = " アイテムの属性は次の通りです :";
SdShowInfoList (szTitle, szMsg, listInfo);

// リストを破棄します。
ListDestroy(listID);

end;
```

QueryShellMgr

QueryShellMgr 関数は Microsoft Windows が利用するプログラムシェルの名前を取得します。たとえば、プログラムシェルが Explorer の場合、QueryShellMgr は svShellMgrName に文字列 “Explorer.exe” を戻します。

構文

```
QueryShellMgr ( svShellMgrName );
```

パラメーター

テーブル 3 ・ QueryShellMgr のパラメーター

パラメーター	説明
svShellMgrName	現在実行中のシェルマネージャーの非完全修飾名 (ドライブ指定またはパス指定無し) を戻します。

戻り値

テーブル 4 ・ QueryShellMgr の戻り値

戻り値	説明
0	関数がプログラムシェルの名前を読み出したことを示します。
< 0	関数がプログラムシェルの名前を読み出すことができなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

ターゲットシステムのシェルが Explorer 以外の場合、LaunchApp 関数を使ってシェルを起動する必要があります。プログラムフォルダーやプログラムアイコンを作成する InstallScript 関数は、プログラムフォルダーやプログラムアイコンを作成するシェルとの DDE 対話を利用します。Norton Desktop など、ほとんどの代替シェルは Explorer シェルをエミュレートしています。このためプログラムフォルダーやアイテムを作成することが可能となります。

Explorer シェルをエミュレートしないシェルで、InstallShield はプログラムフォルダー関数やプログラムアイテム関数を利用してプログラムフォルダーやプログラムアイテムを作成または変更することはできません。シェルのメーカーに問い合わせて、Microsoft DDE 仕様を利用して特定のプログラムフォルダーやプログラムアイテムをどのように処理するのかを判断してください。

QueryShellMgr の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* QueryShellMgr 関数のデモンストレーションを行います。
```

```
*
* QueryShellMgr が呼び出され、シェルマネージャーの名前を検索
* します。そして名前がメッセージボックスに表示されます。
*
¥*-----------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_QueryShellMgr(HWND);

function ExFn_QueryShellMgr(hMSI)
   STRING svShellMgrName, szTitle, szMsg;
   NUMBER nReturn;
begin

   // プログラムシェルの名前を取得します。
   nReturn = QueryShellMgr (svShellMgrName);

   if (nReturn < 0) then
      // エラーを報告します。
      MessageBox ("プログラムシェルを読み出すことができませんでした。", SEVERE);
   else
      // プログラムシェルの名前を表示します。
      MessageBox ("シェルマネージャーは " + svShellMgrName + " です。", INFORMATION);
   endif;

end;
```

ReadArrayProperty

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ReadArrayProperty 関数は、値が配列である指定のプロパティの値を読み取るオブジェクトスクリプトで呼び出されます。

構文

```
ReadArrayProperty ( nPropertyBag, szPropertyName, ArrayPointer );
```

パラメーター

テーブル 5 - ReadArrayProperty のパラメーター

パラメーター	説明
nPropertyBag	プロパティの値が保存される、オブジェクトのプロパティバッグオブジェクトへの参照を指定します (セットアップエンジンは nPropertyBag の値を ReadBoolProperty が呼び出される ReadProperties 関数ブロックへ渡します。この値は [InstallScript] ビューで [新規プロパティの追加] ダイアログを利用するときに オブジェクトスクリプトに配置されます。
szPropertyName	値を読み取るプロパティの名前を指定します。
ArrayPointer	指定の配列プロパティにポインターを返します。

戻り値

テーブル 6 - ReadArrayProperty の戻り値

戻り値	説明
0	この関数は常にゼロ (0) を返します。

ReadBoolProperty

プロジェクト - *この情報は、InstallScript プロジェクトに適用します。*

ReadBoolProperty 関数は、その値がブール型である指定したプロパティの値を読み取るオブジェクト スクリプトで呼び出されます。

構文

ReadBoolProperty (nPropertyBag, szPropertyName, bvPropertyValue);

パラメーター

テーブル 7 - ReadBoolProperty のパラメーター

パラメーター	説明
nPropertyBag	プロパティの値が保存される、オブジェクトのプロパティバッグオブジェクトへの参照を指定します (セットアップエンジンは nPropertyBag の値を ReadBoolProperty が呼び出される ReadProperties 関数ブロックへ渡します。この値は [InstallScript] ビューで [新規プロパティの追加] ダイアログを利用するときに オブジェクトスクリプトに配置されます。
szPropertyName	値を読み取るプロパティの名前を指定します。
bvPropertyValue	指定のプロパティの値を返します。

戻り値

テーブル 8 - ReadBoolProperty の戻り値

戻り値	説明
0	この関数は常にゼロ (0) を返します。

ReadBytes

ReadBytes 関数は、現在のファイルのポインター位置からファイルの特定バイト数を読み取ります。この関数が戻る際、InstallShield はファイルポインターをファイルから読み取ったバイトの最後の新しい位置へ再配置します。

メモ - *ファイル (インターネット上のファイルも可能) から読み取る前に OpenFileMode と OpenFile を呼び出してバイナリモードでファイルを開かなくてはなりません。*

nIndex パラメーターは svString が指定する値へのインデックスです。パラメーター nBytes を利用して nIndex から先にあるバイトをいくつファイルから読み取るのかを指定します。nIndex と nBytes を足した値が svString よりも長い場合、ファイルからは文字列のインデックスから文字列の終わりのバイト数のみを読み取ります。例えば、svString の長さが 100 バイト と宣言されていていて、パラメーター nIndex が 50 バイトと宣言されパラメーター nBytes が 75 バイトの場合、49 バイトから 99 バイトの間のみ (75 バイトではなく 50 バイト) を読み取ります。

構文

```
ReadBytes ( nFileHandle, svString, nIndex, nBytes );
```

パラメーター

テーブル 9 ・ ReadBytes のパラメーター

パラメーター	説明
nFileHandle	バイナリモードで開いたファイルへのファイルハンドルを指定します。
svString	ファイルから読み出したバイトを戻します。このパラメーターで渡される変数はサイズが明示的に宣言されていなくてはなりません。そして、サイズは nBytes が指定するバイト数を受け取るのに充分な大きさでなくてはなりません。
nIndex	svString; へのインデックスを指定します。ファイルからのデータはこの位置で文字列に挿入されます。
nBytes	ファイルから読み取るバイト数を指定します。現在のファイル ポインターの位置からバイト数の読み取りが開始されます。InstallShield は、バイトの読み取りに応じてファイルポインターを移動させます。

戻り値

テーブル 10 ・ ReadBytes の戻り値

戻り値	説明
X	関数がファイルからバイトを読み取ったことを示します。X は実際には svString に戻されたバイト数です。
< 0	関数がファイルから読み取ることができなかったことを示します。

ReadBytes の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
```

```
＊ReadBytes 関数と SeekBytes 関数のデモンストレーションを行います。
＊
＊SeekBytes が呼び出され、バイナリ モードで開かれたファイルの
＊特定の場所へファイルポインターを配置
＊します。そして ReadBytes はこの場所からはじまる特定の数のバイトを
＊読み取ります。バイトは文字列に読み込まれ、
＊そしてメッセージボックスに表示されます。
＊
＊メモ : 定義された定数 EXAMPLE_DIR and EXAMPLE_BIN は
＊　　ターゲットシステムの既存のディレクトリとファイルに設定しなくては
＊　 設定してください。
＊
¥*---------------------------------------------------------------*/

#define EXAMPLE_DIR  "C:¥¥"
#define EXAMPLE_BIN "Example.bin"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ReadBytes(HWND);

function ExFn_ReadBytes(hMSI)
   STRING svString;
   NUMBER nvFileHandle;
begin

   // ファイルモードを読み取り / 書き込みモードに設定します。
   OpenFileMode (FILE_MODE_BINARY);

   // バイナリファイルを開きます。
   if (OpenFile (nvFileHandle, EXAMPLE_DIR, EXAMPLE_BIN) < 0) then
      // エラーをレポートしてから、中止します。
      SprintfBox (SEVERE, "CopyBytes の例 ", "%s を開くことができませんでした。",
            EXAMPLE_BIN);
      abort;
   endif;

   // ファイルポインターをファイルの 16 番目のバイトへ設定します。
   SeekBytes (nvFileHandle, 15, FILE_BIN_START);

   // 次の 28 バイトを svString へ読み込みます。
   if (ReadBytes (nvFileHandle, svString, 0, 28) < 0) then
      // エラーを報告します。
      MessageBox ("ReadBytes が失敗しました。", SEVERE);
   else
      // 文字列を表示します。
      SprintfBox (INFORMATION, "ReadBytes の例 ", " バイトは : %s",
            svString);
   endif;

   // ファイルを閉じます。
   CloseFile (nvFileHandle);

end;
```

ReadNumberProperty

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ReadNumberProperty 関数は、値が数値である指定のプロパティの値を読み取るオブジェクトスクリプトで呼び出されます。

構文

ReadNumberProperty (nPropertyBag, szPropertyName, nvPropertyValue);

パラメーター

テーブル 11 · ReadNumberProperty のパラメーター

パラメーター	説明
nPropertyBag	プロパティの値が保存される、オブジェクトのプロパティバッグオブジェクトへの参照を指定します (セットアップエンジンは nPropertyBag の値を ReadBoolProperty が呼び出される ReadProperties 関数ブロックへ渡します。この値は [InstallScript] ビューで [新規プロパティの追加] ダイアログを利用するときに オブジェクトスクリプトに配置されます。
szPropertyName	値を読み取るプロパティの名前を指定します。
nvPropertyValue	指定のプロパティの値を返します。

戻り値

テーブル 12 · ReadNumberProperty の戻り値

戻り値	説明
0	この関数は常にゼロ (0) を返します。

ReadStringProperty

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ReadStringProperty 関数は、値が文字列である指定のプロパティの値を読み取るオブジェクトスクリプトで呼び出されます。

構文

ReadStringProperty (nPropertyBag, szPropertyName, svPropertyValue);

パラメーター

テーブル 13 ・ ReadStringProperty のパラメーター

パラメーター	説明
nPropertyBag	プロパティの値が保存される、オブジェクトのプロパティバッグオブジェクトへの参照を指定します (セットアップエンジンは nPropertyBag の値を ReadBoolProperty が呼び出される ReadProperties 関数ブロックへ渡します。この値は [InstallScript] ビューで [新規プロパティの追加] ダイアログを利用するときに オブジェクトスクリプトに配置されます。
szPropertyName	値を読み取るプロパティの名前を指定します。
svPropertyValue	指定のプロパティの値を返します。

戻り値

テーブル 14 ・ ReadStringProperty の戻り値

戻り値	説明
0	この関数は常にゼロ (0) を返します。

RebootDialog

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

RebootDialog 関数は、エンド ユーザーがコンピューターを再起動するかどうかを指定できるメッセージ ボックスを表示します。選択したオプションは、インストールの最後に実行されます。

SHAREDFILE オプションまたは LOCKEDFILE オプションを指定して関数を呼び出した際に、ロックされた .dll または .exe ファイルがあると、ロックされたファイルの最新バージョンがターゲットシステムにコピーされ、システム変数 BATCH_INSTALL が TRUE に設定されます。ユーザーが **[後でコンピューターを再起動する]** オプションを選択しない限り、システムが再起動されると **RebootDialog** がアップデート用にロックされたファイルを自動的にコミットします。

InstallScript エンジンは、インストールの別のインスタンスが実行中、できる限りシステムの再起動を回避しようとします。このため、**RebootDialog** を呼び出す前にその他すべてのインスタンスを確実にシャットダウンするようにして下さい。エンドユーザーへのメッセージにも、システムを再起動する前に他のアプリケーションがすべて閉じていることを確認する内容を加えてください。

*メモ・**RebootDialog** 関数の代わりに SdFinishReboot 利用すると、**RebootDialog** ダイアログより使いやすく見た目の良いダイアログを作成することができます。*

構文

RebootDialog (szTitle, szMsg, nDefChoice);

パラメーター

テーブル 15 · RebootDialog のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル ([Windows の再起動]) を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
nDefChoice	デフォルトのオプションボタン選択を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · SYS_BOOTMACHINE ― コンピューターを再起動するオプション ([今すぐコンピューターを再起動する]) が、オプションボタン選択のデフォルトになります。 · 0 ― コンピューターを再起動しないオプション ([後でコンピューターを再起動する]) が、オプションボタン選択のデフォルトになります。

戻り値

テーブル 16 · RebootDialog の戻り値

戻り値	説明
WILL_REBOOT	エンドユーザーが [今すぐコンピューターを再起動する] オプションボタンを選択したことを示します。
0	エンドユーザーが [後でコンピューターを再起動する] オプションオボタンを選択したことを示します。

追加情報

RebootDialog 関数によって表示されるメッセージ ボックスは、スキンを使って表示することはできません。スキンの指定に関わらず、同じように表示されます。

RebootDialog Example

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * RebootDialog 関数のデモンストレーションを行います。
 *
 * このスクリプトは RebootDialog を呼び出して、ユーザーにコンピューターを再起動するかどうかを
 * 質問するダイアログを表示します。ここで
 * RebootDialog は、パラメーター 2 にヌル文字列を渡して既定メッセージを
 * 表示し、パラメーター 3 に 0 を渡してデフォルトの選択を
 * " いいえ、あとでコンピューターを再起動します。" に設定します。
 *
 * 警告 : エンドユーザーがダイアログで [ はい ] を選択した場合、
 *        コンピューターが再起動されます。
 *
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "RebootDialog の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RebootDialog(HWND);

function ExFn_RebootDialog(hMSI)
   NUMBER nvDefChoice;
begin

   // コンピューターの再起動を問い合わせます。
   RebootDialog (TITLE_TEXT, "", 0);

end;
```

RegDBConnectRegistry

プロジェクト・InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。

RegDBConnectRegistry 関数はリモートレジストリへ接続します。接続が完了したあと、ローカルレジストリと同様にリモートレジストリ上にあるレジストリキー、値名、そして値ペアを作成、削除または読み出しが可能になります。この機能は、64 ビット システムでもサポートされていますが、いくつか制限があります。

メモ・この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。

RegDBConnectRegistry では、リモートレジストリが開かれるたびに 1 つのレジストリルートキーのみを変更することができます。また HKEY_LOCAL_MACHINE または HKEY_USERS の何れかにあるキーと値のみを編集することができます。RegDBConnectRegistry を呼び出す時、どのルートキーを編集するのかを指定しなくてはなりません。その他のルートキーまたはそのサブキーを編集するには、接続を一度閉じてから再度開かなくてはなりません。

***注意**・RegDBConnectRegistry を呼び出してルートキーを設定したため、リモートレジストリへの接続を設立した後に RegDBSetDefaultRoot を呼び出すことはできません。RegDBDisConnectRegistry を呼び出したとき、レジストリ関連関数の呼び出しはすべてローカルレジストリに影響します。その後 RegDBSetDefaultRoot を呼び出してルートキーを変更します。*

メモ・リモート Windows システム上にあるレジストリを開く場合は、管理者権限が必要です。この関数はネットワークインストールを行うシステム管理者専用です。

構文

RegDBConnectRegistry (szRemoteSystem, nKeyType, nReserved);

パラメーター

テーブル 17 · RegDBConnectRegistry のパラメーター

パラメーター	説明
szRemoteSystem	“RemoteSys” など、接続先システムの名前を指定します。このパラメーターでヌル文字列 (“”) を渡すと、ローカル レジストリへの接続が作成されます。
nKeyType	定数 HKEY_LOCAL_MACHINE または HKEY_USERS のどちらかを指定します :
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。

戻り値

テーブル 18 · RegDBConnectRegistry の戻り値

戻り値	説明
0	関数がシステムレジストリへの接続を完了したことを示します。
REGDB_ERR_CONNECTIONEXISTS (-6)	既存のリモートレジストリへの接続。これは RegDBConnectRegistry を再び呼び出す前に RegDBDisConnectRegistry を使って閉じなくてはなりません。
REGDB_ERR_CORRUPTEDREGISTRY (-4)	リモートレジストリが破損しているまたはアクセス不可能であることを示します。
REGDB_ERR_INITIALIZATION (-2)	レジストリサービスが初期化できなかったことを示します。[リモート管理] が有効となっていること、そしてレジストリへ書き込むことのできる適切な権限を持っていることを確認してください。
REGDB_ERR_INVALIDHANDLE (-5)	リモートレジストリ用のキー名が無効です。
REGDB_ERR_INVALIDNAME (-3)	szRemoteSystem のシステムが検出されなかったことを示します。名前を確認してからもう一度実行してください。
-1	その他のエラー。

大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、“<my registy entry text>”) は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。

RegDBConnectRegistry の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* RegDBConnectRegistry 関数 と RegDBDisConnectRegistry 関数の
* デモンストレーションを行います。
*
* メモ : このスクリプトを適切に実行するため、
*      プリプロセッサ定数をリモート管理者が有効になっている
*      適切なリモートコンピューターへ設定しなくてはなりません。両方の
*    コンピューターでリモートレジストリサービスが有効でなくてはなりません。
*
¥*---------------------------------------------------------------*/

#define REMOTE "IShield_NT1"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RegDBConnectRegistry(HWND);

function ExFn_RegDBConnectRegistry(hMSI)
   STRING szRemoteMachine, szKey, szTitle, szMsg;
   NUMBER nKeyType, nReturn;
begin

   szTitle       = "RegDBConnectRegistry & RegDBDisConnectRegistry";
   szRemoteMachine = REMOTE;
   nKeyType      = HKEY_LOCAL_MACHINE;
   szMsg         = " セットアップは %s へ接続します。";

   SprintfBox (INFORMATION, szTitle, szMsg, szRemoteMachine);

   // リモートコンピューターのレジストリへ接続します。すべてのレジストリ関連の関数呼び出しは
   // リモートコンピューターのみを変更します。
   nReturn = RegDBConnectRegistry (szRemoteMachine, nKeyType, 0);

   if (nReturn < 0) then
      szMsg = "RegDBConnectRegistry が失敗しました。¥n¥n リモートシステムへ接続することができませんでした。" +
            "";
      MessageBox (szMsg, SEVERE);
      abort;
   else
      szMsg      = "%s へ接続されました。";
      SprintfBox (INFORMATION, szTitle, szMsg, szRemoteMachine);
   endif;

   // リモートコンピューター上でキーを作成します。
   szKey   = "SOFTWARE¥¥InstallShield¥¥Test Key";
   nReturn = RegDBCreateKeyEx(szKey, "");
```

```
    if (nReturn < 0) then
        szMsg = "RegDBCreateKeyEx が失敗しました。¥n¥n¥ リモートマシンでキーを作成することができませんでした。" +
                "";
        MessageBox (szMsg, SEVERE);
    else
        szMsg = "%s に %s を作成しました。";
        SprintfBox (INFORMATION, szTitle, szMsg, szKey, szRemoteMachine);

        // キーがリモートレジストリに存在することを確認します。
        nReturn = RegDBKeyExist(szKey);

        if (nReturn < 0) then
            szMsg = "RegDBKeyExist が失敗しました。¥n¥n リモートキーが存在しません。";
            MessageBox (szMsg, SEVERE);
        else
            szMsg = "%s が存在します。";
            SprintfBox (INFORMATION, szTitle, szMsg, szKey);
        endif;
    endif;

    // リモートコンピューターで作成されたキーを削除します。
    nReturn = RegDBDeleteKey(szKey);

    if (nReturn < 0) then
        MessageBox("RegDBDeleteKey が失敗しました。¥n¥n リモートキーを削除することができませんでした。",
                   INFORMATION);
    else
        szMsg = "%s の %s を削除しました。";
        SprintfBox (INFORMATION, szTitle, szMsg, szKey, szRemoteMachine);
    endif;

    // リモートレジストリから切断します。すべてのレジストリ関連の関数は
    // ローカルレジストリのみを変更します。
    nReturn = RegDBDisConnectRegistry(0);

    if (nReturn < 0) then
        MessageBox("RegDBDisConnectRegistry が失敗しました。¥n¥n リモートレジストリに " +
                   " 接続されています。", SEVERE);
    else
        MessageBox("RegDBDisConnectRegistry が成功しました。¥n¥n リモートレジストリが " +
                   " 未接続状態です。", INFORMATION);
    endif;

end;
```

RegDBCopyKeys

プロジェクト・*InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBCopyKeys 関数は szSourceKe が指定するキーの下にあるレジストリキーおよび値を szTargetKey が指定するキーへコピーします。

メモ・この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。

構文

RegDBCopyKeys (szSourceKey, szTargetKey, nRootKeySource, nRootKeyTarget);

パラメーター

テーブル 19 - RegDBCopyKeys のパラメーター

パラメーター	説明
szSourceKey	そのサブキーと値をコピーするキーの名前を指定します。2 つの円記号 (¥¥) を使って、サブキーの異なるレベルを分割してください。
szTargetKey	コピーする先のキーの名前を指定します。2 つの円記号 (¥¥) を使って、サブキーの異なるレベルを分割してください。キーが存在しない場合は、RegDBCopyKeys が作成します。キーが存在しない場合、szSourceKey の下にある値と同じ名前を持つキーの下にある既存値が上書きされます。これには同一名のサブキーの下にある値が含まれます。
nRootKeySource	szSourceKey のルートキーを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • HKEY_CLASSES_ROOT • HKEY_CURRENT_USER • HKEY_LOCAL_MACHINE • HKEY_USERS • HKEY_CURRENT_CONFIG • HKEY_DYN_DATA
nRootKeyTarget	szTargetKey のルートキーを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • HKEY_CLASSES_ROOT • HKEY_CURRENT_USER • HKEY_LOCAL_MACHINE • HKEY_USERS • HKEY_CURRENT_CONFIG • HKEY_DYN_DATA

戻り値

テーブル 20 - RegDBCopyKeys の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がキーと値をコピーしたことを示します。

テーブル 20 - RegDBCopyKeys の戻り値 (続き)

戻り値	説明
くISERR_SUCCESS	関数がキーおよび値をコピーできなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、"<my registy entry text>") は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。

RegDBCopyValues

プロジェクト - *InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBCopyValues 関数は、szSourceKe が指定するキーの下にあるレジストリ値を szTargetKey が指定するキーへコピーします。

メモ - *この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。*

構文

RegDBCopyValues (szSourceKey, szTargetKey, nRootKeySource, nRootKeyTarget);

パラメーター

テーブル 21 ・ RegDBCopyValues のパラメーター

パラメーター	説明
szSourceKey	その値をコピーするキーの名前を指定します。2 つの円記号 (¥¥) を使って、サブキーの異なるレベルを分割してください。
szTargetKey	コピーする先のキーの名前を指定します。2 つの円記号 (¥¥) を使って、サブキーの異なるレベルを分割してください。キーが存在しない場合は、RegDBCopyKeys が作成します。キーが存在しない場合、szSourceKey の下にある値と同じ名前を持つキーの下にある既存値が上書きされます。
nRootKeySource	szSourceKey のルートキーを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ HKEY_CLASSES_ROOT ・ HKEY_CURRENT_USER ・ HKEY_LOCAL_MACHINE ・ HKEY_USERS ・ HKEY_CURRENT_CONFIG ・ HKEY_DYN_DATA
nRootKeyTarget	szTargetKey のルートキーを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ HKEY_CLASSES_ROOT ・ HKEY_CURRENT_USER ・ HKEY_LOCAL_MACHINE ・ HKEY_USERS ・ HKEY_CURRENT_CONFIG ・ HKEY_DYN_DATA

戻り値

テーブル 22 ・ RegDBCopyValues の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数が値をコピーしたことを示します。
< ISERR_SUCCESS	関数が値をコピーできなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、"<my registy entry text>") は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。

RegDBCreateKeyEx

プロジェクト・*InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBCreateKeyEx 関数は、レジストリでキーを作成します。また、クラスオブジェクトを新規に作成されたキーに関連付けることも可能です (上級ユーザーのみ)。新規に作成されたキーは、関連付けられた値を持っていません。

ログが有効な場合、InstallScript エンジンは、szKey を通して **RegDBCreateKeyEx** に渡されるパスに含まれる各キーをログ記録します。たとえば、szKey で次のキーをパスできます :

"Software" ^ IFX_COMPANY_NAME ^ IFX_PRODUCT_NAME

この場合、InstallScript エンジンはいくつかのキーをログ記録します :

- ソフトウェア
- IFX_COMPANY_NAME 変数の値
- IFX_PRODUCT_NAME 変数の値

キーが既に存在しなかった場合、ログではそのキーが “created” としてフラグされます。前述の例で、Software キーはすべてのターゲット システム (HKEY_LOCAL_MACHINE or HKEY_CURRENT_USER) に既存します。Software キーはログされますが、“created” フラグは false で、アンインストール中にそのキーが削除されないことを示します。作成およびログ記録されたキーは、インストールの実行時に既に存在していなかったため、すべてアンインストール中に削除されます。

キーがアンインストールされたとき、そのサブキーもアンインストールされます。そのため、**RegDBCreateKeyEx** を使って既にアンインストール向けにログ記録されているキーの下にキー (単数または複数) を作成した場合、作成したキーは、それよりも高い階層のキーがアンインストールされた時に一緒にアンインストールされます。この動作はインストールがキーを作成したときにログ記録が有効であったかどうか、およびインストールがキーを作成した順番には影響されません。したがって前述の例では、最初のインストールが作成した IFX_COMPANY_NAME キーの下に、同じく **RegDBCreateKeyEx** を使って異なる製品のキーを作成する 2 番目のインストールでもログ記録が有効である場合に、エンド ユーザーが最初の製品をアンインストールしたとき、IFX_COMPANY_NAME キー全体が削除されて、両方の製品のサブキーが削除されます。これによって、2 番目の製品が誤作動を起こす可能性があります。

複数のインストール間でレジストリ キーの共有を可能にするためには、**RegDBCreateKeyEx** 関数ではなく、[レジストリ] ビューを使ってレジストリ エントリを構成することが推奨されます。InstallScript プロジェクトの [レジストリ] ビューでは、レジストリ キーを共有としてマークすることができます (キーを右クリックしてから **[複数のアプリケーション間で共有]** をクリックします)。アンインストール中、InstallScript エンジンはキーを共有するその他のログ記録されているインストールがマシンに残っていない場合のみ、共有キーを削除します。

インストール中にログ記録されたすべてのレジストリ キーを参照して、各キーの “created” フラグの状態を確認するには、InstallShield キャビネット & ログ ファイル ビューアーを使用します。

ログ記録に関する詳細については、「アンインストール用にログされた InstallScript 関数」を参照してください。

メモ・この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。

構文

RegDBCreateKeyEx (szKey, szClass);

パラメーター

テーブル 23 ・ RegDBCreateKeyEx のパラメーター

パラメーター	説明
szKey	作成するキーの名前を指定します。2 つの円記号 (¥¥) を使って、サブキーの異なるレベルを分割してください。 ルート キーが HKEY_CLASSES_ROOT の場合、このパラメーターで HKEY_CLASSES_ROOT を指定する必要はありません。特に指定しない限り、InstallScript エンジンはキーを HKEY_CLASSES_ROOT のサブキーとして作成します。別のルート キーを指定する場合、**RegDBCreateKeyEx** を呼び出す前に RegDBSetDefaultRoot を呼び出します。
szClass	このキーと関連付けるクラス名を指定します。

戻り値

テーブル 24 ・ RegDBCreateKeyEx の戻り値

戻り値	説明
0	関数がサブキーを作成したことを示します。
< 0	関数がサブキーを作成できなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

RegDBCreateKeyEx はレジストリ関連の一般関数で、レジストリ関連の特殊関数が処理するものを含んで、すべてのレジストリキーに利用することが可能です。レジストリ関連の特殊関数については、「レジストリ関連の特殊関数」を参照してください。

メモ・Windows では、HKEY_LOCAL_MACHINE または HKEY_USERS の下に直接キーを作成することはできません。

デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、"<my registy entry text>") は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 定数を使って **Disable** を呼び出します。

RegDBCreateKeyEx の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * RegDBCreateKeyEx 関数と RegDBKeyExist 関数のデモンストレーションを行います。
 *
 * まず、RegDBCreateKeyEx を呼び出して HKEY_CLASSES_ROOT キーに
 * クラス値を持たないサブキーを作成します。そして、RegDBKeyExist を呼び出し
 * キーが作成されたことを確認します。
 *
 * RegDBCreateKey を再び呼び出し、HKEY_CLASSES_ROOT の下に
 * 複数階層のサブキーをその関連値と共に作成します。
 * そして RegDBKeyExist を再び呼び出し、新規キーの存在を
 * 確認します。
 *
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "RegDBCreateKeyEx & RegDBKeyExist"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RegDBCreateKeyEx(HWND);

function ExFn_RegDBCreateKeyEx(hMSI)
   STRING szKey, szClass, szKeyRoot, szMsg, svLogFile;
   NUMBER nResult1, nResult2;
begin

   // クラス値を持たないキーを作成します。
   szKey   = "CreateKeyExample";
   szClass = "";

   if (RegDBCreateKeyEx(szKey, szClass) < 0) then
      MessageBox ("RegDBCreateKeyEx への最初の呼び出しに失敗しました。", SEVERE);
      abort;
   else
      SprintfBox (INFORMATION, TITLE_TEXT, "%s が正しく作成されました。", szKey);

      // 作成したばかりのキーが存在するかどうかを確認します。
      if (RegDBKeyExist (szKey) < 0) then
         MessageBox ("RegDBKeyExist への最初の呼び出しに失敗しました。", SEVERE);
      else
         SprintfBox (INFORMATION, TITLE_TEXT, "%s が存在します。.", szKey);
```

```
    endif;
endif;

if (RegDBDeleteKey (szKey) < 0) then
    MessageBox ("RegDBDeleteKey が失敗しました。", SEVERE);
endif;

// 複数のサブ階層やクラス値を持つキーを作成します。
szKey     = "ShareWare¥¥Games¥¥CoolChess";
szClass   = "LastPlayed";
szKeyRoot = "ShareWare";

if (RegDBCreateKeyEx(szKey, szClass) < 0) then
    MessageBox ("RegDBCreateKeyEx への 2 回目の呼び出しに失敗しました。", SEVERE);
    abort;
else
    SprintfBox (INFORMATION, TITLE_TEXT, "%s が正しく作成されました。", szKey);

    // 新しく作成した複数階層キーが存在するかどうかを確認します。
    if (RegDBKeyExist (szKeyRoot) < 0) then
        MessageBox ("RegDBKeyExist への 2 回目の呼び出しに失敗しました。", SEVERE);
    else
        SprintfBox (INFORMATION, TITLE_TEXT, "%s が存在します。.", szKey);
    endif;
endif;

if (RegDBDeleteKey (szKey) < 0) then
    MessageBox ("RegDBDeleteKey が失敗しました。", SEVERE);
endif;

end;
```

RegDBDeleteItem

プロジェクト・*InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBDeleteItem はレジストリ関連の特殊な関数で、特定の定義済みレジストリキーと共に動作します。**RegDBDeleteItem** 関数は、nItem の値に従ってアプリケーションごとのパスキーまたはアプリケーション アンインストール キーの下にある値を削除します。**RegDBDeleteItem** を REGDB_APPPATH または REGDB_APPPATH_DEFAULT オプションを使って呼び出すと、アプリケーションごとのパス キーが削除されます。

メモ・*InstallScript エンジンは、現在、レジストリの 64 ビット部分にある [プログラムの追加と削除] 情報の読み書きをサポートしません。そのため、REGDB_OPTIONS システム変数を使った REGDB_OPTION_WOW64_64KEY オプションをこのレジストリ関数で使用することはできません。REGDB_OPTION_WOW64_64KEY オプションを有効にしても、レジストリエントリがこの関数によって作成される場所に影響しません。*

構文

RegDBDeleteItem(nItem);

パラメーター

テーブル 25 · RegDBDeleteItem のパラメーター

パラメーター	説明
nItem	削除するアイテムを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • REGDB_APPPATH – アプリケーションごとのパス キーの下の Path 値のデータ。このキーは、CreateInstallationInfo を呼び出した結果としてインストールによって削除されます。**RegDBDeleteItem** を呼び出す前に、**CreateInstallationInfo** を呼び出してこのキーを作成する必要があります。(イベントベースのスクリプトでは、**CreateInstallationInfo** がデフォルトの **OnMoveData** イベント ハンドラーコードで呼び出されます。) • REGDB_APPPATH_DEFAULT – アプリケーションごとのパス キーの下の DefaultPath 値のデータ。このキーは、**CreateInstallationInfo** を呼び出した結果としてインストールによって作成されます。**RegDBDeleteItem** を呼び出す前に、**CreateInstallationInfo** を呼び出してこのキーを作成する必要があります。(イベントベースのスクリプトでは、**CreateInstallationInfo** がデフォルトの **OnMoveData** イベント ハンドラーコードで呼び出されます。) • REGDB_UNINSTALL_COMMENTS – アンインストール キーの下の Comments 値のデータ。 • REGDB_UNINSTALL_CONTACT – アンインストール キーの下の Contact 値のデータ。 • REGDB_UNINSTALL_DISPLAY_VERSION – アンインストールキーの下の DisplayVersion 値のデータ。 • REGDB_UNINSTALL_DISPLAYICON – アンインストールキーの下の DisplayIcon 値のデータ。 • REGDB_UNINSTALL_HELPLINK – アンインストール キーの下の HelpLink 値のデータ。 • REGDB_UNINSTALL_HELPTELEPHONE – アンインストール キーの下の HelpTelephone 値のデータ。 • REGDB_UNINSTALL_INSTALLDATE – アンインストール キーの下の InstallDate 値のデータ。 • REGDB_UNINSTALL_INSTALLLOC – アンインストール キーの下の InstallLocation 値のデータ。 • REGDB_UNINSTALL_INSTALLSOURCE – アンインストール キーの下の InstallSource 値のデータ。 • REGDB_UNINSTALL_LANGUAGE – アンインストールキーの下の Language 値のデータ。 • REGDB_UNINSTALL_LOGFILE – アンインストールキーの下の LogFile 値のデータ。 • REGDB_UNINSTALL_MAINT_OPTION – アンインストールキーの下の LogMode 値のデータ。

テーブル 25 · RegDBDeleteItem のパラメーター (続き)

パラメーター	説明
nItem (続き)	· REGDB_UNINSTALL_MAJOR_VERSION – アンインストールキーの下の VersionMajor 値のデータ。 · REGDB_UNINSTALL_MAJOR_VERSION_OLD – アンインストールキーの下の MajorVersion 値のデータ。 · REGDB_UNINSTALL_MINOR_VERSION – アンインストールキーの下の VersionMinor 値のデータ。 · REGDB_UNINSTALL_MINOR_VERSION_OLD – アンインストールキーの下の MinorVersion 値のデータ。 · REGDB_UNINSTALL_MODIFYPATH – アンインストール キーの下の ModifyPath の値。 · REGDB_UNINSTALL_NAME – アンインストールキーの下の DisplayName 値のデータ。 · REGDB_UNINSTALL_NOMODIFY – アンインストール キーの下の NoModif の値。 · REGDB_UNINSTALL_NOREMOVE – アンインストール キーの下の NoRemove の値。 · REGDB_UNINSTALL_NOREPAIR – アンインストールキーの下の NoRepair 値のデータ。 · REGDB_UNINSTALL_PRODUCTGUID – アンインストールキーの下の ProductGuid 値のデータ。 · REGDB_UNINSTALL_PRODUCTID – アンインストールキーの下の ProductId 値のデータ。 · REGDB_UNINSTALL_PUBLISHER – アンインストールキーの下の Publisher 値のデータ。 · REGDB_UNINSTALL_README – アンインストールキーの下の Readme 値のデータ。 · REGDB_UNINSTALL_REGCOMPANY – アンインストールキーの下の RegCompany 値のデータ。 · REGDB_UNINSTALL_REGOWNER – アンインストールキーの下の RegOwner 値のデータ。

テーブル 25 - RegDBDeleteItem のパラメーター (続き)

パラメーター	説明
nItem (続き)	• REGDB_UNINSTALL_STRING – アンインストール キーの下の UninstallString 値のデータ。UninstallString 値は、MaintenanceStart または DeinstallStart 関数が呼び出されたときに作成されます。 • REGDB_UNINSTALL_SYSTEMCOMPONENT – アンインストール キーの下の SystemComponent 値のデータ。 • REGDB_UNINSTALL_URLINFOABOUT – アンインストール キーの下の URLInfoAbout 値のデータ。 • REGDB_UNINSTALL_URLUPDATEINFO – アンインストール キーの下の URLUpdateInfo 値のデータ。 • REGDB_UNINSTALL_VERSION – アンインストールキーの下の Version 値のデータ。この定数が利用されたとき、szValue は例えば "16777216" のような文字列としてインストールされた製品のバージョン番号を指定します。 ***重要 -*** *REGDB_WINCURRVER_REGOWNER または REGDB_WINCURRVER_REGORGANIZATION をこの関数と共に使用しないでください。これらの値は Windows によって設定されるため、、インストールで削除することはできません。*

戻り値

テーブル 26 - RegDBDeleteItem の戻り値

戻り値	説明
0	関数が値を正常に設定したことを示します。
< 0	関数が失敗したことを示します。

追加情報

- デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、"<my registy entry text>") は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。
- **RegDBDeleteItem** はアンインストール ログ ファイルへ書き込みを行いません。したがって、この関数を呼び出しても、アプリケーションのアンインストール処理に影響はありません。

RegDBDeleteKey

プロジェクト - *InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべ*

てのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。

RegDBDeleteKey 関数はレジストリから特定のキーそその関連値を削除します。削除されたすべてのサブキーも、関連する値と共に削除されます。

InstallShield は szSubKey で指定したキーが HKEY_CLASSES_ROOT のサブキーとみなします。RegDBSetDefaultRoot を使って別のルートキーを指定できます。

RegDBDeleteKey はレジストリ関連の一般関数で、レジストリ関連の特殊関数が処理するものを含んで、すべてのレジストリキーに利用することが可能です。レジストリ関連の特殊関数については、「レジストリ関連の特殊関数」を参照してください。

メモ・この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。

構文

```
RegDBDeleteKey ( szSubKey );
```

パラメーター

テーブル 27 ・ RegDBDeleteKey のパラメーター

パラメーター	説明
szSubKey	削除するキーの名前を指定します。2 つの円記号 (¥¥) を使って、サブキーの異なるレベルを分割してください。

戻り値

テーブル 28 ・ RegDBDeleteKey の戻り値

戻り値	説明
0	関数がキーを削除したことを示します。
< 0	関数がキーを削除できなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

- デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、"<my registy entry text>") は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。
- **RegDBDeleteKey** はアンインストール ログ ファイルへ書き込みを行いません。したがって、この関数を呼び出しても、アプリケーションのアンインストール処理に影響はありません。

RegDBDeleteKey の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * RegDBDeleteKey 関数のデモンストレーションを行います。
 *
 * この例ではレジストリキーを作成して、それを削除します。
 * レジストリ関数と共に提供されている例のほとんどすべては
 * キーを削除するのにこの関数を利用します。さらに詳しい情報はこれらの例を
 * 参照してください。
 *
¥*---------------------------------------------------------------*/

#define TITLE_TEXT "RegDBDeleteKey の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RegDBDeleteKey(HWND);

function ExFn_RegDBDeleteKey(hMSI)
  STRING szKey, szClass, szKeyRoot, szMsg, svLogFile;
  NUMBER nResult1, nResult2;
begin

  // RegDBCreateKeyEx を呼び出すパラメーターをセットアップします。
  szKey   = "DeleteMeKey";
  szClass = "";

  // クラス値を持たないキーを作成します。
  if (RegDBCreateKeyEx (szKey, szClass) < 0) then
    MessageBox ("RegDBCreateKeyEx が失敗しました。", SEVERE);
    abort;
  else
    SprintfBox (INFORMATION, TITLE_TEXT, "%s が作成されました。", szKey);
  endif;

  // RegDBDeleteKey を呼び出して、作成したばかりのキーを削除します。
  if (RegDBDeleteKey (szKey) < 0) then
    MessageBox ("RegDBDeleteKey が失敗しました。", SEVERE);
  else
    SprintfBox (INFORMATION, TITLE_TEXT, "%s が削除されました。.", szKey);
  endif;

end;
```

RegDBDeleteValue

***プロジェクト・**InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBDeleteValue 関数はレジストリ内の特定のキーから値を削除します。InstallShield は szSubKey で指定したキーが HKEY_CLASSES_ROOT のサブキーとみなします。別のキーを指定するには、RegDBSetDefaultRoot を使わなくてはなりません。

RegDBDeleteKey はレジストリ関連の一般関数で、レジストリ関連の特殊関数が処理するものを含んで、すべてのレジストリキーに利用することが可能です。レジストリ関連の特殊関数については、「レジストリ関連の特殊関数」を参照してください。

***メモ・**この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。*

構文

RegDBDeleteValue (szSubKey, szValue);

パラメーター

テーブル 29 · RegDBDeleteValue のパラメーター

パラメーター	説明
szSubKey	削除する値名を含むレジストリキーの名前を指定します。2 つの円記号 (¥¥) を使って、サブキーの異なるレベルを分割してください。
szValue	削除する値の名前を指定します。

戻り値

テーブル 30 · RegDBDeleteValue の戻り値

戻り値	説明
0	関数が値を削除したことを示します。
< 0	関数が値を削除することができなかったことを示します。

追加情報

- デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、"<my registy entry text>") は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。

- **RegDBDeleteValu** はアンインストール ログ ファイルへ書き込みを行いません。したがって、この関数を呼び出しても、アプリケーションのアンインストール処理に影響はありません。

RegDBDeleteValue の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* RegDBDeleteValue 関数のデモンストレーションを行います。
*
* RegDBDeleteValue が呼び出され、次のレジストリキーから
* "Cursive" と名づけられた値を削除します :
*
* "HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Fonts".
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*      ターゲットシステム上の既存のサブキーと値を
*      参照するように設定してください。
*
¥*---------------------------------------------------------------------*/

#define SUBKEY "¥¥Software¥¥Microsoft¥¥Windows¥¥CurrentVersion¥¥Fonts"
#define VALUE"Cursive"
#define TITLE  "RegDBDeleteValue Example"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_RegDBDeleteValue(HWND);

function ExFn_RegDBDeleteValue(hMSI)
   STRING szSubKey, szValue, szTitle;
   NUMBER nReturn;
begin

   // ルートキーを設定します。
   RegDBSetDefaultRoot (HKEY_LOCAL_MACHINE);

   // サブキーの名前を設定します。
   szSubKey = SUBKEY;
   szValue  = VALUE;

   // サブキーを削除します。
   nReturn = RegDBDeleteValue (szSubKey, szValue);

   // 削除の結果をレポートします。
   if (nReturn < 0) then
      MessageBox ("RegDBDeleteValue が失敗しました。", SEVERE);
   else
      SprintfBox (INFORMATION, TITLE, "%s を削除しました。", szValue);
```

```
    endif;

end;
```

RegDBDisConnectRegistry

プロジェクト・*InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBDisConnectRegistry 関数は、RegDBConnectRegistry を呼び出して設定したリモートレジストリへの接続を閉じます。

RegDBDisConnectRegistry を呼び出したあと、InstallScript レジストリ関連の関数へのすべての呼び出しはローカルシステムのレジストリに影響します。レジストリ関連の特殊関数については、「レジストリ関連の特殊関数」を参照してください。

メモ・*この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。*

構文

```
RegDBDisConnectRegistry ( nReserved );
```

パラメーター

テーブル 31 ・ RegDBDisConnectRegistry のパラメーター

パラメーター	説明
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。

戻り値

テーブル 32 ・ RegDBDisConnectRegistry の戻り値

戻り値	説明
0	この関数がリモートシステム上のレジストリへの接続を閉じたことを示します。
< 0	この関数がレジストリ接続を閉じるのに失敗したことを示します。

RegDBDisConnectRegistry の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* RegDBConnectRegistry 関数 と RegDBDisConnectRegistry 関数の
* デモンストレーションを行います。
*
* メモ : このスクリプトを適切に実行するため、
*      プリプロセッサ定数をリモート管理者が有効になっている
*      適切なリモートコンピューターへ設定しなくてはなりません。両方の
*    コンピューターでリモートレジストリサービスが有効でなくてはなりません。
*
¥*---------------------------------------------------------------*/

#define REMOTE "IShield_NT1"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RegDBDisConnectRegistry(HWND);

function ExFn_RegDBDisConnectRegistry(hMSI)
   STRING szRemoteMachine, szKey, szTitle, szMsg;
   NUMBER nKeyType, nReturn;
begin

   szTitle        = "RegDBConnectRegistry & RegDBDisConnectRegistry";
```

```
szRemoteMachine = REMOTE;
nKeyType        = HKEY_LOCAL_MACHINE;
szMsg           = " セットアップは %s へ接続します。";

SprintfBox (INFORMATION, szTitle, szMsg, szRemoteMachine);

// リモートコンピューターのレジストリへ接続します。すべてのレジストリ関連の関数呼び出しは
// リモートコンピューターのみを変更します。
nReturn = RegDBConnectRegistry (szRemoteMachine, nKeyType, 0);

if (nReturn < 0) then
   szMsg = "RegDBConnectRegistry が失敗しました。¥n¥n リモートシステムへ接続することができませんでした。" +
           "";
   MessageBox (szMsg, SEVERE);
   abort;
else
   szMsg       = "%s へ接続されました。";
   SprintfBox (INFORMATION, szTitle, szMsg, szRemoteMachine);
endif;

// リモートコンピューター上でキーを作成します。
szKey   = "SOFTWARE¥¥InstallShield¥¥Test Key";
nReturn = RegDBCreateKeyEx(szKey, "");

if (nReturn < 0) then
   szMsg = "RegDBCreateKeyEx が失敗しました。¥n¥n¥ リモートマシンでキーを作成することができませんでした。" +
           "";
   MessageBox (szMsg, SEVERE);
else
   szMsg = "%s に %s を作成しました。";
   SprintfBox (INFORMATION, szTitle, szMsg, szKey, szRemoteMachine);

   // キーがリモートレジストリに存在することを確認します。
   nReturn = RegDBKeyExist(szKey);

   if (nReturn < 0) then
      szMsg = "RegDBKeyExist が失敗しました。¥n¥n リモートキーが存在しません。";
      MessageBox (szMsg, SEVERE);
   else
      szMsg = "%s が存在します。";
      SprintfBox (INFORMATION, szTitle, szMsg, szKey);
   endif;
endif;

// リモートコンピューターで作成されたキーを削除します。
nReturn = RegDBDeleteKey(szKey);

if (nReturn < 0) then
   MessageBox("RegDBDeleteKey が失敗しました。¥n¥n リモートキーを削除することができませんでした。",
              INFORMATION);
else
   szMsg = "%s の %s を削除しました。";
   SprintfBox (INFORMATION, szTitle, szMsg, szKey, szRemoteMachine);
endif;

// リモートレジストリから切断します。すべてのレジストリ関連の関数は
// ローカルレジストリのみを変更します。
nReturn = RegDBDisConnectRegistry(0);
```

```
    if (nReturn < 0) then
        MessageBox("RegDBDisConnectRegistry が失敗しました。¥n¥n リモートレジストリに " +
                " 接続されています。", SEVERE);
    else
        MessageBox("RegDBDisConnectRegistry が成功しました。¥n¥n リモートレジストリが " +
                " 未接続状態です。", INFORMATION);
    endif;

end;
```

RegDBGetAppInfo

プロジェクト・*InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBGetAppInfo 関数は、レジストリからメインアプリケーション情報キーに属する特定の値名の値を読み出します。RegDBGetAppInfo はレジストリ関連の特殊関数で、特定の定義済みレジストリキーと一緒に動作するように設計されています。

メモ・*InstallScript エンジンは、現在、レジストリの 64 ビット部分にある [プログラムの追加と削除] 情報の読み書きをサポートしません。そのため、REGDB_OPTIONS システム変数を使った REGDB_OPTION_WOW64_64KEY オプションをこのレジストリ関数で使用することはできません。REGDB_OPTION_WOW64_64KEY オプションを有効にしても、レジストリエントリがこの関数によって作成される場所に影響しません。*

構文

```
RegDBGetAppInfo ( szName, nvType, svValue, nvSize );
```

パラメーター

テーブル 33 - RegDBGetAppInfo のパラメーター

パラメーター	説明
szName	読み出す値の値名を指定します。
nvType	svValue に戻されるデータタイプを識別する、次の定義済み定数のひとつを戻します。 • REGDB_STRING – 文字列変数。newline 文字を利用することはできません。 • REGDB_STRING_EXPAND – "%MYPATH%" といった、拡張可能な環境変数式を持つ文字列変数です。 • REGDB_STRING_MULTI – 文字列変数。newline 文字を利用することができます。 • REGDB_NUMBER – 文字列として処理され、文字列変数で渡される数値。 • REGDB_BINARY – 文字列に格納されているバイナリデータ。
svValue	szName で指定した値名の値を戻します。
nvSize	戻り値のサイズをバイトで返します。

戻り値

テーブル 34 - RegDBGetAppInfo の戻り値

戻り値	説明
0	関数が値を戻したことを示します。
< 0	関数が値を読み出せなかったことを示します。

追加情報

デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、"<my registy entry text>") は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。

RegDBGetAppInfo の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
```

```
*
* RegDBSetAppInfo と RegDBGetAppInfo 関数を示します。
*
* これらの関数のいずれかを呼び出す前に、
* InstallationInfo を呼び出してアプリケーション情報を設定する必要があります。
*
¥*---------------------------------------------------------------------*/

#define COMPANY_NAME "Example_Company"
#define  PRODUCT_NAME          "Example_App"
#define  PRODUCT_VERSION       "5.0"
#define  PRODUCT_KEY           "EXAMPLE.EXE"
#define  DEINSTALL_KEY         "Example_DeinstKey"
#define  UNINSTALL_NAME        "Example_App_5.0"
#define  DEFAULT_LOG_PATH      "EXAMPLE"
#define  TITLE                 "RegDBGetAppInfo"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RegDBGetAppInfo(HWND);

function ExFn_RegDBGetAppInfo(hMSI)
   STRING  szStrName, szStrValue, svStrValue, szTitle, szMsg, svLogFile;
   NUMBER  nvSize, nvType;
begin

   // ルートキーを設定します。
   RegDBSetDefaultRoot (HKEY_LOCAL_MACHINE);

   // 使用する名前を REGDB_STRING で設定します。
   szStrName  = "ExampleStringValue";
   szStrValue = "ExampleStringSetting";

   //RegBDSetAppInfo と
   //RegDBGetAppInfo を使用する前に、アプリケーション情報をセットアップします。
   InstallationInfo (COMPANY_NAME, PRODUCT_NAME, PRODUCT_VERSION, PRODUCT_KEY);
   DeinstallStart(DEFAULT_LOG_PATH, svLogFile, DEINSTALL_KEY, 0);

   //REGDB_STRING 型の値を設定します。
   if (RegDBSetAppInfo (szStrName, REGDB_STRING, szStrValue, -1) < 0) then
      MessageBox ("REGDB_STRING 型のキーと値を設定できませんでした。", SEVERE);
      abort;
   endif;

   //RegDBGetAppInfo を呼び出して値を戻し、すべてのセットアップ
   // パラメーターを比較します。
   if (RegDBGetAppInfo (szStrName, nvType, svStrValue, nvSize) < 0) then
      MessageBox (" アプリケーション情報値の取得に失敗しました。", SEVERE);
      abort;
   else
      // 取得された値が設定された値と同じか確認します。
      if (nvType != REGDB_STRING) then
         MessageBox (" タイプの比較に失敗しました。", WARNING);
      endif;

      if (szStrValue != svStrValue) then
         MessageBox (" サブキー値の比較に失敗しました。", WARNING);
      else
```

```
            szMsg = " 値の設定 : %s = %s¥n¥n 戻り値 : %s = %s";
            SprintfBox (INFORMATION, TITLE, szMsg, szStrName, szStrValue,
                    szStrName, svStrValue);
        endif;

        if (nvSize != StrLength(szStrValue)) then
            MessageBox (" サイズの比較に失敗しました。", WARNING);
        else
            szMsg = " 入力されたバイトのサイズ : %d¥n¥n 戻されたバイトのサイズ : %d";
            SprintfBox (INFORMATION, TITLE, szMsg,
                    nvSize, StrLength(szStrValue) + 1);
        endif;
    endif;

end;
```

RegDBGetDefaultRoot

プロジェクト・*InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBGetDefaultRoot 関数は、レジストリ関連の一般関数が利用するルートキーを戻します。

メモ・*この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。*

構文

```
RegDBGetDefaultRoot ( );
```

パラメーター

なし。

戻り値

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_CONFIG
- HKEY_DYN_DATA
- HKEY_USER_SELECTABLE

メモ・戻り値 HKEY_USER_SELECTABLE は、関数が呼び出されたときに ALLUSERS システム変数がゼロ以外の場合は後に続くレジストリ関数呼び出しがルートキーとして HKEY_LOCAL_MACHINE を利用すること、または関数が呼び出されたときに ALLUSERS が FALSE の場合は後に続くレジストリ関数呼び出しがルートキーとして HKEY_CURRENT_USER を利用することを示します。

RegDBGetItem

***プロジェクト・**InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBGetItem 関数は、nItem の値によってアプリケーションごとのパスキーまたはアプリケーションアンインストールキーの値を読み出します。**RegDBSetItem** はレジストリ関連の特殊な関数で、特定の定義済みレジストリキーと共に動作します。

メモ・InstallScript エンジンは、現在、レジストリの 64 ビット部分にある [プログラムの追加と削除] 情報の読み書きをサポートしません。そのため、REGDB_OPTIONS システム変数を使った REGDB_OPTION_WOW64_64KEY オプションをこのレジストリ関数で使用することはできません。REGDB_OPTION_WOW64_64KEY オプションを有効にしても、レジストリエントリがこの関数によって作成される場所に影響しません。

構文

RegDBGetItem (nItem, svValue);

パラメーター

テーブル 35 · RegDBGetItem のパラメーター

パラメーター	説明
nItem	取得する項目を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • REGDB_APPPATH – アプリケーションごとのパス キーの下の Path 値のデータ。 • REGDB_APPPATH_DEFAULT – アプリケーションごとのパス キーの下の DefaultPath 値のデータ。 • REGDB_UNINSTALL_COMMENTS – アンインストール キーの下の Comments 値のデータ。 • REGDB_UNINSTALL_CONTACT – アンインストール キーの下の Contact 値のデータ。 • REGDB_UNINSTALL_DISPLAY_VERSION – アンインストールキーの下の DisplayVersion 値のデータ。 • REGDB_UNINSTALL_DISPLAYICON – アンインストールキーの下の DisplayIcon 値のデータ。 • REGDB_UNINSTALL_HELPLINK – アンインストール キーの下の HelpLink 値のデータ。 • REGDB_UNINSTALL_HELPTELEPHONE – アンインストール キーの下の HelpTelephone 値のデータ。 • REGDB_UNINSTALL_INSTALLDATE – アンインストール キーの下の InstallDate 値のデータ。 • REGDB_UNINSTALL_INSTALLLOC – アンインストール キーの下の InstallLocation 値のデータ。 • REGDB_UNINSTALL_INSTALLSOURCE – アンインストール キーの下の InstallSource 値のデータ。 • REGDB_UNINSTALL_LANGUAGE – アンインストールキーの下の Language 値のデータ。 • REGDB_UNINSTALL_LOGFILE – アンインストールキーの下の LogFile 値のデータ。 • REGDB_UNINSTALL_MAINT_OPTION – アンインストールキーの下の LogMode 値のデータ。

テーブル 35 · RegDBGetItem のパラメーター (続き)

パラメーター	説明
nItem (続き)	· REGDB_UNINSTALL_MAJOR_VERSION – アンインストールキーの下の VersionMajor 値のデータ。存在しない場合、関数はアンインストール キーの下にある MajorVersion 値のデータを確認します。 · REGDB_UNINSTALL_MAJOR_VERSION_OLD – アンインストールキーの下の MajorVersion 値のデータ。 · REGDB_UNINSTALL_MAJOR_VERSION – アンインストールキーの下の VersionMinor 値のデータ。存在しない場合、関数はアンインストール キーの下にある MinorVersion 値のデータを確認します。 · REGDB_UNINSTALL_MINOR_VERSION_OLD – アンインストールキーの下の MinorVersion 値のデータ。 · REGDB_UNINSTALL_MODIFYPATH – アンインストール キーの下の ModifyPath の値。 · REGDB_UNINSTALL_NAME – アンインストールキーの下の DisplayName 値のデータ。この定数が使用されると、szValue はコントロール パネル内のアンインストール可能なアプリケーションのリストで示されるアプリケーション名を指定します。 · REGDB_UNINSTALL_NOMODIFY – アンインストール キーの下の NoModif の値。 · REGDB_UNINSTALL_NOREMOVE – アンインストール キーの下の NoRemove の値。 · REGDB_UNINSTALL_NOREPAIR – アンインストールキーの下の NoRepair 値のデータ。 · REGDB_UNINSTALL_PRODUCTGUID – アンインストールキーの下の ProductGuid 値のデータ。 · REGDB_UNINSTALL_PRODUCTID – アンインストールキーの下の ProductId 値のデータ。 · REGDB_UNINSTALL_PUBLISHER – アンインストールキーの下の Publisher 値のデータ。 · REGDB_UNINSTALL_README – アンインストールキーの下の Readme 値のデータ。 · REGDB_UNINSTALL_REGCOMPANY – アンインストールキーの下の RegCompany 値のデータ。 · REGDB_UNINSTALL_REGOWNER – アンインストールキーの下の RegOwner 値のデータ。

テーブル 35 ・ RegDBGetItem のパラメーター (続き)

パラメーター	説明
nItem (続き)	・ REGDB_UNINSTALL_STRING – アンインストール キーの下の UninstallString 値のデータ。 ・ REGDB_UNINSTALL_SYSTEMCOMPONENT – アンインストール キーの下の SystemComponent 値のデータ。 ・ REGDB_UNINSTALL_URLINFOABOUT – アンインストール キーの下の URLInfoAbout 値のデータ。 ・ REGDB_UNINSTALL_URLUPDATEINFO – アンインストール キーの下の URLUpdateInfo 値のデータ。 ・ REGDB_UNINSTALL_VERSION – アンインストールキーの下の Version 値のデータ。この定数が利用されたとき、szValue は例えば "16777216" のような文字列としてインストールされた製品のバージョン番号を戻します。 ・ REGDB_WINCURRVER_REGORGANIZATION – 現在のバージョン キーの下の RegisteredOrganization 値のデータ。 ・ REGDB_WINCURRVER_REGOWNER – 現在のバージョン キーの下の RegisteredOwner 値のデータ。
svValue	項目の値を返します。

戻り値

テーブル 36 ・ RegDBGetItem の戻り値

戻り値	説明
0	関数は項目の値を正常に取得したことを示します。
< 0	項目の値が取得できなかったことを示します。

追加情報

デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、"<my registy entry text>") は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。

プロジェクト ・ *InstallScript インストールでは、**CreateInstallationInfo** 呼び出しの前に REGDB_APPPATH または REGDB_APPPATH_DEFAULT を使って **RegDBSetItem** を呼び出しても効果がありません。これは、**RegDBSetItem** が作成したレジストリ情報を **CreateInstallationInfo** が上書きするために、呼び出しが無効となるからです。**MaintenanceStart** 呼び出しの前に、その他の定数を使って **RegDBSetItem** を呼び出しても効果はありません。これは、**RegDBSetItem** が作成したレジストリ データを **MaintenanceStart** が上書きするために、呼び出しが無効となるからです。(イベント指向のスクリプトでは、**CreateInstallationInfo** および **MaintenanceStart** がデフォルト OnMoveData イベント ハンドラーコードで呼び出されます。)*

*InstallScript MSI インストールでは、アンインストール情報は Windows Installer によってファイル転送中に作成されます。このため、InstallScript MSI インストールで **RegDBSetItem** を呼び出す場合は、ファイル転送の後に行う必要があります。*

RegDBGetItem の例

***メモ・**基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * RegDBSetItem と RegDBGetItem 関数を示します。
 *
 * このスクリプトによっていくつかのレジストリキーが設定されます。そして
 * それらのキーを取得して現在の値を表示します。
 *
¥*---------------------------------------------------------------*/

#define  COMPANY_NAME       "ExampleCompany"
#define  PRODUCT_NAME       "ExampleProduct"
#define  VERSION_NUMBER     "5.00.00"
#define  PRODUCT_KEY        "EXAMPLE.EXE"
#define  DEINST_KEY         "ExampleDeinstKey"
#define  APP_DEF_LOG_PATH   "C:¥¥EXAMPLE¥¥TEMP"
#define  APP_PATH           "C:¥¥EXAMPLE"
#define  APP_DEF_PATH       "C:¥¥EXAMPLE¥¥TARGET"
#define  UNINSTALL_NAME     "ExampleUninstallName"
#define  TITLE              "RegDBSetItem の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RegDBGetItem(HWND);

function ExFn_RegDBGetItem(hMSI)
   STRING svLogFile, svValue, szTitle;
begin

   // ルートキーを設定します。
   RegDBSetDefaultRoot (HKEY_LOCAL_MACHINE);

   // インストール情報とアンインストール情報を設定して
   //RegDBSetItem と RegDBGetItem を呼び出します。
   InstallationInfo (COMPANY_NAME, PRODUCT_NAME, VERSION_NUMBER, PRODUCT_KEY);
   DeinstallStart (APP_DEF_LOG_PATH, svLogFile, DEINST_KEY, 0);

   // レジストリのアプリケーションパスキーの値を
   //szAppPath の値に設定します。
   if (RegDBSetItem (REGDB_APPPATH, APP_PATH) < 0) then
      MessageBox ("アプリケーションパスキーを設定できません。", SEVERE);
   else
```

```
    SprintfBox (INFORMATION, TITLE, "RegDBSetItem はアプリケーション " +
            " パスキーを %s に設定します。", APP_PATH);
endif;

// レジストリのアプリケーションデフォルトのパスキーの値を
//szAppDefPath の値に設定します。
if (RegDBSetItem(REGDB_APPPATH_DEFAULT, APP_DEF_PATH) < 0) then
    MessageBox (" アプリケーションのデフォルトのパスキーを設定できません。", SEVERE);
else
    SprintfBox (INFORMATION, TITLE, "RegDBSetItem はアプリケーション " +
            " デフォルトのパスキーを %s に設定します。", APP_DEF_PATH);
endif;

// レジストリのアンインストール名キーの値を
//szUninstallName の値に設定します。
if (RegDBSetItem (REGDB_UNINSTALL_NAME, UNINSTALL_NAME) < 0) then
    MessageBox (" アンインストール名キーを設定できません。", SEVERE);
else
    SprintfBox (INFORMATION, TITLE, "RegDBSetItem はアンインストール " +
            " 名キーを %s に設定します。", UNINSTALL_NAME);
endif;

//SprintfBox を呼び出すタイトルパラメーターをセットアップします。
szTitle = "RegDBGetItem";

// レジストリのアプリケーションパスキーの値を取得します。
if (RegDBGetItem (REGDB_APPPATH, svValue) < 0) then
    MessageBox (" アプリケーションパスキーの値を取得できません。", SEVERE);
else
    SprintfBox (INFORMATION, TITLE, "RegDBGetItem は " +
            " アプリケーション パス キーの値を取得します : %s.", svValue);
endif;

// レジストリのアプリケーションデフォルトのパスキーの値を
// 取得します。
if (RegDBGetItem (REGDB_APPPATH_DEFAULT, svValue) < 0) then
    MessageBox (" アプリケーションのデフォルトのパスキーを取得できません ", SEVERE);
else
    SprintfBox (INFORMATION, TITLE, "RegDBGetItem は " +
            " アプリケーションのデフォルト パス キーの値を取得します : %s.", svValue);
endif;

// レジストリからアンインストール名キーの値を取得します。
if (RegDBGetItem (REGDB_UNINSTALL_NAME, svValue) < 0) then
    MessageBox (" アプリケーションアンインストール名キーを取得できません。", SEVERE);
else
    SprintfBox (INFORMATION, TITLE, "RegDBGetItem は " +
            " アンインストール名キーの値を取得しました : %s.", svValue);
endif;

end;
```

RegDBGetKeyValueEx

プロジェクト・*InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBGetKeyValueEx 関数は、レジストリ内の指定されたキーの下にある、特定の値名がついた値を読み出します。デフォルトでは、InstallShield はこのキーを HKEY_CLASSES_ROOT のサブキーであるとみなします。**RegDBSetDefaultRoot** を使って別のルートキーを指定できます。

RegDBGetKeyValueEx はレジストリ関連の一般関数で、レジストリ関連の特殊関数が処理するものを含んで、すべてのレジストリキーに利用することが可能です。

メモ・*この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。*

構文

```
RegDBGetKeyValueEx ( szKey, szName, nvType, svValue, nvSize );
```

パラメーター

テーブル 37 ・ RegDBGetKeyValueEx のパラメーター

パラメーター	説明
szKey	値を読み出すキーの名前を指定します。2 つの円記号 (¥¥) を使って、サブキーの異なるレベルを分割してください。
szName	読み出す値の szKey の下にある値名を指定します。キーのデフォルト値を読み出すには、ヌル文字列 (“”) を渡します。
nvType	svValue に戻されるデータタイプを識別する、次の定義済み定数のひとつを戻します。 ・ REGDB_STRING – 文字列変数。newline 文字を利用することはできません。 ・ REGDB_STRING_EXPAND– “%MYPATH%” といった、拡張可能な環境変数式を持つ文字列変数です。 ・ REGDB_STRING_MULTI– 文字列変数。newline 文字を利用することができます。 ・ REGDB_NUMBER– 文字列として処理され、文字列変数で渡される数値。 ・ REGDB_BINARY – 文字列に格納されているバイナリデータ。 ***メモ・** データ型 REGDB_STRING_MULTI を読み出す場合、ヌル文字列 (“”) と共に **StrGetTokens** を使って複数のヌル文字で終わる文字列を文字列のリストへと解析します。つまり、svValue が **RegDBGetKeyValueEx** の呼び出し結果として複数文字列を持つ場合、StrGetTokens(listID, svValue, “”) を利用して文字列を解析し、文字列リスト (listID) に配置することができます。*
svValue	szKey と svName が指定した値を戻します。数値は文字列として戻されることに注意してください。
nvSize	svValue で返された値のサイズをバイトで返します。

戻り値

テーブル 38 ・ RegDBGetKeyValueEx の戻り値

戻り値	説明
0	関数が値を戻したことを示します。
< 0	関数が値を読み出せなかったことを示します。

RegDBGetKeyValueEx の例

***メモ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクショ*

ンを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* RegDBSetKeyValueEx 関数と RegDBGetKeyValueEx 関数のデモンストレーションを
* デモンストレーションを行います。
*
* RegDBCreateKeyEx が呼び出され、HKEY_CLASSES_ROOT キーに
* テスト サブキーを作成します。キーの値は RegDBSetKeyValueEx 関数の
* REGDB_NUMBER オプションを使って整数形式で
* 設定されます。この値が設定されたあと、RegDBGetKeyValueEx 関数を使って
* その値が読み出され、確認されます。
*
¥*---------------------------------------------------------------------*/

#define TITLE "RegDBSetKeyValueEx & RegDBGetKeyValueEx"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RegDBGetKeyValueEx(HWND);

function ExFn_RegDBGetKeyValueEx(hMSI)
  STRING szKey, szNumName, szNumValue, svNumValue, szTitle, szMsg;
  NUMBER nType, nSize, nvType, nvSize;
begin

  // テストするキーを作成します。
  szKey = "TestKey";

  if (RegDBCreateKeyEx (szKey, "") < 0) then
    MessageBox ("RegDBCreateKeyEx が失敗しました。", SEVERE);
    abort;
  endif;

  // RegDBSetKeyValueEx を呼び出すパラメーターをセットアップします。
  szNumName  = "TestValue";
  szNumValue = "12345";
  nType      = REGDB_NUMBER;
  nSize      = -1;

  // これに関連するキー名と値を設定します。
  if (RegDBSetKeyValueEx (szKey, szNumName, nType, szNumValue,
                          nSize) < 0) then
    MessageBox ("RegDBSetKeyValueEx 失敗しました。", SEVERE);
    abort;
  else
    // RegDBSetKeyValueEx の結果を表示します。
    szMsg = "%s は次の通り設定されました : %s";
    SprintfBox (INFORMATION, TITLE, szMsg, szNumName, szNumValue);
  endif;

  // キー値についての情報を読み出します。
  if (RegDBGetKeyValueEx (szKey, szNumName, nvType, svNumValue,
                          nvSize) < 0) then
    MessageBox ("RegDBGetKeyValueEx が失敗しました。", SEVERE);
```

```
    else
        // 戻された値が設定された値と同じかどうか確認します。
        if (nvType != REGDB_NUMBER) then
            MessageBox (" タイプ比較に失敗しました。", SEVERE);
        endif;

        if (svNumValue != szNumValue) then
            MessageBox (" サブキー値の比較に失敗しました。", SEVERE);
        endif;

        // RegDBGetKeyValueEx が読み出した結果を表示します。
        szMsg = "%s は次の値を持ちます : %s¥n¥n このデータは %d バイトです。";
        SprintfBox (INFORMATION, TITLE, szMsg, szNumName, svNumValue, nvSize);
    endif;

    // 作成されたテストキーを削除します。
    if (RegDBDeleteKey (szKey) < 0) then
        MessageBox ("RegDBDeleteKey が失敗しました。", SEVERE);
    endif;

end;
```

RegDBGetUninstCmdLine

プロジェクト・*InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBGetUninstCmdLine　関数は szUninstallKey が指定したアンインストール用の登録済みコマンドラインを取得し、svUninstCmdLine のコマンドラインを戻します。

メモ・*InstallScript エンジンは、現在、レジストリの 64 ビット部分にある [プログラムの追加と削除] 情報の読み書きをサポートしません。そのため、REGDB_OPTIONS システム変数を使った REGDB_OPTION_WOW64_64KEY オプションをこのレジストリ関数で使用することはできません。REGDB_OPTION_WOW64_64KEY オプションを有効にしても、レジストリエントリがこの関数によって作成される場所に影響しません。*

構文

```
RegDBGetUninstCmdLine ( szUninstallKey, svUninstCmdLine );
```

パラメーター

テーブル 39 - RegDBGetUninstCmdLine のパラメーター

パラメーター	説明
szUninstallKey	ターゲットレジストリの < ルート キー >¥Software¥Microsoft¥Windows¥CurrentVersion¥Uninstall の下のサブキーの名前を指定します。関数はまずルートキー HKEY_CURRENT_USER の下にあるサブキーを確認し、そしてサブキーがそこで見つからない場合、HKEY_LOCAL_MACHINE の下を確認します。InstallShield Professional 5.53 またはそれ以降で作成されたセットアプでは、これは一般的にアプリケーションの名前です。InstallShield Professional 6.0 またはそれ以降では、これは括弧 ({}) を含むアプリケーションの製品 GUID です。
svUninstCmdLine	szUninstallKey's の UninstallString 値のデータで指定されたアンインストールコマンドラインを戻します。

戻り値

テーブル 40 - RegDBGetUninstCmdLine の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がコマンドラインを取得しました。
< ISERR_SUCCESS	関数がコマンドラインの取得に失敗しました。

追加情報

デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、"<my registy entry text>") は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。

RegDBKeyExist

プロジェクト - *InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBKeyExist 関数はレジストリの特定のキーの存在を確認します。デフォルトでは、InstallShield はこのキーを HKEY_CLASSES_ROOT のサブキーであるとみなします。別のメインキーを利用する場合、RegDBSetDefaultRoot を使って別のルートキーを指定します。

RegDBKeyExist はレジストリ関連の一般関数で、レジストリ関連の特殊関数が処理するものを含んで、すべてのレジストリキーに利用することが可能です。レジストリ関連の特殊関数については、「レジストリ関連の特殊関数」を参照してください。

メモ・この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。

構文

RegDBKeyExist (szSubKey);

パラメーター

テーブル 41 ・ RegDBKeyExist のパラメーター

パラメーター	説明
szSubKey	検出するキーの名前を指定します。このパラメーターに HKEY_CLASSES_ROOT キー (または指定した別のルート キー) を含まなくても構いません。2 つの円記号 (¥¥) を使って、サブキーの異なるレベルを分割してください。

戻り値

テーブル 42 ・ RegDBKeyExist の戻り値

戻り値	説明
1	関数がレジストリでキー名を検出したことを示します。
< 0	関数がレジストリでキー名を検出することができなかったことを示します。

この関数は決してゼロ (0) を戻しません。

追加情報

デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、"<my registy entry text>") は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。

RegDBKeyExist の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
*
```

```
＊InstallShield スクリプトの例
＊
＊RegDBCreateKeyEx 関数と RegDBKeyExist 関数のデモンストレーションを行います。
＊
＊まず、RegDBCreateKeyEx を呼び出して HKEY_CLASSES_ROOT キーに
＊クラス値を持たないサブキーを作成します。そして、RegDBKeyExist を呼び出し
＊キーが作成されたことを確認します。
＊
＊RegDBCreateKey を再び呼び出し、HKEY_CLASSES_ROOT の下に
＊複数階層のサブキーをその関連値と共に作成します。
＊そして RegDBKeyExist を再び呼び出し、新規キーの存在を
＊確認します。
＊
¥*-------------------------------------------------------------*/

#define TITLE_TEXT "RegDBCreateKeyEx & RegDBKeyExist"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RegDBKeyExist(HWND);

function ExFn_RegDBKeyExist(hMSI)
   STRING szKey, szClass, szKeyRoot, szMsg, svLogFile;
   NUMBER nResult1, nResult2;
begin

   // クラス値を持たないキーを作成します。
   szKey   = "CreateKeyExample";
   szClass = "";

   if (RegDBCreateKeyEx(szKey, szClass) < 0) then
      MessageBox ("RegDBCreateKeyEx への最初の呼び出しに失敗しました。", SEVERE);
      abort;
   else
      SprintfBox (INFORMATION, TITLE_TEXT, "%s が正しく作成されました。", szKey);

      // 作成したばかりのキーが存在するかどうかを確認します。
      if (RegDBKeyExist (szKey) < 0) then
         MessageBox ("RegDBKeyExist への最初の呼び出しに失敗しました。", SEVERE);
      else
         SprintfBox (INFORMATION, TITLE_TEXT, "%s が存在します。.", szKey);
      endif;
   endif;

   if (RegDBDeleteKey (szKey) < 0) then
      MessageBox ("RegDBDeleteKey が失敗しました。", SEVERE);
   endif;

   // 複数のサブ階層やクラス値を持つキーを作成します。
   szKey     = "ShareWare¥¥Games¥¥CoolChess";
   szClass   = "LastPlayed";
   szKeyRoot = "ShareWare";

   if (RegDBCreateKeyEx(szKey, szClass) < 0) then
      MessageBox ("RegDBCreateKeyEx への 2 回目の呼び出しに失敗しました。", SEVERE);
      abort;
   else
      SprintfBox (INFORMATION, TITLE_TEXT, "%s が正しく作成されました。", szKey);
```

```
    // 新しく作成した複数階層キーが存在するかどうかを確認します。
    if (RegDBKeyExist (szKeyRoot) < 0) then
        MessageBox ("RegDBKeyExist への 2 回目の呼び出しに失敗しました。", SEVERE);
    else
        SprintfBox (INFORMATION, TITLE_TEXT, "%s が存在します。.", szKey);
    endif;
  endif;

  if (RegDBDeleteKey (szKey) < 0) then
    MessageBox ("RegDBDeleteKey が失敗しました。", SEVERE);
  endif;

end;
```

RegDBQueryKey

プロジェクト・*InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBQueryKey 関数は、ユーザーによるキーのサブキーと値名の照会を可能にします。この関数を使ってキーを動的に列挙できます。RegDBQueryKey はレジストリ関連の一般関数で、レジストリ関連の特殊関数が処理するものを含んで、すべてのレジストリキーに利用することが可能です。

メモ・*この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。*

構文

RegDBQueryKey (szSubKey, nItem, listResults);

パラメーター

テーブル 43 - RegDBQueryKey のパラメーター

パラメーター	説明
szSubKey	RegDBSetDefaultRoot への前回の呼び出しで設定されたルーとキーの 1 つの下にあるサブキーを指定します。さらに深いレベルのサブキーを指定するには円記号を使用してください。ルートキーを読み出すには、ヌル文字列 (″″) を渡します。
nItem	リストに配置するアイテムを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • REGDB_KEYS – listResults に戻される 文字列リストは、このキーの下にあるすべてのサブキーのリストを含みます。 • REGDB_NAMES – listResults に戻される 文字列リストは、このキー用に名前の付いたすべての値の名前を含みます。
listResults	照会結果を文字列リストに戻します。listResults によって識別されるリストは、ListCreate への呼び出しによって既に初期化されている必要があります。

戻り値

テーブル 44 - RegDBQueryKey の戻り値

戻り値	説明
0	関数が成功したことを示します。
< 0	関数が失敗したことを示します。

追加情報

デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、″<my registy entry text>″) は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。

RegDBQueryKey の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
```

```
＊RegDBQueryKey 関数のデモンストレーションを行います。
＊
＊最初に、RegDBQueryKey を呼び出して、キー KEY1 の下のサブキーを
＊クエリします。＊RegDBQueryKey が戻したリストがダイアログに
＊表示されます。
＊
＊そして、RegDBQueryKey を呼び出して、キー KEY2 の下のサブキーを
＊クエリします。このリストはダイアログにも表示されます。
＊
¥*-------------------------------------------------------------*/

#define KEY1  "SOFTWARE"
#define KEY2  "SOFTWARE¥¥Microsoft"
#define TITLE "RegDBQueryKey の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RegDBQueryKey(HWND);

function ExFn_RegDBQueryKey(hMSI)
   STRING szMsg;
   NUMBER nReturn, nItem;
   LIST   listSubKeys, listNames;
begin

   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   // RegDBQueryKey が戻した値を収めるリストを作成します。
   listSubKeys  = ListCreate(STRINGLIST);
   listNames    = ListCreate(STRINGLIST);

   if ((listNames = LIST_NULL) || (listSubKeys  = LIST_NULL)) then
      MessageBox (" 必要なリストを作成することができませんでした。.", SEVERE);
      abort;
   endif;

   RegDBSetDefaultRoot(HKEY_LOCAL_MACHINE);

   // サブキーのリストを取得します。
   nReturn = RegDBQueryKey(KEY1, REGDB_KEYS, listSubKeys );

   if (nReturn < 0) then
      MessageBox("RegDBQueryKey への最初の呼び出しに失敗しました。", SEVERE);
   else
      szMsg = "Subkeys under " + KEY1 + " キー :";
      SdShowInfoList(TITLE, szMsg, listSubKeys );
   endif;

   // サブキーのリストを取得します。
    nReturn  = RegDBQueryKey(KEY2, REGDB_NAMES, listNames);

   if (nReturn < 0) then
      MessageBox("RegDBQueryKey への 2 回目の呼び出しに失敗しました。", SEVERE);
   else
      szMsg = KEY2 + " キーの下の指定された値 ";
      SdShowInfoList(TITLE, szMsg, listNames);
   endif;
```

```
    // メモリからリストを削除します。
    ListDestroy (listNames);
    ListDestroy (listSubKeys );

end;
```

RegDBQueryKeyCount

***プロジェクト・**InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBQueryKeyCount 関数は、サブキー、または ｓ ｚ Key の下にある値の数を戻します。nItem は サブキーまたは値がカウントされているかどうかを指定します。

***メモ・**この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。*

構文

```
RegDBQueryKeyCount ( szKey, nItem );
```

パラメーター

テーブル 45 · RegDBQueryKeyCount のパラメーター

パラメーター	説明
szKey	RegDBSetDefaultRoot への前回の呼び出しで設定されたルートキーの下にあるキーを指定します。さらに深いレベルのサブキーを指定するには円記号を使用してください。ルートキーを指定するには、ヌル文字列 (“”) を渡します。
nItem	どのアイテムを数えるかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · REGDB_KEYS – サブキーの数が戻されます。 · REGDB_NAMES – 値の数が戻されます。

戻り値

テーブル 46 · RegDBQueryKeyCount の戻り値

戻り値	説明
X	指定したキーの下にある指定した種類のアイテムの数。
< ISERR_SUCCESS	関数がアイテムの数を判断できなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

- **RegDBQueryKeyCount** はレジストリ関連の一般関数で、レジストリ関連の特殊関数が処理するものを含んで、すべてのレジストリ キーに利用することが可能です。レジストリ関連の特殊関数については、「レジストリ関連の特殊関数」を参照してください。
- デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、”<my registy entry text>”) は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。

RegDBQueryStringMultiStringCount

プロジェクト · *InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBQueryStringMultiStringCount 関数は、szKey が指定したキーの下にある szValue が指定した複数文字列値に含まれる文字列の数を戻します。

メモ・この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。

構文

RegDBQueryStringMultiStringCount (szKey, szValue);

パラメーター

テーブル 47 - RegDBQueryStringMultiStringCount のパラメーター

パラメーター	説明
szKey	RegDBSetDefaultRoot への前回の呼び出しで設定されたルートキーの下にあるキーを指定します。さらに深いレベルのサブキーを指定するには円記号を使用してください。ルートキーを指定するには、ヌル文字列 (“”) を渡します。
szValue	確認する szKey の下にある複数文字列値を指定します。

戻り値

テーブル 48 - RegDBQueryStringMultiStringCount の戻り値

戻り値	説明
X	指定したキーの下にある指定された複数文字列値のサブ文字列の数。
< ISERR_SUCCESS	関数がサブ文字列の数を判断できなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

- **RegDBQueryStringMultiStringCount** はレジストリ関連の一般関数で、レジストリ関連の特殊関数が処理するものを含んで、すべてのレジストリキーに利用することが可能です。レジストリ関連の特殊関数については、「レジストリ関連の特殊関数」を参照してください。
- デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、”<my registy entry text>”) は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。

RegDBSetAppInfo

プロジェクト・*InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBSetAppInfo 関数は、レジストリ内のアプリケーション情報キーに属する、特定の値名がついた値を設定します。RegDBSetAppInfo はレジストリ関連の特殊関数で、特定の定義済みレジストリキーと一緒に動作するように設計されています。

メモ・*InstallScript エンジンは、現在、レジストリの 64 ビット部分にある [プログラムの追加と削除] 情報の読み書きをサポートしません。そのため、REGDB_OPTIONS システム変数を使った REGDB_OPTION_WOW64_64KEY オプションをこのレジストリ関数で使用することはできません。REGDB_OPTION_WOW64_64KEY オプションを有効にしても、レジストリエントリがこの関数によって作成される場所に影響しません。*

構文

```
RegDBSetAppInfo ( szName, nType, szValue, nSize );
```

パラメーター

テーブル 49 - RegDBSetAppInfo のパラメーター

パラメーター	説明
szName	情報を設定する値名を指定します。
nType	設定するデータ型を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • **REGDB_STRING** – 文字列変数。newline 文字を利用することはできません。 • **REGDB_STRING_EXPAND** – "%MYPATH%" といった、拡張可能な環境変数式を持つ文字列変数です。 • **REGDB_STRING_MULTI** – 文字列変数。newline 文字を利用することができます。 • **REGDB_NUMBER** – 文字列として処理され、文字列変数で渡される数値。 • **REGDB_BINARY** – 文字列に格納されているバイナリデータ。
szValue	値名に設定する値を指定します。
nSize	RegDBSetAppInfo に渡すデータサイズをバイトで指定します。このパラメーターに -1 を渡して、InstallShield がデータサイズを決定することを示します。

戻り値

テーブル 50 - RegDBSetAppInfo の戻り値

戻り値	説明
0	関数は値名に値を割り当てたことを示します。
< 0	関数が値を割り当てることができなかったことを示します。

追加情報

デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、"<my registy entry text>") は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。

RegDBSetAppInfo の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * RegDBSetAppInfo と RegDBGetAppInfo 関数を示します。
 *
 * これらの関数のいずれかを呼び出す前に、
 * InstallationInfo を呼び出してアプリケーション情報を設定する必要があります。
 *
¥*---------------------------------------------------------------*/

#define COMPANY_NAME "Example_Company"
#define  PRODUCT_NAME          "Example_App"
#define  PRODUCT_VERSION       "5.0"
#define  PRODUCT_KEY           "EXAMPLE.EXE"
#define  DEINSTALL_KEY         "Example_DeinstKey"
#define  UNINSTALL_NAME        "Example_App_5.0"
#define  DEFAULT_LOG_PATH      "EXAMPLE"
#define  TITLE                 "RegDBGetAppInfo"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RegDBSetAppInfo(HWND);

function ExFn_RegDBSetAppInfo(hMSI)
   STRING  szStrName, szStrValue, svStrValue, szTitle, szMsg, svLogFile;
   NUMBER  nvSize, nvType;
begin

   // ルートキーを設定します。
   RegDBSetDefaultRoot (HKEY_LOCAL_MACHINE);

   // 使用する名前を REGDB_STRING で設定します。
   szStrName  = "ExampleStringValue";
   szStrValue = "ExampleStringSetting";

   //RegBDSetAppInfo と
   //RegDBGetAppInfo を使用する前に、アプリケーション情報をセットアップします。
   InstallationInfo (COMPANY_NAME, PRODUCT_NAME, PRODUCT_VERSION, PRODUCT_KEY);
   DeinstallStart(DEFAULT_LOG_PATH, svLogFile, DEINSTALL_KEY, 0);

   //REGDB_STRING 型の値を設定します。
   if (RegDBSetAppInfo (szStrName, REGDB_STRING, szStrValue, -1) < 0) then
      MessageBox ("REGDB_STRING 型のキーと値を設定できませんでした。", SEVERE);
      abort;
   endif;

   //RegDBGetAppInfo を呼び出して値を戻し、すべてのセットアップ
   // パラメーターを比較します。
```

```
    if (RegDBGetAppInfo (szStrName, nvType, svStrValue, nvSize) < 0) then
        MessageBox (" アプリケーション情報値の取得に失敗しました。", SEVERE);
        abort;
    else
        // 取得された値が設定された値と同じか確認します。
        if (nvType != REGDB_STRING) then
            MessageBox (" タイプの比較に失敗しました。", WARNING);
        endif;

        if (szStrValue != svStrValue) then
            MessageBox (" サブキー値の比較に失敗しました。", WARNING);
        else
            szMsg = " 値の設定 : %s = %s¥n¥n 戻り値 : %s = %s";
            SprintfBox (INFORMATION, TITLE, szMsg, szStrName, szStrValue,
                    szStrName, svStrValue);
        endif;

        if (nvSize != StrLength(szStrValue)) then
            MessageBox (" サイズの比較に失敗しました。", WARNING);
        else
            szMsg = " 入力されたバイトのサイズ : %d¥n¥n 戻されたバイトのサイズ : %d";
            SprintfBox (INFORMATION, TITLE, szMsg,
                    nvSize, StrLength(szStrValue) + 1);
        endif;
    endif;

end;
```

RegDBSetDefaultRoot

プロジェクト・*InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBSetDefaultRoot 関数は、レジストリ関連の一般関数が利用するルートキーを設定します。InstallScript レジストリ関数は、デフォルト レジストリ ハイブとして HKEY_CLASSES_ROOT 上で動作します。この関数を使用すると、HKEY_LOCAL_MACHINE 、HKEY_CURRENT_USER、または HKEY_USERS などの他のキーをルートキーとして指定できます。

RegDBSetDefaultRoot を利用して特殊なレジストリ関連関数を使って作成された、または処理されたキーのルートキーを変更することはできません。詳細については、「レジストリ関連の特殊関数」を参照してください。

メモ・*この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。*

構文

```
RegDBSetDefaultRoot ( nRootKey );
```

パラメーター

テーブル 51 ・ RegDBSetDefaultRoot のパラメーター

パラメーター	説明
nRootKey	ルートキー用に次の定数の 1 つを指定します。 ・ HKEY_CLASSES_ROOT ・ HKEY_CURRENT_USER ・ HKEY_LOCAL_MACHINE ・ HKEY_USERS ・ HKEY_CURRENT_CONFIG ・ HKEY_DYN_DATA ・ HKEY_USER_SELECTABLE 引数として HKEY_USER_SELECTABLE を渡すと、関数が呼び出されたときに ALLUSERS システム変数がゼロ以外の場合は後に続くレジストリ関数呼び出しがルートキーとして HKEY_LOCAL_MACHINE を利用します。また関数が呼び出されたときに ALLUSERS が FALSE の場合は後に続くレジストリ関数呼び出しがルートキーとして HKEY_CURRENT_USER を利用します。 ***メモ・** Windows では、HKEY_LOCAL_MACHINE または HKEY_USERS の下に直接キーを作成することはできません。*

戻り値

テーブル 52 ・ RegDBSetDefaultRoot の戻り値

戻り値	説明
0	関数がキーを設定したことを示します。
< 0	関数がキーを設定できなかったことを示します。

RegDBSetDefaultRoot の例

***メモ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* RegDBSetDefaultRoot 関数のデモンストレーションを行います。
```

```
*
* RegDBSetDefaultRoot が呼び出され、RegDBCreateKeyEx 関数の
* デフォルトのルートキーを設定します。
*
* メモ : Windows では、HKEY_LOCAL_MACHINE または HKEY_USERS の下に
*       直接キーを作成することはできません。
*
¥*---------------------------------------------------------------*/

#define TITLE "RegDBSetDefaultRoot の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RegDBSetDefaultRoot(HWND);

function ExFn_RegDBSetDefaultRoot(hMSI)
   STRING szKey, szClass, szMsg, szTitle;
   NUMBER nRootKey;
begin

   // HKEY_CLASSES_ROOT キー ( デフォルト ) にサブキーを作成します 。
   szKey   = " テスト キー ";
   szClass = "";

   if (RegDBCreateKeyEx (szKey, szClass) < 0) then
      MessageBox ("RegDBCreateKeyEx が失敗しました。", SEVERE);
      abort;
   else
      szMsg = "HKEY_CLASSES_ROOT に %s を作成しました。";
      SprintfBox (INFORMATION, TITLE, szMsg, szKey);
   endif;

   // HKEY_LOCAL_MACHINE へのルートキーを設定します。
   nRootKey = HKEY_LOCAL_MACHINE;

   if (RegDBSetDefaultRoot (nRootKey) < 0) then
      MessageBox ("RegDBSetDefaultRoot への最初の呼び出しに失敗しました。", SEVERE);
   else
      MessageBox (" ルートキーが HKEY_LOCAL_MACHINE へ設定されました。",
            INFORMATION);
   endif;

   // HKEY_LOCAL_MACHINE キーにサブキーを作成します。
   if (RegDBCreateKeyEx (szKey, szClass) < 0) then
      MessageBox ("RegDBCreateKeyEx が失敗しました。", SEVERE);
      abort;
   else
      szMsg = "HKEY_LOCAL_MACHINE に %s を作成しました。";
      SprintfBox (INFORMATION, TITLE, szMsg, szKey);
   endif;

   // HKEY_LOCAL_MACHINE のサブキー例を削除します。
   if (RegDBDeleteKey (szKey) < 0) then
      MessageBox ("RegDBDeleteKey が失敗しました。", SEVERE);
   else
      szMsg = "HKEY_LOCAL_MACHINE の %s を削除しました。";
      SprintfBox (INFORMATION, TITLE, szMsg, szKey);
   endif;
```

```
    // ルートキーを再び HKEY_CLASSES_ROOT に設定します。
    nRootKey = HKEY_CLASSES_ROOT;

    if (RegDBSetDefaultRoot (nRootKey) < 0) then
        MessageBox ("RegDBSetDefaultRoot への 2 回目の呼び出しに失敗しました。", SEVERE);
    else
        MessageBox (" ルートキーを HKEY_CLASSES_ROOT に設定しました。",
                INFORMATION);
    endif;

    if (RegDBDeleteKey (szKey) < 0) then
        MessageBox ("RegDBDeleteKey が失敗しました。", SEVERE);
    else
        szMsg = "HKEY_CLASSES_ROOT の %s を削除しました。";
        SprintfBox (INFORMATION, TITLE, szMsg, szKey);
    endif;

end;
```

RegDBSetItem

プロジェクト・*InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBSetItem はレジストリ関連の特殊な関数で、特定の定義済みレジストリキーと共に動作します。**RegDBSetItem** 関数は、nItem の値によってアプリケーションごとのパスキーまたはアプリケーションインストールキーの下に値を割り当てます。REGDB_APPPATH または REGDB_APPPATH_DEFAULT オプションを使って **RegDBSetItem** を呼び出すと、アプリケーションごとのパスキーが作成されます。

メモ・*InstallScript エンジンは、現在、レジストリの 64 ビット部分にある [プログラムの追加と削除] 情報の読み書きをサポートしません。そのため、REGDB_OPTIONS システム変数を使った REGDB_OPTION_WOW64_64KEY オプションをこのレジストリ関数で使用することはできません。REGDB_OPTION_WOW64_64KEY オプションを有効にしても、レジストリエントリがこの関数によって作成される場所に影響しません。*

構文

```
RegDBSetItem (nItem, szValue);
```

パラメーター

テーブル 53 · RegDBSetItem のパラメーター

パラメーター	説明
nItem	設定するアイテムを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • REGDB_APPPATH – アプリケーションごとのパス キーの下の Path 値のデータ。このキーは、CreateInstallationInfo を呼び出した結果としてインストールによって作成されます。**RegDBSetItem** を呼び出す前に、**CreateInstallationInfo** を呼び出してこのキーを作成する必要があります。(イベントベースのスクリプトでは、**CreateInstallationInfo** がデフォルトの **OnMoveData** イベント ハンドラーコードで呼び出されます。) • REGDB_APPPATH_DEFAULT – アプリケーションごとのパス キーの下の DefaultPath 値のデータ。このキーは、**CreateInstallationInfo** を呼び出した結果としてインストールによって作成されます。**RegDBSetItem** を呼び出す前に、**CreateInstallationInfo** を呼び出してこのキーを作成する必要があります。(イベントベースのスクリプトでは、**CreateInstallationInfo** がデフォルトの **OnMoveData** イベント ハンドラーコードで呼び出されます。) • REGDB_UNINSTALL_COMMENTS – アンインストール キーの下の Comments 値のデータ。 • REGDB_UNINSTALL_CONTACT – アンインストール キーの下の Contact 値のデータ。 • REGDB_UNINSTALL_DISPLAY_VERSION – アンインストールキーの下の DisplayVersion 値のデータ。 • REGDB_UNINSTALL_DISPLAYICON – アンインストールキーの下の DisplayIcon 値のデータ。 • REGDB_UNINSTALL_HELPLINK – アンインストール キーの下の HelpLink 値のデータ。 • REGDB_UNINSTALL_HELPTELEPHONE – アンインストール キーの下の HelpTelephone 値のデータ。 • REGDB_UNINSTALL_INSTALLDATE – アンインストール キーの下の InstallDate 値のデータ。 • REGDB_UNINSTALL_INSTALLLOC – アンインストール キーの下の InstallLocation 値のデータ。 • REGDB_UNINSTALL_INSTALLSOURCE – アンインストール キーの下の InstallSource 値のデータ。 • REGDB_UNINSTALL_LANGUAGE – アンインストールキーの下の Language 値のデータ。 • REGDB_UNINSTALL_LOGFILE – アンインストールキーの下の LogFile 値のデータ。 • REGDB_UNINSTALL_MAINT_OPTION – アンインストールキーの下の LogMode 値のデータ。

テーブル 53 · RegDBSetItem のパラメーター (続き)

パラメーター	説明
nItem (続き)	• REGDB_UNINSTALL_MAJOR_VERSION – アンインストールキーの下の VersionMajor 値のデータ。 • REGDB_UNINSTALL_MAJOR_VERSION_OLD – アンインストールキーの下の MajorVersion 値のデータ。 • REGDB_UNINSTALL_MINOR_VERSION – アンインストールキーの下の VersionMinor 値のデータ。 • REGDB_UNINSTALL_MINOR_VERSION_OLD – アンインストールキーの下の MinorVersion 値のデータ。 • REGDB_UNINSTALL_MODIFYPATH – アンインストール キーの下の ModifyPath の値。 • REGDB_UNINSTALL_NAME – アンインストールキーの下の DisplayName 値のデータ。この定数が使用されると、szValue はコントロール パネル内のアンインストール可能なアプリケーションのリストで示されるアプリケーション名を指定します。 InstallScript または InstallScript MSI プロジェクトでは、システム変数 UNINSTALL_DISPLAYNAME を直接設定する方法がより簡単です。 • REGDB_UNINSTALL_NOMODIFY – アンインストール キーの下の NoModif の値。 • REGDB_UNINSTALL_NOREMOVE – アンインストール キーの下の NoRemove の値。 • REGDB_UNINSTALL_NOREPAIR – アンインストールキーの下の NoRepair 値のデータ。 • REGDB_UNINSTALL_PRODUCTGUID – アンインストールキーの下の ProductGuid 値のデータ。 • REGDB_UNINSTALL_PRODUCTID – アンインストールキーの下の ProductId 値のデータ。 • REGDB_UNINSTALL_PUBLISHER – アンインストールキーの下の Publisher 値のデータ。 • REGDB_UNINSTALL_README – アンインストールキーの下の Readme 値のデータ。 • REGDB_UNINSTALL_REGCOMPANY – アンインストールキーの下の RegCompany 値のデータ。 • REGDB_UNINSTALL_REGOWNER – アンインストールキーの下の RegOwner 値のデータ。

テーブル 53 ・ RegDBSetItem のパラメーター (続き)

パラメーター	説明
nItem (続き)	・ REGDB_UNINSTALL_STRING – アンインストール キーの下の UninstallString 値のデータ。UninstallString 値は、MaintenanceStart または DeinstallStart 関数が呼び出されたときに作成されます。 ・ REGDB_UNINSTALL_SYSTEMCOMPONENT – アンインストール キーの下の SystemComponent 値のデータ。 ・ REGDB_UNINSTALL_URLINFOABOUT – アンインストール キーの下の URLInfoAbout 値のデータ。 ・ REGDB_UNINSTALL_URLUPDATEINFO – アンインストール キーの下の URLUpdateInfo 値のデータ。 ・ REGDB_UNINSTALL_VERSION – アンインストールキーの下の Version 値のデータ。この定数が利用されたとき、szValue は例えば "16777216" のような文字列としてインストールされた製品のバージョン番号を指定します。 ***重要 ・ REGDB_WINCURRVER_REGOWNER または REGDB_WINCURRVER_REGORGANIZATION をこの関数と共に使用しないでください。これらの値は Windows によって設定されるため、、インストールで設定することはできません。***
szValue	指定の項目に割り当てる値を指定します。

戻り値

テーブル 54 ・ RegDBSetItem の戻り値

戻り値	説明
0	関数が値を正常に設定したことを示します。
< 0	関数が失敗したことを示します。

追加情報

デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、"<my registy entry text>") は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。

RegDBSetItem の例

メモ ・ 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * RegDBSetItem と RegDBGetItem 関数を示します。
 *
 * このスクリプトによっていくつかのレジストリキーが設定されます。そして
 * それらのキーを取得して現在の値を表示します。
 *
¥*---------------------------------------------------------------------*/

#define  COMPANY_NAME       "ExampleCompany"
#define  PRODUCT_NAME       "ExampleProduct"
#define  VERSION_NUMBER     "5.00.00"
#define  PRODUCT_KEY        "EXAMPLE.EXE"
#define  DEINST_KEY         "ExampleDeinstKey"
#define  APP_DEF_LOG_PATH   "C:¥¥EXAMPLE¥¥TEMP"
#define  APP_PATH           "C:¥¥EXAMPLE"
#define  APP_DEF_PATH       "C:¥¥EXAMPLE¥¥TARGET"
#define  UNINSTALL_NAME     "ExampleUninstallName"
#define TITLE               "RegDBSetItem の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RegDBSetItem(HWND);

function ExFn_RegDBSetItem(hMSI)
   STRING svLogFile;
   STRING svValue, szTitle;
begin

   // ルートキーを設定します。
   RegDBSetDefaultRoot (HKEY_LOCAL_MACHINE);

   // インストール情報とアンインストール情報を設定して
   //RegDBSetItem と RegDBGetItem を呼び出します。
   InstallationInfo (COMPANY_NAME, PRODUCT_NAME, VERSION_NUMBER, PRODUCT_KEY);
   DeinstallStart (APP_DEF_LOG_PATH, svLogFile, DEINST_KEY, 0);

   // レジストリのアプリケーションパスキーの値を
   //szAppPath の値に設定します。
   if (RegDBSetItem (REGDB_APPPATH, APP_PATH) < 0) then
      MessageBox (" アプリケーションパスキーを設定できません。", SEVERE);
   else
      SprintfBox (INFORMATION, TITLE, "RegDBSetItem はアプリケーション " +
            " パスキーを %s に設定します。", APP_PATH);
   endif;

   // レジストリのアプリケーションデフォルトのパスキーの値を
   //szAppDefPath の値に設定します。
   if (RegDBSetItem(REGDB_APPPATH_DEFAULT, APP_DEF_PATH) < 0) then
      MessageBox (" アプリケーションのデフォルトのパスキーを設定できません。", SEVERE);
   else
      SprintfBox (INFORMATION, TITLE, "RegDBSetItem はアプリケーション " +
            " デフォルトのパスキーを %s に設定します。", APP_DEF_PATH);
   endif;

   // レジストリのアンインストール名キーの値を
```

```
    //szUninstallName の値に設定します。
    if (RegDBSetItem (REGDB_UNINSTALL_NAME, UNINSTALL_NAME) < 0) then
        MessageBox (" アンインストール名キーを設定できません。", SEVERE);
    else
        SprintfBox (INFORMATION, TITLE, "RegDBSetItem はアンインストール " +
                " 名キーを %s に設定します。", UNINSTALL_NAME);
    endif;

    //SprintfBox を呼び出すタイトルパラメーターをセットアップします。
    szTitle = "RegDBGetItem";

    // レジストリのアプリケーションパスキーの値を取得します。
    if (RegDBGetItem (REGDB_APPPATH, svValue) < 0) then
        MessageBox (" アプリケーションパスキーの値を取得できません。", SEVERE);
    else
        SprintfBox (INFORMATION, TITLE, "RegDBGetItem は " +
                " アプリケーション パス キーの値を取得します : %s.", svValue);
    endif;

    // レジストリのアプリケーションデフォルトのパスキーの値を
    // 取得します。
    if (RegDBGetItem (REGDB_APPPATH_DEFAULT, svValue) < 0) then
        MessageBox (" アプリケーションのデフォルトのパスキーを取得できません ", SEVERE);
    else
        SprintfBox (INFORMATION, TITLE, "RegDBGetItem は " +
                " アプリケーションのデフォルト パス キーの値を取得します : %s.", svValue);
    endif;

    // レジストリからアンインストール名キーの値を取得します。
    if (RegDBGetItem (REGDB_UNINSTALL_NAME, svValue) < 0) then
        MessageBox (" アプリケーションアンインストール名キーを取得できません。", SEVERE);
    else
        SprintfBox (INFORMATION, TITLE, "RegDBGetItem は " +
                " アンインストール名キーの値を取得しました : %s.", svValue);
    endif;

end;
```

RegDBSetKeyValueEx

プロジェクト・*InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBSetKeyValueEx 関数は、レジストリ内の指定されたキーに属する、特定の値名がついた値を設定します。

値名が既に存在しない場合、**RegDBSetKeyValueEx** が作成します。値データが既に存在する場合、**RegDBSetKeyValueEx** がそれを上書きします。

キーが存在しない場合、**RegDBSetKeyValueEx** が RegDBCreateKeyEx を呼び出して、szKey パラメーターからキーを渡してそれを作成します。

RegDBSetKeyValueEx はレジストリ関連の一般関数で、レジストリ関連の特殊関数が処理するものを含んで、すべてのレジストリ キーに利用することが可能です。

メモ・この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。

構文

RegDBSetKeyValueEx (szKey, szName, nType, szValue, nSize);

パラメーター

テーブル 55 · RegDBSetKeyValueEx のパラメーター

パラメーター	説明
szKey	キーの名前を指定します。2 つの円記号 (¥¥) を使って、サブキーの異なるレベルを分割してください。 ルート キーが HKEY_CLASSES_ROOT の場合、このパラメーターで HKEY_CLASSES_ROOT を指定する必要はありません。特に指定しない限り、InstallScript エンジンは HKEY_CLASSES_ROOT キーの下に設定されている値を使用します。別のルート キーを指定する場合、**RegDBSetKeyValueEx** を呼び出す前に RegDBSetDefaultRoot を呼び出します。
szName	設定する値データの値名を指定します。szKey で指定されたキーのデフォルト値を設定するには、このパラメーターでヌル文字列 (″″) を渡します。
nType	設定するデータの種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · REGDB_STRING – 文字列変数。newline 文字を使用することはできません。 · REGDB_STRING_EXPAND – ″%MYPATH%″ といった、拡張可能な環境変数式を持つ文字列変数。 · REGDB_STRING_MULTI – 文字列変数。newline 文字を利用することができます。 · REGDB_NUMBER – 文字列として処理され、文字列変数で渡される数値。タイプ DWORD のデータを作成します。nType が REGDB_NUMBER の場合、szValue で十進数文字列を渡すと数値がレジストリに格納されます。そしてこの数値は 16 進数として表示され、後に括弧で囲まれた対応する十進数が続きます。szValue の文字列に 16 進数を渡すことはできません。*十進数でなくてはなりません。* · REGDB_BINARY – 文字列に格納されているバイナリデータ。
szValue	値名と関連付ける値を指定します。すべての値は文字列変数として渡さなくてはなりません。数値は文字列として表示しなくてはなりません。InstallScript エンジンが内部で数字に変換します。
nSize	設定するデータのサイズをバイトで指定します。nType が REGDB_STRING、REGDB_STRING_EXPAND、または REGDB_NUMBER の時、このパラメーターに -1 を指定することができ、InstallScript エンジンがサイズを設定します。しかし、REGDB_BINARY や REGDB_STRING_MULTI では、格納するバイナリデータの必ずバイト数を指定しなくてはなりません。

戻り値

テーブル 56 · RegDBSetKeyValueEx の戻り値

戻り値	説明
0	関数がキーを設定したことを示します。

テーブル 56 ・ RegDBSetKeyValueEx の戻り値 (続き)

戻り値	説明
< 0	関数がキーを設定できなかったことを示します。

追加情報

デフォルトで、この関数の文字列引数で山かっこで囲まれているテキスト (例、"<my registy entry text>") は、テキスト置換として解釈され、それに従って処理されます。レジストリ関数の文字列引数のテキスト置換処理を無効にするには、REGISTRYFUNCTIONS_USETEXTSUBS 引数を使って **Disable** を呼び出します。

RegDBSetKeyValueEx の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* RegDBSetKeyValueEx 関数と RegDBGetKeyValueEx 関数のデモンストレーションを
* デモンストレーションを行います。
*
* RegDBCreateKeyEx が呼び出され、HKEY_CLASSES_ROOT キーに
* テスト サブキーを作成します。キーの値は RegDBSetKeyValueEx 関数の
* REGDB_NUMBER オプションを使って整数形式で
* 設定されます。この値が設定されたあと、RegDBGetKeyValueEx 関数を使って
* その値が読み出され、確認されます。
*
¥*---------------------------------------------------------------*/

#define TITLE "RegDBSetKeyValueEx & RegDBGetKeyValueEx"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RegDBSetKeyValueEx(HWND);

function ExFn_RegDBSetKeyValueEx(hMSI)
   STRING szKey, szNumName, szNumValue, svNumValue, szTitle, szMsg;
   NUMBER nType, nSize, nvType, nvSize;
begin

   // テストするキーを作成します。
   szKey = "TestKey";
   if (RegDBCreateKeyEx (szKey, "") < 0) then
      MessageBox ("RegDBCreateKeyEx が失敗しました。", SEVERE);
      abort;
   endif;

   // RegDBSetKeyValueEx を呼び出すパラメーターをセットアップします。
   szNumName  = "TestValue";
```

```
    szNumValue = "12345";
    nType      = REGDB_NUMBER;
    nSize      = -1;

    // これに関連するキー名と値を設定します。
    if (RegDBSetKeyValueEx (szKey, szNumName, nType, szNumValue,
                            nSize) < 0) then
        MessageBox ("RegDBSetKeyValueEx 失敗しました。", SEVERE);
        abort;
    else
        // RegDBSetKeyValueEx の結果を表示します。
        szMsg = "%s は次の通り設定されました : %s";
        SprintfBox (INFORMATION, TITLE, szMsg, szNumName, szNumValue);
    endif;

    // キー値についての情報を読み出します。
    if (RegDBGetKeyValueEx (szKey, szNumName, nvType, svNumValue,
                            nvSize) < 0) then
        MessageBox ("RegDBGetKeyValueEx が失敗しました。", SEVERE);
    else
        // 戻された値が設定された値と同じかどうか確認します。
        if (nvType != REGDB_NUMBER) then
            MessageBox (" タイプ比較に失敗しました。", SEVERE);
        endif;

        if (svNumValue != szNumValue) then
            MessageBox (" サブキー値の比較に失敗しました。", SEVERE);
        endif;

        // RegDBGetKeyValueEx が読み出した結果を表示します。
        szMsg = "%s は次の値を持ちます : %s¥n¥n このデータは %d バイトです。";
        SprintfBox (INFORMATION, TITLE, szMsg, szNumName, svNumValue, nvSize);
    endif;

    // 作成されたテストキーを削除します。
    if (RegDBDeleteKey (szKey) < 0) then
        MessageBox ("RegDBDeleteKey が失敗しました。", SEVERE);
    endif;

end;
```

RegDBSetVersion

プロジェクト・*InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

RegDBSetVersion 関数は、システム変数 IFX_PRODUCT_VERSION の値をアプリケーションのアンインストールレジストリキーの下の Version 値のデータとして配置し、レジストリ値が既に存在しない場合はそれを作成します。IFX_PRODUCT_VERSION が パックされた DWORD 形式 でない場合、関数は失敗します。

メモ・この関数は、REGDB_OPTION_WOW64_64KEY オプションを使用してレジストリの 64 ビット部分をサポートします。詳細については、「REGDB_OPTIONS」を参照してください。

構文

RegDBSetVersion ( );

パラメーター

なし。

戻り値

テーブル 57 - RegDBSetVersion の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がデータをレジストリ値に配置しました。
< ISERR_SUCCESS	関数がデータをレジストリ値に配置することができませんでした。

追加情報

RegDBSetVersion が、OnMoveData イベント ハンドラー関数のデフォルトコードによって呼び出されました。

RegisterFontResource

RegisterFontResource 関数は szFileName が指定するフォントリソースを登録または登録解除します。この関数は OnInstalledFontFile および OnUninstallingFontFile イベントハンドラー関数のデフォルトコードによって呼び出されます。

構文

RegisterFontResource (szFileName, svFontTitle, bRegister, nOptions);

パラメーター

テーブル 58 · RegisterFontResource のパラメーター

パラメーター	説明
szFileName	登録または登録解除するフォントファイル (.fnt, .fon, .fot, .mmm, .otf, .pfb, .pfm, .ttc, or .ttf ファイル) の完全修飾名を指定します。フォントファイルは一般的に FOLDER_FONTS へインストールします。ターゲットシステムに存在するフォントファイルのみを登録することができます。また、フォントファイルをシステムから削除する場合は、先に登録解除しなくてはなりません。
svFontTitle	このパラメーターはリテラル値ではなく変数名でなくてはなりません。RegisterFontResource が呼び出されたときに、その値がヌル文字列 ("") 以外の場合、このパラメーターはフォントのタイトルを指定します。RegisterFontResource が戻るとき、このパラメーターはフォントファイルを登録または登録解除するときに利用されるタイトルを含みます。 登録または登録解除しているフォントファイルが TrueType フォントファイルの場合、このパラメーターに値がヌル文字列の文字列変数を指定することができます。この場合、RegisterFontResource は内部で GetTrueTypeFontFileInfo を呼び出してタイトルを取得しようとします。これに失敗すると、フォントファイルのファイル名がタイトルとして利用されます。 この関数の 4 番目の引数で REGFONT_OPTION_DONTUPDATEREGISTRY が指定されると、svFontTitle は無視されます。
bRegister	フォントリソースを登録するか (TRUE)、フォントリソースの登録を解除するか (FALSE) を指定します。

テーブル 58 · RegisterFontResource のパラメーター (続き)

パラメーター	説明
nOptions	各種オプションを指定します。このパラメーターに次の定数のひとつを渡します、またはこれらの定数を OR 演算子 (|) を利用して組み合わせます。 · REGFONT_OPTION_DEFAULT ― 関数がそのデフォルトの動作を行うことを指定します。 · REGFONT_OPTION_DONTBROADCASTFONTCHANGEMSG ― フォント情報が登録後に変更されたこと、またはフォントの登録が解除されたことを示すメッセージを関数がトップレベル ウィンドウに送らないことを指定します。このフラグは、関数を呼び出した後にインストールがメッセージを送る手段として手動を選択した場合に指定します。手動でメッセージを送るには、次の例のように Windows API 関数 PostMessage を呼び出します。 PostMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0); このフラグは一般的に、複数のフォントをアップデートするために RegisterFontResource を複数回にわたって呼び出してから PostMessage を呼び出す場合に利用します。 · REGFONT_OPTION_DONTUPDATEREGISTRY ― 関数がフォント登録情報をレジストリに追加しない、またはフォント情報をレジストリから削除することを指定します。フォント登録でこのオプションを指定した場合、次回システムが再起動されるときまではフォントを利用することができます。このオプションが指定されている場合、svFontTitle は無視される点にご注意ください。

戻り値

テーブル 59 · RegisterFontResource の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がフォントを登録、または登録解除したことを示します。
< ISERR_SUCCESS	関数がフォントを登録または登録解除できなかったことを示します。

追加情報

· フォント情報は常に HKEY_LOCAL_MACHINE へ書き込まれるため通常、フォントをインストールするにはエンドユーザーがシステム管理者権限を有する必要があります。

- 関数は WM_FONTCHANGE メッセージを送るために SendMessage の代わりに PostMessage を呼び出します。これは SendMessage が (HWND_BROADCAST と共に利用されたときに) 戻る前に開いているすべてのウィンドウがメッセージへ応答するまで待機するためです。このため、システム上で開いているウィンドウがメッセージへの応答に失敗した場合はインストールがフリーズします。関数が PostMessage を呼び出すため、Windows がメッセージを処理し、フォントキャッシュからフォントのリリースを完了する前に関数が戻る場合があります。従って、この関数を使ってフォントをアンインストールする場合、フォントファイルを削除する前に (Delay を呼び出して) 数秒間待機することが推奨されます。
- この関数を機能イベントまたはファイルインストールイベント (たとえば、OnInstallingFile または OnInstalledFile) で呼び出す場合、フォント登録は対応する機能と関連付けられ、フォントはファイルまたは機能がアンインストールされる前に自動的に登録解除されます。

RegisterFontResource の例

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * RegisterFontResource 関数のデモンストレーションを行います。
 *
¥*---------------------------------------------------------------*/

function OnBegin()
    string szFileName, svFontTitle;
begin
    szFileName = FOLDER_FONTS ^ "Estre.ttf";
    svFontTitle = "Estrangelo Edessa";
    RegisterFontResource ( szFileName, svFontTitle, TRUE, REGFONT_OPTION_DEFAULT );
    // このサンプル スクリプトをプロジェクトにカット アンド ペーストして実行すると、
    // 次の行によって、スクリプトの実行が中止されます。
    abort;
end;
```

ReleaseDialog

ReleaseDialog 関数は、szDialogName で識別されたカスタムダイアログと関連付けられたすべてのメモリを開放します。EndDialog を呼び出した後にこの関数を呼び出してください。メッセージハンドル ケースステートメントの外でこの関数を呼び出します。

構文

```
ReleaseDialog (szDialogName);
```

パラメーター

テーブル 60 ・ ReleaseDialog のパラメーター

パラメーター	説明
szDialogName	破棄するダイアログの名前を指定します。

戻り値

テーブル 61 ・ ReleaseDialog の戻り値

戻り値	説明
0	関数が、カスタムダイアログと関連付けられたメモリすべてを開放したことを示します。
DLG_ERR (-1)	関数は失敗しました。ダイアログ名が無効である可能性があります。
DLG_ERR_ENDDLG (-2)	ReleaseDialog は、EndDialog の前に呼び出されました。EndDialog を先に呼び出してダイアログを削除しなくてはなりません。

ReleaseDialog の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* DefineDialog 関数、EndDialog 関数、そして ReleaseDialog 関数の
* デモンストレーションを行います。
*
* このスクリプトは ビットマップを表示するシンプルなカスタムダイアログを
* 開きます。ダイアログは次の 3 つのボタンで閉じることが
* [ 戻る ]、[ 次へ ]、および [ キャンセル ]。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SdAskOptions が表示する
* InstallShield 標準ダイアログです。このダイアログは
* インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
* カスタム ダイアログとしてスクリプトで利用することが
* 表示されます。
*
* このダイアログをカスタムダイアログとして利用するためには、
* DefineDialog を呼び出してそれをスクリプトで定義します。その後
* WaitOnDialog を呼び出してダイアログを表示します。イベントが
```

```
＊ダイアログの処理を終了するとき、それを閉じるために EndDialog が
＊表示されます。次いで、ReleaseDialog への呼び出しによって、
＊メモリからダイアログがリリースされます。
＊
¥*-------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID     12027   // ダイアログ自身の ID
#define RES_PBUT_NEXT         1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL       9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK        12   //[ 戻る ] ボタンの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ReleaseDialog(HWND);

function ExFn_ReleaseDialog(hMSI)
    STRING  szDialogName, szDLLName, szDialog;
    NUMBER  nDialog, nResult, nCmdValue;
    BOOL    bDone;
    HWND    hInstance, hwndParent, hwndDlg;
begin

    // DefineDialog への最初のパラメーターとして渡すダイアログの
    // 名前を定義します。
    szDialogName = "ExampleDialog";

    // DefineDialog の 2 番目のパラメーターは 0 となります。
    // これは .dll ファイルが _isres.dll の中にあるためです。
    hInstance = 0;

    // DefineDialog の 3 番目のパラメーターは ヌルです。インストールは
    // _isuser.dll と _isres.dll にあるダイアログを検索します。
    szDLLName = "";

    // DefineDialog の 5 番目のパラメーターは 0 となります。なぜなら、
    // 4 番目のパラメーターにある ID によってダイアログが認識されるためです。
    szDialog = "";

    // この値は保存され、0 でなくてはなりません。
    hwndParent = 0;

    // ダイアログを定義します。インストールのメインウィンドウがダイアログ ボックスを保有します
    // ( パラメーター 7 内の HWND_INSTALL で表示されます ) を保有します。
    nResult  = DefineDialog (szDialogName, hInstance, szDLLName,
                        RES_DIALOG_ID, szDialog, hwndParent,
                        HWND_INSTALL, DLG_MSG_STANDARD|DLG_CENTERED);

    // エラーをチェックします。
    if (nResult < 0) then
        MessageBox (" ダイアログを定義中にエラーが発生しました。", SEVERE);
        bDone = TRUE;
        abort;
    endif;

   // while ループを制御するのに使われるインジケーターを初期化します。
   bDone = FALSE;
```

```
    // 完了するまでループします。
    repeat

        // ダイアログを表示して次のダイアログ イベントを戻します。
        nCmdValue = WaitOnDialog(szDialogName);

        // イベントに応答します。
        switch (nCmdValue)
            case DLG_CLOSE:
                // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
                Do (EXIT);
            case DLG_ERR:
                MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
                abort;
            case DLG_INIT:
                // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
                // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
                // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
                // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
                hwndDlg = CmdGetHwndDlg (szDialogName);
                SdGeneralInit(szDialogName, hwndDlg, 0, "");
            case RES_PBUT_CANCEL:
                // ユーザーが [ キャンセル ] ボタンをクリックしました。
                Do (EXIT);
            case RES_PBUT_NEXT:
                bDone = TRUE;
            case RES_PBUT_BACK:
                bDone = TRUE;
        endswitch;

    until bDone;

    // ダイアログを閉じます
    EndDialog (szDialogName);

    // メモリからダイアログを解放します。
    ReleaseDialog (szDialogName);

end;
```

RenameFile

RenameFile 関数はファイルまたはディレクトリの名前を変更、および / またはファイルまたはディレクトリ (サブディレクトリおよびファイルを含む) を親ディレクトリから別のディレクトリへ移動します。

メモ・次の事項に注意してください。

- *szSource および szTarget が同じディレクトリを参照する場合、または修飾されていない名前を使用する場合は InstallScript インストールで SRCDIR および TARGETDIR が同じディレクトリを参照する場合、(または基本の MSI または InstallScript MSI インストールで、SRCDIR および INSTALLDIR が同じディレクトリを参照する場合)、そのディレクトリに szTarget が指定する名前のファイルまたはディレクトリが既に存在しない限り、szSource が指定するファイルまたはディレクトリの名前は変更されます。*
- *szSource および szTarget が異なるディレクトリを参照する場合、または修飾されていない名前を使用する場合は InstallScript インストールで SRCDIR および TARGETDIR が異なるディレクトリを参照する場合、(または*

基本の MSI または InstallScript MSI インストールで、SRCDIR および INSTALLDIR が異なるディレクトリを参照する場合)、そのディレクトリに同じ名前のファイルまたはディレクトリが既に存在しない限り、szSource が指定するファイルまたはディレクトリは新しいディレクトリに移動して、szTarget が指定する名前が付けられます。

構文

```
RenameFile ( szSource, szTarget );
```

パラメーター

テーブル 62 ・ RenameFile のパラメーター

パラメーター	説明
szSource	名前を変更、または移動するファイルまたはディレクトリの名前を指定します。szFileOld が完全修飾ファイル名またはディレクトリ (パスを含む) を指定する場合、**RenameFile** は指定されたディレクトリにあるファイルまたはディレクトリの名前変更または移動を行います。szSource に完全修飾のファイル名またはディレクトリ名が含まれていない (つまり、パス情報が無い) 場合、**RenameFile** はシステム変数 SRCDIR が指定するディレクトリにあるファイルまたはディレクトリの名前を変更または移動を行います。 ***メモ・ワイルドカード文字を使用することはできません。RenameFile への書く呼び出しで、ファイルまたはディレクトリを 1 つだけ (サブディレクトリとフォルダーを含む) 名前を変更、または移動させることができます。***
szTarget	ファイルまたはディレクトリの新しい名前および / または配置場所を指定します。szFileNew が完全修飾ファイル名またはディレクトリ名 (パスを含む) を指定する場合、**RenameFile** はファイル名を変更または、指定されたディレクトリへ移動します。szFileNew が修飾されていない (パス情報がない) ファイル名またはディレクトリ名を持つ場合、**RenameFile** はファイルまたはディレクトリの名前を変更するか、システム変数 TARGETDIR (InstallScript インストールの場合) またはシステム変数 INSTALLDIR (基本の MSI または InstallScript MSI インストールの場合) が指定するディレクトリにファイルまたはディレクトリを移動します。

戻り値

テーブル 63 ・ RenameFile の戻り値

戻り値	説明
0	関数がファイルまたはディレクトリの名前を変更、もしくはファイルまたはディレクトリを移動したことを示します。
< 0	関数がファイルまたはフォルダーの名前を変更することができなかった、もしくはファイルまたはフォルダーを移動させることができなかったことを示します。

RenameFile の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* RenameFile 関数のデモンストレーションを行います。
*
* まず FILENAME1 を FILENAME2 へ名前の変更を行うため、RenameFile が呼び出されます。
* そして FILENAME2 ファイルを TARGET ディレクトリへ移動するためにもう一度
* 呼び出されます。
*
* これは RenameFile への 1 度の呼び出しで行うこともできます。RenameFile への
* 3 回目の呼び出しで FILENAME1 を使ってデモンストレーションされていて、
* FILENAME2 ファイルの名前変更し、TARGET ディレクトリから
* SOURCE ディレクトリへ移動させます。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*     ターゲットシステム上の有効なファイル名とパスを指定するように
*     設定してください。
*
¥*---------------------------------------------------------------*/

#define FILENAME1  "ISExampl.txt"
#define FILENAME2  "ISExampl.bak"
#define SOURCE_DIR "C:¥¥ISExampl¥¥Source"
#define TARGET_DIR "C:¥¥ISExampl¥¥Target"
#define TITLE      "RenameFile の例 "


#include "ifx.h"

function OnBegin()
begin

  // 名前変更作業のためのシステム変数を設定します。
  SRCDIR    = SOURCE_DIR;
  TARGETDIR = SOURCE_DIR;

  // FILENAME1 の名前を FILENAME2 へ変更します。
  if (RenameFile (FILENAME1, FILENAME2) < 0) then
    MessageBox("RenameFile への最初の呼び出しに失敗しました。", SEVERE);
    abort;
  else
    szMsg = "%s の名前を %s へ変更することができました。";
    SprintfBox(INFORMATION, szTitle, szMsg, FILENAME1, FILENAME2);
  endif;

  // ファイルをディレクトリ間で移動させるためのシステム変数を設定します。
  SRCDIR    = SOURCE_DIR;
  TARGETDIR = TARGET_DIR;

  // SOURCE ディレクトリから TARGET ディレクトリへファイルを移動します。
  if (RenameFile(FILENAME2, FILENAME2) < 0) then
    MessageBox("RenameFile への 2 回目の呼び出しに失敗しました。", SEVERE);
    abort;
  else
    szMsg = "%s を %s へ移動することができました。";
    SprintfBox(INFORMATION, TITLE, szMsg, FILENAME2, TARGETDIR);
  endif;
```

```
    // ファイルをオリジナルの場所へ移動させるシステム変数を設定します。
    SRCDIR    = TARGET_DIR;
    TARGETDIR = SOURCE_DIR;

    // ファイルの名前を変更し、TARGET ディレクトリから SOURCE ディレクトリへ
    // 移動させます。
    if (RenameFile(FILENAME2, FILENAME1) < 0) then
      MessageBox("RenameFile への 3 回目の呼び出しに失敗しました。", SEVERE);
      abort;
    else
      szMsg = "%s は %s の名前を変更し、ディレクトリ %s へ移動しました。";
      SprintfBox(INFORMATION, TITLE, szMsg, FILENAME2, FILENAME1, TARGETDIR);
    endif;

end;
```

ReplaceFolderIcon

ReplaceShortcut 関数は **ReplaceFolderIcon** 関数に優先します。

ReplaceFolderIcon 関数は、特定のフォルダーのショートカットを置換します。CreateProgramFolder 関数を使って作成したフォルダー、またはユーザーのシステムに既に存在するフォルダーを指定しなくてはなりません。

構文

```
ReplaceFolderIcon ( szProgramFolder, szItemName, szNewItem, szCmdLine, szWorkingDir, szIconPath, nIcon, szShortCutKey, nFlag );
```

パラメーター

テーブル 64 • ReplaceFolderIcon のパラメーター

パラメーター	説明
szProgramFolder	置換するショートカットを含むフォルダーの名前を指定します。
szItemName	置換するショートカットの名前を指定します。
szNewItem	置換した後に表示されるショートカットの名前を指定します。
szCmdLine	次のうちの 1 つを指定します : • コマンドラインパラメーターをすべて含む、アイコンに関連する実行可能ファイルの完全修飾名。 • szItemName がサブフォルダーの場合、完全修飾パス。
szWorkingDir	アプリケーションのプログラムファイルが保管されているディレクトリを指定します。(szItemName がサブフォルダーの場合を除きます)。このパラメーターにヌル文字列 ("") を渡すと、プログラムファイルが含まれるディレクトリが作業ディレクトリになります。
szIconPath	アイコンファイル、または新しいアイコンを含む有効な Windows 実行可能ファイルの名前を指定します。
nIcon	szIconPath に実行可能ファイルを指定する場合、実行可能ファイル内のアイコン インデックスを指定します。そうでない場合は、nIcon に数字 0 を入力します。
szShortCutKey	ユーザーが押すとプログラムがスタートするショートカットキー シーケンスを含む文字列を指定します。たとえば、ユーザーが Ctrl と Alt キーを押しながら 1 を押すとアプリケーションが開くようにする場合は、このパラメーターに "Ctrl+Alt+1" を指定します。
nFlag	1 つ以上のオプションを指定します。このパラメーターに、あらかじめ定義されている次の定数を渡します : 複数のオプションを指定するには、定数と OR (|) 演算子を組み合わせます。 • NULL – オプションがないことを示します。 • REPLACE – 既存アイコンを新しいアイコンと置き換えることを示します。 • RUN_MAXIMIZED – プログラムの起動時に最大化されることを示します。 • RUN_MINIMIZED – プログラムの起動時に最小化されることを示します。

戻り値

テーブル 65 - ReplaceFolderIcon の戻り値

戻り値	説明
0	関数がショートカットを置換したことを示します。
< 0	関数がアイコンを置き換えられなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

ReplaceFolderIcon の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ReplaceFolderIcon 関数のデモンストレーションを行います。
*
* メモ : このスクリプトを正しく実行するため、
*     プリプロセッサ定数をターゲットシステムの有効なファイル名とパスに
*     設定してください。この例に使用するショートカットを簡単に作成するには、
*     AddFolderIcon の例 #3 を実行してください。
*
¥*---------------------------------------------------------------*/

#define FOLDER     "C:¥¥Windows¥¥"
#define NEW_PROGRAM "C:¥¥WINDOWS¥¥WRITE.EXE"
#define NEW_PARAM   "C:¥¥WINDOWS¥¥README.TXT"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_ReplaceFolderIcon(HWND);

function ExFn_ReplaceFolderIcon(hMSI)
   STRING szProgramFolder, szItemName, szNewItem, szCmdLine, szWorkingDir;
   STRING szShortCutKey, szIconPath, szProgram, szParam;
   NUMBER nIcon, nFlag;
begin

   szProgramFolder = FOLDER ^ " 例のフォルダー ";
   szItemName      = " メモ帳の例 ";
   szNewItem       = " 新規 Wordpad の例 ";

   // 空白スペースが区切り文字として間違えられないよう注意してください。
   szProgram = NEW_PROGRAM;
```

```
    LongPathToQuote (szProgram, TRUE);

    szParam = NEW_PARAM;
    LongPathToShortPath(szParam);

    szCmdLine      = szProgram + " " + szParam;
    szWorkingDir   = "";
    szIconPath     = "";
    nIcon          = 0;
    szShortCutKey  = "";
    nFlag          = REPLACE|RUN_MAXIMIZED;

    // フォルダーをスクリーンに表示します。
    ShowProgramFolder (szProgramFolder, SW_SHOW);

    // "Notepad の例 " アイコンを " 新規 Wordpad の例 " と置き換えます。
    if (ReplaceFolderIcon (szProgramFolder, szItemName, szNewItem, szCmdLine,
                           szWorkingDir, szIconPath, nIcon, szShortCutKey,
                           nFlag) < 0) then
        MessageBox ("ReplaceFolderIcon が失敗しました。", SEVERE);
    else
        MessageBox (" アイコンが置換されました。", INFORMATION);
    endif;

end;
```

ReplaceProfString

Windows Installer ベースのプロジェクトでは、すべての .ini ファイルは IDE の [INI ファイル] ビューで作成します。この方法ですべての .ini ファイルの変更を処理すると、Windows Installer サービスを通したクリーンアンインストールが可能となります。

ReplaceProfString 関数は、.ini ファイルのプロファイル文字列を置換します。この関数は、System.ini ファイル (device = ...) の [386Enh] セクションにあるような重複キー (非固有キー) の置換にも使用できます。関数は szKeyName = szOrigValue を検索し、行を置き換えます。検出されなかった場合、szSectionName セクションの始まりに szKeyName = szReplaceValue 行を追加します。

固有のキー (指定のセクションですべて違うキーなど) を追加する場合は、WriteProfString 関数を使用します。

この関数は、System.ini ファイルの device=line などの非固有キー名の置換にのみ使用してください。

構文

```
ReplaceProfString (szFileName, szSectionName, szKeyName, szOrigValue, szReplaceValue);
```

パラメーター

テーブル 66 - ReplaceProfString のパラメーター

パラメーター	説明
szFileName	プロファイル文字列を置換する .ini ファイル名を指定します。ファイル名が非完全修飾名 (ドライブ指定とパスが*含まれていない*) の場合、InstallShield は Windows フォルダーのファイルを検索します。ファイルが存在しないと、指定のフォルダーに作成されます。パスがファイル名に含まれていない場合、ファイルは Windows フォルダーに作成されます。ファイル名が、存在しないパス名で修飾されていると、ReplaceProfString は失敗します。
szSectionName	szKeyName を検索する、.ini ファイルセクション名を指定します。ここで指定するセクション名は、角括弧 ([ ]) で*囲まない*でください。この名前の検索は大文字と小文字を*区別しません*。
szKeyName	置換するキーの名前を指定します。キーが存在しない場合は、作成されます。
szOrigValue	szKeyName で指定したキーの現在の値を指定します。
szReplaceValue	szKeyName に割り当てる新しい値を指定します。このパラメーターの値は、プロファイル文字列の等号の右側 (szKeyName = szValue) に表示されます。

戻り値

テーブル 67 - ReplaceProfString の戻り値

戻り値	説明
0	関数がプロファイル文字列を置換、または追加したことを示します。
< 0	関数がプロファイル文字列を置換、または追加することができなかったことを示します。

追加情報

.ini ファイルに加えた変更は、アンインストール用にログ記録することができます。ただし、いくつかの重要な制限事項があります。詳細は、「 初期設定 (.ini) ファイルエントリのアンインストール」を参照してください。

ReplaceProfString の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ReplaceProfString 関数のデモンストレーションを行います。
*
* ReplaceProfString への最初の呼び出しでは、szKeyName キー値
* szOrigValue を 値 szReplaceValue と
置き換えます。そして、ReplaceProfString が再び呼び出され、
* 新しく設定された szKeyName、szReplaceValue の値を
* szOrigValue と置き換えます。
*
* メモ : このスクリプトを適切に実行するため、
*       定数 EXAMPLE_INI がターゲットシステム上の初期化ファイルを
*      参照するように設定してください。そのファイルには
*      次の行を含みます :
*
*      [Old Section]
*      Old Key=Old value
*
¥*---------------------------------------------------------------*/

#define EXAMPLE_INI "C:¥¥ISExampl.ini"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_ReplaceProfString(HWND);

function ExFn_ReplaceProfString(hMSI)
   STRING szSectionName, szKeyName, szOrigValue, szReplaceValue;
begin

   szSectionName  = " 古いセクション ";
   szKeyName      = " 古いキー ";
   szOrigValue    = " 古い値 ";
   szReplaceValue = " 新しい値 ";

   // szOrigValue を szReplaceValue と入れ替えます。
   if (ReplaceProfString (EXAMPLE_INI, szSectionName, szKeyName, szOrigValue,
                          szReplaceValue) < 0) then
      MessageBox("ReplaceProfString が失敗しました。", SEVERE);
      abort;
   else
      SprintfBox (INFORMATION, " 置換に成功しました。",
                  " 元の値 :%s¥n 新しい値 : %s", szOrigValue, szReplaceValue);
   endif;

   // szReplaceValue を szOrigValue と入れ替えます。
   if (ReplaceProfString(EXAMPLE_INI, szSectionName, szKeyName, szReplaceValue,
```

```
                    szOrigValue) < 0) then;
        MessageBox("ReplaceProfString が失敗しました。", SEVERE);
    else
        SprintfBox (INFORMATION, " 置換に成功しました。",
                " 元の値 :%s¥n 新しい値 : %s", szReplaceValue, szOrigValue);
    endif;

end;
```

ReplaceShortcut

ReplaceShortcut 関数は、特定のフォルダーのショートカットを置換します。CreateShortcutFolder 関数を使って作成したフォルダー、またはエンド ユーザーのシステムに既に存在するフォルダーを指定しなくてはなりません。

構文

```
ReplaceShortcut (szShortcutFolder, szName, szNewItem, szCmdLine, szWorkingDir, szIconPath, nIcon, szShortCutKey, nFlag);
```

パラメーター

テーブル 68 • ReplaceShortcut のパラメーター

パラメーター	説明
szShortcutFolder	置換するショートカットを含むフォルダーの名前を指定します。
szName	置換するショートカットの名前を指定します。
szNewItem	置換した後に表示されるショートカットの名前を指定します。
szCmdLine	次のうちの 1 つを指定します : • コマンドライン パラメーターをすべて含む、アイコンに関連する実行可能ファイルの完全修飾名。 • szName がサブフォルダーの場合、完全修飾パス。
szWorkingDir	szName がサブフォルダーでない場合、製品のプログラム ファイルを含むディレクトリを指定します。 このパラメーターにヌル文字列 ("") を渡すと、製品のプログラムファイルが含まれるディレクトリが作業ディレクトリになります。
szIconPath	代替アイコン ファイル、または新しいアイコンを含む有効な実行可能ファイルの名前を指定します。
nIcon	szIconPath に実行可能ファイルを指定する場合、実行可能ファイル内のアイコン インデックスを指定します。そうでない場合は、nIcon に数字 0 を入力します。
szShortCutKey	エンド ユーザーが押すとプログラムがスタートするショートカットキーシーケンスを含む文字列を指定します。たとえば、エンド ユーザーが Ctrl と Alt キーを押しながら 1 を押すとアプリケーションが開くようにする場合は、このパラメーターに "Ctrl+Alt+1" を指定します。

テーブル 68 • ReplaceShortcut のパラメーター (続き)

パラメーター	説明
nFlag	このパラメーターには、次の定義済み定数のいずれかを指定します。このパラメーターに複数の定義済み定数を渡す場合、各定数をビット単位 OR 演算子 (|) で区切ってください。 • CS_OPTION_FLAG_REPLACE_EXISTING – 既存のショートカットを置換します。 • CS_OPTION_FLAG_RUN_MAXIMIZED – 起動時にショートカットのターゲットが最大化されます。 • CS_OPTION_FLAG_RUN_MINIMIZED – 起動時にショートカットのターゲットが最小化されます。 • CS_OPTION_FLAG_PREVENT_PINNING – Windows Vista 以降のシステムで、[スタート] メニューまたはタスクバーにショートカットをピン留めすることを防ぎます。このオプションは、エンド ユーザーがタスクバーおよび [スタート] メニューにショートカットをピン留めするためのコンテキスト メニュー コマンドを隠します。 インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを防ぎたい場合があります。 • CS_OPTION_FLAG_NO_NEW_INSTALL_HIGHLIGHT – エンド ユーザーが製品を Windows Visata 以降のシステム上にインストールした後、ショートカットを新しくインストールされたプログラムとして強調表示しません。これは、ターゲット システム上で個別のアイテムに対して [[スタート] メニューのカスタマイズ] ダイアログ ボックスで **[新しくインストールされたプログラムを強調表示する]** チェック ボックスをクリアした場合と同じ効果を持ちます。 インストールの一部であるツールまたは従属的な製品のショートカットのオプションを使用したい場合があります。 • CS_OPTION_FLAG_NO_STARTSCREEN_PIN – Windows 8 ターゲット システム上で、デフォルトで [スタ — ト] 画面にショートカットをピン留めしません。この定数を渡すと、インストールは Windows 8 で使用可能になった Windows シェル プロパティを設定します。 インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを防ぎたい場合があります。 • NULL – オプションがないことを示します。 CS_OPTION_FLAG_PREVENT_PINNING および CS_OPTION_FLAG_NO_STARTSCREEN_PIN についての詳細は、追加情報セクションを参照してください。

戻り値

テーブル 69 - ReplaceShortcut の戻り値

戻り値	説明
0	関数がショートカットを置換したことを示します。
<0	関数がショートカットを置換できなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

Additional Information

2 つの nFlag 定数について、以下にご注意ください。

CS_OPTION_FLAG_PREVENT_PINNING

ピン留めを行わないようにショートカットを構成した場合、[スタート] メニューの最もよく使われている製品のリストに、ショートカットのターゲットを含められなくなります。

特定の文字列を含むショートカットは、タスクバーまたは [スタート] メニューにピン留めすることができません。また、それらを最もよく使う製品リストに表示することもできません。例 :

- Documentation
- ヘルプ
- Install
- 削除
- Setup
- Support

CS_OPTION_FLAG_NO_STARTSCREEN_PIN

Windows 8 は、アプリケーションのアンインストールによってショートカットが削除された後でも、ショートカットの [スタート] 画面へのピン留めに関する情報を保持します。そのため、ショートカットがインストール済みの場合、ターゲット システム上で CS_OPTION_FLAG_NO_STARTSCREEN_PIN 定数は効果を持ちません。この機能をテストする際、ショートカットとそのターゲットが既にインストールされていない、クリーン マシン上でテストするようにして下さい。

ReplaceShortcut の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * ReplaceShortcut 関数のデモンストレーションを行います。
 *
 * メモ : このスクリプトを正しく実行するため、
 *     プリプロセッサ定数をターゲットシステムの有効なファイル名とパスに
 *     設定してください。この例に使用するショートカットを簡単に作成するには、
 *     CreateShortcut の例 #3 を実行してください。
 *
¥*---------------------------------------------------------------------*/

#define FOLDER      "C:¥¥Windows¥¥"
#define NEW_PROGRAM "C:¥¥WINDOWS¥¥WRITE.EXE"
#define NEW_PARAM   "C:¥¥WINDOWS¥¥README.TXT"

function OnFirstUIAfter( )
    STRING szShortcutFolder, szName, szNewItem, szCmdLine, szWorkingDir;
    STRING szIconPath, szShortCutKey, szProgram, szParam;
    NUMBER nIcon, nFlag;
begin

    szShortcutFolder = FOLDER ^ " 例のフォルダー 3";
    szName           = " メモ帳の例 3";
    szNewItem        = " 新規 Wordpad の例 ";

    // 空白スペースが区切り文字として間違えられないよう注意してください。
    szProgram = NEW_PROGRAM;
    LongPathToQuote (szProgram, TRUE);

    szParam = NEW_PARAM;
    LongPathToShortPath(szParam);

    szCmdLine      = szProgram + " " + szParam;
    szWorkingDir   = "";
    szIconPath     = "";
    nIcon          = 0;
    szShortCutKey  = "";
    nFlag          = CS_OPTION_FLAG_REPLACE_EXISTING|CS_OPTION_FLAG_RUN_MAXIMIZED;

    // フォルダーをスクリーンに表示します。
    ShowProgramFolder (szShortcutFolder, SW_SHOW);

    // "Notepad の例 3" ショートカット " 新規 Wordpad の例 " と置き換えます。
    if (ReplaceShortcut (szShortcutFolder, szName, szNewItem, szCmdLine,
                         szWorkingDir, szIconPath, nIcon, szShortCutKey,
                         nFlag) < 0) then
        MessageBox ("ReplaceShortcut が失敗しました。", SEVERE);
    else
        MessageBox (" ショートカットが正しく置換されました。", INFORMATION);
    endif;

end;
```

Resize

Resize 関数は、InstallScript 配列のサイズを変更します。

構文

Resize (Array , nNewSize);

パラメーター

テーブル 70 - Resize のパラメーター

パラメーター	説明
配列	配列変数の名前を指定します。
nNewSize	配列に適用する新しいサイズを指定します。

戻り値

Resize 関数は、配列の新しいサイズを戻します。

RGB

RGB 関数は、SetColor と SetTitle と共に利用できるカスタムカラー値を作成します。

構文

RGB (constRed, constGreen, constBlue);

パラメーター

テーブル 71 - RGB のパラメーター

パラメーター	説明
constRed	カスタムカラーの赤の量を示す、0 から 255 までの範囲の数値定数を指定します。
constGreen	カスタムカラーの緑の量を示す、0 から 255 までの範囲の数値定数を指定します。
constBlue	カスタムカラーの緑の量を示す、0 から 255 までの範囲の数値定数を指定します。

戻り値

この関数は、SetColor と SetTitle を呼び出す際に利用できるカスタムカラーの数値を戻します。

RGB の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * RGB 関数のデモンストレーションを行います。
 *
 * RGB への最初の呼び出しは、背景色の値をグレーに
 * 戻します。2 番目呼び出しは、背景色赤に戻します。
 * 3 番目の呼び出しは、背景色青に戻します。
 * RGB への呼び出しで戻された値は SetColor へ渡され、
 * 背景色を変更します。
 *
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_RGB(HWND);

function ExFn_RGB(hMSI)
begin

   Enable ( BACKGROUND );

   // 背景色をライトグレーに変更します。
   SetColor (BACKGROUND, RGB(198,198,198));

   Delay (3);

   // 背景色を赤に変更します。
   SetColor (BACKGROUND, RGB(255,0,0));

   Delay (3);

   // 背景色を青に変更します。
   SetColor (BACKGROUND, RGB(0, 0, 255));

   Delay (3);

end;
```

ビルトイン関数 (S-T)

カテゴリ別の関数一覧は、「カテゴリー別ビルトイン関数」を参照してください。

SdAskDestPath

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

SdAskDestPath 関数は、エンドユーザーが別のインストール先パスを選択するために使用するダイアログを作成します。エンド ユーザーがこのダイアログの [参照] ボタンをクリックすると SelectDir 関数が呼び出され、エンド ユーザーが既存のフォルダーを選択したり新規のフォルダー名を入力することができる第 2 のダイアログ ボックスが開きます。

構文

SdAskDestPath (szTitle, szMsg, svDir, nReserved);

パラメーター

テーブル 1 · SdAskDestPath のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (「フォルダーの選択」) を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するテキストを指定します。このテキストは静的コントロールと認識されます。メッセージ文字列の %P プレースホルダーを使用して、以前に SdProductName を呼び出した際に指定された製品名を挿入します。このダイアログのデフォルトの指示を表示するには、ヌル文字列 ("") を渡します。
svDir	デフォルトによって選択されるディレクトリの名前を指定します。エンドユーザーによって選択されたディレクトリの名前を返します。
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。

戻り値

テーブル 2 · SdAskDestPath の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。
- (エンドユーザーが 2 番目のダイアログに不完全、無効、または書き込み禁止のパスを指定した場合、エラーが表示されます。エンドユーザーが書き込み不可能なファイルを選択できるようにするには、AskPath 関数を代わりに呼び出します。
- SdAskDestPath を呼び出す前に新規フォルダーが存在しない場合、サイレントモードで実行するセットアップは新規フォルダーを作成します。これによって、確認ダイアログは表示されません。このステップを踏まない場合は、2 つの条件を処理するために 2 つの応答ファイルが必要です。
- InstallShield Professional の以前のバージョンでは、エンドユーザーが [フォルダーの選択] ダイアログで存在しないフォルダーを選択すると、フォルダーを作成するかどうかを問い合わせる確認メッセージが表示されました。このメッセージは多くのエンドユーザーに混乱を招くため、InstallShield では削除されました。

SdAskDestPath の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdAskDestPath 関数のデモンストレーションを行います。
 *
 * ユーザーへパスを要求するため、SdAskDestPath が呼び出されます。
 * そして選択したパスがシステム変数 INSTALLDIR に割り当てられ、
 * メッセージ ボックスに表示されます。
 *
¥*---------------------------------------------------------------------*/

#define  TITLE_TEXT "SdAskDestPath の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

function OnBegin()
    STRING svDir;
begin

    // セットアップダイアログで [ 戻る ] ボタンを無効にします。
    Disable(BACKBUTTON);

    // SdAskDestPath への呼び出し用にデフォルト フォルダー名を設定します。
    svDir = "C:¥¥ISExampl¥¥Target";

    // SdAskDestPath ダイアログを表示します。既定メッセージを
    // 表示するため、2 番目のパラメーターにヌル文字列を渡します。
    if (SdAskDestPath (TITLE_TEXT, "", svDir, 0) = NEXT) then
        INSTALLDIR = svDir;
    endif;

    // 新しいターゲットディレクトリを表示します。
    SprintfBox (INFORMATION, "SdAskDestPath", " 成功しました。¥n¥n ターゲット " +
            " ディレクトリ : " + INSTALLDIR);

end;
```

SdAskDestPath2

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdAskDestPath 関数は、エンドユーザーが別のインストール先パスを選択するために使用するダイアログを作成します。このダイアログ ボックスの [変更] ボタンをクリックすると、SelectDir 関数が呼び出され、エンドユーザーが既存のフォルダーを選択したり新規のフォルダー名を入力することができる第 2 ダイアログ ボックスが開きます。

構文

SdAskDestPath2 (szTitle, szMsg, svDir);

パラメーター

テーブル 3 · SdAskDestPath2 のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (「インストール先の選択」) を表示するには、このパラメーターにヌル文字列 (“”) を渡します。
szMsg	ダイアログに表示するテキストを指定します。このテキストは静的コントロールと認識されます。メッセージ文字列の %P プレースホルダーを使用して、以前に SdProductName を呼び出した際に指定された製品名を挿入します。このダイアログのデフォルトの指示を表示するには、ヌル文字列 (“”) を渡します。
svDir	デフォルトによって選択されるディレクトリの名前を指定します。エンドユーザーによって選択されたディレクトリの名前を返します。

戻り値

テーブル 4 · SdAskDestPath2 の戻り値

戻り値	説明
NEXT	ユーザーが、[次へ] ボタンを選択したことを示します。
BACK	ユーザーが、[戻る] ボタンを選択したことを示します。
< ISERR_SUCCESS	ダイアログが表示されなかったことを示します。

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。
- (エンドユーザーが 2 番目のダイアログに不完全、無効、または書き込み禁止のパスを指定した場合、エラーが表示されます。エンドユーザーが書き込み不可能なファイルを選択できるようにするには、AskPath 関数を代わりに呼び出します。
- SdAskDestPath2 を呼び出す前に新規フォルダーが存在しない場合、サイレントモードで実行するインストールは新規フォルダーを作成します。これによって、確認ダイアログは表示されません。このステップを踏まない場合は、2 つの条件を処理するために 2 つの応答ファイルが必要です。

SdAskDestPath2 の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdAskDestPath2 関数のデモンストレーションを行います。
 *
 * SdProductName が呼び出されて、製品名が SdAskDestPath2
 * ダイアログ ボックスの %P プレースホルダーの位置に
 * 表示されるよう、製品名を設定します。そして、SdAskDestPath2 が
 * 呼び出されて、ユーザーへパスを問い合わせます。最後に、選択したパスが
 * システム変数 TARGETDIR に割り当てられ、これが
 * メッセージボックスに表示されます。
 *
¥*---------------------------------------------------------------------*/

#define  TITLE_TEXT "SdAskDestPath2 の例 "

   STRING svDir;

#include "ifx.h"

function OnBegin()
begin

   // 製品名を設定します。
   SdProductName (" 製品 5.2 の例 ");

   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   // SdAskDestPath2 への呼び出し用デフォルト フォルダー名を設定します。
   svDir = "C:¥¥ISExampl¥¥Target";

   // SdAskDestPath2 ダイアログを表示します。既定メッセージを
   // 表示するため、2 番目のパラメーターにヌル文字列を渡します。
   if (SdAskDestPath2 (TITLE_TEXT, "", svDir ) = NEXT) then
      TARGETDIR = svDir;
   endif;

   // 新しいターゲットディレクトリを表示します。
   SprintfBox (INFORMATION, "SdAskDestPath2", " 成功しました。¥n¥n ターゲット " +
            " ディレクトリ : " + TARGETDIR);

end;
```

SdAskOptions

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdAskOptions 関数は、インストールのオプションを提供するダイアログを作成します。選択方法としてにはチェック ボックスまたはオプションボタンを使用できます。ボタンの横に表示される情報は、オプションのグループから取得されます。デフォルトのオプション数は 4 です。必要に応じて、複数のオプションをグループで追加したり削除できます。

構文

SdAskOptions (szTitle, szMsg1, szMsg2, szId, szFeatures, nExclusiveFlag);

パラメーター

テーブル 5 · SdAskOptions のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「コンポーネントの選択」を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg1	ダイアログに表示するメッセージを指定します。このスタティックフィールドには、801 という ID があります。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg2	ダイアログに表示する 2 番目のメッセージを指定します。このスタティックフィールドには、802 という ID があります。
szId	代替の数値ダイアログ ID を指定します。ID には、文字列形式の数値を使用します (たとえば、ID 13001 は「13001」と指定します)。SdAskOptions ダイアログリソースをコピーし、変更を加えて固有な数値 ID を指定してから、szId の文字列としてその ID を渡してダイアログを呼び出すこともできます。(上記参照) オプションが 4 つある標準の SdAskOptions ダイアログを作成するには、このパラメーターでヌル文字列 ("") を渡します。
szFeatures	表示するサブ機能を含む機能の名前を指定します。サブ機能は、チェック ボックスまたはオプションボタンよりも優先されます。トップレベルの機能をすべて表示するには、このパラメーターでヌル文字列 ("") を渡します。 SdAskOptions は、ファイルメディアライブラリ、またはシステム変数 MEDIA が指定するスクリプト作成の機能セット内で、要求された機能を検索します。
nExclusiveFlag	ダイアログに表示するボタンの種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · EXCLUSIVE Ñ オプション ボタンを指定します。 · NONEXCLUSIVE Ñ チェック ボックスを指定します。

戻り値

テーブル 6 · SdAskOptions の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。

追加情報

· インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

- 必須の可視コンポーネントがセットアップに含まれている場合、インストールオプションを取得するために SdAskOptions を呼び出さないでください。代わりに、FeatureDialog、SdFeatureDialog、SdFeatureDialogAdv、SdFeatureMult、または SdAskOptionsList 非限定モードで呼び出します。
- セットアップでセットアップの種類のダイアログを使用しない場合は、SdAskOptions を呼び出す前に FeatureSetupTypeSet を必ず呼び出して、IDE の [セットアップの種類] ビューで定義されているセットアップの種類を指定してください。
- SdAskOptions は、MEDIA システム変数によって指定される現在のメディアで実行されます。セットアップ初期設定中、インストールはファイル メディア ライブラリ (**Data1.cab**) に関連付けられたメディア名を MEDIA に割り当てます。

タスク ***スクリプト作成の機能を表示するには:***

1. 現在の MEDIA 値を文字列変数の形式で保存します。たとえば、szSaveMEDIAValue。
2. スクリプト作成コンポーネントセットの名前を MEDIA に割り当てます。
3. SdAskOptions を呼び出して、エンドユーザーの選択を取得します。
4. ステップ 1 で保存した値を MEDIA に割り当てます。必ず FeatureTransferData を呼び出す前に行います。

   リソース エディターを使用して (_isres.dll にある) SdAskOptions ダイアログ リソースをコピーし、コピーに変更を加えて固有な ID を指定することにより、SdAskOptions タイプのダイアログを複数作成することができます。コピーは _isuser.dll に保存してください。SdAskOptions を呼び出して、ダイアログのカスタマイズされたコピーの ID をパラメーター szId で渡す時、カスタマイズされたコピーが表示されます。変更は、既存のスタティックテキストフィールドの編集とスタティックテキストフィールドの追加にとどめてください。処理が必要なコントロールの追加には SdAskOptions ソーススクリプトの変更が必要となるため、お勧めできません。

SdAskOptions の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdAskOptions 関数を示します。
*
* このスクリプトは、インストール
* オプションを提供するダイアログを表示します。
*
 メモ : このスクリプト例を実行するには、いくつかの機能および
*       / またはサブ機能を含むプロジェクトを
*       作成 ( またはプロジェクトに挿入 ) してください。
*
¥*---------------------------------------------------------------------*/
```

```
// ここで機能名を指定します。これらの名前は、
// IDE で機能に付けた名前です。NULL ("") 文字列は基本の機能を指定します
#define FEATURE          ""
#define SDASKOPTSTITLE    " コンポーネントの選択 "
#define SDASKOPTSMSG1     " インストールするコンポーネントを選択してください。"
#define SDASKOPTSMSG2     " 選択にしたがってファイル転送を行います。"
#define APPBASE_PATH      " 会社名 ¥¥ 製品名 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "ifx.h"

function OnFirstUIBefore()
begin

   // デフォルトのインストール先パスを設定します。
   INSTALLDIR = PROGRAMFILES ^ APPBASE_PATH;

   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   // インストール オプションを取得します。
   SdAskOptions (SDASKOPTSTITLE, SDASKOPTSMSG1, SDASKOPTSMSG2,
            "", FEATURE, NONEXCLUSIVE);

end;
```

SdAskOptionsList

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdAskOptionsList 関数は、カスタムインストール用に機能のリストを表示するダイアログを作成します。

構文

```
SdAskOptionsList ( szTitle, szMsg, szFeatures, nStyle );
```

パラメーター

テーブル 7 · SdAskOptionsList のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「コンポーネントの選択」を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szFeatures	表示するサブ機能を含む機能の名前を指定します。サブ機能は、チェック ボックスまたはオプションボタンよりも優先されます。トップレベルの機能をすべて表示するには、このパラメーターでヌル文字列 ("") を渡します。 SdAskOptionsList は、ファイルメディアライブラリ、またはシステム変数 MEDIA が指定するスクリプト作成の機能セット内で、要求された機能を検索します。
nStyle	エンドユーザーの選択が制限されるかどうかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · EXCLUSIVE— エンドユーザーは一覧から項目を 1 つだけ選択できます。szFeatures のサブ機能のいづれかが機能を必要とする場合は、EXCLUSIVE モードは利用しないで下さい。 · NONEXCLUSIVE— エンドユーザーは一覧から項目を 1 つ以上 (隣接していない複数の選択を含む) 選択できます。すべてのオプションを選択する [すべて選択] と、すべての選択をクリアする [すべてクリア] の 2 つのボタンも表示します。

戻り値

テーブル 8 · SdAskOptionsList の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。
- セットアップでセットアップの種類のダイアログを使用しない場合は、SdAskOptionsList を呼び出す前に FeatureSetupTypeSet を*必ず*呼び出して、IDE の [セットアップの種類] ビューで定義されているセットアップの種類を指定してください。

- SdAskOptionsList は、MEDIA システム変数によって指定される現在のメディアで実行されます。セットアップ初期設定中、インストールはファイル メディア ライブラリ (**Data1.cab**) に関連付けられたメディア名を MEDIA に割り当てます。

タスク ***スクリプト作成の機能を表示するには:***

1. 現在の MEDIA 値を文字列変数の形式で保存します。たとえば、szSaveMEDIAValue。
2. スクリプト作成コンポーネントセットの名前を MEDIA に割り当てます。
3. SdAskOptionsList を呼び出して、エンドユーザーの選択を取得します。
4. ステップ 1 で保存した値を MEDIA に割り当てます。必ず FeatureTransferData を呼び出す前に行います。

SdAskOptionsList の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * NONEXCLUSIVE と EXCLUSIVE オプションを利用した
 *  SdAskOptionsList 関数のデモンストレーションを行います。
 *
¥*---------------------------------------------------------------*/

// 文字列を定義します。実際のセットアップでは文字列テーブルでこれらを定義し、
// 各定数の前に「@」を付けてスクリプトで使用します。
#define COMP_SELECT_TITLE          " コンポーネントの選択 "
#define COMP_SELECT_MSG            " インストールするコンポーネントを選択してください。"
#define MY_FEATURE_NAME"DefaultFeature"

#include "ifx.h"

function OnFirstUIBefore()
begin

  // セットアップダイアログで [ 戻る ] ボタンを無効にします。
  Disable(BACKBUTTON);

  // ユーザにトップレベルのコンポーネントから非排他的に選択させます。
  SdAskOptionsList(COMP_SELECT_TITLE, COMP_SELECT_MSG + " NONEXCLUSIVE", "", NONEXCLUSIVE);

  // ユーザーに MY_FEATURE_NAME のサブコンポーネントから排他的に選択させます。
  SdAskOptionsList(COMP_SELECT_TITLE, COMP_SELECT_MSG + " EXCLUSIVE", MY_FEATURE_NAME, EXCLUSIVE);

end;
```

SdBitmap

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdBitmap 関数は、ダイアログにビットマップを表示します。ビットマップの最大許容サイズは、幅 440 ピクセル、高さ 275 ピクセルです。また、メッセージを表示するコントロールが可視になるよう、リソースエディターを使用して SdBitmap ダイアログのリソースを変更した場合のみ、メッセージを SdBitmap ダイアログに表示することもできます。

構文

SdBitmap (szTitle, szMsg, szBitmap);

パラメーター

テーブル 9 · SdBitmap のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「ようこそ」を表示するには、このパラメーターにヌル文字列 (“”) を渡します。
szMsg	メッセージを表示するためにリソースエディターを使用して SdBitmap ダイアログを変更しない限り、このパラメーターでヌル文字列 (“”) を渡します。下の「追加情報」セクションを参照してください。
szBitmap	表示するビットマップのファイル名、およびオプションとしてビットマップ属性のセットを指定します。ビットマップ属性が含まれている場合、このパラメーターに渡された文字列は次のようにフォーマットする必要があります。 “ ビットマップファイル名 ; 透明フラグ ;3-D フラグ ; 背景色 ” · **ビットマップ ファイル名** – ビットマップファイルの名前を指定します。ファイル名が完全でない場合 (つまり、ファイル名にドライブインストール先およびパスが含まれていない場合)、InstallShield では、SUPPORTDIR でビットマップを検索します。 · **透過フラグ** – ビットマップを透過的に示すかどうかを示します。このフラグが 1 (true) の場合、マジェンタ (RGB 値 : 255,0,255) であるビットマップの全部分が透明に表示されます。このパラメーターのデフォルト値は 0 (非透明) です。 · **3-D フラグ** – ビットマップが含まれるスタティック フィールドの縁に 3-D 輪郭線を追加するかどうかを示します。このパラメーターのデフォルト値は 0 (3D 輪郭線なし) です。 · **背景色** – スタティック テキスト フィールドの背景に使用する色を示します。この色は、ビットマップが表示される静的テキストフィールドより小さい場合、または透明フラグが 1 に設定されており、ビットマップに透明な領域がある場合にのみ表示されます。背景色は、コンマで区切られた 3 つの数値である RGB 値として表される必要があります。 次の例では、SUPPORTDIR フォルダーにある MyBitmap.bmp ファイルからのビットマップを表示します。ビットマップは黒背景に表示され、３ D 輪郭線はありません。マジェンタであるビットマップはいずれの部分も、黒い背景色で表示されます。 “MyBitmap.bmp;1;1;0,0,0”

戻り値

テーブル 10 · SdBitmap の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。

テーブル 10 ・ SdBitmap の戻り値 (続き)

戻り値	説明
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。

追加情報

szMsg パラメーターとして渡されたメッセージ文字列が SdBitmap ダイアログに表示されるよう、リソース エディターを使用して SdBitmap ダイアログのリソースを変更することができます。

SdBitmap ダイアログ リソースは _isres.dll に含まれます。リソースは、パラメーター szMsg として渡される文字列を受け取るスタティック テキスト コントロールを含みます。しかし、デフォルトでは、この静的テキストコントロールは SdBitmap ダイアログのビューの外側 (ダイアログの下) に配置されます。また、SdBitmap は、静的テキストコントロールを使用して、ビットマップイメージも表示します。ビットマップ イメージの静的テキスト コントロールのサイズは変更でき、メッセージの静的テキスト コントロールはダイアログのビューに移動することができます。szMsg のメッセージは、SdBitmap が呼び出されると表示されます。

ビットマップイメージの静的テキストコントロールのサイズを変更すると、ビットマップイメージの表示に影響を与えるので、注意してください。ビットマップイメージは、SdBitmap がビットマップイメージの静的テキストコントロールに中央付けられる際に切り取られないよう、小さくなければなりません。

この機能では、透明ビットマップをサポートしていません。透明ビットマップをこの機能で使用した場合、透明部分は通常どおりに表示されます。

SdBitmap はメタファイルをサポートしません。

SdBitmap の例

プロジェクト・*この情報は、次のプロジェクトの種類に適用します :*

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdBitmap 関数のデモンストレーションを行います。
 *
 * メモ : このスクリプトを実行する前に、定義済み定数
 *      BITMAP_FILE が [ サポート ファイル / ビルボード ] ビューに含まれる
 *      ビットマップ ファイルを参照するように設定してください。
 *
¥*---------------------------------------------------------------------*/

// 表示するビットマップ
#define BITMAP_FILE SUPPORTDIR ^ "MyBitmap.bmp"

// SdBitmap ダイアログに利用するタイトル。
#define TITLE_TEXT "SdBitmap の例 "

#include "Ifx.h"
```

```
function OnBegin()
begin
  // セットアップダイアログで [ 戻る ] ボタンを無効にします。
  Disable(BACKBUTTON);

  // ダイアログに指定したビットマップを表示します。ダイアログは
  //  メッセージを表示するようカスタマイズされていないので、
  // 2 番目のパラメーターでヌル文字列を渡してください。
  SdBitmap (TITLE_TEXT, "", BITMAP_FILE);

end;
```

SdConfirmNewDir

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdConfirmNewDir 関数は、フォルダー名をプロンプトして確認を求めるダイアログを作成します。エンドユーザーが [はい] をクリックした場合、この関数は自動的に新規フォルダーを作成します。

構文

```
SdConfirmNewDir (szTitle, szDir, nReserved);
```

パラメーター

テーブル 11 ・ SdConfirmNewDir のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「フォルダー選択の確認」を表示するには、このパラメーターでヌル文字列 (“”) を渡します。
szDir	確認するディレクトリの名前を指定します。(この情報は SdAskDestPath を呼び出して取得します。)
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。

戻り値

テーブル 12 ・ SdConfirmNewDir の戻り値

戻り値	説明
YES (1)	[はい] ボタンがクリックされたことを示します。ディレクトリが確認されて作成されます。
NO (0)	[いいえ] ボタンがクリックされたことを示します。指定されたディレクトリは作成されません。
< 0	[はい] ボタンがクリックされたが、関数が新規ディレクトリを作成できなかったことを示します。

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。
- この関数によって表示されるダイアログは、スキンを使って表示することはできません。スキンの指定に関わらず、同じように表示されます。

SdConfirmNewDir の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdConfirmNewDir 関数のデモンストレーションを行います。
```

```
*
* このスクリプト例では、まず SdAskDestPath を呼び出してユーザーから
* インストール先フォルダーを取得します。フォルダーが存在しない場合、
* SdConfirmNewDir が呼び出され、ユーザーがフォルダーを作成するかどうか
* 問い合わせます。
*
* メモ : このスクリプトはローカルハードディスク上にディレクトリを作成します。
*
¥*---------------------------------------------------------------*/

#define DEFAULT_TARGET_FOLDER "C:¥¥NONEXIST¥¥DIR";
#define TITLE_TEXT "SdConfirmNewDir の例 "

#include "ifx.h"

function OnBegin()
    NUMBER nResult;
    STRING szMsg, svDir;
begin

    // セットアップダイアログで [ 戻る ] ボタンを無効にします。
    Disable(BACKBUTTON);

start:
    // SdAskDestPath への呼び出し用パラメーターをセットアップします。
    szMsg = " インストール先フォルダーを選択します :";
    svDir = DEFAULT_TARGET_FOLDER

    // ユーザーからインストール先フォルダーを読み出します。
    nResult = SdAskDestPath (TITLE_TEXT, szMsg, svDir, 0);

    // 選択されたフォルダーが存在するかどうかを確認します。
    if (ExistsDir (svDir) = EXISTS) then
        // ユーザーに対して指定したファイルが既に存在することを通知します。
        szMsg = " フォルダー '%s' は既に存在します。¥n¥n この例を適切に実行するため、" +
                " 既存しないフォルダーを指定してください。";
        SprintfBox (INFORMATION, TITLE_TEXT, szMsg, svDir);

        // 最初からやり直します。
        goto start;
    else
        // 指定したフォルダーが存在しません。Request user
        // 作成を確認します。
        nResult = SdConfirmNewDir (TITLE_TEXT, svDir, 0);

        if (nResult = NO) then
            // ユーザーが作成しない選択をしました。
            MessageBox (" 選択されたフォルダーは作成されませんでした。", INFORMATION);

            // 最初からやり直します。
            goto start;
        elseif (nResult = YES) then
            // ユーザーはフォルダーの作成を希望します。
            SprintfBox (INFORMATION, TITLE_TEXT, "%s が作成されました。", svDir);
        elseif (nResult < 0) then
            // エラーを報告し、中止します。
            MessageBox ("SdConfirmNewDir が失敗しました。", SEVERE);
            abort;
        endif;
```

```
    endif;

end;
```

SdConfirmRegistration

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdConfirmRegistration 関数は、ユーザー名、会社名、およびシリアル番号を表示するメッセージ ボックスを作成します。ヌル文字列 ("") がダイアログのフィールドに入力されている場合、表示されるフィールドは空になります。

構文

```
SdConfirmRegistration (szTitle, szName, szCompany, szSerial, nReserved);
```

パラメーター

テーブル 13 - SdConfirmRegistration のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「登録の確認」を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szName	エンドユーザーの名前を指定します。
szCompany	会社名を指定します。
szSerial	シリアル番号を指定します。パラメーターにヌル文字列 ("") が含まれている場合、シリアル番号のフィールドはダイアログに表示されません。
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。

戻り値

テーブル 14 - SdConfirmRegistration の戻り値

戻り値	説明
YES (1)	[はい] ボタンがクリックされたことを示します。
NO (0)	[いいえ] ボタンがクリックされたことを示します。

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。
- この関数によって表示されるダイアログは、スキンを使って表示することはできません。スキンの指定に関わらず、同じように表示されます。
- シリアル番号およびエンド ユーザーの名前と会社名を取得するには、**SdRegisterUserEx** を呼び出します。エンド ユーザーの名前と会社名のみを取得するには、**SdRegisterUser** を呼び出します。

SdConfirmRegistration の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
```

```
*
* SdRegisterUser 関数と SdConfirmRegistration 関数のデモンストレーションを
* デモンストレーションを行います。
*
* SdRegisterUser が呼び出されて、ユーザー名と会社名を
* 問い合わせます。これらのエントリは SdConfirmRegistration が呼び出されたときに
* 確認されます。
*
¥*---------------------------------------------------------------------*/

#define REG_TITLE     "SdRegisterUser の例 "
#define REG_MSG       " ここで製品を登録してください。"
#define CONFIRM_TITLE "SdConfirmRegistration の例 "

#include "Ifx.h"

function OnBegin()
   STRING  svName, svCompany;
   NUMBER  nResult;
begin

   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   repeat
      // ユーザー名と会社名を取得します。
      SdRegisterUser (REG_TITLE, REG_MSG, svName, svCompany);

      // 情報が正しいことを確認します。SdRegisterUser は
      // シリアル番号を取得しないので、パラメーター 4 に文字列を渡します。
      nResult = SdConfirmRegistration (CONFIRM_TITLE, svName, svCompany, "", 0);
   until nResult = YES;

end;
```

SdCustomerInformation

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdCustomerInformation 関数は、エンド ユーザーがインストール中の製品のユーザー名および会社名を指定できるダイアログを表示します。このダイアログには、エンド ユーザーが製品をすべてのユーザーにインストールするか、または現在のユーザーにのみインストールするかを指定できるラジオボタンを含めることもできます。

適切なパラメーターを指定することで、これらのフィールドにデフォルト値を設定できます。ヌル文字列 ("") を指定すると、関数は適切なスクリプト変数を使用します。

データが両方の編集フィールドに存在する場合のみ、[次へ] ボタンは有効になります。エンドユーザーはフィールドを空白にすることはできません。

構文

SdCustomerInformation (szTitle, svName, svCompany, bvAllUsers);

パラメーター

テーブル 15 · SdCustomerInformation のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「ユーザー情報」を表示するには、このパラメーターでヌル文字列 ("") を渡します。
svName	関数が呼び出されたときの " 名前 " 編集フィールドのデフォルト値を指定します。 ヌル文字列 ("") が指定された場合、デフォルト値は、IFX_PRODUCT_REGISTEREDOWNER 変数の現在の値となります。InstallScript プロジェクトの場合、この変数は初回インストールの場合はレジストリから、またメンテナンス モードの場合は対応するテキスト置換から読み込まれます。InstallScript MSI プロジェクトの場合、変数のデフォルト値は常に Windows Installer プロパティ **USERNAME** から読み込まれます。 関数は、エンド ユーザーがこのパラメーターで指定した値を返します。InstallScript インストールの場合、関数はさらに *IFX_PRODUCT_REGISTEREDOWNER* の値をエンド ユーザーが指定した値に設定します。InstallScript MSI インストールの場合、関数は Windows Installer プロパティ **USERNAME** を自動的に更新します。
svCompany	関数が呼び出されたときの " 会社 " 編集フィールドのデフォルト値を指定します。 ヌル文字列 ("") が指定された場合、デフォルト値は、IFX_PRODUCT_REGISTEREDCOMPANY 変数の現在の値となります。InstallScript プロジェクトの場合、この変数は初回インストールの場合はレジストリから、またメンテナンス モードの場合は対応するテキスト置換から読み込まれます。InstallScript MSI プロジェクトの場合、変数のデフォルト値は常に Windows Installer プロパティ **COMPANYNAME** から読み込まれます。 関数は、エンド ユーザーがこのパラメーターで指定した値を返します。InstallScript インストールの場合、関数はさらに *IFX_PRODUCT_REGISTEREDCOMPANY* の値をエンド ユーザーが指定した値に設定します。InstallScript MSI インストールの場合、関数は Windows Installer プロパティ **COMPANYNAME** を自動的に更新します。

テーブル 15 ・ SdCustomerInformation のパラメーター (続き)

パラメーター	説明
bvAllUsers	エンド ユーザーが選択したオプションを返します。関数が戻ったあと、bvAllUsers が次の値の 1 つに設定されます : ・ TRUE Ñ [**このコンピューターの利用者すべて [すべてのユーザー]**] オプションが選択されます。 InstallScript インストールでエンド ユーザーがこのオプションを選択すると、関数はシステム変数 ALLUSERS をゼロ以外の値に設定します。 InstallScript MSI インストールでエンド ユーザーがこのオプションを選択すると、関数は **ALLUSERS** プロパティを 2 に設定します。 ・ FALSE—[**自分だけ [ユーザー名]**] オプションがデフォルトで選択されます。 InstallScript インストールでエンド ユーザーがこのオプションを選択すると、関数は ALLUSERS を FALSE に設定します。 デフォルト オプションは、bvAllUsers パラメーターの現在の値ではなく、InstallScript MSI インストールでは Windows Installer プロパティ **ALLUSERS** に基づき、InstallScript インストールでは ALLUSERS システム変数に基づきます。 ・ **ALLUSERS** プロパティが 2、または ALLUSERS システム変数がゼロ以外の場合、デフォルトで [すべてのユーザー] オプションが選択されます。 ・ **ALLUSERS** プロパティが 1、または ALLUSERS システム変数が FALSE の場合、デフォルトで [ユーザーごと] オプションが選択されます。 次のようにスクリプト変数をアップデートすると、ラジオ ボタンのどちらか、またはその両方を無効 / 非表示にすることができます : ・ DISABLE_PERUSERBTN Ñ [ユーザーごと] オプションを、通常は有効であるところを無効 (または HIDE_DISABLED_BTNS が TRUE の場合は非表示) にすることを示します。この変数のデフォルト値は、FALSE です。Windows 9x プラットフォーム上では、この変数の値に関わらず、[ユーザーごと] オプションは常に非表示となります。 ・ DISABLE_ALLUSERBTN Ñ [すべてのユーザー] オプションを、通常は有効であるところを無効 (または非表示) にすることを示します。この変数のデフォルト値は、FALSE です。インストールが管理者権限またはパワー ユーザー権限なしで実行されている場合、この変数の値に関わらず、[すべてのユーザー] オプションは常に非表示となります。 ・ HIDE_DISABLED_BTNS Ñ 両方のオプションが非表示ではなく無効であることを示します。この変数のデフォルト値は、TRUE です。この変数が TRUE に設定されると、オプションのどちらかが無効である場合、両方のオプションが非表示となります。

戻り値

テーブル 16 - SdCustomerInformation の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdCustomerInformation の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 *  SdCustomerInformation 関数のデモンストレーションを行います。
 *  SdCustomerInformation はエンドユーザーに対して、ユーザー名と会社名の
 *  入力を要求し、インストールが現在のユーザー専用かターゲットシステムの
 * すべてのユーザーが対象なのかを問い合わせます。
 *
¥*---------------------------------------------------------------------*/

#include "ifx.h"

function OnFirstUIBefore( )
    // ... その他の変数宣言 ...
    STRING  svName, svCompany, szMsg;
    NUMBER  nvUser, nReturn;
begin

// ... その他のダイアログを表示します ...

    // ユーザー名と会社名を取得します
    SdCustomerInformation("", svName, svCompany, nvUser);

    if (nvUser = 0) then
        szMsg = " ユーザーごとのインストール ";
    else
        szMsg = " すべてのユーザー用インストール ";
    endif;
```

```
    MessageBox(" 入力内容 :¥n¥n" +
        " 名前 : " + svName + "¥n" +
        " 会社 : " + svCompany + "¥n" +
        " 種類 : " + szMsg,
        INFORMATION);

// ... 他のダイアログ ...

end;
```

SdCustomerInformationEx

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdCustomerInformationEx 関数は、エンド ユーザーがインストール中の製品のユーザー名、会社名、およびシリアル番号を指定できるダイアログを表示します。このダイアログには、エンド ユーザーが製品をすべてのユーザーにインストールするか、または現在のユーザーにのみインストールするかを指定できるラジオボタンを含めることもできます。

適切なパラメーターを指定することで、これらのフィールドにデフォルト値を設定できます。ヌル文字列 ("") を指定すると、関数は適切なスクリプト変数を使用します。

データが この 3 つの編集フィールドすべてに存在する場合のみ、[次へ] ボタンは有効になります。エンドユーザーはフィールドを空白にすることはできません。

メモ・SdCustomerInformationEx** 関数はシリアル番号の検証を行いません。シリアル番号を検証するコードを追加する方法については、サンプル シリアル番号検証プロジェクトを参照してください。このサンプル プロジェクトは、InstallShield プログラム ファイル フォルダー内の Samples サブフォルダーの 1 つにあります。デフォルトのインストール先は、C:¥Program Files¥InstallShield¥2016¥Samples¥InstallScript¥Serial Number Validation Sample Project** です。*

構文

```
SdCustomerInformationEx ( szTitle, svName, svCompany, svSerial, bvAllUsers );
```

パラメーター

テーブル 17 · SdCustomerInformationEx のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「ユーザー情報」を表示するには、このパラメーターでヌル文字列 ("") を渡します。
svName	関数が呼び出されたときの " 名前 " 編集フィールドのデフォルト値を指定します。 ヌル文字列 ("") が指定された場合、デフォルト値は、IFX_PRODUCT_REGISTEREDOWNER 変数の現在の値となります。InstallScript プロジェクトの場合、この変数は初回インストールの場合はレジストリから、またメンテナンス モードの場合は対応するテキスト置換から読み込まれます。InstallScript MSI プロジェクトの場合、変数のデフォルト値は常に Windows Installer プロパティ **USERNAME** から読み込まれます。 関数は、エンド ユーザーがこのパラメーターで指定した値を返します。InstallScript インストールの場合、関数はさらに *IFX_PRODUCT_REGISTEREDOWNER* の値をエンド ユーザーが指定した値に設定します。InstallScript MSI インストールの場合、関数は Windows Installer プロパティ **USERNAME** を自動的に更新します。
svCompany	関数が呼び出されたときの " 会社 " 編集フィールドのデフォルト値を指定します。 ヌル文字列 ("") が指定された場合、デフォルト値は、IFX_PRODUCT_REGISTEREDCOMPANY 変数の現在の値となります。InstallScript プロジェクトの場合、この変数は初回インストールの場合はレジストリから、またメンテナンス モードの場合は対応するテキスト置換から読み込まれます。InstallScript MSI プロジェクトの場合、変数のデフォルト値は常に Windows Installer プロパティ **COMPANYNAME** から読み込まれます。 関数は、エンド ユーザーがこのパラメーターで指定した値を返します。InstallScript インストールの場合、関数はさらに *IFX_PRODUCT_REGISTEREDCOMPANY* の値をエンド ユーザーが指定した値に設定します。InstallScript MSI インストールの場合、関数は Windows Installer プロパティ **COMPANYNAME** を自動的に更新します。
svSerial	関数が呼び出されたときの " シリアル番号 " 編集フィールドのデフォルト値を指定します。 ヌル文字列 ("") が指定された場合、デフォルト値は、IFX_PRODUCT_REGISTEREDSERIALNUM 変数の現在の値となります。この変数は初回インストールの場合はレジストリから、またメンテナンス モードの場合は対応するテキスト置換から読み込まれます。 関数は、エンド ユーザーがこのパラメーターで指定した値を返します。関数はさらに *IFX_PRODUCT_REGISTEREDSERIALNUM* の値をエンド ユーザーが指定した値に設定します。

テーブル 17 • SdCustomerInformationEx のパラメーター (続き)

パラメーター	説明
bvAllUsers	エンド ユーザーが選択したオプションを返します。関数が戻ったあと、bvAllUsers が次の値の 1 つに設定されます : • TRUE Ñ [**このコンピューターの利用者すべて [すべてのユーザー]**] オプションが選択されます。 InstallScript インストールでエンド ユーザーがこのオプションを選択すると、関数はシステム変数 ALLUSERS をゼロ以外の値に設定します。 InstallScript MSI インストールでエンド ユーザーがこのオプションを選択すると、関数は **ALLUSERS** プロパティを 2 に設定します。 • FALSE—[**自分だけ [ユーザー名]**] オプションがデフォルトで選択されます。 InstallScript インストールでエンド ユーザーがこのオプションを選択すると、関数は ALLUSERS を FALSE に設定します。 デフォルト オプションは、bvAllUsers パラメーターの現在の値ではなく、InstallScript MSI インストールでは Windows Installer プロパティ **ALLUSERS** に基づき、InstallScript インストールでは ALLUSERS システム変数に基づきます。 • **ALLUSERS** プロパティが 2、または ALLUSERS システム変数がゼロ以外の場合、デフォルトで [すべてのユーザー] オプションが選択されます。 • **ALLUSERS** プロパティが 1、または ALLUSERS システム変数が FALSE の場合、デフォルトで [ユーザーごと] オプションが選択されます。 次のようにスクリプト変数をアップデートすると、ラジオ ボタンのどちらか、またはその両方を無効 / 非表示にすることができます : • DISABLE_PERUSERBTN Ñ [ユーザーごと] オプションを、通常は有効であるところを無効 (または HIDE_DISABLED_BTNS が TRUE の場合は非表示) にすることを示します。この変数のデフォルト値は、FALSE です。Windows 9x プラットフォーム上では、この変数の値に関わらず、[ユーザーごと] オプションは常に非表示となります。 • DISABLE_ALLUSERBTN Ñ [すべてのユーザー] オプションを、通常は有効であるところを無効 (または非表示) にすることを示します。この変数のデフォルト値は、FALSE です。インストールが管理者権限またはパワー ユーザー権限なしで実行されている場合、この変数の値に関わらず、[すべてのユーザー] オプションは常に非表示となります。 • HIDE_DISABLED_BTNS Ñ 両方のオプションが非表示ではなく無効であることを示します。この変数のデフォルト値は、TRUE です。この変数が TRUE に設定されると、オプションのどちらかが無効である場合、両方のオプションが非表示となります。

戻り値

テーブル 18 - SdCustomerInformationEx の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdCustomerInformationEx の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 *  SdCustomerInformationEx 関数のデモンストレーションを行います。
 *  SdCustomerInformationEx はエンドユーザーに対して、ユーザー名、会社名、
 * そしてシリアル番号の入力を要求し、またこのインストールが
 * ターゲットシステムを利用するすべてのユーザー用なのか、現在の
 * ユーザー専用なのかを問い合わせます。
 *
¥*---------------------------------------------------------------------*/

#include "ifx.h"

function OnFirstUIBefore( )
    // ... その他の変数宣言 ...
    STRING  svName, svCompany, svSerial, szMsg;
    NUMBER  nvUser, nReturn;
begin

// ... その他のダイアログを表示します ...

    // ユーザー名と会社名を取得します
    SdCustomerInformationEx("", svName, svCompany, svSerial, nvUser);

    if (nvUser = 0) then
        szMsg = " ユーザーごとのインストール ";
    else
        szMsg = " すべてのユーザー用インストール ";
    endif;
```

```
    MessageBox(″ 入力内容 :¥n¥n″ +
        ″ 名前 : ″ + svName + ″¥n″ +
        ″ 会社 : ″ + svCompany + ″¥n″ +
        ″ シリアル番号 : ″ + svSerial + ″¥n″ +
        ″ 種類 : ″ + szMsg,
        INFORMATION);

// ... 他のダイアログ ...

end;
```

SdDiskSpace2

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdDiskSpace2 関数は、次のいずれかを表示するダイアログを表示します :

- ボリューム、必要なディスク容量、利用可能なディスク容量、および必要なディスク容量と使用可能なディスク容量との差異のリスト。
- ターゲット システムに、インストールで必要なディスク容量が不足していることを示す警告メッセージ。ダイアログには、ボリューム、必要な容量、使用可能容量および必要な容量と使用可能な容量の差についてのリストビューも表示されます。

SdDiskSpace2 関数は、**SdDiskSpaceRequirements** および **SdOutOfDiskSpace** 関数に優先します。

構文

```
SdDiskSpace2 (szTitle, szMsg, bUseOutOfSpaceDialog);
```

パラメーター

テーブル 19 - SdDiskSpace2 のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「必要ディスク領域」または 「ディスク容量不足」を表示するには、このパラメーターでヌル文字列 (“”) を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このテキストはスタティック コントロールとみなされます。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 (“”) を渡します。
bUseOutOfSpaceDialog	インストールがディスク容量不足であることを警告するダイアログ、または必要ディスク容量を示すダイアログを表示するかどうかを示します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • TRUE Ñ ターゲット システムに、インストールで必要なディスク容量が不足していることを示す警告メッセージを表示します。ダイアログには、ボリューム、必要な容量、使用可能容量および必要な容量と使用可能な容量の差についてのリストビューも表示されます。 • FALSE Ñ ボリューム、必要な容量、使用可能容量および必要な容量と使用可能な容量の差についてのリスト ビューを表示するダイアログを表示します。

戻り値

テーブル 20 - SdDiskSpace2 Return の値

戻り値	説明
NEXT (1)	エンド ユーザーが、[OK] ボタンをクリックしたことを示します。

追加情報

プロジェクト - *InstallScript MSI インストールでは、OnOutOfDiskSpace イベント ハンドラーは Out Of Disk Space イベントに応答します。OnOutOfDiskSpace のデフォルト実装は、**SdDiskSpace2** ダイアログを表示してからインストールを終了します。*

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdDiskSpace2 の例

プロジェクト - *この情報は、次のプロジェクトの種類に適用します :*

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 *  SdDiskSpace2 関数のデモンストレーション。
 *  SdDiskSpace2 は、ボリューム、必要なディスク容量、利用可能なディスク容量、および
 * 利用可能な容量と必要な容量との差を一覧にする
 * リストビューを表示します。
 *
¥*---------------------------------------------------------------------*/

#include "ifx.h"

function OnFirstUIBefore( )
begin

// ... その他のダイアログを表示します ...

   // 必要ディスク容量を表示します
   SdDiskSpace2 ("",
      " 有効な容量と必要なディスク容量を検討し、" +
      " アプリケーションのインストール先を決定します。");
      FALSE);

// ... 他のダイアログ ...

end;
```

SdDiskSpaceRequirements

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdDiskSpace2 関数は、ボリューム、必要なディスク容量、利用可能なディスク容量、および必要なディスク容量と使用可能なディスク容量を一覧にするリストビューを表示します。

SdDiskSpace2 関数は、**SdDiskSpaceRequirements** 関数に優先します。

構文

```
SdDiskSpaceRequirements ( szTitle, szMsg );
```

パラメーター

テーブル 21 - SdDiskSpaceRequirements のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「必要ディスク領域」を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このテキストはスタティック コントロールとみなされます。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。

戻り値

テーブル 22 - SdDiskSpaceRequirements の戻り値

戻り値	説明
NEXT (1)	エンド ユーザーが、[OK] ボタンをクリックしたことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdDisplayTopics

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

- *InstallScript*
- *InstallScript MSI*

SdDisplayTopics 関数は、トピック データに基づいて情報を表示するダイアログを作成します。ダイアログには、見出し、およびタイトルと説明のトピックが表示されます。このダイアログを使用して、ヘルプのトピック、例などを表示できます。

構文

SdDisplayTopics (szTitle, szMsg, listTopics, listDetails, nReserved);

パラメーター

テーブル 23 · SdDisplayTopics のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「カスタム インストール ヘルプ」を表示するには、このパラメーターでヌル文字列 (“”) を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 (“”) を渡します。
listTopics	表示するトピックを含む文字列リストを指定します。
listDetails	各トピックの説明を含む文字列リストを指定します。
nReserved	このパラメーターでゼロ (0) を渡します。他の値は使用できません。

戻り値

テーブル 24 · SdDisplayTopics の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。
- InstallScript MSI インストールでは、エンドユーザーが [次へ] をクリックしたとき、USERNAME および COMPANYNAME プロパティ は svName フィールドおよび svCompany フィールドそれぞれに含まれる情報を使って設定されます。
- 説明テキストのフォントスタイルを変更して、タイトル (トピック) テキストと区別することができます。メッセージおよびトピックタイトルは、常に太字で表示されます。
- メッセージの静的フィールドには、ID 801 がなければなりません。トピック ID には、802 から 849 の範囲の数字を含みます。説明フィールドには、851 から 899 の範囲の ID を含みます。
- 説明の静的フィールドの間隔は、ダイアログのサイズで決定します。listDetails リストの間隔は、動的に変更できません。トピックおよび説明の数が番号の静的フィールドより少ない場合、空白には何も表示されず、ダイアログのサイズも変更されません。

SdDisplayTopics の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdDisplayTopics 関数のデモンストレーションを行います。
 *
 * このスクリプト例は 2 つのリストを作成します。1 つ目はトピックのタイトル用、
 * もうひとつはトピックの説明用です。そして SdDisplayTopics を呼び出して
 * トピックと説明を表示します。
 *
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "SdDisplayTopics の例 "
#define MSG_TEXT " カスタム セットアップ オプションは、YourApp のどの部分をインストールするか選択を可能にします。"

#define TOPIC1 "YourApp プログラム :"
#define TOPIC2 "YourApp ヘルプ :"
#define TOPIC3 "YourApp の例 :"

#define DESC1  "YourApp を実行、および書き込むすべてのファイルを含みます。"
#define DESC2  "YourApp を利用したプログラムの作成方法をデモンストレーションする、コンピューターを使ったチュートリアル。"
#define DESC3  "YourApp を利用して作成したアプリケーションのいくつかの例 "

#include "Ifx.h"

function OnBegin()
    LIST listDescriptions, listTopics;
begin

    // トピックのリストを作成します。
    listTopics = ListCreate (STRINGLIST);

    // トピックの説明のリストを作成します。
    listDescriptions = ListCreate (STRINGLIST);

    if (listTopics = LIST_NULL) || (listDescriptions = LIST_NULL) then
        // エラーを報告し、中止します。
        MessageBox(" リストを作成することができませんでした。", INFORMATION);
        abort;
    endif;

    // トピックのリストをビルドします。
    ListAddString (listTopics, TOPIC1, AFTER);
    ListAddString (listTopics, TOPIC2, AFTER);
    ListAddString (listTopics, TOPIC3, AFTER);

    // トピックの説明のリストをビルドします。
    ListAddString (listDescriptions, DESC1, AFTER);
```

```
    ListAddString (listDescriptions, DESC2, AFTER);
    ListAddString (listDescriptions, DESC3, AFTER);

    // トピックと説明を表示します。
    SdDisplayTopics (TITLE_TEXT, MSG_TEXT, listTopics, listDescriptions, 0);

    // メモリからリストを削除します。
    ListDestroy (listTopics);
    ListDestroy (listDescriptions);

end;
```

SdExceptions

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdExceptions 関数は、共有、ロック (使用中)、または読取り専用のファイルが検出されたとき、エンド ユーザーにメッセージ ボックスを表示して適切なオプションを選択できるようにします。

構文

```
SdExceptions (nExceptionType, szFilename);
```

パラメーター

テーブル 25 - SdExceptions のパラメーター

パラメーター	説明
nExceptionType	発生したファイルエラーの種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • SHARED— 共有ファイルの参照カウントが、ゼロになっています。 • READONLY— 読み取り専用ファイルが検出されています。 • LOCKED— ロックファイルが検出されています。
szFilename	問題の原因となっているファイルの名前を指定します。

戻り値

テーブル 26 - SdExceptions の戻り値

戻り値	説明
ERR_RETRY (4)	[再試行] ボタンが選択されたことを示します。
ERR_IGNORE (5)	[無視] ボタンが選択されたことを示します。
ERR_YES (6)	[はい] ボタンが選択されたことを示します。
ERR_NO (7)	[いいえ] ボタンが選択されたことを示します。
ERR_PERFORM_AFTER_REBOOT (100)	[再起動] ボタンが選択されたことを示します。
< 0	ダイアログが表示されなかったことを示します。

追加情報

この関数によって表示されるダイアログは、スキンを使って表示することはできません。スキンの指定に関わらず、同じように表示されます。

SdExceptions の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
```

```
＊SdExceptions 関数はダイアログを表示して エンドユーザーに
共有ファイル、ロックされた ( 使用中 ) ファイル、読み取り専用ファイルであることを知らせるダイアログ ボックスが表示されます。
＊適切なオプションも提供されます。
＊
＊SdExcpetions 関数は、
＊次のその他のイベント ハンドらの既定コードに使用されます :
＊
＊        OnFileLocked
＊        OnFileReadOnly
＊        OnRemovingSharedFile
＊
＊以下のサンプル スクリプトは、OnFileReadOnly イベントを使用します。ここで
＊SdExceptions プロンプトを出すには、セットアップが読み取り専用ファイルの
＊上書きまたはアンインストールを試みる必要があります。
＊
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

//---------------------------------------------------------------------------
// OnFileReadOnly
//
// OnFileReadOnly は、読み取り専用ファイルがインストールまたはアンインストール
// される時に呼び出されます。
//
// szFile は、イベントが呼び出されたときに読み取り専用ファイルの完全パスを
// 含みます。
//
// イベントは、次の値のひとつを戻します :
//
// ERR_YES - ファイルをインストールまたはアンインストールすべきであることを示します。
//
// ファイルをインストールまたはアンインストールすべきではないことを示します。
//---------------------------------------------------------------------------
function OnFileReadOnly(szFile)
begin
    // TODO: ERR_YES を返し、確認なしで
    // 読み取り専用ファイルをインストールまたはアンインストールする場合、このコードを有効にします。
    // return ERR_YES;

    return SdExceptions(READONLY, szFile);

end;
```

SdFeatureDialog

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdFeatureDialog 関数は、エンドユーザーがインストールできるセットアップの機能のリスト、および各機能が使用するディスク容量を表示するダイアログを作成します。この機能は、SdFeatureDialogAdv と全く同じです。

エンドユーザーは [参照] ボタン、そして [空きディスク容量] ボタンのをクリックしてその他のドライブの空き容量を確認し、インストール先フォルダーを変更することができます。

構文

SdFeatureDialog (szTitle, szMsg, svDir, szFeatures);

パラメーター

テーブル 27 - SdFeatureDialog のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (" 機能の選択 ") を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
svDir	デフォルトで選択するフォルダーの名前を指定し、エンドユーザーが選択したフォルダーの名前を戻します。svDir によって指定されたインストール先のフォルダーは、INSTALLDIR、TARGETDIR または他のシステム変数に自動的に割り当てられません。svDir の値をインストールに適用するには、その値を INSTALLDIR へ (InstallScript MSI インストールの場合)、または TARGETDIR (InstallScript インストールの場合) へ割り当て、システム変数が使用中の場合はスクリプト定義の変数へ割り当てなくてはなりません。 svDir によって指定されたデフォルトのフォルダーがエンドユーザーのシステムに存在しない場合、エンドユーザーが [参照] ボタンをクリックして、[フォルダーの選択] ダイアログのステップに従ってフォルダーを作成しない限り、フォルダーは作成されません。従って、デフォルトフォルダーを指定するときはいつでも、FeatureDialog が戻るときに ExistsDir を呼び出し、そのフォルダーの存在を確認しなくてはなりません。フォルダーが存在しない場合、CreateDir を呼び出して、エンド ユーザーのシステムでそのフォルダーを作成します。
szFeatures	サブ機能を表示する機能の名前を指定します。トップレベルの機能をすべて表示するには、このパラメーターでヌル文字列 ("") を渡します。トップ階層の機能やサブ機能の参照方法は、「関数呼び出しで機能やサブ機能を指定する」を参照してください。 SdFeatureDialog はシステム変数 MEDIA が指定するスクリプト作成の機能セットで必要な機能を検索します。追加情報セクションを参照してください。

戻り値

テーブル 28 - SdFeatureDialog の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

- 機能のサイズは、選択されるまで 0 と表示されます。サイズが選択されると、実際のサイズが表示されます。
- ダイアログに表示される必要ディスク容量には、メンテナンスとアンインストーラーを使用するためにインストールするファイルのサイズが含まれます。すべてのアプリケーションコンポーネントが選択解除されても、これらのファイルのサイズは表示されたままです。
- インストールでセットアップ タイプ ダイアログを使用しない場合は、**SdFeatureDialog** を呼び出す前に **FeatureSetupTypeSet** を呼び出して、[セットアップの種類] ビューで定義されているセットアップ タイプを指定してください。
- **SdFeatureDialog** は、MEDIA システム変数によって指定される現在のメディアで実行されます。セットアップ初期設定中、インストールはファイル メディア ライブラリ (**Data1.cab**) に関連付けられたメディア名を MEDIA に割り当てます。

タスク ***スクリプト作成の機能を表示するには:***

1. 現在の MEDIA 値を文字列変数の形式で保存します。たとえば、szSaveMEDIAValue。
2. スクリプト作成コンポーネントセットの名前を MEDIA に割り当てます。
3. **SdFeatureDialog** を呼び出して、エンドユーザーの選択を取得します。
4. ステップ 1 で保存した値を MEDIA に割り当てます。必ず **FeatureTransferData** を呼び出す前に行います。

- 最大の機能サイズを表示できるように、機能名は必要に応じて切り詰められます。このサイズを表示するために必要な領域は、最大機能サイズ (2 GB)、現在使用されている機能サイズオプション、およびダイアログ機能情報を表示するために使用されるフォントによって異なります。機能サイズのオプションは、**DialogSetInfo** 関数で設定します。

  最大サイズの表示に必要な領域が決定されると、すべての機能名は残りの領域に収まるよう、必要に応じて自動的に切り詰められます。この方法では、サイズの表示に必要な領域が小さい (またはサイズが選択されていない) 機能の名前も切り詰められます。機能名がすべて確実に表示されて充分活用できるように、機能名または表示名をダイアログの有効スペースより短くしてください。
- [空きディスク容量] ダイアログではスキンを利用できません。スキンを選択しても同じ外観となります。
- [ディスク容量] ボタンの ID は 101 です。このボタンは、有効なディスク容量ダイアログを自動的に表示します。このボタン / オプションは削除できます。ディレクトリのスタティック フィールドには、ID 851 を含みます。リスト ボックス ID は、複数選択スタイルを持ちます。
- InstallShield Professional の以前のバージョンで作成されたインストールでは、エンド ユーザーが [フォルダーの選択] ダイアログで存在しないフォルダーを選択すると、フォルダーを作成するかどうかを問い合わせる確認メッセージが表示されました。このメッセージは多くのエンドユーザーに混乱を招くため、InstallShield では削除されました。

SdFeatureDialog の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdFeatureDialog 関数のデモンストレーションを行います。
 *
 * このスクリプト例は、ユーザーがインストールできるセットアップに含まれる
 * 各機能に必要なディスク容量を表示する
 * ダイアログ ボックスを表示します。
 *
 * コメント : このスクリプト例を実行するには、いくつかの機能および
 *          / またはファイルを持つコンポーネントを含むサブ機能を
 *          持つプロジェクトを作成 ( またはプロジェクトに挿入 ) します。
 *
 *
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
    STRING  szTitle, szMsg, svDir;
begin
    svDir   = TARGETDIR;
    szTitle = " 機能の選択 ";
    szMsg   = " コンピューターへインストールする機能を選択してください。";

    //   使用できるすべてのトップレベルの機能を表示します。
    SdFeatureDialog (szTitle, szMsg, svDir, "");

end;
```

SdFeatureDialog2

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdFeatureDialog2 関数は次を表示するダイアログを作成します :

- ユーザーがインストールできる機能のリスト。
- 選択した機能への必要ディスク容量と、インストール先の空きディスク容量。機能のサイズは、選択されるまで 0 と表示されます。機能が選択されたとき、その実際のサイズが表示されます。
- 選択された機能の説明 (機能の " 説明 " 設定の値)。

特定の機能にサブ機能がある場合、ユーザーが機能をクリックしたときに [変更] ボタンが有効になります。[変更] ボタンをクリックすると [サブ機能の選択] ダイアログが起動し、ここでエンドユーザーが詳細を選択することができます。

構文

```
SdFeatureDialog2 ( szTitle, szMsg, szDir, szFeatures );
```

パラメーター

テーブル 29 · SdFeatureDialog2 のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (" 機能の選択 ") を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szDir	デフォルトで選択するフォルダーの名前を指定し、エンドユーザーが選択したフォルダーの名前を戻します。szDir によって指定されたインストール先のフォルダーは、INSTALLDIR、TARGETDIR または他のシステム変数に自動的に割り当てられません。szDir の値をインストールに適用するには、その値を INSTALLDIR へ (InstallScript MSI インストールの場合)、または TARGETDIR (InstallScript インストールの場合) へ割り当て、システム変数が使用中の場合はスクリプト定義の変数へ割り当てなくてはなりません。 szDir によって指定されたデフォルトのフォルダーがエンドユーザーのシステムに存在しない場合、エンドユーザーが [参照] ボタンをクリックして、[フォルダーの選択] ダイアログのステップに従ってフォルダーを作成しない限り、フォルダーは作成されません。従って、デフォルトフォルダーを指定するときはいつでも、**FeatureDialog** が戻るときに ExistsDir を呼び出し、そのフォルダーの存在を確認しなくてはなりません。フォルダーが存在しない場合、CreateDir を呼び出して、エンド ユーザーのシステムでそのフォルダーを作成します。
szFeatures	サブ機能を表示する機能の名前を指定します。トップレベルの機能をすべて表示するには、このパラメーターでヌル文字列 ("") を渡します。トップ階層の機能やサブ機能の参照方法は、「関数呼び出しで機能やサブ機能を指定する」を参照してください。 SdFeatureDialog2 はシステム変数 MEDIA が指定するスクリプト作成の機能セットで必要な機能を検索します。追加情報セクションを参照してください。

戻り値

テーブル 30 · SdFeatureDialog2 の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

- デフォルトの選択設定は、ダイアログに表示されている機能またはサブ機能をエンドユーザーが選択する際にクリアされます。エンドユーザーが選択した機能をクリアした場合、すべてのサブ機能もクリアされます。逆に、エンドユーザーが機能について選択したすべてのサブ機能をクリアした場合、機能の選択はクリアされます。

  デフォルトで機能が選択されていないとき、サブ機能は選択されていません。デフォルトで機能のサブ機能が全く選択されていない場合、親機能をデフォルトで選択しません。デフォルト機能とサブ機能の選択設定についての詳細は、「FeatureAddItem」を参照してください。
- 機能のサイズは、選択されるまで 0 と表示されます。サイズが選択されると、実際のサイズが表示されます。
- ダイアログに表示される必要ディスク容量には、メンテナンス セットアップとアンインストーラーを使用するためにインストールするファイルのサイズが含まれます。すべてのアプリケーションコンポーネントが選択解除されても、これらのファイルのサイズは表示されたままです。
- インストールでセットアップの種類ダイアログを使用しない場合は、**SdFeatureDialog2** を呼び出す前に FeatureSetupTypeSet を呼び出して、[セットアップの種類] ビューで定義されているセットアップ タイプを指定してください。
- **SdFeatureDialog2** は、MEDIA システム変数によって指定される現在のメディアで実行されます。セットアップ初期設定中、インストールはファイル メディア ライブラリ (**Data1.cab**) に関連付けられたメディア名を MEDIA に割り当てます。

タスク ***スクリプト作成の機能を表示するには:***

1. 現在の MEDIA 値を文字列変数の形式で保存します。たとえば、szSaveMEDIAValue。
2. スクリプト作成コンポーネントセットの名前を MEDIA に割り当てます。
3. **SdFeatureDialog2** を呼び出して、エンド ユーザーの選択を取得します。
4. ステップ 1 で保存した値を MEDIA に割り当てます。必ず FeatureTransferData を呼び出す前に行います。

- 最大の機能サイズを表示できるように、機能名は必要に応じて切り詰められます。このサイズを表示するために必要な領域は、最大機能サイズ (2 GB)、現在使用されている機能サイズオプション、およびダイアログ機能情報を表示するために使用されるフォントによって異なります。機能サイズのオプションは、DialogSetInfo 関数で設定します。

  最大サイズの表示に必要な領域が決定されると、すべての機能名は残りの領域に収まるよう、必要に応じて自動的に切り詰められます。この方法では、サイズの表示に必要な領域が小さい (またはサイズが選択されていない) 機能の名前も切り詰められます。機能名がすべて確実に表示されて充分活用できるように、機能名または表示名をダイアログの有効スペースより短くしてください。
- [サブ機能の選択] ダイアログではスキンを利用できません。スキンを選択しても同じ外観となります。

SdFeatureDialog2 の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdFeatureDialog2 関数のデモンストレーションを行います。
 *
 * このスクリプト例は、ユーザーがインストールできるセットアップに含まれる
 * 各機能に必要なディスク容量を表示する
 * ダイアログ ボックスを表示します。
 *
 * コメント : このスクリプト例を実行するには、いくつかの機能および
 *          / またはファイルを持つコンポーネントを含むサブ機能を
 *          持つプロジェクトを作成 ( またはプロジェクトに挿入 ) します。
 *
 *
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
    STRING  szTitle, szMsg, svDir;
begin
    svDir   = TARGETDIR;
    szTitle = " 機能の選択 ";
    szMsg   = " コンピューターへインストールする機能を選択してください。";

    //   使用できるすべてのトップレベルの機能を表示します。
    SdFeatureDialog2 (szTitle, szMsg, svDir, "");

end;
```

SdFeatureDialogAdv

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdFeatureDialogAdv 関数は、エンドユーザーがインストールできるセットアップの機能のリスト、および各機能が使用するディスク容量を表示するダイアログを作成します。この機能は、SdFeatureDialog と全く同じです。

エンドユーザーは [参照] ボタン、そして [空きディスク容量] ボタンのをクリックしてその他のドライブの空き容量を確認し、インストール先フォルダーを変更することができます。

構文

```
SdFeatureDialogAdv ( szTitle, szMsg, svDir, szFeatures );
```

パラメーター

テーブル 31 · SdFeatureDialogAdv のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (" 機能の選択 ") を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
svDir	デフォルトで選択するフォルダーの名前を指定し、エンドユーザーが選択したフォルダーの名前を戻します。svDir によって指定されたインストール先のフォルダーは、INSTALLDIR、TARGETDIR または他のシステム変数に自動的に割り当てられません。svDir の値をセットアップに適用するには、その値を INSTALLDIR へ (InstallScript MSI インストールの場合)、または TARGETDIR (InstallScript インストールの場合) へ割り当て、システム変数が使用中の場合はスクリプト定義の変数へ割り当てなくてはなりません。 svDir によって指定されたデフォルトのフォルダーがエンドユーザーのシステムに存在しない場合、エンドユーザーが [参照] ボタンをクリックして、[フォルダーの選択] ダイアログのステップに従ってフォルダーを作成しない限り、フォルダーは作成されません。従って、デフォルトフォルダーを指定するときはいつでも、FeatureDialog が戻るときに ExistsDir を呼び出し、そのフォルダーの存在を確認しなくてはなりません。フォルダーが存在しない場合、CreateDir を呼び出して、エンドユーザーのシステムでそのフォルダーを作成します。
szFeatures	サブ機能を表示する機能の名前を指定します。トップレベルの機能をすべて表示するには、このパラメーターでヌル文字列 ("") を渡します。トップ階層の機能やサブ機能の参照方法は、「関数呼び出しで機能やサブ機能を指定する」を参照してください。 SdCFeatureDialogAdv はシステム変数 MEDIA が指定するスクリプト作成の機能セットで必要な機能を検索します。追加情報セクションを参照してください。

戻り値

テーブル 32 · SdFeatureDialogAdv の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

- 機能のサイズは、選択されるまで 0 と表示されます。サイズが選択されると、実際のサイズが表示されます。
- ダイアログに表示される必要ディスク容量には、メンテナンス セットアップとアンインストーラーを使用するためにインストールするファイルのサイズが含まれます。すべてのアプリケーションコンポーネントが選択解除されても、これらのファイルのサイズは表示されたままです。
- セットアップでセットアップの種類のダイアログを使用しない場合は、SdFeatureDialog を呼び出す前に SdFeatureDialogAdv を*必ず*呼び出して、IDE の [セットアップの種類] ビューで定義されているセットアップの種類を指定してください。
- SdFeatureDialogAdv は、MEDIA システム変数によって指定される現在のメディアで実行されます。セットアップ初期設定中、インストールはファイル メディア ライブラリ (**Data1.cab**) に関連付けられたメディア名を MEDIA に割り当てます。

タスク ***スクリプト作成の機能を表示するには:***

1. 現在の MEDIA 値を文字列変数の形式で保存します。たとえば、szSaveMEDIAValue。
2. スクリプト作成コンポーネントセットの名前を MEDIA に割り当てます。
3. SdFeatureDialogAdv を呼び出して、エンドユーザーの選択を取得します。
4. ステップ 1 で保存した値を MEDIA に割り当てます。必ず FeatureTransferData を呼び出す前に行います。

- 最大の機能サイズを表示できるように、機能名は必要に応じて切り詰められます。このサイズを表示するために必要な領域は、最大機能サイズ (2 GB)、現在使用されている機能サイズオプション、およびダイアログ機能情報を表示するために使用されるフォントによって異なります。機能サイズのオプションは、DialogSetInfo 関数で設定します。

  最大サイズの表示に必要な領域が決定されると、すべての機能名は残りの領域に収まるよう、必要に応じて自動的に切り詰められます。この方法では、サイズの表示に必要な領域が小さい (またはサイズが選択されていない) 機能の名前も切り詰められます。機能名がすべて確実に表示されて充分活用できるように、機能名または表示名をダイアログの有効スペースより短くしてください。
- [空きディスク容量] ダイアログではスキンを利用できません。スキンを選択しても同じ外観となります。
- [ディスク容量 ...] ボタンの ID は 101 です。このボタンは、有効なディスク容量ダイアログを自動的に表示します。このボタン / オプションは削除できます。ディレクトリのスタティック フィールドには、ID 851 を含みます。リスト ボックス ID は、複数選択スタイルを持ちます。
- InstallShield Professional の以前のバージョンでは、エンドユーザーが [フォルダーの選択] ダイアログで存在しないフォルダーを選択すると、フォルダーを作成するかどうかを問い合わせる確認メッセージが表示されました。このメッセージは多くのエンドユーザーに混乱を招くため、InstallShield では削除されました。

SdFeatureDialogAdv の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdFeatureDialogAdv 関数のデモンストレーションを行います。
 *
 * このスクリプト例は、ユーザーがインストールできるセットアップに含まれる
 * 各機能に必要なディスク容量を表示する
 * ダイアログ ボックスを表示します。
 *
 * コメント : このスクリプト例を実行するには、いくつかの機能および
 *          / またはファイルを持つコンポーネントを含むサブ機能を
 *          持つプロジェクトを作成 ( またはプロジェクトに挿入 ) します。
 *
 *
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
   STRING  szTitle, szMsg, svDir;
begin

   svDir   = TARGETDIR;
   szTitle = " 機能の選択 ";
   szMsg   = " コンピューターへインストールする機能を選択してください。";

   //   使用できるすべてのトップレベルの機能を表示します。
   SdFeatureDialogAdv (szTitle, szMsg, svDir, "");

end;
```

SdFeatureMult

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdFeatureMult 関数は次を表示するダイアログを作成します :

- エンドユーザーがインストールを選択できる機能とサブ機能のリストダイアログには、2 つの機能リストがあります。最初のリストで選択された機能にサブ機能が含まれる場合、サブ機能がもうひとつのリストに表示されます。
- 選択した機能への必要ディスク容量と、インストール先の空き容量。機能のサイズは、選択されるまで 0 と表示されます。エンドユーザーが機能を選択すると、実際のサイズが表示されます。
- 機能またはサブ機能の説明。エンドユーザーは機能またはサブ機能をクリックして説明を読むことができます。

構文

```
SdFeatureMult ( szTitle, szMsg, svDir, szFeatures );
```

パラメーター

テーブル 33 · SdFeatureMult のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (" 機能の選択 ") を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
svDir	デフォルトで選択するフォルダーの名前を指定し、エンドユーザーが選択したフォルダーの名前を戻します。svDir によって指定されたインストール先のフォルダーは、INSTALLDIR、TARGETDIR または他のシステム変数に自動的に割り当てられません。svDir の値をセットアップに適用するには、その値を INSTALLDIR へ (InstallScript MSI インストールの場合)、または TARGETDIR (InstallScript インストールの場合) へ割り当て、システム変数が使用中の場合はスクリプト定義の変数へ割り当てなくてはなりません。 svDir によって指定されたデフォルトのフォルダーがエンドユーザーのシステムに存在しない場合、エンドユーザーが [参照] ボタンをクリックして、[フォルダーの選択] ダイアログのステップに従ってフォルダーを作成しない限り、フォルダーは作成されません。従って、デフォルトフォルダーを指定するときはいつでも、FeatureDialog が戻るときに ExistsDir を呼び出し、そのフォルダーの存在を確認しなくてはなりません。フォルダーが存在しない場合、CreateDir を呼び出して、エンドユーザーのシステムでそのフォルダーを作成します。
szFeatures	サブ機能を表示する機能の名前を指定します。トップレベルの機能をすべて表示するには、このパラメーターでヌル文字列 ("") を渡します。トップ階層の機能やサブ機能の参照方法は、「関数呼び出しで機能やサブ機能を指定する」を参照してください。 SdFeatureMult はシステム変数 MEDIA が指定するスクリプト作成の機能セットで必要な機能を検索します。追加情報セクションを参照してください。

戻り値

テーブル 34 · SdFeatureMult の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

- デフォルトの選択設定は、ダイアログに表示されている機能またはサブ機能をエンドユーザーが選択する際にクリアされます。エンドユーザーが選択した機能をクリアした場合、すべてのサブ機能もクリアされます。逆に、エンドユーザーが機能について選択したすべてのサブ機能をクリアした場合、機能の選択はクリアされます。

  デフォルトで機能が選択されていないとき、サブ機能は選択されていません。デフォルトで機能のサブ機能が全く選択されていない場合、親機能をデフォルトで選択しません。デフォルトの機能とサブ機能の選択設定については、FeatureAddItem を参照してください。
- 機能のサイズは、選択されるまで 0 と表示されます。サイズが選択されると、実際のサイズが表示されます。
- ダイアログに表示される必要ディスク容量には、メンテナンス セットアップとアンインストーラーを使用するためにインストールするファイルのサイズが含まれます。すべてのアプリケーションコンポーネントが選択解除されても、これらのファイルのサイズは表示されたままです。
- セットアップでセットアップの種類のダイアログを使用しない場合は、SdFeatureMult を呼び出す前に FeatureSetupTypeSet を*必ず*呼び出して、IDE の [セットアップの種類] ビューで定義されているセットアップの種類を指定してください。
- SdFeatureMult は、MEDIA システム変数によって指定される現在のメディアで実行されます。セットアップ初期設定中、インストールはファイル メディア ライブラリ (**Data1.cab**) に関連付けられたメディア名を MEDIA に割り当てます。

タスク ***スクリプト作成の機能を表示するには:***

1. 現在の MEDIA 値を文字列変数の形式で保存します。たとえば、szSaveMEDIAValue。
2. スクリプト作成コンポーネントセットの名前を MEDIA に割り当てます。
3. SdFeatureMult を呼び出して、エンドユーザーの選択を取得します。
4. ステップ 1 で保存した値を MEDIA に割り当てます。必ず FeatureTransferData を呼び出す前に行います。

- 最大の機能サイズを表示できるように、機能名は必要に応じて切り詰められます。このサイズを表示するために必要な領域は、最大機能サイズ (2 GB)、現在使用されている機能サイズオプション、およびダイアログ機能情報を表示するために使用されるフォントによって異なります。機能サイズのオプションは、DialogSetInfo 関数で設定します。

  最大サイズの表示に必要な領域が決定されると、すべての機能名は残りの領域に収まるよう、必要に応じて自動的に切り詰められます。この方法では、サイズの表示に必要な領域が小さい (またはサイズが選択されていない) 機能の名前も切り詰められます。機能名がすべて確実に表示されて充分活用できるように、機能名または表示名をダイアログの有効スペースより短くしてください。
- [サブ機能の選択] ダイアログではスキンを利用できません。スキンを選択しても同じ外観となります。

SdFeatureMult の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdFeatureMult 関数のデモンストレーションを行います。
*
* このスクリプト例は、ユーザーによる
* 機能やサブ機能の選択を可能にし、機能の説明や
* 選択した機能に必要なディスク容量と
* インストール先フォルダー上の空き容量を表示します。
*
* コメント : このスクリプト例を実行するには、いくつかの機能および
*          / またはファイルを持つコンポーネントを含むサブ機能を
*          持つプロジェクトを作成 ( またはプロジェクトに挿入 ) します。
*
*
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
  STRING  szTitle, szMsg, svDir;
begin

   svDir   = TARGETDIR;
   szTitle = " 機能の選択 ";
   szMsg   = " コンピューターへインストールする機能を選択してください。";

   //   使用できるすべてのトップレベルの機能を表示します。
   SdFeatureMult (szTitle, szMsg, svDir, "");

end;
```

SdFeatureTree

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdFeatureTree 関数は、以下を含むダイアログを表示します :

- エンドユーザーがシステムに必要な機能を選択したり、システムに必要でない機能をクリアできるツリーコントロール。
- 選択された機能の説明 (機能の " 説明 " プロパティのテキスト)。
- ツリーコントロールで選択したファイル処理に必要なディスク容量、そして szDir が指定するパスのドライブ上で有効な容量。必要ディスク容量の計算には、szDir ドライブ上のクラスタサイズも考慮されます。

構文

```
SdFeatureTree ( szTitle, szMsg, szDir, szFeatures, nLevel );
```

パラメーター

テーブル 35 · SdFeatureTree のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (" 機能の選択 ") を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szDir	必要なドライブ容量および空き容量の計算に使用されるパスを指定します。このパラメーターには、必ずスクリプト変数 TARGETDIR を指定してください。
szFeatures	サブ機能を表示する機能の名前を指定します。トップレベルの機能をすべて表示するには、このパラメーターでヌル文字列 ("") を渡します。トップ階層の機能やサブ機能の参照方法は、「関数呼び出しで機能やサブ機能を指定する」を参照してください。
nLevel	ダイアログが最初に表示されたとき、何階層の機能とサブ機能をツリーで開いた状態にするかを指定します。(たとえば nLevel 2 は、ダイアログが最初に表示されたとき、3 番目とそれ以下の階層にあるサブ機能がコントロールで閉じた状態となります。)

戻り値

テーブル 36 · SdFeatureTree の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。
- ダイアログに表示される必要ディスク容量には、メンテナンス インストールとアンインストーラーを使用するためにインストールするファイルのサイズが含まれます。すべてのアプリケーションコ機能が選択解除されても、これらのファイルのサイズは表示されたままです。
- **SdFeatureTree** は、MEDIA システム変数によって指定される現在のメディアで実行されます。セットアップ初期設定中、インストールはファイル メディア ライブラリ (**Data1.cab**) に関連付けられたメディア名を MEDIA に割り当てます。

タスク ***スクリプト作成の機能を表示するには:***

1. 現在の MEDIA 値を文字列変数の形式で保存します。たとえば、szSaveMEDIAValue。
2. スクリプト作成コンポーネントセットの名前を MEDIA に割り当てます。
3. **SdFeatureTree** を呼び出して、エンドユーザーの選択を取得します。
4. ステップ 1 で保存した値を MEDIA に割り当てます。必ず **FeatureTransferData** を呼び出す前に行います。

SdFeatureTree の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdFeatureTree 関数のデモンストレーションを行います。
 *
 * このスクリプト例は、ユーザーによる
 * 機能やサブ機能の選択を可能にし、機能の説明や
 * 選択した機能に必要なディスク容量と
 * インストール先フォルダー上の空き容量を表示します。
 *
 *  コメント : このスクリプト例を実行するには、いくつかの機能および
 *          / またはファイルを持つコンポーネントを含むサブ機能を
 *          持つプロジェクトを作成 ( またはプロジェクトに挿入 ) します。
 *
 *
¥*---------------------------------------------------------------*/

#include "ifx.h"

function OnBegin()
   STRING  szTitle, szMsg, svDir;
begin

   svDir   = TARGETDIR;
   szTitle = " 機能の選択 ";
   szMsg   = " コンピューターへインストールする機能を選択してください。";

   //   最初にダイアログが表示されたとき、3 番目とそれ以下のレベルにある
   //   サブ機能がツリーコントロール上で閉じた状態で
   // 提供されているすべてのトップレベルの機能を表示します。
   SdFeatureTree (szTitle, szMsg, svDir, "", 2);

end;
```

SdFilesInUse

プロジェクト・*InstallScript MSI プロジェクトの種類は、****SdFilesInUse*** *関数をサポートします:*

*InstallScript プロジェクト、InstallScript MSI プロジェクト、および InstallScript カスタム アクションがある基本の MSI プロジェクトでは、****SdFilesInUse*** *ダイアログを手動で呼び出すことができます。ただし、インストールはロックしているファイルのアプリケーション リストを nvlistApps パラメーターを使って提供し、戻り値を適切に処理する必要があります。*

SdFilesInUse 関数は、開いた状態でファイルをロックしているアプリケーションの一覧を表示するリスト ボックスを含むダイアログを表示します。

通常、OnFilesInUse イベント ハンドラーはこのダイアログを InstallScript MSI インストールで、Windows Installer から送信された INSTALLMESSAGE_FILESINUSE メッセージへの応答として表示します。これが発生したとき、Windows Installer はインストールにファイルをロックしているアプリケーションのリストを提供します。アプリケーションのリストは szMessage パラメーターを通して OnFilesInUse イベントに渡されます。イベントは szMessage パラメーターを通して、この情報を **SdFilesInUse** イベントに渡します。その次、イベントは関数からの戻り値をイベントの戻り値として渡します。これにより、Windows Installer は適切に動作します。

構文

```
SdFilesInUse ( byval string szTitle, byval string szMsg, byval string szFilesInUse, byref LIST nvlistApps );
```

パラメーター

テーブル 37 · SdFilesInUse のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「使用中のファイル」を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szFilesInUse	InstallScript MSI インストールで、OnFilesInUse イベント ハンドラーによって **SdFilesInUse** が呼び出された場合、このパラメーターは Windows Installer によって検出されたファイルをロックしているアプリケーションのリストを指定します。関数はこの文字列を解析して、ダイアログのリスト ボックスにアプリケーションのリストを表示します。 **SdFilesInUse** を手動で呼び出す場合、空文字列 ("") をこのパラメーターに渡します。nvlistApps が有効な文字列リストであるとき、このパラメーターは無視されますので注意してください。
nvlistApps	InstallScript MSI インストールで、OnFilesInUse イベント ハンドラーによって **SdFilesInUse** が 呼び出された場合、このパラメーターは初期化されていないリスト変数 (つまり、値が 0 の変数) として指定されます。 **SdFilesInUse** を手動で呼び出す場合、ファイルをロックしていて、かつダイアログに表示する必要があるアプリケーションの文字列リストを指定します。リストの各文字列は、1 つのアプリケーションを表します。(**ListCreate** 関数と関連するリスト関数を使用して、文字列リストの作成と初期化を行います。)  ***メモ ·** このパラメーターの変数を提供する必要があります。リテラル値を指定することはできません。*

戻り値

テーブル 38 · SdFilesInUse の戻り値

戻り値	説明
IDRETRY (4)	エンドユーザーが [再試行] ボタンをクリックしたことを示します。 **SdFilesInUse** を手動で呼び出す場合、この戻り値を使ってロックされたファイルを再チェックし、必要に応じてダイアログをもう一度表示します。 **SdFilesInUse** が OnFilesInUse イベント ハンドラーによって呼び出される場合、この値はイベント ハンドラーによって返され、Windows Installer が処理します。
IDIGNORE (5)	エンドユーザーが [無視する] ボタンをクリックしたことを示します。

テーブル 38 ・ SdFilesInUse の戻り値 (続き)

戻り値	説明
IDCANCEL	エンドユーザーが [終了] ボタンをクリックしたことを示します。 ***メモ ・** 他のスクリプト ダイアログとは違い、このダイアログは、ユーザーが [終了] ボタンをクリックしたとき、OnCanceling イベント ハンドラーを呼び出しませんので注意してください。したがって、この関数を手動で呼び出す場合、[セットアップの取り消し] ダイアログを表示するには、この戻り値を手動で処理する必要があります。*

SdFilesInUse の例

***プロジェクト ・** InstallScript MSI プロジェクトの種類は、**SdFilesInUse** 関数をサポートします :*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 *  SdFilesInUse 関数のデモンストレーションを行います。SdFilesInUse は
 * ファイルをロックしているアプリケーションをリストにするリスト ボックス ダイアログを
 *
 *
 * Notepad.exe が実行中でユーザー [ 再試行 ] をクリックしたときに
 * SdFilesInUse を表示し続けます。
 *
¥*---------------------------------------------------------------------*/

#include "ifx.h"


function OnFirstUIBefore( )
    // ... 他の変数 ...
    LIST   listID;
    NUMBER nReturn;
begin

    // ... 他のダイアログを表示し ...

// Notepad.exe が実行中の場合のみ、ファイルロック中ダイアログを表示します
if (Is(FILE_LOCKED, WINDIR ^ "Notepad.exe")) then
    // 文字列リストを作成します。
    listID = ListCreate(STRINGLIST);

    // エラーが発生した場合、それをレポートして終了させます。
    if (listID = LIST_NULL) then
        MessageBox ("リストを作成できませんでした。", SEVERE);
        abort;
    endif;

    ListAddString(listID, "Notepad.exe", AFTER);
```

```
    AskAgain:

    // [ ファイルの使用中 ] ダイアログを表示します。
    nReturn = SdFilesInUse("", " 次のアプリケーションが現在ロックしているファイルです。", "", listID);

    // ユーザーが RETRY をクリックした場合、ファイルロック中ダイアログを再度表示します ;
    // その他の場合、ファイル転送処理に任せます。これには再起動が必要な場合があります。
    if (nReturn = IDRETRY) then
        if (Is(FILE_LOCKED, WINDIR ^ "Notepad.exe")) then
            goto AskAgain;
        endif;
    endif;

    // メモリからリストを削除します。
    ListDestroy(listID);
endif;

end;
```

SdFinish

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdFinish 関数は、エンド ユーザーにユーザーにインストールの完了を通知し、情報またはオプションを示すダイアログを表示します。SdFinish のダイアログは、メッセージおよびチェック ボックスの選択オプションをそれぞれ 2 つまで表示します。たとえば、エンドユーザーに対して、README ファイルの表示またはアプリケーションの起動のいずれかを選択するようなオプションを提供できます。

メッセージおよびチェック ボックスの説明に製品名を挿入するには、szMsg1、szMsg2、szOpt1、および szOpt2 で渡された文字列の %P プレースホルダーを使用します。

メモ・SdFinish には、セットアップを終了して、エンドユーザーのコンピューターを再起動するオプションはありません。SdFinish が戻されると、セットアップは続けて実行します。エンドユーザーに再起動オプションを提供するには、代わりに SdFinishReboot を呼び出してください。

構文

```
SdFinish (szTitle, szMsg1, szMsg2, szOpt1, szOpt2, bvOpt1, bvOpt2);
```

パラメーター

テーブル 39 ・ SdFinish のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「セットアップの完了」を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg1	ダイアログの上部に表示するメッセージを指定します。インストールが完了したことをユーザーに通知するデフォルトのインストラクションを表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg2	ダイアログの下部に表示するメッセージを指定します。デフォルトのタイトル「[完了] ボタンをクリックして、セットアップを終了してください。」を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szOpt1	最初のチェック ボックスの横に表示するテキストを指定します。このパラメーターでヌル文字列 ("") を渡すと、チェック ボックスを非表示にできます。
szOpt2	2 番目のチェック ボックスの横に表示するテキストを指定します。このパラメーターでヌル文字列 ("") を渡すと、チェック ボックスを非表示にできます。
bvOpt1	最初のチェック ボックスの選択状態 (TRUE または FALSE) を返します。
bvOpt2	2 番目のチェック ボックスの選択状態 (TRUE または FALSE) を返します。

戻り値

テーブル 40 ・ SdFinish の戻り値

戻り値	説明
NEXT (1)	[完了] ボタンがクリックされたことを示します。

***メモ ・** SdFinish はインストール作業の完了を知らせるため、[戻る] ボタンは無効です。*

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdFinish の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdFinish 関数のデモンストレーションを行います。
 *
 * メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
 *      [ サポート ファイル / ビルボード ] ビューの Windows Notepad 実行可能ファイルと
 *      完全修飾名と有効なテキスト ファイルを
 *      参照するように設定します。
 *
¥*---------------------------------------------------------------------*/

// README ファイル へテキスト ファイルの名前を割り当てます
#define NOTEPAD WINDIR ^ "Notepad.exe"
#define README SUPPORTDIR ^ "ReadMe.txt"

#include "Ifx.h"

function OnBegin()
    STRING   szProductName, szTitle;
    STRING   szMsg1, szMsg2, szOpt1, szOpt2;
    BOOL     bvOpt1, bvOpt2;
    NUMBER nReturn;
begin

    // 製品名を %P プレースホルダーの代わりに設定します。
    szProductName = " 自分のアプリケーション ";
    SdProductName (szProductName);

    // SdFinish に渡されるセットアップ パラメーター。
    szTitle = "SdFinish の例 ";
    szMsg1  = "%P セットアップがほぼ完了しました。¥n" +
              " 下のオプションを選択してください。";
    szMsg2   = "[ 完了 ] をクリックして %P セットアップを終了します。";
    szOpt1  = "README を読む。";
    szOpt2  = "%P を起動する。";

    // SdFinish ダイアログを表示します。
    SdFinish (szTitle, szMsg1, szMsg2, szOpt1, szOpt2, bvOpt1, bvOpt2);

    if (bvOpt1) then
        // readme ファイルを表示します。
        LaunchAppAndWait (NOTEPAD, README, WAIT);
    endif;

    if (bvOpt2) then
        // この例は実際にアプリケーションをインストールしないので
        // メッセージボックスは、通常 LaunchApp の呼び出しが行われる
        // この場所に表示されます。
        // 例えば :
        //   LaunchApp (TARGETDIR ^ "MyApp.exe","");

        SprintfBox (INFORMATION, szTitle, "%s をここで起動します。", szProductName);
    endif;
```

```
end;
```

SdFinishEx

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdFinishEx 関数は SdFinish または SdFinishReboot を呼び出して、エンドユーザーに対してインストールの完了を通知し、情報またはオプションを提供するダイアログを表示します。BATCH_INSTALL システム変数が FALSE (セットアップ中にロックファイルが検出されなかったことを示す) の場合、SdFinishEx が SdFinish を呼び出して、ダイアログを表示します。BATCH_INSTALL の値がゼロ以外の場合は、SdFinishEx が SdFinishReboot を呼び出して、ダイアログを表示します。

メッセージおよびチェック ボックスの説明に製品名を挿入するには、szMsg1、szMsg2、szOpt1、および szOpt2 で渡された文字列の %P プレースホルダーを使用します。

構文

```
SdFinishEx (szTitle, szMsg1, szMsg2, szOpt1, szOpt2, bvOpt1, bvOpt2);
```

パラメーター

パラメーターは、SdFinish のパラメーターと同一です。BATCH_INSTALL が TRUE の場合、これらのパラメーターは無視され、SdFinishReboot("", "", SYS_BOOTMACHINE, "", 0) が呼び出されます。

戻り値

テーブル 41 ・ SdFinishEx の戻り値

戻り値	説明
0	SdFinish が呼び出されたことを示します。
NEXT (1)	SdFinishReboot が呼び出され、ユーザーがコンピューターの再起動を選択しなかったことを示します。
< 0	SdFinishReboot が呼び出され、ユーザーがコンピューターの再起動を選択したが、再起動に失敗したことを示します。

追加情報

デフォルト以外のテキストを SdFinishReboot ダイアログに表示する場合、SdFinishEx を呼び出さないで下さい。次のようなスクリプトコードを挿入します。

```
if (!BATCH_INSTALL) then
   SdFinish( szTitle, szMsg1, szMsg2,
      szOption1, szOption2, bOpt1, bOpt2);
else
   SdFinishReboot( szRebootTitle, szRebootMsg1,
```

```
        SYS_BOOTMACHINE, szRebootMsg2, 0 );
endif;
```

SdFinishEx の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdFinishEx 関数のデモンストレーションを行います。
 *
 * SdFinishEx 関数は、SdFinish または SdFinishReboot
 * を呼び出して、エンドユーザーに
 * インストールが完了したことを通知し、ユーザー情報または
 * オプションを提供するダイアログを表示します。システム変数 BATCH_INSTALL が
 * FALSE (0) に等しく、ロックされているファイルが
 * セットアップ中に発生しなかった場合、SdFinishEx は SdFinish を呼び出してダイアログを表示
 * 表示されます。BATCH_INSTALL が非ゼロ値に等しかった場合、
 * SdFinishEx は SdFinishReboot を呼び出してそのダイアログを表示します。
 *
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
    STRING szTitle, szMsg1, szMsg2, szOption1, szOption2;
    NUMBER bOpt1, bOpt2;
begin

    bOpt1  = FALSE;
    bOpt2  = FALSE;
    szMsg1 = SdLoadString(IFX_SDFINISH_MSG1);
    SdFinishEx(szTitle, szMsg1, szMsg2, szOption1, szOption2, bOpt1, bOpt2);
end;
```

SdFinishReboot

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdFinishReboot 関数は、インストール終了時にインストールの完了を知らせ、エンド ユーザーがシステムの再起動を選択できます。システムの再起動によって、**Autoexec.bat**、**Config.sys** および ini ファイルへの変更が可能となります。

SdFinishReboot ダイアログには、静的テキスト フィールドに最高 2 つのメッセージが表示されます。パラメーター szMsg1、szMsg2 を使用して、これらのフィールドの値を設定します。フィールドに表示されるメッセージに製品名を挿入するには、 s zMsg1 と szMsg2 で渡される文字列に %P プレースフォルダーを配置します。

構文

SdFinishReboot (szTitle, szMsg1, nDefOption, szMsg2, nReserved);

パラメーター

テーブル 42 · SdFinishReboot のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「セットアップの完了」を表示するには、このパラメーターでヌル文字列 (“”) を渡します。
szMsg1	インストールの終了をユーザーに通知するためのダイアログの上に表示するテキストを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 (“”) を渡します。
nDefOption	デフォルトのラジオボタンオプションの選択項目を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · SYS_BOOTMACHINE Ñ セットアップが終了したときに、コンピューターを再起動します。 · 0 Ñ コンピューターを再起動しません。
szMsg2	ユーザーに対して次の操作について説明する、ダイアログの下に表示するテキストを入力します。デフォルト指示を表示するには、ヌル文字列 (“”) を渡してください。
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。

戻り値

テーブル 43 · SdFinishReboot の戻り値

戻り値	説明
WILL_REBOOT	ユーザーがシステムを再起動を選択したことを示します。
NEXT (1)	ユーザーがシステムの再起動、または Windows の再起動を選択しなかったことを示します。
< 0	システムの再起動または Windows の再起動が選択されましたが、再起動に失敗しました。

***メモ · SdFinishReboot** はインストール作業の完了を知らせるため、[戻る] ボタンは無効です。*

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。
- インストールでは、別のインスタンスのインストールが実行中、コンピューターを*再起動させない*よう最大限の試みが行われます。このため、**SdFinishReboot** が Windows またはシステムの再起動を行う前に、必ず インストールのその他すべてのインスタンスをシャットダウンする必要があります。また、エンドユーザーに対してもメッセージを表示して、インストール完了後にシステムを再起動する際には、まず、他のアプリケーションがすべて終了していることを確認するよう指示してください。
- インストールは、システムが再度起動されたとき、ロックされている .dll ファイルおよび .exe ファイルが更新済みの状態になっていることを自動的に確認します。

SdFinishReboot の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdFinishReboot ダイアログのデモンストレーションを行います。
 *
 * 警告 : 再起動オプションを選択した場合、コンピューターが再起動
 *        します。このオプションを選択する時、すべてのファイルが保存済みであることを
 *        確認してください。
 *
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
    STRING szTitle, szMsg1, szMsg2;
    NUMBER nDefOption, nReserved;
begin

    // SdFinishReboot へのパラメーターとして渡す変数をセットアップします。
    szTitle   = "SdFinishReboot の例 ";
    szMsg1    = " セットアップが %P のインストールを完了しました。";

    // nDefOptions - デフォルトのラジオボタンオプションの選択項目を指定します。
    // SYS_BOOTMACHINE - セットアップが終了した際に、コンピューターを再起動します。
    // 0 - コンピューターを再起動しません。
    nDefOption   = 0;
    szMsg2    = "[ 完了 ] をクリックして %P セットアップを完了します。";
    nReserved = 0;

    // SdFinishReboot ダイアログを表示します。
    if (SdFinishReboot (szTitle, szMsg1, nOption, szMsg2, nReserved) < 0) then
```

```
    // ユーザーがシステムを再起動するか、
    //Windows を再起動することを選択しましたが、再起動は失敗しました。
      MessageBox ("SdFinishReboot が失敗しました。", SEVERE);
  endif;

end;
```

SdFinishUpdate

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdFinishUpdateEx 関数は **SdFinishUpdate** 関数に優先します。**SdFinishUpdate** は次を呼び出します :

SdFinishUpdateEx(szTitle, szMsg1, szMsg2, "", "", bDefOption);

SdFinishUpdate 関数は、インストールの終わりに、それが完了したことを示すダイアログを表示します。このダイアログには、アプリケーションのアップデートを確認するためのオプションが含まれます。

***メモ・SdFinishUpdate** はアップデートの確認を行いません。アップデートの確認をするためには、InstallScript コードに FlexNet Connect API の呼び出しを追加します。詳細は、FlexNet Connect SDK ドキュメントを参照してください。*

構文

SdFinishUpdate (szTitle, szMsg1, szMsg2, bDefOption);

SdFinishUpdateEx

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdFinishUpdateEx 関数は、インストールの終わりに、それが完了したことを示すダイアログを表示します。このダイアログには、アプリケーションのアップデートを確認するためのオプションが含まれます。

***メモ・SdFinishUpdateEx** はアップデートの確認を行いません。アップデートの確認をするためには、InstallScript コードに FlexNet Connect API の呼び出しを追加します。詳細は、FlexNet Connect SDK ドキュメントを参照してください。*

構文

SdFinishUpdateEx (szTitle, szMsg1, szMsg2, szOpt1, szOpt2, bDefOption);

パラメーター

テーブル 44 · SdFinishUpdateEx のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「セットアップの完了」を表示するには、このパラメーターでヌル文字列 (“”) を渡します。
szMsg1	インストールの終了をユーザーに通知するためのダイアログの上に表示するテキストを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 (“”) を渡します。
szMsg2	エンドユーザーに次の操作を指示するためのダイアログの一番下に表示するテキストを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 (“”) を渡します。
szOpt1	最初のラジオ ボタンの横に表示するテキストを指定します。デフォルトの指示 (“はい、プログラムのアップデートを確認します。(推奨)¥n 処理を続行する前に、インターネットへ接続されていることを確認してください。”) を表示するには、このパラメーターでヌル文字列 (“”) を渡します。
szOpt2	2 番目のラジオ ボタンの横に表示するテキストを指定します。デフォルトの指示 (“ いいえ、確認しません。”) を表示するには、このパラメーターでヌル文字列 (“”) を渡します。
bDefOption	デフォルトで選択したオプションボタンを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · TRUE Ñ デフォルトの選択にする最初のオプションボタンを指定します。 · FALSE Ñ デフォルトの選択にする 2 番目のオプション ボタンを指定します。

戻り値

テーブル 45 · SdFinishUpdateEx の戻り値

戻り値	説明
TRUE	**[プログラムアップデートを確認する]** オプションボタンが選択されたことを示します。
FALSE	**[この手順をスキップする]** オプションが選択されたことを示します。

メモ · *SdFinishUpdateEx はインストール作業の完了を知らせるため、[戻る] ボタンは無効です。*

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdFinishUpdateReboot

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

SdFinishUpdateReboot 関数は、インストールの終わりに、それが完了したことを示すダイアログを表示します。ダイアログは、エンド ユーザーに対してシステムの再起動オプションを提供し、アプリケーションのアップデートも確認します。システムの再起動によって、**Autoexec.bat**、**Config.sys** および ini ファイルへの変更が有効となります。

***メモ・SdFinishUpdateReboot** はアップデートの確認を行いません。アップデートの確認をするためには、InstallScript コードに FlexNet Connect API の呼び出しを追加します。詳細は、FlexNet Connect SDK ドキュメントを参照してください。*

構文

SdFinishUpdateReboot (szTitle, szMsg1, nDefOption, szMsg2, nChkUpdate, nReserved);

パラメーター

テーブル 46 ・ SdFinishUpdateReboot のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「セットアップの完了」を表示するには、このパラメーターでヌル文字列 (″″) を渡します。
szMsg1	インストールの終了をエンド ユーザーに通知するためのダイアログの上に表示するテキストを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 (″″) を渡します。
nDefOption	デフォルトのラジオ ボタン選択を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ SYS_BOOTMACHINE Ñ セットアップが終了したときに、コンピューターを再起動します。 ・ 0 Ñ コンピューターを再起動しません。
szMsg2	エンド ユーザーに対して次の操作について説明する、ダイアログの下に表示するテキストを入力します。デフォルト指示を表示するには、ヌル文字列 (″″) を渡してください。
nvChkUpdate	FlexNet Connect チェック ボックスの状態を戻します : ・ 0 Ñ チェック ボックスが選択されていません。アプリケーションのアップデートを確認しません。 ・ 0 以外 Ñ チェック ボックスが選択されています。アプリケーションアップデートを確認します。
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。

戻り値

テーブル 47 ・ SdFinishUpdateReboot の戻り値

戻り値	説明
WILL_REBOOT	エンドユーザーがシステムの再起動を選択したことを示します。
NEXT (1)	エンドユーザーがシステムの再起動、または Windows の再起動を選択しなかったことを示します。
< 0	エンドユーザーがシステムの再起動または Windows の再起動が選択されましたが、再起動に失敗したことを示します。

メモ ・ *SdFinishUpdateReboot はインストール作業の完了を知らせるため、[戻る] ボタンは無効です。*

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdFinishUpdateReboot の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdFinishUpdateReboot ダイアログのデモンストレーションを行います。
 *
 * このダイアログは、インストレーションが完了したことを通知し、ユーザーに対して
 * システムの再起動やアプリケーションのアップデートの確認オプションを提供するのに
 * 提供するのにインストールの最後で利用されます。
 *
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプ用に Ifx.h を含みます
#include "ifx.h"

export prototype ExFn_SdFinishUpdateReboot(HWND);

function ExFn_SdFinishUpdateReboot(hMSI)
    STRING szTitle, szMsg1, szMsg2;
    NUMBER nDefOption, nChkUpdate, nReserved;
begin

    szTitle = "SdFinishUpdateReboot の例 ";
    szMsg1  = "";
    szMsg2  = "";

// nDefOption - デフォルトで選択するオプション ボタンを指定します。
    // 1 - はい、プログラムのアップデートを確認します
    // 0 - いいえ、このステップを飛ばします
    nDefOption = 1;

    // nChkUpdate - FlexNet Connect チェック ボックスのデフォルトの状態を指定します。
    // nChkUpdate - FlexNet Connect チェック ボックスの状態を戻します。
    // 0 - チェック ボックスが選択されていません。アプリケーションのアップデートを確認しません。
    // 0 以外 - チェック ボックスが選択されています。アプリケーションアップデートを確認します。
    nChkUpdate = 1;
    nReserved = 0;
    SdFinishUpdateReboot ( szTitle, szMsg1, nDefOption, szMsg2, nChkUpdate, nReserved );

end;
```

SdGeneralInit

SdGeneralInit 関数は、[次へ]、[戻る]、および [キャンセル] ボタンの状態を有効または無効に設定する処理を含む、標準ダイアログの初期化を行います。この関数はまた、コントロール ID 700 から 724、および 202 を含むスタティック コントロール上のすべての %P、%VS、および %VI インスタンスを IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および IFX_INSTALLED_DISPLAY_VERSION で置換します。

構文

SdGeneralInit (szDialog, hwndDialg, nUnused, szUnused);

パラメーター

テーブル 48 · SdGeneralInit のパラメーター

パラメーター	説明
szDialog	EzDefineDialog または DefineDialog で作成されたダイアログの名前。
hwndDialog	現在のダイアログへのハンドル。これは、CmdGetHwndDlg の呼び出しで取得します。
nUnused	未使用のパラメーター、0 を渡します。
szUnused	未使用のパラメーター、空白文字列 ("") を渡します。

戻り値

なし。

SdGeneralInit の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdGeneralInit のデモンストレーション。この関数は、
* EzDefineDialog 関数を使って作成されたダイアログの初期化に使います。
*
* このスクリプトは ビットマップを表示するシンプルなカスタムダイアログを
* 開きます。ダイアログは次の 3 つのボタンで閉じることが
* [ 戻る ]、[ 次へ ]、および [ キャンセル ]。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SdAskOptions が表示する
* InstallShield 標準ダイアログです。このダイアログは
* インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
* カスタム ダイアログとしてスクリプトで利用することが
* 表示されます。
*
* このカスタムダイアログを利用するため、スクリプトはまず EzDefineDialog を
* 呼び出して定義します。そしてダイアログを表示して
* WaitOnDialog を呼び出してダイアログ イベントを取得します。その後、
```

```
＊イベントがダイアログの処理を終了するとき、EndDialog が呼び出されて
＊ダイアログを閉じます。次いで、ReleaseDialog への呼び出しによって、
＊メモリからダイアログがリリースされます。
＊
¥*---------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID     12027   // ダイアログ自身の ID
#define RES_PBUT_NEXT         1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL       9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK        12   //[ 戻る ] ボタンの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_EzDefineDialog(HWND);

function ExFn_EzDefineDialog(hMSI)
   STRING  szDialogName, szDLLName, szDialog;
   NUMBER  nDialog, nResult, nCmdValue;
   BOOL    bDone;
   HWND    hInstance, hwndParent, hwndDlg;
begin

   // このインストールでカスタム ボックスを認識するための名前を指定します。
   szDialogName = "CustomDialog";

   // ダイアログを定義します。ヌル文字列を 2 番目のパラメーターで渡し、
   // _isuser.dll または _isres.dll からダイアログを取得します。ヌル文字列を
   // 3 番目のパラメータでヌル文字列を渡します。これは、ダイアログが
   // 4 番目のパラメーターにある ID によって識別されるためです。
   nResult = EzDefineDialog (szDialogName, "", "", RES_DIALOG_ID);

   if (nResult < 0) then
      // エラーを報告し、終了します。
      MessageBox ("ダイアログの定義エラー", SEVERE);
      abort;
   endif;

   // while ループを制御するのに使われるインジケーターを初期化します。
   bDone = FALSE;

   // 完了するまでループします。
   repeat

      // ダイアログを表示して次のダイアログ イベントを戻します。
      nCmdValue = WaitOnDialog(szDialogName);

      // イベントに応答します。
      switch (nCmdValue)
         case DLG_CLOSE:
            // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
            Do (EXIT);
         case DLG_ERR:
            MessageBox ("ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
            abort;
         case DLG_INIT:
            // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
            // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
```

```
            // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
            // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
            hwndDlg = CmdGetHwndDlg (szDialogName);
            SdGeneralInit(szDialogName, hwndDlg, 0, "");
        case RES_PBUT_CANCEL:
            // ユーザーが [ キャンセル ] ボタンをクリックしました。
            Do (EXIT);
        case RES_PBUT_NEXT:
            bDone = TRUE;
        case RES_PBUT_BACK:
            bDone = TRUE;
        if (SdIsStdButton( nCmdValue ) && SdDoStdButton( nCmdValue )) then
            bDone = TRUE;
        endif;
    endswitch;

    until bDone;

    // ダイアログを閉じます
    EndDialog (szDialogName);

    // メモリからダイアログを解放します。
    ReleaseDialog (szDialogName);

end;
```

SdInit

SdInit 関数は、必要なリソース文字列のロード、インストールのウィンドウが最小化されている場合のウィンドウの復元、および Sd ダイアログでの Windows 95 スタイルのチェック ボックスの指定を行い、インストールで Sd ダイアログ 関数を呼び出すための準備を行います。

メモ・この関数は、それぞれの Sd 関数によって自動的に呼び出されます。標準ダイアログ関数を呼び出す前にスクリプトで DialogSetInfo が呼び出される場合以外は、SdInit を明示して呼び出す必要はありません。その場合は、DialogSetInfo を呼び出す前にスクリプトで SdInit を呼び出す必要があります。呼び出さないと、DialogSetInfo への呼び出しは有効になりません。

構文

```
SdInit ( );
```

パラメーター

この関数にパラメーターは使用できません。

戻り値

テーブル 49 - SdInit の戻り値

戻り値	説明
0	Sd ダイアログ関数の呼び出しのセットアップが初期化されたことを示します。
1	Sd ダイアログ関数呼び出しのセットアップがすでに初期化されていることを示します。

SdInit の例

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdInit 関数のデモンストレーションを行います。
*
* SdInit は、それぞれの Sd 関数によって自動的に呼び出されます。
* Sd ダイアログ関数を呼び出す前に DialogSetInfo をスクリプトが呼び出さない限り、
* SdInit を明示的に呼び出す必要はありません。
* スクリプトで DialogSetInfo を呼び出した場合は、DialogSetInfo を呼び出す前に
* SdInit を呼び出す必要があります。そうでなければ、DialogSetInfo への呼び出しは有効になりません。
*
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
    STRING szInfoString;
begin

    // 標準ダイアログ関数を呼び出すため、セットアップを初期化します。
    SdInit ();

    // 関数選択用のチェック ボックススタイルを設定します。
    DialogSetInfo ( DLG_INFO_CHECKSELECTION, szInfoString, CHECKBOX );

    // 機能選択を取得します。
    SdFeatureDialog2 ( "", "", TARGETDIR, "" );

end;
```

SdLicense

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

- *InstallScript*
- *InstallScript MSI*

SdLicenseEx 関数は **SdLicense** 関数に優先します。

SdLicense 関数は、複数行編集フィールドに使用許諾契約書が記載されたダイアログを表示します。ライセンス同意書は、パラメーター szLicenseFile で指定されるテキストファイルに格納されます。

エンド ユーザーはスクロール アップ / ダウンして、使用許諾契約書を読むことができます。エンド ユーザーは [はい]、[いいえ]、あるいは有効な場合は [戻る] ボタンのいずれかを選択しなくてはなりません。通常は最初に表示されるダイアログなので、[戻る] ボタンを無効にすることもできます。ユーザーが [はい] を選択すると、インストールは続行します。ユーザーが [いいえ] を選択した場合、インストールは ExitSetup ダイアログを表示します。

構文

```
SdLicense (szTitle, szMsg, szQuestion, szLicenseFile);
```

パラメーター

テーブル 50 ・ SdLicense のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (「ソフトウェア使用許諾契約書」) を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg	複数行編集フィールドの上のスタティック テキストフィールドに表示するメッセージを指定します。デフォルトの指示を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szQuestion	複数行編集フィールドの下のスタティック テキスト フィールドに表示するテキストを指定します。ユーザが [はい] か [いいえ] を選択することによって応答する質問をここに配置します。デフォルトの指示を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szLicenseFile	使用許諾契約書書を含む ANSI テキストファイルの名前を指定します。このファイルは、[サポートファイル / ビルボード] ビューで適切な言語フォルダーに追加する必要があります。szLicenseFile は、引用符で囲んだ完全修飾名、または UNC パスを入力して指定することもできます。. ***メモ・*** *このファイルはテキスト ファイル (.txt) でなくてはなりません。*

戻り値

テーブル 51 ・ SdLicense の戻り値

戻り値	説明
YES (1)	エンドユーザーが、[はい] ボタンを選択しました。
BACK (12)	ユーザーが、[戻る] ボタンを選択したことを示します。

メモ・ *エンドユーザーが [いいえ] ボタンをクリックした場合は、ExitSetup ダイアログが表示されるので、関数は NO を戻しません。*

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdLicense の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdLicense 関数のデモンストレーションを行います。
 *
 * このスクリプトは SdLicense を呼び出し、使用許諾契約を表示してユーザーに対して
 * ユーザーに対して契約に同意するか同意しないかを問い合わせます。
 *
 * メモ : 使用許諾契約書は定数 LICENSE_PATH が指定する場所に
 *     格納してある RTF ファイルから
 *     読み込まれます。このスクリプトを実行する前に、定数が
 ターゲットシステムで既存のテキスト ファイルを参照するように
 *      参照するように設定してください。
 *
¥*---------------------------------------------------------------------*/

#define LICENSE_PATH SUPPORTDIR ^ "License.txt"
#define TITLE "SdLicense の例 "

#include "Ifx.h"

function OnBegin()
    STRING szMsg, szQuestion;
begin

    // セットアップダイアログで [ 戻る ] ボタンを無効にします。
    Disable(BACKBUTTON);

    // SdLicense へのパラメーターとして渡す変数をセットアップします。
    szMsg      = " 次の使用許諾契約をお読みください。" +
                 " スクロールバーを利用して ¥n 残りの契約内容をご覧下さい。";
    szQuestion = " 契約に同意するには [ はい ] を選択します。¥n" +
                 " セットアップをキャンセルするには [ いいえ ] を選択します。";

    // SdLicense ダイアログを表示します。
    if (SdLicense (TITLE, szMsg, szQuestion, LICENSE_PATH) = YES) then
        MessageBox (" インストールを続行します。", INFORMATION);
    endif;

end;
```

SdLicense2

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdLicense2Ex 関数は、**SdLicense2** 関数に優先します。

SdLicense2 関数は、複数行編集フィールドに使用許諾契約書が記載されたダイアログを表示します。ライセンス同意書は、パラメーター szLicenseFile で指定されるテキストファイルに格納されます。

エンド ユーザーはスクロール アップ / ダウンして、使用許諾契約書を読むことができます。インストールの続行を可能にする [次へ] ボタンを有効にするためには、エンド ユーザーは上のオプションを選択しなくてはなりません。通常は最初に表示されるダイアログなので、[戻る] ボタンを無効にすることもできます。

構文

```
SdLicense2 ( szTitle, szOpt1, szOpt2, szLicenseFile, bLicenseAccepted );
```

パラメーター

テーブル 52 · SdLicense2 のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (「使用許諾契約書」) を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szOpt1	上のオプションボタンの隣にあるスタティック テキスト フィールドに表示するテキストを指定します。デフォルトのテキスト (「ソフトウェア使用許諾契約に同意します」) を表示するには、このパラメーター内にヌル文字列 ("") を渡します。
szOpt2	下のオプション ボタンの隣にあるスタティック テキスト フィールドに表示するテキストを指定します。デフォルトのテキスト (「ソフトウェア使用許諾契約に同意しません」) を表示するには、このパラメーター内にヌル文字列 ("") を渡します。
szLicenseFile	使用許諾契約書書を含む ANSI テキストファイルの名前を指定します。このファイルは、[サポートファイル / ビルボード] ビューで適切な言語フォルダーに追加する必要があります。szLicenseFile は、引用符で囲んだ完全修飾名、または UNC パスを入力して指定することもできます。. ***メモ ·** このファイルはテキスト ファイル (.txt) でなくてはなりません。*
bLicenseAccepted	デフォルトで選択するオプション ボタンを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · TRUE Ñ 上のオプション ボタンをデフォルトの選択として指定します。 · FALSE Ñ 下のオプション ボタンをデフォルトの選択として指定します。

戻り値

テーブル 53 · SdLicense2 の戻り値

戻り値	説明
NEXT	エンド ユーザーが上のオプションボタンと [次へ] ボタンを選択しました。
BACK	エンドユーザーが、[戻る] ボタンを選択しました。
くISERR_SUCCESS	ダイアログを表示することができませんでした。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdLicense2 の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 *  SdLicense2 関数のデモンストレーションを行います。
 *
 * このスクリプトは SdLicense2 を呼び出し、使用許諾契約書を表示して
 * ユーザーに対して契約に同意するか同意しないかを問い合わせます。
 *
 * メモ : 使用許諾契約書は定数 LICENSE_PATH が指定する場所に
 *      格納してある RTF ファイルから
 *      読み込まれます。このスクリプトを実行する前に、定数が
 ターゲットシステムで既存のテキスト ファイルを参照するように
 *       設定してください。
 *
¥*---------------------------------------------------------------------*/



#define LICENSE_PATH "License.txt"
#define TITLE "SdLicense2 の例 "

#include "ifx.h"

function OnBegin()
begin

  // セットアップダイアログで [ 戻る ] ボタンを無効にします。
  Disable(BACKBUTTON);


  // SdLicense2 ダイアログを表示します。
  if (SdLicense2 (TITLE, "", "", LICENSE_PATH, FALSE) = NEXT) then
    MessageBox ("インストールを続行します。", INFORMATION);
  endif;

end;
```

SdLicense2Ex

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdLicense2Ex 関数は、複数行編集フィールドに使用許諾契約書が記載されたダイアログを表示します。使用許諾契約書は、パラメーター szLicenseFile で指定されるテキスト ファイル (.txt) またはリッチテキスト形式ファイル (.rtf) で格納されます。

エンド ユーザーはスクロール アップ / ダウンして、使用許諾契約書を読むことができます。インストールの続行を可能にする [次へ] ボタンを有効にするためには、エンド ユーザーは上のオプションを選択しなくてはなりません。通常は最初に表示されるダイアログなので、[戻る] ボタンを無効にすることもできます。

構文

```
SdLicense2Ex (byval string szTitle, byval string szOpt1, byval string szOpt2, byval string szLicenseFile, byval bool bLicenseAccepted, byval
    bool bRtf);
```

パラメーター

テーブル 54 · SdLicense2Ex のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (「使用許諾契約書」) を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szOpt1	上のオプションボタンの隣にあるスタティック テキスト フィールドに表示するテキストを指定します。デフォルトのテキスト (「ソフトウェア使用許諾契約に同意します」) を表示するには、このパラメーター内にヌル文字列 ("") を渡します。
szOpt2	下のオプション ボタンの隣にあるスタティック テキスト フィールドに表示するテキストを指定します。デフォルトのテキスト (「ソフトウェア使用許諾契約に同意しません」) を表示するには、このパラメーター内にヌル文字列 ("") を渡します。
szLicenseFile	使用許諾契約書を含む ANSI テキストファイル、または .rtf ファイルの名前を指定します。このファイルは、[サポートファイル / ビルボード] ビューで適切な言語フォルダーに追加する必要があります。szLicenseFile は、引用符で囲んだ完全修飾名、または UNC パスを入力して指定することもできます。. ***メモ ·** .rtf ファイルを使用する場合、ファイルのサイズ制限は 16 MB です。*
bLicenseAccepted	デフォルトで選択するオプション ボタンを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · TRUE Ñ 上のオプション ボタンをデフォルトの選択として指定します。 · FALSE Ñ 下のオプション ボタンをデフォルトの選択として指定します。
bRtf	ダイアログにリッチ テキスト編集コントロールを使用するかどうかを示します。選択可能なオプションは以下のとおりです : · TRUE Ñ リッチ テキスト編集コントロールを使用します。このオプションを選択する場合、szLicenseFile に指定するファイルは .rtf ファイルでなくてはなりません。 · FALSE Ñ 標準編集コントロールを使用します。このオプションを選択する場合、szLicenseFile に指定するファイルは .txt ファイルでなくてはなりません。

戻り値

テーブル 55 · SdLicense2Ex の戻り値

戻り値	説明
NEXT	エンド ユーザーが上のオプションボタンと [次へ] ボタンを選択しました。
BACK	エンドユーザーが、[戻る] ボタンを選択しました。
くISERR_SUCCESS	ダイアログを表示することができませんでした。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdLicense2Rtf

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdLicense2Ex 関数は、**SdLicense2Rtf** 関数に優先します。

SdLicense2Rtf 関数は、複数行編集フィールドに使用許諾契約書が記載されたダイアログを表示します。使用許諾契約書は、パラメーター szLicenseFile で指定されるテキスト ファイルまたはリッチテキスト形式ファイル (.rtf) に格納されます。

ユーザーは使用許諾契約を読むために上下にスクロールすることができ、[次へ] ボタンを有効にしてセットアップを続行するためには上のオプションボタンを選択しなくてはなりません。これは通常、最初に表示されるダイアログなので、[戻る] ボタンは無効な場合があります。

構文

```
SdLicense2Rtf ( szTitle, szOpt1, szOpt2, szLicenseFile, bLicenseAccepted );
```

パラメーター

テーブル 56 · SdLicense2Rtf のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (「使用許諾契約書」) を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szOpt1	上のオプションボタンの隣にあるスタティック テキスト フィールドに表示するテキストを指定します。デフォルトのテキスト (「ソフトウェア使用許諾契約に同意します」) を表示するには、このパラメーター内にヌル文字列 ("") を渡します。
szOpt2	下のオプション ボタンの隣にあるスタティック テキスト フィールドに表示するテキストを指定します。デフォルトのテキスト (「ソフトウェア使用許諾契約に同意しません」) を表示するには、このパラメーター内にヌル文字列 ("") を渡します。
szLicenseFile	使用許諾契約書を含む .rtf ファイルまたは ANSI テキストファイルの名前を指定します。このファイルは、[サポートファイル / ビルボード] ビューで適切な言語フォルダーに追加する必要があります。szLicenseFile は、引用符で囲んだ完全修飾名、または UNC パスを入力して指定することもできます。. ***メモ ·** .rtf ファイルを使用する場合、ファイルのサイズ制限は 16 MB です。*
bLicenseAccepted	デフォルトで選択したオプションボタンを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · TRUE— 上のオプションボタンをデフォルトの選択として指定します。 · FALSE Ñ 下のオプション ボタンをデフォルトの選択として指定します。

戻り値

テーブル 57 · SdLicense2Rtf の戻り値

戻り値	説明
NEXT	エンド ユーザーが上のオプションボタンと [次へ] ボタンを選択しました。
BACK	エンドユーザーが、[戻る] ボタンを選択しました。
< ISERR_SUCCESS	ダイアログを表示することができませんでした。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdLicense2Rtf の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 *  SdLicense2Rtf 関数のデモンストレーションを行います。
 *
 * このスクリプトは SdLicense2Rtf を呼び出し、使用許諾契約書を表示して
 * ユーザーに対して契約に同意するか同意しないかを問い合わせます。
 *
 * メモ : 使用許諾契約書は定数 LICENSE_PATH が指定する場所に
 *      格納してある RTF ファイルから
 *      読み込まれます。このスクリプトを実行する前に、定数が
 *      [ サポート ファイル / ビルボード ] ビューで既存の RTF ファイルを
 *      設定してください。
 *
¥*---------------------------------------------------------------------*/



#define LICENSE_PATH "License.rtf"
#define TITLE "SdLicense2Rtf の例 "

#include "ifx.h"

function OnBegin()
begin

  // セットアップダイアログで [ 戻る ] ボタンを無効にします。
  Disable(BACKBUTTON);


  // SdLicense2Rtf ダイアログを表示します。
  if (SdLicense2Rtf (TITLE, "", "", LICENSE_PATH, FALSE) = NEXT) then
    MessageBox ("インストールを続行します。", INFORMATION);
  endif;

end;
```

SdLicenseEx

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdLicenseEx 関数は、複数行編集フィールドに使用許諾契約書が記載されたダイアログを表示します。使用許諾契約書は、テキスト ファイル (.txt) またはリッチ テキスト形式ファイル (.rtf) で保存されます。

エンド ユーザーはスクロール アップ / ダウンして、使用許諾契約書を読むことができます。エンド ユーザーは [はい]、[いいえ]、あるいは有効な場合は [戻る] ボタンのいずれかを選択しなくてはなりません。通常は最初に表示されるダイアログなので、[戻る] ボタンを無効にすることもできます。ユーザーが [はい] を選択すると、インストールは続行します。ユーザーが [いいえ] を選択した場合、インストールは ExitSetup ダイアログを表示します。

構文

```
SdLicenseEx (byval string szTitle, byval string szMsg, byval string szQuestion, byval string szLicenseFile, byval bool bRtf);
```

パラメーター

テーブル 58 · SdLicenseEx のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (「ソフトウェア使用許諾契約書」) を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg	複数行編集フィールドの上のスタティック テキストフィールドに表示するメッセージを指定します。デフォルトの指示を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szQuestion	複数行編集フィールドの下のスタティック テキスト フィールドに表示するテキストを指定します。ユーザが [はい] か [いいえ] を選択することによって応答する質問をここに配置します。デフォルトの指示を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szLicenseFile	使用許諾契約書を含む ANSI テキストファイル、または .rtf ファイルの名前を指定します。このファイルは、[サポートファイル / ビルボード] ビューで適切な言語フォルダーに追加する必要があります。szLicenseFile は、引用符で囲んだ完全修飾名、または UNC パスを入力して指定することもできます。. ***メモ ·** .rtf ファイルを使用する場合、ファイルのサイズ制限は 16 MB です。*
bRtf	ダイアログにリッチ テキスト編集コントロールを使用するかどうかを示します。選択可能なオプションは以下のとおりです : · TRUE Ñ リッチ テキスト編集コントロールを使用します。このオプションを選択する場合、szLicenseFile に指定するファイルは .rtf ファイルでなくてはなりません。 · FALSE Ñ 標準編集コントロールを使用します。このオプションを選択する場合、szLicenseFile に指定するファイルは .txt ファイルでなくてはなりません。

戻り値

テーブル 59 · SdLicenseEx の戻り値

戻り値	説明
YES (1)	エンドユーザーが、[はい] ボタンを選択しました。
BACK (12)	エンドユーザーが、[戻る] ボタンを選択しました。

***メモ ·** エンドユーザーが [いいえ] ボタンをクリックした場合は、ExitSetup ダイアログが表示されるので、関数は NO を戻しません。*

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdLicenseRtf

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdLicenseEx 関数は **SdLicenseRtf** 関数に優先します。

SdLicenseRtf 関数は、複数行編集フィールドに使用許諾契約書が記載されたダイアログを表示します。使用許諾契約書は、パラメーター szLicenseFile で指定されるテキスト ファイルまたはリッチテキスト形式ファイル (.rtf) に格納されます。

ユーザーは、同意書を上下にスクロールして読むことができ、読んだ後に [はい] か [いいえ] あるいは、有効であれば [戻る] ボタンを選択します。これは通常、最初に表示されるダイアログなので、[戻る] ボタンは無効な場合があります。ユーザーが [はい] を選択すると、インストールは続行します。ユーザーが [いいえ] を選択した場合、インストールは ExitSetup ダイアログを表示します。

構文

```
SdLicenseRtf ( szTitle, szMsg, szQuestion, szLicenseFile );
```

パラメーター

テーブル 60 ・ SdLicenseRtf のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (「ソフトウェア使用許諾契約書」) を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg	複数行編集フィールドの上のスタティック テキストフィールドに表示するメッセージを指定します。デフォルトの指示を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szQuestion	複数行編集フィールドの下のスタティック テキスト フィールドに表示するテキストを指定します。ユーザが [はい] か [いいえ] を選択することによって応答する質問をここに配置します。デフォルトの指示を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szLicenseFile	使用許諾契約書を含む .rtf ファイルまたは ANSI テキストファイルの名前を指定します。このファイルは、[サポートファイル / ビルボード] ビューで適切な言語フォルダーに追加する必要があります。szLicenseFile は、引用符で囲んだ完全修飾名、または UNC パスを入力して指定することもできます。. ***メモ・****.rtf ファイルを使用する場合、ファイルのサイズ制限は 16 MB です。*

戻り値

テーブル 61 ・ SdLicenseRtf の戻り値

戻り値	説明
YES (1)	エンドユーザーが、[はい] ボタンを選択しました。
BACK (12)	エンドユーザーが、[戻る] ボタンを選択しました。

メモ・*エンドユーザーが [いいえ] ボタンをクリックした場合は、ExitSetup ダイアログが表示されるので、関数は NO を戻しません。*

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdLicenseRtf の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdLicenseRtf 関数のデモンストレーションを行います。
 *
 * このスクリプトは SdLicenseRtf を呼び出し、使用許諾契約書を表示してユーザーに対して
 * ユーザーに対して契約に同意するか同意しないかを問い合わせます。
 *
 * メモ : 使用許諾契約書は定数 LICENSE_PATH が指定する場所に
 *     格納してある RTF ファイルから
 *     読み込まれます。このスクリプトを実行する前に、定数が
 *     [ サポート ファイル / ビルボード ] ビューで既存の RTF ファイルを
 *     参照するように設定してください。
 *
¥*---------------------------------------------------------------------*/

#define LICENSE_PATH SUPPORTDIR ^ "License.rtf"
#define TITLE        "SdLicenseRtf の例 "

#include "Ifx.h"

function OnBegin()
   STRING szMsg, szQuestion;
begin

   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   // SdLicense へのパラメーターとして渡す変数をセットアップします。
   szMsg      = " 次の使用許諾契約をお読みください。" +
                " スクロールバーを利用して ¥n 残りの契約内容をご覧下さい。";
   szQuestion = " 契約に同意するには [ はい ] を選択します。¥n" +
                " セットアップをキャンセルするには [ いいえ ] を選択します。";

   // SdLicenseRtf ダイアログを表示します。
   if (SdLicenseRtf (TITLE, szMsg, szQuestion, LICENSE_PATH) = YES) then
      MessageBox (" インストールを続行します。", INFORMATION);
   endif;

end;
```

SdLoadString

SdLoadString 関数は、指定されたリソース ID に関連付けられた文字列値を戻します。InstallScript エンジンは、そのファイルの存在をまず **_isuser.dll** のリソース内で検索します。リソースが検出されなかった場合、InstallScript エンジンは **_isres.dll** を調査します。

構文

SdLoadString (nID);

パラメーター

テーブル 62 · SdLoadString のパラメーター

パラメーター	説明
nID	**_isuser.dll** または **_isres.dll** の文字列リソースの ID を指定します。いくつかの有効な nID の値は、*InstallShield Program Files フォルダー*¥**Script¥Ifx¥Include** サブフォルダーにある **Ifx.h** ファイルにあります。デフォルトの OnMaintUIBefore イベント ハンドラー関数コードは、次の識別子を使って対応する文字列値を判別します : · IFX_MAINTUI_MSG Ñ 選択したアプリケーション、およびすべてのコンポーネントを完全に削除しますか ? · IFX_ONMAINTUI_CAPTION— アンインストールの確認

戻り値

テーブル 63 · SdLoadString の戻り値

戻り値	説明
非ヌル文字列値	**SdLoadString** が文字列値を正常に取得しました。
ヌル文字列値 ("")	**SdLoadString** が失敗したことを示します。

SdLoadString の例

メモ · *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdLoadString 関数のデモンストレーションを行います。
*
* このスクリプトでは _isres.dll に ID IFX_MAINTUI_MSG と共に格納されている
* 文字列リソースを取得するために SdLoadString を呼び出します。
* そして文字列リソースがメッセージボックスに表示されます。
*
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
```

```
    STRING  svResource;
begin

    // 文字列リソースを取得します。
    svResource = SdLoadString (IFX_MAINTUI_MSG);

    if (svResource = "") then
        // エラーを報告します。
        MessageBox ("SdLoadString が失敗しました。", WARNING);
    else
        // 文字列リソースを表示します。
        SprintfBox ( INFORMATION , "SdLoadString" ,
                "文字列リソース %ld の値 :¥n%s" , IFX_MAINTUI_MSG,
                svResource );
    endif;

end;
```

SdLogonUserBrowse

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdLogonUserBrowse 関数は、エンドユーザーが指定のドメインまたはサーバー、およびユーザー名を選択できるダイアログを表示します。このダイアログは、エンド ユーザーが **SdLogonUserInformation** ダイアログで [参照] ボタンをクリックしたときに表示されます。

構文

```
SdLogonUserBrowse();
```

SdLogonUserCreateUser

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdLogonUserCreateUser 関数は、エンド ユーザーが **SdLogonUserInformation** ダイアログの [新規ユーザー情報] ボタンをクリックして、新しいユーザー情報を入力することができるダイアログを表示します。

構文

```
SdLogonUserCreateUser();
```

SdLogonUserInformation

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdLogonUserInformation 関数は、アカウントがインストール中に作成される場合、エンドユーザーに既存のユーザー アカウント情報または新しいユーザー情報を入力するようプロンプトするダイアログを表示します。

エンド ユーザーが [ユーザー] ボックスの隣にある [参照] ボタンをクリックすると、**SdLogonUserBrowse** ダイアログが表示されます。

構文

```
SdLogonUserInformation (byval string szTitle, byval string szMsg, byref string szAssociatedAccountName, byref string
    szAssociatePassword);
```

パラメーター

テーブル 64 - SdLogonUserInformation のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトルを表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログでメッセージを指定します。デフォルトのタイトルを表示するには、このパラメーターでヌル文字列 ("") を渡します。
szAssociatedAccountName	アカウント名の BYREF 値を指定します。例 : INSTALLSHIELD¥John Smith
szAssociatePassword	パスワードの BYREF 値を指定します。

戻り値

テーブル 65 - SdLogonUserInformation の戻り値

戻り値	説明
NEXT	ユーザーが、[次へ] ボタンを選択したことを示します。
BACK	ユーザーが、[戻る] ボタンを選択したことを示します。
< ISERR_SUCCESS	ダイアログが表示されなかったことを示します。

SdLogonUserListGroups

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

- *InstallScript*
- *InstallScript MSI*

SdLogonUserListGroups 関数は、エンドユーザーが指定のサーバーからグループを選択して、SdLogonUserCreateUser ダイアログの " グループ " フィールドに挿入することができるダイアログを表示します。

構文

```
SdLogonUserListGroups();
```

SdLogonUserListServers

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

- *InstallScript*

- *InstallScript MSI*

SdLogonUserListServers 関数は、エンドユーザーがユーザーアカウントが関連付けられているドメインまたはサーバーを参照することができるダイアログを表示します。

構文

SdLogonUserListServers();

SdLogonUserListUsers

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

SdLogonUserListUsers 関数は、エンドユーザーが指定のドメインまたはサーバーに既存のユーザーを参照して選択できるダイアログを表示します。

構文

SdLogonUserListUsers();

SdMakeName

SdMakeName 関数は、カスタムダイアログのセクション名を作成します。このセクション名は、.iss ファイルへの書き込みと .iss ファイルからの読み取りで利用されます。これは、InstallScript ベースのインストール プロジェクトで Setup.exe を サイレントモードで実行する際に必要となります。

構文

SdMakeName (svSection, szDlg, szUnused, nvDlgName);

パラメーター

テーブル 66 ・ SdMakeName のパラメーター

パラメーター	説明
svSection	セクション名を指定します (たとえば、"MyDlg-0")。InstallShield は szDlg および nvDlgName 変数を使用して、この変数に値を配置します。この値は、SilentReadData および SilentWriteData で使用されます。
szDlg	セクション名を作成するカスタムダイアログ (例 MyDlg) の名前を指定します。
szUnused	このパラメーターは使用されません。このパラメーターには、ヌル文字列 ("") を渡します。
nvDlgName	szDlg で指定されたダイアログの SdMakeName 呼び出し回数を記録するカウンターを指定します。このカウンターは InstallShield によって自動的に 1 つずつ増加されます。 セクションに正しい名前を付けるには、各カスタム ダイアログのこのパラメーターに固有な変数名を使用する必要があります。szDlg のダイアログ名を変数名として使用するのが、簡単な方法です。たとえば、szDlg が "MyDlgOne" の場合、このパラメーターの変数に nvMyDlgOne という名前を付け、szDlg が "MyDlgTwo" の場合は、nvMyDlgTwo という名前を付けます。

戻り値

なし。

SdMakeName の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdMakeName 関数、SilentReadData 関数、そして SilentWriteData 関数の
* デモンストレーションを行います。
*
* このスクリプト例ではサイレント インストールでのカスタム ダイアログの
* 処理方法を説明します。下に表示したカスタムダイアログボックス例の
* リソース .dll は、の [ サポートファイル / ビルボード ] ビューに、
* 圧縮された形で格納されます。
*
* ダイアログ例は InstallShield が提供する
* カスタム ダイアログ テンプレートからビルドされたものです。
*
* ダイアログ コントロール ID とその他の情報は
```

```
    * Resource.h ファイルに含まれます ( 表示されません )。例の最初の行に含まれる
    * このファイルは、InstallShield の [InstallScript] ビューに
    * 挿入されなくてはなりません。
    *
    * 例では、Cominit.txt と名づけられたテキストファイルを作成します。インストールが
    * サイレントモードで実行される場合、SilentReadData が呼び出され、
    * カスタムダイアログコントロール選択が .iss  ファイルから
    * 読み込まれます。そして選択は、.iss ファイルから
    * 無事に読み込まれることをデモンストレーションする意味で、Cominit.txt
    * ファイルに保存されます。インストールが
    * 標準モードで実行された場合、カスタムダイアログが
    * 表示され、選択肢が .iss ファイルに記録されて
    *  メッセージボックスに表示されます。.ISS ファイルの初期テキストは
    * スクリプト例の後に表示されています。
    *
    ¥*---------------------------------------------------------------*/

    #include "Resource.h"

    // ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
    #include "Ifx.h"

        export prototype ExFn_SdMakeName(HWND);

    function ExFn_SdMakeName(hMSI)
        BOOL   bDone;
        STRING svSection, svComPort, svPulse, svTone, svDial9, svVal;
        NUMBER nvCommDialog, nCmdValue, nPulseState, nToneState;
        NUMBER nDial9State, nResult, nvHandle;
        LIST   listID;
        HWND hwndDlg;
    begin

        //COMINIT.TXT テキストファイルを開いて、.ISS ファイルからのカスタムダイアログ選択を
        // そこに保存します。
        OpenFileMode (FILE_MODE_APPEND);
        OpenFile (nvHandle, "c:¥¥rul", "cominit.txt");

        // サイレントモードで実行中の場合、.ISS ファイルから読み込まれます。
        if (MODE=SILENTMODE) then
            SdMakeName (svSection, "COMM_DIALOG", "", nvCommDialog);

            SilentReadData (svSection, "Result", DATA_NUMBER, svVal, nResult);

            if (nResult = 1) then

                // .ISS ファイルからデータが読み込まれます。テキストファイルに
                // 結果を書き込む為に、データを文字列として
                // 読み込みます。
                SilentReadData (svSection, "nPulseState", DATA_STRING,
                            svPulse, nResult);

                SilentReadData (svSection, "nToneState", DATA_STRING,
                            svTone, nResult);

                SilentReadData (svSection, "nDial9State", DATA_STRING,
                            svDial9, nResult);

                // カスタムダイアログの選択をテキストファイル
```

```
        // COMINIT.TXT に格納します。
        svVal = " パルスボックスは : " ^ svPulse;
        WriteLine(nvHandle, svVal);

        svVal = " トーン ボックスは : " ^ svTone;
        WriteLine(nvHandle, svVal);

        svVal = "Dial9 ボックスは : " ^ svDial9;
        WriteLine(nvHandle, svVal);
    endif;

// サイレントモード以外の場合、通常通りカスタムダイアログを
// 呼び出し、処理します。
else
    listID = ListCreate (STRINGLIST);
    ListAddString (listID,  "COMM1:", AFTER);
    ListAddString (listID,  "COMM2:", AFTER);
    ListAddString (listID,  "COMM3:", AFTER);
    ListAddString (listID,  "COMM4:", AFTER);

    EzDefineDialog ("MYCOMDIALOG", SUPPORTDIR^"RESOURCE.DLL",
                "COMM_DIALOG",0);

    bDone = FALSE;

    while (bDone=FALSE)
        nCmdValue = WaitOnDialog ("MYCOMDIALOG");

        switch (nCmdValue)
            case DLG_INIT:
                // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
                // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
                // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
                // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
                //
                hwndDlg = CmdGetHwndDlg("MYCOMDIALOG");
                SdGeneralInit("MYCOMDIALOG", hwndDlg, 0, "");
                CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_CHECKED);
                CtrlSetList ("MYCOMDIALOG", ID_COMPORT, listID);
                CtrlSetState ("MYCOMDIALOG", ID_DIAL9, BUTTON_CHECKED);
            case OK:
                CtrlGetCurSel ("MYCOMDIALOG", ID_COMPORT, svComPort);
                nPulseState = CtrlGetState ("MYCOMDIALOG", ID_PULSE);
                nToneState = CtrlGetState ("MYCOMDIALOG", ID_TONE);
                nDial9State = CtrlGetState ("MYCOMDIALOG", ID_DIAL9);
                nResult = NEXT;
                bDone = TRUE;
            case BACK:
                nResult = BACK;
                bDone = TRUE;
            case RES_PBUT_CANCEL:
                // ユーザーが [ キャンセル ] ボタンをクリックしました。
                Do (EXIT);
            case DLG_CLOSE:
                // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
                Do (EXIT);
            case ID_PULSE:
                nPulseState = CtrlGetState ("MYCOMDIALOG", ID_PULSE);
```

```
            if (nPulseState = BUTTON_CHECKED) then
                CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_UNCHECKED);
                CtrlSetState ("MYCOMDIALOG", ID_PULSE, BUTTON_CHECKED);
            else
                CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_CHECKED);
                CtrlSetState ("MYCOMDIALOG", ID_PULSE, BUTTON_UNCHECKED);
            endif;
        case ID_TONE:
            nToneState = CtrlGetState ("MYCOMDIALOG", ID_TONE);

            if (nPulseState = BUTTON_CHECKED) then
                CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_CHECKED);
                CtrlSetState ("MYCOMDIALOG", ID_PULSE, BUTTON_UNCHECKED);
            else
                CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_UNCHECKED);
                CtrlSetState ("MYCOMDIALOG", ID_PULSE, BUTTON_CHECKED);
            endif;
        case DLG_ERR:
            MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
            abort;
    endswitch;

endwhile;

EndDialog ("MYCOMDIALOG");
ReleaseDialog ("MYCOMDIALOG");

SdMakeName (svSection, "COMM_DIALOG", "", nvCommDialog);

SilentWriteData (svSection, "nPulseState", DATA_NUMBER,
                 svPulse, nPulseState);

SilentWriteData (svSection, "nToneState", DATA_NUMBER,
                 svTone, nToneState);
SilentWriteData (svSection, "nDial9State", DATA_NUMBER,
                 svDial9, nResult);

if (nPulseState = BUTTON_CHECKED) then
    MessageBox (" パルスボタンが選択されました。", INFORMATION);
else
    MessageBox (" パルスボタンの選択が解除されました。", INFORMATION);
endif;

if (nToneState = BUTTON_CHECKED) then
    MessageBox (" トーンボタンが選択されました。.", INFORMATION);
else
    MessageBox (" トーンボタンの選択が解除されました。", INFORMATION);
endif;

if (nDial9State = BUTTON_CHECKED) then
    MessageBox ("Dial9 ボタンが選択されました。", INFORMATION);
else
    MessageBox ("Dial9 の選択が解除されました。", INFORMATION);
endif;

endif;

// テキストファイル COMINIT.TXT を閉じます。
CloseFile (nvHandle);
```

```
end;


/* 以下のコードは、前述の例の初期 .iss ファイルテキストです。ここでは、角括弧を含んだ <PRODUCT_GUID> が プロジェクトの GUID を
示します。-1001 は BUTTON_CHECKED の数値で、-1002 は BUTTON_UNCHECKED の数値です。

[InstallShield Silent]
Version=v9.00
File=Response File
[Application]
Name=MyDialog
Version=4.0
Company= ソフトウェア会社名
[<PRODUCT_GUID>-DlgOrder]
Dlg0=<PRODUCT_GUID>-COMM_DIALOG-0
Count=1
[<PRODUCT_GUID>-COMM_DIALOG-0]
nPulseState=-1001
nToneState=-1002
nDial9State=-1001
Result=1
*/
```

SdOptionsButtons

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdOptionsButtons 関数はビットマップ イメージを持つ 1 個から 4 個のプッシュボタン コントロールを含むダイアログを表示します。各コントロールの横に、短いテキスト説明が表示されます。

***重要・SdOptionsButtons** ダイアログ上のプッシュボタン コントロールのコントロール ID を変更したり、プッシュボタン コントロールの 1 つをデフォルトのコントロールとして設定したりすることはできません。コントロール ID を変更する、またはこれらのコントロールの 1 つをダイアログのデフォルト コントロールとして設定すると、**SdOptionsButtons** ダイアログが予定通りに動作しなくなります。*

***メモ・SdOptionsButtons** はセットアップの種類ダイアログとして使用できますが、**SdSetupTypeEx** ダイアログはカスタマイズを必要としないため、エンド ユーザーがセットアップの種類を選択するためのダイアログとして推奨されます。エンド ユーザーが選択したセットアップの種類を取得するために **SdOptionsButtons** を呼び出した場合、次に **FeatureSetupTypeSet** を呼び出して、選択されたセットアップの種類を設定する必要があります。*

構文

```
SdOptionsButtons (szTitle, szMsg, listButton, listDescription);
```

パラメーター

テーブル 67 • SdOptionsButtons のパラメーター

パラメーター	説明
szTitle	ダイアログ タイトルを指定します。デフォルトのタイトル (" 機能の選択 ") を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログ ボックスのデフォルトの指示「インストールする機能を選択してください」を表示するには、このパラメーターにヌル文字列 ("") を渡します。
listButton	1 つから 4 つまでの要素を含む文字列リストを指定します。各要素は形式化された文字列で、ボタンに表示されるイメージを指定します。リスト内の各文字列要素に対しボタンが 1 つずつ表示されます。文字列リスト要素の形式は次のとおりです。 @@ResourceID;ScaleFactor *ResourceID* は、.png イメージ (リソース タイプ PNG として格納される) またはビットマップ イメージ (リソース タイプ RT_BITMAP として格納される) のいずれかを検出するためのリソース識別番号を示します。*ScaleFactor* は、イメージが意図されている DPI スケール率を示します。 たとえば、スケール因子は 100% (96 DPI)、125% (120 DPI)、150% (144 DPI)、200% (192 DPI) となります。イメージに指定されているスケール因子が 200 の場合、200% DPI スケール以下で実行中のターゲット システム上で、イメージは縮小表示されます。200% ターゲットシステム上で、これは 1:1 の割合で表示されます。イメージに指定されているスケール因子が 100 の場合、200% DPI スケールで実行中のターゲット システム上で、イメージは拡大表示されます。

テーブル 67 ・ SdOptionsButtons のパラメーター (続き)

パラメーター	説明
listButton (続き)	ビットマップ イメージ (.bmp) を識別する以前のフォーマットも使用できますが、スケーリングまたは .png イメージがサポートされていません : @BitmapResourceID;TransparentFlag;3-DFlag;<unused>;TransparentColorKey *BitmapResourceID* は、ビットマップ イメージのリソース ID 番号を示します。*TransparentFlag* は、1 (true) または 0 (false) のいずれかで示され、ビットマップを表示する際、*TransparentColorKey* フィールドで指定した色が透明かそうでないかを示します。*TransparentColorKey* は、ビットマップの透過色を示す RGB 値を指定します。 InstallShield には、この関数で使用できる 4 つのデフォルトのビットマップが、**_isres.dll** にあります。これらのビットマップの ID は 12001 から 12004 までで、このスクリプト例にあるセットアップの種類の通常、ポータブル、コンパクト、カスタムに対応します。このダイアログを別の目的に使用する場合、または、**_isres.dll** に含まれているタイプ以外のセットアップの種類を使用する場合は、**_isres.dll** ダイアログのテンプレートにカスタムボタンを追加し、カスタマイズしたインストールに **_isres.dll** を含める必要があります。 ***ヒント・****ダイアログエディターで作成されたビットマップコントロールのビットマップ ID を決定するには、プロジェクトをビルドして、.rc ファイルを開いてビットマップ ID を見つける必要があります。.rc ファイルはプロジェクトフォルダーにあります (デフォルトでは、C:¥InstallShield 2016 Projects¥プロジェクト名 .ism)。追加ビットマップコントロールを追加すると、ビットマップ ID が変更される場合があります。このため、プロジェクトを再ビルドして .rc ファイルを再び開き、アップデートされたビットマップ ID を探す必要があります。*
listDescription	listButton のパラメーターにある文字列に対応する、1 つから 4 つまでの文字列要素を含む文字列リストを指定します。各文字列は記述テキストで、対応するボタンとともに表示されます。

戻り値

テーブル 68 - SdOptionsButtons の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。
101	listButton の最初の文字列要素に対応するボタンが選択されたことを示します。
102	listButton の 2 番目の文字列要素に対応するボタンが選択されたことを示します。
103	listButton の 3 番目の文字列要素に対応するボタンが選択されたことを示します。
104	listButton の 4 番目の文字列要素に対応するボタンが選択されたことを示します。

メモ - *ユーザーが必要なボタンをクリックしないでダイアログを終了するのを防ぐには、**SdOptionsButtons** を呼び出す前に、**Disable** 関数を呼び出して、[次へ] ボタンを無効にします。*

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdOptionsButtons の例

プロジェクト - *この情報は、次のプロジェクトの種類に適用します :*

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdOptionsButtons 関数のデモンストレーションを行います。
*
* このスクリプトはユーザーによるセットアップタイプの選択を可能にします。まず、
* 2 つのリストが作成されます。ひとつはセットアップの種類ボタンアイコン、
* もうひとつはセットアップの種類の説明です。そして、ボタンと説明が
* リストに追加されます。次に、ダイアログが
* 表示されます。ユーザーがセットアップの種類ボタンをクリックしたとき、
* ダイアログが閉じ、ユーザーの選択がメッセージボックスに表示
```

```
＊されます。
＊
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
    STRING  szTitle, szMsg;
    LIST    listButton, listDesc;
    NUMBER  nResult;
begin

    // セットアップダイアログで [ 戻る ] ボタンと [ 次へ ] ボタンを無効にします。
    Disable(BACKBUTTON);
    Disable (NEXTBUTTON);

    // ボタンと説明のリストを作成します。
    listButton = ListCreate (STRINGLIST);
    listDesc = ListCreate (STRINGLIST);

    if (listButton = LIST_NULL) || (listDesc = LIST_NULL) then
        // エラーを報告し、中止します。
        MessageBox(" リストを作成することができませんでした。", INFORMATION);
        abort;
    endif;

    // listButton にビットマップ ボタンを追加します。
    ListAddString (listButton, "@12001;1;255,0,255", AFTER);
    ListAddString (listButton, "@12002;1;255,0,255", AFTER);
    ListAddString (listButton, "@12003;1;255,0,255", AFTER);
    ListAddString (listButton, "@12004;1;255,0,255", AFTER);

    // listDesc へ説明を追加します。
    ListAddString (listDesc, " 標準 ¥n" +
                " ほとんどのコンピューターに推奨します。.", AFTER);
    ListAddString (listDesc, " ポータブル ¥n" +
                " アプリケーションはポータブルコンピューターに役立つ " +
                " オプションと共にセットアップされます。" +
                "", AFTER);
    ListAddString (listDesc, " コンパクト ¥n" +
                " ディスク容量を保持するため、オプション機能は " +
                " どれもインストール " +
                " されません。", AFTER);
    ListAddString (listDesc, " カスタム ¥n" +
                " 上級ユーザーとシステム管理者 " +
                " 専用です。利用可能なすべてのセットアップオプションを  " +
                " カスタマイズすることが " +
                " できます。", AFTER);

    // ダイアログを表示します。
    nResult = SdOptionsButtons (szTitle, szMsg, listButton, listDesc);

    // どのボタンが選択されたのかを示すメッセージを表示します。
    switch (nResult)
        case 101:
            MessageBox (" 標準インストールが選択されました。", INFORMATION);
        case 102:
            MessageBox (" ポータブルインストールが選択されました。", INFORMATION);
        case 103:
```

```
            MessageBox ("Compact インストールが選択されました。", INFORMATION);
        case 104:
            MessageBox (" カスタムインストールが選択されました。", INFORMATION);
        デフォルト :
            MessageBox ("SdOptionsButtons:¥n¥n エラーが発生しました。", SEVERE);
    endswitch;

    Enable(NEXTBUTTON);

    // リストを破棄します。
    ListDestroy (listButton);
    ListDestroy (listDesc);

end;
```

SdOutOfDiskSpace

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

SdOutOfDiskSpace 関数は、アプリケーションのインストールに充分な空き容量がターゲットシステムにないことをユーザーに警告するダイアログを表示します。またボリューム、必要な容量、使用可能容量および必要な容量と使用可能な容量の差についてのリストビューを表示します。

InstallScript MSI インストールでは、この関数は INSTALLMESSAGE_OUTOFDISKSPACE メッセージが検出された時にトリガーされます。このメッセージは MSI エンジンから送られます。

SdDiskSpace2 関数は、**SdOutOfDiskSpace** 関数に優先します。

構文

```
SdOutOfDiskSpace ( szTitle, szMsg );
```

パラメーター

テーブル 69 - SdOutOfDiskSpace のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「ディスク容量不足」を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このテキストはスタティック コントロールとみなされます。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。

戻り値

テーブル 70 - SdOutOfDiskSpace の戻り値

戻り値	説明
NEXT (1)	エンド ユーザーが、[OK] ボタンをクリックしたことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdPatchWelcome

***プロジェクト -** この情報は、InstallScript MSI プロジェクトに適用します。*

SdPatchWelcome 関数は、パッチインストール中にエンドユーザーに対してようこそメッセージを表示するダイアログを作成します。

構文

SdPatchWelcome (szTitle, szMsg);

パラメーター

テーブル 71 - SdPatchWelcome のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルに表示するテキストを指定します。デフォルトのタイトル「ようこそ」を表示するには、このパラメーターにヌル文字列 (“”) を渡します。
szMsg	[ようこそ] ダイアログに表示するメッセージを指定します。以前に SdProductName を呼び出した際に設定した製品名をこのメッセージに含めるには、メッセージ文字列の任意の場所に %P プレースホルダーを挿入します。メッセージが表示されると、%P は製品名に置き換えられます。デフォルトのタイトル ようこそメッセージを表示するには、このパラメーターにヌル文字列 (“”) を渡します。

戻り値

テーブル 72 - SdPatchWelcome の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。
- **SdPatchWelcome** を呼び出したときに、SD_NDLG_PATCHWELCOM (ダイアログ リソース : 12059) がインストールの **_isres.dll** ファイル内に見つからない場合、**SdPatchWelcome** の代わりに **SdWelcome** が自動的に呼び出され、**SdPatchWelcome** ダイアログに類似したダイアログが表示されます。この場合、szTitle および szMsg の値が **SdWelcome** 関数に渡されるか、szTitle および szMsg が空白の場合はカスタム 文字列が使用されます。これは、インストールが InstallShield Developer 7 で作成された製品をパッチする場合、またはダイアログ テンプレートが手動で削除されている場合にのみ発生します。

  この処理が発生するとき、ダイアログは **SdWelcome** と同じテンプレートを使用します。従って、**SdWelcome** ダイアログ テンプレートに追加された変更は、**SdPatchWelcome** ダイアログにも反映されます。

  このシナリオでは、サイレント モードはサポートされていないため、正しく動作しません。

SdPatchWelcome の例

***プロジェクト -** この情報は、InstallScript MSI プロジェクトに適用します。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
```

```
*
* SdPatchWelcome 関数のデモンストレーションを行います。
*
*   この関数は、パッチのインストール中にエンドユーザーに表示される
* ようこそメッセージを表示します。この関数は、OnPatchUIBefore イベントのデフォルトのスクリプトで
* 呼び出されます。これは InstallScript MSI プロジェクトの「その他」のイベント カテゴリに
* あります。
*
¥*---------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
    STRING szTitle, szMsg;
begin
    // デフォルトのタイトルとメッセージと一緒に SdPatchWelcome を表示します。
    szTitle = "";
    szMsg   = "";
    SdPatchWelcome(szTitle, szMsg);

end;
```

SdProductName

SdProductName 関数はシステム変数 IFX_PRODUCT_DISPLAY_NAME の値を設定します。この値によって製品名が %P プレースホルダーのインスタンスすべてに対して有効となります。%P プレースホルダーは、Sd ダイアログの静的テキストフィールドにある場合があります。さらに、標準ダイアログ関数の中には SdFinish のように、パラメーターとして関数に渡す文字列に %P を含めることができるものもあります。

構文

```
SdProductName (szProductName);
```

パラメーター

テーブル 73 - SdProductName のパラメーター

パラメーター	説明
szProductName	インストールされた製品名を指定します。製品名のプレースホルダー (%P) が、標準ダイアログの該当する静的フィールドにある場合は、この名前に置き換えられます。 ***注意 -*** *製品名にアンパサンド (&) を含める場合、エンドユーザーダイアログで名前を正しく表示するには、2 つのアンパサンド (&&) を使用する必要があります。たとえば "New & Improved Product" と表示する場合、製品名を "New && Improved Product" と入力します。*

戻り値

この関数は値を返しません。

追加情報

SdProductName を呼び出す代わりの例として、

SdProductName (" 製品名 ");

シンプルに適切な値を IFX_PRODUCT_DISPLAY_NAME に割り当てることができます。例えば、

IFX_PRODUCT_DISPLAY_NAME = " 製品名 ";

SdProductName の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdProductName 関数のデモンストレーションを行います。
*
* まず、SdProductName が呼び出され、製品名を % プレースホルダーの
* 代替として設定します。これは、SdWelcome 関数、
* Sd ダイアログへ渡されるメッセージ文字列に埋め込まれます。
*
¥*---------------------------------------------------------------------*/

#include "ifx.h"

function OnBegin()
    STRING    szProductName, szTitle;
begin

    // 製品名を %P プレースホルダーの代わりに設定します。
    szProductName = " 自分のアプリケーション ";
    SdProductName (szProductName);

    // セットアップダイアログで [ 戻る ] ボタンを無効にします。
    Disable(BACKBUTTON);

    // SdWelcome ダイアログを表示します。2 番目のパラメーターのヌル文字列は
    // デフォルト メッセージを指定します。これは %P プレースホルダーを利用します。
    szTitle = "SdProductName の例 ";
    SdWelcome (szTitle, "");

end;
```

SdRegisterUser

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdRegisterUser 関数は、エンド ユーザーがインストール中の製品のユーザー名および会社名を指定できるダイアログを表示します。

適切なパラメーターを指定することで、これらのフィールドにデフォルト値を設定できます。ヌル文字列 ("") を指定すると、関数は適切なスクリプト変数を使用します。

データが両方の編集フィールドに存在する場合のみ、[次へ] ボタンは有効になります。エンドユーザーはフィールドを空白にすることはできません。

構文

```
SdRegisterUser (szTitle, szMsg, svName, svCompany);
```

パラメーター

テーブル 74 · SdRegisterUser のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「ユーザー情報」を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
svName	関数が呼び出されたときの " 名前 " 編集フィールドのデフォルト値を指定します。 ヌル文字列 ("") が指定された場合、デフォルト値は、IFX_PRODUCT_REGISTEREDOWNER 変数の現在の値となります。InstallScript プロジェクトの場合、この変数は初回インストールの場合はレジストリから、またメンテナンス モードの場合は対応するテキスト置換から読み込まれます。InstallScript MSI プロジェクトの場合、変数のデフォルト値は常に Windows Installer プロパティ **USERNAME** から読み込まれます。 関数は、エンド ユーザーがこのパラメーターで指定した値を返します。InstallScript インストールの場合、関数はさらに *IFX_PRODUCT_REGISTEREDOWNER* の値をエンド ユーザーが指定した値に設定します。InstallScript MSI インストールの場合、関数は Windows Installer プロパティ **USERNAME** を自動的に更新します。
svCompany	関数が呼び出されたときの " 会社 " 編集フィールドのデフォルト値を指定します。 ヌル文字列 ("") が指定された場合、デフォルト値は、IFX_PRODUCT_REGISTEREDCOMPANY 変数の現在の値となります。InstallScript プロジェクトの場合、この変数は初回インストールの場合はレジストリから、またメンテナンス モードの場合は対応するテキスト置換から読み込まれます。InstallScript MSI プロジェクトの場合、変数のデフォルト値は常に Windows Installer プロパティ **COMPANYNAME** から読み込まれます。 関数は、エンド ユーザーがこのパラメーターで指定した値を返します。InstallScript インストールの場合、関数はさらに *IFX_PRODUCT_REGISTEREDCOMPANY* の値をエンド ユーザーが指定した値に設定します。InstallScript MSI インストールの場合、関数は Windows Installer プロパティ **COMPANYNAME** を自動的に更新します。

戻り値

テーブル 75 · SdRegisterUser の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdRegisterUser の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdRegisterUser 関数と SdConfirmRegistration 関数のデモンストレーションを
 * デモンストレーションを行います。
 *
 * SdRegisterUser が呼び出されて、ユーザー名と会社名を
 * 問い合わせます。これらのエントリは SdConfirmRegistration が呼び出されたときに
 * 確認されます。
 *
¥*---------------------------------------------------------------------*/

#define REG_TITLE     "SdRegisterUser の例 "
#define REG_MSG       " ここで製品を登録してください。"
#define CONFIRM_TITLE "SdConfirmRegistration の例 "

#include "ifx.h"

function OnBegin()
   STRING  svName, svCompany;
   NUMBER  nResult;
begin

   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   repeat
      // ユーザー名と会社名を取得します。
      SdRegisterUser (REG_TITLE, REG_MSG, svName, svCompany);

      // 情報が正しいことを確認します。SdRegisterUser は
      // シリアル番号を取得しないので、パラメーター 4 に文字列を渡します。
      nResult = SdConfirmRegistration (CONFIRM_TITLE, svName, svCompany, "", 0);
   until nResult = YES;

end;
```

SdRegisterUserEx

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

- *InstallScript*
- *InstallScript MSI*

SdRegisterUserEx 関数は、エンド ユーザーがインストール中の製品のユーザー名、会社名、およびシリアル番号を指定できるダイアログを表示します。

適切なパラメーターを指定することで、これらのフィールドにデフォルト値を設定できます。ヌル文字列 ("") を指定すると、関数は適切なスクリプト変数を使用します。

データが この 3 つの編集フィールドすべてに存在する場合のみ、[次へ] ボタンは有効になります。エンドユーザーはフィールドを空白にすることはできません。

メモ・SdRegisterUserEx** 関数はシリアル番号の検証を行いません。シリアル番号を検証するコードを追加する方法については、サンプル シリアル番号検証プロジェクトを参照してください。このサンプル プロジェクトは、InstallShield プログラム ファイル フォルダー内の Samples サブフォルダーの 1 つにあります。デフォルトのインストール先は、C:¥Program Files¥InstallShield¥2016¥Samples¥InstallScript¥Serial Number Validation Sample Project** です。*

構文

SdRegisterUserEx (szTitle, szMsg, svName, svCompany, svSerial);

パラメーター

テーブル 76 • SdRegisterUserEx のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「ユーザー情報」を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
svName	関数が呼び出されたときの " 名前 " 編集フィールドのデフォルト値を指定します。 ヌル文字列 ("") が指定された場合、デフォルト値は、IFX_PRODUCT_REGISTEREDOWNER 変数の現在の値となります。InstallScript プロジェクトの場合、この変数は初回インストールの場合はレジストリから、またメンテナンス モードの場合は対応するテキスト置換から読み込まれます。InstallScript MSI プロジェクトの場合、変数のデフォルト値は常に Windows Installer プロパティ **USERNAME** から読み込まれます。 関数は、エンド ユーザーがこのパラメーターで指定した値を返します。InstallScript インストールの場合、関数はさらに *IFX_PRODUCT_REGISTEREDOWNER* の値をエンド ユーザーが指定した値に設定します。InstallScript MSI インストールの場合、関数は Windows Installer プロパティ **USERNAME** を自動的に更新します。
svCompany	関数が呼び出されたときの " 会社 " 編集フィールドのデフォルト値を指定します。 ヌル文字列 ("") が指定された場合、デフォルト値は、IFX_PRODUCT_REGISTEREDCOMPANY 変数の現在の値となります。InstallScript プロジェクトの場合、この変数は初回インストールの場合はレジストリから、またメンテナンス モードの場合は対応するテキスト置換から読み込まれます。InstallScript MSI プロジェクトの場合、変数のデフォルト値は常に Windows Installer プロパティ **COMPANYNAME** から読み込まれます。 関数は、エンド ユーザーがこのパラメーターで指定した値を返します。InstallScript インストールの場合、関数はさらに *IFX_PRODUCT_REGISTEREDCOMPANY* の値をエンド ユーザーが指定した値に設定します。InstallScript MSI インストールの場合、関数は Windows Installer プロパティ **COMPANYNAME** を自動的に更新します。

テーブル 76 - SdRegisterUserEx のパラメーター (続き)

パラメーター	説明
svSerial	関数が呼び出されたときの " シリアル番号 " 編集フィールドのデフォルト値を指定します。 ヌル文字列 ("") が指定された場合、デフォルト値は、IFX_PRODUCT_REGISTEREDSERIALNUM 変数の現在の値となります。この変数は初回インストールの場合はレジストリから、またメンテナンス モードの場合は対応するテキスト置換から読み込まれます。 関数は、エンド ユーザーがこのパラメーターで指定した値を返します。関数はさらに *IFX_PRODUCT_REGISTEREDSERIALNUM* の値をエンド ユーザーが指定した値に設定します。

戻り値

テーブル 77 - SdRegisterUserEx の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdRegisterUserEx の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 *  SdRegisterUserEx 関数のデモンストレーションを行います。
 *
 *    SdRegisterUserEx を呼び出してユーザーからインストール情報を
 * 取得します。情報はリストに格納され、
 * SdShowInfoList に表示されます。
 *
¥*---------------------------------------------------------------------*/

#include "Ifx.h"
```

```
function OnBegin()
    STRING  szTitle, szMsg, svName, svCompany, svSerial;
    LIST    listData;
begin

    // リストを作成します。
    listData = ListCreate (STRINGLIST);

    // SdRegisterUserEx を呼び出すパラメーターをセットアップします。
    szTitle = "SdRegisterUserEx の例 ";
    szMsg = " 名前、会社名、及びシリアル番号を入力してください。";

    // 登録情報を読み出します。
    SdRegisterUserEx (szTitle, szMsg, svName, svCompany, svSerial);

    // リストへ情報を追加します。
    ListAddString (listData, " ユーザー情報 : ", AFTER);
    ListAddString (listData, "          " + svName, AFTER);
    ListAddString (listData, "          " + svCompany, AFTER);
    ListAddString (listData, "          " + svSerial, AFTER);
    ListAddString (listData, "", AFTER);

    // SdShowInfoList でリストからユーザーの選択を表示します。
    szMsg = " 名前、会社名、及びシリアル番号を入力してください。" +
            "SdRegisterUserEx. に入力した情報です。";
    SdShowInfoList(szTitle, szMsg, listData);

end;
```

SdRMFilesInUse

プロジェクト・*InstallScript MSI プロジェクトの種類は、****SdFilesInUse*** *関数をサポートします :*

SdRMFilesInUse 関数は、開いた状態でファイルをロックしているアプリケーションの一覧を表示するリスト ボックスを含むダイアログを表示します。ダイアログにはまた、インストールが、1) 再起動マネージャーを使用して、ファイルをロックしているアプリケーションを閉じる試みをするか、または 2) ロックされているファイルを上書きする試みをするか (結果として、インストールの完了に再起動が必要になる可能性が高くなります) エンドユーザーが指定できる 2 つのラジオ ボタンがあります。

通常、OnRMFilesInUse イベント ハンドラーはこのダイアログを InstallScript MSI インストールで、Windows Installer から送信された INSTALLMESSAGE_FILESINUSE メッセージへの応答として表示します。これが発生したとき、Windows Installer はインストールにファイルをロックしているアプリケーションのリストを提供します。アプリケーションのリストは szMessage パラメーターを通して OnRMFilesInUse イベントに渡されます。イベントは szMessage パラメーターを通して、この情報を **SdRMFilesInUse** イベントに渡します。その次、イベントは関数からの戻り値をイベントの戻り値として渡します。これにより、Windows Installer は適切に動作します。

プロジェクト・*InstallScript プロジェクトおよび InstallScript MSI プロジェクトでは、****SdRMFilesInUse*** *ダイアログを手動で呼び出すことができます。ただし、インストールはロックしているファイルのアプリケーション リストを nvlistApps パラメーターを使って提供し、戻り値を適切に処理する必要があります。また、エンドユーザーが*

SdRMFilesInUse ダイアログで対応するオプションを選択した場合、再起動マネージャーと共にインタラクトして、ロックしているアプリケーションをシャットダウンする試みも行う必要があります。

構文

SdRMFilesInUse (byval string szTitle, byval string szMsg, byval string szFilesInUse, byref LIST nvlistApps, byref BOOL bvCloseRestart);

パラメーター

テーブル 78 · SdRMFilesInUse のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「使用中のファイル」を表示するには、このパラメーターでヌル文字列 (“”) を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 (“”) を渡します。
szFilesInUse	InstallScript MSI インストールで、OnRMFilesInUse イベント ハンドラーによって **SdRMFilesInUse** が呼び出された場合、このパラメーターは Windows Installer によって検出されたファイルをロックしているアプリケーションのリストを指定します。関数はこの文字列を解析して、ダイアログのリスト ボックスにアプリケーションのリストを表示します。 **SdFilesInUse** を手動で呼び出す場合、空文字列 (“”) をこのパラメーターに渡します。nvlistApps が有効な文字列リストであるとき、このパラメーターは無視されますので注意してください。
nvlistApps	InstallScript MSI インストールで、OnRMFilesInUse イベント ハンドラーによって **SdRMFilesInUse** が 呼び出された場合、このパラメーターは初期化されていないリスト変数 (つまり、値が 0 の変数) として指定されます。 **SdRMFilesInUse** を手動で呼び出す場合、ファイルをロックしていて、かつダイアログに表示する必要があるアプリケーションの文字列リストを指定します。リストの各文字列は、1 つのアプリケーションを表します。(**ListCreate** 関数と関連するリスト関数を使用して、文字列リストの作成と初期化を行います。) ***メモ ·** このパラメーターの変数を提供する必要があります。リテラル値を指定することはできません。*
bvCloseRestart	上記のラジオ ボタンのデフォルト値を示します。TRUE は、″ アプリケーションを閉じて再起動する ″ ラジオ ボタンがデフォルトであることを示します。FALSE は、″ 無視して、後で再起動する ″ ラジオ ボタンがデフォルトであることを示します。 ***プロジェクト ·** InstallScript MSI プロジェクトでは、OnRMFilesInUse イベントが、MSIRESTARTMANAGERCONTROL プロパティ という Windows Installer プロパティの現在の値に基づいて、bvCloseRestart の値を設定します。* **SdRMFilesInUse** 関数が戻ったとき、この値はエンドユーザーが選択したラジオ ボタンを示します。TRUE の場合は、[アプリケーションを閉じて再起動する] オプション、FALSE の場合は [無視して、後で再起動する] オプションです。この関数は、関数の戻り値にも同じ情報を返します。

戻り値

テーブル 79 ・ SdRMFilesInUse の戻り値

戻り値	説明
IDOK	エンドユーザーが [後でコンピューターを再起動する] オプションを選択して、[OK] ボタンをクリックしたことを示します。
IDIGNORE (5)	エンドユーザーが [無視して、後で再起動する] オプションを選択して、[OK] ボタンをクリックしたことを示します。この値はダイアログに [無視する] ボタンが含まれていない場合でも返されますので、ご注意ください。
IDCANCEL	エンドユーザーが [終了] ボタンをクリックしたことを示します。 ***メモ ・*** *他のスクリプト ダイアログとは違い、このダイアログは、ユーザーが [終了] ボタンをクリックしたとき、OnCanceling イベント ハンドラーを呼び出しませんので注意してください。したがって、この関数を手動で呼び出す場合、[セットアップの取り消し] ダイアログを表示するには、この戻り値を手動で処理する必要があります。*

SdSelectFolder

プロジェクト ・ *この情報は、次のプロジェクトの種類に適用します :*

- *InstallScript*
- *InstallScript MSI*

SdSelectFolder 関数は、選択のためにフォルダーを表示します。**SdSelectFolder** を使用して、デフォルトの選択を表示することができます。また、エンド ユーザーが新しいフォルダー名を入力することもできます。**SdSelectFolder** は、選択または入力されたフォルダー名のみを戻します。したがって、フォルダーは作成できません。

構文

SdSelectFolder (szTitle, szMsg, svDefGroup);

パラメーター

テーブル 80 - SdSelectFolder のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「プログラム フォルダーの選択」を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このテキストはスタティック コントロールとみなされます。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
svDefGroup	選択されたフォルダー名を返します。

戻り値

テーブル 81 - SdSelectFolder の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。
- **ProgDefGroupType** が **SdSelectFolder** の前に呼び出されると、**SdSelectFolder** によって表示されるプフォグラム フォルダー (共通または個人) は、**ProgDefGroupType**. に渡されたパラメーターによって異なります。

SdSelectFolder の例

プロジェクト - *この情報は、次のプロジェクトの種類に適用します :*

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdSelectFolder 関数のデモンストレーションを行います。
*
* ユーザーへフォルダーの選択を求めるため、SdSelectFolder が
* ハイライト表示します。
*
```

```
＊メモ：このスクリプトを実行する前に、定数 DEF_FOLDER が
＊ターゲットコンピューター上の既存フォルダーを参照することを
＊　確認してください。
＊
¥*---------------------------------------------------------------*/

#define DEF_FOLDER "Startup"

#include "Ifx.h"

function OnBegin()
   STRING  szTitle, szMsg, svDefGroup;
begin

   // SdSelectFolder を呼び出すパラメーターをセットアップします。
   szTitle  = "SdSelectFolder の例 ";
   szMsg  = "";
   svDefGroup = DEF_FOLDER;

   // ユーザーのフォルダー選択を取得します。
   SdSelectFolder (szTitle, szMsg, svDefGroup);

end;
```

SdSetupCompleteError

プロジェクト・この情報は、次のプロジェクトの種類に適用します：

- *InstallScript*
- *InstallScript MSI*

SdSetupCompleteError 関数は、エンドユーザーに対してインストレールが完了する前に中断されたことを通知するダイアログを表示します。ここではまた、製品がインストールされておらず、ターゲットシステムも変更されていないことも通知します。

構文

```
SdSetupCompleteError ( szTitle, szMsg1, szMsg2 );
```

パラメーター

テーブル 82 - SdSetupCompleteError のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「InstallShield Wizard の完了」を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg1	ダイアログの上部に表示するメッセージを指定します。インストールが中断され、製品がインストールされなかったことをユーザーに通知するデフォルト指示を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg2	ダイアログの下部に表示するメッセージを指定します。デフォルトの指示「[完了] ボタンをクリックして、ウィザードを終了してください。」を表示するには、このパラメーターにヌル文字列 ("") を渡します。

戻り値

テーブル 83 - SdSetupCompleteError の戻り値

戻り値	説明
NEXT (1)	[完了] ボタンがクリックされたことを示します。

追加情報

独自の手続き型スクリプトを作成した場合、つまり program/endprogram ブロックを含む場合、エンド ユーザーがインストールをキャンセルした場合は **SdSetupCompleteError** 関数を使ってダイアログを表示することができます。デフォルトのスクリプト イベント モデルを利用している場合、この関数を呼び出す必要はありません。

SdSetupCompleteError はインストールが中断されたため完了できなかったことを示すため、[戻る] ボタンは無効です。

SdSetupCompleteError ダイアログは、**SdFinish** ダイアログの一種です。サイレント応答ファイル (.iss) では、**SdSetupCompleteError** ダイアログは **SdFinish** と呼ばれます。

SdSetupCompleteError の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdSetupCompleteError 関数のデモンストレーションを行います。
```

```
*
* SdSetupCompleteError 関数はダイアログを表示してユーザーに対して
* インストールが完了する前に中断されたことを通知する
* ダイアログ ボックスを表示します。また、エンドユーザーに対してアプリケーションが
* インストールされなかったことと、ターゲットシステムが変更されていないことを
* 通知します。この関数は MSI ベースプロジェクトのみで
* 利用可能です。
*
¥*---------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
    STRING  szTitle, szMsg1, szMsg2;
begin
    // SdSetupCompleteError への呼び出用パラメーターをセットアップします。
    szTitle = "SdSetupCompleteError の例 ";
    szMsg1  = "";
    szMsg2  = "";
    SdSetupCompleteError ( szTitle, szMsg1, szMsg2 );

end;
```

SdSetupType

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdSetupType 関数は、エンドユーザーが、3 種類の標準セットアップ ([標準]、[最小]、[カスタム]) の中から 1 つを選択できるダイアログを表示します。これらのセットアップオプションは標準の記述テキストで表示されます。他のセットアップの種類を追加したり、表示されたセットアップの名前や内容を変更したい場合は、SetupType 関数ではなく、SdSetupTypeEx を呼び出してください。

ダイアログにはデフォルトのインストール先パスも表示されます。[参照] ボタンはダイアログを起動し、エンドユーザーが、新しいフォルダー名を入力するか、または既存のフォルダーをリストから選択して、インストール先パスを変更できるようにします。エンドユーザーが存在しないフォルダー名を入力した場合は、この関数によって指定のフォルダーが自動的に作成されます。指定されたフォルダーの完全修飾パスが、svDir に返されます。

注意・エンドユーザーが機能ダイアログを使用して、選択したセットアップに対応する機能を選択または選択解除した後で、SdSetupType ダイアログに戻った場合、これらの選択は失われます。この問題が発生するのは、SdSetupType 関数が呼び出される度に、デフォルトの機能選択が自動的にリセットされるからです。

構文

```
SdSetupType (szTitle, szMsg, svDir, nReserved);
```

パラメーター

テーブル 84 - SdSetupType のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「セットアップの種類」を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このテキストはスタティック コントロールとみなされます。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
svDir	デフォルトのフォルダー名を指定します。エンドユーザーが選択したフォルダーの名前を返します。
nReserved	後で使用するために予約されています。このパラメーターでゼロ (0) を渡します。

戻り値

テーブル 85 - SdSetupType の戻り値

戻り値	説明
TYPICAL (301)	ユーザーが標準セットアップを選択したことを示します。
COMPACT (302)	ユーザーがコンパクトセットアップを選択したことを示します。
CUSTOM (303)	ユーザーがカスタム セットアップを選択したことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。

追加情報

- InstallScript プロジェクトでセットアップの種類 ダイアログを表示しない場合、スクリプトは以下の手順の 1 つを実行しなくてはなりません。
  - セットアップの種類を選択する。
  - たとえば SdFeatureTree のような機能選択ダイアログ関数を呼び出す。
  - 直接機能を選択する。
- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdSetupType の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdSetupType 関数のデモンストレーションを行います。
 *
 *   この関数は、3 つの標準セットアップの種類 [ 標準 ]、[ 最小 ]、
 *   または [ カスタム ] のひとつをエンドユーザーが選択することが
 *   できるダイアログ ボックスを表示します。
 *
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
    STRING  szTitle, szMsg, svDir;
    NUMBER   nReserved, nResult;
begin

    // SdSetupType ダイアログを表示します。
    szTitle   = "SdSetupType の例 ";
    szMsg   = "";
    // ダイアログに表示されるデフォルトのインストール先フォルダー。
    svDir   = "C:¥¥Example";
    nReserved = 0;
    nResult = SdSetupType (szTitle, szMsg, svDir, nReserved);

    //TARGETDIR をユーザーが選択したインストール先フォルダーに設定します。
    TARGETDIR = svDir;

    // ユーザーが選択したセットアップの種類を取得します。
    switch(nResult)

        case CUSTOM: MessageBox (" カスタム セットアップの種類が選択されました。",  0);

        case TYPICAL: MessageBox(" 標準セットアップ タイプが選択されました。", 0);

        case COMPACT: MessageBox(" 最小セットアップ タイプが選択されました。", 0);

    endswitch;

end;
```

SdSetupType2

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdSetupType2 関数は、エンドユーザーが、2 種類の標準セットアップ ([標準] または [カスタム]) の中から 1 つを選択できるダイアログを表示します。これらのセットアップオプションは標準の記述テキストで表示されます。他のセットアップの種類を追加したり、表示されたセットアップの名前や内容を変更したい場合は、SetupType 関数ではなく、SdSetupTypeEx を呼び出してください。

ダイアログにはデフォルトのインストール先パスも表示されます。[参照] ボタンはダイアログを起動し、エンドユーザーが、新しいフォルダー名を入力するか、または既存のフォルダーをリストから選択して、インストール先パスを変更できるようにします。エンドユーザーが存在しないフォルダー名を入力した場合は、この関数によって指定のフォルダーが自動的に作成されます。指定されたフォルダーの完全修飾パスが、svDir に返されます。

注意 *・エンドユーザーが機能ダイアログを使用して、選択したセットアップに対応する機能を選択または選択解除した後で、SdSetupType2 ダイアログに戻った場合、これらの選択は失われます。この問題が発生するのは、SdSetupType2 関数が呼び出される度に、デフォルトの機能選択が自動的にリセットされるからです。*

構文

```
SdSetupType2 ( szTitle, szMsg, svDir, nReserved );
```

パラメーター

テーブル 86 - SdSetupType2 のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「セットアップの種類」を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このテキストはスタティック コントロールとみなされます。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
svDir	デフォルトのフォルダー名を指定します。エンドユーザーが選択したフォルダーの名前を返します。
nReserved	後で使用するために予約されています。このパラメーターでゼロ (0) を渡します。

戻り値

テーブル 87 - SdSetupType2 の戻り値

戻り値	説明
COMPLETE (304)	ユーザーが、[完全] セットアップの種類を選択したことを示します。
TYPICAL (301)	ユーザーが、[標準] セットアップの種類を選択したことを示します。
CUSTOM (303)	ユーザーが、[カスタム] セットアップの種類を選択したことを示します。
BACK (12)	ユーザーが、[戻る] ボタンをクリックしたことを示します。

追加情報

- InstallScript プロジェクトでセットアップの種類 ダイアログを表示しない場合、スクリプトは以下の手順の 1 つを実行しなくてはなりません。
  - セットアップの種類を選択する。
  - たとえば SdFeatureTree のような機能選択ダイアログ関数を呼び出す。
  - 直接機能を選択する。
- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdSetupType2 の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdSetupType2 関数のデモンストレーションを行います。
 *
 *   SdSetupType2   を呼び出してユーザーからインストール情報を
 * 取得します。
 *
¥*---------------------------------------------------------------------*/

#include "ifx.h"

function OnBegin()
   STRING  szTitle, szMsg, svDir;
   STRING  svSetupType;
   NUMBER   nResult;
begin
   // [ 戻る ] ボタンの無効化。
   Disable(BACKBUTTON);

   // SdSetupType2 を呼び出すパラメーターの設定。
   szTitle = "SdSetupType2 の例 ";
   szMsg = " ボタンの 1 つをクリックしてインストールする種類を選んでください。";

   // セットアップの種類とディレクトリ情報を読み出します。
   svDir = TARGETDIR;
   nResult = SdSetupType2 (szTitle , szMsg, svDir, 0);
   TARGETDIR = svDir;

   // 選択されたセットアップの種類を説明する文字列を作成します。
   switch (nResult)

      case COMPLETE:
                svSetupType = "COMPLETE: アプリケーションは " +
                " すべてのオプションを含みます。";
      case CUSTOM:
                svSetupType = "CUSTOM: インストールするオプション " +
                " を選択します。";
      デフォルト :
                MessageBox (" 選択したセットアップの種類は無効です !", SEVERE);
                abort;
   endswitch;

   MessageBox(" セットアップの種類 : " + svSetupType, 0);

end;
```

SdSetupTypeEx

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

SdSetupTypeEx 関数は、エンドユーザーが [完全] および [カスタム] 以外のセットアップタイプを指定する際に、セットアップタイプを選択するためのダイアログを表示します。ダイアログは [セットアップの種類] ビューで指定したセットアップ タイプの名前を表示します。

構文

```
SdSetupTypeEx (szTitle, szMsg, szReserved, svSetupType, nReserved);
```

パラメーター

テーブル 88 ・ SdSetupTypeEx のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「セットアップの種類」を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szReserved	このパラメーターにヌル文字列 ("") を渡します。他の値は使用できません。
svSetupType	デフォルトのセットアップの種類を指定し、エンドユーザーが選択したセットアップの種類を返します。このパラメーターで返された文字列は、IDE で指定したセットアップの種類の名前と一致します。
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。

戻り値

テーブル 89 ・ SdSetupTypeEx の戻り値

戻り値	説明
0	SdSetupTypeEx が成功したことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。

追加情報

- InstallScript プロジェクトでセットアップの種類 ダイアログを表示しない場合、スクリプトは以下の手順の 1 つを実行しなくてはなりません。
  - セットアップの種類を選択する。
  - たとえば SdFeatureTree のような機能選択ダイアログ関数を呼び出す。

  - 直接機能を選択する。
- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdSetupTypeEx の例

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdSetupTypeEx 関数のデモンストレーションを行います。
*
*   SdSetupTypeEx 関数は、[ 完全 ] と [ カスタム ] 以外のセットアップの種類を指定したときに
* エンドユーザーがセットアップの種類を選択することができるダイアログを
* 表示します。ダイアログには、
IDE の [ セットアップの種類 ] ビューで作成したセットアップの種類の名前が
* 表示されます。
*
* このサンプルは、SdSetupTypeEx ダイアログのデフォルトのセットアップの種類を
* [ 完全 ] に設定します。
*
¥*---------------------------------------------------------------------*/

#include "ifx.h"

function OnBegin()
    STRING szTitle, szMsg, szReserved, svSetupType;
    NUMBER nReserved;
begin
    // セットアップダイアログで [ 戻る ] ボタンを無効にします。
    Disable(BACKBUTTON);

    // SdSetupTypeEx へのパラメーターとして渡す変数をセットアップします。
    szTitle= "SdSetupTypeEx のサンプル ";
    szMsg= "";
    szReserved  = "";

    // デフォルトのセットアップの種類 ([ セットアップの種類 ] ビューで指定されたとおりに指定 )
    // エンドユーザーが選択したセットアップの種類を返します。
    svSetupType = " 完全 ";
    nReserved = 0;

    // SdSetupTypeEx ダイアログを表示します。
    SdSetupTypeEx(szTitle, szMsg, szReserved, svSetupType, nReserved);

    // エンドユーザーが選択したセットアップの種類で essageBox を表示します。
    MessageBox(" 選択されたセットアップの種類 : " + svSetupType, 0);

end;
```

SdShowAnyDialog

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

SdShowAnyDialog 関数は、カスタム ダイアログまたは変更されたダイアログを表示します。この関数は経験の豊富なユーザーのみに推奨します。

構文

SdShowAnyDialog (szTitle, szID, nID, nReserved);

パラメーター

テーブル 90 ・ SdShowAnyDialog のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「ようこそ」を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szID	ダイアログを識別する文字列識別子を指定します。このパラメーターがヌル文字列 ("") を含んでいる場合、SdShowAnyDialog は nID の値を使用します。
nID	ダイアログを識別する数値を指定します。szID に値を入力すると、このパラメーターは無視されます。
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。

戻り値

テーブル 91 ・ SdShowAnyDialog の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。

追加情報

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。
- SdShowAnyDialog 関数を使用するには、表示する、_isres.dll 内にある変更済みのダイアログまたは _isuser.dll 内にあるカスタム ダイアログのいずれかの ID が必要です。

- このダイアログに静的コントロールしかない場合、SdShowAnyDialog クリプトファイルを変更する必要はありません。しかし、このダイアログに他のコントロールがある場合は、ユーザーからのフィードバックを処理するために、InstallShield プログラム ファイル フォルダーの Script¥Isrt¥Src フォルダーにある Sdsadlg.rul ファイルを変更する必要があります。

SdShowAnyDialog の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdShowAnyDialog 関数のデモンストレーションを行います。
 *
 * SdShowAnyDialog は 2 回呼び出されます。最初は [ ようこそ ] ダイアログ ボックスの
 * ID と共に、そして次に [ 終了 ] ダイアログの ID と共に呼び出
 * されます。
 *
¥*---------------------------------------------------------------------*/

#define TITLE "SdShowAnyDialog の例 "

#include "Ifx.h"

function OnBegin()
begin

    // セットアップダイアログで [ 戻る ] ボタンを無効にします。
    Disable(BACKBUTTON);

    // 標準ダイアログ ボックスの %P プレースホルダーの位置に
    // 表示されるよう、製品名を設定します。
    SdProductName ("ExampApp");

    // SdWelcome ダイアログを表示します。
    SdShowAnyDialog (TITLE, "", SD_NDLG_WELCOME, 0);

    // SdFinish ダイアログを表示します。
    SdShowAnyDialog (TITLE, "", SD_NDLG_FINISH, 0);

end;
```

SdShowDlgEdit1

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*

- *InstallScript MSI*

SdShowDlgEdit1 関数は、メッセージおよび 1 つの単一行編集フィールドを表示する汎用ダイアログを作成します。ダイアログのタイトルを指定できます。

構文

SdShowDlgEdit1 (szTitle, szMsg, szField1, svEdit1);

パラメーター

テーブル 92 - SdShowDlgEdit1 のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「データの編集」を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。以前に SdProductName を呼び出した際に設定した製品名をこのメッセージに含めるには、メッセージ文字列の任意の場所に %P プレースホルダーを挿入します。メッセージが表示されると、%P は製品名に置き換えられます。
szField1	編集フィールドの左側に表示するフィールド名を指定します。デフォルトのフィールド名は “Field 1:” です。デフォルトの名前を表示するには、このパラメーターにヌル文字列 ("") を渡します。表示可能な最大文字数は、およそ 10 です。実際の最大数は、フィールド名の文字幅の組み合わせによります。フィールド名が使用可能な領域を越える場合は、ダイアログが表示される際に右側が切り詰められます。
svEdit1	編集フィールドの初期値を指定し、ダイアログを閉じる際に、その編集フィールドの値を返します。

戻り値

テーブル 93 - SdShowDlgEdit1 の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdShowDlgEdit1 の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*
 *
 * InstallShield スクリプトの例
 *
 * SdShowDlgEdit1 関数のデモンストレーションを行います。
 *
 *  このスクリプト例は、SdShowDlgEdit1 を呼び出してフォルダー名を
 * 取得し、そしてメッセージボックスに表示します。
 *
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
   STRING szTitle, szMsg, szField1, svEdit1;
begin
   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   // SdShowDlgEdit1 を呼び出すパラメーターをセットアップします。
   szTitle  = "SdShowDlgEdit1 の例 ";
   szMsg    = "YourApp 用のフォルダーを選択してください :";
   szField1 = " ターゲット :";
   svEdit1  = "C:¥¥Example¥¥Target¥¥YourApp";

   // ユーザーからターゲットフォルダー名を取得します。
   if (SdShowDlgEdit1 (szTitle, szMsg, szField1, svEdit1) < 0) then
      // エラーを報告します。
      MessageBox ("SdShowDlgEdit1 が失敗しました。", SEVERE);

   else
      // svEdit1 文字列変数を表示します。
      SprintfBox (INFORMATION, szTitle, "% を選択しました。", svDir);

   endif;

end;
```

SdShowDlgEdit2

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdShowDlgEdit1 関数は、メッセージおよび 2 つの単一行編集フィールドを表示する汎用ダイアログを作成します。ダイアログのタイトルを指定できます。

構文

SdShowDlgEdit2 (szTitle, szMsg, szField1, szField2, svEdit1, svEdit2);

パラメーター

テーブル 94 · SdShowDlgEdit2 のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。このパラメーターにヌル文字列 ("") を渡すと、デフォルト タイトル「データの編集」が表示されます。
szMsg	ダイアログに表示するメッセージを指定します。以前に SdProductName を呼び出した際に設定した製品名をこのメッセージに含めるには、メッセージ文字列の任意の場所に %P プレースホルダーを挿入します。メッセージが表示されると、%P は製品名に置き換えられます。
szField1	最初の編集フィールドの左側に表示するフィールド名を指定します。デフォルトのフィールド名は "Field 1:" です。デフォルト名を表示するには、このパラメーターにヌル文字列 ("") を渡します。表示可能な最大文字数は、およそ 10 です。実際の最大数は、フィールド名の文字幅の組み合わせによります。フィールド名が使用可能な領域を越える場合は、ダイアログが表示される際に右側が切り詰められます。
szField2	2 番目の編集フィールドのフィールド名を指定します。"Field 2:" がデフォルトです。
svEdit1	最初の編集フィールドの初期値を指定し、ダイアログを閉じる際に、その最初の編集フィールドの値を返します。
svEdit2	2 番目の編集フィールドの初期値を指定し、ダイアログを閉じる際に、その 2 番目の編集フィールドの値を返します。

戻り値

テーブル 95 · SdShowDlgEdit2 の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdShowDlgEdit2 の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdShowDlgEdit2 関数のデモンストレーションを行います。
 *
 * このスクリプトは、ユーザーに対してソース フォルダーとターゲット フォルダーの指定を求める
 * ダイアログを表示します。そしてメッセージボックスに
 * ユーザーの選択を表示します。
 *
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
  STRING szTitle, szMsg, szField1, szField2, svEdit1, svEdit2;
begin

  // セットアップダイアログで [ 戻る ] ボタンを無効にします。
  Disable(BACKBUTTON);

  // SdShowDlgEdit2 を呼び出すパラメーターをセットアップします。
  szTitle  = "SdShowDlgEdit2 の例 ";
  szMsg    = " ソースディレクトリのファイルはすべて " +
             " ターゲットディレクトリへコピーされます。";
  szField1 = " ソース :";
  szField2  = " ターゲット ";
  svEdit1  = "C:¥¥Example¥¥Source";
  svEdit2  = "C:¥¥Example¥¥Target";

  // ソースフォルダーとターゲットフォルダーの名前を取得します。
  if (SdShowDlgEdit2 (szTitle, szMsg, szField1, szField2,
                      svEdit1,  svEdit2) < 0) then
    // エラーを報告します。
    MessageBox ("SdShowDlgEdit2 が失敗しました。", SEVERE);

  else
    // ユーザーの選択を表示します。
    SprintfBox (INFORMATION, szTitle, "svEdit1: %s¥n¥nsvEdit2: %s",
                svEdit1, svEdit2);
  endif;

end;
```

SdShowDlgEdit3

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdShowDlgEdit3 関数は、メッセージおよび 3 つの単一行編集フィールドを表示する汎用ダイアログを作成します。szTitle のダイアログのタイトルを指定できます。

構文

SdShowDlgEdit3 (szTitle, szMsg, szField1, szField2, szField3, svEdit1, svEdit2, svEdit3);

パラメーター

テーブル 96 · SdShowDlgEdit3 のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。このパラメーターにヌル文字列 (“”) を渡すと、デフォルト タイトル「データの編集」が表示されます。
szMsg	ダイアログに表示するメッセージを指定します。以前に SdProductName を呼び出した際に設定した製品名をこのメッセージに含めるには、メッセージ文字列の任意の場所に %P プレースホルダーを挿入します。メッセージが表示されると、%P は製品名に置き換えられます。
szField1	最初の編集フィールドの左側に表示するフィールド名を指定します。デフォルトのフィールド名は “Field 1:” です。デフォルトの名前を表示するには、このパラメーターにヌル文字列 (“”) を渡します。表示可能な最大文字数は、およそ 10 です。実際の最大数は、フィールド名の文字幅の組み合わせによります。フィールド名が使用可能な領域を越える場合は、ダイアログが表示される際に右側が切り詰められます。
szField2	2 番目の編集フィールドのフィールド名を指定します。“Field 2:” がデフォルトです。
szField3	3 番目の編集フィールドのフィールド名を指定します。“Field 3:” がデフォルトです。
svEdit1	最初の編集フィールドの初期値を指定し、ダイアログを閉じる際に、その最初の編集フィールドの値を返します。
svEdit2	2 番目の編集フィールドの初期値を指定し、ダイアログを閉じる際に、その 2 番目の編集フィールドの値を返します。
svEdit3	3 番目の編集フィールドの初期値を指定し、ダイアログを閉じる際に、その 3 番目の編集フィールドの値を返します。

戻り値

テーブル 97 · SdShowDlgEdit3 の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdShowDlgEdit3 の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdShowDlgEdit3 関数のデモンストレーションを行います。
 *
 * このスクリプトは SdShowDlgEdit3 を呼び出し、ユーザーから 3 つのフォルダー名を
 * 取得します。* そして、これらがメッセージボックスに表示されます。
 *
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
  STRING szTitle, szMsg, szField1, szField2, szField3, svEdit1, svEdit2;
  STRING svEdit3;
begin

  // セットアップダイアログで [ 戻る ] ボタンを無効にします。
  Disable(BACKBUTTON);

  // SdShowDlgEdit3 を呼び出すパラメーターをセットアップします。
  szTitle  = "SdShowDlgEdit3 の例 ";
  szMsg = " バックアップの有効なディレクトリを 2 つまで選択してください : または " +
          " 代替 : ソースディレクトリをバックアップします。¥n¥n";
  szField1  = " ディレクトリ ";
  szField2  = " バックアップ ";
  szField3  = " 代替 ";
  svEdit1   = "C:¥¥YourApp";
  svEdit2   = "C:¥¥Backup1";
  svEdit3   = "C:¥¥Backup2";

  // 3 つのフォルダー名をユーザーから取得します。
  if (SdShowDlgEdit3 (szTitle, szMsg, szField1, szField2, szField3,
                      svEdit1, svEdit2, svEdit3) < 0) then
    // エラーを報告します。
    MessageBox ("SdShowDlgEdit3 が失敗しました。", SEVERE);

  else
    // ユーザーが指定したフォルダー名を表示します。
    szMsg = "svEdit1: %s¥n¥nsvEdit2: %s¥n¥nsvEdit3: %s";
    SprintfBox (INFORMATION, szTitle, szMsg, svEdit1, svEdit2, svEdit3);

  endif;

end;
```

SdShowFileMods

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdShowFileMods 関数は、ファイルに対して行う変更を表示するダイアログを作成します。以下の選択肢が表示されます。

- ターゲットファイルを変更する。
- ターゲットファイルのコピーである代替ファイルを変更するが、変更を組み込む。
- 変更をしない。SdShowFileMods はファイルを変更しません。該当するファイル関数を使って、これらの変更をスクリプトに書き込む必要があります。

構文

SdShowFileMods (szTitle, szMsg, szTargetFile, szAltFile, listChanges, nvSelection);

パラメーター

テーブル 98 - SdShowFileMods のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「ファイルの変更」を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szTargetFile	変更するファイルの名前を指定します。この名前は、最初のラジオ ボタンと共に表示されます。
szAltFile	エンドユーザーが変更を行う場合、ファイルにつける別の名前を指定します。この名前は、2 番目のラジオ ボタンと共に表示されます。拡張子 .bak を使用した szTargetFile で指定したファイルの名前を使用するには、このパラメーターにヌル文字列 ("") を渡します。
listChanges	ファイルに行う変更のリストを含む文字列リストの名前を指定します。このリストは複数行の編集フィールドに置かれ、エンドユーザーはこれを使って実行する変更を選択できます。
nvSelection	エンドユーザーが選択したボタンの ID を戻します : • 101—" セットアップに <szTargetFile> ファイルを変更させる。" • 102—"<szAltFile> ファイルに必要な変更を保存する。" • 103—" 変更しない。"

戻り値

テーブル 99 - SdShowFileMods の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdShowFileMods の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdShowFileMods 関数のデモンストレーションを行います。
 *
 * このスクリプトは特定ファイルの変更点のリストを表示します。
 * ます。ユーザーはこれらの変更を行うのか、新しいファイルに保存するのか、
 * または、変更を保存しないかを選択することができます。
 *
 * メモ : SdShowFileMods そのものはファイルを更新または作成しません。
 *    ユーザーの選択を取得するのみです。
 *
¥*---------------------------------------------------------------------*/

#include "ifx.h"

function OnBegin()
   STRING szTitle, szMsg, szTargetFile, szAltFile;
   NUMBER nvSelection;
   LIST   listID;
begin

   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   // SdShowFileMods を呼び出す為の変数をセットアップします。
   szTitle = "SdShowFileMods の例 ";

   szMsg = " 実行するオプションを選択します。"
   szTargetFile = "Example.txt";
   szAltFile = "Example.new";

   // ファイル変更の詳細を収めるリストを作成します。
   listID = ListCreate (STRINGLIST);

   // ファイル変更の詳細をリストへ追加します。
   ListAddString (listID, "PATH = C:¥¥Example", AFTER);
   ListAddString (listID, "FILES = 40", AFTER);

   // オプションを表示してユーザーの選択を取得します。
   SdShowFileMods (szTitle, szMsg, szTargetFile, szAltFile,
                   listID, nvSelection);

   // ユーザーの選択を処理します。
   switch(nvSelection)

      case 101:
         SprintfBox (INFORMATION, szTitle, " セットアップは %s ファイルを更新しました。",
```

```
                szTargetFile);

    case 102:
      SprintfBox (INFORMATION, szTitle, " 必要な変更は " +
                "%s ファイルに保存されました。", szAltFile);

    case 103:
      SprintfBox (INFORMATION, szTitle, " 変更されませんでした。.");

  endswitch;

end;
```

SdShowInfoList

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdShowInfoList 関数は、スクロール可能なメッセージのリストを表示するダイアログを作成します。SdShowInfoList は最大でおよそ 57,200 文字のリストを表示できます。

構文

```
SdShowInfoList ( szTitle, szMsg, listID );
```

パラメーター

テーブル 100 ・ SdShowInfoList のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「情報」を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg	情報ボックスの上の 1 行に表示するメッセージを指定します。メッセージが長すぎて 1 行におさまらない場合、右側に切り詰められます。メッセージに改行エスケープシーケンス (¥n) が含まれる場合、このエスケープシーケンスの後のテキストは表示されません。デフォルトのメッセージ「テキスト」を表示するには、このパラメーターにヌル文字列 ("") を渡します。
listID	ダイアログに表示するメッセージのリストを指定します。このダイアログに表示されるメッセージは、すべて読み取り専用です。

戻り値

テーブル 101 ・ SdShowInfoList の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。
BACK (12)	[戻る] ボタンがクリックされたことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdShowInfoList の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdShowInfoList 関数のデモンストレーションを行います。
*
* このスクリプトは GetSystemInfo を呼び出して、ユーザーのシステム情報を
* 読み出します。ListAdd 関数は文字列リストへ情報を追加するのに
* 利用されます。これは SdShowInfoList 関数への呼び出しで
```

```
＊表示されます。
＊
¥*-------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
  STRING  szTitle, szMsg, svReturn, szInfo;
  NUMBER  nvReturn;
  LIST    listInfo;
begin

  // システム情報を収めるリストを作成します。
  listInfo = ListCreate (STRINGLIST);

  // システムに CD-ROM ドライブがあるかどうかを確認します。
  GetSystemInfo (CDROM, nvReturn, svReturn);

  if (nvReturn = TRUE) then
    szInfo = " マシンには CD-ROM ドライブがあります。";
  else
    szInfo = " マシンには CD-ROM ドライブがありません。";
  endif;

  // リストへ CD-RP 情報を追加します。
  ListAddString (listInfo, szInfo, AFTER);

  // システムの時刻を確認します。
  GetSystemInfo (TIME, nvReturn, svReturn);
  Sprintf (szInfo, " 現在の時刻は %s です。", svReturn);

  // リストへ時刻を追加します。
  ListAddString (listInfo, szInfo, AFTER);

  // セットアップダイアログで [ 戻る ] ボタンを無効にします。
  Disable(BACKBUTTON);

  // SdShowInfoList を呼び出すタイトルとメッセージパラメーターをセットアップします。
  szTitle  = "SdShowInfoList の例 ";
  szMsg    = " 次に示すのは、ご利用中のシステムに関連する情報です :";

  // 情報を表示します。
  SdShowInfoList (szTitle, szMsg, listInfo);

end;
```

SdShowMsg

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdShowMsg 関数は、スクリプト処理を続けながら、画面に残しておく情報メッセージを簡単に表示する方法を備えています。

SdShowMsg 関数は、szMsg で指定するメッセージを表示する小さなモードレスのウィンドウを開いたり閉じたりします。bShow が TRUE の場合ウィンドウが開いてメッセージがウィンドウに表示され、スクリプトの次のステートメントにより処理が続きます。**SdShowMsg** ウィンドウはセットアップ ウィンドウの中央に位置するので、注意してください。bShow が FALSE の場合、szMsg は無視され **SdShowMsg** ウィンドウは閉じます。

***メモ・SdShowMsg** ウィンドウが開いている場合、2 番目のパラメーターを TRUE にした **SdShowMsg** の次の呼び出しは無視されます。メッセージを変更するには、最初に 2 番目のパラメーターを FALSE にして **SdShowMsg** を呼び出してウィンドウを閉じ、次に szMsg に新しいメッセージを使い 2 番目のパラメーターを TRUE にして **SdShowMsg** をもう一度呼び出す必要があります。*

構文

SdShowMsg (szMsg, bShow);

パラメーター

テーブル 102・SdShowMsg のパラメーター

パラメーター	説明
szMsg	ウィンドウに表示するメッセージを指定します。デフォルトのメッセージ「セットアップはインストールした機能を探しています」を表示するには、このパラメーターにヌル文字列 (“”) を渡します。このパラメーターは、bShow が FALSE の場合無視されます。 **SdShowMsg** は、メッセージを 1 行で表示するように作られています。改行文字 (¥n) を szMsg に組み込まないでください。 ***メモ・**szMsg の値を 1 行で表示する **SdShowMsg** ウィンドウの左右のサイズが指定されます。メッセージの長さがウィンドウの最大幅を超える場合、メッセージはウィンドウにおさまるよう切り詰められます。*
bShow	ウィンドウを開くか閉じるかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ TRUE— ウィンドウが開いていない場合は、開きます。 ・ FALSE— ウィンドウが開いている場合は、閉じます。

戻り値

この関数は常に 0 を返します。

追加情報

・ **SdShowMsg** 関数によって表示されるダイアログは、スキンを使って表示することはできません。スキンの指定に関わらず、同じように表示されます。

- ダイレクト エディターはタイトル バーを持たない **SdShowMsg** などのダイアログをサポートしません。したがって、このダイアログは、編集可能なダイアログの 1 つとして [ダイアログ] ビューには表示されません。このダイアログをカスタマイズするには、**SdShowMsg** 呼び出しを使います。

SdShowMsg の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SdShowMsg ダイアログのデモンストレーションを行います。
 *
 * メッセージを 3 秒間表示するため SdShowMsg が呼び出されます。
 * そしてメッセージを画面から削除するために、再び呼び出されます。
 * SdShowMsg への最後の呼び出しで別のメッセージを表示します。
 *
¥*---------------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
  STRING szMsg;
begin

  // 画面にメッセージを表示します。
  szMsg = " このメッセージは 3 秒間表示されます。";
  SdShowMsg (szMsg, TRUE);

  // 3 秒間遅延させます。
  Delay (3);

  // 画面からメッセージを削除します。
  SdShowMsg (szMsg, FALSE);

  // 画面に別のメッセージを表示します。
  szMsg = " これは別のメッセージで、" +
    "3 秒間表示されます。";
  SdShowMsg (szMsg, TRUE);
  Delay(3);

  // 画面からメッセージを削除します。
  SdShowMsg (szMsg, FALSE);

end;

// ソースファイル :Is5fn157.rul
```

SdStartCopy

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdStartCopy 関数は複数行の編集フィールドを作成して、インストール時に行われた設定や選択を表示します。エンドユーザーは必要に応じて設定を変更するために、ダイアログの [戻る] ボタンをクリックして前のダイアログに戻ることができます。ユーザーの選択を取り出して、ファイル転送処理が始まる前に SdStartCopy を呼び出します。

文字列リストを使って、インストール中に取得した情報を収集します。次にこの文字列をパラメーター listData の SdStartCopy に渡します。SdStartCopy が表示するリストを使って、ユーザーはファイル転送処理を続ける前に、情報が正しいことを確認できます。

構文

SdStartCopy (szTitle, szMsg, listData);

パラメーター

テーブル 103 - SdStartCopy のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル「ファイル コピーの開始」を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg	複数行編集フィールドの上の静的テキストフィールドに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
listData	エンドユーザーから取り出した情報の文字列リストを指定します。SdStartCopy は各要素を自動的に複数行の編集フィールドに配置します。ListCreate を呼び出すことによって listData が初期化されていない場合、複数行の編集フィールドは非表示になって静的テキストフィールドだけが表示されます。

戻り値

テーブル 104 - SdStartCopy の戻り値

戻り値	説明
NEXT (1)	ユーザーが、[次へ] ボタンを選択したことを示します。
BACK (12)	ユーザーが、[戻る] ボタンを選択したことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdStartCopy の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdRegisterUserEx と SdSetupType を利用した
* SdStartCopy 関数をデモンストレーションします。
*
* SdRegisterUserEx と SdSetupType が呼び出され、ユーザーからインストール情報が
* 収集されます。情報は
```

```
＊SdStartCopy が表示します。
＊
¥*---------------------------------------------------------------*/

#include "Ifx.h"

function OnBegin()
    STRING  szTitle, szMsg, svName, svCompany, svSerial, svDir, svSetupType;
    LIST    listData;
    NUMBER  nResult;
begin

start:
    // リストを作成します。
    listData = ListCreate (STRINGLIST);

    // SdRegisterUserEx への呼び出し用メッセージパラメーターをセットアップします。
    szMsg = " 名前、会社名、及びシリアル番号を入力してください。";

    // 登録情報を読み出します。
    SdRegisterUserEx ("Registration", szMsg, svName, svCompany, svSerial);

    // リストへ情報を追加します。
    ListAddString (listData, " ユーザー情報 : ", AFTER);
    ListAddString (listData, "          " + svName, AFTER);
    ListAddString (listData, "          " + svCompany, AFTER);
    ListAddString (listData, "          " + svSerial, AFTER);
    ListAddString (listData, "", AFTER);

SetupTypeLabel:
   // SdSetupType を呼び出すパラメーターをセットアップします。
    szMsg = " ボタンの 1 つをクリックしてインストールする種類を選んでください。";
    svDir = TARGETDIR;

    // セットアップの種類とディレクトリ情報を読み出します。
    nResult = SdSetupType(" セットアップの種類を選択 ", szMsg, svDir, 0);

    // 選択されたセットアップの種類を説明する文字列を作成します。
    switch (nResult)
        case TYPICAL:
            svSetupType = "TYPICAL: アプリケーションは " +
                          " 最も一般的なオプションと共にインストールされます。";
        case COMPACT:
            svSetupType = "COMPACT: アプリケーションは " +
                          " 必要最小限のオプションと共にインストールされます。";
        case CUSTOM:
            svSetupType = "CUSTOM: インストールするオプション " +
                          " を選択します。";
        case BACK:
            goto start;
        デフォルト :
            MessageBox (" 選択したセットアップの種類は無効です !", SEVERE);
            abort;
    endswitch;

    // リストへセットアップの種類情報を追加します。
    ListAddString(listData, " セットアップの種類 :", AFTER);
    ListAddString(listData, "          " + svSetupType, AFTER);
    ListAddString(listData, "", AFTER);
```

```
    ListAddString(listData, " インストール先ディレクトリ :", AFTER);
    ListAddString(listData, "              " + svDir, AFTER);

    // SdStartCopy を呼び出すタイトルとメッセージパラメーターをセットアップします。
    szTitle = "SdStartCopy の例 ";
    szMsg   = " セットアップには、ファイル転送処理を開始するのに充分な情報があります ¥n" +
              " 設定を確認するには ¥n" +
              "[ 戻る ] をクリックしてください。設定に満足な場合は、¥n" +
              "[ 次へ ] をクリックしてファイルのコピーを開始します。";

    // ユーザーの選択を表示するため、SdStartCopy を呼び出します。
    nResult = SdStartCopy (szTitle, szMsg, listData);

    // ユーザーが SdStartCopy ダイアログを終了する処理を行います。
    switch(nResult)
        case NEXT:
            MessageBox ("SdStartCopy が成功しました。",INFORMATION);
        case BACK:
            goto SetupTypeLabel;
        デフォルト :
            MessageBox ("SdStartCopy が失敗しました。", SEVERE);
    endswitch;

end;
```

SdStartCopy2

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdStartCopy2 関数は、ユーザーに対して転送処理の開始を通知します。ユーザーは必要に応じて設定を変更するために、[戻る] ボタンをクリックして前のダイアログに戻ることができます。ユーザーの選択を取り出して、ファイル転送処理が始まる前に SdStartCopy2 を呼び出します。

構文

```
SdStartCopy2 ( szTitle, szMsg );
```

パラメーター

テーブル 105 - SdStartCopy2 のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (「プログラムのインストールの準備完了」) を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg	静的テキストフィールドに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。

戻り値

テーブル 106 - SdStartCopy2 の戻り値

戻り値	説明
NEXT	ユーザーが、[インストール] ボタンを選択したことを示します。
BACK	ユーザーが、[戻る] ボタンを選択したことを示します。
< ISERR_SUCCESS	ダイアログが表示されなかったことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdStartCopy2 の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdProductName 関数、SdWelcome 関数、SdStartCopy2 関数、そして SdFinish 関数の
* デモンストレーションを行います。
*
* まず、SdProductName が呼び出され、製品名を % プレースホルダーの
* 代替として設定します。これは、SdWelcome 関数、
* SdWelcome 関数、そして SdFinish 関数へ渡されるメッセージ文字列に埋め込まれます。
*
* 次に、SdWelcome が呼び出され、ようこそメッセージが表示されてから、
```

```
＊SdStartCopy2 が呼び出されます。最後に、SdFinish への呼び出しで
＊ユーザーに最後のオプションを提供し、セットアッププロセスを完了します。
＊
＊メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
＊　　[ サポート ファイル / ビルボード ] ビューの Windows Notepad 実行可能ファイルと
＊　　完全修飾名と有効なテキスト ファイルを
＊　　参照するように設定してください。
＊
¥*---------------------------------------------------------------*/



// README FILE へテキスト ファイルの名前を割り当てます

#define NOTEPAD "C:¥¥Windows¥¥Notepad.exe"
#define README FILE ""

  STRING   szProductName, szTitle, szMsg, szFeatures;
  STRING   szMsg1, szMsg2, szOpt1, szOpt2;
  BOOL     bvOpt1, bvOpt2;
  NUMBER nReturn;

#include "ifx.h"

function OnBegin()
begin

  // 製品名を %P プレースホルダーの代わりに設定します。
  szProductName = " 自分のアプリケーション ";
  SdProductName (szProductName);

SdWelcomeLabel:
  szTitle = "SdWelcome の例 ";

  // セットアップダイアログで [ 戻る ] ボタンを無効にします。
  Disable(BACKBUTTON);

  // SdWelcome ダイアログを表示します。2 番目のパラメーターのヌル文字列は
  // デフォルト メッセージを指定します。これは %P プレースホルダーを利用します。
  SdWelcome (szTitle, "");

  // セットアップダイアログで [ 戻る ] ボタンを有効にします。
  Enable(BACKBUTTON);

SdStartCopy2Label:
  szTitle = "SdStartCopy2 の例 ";
  // SdStartCopy2 ダイアログを表示します。
  if (SdStartCopy2 (szTitle, szMsg) = BACK) then
    goto SdWelcomeLabel;
  endif;

  // %P プレースホルダーは、SdFinishUpdate に渡される文字列パラメーターの
  // いくつかの文字列パラメーターに埋め込まれます。
  szTitle = "SdFinish の例 ";
  szMsg1  = "%P セットアップがほぼ完了しました。¥n" +
    " 下のオプションを選択してください。";
  szMsg2   = "[ 完了 ] をクリックして %P セットアップを終了します。";
  szOpt1  = "README を読む。";
  szOpt2  = "%P を起動する。";
```

```
  // SdFinish ダイアログを表示します。
  SdFinish (szTitle, szMsg1, szMsg2, szOpt1, szOpt2, bvOpt1, bvOpt2);

  if (bvOpt1) then
    // readme ファイルを表示します。
    LaunchAppAndWait (NOTEPAD, README.FILE, LAAW_OPTION_WAIT );
  endif;

  if (bvOpt2) then
    // この例は実際にアプリケーションをインストールしないので
    // メッセージボックスは、通常 LaunchApp の呼び出しが行われる
    // この場所に表示されます。
    // 例えば :
    //   LaunchApp (TARGETDIR^PROGRAMEXECUTABLE,"");

    SprintfBox (INFORMATION, szTitle, "%s をここで起動します。", szProductName);
  endif;

end;
```

SdSubstituteProductInfo

SdSubstituteProductInfo 関数は、svString 内にあるすべての %P、%VS、そして %VI プレースホルダーをシステム変数 IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、そして IFX_INSTALLED_DISPLAY_VERSION の値と置き換えます。この関数は、文字列を表示する前にこの置換を自動的に実行しない MessageBox などの関数を呼び出す前に利用することができます。

構文

```
SdSubstituteProductInfo ( svString );
```

パラメーター

テーブル 107 - SdSubstituteProductInfo のパラメーター

パラメーター	説明
svString	プレースホルダーを含む文字列を指定し、プレースホルダーが置換された文字列を戻します。

戻り値

この関数は値を返しません。

SdWelcome

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

- *InstallScript*
- *InstallScript MSI*

SdWelcome 関数は、エンドユーザーに対してようこそメッセージを表示するダイアログを作成します。

構文

SdWelcome (szTitle, szMsg);

パラメーター

テーブル 108 - SdWelcome のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルに表示するテキストを指定します。デフォルトのタイトル「ようこそ」を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg	[ようこそ] ダイアログに表示するメッセージを指定します。以前に SdProductName を呼び出した際に設定した製品名をこのメッセージに含めるには、メッセージ文字列の任意の場所に %P プレースホルダーを挿入します。メッセージが表示されると、%P は製品名に置き換えられます。デフォルトのタイトル ようこそメッセージ「%P セットアップ プログラムへようこそ。このプログラムはお使いのコンピューターに %P をインストールします」を表示するには、このパラメーターにヌル文字列 ("") を渡します。

戻り値

テーブル 109 - SdWelcome の戻り値

戻り値	説明
NEXT (1)	[次へ] ボタンがクリックされたことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdWelcome の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdWelcome 関数のデモンストレーションを行います。
```

```
 *
 * SdWelcome 関数は、エンドユーザーに [ ようこそ ] ダイアログを
 * 表示します。
 *
¥*---------------------------------------------------------------------*/

#include "ifx.h"

function OnBegin()
    STRING  szTitle, szProductName;
begin

    // セットアップダイアログで [ 戻る ] ボタンを無効にします。
    Disable(BACKBUTTON);

    // 製品名を %P プレースホルダーの代わりに設定します。
    szProductName = " 自分のアプリケーション ";
    SdProductName (szProductName);

    szTitle = "SdWelcome の例 ";

    // SdWelcome ダイアログを表示します。2 番目のパラメーターのヌル文字列は
    // デフォルト メッセージを指定します。これは %P プレースホルダーを利用します。
    SdWelcome (szTitle, "");

end;
```

SdWelcomeMaint

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SdWelcomeMaint 関数は、メンテナンス セットアップ (つまり、すでに実行したセットアップの再実行) の開始時に使用するためのダイアログを表示します。ダイアログには [変更] 、[修復]、および [削除] オプションボタンが含まれています。

構文

```
SdWelcomeMaint (szTitle, szMsg, nType);
```

パラメーター

テーブル 110 ・ SdWelcomeMaint のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル (「Welcome」) を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
nType	デフォルトの選択にするオプションボタンを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ MODIFY—[変更] ボタンがデフォルトの選択になります。 ・ REPAIR—[修復] ボタンがデフォルトの選択になります。 ・ REMOVEALL—[削除] ボタンがデフォルトの選択になります。

戻り値

テーブル 111 ・ SdWelcomeMaint の戻り値

戻り値	説明
MODIFY (301)	エンドユーザが [次へ] ボタンをクリックしたとき、[変更] ボタンが選択されたことを示します。
REPAIR (302)	エンドユーザが [次へ] ボタンをクリックしたとき、[修正] ボタンが選択されたことを示します。
REMOVEALL (303)	エンドユーザが [次へ] ボタンをクリックしたとき、[削除] ボタンが選択されたことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SdWelcomeMaint の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------*¥
 *
```

```
＊InstallShield スクリプトの例
＊
＊SdWelcomeMaint 関数のデモンストレーションを行います。
＊
＊  SdWelcomeMaint 関数はメンテナンスセットアップ
＊の開始時に使用するメッセージを表示します。
＊( 既に実行されているセットアップの再実行 )。. ダイアログは  [ 更新 ]、
＊[ 修復 ]、および [ 削除 ] オプション ボタンが含まれています。
＊
¥*---------------------------------------------------------*/

#include "ifx.h"

function OnBegin()
   STRING  szTitle, szProductName;
   NUMBER  nType, nReturn;
begin

   // セットアップダイアログで [ 戻る ] ボタンを無効にします。
   Disable(BACKBUTTON);

   // 製品名を %P プレースホルダーの代わりに設定します。
   szProductName = " 自分のアプリケーション ";
   SdProductName (szProductName);

   szTitle = "SdWelcomeMaint の例 ";
   // デフォルトの選択にするオプションボタンを指定します。
   nType = REMOVEALL;

   // SdWelcomeMaint ダイアログを表示します。2 番目のパラメーターのヌル文字列は
   // デフォルト メッセージを指定します。これは %P プレースホルダーを利用します。
   //
   nReturn = SdWelcomeMaint (szTitle, "", nType);

   switch(nReturn)

     case MODIFY: MessageBox("SdWelcomeMaint 選択 : Modify", 0);

     case REPAIR: MessageBox("SdWelcomeMaint selection: 修復 ", 0);

     case REMOVEALL: MessageBox("SdWelcomeMaint selection: 削除 ", 0);

   endswitch;

end;
```

SeekBytes

SeekBytes 関数は、開いたバイナリファイル内でファイルポインターを再配置します。ファイルポインターは、現在の位置に関連付けて、またはファイルの最初および終わりと関連付けて特定のバイト数移動させることができます。

***メモ・** SeekBytes を呼び出す前に、OpenFileMode や OpenFile を呼び出してファイルをバイナリモードで開かなくてはなりません (インターネット上のファイルも可)。ファイルへバイトを書き込んだ後、CloseFile を呼び出してファイルを選択します。*

構文

```
SeekBytes ( nFileHandle, nBytes, nPosition );
```

パラメーター

テーブル 112 ・ SeekBytes のパラメーター

パラメーター	説明
nFileHandle	バイナリモードで開いたファイルへのファイルハンドルを指定します。
nBytes	nPosition が指定した位置に関連付けてファイルポインターの移動バイト数を指定します。 n Bytes が正数の場合、ファイルポインターはファイルの終わりに移動します。 n Bytes が負数の場合、ファイルポインターはファイルの始まりに移動します。
nPosition	ポインター nBytes を移動するファイルの場所を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ FILE_BIN_CUR Ñ ポインター nBytes を現在の位置から移動します。 ・ FILE_BIN_END Ñ ポインター nBytes をファイルの終わりから移動します。 ・ FILE_BIN_START Ñ ポインター nBytes をファイルの始まりから移動します。

戻り値

テーブル 113 ・ SeekBytes の戻り値

戻り値	説明
0	関数がポインターを再配置したことを示します。
< 0	関数がポインターを再配置できなかったことを示します。

SeekBytes の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
```

```
*
* ReadBytes 関数と SeekBytes 関数のデモンストレーションを行います。
*
* SeekBytes が呼び出され、バイナリ モードで開かれたファイルの
* 特定の場所へファイルポインターを配置
* します。そして ReadBytes はこの場所からはじまる特定の数のバイトを
* 読み取ります。バイトは文字列に読み込まれ、
* そしてメッセージボックスに表示されます。
*
* メモ : 定義された定数 EXAMPLE_DIR and EXAMPLE_BIN は
*       ターゲットシステムの既存のディレクトリとファイルに設定しなくては
*      設定してください。
*
¥*---------------------------------------------------------------*/

#define EXAMPLE_DIR  "C:¥¥"
#define EXAMPLE_BIN "Example.bin"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_SeekBytes(HWND);

function ExFn_SeekBytes(hMSI)
   STRING svString;
    NUMBER nvFileHandle;
begin

   // ファイルモードを読み取り / 書き込みモードに設定します。
   OpenFileMode (FILE_MODE_BINARY);

   // バイナリファイルを開きます。
   if (OpenFile (nvFileHandle, EXAMPLE_DIR, EXAMPLE_BIN) < 0) then

      // エラーをレポートしてから、中止します。
      SprintfBox (SEVERE, "CopyBytes の例 ", "%s を開くことができませんでした。",
                  EXAMPLE_BIN);
      abort;

   endif;

   // ファイルポインターをファイルの 16 番目のバイトへ設定します。
   SeekBytes (nvFileHandle, 15, FILE_BIN_START);

   // 次の 28 バイトを svString へ読み込みます。
   if (ReadBytes (nvFileHandle, svString, 0, 28) < 0) then

      // エラーを報告します。
      MessageBox ("ReadBytes が失敗しました。", SEVERE);
   else

      // 文字列を表示します。
      SprintfBox (INFORMATION, "ReadBytes の例 ", " バイトは : %s",
svString);

   endif;

   // ファイルを閉じます。
   CloseFile (nvFileHandle);
```

```
end;
```

SelectDir

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

- *InstallScript*
- *InstallScript MSI*

SelectDir 関数の代わりに、**SelectDirEx** 関数が使われるようになりました。新しいインストール プログラムの作成には **SelectDirEx** 関数を使用してください。nFlags を BIF_RETURNONLYFSDIRS | BIF_EDITBOX に設定して **SelectDirEx** 関数を呼び出すと、bCreate を FALSE に設定して **SelectDir** 関数を呼び出した時と同じ結果が得られます。

SelectDir 関数は、エンドユーザーがアプリケーションをインストールするフォルダーを指定できるダイアログを表示します。エンドユーザーは完全修飾フォルダー名を入力したり既存のフォルダーをリストから選択することができます。エンドユーザーが無効のフォルダー名や完全でないフォルダー名を入力した場合、メッセージボックスが表示されエンドユーザーに有効な名前を入力するよう要求します。選択したフォルダーの完全修飾名が、svDir で返されます。

指定されたフォルダーが存在せず、bCreate パラメーターが TRUE の場合、**SelectDir** は自動的に指定されたフォルダーを作成します。パラメーター bCreate が FALSE に設定され存在しないフォルダーが選択されると、エンドユーザーには通知されず、**SelectDir** はフォルダーを作成しません。この場合、svDir に含まれる選択の操作はお任せします。

***メモ・SelectDir** は、エンドユーザーが **AskDestPath**、**SdAskDestPath** や、フォルダー名を取得する他の InstallShield 関数によって表示されるダイアログの 参照 ボタンをクリックすると自動的に呼び出されます。*

Windows はこのダイアログを表示するため、インストールがダイアログ上にあるボタンのテキストを変更することはできません。Windows が、ボタンテキストをオペレーティング システムの言語で表示 (英語システムでは "Yes" または "No") するため、このテキストを手動でローカライズする必要はありません。さらに高度な柔軟性があるダイアログが必要な場合は、Windows API 関数を直接呼び出すか、カスタム ダイアログを使用します。

構文

```
SelectDir (szTitle, szMsg, svDir, bCreate);
```

パラメーター

テーブル 114 · SelectDir のパラメーター

パラメーター	説明
szTitle	このダイアログのタイトルを指定します。デフォルトのタイトル (「フォルダーの選択」) を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	このダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
svDir	デフォルトの選択として表示するフォルダーの名前を指定します。エンドユーザーが選択したフォルダーの完全修飾名を返します。
bCreate	指定したフォルダーが存在しない場合、InstallShield で作成するかどうかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · TRUE Ñ フォルダーがない場合は、作成することを示します。 · FALSE– フォルダーがない場合、作成しないことを示します。

戻り値

bCreate が FALSE の場合、この関数は次の値のうちの 1 つを戻します :

テーブル 115 · bCreate が FALSE の場合の SelectDir の戻り値

戻り値	説明
IDOK (1)	[OK] ボタンが選択されたことを示します。
IDCANCEL (2)	[キャンセル] ボタンが選択されたことを示します。
< 0	関数がダイアログを表示できなかったことを示します。

bCreate が TRUE の場合、この関数は次の値のうちの 1 つを戻します :

テーブル 116 · bCreate が TRUE の場合の SelectDir の戻り値

戻り値	説明
IDCANCEL (2)	[キャンセル] ボタンが選択されたことを示します。
0	[OK] ボタンが選択されて、必要に応じて関数が指定のフォルダーを作成したことを示します。
< 0	関数がダイアログを表示できなかったか、または指定のフォルダーを作成できなかったことを示します。

追加情報

- この関数によって表示されるダイアログは、スキンを使って表示することはできません。スキンの指定に関わらず、同じように表示されます。
- InstallShield Professional の以前のバージョンでは、bCreate が TRUE に設定されていて、エンドユーザーが存在しないフォルダーを選択すると、フォルダーを作成するかどうかをたずねる確認メッセージが表示されていました。このメッセージは多くのエンドユーザーに混乱を招くため、InstallShield では削除されました。

SelectDir の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SelectDir 関数のデモンストレーションを行います。
 *
 * このスクリプトは SelectDir を呼び出して、ユーザーがフォルダーを特定することができる
 * ダイアログを表示します。指定されたフォルダーが
 * 存在しない場合、それが作成されます。その代わり、エラーメッセージが
 * が表示され、SelectDir ダイアログ ボックスが再度表示されます。
 *
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "SelectDir の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_SelectDir(HWND);

function ExFn_SelectDir(hMSI)
   STRING  szTitle, szMsg, svDir;
    BOOL    bCreate, bFolderExists;
    NUMBER  nResult;
begin

  // ユーザーがキャンセルするか既存のフォルダーを選択するまでループします。
  repeat

     //SelectDir ダイアログのデフォルト フォルダーを設定します。
     svDir = INSTALLDIR;

     //SelectDir ダイアログ ボックスに表示するメッセージを設定します。
     szMsg   = " 既存のフォルダーを選択してください。";

     // ユーザーから既存のフォルダー名を取得します。4 番目の
     // パラメーターは、存在しないフォルダーは作成されないことを
     // 示します。
     nResult = (SelectDir (TITLE_TEXT, szMsg, svDir, FALSE) < 0) ;
```

```
    if nResult = 0 then

      // そのフォルダーが存在するか判断します。
      bFolderExists = ExistsDir (svDir);

      if bFolderExists = NOTEXISTS then
        // フォルダーは存在しません。ユーザーに再度選択するよう求めます。
        szMsg = "%s は存在しません。¥n 既存のフォルダーを選択してください。";
        SprintfBox  (WARNING, szTitle, szMsg, svDir);
      endif;

    endif;

  until (nResult = CANCEL) || (bFolderExists = EXISTS);

  if (bFolderExists = EXISTS) then

    // 選択されたフォルダー名を表示します。
    SprintfBox (INFORMATION, szTitle, "%s を選択しました。", svDir);

  endif;

end;
```

SelectDirEx

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SelectDirEx 関数は、エンドユーザーがアプリケーションをインストールするフォルダーを指定できるダイアログを表示します。新しいフォルダーを指定できるように編集ボックスを表示することもできます。

この関数は Windows API 関数 SHBrowseForFolde を呼び出して、ダイアログを表示します。SHBrowseForFolder についてのさらに詳しい情報は Windows API マニュアル を参照してください。

メモ・Windows はこのダイアログを表示するため、インストールがダイアログ上にあるボタンのテキストを変更することはできません。Windows が、ボタンテキストをオペレーティング システムの言語で表示 (英語システムでは "Yes" または "No") するため、このテキストを手動でローカライズする必要はありません。さらに高度な柔軟性があるダイアログが必要な場合は、Windows API 関数を直接呼び出すか、カスタム ダイアログを使用します。

構文

```
SelectDirEx ( szTitle, szMsg, szEditBoxStatusText, szTreeControlStatusText, nFlags, svDir );
```

パラメーター

テーブル 117 ・ SelectDirEx のパラメーター

パラメーター	説明
szTitle	このダイアログのタイトルを指定します。デフォルトのタイトル ([フォルダーの選択]) を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szMsg	このダイアログに表示するメッセージを指定します。このダイアログのデフォルトの指示を表示するには ([インストールフォルダーを選択してください])、このパラメーターにヌル文字列 ("") を渡してください。
szEditBoxStatusText	nFlags が BIF_EDITBOX. を指定した場合、編集ボックスに伴う静的テキストを指定します。nFlags が BIF_EDITBOX を指定しない場合、このパラメーターは無視されます。
szTreeControlStatusText	nFlags が BIF_STATUSTEXT を指定すると、静的テキストがダイアログのツリーコントロールを伴うよう指定します。nFlags が BIF_STATUSTEXT を指定しない場合、このパラメーターは無視されます。

テーブル 117 • SelectDirEx のパラメーター (続き)

パラメーター	説明
nFlags	Windows の BROWSEINFO 構造で使用するフラグと同じフラグを指定することにより、関数によって表示されるダイアログの外観と機能を指定します。BIF_BROWSEFORCOMPUTER または BIF_BROWSEFORPRINTER を渡した場合、編集ボックスは表示されません。以下の定数のどれかを渡します。 • BIF_BROWSEFORCOMPUTER Ñ エンドユーザーはネットワーク上にあるコンピューターを選択することができます。ツリーコントロール内で有効なコンピューターが選択された場合のみ、[ネットワークコンピューター] があらかじめ選択され、[OK] ボタンが有効になります。BIF_EDITBOX が指定されても、編集ボックスは表示されません。 • BIF_BROWSEFORPRINTER Ñ エンドユーザーがプリンターを選択することができます。[マイコンピューター] フォルダーがツリーコントロールであらかじめ選択されます。少なくとも 1 台のプリンターを持つコンピューターだけが [ネットワークコンピューター] フォルダー下に表示されます。ツリーコントロールで有効なコンピューター名を選択した場合だけ、[OK] ボタンが有効になります。BIF_EDITBOX が指定されても、編集ボックスは表示されません。 • BIF_DONTGOBELOWDOMAIN Ñ ドメイン レベルの下にあるネットワークフォルダーは、ツリー コントロールには表示されません。 • BIF_RETURNFSANCESTORS Ñ svDir の上のフォルダーを表示します。. • BIF_RETURNONLYFSDIRS Ñ ファイル システム フォルダーを参照します。 次の定数は、ダイアログの他の側面を指定します : • BIF_STATUSTEXT Ñ ダイアログにステータス テキストを表示します。 • BIF_EDITBOX Ñ [参照] ダイアログへの編集フィールドを追加します。エンドユーザーは編集ボックスにフォルダー名を入力することができます。SzEditBoxStaticText パラメーターに指定したテキストは、BIF_BROWSEFORCOMPUTER または BIF_BROWSEFORPRINTER が指定されない限り、編集ボックスの上に表示されます。エンドユーザーが [OK] をクリックすると、SelectDirEx は有効なフォルダー名が入力されたかをチェックします。それ以外の場合、エラーメッセージが表示され、ダイアログが閉じます。
svDir	デフォルトの選択として表示するフォルダーの名前を指定します。エンドユーザーが選択したフォルダーの完全修飾名を返します。このパラメーターでターゲットシステムに存在する有効なフォルダー名を指定した場合、ツリーコントロールで指定されたフォルダーが前もって選択されます。

戻り値

テーブル 118 - SelectDirEx の戻り値

戻り値	説明
IDOK (1)	エンド ユーザーが、[OK] ボタンをクリックしたことを示します。
IDCANCEL (2)	エンド ユーザーが、[キャンセル] ボタンをクリックしたことを示します。
< 0	関数がダイアログを表示できなかったことを示します。

追加情報

この関数によって表示されるダイアログは、スキンを使って表示することはできません。スキンの指定に関わらず、同じように表示されます。

SelectDirEx の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SelectDirEx 関数を示します。
 *
 * このスクリプト例では SelectDirEx を呼び出して、ユーザーがフォルダーを指定することができる
 * ダイアログを表示します。指定したフォルダーが存在しない
 * 場合は、それが作成されます。またはエラーメッセージが
 * 表示され、SelectDirEx ダイアログが再度表示されます。
 *
 *
¥*---------------------------------------------------------------------*/

#include "ifx.h"

function OnBegin()
  STRING  szTitle, szMsg, svDir;
  BOOL    bCreate, bFolderExists;
  NUMBER  nResult;
begin
  // ユーザーがキャンセルするか既存のフォルダーを選択するまでループします。
  repeat
    //SelectDirEx ダイアログのデフォルト フォルダーを設定します。
    svDir = TARGETDIR;

    //SelectDirEx ダイアログに表示するタイトルを設定します。
    szTitle = "SelectDirEx の例 ";
```

```
    //SelectDirEx ダイアログに表示するメッセージを設定します。
    szMsg  = " 既存のフォルダーを選択してください。";

    // ユーザーから既存のフォルダー名を取得します。
    nResult = (SelectDirEx (szTitle, szMsg, "", "",
      BIF_RETURNONLYFSDIRS | BIF_EDITBOX, svDir ) < 0) ;

    if nResult = 0 then
      // そのフォルダーが存在するか判断します。
      bFolderExists = ExistsDir (svDir);

      if bFolderExists = NOTEXISTS then
        // フォルダーは存在しません。ユーザーに再度選択するよう求めます。
        szMsg = "%s は存在しません。¥n 既存のフォルダーを選択してください。";
        SprintfBox  (WARNING, szTitle, szMsg, svDir);
      endif;
    endif;
  until (nResult = CANCEL) || (bFolderExists = EXISTS);

  if (bFolderExists = EXISTS) then
    // 選択されたフォルダー名を表示します。
    SprintfBox (INFORMATION, szTitle, "%s を選択しました。", svDir);
  endif;
end;
```

SelectFolder

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SelectFolder 関数は、エンド ユーザーがプログラムフォルダーの名前を編集フィールドに入力したり、プログラムフォルダーをリストから選択できるダイアログを表示します。この関数は、システムのすべてのプログラムフォルダーを自動的に表示します。svDefFolder で渡されるデフォルドのフォルダー名は、編集フィールドに表示されます。選択したフォルダー名は svResultFolder で返されます。

注意・フォルダーが存在しない場合、CreateProgramFolder を呼び出して作成する必要があります。SelectFolder はフォルダーを作成しません。

構文

SelectFolder (szTitle, szDefFolder, svResultFolder);

パラメーター

テーブル 119 ・ SelectFolder のパラメーター

パラメーター	説明
szTitle	ダイアログのタイトルを指定します。デフォルトのタイトル [プログラムフォルダーの選択] を表示するには、このパラメーターにヌル文字列 (″″) を渡します。
szDefFolder	デフォルトフォルダーとして表示するフォルダーの名前を指定します。
svResultFolder	エンドユーザーが選択または指定したフォルダーの名前を返します。

戻り値

テーブル 120 ・ SelectFolder の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。
< 0	関数が失敗したことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SelectFolder の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SelectFolder 関数のデモンストレーションを行います。
 *
 * まず、SelectFolder が呼び出され、ユーザーからフォルダーの選択を
 *   取得します。選択されたフォルダー名をが表示されます。
 *
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT ″SelectFolder の例 ″
```

```
// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_SelectFolder(HWND);

function ExFn_SelectFolder(hMSI)
  STRING svResultFolder;
   NUMBER nReturn;
begin

  // エンドユーザーからフォルダーの選択を取得します。"Startup" をデフォルトの選択に
  // します。
  nReturn = SelectFolder (TITLE_TEXT, "Startup", svResultFolder);

  if (nReturn < 0) then

    // エラーを報告します。
    MessageBox ("SelectFolder が失敗しました。", SEVERE);

  else

    // 選択されたフォルダー名を表示します。
    SprintfBox (INFORMATION, TITLE_TEXT,
          " 選択されたフォルダー : %s", svResultFolder);

  endif;

end;
```

SendMessage

SendMessage 関数はメッセージを 1 つまたは複数のウィンドウに送ります。SendMessage はメッセージが処理されるまでセットアップスクリプトへコントロールを戻しません。SendMessage 関数は Windows API SendMessage への直接パススルーです。詳しい情報は、Windows プログラミングマニュアルを参照してください。

メモ・パラメーター nMsg を利用してメッセージを送る、または戻り値を処理するには、Windows.h で定義された定数に対応するスクリプト内の定数を定義しなくてはなりません。#include を使って Windows.h をスクリプトに含めることはできません。

構文

```
SendMessage ( nHwnd, nMsg, nwParam, nlParam );
```

パラメーター

テーブル 121 ・ SendMessage のパラメーター

パラメーター	説明
nHwnd	メッセージを受け取るウィンドウを識別するハンドルを指定します。
nMsg	ウィンドウへ送るメッセージを指定します。
nwParam	追加メッセージ情報を指定します。
nlParam	追加メッセージ情報を指定します。

戻り値

SendMessage は、同じ名前の Windows API の呼び出しから受け取った値を戻します。戻り値は Windows API SendMessage が受け取ったメッセージによって異なります。Windows API SendMessage が戻すメッセージについての詳細情報は、Windows プログラミングマニュアルを参照してください。

追加情報

タスク ***SendMessage の 4 番目のパラメーターに文字列データを渡すには、以下の手順を実行します。***

1. InstallShield Professional から変換されたプロジェクトの場合、次のいずれかを実行して必要な関数宣言を含めます。
   - [設定] ダイアログ ボックスの [コンパイル / リンク] タブにある [プリプロセッサ定義] ボックスから、プリプロセッサ定数 ISINCLUDE_NO_WINAPI_H を削除します。(このダイアログ ボックスには、[ビルド] メニューの [設定] をクリックしてアクセスします。)
   - 次のコードをスクリプトに配置します。

     ```
     prototype USER.SendMessageA(HWND, NUMBER, NUMBER, BYREF STRING);
     ```

     (InstallShield で作成されたプロジェクトの場合、デフォルトでは、**ISRTWindows.h** は **Ifx.h** に含まれます。このとき、Ifx.h もデフォルトでスクリプトに含まれます。)
2. 次の例に従って関数を呼びます。

   ```
   USER.SendMessageA( hWnd, LB_GETTEXT, nResult, sResult);
   ```

SendMessage の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
```

```
*
* FindWindow 関数と SendMessage 関数のデモンストレーションを行います。
*
* このスクリプトは Windows Notepad を起動し、
* FindWindow を呼び出して Notepad ウィンドウを検出します。次に、
* SendMessage を呼び出してウィンドウを最大化し、3 秒後に
* SendMessage を再び呼び出してウィンドウを最小化
* します。スクリプトが終了したとき、Windows NotePad は開いた状態ですが、
* 最小化されています。SendMessage へ渡されるパラメーターは
* Windows システム メッセージで、その値は
* スクリプトでは定数として定義されいることに注意してください。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*      定数 NOTEPAD を設定して、Windows Notepad 実行可能ファイルの完全
*    修飾名が参照されるようにしてください。
*
¥*---------------------------------------------------------------------*/

#define NOTEPAD "C:¥¥Windows¥¥Notepad.exe"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_SendMessage(HWND);

function ExFn_SendMessage(hMSI)
   NUMBER nMsg, nwParam, nlParam;
   HWND nHwnd;
begin

   // セットアップの背景ウィンドウを表示しません。
   Disable(BACKGROUND);

   // Windows Notepad を開きます。
   if (LaunchApp (NOTEPAD, "") < 0 ) then
      MessageBox ("Notepad を起動できませんでした。", SEVERE);
      abort;
   endif;

   // 3 秒間待機して、最大化される前にそのウィンドウを
   // 参照できるようにします。
   Delay (3);

   // Notepad ウィンドウのハンドルを読み出します。最初の
   // パラメーターはウィンドウクラスです。2 番目のパラメーターにある
   // ヌル文字列は一番最初の Notepad ウィンドウを指定します。
   nHwnd = FindWindow ("NotePAD", "");

   if (nHwnd = NULL) then
      MessageBox ("Notepad ウィンドウを検出できませんでした。", SEVERE);
   else
      // システム コマンドを送り、ウィンドウを最大化します。
      SendMessage (nHwnd, WM_SYSCOMMAND, SC_MAXIMIZE, 0);

      // 3 秒間待機して、最小化される前にそのウィンドウを
      // 参照できるようにします。
      Delay (3);

      // システム コマンドを送り、ウィンドウを最小化します。
```

```
        SendMessage (nHwnd, WM_SYSCOMMAND, SC_MINIMIZE, nlParam);
    endif;

end;
```

ServiceAddService

ServiceAddService 関数は szServiceName が指定したサービスをシステムで登録されているサービスのリストへ追加します。SERVICE_IS_PARAMS 構造をカスタマイズして、追加サービス作成要素をコントロールすることができます。**ServiceAddService** によってインストールされるファイルはありません。インストール中に各自の責任でサービスをインストールする必要があります。

szServiceName が指定したサービスが既に存在する場合、**ServiceAddService** はそれを関数呼び出しで指定されたパラメーターへ再構成します。**ServiceAddService** が呼び出されたときに既存サービスが実行中の場合、インストーラーはサービスを再構成する前にサービスを停止します。サービスが停止できない場合、**ServiceAddService** はサービスを再構成し、Windows が再起動の後にサービスを再構成できるように BATCH_INSTALL を設定します。

構文

```
ServiceAddService ( szServiceName, szServiceDisplayName, szServiceDescription, szServicePathFile, bStartService, szStartServiceArgs
    );
```

パラメーター

テーブル 122 · ServiceAddService のパラメーター

パラメーター	説明
szServiceName	サービスの名前を指定します。
szServiceDisplayName	サービスの表示名を指定します。
szServiceDescription	サービスの説明を指定します。
szServicePathFile	サービス実行可能ファイルへのパスを指定、またオプションでサービスが開始された時に常に渡されるコマンドラインパラメーターを指定します。
bStartService	追加したあとにサービスを開始するかどうかを指定します。
szStartServiceArgs	インストーラーがサービスをスタートした場合のみサービスへ渡すコマンドライン引数を指定します。

戻り値

テーブル 123 · ServiceAddService の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がサービスを追加または再構成したことを示します。
< ISERR_SUCCESS	関数がサービスを追加または再構成できなかったことを示します。 この関数が失敗した場合、**GetExtendedErrInfo** を呼び出し、3 番目の引数の値を確認することでて追加エラー情報が利用できる場合もあります。(この値は一般的に、Windows API 関数への呼び出しに失敗した後の Windows API 関数 **GetLastError** の内部呼び出しの結果を示します。) 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

ServiceAddService が呼び出されたときにログ記録が有効な場合、サービスが既に存在したかどうか、またアプリケーションがアンインストールされた時にサービスがアンインストールされたかについてアンインストール用にログ記録されます。アンインストールしないサービスを追加する場合、この関数を呼び出す前にログ記録を無効にします。

パブリックで定義された構造 (SERVICE_IS_PARAMS および SERVICE_IS_STATUS) は、サービスが自動的にアンインストールされた時には使用されません。サービスをアンインストールする際にデフォルト以外の設定を指定するには、サービスを追加する際にログ記録を無効にして、アンインストール中にスクリプトから直接 **ServiceRemoveService** を呼び出して手動でサービスを削除します。

ServiceExistsService

ServiceExistsService 関数は szServiceName が指定したサービスがシステムで登録されているかどうかを判断します。

構文

ServiceExistsService (szServiceName);

パラメーター

テーブル 124 · ServiceExistsService のパラメーター

パラメーター	説明
szServiceName	サービスの名前を指定します。

戻り値

テーブル 125 · ServiceExistsService の戻り値

戻り値	説明
TRUE	サービスがシステムで登録されていることを示します。
FALSE	サービスがシステムで登録されていないこと、またはサービスがシステムで登録されているかどうかをセットアップが判断できなかったことを示します。

ServiceGetServiceState

ServiceGetServiceState 関数は svServiceState に szServiceNam が指定したサービスの状態を戻します。

構文

ServiceGetServiceState (szServiceName, nvServiceState);

パラメーター

テーブル 126 · ServiceGetServiceState のパラメーター

パラメーター	説明
szServiceName	サービスの名前を指定します。
nvServiceState	サービスの状態を指定する、次の Windows 定義定数の 1 つを戻します : · SERVICE_STOPPED · SERVICE_START_PENDING · SERVICE_STOP_PENDING · SERVICE_RUNNING · SERVICE_CONTINUE_PENDING · SERVICE_PAUSE_PENDING · SERVICE_PAUSED

戻り値

テーブル 127 · ServiceGetServiceState の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がサービスの状態を読み出したことを示します。サービス状態についての詳しい情報は SERVICE_IS_STATUS システム変数を参照してください。
< ISERR_SUCCESS	関数がサービスの状態を読み出すことができなかったことを示します。 この関数が失敗した場合、**GetExtendedErrInfo** を呼び出し、3 番目の引数の値を確認することでて追加エラー情報が利用できる場合もあります。(この値は一般的に、Windows API 関数への呼び出しに失敗した後の Windows API 関数 **GetLastError** の内部呼び出しの結果を示します。) 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

ServiceInitParams

ServiceInitParams 関数は、SERVICE_IS_PARAMS システム変数のメンバーを次のデフォルト値に初期化します。この関数はセットアップ初期化中に自動的に呼び出されます。

構文

```
ServiceInitParams ( );
```

システム変数メンバー

テーブル 128 · ServiceInitParams のシステム変数メンバー

パラメーター	デフォルト値
SERVICE_IS_PARAMS.lpMachineName	NULL
SERVICE_IS_PARAMS.lpDatabaseName	NULL
SERVICE_IS_PARAMS.dwDesiredAccess	SERVICE_ALL_ACCESS
SERVICE_IS_PARAMS.dwServiceType	SERVICE _WIN32_OWN_PROCESS
SERVICE_IS_PARAMS.dwStartType	SERVICE_AUTO_START
SERVICE_IS_PARAMS.dwErrorControl	SERVICE_ERROR_IGNORE
SERVICE_IS_PARAMS.lpLoadOrderGroup	NULL
SERVICE_IS_PARAMS.lpdwTagId	NULL
SERVICE_IS_PARAMS.lpDependencies	NULL
SERVICE_IS_PARAMS.lpServiceStartName	NULL
SERVICE_IS_PARAMS.lpPassword	NULL
SERVICE_IS_PARAMS.nStartServiceWaitCount	INFINITE
SERVICE_IS_PARAMS.nStopServiceWaitCount	INFINITE

パラメーター

なし。

戻り値

テーブル 129 · ServiceInitParams の戻り値

戻り値	説明
ISERR_SUCCESS	この関数は常に ISERR_SUCCESS を戻します。

ServiceRemoveService

ServiceRemoveService 関数は、サービスコントロールマネージャーデータベースから szServiceName が指定したサービスを削除します。関数が呼び出されたときにサービスが実行中の場合、インストールはサービスを停止してサービスを削除します。

構文

ServiceRemoveService (szServiceName);

パラメーター

テーブル 130 ・ ServiceRemoveService のパラメーター

パラメーター	説明
szServiceName	サービスの名前を指定します。

戻り値

テーブル 131 ・ ServiceRemoveService の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がサービスコントロールマネージャーデータベースからサービスを削除したことを示します。
< ISERR_SUCCESS	関数がサービスを削除できなかったことを示します。 この関数が失敗した場合、**GetExtendedErrInfo** を呼び出し、3 番目の引数の値を確認することでて追加エラー情報が利用できる場合もあります。(この値は一般的に、Windows API 関数への呼び出しに失敗した後の Windows API 関数 **GetLastError** の内部呼び出しの結果を示します。) 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

ServiceStartService

ServiceStartService 関数は szServiceName が指定したサービスを開始します。関数が呼び出されたときにサービスが実行中の場合、インストールは停止してサービスを再開始します。

この関数は、サービスが実行状態に到達するのを待ってから値を返します。少なくとも dwWaitHint ミリ秒おきに SERVICE_IS_STATUS 構造の dwCheckPoint メンバーがアップデートされている間は、いつまでも待ちます。一定期間の後、サービスが開始されたかどうかに関わらず値を戻すことを強制する場合は、SERVICE_IS_PARAMS 構造変数の nStartServiceWaitCount メンバーを適切な秒数に変更します。

構文

ServiceStartService (szServiceName, szStartServiceArgs);

パラメーター

テーブル 132 · ServiceStartService のパラメーター

パラメーター	説明
szServiceName	サービスの名前を指定します。
szStartServiceArgs	サービスへの引数のコンマ区切り文字列を指定します。

戻り値

テーブル 133 · ServiceStartService の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がサービスを開始したことを示します。
< ISERR_SUCCESS	関数がサービスを開始できなかったことを示します。 この関数が失敗した場合、**GetExtendedErrInfo** を呼び出し、3 番目の引数の値を確認することでて追加エラー情報が利用できる場合もあります。(この値は一般的に、Windows API 関数への呼び出しに失敗した後の Windows API 関数 **GetLastError** の内部呼び出しの結果を示します。) 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

ServiceStopService

ServiceStopService 関数は szServiceName が指定したサービスを停止します。

この関数は、結果を戻す前にサービスが停止するまで待機します。つまり、サービスが最低 dwWaitHint ミリ秒単位で SERVICE_IS_STATUS 構造体の dwCheckPoint メンバーを更新している間は無期限で待機します。一定期間の後、サービスが停止されたかどうかに関わらず値を戻すことを強制する場合は、SERVICE_IS_PARAMS 構造変数の nStopServiceWaitCount メンバーを適切な秒数に変更します。

構文

```
ServiceStopService ( szServiceName );
```

パラメーター

テーブル 134 ・ ServiceStopService のパラメーター

パラメーター	説明
szServiceName	サービスの名前を指定します。

戻り値

テーブル 135 ・ ServiceStopService の戻り値

戻り値	説明
>= **ISERR_SUCCESS**	関数がサービスを停止したことを示します。
< **ISERR_SUCCESS**	関数がサービスを停止できなかったことを示します。 この関数が失敗した場合、**GetExtendedErrInfo** を呼び出し、3 番目の引数の値を確認することでて追加エラー情報が利用できる場合もあります。(この値は一般的に、Windows API 関数への呼び出しに失敗した後の Windows API 関数 **GetLastError** の内部呼び出しの結果を示します。) 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

SetColor

SetColor 関数はセットアップ背景色を設定します。

メモ・この関数は、基本の MSI セットアッププロジェクトでのみ使用できます。

構文

```
Handler (nObject, Label);
```

パラメーター

テーブル 136 ・ SetColor のパラメーター

パラメーター	説明
nObject	変更するユーザーインターフェイスオブジェクトを指定します。このパラメーターに、あらかじめ定義されている次の定数を渡します。 ・ BACKGROUND Ñ セットアップ ウィンドウの背景を示します。デフォルトの色は青緑単色 (RGB (0,128,128)) です。

テーブル 136 ・ SetColor のパラメーター (続き)

パラメーター	説明
nColor	背景の色を指定します。 背景色をグラデーションする場合、次の定数の 1 つを渡します : ・ BK_BLUE ・ BK_GREEN ・ BK_MAGENTA ・ BK_ORANGE ・ BK_PINK ・ BK_RED ・ BK_YELLOW 単一背景色には、次の定数の 1 つを渡すか、RGB 関数を呼び出して特定の色 (括弧内にある値を利用する) を指示します : ・ BK_SOLIDBLACK (0, 0, 0) ・ BK_SOLIDBLUE (0, 0, 255) ・ BK_SOLIDGREEN (0, 255, 0) ・ BK_SOLIDMAGENTA (255, 0, 127) ・ BK_SOLIDORANGE (255, 127, 0) ・ BK_SOLIDPINK (255, 0, 255) ・ BK_SOLIDRED (255, 0, 0) ・ BK_SOLIDWHITE (255, 255, 255) ・ BK_SOLIDYELLOW (255, 255, 0) カスタムカラーの場合、このパラメーターに関数 RGB を渡します。  ***メモ ・*** *RGB 値を使用すると、Microsoft Windows のプログラミングマニュアルで説明しているメソッドと同じ手法を使うことができます RED、GREEN、および BLUE 色の混合を指定して、色を「混合」することができます。0 から 255 の番号を使って、使用する色の量を表します。リテラル数を RGB マクロのパラメーターに使用しなければなりません。* *RGB ステートメントのかわりに RGB カラーを表す長い値を使うこともできます。* *背景をカスタムカラーでペイントする場合の効果をスムーズにする (グラデーション) ためには、定義済みの定数 BK_SMOOTH と色をビットワイズで OR します。スムージング効果は 256 色を有効にすると向上します。*

戻り値

テーブル 137 ・ SetColor の戻り値

戻り値	説明
0	関数はオブジェクトの色を正しく設定したことを示します。
< 0	関数はユーザーインターフェイスオブジェクトの色を設定できなかったことを示します。

SetColor の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SetColor 関数のデモンストレーションを行います。
*
* SetColor への最初の呼び出しで背景色を単一青に設定
* します。2 回目呼び出しで背景色をグラデーション赤に設定
* します。最後の呼び出しで背景色を RGB 値に設定します。
*
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_SetColor(HWND);

function ExFn_SetColor(hMSI)
begin

  // 背景色を単一青に変更します。
  if (SetColor (BACKGROUND, BK_SOLIDBLUE) < 0) then
    MessageBox ("SetColor が失敗しました。", SEVERE);
  endif;

  // 3 秒間遅延させます。
  Delay (3);

  // 背景色をグラデーション赤に変更します。
  if (SetColor (BACKGROUND, BK_RED) < 0) then
    MessageBox ("SetColor が失敗しました。", SEVERE);
  endif;

  // 3 秒間遅延させます。
  Delay (3);

  // 背景色をカスタムマジェンタに変更します。
  if (SetColor (BACKGROUND, RGB(100, 50, 150))  0) then
```

```
    MessageBox ("SetColor が失敗しました。", SEVERE);
  endif;

  Delay (3);

end;
```

SetDialogTitle

SetDialogTitle 関数は、一部の共通ビルトイン ダイアログのタイトルバーに表示されるタイトルを変更します。パラメーター nDialogId を使って、ダイアログを指定します。SetDialogTitle を使用しない場合、デフォルトのタイトルが表示されます。

特定のダイアログにタイトルを設定すると、SetDialogTitle を使って再度タイトルを変更するまで、InstallShield はは設定したダイアログのタイプのすべてのインスタンスにこのタイトルを使います。タイトルを変更するダイアログの各タイプに SetDialogTitle を別個に呼び出す必要があります。

メモ・*InstallShield は、標準の Windows メッセージボックス関数を使ってメッセージボックスを作成します。Windows は、これらのメッセージボックスの [OK] ボタンと [キャンセル] ボタンのテキストを判別します。InstallShield は、Windows メッセージボックス内のボタンに使用しているテキストを制御しません。*

構文

```
SetDialogTitle (nDialogId, szTitle);
```

パラメーター

テーブル 138 ・ SetDialogTitle のパラメーター

パラメーター	説明
nDialogId	タイトルの変更を変更するビルトインのダイアログを識別します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ DLG_ASK_OPTIONS Ñ AskOptions ダイアログのタイトルを変更します。 ・ DLG_ASK_PATH–AskPath ダイアログのタイトルを変更します。 ・ DLG_ASK_OPTIONS Ñ AskText ダイアログのタイトルを変更します。 ・ DLG_ASK_YESNO Ñ AskYesNo ダイアログのタイトルを変更します。 ・ DLG_ENTER_DISK Ñ EnterDisk ダイアログのタイトルを変更します。 ・ DLG_MSG_INFORMATION– 情報スタイル MessageBox のタイトルを変更します。 ・ DLG_MSG_SEVERE Ñ Severe スタイルの MessageBox のタイトルを変更します。 ・ DLG_STATUS Ñ ダイアログ スタイルの進行状況インジケーターのタイトルを変更します。ダイアログスタイルの進行状況インジケーターを再度有効にする必要があり、それにはタイトルの変更を有効にするために DLG_STATUS オプションを使って SetDialogTitle を呼び出した後に Enable(STATUSDLG)) を呼び出します。 ・ DLG_MSG_WARNING Ñ Warning スタイルの MessageBox のタイトルを変更します。 ・ DLG_USER_CAPTION– ユーザー定義のメッセージ ボックス スタイルを使用する場合、MessageBox キャプションを変更します。
szTitle	新しいタイトルを指定します。

戻り値

テーブル 139 ・ SetDialogTitle の戻り値

戻り値	説明
0	関数がダイアログのタイトルを正常に変更したことを示します。
< 0	関数がダイアログのタイトルを変更できなかったことを示します。

SetDialogTitle の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
```

```
*
* SetDialogTitle 関数のデモンストレーションを行います。
*
* SetDialogTitle が呼び出され、AskYesNo ダイアログのタイトルを
* 変更します。
*
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "SetDialogTitle の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_SetDialogTitle(HWND);

function ExFn_SetDialogTitle(hMSI)
  NUMBER nCheck1, nCheck2;
begin

  // AskYesNo ダイアログのタイトルを設定します。
  if (SetDialogTitle (DLG_ASK_YESNO, TITLE_TEXT) < 0) then

    // エラーを報告します。
    MessageBox ("SetDialogTitle が失敗しました。", SEVERE);

  else

    // AskYesNo ダイアログを新しいタイトルと共に表示します。
    AskYesNo ("SetDialogTitle がこのタイトルを変更しましたか ?", YES);

  endif;

end;
```

SetDisplayEffect

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SetDisplayEffect 関数は、ビットマップまたはメタファイルを PlaceBitmap 関数を使って表示するとき、またはビルボードを表示するときに利用する表示効果を指定します。表示効果が設定された後に続いて PlaceBitmap が表示するすべてのビットマップ、またはビルボードは、SetDisplayEffect への呼び出しで別の効果が設定されるまで同じ効果を使って表示されます。

表示効果はビットマップまたはビルボードを配置する場合のみ有効です。ビットマップまたはビルボードを削除するときには効果は利用されません。

メモ・BITMAPICON、FULLSCREEN、FULLSCREENSIZE、または TILED オプションを使って PlaceBitmap が表示するビットマップには表示効果は利用されません。代わりに標準で表示されます。詳細については、「PlaceBitmap」を参照してください。

構文

SetDisplayEffect (nEffect);

パラメーター

テーブル 140 ・ SetDisplayEffect のパラメーター

パラメーター	説明
nEffect	表示効果を指定します。このパラメーターに、あらかじめ定義されている以下の定数のうちの 1 つを渡します。これらの定数は同時に使用することはできません。つまり、ビットワイズ OR 演算子を使用して、複数指定することはできません。さらに、このパラメーターは PlaceBitmap と共にビットマップを表示するときに BITMAPICON、FULLSCREEN、FULLSCREENSIZE、または TILED が指定された場合、効果を持ちません。 ・ EFF_FADE Ñ ビットマップまたはビルボードはゆっくりとフェードイン、フェードアウトします。 ・ EFF_REVEAL– ビットマップまたはビルボードは中央から徐々に四隅に向かって広がります。 ・ EFF_HORZREVEAL Ñ ビットマップまたはビルボードは中央から水平に広がります。 ・ EFF_HORZSTRIPE Ñ ビットマップまたはビルボードファイルのセクションは外側から内側に向けて水平に埋まります。次に残りが中央から外に向けて広がります。 ・ EFF_VERTSTRIPE– ビットマップまたはビルボードファイルのセクションは外側から内側に向けて水平に埋まります。次に残りが中央から外に向けて広がります。 ・ EFF_BOXSTRIPE Ñ ビットマップまたはビルボードのセクションはすべての端から内側に向かって埋まり、残りが内側から端に向かって広がります。 ・ EFF_NONE Ñ このオプションがデフォルトの設定です。その他のオプションの 1 つを呼び出した後、表示効果をクリアするのに利用します。 ***メモ・** EFF_REVEAL と EFF_HORZREVEAL のみメタファイルで利用できます。*

戻り値

テーブル 141 ・ SetDisplayEffect の戻り値

戻り値	説明
0	関数が表示効果を設定したことを示します。
< 0	関数が表示効果を設定できなかったことを示します。

SetDisplayEffect の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SetDisplayEffect 関数のデモンストレーションを行います。
*
```

```
＊このスクリプトは、違う効果をつけて同じビットマップを
＊7 回表示します。
＊
＊メモ : このスクリプトを適切に実行するため、
＊　定数 BITMAP_FILE がターゲットシステム上の既存の
＊　ビットマップファイルを参照するように設定しなくてはなりません。その後
＊　例では、BITMAP_ID には任意の整数が可能です。
＊　これはビットマップが DLL からロードされず、画面上に
＊　その他のビットマップがないためです。
＊
¥*---------------------------------------------------------------*/

#define BITMAP_FILE "C:¥¥Windows¥¥Forest.BMP"

// ビットマップに使う ID
#define BITMAP_ID 1

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_SetDisplayEffect(HWND);

function ExFn_SetDisplayEffect(hMSI)
begin

  // ビットマップが表示される度に効果名が
  // タイトルバーに表示されるよう、背景ウィンドウを
  // 開きます。
  Enable (FULLWINDOWMODE);
  Enable (BACKGROUND);

  // 1. ボックス ストライプ効果を使ってビットマップを表示します。
  SetDisplayEffect (EFF_FADE);

  SetTitle (" フェード効果 ", 0, BACKGROUNDCAPTION);
  PlaceBitmap (BITMAP_FILE, BITMAP_ID, 0, 0, CENTERED);
  Delay (3);
  PlaceBitmap ("", BITMAP_ID, 0, 0, REMOVE);

  // 2. リビール効果を使ってビットマップを表示します。
  SetDisplayEffect (EFF_REVEAL);

  SetTitle(" リビール効果 ", 0, BACKGROUNDCAPTION);
  PlaceBitmap (BITMAP_FILE, BITMAP_ID, 0, 0, CENTERED);
  Delay (3);
  PlaceBitmap ("", BITMAP_ID, 0, 0, REMOVE);

  // 3. 水平リビール効果を使ってビットマップを表示します。
  SetDisplayEffect (EFF_HORZREVEAL);

  ビットマップまたはビルボードは中央から徐々に四隅に向かって広がります。
  PlaceBitmap (BITMAP_FILE, BITMAP_ID, 0, 0, CENTERED);
  Delay (3);
  PlaceBitmap ("", BITMAP_ID, 0, 0, REMOVE);

  // 4. 水平ストライプ効果を使ってビットマップを表示します。
  SetDisplayEffect (EFF_HORZSTRIPE);

  SetTitle (" 水平ストライプ効果 ", 0, BACKGROUNDCAPTION);
```

```
    PlaceBitmap (BITMAP_FILE, BITMAP_ID, 0, 0, CENTERED);
    Delay (3);
    PlaceBitmap ("", BITMAP_ID, 0, 0, REMOVE);

    // 5. 垂直ストライプ効果を使ってビットマップを表示します。
    SetDisplayEffect (EFF_VERTSTRIPE);

    SetTitle (" 垂直ストライプ効果 ", 0, BACKGROUNDCAPTION);
    PlaceBitmap (BITMAP_FILE, BITMAP_ID, 0, 0, CENTERED);
    Delay (3);
    PlaceBitmap ("", BITMAP_ID, 0, 0, REMOVE);

    // 6. ボックス ストライプ効果を使ってビットマップを表示します。
    SetDisplayEffect (EFF_BOXSTRIPE);

    SetTitle (" ボックスストライプ効果 ", 0, BACKGROUNDCAPTION);
    PlaceBitmap (BITMAP_FILE, BITMAP_ID, 0, 0, CENTERED);
    Delay (3);
    PlaceBitmap ("", BITMAP_ID, 0, 0, REMOVE);
    Delay (1);

    // 7. 効果すべてをクリア
    SetDisplayEffect (EFF_NONE);

    SetTitle (" 効果なし ", 0, BACKGROUNDCAPTION);
    PlaceBitmap (BITMAP_FILE, BITMAP_ID, 0, 0, CENTERED);
    Delay (3);
    PlaceBitmap ("", BITMAP_ID, 0, 0, REMOVE);

    Delay (1);

end;
```

SetErrorMsg

ディスク エラーが発生したとき、**SetErrorMsg** 関数は **EnterDiskError** 関数によって表示される対応エラー メッセージを設定します。

構文

```
SetErrorMsg ( nErrorID, szText );
```

パラメーター

テーブル 142 · SetErrorMsg のパラメーター

パラメーター	説明
nErrorID	カスタマイズするエラー メッセージを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · ERR_BOX_BADPATH Ñ このメッセージは、**EnterDiskError** がユーザーが入力した無効なパスを検出した場合に表示されます。 · ERR_BOX_BADTAGFILE Ñ このメッセージは、指定したタグ ファイルがディスク上に存在しないことを **EnterDiskError** が検出したときに表示されます。 · ERR_BOX_DISKID Ñ このメッセージは、ユーザーが指定したドライブが存在しないことを **EnterDiskError** が検出したときに表示されます。
szText	メッセージボックスに表示するエラーメッセージを指定します。

戻り値

テーブル 143 · SetErrorMsg の戻り値

戻り値	説明
0	関数がエラーメッセージを変更したことを示します。
< 0	関数がエラーメッセージを変更できなかったことを示します。

SetErrorMsg の例

***メモ ·** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SetErrorMsg 関数のデモンストレーションを行います。
*
* このスクリプトは一連のセットアップディスクの次のディスクが
* 指定されたドライブに準備できていない場合に、EnterDisk への呼び出しの後に
* 表示されるエラーメッセージをカスタマイズします。
*
¥*---------------------------------------------------------------------*/

// EnterDisk エラー用のテキストメッセージを定義します。
#define MSG_DRIVE_DOOR "ドライブの扉が開いています。"
#define MSG_BAD_PATH   "パスが存在しません。"
#define MSG_BAD_TAG    "タグファイルが無効です。"
```

```
#define MSG_BAD_DRIVE  " 指定したドライブが存在しません。"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_SetErrorMsg(HWND);

function ExFn_SetErrorMsg(hMSI)
begin

  // EnterDisk 関数のエラーボックス用メッセージを設定します。
  SetErrorMsg (ERR_BOX_DRIVEOPEN, MSG_DRIVE_DOOR);
  SetErrorMsg (ERR_BOX_BADPATH, MSG_BAD_PATH  );
  SetErrorMsg (ERR_BOX_BADTAGFILE, MSG_BAD_TAG);
  SetErrorMsg (ERR_BOX_DISKID, MSG_BAD_DRIVE);

  // ユーザーにディスクの指定を求めます。
  EnterDisk ("' 'Examples' ディスクを挿入して下さい。:", "Example.exe");

end;
```

SetErrorTitle

ディスク エラーが発生したとき、**SetErrorTitle** 関数は **EnterDiskError** 関数によって表示されるエラー メッセージのタイトルバーを設定します。

構文

```
SetErrorTitle ( nErrorID, szText );
```

パラメーター

テーブル 144 · SetErrorTitle のパラメーター

パラメーター	説明
nErrorID	タイトルをカスタマイズするエラー メッセージ ボックスを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · ERR_BOX_BADPATH Ñ このメッセージは、**EnterDiskError** が無効なパスを検出した場合に表示されます。 · ERR_BOX_BADTAGFILE Ñ このメッセージは、指定したタグ ファイルがディスク上に存在しないことを **EnterDiskError** が検出したときに表示されます。 · ERR_BOX_DISKID Ñ このメッセージボックスは、指定したドライブが存在しないことを **EnterDiskError** が検出したときに表示されます。
szText	エラーメッセージボックスに表示するタイトルを指定します。

戻り値

テーブル 145 · SetErrorTitle の戻り値

戻り値	説明
0	関数がタイトルバーのテキストを変更したことを示します。
< 0	関数がタイトルバーのテキストを変更できなかったことを示します。

SetErrorTitle の例

***メモ·** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SetErrorTitle 関数のデモンストレーションを行います。
*
* このスクリプトは、一連のセットアップディスクの次のディスクが
* このスクリプトは、一連のセットアップディスクの次のディスクが
* 表示されるメッセージボックス用のタイトルテキストをカスタマイズします。
*
¥*---------------------------------------------------------------------*/

// EnterDisk エラー用のメッセージボックスタイトルテキストを定義します。
#define MSG_DRIVE_DOOR ″ドライブの扉が開いています。″
#define MSG_BAD_PATH   ″パスが見つかりません。″
#define MSG_BAD_TAG    ″タグファイルが無効です。″
```

```
#define MSG_BAD_DRIVE "ドライブが見つかりません。"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_SetErrorTitle(HWND);

function ExFn_SetErrorTitle(hMSI)
  STRING szText;
   NUMBER nErrorID;
begin

  // EnterDisk への呼び出しの後に発生する各エラー用の
  // メッセージボックスタイトルを設定します。
  SetErrorTitle (ERR_BOX_DRIVEOPEN,  MSG_DRIVE_DOOR);
  SetErrorTitle (ERR_BOX_DISKID,     MSG_BAD_DRIVE);
  SetErrorTitle (ERR_BOX_BADTAGFILE, MSG_BAD_TAG);
  SetErrorTitle (ERR_BOX_BADPATH,    MSG_BAD_PATH);

  // ドライブ A をデフォルトとします。

  // ユーザーにディスクの指定を求めます。
  EnterDisk ("'Examples' ディスクを挿入して下さい。:", "Example.exe");

end;
```

SetExtendedErrInfo

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SetExtendedErrInfo 関数は、**GetExtendedErrInfo** が読み出すことのできるエラー情報を設定します。

構文

```
SetExtendedErrInfo ( szScriptFile, nLineNumber, nError );
```

パラメーター

テーブル 146 · SetExtendedErrInfo のパラメーター

パラメーター	説明
szScriptFile	エラーが発生したスクリプトファイルを指定します。現在のスクリプト ファイル (つまり SetExtendedErrInfo への呼び出しを含むスクリプト ファイル) を指定するには、予約識別子 _FILE_ を渡します。
nLineNumber	エラーが発生した行番号を指定します。現在の行番号 (つまり SetExtendedErrInfo への呼び出しの行番号) を指定するには、予約識別子 _LINE_ を渡します。
nError	エラーのエラーコードを指定します。

戻り値

テーブル 147 · SetExtendedErrInfo の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がエラー情報を設定しました。
< ISERR_SUCCESS	関数がエラー情報の設定に失敗しました。

SetFileInfo

SetFileInfo 関数によって、既存のファイルの日付またはタイムスタンプが設定されるか、ファイルの属性が変更されます。ファイルの日付と時間を両方変更する場合、日付を変更するためと時間を変更するために SetFileInfo を 2 度呼び出す必要があります。ただし、nAttribute の定数を OR (|) 演算子と組み合わせることによって、SetFileInfo を 1 度呼び出すだけで複数のファイル属性を設定できます。

***メモ ·** この関数を使ってフォルダーの属性を変更することもできます。例えば、この関数を使って非表示フォルダーを作成することができます。*

構文

SetFileInfo (szPathFile, nType, nAttribute, szValue);

パラメーター

テーブル 148 · SetFileInfo のパラメーター

パラメーター	説明
szPathFile	日付、時刻、または属性を変えるフォルダーの完全修飾名を指定します。
nType	変更するファイルの特徴を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · FILE_ATTRIBUTE Ñ ファイルの 1 つ以上ファイルの属性が変更されることを示します。 · FILE_DATE Ñ ファイルの日付スタンプが変更されることを示します。 · FILE_TIME Ñ ファイルのタイムスタンプが変更されることを示します。
nAttribute	nType が FILE_ATTRIBUTE のときのファイル属性を指定します。(nType が FILE_DATE または FILE_TIME のとき、このパラメーターに 0 を渡します。)2 つ以上の属性を指定する場合は、次の定義済みの定数と OR (|) 演算子を組み合わせます。 · FILE_ATTR_ARCHIVED Ñ アーカイブ属性を設定します。 · FILE_ATTR_HIDDEN Ñ 非表示属性を設定します。 · FILE_ATTR_READONLY Ñ 読み取り専用属性を設定します。 · FILE_ATTR_SYSTEM Ñ システム属性を設定します。 · FILE_ATTR_NORMAL Ñ この定数だけを指定した場合、すべてのファイル属性がクリアされます。この定数と他のファイル属性定数が OR で組み合わされている場合、OR 式に含まれていない属性はクリアされます。
szValue	nType で渡される値に基づいて、次のどれかを指定します : · FILE_DATE Ñ 日付を YYYY/MM/DD または YYYY¥MM¥DD の形式で指定します。日付は「1980/01/01」以降を使用してください。区切り文字として円記号を使用すると、円記号はエスケープ シーケンスとして文字列に挿入されます (たとえば、1980¥¥01¥¥01)。 · FILE_TIME Ñ 24 時間形式で HH:MM:SS で時間を指定します。秒は 2 の倍数です。 · FILE_ATTRIBUTE Ñ このパラメーターにヌル文字列 ("") を渡します。

戻り値

テーブル 149 · SetFileInfo の戻り値

戻り値	説明
0	ファイルの日付スタンプ、タイムスタンプ、または属性が正常に設定されたことを示します。

テーブル 149 ・ SetFileInfo の戻り値 (続き)

戻り値	説明
< 0	ファイルの日付スタンプ、タイムスタンプ、または属性が設定できなかったことを示します。

SetFileInfo の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SetFileInfo 関数のデモンストレーションを行います。
*
* SetFileInfo が呼び出され、ファイルの日付、時刻、属性を
* 設定します。
*
* メモ : このスクリプトを実行する前に、C ドライブのルートへ
*    ISExampl.txt と名づけられたファイルを作成します。
*
¥*---------------------------------------------------------------------*/

#define EXAMPLE_FILE "C:¥¥ISExampl.txt"

#define TITLE_TEXT   "SetFileInfo の例 "

#define NEW_FILE_DATE "2003/09/12"
#define NEW_FILE_TIME "18:30:00"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_SetFileInfo(HWND);

function ExFn_SetFileInfo(hMSI)
   LIST   listID;
begin

   // メッセージを収めるリストを作成します。
   listID = ListCreate (STRINGLIST);

   // エラーが発生した場合にそれを報告し、終了します。
   if (listID = LIST_NULL) then
      MessageBox (" この例に必要なリストを作成できませんでした。", SEVERE);
      abort;
   endif;

   // ファイルの日付を設定します。
   if (SetFileInfo (EXAMPLE_FILE, FILE_DATE, 0, NEW_FILE_DATE) < 0) then
```

```
    ListAddString (listID, " ファイル ¥ の日付を変更できませんでした。", AFTER);

  else
    ListAddString (listID, " ファイル ¥ の日付を " + NEW_FILE_DATE + " に変更しました。",
AFTER);

  endif;

 // ファイルの時刻を設定します。
  if (SetFileInfo (EXAMPLE_FILE, FILE_TIME, 0, NEW_FILE_TIME) < 0) then
    ListAddString (listID, " ファイル ¥ の時刻を変更できませんでした。", AFTER);

  else
    ListAddString (listID, " ファイル ¥ の時刻を " + NEW_FILE_TIME + " へ変更しました。",
AFTER);

  endif;

  // ファイルの属性をクリアします。
  if (SetFileInfo (EXAMPLE_FILE, FILE_ATTRIBUTE, FILE_ATTR_NORMAL, "") < 0)
then
    ListAddString (listID, " ファイルの属性をクリアすることができませんでした。", AFTER);

  else
    ListAddString (listID, " ファイル属性をクリアしました。", AFTER);

  endif;

  // 結果をレポートします。
  SdShowInfoList (TITLE_TEXT, " 変更 " + EXAMPLE_FILE, listID);

  // メモリからリストを削除します。
  ListDestroy(listID);

end;
```

SetFont

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SetFont 関数は、テキスト文字列を表示する時のフォントとスタイルを設定します。この関数では Windows 標準フォントを利用することもできます。

構文

```
SetFont ( nItemID, nFontStyle, szFontName );
```

パラメーター

テーブル 150 ・ SetFont のパラメーター

パラメーター	説明
nItemID	フォントとスタイルを設定するアイテムを指定します。このパラメーターに、あらかじめ定義されている次の定数を渡します。 ・ FONT_TITLE– セットアップウィンドウの左上角に表示するセットアップ処理のメインタイトルを指定します。
nFontStyle	フォントスタイルを指定します。このパラメーターには、次の定義済み定数のいずれかを指定します。STYLE_NORMAL 以外はビット単位で OR して複数のスタイルを指定することができます。 ・ STYLE_NORMAL Ñ 太字、イタリック、または影付きは不可 (OR は使用できません) ・ STYLE_BOLD Ñ 太字テキスト ・ STYLE_ITALIC Ñ イタリック テキスト ・ STYLE_SHADOW Ñ 影付きテキスト ・ STYLE_UNDERLINE Ñ 下線付きテキスト
szFontName	有効な Windows フォントの名前を指定します。有効なフォント名には Courier、Helv、Helvetica、Modern、Roman、Script、Terminal、Times、そして TmsRmn が含まれます。指定されたスタイルで指定したフォントが見つからなかった場合、Arial が利用されます。

戻り値

テーブル 151 ・ SetFont の戻り値

戻り値	説明
0	関数がフォントを設定したことを示します。
< 0	関数がフォントを設定できなかったことを示します。

SetFont の例

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SetFont 関数のデモンストレーションを行います。
*
* このスクリプトでは、セットアップのメインウィンドウに 3 つの異なるタイトルが
* 表示されます。各タイトルは画面上に 3 秒間
* 表示されます。タイトルは、SetTitle への呼び出しで
* 表示され、ここでタイプサイズと色も設定します。
```

```
＊タイトルのフォント色とフォントスタイルを制御する為に、
＊スクリプトは SetTitle への各呼び出しの前に SetFont を呼び出します。
＊
＊        フォント        サイズ   スタイル        色
＊
＊タイトル 1   Times New Roman    36    標準          赤
＊タイトル 2   Courier New       48    イタリック    黄
＊タイトル 3   Arial            60    太字、影つき  青
＊
¥*---------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_SetFont(HWND);

function ExFn_SetFont(hMSI)
begin

   Enable ( BACKGROUND );

   // タイトル 1: Times Roman、36pt, 標準、赤。
   if (SetFont (FONT_TITLE, STYLE_NORMAL, "Times New Roman") < 0) then
      MessageBox ("SetFont が失敗しました。", SEVERE);
   endif;

   SetTitle ("SetFont の例 1", 36, RGB(255, 0, 0));
   Delay (3);

   // タイトル 2: Courier New、48pt、イタリック、黄。
   if (SetFont (FONT_TITLE, STYLE_ITALIC, "Courier New") < 0) then
      MessageBox ("SetFont が失敗しました。", SEVERE);
   endif;

   SetTitle ("SetFont の例 2", 48, RGB(255, 255, 0));
   Delay (3);

   // タイトル 3: Arial、60pt、太字、影付き、青。
   if (SetFont (FONT_TITLE, STYLE_BOLD | STYLE_SHADOW, "Arial") < 0) then
      MessageBox ("SetFont が失敗しました。", SEVERE);
   endif;

   SetTitle ("SetFont の例 3", 60, RGB(0, 0, 255));
   Delay (3);

end;
```

SetInstallationInfo

SetInstallationInfo 関数はシステム変数 IFX_COMPANY_NAME、IFX_PRODUCT_NAME、IFX_PRODUCT_VERSION、および IFX_PRODUCT_KEY の値を設定します。CreateInstallationInfo が利用できるように SetInstallationInfo はこの情報を指定します。CreateInstallationInfo はイベント指向のスクリプトではデフォルト OnShowUI イベント ハンドラーコードで呼び出され、インストールするプログラムのアプリケーション情報キーとアプリケーションごとのパスキーを作成します。

SetInstallationInfo はレジストリ関連の特殊関数で、特定の定義済みレジストリキーと一緒に動作するように設計されています。レジストリ関連の特殊関数については、「レジストリ関連の特殊関数」を参照してください。

構文

SetInstallationInfo (szCompany, szProduct, szVersion, szProductKey);

パラメーター

テーブル 152 ・ SetInstallationInfo のパラメーター

パラメーター	説明
szCompany	会社名を指定します。CreateInstallationInfo は szCompany を使って 、レジストリの HKEY_LOCAL_MACHINE¥Software キーの下に ¥< 会社 > キーを作成します。
szProduct	インストールする製品の名前を指定します。CreateInstallationInfo は szProduct を使って 、レジストリの HKEY_LOCAL_MACHINE¥Software¥ 会社 キーの下に ¥< 製品 > キーを作成します。szProduct の中の値は、[ようこそ] ダイアログのメッセージテキストの最初のパラグラフにも挿入されます。
szVersion	製品のバージョン番号を入力します。CreateInstallationInfo は szVersion を使って 、レジストリの HKEY_LOCAL_MACHINE¥Software¥< 会社 >¥< 製品 > キーの下に ¥< バージョン > キーを作成します。合わせて、¥< 会社 > キー (szCompany)、¥< 製品 > キー (szProduct)、および ¥< バージョン > キー (szVersion) がアプリケーションの情報キーとして参照されます。アプリケーション情報キーは CreateInstallationInfo を呼び出すとすぐに作成されます。
szProductKey	アプリケーションのメイン実行ファイルの名前を指定します。製品が複数の実行可能ファイルを使用する場合、製品を最もよく表す実行ファイルを指定します。CreateInstallationInfo は szProductKey を使って、アプリケーションごとのパスキーを、キー HKEY_LOCAL_MACHINE¥Software¥Microsoft¥Windows¥CurrentVersion¥App Paths の下に作成します。アプリケーションごとのパスキーは、RegDBSetItem を呼び出して値名と値のペアをそのキーの下に作成するまで、レジストリには作成されません。

戻り値

この関数は常に 0 を返します。

SetObjectPermissions

SetObjectPermissions 関数は、ファイル、フォルダー、またはレジストリ キーのアクセス許可を設定します。ファイル、フォルダー、またはレジストリ キーは、インストールの一部としてインストールでき、またターゲット システムに既に存在する場合もあります。

構文

```
SetObjectPermissions (byval string szObject, byval number nType, byval string szDomain, byval string szUser, byval number nPermissions,
    byval number nOptions);
```

パラメーター

テーブル 153 · SetObjectPermissions のパラメーター

パラメーター	説明
szObject	アクセス許可を設定するオブジェクト (ファイル、フォルダー、またはレジストリ キー) を指定します。 ファイルとフォルダーには、完全パスを指定します。 レジストリ キーには、パスに以下のいずれかを使用します : · CLASSES_ROOT Ñ HKEY_CLASSES_ROOT ハイブを示します。 · CURRENT_USER Ñ HKEY_CURRENT_USER ハイブを示します。 · MACHINE Ñ HKEY_LOCAL_MACHINE ハイブを示します。 · USERS Ñ HKEY_USERS ハイブを示します。 次の例は HKEY_LOCAL_MACHINE 内のキーのアクセス許可を設定します : SetObjectPermissions("MACHINE¥¥Software¥¥MyProduct¥¥Example", IS_PERMISSIONS_TYPE_REGISTRY, "", "Users", KEY_CREATE_SUB_KEY, IS_PERMISSIONS_OPTION_DENY_ACCESS);
nType	szObject パラメーターを使って渡されるオブジェクトの種類を示します。有効なオプションは次のとおりです : · IS_PERMISSIONS_TYPE_FILE Ñ szObject はファイルです。 · IS_PERMISSIONS_TYPE_FOLDER Ñ szObject はフォルダーです。 · IS_PERMISSIONS_TYPE_REGISTRY Ñ szObject はレジストリ キーです。
szDomain	アクセス許可を設定するユーザーのドメイン名を指定します。 ローカル マシンをドメインとして使用する場合、このパラメーターに空白文字列 ("") を渡します。

テーブル 153 · SetObjectPermissions のパラメーター (続き)

パラメーター	説明
szUser	アクセス許可を設定するユーザーのドメイン名を指定します。選択可能なオプションは以下のとおりです : • Administrators • Authenticated Users • Creator Owner • Everyone • Guests • Interactive • IUSR • Local Service • Local System • Network Service • Power Users • Remote Desktop Users • Users ユーザーはインストール中に作成されるか、実行時にターゲット システム上に既存するかを問いません。

テーブル 153 • SetObjectPermissions のパラメーター (続き)

パラメーター	説明
nPermissions	指定のユーザー向けにオブジェクトに適用するアクセス許可を指定するには、このパラメーターに次の定義済み定数の 1 つを渡します。ビット単位 OR 演算子 (|) を使用して、これらの定数を組み合わせることができます。 選択可能なオプションは以下のとおりです : • DELETE • GENERIC_ALL • GENERIC_EXECUTE • GENERIC_WRITE • GENERIC_READ • READ_CONTROL • STANDARD_RIGHTS_ALL • STANDARD_RIGHTS_EXECUTE • STANDARD_RIGHTS_READ • STANDARD_RIGHTS_REQUIRED • STANDARD_RIGHTS_WRITE • SYNCHRONIZE • WRITE_DAC • WRITE_OWNER ファイルとフォルダーには、以下のオプションが使用できます : • FILE_LIST_DIRECTORY (フォルダー) • FILE_READ_DATA (ファイル) • FILE_WRITE_DATA (ファイル) • FILE_ADD_FILE (フォルダー) • FILE_APPEND_DATA (ファイル) • FILE_ADD_SUBDIRECTORY (フォルダー) • FILE_READ_EA (ファイルとフォルダー) • FILE_WRITE_EA (ファイルとフォルダー) • FILE_EXECUTE (ファイル) • FILE_TRAVERSE (フォルダー) • FILE_DELETE_CHILD (フォルダー) • FILE_READ_ATTRIBUTES (ファイルとフォルダー) • FILE_WRITE_ATTRIBUTES (ファイルとフォルダー) • FILE_ALL_ACCESS

テーブル 153 ・ SetObjectPermissions のパラメーター (続き)

パラメーター	説明
nPermissions (続き)	レジストリ キーには次のオプションが使用できます : ・ KEY_QUERY_VALUE ・ KEY_SET_VALUE ・ KEY_CREATE_SUB_KEY ・ KEY_ENUMERATE_SUB_KEYS ・ KEY_NOTIFY ・ KEY_CREATE_LINK 各値についての詳細は、MSDN ライブラリ の「Registry Key Security and Access Rights」、「File Security and Access Rights」、および「Registry Key Security and Access Rights」 を参照してください。
nOptions	このパラメーターには、次の定義済み定数のいずれかを指定します。 ・ IS_PERMISSIONS_OPTION_64BIT_OBJECT – REGDB_OPTION_WOW64_64KEY オプションが有効になっているなっていないにかかわらず、nPermissions で指定されたアクセス許可セットは、64 ビット キーに対して指定する必要があります。この定数を 32 ビット ターゲット システムで渡すことはできません。また、この定数によって、ファイルまたはフォルダーのアクセス許可が影響を受けることはありません。 ・ IS_PERMISSIONS_OPTION_DENY_ACCESS Ñ nPermissions で指定されたアクセス許可セットは拒否されます。 ・ IS_PERMISSIONS_OPTION_NO_APPLYDOWN Ñ アクセス許可は、指定されたオブジェクトにのみ適用され、子オブジェクトには一切反映されません。 ・ IS_PERMISSIONS_OPTION_DENY_ACCESS Ñ nPermissions で指定されたアクセス許可セットが許可されます。 ビット単位 OR 演算子 (|) を使用して、複数の定数を組み合わせることができます。IS_PERMISSIONS_OPTION_DENY_ACCESS および IS_PERMISSIONS_OPTION_ALLOW_ACCESS 定数は、相互に排他的であるため組み合わせることはできません。nOptions にこれらのオプションの両方を指定した場合、アクセス許可が拒否されます。

戻り値

テーブル 154 ・ SetObjectPermissions の戻り値

戻り値	説明
ISERR_SUCCESS	関数がアクセス許可を正しく設定しました。
!= ISERR_SUCCESS	関数がアクセス許可を設定できませんでした。戻り値は Win32 エラーです。Win32 エラーについての詳細は、「MSDN ライブラリ」を参照してください。

追加情報

SetObjectPermissions は、REGDB_OPTION_WOW64_64KEY オプションが有効になっていると、64 ビット キーのアクセス許可を設定しようと試みます。

64 ビット System32 フォルダーにあるファイルのアクセス許可を設定するには、**SetObjectPermissions** 関数が呼び出される前に、ファイル システムのリダイレクトが無効になっている必要があります。ファイル システムのリダイレクトを無効にするには、次のコードで示されているように、WOW64FSREDIRECTION 定数を使用します :

```
Disable(WOW64FSREDIRECTION );
```

いったん操作が終了したら、ファイル システムのリダイレクトを無効にします :

```
Enable(WOW64FSREDIRECTION);
```

SetObjectPermissions の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
//---------------------------------------------------------------------------
//
// InstallScript スクリプト例
//
// SetObjectPermissions 関数のデモンストレーションを行います。
//
// このサンプルは、ローカル管理者がファイルまたはその属性を
// 変更できないように防ぐ方法を説明します。
//
//---------------------------------------------------------------------------

function OnFirstUIAfter( )
    STRING szTitle, szMsg1, szMsg2, szOpt1, szOpt2;
    NUMBER bvOpt1, bvOpt2;
begin

    SetObjectPermissions (TARGETDIR+"MyFile.exe",IS_PERMISSIONS_TYPE_FILE, "", "administrator",
                          FILE_WRITE_DATA|FILE_APPEND_DATA|FILE_WRITE_EA|FILE_WRITE_ATTRIBUTES,
                          IS_PERMISSIONS_OPTION_DENY_ACCESS);

end;
```

SetShortcutProperty

SetShortcutProperty 関数では、インストール実行時に Windows シェルによる設定が必要な 1 つ以上のショートカット プロパティを指定できます。たとえば、**SetShortcutProperty** には次の動作を制御するシェル プロパティを設定するためのビルトイン サポートが搭載されています :

- Windows 8 [スタート] 画面にショートカットをピン留めするかどうかを指定する
- Windows 7 以降のシステム上でエンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めできるかどうかを指定する
- Windows 7 以降のシステム上では [スタート] メニューのショートカットが新しくインストールされたアイテムとして強調表示されないように防ぎます。

SetShortcutProperty はまた、Windows シェルがサポートする追加プロパティを設定することもできます。

***メモ・SetShortcutProperty** を呼び出すためには、そのショートカットとそのターゲットが既にターゲット システム上に存在している必要があります。*

***SetShortcutProperty** はインターネット ショートカットの構成をサポートしません。*

構文

```
SetShortcutProperty (szShortcutFolder, szName, szPropertyKey, szValue);
```

パラメーター

テーブル 155 ・ SetShortcutProperty のパラメーター

パラメーター	説明
szShortcutFolder	プロパティを構成するショートカットのパスを指定します。 特定のフォルダー内のショートカットを構成する場合、以下のような完全修飾パスを指定します : **C:¥¥ProgramData¥¥Microsoft¥¥Windows¥¥Start Menu¥¥Programs** [スタート] メニューにある [プログラム] メニューにショートカットがある場合、このパラメーターにヌル文字列 (””) を渡します。 次の InstallScript システム変数の 1 つをこのパラメーターで渡すことができます : ・ FOLDER_DESKTOP – デスクトップにショートカットを追加します。 ・ FOLDER_STARTUP – スタートアップ メニューにショートカットを追加します。 ・ FOLDER_STARTMENU – スタート メニューにショートカットを追加します。 ・ FOLDER_PROGRAMS – スタート ¥ プログラム メニューにショートカットを追加します。 InstallScript システム変数によって識別されるフォルダーの相対パスを指定することもできます。例 : **FOLDER_PROGRAMS ^ ″ACCESSORIES¥¥GAMES″**
szName	構成するショートカットの名前を指定します。

テーブル 155 ・ SetShortcutProperty のパラメーター (続き)

パラメーター	説明
szPropertyKey	設定するプロパティ キー名を指定します。設定できるプロパティは、Windows SDK に含まれている **propkey.h** で定義されています。 InstallScript で定義済みのプロパティ キーには、次から任意の定数を渡すことができます : ・ SSP_PROPERTY_PREVENT_PINNING – Windows 7 以降のシステムで、[スタート] メニューまたはタスクバーにショートカットをピン留めすることを防ぎます。このオプションは、エンド ユーザーがタスクバーおよび [スタート] メニューにショートカットをピン留めするためのコンテキスト メニュー コマンドを隠します。 インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを防ぎたい場合があります。 このプロパティを有効にするには、szValue を 1 に設定します。 ・ SSP_PROPERTY_NO_NEW_INSTALL_HIGHLIGHT – エンド ユーザーが製品を Windows 7 以降のシステム上にインストールした後、ショートカットを新しくインストールされたプログラムとして強調表示しません。これは、ターゲット システム上で個別のアイテムに対して [[スタート] メニューのカスタマイズ] ダイアログ ボックスで [新しくインストールされたプログラムを強調表示する] チェック ボックスをクリアした場合と同じ効果を持ちます。 インストールの一部であるツールまたは従属的な製品のショートカットのオプションを使用したい場合があります。 このプロパティを有効にするには、szValue を 1 に設定します。 ・ SSP_PROPERTY_NO_STARTSCREEN_PIN – Windows 8 ターゲット システム上で、デフォルトで [スタ ― ト] 画面にショートカットをピン留めしません。この定数を渡すと、インストールは Windows 8 で使用可能になった Windows シェル プロパティを設定します。 インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを防ぎたい場合があります。 このプロパティを有効にするには、szValue を 1 に設定します。 SSP_PROPERTY_PREVENT_PINNING および SSP_PROPERTY_NO_STARTSCREEN_PIN についての詳細は、追加情報セクションを参照してください。
szValue	設定するプロパティの値 (文字列として表示) を指定します。指定されたプロパティで使用する値についての情報は、Windows SDK に含まれる **propkey.h** を参照してください。 InstallScript エンジンは、文字列値を szPropertyKey で指定されているプロパティに対応するデータ型に変換します。

戻り値

テーブル 156:

戻り値	説明
>= ISERR_SUCCESS	関数がショートカット プロパティを正しく設定したことを示します。
< ISERR_SUCCESS	関数がショートカット プロパティを設定できなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

Additional Information

2 つの nFlag 定数について、以下にご注意ください。

SSP_PROPERTY_PREVENT_PINNING

ピン留めを行わないようにショートカットを構成した場合、[スタート] メニューの最もよく使われている製品のリストに、ショートカットのターゲットを含められなくなります。

特定の文字列を含むショートカットは、タスクバーまたは [スタート] メニューにピン留めすることができません。また、それらを最もよく使う製品リストに表示することもできません。例 :

- Documentation
- ヘルプ
- Install
- 削除
- Setup
- Support

SSP_PROPERTY_NO_STARTSCREEN_PIN

Windows 8 は、アプリケーションのアンインストールによってショートカットが削除された後でも、ショートカットの [スタート] 画面へのピン留めに関する情報を保持します。そのため、ショートカットがインストール済みの場合、ターゲット システム上で SSP_PROPERTY_NO_STARTSCREEN_PIN 定数は効果を持ちません。この機能をテストする際、ショートカットとそのターゲットが既にインストールされていない、クリーン マシン上でテストするようにして下さい。

SetShortcutProperty の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SetShortcutProperty 関数のデモンストレーションを行います。
*
* この例は、ショートカットのシェル プロパティを設定します。その後
* プロパティは、エンド ユーザーがショートカットをタスクバー
* および [ スタート ] メニューにピン留めできるようにするコンテキスト
* メニュー コマンドを隠します
*
* メモ : このスクリプトを適切に実行するため、
* プロジェクトに実行可能ファイルを追加して、[ ショートカット ] ビューに
* My Shortcut を作成します。ショートカットのターゲットは、
プロジェクトに追加済みの実行可能ファイルでなくては
* なりません。ショートカットの場所は「デスクトップ」です。
*
*
¥*---------------------------------------------------------------*/

function OnFirstUIAfter( )
    STRING szName, szValue;
begin

    // SetShortcutProperty の呼び出し用のパラメーターを設定します。
    szName  = "My Shortcut";
    szValue = "1";

    // タスクバーおよび [ スタート ] メニューへのピン留めを防ぐショートカット プロパティを設定します。
    if (SetShortcutProperty (FOLDER_DESKTOP, szName, SSP_PROPERTY_PREVENT_PINNING,
                             szValue) < 0) then
        MessageBox ("SetShortcutProperty が失敗しました ", SEVERE);
    else
        SprintfBox (INFORMATION, "SetShortcutProperty", "%s が正しく構成されました。",
                    szName);
    endif;

end;
```

SetStatus

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SetStatus 関数はオブジェクトスクリプトで呼び出され、オブジェクトの Status.Number や Status.Description プロパティを設定します。オブジェクトのその他のステータスプロパティを設定するには、SetStatusEx を呼び出します。

構文

```
SetStatus ( nNumber, szDescription );
```

パラメーター

テーブル 157 ・ SetStatus のパラメーター

パラメーター	説明
nNumber	Status.Number の値を指定します。
szDescription	Status.Description の値を指定します。

戻り値

テーブル 158 ・ SetStatus の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がステータスプロパティを設定しました。
< ISERR_SUCCESS	関数がステータスプロパティの設定に失敗しました。

SetStatusEx

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SetStatusEx は、オブジェクトのステータスプロパティを設定するためにオブジェクトスクリプトで呼び出されます。

構文

```
SetStatusEx ( nNumber, szDescription, szSource, szScriptFile, nScriptLine, nScriptError );
```

パラメーター

テーブル 159 · SetStatusEx のパラメーター

パラメーター	説明
nNumber	Status.Number の値を指定します。
szDescription	Status.Description の値を指定します。
szSource	Status.szSource の値を指定します。
szScriptFile	Status.szScriptFile の値を指定します。現在のスクリプト ファイル (つまり SetStatusEx への呼び出しを含むスクリプト ファイル) を指定するには、予約識別子 _FILE_ を渡します。
nScriptLine	Status.nScriptLine の値を指定します。現在の行番号 (つまり SetStatusEx への呼び出しの行番号) を指定するには、予約識別子 _LINE_ を渡します。
nScriptError	Status.nScriptError の値を指定します。

戻り値

テーブル 160 · SetStatusEx の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がステータスプロパティを設定しました。
< ISERR_SUCCESS	関数がステータスプロパティの設定に失敗しました。

SetStatusExStaticText

SetStatusExStaticText 関数は、STATUSEX ダイアログでステータステキストの上で表示されるステータステキストを設定します。イベント OnFirstUIBefore、OnMaintUIBefore、OnUpdateUIBefore は自動的に、この関数を呼んで、ステータステキストを適切に設定します。

構文

```
SetStatusExStaticText ( szString);
```

パラメーター

テーブル 161 ・ SetStatusExStaticText のパラメーター

パラメーター	説明
szString	静的テキストとして設定されるテキスト。次の標準文字列は、SdLoadString 関数を使用して利用できます。 ・ IDS_IFX_STATUSEX_STATICTEXT_FIRSTUI Ñ InstallShield Wizard は %P をインストールしています。 ・ IDS_IFX_STATUSEX_STATICTEXT_MAINTUI_MODIFY Ñ InstallShield Wizard は %P を変更しています。 ・ IDS_IFX_STATUSEX_STATICTEXT_MAINTUI_REPAIR Ñ InstallShield Wizard は %P を修復しています。 ・ IDS_IFX_STATUSEX_STATICTEXT_MAINTUI_REMOVEALL Ñ InstallShield Wizard は %P を削除しています。 ・ IDS_IFX_STATUSEX_STATICTEXT_UPDATEUI Ñ InstallShield Wizard は %VI バージョンの %P を %VS へアップデートしています ***メモ・**ステータス ダイアログはテキスト置換、%P をサポートしていますが、%VI や %VS などその他のテキスト置換はサポートしていません。%P 以外のテキスト置換を使用して静的テキストの文字列を設定する場合は、SetStatusExStaticText を呼び出す前に SdSubstituteProductInfo 関数を呼び出してこれらのテキスト置換を正しい値に更新する必要があります。*

戻り値

テーブル 162 ・ SetStatusExStaticText の戻り値

戻り値	説明
ISERR_SUCCESS	関数が成功したことを示します。
< ISERR_SUCCESS	関数の実行に失敗したことを示します。

SetStatusWindow

SetStatusWindow 関数は、進行状況インジケーター (ステータスバー) の完了率インジケーターの初期値や現在の値を設定して、進行状況インジケーター (ステータスバー) の一番上の行に表示する現在のメッセージを指定します。

FeatureMoveData を利用してファイルをインストールするセットアップでは、FeatureMoveData を呼び出す前に SetStatusWindow を呼び出し、完了率インジケーターを 0% に設定し、インジケーターの一番上の行を消去しなくてはなりません。SetStatusWindow をさらに呼び出す必要はありません。機能のファイルをインストール中、各機能の [ステータステキスト] 文字列がステータスバーのトップの行に自動的に表示されます。

FeatureMoveData を呼び出す前に、セットアップは StatusUpdate 関数を呼び出して、ファイル転送中の完了率インジケーターの自動アップデートを有効にする必要があります。インストールするファイルやパスの名前をステータスバーの 2 番目の行に表示できるようにするには、FeatureMoveData を呼び出す前に INDVFILESTATUS パラメーターを使って Enable を呼び出します。これらの呼び出しの後、完了率インジケーターはファイル転送の最中スムーズに更新され、FeatureMoveData 関数が呼び出されたとき、転送中の各ファイルの名前が表示されます。

CopyFile または XCopyFile 関数を使用してファイルをインストールするセットアップでは、後に続く CopyFile または XCopyFile への呼び出しの間でインジケーターの一番上の行のメッセージを変更するために、SetStatusWindow を複数回呼び出さなければならないことがあります。StatusUpdate と Enable は (パラメーターを INDVFILESTATUS にした状態で)、通常、CopyFile と XCopyFile で使用できます。StatusUpdate を呼び出して自動アップデートを有効にした場合、完了率インジケーターを変更するのに、後に続く CopyFile または XCopyFile との間に SetStatusWindow を呼び出す必要はありません。

構文

SetStatusWindow (nPercent, szString);

パラメーター

テーブル 163 · SetStatusWindow のパラメーター

パラメーター	説明
nPercent	完了率インジケーターによって表示されるパーセンテージを表す 0 と 100 の間の値を指定します。FeatureMoveData を呼び出す前にインジケーターをリセットする場合は、0 を指定します。完了率インジケーターを変更せずにステータスバーの一番上の行に表示されるメッセージを変更するには、このパラメーターに -1 を指定します。
szString	ステータスバーの一番上の行に表示する文字列を指定します。'DisplayText' パラメーターがこの機能に (IDE で) 指定された場合、FeatureMoveData を呼び出すと、指定した文字列は自動的にこのパラメーターに指定したテキストをすべて上書きするので注意してください。

戻り値

この関数は値を返しません。

SetStatusWindow の例

この関数を基本の MSI インストールで利用することはできません。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
```

```
*
* SetStatusWindow 関数のデモンストレーションを行います。
*
* SetStatusWindow が呼び出されて、プログレスバーが設定され、
* 進行状況インジケーターにテキストが表示されます。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が参照する
*    ディレクトリとファイルを作成してください。
*
¥*--------------------------------------------------------------*/

#define  SOURCE_DIR  "C:¥¥Source"
#define  TARGET_DIR  "C:¥¥Target"
#define  TARGET_FILE "ISExampl.txt"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

  export prototype ExFn_SetStatusWindow(HWND);

function ExFn_SetStatusWindow(hMSI)
begin

    // 進行状況インジケーターを有効にします。
    Enable (STATUS);

    // プログレスバーを 33 % に設定してメッセージを表示します。
    SetStatusWindow (33, " ファイルをコピーしています ...");

    // ウィンドウが適切に表示されることを確認するため、2 秒間遅延させます。
    Delay (2);

    // ソースファイルディレクトリのファイルをターゲットディレクトリへコピーします
    CopyFile (SOURCE_DIR ^ TARGET_FILE, TARGET_DIR ^ TARGET_FILE);

    // プログレスバーを 66% に設定してメッセージを表示します。
    SetStatusWindow (66, " ファイルを削除しています ...");
    Delay (2);

    // ターゲットディレクトリにコピーされたファイルを削除します。
    DeleteFile (TARGET_FILE);

    // 完了率インジケーターを 100% に設定してメッセージを表示します。
    SetStatusWindow (100, "SetStatusWindow の例が完了しました。");
    Delay (2);

end;
```

SetTitle

プロジェクト・この情報は、基本の MSI プロジェクトには適用されません。

SetTitle 関数は、nColor の値に従ってメインウィンドウのタイトルバーまたはメインウィンドウの内部にタイトルを表示します。

***ヒント・**セットアップの背景色を設定するには、SetColor を呼び出します。*

背景ウィンドウの内部に表示されるタイトルのフォントとフォントスタイルを設定するには、SetFont を呼び出します。

構文

SetTitle (szTitle, nPointSize, nColor);

パラメーター

テーブル 164 ・ SetTitle のパラメーター

パラメーター	説明
szTitle	メインウィンドウのタイトルバーまたはメインウィンドウの内部のどちらにタイトルを表示するのかを指定します。タイトルをウィンドウのタイトルバーに表示するには、3 番目のパラメーターで定義済み定数 BACKGROUNDCAPTION を指定しなくてはなりません。タイトルがタイトルバーの制限文字数に収まらない場合は右側が切り詰められ、行末に省略記号が付きます。デフォルトのタイトルバーのタイトルは「セットアップ」です。 ***メモ ・** BACKGROUNDCAPTION オプションは下位互換性のみにサポートされています。メインウィンドウのタイトルバーの設定する方法としては、システム変数 IFX_SETUP_TITLE の値を設定するのが最も望ましい方法です。* *色値が 3 番目のパラメーターに渡された場合、タイトルはメインウィンドウの上部に左揃えで表示されます。メインウィンドウに収まらないタイトルは右側が切り詰められて表示されます。一行以上に及ぶタイトルを作成するには、改行する場所に改行文字を埋め込みます。*
nPointSize	メインウィンドウ内でタイトルを表示するサイズをポイントで指定します。推奨されるサイズは 24 ポイントです。3 番目のパラメーターが BACKGROUNDCAPTION の場合、このパラメーターは無視されることに注意してください。

テーブル 164 ・ SetTitle のパラメーター (続き)

パラメーター	説明
nColor	色または定数 BACKGROUNDCAPTION のどちらかを指定します。 ***メモ ・** nColor を BACKGROUNDCAPTION に設定して SetTitle を呼び出しても、ウィンドウモードで実行しないセットアップには影響しません。セットアップを標準ウィンドウで実行するには、まず Enable をパラメーターを DEFWINDOWMODE または FULLWINDOWMODE で呼び出してから、Enable をパラメーター BACKGROUND で呼び出してウィンドウを表示します。* ・ szTitle の値がメインウィンドウのタイトルバーに表示されることを示すには、定義済み定数 BACKGROUNDCAPTION をこのパラメーターで渡します。BACKGROUNDCAPTION を指定するとき、nPointSize は無視されます。タイトルバーのタイトルの色とサイズはエンドユーザーのシステム設定によって決定されます。このオプションは、ウィンドウモードで実行しないセットアップには影響しないので、ご注意ください。 ・ szTitle の値をセットアップウィンドウの内部に表示することを示すには、そのタイトルの色を次の定数 (対応する RGB 値は括弧に示されています) のひとつを渡して指定するか、または RGB 関数を呼び出して特定のカスタム色を示します : BLACK Ñ (0, 0, 0) BLUE Ñ (0, 0, 255) GREEN Ñ (0, 255, 0) MAGENTA Ñ (255, 0, 127) RED Ñ (255, 0, 0) WHITE Ñ (255, 255, 255) YELLOW Ñ (255, 255, 0) ***メモ ・** マジェンタは Windows 16 色パレットの標準色ではありません。*

戻り値

テーブル 165 ・ SetTitle の戻り値

戻り値	説明
0	関数がセットアップのタイトルを正しく設定したことを示します。
< 0	関数がセットアップのタイトルを設定できなかったことを示します。

SetTitle の例

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SetTitle 関数のデモンストレーションを行います。
 *
 * このスクリプトはタイトルを黄色、24 ポイントのタイプで表示します。
 *
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_SetTitle(HWND);

function ExFn_SetTitle(hMSI)
begin

   // 背景ウィンドウを標準最大化されたウィンドウにします。
   Enable (FULLWINDOWMODE);

   // 背景色を青に設定します。
   SetColor (BACKGROUND, BK_BLUE);

   // ウィンドウにタイトルを表示します。タイトル文字列の改行文字は
   // 次の行で強制的に "Example" を表示します。
   SetTitle ("SetTitle¥nExample", 24, YELLOW);

   // 背景ウィンドウを表示します
   Enable (BACKGROUND);

   // ウィンドウを 3 秒間開いた状態にします。
   Delay (3);

end;
```

SetUpdateStatus

SetUpdateStatus 関数は現在使用されていません。この関数を呼び出すと、ISERR_NOT_IMPL が返されます。

構文

void SetUpdateStatus(BOOL);

SetUpdateStatusReboot

SetUpdateStatusReboot 関数は現在使用されていません。この関数を呼び出すと、ISERR_NOT_IMPL が返されます。

構文

void SetUpdateStatusReboot(BOOL);

SetupType

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

- *InstallScript*
- *InstallScript MSI*

SetupType 関数は、エンドユーザーが、3 種類の標準セットアップ ([標準]、[最小]、[カスタム]) の中から 1 つを選択できるダイアログを表示します。これらのセットアップオプションは標準の記述テキストで表示されます。他のセットアップの種類を追加したり、表示されたセットアップの名前や内容を変更したい場合は、SetupType 関数ではなく、SdSetupTypeEx を呼び出してください。

***注意・**エンドユーザーが機能ダイアログを使用して、選択したセットアップに対応する機能を選択または選択解除した後で [セットアップ] ダイアログに戻った場合、これらの選択は失われます。この問題が発生するのは、SetupType 関数が呼び出される度に、デフォルトの機能選択が自動的にリセットされるからです。*

構文

SetupType (szTitle, szMsg, szReserved, nType, nReserved);

パラメーター

テーブル 166 · SetupType のパラメーター

パラメーター	説明
szTitle	このダイアログのタイトルを指定します。デフォルトのタイトル [セットアップタイプ] を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg	ダイアログの上部に表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szReserved	このパラメーターにヌル文字列 ("") を渡します。別の値を渡すことはできません。
nType	ダイアログが開くときのデフォルトのセットアップの種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · TYPICAL– デフォルトのセットアップの種類を [標準] に定義します。 · COMPACT– デフォルトのセットアップの種類を [コンパクト] に定義します。 · CUSTOM– デフォルトのセットアップの種類を [カスタム] に定義します。
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。

戻り値

テーブル 167 · SetupType の戻り値

戻り値	説明
TYPICAL (301)	エンドユーザーが、[標準] セットアップの種類を選択したことを示します。
COMPACT (302)	エンドユーザーが、[コンパクト] セットアップの種類を選択したことを示します。
CUSTOM (303)	エンドユーザーが、[カスタム] セットアップの種類を選択したことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。

追加情報

- InstallScript プロジェクトでセットアップの種類 ダイアログを表示しない場合、スクリプトは以下の手順の 1 つを実行しなくてはなりません。
  - セットアップの種類を選択する。
  - たとえば SdFeatureTree のような機能選択ダイアログ関数を呼び出す。
  - 直接機能を選択する。

- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SetupType の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SetupType 関数のデモンストレーションを行います。
 *
 * コメント : このスクリプト例を実行するには、いくつかの機能および
 いくつかの機能および / またはサブ機能がある
 プロジェクトを作成 ( またはプロジェクトに挿入 ) します。
 *          この例にはアンインストール機能のセットアップが
 *          含まれます。
 *
 ¥*---------------------------------------------------------------------*/

// ここで機能名を指定します。これらの名前は、
// IDE で機能に付けた名前です。NULL ("") 文字列は基本の機能を指定します
#define ASKDESTTITLE      " インストール先の選択 "
#define ASKDESTMSG        " アプリケーションのインストール先を選択してください。"
#define SETUPTYPETITLE    " セットアップの種類の選択 "
#define SETUPTYPEMSG      " セットアップの種類を選択してください。"
#define FEATURE           ""
#define SDFEATDLGTITLE    " 機能の選択 "
#define SDFEATDLGMSG      " インストールする機能とインストール先を選択してください。"
#define APPBASE_PATH              " 会社名 ¥¥ ワードプロセッサ "
#define COMPANY_NAME " 会社名 "
#define PRODUCT_NAME              " ワードプロセッサ "
#define  PRODUCT_VERSION     "1.0"
#define PRODUCT_KEY               " ワードプロセッサ "
#define DEINSTALL_KEY             " ワードプロセッサ "
#define UNINSTALL_NAME            " ワードプロセッサ "

    prototype HandleFeatureError (NUMBER);


// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_SetupType(HWND);

function ExFn_SetupType(hMSI)
    STRING svLogFile;
    NUMBER nvDisk, nResult;
begin
```

```
    // セットアップダイアログで [ 戻る ] ボタンを無効にします。
    Disable(BACKBUTTON);

    // インストール先の場所を取得します。
    INSTALLDIR = PROGRAMFILES ^ APPBASE_PATH;
    AskDestPath (ASKDESTTITLE, ASKDESTMSG, INSTALLDIR , 0);

    //SdSetupType を使ってセットアップの種類とターゲットの場所を取得します。
    INSTALLDIR = PROGRAMFILES ^ APPBASE_PATH;
    nResult = SetupType (SETUPTYPETITLE, SETUPTYPEMSG, "", TYPICAL, 0);

    // [ カスタム ] セットアップの種類が選択された場合、インストールする機能や
    // 必要な場合はインストール場所をユーザーが指定できるようにします。
    if (nResult = CUSTOM) then
       SdFeatureDialogAdv (SDFEATDLGTITLE, SDFEATDLGMSG,
                           INSTALLDIR, FEATURE);
    endif;

    // インストールをセットアップします。
    InstallationInfo (COMPANY_NAME, PRODUCT_NAME,
                      PRODUCT_VERSION, PRODUCT_KEY);

    svLogFile = "Uninst.isu";
    DeinstallStart (INSTALLDIR, svLogFile, DEINSTALL_KEY, 0);
    RegDBSetItem (REGDB_UNINSTALL_NAME, UNINSTALL_NAME);

    // 機能選択に基づいてファイルを転送します。エラーを処理します。
    Enable (STATUSDLG);
    Enable (INDVFILESTATUS);
    StatusUpdate (ON, 100);
    nResult = FeatureMoveData (MEDIA, nvDisk, 0);
    HandleFeatureError (nResult);

    Disable (INDVFILESTATUS);
    Disable (STATUSDLG);

end;

/*---------------------------------------------------------------------------*¥
 *
 * 関数 :  HandleFeatureError
 *
 * 目的 :  この関数は、機能関数が戻した値を評価し、
 *          関数によって返された値を評価し、その値が 0 以下だとエラー
 *          番号を返してセットアップを終了します。
 *
/*---------------------------------------------------------------------------*¥
function HandleFeatureError (nResult)
  NUMBER  nvError;
  STRING  svFeatureSource, svFeature, svComponent, svFile;
begin
  if (nResult < 0) then
    ComponentError (svFeatureSource, svFeature, svComponent, svFile, nvError);
    SprintfBox (INFORMATION, " データ転送エラー情報 ",
            "FeatureError は " +
            " 次のデータ転送エラーを戻しました : ¥n" +
            " セットアップは終了します。¥n¥n" +
            " メディア名 : %s¥n 機能 : %s¥n コンポーネント : %s¥n" +
            " ファイル : %s¥n エラー番号 : %ld",
```

```
            svFeatureSource, svFeature, svComponent, svFile, nvError);
    abort;
  endif;
end;
```

SetupType2

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SetupType2 関数は、エンドユーザーが、2 種類の標準セットアップ ([完全] または [カスタム]) の中から 1 つを選択できるダイアログを表示します。これらのセットアップオプションは標準の記述テキストで表示されます。他のセットアップの種類を追加したり、表示されたセットアップの名前や内容を変更したい場合は、SetupType 関数ではなく、SdSetupTypeEx を呼び出してください。

注意・エンドユーザーが機能ダイアログを使用して、選択したセットアップに対応する機能を選択または選択解除した後で [セットアップ] ダイアログに戻った場合、これらの選択は失われます。この問題が発生するのは、SetupType2 関数が呼び出される度に、デフォルトの機能選択が自動的にリセットされるからです。

構文

```
SetupType2 ( szTitle, szMsg, szReserved, nType, nReserved );
```

パラメーター

テーブル 168 · SetupType2 のパラメーター

パラメーター	説明
szTitle	このダイアログのタイトルを指定します。デフォルトのタイトル「Setup Type2」を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szMsg	ダイアログの上部に表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szReserved	このパラメーターにヌル文字列 ("") を渡します。他の値は使用できません。
nType	ダイアログが開くときのデフォルトのセットアップの種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · COMPLETE– デフォルトのセットアップの種類を [完全] に定義します。 · CUSTOM– デフォルトのセットアップの種類を [カスタム] に定義します。
nReserved	このパラメーターでゼロを渡します。他の値は使用できません。

戻り値

テーブル 169 · SetupType2 の戻り値

戻り値	説明
COMPLETE (304)	[完全] セットアップの種類が選択されたことを示します。
CUSTOM (303)	エンドユーザーが、[カスタム] セットアップの種類を選択したことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。
< ISERR_SUCCESS	ダイアログが表示されなかったことを示します。

追加情報

- InstallScript プロジェクトでセットアップの種類 ダイアログを表示しない場合、スクリプトは以下の手順の 1 つを実行しなくてはなりません。
  - セットアップの種類を選択する。
  - たとえば SdFeatureTree のような機能選択ダイアログ関数を呼び出す。
  - 直接機能を選択する。
- インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

SetupType2 の例

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 *  SetupType2 関数のデモンストレーションを行います。
 *
 * コメント : このスクリプト例を実行するには、いくつかの機能および
 いくつかの機能および / またはサブ機能がある
 プロジェクトを作成 ( またはプロジェクトに挿入 ) します。
 *          この例にはアンインストール機能のセットアップが
 *          含まれます。
 *
 ¥*---------------------------------------------------------------------*/

// ここで機能名を指定します。これらの名前は、
// InstallShield の機能につけたものです。ヌル ("") 文字列は
// 基本の機能を指定します。
#define ASKDESTTITLE      " インストール先の選択 "
#define ASKDESTMSG        " アプリケーションのインストール先を選択してください。"
#define SETUPTYPETITLE    " セットアップの種類の選択 "
#define SETUPTYPEMSG      " セットアップの種類を選択してください。"
#define FEATURE           ""
#define SDFEATDLGTITLE    " 機能の選択 "
#define SDFEATDLGMSG      " インストールする機能とインストール先を選択してください。"
#define APPBASE_PATH              " 会社名 ¥¥ ワードプロセッサ "
#define COMPANY_NAME " 会社名 "
#define PRODUCT_NAME              " ワードプロセッサ "
#define  PRODUCT_VERSION     "1.0"
#define PRODUCT_KEY               " ワードプロセッサ "
#define DEINSTALL_KEY             " ワードプロセッサ "
#define UNINSTALL_NAME            " ワードプロセッサ "

    prototype HandleFeatureError (NUMBER);


// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_SetupType2(HWND);

function ExFn_SetupType2(hMSI)
    STRING svLogFile;
    NUMBER nvDisk, nResult;
begin

    // セットアップダイアログで [ 戻る ] ボタンを無効にします。
    Disable(BACKBUTTON);

    // インストール先の場所を取得します。
```

```
    INSTALLDIR = PROGRAMFILES ^ APPBASE_PATH;
    AskDestPath (ASKDESTTITLE, ASKDESTMSG, INSTALLDIR , 0);

    // SdSetupType2 を使ってセットアップの種類とターゲット場所を取得します。
    INSTALLDIR = PROGRAMFILES ^ APPBASE_PATH;
    nResult = SetupType2 (SETUPTYPETITLE, SETUPTYPEMSG, "", COMPLETE, 0);

    // [ カスタム ] セットアップの種類が選択された場合、インストールする機能や
    // 必要な場合はインストール場所をユーザーが指定できるようにします。
    if (nResult = CUSTOM) then
        SdFeatureDialogAdv (SDFEATDLGTITLE, SDFEATDLGMSG,
                            INSTALLDIR, FEATURE);
    endif;

    // インストールをセットアップします。
    InstallationInfo (COMPANY_NAME, PRODUCT_NAME,
                      PRODUCT_VERSION, PRODUCT_KEY);

    svLogFile = "Uninst.isu";
    DeinstallStart (INSTALLDIR, svLogFile, DEINSTALL_KEY, 0);
    RegDBSetItem (REGDB_UNINSTALL_NAME, UNINSTALL_NAME);

    // 機能選択に基づいてファイルを転送します。エラーを処理します。
    Enable (STATUSDLG);
    Enable (INDVFILESTATUS);
    StatusUpdate (ON, 100);
    nResult = FeatureMoveData (MEDIA, nvDisk, 0);
    HandleFeatureError (nResult);

    Disable (INDVFILESTATUS);
    Disable (STATUSDLG);

end;

/*---------------------------------------------------------------------------*¥
 *
 * 関数 :  HandleFeatureError
 *
 * 目的 :  この関数は、機能関数が戻した値を評価し、
 *          関数によって返された値を評価し、その値が 0 以下だとエラー
 *          番号を返してインストールを終了します。
 *
/*---------------------------------------------------------------------------*¥
function HandleFeatureError (nResult)
  NUMBER  nvError;
  STRING  svFeatureSource, svFeature, svComponent, svFile;
begin
  if (nResult < 0) then
    ComponentError (svFeatureSource, svFeature, svComponent, svFile, nvError);
    SprintfBox (INFORMATION, " データ転送エラー情報 ",
            "FeatureError は " +
            " 次のデータ転送エラーを戻しました : ¥n" +
            " セットアップは終了します。¥n¥n" +
            " メディア名 : %s¥n 機能 : %s¥n コンポーネント : %s¥n" +
            " ファイル : %s¥n エラー番号 : %ld",
            svFeatureSource, svFeature, svComponent, svFile, nvError);
    abort;
  endif;
end;
```

ShowObjWizardPages

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

ShowObjWizardPages 関数は、プロジェクトの UI イベントハンドラーで呼び出され、インクルードされた各オブジェクトの対応 UI イベントハンドラーが実行されます。たとえば、**ShowObjWizardPages** をプロジェクトの OnFirstUIBefore イベント ハンドラーの中で呼び出すと、インクルードされた各オブジェクトの対応 OnFirstUIBefore コードが実行されます。

構文

```
ShowObjWizardPages ( nDirection );
```

パラメーター

テーブル 170 · ShowObjWizardPages のパラメーター

パラメーター	説明
nDirection	測定単位を示すために、以下のあらかじめ定義された定数のうち 1 つを渡します。 • **NEXT** Ñ 以下を指定します : 呼び出された最初の UI イベントハンドラーは、プロジェクトの [コンポーネント] ペインで最初に一覧表示されているオブジェクトの UI イベントハンドラーで、2 番目に呼び出された UI イベントハンドラーは、プロジェクトの [コンポーネント] ペインで 2 番目に一覧表示されているオブジェクトの UI イベントハンドラー、というようになります。 WizardDirection 関数がオブジェクトの UI イベントハンドラーで呼び出されると、NEXT という値が返されます。 • **BACK** Ñ 以下を指定します : 呼び出された最初の UI イベントハンドラーは、プロジェクトの [コンポーネント] ペインで最後に一覧表示されているオブジェクトの UI イベントハンドラーで、2 番目に呼び出された UI イベントハンドラーは、プロジェクトの [コンポーネント] ペインで最後から 2 番目に一覧表示されているオブジェクトの UI イベントハンドラー、というようになります。 WizardDirection 関数がオブジェクトの UI イベント ハンドラーで呼び出されると、BACK という値が返されます。

戻り値

最も最近実行されたオブジェクト UI イベントハンドラーによって返される値。

追加情報

オブジェクトのダイアログは、nDirection の値に基づいて、2 つの順序のうちのいずれかで表示されます。オブジェクトのダイアログの表示順をさらに制御するには、オブジェクトの ShowxxxxxUIyyyyy メソッドを呼び出してください。オブジェクト UI イベント ハンドラーの詳細は、「オブジェクトのランタイム UI を作成する」を参照してください。

ShowProgramFolder

ShowProgramFolder 関数はプログラムフォルダーを表示します。

構文

```
ShowProgramFolder ( szFolder, nCommand );
```

パラメーター

テーブル 171 ・ ShowProgramFolder のパラメーター

パラメーター	説明
szFolder	表示するフォルダーの名前を指定します。
nCommand	フォルダーの状態を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ SW_SHOW Ñ フォルダーを標準で表示します。 ・ SW_MAXIMIZE Ñ フォルダーを最大化します。 ・ SW_MINIMIZE Ñ フォルダーを最小化します。 ・ SW_RESTORE Ñ ウィンドウを元のサイズに戻します。

戻り値

この関数は値を返しません。

ShowProgramFolder の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* ShowProgramFolder 関数のデモンストレーションを行います。
*
* ShowProgramFolder はフォルダーを表示してから、
```

```
＊フォルダーの状態を変更します。
＊
＊メモ : このスクリプトは、Startup というプログラム フォルダーがなければ
＊　"Startup" というプログラムフォルダーがなければ正しく動作しません。"Startup" と名付けたプログラムフォルダーを
作成するか、
＊　既存のプログラムフォルダーを参照するように定数 FOLDER を
＊　設定してください。さらに、
＊　このスクリプトを実行する際、指定したフォルダーは閉じた状態
＊　または最小化された状態でなくてはなりません。
＊
¥*-------------------------------------------------------------*/

#define  FOLDER  "Startup"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_ShowProgramFolder(HWND);

function ExFn_ShowProgramFolder(hMSI)
begin

   // 指定されたフォルダーを表示します。
   ShowProgramFolder (FOLDER, SW_SHOW);
   Delay (3);

   // フォルダーを最大化します。
   ShowProgramFolder (FOLDER, SW_MAXIMIZE);
   Delay (3);

   // フォルダーを以前の状態に戻します。
   ShowProgramFolder (FOLDER, SW_RESTORE);
   Delay (3);

   // フォルダーを最小化します。
   ShowProgramFolder (FOLDER, SW_MINIMIZE);
   Delay (3);

   // フォルダーを以前の状態に戻します。
   ShowProgramFolder (FOLDER, SW_RESTORE);
   Delay (3);

end;
```

ShowWindow

ShowWindow 関数 は指定されたウィンドウの表示状態を設定します。

構文

```
BOOL ShowWindow(
       HWND hWnd, // ウィンドウのハンドル
       int nCmdShow // 状態を表示
      );
```

パラメーター

テーブル 172 ・ ShowWindow のパラメーター

パラメーター	説明
hWnd	ウィンドウへのハンドルを指定します。

テーブル 172 ・ ShowWindow のパラメーター (続き)

パラメーター	説明
nCmdShow	ウィンドウの表示状態を指定します。アプリケーションを起動したプログラムが STARTUPINFO 構造を含む場合、アプリケーションが ShowWindow を最初に呼び出すときこのパラメーターは無視されます。それ以外の場合、ShowWindow が最初に呼び出されたとき、値は WinMain 関数が *nCmdShow* パラメーター内に取得した値となります。後に続く呼び出しでは、このパラメーターには次の値のひとつが可能です。 ・ SW_FORCEMINIMIZE ― そのウィンドウを保有するスレッドが応答しない場合でも、ウィンドウを最小化します。値は別のスレッドからウィンドウを最小化する場合のみ利用します。 ・ SW_HIDE Ñ ウィンドウを非表示にして別のウィンドウをアクティブにします。 ・ SW_MAXIMIZE Ñ 指定のウィンドウを最大化します。 ・ SW_MINIMIZE Ñ 指定したウィンドウを最小化し、Z 順で次にトップ階層にあるウィンドウをアクティブにします。Z 順は、奥行きを示す z 軸上にあるオーバーラップしたウィンドウの重なりにあるウィンドウの位置です。 ・ SW_RESTORE Ñ ウィンドウをアクティブにして表示します。ウィンドウが最小化または最大化された場合に、システムが元のサイズと位置へ戻します。最小化されたウィンドウを元に戻す場合、アプリケーションはこの値を指定します。 ・ SW_SHOW Ñ ウィンドウをアクティブにし、現在のサイズと位置でそれを表示します。 ・ SW_SHOWDEFAULT Ñ アプリケーションを開始したプログラムによって CreateProcess 関数へ渡された STARTUPINFO 構造 で指定された SW_ 値に基づいて表示状態を設定します。 ・ SW_SHOWMAXIMIZED Ñ ウィンドウをアクティブにし、それを最大化して表示します。 ・ SW_SHOWMINIMIZED Ñ ウィンドウをアクティブにし、それを最小化して表示します。 ・ SW_SHOWMINNOACTIVE Ñ ウィンドウを最小化して表示します。アクティブウィンドウはアクティブの状態のままです。 ・ SW_SHOWNA Ñ ウィンドウを現在のサイズと位置で表示します。アクティブウィンドウはアクティブの状態のままです。 ・ SW_SHOWNOACTIVATE Ñ ウィンドウを一番最後に設定されたサイズと位置で表示します。アクティブウィンドウはアクティブの状態のままです。 ・ SW_SHOWNORMAL Ñ ウィンドウをアクティブにして表示します。ウィンドウが最小化または最大化された場合に、元のサイズと位置へ戻します。ウィンドウを最初に表示する際に、アプリケーションがこの値を指定しなくてはなりません。

戻り値

テーブル 173 ・ ShowWindow の戻り値

戻り値	説明
ゼロ以外	ウィンドウが表示されている場合は戻り値はゼロ以外です。
0	ウィンドウが非表示の場合は戻り値は 0 です。

追加情報

この情報は MSDN トピック ShowWindow 関数 から採用されています。

SilentReadData

SilentReadData 関数は、インストールがサイレントモードで実行されている時に (**Setup.exe** と共に -s スイッチを利用する場合) InstallShield Silent に対し、カスタムダイアログ用の .iss ファイルダイアログデータの読み方を指示します。.iss ファイルは **SilentWriteData** を呼び出して作成することもできます。

スクリプトで **SilentReadData** を利用するには、インストールがサイレントモードで実行していることを最初に確認するようにロジックを構築します。次に示したように、システム変数 MODE のテストに基づいて **SilentReadData** 関数を if-else ステートメントの内部に配置します :

```
if (MODE=SILENTMODE) then
   // ここで SilentReadData を呼び出します
else
   // ここで標準の非サイレント関数を呼び出します
endif;
```

カスタム ダイアログは **EzDefineDialog** や **WaitOnDialog** といった関数を利用したインストールスクリプトで呼び出したり処理するリソースとすることも可能です。または全く外部で、DLL の関数への呼び出しとして実行することもできます。どちらの場合も、ダイアログ ボタンの戻り値 ([次へ]、[戻る]、[キャンセル] など) や、変数で設定または戻される値を .iss ファイルから読み出すには、**SilentReadData** を利用しなくてはなりません。

構文

```
SilentReadData (szSection, szValName, nValType, svVal, nvVal);
```

パラメーター

テーブル 174 · SilentReadData のパラメーター

パラメーター	説明
szSection	.iss ファイルのダイアログ データ セクションの名前を指定します。角括弧 ([ ]) は含めないでください。パラメーター szSection は < 関数名 >-< 数字 > の形式をとります。< 関数名 > はスクリプトで利用されるのと同じダイアログ関数の名前で、< 数字 > は 0 (ゼロ) から始まる、そのスクリプト内でダイアログが利用された回数です。例えば、最初に表示される MyDialog 関数ダイアログは szSection に 値 "MyDialog-0" を持ち、2 回目に表示されるときには "MyDialog-1"、3 回目は "MyDialog-2" といった要領です。
szValName	.iss ファイルのダイアログ データ セクションに表示される値名を指定します。各ダイアログは最低 1 つの szValName ("Result") を持ち、これによりダイアログ ボタン コントロール ([戻る]、[次へ]、[OK]、または [キャンセル]) が戻した値を識別します。その他の値名は、別のダイアログ コントロールに関連付けられた値やデータを識別するのに利用されます。
nValType	ValName の値名に割り当てられた値のデータ型を指定します。値そのものは、nValType の値によって svVal または nvVal に格納されます。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • DATA_STRING Ñ ValName で値名に割り当てられる値は、STRING 型 です。その値は svVal に格納されます。 • DATA_NUMBER Ñ ValName で値名に割り当てられる値は、NUMBER 型です。その値は nvVal に格納されます。 • DATA_COMPONENT Ñ ValName で値名に割り当てられる値は、コンポーネントの名前です。これは ｓ ｖ Val に格納されます。 • DATA_LIST Ñ szValName の値名に割り当てられる値は、InstallScript リスト用のリスト ID です。これは nvVal に格納されます。
svVal	nValType が DATA_STRING または DATA_COMPONENT のとき、szValName の値名に割り当てられる値を指定します。
nvVal	nValType が DATA_NUMBER または DATA_LIST のとき、szValName の値名に割り当てられる値を指定します。

戻り値

テーブル 175 · SilentReadData の戻り値

戻り値	説明
0	**SilentReadData** が InstallShield Silent に対して、カスタム ダイアログ用のダイアログ データの読み取り方法を指示しました。

テーブル 175 · SilentReadData の戻り値 (続き)

戻り値	説明
< 0	**SilentReadData** が、サイレントで実行していないインストールで呼び出されました。 ***メモ · SilentReadData** は、応答ファイル内のカスタム ダイアログ データを読み込めません。インストールが中止します。*

SilentReadData の例

メモ · 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdMakeName 関数、SilentReadData 関数、そして SilentWriteData 関数の
* デモンストレーションを行います。
*
* このスクリプト例ではサイレント インストールでのカスタム ダイアログの
* 処理方法を説明します。下に表示したカスタムダイアログボックス例の
* リソース .dll は、の [ サポートファイル / ビルボード ] ビューに、
* 圧縮された形で格納されます。
*
* ダイアログ例は InstallShield が提供する
* カスタム ダイアログ テンプレートからビルドされたものです。
*
* ダイアログ コントロール ID とその他の情報は
* RESOURCE.H ファイルに含まれます ( 表示されません )。例の最初の行に含まれる
* このファイルは、InstallShield の [InstallScript] ビューに
* 挿入されなくてはなりません。
*
* 例では、Cominit.txt と名づけられたテキストファイルを作成します。インストールが
* サイレントモードで実行される場合、SilentReadData が呼び出され、
* カスタムダイアログコントロール選択が .iss ファイルから
* 読み取られます。そして選択は、.iss ファイルから
* 無事に読み込まれることをデモンストレーションする意味で、Cominit.txt
* ファイルに保存されます。インストールが
* 標準モードで実行された場合、カスタムダイアログが
* 表示され、選択肢が .iss ファイルに記録されて
*  メッセージボックスに表示されます。.ISS ファイルの初期テキストは
* スクリプト例の後に表示されています。
*
¥*---------------------------------------------------------------------*/

#include "Resource.h"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
```

```
#include "Ifx.h"

    export prototype ExFn_SilentReadData(HWND);

function ExFn_SilentReadData(hMSI)
    BOOL   bDone;
    STRING svSection, svComPort, svPulse, svTone, svDial9, svVal;
    NUMBER nvCommDialog, nCmdValue, nPulseState, nToneState;
    NUMBER nDial9State, nResult, nvHandle;
    LIST   listID;
    HWND hwndDlg;
begin

    //COMINIT.TXT テキストファイルを開いて、.ISS ファイルからのカスタムダイアログ選択を
    // そこに保存します。
    OpenFileMode (FILE_MODE_APPEND);
    OpenFile (nvHandle, "c:¥¥rul", "cominit.txt");

    // サイレントモードで実行中の場合、.ISS ファイルから読み込まれます。
    if (MODE=SILENTMODE) then
        SdMakeName (svSection, "COMM_DIALOG", "", nvCommDialog);

        SilentReadData (svSection, "Result", DATA_NUMBER, svVal, nResult);

        if (nResult = 1) then
            // .ISS ファイルからデータが読み込まれます。テキストファイルに
            // 結果を書き込む為に、データを文字列として
            // 読み込みます。
            SilentReadData (svSection, "nPulseState", DATA_STRING,
                        svPulse, nResult);

            SilentReadData (svSection, "nToneState", DATA_STRING,
                        svTone, nResult);

            SilentReadData (svSection, "nDial9State", DATA_STRING,
                        svDial9, nResult);

            // カスタムダイアログの選択をテキストファイル
            // COMINIT.TXT に格納します。
            svVal = " パルスボックスは : " ^ svPulse;
            WriteLine(nvHandle, svVal);

            svVal = " トーン ボックスは : " ^ svTone;
            WriteLine(nvHandle, svVal);

            svVal = "Dial9 ボックスは : " ^ svDial9;
            WriteLine(nvHandle, svVal);
        endif;

    // サイレントモード以外の場合、通常通りカスタムダイアログを
    // 呼び出し、処理します。
    else
        listID = ListCreate (STRINGLIST);
        ListAddString (listID,  "COMM1:", AFTER);
        ListAddString (listID,  "COMM2:", AFTER);
        ListAddString (listID,  "COMM3:", AFTER);
        ListAddString (listID,  "COMM4:", AFTER);

        EzDefineDialog ("MYCOMDIALOG", SUPPORTDIR^"RESOURCE.DLL",
```

```
            "COMM_DIALOG",0);

    bDone = FALSE;

    while (bDone=FALSE)
        nCmdValue = WaitOnDialog ("MYCOMDIALOG");

        switch (nCmdValue)
            case DLG_INIT:
                // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
                // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
                // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
                // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
                //
                hwndDlg = CmdGetHwndDlg("MYCOMDIALOG");
                SdGeneralInit("MYCOMDIALOG", hwndDlg, 0, "");
                CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_CHECKED);
                CtrlSetList ("MYCOMDIALOG", ID_COMPORT, listID);
                CtrlSetState ("MYCOMDIALOG", ID_DIAL9, BUTTON_CHECKED);
            case OK:
                CtrlGetCurSel ("MYCOMDIALOG", ID_COMPORT, svComPort);
                nPulseState = CtrlGetState ("MYCOMDIALOG", ID_PULSE);
                nToneState = CtrlGetState ("MYCOMDIALOG", ID_TONE);
                nDial9State = CtrlGetState ("MYCOMDIALOG", ID_DIAL9);
                nResult = NEXT;
                bDone = TRUE;
            case BACK:
                nResult = BACK;
                bDone = TRUE;
            case RES_PBUT_CANCEL:
                // ユーザーが [ キャンセル ] ボタンをクリックしました。
                Do (EXIT);
            case DLG_CLOSE:
                // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
                Do (EXIT);
            case ID_PULSE:
                nPulseState = CtrlGetState ("MYCOMDIALOG", ID_PULSE);

                if (nPulseState = BUTTON_CHECKED) then
                    CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_UNCHECKED);
                    CtrlSetState ("MYCOMDIALOG", ID_PULSE, BUTTON_CHECKED);
                else
                    CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_CHECKED);
                    CtrlSetState ("MYCOMDIALOG", ID_PULSE, BUTTON_UNCHECKED);
                endif;
            case ID_TONE:
                nToneState = CtrlGetState ("MYCOMDIALOG", ID_TONE);

                if (nPulseState = BUTTON_CHECKED) then
                    CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_CHECKED);
                    CtrlSetState ("MYCOMDIALOG", ID_PULSE, BUTTON_UNCHECKED);
                else
                    CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_UNCHECKED);
                    CtrlSetState ("MYCOMDIALOG", ID_PULSE, BUTTON_CHECKED);
                endif;
            case DLG_ERR:
                MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
                abort;
        endswitch;
```

```
        endwhile;

        EndDialog ("MYCOMDIALOG");
        ReleaseDialog ("MYCOMDIALOG");

        SdMakeName (svSection, "COMM_DIALOG", "", nvCommDialog);

        SilentWriteData (svSection, "nPulseState", DATA_NUMBER,
                svPulse, nPulseState);

        SilentWriteData (svSection, "nToneState", DATA_NUMBER,
                svTone, nToneState);
        SilentWriteData (svSection, "nDial9State", DATA_NUMBER,
                svDial9, nResult);

        if (nPulseState = BUTTON_CHECKED) then
            MessageBox ("パルスボタンが選択されました。", INFORMATION);
        else
            MessageBox ("パルスボタンの選択が解除されました。", INFORMATION);
        endif;

        if (nToneState = BUTTON_CHECKED) then
            MessageBox ("トーンボタンが選択されました。.", INFORMATION);
        else
            MessageBox ("トーンボタンの選択が解除されました。", INFORMATION);
        endif;

        if (nDial9State = BUTTON_CHECKED) then
            MessageBox ("Dial9 ボタンが選択されました。", INFORMATION);
        else
            MessageBox ("Dial9 の選択が解除されました。", INFORMATION);
        endif;

    endif;

    // テキストファイル COMINIT.TXT を閉じます。
    CloseFile (nvHandle);

end;
```

/* 以下のコードは、前述の例の初期 .iss ファイルテキストです。ここでは、角括弧を含んだ <PRODUCT_GUID> が プロジェクトの GUID を
示します。-1001 は BUTTON_CHECKED の数値で、-1002 は BUTTON_UNCHECKED の数値です。

```
[InstallShield Silent]
Version=v9.00
File=Response File
[Application]
Name=MyDialog
Version=4.0
Company= ソフトウェア会社名
[<PRODUCT_GUID>-DlgOrder]
Dlg0=<PRODUCT_GUID>-COMM_DIALOG-0
Count=1
[<PRODUCT_GUID>-COMM_DIALOG-0]
nPulseState=-1001
nToneState=-1002
```

```
nDial9State=-1001
Result=1
*/
```

SilentWriteData

SilentWriteData 関数は、インストールの最中にカスタム ダイアログで選択された内容を記録します。この選択データは、InstallShield Silent 用に .iss ファイルに書き込まれます。インストール中に .iss ファイルを書き込むには、Setup.exe で -r スイッチを利用してインストールを実行してください。

カスタム ダイアログは EzDefineDialog や WaitOnDialog といった関数を利用したインストールスクリプトで呼び出したり処理するリソースとすることも可能です。または全く外部で、DLL の関数への呼び出しとして実行することもできます。どちらの場合も、ダイアログ ボタンの戻り値 ([次へ]、[戻る]、[キャンセル] など) や、変数で設定または戻されるすべての値を記録するには、SilentWriteData を利用しなくてはなりません。

構文

```
SilentWriteData ( szSection, szValName, nValType, szVal, nVal );
```

パラメーター

テーブル 176 ・ SilentWriteData のパラメーター

パラメーター	説明
szSection	.iss ファイルのダイアログ データ セクションの名前を指定します。角括弧 ([ ]) は含めないでください。パラメーター szSection は < 関数名 >-< 数字 > の形式をとります。< 関数名 > はスクリプトで利用されるのと同じダイアログ関数の名前で、< 数字 > は 0 (ゼロ) から始まる、そのスクリプト内でダイアログが利用された回数です。例えば、最初に表示される MyDialog 関数ダイアログは szSection に 値 "MyDialog-0" を持ち、2 回目に表示されるときには "MyDialog-1"、3 回目は "MyDialog-2" といった要領です。
szValName	.iss ファイルのダイアログ データ セクションに表示される値名を指定します。各ダイアログは最低 1 つの szValName ("Result") を持ち、これによりダイアログ ボタン コントロール ([戻る]、[次へ]、[OK]、または [キャンセル]) が戻した値を識別します。その他の値名は、別のダイアログ コントロールに関連付けられた値やデータを識別するのに利用されます。
nValType	ValName の値名に割り当てられた値のデータ型を指定します。値そのものは、nValType の値によって szVal または nVal に格納されます。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • DATA_STRING Ñ ValName で値名に割り当てられる値は、STRING 型 です。その値は szVal に格納されます。 • DATA_NUMBER Ñ ValName で値名に割り当てられる値は、NUMBER 型です。その値は n Val に格納されます。 • DATA_COMPONENT Ñ ValName で値名に割り当てられる値は、コンポーネントの名前です。これは szVal に格納されます。 • DATA_LIST Ñ szValName の値名に割り当てられる値は、InstallShield リスト用のリスト ID です。これは nVal に格納されます。
szVal	nValType が DATA_STRING または DATA_COMPONENT のとき、szValName の値名に割り当てられる値を指定します。
nVal	nValType が DATA_NUMBER または DATA_LIST のとき、szValName の値名に割り当てられる値を指定します。

戻り値

テーブル 177 ・ SilentWriteData の戻り値

戻り値	説明
0	SilentWriteData がカスタム ダイアログ用のダイアログ データを Setup.iss へ書き込みました。
< 0	SilentReadData が、カスタム ダイアログ用のダイアログ データを Setup.iss へ書き込むことができませんでした。

SilentWriteData の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SdMakeName 関数、SilentReadData 関数、そして SilentWriteData 関数の
* デモンストレーションを行います。
*
* このスクリプト例ではサイレント インストールでのカスタム ダイアログの
* 処理方法を説明します。下に表示したカスタムダイアログボックス例の
* リソース .dll は、の [ サポートファイル / ビルボード ] ビューに、
* 圧縮された形で格納されます。
*
* ダイアログ例は InstallShield が提供する
* カスタム ダイアログ テンプレートからビルドされたものです。
*
* ダイアログ コントロール ID とその他の情報は
* RESOURCE.H ファイルに含まれます ( 表示されません )。例の最初の行に含まれる
* このファイルは、InstallShield の [InstallScript] ビューに
* 挿入されなくてはなりません。
*
* 例では、Cominit.txt と名づけられたテキストファイルを作成します。インストールが
* サイレントモードで実行される場合、SilentReadData が呼び出され、
* カスタムダイアログコントロール選択が .iss ファイルから
* 読み取られます。そして選択は、.iss ファイルから
* 無事に読み込まれることをデモンストレーションする意味で、Cominit.txt
* ファイルに保存されます。インストールが
* 標準モードで実行された場合、カスタムダイアログが
* 表示され、選択肢が .iss ファイルに記録されて
*  メッセージボックスに表示されます。.ISS ファイルの初期テキストは
* スクリプト例の後に表示されています。
*
¥*---------------------------------------------------------*/

#include "Resource.h"
```

```
// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_SilentWriteData(HWND);

function ExFn_SilentWriteData(hMSI)
    BOOL   bDone;
    STRING svSection, svComPort, svPulse, svTone, svDial9, svVal;
    NUMBER nvCommDialog, nCmdValue, nPulseState, nToneState;
    NUMBER nDial9State, nResult, nvHandle;
    LIST   listID;
    HWND hwndDlg;
begin

    //COMINIT.TXT テキストファイルを開いて、.ISS ファイルからのカスタムダイアログ選択を
    // そこに保存します。
    OpenFileMode (FILE_MODE_APPEND);
    OpenFile (nvHandle, "c:¥¥rul", "cominit.txt");

    // サイレントモードで実行中の場合、.ISS ファイルから読み込まれます。
    if (MODE=SILENTMODE) then
        SdMakeName (svSection, "COMM_DIALOG", "", nvCommDialog);

        SilentReadData (svSection, "Result", DATA_NUMBER, svVal, nResult);

        if (nResult = 1) then

            // .ISS ファイルからデータが読み込まれます。テキストファイルに
            // 結果を書き込む為に、データを文字列として
            // 読み込みます。
            SilentReadData (svSection, "nPulseState", DATA_STRING,
                        svPulse, nResult);

            SilentReadData (svSection, "nToneState", DATA_STRING,
                        svTone, nResult);

            SilentReadData (svSection, "nDial9State", DATA_STRING,
                        svDial9, nResult);

            // カスタムダイアログの選択をテキストファイル
            // COMINIT.TXT に格納します。
            svVal = " パルスボックスは : " ^ svPulse;
            WriteLine(nvHandle, svVal);

            svVal = " トーン ボックスは : " ^ svTone;
            WriteLine(nvHandle, svVal);

            svVal = "Dial9 ボックスは : " ^ svDial9;
            WriteLine(nvHandle, svVal);
        endif;

    // サイレントモード以外の場合、通常通りカスタムダイアログを
    // 呼び出し、処理します。
    else
        listID = ListCreate (STRINGLIST);
        ListAddString (listID,  "COMM1:", AFTER);
        ListAddString (listID,  "COMM2:", AFTER);
        ListAddString (listID,  "COMM3:", AFTER);
        ListAddString (listID,  "COMM4:", AFTER);
```

```
EzDefineDialog ("MYCOMDIALOG", SUPPORTDIR^"RESOURCE.DLL",
            "COMM_DIALOG",0);

bDone = FALSE;

while (bDone=FALSE)
   nCmdValue = WaitOnDialog ("MYCOMDIALOG");

   switch (nCmdValue)
      case DLG_INIT:
         // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
         // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
         // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
         // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
         //
         hwndDlg = CmdGetHwndDlg("MYCOMDIALOG");
         SdGeneralInit("MYCOMDIALOG", hwndDlg, 0, "");
         CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_CHECKED);
         CtrlSetList ("MYCOMDIALOG", ID_COMPORT, listID);
         CtrlSetState ("MYCOMDIALOG", ID_DIAL9, BUTTON_CHECKED);
      case OK:
         CtrlGetCurSel ("MYCOMDIALOG", ID_COMPORT, svComPort);
         nPulseState = CtrlGetState ("MYCOMDIALOG", ID_PULSE);
         nToneState = CtrlGetState ("MYCOMDIALOG", ID_TONE);
         nDial9State = CtrlGetState ("MYCOMDIALOG", ID_DIAL9);
         nResult = NEXT;
         bDone = TRUE;
      case BACK:
         nResult = BACK;
         bDone = TRUE;
      case RES_PBUT_CANCEL:
         // ユーザーが [ キャンセル ] ボタンをクリックしました。
         Do (EXIT);
      case DLG_CLOSE:
         // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
         Do (EXIT);
      case ID_PULSE:
         nPulseState = CtrlGetState ("MYCOMDIALOG", ID_PULSE);

         if (nPulseState = BUTTON_CHECKED) then
            CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_UNCHECKED);
            CtrlSetState ("MYCOMDIALOG", ID_PULSE, BUTTON_CHECKED);
         else
            CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_CHECKED);
            CtrlSetState ("MYCOMDIALOG", ID_PULSE, BUTTON_UNCHECKED);
         endif;
      case ID_TONE:
         nToneState = CtrlGetState ("MYCOMDIALOG", ID_TONE);

         if (nPulseState = BUTTON_CHECKED) then
            CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_CHECKED);
            CtrlSetState ("MYCOMDIALOG", ID_PULSE, BUTTON_UNCHECKED);
         else
            CtrlSetState ("MYCOMDIALOG", ID_TONE, BUTTON_UNCHECKED);
            CtrlSetState ("MYCOMDIALOG", ID_PULSE, BUTTON_CHECKED);
         endif;
      case DLG_ERR:
         MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
```

```
                abort;
            endswitch;

        endwhile;

        EndDialog ("MYCOMDIALOG");
        ReleaseDialog ("MYCOMDIALOG");

        SdMakeName (svSection, "COMM_DIALOG", "", nvCommDialog);

        SilentWriteData (svSection, "nPulseState", DATA_NUMBER,
                svPulse, nPulseState);

        SilentWriteData (svSection, "nToneState", DATA_NUMBER,
                svTone, nToneState);
        SilentWriteData (svSection, "nDial9State", DATA_NUMBER,
                svDial9, nResult);

        if (nPulseState = BUTTON_CHECKED) then
            MessageBox (" パルスボタンが選択されました。", INFORMATION);
        else
            MessageBox (" パルスボタンの選択が解除されました。", INFORMATION);
        endif;

        if (nToneState = BUTTON_CHECKED) then
            MessageBox (" トーンボタンが選択されました。.", INFORMATION);
        else
            MessageBox (" トーンボタンの選択が解除されました。", INFORMATION);
        endif;

        if (nDial9State = BUTTON_CHECKED) then
            MessageBox ("Dial9 ボタンが選択されました。", INFORMATION);
        else
            MessageBox ("Dial9 の選択が解除されました。", INFORMATION);
        endif;

    endif;

    // テキストファイル COMINIT.TXT を閉じます。
    CloseFile (nvHandle);

end;
```

/* 以下のコードは、前述の例の初期 .iss ファイルテキストです。ここでは、角括弧を含んだ <PRODUCT_GUID> が プロジェクトの GUID を
示します。-1001 は BUTTON_CHECKED の数値で、-1002 は BUTTON_UNCHECKED の数値です。

```
[InstallShield Silent]
Version=v9.00
File=Response File
[Application]
Name=MyDialog
Version=4.0
Company= ソフトウェア会社名
[<PRODUCT_GUID>-DlgOrder]
Dlg0=<PRODUCT_GUID>-COMM_DIALOG-0
Count=1
[<PRODUCT_GUID>-COMM_DIALOG-0]
```

```
nPulseState=-1001
nToneState=-1002
nDial9State=-1001
Result=1
*/
```

SizeOf

SizeOf 関数は InstallScript 配列 の要素数、または引数として渡される変数のサイズを読み出します。

文字列変数の値にある文字数を判断するには、SizeOf の代わりに StrLength を使用します。

構文

```
SizeOf ( Item );
```

パラメーター

テーブル 178 ・ SizeOf のパラメーター

パラメーター	説明
項目	変数の名前を指定します。

戻り値

SizeOf は配列の要素数、または引数として渡される変数のサイズを戻します。

SizeWindow

特定のユーザーインターフェイス要素のサイズを変更するには **SizeWindow** 関数を利用します。新しいサイズをピクセル単位で指定します。

インストールは様々なスクリーン解像度で実行する場合もあります。これを実現するため、**GetExtents** 関数を使って画面全体のサイズを決定してから **SizeWindow** 関数呼び出しのパラメーターの割合を利用してユーザー インターフェイス オブジェクトのサイズを指定します。

メモ・この関数は上級開発者のみに推奨します。

構文

```
SizeWindow ( nObject, nDx, nDy );
```

パラメーター

テーブル 179 · SizeWindow のパラメーター

パラメーター	説明
nObject	サイズを変更するオブジェクトを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · BACKGROUND Ñ メイン背景ウィンドウを示します。 · METAFILE Ñ ファイル転送処理中に利用されるビルボードを示します。**SizeWindow** はビットマップ (.bmp) ファイルをサポートしません。 ***メモ ·*** *このパラメーターは、SdBitmap 関数で表示されるメタファイルには影響を与えません。**SdBitmap** はメタファイルが表示されるときに自動的にそのサイズを変更します。* *また、このパラメーターは* Enable(BBRD) *が呼び出されたときに進行状況ダイアログに表示されるビルボードの種類には影響を与えません。* · MMEDIA_AVI Ñ 次に再生される AVI ファイルのウィンドウ サイズを設定します。すべての AVI ビデオはデフォルトサイズで表示されます。サイズを変更すると、ビデオの解像度や鮮明度も変更される可能性があります。 · MMEDIA_AVI Ñ 次に再生される Adobe Flash アプリケーション ファイル (.swf) のウィンドウ サイズを設定します。
nDx	オブジェクトの水平軸上のサイズをピクセル単位で指定します。
nDy	オブジェクトの垂直軸上のサイズをピクセル単位で指定します。

戻り値

テーブル 180 · SizeWindow の戻り値

戻り値	説明
0	関数は、ウィンドウのサイズを正しく変更しました。
< 0	関数はウィンドウのサイズを変更できませんでした。

追加情報

インストールが Adobe Flash アプリケーション ファイル (.swf) または AVI ファイルを実行する場合は、**PlayMMedia** 関数を使用します。

SizeWindow の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * SizeWindow 関数のデモンストレーションを行います。
 *
 * GetExtents が呼び出されて画面の大きさを読み出します。
 *  そして SizeWindow が呼び出されて背景を元のサイズの半分にサイズ調整
 * します。
 *
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_SizeWindow(HWND);

function ExFn_SizeWindow(hMSI)
   NUMBER  nDx, nDy, nObject;
begin

   // 背景を有効にします。
   Enable (BACKGROUND);

   MessageBox (" 背景は元のサイズです。", INFORMATION);

   // 画面の範囲を取得します。
   GetExtents (nDx, nDy);

   // サイズ調整するオブジェクトを設定します。
   nObject = BACKGROUND;

   // 背景ウィンドウを元のサイズの半分のサイズに調整します。
   if (SizeWindow (nObject, (nDx / 2), (nDy / 2)) < 0) then
      MessageBox ("SizeWindow が失敗しました。", SEVERE);
   endif;

   MessageBox ("SizeWindow の呼び出しの後の背景。", INFORMATION);

end;
```

Sprintf

Sprintf 関数は、形式指定子と一致する変数を使用して、変数データから文字列を作成します。Sprintf 関数は、Microsoft Windows API wsprintf 同様に機能します。

構文

Sprintf (svResult, szFormat [,arg] [,...]);

パラメーター

テーブル 181 ・ Sprintf のパラメーター

パラメーター	説明
svResult	3 番目とそれに続くパラメーターに渡された引数から作成され、フォーマットされた文字列を、2 番目のパラメーター、szFormat の指定に従って返します。
szFormat	svResult によって返された文字列に組み込まれている各引数のリテラルテキストを含み、1 つの 書式指定子 を持つ文字列を指定します。
arg	メッセージに含める引数を 9 個まで指定します。メッセージの書式指定子には少なくとも 1 つ引数を指定する必要があります。各引数の型は、各書式指定子の型と一致させる必要があります。 以下の条件の場合、Sprintf はコンパイラエラーを生成したり、実行時に失敗します : ・ 9 個を超える書式指定子と引数が指定されている場合。これはコンパイラエラーの原因となります。 ・ 引数の数と書式指定子の数が一致しない場合。指定子に対応する引数がない場合、結果文字列のその指定子の位置に予期しない文字が入ります。指定子の数を超える引数を指定した場合、余分な引数は結果文字列に挿入されません。 ・ 変数が各書式指定子の型と一致しない場合。結果文字列のその指定子の位置に無効な文字が入ります。

戻り値

Sprintf 関数が成功すると、戻り値は svResult 変数に格納されている文字列の長さ (文字数) になり、これには最後のヌル文字は含まれません。

Sprintf の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* Sprintf 関数のデモンストレーションを行います。
*
```

```
＊このスクリプトは C: ドライブの容量を取得し、Sprintf を
＊呼び出してフォーマットされたメッセージを作成します。その
＊変数に格納されたドライブ名とディスク サイズを
＊変数で格納されます。そしてメッセージが表示されます。
＊
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_Sprintf(HWND);

function ExFn_Sprintf(hMSI)
   STRING  svResult, svMssg;
   NUMBER  nvResult;
begin

   // GetSystemInfo への呼び出し用の字列パラメーターをセットアップします。
   svResult = "C:";

   // C ドライブの容量を取得します。
   GetSystemInfo (DISK_TOTALSPACE, nvResult, svResult);

   // GetSystemInfo が戻した値と組み合わせるメッセージを
   // ビルドします。
   Sprintf (svMssg, " %s ドライブの合計ディスク容量は %ld バイトです。",
         svResult, nvResult);

   // メッセージを表示します。
   MessageBox (svMssg, INFORMATION);

end;
```

SprintfBox

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

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

SprintfBox 関数を使用して、3 種類のアイコンのうちの 1 つ、タイトル、およびフォーマットされたメッセージが含まれるメッセージボックスを表示できます。メッセージ中では変数を使用することができ、この変数の形式は入力するコマンドに応じて決定されます。

SprintfBox は **MessageBox** と似ていますが、**SprintfBox** の方がより柔軟に表示項目をコントロールできます。

注意・外部機能に渡す目的以外で複数ヌル区切り文字列を利用することはできません。

SprintfBox 関数は Microsoft Windows API **MessageBox** を使用してメッセージ ボックスを作成します。メッセージ ボックスのサイズと位置を決定するのは、InstallShield ではなく、操作環境です。また、動作環境によって [OK] ボタンのテキストがローカル言語 (実行する OS の言語) で生成されます。このボタンのテキストを変更することはできません。さまざまな **MessageBox** タイプの使用方法の詳細については、該当する Windows SDK で **MessageBox** Windows API 関数の説明を参照してください。

Windows メッセージボックス定数を使用する場合は、以下の点に注意してください。

- 一部の Windows **MessageBox** タイプ定数は、*InstallShield Program Files フォルダー*¥Script¥Isrt¥Include フォルダーにある **ISRTWindows.h** ファイルで既に定義されています。このファイルは、スクリプトに **Ifx.h** を含めると、自動的にインストールに含まれます。**ISRTWindows.h** に定義されている定数を再定義する必要はありません。再定義すると、コンパイラ警告が発生します。どの定数が定義済みかを判別するには、**ISRTWindows.h** ファイルを参照します。
- **ISRTWindows.h** に定義されていない定数を使用する場合は、インストールスクリプトの宣言ブロックに必ずそれらを定義しなければなりません (#define を使用)。通常 C++ プログラムの一部である **Windows.h** ファイルを、単純挿入することはできません。未定義定数に割り当てる必要がある値は、通常は、該当する Windows SDK や開発ツール付属のインクルードファイルに含まれています。(Microsoft Visual C++ の場合、ほとんどの定数は *InstallShield Program Files フォルダー*¥Script¥Resource フォルダーにある **Winuser.h** ファイルに含まれています。)
- 単一のインストールで Windows のメッセージボックス定数と InstallShield のメッセージボックス定数を併用することはできません。OR 演算子 (|) を使用して InstallShield メッセージボックス定数と Windows メッセージボックス定数と組み合わせても、Windows メッセージボックス定数は無視されます。
- Windows プラットフォームの中には、特定の Windows メッセージボックススタイルをサポートしていないものがあります。インストール先のオペレーティングシステムで特定のスタイルがサポートされているかどうかを判断するには、該当する Windows SDK を参照してください。

構文

SprintfBox (nType, szTitle, szFormat [,arg] [,...]);

パラメーター

テーブル 182 ・ SprintfBox のパラメーター

パラメーター	説明
nType	メッセージボックスに表示するアイコンの種類を指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ INFORMATION ・ WARNING ・ SEVERE Windows API に精通した上級の開発者の方は、パラメーター nType で Windows のネイティブメッセージボックススタイルを使用することにより、メッセージボックスのスタイルを自由に指定できます。ご使用の操作環境のプログラミングマニュアルで、MessageBox または WinMessageBox API の説明を参照してください。ネイティブのメッセージボックススタイルを使用している場合、InstallShield SprintfBox 関数は、Windows API からの戻り値を戻します。したがって、スクリプトで Windows API の戻り値を使用する必要があります。 たとえば、最初のパラメーターとして YES|NO|CANCEL を渡すと、SprintfBox メッセージボックスに [はい]、[いいえ]、そして [キャンセル] ボタンが表示されます。各ボタンの戻り値は、Windows API MessageBox の定義と同じ 6 (IDYES)、7 (IDNO)、および 2 (IDCANCEL) です。スクリプトでは、適切な定数値を数値リテラル、またはスクリプトの宣言ブロックで数値リテラルとして定義されている定数として使用する必要があります。 ***ヒント・**上級の開発者の方は、SprintfBox 関数の最初のパラメーターとして直接 MB_STYLE を使用して、定数 SEVERE、WARNING、または INFORMATION を置き換えることができます。MB_STYLE の値は Windows.h ファイルにリストされています。パラメーター nType に値を直接入力するか、または #define プリプロセッサ命令を使用して、値と関連する定数を定義できます。*
szTitle	メッセージボックスのタイトルを指定します。デフォルトのタイトル [エラー] を表示するには、このパラメーターにヌル文字列 ("") を渡します。
szFormat	メッセージに含む各引数に対して、書式指定子を含む文字列を指定します。

テーブル 182 • SprintfBox のパラメーター (続き)

パラメーター	説明
arg	メッセージに含める引数を 10 個まで指定します。メッセージの書式指定子には少なくとも 1 つ引数を指定する必要があります。各引数の型は、各書式指定子の型と一致させる必要があります。以下の条件の場合、SprintfBox はコンパイラエラーを生成したり、実行時に失敗します : • 10 個を超える書式指定子と引数が指定されている場合、コンパイル エラーが発生します。 • 引数の数と書式指定子の数が一致しない場合。指定子に対応する引数がない場合、結果文字列のその指定子の位置に予期しない文字が入ります。指定子の数を超える引数を指定した場合、余分な引数は結果文字列に挿入されません。 • 変数が各書式指定子の型と一致しない場合。 • 数値 ASCII 値または STRTOCHAR 関数の戻り値ではなく文字そのものが、書式指定子 %c と共に渡された場合。

戻り値

戻り値は、Microsoft Windows 標準のメッセージボックスのスタイルを使用しない限り、重要ではありません。これらのスタイルを使用する場合、戻り値は MessageBox API 関数の戻り値と同じになります。

追加情報

SprintfBox 関数によって表示されるダイアログは、スキンを使って表示することはできません。スキンの指定に関わらず、同じように表示されます。

SprintfBox の例

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

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

***メモ •** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* SprintfBox 関数のデモンストレーションを行います。
*
* このスクリプトは C: ドライブの容量を取得し、Sprintf を
```

```
＊呼び出してフォーマットされたメッセージを表示します。その
＊メッセージには、変数で格納されているドライブ名とディスクサイズが
＊含まれます。
＊
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_SprintfBox(HWND);

function ExFn_SprintfBox(hMSI)
   STRING  svResult;
   NUMBER  nvResult;
begin

   // GetSystemInfo への呼び出し用の字列パラメーターをセットアップします。
   svResult = "C:";

   // C ドライブの容量を取得します。
   GetSystemInfo (DISK_TOTALSPACE, nvResult, svResult);

   // GetSystemInfo が戻した値と組み合わせる
   // メッセージをビルドして表示します。
   SprintfBox (INFORMATION, " システム情報 ",
         " %s ドライブの合計ディスク容量は %ld バイトです。",
         svResult, nvResult);

end;
```

SprintfMsiLog

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *基本の MSI*
- *InstallScript MSI*

SprintfMsiLog 関数は Windows Installer ログ ファイルに直接メッセージを直接書き込みます。

構文

```
SprintfMsiLog ( szFormat [,arg] [,...]);
```

パラメーター

テーブル 183 · SprintfMsiLog のパラメーター

パラメーター	説明
szFormat	svResult によって返された文字列に組み込まれている各引数のリテラルテキストを含み、1 つの 書式指定子 を持つ文字列を指定します。
arg	メッセージに含める引数を 9 個まで指定することができます。メッセージ内の各書式指定子に 1 つの引数を含まなくてはなりません。各引数のタイプは、それぞれの書式指定子のタイプと一致しなくてはなりません。 以下の条件の場合、SprintfMsiLog はコンパイラエラーを生成したり、実行時に失敗します : · 9 個を超える書式指定子と引数が指定されている場合。これはコンパイラエラーの原因となります。 · 引数の数と書式指定子の数が一致しない場合。指定子に対応する引数がない場合、結果文字列のその指定子の位置に予期しない文字が入ります。指定子の数を超える引数を指定した場合、余分な引数は結果文字列に挿入されません。 · 変数が各書式指定子の型と一致しない場合。結果文字列のその指定子の位置に無効な文字が入ります。

戻り値

この関数への戻り値はありません。関数が成功した場合は、値が Windows Installer ログ ファイルに書き込まれます。関数が失敗した場合は、値が Windows Installer ログ ファイルに書き込まれません。

SQLBrowse

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

- *InstallScript*
- *InstallScript MSI*

SQLBrowse2 関数は **SQLBrowse** 関数に優先します。

SQLBrowse 関数は、エンドユーザーがネットワーク上で提供されているすべての SQL Server のリストを表示することができるダイアログを作成します。

この関数は、InstallScript プロジェクトでは **SQLRT.obl** ファイルに、また InstallScript MSI プロジェクトでは **SQLCONV.obl** ファイルにあります。InstallShield の [SQL スクリプト] ビューを使用する場合、適切なファイルは自動的にリンカの設定に追加されます。このビューを使用していない場合は、適切なファイルを次のようにリンカの設定に追加します : [ビルド] メニューで、[設定] をクリックして、関数をライブラリ (.obl) ボックスに追加します。

構文

SQLBrowse(svServer);

パラメーター

テーブル 184 ・ SQLBrowse のパラメーター

パラメーター	説明
svServer	関数が戻されると、このパラメーターにはエンドユーザーが選択したサーバー名を含んだ文字列が含まれています。 SQL Server データベースへの接続にエイリアス名が使用された場合、このパラメーターにはそのエイリアス名が含まれます。

戻り値

テーブル 185 ・ SQLBrowse の戻り値

戻り値	説明
NEXT	ユーザーが、[OK] ボタンをクリックしました。
CANCEL	ユーザーが、[キャンセル] ボタンをクリックしました。
くISERR_SUCCESS	ダイアログを表示することができませんでした。

SQLBrowse2

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SQLBrowse2 関数は、エンド ユーザーが、接続に指定されたデータベース テクノロジ用にネットワーク上で提供されているすべてのデータベース サーバーのリストを表示できるようにするダイアログを作成します。

構文

SQLBrowse2(szConnection, svServer);

パラメーター

テーブル 186 ・ SQLBrowse2 のパラメーター

パラメーター	説明
szConnection	[SQL スクリプト] ビューで作成した接続の名前を指定します。
svServer	関数が戻されると、このパラメーターにはエンドユーザーが選択したサーバー名を含んだ文字列が含まれています。 SQL Server データベースへの接続にエイリアス名が使用された場合、このパラメーターにはそのエイリアス名が含まれます。

戻り値

テーブル 187 ・ SQLBrowse2 の戻り値

戻り値	説明
NEXT	ユーザーが、[OK] ボタンをクリックしました。
CANCEL	ユーザーが、[キャンセル] ボタンをクリックしました。
< ISERR_SUCCESS	ダイアログを表示することができませんでした。

SQLDatabaseBrowse

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

- *InstallScript*
- *InstallScript MSI*

SQLDatabaseBrowse は、エンド ユーザーが、指定されたデータベース サーバー上で提供されているすべてのデータベース カタログを一覧表示することができるダイアログを作成します。この関数は **SQLRTGetDatabases** を呼び出して、InstallScript プロジェクトの場合は **SQLRT.dll** を、InstallScript MSI プロジェクトの場合は **ISSQLSRV.dll** を使用します。

***メモ・SQLDatabaseBrowse** 関数は SQL 設定ファイルの設定を使用します。したがって、この関数は SQLRTInitialize2 が呼び出された後にのみ呼び出すことができます。*

構文

```
SQLDatabaseBrowse( szConnection, szServer, bvWindowsLogin, szUser, szPassword, svDBCatalog );
```

パラメーター

テーブル 188 ・ SQLDatabaseBrowse のパラメーター

パラメーター	説明
szConnection	[SQL スクリプト] ビューで作成した接続の名前を指定します。
svServer	関数が戻されると、このパラメーターにはエンドユーザーが選択したサーバー名を含んだ文字列が含まれています。
bvWindowsLogin	SQL Server 認証情報ではなく、Windows 認証情報を使用するログインを指定するラジオボタンの初期状態を指定します。これが TRUE の場合、″ パスワード ″ フィールドと ″ ログイン ID″ フィールドは無効になります。FALSE の場合、SQL Server 認識情報が使用されるというメッセージが表示され、パスワードおよびログインフィールドが有効にされます。 一度関数が戻されると、このパラメーターには , エンドユーザーが Windows 認証情報ラジオ ボタンを選択した場合 TRUE が含まれ、SQL Server 認証情報ラジオ ボタンが代わりに選択された場合は FALSE が含まれています。
szUser	″ ログイン ID″ 編集ボックスに表示するデフォルトの文字列を指定します。一度ダイアログ関数が戻されると、このパラメーターにはエンドユーザーが編集フィールドで入力したログイン ID が含まれています。この情報はエンドユーザーが Windows 認証情報の代わりに SQL ログインの使用を選択したときのみ適切です。
szPassword	パスワード 編集フィールドに表示するデフォルトの文字列を指定します。一度ダイアログ関数が戻されると、このパラメーターにはエンドユーザーが編集フィールドで入力したパスワードが含まれています。この情報はエンドユーザーが Windows 認証情報の代わりに SQL ログインの使用を選択したときのみ適切です。
svDBCatalog	SQL データベース カタログの名前を指定します。

戻り値

テーブル 189 ・ SQLDatabaseBrowse の戻り値

戻り値	説明
NEXT	ユーザーが、[OK] ボタンをクリックしました。
CANCEL	ユーザーが、[キャンセル] ボタンをクリックしました。
< ISERR_SUCCESS	ダイアログを表示することができませんでした。

SQLRTComponentInstall

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SQLRTComponentInstall 関数は、スクリプトがインストール時に実行されるようにスケジュールされている場合、指定されたコンポーネントに関連付けられている SQL スクリプトを実行します。

メモ・OnSQLServerInitialize イベント ハンドラー (InstallScript プロジェクトの場合) または OnSQLLogin イベント ハンドラー (InstallScript MSI プロジェクトの場合) が呼び出される前に SQL 関連のビルトイン関数を呼び出す場合、SQLRTInitialize2 関数を先に呼び出します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。

構文

SQLRTComponentInstall(szComponent);

パラメーター

テーブル 190・SQLRTComponentInstall のパラメーター

パラメーター	説明
szComponent	SQL スクリプトを含むコンポーネントの名前を指定します。

戻り値

テーブル 191・SQLRTComponentInstall の戻り値

戻り値	説明
>=ISERR_SUCCESS	この関数は SQL スクリプトを実行することができました。
<ISERR_SUCCESS	この関数は SQL スクリプトを実行することができませんでした。
SQL_ERROR_NOT_INITIALIZED (-10)	**SQLRT.dll** がロードされていません。

SQLRTComponentUninstall

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SQLRTComponentUninstall 関数は、スクリプトがアンインストール時に実行されるようにスケジュールされている場合、指定されたコンポーネントに関連付けられている SQL スクリプトを実行します。

メモ・OnSQLServerInitialize イベント ハンドラー (InstallScript プロジェクトの場合) または OnSQLLogin イベント ハンドラー (InstallScript MSI プロジェクトの場合) が呼び出される前に SQL 関連のビルトイン関数を呼び出す場

合、SQLRTInitialize2 関数を先に呼び出します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。

構文

SQLRTComponentUninstall(szComponent);

パラメーター

テーブル 192 - SQLRTComponentUninstall のパラメーター

パラメーター	説明
szComponent	SQL スクリプトを含むコンポーネントの名前を指定します。

戻り値

テーブル 193 - SQLRTComponentUninstall の戻り値

戻り値	説明
>=ISERR_SUCCESS	この関数は SQL スクリプトを実行することができました。
<ISERR_SUCCESS	この関数は SQL スクリプトを実行することができませんでした。
SQL_ERROR_NOT_INITIALIZED (-10)	**SQLRT.dll** がロードされていません。

SQLRTConnect

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

SQLRTConnect2 関数は **SQLRTConnect** 関数に優先します。

SQLRTConnect 関数は指定された認証情報を使用して接続を確立します。

***メモ -** OnSQLServerInitialize イベント ハンドラー (InstallScript プロジェクトの場合) または OnSQLLogin イベント ハンドラー (InstallScript MSI プロジェクトの場合) が呼び出される前に SQL 関連のビルトイン関数を呼び出す場合、SQLRTInitialize2 関数を先に呼び出します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

SQLRTConnect(szConnection, szServer, bTrust, szUserName, szPassword);

パラメーター

テーブル 194 · SQLRTConnect のパラメーター

パラメーター	説明
szConnection	[SQL スクリプト] ビューで作成した接続の名前を指定します。
szServer	SQL Server を指定します。
bTrust	SQL Server 認証情報ではなく、Windows 認証情報を使用するログインを指定するラジオボタンの初期状態を指定します。これが TRUE の場合、″ パスワード ″ フィールドと ″ ログイン ID″ フィールドは無効になります。FALSE の場合、SQL Server 認識情報が使用されるというメッセージが表示され、パスワードおよびログインフィールドが有効にされます。 一度関数が戻されると、このパラメーターには , エンドユーザーが Windows 認証情報ラジオ ボタンを選択した場合 TRUE が含まれ、SQL Server 認証情報ラジオ ボタンが代わりに選択された場合は FALSE が含まれています。
szUserName	SQL Server ログイン情報を指定します。
szPassword	ユーザー アカウントに関連付けられたパスワードを指定します。

戻り値

テーブル 195 · SQLRTConnect の戻り値

戻り値	説明
>=ISERR_SUCCESS	この関数は接続を確立できました。
<ISERR_SUCCESS	この関数は接続の確立に失敗しました。
SQL_ERROR_NOT_INITIALIZED (-10)	**SQLRT.dll** がロードされていません。

SQLRTConnect2

***プロジェクト ·** この情報は、InstallScript プロジェクトに適用します。*

SQLRTConnect2 関数は接続を確立します。インストール中にスクリプトを実行するために接続が利用される場合、この関数をファイル転送の前に呼び出す必要があります。**SQLRTConnect2** は、接続の確立に失敗した場合、データベース サーバー名を戻します。

***メモ・SQLRTConnect2** は* **SQLRT.dll** *を使用します。したがって、この関数は、SQLRTInitialize2 が呼び出されたあとにのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

SQLRTConnect2(szConnection, szServer, bWinLogin, szUser, szPassword, szDatabaseServer);

パラメーター

テーブル 196 ・ SQLRTConnect2 のパラメーター

パラメーター	説明
szConnection	[SQL スクリプト] ビューで作成した接続の名前を指定します。
szServer	SQL Server を指定します。
bWinLogin	SQL Server 認証情報ではなく、Windows 認証情報を使用するログインを指定するラジオボタンの初期状態を指定します。これが TRUE の場合、″ パスワード ″ フィールドと ″ ログイン ID″ フィールドは無効になります。FALSE の場合、SQL Server 認識情報が使用されるというメッセージが表示され、パスワードおよびログインフィールドが有効にされます。 一度関数が戻されると、このパラメーターには , エンドユーザーが Windows 認証情報ラジオ ボタンを選択した場合 TRUE が含まれ、SQL Server 認証情報ラジオ ボタンが代わりに選択された場合は FALSE が含まれています。
szUser	SQL Server ログイン情報を指定します。
szPassword	ユーザー アカウントに関連付けられたパスワードを指定します。
szDatabaseServer	データベース サーバーを指定します。

戻り値

テーブル 197 ・ SQLRTConnect2 の戻り値

戻り値	説明
>=ISERR_SUCCESS	この関数は接続を確立できました。
<ISERR_SUCCESS	この関数は接続の確立に失敗しました。
SQL_ERROR_NOT_INITIALIZED (-10)	**SQLRT.dll** がロードされていません。

追加情報

接続するカタログを指定できるようにする場合、OnSQLServerInitialize イベントでの **SQLRTConnect2** 関数の呼び出しを、**SQLRTConnectDB** 関数の呼び出しで置換します。詳細については、「SQLRTConnectDB」を参照してください。

SQLRTConnectDB

***プロジェクト ・** この情報は、InstallScript プロジェクトに適用します。*

特定のカタログへの接続を確立するには、OnSQLServerInitialize イベントでの **SQLRTConnect2** 関数の呼び出しを、**SQLRTConnectDB** 関数の呼び出しで置換します。カタログ名を **SQLRTConnectDB** の szDB パラメーターとして渡します。

実行時に、エンドユーザーがカタログを指定できるようにするには、エンドユーザー定義の変数を szDB として渡します。

メモ・SQLRTConnectDB *は* **SQLRT.dll** *を使用します。したがって、この関数は、SQLRTInitialize2 が呼び出されたあとにのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

SQLRTConnectDB(szConnection, szDB, szServer, bWinLogin, szUser, szPassword);

パラメーター

テーブル 198 · SQLRTConnectDB パラメーター

パラメーター	説明
szConnection	SQL Server 接続を指定します。
szDB	接続するカタログを指定します。 複数のカタログを指定するには、各カタログをセミコロン (;) で区切ります。 実行時に、エンドユーザーがカタログを指定できるようにするには、エンドユーザー定義の変数を szDB として渡します。
szServer	SQL Server を指定します。
bWinLogin	SQL Server 認証情報ではなく、Windows 認証情報を使用するログインを指定するラジオボタンの初期状態を指定します。これが TRUE の場合、″ パスワード ″ フィールドと ″ ログイン ID″ フィールドは無効になります。FALSE の場合、SQL Server 認識情報が使用されるというメッセージが表示され、パスワードおよびログインフィールドが有効にされます。 一度関数が戻されると、このパラメーターには , エンドユーザーが Windows 認証情報ラジオボタンを選択した場合 TRUE が含まれ、SQL Server 認証情報ラジオ ボタンが代わりに選択された場合は FALSE が含まれています。
szUser	SQL Server ログイン情報を指定します。
szPassword	ユーザー アカウントに関連付けられたパスワードを指定します。

戻り値

テーブル 199 · SQLRTConnectDB 戻り値

戻り値	説明
>=ISERR_SUCCESS	この関数は接続を確立できました。
<ISERR_SUCCESS	この関数は接続の確立に失敗しました。
SQL_ERROR_NOT_INITIALIZED (-10)	**SQLRT.dll** がロードされていません。

SQLRTDoRollbackAll

***プロジェクト ·** この情報は、InstallScript プロジェクトに適用します。*

SQLRTDoRollbackAll 関数はロールバック中に実行されるようにスケジュールされた SQL スクリプトをすべて実行します。

***メモ・SQLRTDoRollbackAll** は* **SQLRT.dll** *を使用します。したがって、この関数は、SQLRTInitialize2 が呼び出されたあとにのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

SQLRTDoRollbackAll();

パラメーター

SQLRTDoRollbackAll にパラメーターは使用できません。

戻り値

テーブル 200・SQLRTDoRollbackAll の戻り値

戻り値	説明
>=ISERR_SUCCESS	この関数はスクリプトを実行することができました。
<ISERR_SUCCESS	この関数はスクリプトを実行することができませんでした。
SQL_ERROR_NOT_INITIALIZED (-10)	**SQLRT.dll** がロードされていません。

SQLRTGetBatchList

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

SQLRTGetBatchList 関数は、バッチ モードが有効になっているとき実行する必要がある SQL スクリプトに関連付けられているコンポーネントの一覧を返します。

***メモ・SQLRTGetBatchList** は* **SQLRT.dll** *を使用します。したがって、この関数は、SQLRTInitialize2 が呼び出されたあとにのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

SQLRTGetBatchList(nOperation);

パラメーター

テーブル 201 ・ SQLRTGetBatchList パラメーター

パラメーター	説明
nOperation	このパラメーターを使って、以下の定数のいずれかを渡します。 • SQL_BATCH_INSTALL Ñ インストール中に実行されるようにスケジュールされた SQL スクリプトの一覧を取得します。 • SQL_BATCH_UNINSTALL Ñ アンインストール中に実行されるようにスケジュールされた SQL スクリプトの一覧を取得します。

戻り値

テーブル 202 ・ SQLRTGetBatchList 戻り値

戻り値	説明
listConnections	コンポーネント名の一覧を返します。

追加情報

バッチ モードに関する詳細については、「接続に関連付けられている複数 SQL スクリプトの実行順序を指定する」をご覧ください。

SQLRTGetBatchMode

プロジェクト・*この情報は、InstallScript プロジェクトに適用します。*

SQLRTGetBatchMode 関数は、バッチ モードが有効か、または無効かを戻します。

メモ・SQLRTGetBatchMode *は* **SQLRT.dll** *を使用します。したがって、この関数は、SQLRTInitialize2 が呼び出されたあとにのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

```
SQLRTGetBatchMode();
```

パラメーター

SQLRTGetBatchMode にパラメーターは使用できません。

戻り値

テーブル 203 - SQLRTGetBatchMode の戻り値

戻り値	説明
TRUE	バッチ モードが有効です。
FALSE	バッチ モードが無効です。

追加情報

バッチ モードに関する詳細については、「接続に関連付けられている複数 SQL スクリプトの実行順序を指定する」をご覧ください。

SQLRTGetBrowseOption

プロジェクト - *この情報は、InstallScript プロジェクトに適用します。*

SQLRTGetBrowseOption 関数は、SQL Server の参照コンボ ボックスとリスト ボックス コントロールの参照オプションの現在の値を返します。これらでは、ローカル サーバー、リモート サーバー、サーバー エイリアス、およびこれらのサーバーの組み合わせを表示できます。

メモ - *OnSQLServerInitialize イベント ハンドラー (InstallScript プロジェクトの場合) または OnSQLLogin イベント ハンドラー (InstallScript MSI プロジェクトの場合) が呼び出される前に SQL 関連のビルトイン関数を呼び出す場合、SQLRTInitialize2 関数を先に呼び出します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

```
SQLRTGetBrowseOption( );
```

パラメーター

SQLRTGetBrowseOption にパラメーターは使用できません。

戻り値

テーブル 204 - SQLRTGetBrowseOption Return の値

戻り値	説明
0	SQL Server の参照コンボ ボックスとリスト ボックス コントロールでは、SQL Server (ローカル サーバー、リモート サーバー、およびサーバー エイリアス) が表示されます。 これは、**SQL_BROWSE_ALL** またはゼロ (**0**) が **SQLRTSetBrowseOption** の nBrowseOption パラメーターに渡されたとき、または、**SQLRTSetBrowseOption** 関数が呼び出されなかったときの戻り値です。
1	SQL Server の参照コンボ ボックスとリスト ボックス コントロールでは、すべての使用可能なローカル SQL Server が表示されます。 これは、**SQL_BROWSE_LOCAL** または **1** が **SQLRTSetBrowseOption** の nBrowseOption パラメーターに渡されたときの戻り値です。
2	SQL Server の参照コンボ ボックスとリスト ボックス コントロールでは、すべての使用可能なリモート SQL Server が表示されます。 これは、**SQL_BROWSE_REMOTE** または **2** が **SQLRTSetBrowseOption** の nBrowseOption パラメーターに渡されたときの戻り値です。
3	SQL Server の参照コンボ ボックスとリスト ボックス コントロールでは、すべての使用可能なローカル SQL Server が表示し、SQL Server を削除します。 これは、**SQL_BROWSE_LOCAL | SQL_BROWSE_REMOTE** または **3** が **SQLRTSetBrowseOption** の nBrowseOption パラメーターに渡されたときの戻り値です。
4	SQL Server の参照コンボ ボックスとリスト ボックス コントロールでは、すべての使用可能な SQL Server のエイリアスが表示されます。 これは、**SQL_BROWSE_ALIAS** または **4** が **SQLRTSetBrowseOption** の nBrowseOption パラメーターに渡されたときの戻り値です。
5	SQL Server の参照コンボ ボックスとリスト ボックス コントロールでは、すべての使用可能なローカル SQL Server と SQL Server のエイリアスが表示されます。 これは、**SQL_BROWSE_LOCAL | SQL_BROWSE_ALIAS** または **5** が **SQLRTSetBrowseOption** の nBrowseOption パラメーターに渡されたときの戻り値です。
6	SQL Server の参照コンボ ボックスとリスト ボックス コントロールでは、すべての使用可能なリモート SQL Server と SQL Server のエイリアスが表示されます。 これは、**SQL_BROWSE_REMOTE | SQL_BROWSE_ALIAS** または **6** が **SQLRTSetBrowseOption** の nBrowseOption パラメーターに渡されたときの戻り値です。

テーブル 204 • SQLRTGetBrowseOption Return の値 (続き)

戻り値	説明
7	SQL Server の参照コンボ ボックスとリスト ボックス コントロールでは、SQL Server (ローカル サーバー、リモート サーバー、およびサーバー エイリアス) が表示されます。 これは、**SQL_BROWSE_LOCAL** | **SQL_BROWSE_REMOTE** | **SQL_BROWSE_ALIAS** または **7** が **SQLRTSetBrowseOption** の nBrowseOption パラメーターに渡されたときの戻り値です。

SQLRTGetComponentScriptError

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SQLRTGetComponentScriptError2 関数は **SQLRTGetComponentScriptError** 関数に優先します。

SQLRTGetComponentScriptError 関数は、コンポーネントに関連付けられている SQL スクリプトの実行中に発生した最後のエラーを取得します。

メモ・SQLRTGetComponentScriptError は SQLRT.dll を使用します。したがって、この関数は、SQLRTInitialize2 が呼び出されたあとにのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。

構文

SQLRTGetComponentScriptError(szComponent, szMessage, nvErrorType, nvErrorLine);

パラメーター

テーブル 205 · SQLRTGetComponentScriptError のパラメーター

パラメーター	説明
szComponent	スクリプトに関連付けられているコンポーネントの名前を指定します。
szMessage	データベース サーバーが返したエラー メッセージを指定します。
nvErrorType	エラーの種類を指定します。以下は有効な種類です : · SQL_ERROR_SCRIPT_UNABLE_OPEN_FILE Ñ スクリプト ファイルを開くことができませんでした。 · SQL_ERROR_SCRIPT_CONNECTION_NOT_OPEN Ñ 指定された接続が開いていません。 · SQL_ERROR_GET_SCHEMA_VERSION Ñ ターゲット データベースからスキーマ バージョンを取得中に、エラーが発生しました。 · SQL_ERROR_SET_SCHEMA_VERSION Ñ ターゲット データベースにスキーマ バージョンを設定中に、エラーが発生しました。 · SQL_ERROR_SCRIPT_COMMAND_ERROR Ñ SQL バッチの実行に失敗しました。
nvErrorLine	エラーの行番号を指定します。

戻り値

テーブル 206 · SQLRTGetComponentScriptError の戻り値

戻り値	説明
TRUE	1 つまたは複数のエラーが発生しました。
FALSE	エラーはありませんでした。

SQLRTGetComponentScriptError2

***プロジェクト·** この情報は、InstallScript プロジェクトに適用します。*

SQLRTGetComponentScriptError2 関数は、コンポーネントに関連付けられている SQL スクリプトの実行中に発生した最後のエラーを取得します。この関数では、**SQLRTGetComponentScriptError** 関数で使用できないパラメーター がいくつか使用できます (szScriptName、szTechnology、szServer および szDB)。

メモ・SQLRTGetComponentScriptError *は* **SQLRT.dll** *を使用します。したがって、この関数は、SQLRTInitialize2 が呼び出されたあとにのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

SQLRTGetComponentScriptError2(szComponent, szMessage, nvErrorType, nvErrorLine, szScriptName, szTechnology, szServer, szDB);

パラメーター

テーブル 207 · SQLRTGetComponentScriptError2 のパラメーター

パラメーター	説明
szComponent	スクリプトに関連付けられているコンポーネントの名前を指定します。
szMessage	データベース サーバーが返したエラー メッセージを指定します。
nvErrorType	エラーの種類を指定します。以下は有効な種類です : · SQL_ERROR_SCRIPT_UNABLE_OPEN_FILE Ñ スクリプト ファイルを開くことができませんでした。 · SQL_ERROR_SCRIPT_CONNECTION_NOT_OPEN Ñ 指定された接続が開いていません。 · SQL_ERROR_GET_SCHEMA_VERSION Ñ ターゲット データベースからスキーマ バージョンを取得中に、エラーが発生しました。 · SQL_ERROR_SET_SCHEMA_VERSION Ñ ターゲット データベースにスキーマ バージョンを設定中に、エラーが発生しました。 · SQL_ERROR_SCRIPT_COMMAND_ERROR Ñ SQL バッチの実行に失敗しました。
nvErrorLine	エラーの行番号を指定します。
szScriptName	スクリプトの名前を指定します。
szTechnology	データベース サーバー製品の名前を指定します (Microsoft SQL Server、Oracle または MySQL)。
szServer	ターゲット データベース サーバーの名前を指定します。
szDB	ターゲット データベース カタログの名前を指定します。

戻り値

テーブル 208 · SQLRTGetComponentScriptError の戻り値

戻り値	説明
TRUE	1 つまたは複数のエラーが発生しました。
FALSE	エラーはありませんでした。

SQLRTGetConnectionAuthentication

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

SQLRTGetConnectionAuthentication 関数は、デフォルトの SQL Server 接続認証タイプを取得します。

***メモ・**OnSQLServerInitialize イベント ハンドラー (InstallScript プロジェクトの場合) または OnSQLLogin イベント ハンドラー (InstallScript MSI プロジェクトの場合) が呼び出される前に SQL 関連のビルトイン関数を呼び出す場合、SQLRTInitialize2 関数を先に呼び出します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

SQLRTGetConnectionAuthentication(szConnection);

パラメーター

テーブル 209・SQLRTGetConnectionAuthentication のパラメーター

パラメーター	説明
szConnection	SQL Server 接続を指定します。

戻り値

テーブル 210・SQLRTGetConnectionAuthentication の戻り値

戻り値	説明
TRUE	Windows 認証を実行します。
FALSE	SQL 認証を実行します。

SQLRTGetConnectionInfo

プロジェクト・この情報は、次のプロジェクトの種類に適用します:

- *InstallScript*
- *InstallScript MSI*

SQLRTGetConnectionInfo 関数は、接続情報 (デフォルト サーバー、データベース、デフォルト ユーザー名、デフォルト パスワード) が含まれた文字列を取得します。

メモ・SQLRTGetConnectionInfo *は* **SQLRT.dll** *(InstallScript プロジェクトの場合)および* **ISSQLSRV.dll** *(InstallScript MSI プロジェクトの場合)を使用します。したがって、この関数は、SQLRTInitialize2 が呼び出されたあとにのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

SQLRTGetConnectionInfo(szConnection, szServer, szDB, szUser, szPassword);

パラメーター

テーブル 211 ・ SQLRTGetConnectionInfo のパラメーター

パラメーター	説明
szConnection	SQL Server 接続を指定します。
szServer	SQL Server を指定します。
szDB	SQL Server データベースを指定します。
szUser	SQL Server ログイン情報を指定します。
szPassword	ユーザー アカウントに関連付けられたパスワードを指定します。

戻り値

テーブル 212 ・ SQLRTGetConnectionInfo の戻り値

戻り値	説明
>=ISERR_SUCCESS	この関数は接続を確立できました。
<ISERR_SUCCESS	この関数は接続の確立に失敗しました。
SQL_ERROR_NOT_INITIALIZED (-10)	**SQLRT.dll** がロードされていません。

SQLRTGetConnections

プロジェクト・*この情報は、次のプロジェクトの種類に適用します:*

- *InstallScript*
- *InstallScript MSI*

SQLRTGetConnections 関数は、設定ファイルに存在する接続の文字列リストを取得します。

メモ・SQLRTGetConnections は **SQLRT.dll** *(InstallScript プロジェクトの場合) および* **ISSQLSRV.dll** *(InstallScript MSI プロジェクトの場合) を使用します。したがって、この関数は、SQLRTInitialize2 が呼び出されたあとにのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

```
SQLRTGetConnections( );
```

パラメーター

SQLRTGetConnections にパラメーターは使用できません。

戻り値

テーブル 213 ・ SQLRTGetConnections の戻り値

戻り値	説明
listConnections;	接続名のリストを戻します。

SQLRTGetDatabases

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SQLRTGetDatabases 関数は、指定されたデータベース サーバーで提供されているデータベース カタログのリストを返します。

メモ・OnSQLServerInitialize イベント ハンドラー (InstallScript プロジェクトの場合) または OnSQLLogin イベント ハンドラー (InstallScript MSI プロジェクトの場合) が呼び出される前に SQL 関連のビルトイン関数を呼び出す場合、SQLRTInitialize2 関数を先に呼び出します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。

構文

```
SQLRTGetDatabases( szConnection, szServer, bTrust, szUserName, szPassword );
```

パラメーター

テーブル 214 ・ SQLRTGetDatabases のパラメーター

パラメーター	説明
szConnection	[SQL スクリプト] ビューで作成した接続の名前を指定します。
szServer	SQL Server を指定します。
bTrust	SQL Server 認証情報ではなく、Windows 認証情報を使用するログインを指定するラジオボタンの初期状態を指定します。これが TRUE の場合、″ パスワード ″ フィールドと ″ ログイン ID″ フィールドは無効になります。FALSE の場合、SQL Server 認識情報が使用されるというメッセージが表示され、パスワードおよびログインフィールドが有効にされます。 一度関数が戻されると、このパラメーターには , エンドユーザーが Windows 認証情報ラジオ ボタンを選択した場合 TRUE が含まれ、SQL Server 認証情報ラジオ ボタンが代わりに選択された場合は FALSE が含まれています。
szUserName	SQL Server ログイン情報を指定します。
szPassword	ユーザー アカウントに関連付けられたパスワードを指定します。

戻り値

データベース サーバーで提供されているデータベース カタログのリストを返します。

SQLRTGetErrorMessage

プロジェクト・*この情報は、InstallScript プロジェクトに適用します。*

SQLRTGetErrorMessage 関数は、接続を開いているときに、SQL ランタイムで最後に発生したエラーについての説明を返します。

メモ・SQLRTGetErrorMessage *は、***SQLRT.dll** *と SQL 設定ファイルからの設定を使用する SQLRTGetLastError2 を呼び出します。したがって、この関数は SQLRTInitialize2 が呼び出された後にのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

```
SQLRTGetErrorMessage( svMessage );
```

パラメーター

テーブル 215 - SQLRTGetErrorMessage のパラメーター

パラメーター	説明
svMessage	SQL エラー メッセージを指定します。

戻り値

テーブル 216 - SQLRTGetErrorMessage の戻り値

戻り値	説明
ISERR_SUCCESS	関数は正常に実行されました。
SQL_ERROR_NOT_INITIALIZED	**SQLRT.dll** がロードされていません。

SQLRTGetLastError

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

SQLRTGetLastError2 関数は **SQLRTGetLastError** 関数に優先します。

SQLRTGetLastError 関数は、SQL ランタイムで最後に発生したエラーのテキストを戻します。**SQLRTGetLastError** はまた、適切な SQL エラー メッセージもロードします。

***メモ - SQLRTGetLastError** は、**SQLRT.dll** と SQL 設定ファイルからの設定を使用します。したがって、この関数は SQLRTInitialize2 が呼び出された後にのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

```
SQLRTGetLastError( szError );
```

パラメーター

テーブル 217 - SQLRTGetLastError のパラメーター

パラメーター	説明
szError	SQL エラーを指定します。

戻り値

なし

SQLRTGetLastError2

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SQLRTGetLastError2 関数は、SQL ランタイムで最後に発生したエラーの詳細情報を返します。**SQLRTGetLastError2** はまた、適切な SQL エラー メッセージもロードします。

***メモ・SQLRTGetLastError2** は、**SQLRT.dll** と SQL 設定ファイルからの設定を使用します。したがって、この関数は SQLRTInitialize2 が呼び出された後にのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

SQLRTGetLastError2(nvErrorCode, svMessage, svTechnology, svConnection, svServer, svDBCatalog);

パラメーター

テーブル 218・SQLRTGetLastError2 のパラメーター

パラメーター	説明
nvErrorCode	SQL エラー コードを指定します。
svMessage	SQL エラー メッセージを指定します。
svTechnology	使用されたサーバーの種類を指定します (Microsoft SQL Server、Oracle など)。
svConnection	SQL エラー コードを指定します。
svServer	SQL データベース サーバーの名前を指定します。
svDBCatalog	SQL データベース カタログの名前を指定します。

戻り値

なし

SQLRTGetScriptErrorMessage

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

SQLRTGetScriptErrorMessage 関数は SQL スクリプトが実行しているときに、SQL ランタイムで最後に発生したエラーについての説明を返します。

***メモ・SQLRTGetScriptErrorMessage** は、SQLRT.dll と SQL 設定ファイルからの設定を使用する SQLRTGetComponentScriptError2 を呼び出します。したがって、この関数は SQLRTInitialize2 が呼び出された後にのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

SQLRTGetScriptErrorMessage(svMessage);

パラメーター

テーブル 219・SQLRTGetScriptErrorMessage のパラメーター

パラメーター	説明
svMessage	SQL エラー メッセージを指定します。

戻り値

テーブル 220・SQLRTGetScriptErrorMessage の戻り値

戻り値	説明
ISERR_SUCCESS	関数は正常に実行されました。
SQL_ERROR_NOT_INITIALIZED	**SQLRT.dll** がロードされていません。

SQLRTGetServers

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します:*

- *InstallScript*
- *InstallScript MSI*

SQLRTGetServers 関数は、インストールに含まれているすべてのデータベース テクノロジについて、ネットワークで提供されているデータベース サーバーのリストを返します。bFilter が TRUE のとき、ローカル データベース サーバーのみを返します。

***メモ・**OnSQLServerInitialize イベント ハンドラー (InstallScript プロジェクトの場合) または OnSQLLogin イベント ハンドラー (InstallScript MSI プロジェクトの場合) が呼び出される前に SQL 関連のビルトイン関数を呼び出す場合、SQLRTInitialize2 関数を先に呼び出します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

SQLRTGetServers(bFilter);

パラメーター

テーブル 221 ・ SQLRTGetServers のパラメーター

パラメーター	説明
bFilter	返されるデータベース サーバーのリストにローカル データベース サーバーのみが含まれるかどうかを指定します。 ・ TRUE Ñ ローカル データベース サーバーのみ返します。 ・ FALSE Ñ すべてのデータベース サーバーを返します。

戻り値

インストールに含まれているすべてのデータベース テクノロジについて、ネットワークで提供されているデータベース サーバーのリストを返します。

SQLRTGetServers2

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

- *InstallScript*
- *InstallScript MSI*

SQLRTGetServers2 関数は、接続に指定されたデータベース テクノロジ用のデータベース サーバーのリストを返します。szConnection が空のとき、この関数は SQLRTGetServers のように動作します。

***メモ・**OnSQLServerInitialize イベント ハンドラー (InstallScript プロジェクトの場合) または OnSQLLogin イベント ハンドラー (InstallScript MSI プロジェクトの場合) が呼び出される前に SQL 関連のビルトイン関数を呼び出す場合、SQLRTInitialize2 関数を先に呼び出します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

```
SQLRTGetServers2( szConnection, bFilter );
```

パラメーター

テーブル 222 ・ SQLRTGetServers2 のパラメーター

パラメーター	説明
szConnection	接続を指定します。このパラメーターが空のとき、関数は、インストールに含まれているすべてのデータベース テクノロジについて、ネットワークで提供されているデータベース サーバーのリストを返します。
bFilter	返されるデータベース サーバーのリストにローカル データベース サーバーのみが含まれるかどうかを指定します。 ・ TRUE Ñ ローカル データベース サーバーのみ返します。 ・ FALSE Ñ すべてのデータベース サーバーを返します。

戻り値

接続に指定されたデータベース テクノロジについて、ネットワークで提供されているデータベース サーバーのリストを返します。

SQLRTInitialize

***プロジェクト・**この情報は、InstallScript プロジェクトに適用します。*

SQLRTInitialize2 関数は **SQLRTInitialize** 関数に優先します。

SQLRTInitialize が使用される場合、SQLRT で呼び出される最初の関数として指定する必要があります。**SQLRTInitialize** は **SQLRT.dll** をロードし、設定ファイルを使用してこれを初期化します。

構文

```
SQLRTInitialize( szSettingsFile );
```

パラメーター

テーブル 223 ・ SQLRTInitialize のパラメーター

パラメーター	説明
szSettingsFile	**SQLRT.ini** ファイルへのパスを指定します。次のパスがこのパラメーターに指定されています : **SUPPORTDIR ^ "SQLRT.ini"** **SQLRT.ini** ファイルは変更できません。これを変更すると、実行時に予期しない動作が発生する可能性があります。このファイルは、[SQL スクリプト] ビューで入力した SQL の設定に基づいて作成されます。

戻り値

なし

SQLRTInitialize2

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

- *InstallScript*
- *InstallScript MSI*

SQLRTInitialize2 は、SQL ランタイムで呼び出される最初の関数として指定する必要があります。**SQLRTInitialize2** は InstallScript プロジェクトでは **SQLRT.dll** ファイルをロードし、InstallScript MSI プロジェクトでは **ISSQLSRV.dll** ファイルをロードします。また SQLRTInitialize2 は、設定ファイルを使用して .dll ファイルを初期化します。

***メモ ・** OnSQLServerInitialize イベント ハンドラー (InstallScript プロジェクトの場合) または OnSQLLogin イベント ハンドラー (InstallScript MSI プロジェクトの場合) が呼び出される前に SQL 関連のビルトイン関数を呼び出す場合、SQLRTInitialize2 関数を先に呼び出します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

```
SQLRTInitialize2 ();
```

パラメーター

SQLRTInitialize2 にパラメーターは使用できません。

戻り値

なし

SQLRTPutConnectionAuthentication

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

The **SQLRTPutConnectionAuthentication** 関数は、デフォルトの SQL Server 接続認証タイプを設定します。

メモ・OnSQLServerInitialize イベント ハンドラー (InstallScript プロジェクトの場合) または OnSQLLogin イベント ハンドラー (InstallScript MSI プロジェクトの場合) が呼び出される前に SQL 関連のビルトイン関数を呼び出す場合、SQLRTInitialize2 関数を先に呼び出します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。

構文

SQLRTPutConnectionAuthentication(szConnection, bWinLogin);

パラメーター

テーブル 224・SQLRTPutConnectionAuthentication のパラメーター

パラメーター	説明
szConnection	[SQL スクリプト] ビューで作成した接続の名前を指定します。
bWinLogin	SQL Server 認証情報ではなく、Windows 認証情報を使用するログインを指定するラジオボタンの初期状態を指定します。これが TRUE の場合、″ パスワード ″ フィールドと ″ ログイン ID″ フィールドは無効になります。FALSE の場合、SQL Server 認識情報が使用されるというメッセージが表示され、パスワードおよびログインフィールドが有効にされます。 一度関数が戻されると、このパラメーターには , エンドユーザーが Windows 認証情報ラジオ ボタンを選択した場合 TRUE が含まれ、SQL Server 認証情報ラジオ ボタンが代わりに選択された場合は FALSE が含まれています。

戻り値

なし

SQLRTPutConnectionInfo

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SQLRTPutConnectionInfo2 関数は **SQLRTPutConnectionInfo** 関数に優先します。

SQLRTPutConnectionInfo 関数は、接続情報 (デフォルト サーバー、デフォルト ユーザー名、デフォルト パスワード) を設定します。これは、[戻る] ボタンのように、エンドユーザーが以前に入力した情報を再び再現する必要がある場合に便利です。

メモ・SQLRTPutConnectionInfo *は* **SQLRT.dll** *(InstallScript プロジェクトの場合) および* **ISSQLSRV.dll** *(InstallScript MSI プロジェクトの場合) を使用します。したがって、この関数は、SQLRTInitialize2 が呼び出されたあとにのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

SQLRTPutConnectionInfo(szConnection, szServer, szUser, szPassword);

パラメーター

テーブル 225 ・ SQLRTPutConnectionInfo のパラメーター

パラメーター	説明
szConnection	SQL Server 接続を指定します。
szServer	SQL Server を指定します。
szUser	SQL Server ログイン情報を指定します。
szPassword	ユーザー アカウントに関連付けられたパスワードを指定します。

戻り値

テーブル 226 ・ SQLRTPutConnectionInfo の戻り値

戻り値	説明
>=ISERR_SUCCESS	この関数は接続を確立できました。
<ISERR_SUCCESS	この関数は接続の確立に失敗しました。
SQL_ERROR_NOT_INITIALIZED (-10)	**SQLRT.dll** がロードされていません。

SQLRTPutConnectionInfo2

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SQLRTPutConnectionInfo2 関数は、接続情報 (デフォルト サーバー、デフォルト データベース カタログ、デフォルト ユーザー名、デフォルト パスワード) を設定します。これは、[戻る] ボタンのように、エンドユーザーが以前に入力した情報を再び再現する必要がある場合に便利です。

***メモ・SQLRTPutConnectionInfo2** は* **SQLRT.dll** *(InstallScript プロジェクトの場合) および* **ISSQLSRV.dll** *(InstallScript MSI プロジェクトの場合) を使用します。したがって、この関数は、SQLRTInitialize2 が呼び出されたあとにのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

SQLRTPutConnectionInfo2(szConnection, szServer, svDBCatalog, szUser, szPassword);

パラメーター

テーブル 227 · SQLRTPutConnectionInfo2 のパラメーター

パラメーター	説明
szConnection	SQL Server 接続を指定します。
szServer	SQL Server を指定します。
svDBCatalog	SQL データベース カタログの名前を指定します。
szUser	SQL Server ログイン情報を指定します。
szPassword	ユーザー アカウントに関連付けられたパスワードを指定します。

戻り値

テーブル 228 · SQLRTPutConnectionInfo2 の戻り値

戻り値	説明
>=ISERR_SUCCESS	この関数は接続を確立できました。
<ISERR_SUCCESS	この関数は接続の確立に失敗しました。
SQL_ERROR_NOT_INITIALIZED (-10)	**SQLRT.dll** がロードされていません。

SQLRTServerValidate

プロジェクト・この情報は、InstallScript MSI プロジェクトに適用します。

SQLRTServerValidate 関数はインストールで指定された接続をテストします。

メモ・OnSQLServerInitialize イベント ハンドラー (InstallScript プロジェクトの場合) または OnSQLLogin イベント ハンドラー (InstallScript MSI プロジェクトの場合) が呼び出される前に SQL 関連のビルトイン関数を呼び出す場合、SQLRTInitialize2 関数を先に呼び出します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。

構文

SQLRTServerValidate¥(hInstall);

パラメーター

テーブル 229・SQLRTServerValidate のパラメーター

パラメーター	説明
hInstall	インストールへのハンドルを指定します。 **IS_SQLSERVER_CONNECTIONS_TO_VALIDATE** が指定されているとき、Windows Installer プロパティの値として指定されている接続のみテストされます。

戻り値

テーブル 230・SQLRTServerValidate の戻り値

戻り値	説明
=ISERR_SUCCESS	この関数は接続を確立できました。
SQL_ERROR_NOT_INITIALIZED	**ISSQLSRV.dll** がロードされていません。

追加情報

結果は、**IS_SQLSERVER_STATUS** プロパティの値を設定するために使用されます。次のテーブルは、使用可能な値の一覧です :

テーブル 231・IS_SQLSERVER_STATUS プロパティの値

戻り値	説明
0	関数は 1 つまたは複数の接続を確立することができました。
1	関数は接続の確立に失敗しました。
2	関数は接続を確立できましたが、ターゲット データベース サーバーから製品バージョンを取得できませんでした。
3	関数は接続を確立できましたが、バージョン要件が満たされませんでした。

テーブル 231 ・ IS_SQLSERVER_STATUS プロパティの値 (続き)

戻り値	説明
4	関数は接続を確立できましたが、MSDE または Express バージョンは許可されていませんでした。
5	関数は接続を確立できましたが、ターゲット データベース カタログからスキーマ バージョンを取得できませんでした。
6	関数は接続を確立できましたが、ターゲット データベース カタログにスキーマ バージョンを設定できませんでした。
7	関数は指定された ODBC ドライバーを見つけることができませんでした。
8	関数はターゲット データベース カタログへの接続を確立できましたが、([SQL スクリプト] ビューにある **[存在しない時、カタログを作成する]** チェック ボックスが選択されているとき) ターゲット データベース カタログを作成できませんでした。
9	関数はデフォルト データベース カタログへの接続を確立できましたが、ターゲット データベース カタログへの接続に失敗しました。
10	関数は **ISSQLDBMetaData** テーブルで有効なレコードを見つけることができませんでした。

SQLRTSetBrowseOption

プロジェクト・*この情報は、InstallScript プロジェクトに適用します。*

SQLRTSetBrowseOption 関数を利用して、SQL Server の参照コンボ ボックスとリスト ボックス コントロールで、ローカル サーバー、リモート サーバー、サーバーのエイリアス、またはこれらのサーバーの組み合わせを表示するかどうかを指定できます。

メモ・*OnSQLServerInitialize イベント ハンドラー (InstallScript プロジェクトの場合) または OnSQLLogin イベント ハンドラー (InstallScript MSI プロジェクトの場合) が呼び出される前に SQL 関連のビルトイン関数を呼び出す場合、SQLRTInitialize2 関数を先に呼び出します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

```
SQLRTSetBrowseOption( nBrowseOption );
```

パラメーター

テーブル 232 ・ SQLRTSetBrowseOption のパラメーター

パラメーター	説明
nBrowseOption	このパラメーターには、次の定義済み定数のいずれかを指定します : ・ SQL_BROWSE_ALL (0) Ñ すべての使用可能な SQL Server を表示します。この定数の指定は、SQL_BROWSE_LOCAL、SQL_BROWSE_REMOTE、および SQL_BROWSE_ALIAS 定数の選択と同じ意味を持ちます。 ・ SQL_BROWSE_LOCAL (1) Ñ ローカルの SQL Server を表示します。 ・ SQL_BROWSE_REMOTE (2) Ñ リモート SQL Server を表示します。 ・ SQL_BROWSE_ALIAS (4) Ñ SQL Server のエイリアスを表示します。 定数とビット単位 OR 演算子 (|) を組み合わせて複数の参照オプションを指定することができます。

戻り値

なし

SQLRTTestConnection

***プロジェクト・**この情報は、InstallScript MSI プロジェクトに適用します。*

SQLRTTestConnection2 関数は **SQLRTTestConnection** 関数に優先します。

SQLRTTestConnection 関数は、指定された認証情報を使ってインストールで指定された接続をすべてテストします。

***メモ・**OnSQLServerInitialize イベント ハンドラー (InstallScript プロジェクトの場合) または OnSQLLogin イベント ハンドラー (InstallScript MSI プロジェクトの場合) が呼び出される前に SQL 関連のビルトイン関数を呼び出す場合、SQLRTInitialize2 関数を先に呼び出します。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

```
SQLRTTestConnection( szServer, szDB, szUserName, szPassword, bTrust );
```

パラメーター

テーブル 233 · SQLRTTestConnection のパラメーター

パラメーター	説明
szServer	SQL Server を指定します。
szDB	カタログを指定します。
szUserName	SQL Server ログイン情報を指定します。
szPassword	ユーザー アカウントに関連付けられたパスワードを指定します。
bTrust	SQL Server 認証情報ではなく、Windows 認証情報を使用するログインを指定するラジオボタンの初期状態を指定します。これが TRUE の場合、″パスワード″フィールドと″ログイン ID″フィールドは無効になります。FALSE の場合、SQL Server 認識情報が使用されるというメッセージが表示され、パスワードおよびログインフィールドが有効にされます。 一度関数が戻されると、このパラメーターには , エンドユーザーが Windows 認証情報ラジオ ボタンを選択した場合 TRUE が含まれ、SQL Server 認証情報ラジオ ボタンが代わりに選択された場合は FALSE が含まれています。

戻り値

テーブル 234 · SQLRTTestConnection の戻り値

戻り値	説明
=ISERR_SUCCESS	この関数は接続を確立できました。
SQL_ERROR_NOT_INITIALIZED	**ISSQLSRV.dll** がロードされていません。

追加情報

結果は、**IS_SQLSERVER_STATUS** プロパティの値を設定するために使用されます。次のテーブルは、使用可能な値の一覧です :

テーブル 235 · IS_SQLSERVER_STATUS プロパティの値

戻り値	説明
0	関数は 1 つまたは複数の接続を確立することができました。
1	関数は接続の確立に失敗しました。
2	関数は接続を確立できましたが、ターゲット データベース サーバーから製品バージョンを取得できませんでした。
3	関数は接続を確立できましたが、バージョン要件が満たされませんでした。

テーブル 235 ・ IS_SQLSERVER_STATUS プロパティの値 (続き)

戻り値	説明
4	関数は接続を確立できましたが、MSDE または Express バージョンは許可されていませんでした。
5	関数は接続を確立できましたが、ターゲット データベース カタログからスキーマ バージョンを取得できませんでした。
6	関数は接続を確立できましたが、ターゲット データベース カタログにスキーマ バージョンを設定できませんでした。
7	関数は指定された ODBC ドライバーを見つけることができませんでした。
8	関数はターゲット データベース カタログへの接続を確立できましたが、([SQL スクリプト] ビューにある **[存在しない時、カタログを作成する]** チェック ボックスが選択されているとき) ターゲット データベース カタログを作成できませんでした。
9	関数はデフォルト データベース カタログへの接続を確立できましたが、ターゲット データベース カタログへの接続に失敗しました。
10	関数は **ISSQLDBMetaData** テーブルで有効なレコードを見つけることができませんでした。

SQLRTTestConnection2

プロジェクト・*この情報は、InstallScript MSI プロジェクトに適用します。*

SQLRTTestConnection2 関数は接続を確立します。

メモ・SQLRTTestConnection2 *は* **ISSQLSRV.dll** *を使用します。したがって、この関数は、SQLRTInitialize2 が呼び出されたあとにのみ呼び出すことができます。詳しくは、「InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する」を参照してください。*

構文

SQLRTTestConnection2(szConnection, szServer, szDB, szUserName, szPassword, bTrust);

パラメーター

テーブル 236 · SQLRTTestConnection2 のパラメーター

パラメーター	説明
szConnection	SQL Server 接続を指定します。
szServer	SQL Server を指定します。
svDB	カタログを指定します。
szUser	SQL Server ログイン情報を指定します。
szPassword	ユーザー アカウントに関連付けられたパスワードを指定します。
bTrust	SQL Server 認証情報ではなく、Windows 認証情報を使用するログインを指定するラジオボタンの初期状態を指定します。これが TRUE の場合、″パスワード″フィールドと″ログイン ID″フィールドは無効になります。FALSE の場合、SQL Server 認識情報が使用されるというメッセージが表示され、パスワードおよびログインフィールドが有効にされます。 一度関数が戻されると、このパラメーターには , エンドユーザーが Windows 認証情報ラジオ ボタンを選択した場合 TRUE が含まれ、SQL Server 認証情報ラジオ ボタンが代わりに選択された場合は FALSE が含まれています。

戻り値

テーブル 237 · SQLRTTestConnection2 の戻り値

戻り値	説明
=ISERR_SUCCESS	この関数は接続を確立できました。
SQL_ERROR_NOT_INITIALIZED	**ISSQLSRV.dll** がロードされていません。

追加情報

結果は、**IS_SQLSERVER_STATUS** プロパティの値を設定するために使用されます。次のテーブルは、使用可能な値の一覧です :

テーブル 238 · IS_SQLSERVER_STATUS プロパティの値

戻り値	説明
0	関数は 1 つまたは複数の接続を確立することができました。
1	関数は接続の確立に失敗しました。
2	関数は接続を確立できましたが、ターゲット データベース サーバーから製品バージョンを取得できませんでした。

テーブル 238 ・ IS_SQLSERVER_STATUS プロパティの値 (続き)

戻り値	説明
3	関数は接続を確立できましたが、バージョン要件が満たされませんでした。
4	関数は接続を確立できましたが、MSDE または Express バージョンは許可されていませんでした。
5	関数は接続を確立できましたが、ターゲット データベース カタログからスキーマ バージョンを取得できませんでした。
6	関数は接続を確立できましたが、ターゲット データベース カタログにスキーマ バージョンを設定できませんでした。
7	関数は指定された ODBC ドライバーを見つけることができませんでした。
8	関数はターゲット データベース カタログへの接続を確立できましたが、([SQL スクリプト] ビューにある **[存在しない時、カタログを作成する]** チェック ボックスが選択されているとき) ターゲット データベース カタログを作成できませんでした。
9	関数はデフォルト データベース カタログへの接続を確立できましたが、ターゲット データベース カタログへの接続に失敗しました。
10	関数は **ISSQLDBMetaData** テーブルで有効なレコードを見つけることができませんでした。

SQLServerLogin

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

- *InstallScript*
- *InstallScript MSI*

SQLServerLogin 関数は SQL ログイン認証情報を指定するスクリプトで使われるダイアログを作成します。情報には、ログイン ID とパスワードが含まれます。

この関数は、InstallScript プロジェクトでは **SQLRT.obl** ファイルに、また InstallScript MSI プロジェクトでは **SQLCONV.obl** ファイルにあります。InstallShield の [SQL スクリプト] ビューを使用する場合、適切なファイルは自動的にリンカの設定に追加されます。このビューを使用していない場合は、適切なファイルを次のようにリンカの設定に追加します : [ビルド] メニューで、[設定] をクリックして、関数をライブラリ (.obl) ボックスに追加します。

構文

```
SQLServerLogin( szMsg, svUser, svPassword );
```

パラメーター

テーブル 239 ・ SQLServerLogin のパラメーター

パラメーター	説明
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
svUser	" ログイン ID" 編集ボックスに表示するデフォルトの文字列を指定します。一度ダイアログ関数が戻されると、このパラメーターにはエンドユーザーが編集ボックスで入力したログイン ID が含まれています。この情報はエンドユーザーが Windows 認証情報の代わりに SQL ログインの使用を選択したときのみ適切です。
svPassword	パスワード 編集フィールドに表示するデフォルトの文字列を指定します。一度ダイアログ関数が戻されると、このパラメーターにはエンドユーザーが編集フィールドで入力したパスワードが含まれています。この情報はエンドユーザーが Windows 認証情報の代わりに SQL ログインの使用を選択したときのみ適切です。

戻り値

テーブル 240 ・ SQLServerLogin の戻り値

戻り値	説明
NEXT	エンドユーザーが、[次へ] ボタンを選択しました。
BACK	エンドユーザーが、[戻る] ボタンを選択しました。
< ISERR_SUCCESS	ダイアログを表示することができませんでした。

SQLServerSelect

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

- *InstallScript*
- *InstallScript MSI*

SQLServerSelect 関数は、エンド ユーザーがデータベース サーバーを選択できるダイアログを作成します。

この関数は、InstallScript プロジェクトでは **SQLRT.obl** ファイルに、また InstallScript MSI プロジェクトでは **SQLCONV.obl** ファイルにあります。InstallShield の [SQL スクリプト] ビューを使用する場合、適切なファイルは自動的にリンカの設定に追加されます。このビューを使用していない場合は、適切なファイルを次のようにリンカの設定に追加します : [ビルド] メニューで、[設定] をクリックして、関数をライブラリ (.obl) ボックスに追加します。

構文

```
SQLServerSelect( szMsg, svServer );
```

パラメーター

テーブル 241 ・ SQLServerSelect のパラメーター

パラメーター	説明
szMsg	ダイアログに表示するメッセージを指定します。このダイアログにデフォルトの指示を表示するには、このパラメーターでヌル文字列 ("") を渡します。
szServer	Server コンボボックスで初回に表示するデフォルトサーバーを指定します。ダイアログ関数が戻されると、このパラメーターにはエンドユーザーが Server コンボ ボックスで選択または入力したサーバー名が含まれています。

戻り値

テーブル 242 ・ SQLServerSelect の戻り値

戻り値	説明
NEXT	エンドユーザーが、[次へ] ボタンを選択しました。
BACK	エンドユーザーが、[戻る] ボタンを選択しました。
< ISERR_SUCCESS	ダイアログを表示することができませんでした。

SQLServerSelectLogin

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

SQLServerSelectLogin2 関数は **SQLServerSelectLogin** 関数に優先します。

SQLServerSelectLogin 関数はデフォルト スクリプトで使用されるダイアログを作成します。ターゲット エンドユーザーは、それを利用して、今ある接続に使用する SQL Server およびそのログイン情報を指定することができます。ダイアログが表示されると、コンボボックスは DSN を通してアクセスされた SQL Server のリストを表示します。参照ボタンを利用して、エンドユーザーは、ネットワーク上で提供されている SQL Server のリストを表示することもできますし、コンボ ボックスにサーバー名を直接入力することもできます。エンドユーザーはまた、オプションで Windows ログイン認証情報を使用したり、SQL Server ログイン ID とパスワードを入力したりすることもできます。

この関数は、InstallScript プロジェクトでは **SQLRT.obl** ファイルに、また InstallScript MSI プロジェクトでは **SQLCONV.obl** ファイルにあります。InstallShield の [SQL スクリプト] ビューを使用する場合、適切なファイルは自動的にリンカの設定に追加されます。このビューを使用していない場合は、適切なファイルを次のようにリンカの設定に追加します : [ビルド] メニューで、[設定] をクリックして、関数をライブラリ (.obl) ボックスに追加します。

構文

SQLServerSelectLogin (byref string svServer, byref string svUser, byref string svPassword, byref BOOL bvWindowsLogin);

パラメーター

テーブル 243 · SQLServerSelectLogin のパラメーター

パラメーター	説明
szServer	Server コンボボックスで初回に表示するデフォルトサーバーを指定します。ダイアログ関数が戻されると、このパラメーターにはエンドユーザーが [サーバー] コンボ ボックスで選択または入力したサーバー名が含まれています。 このパラメーターは SQL Server データベース接続でエイリアス名をサポートします。
svUser	″ ログイン ID″ 編集ボックスに表示するデフォルトの文字列を指定します。一度ダイアログ関数が戻されると、このパラメーターにはエンドユーザーが編集ボックスで入力したログイン ID が含まれています。この情報はエンドユーザーが Windows 認証情報の代わりに SQL ログインの使用を選択したときのみ適切です。
svPassword	パスワード 編集フィールドに表示するデフォルトの文字列を指定します。一度ダイアログ関数が戻されると、このパラメーターにはエンドユーザーが編集フィールドで入力したパスワードが含まれています。この情報はエンドユーザーが Windows 認証情報の代わりに SQL ログインの使用を選択したときのみ適切です。
bvWindowsLogin	SQL Server 認証情報ではなく、Windows 認証情報を使用するログインを指定するラジオボタンの初期状態を指定します。これが TRUE の場合、″ パスワード ″ フィールドと ″ ログイン ID″ フィールドは無効になります。FALSE の場合、SQL Server 認識情報が使用されるというメッセージが表示され、パスワードおよびログイン フィールドが有効にされます。 一度関数が戻されると、このパラメーターには , エンドユーザーが Windows 認証情報ラジオ ボタンを選択した場合 TRUE が含まれ、SQL Server 認証情報ラジオ ボタンが代わりに選択された場合は FALSE が含まれています。

戻り値

テーブル 244 · SQLServerSelectLogin の戻り値

戻り値	説明
NEXT	エンドユーザーが、[次へ] ボタンを選択しました。
BACK	エンドユーザーが、[戻る] ボタンを選択しました。
< ISERR_SUCCESS	ダイアログを表示することができませんでした。

SQLServerSelectLogin2

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

- *InstallScript*
- *InstallScript MSI*

SQLServerSelectLogin2 関数はデフォルト スクリプトで使用されるログイン ダイアログを作成します。このダイアログで、ターゲットされたエンド ユーザーは、現在の接続に使用する SQL Server と使用するログインの認証情報を指定できます。ダイアログでは、DSN を通してアクセスされた SQL Server のリストを含むコンボ ボックスが表示されます。エンドユーザーは、このコンボ ボックスでサーバー名を入力することもできますし、[サーバー名] コンボ ボックスの隣にある [参照] ボタンをクリックすることもできます。このボタンをクリックすると、インターネットで提供されている SQL Servers がすべて表示されます。エンドユーザーは、オプションで Windows ログイン認証情報を使用したり、SQL Server ログイン ID とパスワードを入力したりすることもできます。

bShowCxnName が TRUE に設定されている場合、ダイアログで接続情報に関連付けられている接続名が表示されます。また bShowDBCatalog が TRUE に設定されている場合、エンドユーザーは現在の接続に使用するデータベース カタログを指定することができます。エンドユーザーは編集ボックスでカタログ名を入力することもできますし、[データベース カタログ名] 編集ボックスの隣にある [参照] ボタンをクリックすることもできます。[参照] ボタンをクリックすると、指定されたデータベース サーバーで提供されているすべてのデータベース カタログのリストが表示されます。

構文

```
SQLServerSelectLogin2(szConnection, svServer, svUser, svPassword, bvWindowsLogin, svCatalog, bShowCxnName, bShowDBCatalog);
```

パラメーター

テーブル 245 ・ SQLServerSelectLogin2 のパラメーター

パラメーター	説明
szConnection	SQL Server 接続を指定します。
szServer	Server コンボボックスで初回に表示するデフォルトサーバーを指定します。ダイアログ関数が戻されると、このパラメーターにはエンドユーザーが [サーバー] コンボ ボックスで選択または入力したサーバー名が含まれています。 このパラメーターは SQL Server データベース接続でエイリアス名をサポートします。
svUser	″ ログイン ID″ 編集ボックスに表示するデフォルトの文字列を指定します。一度ダイアログ関数が戻されると、このパラメーターにはエンドユーザーが編集ボックスで入力したログイン ID が含まれています。この情報はエンドユーザーが Windows 認証情報の代わりに SQL ログインの使用を選択したときのみ適切です。
svPassword	パスワード 編集フィールドに表示するデフォルトの文字列を指定します。一度ダイアログ関数が戻されると、このパラメーターにはエンドユーザーが編集フィールドで入力したパスワードが含まれています。この情報はエンドユーザーが Windows 認証情報の代わりに SQL ログインの使用を選択したときのみ適切です。
bvWindowsLogin	SQL Server 認証情報ではなく、Windows 認証情報を使用するログインを指定するラジオボタンの初期状態を指定します。これが TRUE の場合、″ パスワード ″ フィールドと ″ ログイン ID″ フィールドは無効になります。FALSE の場合、SQL Server 認識情報が使用されるというメッセージが表示され、パスワードおよびログイン フィールドが有効にされます。 一度関数が戻されると、このパラメーターには , エンドユーザーが Windows 認証情報ラジオ ボタンを選択した場合 TRUE が含まれ、SQL Server 認証情報ラジオ ボタンが代わりに選択された場合は FALSE が含まれています。
svCatalog	データベース カタログの名前を指定します。
bShowCxnName	ダイアログが接続情報に関連付けられている接続名を表示するかどうかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ TRUE Ñ 接続の名前を表示します。 ・ FALSE Ñ 接続の名前を表示しません。

テーブル 245 · SQLServerSelectLogin2 のパラメーター (続き)

パラメーター	説明
bShowDBCatalog	エンドユーザーが現在の接続に使用するデータベース カタログを選択できるかどうかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 · TRUE Ñ エンド ユーザーが現在の接続に使用するデータベース カタログを選択するのを許可します。エンドユーザーは編集ボックスでカタログ名を入力することもできますし、[データベース カタログ名] 編集ボックスの隣にある [参照] ボタンをクリックすることもできます。[参照] ボタンをクリックすると、指定されたデータベース サーバーで提供されているすべてのデータベース カタログのリストが表示されます。 · FALSE Ñ エンドユーザーが現在の接続に使用するデータベース カタログを選択するのを許可しません。

戻り値

テーブル 246 · SQLServerSelectLogin2 の戻り値

戻り値	説明
NEXT	エンドユーザーが、[次へ] ボタンを選択しました。
BACK	エンドユーザーが、[戻る] ボタンを選択しました。
< ISERR_SUCCESS	ダイアログを表示することができませんでした。

SQLServerSelectLoginEx

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

- *InstallScript*
- *InstallScript MSI*

SQLServerSelectLogin2 関数は **SQLServerSelectLoginEx** 関数に優先します。

SQLServerSelectLoginEx 関数はデフォルト スクリプトで使用されるログイン ダイアログを作成します。このダイアログで、ターゲットされたエンド ユーザーは、現在の接続に使用する SQL Server と使用するログインの認証情報を指定できます。ダイアログには、接続情報に関連付けられた接続名も表示されます。

ダイアログでは、DSN を通してアクセスされた SQL Server のリストを含むコンボ ボックスが表示されます。エンドユーザーは、このコンボ ボックスでサーバー名を入力することもできますし、[サーバー名] コンボ ボックスの隣にある [参照] ボタンをクリックすることもできます。このボタンをクリックすると、インターネットで提供されている SQL Servers がすべて表示されます。エンドユーザーは、オプションで Windows ログイン認証情報を使用したり、SQL Server ログイン ID とパスワードを入力したりすることもできます。

構文

SQLServerSelectLoginEx (byval string szConnection, byref string svServer, byref string svUser,
byref string svPassword, byref BOOL bvWindowsLogin)

パラメーター

テーブル 247 · SQLServerSelectLoginEx のパラメーター

パラメーター	説明
szConnection	SQL Server 接続を指定します。
szServer	Server コンボボックスで初回に表示するデフォルトサーバーを指定します。ダイアログ関数が戻されると、このパラメーターにはエンドユーザーが [サーバー] コンボ ボックスで選択または入力したサーバー名が含まれています。 このパラメーターは SQL Server データベース接続でエイリアス名をサポートします。
svUser	″ ログイン ID″ 編集ボックスに表示するデフォルトの文字列を指定します。一度ダイアログ関数が戻されると、このパラメーターにはエンドユーザーが編集ボックスで入力したログイン ID が含まれています。この情報はエンドユーザーが Windows 認証情報の代わりに SQL ログインの使用を選択したときのみ適切です。
svPassword	パスワード 編集フィールドに表示するデフォルトの文字列を指定します。一度ダイアログ関数が戻されると、このパラメーターにはエンドユーザーが編集フィールドで入力したパスワードが含まれています。この情報はエンドユーザーが Windows 認証情報の代わりに SQL ログインの使用を選択したときのみ適切です。
bvWindowsLogin	SQL Server 認証情報ではなく、Windows 認証情報を使用するログインを指定するラジオボタンの初期状態を指定します。これが TRUE の場合、″ パスワード ″ フィールドと ″ ログイン ID″ フィールドは無効になります。FALSE の場合、SQL Server 認識情報が使用されるというメッセージが表示され、パスワードおよびログイン フィールドが有効にされます。 一度関数が戻されると、このパラメーターには , エンドユーザーが Windows 認証情報ラジオ ボタンを選択した場合 TRUE が含まれ、SQL Server 認証情報ラジオ ボタンが代わりに選択された場合は FALSE が含まれています。

戻り値

テーブル 248 · SQLServerSelectLoginEx の戻り値

戻り値	説明
NEXT	エンドユーザーが、[次へ] ボタンを選択しました。
BACK	エンドユーザーが、[戻る] ボタンを選択しました。
< ISERR_SUCCESS	ダイアログを表示することができませんでした。

StatusUpdate

StatusUpdate 関数は、ファイル転送処理とステータスバーの進行状況インジケーターとの間のリンクを有効、または無効にします。bLink を ON に設定するとリンクが有効になり、nFinalPercent を使って、次のファイル転送の

最後に最終的なパーセンテージを表示するよう指定できます。ファイル転送中は、ステータスバーが現在の値から nFinalPercent で指定された値まで滑らかに更新されます。bLink を OFF に設定するとリンクは無効になり、それ以降、ファイル転送時にステータスバーの進行状況インジケーターは自動的に更新されなくなります。

この関数は、ファイル転送関数によって転送される総バイト数を計算することにより機能します。続いて、どの程度の頻度でプログレスバーを現在の位置から nFinalPercent で指定された最大値まで増加させるかを計算します。

StatusUpdate 関数を VerUpdateFile 関数や VerSearchAndUpdateFile 関数と併用することはできません。これらの関数を呼び出す場合、ステータスバーを無効にするか、または手動で更新する必要があります。

ステータスバーをパーセンテージの初期値に設定するには、StatusUpdate より先に SetStatusWindow を呼び出します。

__ヒント__・ファイル転送中にステータスバーを有効にする場合は、CopyFile または XCopyFile の呼び出しの前に StatusUpdate を呼び出します。FeatureMoveData を呼び出してファイルを転送する場合は、先にパラメーターを ON および 100 に設定して StatusUpdate を呼び出します。これにより、セットアップのファイル転送段階でステータスバーが 100% までスムーズに更新されるよう設定されます。

構文

```
StatusUpdate (bLink, nFinalPercent);
```

パラメーター

テーブル 249 ・ StatusUpdate のパラメーター

パラメーター	説明
bLink	ファイル転送操作とステータスバーの進行状況インジケーターとの間のリンクを有効、または無効にします。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 ・ ON Ñ ステータスバーの進行状況インジケーターがファイル転送操作とリンクされるよう指定します。 ・ OFF Ñ ファイル転送操作とステータスバーの進行状況インジケーターとの間のリンクが無効になるよう指定します。bLink を ON に設定して再び StatusUpdate を呼び出すことでリンクを確立し直すまで、リンクは無効のままです。 ***メモ・ステータスバーは、SetStatusWindow を呼び出すことにより手動で更新できます。***
nFinalPercent	bLink の設定が ON の場合に、次のファイル転送操作の最後にステータスバーの進行状況インジケーターが到達する最終的なパーセンテージの値を指定します。nFinalPercent から渡された値がステータスバーの進行状況インジケーターの現在値よりも小さい場合、進行状況インジケーターは変化しません。bLink の設定が OFF の場合、このパラメーターは無視されます。

戻り値

テーブル 250 ・ StatusUpdate の戻り値

戻り値	説明
0	関数が成功したことを示します。
< 0	関数の実行に失敗したことを示します。

StatusUpdate の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
```

```
＊StatusUpdate 関数のデモンストレーションを行います。
＊
＊このスクリプトはソース ディレクトリからターゲット ディレクトリへすべてのファイルを
＊コピーします。StatusUpdate が呼び出されてファイルのコピーと共に
＊進行状況バーの限界を設定し、進行状況インジケーターを
＊統制します。
＊
＊メモ : このスクリプトを実行する前に、プリプロセッサ定数が参照する
＊　が参照するディレクトリを作成し、SOURCE_DIR へ 2 つ以上のファイルを
＊　作成してください。
＊
¥*---------------------------------------------------------*/

#define SOURCE_DIR "C:¥¥ISExampl¥¥Source"
#define TARGET_DIR "C:¥¥ISExampl¥¥Target";

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_StatusUpdate(HWND);

function ExFn_StatusUpdate(hMSI)
begin

  // 進行状況バーを有効にします。
  Enable(STATUS);

  // 進行状況バーの限界を 99% 完了に設定します。
  StatusUpdate (ON, 99);

  // ファイルをコピーします。
  if (XCopyFile (SOURCE_DIR ^ "*.*", TARGET_DIR ^ "*.*", COMP_NORMAL) < 0 ) then
    MessageBox (" ファイルのコピー中にエラーが発生しました。.",SEVERE);
  endif;

  // メッセージを表示します ; 進行状況バーは変更しません。
  SetStatusWindow (-1, " ファイルコピーが 99% で完了しました。");
  Delay (3);

  // 進行状況バーを 100% に設定してメッセージを表示します。
  SetStatusWindow (100, "StatusUpdate 例が完了しました。終了しています ...");
  Delay (3);

end;
```

StrAddLastSlash

StrAddLastSlash 関数は、パス指定の末尾に円記号がない場合、それを追加します。

構文

```
StrAddLastSlash ( svPath );
```

パラメーター

テーブル 251 ・ StrAddLastSlash のパラメーター

パラメーター	説明
svPath	値がパス指定である文字列を指定すると、行末に円記号付きでパスが戻ります。パスが行末の円記号を含む場合、変更されないまま戻されます。

戻り値

テーブル 252 ・ StrAddLastSlash の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数が行末の円記号を追加したこと、またはパスの行末に円記号を含むことを示します。

StrCompare

StrCompare 関数は 2 つの文字列を比較します。比較は大文字と小文字を区別しません。

構文

```
StrCompare ( szStringA, szStringB );
```

パラメーター

テーブル 253 ・ StrCompare のパラメーター

パラメーター	説明
szStringA	比較する初めの文字列を指定します。
szStringB	比較する 2 番目の文字列を指定します。

戻り値

テーブル 254 ・ StrCompare の戻り値

戻り値	説明
< 0	szStringA の文字列が szStringB の文字列よりも低い値を持っていることを示します。
0	2 つの文字列が等しいことを示します。
> 0	szStringA の文字列が szStringB の文字列よりも大きい値を持っていることを示します。

追加情報

StrCompare 関数は、各文字列の最初の文字、各文字列の 2 番目の文字、と続けて相違する文字を検出するまで、または文字列の終わりに到達するまで続けて 2 つの文字列を比較します。

選択した言語の言語ドライバーはどちらの文字列が大きいか、または文字列が同じかを判断します。言語ドライバーを利用しない場合、Windows は内部関数を利用します。Windows のダブルバイト文字セット (DBCS) バージョンでは、この関数は 2 つの DBCS 文字列を比較することができます。

StrCompare の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* StrCompare 関数のデモンストレーションを行います。
*
* StrCompare が呼び出されて 2 つの文字列を比較します。比較結果が
* メッセージボックスに表示されます。
*
¥*---------------------------------------------------------------*/

#define TITLE_TEXT "StrCompare の例 "
```

```
#define MSG_TEXT   " 比較する 2 つの文字列を入力してください :"
#define FIELD_A    " 文字列 A:"
#define FIELD_B    " 文字列 B:"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_StrCompare(HWND);

function ExFn_StrCompare(hMSI)
   STRING svStringA, svStringB;
   NUMBER nResult;
begin

   // ユーザーから 2 つの文字列を取得します。
   SdShowDlgEdit2 (TITLE_TEXT, MSG_TEXT, FIELD_A, FIELD_B, svStringA, svStringB);

   // 文字列を比較します。
   nResult = StrCompare (svStringA, svStringB);

   // 結果を表示します。
   if (nResult = 0) then
      MessageBox (svStringA + " と " +  svStringB + " は同じです。", INFORMATION);
   elseif (nResult < 0) then
      MessageBox (svStringB + " は " + svStringA + " よりも大きいです ", INFORMATION);
   else
      MessageBox (svStringA + " の値は " + svStringB  + " よりも大きいです ", INFORMATION);
   endif;

end;
```

StrConvertSizeUnit

StrConvertSizeUnit 関数は、指定された InstallScript サイズ単位定数の適切な表示文字列を返します。

構文

```
StrConvertSizeUnit (byval string nUnit);
```

パラメーター

テーブル 255 ・ StrConvertSizeUnit のパラメーター

パラメーター	説明
nUnit	UI 文字列に変換する単位定数を指定します。次の定数がサポートされています : • BYTES • KBYTES • MBYTES • GBYTES • TBYTES

戻り値

テーブル 256 ・ StrConvertSizeUnit の戻り値

戻り値	説明
バイト	BYTES 定数を変換した関数。
KB	KBYTES 定数を変換した関数。
MB	MBYTES 定数を変換した関数。
GB	GBYTES 定数を変換した関数。
TB	TBYTES 定数を変換した関数。
ヌル ("")	関数が nUnit を変換できませんでした。

StreamFileFromBinary

StreamFileFromBinary 関数はバイナリキーとファイルをストリームします。

構文

StreamFileFromBinary (hInstall(HWND), svObjectBinaryKey, szFileName);

パラメーター

テーブル 257 ・ StreamFileFromBinary のパラメーター

パラメーター	説明
hInstall (HWND)	現在実行中のアプリケーションへのハンドルを提供します。
svObjectBinaryKey	バイナリテーブルからストリームアウトするキーを示します。
szFileName	抽出されたファイルへの完全パスを示します。

戻り値

テーブル 258 ・ StreamFileFromBinary の戻り値

戻り値	説明
0	関数がファイルと共にバイナリキーをストリームしました。
-1	関数がファイルと共にバイナリキーをストリームすることができませんでした。

StrFind

StrFind 関数は、パラメーター szFindMe で渡された文字列がパラメーター szString で渡された文字列内で検出されたかどうかを判断します。szFindMe が szString で検出された場合、**StrFind** が szString 内での szFindMe の最初の文字の位置を戻します。(注 : szString の最初の文字位置はゼロです。) この関数は大文字と小文字を区別しません。また szString 内で最初に登場する szFindMe を検出するためのみに利用されます。

StrFind は次を呼び出します :

StrFindEx(szString, szFindMe, 0);

StrFind のパラメーターと戻り値については、「StrFindEx」を参照してください。

StrFind の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
```

```
 * StrFind 関数のデモンストレーションを行います。
 *
 * StrFind が呼び出されて文字列内のサブ文字列を検索します。
 * 成功した場合、StrFind はサブ文字列の位置を戻します。
 *
¥*---------------------------------------------------------------*/

#define TITLE_TEXT "StrFind の例 ";

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_StrFind(HWND);

function ExFn_StrFind(hMSI)
   STRING szString, szFindMe, szTitle, szMsg;
   NUMBER nLocation;
begin

   // StrFind を呼び出すパラメーターをセットアップします。
   szString = " これは、検索される文字列の例です。";
   szFindMe = " 文字列の例 ";

   // szFindMe が指定したサブ文字列を検出します。
   nLocation = StrFind (szString, szFindMe);

   // 検出された場合に、そのテキストの場所を表示します。
   if (nLocation < 0) then
      MessageBox (szFindMe + " は " + szString + " で検出されませんでした。", WARNING);
   else
      szMsg    = "'%s' は ¥n'%s' のバイト %d で始まります ";
      SprintfBox (INFORMATION, szTitle, szMsg, szFindMe, nLocation, szString);
   endif;

end;
```

StrFindEx

StrFind 関数は、パラメーター szFindMe で渡された文字列がパラメーター szString で渡された文字列内で検出されたかどうかを判断します。関数は nStart が指定した場所で検索を開始します。szFindMe が検出された場合、StrFind は szFindMe の最初の文字の szString 内の位置を戻します。この関数は大文字と小文字を区別しません。また szString 内で最初に登場する szFindMe を検出するためのみに利用されます。

構文

```
StrFindEx ( szString, szFindMe, nStart );
```

パラメーター

テーブル 259 • StrFindEx のパラメーター

パラメーター	説明
szString	検索する文字列を指定します。
szFindMe	szString で検索する文字列を指定します。
nStart	szFindMe の検索を開始する文字を識別する szString へのオフセットを指定します。 szString の最初の文字の場所はゼロ (0) です。

戻り値

テーブル 260 • StrFindEx の戻り値

戻り値	説明
X	szString が szFindMe を含む場合、X は szFindMe 内の最初の文字の数値位置です。 szString の最初の文字の場所は 0 (ゼロ) です。
< 0	szString が szFindMe を含まないことを示します。

追加情報

下の例では、StrFind は値 19 を戻します。

```
nStartPos = StrFind("Scripting means having fun","ing",7);
```

ある文字列がもうひとつの文字列を含むか否かを示す TRUE または FALSE 戻り値のみが必要な場合 (つまり、サブ文字列の位置が重要でない場合)、下の様に 文字列検索演算子 (%) を利用します。

```
if (szString % szFindMe) then ...
```

文字列検索演算子 (%) は、if ステートメントで解決されるブール型の式でのみ利用することができます。これを repeat ステートメントや while ステートメントで利用することはできません。

StrGetTokens

StrGetTokens 関数は、szString が指定した文字列の部分文字列 (トークン) を抽出して、listID に指定されているリストに配置します。szString の部分文字列は、szDelimiterSet で指定された 1 文字以上の文字を使って区切る必要があります。

たとえば、"One;Two;Three;Four;Five" を 2 番目のパラメーター、";" を 3 番目のパラメーターとして StrGetTokens を呼び出すと、listID は 5 つの文字列 ("One"、"Two"、"Three"、"Four"、および "Five") と共に返されます。リストの各トークンにアクセスするには、ListGetFirstString や ListGetNextString のようなリスト関数を使用してください。

メモ・szString の最初 (または最後) の文字が szDelimiterSet の文字と一致すると、ヌル文字列 ("") は、最初 (または最後) の要素としてリストに挿入されません。代わりに、最初と 2 番目 (または最後と最後から 2 番目) の区切り文字の間の文字が最初 (または最後) の要素としてリストに挿入されます。

構文

StrGetTokens (listID, szString, szDelimiterSet);

パラメーター

テーブル 261 ・ StrGetTokens のパラメーター

パラメーター	説明
listID	トークンのリストを返します。listID によって識別される文字列リストは、ListCreate への呼び出しによって既に初期化されている必要があります。
szString	解析する文字列を指定します。
szDelimiterSet	1 つ以上の区切り文字を指定します。各区切り文字は 1 文字 (1 バイト) です。このパラメーターにヌル文字列を渡すと、関数はヌル文字 ("") を区切り文字として検索します。これは、GetProfString 関数を使用している場合に便利です。 *メモ・区切り文字として空白文字が指定されている場合、StrGetTokens は連続した空白を 1 つの区切り文字として処理します。*

戻り値

テーブル 262 ・ StrGetTokens の戻り値

戻り値	説明
0	関数が正常に文字列を区別し、指定のリストにトークンを挿入したことを示します。
< 0	関数が文字列を区別できず、指定のリストにトークンを挿入できなかったことを示します。

StrGetTokens の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * StrGetTokens 関数のデモンストレーションを行います。
 *
 * まず、キーの値が初期化ファイルから読み出され
 * ます。戻された値は検索パスです。Then
 * StrGetTokens が呼び出され、検索パスで検出されたパスのリストを
 * ビルドします。最後に、パスが表示されます。
 *
 * このスクリプトを適切に実行するため、初期化 ( テキスト ) ファイルを作成して
 * そこに次の行を追加しなくてはなりません :
 *
 *
 *   [Test Section]
 *    searchpath=C:¥Windows;C:¥Windows¥System;C:¥Windows¥Command
 *
 * それから、定数 EXAMPLE_INI をそのファイルの完全修飾名
 * に設定します。
 *
¥*---------------------------------------------------------------------*/

#define FILE_NAME    "C:¥¥ISExampl.ini"
#define SECTION_NAME " テストセクション "
#define KEY_NAME     "searchpath"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_StrGetTokens(HWND);

function ExFn_StrGetTokens(hMSI)
    LIST   listID;
    STRING svSearchPath;
    STRING szTitle, szMsg;
begin

    // 特定のキーの値を初期化ファイルの指定したセクションから
    // 取得します。
    if GetProfString (FILE_NAME, SECTION_NAME, KEY_NAME, svSearchPath) < 0 then
      // エラーを報告します。
      MessageBox (" "+ FILE_NAME + " から値を読み出すことができませんでした。", INFORMATION);
    else
        // GetProfString が戻した検索パスとなるパスのリストを格納する
        // リストを作成します。
        listID = ListCreate (STRINGLIST);

        // 検索パスからリストへ各パスを取得します。
        if (StrGetTokens (listID, svSearchPath, ";") < 0) then
            // エラーを報告します。
            MessageBox ("StrGetTokens が失敗しました。", SEVERE);
        else
            // リストから読み出した各パスを表示します。
            Sprintf (szMsg, " %s %s でパスが見つかりませんでした ", SECTION_NAME,KEY_NAME);
            SdShowInfoList (" 検索パス ", szMsg, listID);
        endif;
    endif;
```

```
    // メモリからリストを削除します。
    ListDestroy(listID);

end;
```

StrLength

StrLength は、指定の文字列変数で最初のヌル文字までの文字数 (つまり、UTF-16 エンコード文字列のコード ユニット数) を返します。

StrLength および StrLengthChars は、同じ結果を返します。

埋め込まれたヌルを含む文字列の文字数を取得するには、SizeOf を使用してください。

構文

```
StrLength (szString);
```

パラメーター

テーブル 263 · StrLength のパラメーター

パラメーター	説明
szString	サイズを決定する文字列を指定します。

戻り値

テーブル 264 · StrLength の戻り値

戻り値	説明
X	X は文字列内の文字数です。
< 0	関数が文字列内の文字数を判断できなかったことを示します。

StrLength の例

***メモ ·** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* StrLength 関数のデモンストレーションを行います。
*
* StrLength が呼び出され、ユーザーが入力した文字列の
* 文字数を取得します。文字の数は
```

```
    * メッセージボックスに表示されます。
    *
    ¥*---------------------------------------------------------------*/

    #define TITLE_TEXT "StrLength の例 "

    // ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
    #include "Ifx.h"

    export prototype ExFn_StrLength(HWND);

    function ExFn_StrLength(hMSI)
        STRING svString, szTitle, szMsg;
        NUMBER nLength;
    begin

        // エンド ユーザーからテキスト行を取得します。
        AskText (" テキスト行を入力します。", "", svString);

        // 文字列の文字数を取得します。
        nLength = StrLength (svString);

        if (nLength < 0) then
            MessageBox ("StrLength が失敗しました。", SEVERE);
        else
            // 文字列の長さを表示します。
            szMsg    = " 文字列 '%s' の長さは %d 文字です。";
            SprintfBox (INFORMATION, TITLE_TEXT, szMsg, svString, nLength);
        endif;

    end;
```

StrLengthChars

StrLengthChars は、指定の文字列変数で最初のヌル文字までの文字数 (つまり、UTF-16 エンコード文字列のコード ユニット数) を返します。

StrLengthChars および StrLength は、同じ結果を返します。

埋め込まれたヌルを含む文字列の文字数を取得するには、SizeOf を使用してください。

構文

```
StrLengthChars ( szString );
```

パラメーター

テーブル 265 ・ StrLengthChars のパラメーター

パラメーター	説明
szString	サイズを決定する文字列を指定します。

戻り値

テーブル 266 ・ StrLengthChars の戻り値

戻り値	説明
X	X は文字列内の文字数です。
< 0	関数が文字列内の文字数を判断できなかったことを示します。

StrLengthChars の例

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * StrLengthChars 関数のデモンストレーションを行います。
 *
 * StrLengthChars が呼び出され、エンド ユーザーが入力した文字列の
 * 文字数を取得します。文字の数は
 * メッセージボックスに表示されます。
 *
¥*---------------------------------------------------------------*/

#define TITLE_TEXT "StrLengthChars の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_StrLengthChars(HWND);

function ExFn_StrLengthChars(hMSI)
   STRING svString, szTitle, szMsg;
   NUMBER nLength;
begin

   // エンド ユーザーからテキスト行を取得します。
   AskText (" テキスト行を入力します。", "", svString);

   // 文字列の文字数を取得します。
   nLength = StrLengthChars (svString);

   if (nLength < 0) then
      MessageBox ("StrLengthChars が失敗しました。", SEVERE);
   else
      // 文字列のコード単位の数を表示します。
      szMsg  = " 文字列 '%s' のコード単位の長さは %d です。";
```

```
        SprintfBox (INFORMATION, TITLE_TEXT, szMsg, svString, nLength);
    endif;

end;
```

StrPutTokens

StrPutTokens 関数は、listID で指定された文字列リストからリスト項目を抽出して、svString で指定された文字列に配置します。listID から svString に配置されたサブ文字列は、表示される文字列で szSeparator によって区切られます。szSeparator のインスタンスは文字列の最初と最後には *配置されません* 。文字列に追加されるリスト要素の間にのみ配置されます。

たとえば、最初のパラメーターに「One」、「Two」、「Three」、「Four」および「Five」という項目を含み、3 番目のパラメーターに「;」を含むリストと一緒に StrPutTokens を呼び出すと、svString にはその関数が呼び出された後で「One;Two;Three;Four;Five」が含まれます。

bNull の値は、結果として生じる文字列のサブ文字列を区切るのにヌル文字を使用するかどうかを判断します。

構文

```
StrPutTokens( listID, svString, szSeparator, bNull );
```

パラメーター

テーブル 267 · StrPutTokens のパラメーター

パラメーター	説明
listID	処理する文字列リストを指定します。listID によって識別される文字列リストは、ListCreate への呼び出しによって既に初期化されている必要があります。この関数は番号リストと一緒には使用できません。
svString	結果の文字列を指定します。
szSeparator	結果の文字列で各リスト項目を区切るのに使用する文字列を指定します。StrGetTokens 関数と違って、このパラメーターにヌル文字列を渡しても*ヌル区切りの文字列はできません*。区切り文字なしのサブ文字列が順番に配置されるだけです。bNULL を TRUE に設定するとヌル文字で区切られた文字列が作成されます。
bNull	bNull が TRUE に設定されている場合、結果の文字列のサブ文字列を区切るのにヌル文字が使われ、szSeparator は無視されます。この場合、結果の文字列は 2 つのヌル文字で終わります。bNull が FALSE に設定されている場合、結果の文字列のサブ文字列は szSeparator によって区切られます。

戻り値

テーブル 268 · StrPutTokens の戻り値

戻り値	説明
0	指定の文字列リストから文字列が正常に作成されたことを示します。
< 0	指定の文字列リストから指定の文字列が作成できなかったことを示します。

StrRemoveLastSlash

StrRemoveLastSlash 関数は、パス指定から行末の円記号を削除します。StrRemoveLastSlash の目的は有効なパスを作成することなので、ルートディレクトリ指定 "A:¥" または "C:¥" などから円記号を削除することはありません。

構文

```
StrRemoveLastSlash(svPath);
```

パラメーター

テーブル 269 • StrRemoveLastSlash のパラメーター

パラメーター	説明
svPath	値がパス指定である文字列を指定すると、行末の円記号無しにパスが戻ります。パスが行末の円記号を含まない場合、変更されないまま戻されます。

戻り値

テーブル 270 • StrRemoveLastSlash の戻り値

戻り値	説明
0	関数が行末の円記号を削除したこと、またはパスの行末に円記号が含まれないことを示します。
< 0	関数が行末の円記号をを削除できなかったことを示します。

追加情報

StrRemoveLastSlash は、AskPath や ParsePath が戻したパスから行末の円記号を削除するのに便利です。StrRemoveLastSlash の目的は有効なパスを作成することなので、ルートディレクトリ指定 "A:¥" または "C:¥" などから円記号を削除することはありません。有効なパスがドライブ指定に変更されてしまうからです。すべてパスから行末の円記号を削除する必要がある場合は、次の部分スクリプトを参考にして下さい。

```
AskPath("", "", svPath);

if (StrLength(svPath) = 3)
    && (svPath[1] = ":")
    && (svPath[2] = "¥¥") then

    svTempString = svPath;
    StrSub(svPath,svTempString,0,2);
else
   StrRemoveLastSlash(svPath);
endif;
```

StrRemoveLastSlash の例

メモ • *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* StrRemoveLastSlash 関数のデモンストレーションを行います。
*
```

```
＊AskPath ダイアログが表示され、ユーザーに対してパスを指定するよう
＊要求します。そして StrRemoveLastSlash  が呼び出されて
＊AskPath が戻すパスから末尾の円記号を削除します。
＊
¥*---------------------------------------------------------------*/

#define  TITLE_TEXT "StrRemoveLastSlash の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_StrRemoveLastSlash(HWND);

function ExFn_StrRemoveLastSlash(hMSI)
   STRING szDefaultPath, svString, szMsg, szTitle;
begin

   // ユーザーからパスを取得します。AskPath は末尾に
   // 円記号をもつパスを戻します。
   AskPath (" パスを指定します :", INSTALLDIR, svString);

   // パスから末尾の円記号を削除します。
   if (StrRemoveLastSlash (svString) < 0) then
      // エラーを報告します。
      MessageBox ("StrRemoveLastSlash が失敗しました。", SEVERE);
   else
      // 最後の円記号が削除された後にフォルダー名を
      // 表示します。
      MessageBox (" 指定のパス h: " + svString, INFORMATION);
   endif;

end;
```

StrReplace

StrReplace 関数は nStart が指定した場所から svResult を検索し、見つかった szFind のインスタンスすべてを szReplace で置換します。

構文

```
StrReplace ( svResult, szFind, szReplace, nStart );
```

パラメーター

テーブル 271 • StrReplace のパラメーター

パラメーター	説明
svResult	検索する文字列を指定します。修正済み文字列を戻します。
szFind	svResult で検索する文字列を指定します。
szReplace	検出された szFind のインスタンスと置換する文字列を指定します。
nStart	szFind の検索を開始する文字を識別する svResult へのオフセットを指定します。szString の最初の文字の場所はゼロ (0) です。svResult にある szFind のすべてのインスタンスを置換する場合、nStart に 0 を指定します。

戻り値

テーブル 272 • StrReplace の戻り値

戻り値	説明
X	szReplace による szFind の置換合計数。
< ISERR_SUCCESS	関数が失敗したことを示します。szFind が ヌル文字列 ("") の場合、または nStart が svResult の長さよりも大きい場合には、ISERR_SUCCESS よりも小さい値が戻されます。

StrSub

StrSub 関数は、nStart が指定する場所から始まる szString が指定する文字列の一部をコピーします。パラメーター nLength はコピーする文字数を指定します。

構文

StrSub (svSubStr, szString, nStart, nLength);

パラメーター

テーブル 273 ・ StrSub のパラメーター

パラメーター	説明
svSubStr	szString からコピーされたサブ文字列を戻します。
szString	サブ文字列がコピーされる元の文字列を指定します。
nStart	コピーする最初の文字を識別する szString へのオフセットを指定します。szString の最初の文字の場所はゼロ (0) です。nStart で渡される値が szString の長さを超える、または等しい場合、svSubStr にヌル文字列 ("") が戻されます。
nLength	szString からコピーする文字数を指定します。この値が nStart と szString の終わりまでの間にある文字数よりも大きい数を指定した場合、nStart から文字列の終わりまでにあるすべての文字は svSubStr に戻されます。

戻り値

テーブル 274 ・ StrSub の戻り値

戻り値	説明
X	ここで X は svSubStr の文字数と同じです。

StrSub の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* StrSub 関数のデモンストレーションを行います。
*
* このスクリプトはユーザーから日付を取得し、StrSub を呼び出して
* 年を 4 桁で抽出します。最後に、年が
* メッセージボックスに表示されます。
*
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "StrSub の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_StrSub(HWND);
```

```
function ExFn_StrSub(hMSI)
    STRING svYear, svDate, szQuestion, szTitle, szMsg;
    NUMBER nYear;
    BOOL bDateOk;
begin

    // AskText を呼び出すメッセージパラメーターをセットアップします。
    szQuestion = " 日付を MM/DD/YYYY の形式で入力してください :";

    repeat
        // ユーザーから日付を取得します。
        AskText (szQuestion, "09/28/1998", svDate);

        // ユーザーが入力した日付のフォーマットを確認します。
        bDateOk = (StrLength(svDate) = 10) &&
                  (svDate[2] = "/") &&
                  (svDate[5] = "/");

        // 日付の形式が不適切な場合、エラーを報告します。
        if !bDateOk then
            MessageBox(svDate + " は指定された形式ではありません。", WARNING);
        endif;
    until bDateOk ;

    // バイト 6 で始まる 4 桁の年を読み出します。
    if (StrSub (svYear, svDate, 6, 4) < 0) then
        MessageBox ("StrSub が失敗しました。", SEVERE);
    endif;

    // 年フィールドを有効にします。
    if StrToNum(nYear, svYear) = 0 then
        // 編集された文字列を表示します。
        szMsg = " %s 年を指定しました ";
        SprintfBox (INFORMATION, TITLE_TEXT, szMsg, svYear);
    else
        MessageBox(svDate + " は指定された形式ではありません。", WARNING);
    endif;

end;
```

STRTOCHAR

STRTOCHAR 関数は、タイプ CHAR のデータとして szString の最初の文字を戻します。

構文

```
STRTOCHAR ( szString );
```

パラメーター

テーブル 275 • STRTOCHAR のパラメーター

パラメーター	説明
szString	最初の文字が CHAR 型のデータとして戻される文字列を指定します。

戻り値

テーブル 276 • STRTOCHAR の戻り値

戻り値	説明
char	szString の最初の文字は CHAR 型のデータとして表記されます。

追加情報

STRTOCHAR 関数は文字列リテラルを関数へ渡すときに便利です。InstallScript は単一及び二重引用符を文字列区切りとして認識するため、例えば文字を受け取る関数へ 'a' のようなリテラル文字を渡すとコンパイラエラーの原因となります。この問題を回避するため、文字リテラルを STRTOCHAR へ渡し、この関数の呼び出し結果を引数として渡します。例 :

```
Sprintf( szString, "%c", STRTOCHAR('a') );
```

StrToLower

StrToLower 関数は、文字列内のすべての文字を小文字に変換します。この関数は非アルファベット文字には無効です。

構文

```
StrToLower ( svTarget, szSource );
```

パラメーター

テーブル 277 • StrToLower のパラメーター

パラメーター	説明
svTarget	szSource の文字列を、すべての文字を小文字に変換して戻します。
szSource	すべて小文字に変換する文字列を指定します。

戻り値

テーブル 278 • StrToLower の戻り値

戻り値	説明
0	関数が文字列を大文字 / 小文字に変更したことを示します。
< 0	関数が文字列を大文字 / 小文字に変更できなかったことを示します。

StrToLower の例

メモ • *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* StrToUpper 関数と StrToLower 関数のデモンストレーションを行います。
*
* StrToUpper が呼び出され、szSource にある文字を小文字から
* 大文字に変換します。
*
* そして StrToLower が呼び出され、大文字から小文字に
* 変換します。
*
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "StrToUpper & StrToLower"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_StrToLower(HWND);

function ExFn_StrToLower(hMSI)
```

```
    STRING szSource, svTarget, szTitle, szMsg;
    NUMBER nReturn;
begin

    // StrToUpper を呼び出すパラメーターをセットアップします。
    szSource = "aBcDeF";

    // すべての文字を大文字に変換します。
    nReturn = StrToUpper (svTarget, szSource);

    if (nReturn < 0) then
        MessageBox ("StrToUpper が失敗しました。", SEVERE);
    endif;

    szMsg = " 元 : %s¥n¥n 更新済み : %s";

    // 修正済み文字列を表示します。
    SprintfBox (INFORMATION, szTitle, szMsg, szSource, svTarget);

    // StrToLower を呼び出すパラメーターをセットアップします。
    szSource = "ABC123*&?d";

    // すべての文字を小文字に変換します。
    nReturn = StrToLower (svTarget, szSource);

    if (nReturn < 0) then
        MessageBox ("StrToLower が失敗しました。", SEVERE);
    endif;

    // 修正済み文字列を表示します。
    szMsg = " オリジナル :%s¥n¥n 修正済み :%s¥n( 注 -- アルファベット以外の文字は " +
            " 変更されません )";
    SprintfBox (INFORMATION, TITLE_TEXT, szMsg, szSource, svTarget);

end;
```

StrToNum

StrToNum 関数は C 関数 atol() とほぼ同様に文字列を数字に変換します。svString について、文字位置 0 から始まって文字列の終わりに到達するまで、または "0"..."9" に当てはまらない文字を検出するまで調査します。(最初の文字にはプラス記号やマイナス記号も可能です。)

そして次のうちのひとつの処理が行われます :

- 文字列内の文字すべてが "0"..."9" に当てはまる場合、文字列が象徴する数字は nvVar へ割り当てられます。
- 文字列が "0".."9" に当てはまる 1 つまたは複数の文字で始まり、1 つまたは複数の数値以外の文字を含む場合、最初の数値以外の文字の左側にある文字に基づいた番号が nvVar に割り当てられます。例えば、szString が "-123ABC456" の場合、nvResult は -123 となります。
- 文字列の最初の文字が "0".."9" に当てはまらず、プラス記号またはマイナス記号以外の時は関数が失敗します。
- 文字列の最初の文字がプラス記号またはマイナス記号で、2 番目の文字が "0".."9" に当てはまらない場合は関数が失敗します。

構文

StrToNum (nvVar, szString);

パラメーター

テーブル 279 ・ StrToNum のパラメーター

パラメーター	説明
nvVar	szString の文字列から作成された番号を戻します。
szString	数値へ変換する文字列を指定します。文字列が、数値変数に許された値以外の数値に対応していると、この関数は予期しない結果を生み出します。

戻り値

テーブル 280 ・ StrToNum の戻り値

戻り値	説明
0	関数が文字列を数値に変換したことを示します。
< 0	関数が文字列を数値に変換できなかったことを示します。

StrToNum の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* StrToNum 関数のデモンストレーションを行います。
*
* StrToNum が呼び出され、"1222240" を 1222240 へ、
* "1222ABC40" を 1222 へ変換します。
*
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "StrToNum の例 "
#define MSG_TEXT   " 文字列 : %s¥n¥n 番号 : %d"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
```

```
#include "Ifx.h"

    export prototype ExFn_StrToNum(HWND);

function ExFn_StrToNum(hMSI)
   STRING szString, szMsg;
   NUMBER nVar;
begin

   // 数値を含む文字列を数字に変換します。
   szString = "1222240";

   if (StrToNum (nVar, szString) < 0) then
      MessageBox ("StrToNum が失敗しました。", SEVERE);
   else
      SprintfBox (INFORMATION, TITLE_TEXT, MSG_TEXT, szString, nVar);
   endif;

   // 数値以外を含む文字列を数字に変換します。
   szString = "1222ABC40";

   if (StrToNum (nVar, szString) < 0) then
      MessageBox ("StrToNum が失敗しました。", SEVERE);
   else
      SprintfBox (INFORMATION, TITLE_TEXT, MSG_TEXT, szString, nVar);
   endif;

end;
```

StrToNumHex

StrToNumHex 関数は文字列を数値へ変換します。szString について、文字位置 0 から始まって文字列の終わりに到達するまで、または "0"..."9"、"a"..."f"、"A"..."F" に当てはまらない文字を検出するまで調査します。(文字列の最初の 2 文字には "0x" または "0X" が可能です。) そして次のうちのひとつの処理が行われます :

- 文字列内の文字すべてが "0"..."9" に当てはまる場合、文字列が象徴する 16 進数は nvVar へ割り当てられます。
- 文字列が 16 進法に当てはまる単数または複数の文字で始まり、ひとつまたは複数の 16 進法以外の文字を含む場合、最初の 16 進法以外の文字の左側にある文字に基づいた番号が nvVar に割り当てられます。たとえば、szString が "0x1A2GHI456" の場合、nvResult は 418 (0x1A2) となります。
- 文字列の最初の文字が 16 進法に当てはまらない場合、関数は失敗します。
- 文字列の最初の 2 文字が "0x" または "0X" で、3 番目の文字が 16 進法に当てはまらない場合、関数は失敗します。

構文

```
StrToNumHex ( nvVar, szString );
```

パラメーター

テーブル 281 ・ StrToNumHex のパラメーター

パラメーター	説明
nvVar	szString の文字列から作成された番号を戻します。
szString	数値へ変換する文字列を指定します。

戻り値

テーブル 282 ・ StrToNumHex の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数が文字列を数値に変換したことを示します。
< ISERR_SUCCESS	関数が文字列を数値に変換できなかったことを示します。

StrToUpper

StrToUpper 関数は、文字列内のすべての文字を大文字に変換します。この関数は非アルファベット文字には無効です。

構文

```
StrToUpper ( svTarget, szSource );
```

パラメーター

テーブル 283 ・ StrToUpper のパラメーター

パラメーター	説明
svTarget	szSource の文字列を、すべての文字を大文字に変換して戻します。
szSource	すべて大文字に変換する文字列を指定します。

戻り値

テーブル 284 ・ StrToUpper の戻り値

戻り値	説明
0	関数が文字列を大文字 / 小文字に変更したことを示します。
< 0	関数が文字列を大文字 / 小文字に変更できなかったことを示します。

StrToUpper の例

***メモ ・** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* StrToUpper 関数と StrToLower 関数のデモンストレーションを行います。
*
* StrToUpper が呼び出され、szSource にある文字を小文字から
* 大文字に変換します。
*
* そして StrToLower が呼び出され、大文字から小文字に
* 変換します。
*
¥*---------------------------------------------------------------------*/

#define TITLE_TEXT "StrToUpper & StrToLower"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_StrToUpper(HWND);

function ExFn_StrToUpper(hMSI)
```

```
    STRING szSource, svTarget, szTitle, szMsg;
    NUMBER nReturn;
begin

    // StrToUpper を呼び出すパラメーターをセットアップします。
    szSource = "aBcDeF";

    // すべての文字を大文字に変換します。
    nReturn = StrToUpper (svTarget, szSource);

    if (nReturn < 0) then
        MessageBox ("StrToUpper が失敗しました。", SEVERE);
    endif;

    szMsg = " 元 : %s¥n¥n 更新済み : %s";

    // 修正済み文字列を表示します。
    SprintfBox (INFORMATION, szTitle, szMsg, szSource, svTarget);

    // StrToLower を呼び出すパラメーターをセットアップします。
    szSource = "ABC123*&?d";

    // すべての文字を小文字に変換します。
    nReturn = StrToLower (svTarget, szSource);

    if (nReturn < 0) then
        MessageBox ("StrToLower が失敗しました。", SEVERE);
    endif;

    // 修正済み文字列を表示します。
    szMsg = " オリジナル :%s¥n¥n 修正済み :%s¥n( 注 -- アルファベット以外の文字は " +
            " 変更されません )";
    SprintfBox (INFORMATION, TITLE_TEXT, szMsg, szSource, svTarget);

end;
```

StrTrim

StrTrim 関数は、文字列から先頭と行末の空白およびタブを削除します。

構文

```
StrTrim (byref string svString);
```

パラメーター

テーブル 285 ・ StrTrim のパラメーター

パラメーター	説明
svString	切り詰める文字列を指定します。

戻り値

テーブル 286 ・ StrTrim の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数が文字列を正しく切り詰めました。
< ISERR_SUCESS	関数は、文字列の切り詰めに失敗しました。

SuiteFormatString

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

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

***メモ ・** この関数は、アドバンスト UI またはスイート / アドバンスト UI インストールに InstallScript パッケージとして含める可能性がある InstallScript インストールに使用できます。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート / アドバンスト UI プロジェクトに追加する」をご覧ください。*

この関数は、スイート / アドバンスト UI インストールに含まれている InstallScript アクションでも使用できます。詳細については、「スイート / アドバンスト UI インストールに含まれる InstallScript コードを実行するアクションでの作業について」を参照してください。

この関数は、次のシナリオで、エラーを返します :

- *関数が、直接起動された (つまり、アドバンスト UI またはスイート / アドバンスト UI インストールから起動されなかった) InstallScript インストールで呼び出されたとき。*
- *関数が、アドバンスト UI またはスイート / アドバンスト UI インストールに実行可能パッケージとして含められている InstallScript インストールで呼び出されたとき。*

SuiteFormatString 関数は、プロパティ名、環境変数リファレンス、およびその他の特殊な文字列を含む形式化された式を解決します。実行時、アドバンスト UI またはスイート / アドバンスト UI インストールは、これらの式の値を展開します。関数によって、文字列変数に、結果の文字列が格納されます。

構文

SuiteFormatString (string szValue, string svFormattedValue);

パラメーター

テーブル 287 · SuiteFormatString のパラメーター

パラメーター	説明
szValue	プロパティ名、環境変数リファレンス、またはその他の特殊文字列を含む 1 つ以上の形式化された式を使った文字列を指定します。これらの式で使用できる構文については、「アドバンスト UI およびスイート / アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。
svFormattedValue	形式化された文字列または式を含める文字列変数を指定します。実行時、インストールは有効な形式化された式の値を展開します。無効な式は、空白文字列 ("") に解決します。

戻り値

テーブル 288 · SuiteFormatString の戻り値

戻り値	説明
ISERR_SUCCESS	関数が形式化された文字列を正しく解決しました。
ISERR_ISERVICE_NOT_ENABLED	関数は、アドバンスト UI またはスイート / アドバンスト UI インストールから起動された InstallScript パッケージ、またはスイート / アドバンスト UI インストールに含まれる InstallScript アクションから起動された InstallScript パッケージで呼び出されたものではありません。関数は、直接起動された InstallScript インストール、または、アドバンスト UI またはスイート / アドバンスト UI インストールに実行可能パッケージとして含められている InstallScript インストールから呼び出されました。

SuiteFormatString の例

```
//---------------------------------------------------------------------------
//
// InstallScript スクリプト例
//
// 次の関数のデモンストレーションを行います。
//    *SuiteSetProperty
//    *SuiteGetProperty
//    *SuiteResolveString
//    *SuiteFormatString
//    *SuiteLogInfo
//
// このサンプルは、アドバンスト UI またはスイート / アドバンスト UI インストールから
// 起動された InstallScript パッケージで呼び出すことができます。このコードによって、
// 次のラインが、アドバンスト UI またはスイート / アドバンスト UI のデバッグ ログに記録されます :
//  SetGet: MyPropertyValue; Resolved: English (United States); Formatted: fmt MyPropertyValue;
//
//---------------------------------------------------------------------------
```

```
function OnBegin()

STRING szValue;
STRING szResolved;
STRING szFormatted;

begin

   if SUITE_HOSTED then
      SuiteSetProperty ("MyPropertyName", "MyPropertyValue");
      SuiteGetProperty ("MyPropertyName", szValue);
      SuiteResolveString ("IDS_LANGUAGE_1033", szResolved);
      SuiteFormatString ("fmt [MyPropertyName]", szFormatted);
      SuiteLogInfo ("SetGet: %s; 解決済み : %s; フォーマット済み : %s;",
         szValue, szResolved, szFormatted);
   endif;

end;
```

SuiteGetProperty

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

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

***メモ・**この関数は、アドバンスト UI またはスイート / アドバンスト UI インストールに InstallScript パッケージとして含める可能性がある InstallScript インストールに使用できます。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート / アドバンスト UI プロジェクトに追加する」をご覧ください。*

この関数は、スイート / アドバンスト UI インストールに含まれている InstallScript アクションでも使用できます。詳細については、「スイート / アドバンスト UI インストールに含まれる InstallScript コードを実行するアクションでの作業について」を参照してください。

この関数は、次のシナリオで、エラーを返します :

- *関数が、直接起動された (つまり、アドバンスト UI またはスイート / アドバンスト UI インストールから起動されなかった) InstallScript インストールで呼び出されたとき。*
- *関数が、アドバンスト UI またはスイート / アドバンスト UI インストールに実行可能パッケージとして含められている InstallScript インストールで呼び出されたとき。*

SuiteGetProperty 関数は、InstallScript パッケージを実行中のアドバンスト UI またはスイート / アドバンスト UI インストールからのアドバンスト UI またはスイート / アドバンスト UI プロパティの値、または InstallScript アクションを実行中のスイート / アドバンスト UI インストールからの値を取得します。関数によって、文字列変数に、プロパティの値が格納されます。

構文

```
SuiteGetProperty (string szName, string svValue);
```

パラメーター

テーブル 289 · SuiteGetProperty のパラメーター

パラメーター	説明
szName	値を取得するアドバンスト UI またはスイート / アドバンスト UI プロパティの名前を指定します。
svValue	アドバンスト UI またはスイート / アドバンスト UI プロパティの値を含める文字列変数を指定します。

戻り値

テーブル 290 · SuiteGetProperty 戻り値

戻り値	説明
ISERR_SUCCESS	関数によって、プロパティの値が正しく取得されました。
ISERR_ISERVICE_NOT_ENABLED	関数は、アドバンスト UI またはスイート / アドバンスト UI インストールから起動された InstallScript パッケージ、またはスイート / アドバンスト UI インストールに含まれる InstallScript アクションから起動された InstallScript パッケージで呼び出されたものではありません。関数は、直接起動された InstallScript インストール、または、アドバンスト UI またはスイート / アドバンスト UI インストールに実行可能パッケージとして含められている InstallScript インストールから呼び出されました。

SuiteGetProperty の例

```
//---------------------------------------------------------------------------
//
// InstallScript スクリプト例
//
// 次の関数のデモンストレーションを行います。
//    *SuiteSetProperty
//    *SuiteGetProperty
//    *SuiteResolveString
//    *SuiteFormatString
//    *SuiteLogInfo
//
// このサンプルは、アドバンスト UI またはスイート / アドバンスト UI インストールから
// 起動された InstallScript パッケージで呼び出すことができます。このコードによって、
// 次のラインが、アドバンスト UI またはスイート / アドバンスト UI のデバッグ ログに記録されます :
//  SetGet: MyPropertyValue; Resolved: English (United States); Formatted: fmt MyPropertyValue;
//
//---------------------------------------------------------------------------

function OnBegin()

STRING szValue;
STRING szResolved;
STRING szFormatted;
```

```
begin

    if SUITE_HOSTED then
        SuiteSetProperty ("MyPropertyName", "MyPropertyValue");
        SuiteGetProperty ("MyPropertyName", szValue);
        SuiteResolveString ("IDS_LANGUAGE_1033", szResolved);
        SuiteFormatString ("fmt [MyPropertyName]", szFormatted);
        SuiteLogInfo ("SetGet: %s; 解決済み : %s; フォーマット済み : %s;",
            szValue, szResolved, szFormatted);
    endif;

end;
```

SuiteLogInfo

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

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

***メモ・**この関数は、アドバンスト UI またはスイート / アドバンスト UI インストールに InstallScript パッケージとして含める可能性がある InstallScript インストールに使用できます。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート / アドバンスト UI プロジェクトに追加する」をご覧ください。*

この関数は、スイート / アドバンスト UI インストールに含まれている InstallScript アクションでも使用できます。詳細については、「スイート / アドバンスト UI インストールに含まれる InstallScript コードを実行するアクションでの作業について」を参照してください。

この関数は、次のシナリオで、エラーを返します :

- *関数が、直接起動された (つまり、アドバンスト UI またはスイート / アドバンスト UI インストールから起動されなかった) InstallScript インストールで呼び出されたとき。*
- *関数が、アドバンスト UI またはスイート / アドバンスト UI インストールに実行可能パッケージとして含められている InstallScript インストールで呼び出されたとき。*

SuiteLogInfo 関数は、アドバンスト UI またはスイート / アドバンスト UI インストールで実行中の InstallScript パッケージについての情報を、アドバンスト UI またはスイート / アドバンスト UI デバッグ ログに記録します。スイート / アドバンスト UI インストールで実行中の InstallScript アクションについての情報を、アドバンスト UI またはスイート / アドバンスト UI デバッグ ログに記録します。

アドバンスト UI またはスイート / アドバンスト UI デバッグ ログの生成については、「アドバンスト UI またはスイート / アドバンスト UI インストールのトラブルシューティング」をご覧ください。

構文

```
SuiteLogInfo (string szFormat, arglist);
```

パラメーター

テーブル 291 • SuiteLogInfo のパラメーター

パラメーター	説明
szFormat	アドバンスト UI またはスイート / アドバンスト UI インストールのデバッグ ログで表示および書き込まれる標準 printf 書式の文字列を指定します。
arglist	szFormat で、同じ番号の書式指定子に対応する引数を指定します （複数可）。

戻り値

テーブル 292 • SuiteLogInfo の戻り値

戻り値	説明
ISERR_SUCCESS	関数によって、情報が正しくログ記録されました。
ISERR_ISERVICE_NOT_ENABLED	関数は、アドバンスト UI またはスイート / アドバンスト UI インストールから起動された InstallScript パッケージ、またはスイート / アドバンスト UI インストールに含まれる InstallScript アクションから起動された InstallScript パッケージで呼び出されたものではありません。関数は、直接起動された InstallScript インストール、または、アドバンスト UI またはスイート / アドバンスト UI インストールに実行可能パッケージとして含められている InstallScript インストールから呼び出されました。

SuiteLogInfo の例

```
//---------------------------------------------------------------------------
//
// InstallScript スクリプト例
//
// 次の関数のデモンストレーションを行います。
//    *SuiteSetProperty
//    *SuiteGetProperty
//    *SuiteResolveString
//    *SuiteFormatString
//    *SuiteLogInfo
//
// このサンプルは、アドバンスト UI またはスイート / アドバンスト UI インストールから
// 起動された InstallScript パッケージで呼び出すことができます。このコードによって、
// 次のラインが、アドバンスト UI またはスイート / アドバンスト UI のデバッグ ログに記録されます :
//  SetGet: MyPropertyValue; Resolved: English (United States); Formatted: fmt MyPropertyValue;
//
//---------------------------------------------------------------------------

function OnBegin()

STRING szValue;
STRING szResolved;
STRING szFormatted;
```

```
begin

    if SUITE_HOSTED then
        SuiteSetProperty ("MyPropertyName", "MyPropertyValue");
        SuiteGetProperty ("MyPropertyName", szValue);
        SuiteResolveString ("IDS_LANGUAGE_1033", szResolved);
        SuiteFormatString ("fmt [MyPropertyName]", szFormatted);
        SuiteLogInfo ("SetGet: %s; 解決済み : %s; フォーマット済み : %s;",
            szValue, szResolved, szFormatted);
    endif;

end;
```

SuiteReportError

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

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

***メモ・**この関数は、アドバンスト UI またはスイート / アドバンスト UI インストールに InstallScript パッケージとして含める可能性がある InstallScript インストールに使用できます。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート / アドバンスト UI プロジェクトに追加する」をご覧ください。*

この関数は、次のシナリオで、エラーを返します :

- *関数が、直接起動された (つまり、アドバンスト UI またはスイート / アドバンスト UI インストールから起動されなかった) InstallScript インストールで呼び出されたとき。*
- *関数が、アドバンスト UI またはスイート / アドバンスト UI インストールに実行可能パッケージとして含められている InstallScript インストールで呼び出されたとき。*

SuiteReportError 関数は、アドバンスト UI またはスイート / アドバンスト UI のユーザー インターフェイスで、アドバンスト UI またはスイート / アドバンスト UI インストールで InstallScript パッケージの実行中に発生したエラーを通知するメッセージ ボックスを表示します。

構文

```
SuiteReportError (string szMessage, nFlags);
```

パラメーター

テーブル 293 ・ SuiteReportError のパラメーター

パラメーター	説明
szMessage	エンドユーザーに表示するエラー メッセージを指定します。
nFlags	有効な MessageBox フラグを指定します。有効なフラグについては、MSDN の MessageBox Function を参照してください。

戻り値

テーブル 294 ・ SuiteReportError の戻り値

戻り値	説明
ISERR_ISERVICE_NOT_ENABLED	関数が、アドバンスト UI またはスイート / アドバンスト UI インストールから起動された InstallScript パッケージで呼び出されませんでした。関数は、直接起動された InstallScript インストール、または、アドバンスト UI またはスイート / アドバンスト UI インストールに実行可能パッケージとして含められている InstallScript インストールから呼び出されました。
MessageBox の戻り値	使用されている戻り値については、MSDN の MessageBox Function を参照してください。

SuiteResolveString

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

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

***メモ ・** この関数は、アドバンスト UI またはスイート / アドバンスト UI インストールに InstallScript パッケージとして含める可能性がある InstallScript インストールに使用できます。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート / アドバンスト UI プロジェクトに追加する」をご覧ください。*

この関数は、スイート / アドバンスト UI インストールに含まれている InstallScript アクションでも使用できます。詳細については、「スイート / アドバンスト UI インストールに含まれる InstallScript コードを実行するアクションでの作業について」を参照してください。

この関数は、次のシナリオで、エラーを返します :

- *関数が、直接起動された （つまり、アドバンスト UI またはスイート / アドバンスト UI インストールから起動されなかった）InstallScript インストールで呼び出されたとき。*

- *関数が、アドバンスト UI またはスイート / アドバンスト UI インストールに実行可能パッケージとして含められている InstallScript インストールで呼び出されたとき。*

SuiteResolveString 関数は、アドバンスト UI またはスイート / アドバンスト UI の文字列 ID を、アドバンスト UI またはスイート / アドバンスト UI インストールで実行中の InstallScript パッケージ内、またはスイート / アドバンスト UI インストールを実行中の InstallScript アクション内の対応する文字列値で置き換えます。

構文

SuiteResolveString (string szStringID, string svResolvedString);

パラメーター

テーブル 295 - SuiteResolveString のパラメーター

パラメーター	説明
szStringID	置き換えるアドバンスト UI またはスイート / アドバンスト UI の文字列 ID を指定します。
svResolvedString	アドバンスト UI またはスイート / アドバンスト UI 文字列 ID の値を格納する文字列変数を指定します。値は、アドバンスト UI またはスイート / アドバンスト UI インストールで選択された言語に基づきます。不明の文字列 ID は、空の文字列 ("") を返します。

戻り値

テーブル 296 - SuiteResolveString の戻り値

戻り値	説明
ISERR_SUCCESS	関数によって、文字列 ID が、対応する文字列値で正しく置換されました。
ISERR_ISERVICE_NOT_ENABLED	関数は、アドバンスト UI またはスイート / アドバンスト UI インストールから起動された InstallScript パッケージ、またはスイート / アドバンスト UI インストールに含まれる InstallScript アクションから起動された InstallScript パッケージで呼び出されたものではありません。関数は、直接起動された InstallScript インストール、または、アドバンスト UI またはスイート / アドバンスト UI インストールに実行可能パッケージとして含められている InstallScript インストールから呼び出されました。

SuiteResolveString の例

```
//---------------------------------------------------------------------------
//
// InstallScript スクリプト例
//
// 次の関数のデモンストレーションを行います。
//    *SuiteSetProperty
//    *SuiteGetProperty
//    *SuiteResolveString
//    *SuiteFormatString
```

```
//    *SuiteLogInfo
//
// このサンプルは、アドバンスト UI またはスイート / アドバンスト UI インストールから
// 起動された InstallScript パッケージで呼び出すことができます。このコードによって、
// 次のラインが、アドバンスト UI またはスイート / アドバンスト UI のデバッグ ログに記録されます :
//  SetGet: MyPropertyValue; Resolved: English (United States); Formatted: fmt MyPropertyValue;
//
//---------------------------------------------------------------------------

function OnBegin()

STRING szValue;
STRING szResolved;
STRING szFormatted;

begin

    if SUITE_HOSTED then
        SuiteSetProperty ("MyPropertyName", "MyPropertyValue");
        SuiteGetProperty ("MyPropertyName", szValue);
        SuiteResolveString ("IDS_LANGUAGE_1033", szResolved);
        SuiteFormatString ("fmt [MyPropertyName]", szFormatted);
        SuiteLogInfo ("SetGet: %s; 解決済み : %s; フォーマット済み : %s;",
            szValue, szResolved, szFormatted);
    endif;

end;
```

SuiteSetProperty

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

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

***メモ・**この関数は、アドバンスト UI またはスイート / アドバンスト UI インストールに InstallScript パッケージとして含める可能性がある InstallScript インストールに使用できます。詳しい情報は、「InstallScript パッケージをアドバンスト UI またはスイート / アドバンスト UI プロジェクトに追加する」をご覧ください。*

この関数は、スイート / アドバンスト UI インストールに含まれている InstallScript アクションでも使用できます。詳細については、「スイート / アドバンスト UI インストールに含まれる InstallScript コードを実行するアクションでの作業について」を参照してください。

この関数は、次のシナリオで、エラーを返します :

- *関数が、直接起動された (つまり、アドバンスト UI またはスイート / アドバンスト UI インストールから起動されなかった) InstallScript インストールで呼び出されたとき。*
- *関数が、アドバンスト UI またはスイート / アドバンスト UI インストールに実行可能パッケージとして含められている InstallScript インストールで呼び出されたとき。*

SuiteSetProperty 関数は、実行中の InstallScript パッケージを起動したアドバンスト UI またはスイート / アドバンスト UI インストールに含まれるアドバンスト UI またはスイート / アドバンスト UI プロパティの値、または実行中の InstallScript アクションを起動したスイート / アドバンスト UI インストールに含まれるスイート / アドバンスト UI プロパティの値を設定します。

構文

SuiteSetProperty (string szName, string szValue);

パラメーター

テーブル 297 · SuiteSetProperty のパラメーター

パラメーター	説明
szName	値を設定するアドバンスト UI またはスイート / アドバンスト UI プロパティの名前を指定します。
szValue	設定するアドバンスト UI またはスイート / アドバンスト UI プロパティの値を指定します。

戻り値

テーブル 298 · SuiteSetProperty の戻り値

戻り値	説明
ISERR_SUCCESS	関数によって、プロパティの値が正しく設定されました。
ISERR_ISERVICE_NOT_ENABLED	関数は、アドバンスト UI またはスイート / アドバンスト UI インストールから起動された InstallScript パッケージ、またはスイート / アドバンスト UI インストールに含まれる InstallScript アクションから起動された InstallScript パッケージで呼び出されたものではありません。関数は、直接起動された InstallScript インストール、または、アドバンスト UI またはスイート / アドバンスト UI インストールに実行可能パッケージとして含められている InstallScript インストールから呼び出されました。

SuiteSetProperty の例

```
//---------------------------------------------------------------------------
//
// InstallScript スクリプト例
//
// 次の関数のデモンストレーションを行います。
//    *SuiteSetProperty
//    *SuiteGetProperty
//    *SuiteResolveString
//    *SuiteFormatString
//    *SuiteLogInfo
//
// このサンプルは、アドバンスト UI またはスイート / アドバンスト UI インストールから
// 起動された InstallScript パッケージで呼び出すことができます。このコードによって、
```

```
// 次のラインが、アドバンスト UI またはスイート / アドバンスト UI のデバッグ ログに記録されます :
//  SetGet: MyPropertyValue; Resolved: English (United States); Formatted: fmt MyPropertyValue;
//
//--------------------------------------------------------------------------

function OnBegin()

STRING szValue;
STRING szResolved;
STRING szFormatted;

begin

    if SUITE_HOSTED then
        SuiteSetProperty ("MyPropertyName", "MyPropertyValue");
        SuiteGetProperty ("MyPropertyName", szValue);
        SuiteResolveString ("IDS_LANGUAGE_1033", szResolved);
        SuiteFormatString ("fmt [MyPropertyName]", szFormatted);
        SuiteLogInfo ("SetGet: %s; 解決済み : %s; フォーマット済み : %s;",
            szValue, szResolved, szFormatted);
    endif;

end;
```

System

System 関数は InstallShield Professional との下位互換性を保つために説明されています。InstallShield の新しいバージョンでは、RebootDialog と SdFinishReboot の方が Windows の再起動やシステムのリブートには適しています。この 2 つの関数のうち、SdFinishReboot がほとんどの機能を提供します。それぞれの関数の説明を参照し、自分のニーズに合う方をお選びください。

インストールが完了した後で、System 関数を使用して Windows を再起動する、またはシステムをリブートしてください。System は中止されたセットアップの処理を行いません。つまり、インストール済みのファイルを削除しません。しかし、InstallShield はセットアップを実行するためにシステム上に配置した一時ディレクトリと一時ファイルを削除します。System 関数を呼び出した直後にシステムを再起動する場合は、System 関数を呼び出した直後に exit キーワードを使用してください。

一部のシステムは、この関数が呼び出されたときに再スタートしない、またはハング状態になることがあります。多くのセットアップルーチン (MS-DOS などのシステムソフトウェア用インストレーションを含む) は、システムを再スタートする前にユーザーに対して警告メッセージを表示します。警告メッセージは、現在の状況を説明し、コマンドが失敗した場合はユーザーに対して手動によるシステムの再起動を指示します。

メモ・この関数は Windows API ExitWindows を呼び出します。BIOS 型には様々な種類があるので、この関数にはシステムとの BIOS 対話が大きく影響します。

構文

```
System (nOp);
```

パラメーター

テーブル 299 ・ System のパラメーター

パラメーター	説明
nOp	セットアップを終了した後に実行するアクションを指定します。このパラメーターに、あらかじめ定義されている次の定数を渡します。 ・ SYS_BOOTMACHINE Ñ システムを再起動します。

System の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* System 関数のデモンストレーションを行います。
*
* このスクリプトはコンピューターを再起動します。
*
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_System(HWND);

function ExFn_System(hMSI)
begin

   System (SYS_BOOTMACHINE);

end;
```

TextSubGetValue

TextSubGetValue 関数は、svValue から、szTextSub と関連したテキスト置換文字列を取り出します。bGlobalOnly パラメーターは、関数がローカルテキスト置換を検索するかどうかを指定します。

構文

```
TextSubGetValue ( szTextSub, svValue, bGlobalOnly, bResolveEmbedded );
```

パラメーター

テーブル 300 ・ TextSubGetValue のパラメーター

パラメーター	説明
szTextSub	テキスト置換を実行したときに svValue によって置換される文字列を指定します。オプションで、"<MYTEXTSUB>" のように文字列値をかぎ括弧で括ることもできます。文字列値をかぎ括弧で括った場合、テキスト置換を実行したときに自動的に削除される追加のテキストをかぎ括弧の外に含めることができます (例 "additional<MYTEXTSUB>text")。
svValue	テキスト置換を実行したときに szTextSub を置換する文字列を戻します。この文字列値自体に埋め込みテキスト置換を含めることもできます。詳細については、「テキスト置換」を参照してください。
bGlobalOnly	TextSubGetValue でグローバルテキスト置換の中だけを検索する (TRUE) か、グローバルとローカルの両方のテキスト置換を検索する (FALSE) かを指定します。詳細については、「テキスト置換」を参照してください。
bResolveEmbedded	埋め込みテキスト置換を実行するかどうかを指定します。詳細については、「テキスト置換」を参照してください。詳細については、次のコード例とコメント行を参照してください。 TextSubSetValue ("<MYTEXTSUB1>", " 最初のテキストサブ ", FALSE); TextSubSetValue ("<MYTEXTSUB2>", " 2 番目のテキストサブ MYTEXTSUB1", FALSE); TextSubGetValue ("<MYTEXTSUB2>", svValue, FALSE, TRUE); // svValue は "2 番目のテキストサブ、最初のテキストサブ " になります。 TextSubGetValue ("<MYTEXTSUB2>", svValue, FALSE, FALSE); // svValue is "2 番目のテキストサブ <MYTEXTSUB1>"

戻り値

テーブル 301 ・ TextSubGetValue の戻り値

戻り値	説明
ISERR_SUCCESS	TextSubGetValue は正常に関連文字列を取り出しました。
< ISERR_SUCCESS	TextSubGetValue は関連文字列の取り出しに失敗しました。

TextSubGetValue の例

```
/*---------------------------------------------------------------*¥
*
```

```
＊InstallShield スクリプトの例
＊
＊TextSub* 関数、TextSubSetValue、
＊TextSubSubtitute、TextSubGetValue および TextSubParseTextSub のデモンストレーションを行います。
＊
¥*---------------------------------------------------------------*/

function OnBegin()
    string svString, svValue;
begin
    TextSubSetValue ( "<MYTEXTSUB1>", " 最初のテキストサブ ", FALSE );
    svString = " テキスト <MYTEXTSUB1> 文字列 ";
    TextSubSubstitute ( svString, FALSE );
    MessageBox (svString, INFORMATION);
    // svString は " テキスト 最初のテキスト サブ文字列 " になります。

    TextSubSetValue ( "<MYTEXTSUB2>", " 2 番目のテキストサブ MYTEXTSUB1", FALSE );
    TextSubGetValue ( "<MYTEXTSUB2>", svValue, FALSE, TRUE );
    MessageBox( svValue, INFORMATION );
    // svValue は "2 番目のテキストサブ、最初のテキストサブ " になります。
    TextSubGetValue ( "<MYTEXTSUB2>", svValue, FALSE, FALSE );
    MessageBox( svValue, INFORMATION );
    // svValue is "2 番目のテキストサブ <MYTEXTSUB1>"

    svString = " テキスト <MYTEXTSUB1> 文字列 ";
    TextSubParseTextSub ( svString );
    MessageBox (svString, INFORMATION);
    // svString は "MYTEXTSUB1" です。
end;
```

TextSubParseTextSub

TextSubParseTextSub 関数は、svTextSub を検索して、かぎ括弧で括られているサブ文字列 (テキスト置換文字列を示す標準的方法) を見つけます。TextSubParseTextSub でそのようなサブ文字列が見つかると、svTextSub で見つかった最初のサブ文字列をかぎ括弧で括らずに返します。それ以外に TextSubParseTextSub は svTextSub を変更しません。

構文

```
TextSubParseTextSub ( svTextSub );
```

パラメーター

テーブル 302 ・ TextSubParseTextSub のパラメーター

パラメーター	説明
svTextSub	検索する文字列を指定します。

戻り値

テーブル 303 ・ TextSubParseTextSub の戻り値

戻り値	説明
ISERR_SUCCESS	この関数は常に ISERR_SUCCESS を戻します。

TextSubParseTextSub の例

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * TextSub* 関数、TextSubSetValue、
 * TextSubSubtitute、TextSubGetValue および TextSubParseTextSub のデモンストレーションを行います。
 *
¥*---------------------------------------------------------------*/

function OnBegin()
    string svString, svValue;
begin
    TextSubSetValue ( "<MYTEXTSUB1>", " 最初のテキストサブ ", FALSE );
    svString = " テキスト <MYTEXTSUB1> 文字列 ";
    TextSubSubstitute ( svString, FALSE );
    MessageBox (svString, INFORMATION);
    // svString は " テキスト 最初のテキスト サブ文字列 " になります。

    TextSubSetValue ( "<MYTEXTSUB2>", " 2 番目のテキストサブ MYTEXTSUB1", FALSE );
    TextSubGetValue ( "<MYTEXTSUB2>", svValue, FALSE, TRUE );
    MessageBox( svValue, INFORMATION );
    // svValue は "2 番目のテキストサブ、最初のテキストサブ " になります。
    TextSubGetValue ( "<MYTEXTSUB2>", svValue, FALSE, FALSE );
    MessageBox( svValue, INFORMATION );
    // svValue is "2 番目のテキストサブ <MYTEXTSUB1>"

    svString = " テキスト <MYTEXTSUB1> 文字列 ";
    TextSubParseTextSub ( svString );
    MessageBox (svString, INFORMATION);
    // svString は "MYTEXTSUB1" です。
end;
```

TextSubSetValue

TextSubSetValue 関数は、szTextSub および szValue の間のテキスト置換関連を作成します。関連がグローバルかローカルかは、bGlobal パラメーターによって指定されます。

構文

```
TextSubSetValue ( szTextSub, szValue, bGlobal );
```

パラメーター

テーブル 304 ・ TextSubSetValue のパラメーター

パラメーター	説明
szTextSub	テキスト置換を実行したときに szValue によって置換される文字列を指定します。オプションで、"<MYTEXTSUB>" のように文字列値をかぎ括弧で括ることもできます。文字列値をかぎ括弧で括った場合、テキスト置換を実行したときに自動的に削除される追加のテキストをかぎ括弧の外に含めることができます (例 "additional<MYTEXTSUB>text")。 ***メモ・内部で作成された既存のテキスト置換やインストールで使用されているオブジェクトとの競合を回避するために、TextSubSetValue で指定する szTextSub の値にインストールに固有のプレフィックスをお使いになることをお勧めします。***
szValue	テキスト置換を実行したときに szTextSub を置換する文字列を指定します。この文字列値自体に埋め込みテキスト置換を含めることもできます。詳細については、「テキスト置換」を参照してください。
bGlobal	szTextSub と szValue の関連がグローバルかローカルかを指定します。詳細は、テキスト置換を参照してください。

戻り値

テーブル 305 ・ TextSubSetValue の戻り値

戻り値	説明
ISERR_SUCCESS	TextSubSetValue は指定した文字列を正常に関連付けました。
< ISERR_SUCCESS	TextSubSetValue は指定した文字列の関連付けに失敗しました。

TextSubSetValue の例

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * TextSub* 関数、TextSubSetValue、
 * TextSubSubtitute、TextSubGetValue および TextSubParseTextSub のデモンストレーションを行います。
 *
¥*---------------------------------------------------------------*/

function OnBegin()
    string svString, svValue;
```

```
begin
    TextSubSetValue ( "<MYTEXTSUB1>", " 最初のテキストサブ ", FALSE );
    svString = " テキスト <MYTEXTSUB1> 文字列 ";
    TextSubSubstitute ( svString, FALSE );
    MessageBox (svString, INFORMATION);
    // svString は " テキスト 最初のテキスト サブ文字列 " になります。

    TextSubSetValue ( "<MYTEXTSUB2>", " 2 番目のテキストサブ MYTEXTSUB1", FALSE );
    TextSubGetValue ( "<MYTEXTSUB2>", svValue, FALSE, TRUE );
    MessageBox( svValue, INFORMATION );
    // svValue は "2 番目のテキストサブ、最初のテキストサブ " になります。
    TextSubGetValue ( "<MYTEXTSUB2>", svValue, FALSE, FALSE );
    MessageBox( svValue, INFORMATION );
    // svValue is "2 番目のテキストサブ <MYTEXTSUB1>"

    svString = " テキスト <MYTEXTSUB1> 文字列 ";
    TextSubParseTextSub ( svString );
    MessageBox (svString, INFORMATION);
    // svString は "MYTEXTSUB1" です。
end;
```

TextSubSubstitute

TextSubSubtitute 関数は、svString でテキスト置換を実行します。bGlobalOnly パラメーターは、関数がローカルテキスト置換を実行するかどうかを指定します。

構文

```
TextSubSubtitute ( svString, bGlobalOnly );
```

パラメーター

テーブル 306 ・ TextSubSubstitute のパラメーター

パラメーター	説明
svString	テキスト置換を実行する文字列を指定します。
bGlobalOnly	TextSubSubstitute でグローバル テキスト置換 (TRUE) を行うか、グローバル テキスト置換とローカル テキスト置換 (FALSE) を行うかを指定します。詳細は、テキスト置換を参照してください。

戻り値

テーブル 307 ・ TextSubSubstitute の戻り値

戻り値	説明
ISERR_SUCCESS	TextSubSubstitute は正常にテキスト置換を実行しました。
< ISERR_SUCCESS	TextSubSubstitute はテキスト置換の実行に失敗しました。

TextSubSubstitute の例

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * TextSub* 関数、TextSubSetValue、
 * TextSubSubtitute、TextSubGetValue および TextSubParseTextSub のデモンストレーションを行います。
 *
¥*---------------------------------------------------------------*/

function OnBegin()
    string svString, svValue;
begin
    TextSubSetValue ( "<MYTEXTSUB1>", " 最初のテキストサブ ", FALSE );
    svString = " テキスト <MYTEXTSUB1> 文字列 ";
    TextSubSubstitute ( svString, FALSE );
    MessageBox (svString, INFORMATION);
    // svString は " テキスト 最初のテキスト サブ文字列 " になります。

    TextSubSetValue ( "<MYTEXTSUB2>", " 2 番目のテキストサブ MYTEXTSUB1", FALSE );
    TextSubGetValue ( "<MYTEXTSUB2>", svValue, FALSE, TRUE );
    MessageBox( svValue, INFORMATION );
    // svValue は "2 番目のテキストサブ、最初のテキストサブ " になります。
    TextSubGetValue ( "<MYTEXTSUB2>", svValue, FALSE, FALSE );
    MessageBox( svValue, INFORMATION );
    // svValue is "2 番目のテキストサブ <MYTEXTSUB1>"
```

```
    svString = " テキスト <MYTEXTSUB1> 文字列 ";
    TextSubParseTextSub ( svString );
    MessageBox (svString, INFORMATION);
    // svString は "MYTEXTSUB1" です。
end;
```

ビルトイン関数 (U-Z)

カテゴリ別の関数一覧は、「カテゴリー別ビルトイン関数」を参照してください。

UninstallApplication

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

UninstallApplication 関数は szUninstallKey が指定したアンインストールを起動します。

構文

UninstallApplication (szUninstallKey, szAdditionalCmdLine, nOptions);

パラメーター

テーブル 1 ・ UninstallApplication のパラメーター

パラメーター	説明
szUninstallKey	ターゲットレジストリの < ルート キー >¥Software¥Microsoft¥Windows¥CurrentVersion¥Uninstall の下のサブキーの名前を指定します。関数はまずルートキー HKEY_CURRENT_USER の下にあるサブキーを確認し、そしてサブキーがそこで見つからない場合、HKEY_LOCAL_MACHINE の下を確認します。InstallShield Professional 5.53 またはそれ以降で作成されたセットアプでは、これは一般的にアプリケーションの名前です。InstallShield Professional 6.0 またはそれ以降では、これは括弧 ({}) を含むアプリケーションの製品 GUID です。現在のセットアップの製品 GUID を入力しないで下さい。
szAdditionalCmdLine	アンインストールに渡す任意の追加コマンドライン引数を指定します。szUninstallKey の UninstallString 値のデータで既に指定されているコマンドライン引数を指定する必要はありません。
nOptions	追加オプションをしていします。LaunchApplication がサポートする任意のオプションを指定することができます。

戻り値

テーブル 2 ・ UninstallApplication の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がアンインストールを起動したことを示します。
< ISERR_SUCCESS	関数がアンインストールを起動できなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

UnUseDLL

UnUseDLL 関数は、メモリから .dll ファイルをアンロードします。**UnUseDLL** は .dll ファイルのロックカウントを 1 つ減少させます。ロックカウントがゼロと等しくなったとき、InstallShield は .dll ファイルをアンロードします。.dll ファイルが必要でなくなった時にそれをメモリに残してシステムリソースを利用しないよう、**UseDLL** への各呼び出しには対応する **UnUseDLL** への呼び出しが必要です。.dll ファイルをアンロードした後、その .dll ファイルで関数を呼び出すことはできません。

UnUseDLL を使って .dll ファイルを適切にアンロードする前にスクリプトが終了、または中止されたとき、.dll ファイルはメモリにロックされます。再び .dll ファイルへアクセスしたとき、スクリプトが失敗する可能性があります。Windows を再起動して .dll ファイルをメモリから削除しなくてはなりません。

***注意*・User32.dll、Gdi32.dll、および Kernel32.dll** *などの Microsoft Windows システム .dll ファイルは、Windows によって自動的にロード及びアンロードされます。これらの .dll ファイルのロードまたはアンロードのために、**UseDLL** 及び **UnUseDLL** を呼び出さないで下さい。*

構文

UnUseDLL (szDLLName);

パラメーター

テーブル 3・UnUseDLL のパラメーター

パラメーター	説明
szDLLName	.dll ファイルの名前を指定します。このパラメーターにパスを含めないでください。

戻り値

テーブル 4・UnUseDLL の戻り値

戻り値	説明
0	関数が .dll ファイルをロック解除し、メモリからアンロードしたことを示します。
< 0	関数が .dll ファイルのロック解除、またはアンロードができなかったことを示します。

UnUseDLL の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------*¥
*
* InstallShield スクリプトの例
```

```
*
* UseDLL 関数と UnUseDLL 関数のデモンストレーションを行います。
*
* UseDLL が呼び出され、.dll ファイル例をメモリへロードします。A
* そしてこの .dll ファイルの 1 つの関数が呼び出され、インストールでの
* .dll 関数の利用法をデモンストレーションします。最後に、
* UnUseDLL を呼び出し、メモリから .dll ファイル例をアンロードします。
*
* メモ : このスクリプトでは、定数 DLL_FILE が
*     フォーマットが下のプロトタイプ宣言に一致する Test という名前の
*     完全修飾名に設定されている必要があります。その関数のフォーマットは、次のプロトタイプ宣言と
*     一致しなくてはなりません。
*
¥*---------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

#define DLL_FILE SUPPORTDIR ^ "TheDLL.dll"

// TheDLL.dll 内の TheExportedFunction をプロトタイプ化します。
prototype TheDLL.TheExportedFunction (INT, WPOINTER);

export prototype ExFn_UnUseDLL(HWND);

function ExFn_UnUseDLL(hMSI)
    STRING   svString;
    INT      nValue;
    WPOINTER psvString;
    NUMBER   nResult;
    BOOL     bDone;
begin

    // .dll ファイルをメモリにロードします。
    nResult = UseDLL (DLL_FILE);

    if (nResult = 0) then
        MessageBox ("UseDLL が成功しました ¥n¥n.dll ファイルがロード されました。", INFORMATION);
    else
        MessageBox ("UseDLL が失敗しました。¥n¥n .dll ファイルをロードできませんでした。", INFORMATION);
        abort;
    endif;

    // bDone が後に続く while ループを制御します。
    bDone = FALSE;

    //  b Done が FALSE の間ループします。
    while (bDone = FALSE)
        // インストールダイアログで [ 戻る ] ボタンを無効にします。
        Disable(BACKBUTTON);

        // ユーザーから文字列を取得します。
        AskText (" 例の文字列を入力します。", " 文字列例。", svString);

        // TheExportedFunction へ渡す文字列へのポインターを取得します。
        psvString = &svString;

        // TheExportedFunction へ渡す文字列の長さを取得します。
        nValue = StrLength (svString);
```

```
    // TheExportedFunction を呼び出します。
    TheExportedFunction (nValue, psvString);

    // TheExportedFunction がどのように変更したかを確認するため、文字列を表示します。
    SprintfBox (INFORMATION, "UseDLL", "TheExportedFunction() changed the string " +
            ": %s", svString);

    // ユーザーに対して、別の例を試す機会を提供します。
    if (AskYesNo (" 別の例を試しますか ?", YES) = NO) then
      bDone = TRUE;
    endif;
  endwhile;

  // メモリから .dll ファイルを削除します。
  if (UnUseDLL (DLL_FILE) < 0) then
    MessageBox ("UnUseDLL が失敗しました。¥n¥n.dll がメモリに残っています。", SEVERE);
  else
    MessageBox ("UnUseDLL が成功しました。.¥n¥n.dll ファイルがメモリから削除されました。",
          INFORMATION);
  endif;

end;
```

UpdateServiceCheckForUpdates

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この関数は現在使用されていません。この関数を呼び出すと、ISERR_NOT_IMPLEMENTED が返されます。

FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

構文

```
UpdateServiceCheckForUpdates ( szProductCode, bWait );
```

UpdateServiceCreateShortcut

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この関数は現在使用されていません。この関数を呼び出すと、ISERR_NOT_IMPLEMENTED が返されます。

FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

構文

```
UpdateServiceCreateShortcut ( szProductCode, szFolder, szItemName );
```

UpdateServiceEnableUpdateManagerInstall

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この関数は現在使用されていません。この関数を呼び出すと、ISERR_NOT_IMPLEMENTED が返されます。

FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

構文

UpdateServiceEnableUpdateManagerInstall (BOOL bEnable);

UpdateServiceGetAgentTarget

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この関数は現在使用されていません。この関数を呼び出すと、ヌル文字列 ("") が返されます。

FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

構文

UpdateServiceGetAgentTarget ( );

UpdateServiceOnEnabledStateChange

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この関数は現在使用されていません。この関数を呼び出すと、ISERR_NOT_IMPLEMENTED が返されます。

FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

構文

UpdateServiceOnEnabledStateChange ( );

UpdateServiceRegisterProduct

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この関数は現在使用されていません。この関数を呼び出すと、ISERR_NOT_IMPLEMENTED が返されます。

FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

構文

UpdateServiceRegisterProduct (szProductCode, szProductVersion, bRegister, nInterval);

UpdateServiceRegisterProductEx

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この関数は現在使用されていません。この関数を呼び出すと、ISERR_NOT_IMPLEMENTED が返されます。

FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

構文

UpdateServiceRegisterProductEx (BOOL bRegister);

UpdateServiceSetHost

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この関数は現在使用されていません。この関数を呼び出すと、ISERR_NOT_IMPLEMENTED が返されます。

FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

構文

UpdateServiceSetHost (szProductCode, szHostURL);

UpdateServiceSetLanguage

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

この関数は現在使用されていません。この関数を呼び出すと、ISERR_NOT_IMPLEMENTED が返されます。

FlexNet Connect のサポートを InstallScript プロジェクトを追加する方法については、ナレッジベースを参照してください。

構文

UpdateServiceSetLanguage (szProductCode, nLanguageID);

UseDLL

UseDLL 関数は、.dll ファイルをメモリにロードします。.dll ファイルが守りにロードされた後、インストールのスクリプトは .d ll ファイルから関数を呼び出すことができます。szDLLName が指定した .dll ファイルが別の .dll ファイルを必要とする場合、これらの別の .dll ファイルは .dll ファイルがロードすることができるフォルダー内に存在しなくてはなりません。通常これは現在のフォルダーです。これらの .dll ファイルが適切に検出されるよう、**UseDLL** を呼び出す前に **ChangeDirectory** を呼び出して、現在のフォルダーをこれらの .dll ファイルの場所へ変更します。これに失敗すると、.dll ファイルが適切にロードされない可能性があります。

.dll ファイルをメモリにロードするたびに、.dll ファイルのロック カウントが増加します。ロック カウントは .dll ファイルを利用するアプリケーションの数を識別します。.dll ファイルの処理が終わった直後に **UnUseDLL** を呼び出して DLL をアンロードしてください。.dll ファイルの処理が終わったときにそれをアンロードしなかった場合、それを必要とするアプリケーションが無いのにも関わらず .dll ファイルがメモリに残った状態になり、システムリソースの無駄遣いとなります。**UseDLL** への各呼び出しはスクリプトの **UseDLL** への呼び出しと一致しなくてはなりません。

注意*・User32.dll、Gdi32.dll、*および* Kernel32.dll *などの Microsoft Windows システム .dll ファイルは、Windows によって自動的にロード及びアンロードされます。これらの .dll ファイルのロードまたはアンロードのために、*UseDLL *及び* UnUseDLL *を呼び出さないで下さい。

構文

UseDLL (szDLLName);

パラメーター

テーブル 5 · UseDLL のパラメーター

パラメーター	説明
szDLLName	ロードする .dll ファイルの名前を指定します。拡張子を指定しなかった場合、InstallScript エンジン はファイルが .dll または .exe 拡張子を持っているものと見なします。このパラメーターにパスを含むことが推奨されますが、これはオプションです。.dll ファイルへのパスがこのパラメーターで指定されなかった場合、InstallScript エンジンは Windows API 関数 **LoadLibrary** が利用した検索順と同じ順序で .dll ファイルを検索します。検索順に関する詳細については、MSDN に記載されている Windows API 関数の説明を参照してください。 インストールに .dll ファイル を含むには、[サポート ファイル] ビューの [言語非依存] フォルダーの適切なサブフォルダーへ追加します。インストールがターゲット システム上で実行中、SUPPORTDIR で指定されている一時ディレクトリ内の .dll ファイルを使用することができます。その後、.dll ファイルを参照するために次のように .dll ファイル名を SUPPORTDIR に追加することができます : szDLLName = SUPPORTDIR ^ "TheDLL.dll"; UseDLL (szDLLName); .dll ファイルを ([サポート ファイル] ビューの適切なフォルダーへ挿入する方法で) インストールに配置しない場合、代わりにアプリケーションのファイルと共にファイルを配布し、ターゲット システムからそれをロードすることができます。ただし、インストールが参照できるよう .dll ファイルのインストール先を指定しなくてはなりません。また、その .dll ファイルがターゲット システムに転送される前にインストールがそれをロードしないように設定しなくてはなりません。 ***メモ ·** szDLLName パラメーターは URL (Uniform Resource Locators) をサポートしません。*

戻り値

テーブル 6 · UseDLL の戻り値

戻り値	説明
0	関数が .dll ファイルをメモリにロードしたことを示します。

テーブル 6 ・ UseDLL の戻り値 (続き)

戻り値	説明
< 0	関数が .dll ファイルをメモリにロードすることができなかったことを示します。 **UseDLL** が失敗した場合、最も一般的に .dll ファイルが見つからないことが原因です。このエラーが発生したとき、パラメーター szDLLName で指定したパスが正しいことを確認してください。 その他、.dll ファイルを利用する際に発生するエラーの一般的な原因は、.dll ファイルの依存関係 (ロードする .dll ファイルがアクセスした .dll ファイル) に関連します。.dll ファイルがアクセスする .dll ファイルが ロードされなかった、または検出されなかった場合、.dll ファイルの呼び出しは失敗します。このエラーが発生した場合、別の .dll ファイルがシステムに存在すること、そしてアクセスが可能であることを確認してください。 **UnUseDLL** を使って .dll ファイルを適切にアンロードする前にスクリプトが終了、または中止されたとき、.dll ファイルはメモリにロックされます。再び .dll ファイルへアクセスしたとき、スクリプトが失敗する可能性があります。Windows を再起動して .dll ファイルをメモリから削除しなくてはなりません。

UseDLL の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* UseDLL 関数と UnUseDLL 関数のデモンストレーションを行います。
*
* UseDLL が呼び出され、.dll ファイル例をメモリへロードします。A
* そしてこの .dll ファイルの 1 つの関数が呼び出され、インストールでの
* .dll 関数の利用法をデモンストレーションします。最後に、
* UnUseDLL を呼び出し、メモリから .dll ファイル例をアンロードします。
*
* メモ : このスクリプトでは、定数 DLL_FILE が
*     フォーマットが下のプロトタイプ宣言に一致する Test という名前の
*     完全修飾名に設定されている必要があります。その関数のフォーマットは、次のプロトタイプ宣言と
*     一致しなくてはなりません。
*
¥*---------------------------------------------------------------------*/

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

#define DLL_FILE SUPPORTDIR ^ "TheDLL.dll"

// TheDLL.dll 内の TheExportedFunction をプロトタイプ化します。
prototype TheDLL.TheExportedFunction (INT, WPOINTER);

export prototype ExFn_UseDLL(HWND);
```

```
function ExFn_UseDLL(hMSI)
    STRING   svString;
    INT      nValue;
    WPOINTER psvString;
    NUMBER   nResult;
    BOOL     bDone;
begin

    // .dll ファイルをメモリにロードします。
    nResult = UseDLL (DLL_FILE);

    if (nResult = 0) then
        MessageBox ("UseDLL が成功しました ¥n¥n.dll ファイルがロード されました。", INFORMATION);
    else
        MessageBox ("UseDLL が失敗しました。¥n¥n .dll ファイルをロード できませんでした。", INFORMATION);
        abort;
    endif;

    // bDone が後に続く while ループを制御します。
    bDone = FALSE;

    //  b Done が FALSE の間ループします。
    while (bDone = FALSE)
        // インストールダイアログで [ 戻る ] ボタンを無効にします。
        Disable(BACKBUTTON);

        // ユーザーから文字列を取得します。
        AskText (" 例の文字列を入力します。", " 文字列例。", svString);

        // TheExportedFunction へ渡す文字列へのポインターを取得します。
        psvString = &svString;

        // TheExportedFunction へ渡す文字列の長さを取得します。
        nValue = StrLength (svString);

        // TheExportedFunction を呼び出します。
        TheExportedFunction (nValue, psvString);

        // TheExportedFunction がどのように変更したかを確認するため、文字列を表示します。
        SprintfBox (INFORMATION, "UseDLL", "TheExportedFunction() changed the string " +
                ": %s", svString);

        // ユーザーに対して、別の例を試す機会を提供します。
        if (AskYesNo (" 別の例を試しますか ?", YES) = NO) then
            bDone = TRUE;
        endif;
    endwhile;

    // メモリから .dll ファイルを削除します。
    if (UnUseDLL (DLL_FILE) < 0) then
        MessageBox ("UnUseDLL が失敗しました。¥n¥n.dll がメモリに残っています。", SEVERE);
    else
        MessageBox ("UnUseDLL が成功しました。.¥n¥n.dll ファイルがメモリから削除されました。",
                INFORMATION);
    endif;

end;
```

VarInit

VarInit 関数は、VarSave と VarRestore 関数で使用する内部リストを初期化または再初期化します。. この関数を呼び出すと、前回 **VarSave** の呼び出しで保存されているが、後に続く **VarRestore** 関数によってまだ使われていない情報が実質的にクリアされます。

構文

VarInit (nType);

パラメーター

テーブル 7 ・ VarInit のパラメーター

パラメーター	説明
nType	リセットする情報を指定します。このパラメーターに、あらかじめ定義されている以下の定数のうちの 1 つを渡します。 ***プロジェクト・****これらの定数は、InstallScript プロジェクトに適用されます。* ・ VAR_LOGGING– 保存されているアンインストール ログ エントリすべてをリセットします。 ・ VAR_CURRENTDIR– 保存されている現在のディレクトリ エントリすべてをリセットします。 ・ VAR_ALLSUPPORTED– 保存されているすべてのエントリをリセットします。 ・ VAR_HKEYCURRENTROOTKEY– 保存されている現在のルートキーエントリをすべてリセットします。 ・ CURRENTROOTKEY– この定数は現在使用されていません。代わりに、VAR_HKEYCURRENTROOTKEY を使用します。 ***プロジェクト・****この定数は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに適用します :* ・ VAR_SRCTARGETDIR– TARGETDIR (InstallScript インストールの場合)、および INSTALLDIR (基本の MSI および InstallScript MSI インストールの場合)、および SRCDIR の保存されている現在のエントリをすべてリセットします。 ***プロジェクト・****この定数は、基本の MSI および InstallScript MSI プロジェクトに適用します :* ・ SRCINSTALLDIR – この定数は現在使用されていません。代わりに、VAR_SRCTARGETDIR を使用します。

戻り値

テーブル 8 ・ VarInit の戻り値

戻り値	説明
ISERR_SUCCESS	システム変数の現在の値がリセットされたことを示します。

VarRestore

VarRestore 関数はシステム変数 TARGETDIR (InstallScript インストールの場合)、INSTALLDIR (基本の MSI および InstallScript MSI インストールの場合)、SRCDIR、または HKEYCURRENTROOTKEY に、以前に行った **VarSave** 呼び出しで保存された値を再び割り当てます。たとえば、**XCopyFile** への呼び出しの前にソース ディレクトリとターゲット ディレクトリを設定する場合など、これらの変数の値を一時的に変更する必要がある場合は常に **VarSave** を呼び出します。後で **VarRestore** を呼び出してこれらの変数を元の値へ戻します。

構文

```
VarRestore ( nType );
```

パラメーター

テーブル 9 ・ VarRestore のパラメーター

パラメーター	説明
nType	復元するシステム変数を指定します。このパラメーターには、次の定義済み定数のいずれかを指定します。複数の定数を OR 演算子 (||) で区切って渡すこともできます。 ***プロジェクト・これらの定数は、InstallScript プロジェクトに適用されます。*** ・ VAR_LOGGING– 保存されているアンインストール ログ エントリすべてをリセットします。 ・ VAR_CURRENTDIR– 保存されている現在のディレクトリ エントリすべてをリセットします。 ・ VAR_ALLSUPPORTED– 保存されているすべてのエントリをリセットします。 ・ VAR_HKEYCURRENTROOTKEY– 保存されている現在のルートキーエントリをすべてリセットします。 ・ VAR_REGOPTIONS – 保存されている現在のレジストリ オプションをすべてをリセットします。 ・ CURRENTROOTKEY– この定数は現在使用されていません。代わりに、VAR_HKEYCURRENTROOTKEY を使用します。 ***プロジェクト・この定数は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに適用します:*** ・ VAR_SRCTARGETDIR または SRCTARGETDIR–TARGETDIR (InstallScript インストールの場合)、および INSTALLDIR (基本の MSI および InstallScript MSI インストールの場合)、および SRCDIR の保存されている現在のエントリをすべてリセットします。 ***プロジェクト・この定数は、基本の MSI および InstallScript MSI プロジェクトに適用します:*** ・ SRCINSTALLDIR – この定数は現在使用されていません。代わりに、VAR_SRCTARGETDIR を使用します。

戻り値

テーブル 10 ・ VarRestore の戻り値

戻り値	説明
0	以前に保存されたシステム変数の値が復元されたことを示します。
< 0	復元する値が保存場所に無いことを示します。このエラーは、VarSave を先に呼び出さずに VarRestore を呼び出したとき、または VarSave の呼び出し回数よりも VarRestore を多く呼び出した場合に発生します。

追加情報

VarSave を呼び出すたびに、InstallScript エンジン はシステム変数の現在の値を内部スタックに、後入れ先だし方式で、積み重ねます。この方法を利用すると、メモリ内で一連の値をスタックすることができます。その後、**VarRestore** を呼び出して、保存したときと逆の順番でこれらの値をスタックから読み出すことができます。

たとえば、(呼び出しの間に **VarRestore** を呼び出さずに) SRCINSTALLDIR をその引数として **VarSave** を 3 回呼び出した場合、スタック上には SRCDIR と INSTALLDIR が 3 セット存在します。SRCINSTALLDIR を引数にした **VarRestore** への最初の呼び出しは、3 番目の **VarSave** への呼び出しからの値を復元します。**VarRestore** への次の呼び出しでは、**VarSave** への 2 番目の呼び出しから値を復元します。**VarRestore** への 3 番目の呼び出しは、**VarSave** への最初の呼び出しから値を復元します。この時点で、スタックは空となります。この処理を具体的に説明する部分スクリプトを参照するには、VarSave Stack の例をクリックしてください。

VarRestore の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* VarSave 関数と VarRestore 関数のデモンストレーションを行います。
*
* このスクリプトは、システム値 SRCDIR と INSTALLDIR の最初の値を
* 表示するところから始まります。そして VarSave を呼び出して
* これらの値を保存します。次に、SRCDIR と INSTALLDIR へ新規値を
* 割り当て、これらの値を表示します。最後に、
* VarRestore を呼び出してオリジナルの値を復元し、
* それを表示します。
*
¥*---------------------------------------------------------------------*/

#define  NEW_SOURCE_DIR "C:¥¥Source"
#define  NEW_INSTALL_DIR "C:¥¥Target"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"
```

```
    export prototype ExFn_VarRestore(HWND);

function ExFn_VarRestore(hMSI)
begin

   // Display the values of SRCDIR and INSTALLDIR.
   SprintfBox (INFORMATION, " ソースフォルダーとターゲットフォルダーを開始します ",
            " ソース :¥n¥n%s¥n¥n ターゲット :¥n¥n%s ",
            SRCDIR, INSTALLDIR);

   // SRCDIR と INSTALLDIR の現在の値を保存します。
   VarSave (SRCTARGETDIR);

   // SRCDIR と INSTALLDIR に新規値を割り当てます。
   SRCDIR    = NEW_SOURCE_DIR;
   INSTALLDIR = NEW_INSTALL_DIR;

   // Display the values of SRCDIR and INSTALLDIR.
   SprintfBox (INFORMATION, " 新しいソースフォルダーとターゲットフォルダー ",
            " 新規ソース :¥n¥n%s¥n¥n 新規ターゲット :¥n¥n%s",
            SRCDIR, INSTALLDIR);

   // 古い値を復元します。
   VarRestore (SRCTARGETDIR);

   // Display the values of SRCDIR and INSTALLDIR.
   SprintfBox (INFORMATION, " ソースフォルダーとターゲットフォルダーを復元しました ",
            " ソース :¥n¥n%s¥n¥n ターゲット :¥n¥n%s ",
            SRCDIR, INSTALLDIR);

end;
```

VarSave

VarSave 関数は、システム変数 TARGETDIR (InstallScript インストールの場合)、INSTALLDIR (基本の MSI および InstallScript MSI インストールの場合)、SRCDIR、または HKEYCURRENTROOTKEY の現在の値を保存します。これらはその他多くの InstallScript 関数が利用します。たとえば、**XCopyFile** への呼び出しの前にソース ディレクトリとターゲット ディレクトリを設定する場合など、これらの変数の値を一時的に変更する必要がある場合は常に **VarSave** を呼び出します。後で **VarRestore** を呼び出してこれらの変数を元の値へ戻します。

構文

```
VarSave (nType);
```

パラメーター

テーブル 11・VarSave のパラメーター

パラメーター	説明
nType	保存するシステム変数を指定します。このパラメーターには、次の定義済み定数のいずれかを指定します。複数の定数を OR 演算子 (||) で区切って渡すこともできます。 ***プロジェクト・この情報は、InstallScript プロジェクトに適用されます。*** ・ VAR_LOGGING— 保存されているアンインストール ログ エントリすべてをリセットします。 ・ VAR_CURRENTDIR— 保存されている現在のディレクトリ エントリすべてをリセットします。 ・ VAR_ALLSUPPORTED— 保存されているすべてのエントリをリセットします。 ・ VAR_HKEYCURRENTROOTKEY— 保存されている現在のルートキーエントリをすべてリセットします。 ・ VAR_REGOPTIONS— 保存されている現在のレジストリ プションをすべてをリセットします。 ・ CURRENTROOTKEY— この定数は現在使用されていません。代わりに、VAR_HKEYCURRENTROOTKEY を使用します。 ***プロジェクト・この定数は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに適用します:*** ・ VAR_SRCTARGETDIR または SRCTARGETDIR—TARGETDIR (InstallScript インストールの場合)、および INSTALLDIR (基本の MSI および InstallScript MSI インストールの場合)、および SRCDIR の保存されている現在のエントリをすべてリセットします。 ***プロジェクト・この定数は、基本の MSI および InstallScript MSI プロジェクトに適用します:*** ・ SRCINSTALLDIR — この定数は現在使用されていません。代わりに、VAR_SRCTARGETDIR を使用します。

戻り値

テーブル 12 ・ VarSave の戻り値

戻り値	説明
0	システム変数の現在の値が保存されたことを示します。
< 0	内部エラーのため値が保存されなかったことを示します。このエラーはシステムの有効メモリが足りない場合のみ発生します。

追加情報

VarSave を呼び出すたびに、InstallScript エンジン はシステム変数の現在の値を内部スタックに、後入れ先だし方式で、積み重ねます。この方法を利用すると、メモリ内で一連の値をスタックすることができます。その後、**VarRestore** を呼び出して、保存したときと逆の順番でこれらの値をスタックから読み出すことができます。

たとえば、(呼び出しの間に **VarRestore** を呼び出さずに) SRCINSTALLDIR をその引数として **VarSave** を 3 回呼び出した場合、スタック上には SRCDIR と INSTALLDIR が 3 セット存在します。SRCINSTALLDIR を引数にした **VarRestore** への最初の呼び出しは、3 番目の **VarSave** への呼び出しからの値を復元します。**VarRestore** への次の呼び出しでは、**VarSave** への 2 番目の呼び出しから値を復元します。**VarRestore** への 3 番目の呼び出しは、**VarSave** への最初の呼び出しから値を復元します。この時点で、スタックは空となります。この処理を具体的に説明する部分スクリプトを参照するには、VarSave Stack の例をクリックしてください。

VarSave の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* VarSave 関数と VarRestore 関数のデモンストレーションを行います。
*
* このスクリプトは、システム値 SRCDIR と INSTALLDIR の最初の値を
* 表示するところから始まります。そして VarSave を呼び出して
* これらの値を保存します。次に、SRCDIR と INSTALLDIR へ新規値を
* 割り当て、これらの値を表示します。最後に、
* VarRestore を呼び出してオリジナルの値を復元し、
* それを表示します。
*
¥*---------------------------------------------------------------------*/

#define  NEW_SOURCE_DIR "C:¥¥Source"
#define  NEW_INSTALL_DIR "C:¥¥Target"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_VarSave(HWND);
```

```
function ExFn_VarSave(hMSI)
begin

   // Display the values of SRCDIR and INSTALLDIR.
   SprintfBox (INFORMATION, " ソースフォルダーとターゲットフォルダーを開始します ",
            " ソース :¥n¥n%s¥n¥n ターゲット :¥n¥n%s ",
            SRCDIR, INSTALLDIR);

   // SRCDIR と INSTALLDIR の現在の値を保存します。
   VarSave (SRCTARGETDIR);

   // SRCDIR と INSTALLDIR に新規値を割り当てます。
   SRCDIR    = NEW_SOURCE_DIR;
   INSTALLDIR = NEW_INSTALL_DIR;

   // Display the values of SRCDIR and INSTALLDIR.
   SprintfBox (INFORMATION, " 新しいソースフォルダーとターゲットフォルダー ",
            " 新規ソース :¥n¥n%s¥n¥n 新規ターゲット :¥n¥n%s",
            SRCDIR, INSTALLDIR);

   // 古い値を復元します。
   VarRestore (SRCTARGETDIR);

   // Display the values of SRCDIR and INSTALLDIR.
   SprintfBox (INFORMATION, " ソースフォルダーとターゲットフォルダーを復元しました ",
            " ソース :¥n¥n%s¥n¥n ターゲット :¥n¥n%s ",
            SRCDIR, INSTALLDIR);

end;
```

VarSave Stack の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
// この部分スクリプトは SRCDIR と INSTALLDIR に割り当てられた
// 値を保存したり読み出したりするのに、内部スタックと共に
//  VarSave と VarRestore がどのように動作するのかを
// 具体的に説明します。

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_VarSave Stack(HWND);

function ExFn_VarSave Stack(hMSI)
begin

   // SRCDIR と INSTALLDIR の最初の値を保存します。
   VarSave(SRCTARGETDIR);

   // SRCDIR と INSTALLDIR に新規値を割り当てます。
   SRCDIR    = "E:¥¥";
```

```
    INSTALLDIR = "C:¥¥Program Files";

    // を参照してください。を参照してください。

    // これらの 値 ("E:¥¥" と "C:¥¥Program Files") を保存します
    VarSave(SRCTARGETDIR);

    // SRCDIR と INSTALLDIR に新規値を割り当てます。
    SRCDIR    = "A:¥¥";
    INSTALLDIR = "C:¥¥Windows";

    // を参照してください。を参照してください。

    // これらの 値 ("A:¥¥" and "C:¥¥Windows") を保存します
    VarSave(SRCTARGETDIR);

    // を参照してください。を参照してください。

    // VarSave への 3 番目の呼び出しから値を復元します。
    VarRestore (SRCTARGETDIR);

    // SRCDIR is now "A:¥¥
    // INSTALLDIR is now "C:¥¥Windows"

    // を参照してください。を参照してください。

    // VarSave への 2 番目の呼び出しから値を復元します。
    VarRestore (SRCTARGETDIR);

    // SRCDIR は現在 "E:¥¥" です
    // INSTALLDIR は現在 "C:¥¥Program Files" です

    // を参照してください。を参照してください。

    // VarSave への 最初の呼び出しから値を復元します。
    VarRestore (SRCTARGETDIR);

    // SRCDIR と INSTALLDIR は現在最初の値です。

end;
```

VerCompare

VerCompare 関数はバージョン情報を含む 2 つの文字列を比較して、最初の文字列が 2 番目の文字列よりも小さいか、大きいか、または等しいかを返します。

重要 *・szVersionInfo1 および szVersionInfo2 パラメーターを使って渡すバージョンのフォーマットは* ***w.x.y.z*** *の形式です。ここで* ***w***、***x***、***y***、および ***z*** *は数値を表します。4 つのすべてのフィールドが存在しなくてはなりません。そうでない場合、****VerCompare*** *は正しく 2 つのバージョン文字列を比較することができません。*

たとえば、有効な zVersionInfo1 および szVersionInfo2 パラメーターのエントリは次のとおりです :

- *"1.0.0.0"*
- *"10.10.20.10"*

- *"3.21.01.2"*

次のエントリは無効なパラメーターです:

- *"1.20"*
- *"1.12.3"*
- *"2"*

バージョン 文字列の 1 つが 4 フィールド以下である場合、必要に応じて連結文字列演算子 (+) を使用して、少数点と数値 0 を追加してください。

構文

VerCompare (szVersionInfo1, szVersionInfo2, nCompareFlag);

パラメーター

テーブル 13 · VerCompare のパラメーター

パラメーター	説明
szVersionInfo1	最初のバージョン文字列を **w.x.y.z** の形式で指定します。ここで **w**、**x**、**y**、および **z** は数値を表します。
szVersionInfo2	2 番目のバージョン文字列を **w.x.y.z** の形式で指定します。ここで **w**、**x**、**y**、および **z** は数値を表します。
nCompareFlag	定義済み定数 VERSION を指定して、バージョン番号の比較を示します。このパラメーターで他の値を使用することはできません。

戻り値

テーブル 14 · VerCompare の戻り値

戻り値	説明
EQUALS (2)	2 つの文字列の値が等しいことを示します。
LESS_THAN (1)	最初の文字列に 2 番目の文字列より小さい値があることを示します。
GREATER_THAN (0)	最初の文字列に 2 番目の文字列より大きい値があることを示します。
ISERR_GEN_FAILURE (-1)	関数が 2 つの文字列を比較できなかったことを示します。

VerCompare の例

***メモ ·** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * VerFindFileVersion 関数と VerCompare 関数のデモンストレーションを行います。
 *
 * このスクリプトは VerFindFileVersion を呼び出してターゲットファイルを検出し、
 * バージョン情報を読み出します。指定したフォルダーが存在しない
 * 検出されなかった場合、ソースファイルが INSTALLDIR へ
 * コピーされます。検出された場合、VerCompare が呼び出され、
 * ターゲットシステム ( ターゲットファイル ) で検出された EXAMPLE ファイルの
 * バージョン番号と、SRCDIR ( ソースファイル ) にあるファイルのバージョン番号を
 * 比較します。ソースファイルバージョン番号が
 * ターゲットファイルよりも新しい場合、ターゲットファイルが
 * ソースファイルで上書きされます。
 *
¥*---------------------------------------------------------------------*/

#define EXAMPLE    "Stirinfc.dll"
#define SOURCE_VER "2.0.1.0"
#define TITLE      "VerCompare と VerFindFileVersion"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_VerFindFileVersion(HWND);

function ExFn_VerFindFileVersion(hMSI)
    STRING szFileName, svPath, svVersionNumber, szExistingVersion, szUpdateVersion;
    STRING szTitle, szMsg;
    NUMBER nResult, nCompareFlag;
begin

    // アップデートするファイルを設定します。
    szFileName = EXAMPLE;

    // ターゲットシステムの szFileName を検出し、バージョン番号を読み出します。
    nResult = VerFindFileVersion(szFileName, svPath, svVersionNumber);

    if (nResult = FILE_NOT_FOUND) then
        // szFileName が検出されなかった場合、ソースファイルを INSTALLDIR へコピーします。
        szMsg = "%s が見つかりません。%s を %s へコピーしています。";
        SprintfBox (INFORMATION, TITLE, szMsg, szFileName, szFileName,
                INSTALLDIR);

        CopyFile (szFileName, szFileName);
        abort;
    elseif (nResult = FILE_NO_VERSION) then

        // バージョン番号が検出されなかった場合、ソースファイルが svPath へコピーして終了します。
        szMsg = "%s バージョン番号が検出されませんでした。%s を %s へコピーしています。";
        SprintfBox (INFORMATION, TITLE, szMsg, szFileName, szFileName,
                INSTALLDIR);

        CopyFile (szFileName, szFileName);
        abort;
    elseif (nResult < 0) then
        MessageBox ("VerFindFileVersion が失敗しました。", SEVERE);
        abort;
```

```
    endif;

    // 2 つのファイルのバージョンを比較します。ソースバージョン番号が
    // 分かっているものと見なします。
    szExistingVersion = svVersionNumber;
    MessageBox (szExistingVersion, INFORMATION);

    szUpdateVersion = SOURCE_VER;
    MessageBox (szUpdateVersion, INFORMATION);

    nCompareFlag = VERSION;

    nResult = VerCompare (szUpdateVersion, szExistingVersion, nCompareFlag);

    // ソースファイルの方が新しい場合、インストールします。
    if (nResult = GREATER_THAN) then
        szMsg      = "%s が %s ディレクトリにアップデートされました。";
        SprintfBox (INFORMATION, TITLE, szMsg, szFileName, INSTALLDIR);

        CopyFile (szFileName, szFileName);

    // ターゲットファイルの方が新しい場合、インストールしません。
    elseif (nResult = LESS_THAN) then
        szMsg = " アップグレードする必要はありません。%s の最新バージョンは " +
                " インストール済みです。";
        SprintfBox (INFORMATION, TITLE, szMsg, szFileName);

    // ターゲットとソースのバージョンが等しい場合、インストールしません。
    elseif (nResult = EQUALS) then
        MessageBox (" バージョンが同じです。アップデートは不要です。.", INFORMATION);
    endif;

end;
```

VerFindFileVersion

VerFindFileVersion 関数は指定されたファイルを検索し、ファイルのバージョンと場所を読み出します。

VerFindFileVersion は、次の検索アルゴリズムを利用してファイルを検出します (次の順番でフォルダーを検索します):

1. Windows フォルダー
2. Windows システム フォルダ
3. TARGETDIR システム変数 (InstallScript インストール) または INSTALLDIR システム変数 (基本の MSI および InstallScript MSI インストール) が指定したフォルダー
4. PATH 環境変数が指定したフォルダー
5. **Setup.exe** が実行されるフォルダー

Windows システムフォルダーの詳細については、InstallScript システム変数 WINSYSDIR の説明を参照してください。

***メモ・VerFindFileVersion** を使用するとき、InstallScript エンジンによって自動的に設定される値の他に TARGETDIR (InstallScript インストールの場合) または INSTALLDIR (基本の MSI および InstallScript インストールの場合) 値の設定が必要な場合があります。関数は、TARGETDIR または INSTALLDIR フォルダーでファイルを検索するため、**VerFindFileVersion** でファイルを確実に検索するためには、一時的にシステム変数の値をリセットしなくてはならない場合もあります。リセットが必要な場合は、**VarSave** を使用して TARGETDIR または INSTALLDIR の値を保存し、別のフォルダーにそれを一時的に設定します。**VerFindFileVersion** 関数を呼び出した後、**VarRestore** を使用して TARGETDIR または INSTALLDIR をリセットします。*

構文

VerFindFileVersion (szFileName, svPath, svVersionNumber);

パラメーター

テーブル 15 · VerFindFileVersion のパラメーター

パラメーター	説明
szFileName	バージョンを取得するファイルの非完全修飾名を指定します。このパラメーターではドライブ指定またはパスを指定しないで下さい。
svPath	ファイルが存在するフォルダーへの完全パス (ドライブ指定を含む) を戻します。
svVersionNumber	ファイルのバージョン番号を次の形式で戻します : < メジャー バージョン >.< マイナー バージョン > 例えば、svVersionNumber が 2.1.2.0 を戻す場合、メジャーバージョン番号は 2.1 でマイナーバージョン番号は 2.0 です。

戻り値

テーブル 16 · VerFindFileVersion の戻り値

戻り値	説明
0	関数がバージョン情報を戻したことを示します。
FILE_NO_VERSION (-8)	ファイルが検出されましたが、バージョン情報を含まないことを示します。
FILE_NOT_FOUND (-2)	ファイルが検出されなかったことを示します。

VerFindFileVersion の例

***メモ ·** 基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* VerFindFileVersion 関数と VerCompare 関数のデモンストレーションを行います。
*
* このスクリプトは VerFindFileVersion を呼び出してターゲットファイルを検出し、
* バージョン情報を読み出します。指定したフォルダーが存在しない
* 検出されなかった場合、ソースファイルが INSTALLDIR へ
* コピーされます。検出された場合、VerCompare が呼び出され、
* ターゲットシステム ( ターゲットファイル ) で検出された EXAMPLE ファイルの
* バージョン番号と、SRCDIR ( ソースファイル ) にあるファイルのバージョン番号を
* 比較します。ソースファイルバージョン番号が
* ターゲットファイルよりも新しい場合、ターゲットファイルが
* ソースファイルで上書きされます。
```

```
 *
¥*---------------------------------------------------------------------*/

#define EXAMPLE    "Stirinfc.dll"
#define SOURCE_VER "2.0.1.0"
#define TITLE      "VerCompare と VerFindFileVersion"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

      export prototype ExFn_VerFindFileVersion(HWND);

function ExFn_VerFindFileVersion(hMSI)
    STRING szFileName, svPath, svVersionNumber, szExistingVersion, szUpdateVersion;
    STRING szTitle, szMsg;
    NUMBER nResult, nCompareFlag;
begin

    // アップデートするファイルを設定します。
    szFileName = EXAMPLE;

    // ターゲットシステムの szFileName を検出し、バージョン番号を読み出します。
    nResult = VerFindFileVersion(szFileName, svPath, svVersionNumber);

    if (nResult = FILE_NOT_FOUND) then
        // szFileName が検出されなかった場合、ソースファイルを INSTALLDIR へコピーします。
        szMsg = "%s が見つかりません。%s を %s へコピーしています。";
        SprintfBox (INFORMATION, TITLE, szMsg, szFileName, szFileName,
                    INSTALLDIR);

        CopyFile (szFileName, szFileName);
        abort;
    elseif (nResult = FILE_NO_VERSION) then

        // バージョン番号が検出されなかった場合、ソースファイルが svPath へコピーして終了します。
        szMsg = "%s バージョン番号が検出されませんでした。%s を %s へコピーしています。";
        SprintfBox (INFORMATION, TITLE, szMsg, szFileName, szFileName,
                    INSTALLDIR);

        CopyFile (szFileName, szFileName);
        abort;
    elseif (nResult < 0) then
        MessageBox ("VerFindFileVersion が失敗しました。", SEVERE);
        abort;
    endif;

    // 2 つのファイルのバージョンを比較します。ソースバージョン番号が
    // 分かっているものと見なします。
    szExistingVersion = svVersionNumber;
    MessageBox (szExistingVersion, INFORMATION);

    szUpdateVersion = SOURCE_VER;
    MessageBox (szUpdateVersion, INFORMATION);

    nCompareFlag = VERSION;

    nResult = VerCompare (szUpdateVersion, szExistingVersion, nCompareFlag);

    // ソースファイルの方が新しい場合、インストールします。
```

```
    if (nResult = GREATER_THAN) then
        szMsg      = "%s が %s ディレクトリにアップデートされました。";
        SprintfBox (INFORMATION, TITLE, szMsg, szFileName, INSTALLDIR);

        CopyFile (szFileName, szFileName);

    // ターゲットファイルの方が新しい場合、インストールしません。
    elseif (nResult = LESS_THAN) then
        szMsg = " アップグレードする必要はありません。%s の最新バージョンは " +
                " インストール済みです。";
        SprintfBox (INFORMATION, TITLE, szMsg, szFileName);

    // ターゲットとソースのバージョンが等しい場合、インストールしません。
    elseif (nResult = EQUALS) then
        MessageBox (" バージョンが同じです。アップデートは不要です。.", INFORMATION);
    endif;

end;
```

VerGetFileLanguages

VerGetFileLanguages 関数は、szFile が指定するファイルによってサポートされている言語リストを読み出します。

構文

```
VerGetFileLanguages ( szFileName, listLanguages );
```

パラメーター

テーブル 17 - VerGetFileLanguages のパラメーター

パラメーター	説明
szFileName	サポートされている言語のリストを読み出すファイルの完全修飾名を指定します。
listLanguages	サポートしている言語の数値言語コード、およびコードページ ID のリストを戻します。各リストの要素は低位ワードにサポートされている言語コードを含み、高位ワードに対応するコードページ ID を含む 32-bit 整数で、LOWORD 関数および HIWORD 関数を使って抽出することができます。 listLanguages が識別する数値リストは、ListCreate(NUMBERLIST) への呼び出しによって既に初期化済みでなくてはなりません。

戻り値

テーブル 18 - VerGetFileLanguages の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がサポートされている言語のリストを読み出しました。
< ISERR_SUCCESS	関数がサポートされている言語のリストを読み出しに失敗しました。

VerGetFileLanguages の例

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * VerGetFileLanguages 関数のデモンストレーションを行います。
 *
¥*---------------------------------------------------------------*/

function OnBegin()
    number nListGetItem, nvLanguageInfo, nLanguageCode, nCodePage;
    string szFileName;
    LIST listLanguages, listLanguageCodes, listCodePages;
begin
    // ファイル言語情報を取得します。
    szFileName = "C:¥¥Program Files¥¥Internet Explorer¥¥Iexplore.exe";
    listLanguages = ListCreate( NUMBERLIST );
    VerGetFileLanguages ( szFileName, listLanguages );
```

```
    // 言語コードおよびコードページ ID を
    // リストアイテムから抽出し、新しいリストへ追加します。
    listLanguageCodes = ListCreate( NUMBERLIST );
    listCodePages = ListCreate( NUMBERLIST );
    nListGetItem = ListGetFirstItem( listLanguages ,nvLanguageInfo);
    while nListGetItem=0
        nLanguageCode = LOWORD( nvLanguageInfo );
        ListAddItem( listLanguageCodes, nLanguageCode, AFTER );
        nCodePage = HIWORD( nvLanguageInfo );
        ListAddItem( listCodePages, nCodePage, AFTER );
        nListGetItem = ListGetNextItem( listLanguages ,nvLanguageInfo);
    endwhile;
end;
```

VerGetFileVersion

VerGetFileVersion 関数は、指定したファイルの数値バージョン情報を読み出します。

メモ・*InstallScript ファイルバージョン関数はバージョン情報を文字列形式で取り出しますが、関数がファイル内で参照するバージョン情報は数値バージョン情報です。InstallScript 関数はファイルの文字列バージョン情報の調査を行わず、戻しません。さらに、Windows Explorer がファイルのプロパティを表示するとき、ファイルの数値バージョン情報と必ずしも対応しない文字列バージョン情報を表示します。この理由から、VerGetFileVersion が svVersionNumber で戻す値は、Windows Express が表示するバージョン情報と一致しない可能性があります。*

ファイル バージョン情報についての詳細は、Windows マニュアルを参照してください。

構文

```
VerGetFileVersion ( szFileName, svVersionNumber );
```

パラメーター

テーブル 19 - VerGetFileVersion のパラメーター

パラメーター	説明
szFileName	数値バージョン情報を読み出すファイルの完全修飾名を指定します。
svVersionNumber	数値バージョン情報を次の形式で文字列として戻します : < メジャー バージョン >.< マイナー バージョン > 例えば、svVersionNumber が 2.1.2.0 を戻す場合、メジャーバージョン番号は 2.1 でマイナーバージョン番号は 2.0 です。

戻り値

テーブル 20 - VerGetFileVersion の戻り値

戻り値	説明
0	関数がバージョン情報を戻したことを示します。
FILE_NOT_FOUND (-2)	指定されたファイルが検出されなかったことを示します。
FILE_NO_VERSION (-8)	ファイルが検出されましたが、バージョン情報を含まないことを示します。

VerGetFileVersion の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* VerGetFileVersion 関数のデモンストレーションを行います。
*
* 下のスクリプトは、VerGetFileVersion を呼び出して Windows Notepad の
* バージョン番号を読み出します。情報は
* メッセージボックスに表示されます。
*
¥*---------------------------------------------------------------------*/

#define EXAMPLE_FILE WINDIR ^ "NotePad.exe"
#define TITLE_TEXT "VerGetFileVersion の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"
```

```
    export prototype ExFn_VerGetFileVersion(HWND);

function ExFn_VerGetFileVersion(hMSI)
    NUMBER nResult;
    STRING szFile, szPath, szMsg, svVersionNumber;
begin

    // 指定したファイルのバージョン番号を取得します。
    nResult = VerGetFileVersion(EXAMPLE_FILE, svVersionNumber);

    // VerGetFileVersion の結果をレポートします。
    if (nResult = FILE_NO_VERSION) then
        szMsg = EXAMPLE_FILE + " はバージョン情報を含みません。";
        MessageBox (szMsg, INFORMATION);
    elseif (nResult = FILE_NOT_FOUND) then
        szMsg = EXAMPLE_FILE + " 検索されませんでした。";
        MessageBox (szMsg, INFORMATION);
    else
        szMsg    = " %s のバージョン番号は %s です ";
        SprintfBox (INFORMATION, TITLE_TEXT, szMsg,
                EXAMPLE_FILE, svVersionNumber);
    endif;

end;
```

VerProductCompareVersions

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

VerProductCompareVersions 関数はバージョン情報を比較して、その結果を示す値を戻します。

構文

```
VerProductCompareVersions ( );
```

パラメーター

なし。

戻り値

テーブル 21 ・ VerProductCompareVersions の戻り値

戻り値	説明
VERSION_COMPARE_RESULT_NOT_INSTALLED	製品のバージョンがインストールされていない、またはインストールされているバージョンが検出されませんでした (IFX_INSTALLED_VERSION がヌル文字列)。
VERSION_COMPARE_RESULT_SAME	アップデート セットアップのバージョンが現在ターゲット システムにインストールされているバージョンと同じです。
VERSION_COMPARE_RESULT_OLDER	アップデート セットアップのバージョンは、現在ターゲット システムにインストールされているバージョンよりも古いです。
VERSION_COMPARE_RESULT_NEWER_NOT_SUPPORTED	アップデート セットアップ バージョンは現在ターゲット システムにインストールされているバージョンよりも新しいが、現在インストールされているバージョンはアップデート セットアップでサポートされていません。
VERSION_COMPARE_RESULT_NEWER	アップデート セットアップ バージョンは現在ターゲット システムにインストールされているバージョンよりも新しく、現在インストールされているバージョンはアップデート セットアップでサポートされています。
< ISERR_SUCCESS	関数が失敗しました。

Comments

この関数はシステム変数 IFX_INSTALLED_VERSION および IFX_PRODUCT_VERSION を比較して、現在ターゲット システムにインストールされているバージョンがアップデート セットアップのバージョンよりも古いかどうかを判断します。古い場合、この関数はシステム変数 IFX_INSTALLED_VERSION と IFX_SUPPORTED_VERSIONS を比較して、アップデート セットアップを現在インストールされているバージョンに適用するかどうかを決定します。

VerProductCompareVersions が、OnSetUpdateMode と OnUpdateUIBefore イベント ハンドラー関数のデフォルト コードによって呼び出されました。

VerProductGetInstalledVersion

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

VerProductGetInstalledVersion 関数は、アプリケーションアンインストール レジストリキーの Version 値がパックされた DWORD の場合、そのデータに対応する文字列を svVersionInstalled に戻します。

構文

VerProductGetInstalledVersion (svVersionInstalled);

パラメーター

テーブル 22 - VerProductGetInstalledVersion のパラメーター

パラメーター	説明
svVersionInstalled	アプリケーションアンインストールレジストリキーの Version の値に等しい文字列を戻します。関数が失敗したとき、svVersionInstalled の値は変更されません。

戻り値

テーブル 23 - VerProductGetInstalledVersion の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数がインストールされたバージョン情報を読み出しました。
< ISERR_SUCCESS	関数がインストールされたバージョン情報の読み出しに失敗しました。レジストリキーまたは値が存在しない、あるいはデータがパックされた DWORD ではありません。

VerProductIsVersionSupported

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

VerProductIsVersionSupported 関数は szVersionCheck の バージョン文字列が szVersionSupported のバージョンの 1 つかどうかを確認します。

構文

VerProductIsVersionSupported (szVersionCheck, szVersionSupported);

パラメーター

テーブル 24 - VerProductIsVersionSupported のパラメーター

パラメーター	説明
szVersionCheck	パックされた DWORD 形式でバージョン文字列を指定します (例えば、システム変数 IFX_PRODUCT_VERSION の初期値。
szVersionSupported	パックされた DWORD 形式で、バージョン文字列の垂直線 (|) 区切りリストを指定します (例えば、システム変数 IFX_SUPPORTED_VERSIONS の初期値)。

戻り値

テーブル 25 - VerProductIsVersionSupported の戻り値

戻り値	説明
TRUE	szVersionCheck は、szVersionSupported、または szVersionSupported のバージョンのひとつがヌル文字列 ("") です。
FALSE	szVersionCheck は、szVersionSupported のバージョンのひとつではありません。

VerProductNumToStr

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

VerProductNumToStr 関数は、nVersion で指定されたパックされた DWORD に対応するバージョン文字列 を svVersion に戻します。

構文

VerProductNumToStr (svVersion, nVersion);

パラメーター

テーブル 26 - VerProductNumToStr のパラメーター

パラメーター	説明
svVersion	nVersion に対応する文字列を、パックされた DWORD 形式で戻します。
nVersion	バージョン情報を 4 バイト値として指定します。つまり、0 から 4294967295 の範囲内の数値です。

戻り値

テーブル 27 - VerProductNumToStr の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数が成功しました。
< ISERR_SUCCESS	関数が失敗しました。

VerProductStrToNum

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

VerProductStrToNum 関数は、svVersion が指定した文字列に対応する パックされた DWORD バージョン情報を nvVersionResult に戻します。

構文

```
VerProductStrToNum ( nvVersionResult, szVersion );
```

パラメーター

テーブル 28 - VerProductStrToNum のパラメーター

パラメーター	説明
nvVersionResult	パックされた DWORD バージョン情報を戻します。
szVersion	バージョン情報を文字列の形式で指定します。szVersion が次の条件のすべてに一致しない場合、関数が失敗します : • szVersion は major.minor.build または major.minor.build.release (nvVersionResult を判断する場合はリリースは無視されます) の形式です。 • メジャーとマイナーは 0 から 255 の範囲内の数値に相応する文字列です。 • ビルドは 0 から 65535 の範囲内の数値に相応する文字列です。

戻り値

テーブル 29 - VerProductStrToNum の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数が成功しました。
< ISERR_SUCCESS	関数が失敗しました。

VerProductVerFromVerParts

プロジェクト - *この情報は、InstallScript プロジェクトに適用します。*

VerProductVerFromVerParts 関数は、nVersionMajor、nVersionMinor、そして nVersionBuild が指定したバージョン部分に対応するパックされた DWORD を読み出します。

構文

```
VerProductVerFromVerParts ( nvVersion, nVersionMajor, nVersionMinor, nVersionBuild );
```

パラメーター

テーブル 30 ・ VerProductVerFromVerParts のパラメーター

パラメーター	説明
nvVersion	nVersionMajor、nVersionMinor、そして nVersionBuild が指定したバージョン部分へ対応する 4 バイト値を戻します。
nVersionMajor	バージョン情報の最初のバイトを指定します。この値が (0) 未満、または 255 より大きい場合、VerProductVerFromVerParts は失敗します。
nVersionMinor	バージョン情報の 2 番目のバイトを指定します。この値が (0) 未満、または 255 より大きい場合、VerProductVerFromVerParts は失敗します。
nVersionBuild	バージョン情報の最後の 2 のバイトを指定します。この値が (0) 未満、または 65535 より大きい場合、VerProductVerFromVerParts は失敗します。

戻り値

テーブル 31 ・ VerProductVerFromVerParts の戻り値

戻り値	説明
>= ISERR_SUCCESS	関数が成功したことを示します。
< ISERR_SUCCESS	関数が失敗したことを示します。

VerProductVerPartsFromVer

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

VerProductVerPartsFromVer 関数は、nVersion が指定したパックされた DWORD のバージョン部分を別の数値として読み出します。

構文

VerProductVerPartsFromVer (nVersion, nvVersionMajor, nvVersionMinor, nvVersionBuild);

パラメーター

テーブル 32 - VerProductVerPartsFromVer のパラメーター

パラメーター	説明
nVersion	バージョン情報を 4 バイト値として指定します。つまり、0 から 4294967295 の範囲内の数値です。
nvVersionMajor	nVersion の最初のバイトを戻します。
nvVersionMinor	nVersion の 2 番目のバイトを戻します。
nvVersionBuild	nVersion の最後の 2 バイトを戻します。

戻り値

テーブル 33 - VerProductVerPartsFromVer の戻り値

戻り値	説明
ISERR_SUCCESS	この関数は常に ISERR_SUCCESS を戻します。

VerSearchAndUpdateFile

VerSearchAndUpdateFile 関数は、指定したファイルを検索し、必要に応じてより新しいバージョンをインストールします。関数がファイルを検出すると、既存ファイルのバージョン番号と新しいファイルのバージョン番号を比較します。既存ファイルの方が古い場合、新しいファイルと置換されます。新しいファイルはシステム変数 SRCDIR が指定するディレクトリに存在しなくてはなりません。関数が既存ファイルを検出できない場合、新規ファイルをターゲットシステムへコピーします。Windows はファイルのインストール場所をファイルの種類によって判断します。たとえば、DLL とシステムドライブは Windows システムフォルダーにインストールされます。Windows システムフォルダーの詳細については、InstallScript システム変数 WINSYSDIR の説明を参照してください。

VerFindFileVersion は、次の検索アルゴリズムを利用してファイルを検出します (次の順番でフォルダーを検索します):

1. Windows フォルダー
2. Windows システム フォルダ
3. TARGETDIR システム変数 (InstallScript インストールの場合) または INSTALLDIR システム変数 (基本の MSI または InstallScript MSI インストールの場合) が指定したフォルダー
4. PATH 環境変数が指定したフォルダー
5. **Setup.exe** が実行されるフォルダー

メモ - XCopyFile は VerSearchAndUpdateFile の代替となるファイル転送方法です。XCopyFile はバージョンチェックを行い、システム再起動の後にアップデートできるようにロックされた .dll や .exe ファイルをマークし、共有 .dll と .exe ファイルのレジストリ参照カウンターを増加させることがます。

構文

VerSearchAndUpdateFile (szFileName, nUpdateFlag, svInstalledFile);

パラメーター

テーブル 34 - VerSearchAndUpdateFile のパラメーター

パラメーター	説明
szFileName	インストールするファイルの非完全修飾名を指定します。このパラメーターではドライブ指定またはパスを指定しないで下さい。
nUpdateFlag	ファイルを無条件に更新するか、ターゲット システムで検出されたファイルのバージョンが送ったファイルのバージョンよりも古い場合にのみ更新するかどうかを指定します。このパラメーターに、以下の定義済み定数のうちの 1 つを渡します。 • VER_UPDATE_COND– ファイルが古いバージョンの場合のみ既存ファイルを更新します。 • VER_UPDATE_ALWAYS– ファイルが新しいバージョンの場合でも既存ファイルを更新します。
svInstalledFile	関数がインストールしたファイルの完全修飾名を戻します。置換するファイルが使用中の場合、ファイルは同じディレクトリに若干異なる名前でインストールされます。拡張子の最初の文字にチルダ文字 (~) を付けてファイル名が変更されます。例えば、Shell.dll をインストールするときにファイルがロックされている場合、そのファイルは Shell.~ll としてコピーされます。名前はこの変数に戻されます。

戻り値

テーブル 35 - VerSearchAndUpdateFile の戻り値

戻り値	説明
FILE_INSTALLED (0)	関数がファイルをインストールに成功しました。
FILE_IS_LOCKED (-4)	ファイルは Windows が使用中で置換できないことを示します。新規ファイルが同じディレクトリに新しい名前と共にコピーされます。
FILE_NO_VERSION (-8)	ファイルが検出されましたが、バージョン情報を含まないことを示します。ファイルの更新は実行されません。
FILE_RD_ONLY (-5)	既存ファイルが書き込み禁止であることを示します。セットアップを続行する前に、スクリプトがインストール先ファイルの読み取り専用フラグをリセットしてからファイルのインストールを再度行わなくてはなりません。
FILE_SRC_OLD (-7)	インストールするファイルが同じ日付、または既存ファイルの方が古いことを示します。
OUT_OF_DISK_SPACE (-6)	インストール先ドライブに充分な空き容量が無いため、関数がファイルを作成できないことを示します。ファイルの更新は実行されません。
VER_DLL_NOT_FOUND (-3)	Ver.dll が検出されなかったことを示します。ファイルの更新は実行されません。

テーブル 35 ・ VerSearchAndUpdateFile の戻り値 (続き)

戻り値	説明
OTHER_FAILURE (-1)	不特定エラーが発生したことを示します。ファイルの更新は実行されません。

VerSearchAndUpdateFile の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* VerSearchAndUpdateFile 関数のデモンストレーションを行います。
*
* VerSearchAndUpdateFile への最初の呼出しで、ターゲットシステムの
* バージョンに関わりなく、定数 UPDATE_FILE1 が指定したファイルを
* 置換します。
*
* VerSearchAndUpdateFile への 2 番目の呼出しで、ソースディレクトリの
* バージョンがターゲットディレクトリよりも新しい場合のみ
* 定数 UPDATE_FILE2 が指定したファイルを置換
* コピーします。
*
¥*---------------------------------------------------------------------*/

#define UPDATE_FILE1  "Example.txt"
#define UPDATE_FILE2  "Readme.txt"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_VerSearchAndUpdateFile(HWND);

function ExFn_VerSearchAndUpdateFile(hMSI)
   STRING szFileName, svInstalledFile, szTitle, szMsg;
   NUMBER nUpdateFlag, nResult;
   BOOL   bDone;
begin

   // VerSearchAndUpdateFile を呼び出すタイトルとメッセージパラメーターを
   // セットアップします。
   szTitle = "VerSearchAndUpdateFile の例 ";
   szMsg   = " 正常にアップデートしました。";

   // バージョン番号に関わらず、UPDATE_FILE1 を更新します。
   if (VerSearchAndUpdateFile (UPDATE_FILE1, VER_UPDATE_ALWAYS,
                               svInstalledFile) = 0) then
      SprintfBox (INFORMATION, szTitle, UPDATE_FILE1 + szMsg);
   endif;
```

```
// ループ終了を制御するインジケーターを設定します。
bDone = FALSE;

// while ループを始めます。
while (bDone = FALSE)
   // 既存ファイルの方が古い場合のみ UPDATE_FILE2 を更新します。
   nResult = VerSearchAndUpdateFile (UPDATE_FILE2, VER_UPDATE_COND,
                                     svInstalledFile);
   switch (nResult)
      case 0:
         // VerSearchAndUpdate が成功しました。
         SprintfBox (INFORMATION, szTitle, UPDATE_FILE2 + szMsg);
         bDone = TRUE;
      // ターゲットファイルにはバージョン番号がありません。
      case FILE_NO_VERSION:
         // ファイルを更新すべきかどうかをユーザーに問い合わせます。
         if (AskYesNo (" バージョン番号が検出されませんでした。¥n " +
                  "" + UPDATE_FILE2 + " を更新しますか ?", YES) = YES) then
            // バージョン番号に関わらず、UPDATE_FILE2 を更新します。
            VerSearchAndUpdateFile (UPDATE_FILE2, VER_UPDATE_ALWAYS,
                                    svInstalledFile);
            bDone = TRUE;
         else
            bDone = TRUE;
         endif;

      // ターゲットファイルがロックされています。
      case FILE_IS_LOCKED:
         MessageBox (" ターゲット ファイルはロックされています。¥n¥n すべての " +
                  " プログラムを閉じてセットアップを再度実行してください。", INFORMATION);
         bDone = TRUE;
      // ターゲットファイルは読み取り専用です。
      case FILE_RD_ONLY:
         // Setup が読み取り専用属性を削除すべきかどうかをユーザーに問い合わせます。
         if (AskYesNo (" ファイルは読み取り専用です。¥n セットアップで " +
                  UPDATE_FILE2 + " の書き込み保護を削除しますか ?", YES) = YES) then
            // ターゲットファイルの属性を標準に変更します。
            SetFileInfo (svInstalledFile, FILE_ATTRIBUTE, FILE_ATTR_NORMAL, "");
            bDone = FALSE;
         else
            bDone = TRUE;
         endif;
      // ターゲットディスクに充分な容量がありません。
      case OUT_OF_DISK_SPACE:
         MessageBox (" この更新を行うにはより多くの空き容量が必要です。", SEVERE);
         bDone = TRUE;
      // 必要な VER.DLL ファイルが検出されませんでした。
      case VER_DLL_NOT_FOUND:
         MessageBox ("VER.DLL が検出されませんでした。.", SEVERE);
         bDone = TRUE;
      // その他のエラーが発生しました。
      case OTHER_FAILURE:
         MessageBox (" 更新が失敗しました。", SEVERE);
         bDone = TRUE;
      デフォルト :
         bDone = TRUE;
   endswitch;

endwhile;
```

```
end;
```

VerUpdateFile

VerUpdateFile 関数は、指定されたファイルのバージョン情報を使って、そのファイルをターゲットディレクトリにインストールするかどうかを決定します。VerUpdateFile は szFileName で指定されたファイル名を取得します。

VerFindFileVersion は、次の検索アルゴリズムを利用してファイルを検出します (次の順番でフォルダーを検索します):

1. Windows フォルダー
2. Windows システム フォルダ
3. TARGETDIR システム変数 (InstallScript インストールの場合) または INSTALLDIR システム変数 (基本の MSI または InstallScript MSI インストールの場合) が指定したフォルダー
4. PATH 環境変数が指定したフォルダー
5. **Setup.exe** が実行されるフォルダー

そして VerUpdateFile は、ターゲットファイルが存在する場合に SRCDIR (ソースファイル) にある同じ名前のファイルのバージョンをこのターゲットファイルのバージョンと比較します。ソースファイルのバージョンがターゲットファイルのバージョンよりも新しい場合は、ターゲットファイルはソースファイルで置き換えられます。ターゲットファイルが存在しない場合には、InstallShield によってソースファイルがターゲット位置にコピーされます。

パラメーター nUpdateFlag で SHAREDFILE または LOCKEDFILE オプションが利用される時に、更新される .dll または .exe ファイルがシステムで使用中の場合には、ソースファイルの名前を変更したコピーがターゲットシステムに転送され、システム変数 BATCH_INSTALL が TRUE に設定されます。そしてセットアップの最後に **RebootDialog** または **SdFinishReboot** が呼び出されてシステムが再起動されるとき、ロックされたファイルが更新されます。ロックされたファイルの更新についての詳細は、RebootDialog と SdFinishReboot を参照してください。システム変数 BATCH_INSTALL をテストすると、ロックされた .dll または .exe ファイルが存在したかどうかを調べることができます。SHAREDFILE と LOCKEDFILE オプションは同時に利用することはできません。どちらかひとつのみを利用しなくてはなりません。

XCopyFile は VerUpdateFile よりも好ましいファイル転送方法です。XCopyFile はバージョンチェックを行い、システム再起動の後にアップデートできるようにロックされた .dll や .exe ファイルをマークし、共有 .dll と .exe ファイルのレジストリ参照カウンターを増加させます。

構文

```
VerUpdateFile (szFileName, nUpdateFlag, svInstalledFilePath);
```

パラメーター

テーブル 36 · VerUpdateFile のパラメーター

パラメーター	説明
szFileName	更新するファイルの完全修飾名または非完全修飾名を指定します。名前が非完全修飾の場合 (つまり、ドライブ指定またはパスを含まない)、InstallShield は Windows または Win95 ディレクトリ、システム ディレクトリ、PATH 環境変数が指定したディレクトリ、そして InstallShield 実行可能ファイルのパスで一致するファイルを検索します。VerUpdateFile は szFileName のファイル名部分を取り出し、ソースファイルとして利用される SRCDIR のファイルを識別するのに利用します。
nUpdateFlag	ファイルを無条件で更新するか、またはターゲットアフィルのバージョンがソースファイルのバージョンより古い場合にのみ更新するかを指定します。このパラメーターに、あらかじめ定義されている以下の定数のうちの 1 つを渡します。定数 SHAREDFILE は、ビット単位 OR 演算子 (|) を使って、その他の定数の 1 つと組み合わせることができます。しかし、SHAREDFILE と LOCKEDFILE を組み合わせることはできません。 • LOCKEDFILE—Windows またはシステムを再起動したときに、更新するロックされた .dll および .exe ファイルが VerUpdateFile 関数によって記録されます。ロックされたファイルとは、InstallShield によってファイルのアクセスまたは更新が試行されたときに、アプリケーションまたはシステムで使用されているファイルのことです。LOCKEDFILE オプションは、SHAREDFILE と同じように機能します。ただし、LOCKEDFILE ではレジストリエントリを作成したりレジストリ参照カウンターを変更することはできません。SHAREDFILE オプションを使用している時に LOCKEDFILE オプションを使用することはできません。スクリプトの作成者がレジストリエントリおよび参照カウンターを必要としない非共有ファイル (シェル拡張子など) もいくつかあります。これらのファイルはアプリケーションによってアンインストールされないかぎり、絶対にアンインストールしないでください。LOCKEDFILE を使用すると、ロックされた非共有ファイルを VerUpdateFile で処理することができます。 • SHAREDFILE—VerUpdateFile がすべてのファイルを共有ファイルとして扱うようにし、Windows またはシステムが再起動されたときに更新するロックされた .dll および .exe ファイルを記録することで、ロックされたファイルと共有ファイルの処理を組み合わせます。RebootDialog と SdFinishReboot を参照してください。 SHAREDFILE オプションを使うと、VerUpdateFile はすべてのファイルを共有ファイルとして処理し、ファイルがターゲット ディレクトリに存在して、0 より大きい参照カウンターを持つ場合、レジストリ参照カウントを 1 増やします。共有ファイルがターゲットディレクトリに存在せず、参照カウンターがない場合、InstallShield はカウンターを作成して 1 に設定します。共有ファイルがターゲット ディレクトリに存在するが、参照カウンターがない場合、InstallShield はカウンターを作成して、アンインストール中に誤って削除されないように、これを 2 に初期化します。

テーブル 36 ・ VerUpdateFile のパラメーター (続き)

パラメーター	説明
nUpdateFlag (続き)	・ SELFREGISTER— 「非バッチ メソッド」で自己登録ファイルをインストールしたときに、自己登録処理をただちに実行します。 Enable (SELFREGISTERBATCH) を呼び出すと、自己登録ファイルは登録キューに配置されます。「バッチメソッド」で自己登録ファイルをインストールした場合には、Do (SELFREGISTRATIONPROCESS) を呼び出したときにファイルは登録されます。 SELFREGISTER オプションと定数 SHAREDFILE は、常にビット単位 OR 演算子 (|) でつなげて一緒に使用します。 ・ VER_UPDATE_ALWAYS— バージョン番号にかかわらず、ファイルを更新します。 ・ VER_UPDATE_COND— 置換するファイルがその古いバージョンの場合のみ、ファイルを更新します。
svInstalledFilePath	インストールされたファイルの完全修飾名を戻します。置換するファイルが使用中の場合、ファイルは同じディレクトリに修正された名前でインストールされます。InstallShield はチルダ文字 (~) を使って、ファイル拡張子の最初の文字を置換します。 たとえば、ファイル Shell.dll をアップデートする時にターゲットがロックされている場合、ソースファイルはターゲットディレクトリに Shell.~ll としてコピーされます。この名前がパラメーター svInstalledFilePath に戻されます。 SHAREDFILE オプションがパラメーター nUpdateFlag で利用され、Windows またはシステムが再起動した場合のアップデート用にロックされたファイルが適切に送られた場合、アップデートが実行された時にファイルの ~ 変更済み名は削除されます。

戻り値

テーブル 37 - VerUpdateFile の戻り値

戻り値	説明
FILE_INSTALLED (0)	関数がファイルをインストールしたことを示します。この定数は 0 (ゼロ) に等しいです。その他のすべての戻り値は ゼロ以下 (0) です。
FILE_IS_LOCKED (-4)	既存ファイルは Windows が使用中で置換できないことを示します。新規ファイルが前述の通り、同じディレクトリに新しい名前と共にコピーされます。
FILE_NO_VERSION (-8)	ファイルが検出されましたが、バージョン情報を含まないことを示します。ファイルの更新は実行されません。
FILE_RD_ONLY (-5)	既存ファイルが書き込み禁止であることを示します。セットアップを続行する前に、インストール先ファイルの読み取り専用部分をリセットしてからファイルのインストールを再度行わなくてはなりません。
FILE_SRC_EQUAL (-9)	インストールするファイルが既存ファイルと同じバージョンを持っている事を示します。VER_UPDATE_COND フラグが設定されている場合、ファイルの更新は実行されません。
FILE_SRC_OLD (-7)	インストールするファイルが、既存ファイルよりも古いバージョンであることを示します。VER_UPDATE_COND フラグが設定されている場合、ファイルの更新は実行されません。
OUT_OF_DISK_SPACE (-6)	インストール先ドライブに充分な空き容量が無いため、関数がファイルを作成できないことを示します。ファイルの更新は実行されません。
-51	自動登録ファイルが登録されませんでした。
OTHER_FAILURE (-1)	不特定エラーが発生したことを示します。ファイルの更新は実行されません。

VerUpdateFile の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* VerUpdateFile 関数のデモンストレーションを行います。
*
* このスクリプトは、VerUpdateFile を 2 回呼び出して Windows
* アクセサリを更新します。
*
¥*---------------------------------------------------------------*/
```

```
#define APPFILE "Notepad.exe"
#define TITLE   "VerUpdateFile の例 "

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_VerUpdateFile(HWND);

function ExFn_VerUpdateFile(hMSI)
   STRING svInstalledFilePath, szTitle, szMsg;
   NUMBER nResult;
   BOOL   bDone;
begin

   // ファイルバージョンにかかわらず、ファイルを更新します。
   nResult = VerUpdateFile (APPFILE, VER_UPDATE_ALWAYS, svInstalledFilePath);

   if (nResult < 0) then
      MessageBox ("VerUpdateFile への最初の呼び出しに失敗しました。", SEVERE);
   else
      szMsg = "%s がアップデートされました。";
      SprintfBox (INFORMATION, TITLE, szMsg, APPFILE);
   endif;

   // ターゲットファイルの方が新しい場合にのみファイルを更新します。
   nResult = VerUpdateFile (APPFILE, VER_UPDATE_COND, svInstalledFilePath);

   if (nResult < 0) then
      MessageBox ("VerUpdateFile への 2 番目の呼び出しに失敗しました。", SEVERE);
   endif;

end;
```

WaitForApplication

WaitForApplication 関数は、実行中のアプリケーションが終了するのを待機してから、戻されます。

関数が、戻る前に、実行中のアプリケーションが終了するのを待機しなかった場合、アプリケーションが起動したサブアプリケーションが終了するまで終了しないことを検証します。**WaitForApplication** 関数は、指定されたアプリケーションのプロセス ハンドルを監視します。アプリケーションがセカンダリ アプリケーションまたはプロセスに制御を渡してから、終了した場合、関数はすぐに終了します。

構文

```
WaitForApplication( byval number hProcess, byval number dwProcessId, byval number nTimeOut, byval number nOptions );
```

パラメーター

テーブル 38 · WaitForApplication のパラメーター

パラメーター	説明
hProcess	待機する実行中のプロセスのプロセス ハンドルを指定します。(アプリケーションが **LaunchAppAndWait** または **LaunchApp** で起動された場合、ハンドルは LAAW_PROCESS_INFORMATION.hProcess メンバーで返されます。)
dwProcessId	待機する実行中のプロセスのプロセス ID を指定します。nOptions で LAAW_OPTION_WAIT_INCL_CHILD を指定した場合のみ。このパラメーターにゼロ以外の値を指定する必要があります。 アプリケーションが **LaunchAppAndWait** または **LaunchApp** で起動され、プロセス ID が判別可能な場合 (LAAW_OPTION_WAIT_INCL_CHILD の説明を参照してください)、このハンドルは LAAW_PROCESS_INFORMATION.dwProcessId メンバーで返されます。
nTimeOut	戻る前に、アプリケーションが完了するのを待機する最長時間 (ミリ秒) を指定します。アプリケーションが終了するまえに、指定された時間が経過した場合、関数は待機を終了して、失敗を返します。関数が無期限に待機することを示す場合、INFINITE を指定します。 (LAAW_PARAMTERS.nTimeOut 値を渡して、**LaunchAppAndWait** 関数の動作を模倣することもできます。)

テーブル 38 · WaitForApplication のパラメーター (続き)

パラメーター	説明
nOptions	このパラメーターに、あらかじめ定義されている以下の定数のうちの 1 つを渡します。LAAW_OPTION_NOWAIT を LAAW_OPTION_WAIT の組み合わせを例外として、ビットごとの OR 演算子 (|) を利用して、これらの定数を組み合わせることができます。 · LAAW_OPTION_WAIT– 数が使用されたアプリケーションが終了するまで待機することを指定します。 ***メモ ·** 指定されたアプリケーションが終了に失敗したときに、nOptions が LAAW_OPTION_WAIT に設定されているとき (および nTimeOut が INFINITE に設定されているとき)、インストールは起動されたアプリケーションが終了するまで無期限に待機します。* · LAAW_OPTION_NOWAIT– 数がすぐに戻ることを指定します。 · LAAW_OPTION_USE_CALLBACK– 果として関数が OnLaunchAppAndWaitCallback イベントを毎秒、またはアプリケーションが終了するまで待機する間に LAAW_PARAMETERS.nCallbackInterval メンバーに設定された間隔で呼び出します。 · LAAW_OPTION_WAIT_INCL_CHILD– 関数が起動されるアプリケーションとその直属の子プロセスを待機することを示します。 LAAW_OPTION_WAIT_INCL_CHILD が使用されたとき、関数は、起動されたプロセスの直接の子プロセスのみ検出し待機します。初回で起動されたプロセスの子プロセスによって起動されたその他の子プロセスは検出 / 待機されません。 LAAW_OPTION_WAIT_INCL_CHILD を使って子プロセスを検出および待機するとき、関数が起動されたプロセスのプロセス ID を受け取って、それを判別できる必要があります。したがって、このオプションが動作するために、dwProcessId が指定されていない場合、関数がプロセス ID を判別できるように Windows API **GetProcessId** がシステムで使用可能である必要があります。Windows API のドキュメントによると、**GetProcessId** は Windows XP SP1 以降および Windows Server 2003 以降で使用可能です。この API が使用できない場合、LAAW_OPTION_WAIT_INCL_CHILD は LAAW_OPTION_WAIT と同じ動作をします。 LaunchApplication オプションはどれでも指定可能ですが、前記のオプションのみ効果があります。

戻り値

テーブル 39 ・ WaitForApplication の戻り値

戻り値	説明
>= ISERR_SUCCESS	アプリケーションが終了した結果、関数が戻りました。
< ISERR_SUCCESS	アプリケーション以外のものが終了した結果、関数が戻りました。LAAW_PARAMETERS.nWaitResult の値を確認して、追加の情報を判別することができます : ・ WAIT_OBJECT_0— アプリケーションが終了したことを示します。 ・ WAIT_OBJECT_0 + 1— インストールが終了のメッセージを受け取ったことを示します。 ・ WAIT_TIMEOUT— タイムアウト間隔が経過したか、またはコールバック関数が LAAW_CALLBACK_RETURN_END_WAIT を返したことを示します。LAAW_PARAMETERS.bCallbackEndedWait の値を確認して、どちらが待機を終了したかを判別します。

WaitOnDialog

WaitOnDialog 関数はカスタム ダイアログを表示します。この関数からの戻り値に基づいてユーザーからの様々な返答を処理するスクリプトを書くことができます。

構文

```
WaitOnDialog ( szDlgName );
```

パラメーター

テーブル 40 ・ WaitOnDialog のパラメーター

パラメーター	説明
szDlgName	表示するダイアログの ID を指定します。

戻り値

テーブル 41 ・ WaitOnDialog の戻り値

戻り値	説明
ダイアログ コントロール ID	WM_COMMAND メッセージを受け取ったダイアログコントロールの ID 。
IDCANCEL (2)	このメッセージはダイアログが閉じる前の合図として受け取られます。
DLG_ERR (-1)	このメッセージはエラーが発生したときに受け取られます。
DLG_INIT (-100)	このメッセージはダイアログが表示される直前に受け取られます。

追加情報

エンド ユーザーが InstallScript ダイアログの右上にある閉じるボタンをクリックしてインストールをキャンセルできるようにするには、ダイアログにコントロール ID プロパティが 2 に設定されているボタン コントロールを含まなくてはなりません。詳細については、「InstallScript を使用してカスタム ダイアログを実装する」を参照してください。

WaitOnDialog の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* DefineDialog 関数、WaitOnDialog 関数、EndDialog 関数、そして ReleaseDialog 関数の
* デモンストレーションを行います。
*
* このスクリプトは ビットマップを表示するシンプルなカスタムダイアログを
* 開きます。ダイアログは次の 3 つのボタンで閉じることが
* [ 戻る ]、[ 次へ ]、および [ キャンセル ]。
*
* このスクリプトで利用される [ カスタム ] ダイアログは、
* 実際、ビルトイン関数 SdAskOptions が表示する
* InstallShield 標準ダイアログです。このダイアログは
* インストールで既に圧縮済みのファイル _isres.dll に保存されているため、
* カスタム ダイアログとしてスクリプトで利用することが
```

```
＊表示されます。
＊
＊このダイアログをカスタムダイアログとして利用するためには、
＊DefineDialog を呼び出してそれをスクリプトで定義します。その後
＊WaitOnDialog を呼び出してダイアログを表示します。イベントが
＊ダイアログの処理を終了するとき、それを閉じるために EndDialog が
＊表示されます。次いで、ReleaseDialog への呼び出しによって、
＊メモリからダイアログがリリースされます。
＊
¥*-------------------------------------------------------------*/

// ダイアログ ID とコントロール ID。
#define RES_DIALOG_ID     12027   // ダイアログ自身の ID
#define RES_PBUT_NEXT       1   // [ 次へ ] ボタンの ID
#define RES_PBUT_CANCEL     9   // [ キャンセル ] ボタンの ID
#define RES_PBUT_BACK      12   //[ 戻る ] ボタンの ID

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_WaitOnDialog(HWND);

function ExFn_WaitOnDialog(hMSI)
    STRING  szDialogName, szDLLName, szDialog;
    NUMBER  nDialog, nResult, nCmdValue;
    BOOL    bDone;
    HWND    hInstance, hwndParent, hwndDlg;
begin

    // DefineDialog への最初のパラメーターとして渡すダイアログの
    // 名前を定義します。
    szDialogName = "ExampleDialog";

    // DefineDialog の 2 番目のパラメーターは 0 となります。
    // d l l が _isres.dll の中にあるためです。
    hInstance = 0;

    // DefineDialog の 3 番目のパラメーターは ヌルとなります。インストールは
    // _isuser.dll と _isres.dll にあるダイアログを検索します。
    szDLLName = "";

    // DefineDialog の 5 番目のパラメーターは 0 となります。なぜなら、
    // 4 番目のパラメーターにある ID によってダイアログが認識されるためです。
    szDialog = "";

    // この値は保存され、0 でなくてはなりません。
    hwndParent = 0;

    // ダイアログを定義します。インストールのメイン ウィンドウがダイアログを所有します
    // ( パラメーター 7 内の HWND_INSTALL で表示されます )。
    nResult  = DefineDialog (szDialogName, hInstance, szDLLName,
                        RES_DIALOG_ID, szDialog, hwndParent,
                        HWND_INSTALL, DLG_MSG_STANDARD|DLG_CENTERED);

    // エラーをチェックします。
    if (nResult < 0) then
        MessageBox (" ダイアログを定義中にエラーが発生しました。", SEVERE);
        bDone = TRUE;
        abort;
```

```
    endif;

    // while ループを制御するのに使われるインジケーターを初期化します。
    bDone = FALSE;

    // 完了するまでループします。
    repeat

        // ダイアログを表示して次のダイアログ イベントを戻します。
        nCmdValue = WaitOnDialog(szDialogName);

        // イベントに応答します。
        switch (nCmdValue)
            case DLG_CLOSE:
                // ユーザーがウィンドウの [ 閉じる ] ボタンをクリックしました。
                Do (EXIT);
            case DLG_ERR:
                MessageBox (" ダイアログを表示できませんでした。セットアップがキャンセルされました。", SEVERE);
                abort;
            case DLG_INIT:
                // このダイアログの [ 戻る ]、[ 次へ ]、および [ キャンセル ] ボタンと有効 / 無効状態を初期化して、
                // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
                // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
                // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
                //
                hwndDlg = CmdGetHwndDlg (szDialogName);
                SdGeneralInit(szDialogName, hwndDlg, 0, "");
            case RES_PBUT_CANCEL:
                // ユーザーが [ キャンセル ] ボタンをクリックしました。
                Do (EXIT);
            case RES_PBUT_NEXT:
                bDone = TRUE;
            case RES_PBUT_BACK:
                bDone = TRUE;
        endswitch;

    until bDone;

    // ダイアログを閉じます
    EndDialog (szDialogName);

    // メモリからダイアログを解放します。
    ReleaseDialog (szDialogName);

end;
```

Welcome

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *InstallScript*
- *InstallScript MSI*

Welcome 関数は、エンドユーザーへの [ようこそ] メッセージを含むダイアログを表示します。

手続き型スクリプトの場合、InstallShield によって Welcome ダイアログのメッセージ テキストの最初の段落に製品名が挿入されるよう、Welcome より先に SdProductName を呼び出す必要がありますイベントベースのスクリプトの場合は、SdProductName が、PRODUCT_NAME 文字列エントリを引数として Begin イベントより先に自動的に呼び出されます。SdProductName を使用して製品名を渡さない場合、InstallShield によって製品名の代わりにスペースが挿入されます。

構文

Welcome (szTitle, nReserved);

パラメーター

テーブル 42 ・ Welcome のパラメーター

パラメーター	説明
szTitle	このダイアログのタイトルを指定します。デフォルトのタイトル ([ようこそ]) を表示するには、このパラメーターでヌル文字列 ("") を渡します。
nReserved	このパラメーターで 0 (ゼロ) を渡します。

戻り値

テーブル 43 ・ Welcome の戻り値

戻り値	説明
NEXT (1)	エンドユーザーが、[次へ] ボタンをクリックしたことを示します。
BACK (12)	エンドユーザーが、[戻る] ボタンをクリックしたことを示します。
< 0	Welcome がダイアログを表示できなかったことを示します。

追加情報

インストール内のこのダイアログまたはその他のダイアログの例を参照するには、ダイアログサンプラーを利用します。InstallShield の [ツール] メニューで [InstallScript] をポイントして、[標準ダイアログサンプラー] または [スキン適用のダイアログ サンプラー] をクリックします。

Welcome の例

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

- *InstallScript*
- *InstallScript MSI*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
```

```
＊Welcome 関数のデモンストレーションを行います。
＊
＊このスクリプトは、installation Welcome ダイアログを表示します。
＊
¥*---------------------------------------------------------------*/

#define PRODUCT "ExampleProduct"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_Welcome(HWND);

function ExFn_Welcome(hMSI)
  STRING svLogFile;
begin

  SdProductName ( PRODUCT );

  // ようこそダイアログを表示します。
  if (Welcome ("Welcome ダイアログ ボックスの例 ", 0) 0 then
    MessageBox ("Welcome ダイアログが失敗しました。", SEVERE);
  endif;

end;
```

WizardDirection

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

WizardDirection 関数は、オブジェクトスクリプトで呼び出され、メインのセットアップ (または親オブジェクト) スクリプトで ShowObjWizardPages 関数またはオブジェクトの ShowxxxxxUIyyyyy メソッドに対する最後の呼び出しに渡された引数を報告します。

構文

```
WizardDirection ( );
```

パラメーター

なし。

戻り値

テーブル 44 - WizardDirection の戻り値

戻り値	説明
NEXT	メインのセットアップ (または親オブジェクト) スクリプトでの ShowObjWizardPages 関数またはオブジェクトの Show*xxxxx*UI*yyyyy* メソッドのひとつに対する最後の呼び出しで、引数として NEXT が渡されたことを示します。
BACK	メインのセットアップ (または親オブジェクト) スクリプトでの ShowObjWizardPages 関数またはオブジェクトの Show*xxxxx*UI*yyyyy* メソッドのひとつに対する最後の呼び出しで、引数として BACK が渡されたことを示します。

追加情報

WizardDirection は、メインのセットアップ (または親オブジェクト) のダイアログ シーケンスからオブジェクトのダイアログシーケンスに最後に移行した時点で、エンドユーザーがダイアログ シーケンスで移動した方向を報告することを目的としています。詳細は、「オブジェクトのランタイム UI を作成する」を参照してください。

WriteArrayProperty

***プロジェクト -** この情報は、InstallScript プロジェクトに適用します。*

WriteArrayProperty 関数は、値が配列の指定プロパティに値を入力するため、オブジェクトスクリプトで呼び出します。

構文

WriteArrayProperty (nPropertyBag, szPropertyName, ArrayPointer);

パラメーター

テーブル 45 ・ WriteArrayProperty のパラメーター

パラメーター	説明
nPropertyBag	プロパティの値が保存される、オブジェクトのプロパティバッグオブジェクトへの参照を指定します
szPropertyName	値を入力するプロパティの名前を指定します。
ArrayPointer	配列プロパティへのポインターを指定します。

戻り値

テーブル 46 ・ WriteArrayProperty の戻り値

戻り値	説明
0	この関数は常にゼロ (0) を返します。

WriteBoolProperty

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

WriteBoolProperty 関数は、値がブール値の指定プロパティに値を入力するため、オブジェクトスクリプトで呼び出します。

構文

```
WriteBoolProperty ( nPropertyBag, szPropertyName, bPropertyValue );
```

パラメーター

テーブル 47 - WriteBoolProperty のパラメーター

パラメーター	説明
nPropertyBag	プロパティの値が保存される、オブジェクトのプロパティバッグオブジェクトへの参照を指定します
szPropertyName	値を入力するプロパティの名前を指定します。
bPropertyValue	プロパティの値を指定します。

戻り値

テーブル 48 - WriteBoolProperty の戻り値

戻り値	説明
0	この関数は常にゼロ (0) を返します。

WriteBytes

WriteBytes 関数はバイナリモードで開かれたファイルへ特定のバイト数を書き込みます。この関数は 現在のファイルポインター位置でバイトの書き込みを開始します。

メモ - *WriteBytes を呼び出す前に、OpenFileMode(FILE_MODE_BINARY) を呼び出してファイルを開いてから OpenFile を呼び出します。*

パラメーター nIndex は svString へのインデックスです。nBytes はファイルへ書き込む nIndex の値を超過するバイト数を指定します。nIndex と nBytes を足した長さが svString を超える場合、InstallShield は文字列へのインデックスから文字列の最後へのバイト数のみを書き込みます。例えば、svString の長さが 100 バイトで、nIndex が 50、nBytes が 75 の場合、51 と 100 の間 (75 バイトではなく 50 バイト) のバイトのみがファイルに書き込まれます。

構文

```
WriteBytes (nFile, svString, nIndex, nBytes);
```

パラメーター

テーブル 49 ・ WriteBytes のパラメーター

パラメーター	説明
nFile	バイナリモードで開いたファイルへのファイルハンドルを指定します。
svString	出力ファイルへ書き込むバイトを含む文字列変数を指定します。
nIndex	svString へのインデックスを指定します。この位置で始まるバイトは出力ファイルに書き込まれます。初めのバイトはインデックス位置 0 です。
nBytes	出力ファイルに書き込むバイト数を指定します。

戻り値

テーブル 50 ・ WriteBytes の戻り値

戻り値	説明
X	X は実際に書き込まれたバイト数を表します。
< 0	関数がバイトを書き込むことができなかったことを示します。

WriteBytes の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* WriteBytes 関数のデモンストレーションを行います。
*
* WriteBytes が 呼び出されて会社名をバイナリファイルへ書き込みます。
*
* メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
*      ターゲットシステム上の既存ディレクトリを参照するよう
*      にします。スクリプトがこのファイルに
*     書き込むため、つまり既存データをすべて上書きするため、
*      ファイルを作成するか、この例で使用する既存の
*      ファイルのコピーを作成してください。
*
¥*---------------------------------------------------------------------*/

#define EXAMPLE_DIR  "C:¥¥"
#define EXAMPLE_FILE "ISExampl.bin"
```

```
// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_WriteBytes(HWND);

function ExFn_WriteBytes(hMSI)
   STRING  szQuestion, svCompany[28];
   NUMBER  nFileHandle, nOffset, nIndex, nBytes;
begin

   // ファイルのオープン モードを設定します。
   OpenFileMode (FILE_MODE_BINARY);

   // ファイルを開いてファイル ハンドルを取得します。
   if (OpenFile (nFileHandle, EXAMPLE_DIR, EXAMPLE_FILE)  0) then
      MessageBox(" ファイルを開くことができません。", SEVERE);
      abort;
   endif;

   // ユーザーに対して会社名を問い合わせます。
   szQuestion = " 会社名を入力してください。; 入力できる最大文字数は " +
            "27 文字です。";
   AskText (szQuestion, "My Software Company", svCompany);

   // ファイルポインターをファイルの最初から 15 バイト移動させます。
   nOffset = 15;
   SeekBytes (nFileHandle, nOffset, FILE_BIN_START);

   // 会社名をファイルに書き込みます。
   nIndex = 0;
   nBytes = 27;
   if (WriteBytes (nFileHandle, svCompany, nIndex, nBytes)  0) then
      MessageBox ("WriteBytes が失敗しました。", SEVERE);
   else
      MessageBox (" バイトがファイルに書き込まれました。", INFORMATION);
   endif;

   // ファイルを閉じます。
   CloseFile (nFileHandle);

end;
```

WriteLine

WriteLine は追加モードで開いたテキストファイルへテキストの行を書き込みます。最初に **OpenFileMode** を利用してファイル モードを追加モードに設定しなくてはなりません。そして **WriteLine** を呼び出す前に **CreateFile** を使ってファイルを作成するか、**OpenFile** を使ってファイルを開きます。この関数はファイルの最後に行を配置します。

WriteLine は、末尾に改行文字や復帰改行文字をもつ行を作成します。バイナリ ファイルへ書き込むには、**WriteBytes** を利用します。

注意・この関数は読み取り専用モードで開いたファイルでは利用できません。

構文

WriteLine (nvFileHandle, szLine);

パラメーター

テーブル 51 ・ WriteLine のパラメーター

パラメーター	説明
nvFileHandle	開いたファイルのファイルハンドルを指定します。ハンドルは **OpenFile** または **CreateFile** から取得します。
szLine	ファイルへ書き込むテキストを含む文字列を指定します。 ***メモ・単一引用符を使って文字列を区切ることで、二重引用符を文字列の内部に埋め込むことができます。たとえば、文字列 " この文字列は二重 " 引用符 " を含みます。 " を書き込むには、次のコードを利用することができます。*** WriteLine(nvFileHandle, ' この文字列は二重 " 引用符 " を含みます。'); ***注意・szLine で渡す文字列に newline 文字を埋め込んで複数行を書き込まないで下さい。次のコードは、印刷不可能な文字を "one" および "This" の間に利用して文字列をひとつの行として書き込みます。*** szString = " これは第 1 行 ¥n これは第 2 行 "; WriteLine(nvFileHandle, szString); **WriteLine** への一度の呼び出しで 2 行書き込むには、改行と newline を (この順序で) 埋め込まなくてはなりません。 szString = " これは第 1 行 ¥r¥n これは第 2 行 ";

戻り値

テーブル 52 ・ WriteLine の戻り値

戻り値	説明
0	関数がファイルへ行を書き込んだこと示します。
< 0	関数がファイルへ行を書き込むことができなかったことを示します。

WriteLine の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * CreateFile 関数と WriteLine 関数のデモンストレーションを行います。
 *
 * 文字列を保存するファイルを作成するため Createfile が呼び出されます。その後
 * 文字列は WriteLine 関数によってファイルに書き込まれます。
 *
 * メモ : このスクリプトを実行する前に、プリプロセッサ定数が、
 *      EXAMPLE_DIR を設定して、ターゲット システム上の既存の
 *      ディレクトリを参照するようにします。EXAMPLE_FILE で、
 *      指定したファイルが既に存在する場合
 *      それが上書きされます。
 *
¥*---------------------------------------------------------------------*/

#define EXAMPLE_DIR  "C:¥¥"
#define EXAMPLE_FILE "ISExampl.txt"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_WriteLine(HWND);

function ExFn_WriteLine(hMSI)
   STRING  szTitle, szMsg;
   NUMBER nvFileHandle;
begin

   // ファイル モードを追加に設定します。
   OpenFileMode (FILE_MODE_APPEND);

   // 新規ファイルを作成して開いた状態にします。
   if (CreateFile (nvFileHandle, EXAMPLE_DIR, EXAMPLE_FILE)  0) then
      // エラーを報告します。
      MessageBox ("CreateFile が失敗しました。", SEVERE);
      abort;
   else
      // ファイルへ書き込むメッセージを設定します。
      szMsg = " この行は、サンプル InstallShield スクリプトによって追加されました。";

      // ファイルへメッセージを追加します。
      if (WriteLine(nvFileHandle, szMsg) < 0) then
         // エラーを報告します。
         MessageBox ("WriteLine が失敗しました。", SEVERE);
      else
         // 成功を報告します。
         szTitle = "CreateFile & WriteLine";
         szMsg = " 作成に成功し、%s へ書き込みました。";
         SprintfBox (INFORMATION, szTitle, szMsg, EXAMPLE_FILE);
      endif;

   endif;

   // ファイルを閉じます。
   CloseFile (nvFileHandle);

end;
```

WriteNumberProperty

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

WriteNumberProperty 関数は、値が数値の指定プロパティに値を入力するため、オブジェクトスクリプトで呼び出します。

構文

WriteNumberProperty (nPropertyBag, szPropertyName, nPropertyValue);

パラメーター

テーブル 53 · WriteNumberProperty のパラメーター

パラメーター	説明
nPropertyBag	プロパティの値が保存される、オブジェクトのプロパティバッグオブジェクトへの参照を指定します
szPropertyName	値を入力するプロパティの名前を指定します。
nPropertyValue	プロパティの値を指定します。

戻り値

テーブル 54 · WriteNumberProperty の戻り値

戻り値	説明
0	この関数は常にゼロ (0) を返します。

WriteProfInt

ヒント・すべての INI ファイルへの変更は、IDE の [INI ファイル] ビューで行います。この方法ですべての INI ファイルの変更を処理すると、Windows Installer サービスを通したクリーンアンインストールが可能となります。

WriteProfInt 関数は、整数値をキーに割り当てるプロファイル文字列を挿入または更新して .ini ファイルを修正します。次の重要な点があります。

- Windows によるファイル変更点のキャッシュ方法の為、**WriteProfString** への呼出の後にキャッシュバッファーをフラッシュする必要があります。
- .ini ファイルに加えた変更は、アンインストール用にログ記録することができます。
- 文字列値を .ini ファイルへ書き込むには、代わりに **WriteProfString** を呼び出します。
- **System.ini** ファイルを修正する場合には **AddProfString** 関数と **ReplaceProfString** 関数を利用します。

構文

WriteProfInt (szFileName, szSectionName, szKeyName, iValue);

パラメーター

テーブル 55 ・ WriteProfInt のパラメーター

パラメーター	説明
szFileName	.ini ファイルの名前を指定します。ファイル名が非完全修飾名 (ドライブ指定とパスが *含まれていない*) の場合、InstallShield は Windows フォルダーのファイルを検索します。ファイルが存在しないと、指定のフォルダーに作成されます。パスがファイル名に含まれていない場合、ファイルは Windows フォルダーに作成されます。ファイル名が、存在しないパス名で修飾されていると、WriteProfInt は失敗します。
szSectionName	szKeyName が挿入されるまたは変更される .ini ファイルセクションの名前を指定します。ここで指定するセクション名は、角括弧 ([ ]) *で囲まないでください*。この名前の検索は大文字と小文字を *区別しません*。
szKeyName	更新する固有キーを指定します。更新するキーが指定されたセクションに存在しない場合は、それが作成されます。 ***メモ・** .ini ファイルからセクション全体を削除するには、szKeyName にヌル文字列 ("") を渡します。.ini ファイルのセクションから単数または複数のキーを削除するには、**WriteProfString** を利用します。*
iValue	szKeyName が認識した固有のキーへ割り当てる整数値を指定します。

戻り値

テーブル 56 ・ WriteProfInt の戻り値

戻り値	説明
0	関数が指定した .ini ファイルを更新したことを示します。
< 0	関数が指定した .ini ファイルを更新できなかったことを示します。

追加情報

- WriteProfInt 関数は、Windows API WritePrivateProfileString を利用して .ini ファイルにアクセスします。したがって、その機能は Windows API が提供する機能に制限されます。.ini ファイルに関するさらに詳しい情報は Microsoft マニュアルを参照してください。

- Windows 95 またはそれ以降は、指定したファイルへの変更の書き込みを遅らせる .ini ファイルをキャッシュします。この結果、CopyFile や XCopyFile への呼出など、後に続くファイル操作に影響する場合があります。したがって、直後にファイル操作を行う場合、WriteProfInt を利用した後はキャッシュバッファーをフラッシュする必要があります。単純に、ヌルパラメーターと一緒に WriteProfInt を呼び出して、次に示すように Windows 95 またはそれ以降でデータを .ini ファイルへ直接書き込むよう強制します :

WriteProfInt ("c:¥¥Test.ini", "Windows", "KeyboardDelay", 100); WriteProfInt ("", "", "", 0); // 最初の 3 つのパラメーターのヌル文字列 //CopyFile はこれで更新ファイルにアクセスできます。.CopyFile ("c:¥¥test.ini", "d:¥¥test.ini");

WriteProfInt の例

メモ・基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。

```
/*---------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * WriteProfInt 関数のデモンストレーションを行います。
 *
 * このスクリプトは Windows ディレクトリの初期化ファイルにある
 * 整数値を更新します。ファイルが存在しない場合は、
 * 作成されます。
 *
¥*---------------------------------------------------------------*/

// 初期化ファイル名を定義します。
#define EXAMPLE_INI WINDIR^"ISExampl.ini"

// 初期化アイテムと、その新規値を定義します。
#define   SECTION  "Windows"
#define   KEYNAME  "OurAppVal"
#define   KEYVAL   0

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_WriteProfInt(HWND);

function ExFn_WriteProfInt(hMSI)
begin

  // 初期化ファイルのフィールドを更新します。
  if (WriteProfInt (EXAMPLE_INI, SECTION, KEYNAME, KEYVAL) 0 ) then
    // エラーを報告します。
    SprintfBox (SEVERE, "WriteProfString",
          "%s は更新できませんでした ", EXAMPLE_INI);
  else
    // 成功を報告します。
    SprintfBox (INFORMATION, "WriteProfString",
          "%s が修正されました。", EXAMPLE_INI);
  endif;
```

```
end;
```

WriteProfString

WriteProfString 関数は、.ini ファイルのプロファイル文字列を書き込みます。WriteProfString へ渡される値に従い、セクションの作成、セクション全体の削除、固有の KEY=VALUE エントリの作成、KEY=VALUE エントリの削除、またはキーの値の更新を行うことができます。

次の重要な点があります。

- 整数値を .ini ファイルへ書き込むには、代わりに WriteProfInt を呼び出します。
- System.ini ファイルを修正する場合には AddProfString 関数と ReplaceProfString 関数を利用します。
- .ini ファイルに加えた変更は、アンインストール用にログ記録することができます。ただし、いくつかの重要な制限事項があります。詳細は、「 初期設定 (.ini) ファイルエントリのアンインストール」を参照してください。
- WriteProfString は Windows API WritePrivateProfileString を利用して .ini ファイルにアクセスします。したがって、その機能は Windows API が提供する機能に制限されます。.ini ファイルに関するさらに詳しい情報は Windows プログラムマニュアルを参照してください。
- Windows は指定したファイルへの変更の書き込みを遅らせる .ini ファイルをキャッシュします。この結果、CopyFile や XCopyFile への呼出など、後に続くファイル操作に影響する場合があります。したがって、直後にファイル操作を行う場合、WriteProfString を利用した後はキャッシュバッファーをフラッシュする必要があります。単純に、ヌルパラメーターで WriteProfString を呼び出して、Windows が即座に .ini ファイルへデータの書き込みを行う様に強制します。

```
WriteProfString ("C:¥¥Test.ini", "Windows", "KeyboardDelay", "100");
WriteProfString ("", "", "", ""); // 全 4 つのパラメーター用のヌル文字列
// CopyFile はこれで更新されたファイルにアクセスできます。CopyFile ("C:¥¥Test.ini", "C:¥¥Temp¥¥Test.ini");
```

構文

```
WriteProfString ( szFileName, szSectionName, szKeyName, szValue );
```

パラメーター

テーブル 57 ・ WriteProfString のパラメーター

パラメーター	説明
szFileName	.ini ファイルの名前を指定します。ファイル名が非完全修飾名 (ドライブ指定とパスが*含まれていない*) の場合、InstallShield は Windows フォルダーのファイルを検索します。ファイルが存在しないと、指定のフォルダーに作成されます。パスがファイル名に含まれていない場合、ファイルは Windows フォルダーに作成されます。ファイル名が、存在しないパス名で修飾されていると、WriteProfString は失敗します。
szSectionName	szKeyName を検索する、.ini ファイルセクション名を指定します。ここで指定するセクション名は、角括弧 ([ ]) で*囲まないでください*。この名前の検索は大文字と小文字を*区別しません*。
szKeyName	更新するまたは削除する固有キーを指定します。更新するキーが存在しない場合は、それが作成されます。ここで指定されたキーを削除するには、szValue にヌル文字列 ("") を渡します。szSectionName が指定するセクション全体並びにそこに含まれるすべてのエントリを削除するには、このパラメーターでヌル文字列 ("") を渡します。
szValue	szKeyName が認識した固有のキーへ割り当てる値を指定します。キーを削除するには、このパラメーターにヌル文字列 ("") を渡します。キーそのものは保持してキーの値のみを削除するには、このパラメーターで空白スペースを渡します。

戻り値

テーブル 58 ・ WriteProfString の戻り値

戻り値	説明
0	関数が指定した .ini ファイルへ文字列を書き込んだこと示します。
< 0	関数が指定した .ini ファイルへ文字列を書き込むことができなかったことを示します。

WriteProfString の例

メモ・*基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------------*¥
 *
 * InstallShield スクリプトの例
 *
 * WriteProfString 関数のデモンストレーションを行います。
 *
 * このスクリプトは Windows ディレクトリの初期化ファイルのフィールドを
 * 更新します。ファイルが存在しない場合は、作成されます。
 *
¥*---------------------------------------------------------------------*/

// 初期化ファイル名を定義します。
#define EXAMPLE_INI WINDIR ^ "ISExampl.ini"

// 初期化アイテムと、その新規値を定義します。
#define    SECTION  "Windows"
#define    KEYNAME  "Keyboard"
#define    KEYVALUE "English"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_WriteProfString(HWND);

function ExFn_WriteProfString(hMSI)
begin

  // 初期化ファイルのフィールドを更新します。
  if (WriteProfString (EXAMPLE_INI, SECTION, KEYNAME, KEYVALUE)  0) then
    // エラーを報告します。
    SprintfBox (SEVERE, "WriteProfString",
          "%s は更新できませんでした ", EXAMPLE_INI);
  else
    // 成功を報告します。
    SprintfBox (INFORMATION, "WriteProfString", "%s が修正されました。", EXAMPLE_INI);
  endif;

end;
```

WriteStringProperty

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

WriteStringProperty 関数は、値が文字列の指定プロパティに値を入力するため、オブジェクトスクリプトで呼び出します。

構文

```
WriteStringProperty ( nPropertyBag, szPropertyName, szPropertyValue );
```

パラメーター

テーブル 59 - WriteStringProperty のパラメーター

パラメーター	説明
nPropertyBag	プロパティの値が保存される、オブジェクトのプロパティバッグオブジェクトへの参照を指定します
szPropertyName	値を入力するプロパティの名前を指定します。
szPropertyValue	プロパティの値を指定します。

戻り値

テーブル 60 - WriteStringProperty の戻り値

戻り値	説明
0	この関数は常にゼロ (0) を返します。

XCopyFile

XCopyFile 関数は、1 つまたは複数のファイルをソースディレクトリからターゲットディレクトリにコピーします。この関数は、必要な場合にターゲット ディレクトリを作成およびログ記録します。この関数は、ファイルだけでなく、サブディレクトリもコピーできます。INCLUDE_SUBDIR 定数が nOp パラメーターに渡された場合、**XCopyFile** は必要に応じてターゲット ディレクトリ上にサブディレクトリを作成します。

この関数を使って、ファイルを WINSYSDIR64 に転送する場合、まず WOW64FSREDIRECTION を使用してファイル システムのリダイレクトを無効にする必要があります。無効化をしない場合、WINSYSDIR64 に転送されるファイルは不適切に 32 ビット SysWOW64 システムフォルダーにリダイレクトされます。インストールが利用する可能性のある Windows 機能にはファイル システム リダイレクトを有効にしておく必要があるため、Windows ドキュメンテーションではリダイレクトを無効にするのはそれが必要な場合のみにとどめることが推奨されています。必要なファイルを WINSYSDIR64 へ転送し終わったら、直ちにシステム ファイルのリダイレクトを有効にすることをお勧めします。詳しくは、「64 ビット オペレーティング システムを InstallScript インストールでターゲットにする」を参照してください。

XCopyFile を使ってファイルの名前を変更することはできません。ファイルコピー中にファイル名を変更するには、**CopyFile** 関数を使用してください。

***ヒント -** コピーの最中にステータスダイアログが表示される場合は、**XCopyFile** 関数を呼び出す前に **Disable** 関数を利用して [キャンセル] ボタンを無効にすることが、強く推奨されます。[キャンセル] ボタンを無効にしなかった場合で、エンド ユーザーがファイルのコピー操作を途中でキャンセルした場合、OnCancelling イベントハンドラーは呼び出されません。その代わり、ファイルのコピー操作はエラー コードを戻します。この場合、スクリプトによって適切なイベントを呼び出してからファイルのコピー操作を再開する必要があります。Enable と Disable 関数を利用して [キャンセル] ボタンを有効および無効にすることができます。*

メモ・完全修飾でないファイル名を使用して、XcopyFile を使用中に SRCDIR と TARGETDIR の値を設定する場合、XCopyFile を呼び出す前に VarSave を使って現在の値を保存し、それから VarRestore を使って復元してください。

構文

```
XCopyFile ( szSrcFile, szTargetPath, nOp );
```

パラメーター

テーブル 61 ・ XCopyFile のパラメーター

パラメーター	説明
szSrcFile	コピーするファイルを指定します。指定のファイルが完全修飾名、つまりパスが含まれる場合、**XCopyFile** はファイルを指定の場所からコピーします。szSrcFile に完全修飾名ではないファイル名が含まれ、パス情報がない場合、**XCopyFile** は SRCDIR システム変数によって識別されたディレクトリからコピーします。複数のファイルをコピーするには、このパラメーターでワイルドカード文字を使ってください。 このパラメーターでは、有効な URL を指定できます。CGI または ASP 要求 (たとえば、"http://www.mydomain.net/login.asp?name=Me&pwd=wow") を渡す場合、その応答は szTargetPath パラメーターで指定されたファイルへ送られます。URL を渡すとき、ワイルドカード文字を含まないで下さい。URL が有効かどうかを確認するには、次を呼び出します : Is (VALID_PATH, szURL);
szTargetPath	szSrcFile で指定されたファイルのコピー先ディレクトリを指定します。このパラメーターにヌル文字列 ("") が含まれる場合、ファイルはシステム変数 TARGETDIR (InstallScript インストールの場合) または INSTALLDIR (基本の MSI および InstallScript MSI インストールの場合) が指定するディレクトリにコピーされます。 このパラメーターで URL は指定できません。ここで指定しても、関数が失敗して ISERR_INVALID_ARG を戻します。

テーブル 61 ・ XCopyFile のパラメーター (続き)

パラメーター	説明
nOp	実行するコピー操作の種類を指定します。このパラメーターに、あらかじめ定義されている以下の定数のうちの 1 つを渡します。複数のオプションを指定するには、定数と OR (|) 演算子を組み合わせます。 ・ COMP_NORMAL─ ファイルをターゲットシステムにコピーし、既に存在する同じ名前のファイルをその日付、時間、バージョン情報にかかわらず、更新します。 ・ COMP_UPDATE_SAME─ ソースファイルの日付、時間、バージョンが、ターゲットファイルと同じであっても更新します。この定数と一緒に、COMP_UPDATE_DATE か COMP_UPDATE_VERSION を指定する必要があります。怠った場合、この定数は無視されます。 ・ COMP_UPDATE_DATE─ ファイルの日時に基づいてファイルを更新します。この定数は、ソースファイルがターゲットファイルより新しい場合にファイルを更新します。 ・ COMP_UPDATE_VERSION─ ファイル バージョンに基づいてファイルを更新します。この定数は、ソースファイルがターゲットファイルより新しい場合にファイルを更新します。ソースファイルにもターゲットファイルにもファイルバージョンが存在しない場合には、その日付と時間を比較します。どちらか 1 つのファイルのバージョンが存在しない場合、InstallShield はバージョン情報を含むファイルが新しいファイルとみなされます。 ・ SELFREGISTER─「非バッチ メソッド」で自己登録ファイルをインストールしたときに、自己登録処理をただちに実行します。 Enable (SELFREGISTERBATCH) を呼び出すと、自己登録ファイルは登録キューに配置されます。「バッチメソッド」で自動登録ファイルをインストールした場合には、Do (SELFREGISTRATIONPROCESS) を呼び出したときにファイルは登録されます。 SELFREGISTER オプションと SHAREDFILE オプションは、常にビット単位 OR 演算子 (|) でつなげて一緒に使用します。 ・ SHAREDFILE─**XCopyFile** がすべてのファイルを共有ファイルとして扱うようにし、Windows またはシステムが再起動されたときに更新するロックされた .dll および .exe ファイルを記録することで、ロックされたファイルと共有ファイルの処理を組み合わせます。詳細については、「RebootDialog」および「SdFinishReboot」を参照してください。 SHAREDFILE オプションを使うと、**XCopyFile** はすべてのファイルを共有ファイルとして処理し、ファイルがターゲット ディレクトリに存在して、0 より大きい参照カウンターを持つ場合、レジストリ参照カウントを 1 増やします。共有ファイルがターゲットディレクトリに存在せず、参照カウンターがない場合、インストール はカウンターを作成して 1 に設定します。共有ファイルがターゲット ディレクトリに存在するが、参照カウンターがない場合、インストール はカウンターを作成して、アンインストール中に誤って削除されないように、これを 2 に初期化します。

テーブル 61 ・ XCopyFile のパラメーター (続き)

パラメーター	説明
	• LOCKEDFILE—Windows またはシステムを再起動したときに、更新するロックされた .dll および .exe ファイルが **XCopyFile** 関数によって記録されます。ロックされたファイルとは、インストールによってファイルのアクセスまたは更新が試行されたときに、アプリケーションまたはシステムで使用されているファイルのことです。LOCKEDFILE オプションは、SHAREDFILE と同じように機能します。ただし、LOCKEDFILE ではレジストリエントリを作成したりレジストリ参照カウンターを変更することはできません。SHAREDFILE オプションを使用している時に LOCKEDFILE オプションを使用することはできません。スクリプトの作成者がレジストリエントリおよび参照カウンターを必要としない非共有ファイル (シェル拡張子など) もいくつかあります。これらのファイルはアプリケーションによってアンインストールされないかぎり、絶対にアンインストールしないでください。LOCKEDFILE を使用すると、ロックされた非共有ファイルを **XCopyFile** で処理することができます。 • EXCLUDE_SUBDIR— ソースパスに含まれるサブディレクトリを除外します。空のサブディレクトリはコピーされません。 • INCLUDE_SUBDIR— ソースパスのサブディレクトリもコピーされなければならないように指定します。 • CLEAR_FILE_ATTR— コピーされたファイルの属性をクリアするように指定します。このパラメーターを使用すると、ターゲット フォルダーのファイルには、デフォルトで設定されたファイルの属性がどれも適用されません。このパラメーターを使用しない場合、**XCopyFile** はコピーされたファイルに対して元のファイルの属性を使用します。

戻り値

テーブル 62 - XCopyFile の戻り値

戻り値	説明
0	関数の実行によってファイルが無事コピーされたことを示します。
ISERR_INVALID_ARG	無効な引数が関数へ渡されたことを示します。
その他すべての負の値	関数がファイルをコピーできなかったことを示します。 大きな負の戻り値と関連付けられたエラーメッセージテキストを取得することができます。たとえば、FormatMessage を呼び出した場合の -2147024891 (0x80070005) です。

追加情報

WriteProfString で .ini ファイルを変更後、**XCopyFile** を利用する前にキャッシュのバッファーをフラッシュしなくてはなりません。すべての .ini ファイルはキャッシュされます。この動作の結果、指定されたファイルへの変更の書き込みが遅れることがあります。これは、後続のファイル操作を妨げることがあります。単純にヌルパラメーターで **WriteProfString** を呼び出して、以下の通り Windows が即座に .ini ファイルへデータの書き込みを行う様に強制します。

```
WriteProfString ("C:¥¥Test.ini", "Windows", "KeyboardDelay", "100");
// 全 4 つのパラメーター用のヌル文字列 ("")
WriteProfString ("", "", "", "");
// XCopyFile はこれで更新されたファイルにアクセスできます。
XCopyFile ("C:¥¥Test.ini", "C:¥¥Temp", EXCLUDE_SUBDIR);
```

XCopyFile の例

メモ - *基本の MSI セットアップでこの関数を呼び出すには、まずエントリーポイント関数用のカスタムアクションを作成し、シーケンスで、またはダイアログのコントロールイベントの結果としてカスタムアクションを実行してから、リリースをビルドします。*

```
/*---------------------------------------------------------------*¥
*
* InstallShield スクリプトの例
*
* XCopyFile 関数のデモンストレーションを行います。
*
* XCopyFile への最初の呼出は、日付、時刻、バージョンに関わらず
* すべてのテキストファイルをコピーします。
*
* 2 番目の呼出ではプログラムファイルをコピーし、これらのファイルを配置するための
* サブディレクトリを作成します。
*
* 3 番目の呼び出しでは、日付に基づいてテンプレートファイルをコピーし、
* ソースファイルと同じまたは以前の日付を持つターゲットファイルを
* 上書きします。
*
* 4 番目の呼び出しでは、バージョン番号に基づいてサンプルファイルをコピーし、
```

```
＊古いバージョン番号を持つターゲットファイルを上書きします。
＊
＊
＊メモ : このスクリプトを適切に実行するため、
＊　　プリプロセッサ定数をターゲットシステムの有効なファイル名
＊　　とパスへ設定しなくてはなりません。
＊
¥*---------------------------------------------------------------*/

#define SDIR          "C:¥¥ISExampl¥¥Source¥¥"
#define SDIR_PROGRAM  "C:¥¥ISExampl¥¥Source¥¥Program¥¥"
#define SDIR_TEMPLATE "C:¥¥ISExampl¥¥Source¥¥Template¥¥"
#define SDIR_SAMPLES  "C:¥¥ISExampl¥¥Source¥¥Samples¥¥"
#define TDIR          "C:¥¥ISExampl¥¥Target¥¥"

// ビルトイン InstallScript 関数プロトタイプに Ifx.h を含みます。
#include "Ifx.h"

    export prototype ExFn_XCopyFile(HWND);

function ExFn_XCopyFile(hMSI)
  STRING szSrcFile;
  NUMBER nResult;
begin

  // ソースファイルのファイル指定へ変数を設定します。
  szSrcFile = "*.txt";

  // ソースディレクトリのすべてのテキストファイルを
  // ターゲットディレクトリへコピーします。
  if (XCopyFile (SDIR ^ szSrcFile, TDIR ^ "*.*", COMP_NORMAL) < 0) then
    MessageBox ("XCopyFile が失敗しました ", SEVERE);
  else
    MessageBox (" テキストファイルがコピーされました。", INFORMATION);
  endif;

  // 新しい変数を設定します。
  szSrcFile = "*.*";

  // ソースサブディレクトリにあるすべてのプログラムファイルを
  // ターゲットディレクトリのサブディレクトリへコピーします。

  if (XCopyFile (SDIR_PROGRAM ^ szSrcFile, TDIR ^ "PROGRAM" ^ "*.*",
    INCLUDE_SUBDIR) < 0) then
    MessageBox ("XCopyFile が失敗しました ", SEVERE);
  else
    MessageBox (" プログラムファイルがコピーされました。", INFORMATION);
  endif;

  // ソースサブディレクトリにあるすべてのテンプレートを
  // ターゲットディレクトリのサブディレクトリへコピーします。
  if (XCopyFile (SDIR_TEMPLATE ^ szSrcFile, TDIR ^ "TEMPLATE" ^ "*.*",
    COMP_UPDATE_SAME | COMP_UPDATE_DATE)  0) then
    MessageBox ("XCopyFile が失敗しました ", SEVERE);
  else
    MessageBox (" テンプレート ファイルがコピーされました。", INFORMATION);
  endif;

  // ソース サブディレクトリ内のすべてのサンプルファイルを
```

```
    // ターゲットディレクトリのサブディレクトリへコピーします。
    if (XCopyFile (SDIR_SAMPLES ^ szSrcFile, TDIR ^ "SAMPLES" ^ "*.*",
        COMP_UPDATE_VERSION) < 0) then
        MessageBox ("XCopyFile が失敗しました ", SEVERE);
    else
        MessageBox (" サンプルファイルがコピーされました。", INFORMATION);
    endif;

end;
```

索引

記号

数字

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

あ

き

く

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

な

に

ね

は

ひ

ふ

へ

ほ

め

も

ゆ

よ

り

る

れ

ろ