Inno Setup 5.0.7

中文帮助

一凡整理于 2005.2.7

QQ：51795745　　Email：xsj_by@163.com

目 录

第一部分 简介

1、Inno Setup 是什么？

Inno Setup 是一个免费的 Windows 安装程序制作软件。第一次发表是在 1997 年，Inno Setup 今天在功能设置和稳定性上的竞争力可能已经超过一些商业的安装程序制作软件。
关键功能:
★ 支持现在所有正在使用的 32 位 Windows 版本: Windows 95，98，2000，Server 2003，XP，Me，NT 4.0 (不需要服务包)。
★ 支持创建单个 EXE 格式的安装程序，使你的程序可以很方便地在网络上发表。同时也支持磁盘延伸。
★ 标准的 Windows 2000/XP 样式向导界面。
★ 定制安装类型，例如：完整安装，最小安装，自定义安装。
★ 完整的卸载功能。
★ 文件安装:
包括完全的“压缩”支持，bzip2 和 7-Zip LZMA 文件压缩。安装程序可以比较文件版本信息，替换正在使用的文件，使用共享文件计数，注册 DLL/OCX 和类型库，以及安装字体。
★ 可以在任意地方创快捷方式建，包括开始菜单和桌面。
★ 创建注册表和 .INI 项目。
★ 完整的 Pascal 脚本引擎。
★ 支持 multilingualLanguagessection 安装。
★ 支持密码和加密安装。
★ 后台安装和后台卸载。
★ 全部源代码公开 (Borland Delphi 2.0-5.0)。

2、文档约定

Windows 98/NT 4+这是“Windows 98，2000，XP，NT 4.0，Me 以及更高版本”的简写。
Windows NT 只要是 Windows NT 就可以，它包括 Windows 2000 和 XP (就是 NT 5)，除非另外说明。
等宽文本 当你在帮助中看到等宽文本，它表示脚本在中输入的引用的文字。

第二部分 怎么使用

1、创建安装程序

安装程序用编译脚本的方式创建，脚本其实就是一个类似 .INI 文件格式的 ASCII 码文本文件。 (它不象你想象的那么复杂!)
脚本用一个“.iss” (表示 Inno Setup Script) 的扩展名。脚本控制着安装程序的所有方面。由它指定哪些文件将被安装到什么地方，在哪里创建快捷方式，且被命名为什么。
脚本文件一般可以用安装程序编译器程序内置的编辑器进行编辑。在你编写完脚本后，下一

个最终步骤就是选择安装程序编译器中的“编译”。创建完成后，就可以运行根据你脚本编译的安装程序了。按默认，这个安装程序创建在包含脚本文件目录下的名为“输出”目录中。如果你想看看它是怎样工作的，启动安装程序编译器，单击“文件 | 打开”，并选择位于 Inno Setup 安装目录下的 Samples 子目录中的一个脚本文件。(你也可以将这些示例脚本作为你自己编写脚本的模板。)

2、脚本格式概述

Inno Setup 准备了一些段。每个段控制一个不同方面的安装程序部分。每个段用包含在括号 [] 中的段名开始，每个段里面是一些相关的条目。
其中有两种不能类型的段：有些就象 [Setup] 段，条目包含指示名和值 (格式为 Directive=Value)，还有一些就象 [Files] 段，条目被参数分隔。
这里是一个例子:

```
[Setup]
AppName=My Program
[Files]
Source: "MYPROG.EXE"; DestDir: "{app}"
```

注意，在脚本中指定多个相同名字的段是合法的。
你可以通过在行起始位置加个分号“;”在脚本中写入“注释” (编译时被编译时忽略)。例如:

```
; 这是一条注释，放在这里只是要提醒我自己...
```

支持 A C-like #include 指示，从个别文件放入行到 #include 指示位置的脚本。语法是:

```
#include "filename.txt"
```

如果文件名中未提供完整的路径，编译将在包含 #include 指示的同一目录中查找。文件名用“compiler:”作为前缀的场合中，在编译器目录中查找文件。

3、段中参数

所有脚本中的段，除 [Setup]、[Messages]、[CustomMessages] 和 [LangOptions] 段，包含的行中可有多个各自的参数。下列是 [Files] 段中的一个举例:

```
[Files]
Source: "MYPROG.EXE"; DestDir: "{app}"
Source: "MYPROG.HLP"; DestDir: "{app}"
Source: "README.TXT"; DestDir: "{app}"; Flags: isreadme
```

每个参数都由一个名字组成，然后跟随一个冒号，然后是一个值。除非另外说明，如果参数未指定，将设定为一个默认值。一行中多个参数用分号隔开，并且可以以任何次序列出。
参数的值如果包含一个用户定义的字符串时，一般来说用双引号 (") 包含，例如象文件名。引号使用不是必需的，但这样做可能会在值中的首位或末位被加入空格，以及分号和双引号。
在要引用的值中使用一个双引号字符，用两个连续的双引号字符，例如:

```
"This "" contains "" embedded "" quotes"
```

安装程序编译器会将它视作:

```
This " contains " embedded " quotes
```

如果你希望参数值是一个单个双引号字符，用四个双引号字符: """"。外面的两个用于包含引用的字符；内部两个写入单个的双引号字符。

4、常量

脚本中的项目大部分可以嵌入常量。这些预定义的字符被包含在括弧 { } 中。安装程序或卸载程序会根据用户选择和系统配置将这些常量翻译为文字值。例如，{win} 在大部分系统中会被翻译为“C:\WINDOWS”。

字符“{”视作为常量开始。如果你想将它作为实际字符使用，你必须使用两个连续的“{”字符。(对于“}”则不需要。)

当在常量后面直接跟随一个反斜杠时，如果常量的值末端已经包含了一个反斜杠号，安装程序或卸载程序将自动删除该反斜杠号。 因此，如果一个特殊常量值是“C:\”，{constantname}\file 将翻译为“C:\file”，而不是“C:\\file”。如果你想防止意外，将反斜框放入 { } 字符中，例如，{app}{\}。

下面是支持的常量列表。

目录常量

{app}

用户在安装向导中的选择目标位置页中选定的应用程序目录。

例如: 如果你在项目中指定了 {app}\MYPROG.EXE，用户选择了“C:\MYPROG”作为应用程序目录，安装程序将该文件安装到“C:\MYPROG\MYPROG.EXE”。

{win}

系统的 Windows 目录。

例如: 如果你在条目中使用了 {win}\MYPROG.INI，且系统的 Windows 目录是“C:\WINDOWS”安装程序或卸载程序将它传送到“C:\WINDOWS\MYPROG.INI”。

{sys}

系统的 Windows System 目录 (在 NT 平台上是 System32)。

例如: 如果你在条目中使用了 {sys}\CTL3D32.DLL，且系统的 Windows System 目录是“ C:\WINDOWS\SYSTEM ”， 安装程序或卸载程序将它传送到“C:\WINDOWS\SYSTEM\CTL3D32.DLL”。

{src}

安装程序文件所在目录。

例如: 你在条目中使用了 {src}\MYPROG.EXE，且用户正在从“S:\”进行安装，安装程序将它传送到“S:\MYPROG.EXE”。

{sd}

Windows 系统所在的驱动器。一般来说是“C:”。在 Windows NT 平台，这个目录常量等同于 SystemDrive 环境变量。

{pf}

程序文件位置。系统的 Program Files 目录的路径，一般来说是“C:\Program Files”。

{cf}

公共文件目录。系统的 Common Files 目录路径，一般来说是“C:\Program Files\Common Files”。

{tmp}

用于安装程序或卸载程序的临时目录。这不是用户的 TEMP 环境变量值。它是在安装程序启动后在用户临时目录中创建的子目录 (象名为“C:\WINDOWS\TEMP\IS-xxxxx.tmp”)。目录中的所有文件和子目录在安装程序或卸载程序退出时删除。在安装时，这主要用于提取在 [Run] 段运行、但安装后不再需要的文件。

{fonts}

字体目录。通常是在 Windows 下面的名字“FONTS”的目录。
{dao}
DAO 目录，等同于 {cf}\Microsoft Shared\DAO。
外壳文件夹常量
Inno Setup 支持其它目录常量设置，作为外壳文件夹常量引用。它们可以与其它目录常量相同方法使用。
下面的“user”常量引用到当前登录的用户配置文件，“common”常量引用到所有用户配置文件。
除非有另外的注释，外壳文件夹常量工作于 Inno Setup 支持的所有 Windows 版本，包括 Windows 95 和 NT 4.0。
* = 如果登录的用户缺少管理员权限，或操作系统是 Windwos 95/98/Me，“common”结构映射到“user”结构。
{group}
开始菜单文件夹路径，由用户在安装程序的选择开始菜单文件夹向导页中选定。在 Windows NT/2000/XP，这个文件夹总是创建在所有用户配置文件下，除非非用户安装程序的用户没有管理员权限，这种情况下它将创建在用户配置文件下。
{localappdata}
本地应用程序数据文件夹。
{sendto}
当前用户的 Send To 文件夹路径。(这里不是指公共 Send To 文件夹。)
{userappdata} 和 {commonappdata}
应用程序数据文件夹路径。
{userdesktop} 和 {commondesktop} *
桌面文件夹路径。
{userdocs} 和 {commondocs}
我的文档 (My Documents) 文件夹路径 (在 NT 4.0，私人文件夹)。
{userfavorites} 和 {commonfavorites} *
收藏夹文件夹路径。这些常量设置必须 至少“4.1, 4” MinVersion 设置。只在 Windows 2000 和更高版本支持 {commonfavorites}；如果在先前的 Windows 版本中使用，它将翻译为等同于 {userfavorites} 目录。
{userprograms} 和 {commonprograms} *
开始菜单中程序文件夹路径。
{userstartmenu} 和 {commonstartmenu} *
开始菜单顶层路径。
{userstartup} 和 {commonstartup} *
开始菜单启动文件夹路径。
{usertemplates} 和 {commontemplates} *
模板文件夹路径。仅在 Windows 2000 和更高版本支持 {commontemplates}，如果使用的是先前的 Windows 版本，它将被翻译为等同于 {usertemplates} 目录。
其它常量
{\}
反斜杠字符。查阅本页顶部的注释获取使用 {\} 和只使用一个 \ 字符之间的差异。
{%NAME|DefaultValue}
嵌入一个环境变量值。

★ NAME 用于指定要使用的环境变量的名字。
★ DefaultValue 确定如果指定的变量在用户系统中不存在时置入的字符串。
★ 如果你想在常量内部包含一个逗号、竖条（“|”），或括弧后半部（“}”），你必须通过“%-encoding.”先用一个“%”字符然后跟随它的两上数字的十六进制代码替换，从而避开它。逗号是“%2c”，竖条是“%7c”括弧后半部是“%7d”。如果你想实际使用“%”，使用“%25”。
★ NAME 和 DefaultValue 可以包含常量。注意，你不需要将常量中的括弧替换成上面所说的字符；上面所说的括弧后半部只在使用于其它地方时需要替换。
示例:
{%COMSPEC}
{%PROMPT|PG}
{cmd}
系统标准命令解释器的完整路径名。在 Windows NT/2000/XP，是 Windows\System32\cmd.exe。在 Windows 95/98/Me，是 Windows\COMMAND.COM。注意当展开这个常量时 COMSPEC 环境变量不使用。
{computername}
正在运行安装程序或卸载程序的电脑名 (等同于由 GetComputerName 函数返回的值)。
{drive:Path}
从指定的路径中提取并返回驱动器卷标和冒号 (例如“C:”)在 UNC 路径的场合中，它返回服务器和共享名 (例如“\\SERVER\SHARE”)。
★ Path 指定路径。
★ 如果你想在常量内部包含一个逗号、竖条（“|”），或括弧后半部（“}”），你必须通过“%-encoding.”先用一个“%”字符然后跟随它的两上数字的十六进制代码替换，从而避开它。逗号是“%2c”，竖条是“%7c”括弧后半部是“%7d”。如果你想实际使用“%”，使用“%25”。
★ 可以包含常量。注意，你不需要将常量中的括弧替换成上面所说的字符；上面所说的括弧后半部只在使用于其它地方时需要替换。
示例:
{drive:{src}}
{drive:c:\path\file}
{drive:\\server\share\path\file}
{groupname}
用户在安装程序向导页的选择开始菜单文件夹中选定的文件夹名。它不同于 {group}，只有名字，不包含路径。
{hwnd}
(特殊用途) 转换为安装程序的背景窗口句柄。
{wizardhwnd}
(特殊用途) 转换为安装程序的向导窗口句柄。如果向导窗口句柄在翻译完成时不能用，这个句柄设置为“0”。
{ini:Filename,Section,Key|DefaultValue}
从 .INI 文件插入一个值。
★ Filename 指定要读取的 .INI 文件的名字。
★ Section 指定读取的段名。
★ Key 指定读取的键名。

★ DefaultValue 确定如果指定的键不存在时要插入的字符。
★ 如果你想在常量内部包含一个逗号、竖条（“|”），或括弧后半部（“}”），你必须通过“%-encoding.”先用一个“%”字符然后跟随它的两上数字的十六进制代码替换，从而避开它。逗号是“%2c”，竖条是“%7c”括弧后半部是“%7d”。如果你想实际使用“%”，使用“%25”。
★ Filename，Section 和 Key 可以包含常量。注意，你不需要将常量中的括弧替换成上面所说的字符；上面所说的括弧后半部只在使用于其它地方时需要替换。
示例:
{ini:{win}\MyProg.ini,Settings,Path|{pf}\My Program}

{language}
选定语言的内部名字。查阅 [Languages] 段帮助文档获取更多信息。
{cm:MessageName}
{cm:MessageName,Arguments}
根据活动语言嵌入一个自定义消息值。
★ MessageName 用于指定要读取的自定义消息名。查阅 [CustomMessages] 段帮助文档获取更多信息。
★ Arguments 可随意在消息值中指定逗号分隔的声明列表。
★ 如果你想在常量内部包含一个逗号，垂直条（“|”），或括号（“}”），你必须使用“%-encoding.”避开它，用“%”字符，后面跟随它的两位数十六进制代码替换。逗号是“%2c”，垂直条是“%7c”，括号是“%7d”，如果你想包含一个实际的“%”字符，用“%25”。
★ 每个 Arguments 中的声明可以包含常量。注意，你不需要避开上面描述的常量中的括号，只有在别处使用这种括号时需要避开。
示例:
{cm:LaunchProgram,Inno Setup}
如果活动语言是简体中文，上面的示例被翻译为“运行 Inno Setup”。
{reg:HKxx\SubkeyName,ValueName|DefaultValue}
插入一个注册表值。
★ HKxx 指定注册表根键；查阅 [Registry] 段帮助文档获取可用根键列表。
★ SubkeyName 指定要读取的子键名。
★ ValueName 指定要读取的值名；如果你想读取键的“默认”值，将 ValueName 留空。
★ DefaultValue 确定在指定的注册表值不存在，或不是一个字符串类型的值 (REG_SZ 或 REG_EXPAND_SZ) 时要插入的字符。
★ 如果你想在常量内部包含一个逗号、竖条（“|”），或括弧后半部（“}”），你必须通过“%-encoding.”先用一个“%”字符然后跟随它的两上数字的十六进制代码替换，从而避开它。逗号是“%2c”，竖条是“%7c”括弧后半部是“%7d”。如果你想实际使用“%”，使用“%25”。
★ SubkeyName，ValueName 和 DefaultValue 可以包含常量。注意，你不需要将常量中的括弧替换成上面所说的字符；上面所说的括弧后半部只在使用于其它地方时需要替换。
示例:
{reg:HKLM\Software\My Program,Path|{pf}\My Program}
{param:ParamName|DefaultValue}
插入一个命令行参数值。
★ ParamName 指定要读取的命令行参数名。

★ DefaultValue 确定如果指定的命令行参数不存在，或它的值不能确定时要插入的字符。
★ 如果你想在常量内部包含一个逗号、竖条（“|”），或括弧后半部（“}”），你必须通过“%-encoding.”先用一个“%”字符然后跟随它的两上数字的十六进制代码替换，从而避开它。逗号是“%2c”，竖条是“%7c”括弧后半部是“%7d”。如果你想实际使用“%”，使用“%25”。
★ ParamName 和 DefaultValue 可以包含常量。注意，你不需要将常量中的括弧替换成上面所说的字符；上面所说的括弧后半部只在使用于其它地方时需要替换。

示例:
{param:Path|{pf}\My Program}
如果指定命令行 /Path="c:\My Program"，上面的例子翻译为 c:\My Program。
{srcexe}
安装程序文件的完整路径名，例如“C:\SETUP.EXE”。
{uninstallexe}
由安装程序提取的卸载程序的完整路径名，例如“C:\Program Files\My Program\unins000.exe”。这个常量一般用于在 [Icons] 段条目创建一个卸载图标。它只在 Uninstallable 设为 yes (默认设置) 时有效。0
{sysuserinfoname}
{sysuserinfoorg}
Windows 已许可的名字和组织，这个信息从注册表中读取。
{userinfoname}
{userinfoorg}
{userinfoserial}
用户在用户信息向导页 (可以通过 UserInfoPage 指示来启用) 中分别输入的名字，组织和序列号。一般来说，这些常量用于在 [Registry] 或 [INI] 条目中保存它们以后要使用的值。
{username}
正在运行安装程序或卸载程序的用户的名字 (也可以用 GetUserName 函数返回)。

5、公共参数

有三个可选的被所有段条目支持的参数，它们是:
Languages
描述:
一个用空格分隔的语言名列表，告诉安装程序条目属于哪种语言。如果最终用户从列表中选择了一个语言，该条目就执行(例如: 安装文件)。
不带 Languages 参数的条目总是安装，除非其它参数中有限制。
示例:
Languages: en nl
除用空格将它们隔开外，你也可以使用 boolean 表达式。查阅组件和任务参数获取 boolean 表达式的示例。
MinVersion
描述:
指定条目要进行处理的最小 Windows 版本 Windows NT 版本。如果你在版本中的一个使用“0”，那么条目将不在平台中进行处理。构建号和/或安全服务包级别可能包含在版

本号中。这将忽略任何在脚本 [Setup] 段中的 MinVersion 指示。
不带 MinVersion 参数的条目总是安装，除非其它参数中有限制。

示例:

MinVersion: 4.0,4.0

OnlyBelowVersion

描述:

基本上是和 MinVersion 相对。指定条目不进行处理的最小 Windows 和 Windows NT 版本。例如，如果你加入 4.1,5.0，用户正在运行 Windows 95 或 NT 4.0，那么条目将进行处理，但如果用户正运行于 Windows 98 (报告它的版本是 4.1) 或 Windows 2000 (报告它的版本是 NT 5.0)，它将不进行处理。版本中的一个放入“0”表示不受版本上限。 构建号和/或安全服务包级别可能包含在版本号中。这将忽略任何在脚本 [Setup] 段中的 MinVersion 指示。
不带 OnlyBelowVersion 参数的条目总是安装，除非其它参数中有限制。

示例:

OnlyBelowVersion: 4.1,5.0

6、组件和任务参数

这里有两个可选的参数，被除 [Types]、[Components] 和 [Tasks] 段以外所有其它段中的条目支持。它们是:

Components

描述:

用空格分隔的组件名列表，告诉安装程序条目属于哪个组件。如果最终用户从列表中选择了一个组件，那么该条目就进行处理 (例如: 安装文件)。

不带组件参数的条目总是安装，除非其它参数对其有限制。

示例:

[Files]

Source: "MYPROG.EXE"; DestDir: "{app}"; Components: main

Source: "MYPROG.HLP"; DestDir: "{app}"; Components: help

Source: "README.TXT"; DestDir: "{app}"

Tasks

描述:

用空格分隔的任务名列表，告诉安装程序条目属于哪个任务。如果最终用户从列表中选择了一个任务，那么该条目就进行处理 (例如: 安装文件)。
不带任务参数的条目总是安装，除非其它参数对其有限制。

“不创建任何快捷方式”选项框不控制 [Icons] 带任务参数的条目 (它们有自己的选项框)。因此，如果你已经定义了带任务参数的快捷方式，安装程序将改变“不创建任何快捷方式”的文字到“不创建开始菜单文件夹”。

示例:

[Icons]

Name: "{group}\My Program"; Filename: "{app}\MyProg.exe"; Components: main; Tasks: startmenu

Name: "{group}\My Program Help"; Filename: "{app}\MyProg.hlp"; Components: help;

Tasks: startmenu

Name: "{userdesktop}\My Program"; Filename: "{app}\MyProg.exe"; Components: main; Tasks: desktopicon

除用空格分隔外，你也可以使用 boolean 表达式作为组件和任务参数。支持包含 not、and 和 or 操作。例如:

[Components]

Name: a; Description: a

Name: b; Description: b

[Tasks]

Name: p; Description: a or b; Components: a or b

Name: q; Description: a and b; Components: a and b

Name: r; Description: not a or b; Components: not a or b

Name: s; Description: not (a or b); Components: not (a or b)

Name: t; Description: a or b - old style; Components: a b

7、安装脚本段

[Setup] 段

这个段包含用于安装程序和卸载程序的全局设置。某些提示对于你创建的任何安装程序都是必需的。这是 [Setup] 段的一个示例:

[Setup]
AppName=My Program
AppVerName=My Program version 1.4
DefaultDirName={pf}\My Program
DefaultGroupName=My Program

下列指示可以放置到 [Setup] 段中:

(粗体为必需项)

编译器相关

★ Compression
★ DiskClusterSize
★ DiskSliceSize
★ DiskSpanning
★ Encryption
★ InternalCompressLevel
★ MergeDuplicateFiles
★ OutputBaseFilename
★ OutputDir
★ OutputManifestFile
★ ReserveBytes
★ SlicesPerDisk
★ SolidCompression
★ SourceDir

★ UseSetupLdr
★ VersionInfoCompany
★ VersionInfoDescription
★ VersionInfoTextVersion
★ VersionInfoVersion

安装程序相关

功能:这些指示影响安装程序的操作，或保存和被卸载程序使用。

★ AllowCancelDuringInstall
★ AllowNoIcons
★ AllowRootDirectory
★ AllowUNCPath
★ AlwaysRestart
★ AlwaysShowComponentsList
★ AlwaysShowDirOnReadyPage
★ AlwaysShowGroupOnReadyPage
★ AlwaysUsePersonalGroup
★ AppendDefaultDirName
★ AppendDefaultGroupName
★ AppComments
★ AppContact
★ AppId
★ AppModifyPath
★ AppMutex
★ AppName
★ AppPublisher
★ AppPublisherURL
★ AppReadmeFile
★ AppSupportURL
★ AppUpdatesURL
★ AppVersion
★ AppVerName
★ ChangesAssociations
★ ChangesEnvironment
★ CreateAppDir
★ CreateUninstallRegKey
★ DefaultDirName
★ DefaultGroupName
★ DefaultUserInfoName
★ DefaultUserInfoOrg
★ DefaultUserInfoSerial
★ DirExistsWarning
★ DisableDirPage
★ DisableFinishedPage
★ DisableProgramGroupPage

★ DisableReadyMemo
★ DisableReadyPage
★ DisableStartupPrompt
★ EnableDirDoesntExistWarning
★ ExtraDiskSpaceRequired
★ InfoAfterFile
★ InfoBeforeFile
★ LanguageDetectionMethod
★ LicenseFile
★ MinVersion
★ OnlyBelowVersion
★ Password
★ PrivilegesRequired
★ RestartIfNeededByRun
★ ShowLanguageDialog
★ TimeStampRounding
★ TimeStampsInUTC
★ Uninstallable
★ UninstallDisplayIcon
★ UninstallDisplayName
★ UninstallFilesDir
★ UninstallLogMode
★ UninstallRestartComputer
★ UpdateUninstallLogAppName
★ UsePreviousAppDir
★ UsePreviousGroup
★ UsePreviousSetupType
★ UsePreviousTasks
★ UsePreviousUserInfo
★ UserInfoPage

修饰:这些指示只用于安装程序的显示目的。

★ AppCopyright
★ BackColor
★ BackColor2
★ BackColorDirection
★ BackSolid
★ FlatComponentsList
★ SetupIconFile
★ ShowComponentSizes
★ ShowTasksTreeLines
★ UninstallStyle
★ WindowShowCaption
★ WindowStartMaximized
★ WindowResizable

★ WindowVisible
★ WizardImageBackColor
★ WizardImageFile
★ WizardImageStretch
★ WizardSmallImageFile

已废弃

★ AdminPrivilegesRequired
★ AlwaysCreateUninstallIcon
★ DisableAppendDir
★ DontMergeDuplicateFiles
★ MessagesFile
★ UninstallIconFile
★ UninstallIconName
★ UninstallStyle
★ WizardSmallImageBackColor
★ WizardStyle

[Types] 段

这个段是可选的。它用来定义安装程序向导在选择组件面时可供选择的所有安装类型。如果你在 [Components] 段定义了组件，但没有定义类型，在编译时将创建一个默认的安装类型设置。如果你正在使用默认 (简体中文) 消息文件，这些类型等同于下面示例中的类型。

这里是一个 [Types] 段的示例:

```
[Types]
Name: "full"; Description: "完全安装"
Name: "compact"; Description: "简洁安装"
Name: "custom"; Description: "自定义安装"; Flags: iscustom
```

下列是所支持的参数列表:

Name (必需)

描述:

类型的内部名字。用于定义 [Components] 段中组件参数，告诉安装程序组件属于哪个类型。

示例:

Name: "full"

Description (必需)

描述:

类型的描述，可以包含常量。这个描述在安装期间显示。

示例:

Description: "完全安装"

Flags

描述:

这个参数是额外选项设置。多个选项可以使用空格隔开。支持下面的选项:

iscustom

告诉安装程序这个类型是自定义类型。只在最终用户在安装期间手动改变了组件选择，安装

程序就将安装类型设置为自定义类型。请注意，如果你未定义自定义类型，安装程序将只允许用户选择一个安装类型，并且不能手动选择/取消选择组件。
示例:

Flags: iscustom

[Components] 段

这个段是可选的。它定义安装程序向导的选择组件页中显示的所有组件，以便于用户定制安装类型。
光用它一个组件不会做任何事情: 它需要“链接”到其它安装条目。请查阅组件和任务参数。
这里是一个 [Components] 段的示例:

[Components]
Name: "main"; Description: "主文件"; Types: full compact custom; Flags: fixed
Name: "help"; Description: "帮助文件"; Types: full
Name: "help\english"; Description: "English"; Types: full
Name: "help\dutch"; Description: "简体中文"; Types: full

上面的示例生成四种组件: “main”组件在最终用户选择一个名为“full”和“compact”的类型时都将安装。带有两个子组件的“help”组件类型仅在最终用户选择“full”类型时安装。
下列是所支持的参数列表:
Name (必需)

描述:

组件的内部名字。

在组件名字中的 \ 或 / 字符合计数是调用组件的层次。任何在层次 1 或更高层次的组件是子组件。在子组件前列出的小于子组件一个层次的组件是上级组件。其它有相同上级组件的组件之间是同级组件。
如果上级组件未选定，则不能选定一个它的子组件。如果所有的子组件均未选定，则上级组件也不能选定，除非组件参数引用上级组件或上级组件包含 checkablealone 标记。

如果同级组件已经标有 exclusive 标记，那么它们之中只有一个可选。

示例:

Name: "help"

Description (必需)

描述:

组件的描述，可以包含常量。这个描述用于在安装期间显示给最终用户参考。

示例:

Description: "帮助文件"

Types

描述:

用空格隔开所属组件的类型列表。如果最终用户从这个列表中选择了一个类型，这个组件将被安装。

如果未使用 fixed 标记 (看下面)，这个列表中的任何自定义类型 (使用 iscustom 标记的类型) 均被安装程序忽略。

示例:

Types: full compact

ExtraDiskSpaceRequired

描述:

这个组件所需要的额外磁盘空间，类似于 [Setup] 段中的 ExtraDiskSpaceRequired。

示例:

ExtraDiskSpaceRequired: 0

Flags

描述:

这个参数是额外选项设置。多个选项可以使用空格隔开。支持下面的选项:

checkablealone

指定当一个组件的子组件选中时，该组件是否可以选中。按默认，如果没有组件参数直接引用到该组件，未选中所有子组件将会使该组件变成未选中状态。

dontinheritcheck

指定当该组件的上级被选中时，该组件应该不自动变成已选中状态。这对顶层的组件不影响，且不能与 exclusive 标记组合使用。

exclusive

告诉安装程序这个组件与它的也使用 exclusive 标记的同级组件是互相排斥的。

fixed

告诉安装程序这个组件不能在安装期间被最终用户手动选择或取消选择。

restart

告诉安装程序如果用户安装了这个组件，将询问用户重新启动系统，不管它是不是需要 (例如，因为 [Files] 段条目用了 restartreplace 标记)。有点象 AlwaysRestart，但不是每个组件。

disablenouninstallwarning

如果这个组件已经安装在用户机器中，重新安装时在用户取消这个组件选择后，这条标记告诉安装程序不警告用户不卸载该组件。

考虑到你的组件的复杂性，你可以尝试使用 [InstallDelete] 段和这个标记为自动“卸载”取消选定的组件。

示例:

Flags: fixed

[Tasks] 段

这个段是只选的。它定义安装程序在执行安装期间所有由用户定制的任务。这些任务以选项框和单选项形式在附加任务向导页中出现。

光任务本身是不会做任何事情的: 它需要“链接”到其它安装条目。查阅组件和任务参数。

这里是一个 [Tasks] 段的示例:

[Tasks]

Name: desktopicon; Description: "创建桌面快捷方式(&D)"; GroupDescription: "添加快捷方式:"; Components: main

Name: desktopicon\common; Description: "对于所有用户"; GroupDescription: "添加快捷方式:"; Components: main; Flags: exclusive

Name: desktopicon\user; Description: "仅对当前用户"; GroupDescription: "添加快捷方式:

quicklaunchicon; Description: "创建快速运行栏快捷方式(&Q)"; GroupDescription: "添加快捷方式:"; Components: main; Flags: unchecked

Name: associate; Description: "文件关联(&A)"; GroupDescription: "其它任务:"; Flags: unchecked

下列是所支持的参数列表:

Name (必需)

描述:

任务的内部名字。

在任务名字中的 \ 或 / 字符合计数是调用任务的层次。任何在层次 1 或更高层次的任务是子任务。在子任务前列出的小于子任务一个层次的任务是上级任务。其它有相同上级任务的任务之间是同级任务。

如果上级任务未选定，则不能选定一个它的子任务。如果所有的子任务均未选定，则上级任务也不能选定，除非任务参数引用上级任务或上级任务包含 checkablealone 标记。

如果同级任务已经标有 exclusive 标记，那么它们之中只有一个可选。

示例:

Name: "desktopicon"

Description (必需)

描述:

任务的描述，可以包含常量。这个描述用于在安装期间显示给最终用户参考。

示例:

Description: "创建桌面快捷方式(&D)"

GroupDescription

描述:

任务组的组描述，可以包含常量。用相同组描述的任务将被连续组合到文字标签下。文字标签显示组描述。

示例:

GroupDescription: "附加图标"

Components

描述:

这个任务属于一个用空格隔开的组件列表。如果最终用户从这个列表中选择了一个组件，这个任务将显示。不带组件参数的任务条目总显示。

示例:

Components: main

Flags

描述:

这个参数是额外选项设置。多个选项可以使用空格隔开。支持下面的选项:

checkablealone

指定当一个任务的子任务选中时，该任务是否可以选中。按默认，如果没有任务参数直接引用到该任务，未选中所有子任务将会使该任务变成未选中状态。

checkedonce

告诉安装程序当安装程序找到已经安装的相同应用程序先前版本时，这个任务开始应该不选中。这个标记不能与 unchecked 标记组合使用。

如果 [Setup] 段的 UsePreviousTasks 指示是 no，这个标记是被禁用的。

dontinheritcheck

指定当该任务的上级被选中时，该任务应该不自动变成已选中状态。这对顶层的任务不影响，且不能与 exclusive 标记组合使用。

exclusive
告诉安装程序这个任务是与同样有 exclusive 标记的同级任务互斥的。
restart
告诉安装程序如果用户安装了这个任务，将询问用户重新启动系统，不管它是不是需要 (例如，因为 [Files] 段条目用了 restartreplace 标记)。有点象 AlwaysRestart，但不是每个任务。
unchecked
告诉安装程序这个任务在最初应该是不选中。这个标记不能与 checkedonce 标记组合使用。
示例:

Flags: unchecked

[Dirs] 段

这个可选段用来定义除创建用户选择的应用程序目录外安装程序自动创建的另外目录。在主应用程序目录下创建子目录对于这个段是公共使用的。
请注意，在使用 [Files] 段安装文件前你无需一定要创建目录，这个段起初用于创建一个空的目录。
这里是一个 [Dirs] 段的示例:

```
[Dirs]
Name: "{app}\data"
Name: "{app}\bin"
```

上面的示例中，在安装程序创建应用程序目录后，又在应用程序目录下创建了两个子目录。
下列是所支持的参数列表:
Name (必需)

描述:

要创建的目录名，通常用一个目录常量开始。

示例:

Name: "{app}\MyDir"

Attribs

描述:

指定目录的其它属性。这可以包含下面属性中的一个或多个: readonly，hidden，system。如果这个参数未指定，安装程序不会在目录中分配任何特殊的属性。
如果目录已经存在，指定的属性将与目录现有的属性组合。

示例:

Attribs: hidden system

Permissions

描述:

指定访问目录 ACL (访问控制列表) 另外的认可权限。如果你不熟悉 ACL 或不知道为什么要改变它们，则不推荐你使用这个参数，因为误用会发生系统冲突，影响安全性。
使用这个参数还有一个问题用户必须在使用 Windows 2000 或更高版本 (由于 API 的 bug，NT 4.0 不支持)，目录必须位于支持 ACL (象 NTFS) 的分区，并且当前的用户有改变目录许可的权限。如果这些条件不满足，不会显示错误消息，并且不会设置任何许可。

这个参数只应该用于你应用程序的目录隐私。决不会改变顶层目录，象 {sys} 或 {pf} 的 ACL，否则你可以打开你的用户系统的安全漏洞。
另外，推荐你避免使用这个参数，同意在包含程序文件的目录的定入访问。例如，每个人都

可以修改 {app} 目录的许可将允许没有权限的用户损坏你的应用程序的程序文件；这个创建会留下潜在的增加别个攻击的问题。(可是，在你的应用程序目录下的子目录的许可是安全的，它们不包含程序文件，例如，{app}\data。)
指定的许可设置不考虑安装前已有的目录。

这个参数可以包含象下列格式一样一个或多个空格分隔的值:

<用户或组标识>-<访问类型>

下列是 [Dirs] 段支持的访问类型:

full
同意“完全控制”许可，它与下面的 modify 一样，但另加上允许指定的用户/组获取目录的所有权并改变它的许可。保守一点使用，一般 modify 就足够了。
modify
同意“修改”许可，允许指定的用户/组读取，执行，修改和删除目录和它的子目录中的文件。
readexec
同意“读取和执行”许可，允许指定的用户/组阅读和执行目录和它的子目录中的文件。
示例:

Permissions: authusers-modify

Flags

描述:

这个参数是额外选项设置。多个选项可以使用空格隔开。支持下面的选项:

deleteafterinstall
告诉安装程序照常创建目录，但当安装完成 (或中断) 后，如果它是空的，则删除它。比如，你在脚本 [Run] 段定义了一个安装时要执行的文件，可以将它提取到临时数据然后执行后这个标记就相当有用。
这个标记不会导致安装前已经存在的目录被删除。
uninsalwaysuninstall
告诉卸载程序如果目录是空的，允许删除该目录。通常卸载程序只尝试删除在安装前不存在的目录。
uninsneveruninstall
告诉卸载程序不要删除目录。按默认，如果在 [Dirs] 段中指定的目录已经是空的，卸载程序将删除目录。
示例:

Flags: uninsneveruninstall

[Files] 段

这是定义安装程序安装文件到用户系统中的可选文件段。
这里是一个 [Files] 段的示例:

```
[Files]
Source: "CTL3DV2.DLL"; DestDir: "{sys}"; Flags: onlyifdoesntexist uninsneveruninstall
Source: "MYPROG.EXE"; DestDir: "{app}"
Source: "MYPROG.HLP"; DestDir: "{app}"
Source: "README.TXT"; DestDir: "{app}"; Flags: isreadme
```

查阅本页底部的备注段获取一些重要注意事项。

下列是所支持的参数列表:

Source (必需)

描述:

来源文件的名字。如果你不指定一个完整的路径名，编译器将预先考虑你的安装程序来源目录的路径。

这里可以在一个条目中用通配符指定一个文件组。当使用通配符时，所有与它匹配的文件使用相同的选项。

当指定 external 标记后,来源必须是发布的媒介或用户系统中现有文件 (或作用通配符)的完整路径 (例如“{src}\license.ini”)。

当指定 external 标记后，只能使用常量，因为编译器自身不能执行常量翻译。

示例:

Source: "MYPROG.EXE"

Source: "Files*"

DestDir (必需)

描述:

文件安装到用户系统中的目录。基本上都是用一个目录常量开头。如果指定的路径在用户系统中不存在，它会自动创建，并在卸载后如果是空的，卸载程序会自动删除。

示例:

DestDir: "{app}"

DestDir: "{app}\subdir"

DestName

描述:

这个参数用来指定将该文件使用新的文件名安装到用户系统中，按默认，安装程序使用 Source 参数中的名字，因此大多数场合中你不需要指定这个参数。

示例:

DestName: "MYPROG2.EXE"

Excludes

描述:

指定要排除的格式列表，用逗号分隔。这个参数不能与 external 标记组合使用。格式可以包含通配符 (“*”和“?”)。

如果格式用反斜杠符号 (“\”) 如果格式用反斜杠符号 (“\”) 开始，表示它只匹配反斜杠符号左边的路径名，否则它一直匹配到路径名末端。因此“\foo”只排除树根部的名为“foo”的文件，“foo”将排除树中任何地方的名字“foo”的文件。

该格式可以包含反斜框符号。“foo\bar”表示排除“foo\bar”和“subdir\foo\bar”两者。“\foo\bar”只排除“foo\bar”。

示例:

Source: "*"; Excludes: "*.~*"

Source: "*"; Excludes: "*.~*,\Temp*"; Flags: recursesubdirs

CopyMode

描述:

你不要在一些新的脚本中使用这个参数。不赞成用这个参数，用 Inno Setup 3.0.5 中用下列标记替换:

CopyMode: normal -> Flags: promptifolder

CopyMode: alwaysskipifsameorolder -> no flags

CopyMode: onlyifdoesntexist -> Flags: onlyifdoesntexist

CopyMode: alwaysoverwrite -> Flags: ignoreversion

CopyMode: dontcopy -> Flags: dontcopy

什么是 CopyMode: alwaysskipifsameorolder 现在是默认动作。(以前默认是 CopyMode: normal。)

Attribs

描述:

指定文件的附加属性。这可以包含下列中的一个或多个: readonly，hidden，system。如果未指定这个参数，安装程序不会在文件中分配任何特殊属性。

示例:

Attribs: hidden system

Permissions

描述:

指定操作 ACL (访问控制列表) 的附加许可。如果不熟悉 ACL 或不知道为什么或更改它们，推荐你不要使用，因为误用会导致冲突，影响系统安全。

用户必须在使用 Windows 2000 或更高版本，这个参数才有效 (NT 4.0 由于 API 的问题，不支持)，文件必须位于支持 supports ACL (象 NTFS) 的分区中，以及当前用户必须可以更改文件许可。如果这些条件不东路，不会显示错误消息，也不会设置许可。

这个参数应该只在你的应用程序么人文件中使用，不要改变共享系统文件的 ACL，否则，你打开了你的用户系统的安全漏洞。

该指定的许可设置不考虑安装前文件是否存在。

这个参数可以包含象下列格式一样一个或多个空格分隔的值:

<用户或组标识>-<访问类型>

[Files] 段支持下列访问类型:

full

同意“完全控制”许可，与修改相同 (看下面)，但又加上允许指定的用户/用户组获取文件所有权，并改变它的许可。通常保守的使用 modify 就足够了。

modify

同意“修改”许可，允许指定的用户/用户组读取、执行、修改和删除文件。

readexec

同意“读取和执行”许可，允许指定的用户/用户组读取和执行文件。

示例:

Permissions: authusers-modify

FontInstall

描述:

告诉安装程序需要安装的文件是一个字体文件，这个参数值是被保存到注册表或 WIN.INI 文件中的字体名，这个名称必须与和在资源管理器中双击字体文件看到的名字相同。注意，安装程序会在字体名后自动添加“(TrueType)”。

如果这个字体文件不是一个 TrueType 字体，你必须在标记参数中标记 fontisnttruetype。

这里建议你在安装字体到 {fonts} 目录时，使用 onlyifdoesntexist 和 uninsneveruninstall 标记。

要在 Windows 2000/XP 中成功安装字体，用户必须是超级用户或管理组成员，在 Windows NT 4.0 及早期版本，任何人都可以安装字体。

示例:

Source: "OZHANDIN.TTF"; DestDir: "{fonts}"; FontInstall: "Oz Handicraft BT"; Flags: onlyifdoesntexist uninsneveruninstall

Flags

描述:

这个参数是额外选项设置。多个选项可以使用空格隔开。支持下面的选项:

allowunsafefiles

禁止编译器自动检查不安全文件。强烈推荐你不要使用这个标记，除非你有绝对的把握。

comparetimestamp

(不推荐，看下面)

如果安装的文件已经在用户系统中存在，以及至少下列条件中的一个为 true，则告诉安装程序进行时间戳比较:

1. 现有的文件和安装的文件都没有版本信息。
2. 条目中同时使用了 ignoreversion 标记。
3. 未使用 replacesameversion 标记，并且现有的文件和要安装的文件版本号相同 (用文件的版本信息决定)。

如果现有的文件的时间戳比安装的文件老，现有的文件将被替换。否则，将不替换。

如果还没有其它方法，不推荐使用这个标记，因为其中有一个内部问题: NTFS 分区在 UTC (不同于 FAT 分区) 贮存时间戳，因本地时间戳原因 -- Inno Setup 按默认工作 -- 只要用户改变他们系统的时间区域，或转到或使夏令时生效。这可能会导致用户不希望替换的文件被替换，或用户希望替换的文件未被替换。

confirmoverwrite

替换现有的文件前总是询问用户确认。

createallsubdirs

按默认，当编译器包含子目录搜索源文件名/通配符时，将跳过空目录。这个标记会使这些目录在安装时被创建 (就象在 [Dirs] 段创建一样)。

必须和 recursesubdirs 组合使用。

deleteafterinstall

告诉安装程序象平常一样安装文件，但一旦安装完成 (或中断) 则删除。这在提取脚本中的 [Run] 段指定要执行的临时程序时有用。

这个标记不会导致安装期间未被替换的现有文件被删除。

这个标记不能与 isreadme，regserver，regtypelib，restartreplace，sharedfile 或 uninsneveruninstall 标记组合使用。

dontcopy

不复制文件到用户系统。如果文件是通过 [Code] 专门处理，这个标记是有用的。

dontverifychecksum

防止安装程序在提取后校验文件。在你想修改的已编译到安装程序中的文件使用这个标记。

必须与 nocompression 组合。

external

这个标记告诉 Inno Setup 不要编译 Source 参数指定的文件到安装程序文件中，改为从分布的媒介或用户系统中复制。查阅 Source 参数的描述获取更多信息。

fontisnttruetype

如果用 FontInstall 参数的条目安装一个非 TrueType 字体，则使用这个标记。

ignoreversion

不比较版本信息；不考虑版本号替换现有的文件。
这个标记应该用于你私人的程序中，不要用于共享的系统文件。

isreadme

表示文件为“自述”文件。安装程序中只有一个文件可以使用这个标记。当文件使用这个标记时，在安装完成后询问用户是否想查看自述文件。如果选择是，安装程序将使用与这个文件类型默认的程序打开该文件。因上，自述文件应该总是使用象 .txt，.wri 或 .doc 扩展名。请注意，如果安装程序重新启动了用户的电脑 (安装了一个带 restartreplace 标记的文件或 [Setup] 段的 AlwaysRestart 指示设为 yes)，用户将没有查看自述文件的选项。

nocompression

预防编译器尝试压缩文件。在你知道压缩没有什么好处的文件 (例如 JPEG 图像) 上使用这个标记，可以加速编译进度，并可以使生成的安装程序更小。

noencryption

防止文件被加密贮存。如果你已经使用了 encryption (用 [Setup] 段指示 Encryption)，但想让它可以在用户输入正确的密码之前用 [Code] 段支持函数 ExtractTemporaryFile 提取该文件，则使用这个标记。

noregerror

当与 regserver 或 regtypelib 标记中的任一一个组合使用时，安装程序将在注册失败时不显示任何错误消息。

onlyifdestfileexists

仅在用户系统中已经存在相同名字的文件时安装文件。如果你的安装程序是已经安装的软件的补丁，并且你希望安装用户没有安装的文件时，这个标记可能有用。

onlyifdoesntexist

仅在用户系统中不存在时安装文件。

overwritereadonly

总是覆盖只读文件。如果不带这个标记，安装程序在遇到只读文件时询问用户是否覆盖。

promptifolder

按默认，当安装的文件比现有的文件是个较老的版本 (或当使用 comparetimestamp 时发现较早的时间戳)安装程序将不替换现有的文件 (查阅这个主题下面的备注段获取详细资料)。当使用这个标记时，安装程序将询问用户是否替换文件，默认的回答是保留现有的文件。

recursesubdirs

告诉编译器或安装程序同时也搜索来源目录下子目录中的源文件名/通配符。

regserver

注册 OLE 服务 (a.k.a. ActiveX 控件)。使用这个标记，安装程序将查找和执行 DLL/OCX 的 DllRegisterServer 输出。卸载程序调用 DllUnregisterServer。当用于与共享文件结合时，DLL/OCX 只将在涉及的计数为零时取消注册。
看这个主题下面的备注段获取更多信息。

regtypelib

注册类型库 (.tlb)。卸载程序将撤销类型库注册 (除非指定了 uninsneveruninstall 标记)。与 regserver 标记一样，当用于与共享文件结合时，文件只将在涉及的计数为零时取消注册。
看这个主题下面的备注段获取更多信息。

replacesameversion

当使用这个标记，并且文件已经在用户系统中存在，以及它与要安装的文件版本号相同，安装程序将比较文件，如果它们的内部不同，则替换现有的文件。
默认动作 (例如，当这个标记未使用) 是不替换版本号相同的已有的文件。

restartreplace
这个标记通常用于替换系统核心文件。如果文件预先已经存在，且发现被锁定，以至于安装程序不能替换，安装程序将注册该文件 (用 WININIT.INI 可通过使用 MoveFileEx，分别用于 Windows 和 Windows NT) 在下次系统重新启动时替换。当出现这种情况时，将在安装完成时提示用户重新启动电脑。
为保持与 Windows 95/98/Me 兼容，不要在这个条目中使用长文件名，只支持“8.3”文件名。(Windows NT 平台没有这个限制。)
重要提示: 这个 restartreplace 标记只能在用户有管理员权限时才能在 Windows NT 平台成功替换使用中的文件。如果用户没有管理员权限，将显示下列消息: “RestartReplace 失败: MoveFileEx 失败; 代码 5。”因此，当使用 restartreplace 时，强烈推荐你在 [Setup] 段设置“PrivilegesRequired=admin”，让你的安装程序必需有管理员权限才能安装。
sharedfile
使用 Windows 共享文件计数功能 (位于注册表 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\SharedDLLs)。这个功能是用于让文件在应用程序之间共享使用，不要不注意将它删除了。每次安装该文件，涉及的文件计数将增加，当使用这个文件的应用程序卸载，涉及的文件计数减少。如果计数减少到零，该文件被删除 (会通知用户确认，除非同时指定了 uninsnosharedfileprompt 标记)。
使用这个标记的大多数文件被安装到 Windows 系统目录，包含 .OCX、.BPL 和 .DPL 文件。
skipifsourcedoesntexist
这个标记告诉编译器 -- 或安装程序，如果同时使用了 external 标记 -- 如果源文件不存在，只是跳过该条目，而不是显示一个错误消息。
sortfilesbyextension
这个标记告诉编译器在按路径名排序前，先按文件扩展名排序压缩找到的文件。如果同时使用 SolidCompression，可以进一步减小安装程序大小。
touch
这个标记命令安装程序设置安装的文件的时间/日期戳为 [Setup] 段 TouchDate 和 TouchTime 指示指定的值。
如果与 external 标记同时使用，这个标记无效。
uninsnosharedfileprompt
当卸载共享文件时，如果共享计数为零，不询问用户自动删除文件。必须与 sharedfile 标记组合使用才有效。
uninsremovereadonly
当卸载文件时，在删除文件前选删除它的只读属性。
uninsrestartdelete
当使用这个标记、并且该文件在卸载时正在被使用，卸载将文件放置到删除队列，直到系统重新启动经。卸载结束时会询问用户是否重新启动。这个标记通常在卸载不能用程序停止的外壳扩展的文件时有用。注意，要使用这个标记生效，在 Windows NT/2000/XP 中，必需要有管理员权限。
uninsneveruninstall
不卸载这个文件。这个标记在安装在任何情况下不删除的公共共享文件时有用，象 MFC DLL。
示例:

Flags: isreadme 组件和任务参数
公共参数
备注
如果文件已经在用户系统中存在，按默认，将根据下列规则替换:
1. 如果现有的文件版本老于安装的文件版本 (通过文件的版本信息确定)，现有的文件将被替换。
2. 如果现有的文件版本与安装的文件版本相同，不替换现有的文件，除非使用了 replacesameversion 标记，且两个文件的内容不一致。
3. 如果现有的文件版本新于安装的文件版本，或如果现有的文件有版本信息，但安装的文件没有，将不替换现有的文件。
4. 如果现有的文件没有版本信息，将被替换。
某些标记象 onlyifdoesntexist，ignoreversion 和 promptifolder 会更改上述的规则。
如果未使用 restartreplace 标记，并且因为其它进程正在使用，安装程序不能替换现有的文件，它将继续进行四次文件替换尝试，每次尝试延时一秒。如果所有尝试失败，将显示一个错误消息。
安装程序用 regserver 或 regtypelib 标记注册所有文件为安装的最后一个步骤。但是，如果 [Setup] 段指示 AlwaysRestart 设为 yes，或如果有用 restartreplace 标记的文件，所有文件在下次重新启动后注册 (通过在 Windows RunOnce 注册表键中创建一个项目)。
当卸载带扩展名 .HLP (Windows 帮助文件) 文件后，相应的 .GID 和 .FTS 文件也同样将自动卸载。

[Icons] 段

这个可选段定义所有创建在开始菜单和/或其它位置 (比如桌面) 的快捷方式。
这里是 [Icons] 段的例子:

```
[Icons]
Name: "{group}\My Program"; Filename: "{app}\MYPROG.EXE"; WorkingDir: "{app}"
Name: "{group}\Uninstall My Program"; Filename: "{uninstallexe}"
```

下列是所支持的参数列表:
Name (必需)
描述:
要创建的快捷方式的名字和位置。在这个参数中可以使用任何外壳文件夹常量或目录常量。请注意，快捷方式是贮存为文字文件，因此在普通文件名中不能使用的字符不这里也同样不能使用。同样，因为不可能有两个相同名字的文件一样，也不可能有两个相同名字的快捷方式。
示例:

```
Name: "{group}\My Program"
Name: "{group}\Subfolder\My Program"
Name: "{userdesktop}\My Program"
Name: "{commonprograms}\My Program"
Name: "{commonstartup}\My Program"
```

Filename (必需)
描述:
快捷方式的命令行文件名，通常用一个目录常量开头。

示例:

Filename: "{app}\MYPROG.EXE"

Filename: "{uninstallexe}"

Parameters

描述:

快捷方式的可选命令行参数，可以包含常量。

示例:

Parameters: "/play filename.mid"

WorkingDir

描述:

快捷方式的工作 (或启动) 目录，就是指示程序在哪个目录开始运行。如果这个参数未指定或是空白的，Windows 将使用一个默认路径，在 Windows 不同版本之间是不同的。这个参数可以包含常量。

示例:

WorkingDir: "{app}"

HotKey

描述:

快捷方式的热键 (或快捷键) 设置，就是可以用于启动程序的组合键。

注意: 如果你改变了快捷键并重新安装了应用程序，Windows 可以继续认可老的快捷键，直到你注销并返回或重新启动系统。

示例:

HotKey: "ctrl+alt+k"

Comment

描述:

指定快捷方式的注释 (或描述) 对象，在 Windows 2000，Me 及更高版本中可以用来弹出提示。早期版本的 Windows 被忽略注释。

示例:

Comment: "This is my program"

IconFilename

描述:

要显示的自定义图标文件名 (位于用户系统)。这可以是一个包含图标文件的可执行映像 (.exe, .dll)或一个 .ico 文件。如果这个参数未指定或空白，Windows 将使用文件的默认图标，这个参数可以包含常量。

示例:

IconFilename: "{app}\myicon.ico"

IconIndex

默认: 0

描述:

用 IconFilename 指定的文件中使用的以零为基点的图标索引。

如果 IconIndex 非零值，以及 IconFilename 未指定或空白，它将默认为 IconFilename 与 Filename 名相同。

示例:

IconIndex: 0

Flags

描述:

这个参数是额外选项设置。多个选项可以使用空格隔开。支持下面的选项:

closeonexit

当设置这个标记时，安装程序将设置快捷方式的“退出时关闭”属性。这个标记只在快捷方式指向 MS-DOS 应用程序时有效 (如果它是 .pif 扩展名)。如果没有这个标记也没有指定 dontcloseonexit 标记，安装程序将尝试改变“退出时关闭”属性。

createonlyiffileexists

当设置这个标记时，安装程序将只在用文件名参数指定的文件存在时尝试创建图标。

dontcloseonexit

等同于 closeonexit，除了它导致安装程序不选中“退出时关闭”属性。

foldershortcut

创建一个特殊的象“文件夹快捷方式”的快捷方式类型。一般来说，文件夹快捷方式出现在开始菜单中，单击该快捷方式会打开资源管理器窗口显示文件夹内容。与此相反，“文件夹快捷方式”将象菜单一样显示目标文件夹的内容，而不是单独打开一个窗口。

文件夹快捷方式只支持 Windows 2000、Me 和最高版本，一些早期版本的 Windows，安装程序在遇到这个标记时，将返回创建一个普通快捷方式。

当使用这个标记，文件夹名必须在文件名参数中指定。指定一个文件名字将导致毫无用处的快捷方式。

runmaximized

当设置这个标记时，安装程序设置图标的“运行”设置为“最大化”，使程序在启动后初始状态为最大化。

runminimized

当设置这个标记时，安装程序设置图标的“运行”设置为“最小化”，使程序在启动后初始状态为最小化。

uninsneveruninstall

通知卸载程序不删除图标。

useapppaths

当设置这个标记时，只在文件名参数中指定文件名 (无路径)，安装程序将从“HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths”注册表键获取路径名，并自动添加到文件名前面。

示例:

```
Flags: runminimized
```

[INI] 段

这是你希望安装程序在用户系统中设置 .INI 文件条目的可选段。

这里是 [INI] 段的例子:

```
[INI]
Filename: "{win}\MYPROG.INI"; Section: "InstallSettings"; Flags: uninsdeletesection
Filename: "{win}\MYPROG.INI"; Section: "InstallSettings"; Key: "InstallPath"; String: "{app}"
```

下列是所支持的参数列表:

Filename (必需)

描述:

你希望安装程序修改的 .INI 文件的名字，可以包含常量。如果这个参数是空白的，它

写入到系统的 Windows 目录中的 WIN.INI 中。

示例:

Filename: "{win}\MYPROG.INI"

Section (必需)

描述:

要创建的用于放置 INI 条目的区段名，可以包含常量。

示例:

Section: "Settings"

Key

描述:

要设置的键名，可以包含常量。如果这个参数未指定或是空白的，将不创建键。

示例:

Key: "Version"

String

描述:

分配给键的值，可以使用常量。如果这个参数未指定，将不创建键。

示例:

String: "1.0"

Flags

描述:

这个参数是额外选项设置。多个选项可以使用空格隔开。支持下面的选项:

createkeyifdoesntexist

仅在如果键名不存在时分配到键。

uninsdeleteentry

当程序卸载时删除条目。这可以写 uninsdeletesectionifempty 标记组合使用。

uninsdeletesection

当程序卸载时，删除条目所在的整个区段。这对于在 Windows 它自身包含的 INI 文件 (象一些 WIN.INI 中的区段) 中使用，这具有很大的危险性。你应该只在应用程序私有的文件中使用这个标记。

uninsdeletesectionifempty

等同于 uninsdeletesection，但只在没有键剩余的情况下删除段，它可以与 uninsdeleteentry 标记组合使用。

示例:

Flags: uninsdeleteentry

[InstallDelete] 段

这个可选段格式与 [UninstallDelete] 一样，除了它的条目处理在安装步骤之前。

[Languages] 段

Inno Setup 支持多语言安装。[Languages] 段用来定义安装程序中可使用的语言。

安装程序用下列顺序确定在消息中默认使用的语言:

1. 搜索 LanguageID 设置的语言 (通常指定在语言文件 .isl 的 [LangOptions] 段) 匹配

首选语言标识和当前用户界面语言或本地 (根据 LanguageDetectionMethod) 子语言标识两者。

2. 如果未发现，只搜索匹配的首选语言标识。如果两个或多个可用文有相同的首选语言标识，它将使用在 [Languages] 段中列出的第一个。

3. 如果未发现，默认为指定在 [Languages] 段的第一个语言。

如果 [Setup] 段的 ShowLanguageDialog 指示设置为 yes (默认)，将显示一个选择语言对话框，使用户有机会忽略语言安装选择。

ShowLanguageDialog 指示设置为 yes (默认)，将显示一个选择语言对话框，使用户有机会忽略语言安装选择。

下面是 [Languages] 段的一个示例。它定义两个语言: 简体中文，根据标准的汉化版的 Default.isl 文件，以及英语，原版的语言文件，汉化版中已包括。

[Languages]

Name: "chs"; MessagesFile: "compiler:Default.isl"

Name: "en"; MessagesFile: "compiler:English.isl"

Name (必需)

描述:

语言的内部名字，你可以设置为你喜欢的任何东西。这可以用作 [LangOptions] 或 [Messages] 段条目的前缀，使这些条目只用一种语言。{language} 常量返回选定语言的内部名字。

示例:

Name: "en"

MessagesFile (必需)

描述:

指定要读取的默认消息的文件名。这个文件必须位于你的安装程序的来源目录，除非你指定了文件的完整路径或用“compiler:”作为前缀，在这种场合下它会在编译器目录下查找文件。

当指定多个文件时，它按指定的顺序读取，因此最后的消息文件将覆盖前面的所有文件。

查阅 [Messages] 段帮助主题获取 .isl 文件格式中的详细资料。

示例:

MessagesFile: "compiler:Dutch.isl"

MessagesFile: "compiler:Default.isl,compiler:MyMessages.isl"

LicenseFile

描述:

指定可选的许可协议文件名字，用 .txt 或 .rtf (富文本) 格式，它显示在用户选择程序的目标目录之前。在运行安装程序编译器时，这个文件必须位于你的安装程序的来源目录，除非你指定了文件的完整路径或用“compiler:”作为前缀，在这种场合下它会在编译器目录下查找文件。

示例:

LicenseFile: "license-Dutch.txt"

InfoBeforeFile

描述:

指定可选的“自述”文件名字，用 .txt 或 .rtf (富文本) 格式，它显示在用户选择程序的目标目录之前。在运行安装程序编译器时，这个文件必须位于你的安装程序的来源目录，除非你指定了文件的完整路径或用“compiler:”作为前缀，在这种场合下它会在编译器目录

下查找文件。
示例:
InfoBeforeFile: "infobefore-Dutch.txt"
InfoAfterFile
描述:
指定可选的“自述”文件名字，用 .txt 或 .rtf (富文本) 格式，它显示在完成安装之后。在运行安装程序编译器时，这个文件必须位于你的安装程序的来源目录，除非你指定了文件的完整路径或用“compiler:”作为前缀，在这种场合下它会在编译器目录下查找文件。
这与 isreadme 文件不同，它里面的文字显示这一个向导页中，而不是一个单独的记事本窗口。
示例:
InfoAfterFile: "infoafter-Dutch.txt"

[Messages] 段

[Messages] 段用于定义显示在安装程序和卸载程序中的消息。通常，你不需要在你的脚本中创建 [Messages] 段，按默认，所有的消息在 Inno Setup 的 Default.isl 文件 (或在 [Languages] 段指定的条目) 中已经包含。

但是，一些特殊的消息可以用创建在脚本文件 [Messages] 段的内容覆盖。要覆盖消息，首先你需要知道你想更的消息 ID。这可以通过搜索 Default.isl 很容易地找到。例如，你想改变向导页“&Next >”按钮为“前进(&F) >”，此消息的 ID 是“ButtonNext”，所以你只要在 [Messages] 段写入下面的内容:
[Messages]
ButtonNext=前进(&F) >
一些消息包含变量,象 %1 和 %2。你可以重新排列变量顺序 (例如将 %2 移到 %1 之前)，也可以在需要的情况下重复使用同一变量 (例如“%1 ... %1 %2”)。在带变量的消息中，使用两个连续的“%”字符，表示要置入单个“%”。“%n”表示换行符。
如果你想翻译所有 Inno Setup 的文本其它语言，而不是修改 Default.isl 或覆盖在你创建的所有脚本中的每个消息，将 Default.isl 复制一份，用其它名字象 MyTranslation.isl 命名。在你想使用 MyTranslation.isl 的安装程序中，创建一个 [Languages] 段 指向该文件。
在一些场合中，这里有多个 [Languages] 段条目，在你的脚本中指定一个 [Messages] 段条目 (与 .isl 文件中指定的条目不同) 将按默认不考虑所有语言消息。要应用一个 [Messages] 段条目到只使用于一种语言，在语言的内部名字前加到前缀并跟随一个点。例如:
en.ButtonNext=前进(&F) >
特殊用途标识
BeveledLabel 消息的特殊用途是可以用于指定显示在向导窗口和卸载程序窗口左下角的文本行，下面是一个示例:
[Messages]
BeveledLabel=Inno Setup

[CustomMessages] 段

[CustomMessages] 段用于定义 {cm:...} 常量的自定义消息值。查阅常量帮助文档内容获取

更多信息。
从 [CustomMessages] 段使用 {cm:...} 常量获取的带描述的任务示例:
[CustomMessages]
CreateDesktopIcon=创建桌面快捷方式(&D)
[Tasks]
Name: desktopicon; Description: "{cm:CreateDesktopIcon}"
消息可以获取从 %1 到 %9 的声明。你可以重新排列它们的顺序 (例如，将 %2 移到 %1 之前)，也可以在需要时重复声明 (例如“%1 ... %1 %2”)。在带声明的消息中，使用两个连续的“%”字符表示嵌入单个“%”。“%n”创建一个换行符。
在一些多个 [Languages] 段条目的场合下，在你的脚本中指定一个 [CustomMessages] 段条目 (对于 .isl 文件) 将按默认覆盖所有语言的消息。要只在一种语言中使用 [CustomMessages] 段条目，用语言的内部名字作为前缀，然后跟随一个句点。例如:
nl.CreateDesktopIcon=Maak een snelkoppeling op het &bureaublad
通常所有有下列自定义消息定义和每种语言的翻译已经包含在各自的语言 .isl 文件中 (以简体中文语言为例):
NameAndVersion=%1 版本 %2
AdditionalIcons=附加快捷方式:
CreateDesktopIcon=创建桌面快捷方式(&D)
CreateQuickLaunchIcon=创建快速运行栏快捷方式(&Q)
ProgramOnTheWeb=%1 网站
UninstallProgram=卸载 %1
LaunchProgram=运行 %1
AssocFileExtension=将 %2 文件扩展名与 %1 建立关联(&A)
AssocingFileExtension=正在将 %2 文件扩展名与 %1 建立关联...
你可以在脚一中使用这些预定的自定义消息。下列是在 UninstallProgram 中使用的示例:
[Icons]
Name: "{group}\{cm:UninstallProgram,My Program}"; Filename: "{uninstallexe}"

[LangOptions] 段

[LangOptions] 段用于定义特殊语言设置，象字体一样，被安装程序和卸载程序使用。通常，你不需要在你的脚本中创建 [LangOptions] 按默认，所有的特殊语言设置在 Inno Setup 的 Default.isl 文件 (或在 [Languages] 段指定的条目) 中已经包含。
下面是 [LangOptions] 段的一个例子。(下面列出的设置是默认值。)
[LangOptions]
LanguageName=English
LanguageID=$0409
LanguageCodePage=0
DialogFontName=
DialogFontSize=8
WelcomeFontName=Verdana
WelcomeFontSize=12
TitleFontName=Arial
TitleFontSize=29

CopyrightFontName=Arial
CopyrightFontSize=8
LanguageName 是语言的名字。它显示在多语言安装程序中的选择语言对话框可用语言列表中。它以 Unicode 字符形式 (在 NT 平台也是这样显示) 贮存在内部。要嵌入 Unicode 字符，使用“<nnnn>”，这里的“nnnn”是 4 位数的十六进制 Unicode 字符集。你可以在 Windows 2000 及更高版本中的附件中用字符映射表查找字符的 Unicode 字符集。
LanguageID 是语言的数字化“语言标识”。查阅 http://msdn.microsoft.com/library/en-us/intl/nls_238z.asp 获取有效的语言标识列表。这是用来自动侦测默认使用的语言，请确保它格式使用正确，总是用一个“$”符号作为开头，因为语言标识是十六进制格式。
LanguageCodePage 用来指定显示语言所需要的“代码页”。当在多语言安装的选择语言对话框中加上可用语言列表时，它的 LanguageCodePage 值与系统代码页进行比较以确定应该列出哪些语言。只有那些 LanguageCodePage 的值与系统代码页匹配的语言才能显示。目的是不显示在用户系统中不能正确显示的语言。例如，俄语文字不能在除代码页为 1251 的系统中正确显示，因此如果系统运行在其它代码页中，俄语没有必要列出。
如果 LanguageCodePage 设为 0，该语言将总是在列表中列出，且不考虑系统的代码页。它可以判断在包含纯 ASCII 的语言中使用 0。例如象 English，因为 ASCII 同样可以在所有代码页中使用。
DialogFontName 和 DialogFontSize 指定在对话框中使用的字体名和大小 (磅)。如果指定的字体名在用户系统中不存在，或是一个空字串，将用 8 磅大小的 Microsoft Sans Serif 或 MS Sans Serif 替换。
WelcomeFontName 和 WelcomeFontSize 指定在欢迎向导页和安装完成向导页中的字体名和大小 (磅)。如果指定的字体名在用户系统中不存在，或是一个空的字符，将用 12 磅大小的 Microsoft Sans Serif 或 MS Sans Serif 替换。
TitleFontName 和 TitleFontSize 指定在背景窗口顶部 (仅在 indowVisible=yes 时可见) 使用的字体名和大小 (磅)。如果指定的字体名在用户系统中不存在，将用 29 磅的 Arial 代替。如果指定的字体名是空的，将用 29 磅的 Microsoft Sans Serif 或 MS Sans Serif 代替。
CopyrightFontName 和 CopyrightFontSize 指定在背景窗口中 AppCopyright 消息 (仅在 indowVisible=yes 时可见) 使用的字体名和大小 (磅)。如果指定的字体名在用户系统中不存在，将用 8 磅的 Arial 代替。如果指定的字体名是空的，将用 8 磅的 Microsoft Sans Serif 或 MS Sans Serif 代替。
在有多个 [Languages] 段条目的情况下，在脚本中指定一个 [LangOptions] 段指示 (不同于 an .isl 文件) 将按默认不覆盖所有语言指示。要应用 [LangOptions] 段指示到其中一种语言，用语言的内部名字跟随一个点作为前缀。例如:
en.LanguageName=English

[Registry] 段

这个可选段用来定义一些你想用安装程序在用户系统中创建、修改或删除的注册表键/值。按默认，用安装程序创建的注册表键和值在卸载时不删除。如果你想让卸载程序删除键或值，你必须包含下面 uninsdelete* 标记中的一个。

下面是 [Registry] 段的一个示例。
[Registry]

Root: HKCU; Subkey: "Software\My Company"; Flags: uninsdeletekeyifempty
Root: HKCU; Subkey: "Software\My Company\My Program"; Flags: uninsdeletekey
Root: HKLM; Subkey: "Software\My Company"; Flags: uninsdeletekeyifempty
Root: HKLM; Subkey: "Software\My Company\My Program"; Flags: uninsdeletekey
Root: HKLM; Subkey: "Software\My Company\My Program"; ValueType: string; ValueName: "InstallPath"; ValueData: "{app}"
下列是所支持的参数列表:
Root (必需)
描述:
根键。必须是下列中的一个:
HKCR (HKEY_CLASSES_ROOT)
HKCU (HKEY_CURRENT_USER)
HKLM (HKEY_LOCAL_MACHINE)
HKU (HKEY_USERS)
HKCC (HKEY_CURRENT_CONFIG)
示例:
Root: HKCU
Subkey (必需)
描述:
子键名，可以包含常量。
示例:
Subkey: "Software\My Company\My Program"

ValueType
描述:
值的数据类型。必须是下面中的一个:
none
string
expandsz
multisz
dword
binary
如果指定了 none （默认设置），安装程序将创建一个没有键值的键，在这种情况下，ValueName 和 ValueData 参数将被忽略。
如果指定了 string，安装程序将创建一个字符串 （REG_SZ） 值。
如果指定了 expandsz，安装程序将创建一个扩展字符串 （REG_EXPAND_SZ） 值。这种数据类型起初是用于 Windows NT/2000/XP 上，但是也支持 Windows 95/98/me。
如果指定了 multisz，安装程序将创建一个多行文本 （REG_MULTI_SZ） 值。
如果指定了 dword，安装程序将创建一个整数 （REG_DWORD） 值。
如果指定了 binary，安装程序将创建一个二进制 （REG_BINARY） 值。
示例:
ValueType: string
ValueName
描述:

要创建的值名，可以包含常量。如果是空白的，将写入到“默认”值。如果 ValueType 参数设置为 none，这个参数被忽略。

示例:

ValueName: "Version"

ValueData

描述:

值的数据。如果 ValueType 参数是 string，expandsz 或 ultisz，这是这一个可以包含常量的字符串。如果数据类型是 dword，这可以是一个十进制整数 (例如“123”)，一个十六进制整数 (例如“$7B”) 或解析为整数的常量。如果数据类型是 binary，这可以是下列形式的十六进制字节序列: “00 ff 12 34”。如果数据类型是 none，将被忽略。

在 string，expandsz 或 multisz 类型值中，你可以在这个参数中使用一个特殊的常量调用 {olddata}。{olddata} 用先前的注册表值数据替换。如果你想添加一个字串到现有的值中，{olddata} 常量非常有用。例如，{olddata};{app}。如果该值不存在或现有的值不是一个字符串类型，{olddata} 常量将被删除。如果创建的值是一个 multisz 类型，但现有的值不是多行字符类型(例如，它是 REG_SZ 或 REG_EXPAND_SZ)，{olddata} 常量也将被删除。反这亦然。

在 multisz 类型值中，你可以在参数中使用一个特殊常量调用 {break} 以插入换行符。

示例:

ValueData: "1.0"

Permissions

描述:

指定登录注册表键 ACL (访问控制列表) 的附加权限。如果你不熟悉 ACL 或你不知道为什么要列改，不推荐你使用这个参数，因为误用会导致重大的系统安全问题。

这个参数在用户必须运行 Windows 2000 或更高版本 (NT 4.0 由于 API 的问题不支持) 以及当前用户必须有更改注册表的权限时才有效。这些条件不满足时，不会显示错误消息，权限也不会被设置。

这个参数应该只使用于你的应用程序特有的注册表键中。不要更改顶级键象 HKEY_LOCAL_MACHINE\SOFTWARE 的 ACL，否则你自己打开了用户系统的安全漏洞。指定的权限不考虑注册表键在安装前是否存在。如果 ValueType 是 none 和使用了 deletekey 或 deletevalue 标记时，不要设置这个权限。

这个参数可以包含象下列格式一样一个或多个空格分隔的值:

<用户或组标识>-<访问类型>

下面是 [Registry] 段支持的访问类型:

full

同意“完全控制”许可，与修改相同 (看下面)，但又加上允许指定的用户/用户组获取注册表键所有权，并改变它的许可。通常保守的使用 modify 就足够了。

modify

同意“修改”许可，允许指定的用户/用户组读取、修改和删除注册表值或子键。

read

同意“读取和执行”许可，允许指定的用户/用户组读取注册表值或子键。

示例:

Permissions: authusers-modify

Flags

描述:

这个参数是额外选项设置。多个选项可以使用空格隔开。支持下面的选项:

createvalueifdoesntexist

当指定了这个标记，安装程序只在如果没有相同名字的值存在时创建值。如果值类型是 none，或如果你指定了 deletevalue 标记，这个标记无效。

deletekey

当指定了这个标记，安装程序在如果条目存在的情况下，先将尝试删除它，包括其中的所有值和子键。如果 ValueType 不是 none，那么它将创建一个新的键和值。

要防止意外，如果子键是空白的或只包含反斜框符号，安装时这个标记被忽略。

deletevalue

当指定了这个标记，安装程序在如果值存在的情况下，先将尝试删除值，如果 ValueType 是 none，那么在键不存在的情况下，它将创建键以及新值。

dontcreatekey

当指定了这个标记，如果键已经在用户系统中不存在，安装程序将不尝试创建键或值。如果键不存在，不显示错误消息。

一般来说，这个键与 uninsdeletekey 标记组合使用，在卸载时删除键，但安装时不创建键。

noerror

如果安装程序因任何原因创建键或值失败，不显示错误消息。

preservestringtype

这只在当 ValueType 参数是 string 或 expandsz 时适用。当指定这个标记，并且值不存在或现有的值不是 string 类型 (REG_SZ 或 REG_EXPAND_SZ)，它将用 ValueType 指定的类型创建。如果值存在，并且是 string 类型，它将用先存在值的相同值类型替换。

uninsclearvalue

当卸载程序时，设置值数据为空字符 (类型 REG_SZ)。这个标记不能与 uninsdeletekey 标记组合使用。

uninsdeletekey

当卸载程序时，删除整个键，包含其中的所有值和子键。这对于 Windows 自身使用的键明显不是一个好方法。你只能用于你的应用程序特有的键中。

为防止意外，安装期间如果子键空白或只包含反斜框符号，这个标记被忽略。

uninsdeletekeyifempty

当程序卸载时，如果这个键的内部没有值或子键，则删除这个键。这个标记可以与 uninsdeletevalue 组合使用。

为防止意外，安装期间如果子键空白或只包含反斜框符号，这个标记被忽略。

uninsdeletevalue

当程序卸载时删除该值。这个标记不能与 uninsdeletekeyifempty 组合使用。

注意: 在早于 1.1 的 Inno Setup 版本中，你可以使用这个标记连同数据类型 none，那么它的功能与“如果空则删除键”标记一样。这个方法已经不支持了。你必须使用 uninsdeletekeyifempty 标记实现。

示例:

Flags: uninsdeletevalue

[Run] & [UninstallRun] 段

[Run] 段是可选的，用来指定程序完成安装后、在安装程序显示最终对话框之前要执行的程序数，[UninstallRun] 段也可样是可选的，用来指定在卸载第一步要执行的程序数。除在下面有注释的外，两个段用相同的语法。

程序按它们在脚本中的出现顺序执行。按默认，当处理 [Run]/[UninstallRun] 段条目时，安装程序/卸载程序将在处理下一个任务之前等待，直到程序终止。除非使用了 nowait，shellexec 或 waituntilidle 标记。

注意，按默认，如果 [Run] 段队列文件中的一个正在执行的程序要在下一次重新启动后替换 (通过调用 MoveFileEx 或通过修改 wininit.ini)，安装程序将进行侦测，并在安装结束后提示用户重新启动电脑。如果你不想这么做，设置 RestartIfNeededByRun 指示为 no。

下面是 [Run] 段的一个示例。

```
[Run]
Filename: "{app}\INIT.EXE"; Parameters: "/x"
Filename: "{app}\README.TXT"; Description: "查看自述文件"; Flags: postinstall shellexec skipifsilent
Filename: "{app}\MYPROG.EXE"; Description: "运行应用程序"; Flags: postinstall nowait skipifsilent unchecked
```

下列是所支持的参数列表:

Filename (必需)

描述:

要执行的程序，或要打开的文件/文件夹。如果 Filename 不是一个可执行文件 (.exe 或 .com) 或批处理文件 (.bat 或 .cmd)，你必须在条目中使用 the shellexec 标记。这个参数可以包含常量。

示例:

```
Filename: "{app}\INIT.EXE"
```

Description

描述:

仅在 [Run] 段有效。这是条目的描述，可以包含常量。这个描述用于带 postinstall 标记的条目。如果条目的描述未指定，安装程序将使用一个默认描述。这个描述根据条目的类型(normal 或 shellexec)。

示例:

```
Description: "查看自述文件"
```

Parameters

描述:

程序的可选命令行参数，可以包含常量。

示例:

```
Parameters: "/x"
```

WorkingDir

描述:

指定程序在哪个目录启动。如果这个参数未指定或是空白的，它使用 Filename 参数中的目录。如果 Filename 不包含路径，它将使用默认目录。这个参数可以包含常量。

示例:

```
WorkingDir: "{app}"
```

StatusMsg

描述:

仅在 [Run] 段有效。确定程序执行时显示在向导页的消息。如果这个参数未指定可是空白的，将使用默认的消息“正在完成安装...”。这个参数可以包含常量。

示例:

StatusMsg: "正在安装 BDE..."

RunOnceId

描述:

仅在 [UninstallRun] 段有效。如果已经安装了相同的应用程序，卸载日志文件中的“run”条目将被复制一个副本。通过分配一个字符给 RunOnceId，可以确保在卸载期间特殊的 [UninstallRun] 条目只执行一次。例如，如果卸载日志中有两个或更多“run”条目用“DelService”的 RunOnceId 设置，只执行最后一个用“DelService”的 RunOnceId 设置的条目；其它的将被忽略。注意 RunOnceId 比校是区分大小写的。

示例:

RunOnceId: "DelService"

Flags

描述:

这个参数是额外选项设置。多个选项可以使用空格隔开。支持下面的选项:

hidewizard

如果指定了这个标记，向导将在程序运行期间隐藏。

nowait

如果指定了这个标记，它将在处理下一个 [Run] 条目前或完成安装前不等待进程执行完成。不能与 waituntilidle 或 waituntilterminated 组合使用。

postinstall

仅在 [Run] 段有效。告诉安装程序在安装完成向导页创建一个选择框，用户可以选中或不选中这个选择框从而决定是否处理这个条目。以前这个标记调用 showcheckbox。

如果安装程序已经重新启动了用户的电脑 (安装了一个带 restartreplace 标记的文件或如果 [Setup] 段的 AlwaysRestart 指示是 yes 引起的)，选择框没有机会出现，因此这些条目不会被处理。

[Files] 段条目中的 isreadme 标记现在已被废弃。如果编译器带 isreadme 标记的条目，它将从 [Files] 段条目中忽略这个标记，并在 [Run] 段条目列表的开头插入一个生成的 [Run] 条目。这相生成的 [Run] 段条目运行自述文件，并带有 shellexec、skipifdoesntexist、postinstall 和 skipifsilent 标记。

runhidden

如果指定了这个标记，它将在隐藏窗口中运行程序。请在执行一个要提示用户输入的程序中不要使用这个标记。

runmaximized

如果指定了这个标记，将在最大化窗口运行程序或文档。

runminimized

如果指定了这个标记，将在最小化窗口运行程序或文档。

shellexec

如果 Filename 不是一个直接可执行文件 (.exe 或 .com 文件)，这个标记是必需的。当设置这个标记时，Filename 可以是一个文件夹或任何已注册的文件类型 -- 包括 .hlp，.doc 等。

该文件将用用户系统中与这个文件类型关联的应用程序打开，与在资源管理器双击文件的方法是相同的。
按默认，当使用 shellexec 标记时，将不等待，直到生成的进程终止。
如果你需要，你必须添加标记 waituntilterminated。注意，如果新进程未生成，它不能执行也将不等待 -- 例如，文件指定指定为一个文件夹。
skipifdoesntexist
如果这个标记在 [Run] 段中指定，如果文件名不存在，安装程序不显示错误消息。
如果这个标记在 [UninstallRun] 段中指定，如果文件名不存在，卸载程序不显示“一些元素不能删除”的警告。
skipifnotsilent
仅在 [Run] 段有效。告诉安装程序如果安装程序未在后台运行则跳过这个条目。
skipifsilent
仅在 [Run] 段有效。告诉安装程序如果安装程序在后台运行则跳过这个条目。
unchecked
仅在 [Run] 段有效。告诉安装程序初始为不选中选择框。如果用户希望处理这个条目，可以通过选取选择框执行。如果 postinstall 标记未同时指定，这个标记被忽略。
waituntilidle
如果指定了这个标记，它将在未输入期间等待，直到进程等待用户输入，而不是等待进程终止。(调用 WaitForInputIdle Win32 函数。) 不能与 nowait 或 waituntilterminted 组合使用。
waituntilterminated
如果指定这个标记，将等待到进程完全终止。注意这是一个默认动作 (也就是你不需要指定这个标记)，除非你使用了 shellexec 标记，在这种情况下，如果你要等待，需要指定这个标记。不能与 nowait 或 waituntilidle 组合使用。
示例:
Flags: postinstall nowait skipifsilent

[UninstallDelete] 段

这个可选段定义你想让卸载程序删除除用 [Files] 或 [Dirs] 条目安装/创建外的其它文件或目录，或由你应用程序创建的一些公共使用的 .INI 文件。卸载程序在卸载时最后一步处理这些条目。
这里是 [UninstallDelete] 段的一个示例:
[UninstallDelete]
Type: files; Name: "{win}\MYPROG.INI"
下列是所支持的参数列表:
Type (必需)
描述:
指定卸载程序要删除的是什么。必须是下面中的一个:
files
该名字参数指定一个详细的文件名，或带通配符的文件名。
filesandordirs
除同时还匹配目录名外，功能与 files 相同，并删除任何名字匹配的目录以及包含它们中的所有文件和子目录。
dirifempty

当使用这个参数时，名字参数必须是目录名，但它不能包含通配符。该目录只在不包含任何文件或子目录的情况下才被删除。
示例:

Type: files

Name (必需)

描述:

要删除的文件或目录名。

注意: 不要尝试在这里使用通配符用来删除 {app} 目录中的所有文件。我强烈推荐你不要这么做有两个原因。首先，用户通常不希望将他们放置在应用程序目录中的数据文件在没有警告的情况下被删除 (例如，如果用户将它移动到其它驱动器，那么有些不用删除的内容将被卸载)。最好是保留它让最终用户在卸载后手动删除。同时，如果用户由于过失刚好将程序安装在危险的目录(例如，C:\WINDOWS)，如果这时卸载将后果严重。再说一遍，不要这么做!
示例:

Name: "{win}\MYPROG.INI"

8、Pascal 脚本

介绍

Pascal 脚本功能 (modern Delphi-like Pascal) 增加了许多新的功能定制你的安装程序和卸载程序，一些例子:
支持自定义条件的情况下的中断安装和卸载。

支持的安装程序运行时添加自定义向导页。

在安装前、安装期间、安装后从 Pascal 脚本提取或调用 DLL 或其它文件。

支持当作任何普通常量的脚本化常量，从注册表读取，从 ini 读取和从命令行读取常量等。

支持运行时自定义条件下的类型、组件和/或任务的删除。
根据自定义条件支持有条件的 [Files]、[Registry]、[Run] 等条目安装。

几乎 Inno Setup 自身可以执行的许多支持函数来自 Pascal 脚本。
也可以使用一个完整的 run-time 调试器调试你的自定义 Pascal 脚本。
用于 Inno Setup 的脚本引擎是来自 Innerfuse 的 Carlo Kok 编写的 Innerfuse Pascal 脚本，Innerfuse Pascal 脚本是免费使用，并带有源代码。查阅 http://www.remobjects.com/?ps 获取更多信息。

创建 [Code] 段

[Code] 段是一个可选的指定 Pascal 脚本的段。Pascal 脚本可以用于通过多种方法定制安装程序或卸载程序。请注意，创建 Pascal 脚本不是很方便，需要有丰富的 Inno Setup 使用经验，以及 Pascal 或至少一种其它类型语言的设计知识。
Inno Setup 目录的“Examples”子目录下有一些“Code*.iss”和“UninstallCode*.iss”文件，包含各种 [Code] 段的使用示例。请在创建你自己的 Pascal 脚本前仔细学习。
注意: 要学习更多 Pascal 程序语言，你可查找一些有用的 Marco Cantu 免费的基本 Pascal 书籍。

事件函数

Pascal 脚本可以包含单独的在适当时间调用的事件函数，对于安装程序，它们是:
function InitializeSetup(): Boolean;
在安装程序初始化时调用，返回 False 中断安装，返回 True 反之。

procedure InitializeWizard();
使用这个事件函数启动时改变向导或向导页。你不能在它触发之后使用 InitializeSetup 事件函数，向导窗体不退出。

procedure DeinitializeSetup();
仅在安装程序终止前调用。注意这个函数在即使用户在任何内容安装之前退出安装程序时也会调用。

procedure CurStepChanged(CurStep: TSetupStep);
你可以用这个事件函数执行你自己的预安装和安装后任务。
在实际安装开始之前用 CurStep=ssInstall 调用，或在实际安装完成之后用 CurStep=ssPostInstall 调用，或在安装程序终止之前和安装完成之后用 CurStep=ssDone 调用。

function NextButtonClick(CurPageID: Integer): Boolean;
当用户单击下一步按钮时调用。如果你返回 True，向导将移到下一页；如果返回 False，它仍保留在当前页 (用 CurPageID 指定)。
注意，这个函数在静寂安装时也会调用，即使没有下一步按钮让用户单击。安装程序会模拟单击下一步按钮。在静寂安装中，如果你的 NextButtonClick 函数在安装之前返回 False，安装程序将自动退出。
function BackButtonClick(CurPageID: Integer): Boolean;
当用户单击上一步按钮时调用。如果你返回 True，向导将移到上一页；如果返回 False，它仍保留在当前页 (用 CurPageID 指定)。

procedure CancelButtonClick(CurPageID: Integer; var Cancel, Confirm: Boolean);
当用户单击取消按钮或单击窗口中的关闭按钮时调用。Cancel 参数指定是否是一般的取消进程；默认为 True。Confirm 参数指定是否显示“退出安装程序吗？”的消息框；一般它默认为 True。如果 Cancel 设为 False，那么 Confirm 值被忽略。
function ShouldSkipPage(PageID: Integer): Boolean;
向导调用这个事件函数确定是否在所有页或不在一个特殊页 (用 PageID 指定) 显示。如果返回 True，将跳过该页；如果你返回 False，该页被显示。
注意: 这个事件函数不被 wpWelcome、wpPreparing 和 wpInstalling 页调用，还有安装程序已经确定要跳过的页也不会调用 (例如，没有包含组件安装程序的 wpSelectComponents)。

procedure CurPageChanged(CurPageID: Integer);
在新向导页 (用 CurPageID 指定) 显示后调用。

function CheckPassword(Password: String): Boolean;
如果安装程序在 Pascal 脚本中发现 CheckPassword 事件函数，它自动显示密码页并调用 CheckPassword 检查密码。返回 True 表示接受密码，返回 False 拒绝。
要避免在编译的安装程序的 [Code] 段内部贮存真实的密码，你应该用其它无用的信息进行比较: 计算你自己密码的 MD5 的无用信息，然后编译到 GetMD5OfString(Password)。通过这种方法保护实际密码值。
注意: 如果你已经用 CheckPassword 事件函数，并且你的用户带“/PASSWORD=”和“/SILENT”安装命令行参数运行安装程序，你的 CheckPassword 函数将在其它事件函数

调用之前调用，包括 InitializeSetup。

function NeedRestart(): Boolean;

返回 True 告诉安装程序提示用户在安装结束时重新启动系统，False 则反之。

function UpdateReadyMemo(Space, NewLine, MemoUserInfoInfo, MemoDirInfo, MemoTypeInfo, MemoComponentsInfo, MemoGroupInfo, MemoTasksInfo: String): String;

如果安装程序在 Pascal 脚本中发现 UpdateReadyMemo 事件函数，当准备安装向导页变为激活页时自动调用。它返回的文字显示在准备安装向导页的备注注册中，该文字是用 NewLine 参数换行的字符。参数空间包含一个带安全可靠的字符。其它参数将包含安装程序用于设置段的字符 (可能是空的)。MemoDirInfo 参数包含象选择目录段的字符。

procedure RegisterPreviousData(PreviousDataKey: Integer);

要在自定义向导页中贮存用户输入的设置，在 Pascal 脚本中放入一个 RegisterPreviousData 事件函数，并调用 SetPreviousData(PreviousDataKey, ...) 替换它，每个设置一次。

function CheckSerial(Serial: String): Boolean;

如果安装程序在 Pascal 脚本中发现 CheckSerial 事件函数，将在用户信息向导页中自动出现一个序列号对象 (必须在你的 [Setup] 段中使用 UserInfoPage=yes!)。返回 True 表示接受序列号，返回 False 拒绝。当使用序列号时，请一定要记住，这个软件无加密可言，况且 Inno Setup 源代码是免费获取的，它对于有经验的人从安装程序中删除序列号保护并不是很困难的事。使用这个只是方便用户在你的应用程序中仔细检查输入的序列号 (贮存在 {userinfoserial} 常量)。

function GetCustomSetupExitCode: Integer;

返回一个非零值命令安装程序返回一个自定义退出代码。这个函数只在安装程序运行完成并且退出代码已是零时调用。同时请查阅安装退出代码。

对于卸载程序，它们是:

function InitializeUninstall(): Boolean;

返回 False 中断卸载，True 则反之。

procedure DeinitializeUninstall();

procedure CurUninstallStepChanged(CurUninstallStep: TUninstallStep);

function UninstallNeedRestart(): Boolean;

返回 True 命令卸载程序提示用户在卸载完成后重新启动系统，False 则反之。

这里是这些函数使用的常量列表:

CurStep values

ssInstall, ssPostInstall, ssDone

CurUninstallStep values

usAppMutexCheck, usUninstall, usPostUninstall, usDone

预定义向导页 CurPageID values

wpWelcome, wpLicense, wpPassword, wpInfoBefore, wpUserInfo, wpSelectDir, wpSelectComponents, wpSelectProgramGroup, wpSelectTasks, wpReady, wpPreparing, wpInstalling, wpInfoAfter, wpFinished

这些函数不需要在 Pascal 脚本中出现。

脚本化常量

Pascal 脚本可以包含当安装程序想知道脚本化的 {code:...} 常量的值时调用的函数，调用的

函数必须有 1 个名为 Param 的字符串参数，并且必须返回一个字符串值。
{code:...} 常量语法是: {code:FunctionName|Param}
FunctionName 指定 Pascal 脚一函数的名字。

Param 指定用于函数的字符参数。如果你省略 Param，将使用一个空字符串。

如果你想在常量内部包含一个逗号，垂直条(“|”)，或括弧 (“}”)，你必须通过“%-encoding”避开它。用“%”字符串替换该字符串，后面再跟随它的两位数的十六进制代码。逗号是“%2c”，垂直条是“%7c”，括弧是“%7d”。如果你想包含一个实际的“%”字符串，使用“%25”。
Param 可以包含常量。请注意，你不需要将常量的括弧替换为上面所说的结构；它只在别处使用时才需要替换。

示例:

DefaultDirName={code:MyConst}\My Program
这里是一个包含上述使用的函数的 [Code] 段的示例。

```
[Code]
function MyConst(Param: String): String;
begin
  Result := ExpandConstant('{pf}');
end;
```

如果通过 {code:...} 常量指定的函数不包含在 [Code] 段，它必须是一个支持的函数。这是一个示例。

[INI]

FileName: "{app}\MyIni.ini"; Section: "MySettings"; Key: "ShortApp"; String: "{code:GetShortName|{app}}"

检查参数

这里是一个被所有段中被参数分开的条目支持的可选参数这就是:
Check

描述:

确定是否被处理或不处理的检查函数名。函数必须是一个 [Code] 段中的自定义函数或支持函数。
除单一的名字外，你也可以使用 boolean 表达式。披阅组件和任务参数获取 boolean 表达式的详细资料。
对于每个检查函数，可以包含一个安装程序用于检查函数的逗号隔开的参数列表。允许的参数类型是字符串，整数和布尔数。字符串参数可以包含常量。
这是一个可以从参数列表内部调用的支持函数: ExpandConstant。
示例:

```
[Files]
Source: "MYPROG.EXE"; DestDir: "{app}"; Check: MyProgCheck
Source: "A\MYFILE.TXT"; DestDir: "{app}"; Check: MyDirCheck({app}\A)
Source: "B\MYFILE.TXT"; DestDir: "{app}"; Check: MyDirCheck({app}\B)
```

所有检查函数必须有一个布尔返回值。如果检查函数 (或布尔表达式) 返回 True，那么条目将被处理，否则跳过。

安装程序可能随时调用检查函数，即使只有一个条目使用检查函数。如果你的函数执行一个长的代码段，你可以通过在全局变量中“缓冲”结果执行代码来优化它。
如果安装程序已经确定不处理条目，那么检查函数不被调用。
这里是上述使用的包含检查函数的 [Code] 段的示例。函数 DirExists 是支持函数，因此不包含在这个 [Code] 段内。

```
[Code]
var
  MyProgChecked: Boolean;
  MyProgCheckResult: Boolean;
function MyProgCheck(): Boolean;
begin
  if not MyProgChecked then begin
    MyProgCheckResult := MsgBox('Do you want to install MyProg.exe to ' + ExtractFilePath(CurrentFileName) + '?', mbConfirmation, MB_YESNO) = idYes;
    MyProgChecked := True;
  end;
  Result := MyProgCheckResult;
end;
function MyDirCheck(DirName: String): Boolean;
begin
  Result := DirExists(DirName);
end;
```

BeforeInstall 和 AfterInstall 参数

这里有两个可选的被除 [Types]、[Components] 和 [Tasks] 之外的所有段中的那些分隔到参数中的条目支持的参数。它们是:
BeforeInstall
描述:
该函数名只在条目安装之前调用一次。该函数必须是 [Code] 段的自定义函数或一个支持函数。可以包含一个安装程序用于函数的逗号隔开的参数列表。允许的参数类型是字符、整数和布尔数。字符参数可以包含常量。
这里是一个可以从参数列表内部调用的支持函数: ExpandConstant。
示例:

```
[Files]
Source: "MYPROG.EXE"; DestDir: "{-app}"; BeforeInstall: MyBeforeInstall
Source: "A\MYFILE.TXT"; DestDir: "{-app}"; BeforeInstall: MyBeforeInstall2('{-app}\A\MYFILE.TXT')
Source: "B\MYFILE.TXT"; DestDir: "{-app}"; BeforeInstall: MyBeforeInstall2('{-app}\B\MYFILE.TXT')
Source: "MYPROG.HLP"; DestDir: "{-app}"; BeforeInstall: Log('Before MYPROG.HLP Install')
```

AfterInstall
描述:

该函数名只在条目安装之后调用一次。该函数必须是 [Code] 段的自定义函数或一个支持函数。可以包含一个安装程序用于函数的参数。这个参数可以包含常量。
示例:

```
[Files]
Source: "MYPROG.EXE"; DestDir: "{-app}"; AfterInstall: MyAfterInstall
Source: "A\MYFILE.TXT"; DestDir: "{-app}"; AfterInstall: MyAfterInstall2('{-app}\A\MYFILE.TXT')
Source: "B\MYFILE.TXT"; DestDir: "{-app}"; AfterInstall: MyAfterInstall2('{-app}\B\MYFILE.TXT')
Source: "MYPROG.HLP"; DestDir: "{-app}"; AfterInstall: Log('After MYPROG.HLP Install')
```

所有 BeforeInstall 和 AfterInstall 函数不需要有返回值。
如果安装程序已经确认不处理条目，则不调用 BeforeInstall 或 AfterInstall 函数。
这里是一个包含上述函数使用的 [Code] 段示例，函数 Log 是一个支持函数，因此不包含在这个 [Code] 段。

```
[Code]
procedure MyBeforeInstall();
begin
  MsgBox('About to install MyProg.exe as ' + CurrentFileName + '.', mbInformation, MB_OK);
end;
procedure MyBeforeInstall2(FileName: String);
begin
  MsgBox('About to install ' + FileName + ' as ' + CurrentFileName + '.', mbInformation, MB_OK);
end;
procedure MyAfterInstall();
begin
  MsgBox('Just installed MyProg.exe as ' + CurrentFileName + '.', mbInformation, MB_OK);
end;
procedure MyAfterInstall2(FileName: String);
begin
  MsgBox('Just installed ' + FileName + ' as ' + CurrentFileName + '.', mbInformation, MB_OK);
end;
```

卸载代码

Pascal 脚本也可以包含卸载时调用的代码。查看事件函数主题获取更多信息。
当设计在卸载时执行的代码时，这是你所要知道的重要内容: 当彼此安装多个版本的应用程序时，卸载时只运行一个 Pascal 脚本。一般来说，脚本从最近安装的进行选择。如果这样，会卸载最新版本的应用程序，你最新版本的 Inno Setup 会被降级。相似情况，当安装较老版本的应用程序时，有可能会覆盖新版本的应用程序。
当生成其它安装程序的补丁安装程序时，补丁安装程序将与原始安装程序共享相同的卸载日志 (就象 Uninstallable 设为 yes 以及 AppId 设置为与原始安装程序相同)，确保补丁包含与原始安装程序完整的 [Code] 段副本，否则，卸载时不会运行代码。

可是，补丁安装程序中 Uninstallable 设为 no，那么安装程序不用知道现有的卸载程序或卸载日志，在这种情况下，补丁安装程序不需要包含与原始安装程序的 [Code] 段副本。

示例

Pascal Scripting 示例脚本位于单独的文件。打开 Inno Setup 安装目录下的“Examples”子目录中的“Code*.iss”文件中的一个。

支持的函数参考

这里是可以从 Pascal 脚本内部调用的支持的函数列表。
安装或卸载信息函数
function GetCmdTail: String;
function ParamCount: Integer;
function ParamStr(Index: Integer): String;
function ActiveLanguage: String;
function SetupMessage(const ID: TSetupMessageID): String;
function WizardDirValue: String;
function WizardGroupValue: String;
function WizardNoIcons: Boolean;
function WizardSetupType(const Description: Boolean): String;
function WizardSelectedComponents(const Descriptions: Boolean): String;
function WizardSelectedTasks(const Descriptions: Boolean): String;
function WizardSilent: Boolean;
function IsUninstaller: Boolean;
function UninstallSilent: Boolean;
function CurrentFileName: String;
function ExpandConstant(const S: String): String;
function ExpandConstantEx(const S: String; const CustomConst, CustomValue: String): String;
function IsComponentSelected(const Components: String): Boolean;
function IsTaskSelected(const Tasks: String): Boolean;
function ExtractTemporaryFile(const FileName: String);
function GetPreviousData(const ValueName, DefaultValueData: String): String;
function SetPreviousData(const PreviousDataKey: Integer; const ValueName, ValueData: String): Boolean;
function Terminated: Boolean;
排除函数
procedure Abort;
procedure RaiseException(const Msg: String);
function GetExceptionMessage: String;
procedure ShowExceptionMessage;
系统函数
function IsAdminLoggedOn: Boolean;
function IsPowerUserLoggedOn: Boolean;

function UsingWinNT: Boolean;
function GetWindowsVersion: Cardinal;
function GetWindowsVersionString: String;
function InstallOnThisVersion(const MinVersion, OnlyBelowVersion: String): Integer;
function GetEnv(const EnvVar: String): String;
function GetUserNameString: String;
function GetComputerNameString: String;
function GetUILanguage: Integer;
function FindWindowByClassName(const ClassName: String): HWND;
function FindWindowByWindowName(const WindowName: String): HWND;
function SendMessage(const Wnd: HWND; const Msg, WParam, LParam: Longint): Longint;
function PostMessage(const Wnd: HWND; const Msg, WParam, LParam: Longint): Longint;
function SendNotifyMessage(const Wnd: HWND; const Msg, WParam, LParam: Longint): Boolean;
function RegisterWindowMessage(const Name: String): Longint;
function SendBroadcastMessage(const Msg, WParam, LParam: Longint): Longint;
function PostBroadcastMessage(const Msg, WParam, LParam: Longint): Boolean;
function SendBroadcastNotifyMessage(const Msg, WParam, LParam: Longint): Boolean;
procedure CreateMutex(const Name: String);
function CheckForMutexes(Mutexes: String): Boolean;
procedure MakePendingFileRenameOperationsChecksum: String;
procedure UnloadDLL(Filename: String);
procedure DLLGetLastError(): Longint;
字符函数
function Chr(B: Byte): Char;
function Ord(C: Char): Byte;
function Copy(S: String; Indx, Count: Integer): String;
function Length(s: String): Longint;
function Lowercase(s: string): String;
function StringOfChar(c: Char; I : Longint): String;
function Uppercase(s: string): String;
procedure Delete(var S: String; Indx, Count: Integer);
procedure Insert(Source: String; var Dest: String; Indx: Integer);
procedure StringChange(var S: String; const FromStr, ToStr: String);
function Pos(SubStr, S: String): Integer;
function AddQuotes(const S: String): String;
function RemoveQuotes(const S: String): String;
function ConvertPercentStr(var S: String): Boolean;
function CompareText(const S1, S2: string): Integer;
function CompareStr(const S1, S2: string): Integer;
function Format1(const Format, S1: String): String;
function Format2(const Format, S1, S2: String): String;
function Format3(const Format, S1, S2, S3: String): String;
function Format4(const Format, S1, S2, S3, S4: String): String;

function Trim(const S: string): String;
function TrimLeft(const S: string): String;
function TrimRight(const S: string): String;
function StrToIntDef(s: string; def: Longint): Longint;
function StrToInt(s: string): Longint;
function IntToStr(i: Longint): String;
function CharLength(const S: String; const Index: Integer): Integer;
function AddBackslash(const S: String): String;
function RemoveBackslashUnlessRoot(const S: String): String;
function RemoveBackslash(const S: String): String;
function AddPeriod(const S: String): String;
function ExtractFileExt(const FileName: string): String;
function ExtractFileDir(const FileName: string): String;
function ExtractFilePath(const FileName: string): String;
function ExtractFileName(const FileName: string): String;
function ExtractFileDrive(const FileName: string): String;
function ExtractRelativePath(const BaseName, DestName: String): String;
function ExpandFileName(const FileName: string): String;
function ExpandUNCFileName(const FileName: string): String;
function GetDateTimeString(const DateTimeFormat: String; const DateSeparator, TimeSeparator: Char): String;
procedure SetLength(var S: String; L: Longint);
procedure CharToOemBuff(var S: String);
procedure OemToCharBuff(var S: String);
function GetMD5OfString(const S: String): String;
function SysErrorMessage(ErrorCode: Integer): String;
排列函数
function GetArrayLength(var Arr: Array): Longint;
procedure SetArrayLength(var Arr: Array; I: Longint);
文件系统函数
function DirExists(const Name: String): Boolean;
function FileExists(const Name: String): Boolean;
function FileOrDirExists(const Name: String): Boolean;
function FileSize(const Name: String; var Size: Integer): Boolean;
function GetSpaceOnDisk(const Path: String; const InMegabytes: Boolean; var Free, Total: Cardinal): Boolean;
function FileSearch(const Name, DirList: string): String;
function FindFirst(const FileName: String; var FindRec: TFindRec): Boolean;
function FindNext(var FindRec: TFindRec): Boolean;
procedure FindClose(var FindRec: TFindRec);
function GetCurrentDir: String;
function SetCurrentDir(const Dir: string): Boolean;
function GetWinDir: String;
function GetSystemDir: String;

function GetTempDir: String;
function GetShellFolder(Common: Boolean; const ID: TShellFolderID): String;
function GetShellFolderByCSIDL(const Folder: Integer; const Create: Boolean): String;
function GetShortName(const LongName: String): String;
function GenerateUniqueName(Path: String; const Extension: String): String;
function GetVersionNumbers(const Filename: String; var VersionMS, VersionLS: Cardinal): Boolean;
function GetVersionNumbersString(const Filename: String; var Version: String): Boolean;
function IsProtectedSystemFile(const Filename: String): Boolean;
function GetMD5OfFile(const Filename: String): String;
文件函数
function Exec(function GetShellFolderByCSIDL(const Folder: Integer; const Create: Boolean): String;
const Filename, Params, WorkingDir: String; const ShowCmd: Integer; const Wait: TExecWait; var ResultCode: Integer): Boolean;
function ShellExec(const Verb, Filename, Params, WorkingDir: String; const ShowCmd: Integer; const Wait: TExecWait; var ErrorCode: Integer): Boolean;
function RenameFile(const OldName, NewName: string): Boolean;
function ChangeFileExt(const FileName, Extension: string): String;
function FileCopy(const ExistingFile, NewFile: String; const FailIfExists: Boolean): Boolean;
function DeleteFile(const FileName: string): Boolean;
procedure DelayDeleteFile(const Filename: String; const Tries: Integer);
function LoadStringFromFile(const FileName: String; var S: String): Boolean;
function LoadStringsFromFile(const FileName: String; var S: TArrayOfString): Boolean;
function SaveStringToFile(const FileName, S: String; const Append: Boolean): Boolean;
function SaveStringsToFile(const FileName: String; const S: TArrayOfString; const Append: Boolean): Boolean;
function CreateDir(const Dir: string): Boolean;
function ForceDirectories(Dir: string): Boolean;
function RemoveDir(const Dir: string): Boolean;
function DelTree(const Path: String; const IsDir, DeleteFiles, DeleteSubdirsAlso: Boolean): Boolean;
function CreateShellLink(const Filename, Description, ShortcutTo, Parameters, WorkingDir, IconFilename: String; const IconIndex, ShowCmd: Integer): String;
procedure RegisterServer(const Filename: String; const FailCriticalErrors: Boolean);
function UnregisterServer(const Filename: String; const FailCriticalErrors: Boolean): Boolean;
procedure RegisterTypeLibrary(const Filename: String);
function UnregisterTypeLibrary(const Filename: String): Boolean
procedure IncrementSharedCount(const Filename: String; const AlreadyExisted: Boolean);
function DecrementSharedCount(const Filename: String): Boolean;
procedure RestartReplace(const TempFile, DestFile: String);
procedure UnregisterFont(const FontName, FontFilename: String);
function ModifyPifFile(const Filename: String; const CloseOnExit: Boolean): Boolean;
注册表函数

function RegKeyExists(const RootKey: Integer; const SubKeyName: String): Boolean;
function RegValueExists(const RootKey: Integer; const SubKeyName, ValueName: String): Boolean;
function RegGetSubkeyNames(const RootKey: Integer; const SubKeyName: String; var Names: TArrayOfString): Boolean;
function RegGetValueNames(const RootKey: Integer; const SubKeyName: String; var Names: TArrayOfString): Boolean;
function RegQueryStringValue(const RootKey: Integer; const SubKeyName, ValueName: String; var ResultStr: String): Boolean;
function RegQueryMultiStringValue(const RootKey: Integer; const SubKeyName, ValueName: String; var ResultStr: String): Boolean;
function RegQueryDWordValue(const RootKey: Integer; const SubKeyName, ValueName: String; var ResultDWord: Cardinal): Boolean;
function RegQueryBinaryValue(const RootKey: Integer; const SubKeyName, ValueName: String; var ResultStr: String): Boolean;
function RegWriteStringValue(const RootKey: Integer; const SubKeyName, ValueName, Data: String): Boolean;
function RegWriteExpandStringValue(const RootKey: Integer; const SubKeyName, ValueName, Data: String): Boolean;
function RegWriteMultiStringValue(const RootKey: Integer; const SubKeyName, ValueName, Data: String): Boolean;
function RegWriteDWordValue(const RootKey: Integer; const SubKeyName, ValueName: String; const Data: Cardinal) Boolean;
function RegWriteBinaryValue(const RootKey: Integer; const SubKeyName, ValueName, Data: String): Boolean;
function RegDeleteKeyIncludingSubkeys(const RootKey: Integer; const SubkeyName: String): Boolean;
function RegDeleteKeyIfEmpty(const RootKey: Integer; const SubkeyName: String): Boolean;
function RegDeleteValue(const RootKey: Integer; const SubKeyName, ValueName: String): Boolean;
INI 文件函数
function IniKeyExists(const Section, Key, Filename: String): Boolean;
function IsIniSectionEmpty(const Section, Filename: String): Boolean;
function GetIniBool(const Section, Key: String; const Default: Boolean; const Filename: String): Boolean
function GetIniInt(const Section, Key: String; const Default, Min, Max: Longint; const Filename: String): Longint;
function GetIniString(const Section, Key, Default, Filename: String): String;
function SetIniBool(const Section, Key: String; const Value: Boolean; const Filename: String): Boolean;
function SetIniInt(const Section, Key: String; const Value: Longint; const Filename: String): Boolean;
function SetIniString(const Section, Key, Value, Filename: String): Boolean;
procedure DeleteIniSection(const Section, Filename: String);

procedure DeleteIniEntry(const Section, Key, Filename: String);

自定义安装程序向导页函数

function CreateInputQueryPage(const AfterID: Integer; const ACaption, ADescription, ASubCaption: String): TInputQueryWizardPage;

function CreateInputOptionPage(const AfterID: Integer; const ACaption, ADescription, ASubCaption: String; Exclusive, ListBox: Boolean): TInputOptionWizardPage;

function CreateInputDirPage(const AfterID: Integer; const ACaption, ADescription, ASubCaption: String; AAppendDir: Boolean; ANewFolderName: String): TInputDirWizardPage;

function CreateInputFilePage(const AfterID: Integer; const ACaption, ADescription, ASubCaption: String): TInputFileWizardPage;

function CreateOutputMsgPage(const AfterID: Integer; const ACaption, ADescription, AMsg: String): TOutputMsgWizardPage;

function CreateOutputMsgMemoPage(const AfterID: Integer; const ACaption, ADescription, ASubCaption, AMsg: String): TOutputMsgMemoWizardPage;

function CreateOutputProgressPage(const ACaption, ADescription: String): TOutputProgressWizardPage;

function CreateCustomPage(const AfterID: Integer; const ACaption, ADescription: String): TWizardPage;

function CreateCustomForm: TSetupForm;

function PageFromID(const ID: Integer): TWizardPage;

function ScaleX(X: Integer): Integer;

function ScaleY(Y: Integer): Integer;

对话框函数

function MsgBox(const Text: String; const Typ: TMsgBoxType; const Buttons: Integer): Integer;

function SuppressibleMsgBox(const Text: String; const Typ: TMsgBoxType; const Buttons, Default: Integer): Integer;

function GetOpenFileName(const Prompt: String; var FileName: String; const InitialDirectory, Filter, DefaultExtension: String): Boolean;

function BrowseForFolder(const Prompt: String; var Directory: String; const NewFolderButton: Boolean): Boolean;

function ExitSetupMsgBox: Boolean;

COM 自动操作对象支持函数

function CreateOleObject(const ClassName: string): Variant;

function GetActiveOleObject(const ClassName: string): Variant;

安装程序记录函数

procedure Log(const S: String);

其它函数

procedure Sleep(const Milliseconds: LongInt);

function Random(const Range: Integer): Integer;

procedure Beep;

procedure BringToFrontAndRestore;

不建议使用的函数

function LoadDLL(const DLLName: String; var ErrorCode: Integer): Longint;

function CallDLLProc(const DLLHandle: Longint; const ProcName: String; const Param1,

Param2: Longint; var Result: Longint): Boolean;
function FreeDLL(const DLLHandle: Longint): Boolean;
function CastStringToInteger(var S: String): Longint;
function CastIntegerToString(const L: Longint): String;
这里是用于这些函数的常量列表:
CurStep 值
ssInstall, ssPostInstall, ssDone
CurPage 值
wpWelcome, wpLicense, wpPassword, wpInfoBefore, wpUserInfo, wpSelectDir, wpSelectComponents, wpSelectProgramGroup, wpSelectTasks, wpReady, wpPreparing, wpInstalling, wpInfoAfter, wpFinished
TMsgBoxType
mbInformation, mbConfirmation, mbError, mbCriticalError
MsgBox - 按钮标记
MB_OK, MB_OKCANCEL, MB_ABORTRETRYIGNORE, MB_YESNOCANCEL, MB_YESNO, MB_RETRYCANCEL, MB_DEFBUTTON1, MB_DEFBUTTON2, MB_DEFBUTTON3, MB_SETFOREGROUND
MsgBox - 返回值
IDOK, IDCANCEL, IDABORT, IDRETRY, IDIGNORE, IDYES, IDNO
TGetShellFolderID
sfDesktop, sfStartMenu, sfPrograms, sfStartup, sfSendTo, sfFonts, sfAppData, sfDocs, sfTemplates, sfFavorites, sfLocalAppData
Reg* - 根键值
HKEY_CLASSES_ROOT, HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE, HKEY_USERS, HKEY_PERFORMANCE_DATA, HKEY_CURRENT_CONFIG, HKEY_DYN_DATA,
HKCR, HKCU, HKLM, HKU, HKCC
TShouldProcessEntryResult
srNo, srYes, srUnknown
InstallOnThisVersion - 返回值
irInstall, irNotOnThisPlatform, irVerTooLow, irVerTooHigh, irInvalid
TSetupMessageID
用 'msg' + 消息名。 示例: SetupMessage(msgSetupAppTitle)
Exec 和 ShellExec - ShowCmd 值
SW_SHOW, SW_SHOWNORMAL, SW_SHOWMAXIMIZED, SW_SHOWMINIMIZED, SW_SHOWMINNOACTIVE, SW_HIDE

支持的类参考

下面是你可以在 Pascal 脚本中使用的支持类的列表。也有两个支持对象可用: 类型 MainForm: TMainForm，和类型 WizardForm: TWizardForm，以及一个特殊常量: 类型 crHand: TControl.Cursor。注意: MainForm 只在 WindowVisible 时可见。
注意: 你也可以从 Borland 的 Delphi Visual Component Library (VCL) 帮助查找，下面的类主要是简单地包装 VCL 类在 Inno Setup 中使用。请查阅

http://info.borland.com/techpubs/delphi/ 和
ftp://ftp.borland.com/pub/delphi/techpubs/delphi3/d3cs.zip。

```
TObject = class
   constructor Create;
   procedure Free;
end;
TPersistent = class(TObject)
   procedure Assign(Source: TPersistent);
end;
TComponent = class(TPersistent)
   function FindComponent(AName: string): TComponent;
   constructor Create(AOwner: TComponent);
   property Owner: TComponent; read write;
   procedure DESTROYCOMPONENTS;
   procedure DESTROYING;
   procedure FREENOTIFICATION(ACOMPONENT:TCOMPONENT);
   procedure INSERTCOMPONENT(ACOMPONENT:TCOMPONENT);
   procedure REMOVECOMPONENT(ACOMPONENT:TCOMPONENT);
   property COMPONENTS[Index: INTEGER]: TCOMPONENT; read;
   property COMPONENTCOUNT: INTEGER; read;
   property COMPONENTINDEX: INTEGER; read write;
   property COMPONENTSTATE: Byte; read;
   property DESIGNINFO: LONGINT; read write;
   property NAME: STRING; read write;
   property TAG: LONGINT; read write;
end;
TStrings = class(TPersistent)
   function Add(S: string): Integer;
   procedure Append(S: string);
   procedure AddStrings(Strings: TStrings);
   procedure Clear;
   procedure Delete(Index: Integer);
   function IndexOf(const S: string): Integer;
   procedure Insert(Index: Integer; S: string);
   property Count: Integer; read;
   property Text: String; read write;
   property CommaText: String; read write;
   procedure LoadFromFile(FileName: string);
   procedure SaveToFile(FileName: string);
   property Strings[Index: Integer]: String; read write;
   property Objects[Index: Integer]: TObject; read write;
end;
TNotifyEvent = procedure(Sender: TObject);
TStringList = class(TStrings)
```

```
    function FIND(S:STRING;var INDEX:INTEGER):BOOLEAN;
    procedure SORT;
    property DUPLICATES: TDUPLICATES; read write;
    property SORTED: BOOLEAN; read write;
    property ONCHANGE: TNOTIFYEVENT; read write;
    property ONCHANGING: TNOTIFYEVENT; read write;
end;
TStream = class(TObject)
    function READ(BUFFER:STRING;COUNT:LONGINT):LONGINT;
    function WRITE(BUFFER:STRING;COUNT:LONGINT):LONGINT;
    function SEEK(OFFSET:LONGINT;ORIGIN:WORD):LONGINT;
    procedure READBUFFER(BUFFER:STRING;COUNT:LONGINT;
    procedure WRITEBUFFER(BUFFER:STRING;COUNT:LONGINT;
    function COPYFROM(SOURCE:TSTREAM;COUNT:LONGINT;
    property POSITION: LONGINT; read write;
    property SIZE: LONGINT; read write;
end;
THandleStream = class(TStream)
    constructor CREATE(AHANDLE:INTEGER);
    property HANDLE: INTEGER; read;
end;
TFileStream = class(THandleStream)
    constructor CREATE(FILENAME:STRING;MODE:WORD);
end;
TGraphicsObject = class(TPersistent)
    property ONCHANGE: TNOTIFYEVENT; read write;
end;
TFontStyle = (fsBold, fsItalic, fsUnderline, fsStrikeOut);
TFontStyles = set of TFontStyle;
TFont = class(TGraphicsObject)
    constructor Create;
    property Handle: Integer; read;
    property Color: Integer; read write;
    property Height: Integer; read write;
    property Name: string; read write;
    property Pitch: Byte; read write;
    property Size: Integer; read write;
    property PixelsPerInch: Integer; read write;
    property Style: TFontStyles; read write;
end;
TCanvas = class(TPersistent)
    procedure Arc(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
    procedure Chord(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
    procedure Draw(X, Y: Integer; Graphic: TGraphic);
```

```
    procedure Ellipse(X1, Y1, X2, Y2: Integer);
    procedure FloodFill(X, Y: Integer; Color: TColor; FillStyle: Byte);
    procedure LineTo(X, Y: Integer);
    procedure MoveTo(X, Y: Integer);
    procedure Pie(X1, Y1, X2, Y2, X3, Y3, X4, Y4: Integer);
    procedure Rectangle(X1, Y1, X2, Y2: Integer);
    procedure Refresh;
    procedure RoundRect(X1, Y1, X2, Y2, X3, Y3: Integer);
    function TextHeight(Text: string): Integer;
    procedure TextOut(X, Y: Integer; Text: string);
    function TextWidth(Text: string): Integer;
    property Handle: Integer; read write;
    property Pixels: Integer Integer Integer; read write;
    property Brush: TBrush; read;
    property CopyMode: Byte; read write;
    property Font: TFont; read;
    property Pen: TPen; read;
end;
TPenMode = (pmBlack, pmWhite, pmNop, pmNot, pmCopy, pmNotCopy, pmMergePenNot,
pmMaskPenNot, pmMergeNotPen, pmMaskNotPen, pmMerge, pmNotMerge, pmMask,
pmNotMask, pmXor, pmNotXor);
TPenStyle = (psSolid, psDash, psDot, psDashDot, psDashDotDot, psClear, psInsideFrame);
TPen = class(TGraphicsObject)
    constructor CREATE;
    property COLOR: TCOLOR; read write;
    property MODE: TPENMODE; read write;
    property STYLE: TPENSTYLE; read write;
    property WIDTH: INTEGER; read write;
end;
TBrushStyle = (bsSolid, bsClear, bsHorizontal, bsVertical, bsFDiagonal, bsBDiagonal, bsCross,
bsDiagCross);
TBrush = class(TGraphicsObject)
    constructor CREATE;
    property COLOR: TCOLOR; read write;
    property STYLE: TBRUSHSTYLE; read write;
end;
TGraphic = class(TPersistent)
    procedure LoadFromFile(const Filename: string);
    procedure SaveToFile(const Filename: string);
    property Empty: Boolean; read write;
    property Height: Integer; read write;
    property Modified: Boolean; read write;
    property Width: Integer; read write;
    property OnChange: TNotifyEvent; read write;
```

```
end;
TBitmap = class(TGraphic)
   procedure LoadFromStream(Stream: TStream);
   procedure SaveToStream( Stream: TStream);
   property Canvas: TCanvas; read write;
   property Handle: HBITMAP; read write;
end;
TAlign = (alNone, alTop, alBottom, alLeft, alRight, alClient);
TControl = class(TComponent)
   constructor Create(AOwner: TComponent);
   procedure BringToFront;
   procedure Hide;
   procedure Invalidate;
   procedure refresh;
   procedure Repaint;
   procedure SendToBack;
   procedure Show;
   procedure Update;
   procedure SetBounds(x,y,w,h: Integer);
   property Left: Integer; read write;
   property Top: Integer; read write;
   property Width: Integer; read write;
   property Height: Integer; read write;
   property Hint: String; read write;
   property Align: TAlign; read write;
   property ClientHeight: Longint; read write;
   property ClientWidth: Longint; read write;
   property ShowHint: Boolean; read write;
   property Visible: Boolean; read write;
   property Enabled: Boolean; read write;
   property Hint: String; read write;
   property Cursor: Integer; read write;
end;
TWinControl = class(TControl)
   property Parent: TWinControl; read write;
   property Handle: Longint; read write;
   property Showing: Boolean; read;
   property TabOrder: Integer; read write;
   property TabStop: Boolean; read write;
   function CANFOCUS:BOOLEAN;
   function FOCUSED:BOOLEAN;
   property CONTROLS[Index: INTEGER]: TCONTROL; read;
   property CONTROLCOUNT: INTEGER; read;
end;
```

```
TGraphicControl = class(TControl)
end;
TCustomControl = class(TWinControl)
end;
TScrollBarKind = (sbHorizontal, sbVertical);
TScrollBarInc = SmallInt;
TCONTROLSCROLLBAR = class('TPersistent')
   property KIND: TSCROLLBARKIND; read;
   property SCROLLPOS: INTEGER; read
   property MARGIN: WORD; read write;
   property INCREMENT: TSCROLLBARINC; read write;
   property RANGE: INTEGER; read write;
   property POSITION: INTEGER; read write;
   property TRACKING: BOOLEAN; read write;
   property VISIBLE: BOOLEAN; read write;
end;
TScrollingWinControl = class(TWinControl)
   procedure SCROLLINVIEW(ACONTROL:TCONTROL);
   property HORZSCROLLBAR: TCONTROLSCROLLBAR; read write;
   property VERTSCROLLBAR: TCONTROLSCROLLBAR; read write;
end;
TFormBorderStyle = (bsNone, bsSingle, bsSizeable, bsDialog, bsToolWindow, bsSizeToolWin);
TBorderIcon = (biSystemMenu, biMinimize, biMaximize, biHelp);
TBorderIcons = set of TBorderIcon;
TPosition = (poDesigned, poDefault, poDefaultPosOnly, poDefaultSizeOnly, poScreenCenter,
poDesktopCenter, poMainFormCenter, poOwnerFormCenter);
TCloseAction = (caNone, caHide, caFree, caMinimize);
TCloseEvent = procedure(Sender: TObject; var Action: TCloseAction);
TCloseQueryEvent = procedure(Sender: TObject; var CanClose: Boolean);
TEShiftState = (ssShift, ssAlt, ssCtrl, ssLeft, ssRight, ssMiddle, ssDouble);
TShiftState = set of TEShiftState;
TKeyEvent = procedure (Sender: TObject; var Key: Word; Shift: TShiftState);
TKeyPressEvent = procedure(Sender: TObject; var Key: Char);
TForm = class(TScrollingWinControl)
   constructor CREATENEW(AOWNER:TCOMPONENT; Dummy: Longint);
   procedure CLOSE;
   procedure HIDE;
   procedure SHOW;
   function SHOWMODAL:INTEGER;
   procedure RELEASE;
   property ACTIVE: BOOLEAN; read;
   property ACTIVECONTROL: TWINCONTROL; read write;
   property BORDERICONS: TBorderIcons; read write;
   property BORDERSTYLE: TFORMBORDERSTYLE; read write;
```

```
    property CAPTION: STRING; read write;
    property AUTOSCROLL: BOOLEAN; read write;
    property COLOR: TCOLOR; read write;
    property FONT: TFONT; read write;
    property FORMSTYLE: TFORMSTYLE; read write;
    property KEYPREVIEW: BOOLEAN; read write;
    property POSITION: TPOSITION; read write;
    property ONACTIVATE: TNOTIFYEVENT; read write;
    property ONCLICK: TNOTIFYEVENT; read write;
    property ONDBLCLICK: TNOTIFYEVENT; read write;
    property ONDROPDOWN: TNotifyEvent; read write;
    property ONCLOSE: TCLOSEEVENT; read write;
    property ONCLOSEQUERY: TCLOSEQUERYEVENT; read write;
    property ONCREATE: TNOTIFYEVENT; read write;
    property ONDESTROY: TNOTIFYEVENT; read write;
    property ONDEACTIVATE: TNOTIFYEVENT; read write;
    property ONHIDE: TNOTIFYEVENT; read write;
    property ONKEYDOWN: TKEYEVENT; read write;
    property ONKEYPRESS: TKEYPRESSEVENT; read write;
    property ONKEYUP: TKEYEVENT; read write;
    property ONRESIZE: TNOTIFYEVENT; read write;
    property ONSHOW: TNOTIFYEVENT; read write;
end;
TCustomLabel = class(TGraphicControl)
end;
TAlignment = (taLeftJustify, taRightJustify, taCenter);
TLabel = class(TCustomLabel)
    property ALIGNMENT: TAlignment; read write;
    property AUTOSIZE: Boolean; read write;
    property CAPTION: String; read write;
    property COLOR: TColor; read write;
    property FOCUSCONTROL: TWinControl; read write;
    property FONT: TFont; read write;
    property WORDWRAP: Boolean; read write;
    property ONCLICK: TNotifyEvent; read write;
    property ONDBLCLICK: TNotifyEvent; read write;
end;
TCustomEdit = class(TWinControl)
    procedure CLEAR;
    procedure CLEARSELECTION;
    procedure SELECTALL;
    property MODIFIED: BOOLEAN; read write;
    property SELLENGTH: INTEGER; read write;
    property SELSTART: INTEGER; read write;
```

```
    property SELTEXT: STRING; read write;
    property TEXT: string; read write;
end;
TBorderStyle = TFormBorderStyle;
TEditCharCase = (ecNormal, ecUpperCase, ecLowerCase);
TEdit = class(TCustomEdit)
    property AUTOSELECT: Boolean; read write;
    property AUTOSIZE: Boolean; read write;
    property BORDERSTYLE: TBorderStyle; read write;
    property CHARCASE: TEditCharCase; read write;
    property COLOR: TColor; read write;
    property FONT: TFont; read write;
    property HIDESELECTION: Boolean; read write;
    property MAXLENGTH: Integer; read write;
    property PASSWORDCHAR: Char; read write;
    property READONLY: Boolean; read write;
    property TEXT: string; read write;
    property ONCHANGE: TNotifyEvent; read write;
    property ONCLICK: TNotifyEvent; read write;
    property ONDBLCLICK: TNotifyEvent; read write;
    property ONKEYDOWN: TKeyEvent; read write;
    property ONKEYPRESS: TKeyPressEvent; read write;
    property ONKEYUP: TKeyEvent; read write;
end;
TCustomMemo = class(TCustomEdit)
    property LINES: TSTRINGS; read write;
end;
TScrollStyle = (ssNone, ssHorizontal, ssVertical, ssBoth);
TMemo = class(TMemo)
    property LINES: TSTRINGS; read write;
    property ALIGNMENT: TAlignment; read write;
    property BORDERSTYLE: TBorderStyle; read write;
    property COLOR: TColor; read write;
    property FONT: TFont; read write;
    property HIDESELECTION: Boolean; read write;
    property MAXLENGTH: Integer; read write;
    property READONLY: Boolean; read write;
    property SCROLLBARS: TScrollStyle; read write;
    property WANTRETURNS: Boolean; read write;
    property WANTTABS: Boolean; read write;
    property WORDWRAP: Boolean; read write;
    property ONCHANGE: TNotifyEvent; read write;
    property ONCLICK: TNotifyEvent; read write;
    property ONDBLCLICK: TNotifyEvent; read write;
```

```
    property ONKEYDOWN: TKeyEvent; read write;
    property ONKEYPRESS: TKeyPressEvent; read write;
    property ONKEYUP: TKeyEvent; read write;
end;
TCustomComboBox = class(TWinControl)
    property DROPPEDDOWN: BOOLEAN; read write;
    property ITEMS: TSTRINGS; read write;
    property ITEMINDEX: INTEGER; read write;
end;
TComboBoxStyle = (csDropDown, csSimple, csDropDownList, csOwnerDrawFixed,
csOwnerDrawVariable);
TComboBox = class(TCustomComboBox)
    property STYLE: TComboBoxStyle; read write;
    property COLOR: TColor; read write;
    property DROPDOWNCOUNT: Integer; read write;
    property FONT: TFont; read write;
    property MAXLENGTH: Integer; read write;
    property SORTED: Boolean; read write;
    property TEXT: string; read write;
    property ONCHANGE: TNotifyEvent; read write;
    property ONCLICK: TNotifyEvent; read write;
    property ONDBLCLICK: TNotifyEvent; read write;
    property ONKEYDOWN: TKeyEvent; read write;
    property ONKEYPRESS: TKeyPressEvent; read write;
    property ONKEYUP: TKeyEvent; read write;
end;
TButtonControl = class(TWinControl)
end;
TButton = class(TButtonControl)
    property CANCEL: BOOLEAN; read write;
    property CAPTION: String; read write;
    property DEFAULT: BOOLEAN; read write;
    property FONT: TFont; read write;
    property MODALRESULT: LONGINT; read write;
    property ONCLICK: TNotifyEvent; read write;
end;
TCustomCheckBox = class(TButtonControl)
end;
TCheckBoxState = (cbUnchecked, cbChecked, cbGrayed);
TCheckBox = class(TCustomCheckBox)
    property ALIGNMENT: TAlignment; read write;
    property ALLOWGRAYED: Boolean; read write;
    property CAPTION: String; read write;
    property CHECKED: Boolean; read write;
```

```
    property COLOR: TColor; read write;
    property FONT: TFont; read write;
    property STATE: TCheckBoxState; read write;
    property ONCLICK: TNotifyEvent; read write;
end;
TRadioButton = class(TButtonControl)
    property ALIGNMENT: TALIGNMENT; read write;
    property CAPTION: String; read write;
    property CHECKED: BOOLEAN; read write;
    property COLOR: TColor; read write;
    property FONT: TFont; read write;
    property ONCLICK: TNotifyEvent; read write;
    property ONDBLCLICK: TNotifyEvent; read write;
end;
TCustomListBox = class(TWinControl)
    property ITEMS: TSTRINGS; read write;
    property ITEMINDEX: INTEGER; read write;
    property SELCOUNT: INTEGER; read;
    property SELECTED[Index: INTEGER]: BOOLEAN; read write;
end;
TListBoxStyle = (lbStandard, lbOwnerDrawFixed, lbOwnerDrawVariable);
TListBox = class(TCustomListBox)
    property BORDERSTYLE: TBorderStyle; read write;
    property COLOR: TColor; read write;
    property FONT: TFont; read write;
    property MULTISELECT: Boolean; read write;
    property SORTED: Boolean; read write;
    property STYLE: TListBoxStyle; read write;
    property ONCLICK: TNotifyEvent; read write;
    property ONDBLCLICK: TNotifyEvent; read write;
    property ONKEYDOWN: TKeyEvent; read write;
    property ONKEYPRESS: TKeyPressEvent; read write;
    property ONKEYUP: TKeyEvent; read write;
end;
TBevelShape = (bsBox, bsFrame, bsTopLine, bsBottomLine, bsLeftLine, bsRightLine,bsSpacer);
TBevelStyle = (bsLowered, bsRaised);
TBevel = class(TGraphicControl)
    property SHAPE: TBEVELSHAPE; read write;
    property STYLE: TBEVELSTYLE; read write;
end;
TCustomPanel = class(TCustomControl)
end;
TPanelBevel = (bvNone, bvLowered, bvRaised,bvSpace);
TBevelWidth = Longint;
```

```
TBorderWidth = Longint;
TPanel = class(TCustomPanel)
   property ALIGNMENT: TAlignment; read write;
   property BEVELINNER: TPanelBevel; read write;
   property BEVELOUTER: TPanelBevel; read write;
   property BEVELWIDTH: TBevelWidth; read write;
   property BORDERWIDTH: TBorderWidth; read write;
   property BORDERSTYLE: TBorderStyle; read write;
   property CAPTION: String; read write;
   property COLOR: TColor; read write;
   property FONT: TFont; read write;
   property ONCLICK: TNotifyEvent; read write;
   property ONDBLCLICK: TNotifyEvent; read write;
end;
TNewStaticText = class(TWinControl)
   property AUTOSIZE: BOOLEAN; read write;
   property CAPTION: String; read write;
   property COLOR: TColor; read write;
   property FOCUSCONTROL: TWinControl; read write;
   property FONT: TFont; read write;
   property SHOWACCELCHAR: Boolean; read write;
   property WORDWRAP: Boolean; read write;
   property ONCLICK: TNotifyEvent; read write;
   property ONDBLCLICK: TNotifyEvent; read write;
end;
TNewCheckListBox = class(TCustomListBox)
   function AddCheckBox(const ACaption, ASubItem: string; ALevel: Byte; AChecked, AEnabled,
AHasInternalChildren, ACheckWhenParentChecked: Boolean; AObject: TObject): Integer;
   function
ADDGROUP(ACAPTION,ASUBITEM:STRING;ALEVEL:BYTE;AOBJECT:TOBJECT):INTE
GER;
   function AddRadioButton(const ACaption, ASubItem: string; ALevel: Byte; AChecked,
AEnabled: Boolean; AObject: TObject): Integer;
   property CHECKED[Index: INTEGER]: BOOLEAN; read write;
   property STATE[Index: INTEGER]: TCHECKBOXSTATE; read write;
   property ITEMENABLED[Index: INTEGER]: BOOLEAN; read write;
   property ITEMLEVEL[Index: INTEGER]: BYTE; read;
   property ITEMOBJECT[Index: INTEGER]: TOBJECT; read write;
   property ITEMSUBITEM[Index: INTEGER]: STRING; read write;
   property ALLOWGRAYED: BOOLEAN; read write;
   property FLAT: BOOLEAN; read write;
   property MINITEMHEIGHT: INTEGER; read write;
   property OFFSET: INTEGER; read write;
   property MULTISELECT: BOOLEAN; read write;
```

```
    property ONCLICKCHECK: TNOTIFYEVENT; read write;
    property BORDERSTYLE: TBORDERSTYLE; read write;
    property COLOR: TColor; read write;
    property FONT: TFont; read write;
    property SORTED: Boolean; read write;
    property STYLE: TListBoxStyle; read write;
    property ONCLICK: TNotifyEvent; read write;
    property ONDBLCLICK: TNotifyEvent; read write;
    property ONKEYDOWN: TKeyEvent; read write;
    property ONKEYPRESS: TKeyPressEvent; read write;
    property ONKEYUP: TKeyEvent; read write;
    property SHOWLINES: BOOLEAN; read write;
    property WANTTABS: BOOLEAN; read write;
end;
TNewProgressBar = class(TWinControl)
    property MIN: LONGINT; read write;
    property MAX: LONGINT; read write;
    property POSITION: LONGINT; read write;
end;
TRichEditViewer = class(TMemo)
    property RTFTEXT: STRING; write;
    property USERICHEDIT: BOOLEAN; read write;
end;
TPasswordEdit = class(TCustomEdit)
    property AUTOSELECT: Boolean; read write;
    property AUTOSIZE: Boolean; read write;
    property BORDERSTYLE: TBorderStyle; read write;
    property COLOR: TColor; read write;
    property FONT: TFont; read write;
    property HIDESELECTION: Boolean; read write;
    property MAXLENGTH: Integer; read write;
    property Password: Boolean; read write;
    property READONLY: Boolean; read write;
    property TEXT: string; read write;
    property ONCHANGE: TNotifyEvent; read write;
    property ONCLICK: TNotifyEvent; read write;
    property ONDBLCLICK: TNotifyEvent; read write;
    property ONKEYDOWN: TKeyEvent; read write;
    property ONKEYPRESS: TKeyPressEvent; read write;
    property ONKEYUP: TKeyEvent; read write;
end;
TCustomFolderTreeView = class(TWinControl)
    procedure ChangeDirectory(const Value: String; const CreateNewItems: Boolean);
    procedure CreateNewDirectory(const ADefaultName: String);
```

```
    property: Directory: String; read write;
end;
TFolderRenameEvent = procedure(Sender: TCustomFolderTreeView; var NewName: String; var Accept: Boolean);
TFolderTreeView = class(TCustomFolderTreeView)
    property OnChange: TNotifyEvent; read write;
    property OnRename: TFolderRenameEvent; read write;
end;
TStartMenuFolderTreeView = class(TCustomFolderTreeView)
    procedure SetPaths(const AUserPrograms, ACommonPrograms, AUserStartup, ACommonStartup: String);
    property OnChange: TNotifyEvent; read write;
    property OnRename: TFolderRenameEvent; read write;
end;
TBitmapImage = class(TGraphicControl)
    property AutoSize: Boolean; read write;
    property BackColor: TColor; read write;
    property Center: Boolean; read write;
    property Bitmap: TBitmap; read write;
    property ReplaceColor: TColor; read write;
    property ReplaceWithColor: TColor; read write;
    property Stretch: Boolean; read write;
end;
TNewNotebook = class(TWinControl)
    function FindNextPage(CurPage: TNewNotebookPage; GoForward: Boolean): TNewNotebookPage;
    property PageCount: Integer; read write;
    property Pages[Index: Integer]: TNewNotebookPage; read;
    property ActivePage: TNewNotebookPage; read write;
end;
TNewNotebookPage = class(TCustomControl)
    property Color: TColor; read write;
    property Notebook: TNewNotebook; read write;
    property PageIndex: Integer; read write;
end;
TWizardPageNotifyEvent = procedure(Sender: TWizardPage);
TWizardPageButtonEvent = function(Sender: TWizardPage): Boolean;
TWizardPageCancelEvent = procedure(Sender: TWizardPage; var ACancel, AConfirm: Boolean);
TWizardPageShouldSkipEvent = function(Sender: TWizardPage): Boolean;
TWizardPage = class(TComponent)
    property ID: Integer; read;
    property Caption: String; read write;
    property Description: String; read write;
    property Surface: TNewNotebookPage; read write;
```

```
    property SurfaceHeight: Integer; read write;
    property SurfaceWidth: Integer; read write;
    property OnActivate: TWizardPageNotifyEvent; read write;
    property OnBackButtonClick: TWizardPageButtonEvent; read write;
    property OnCancelButtonClick: TWizardPageCancelEvent; read write;
    property OnNextButtonClick: TWizardPageButtonEvent; read write;
    property OnShouldSkipPage: TWizardPageShouldSkipEvent; read write;
end;
TInputQueryWizardPage = class(TWizardPage)
    function Add(const APrompt: String; const APassword: Boolean): Integer;
    property Edits[Index: Integer]: TPasswordEdit; read;
    property PromptLabels[Index: Integer]: TNewStaticText; read;
    property SubCaptionLabel: TNewStaticText; read;
    property Values[Index: Integer]: String; read write;
end;
TInputOptionWizardPage = class(TWizardPage)
    function Add(const ACaption: String): Integer;
    function AddEx(const ACaption: String; const ALevel: Byte; const AExclusive: Boolean):
Integer;
    property CheckListBox: TNewCheckListBox; read;
    property SelectedValueIndex: Integer; read write;
    property SubCaptionLabel: TNewStaticText; read;
    property Values[Index: Integer]: Boolean; read write;
end;
TInputDirWizardPage = class(TWizardPage)
    function Add(const APrompt: String): Integer;
    property Buttons[Index: Integer]: TButton; read;
    property Edits[Index: Integer]: TEdit; read;
    property PromptLabels[Index: Integer]: TNewStaticText; read;
    property SubCaptionLabel: TNewStaticText; read;
    property Values[Index: Integer]: String; read write;
end;
TInputFileWizardPage = class(TWizardPage)
    function Add(const APrompt, AFilter, ADefaultExtension: String): Integer;
    property Buttons[Index: Integer]: TButton; read;
    property Edits[Index: Integer]: TEdit; read;
    property PromptLabels[Index: Integer]: TNewStaticText; read;
    property SubCaptionLabel: TNewStaticText; read;
    property Values[Index: Integer]: String; read write;
end;
TOutputMsgWizardPage = class(TWizardPage)
    property MsgLabel: TNewStaticText; read;
end;
TOutputMsgMemoWizardPage = class(TWizardPage)
```

```
    property RichEditViewer: TRichEditViewer; read;
    property SubCaptionLabel: TNewStaticText; read;
end;
TOutputProgressWizardPage = class(TWizardPage)
    procedure Hide;
    property Msg1Label: TNewStaticText; read;
    property Msg2Label: TNewStaticText; read;
    property ProgressBar: TNewProgressBar; read;
    procedure SetProgress(const Position, Max: Longint);
    procedure SetText(const Msg1, Msg2: String);
    procedure Show;
end;
TUIStateForm = class(TForm)
end;
TSetupForm = class(TUIStateForm)
    procedure Center;
    procedure CenterInsideControl(const Ctl: TWinControl; const InsideClientArea: Boolean);
end;
TMainForm = class(TSetupForm)
    procedure ShowAboutBox;
end;
TWizardForm = class(TSetupForm)
    property CANCELBUTTON: TBUTTON; read;
    property NEXTBUTTON: TBUTTON; read;
    property BACKBUTTON: TBUTTON; read;
    property NOTEBOOK1: TNOTEBOOK; read;
    property NOTEBOOK2: TNOTEBOOK; read;
    property WelcomePage: TNewNotebookPage; read;
    property InnerPage: TNewNotebookPage; read;
    property FinishedPage: TNewNotebookPage; read;
    property LicensePage: TNewNotebookPage; read;
    property PasswordPage: TNewNotebookPage; read;
    property InfoBeforePage: TNewNotebookPage; read;
    property UserInfoPage: TNewNotebookPage; read;
    property SelectDirPage: TNewNotebookPage; read;
    property SelectComponentsPage: TNewNotebookPage; read;
    property SelectProgramGroupPage: TNewNotebookPage; read;
    property SelectTasksPage: TNewNotebookPage; read;
    property ReadyPage: TNewNotebookPage; read;
    property PreparingPage: TNewNotebookPage; read;
    property InstallingPage: TNewNotebookPage; read;
    property InfoAfterPage: TNewNotebookPage; read;
    property DISKSPACELABEL: TNewStaticText; read;
    property DIREDIT: TEDIT; read;
```

property GROUPEDIT: TEDIT; read;
property NOICONSCHECK: TCHECKBOX; read;
property PASSWORDLABEL: TNewStaticText; read;
property PASSWORDEDIT: TPasswordEdit; read;
property PASSWORDEDITLABEL: TNewStaticText; read;
property READYMEMO: TMEMO; read;
property TYPESCOMBO: TCOMBOBOX; read;
property BEVEL: TBEVEL; read;
property WizardBitmapImage: TBitmapImage; read;
property WELCOMELABEL1: TNewStaticText; read;
property INFOBEFOREMEMO: TRICHEDITVIEWER; read;
property INFOBEFORECLICKLABEL: TNewStaticText; read;
property MAINPANEL: TPANEL; read;
property BEVEL1: TBEVEL; read;
property PAGENAMELABEL: TNewStaticText; read;
property PAGEDESCRIPTIONLABEL: TNewStaticText; read;
property WizardSmallBitmapImage: TBitmapImage; read;
property READYLABEL: TNewStaticText; read;
property FINISHEDLABEL: TNewStaticText; read;
property YESRADIO: TRADIOBUTTON; read;
property NORADIO: TRADIOBUTTON; read;
property WizardBitmapImage2: TBitmapImage; read;
property WELCOMELABEL2: TNewStaticText; read;
property LICENSELABEL1: TNewStaticText; read;
property LICENSEMEMO: TRICHEDITVIEWER; read;
property INFOAFTERMEMO: TRICHEDITVIEWER; read;
property INFOAFTERCLICKLABEL: TNewStaticText; read;
property COMPONENTSLIST: TNEWCHECKLISTBOX; read;
property COMPONENTSDISKSPACELABEL: TNewStaticText; read;
property BEVELEDLABEL: TNewStaticText; read;
property STATUSLABEL: TNewStaticText; read;
property FILENAMELABEL: TNewStaticText; read;
property PROGRESSGAUGE: TNEWPROGRESSBAR; read;
property SELECTDIRLABEL: TNewStaticText; read;
property SELECTSTARTMENUFOLDERLABEL: TNewStaticText; read;
property SELECTCOMPONENTSLABEL: TNewStaticText; read;
property SELECTTASKSLABEL: TNewStaticText; read;
property LICENSEACCEPTEDRADIO: TRADIOBUTTON; read;
property LICENSENOTACCEPTEDRADIO: TRADIOBUTTON; read;
property USERINFONAMELABEL: TNewStaticText; read;
property USERINFONAMEEDIT: TEDIT; read;
property USERINFOORGLABEL: TNewStaticText; read;
property USERINFOORGEDIT: TEDIT; read;
property PreparingErrorBitmapImage: TBitmapImage; read;

```
    property PREPARINGLABEL: TNewStaticText; read;
    property FINISHEDHEADINGLABEL: TNewStaticText; read;
    property USERINFOSERIALLABEL: TNewStaticText; read;
    property USERINFOSERIALEDIT: TEDIT; read;
    property TASKSLIST: TNEWCHECKLISTBOX; read;
    property RUNLIST: TNEWCHECKLISTBOX; read;
    property DirBrowseButton: TButton; read;
    property GroupBrowseButton: TButton; read;
    property SelectDirBitmapImage: TBitmapImage; read;
    property SelectGroupBitmapImage: TBitmapImage; read;
    property SelectDirBrowseLabel: TNewStaticText; read;
    property SelectStartMenuFolderBrowseLabel: TNewStaticText; read;
    property CurPageID: Integer; read;
    function ADJUSTLABELHEIGHT(ALABEL:TNewStaticText):INTEGER;
    procedure INCTOPDECHEIGHT(ACONTROL:TCONTROL;AMOUNT:INTEGER);
end;
```

使用自定义向导页

该 Pascal 脚本允许你添加自定义向导页到安装程序的向导中。这包括用于公共查询的“预构建”向导页和完全受你控制的自定义向导页。

要使用自定义向导面，先在你的 InitializeWizard 事件函数中创建它们。你可以使用 CreateInput...页和 CreateOutput...页函数创建预构建页，或用 CreateCustomPage 函数创建一个空向导页。查看支持函数主题获取所有创建自定义页函数的列表和说明。

每个页创建之后，你要添加控件，可以调用预构建页指定方法，或手动在向导页中创建控件。大多数创建的向导页函数会获取“page ID”作为它们的第一个参数；这用来定义新创建的页将放置到哪个已有的页之后。这里有几个方法查找已有向导页的“page ID”。你自己创建的向导页有控制它们 ID 的 ID 属性。内置的向导页有预定义的 ID。例如，欢迎向导页的 ID 是 wpWelcome。查阅支持函数主题获取所有预定义 ID 的列表。

自定义向导页创建后，安装程序将象它内置的向导页一样显示和处理这些自定义向导页。这包含有关象 NextButtonClick 和 ShouldSkipPage 的事件函数的所有页面的调用。

在任何安装期间，你可以接收用户输入或使用预构建页特殊属性的值，或通过使用你自己创建的控件属性值。

打开 Inno Setup 安装目录下的“Examples”子目录，找到“CodeDlg.iss”脚本，这个示例说明怎样使用预构建自定义向导页和事件函数。打开“CodeClasses.iss”脚本获取怎样使用完全自定义向导页和控件的示例。

使用 DLL

Pascal 脚本可以调用外部 DLL 函数。这包括标准 Windows DLL 同倍的标准 Win32 API 函数，和自制 DLL 中的函数 (至于如何自制 DLL 超出了本帮助文件讨论范围)。

要调用一个 DLL 函数，你应该先以正规的方法写入函数语法，而不是只写入函数内容，你使用“external”作为关键字指定一个 DLL。如果你的函数有下面示例的语法函数 function

A(B: Integer): Integer;，支持下面三种形态:
function A(B: Integer): Integer;
external '<dllfunctionname>@<dllfilename>';
function A(B: Integer): Integer;
external '<dllfunctionname>@<dllfilename> <callingconvention>';
function A(B: Integer): Integer;
external '<dllfunctionname>@<dllfilename> <callingconvention> <options>';
第一个形态指定将要用默认调用协定的 DLL 函数，这是“stdcall”。所有标准的 Win32 API 函数使用“stdcall”，就象大多数自定义 DLL 函数。
第二种形态指定将要用特殊调协定的 DLL 函数。有效的调用协定是: “stdcall” (默认)，“cdecl”，“pascal”和“register”。
第三种形态指定为载入 DLL 附加一个或多个选项，用空格隔开:
delayload
指定 DLL 应该延时载和。一般来说，Pascal 脚本在启动时检查 DLL 函数是否可以被调用，如果否，拒绝运行。如果你指定用“delayload”延时载入，就不会发生这种情况。如果你想调用不知道是否确实在运行的 DLL 函数时，请使用延时载入: 如果该 DLL 函数还不能调用，Pascal 脚本将仍旧运行，但生成一个表达式，你可以监视处理还没有运行的 DLL 函数。
setuponly
指定该 DLL 只在从安装程序运行脚本时载入。
uninstallonl
指定该 DLL 只在从卸载程序运行脚本时载入。
(第二种形态) 示例，如果 DLL 内部有一个名字“A2”的 DLL 函数，DLL 名字为“MyDll.dll”，并且 DLL 函数使用“stdcall”调用协定:
[Code]
function A(B: Integer): Integer;
external 'A2@MyDll.dll stdcall';
常量可以用于 DLL 文件名。在安装期间，也可以使用一个特殊的“files:”前缀告诉安装程序自动从 [Files] 段提取 DLL。示例:
[Files]
Source: "MyDll.dll"; Flags: dontcopy
[Code]
procedure MyDllFunc(hWnd: Integer; lpText, lpCaption: String; uType: Cardinal);
external 'MyDllFunc@files:MyDll.dll stdcall';
打开 Inno Setup 安装目录中“Examples”子目录，其中有一个使用 DLL 的示例。
“Examples”子目录还包含两个自定义 DLL 示例方案，一个用于 Microsoft Visual C++，另一个用于 Borland Delphi。

使用 COM 自动操作对象

Pascal 脚本可以通过 COM 自动操作对象支持访问 COM (同样也可以是 OLE 或 ActiveX) 程序。这允许你通过 COM Interop 访问象标准 Windows COM 服务器，自定义 COM 服务器，Visual Basic ActiveX DLL 和 .NET 系统。
这里是有关创建 COM 自动操作对象的支持函数: CreateOleObject 和 GetActiveOleObject。使用 CreateOleObject 可以用指定的类名创建一个新的 COM 对象。如果成功，这个函数返

回一个不同的类型变量。返回的值可以用于访问 COM 对象的方法和属性。这个访问不通过“后期连接”，这意味着不检查你正在尝试访问实际存在的方法和属性，直到安装程序运行时确实需要。
使用 GetActiveOleObject 可以用指定的类名连接到一个现有的 COM 对象。如果成功，这个函数返回一个不同的类型变量。在某些程序中，这可以用于侦测程序是否正在运行。
打开 Inno Setup 安装目录中“Examples”子目录下的“CodeAutomation.iss”文件，查阅使用 COM 自动操作对象的使用示例。
如果你提取一个 COM 自动库到临时位置，并想在用后能够删除它，确认你以后不想引用到这个库并调用 CoFreeUnusedLibraries。这个 Windows 函数将尝试撤销库，使你能够删除它。

第三部分 其它信息

1、常见问题解答

常见问题解答现在位于单独的一个文档中。请单击在安装 Inno Setup 时创建在开始菜单中的“Inno Setup 常见问题解答”快捷方式，或打开你的 Inno Setup 安装目录中的“isfaq.htm”文件。
获取更多最新的常见问题解答，转到 http://www.jrsoftware.org/isfaq.php

2、向导页

下面是在安装过程中可以通过激活它们的显示条件来显示的所有向导页的列表。

★ 欢迎
总是显示。
★ 许可协议
如果设置 LicenseFile 则显示。用户只能在选定“我同意该协议”的情况下才能继续到一页。
★ 密码
如果设置了 Password 则显示。用户只能在输入正确的密码后才能继续到下一页。
★ 信息
如果设置了 InfoBeforeFile 则显示。
★ 用户信息
如果设置 UserInfoPage 为 yes 则显示。
★ 选择目标位置
默认为显示，但可以通过 DisableDirPage 禁用。
★ 选择组件
如果设置了 [Components] 条目则显示。
★ 选择开始菜单文件夹
如果设置了 [Icons] 条目则显示，但可以通过 DisableProgramGroupPage 禁用。
★ 选择任务
如果设置了 [Tasks] 条目则显示，除非 [Tasks] 条目全部绑定到未在选择组件页中选定的组件。

★ 准备安装

默认为显示，但可以通过 DisableReadyPage 禁用。

★ 正在准备安装

通常安装程序不会在这个页面中停止。它只会在安装程序确定不能继续时显示。一般来说，它只会在一个或多个指定在 [Files] 段中的文件在使用 (被其它应用程序) 要进行替换或在下一次重新启动时删除时出现。即使这样，它告诉用户需要重新启动电脑，然后再运行安装程序。请注意，这个检查在后台安装时也会显示，但一些消息显示的消息框中，则不是一个向导页。

★ 正在安装

在实际安装进程期间显示。

★ 信息

如果设置了 InfoAfterFile 则显示。

★ 安装完成

默认为显示，用 DisableFinishedPage 在某些场合下禁用。

3、安装顺序

这里是实际安装进程开始后执行的各种安装任务的顺序:

处理 [InstallDelete]。

[UninstallDelete] 中的条目贮存在卸载日志中 (眼下暂时贮存在内存中)。

如果需要，创建应用程序目录。

处理 [Dirs]。

如果需要，预定了卸载日志文件名。

处理 [Files]。 (文件注册暂不发生。)

处理 [Icons]。

处理 [INI]。

处理 [Registry]。

需要注册的文件现在开始注册，除非系统需要重新启动，在某些场合下文件在系统重新启动之前不注册。

如果需要，开始创建程序的添加/删除程序条目。

[UninstallRun] 中的条目贮存到卸载日志中。

卸载程序 EXE 文件和日志文件最终定稿，并保存到磁盘。在此完成后，用户禁止取消安装，以及其后所有的错误将在返回之前不出现。

处理 [Run]，除用 postinstall 标记的条目外，在安装程序完成向导页显示后获取进程。

如果 ChangesAssociations 设置为 yes，立即刷新文件关联。

如果 ChangesEnvironment 设置为 yes，其它应用程序将接到通知。

安装程序所有条目处理顺序是按照它们在段中出现的顺序。

卸载程序按安装程序建立的相反顺序进行卸载，这是因为卸载日志是从结尾到开头进行解析。

在这个示例中:

[INI]

Filename: "{win}\MYPROG.INI"; Section: "InstallSettings"; Flags: uninsdeletesectionifempty

Filename: "{win}\MYPROG.INI"; Section: "InstallSettings"; Key: "InstallPath"; String: "{app}"; Flags: uninsdeleteentry

安装程序在卸载日志中先记录第一个条目的 uninsdeletesectionifempty 标记的数据，创建第二个条目的键，然后在卸载日志中记录 uninsdeleteentry 标记的数据。当卸载程序时，卸载程序先将处理 uninsdeleteentry 标记，删除条目，然后处理 uninsdeletesectionifempty 标记。

4、其它注意事项

如果安装程序在用户系统中发现 Windows 系统目录有写保护，{sys} 目录常量将翻译为用户的 Windows 目录，而不是系统目录。

要方便地自动更新你的应用程序，先让你的应用程序侦测新的 Setup.exe 版本，并使它定位或下载这个新的版本。然后进行自动升级，然后从你的应用程序按下面的命令行示例启动你的 Setup.exe:

/SP- /silent /noicons "/dir=c:\Program Files\My Program"

在启动 setup.exe 后，一有可能就退出你的应用程序。注意，为避免你更新你的 .exe 中出现的问题，当安装程序在后台或绝对后台运行时有一个自动重试功能。

另外你也可以使用 skipifsilent 和 skipifnotsilent 标记让你的应用程序意识到正在使用一个“/updated”参数，例如，显示一个友好的消息框提醒用户升级已完成。

命令行编译器执行

脚本也可以用安装程序编译器从命令行方式进行编译。命令行用法看下面:

compiler /cc <脚本名>

示例:

compil32 /cc "c:\isetup\samples\my script.iss"

象上述的例子中，包含空格的文件名必须用引号。

从命令行运行安装程序编译器不支持普通的进度显示或所有错误消息显示。如果编译成功，安装程序编译器将返回代码 0 退出，如果命令行参数无效，返回 1，编译失败，返回 2。作为选择，你可以用控制台模式编译器 ISCC.exe 编译脚本。命令行用法见下面:

iscc [options] <脚本名>

可从标准输入中读取:

iscc [options] -

示例:

iscc "c:\isetup\samples\my script.iss"

象上述的例子中，包含空格的文件名必须用引号。

有效的选项是: “/O”用来指定输出路径 (高于在脚本中的任何 OutputDir 设置)，“/F”用来指定输出文件名 (高于在脚本中的任何 OutputBaseFilename 设置)，“/Q”用来静寂编译 (仅打印错误消息)，“/?”用来显示帮助。

示例:

iscc /Q /O"My Output" /F"MyProgram-1.0" "c:\isetup\samples\my script.iss"

如果编译成功，ISCC 将返回 0，如果命令行参数无效或发生内部错误，返回 1，如果编译失败，返回 2。

安装脚本向导可以从命令行启动。命令行用法看下面:

compiler /wizard <向导名> <脚本名>

示例:

compil32 /wizard "MyProg Script Wizard" "c:\temp.iss"

象上述的例子中，包含空格的文件名必须用引号。

从命令行运行向导不支持所有错误消息。如果没有发生错误并保存生成的脚本文件到指定的文件名，安装脚本向导退出代码返回 0，如果命令行参数无效，返回 1，如果生成的脚本文件不能保存，返回 2。如果用户取消安装脚本向导，返回退出代码 0，且不保存脚本文件。

5、安装命令行参数

安装程序接受可选的命令行参数。这些对于系统管理员以及其它程序调用安装程序时有用。

/SP-

在安装开始时禁用“这将安装... 你想继续吗？”的提示，当然，如果 [Setup] 段的指示 DisableStartupPrompt 设为 yes 时，这将无效。

/SILENT, /VERYSILENT

告诉安装程序后台或完全后台运行。当安装程序在后台运行时，向导和背景窗口将不显示，但安装进度窗口显示。当安装程序在完全后台安装时，这个安装进度窗口也不显示。其它的事件被象正常安装一样，例如安装期间显示错误消息框，以及启动时提示 (如果你没有在 DisableStartupPrompt 或上面说明的“/SP-”命令行选项中指定) 。

如果需要重新启动，以及未使用“/NORESTART”命令行 (看下面)，并且安装程序在后台运行，将显示“立即重新启动吗”消息框。如果在完全后台安装模式，将在不询问的情况下重新启动。

/SUPPRESSMSGBOXES

命令安装程序可禁止消息框。只在用“/SILENT”和“/VERYSILENT”编译时有效。

在这种情况下默认会有一个选择:

-在“保留新文件吗”中会选择是。

-在“文件存在，确认覆盖。”中会选择否。

-在中断/重试情况下会选择中断。

-在重试/取消情况下会选择取消。

- 在 DiskSpaceWarning/DirExists/DirDoesntExist/NoUninstallWarning/ExitSetupMessage/ConfirmUninstall 情况下会选择是 (继续)。

-在 FinishedRestartMessage/UninstalledAndNeedsRestart 情况下会选择是 (重启)。

有 5 种消息不能禁止:

-中断安装程序消息框。

-“退出安装程序吗”消息框。

-当安装程序需要在新磁盘安装但新磁盘未找到时显示的 FileNotInDir2 消息框。

-在安装程序或卸载程序不能读取命令行参数之前显示的任何 (错误) 消息。

-用 [Code] 段支持函数 MsgBox 显示的消息框。

/LOG

使安装程序在用户的临时目录创建一个记录安装程序安装期间 [Run] 段详细动作的日志文件。这对于安装调试有帮助。例如，在你认为一个文件应该被替换，但你怀疑它没有被替换时 (反之亦然)，该日志文件将告诉你这个文件是否确实被跳过，并且为什么会跳过。日志文件若要当前日期的唯一名字创建(它不覆盖或添加到现有的文件中)。

日志文件中包含的信息是用自然语言，非常容易看懂，对于用户来说不会难以理解。也不用进行机器分析；文件的格式是不预先通知服务改变。

/LOG="filename"

与 /LOG 相同，另外它还允许你为日志文件指定一个固定的路径/文件。如果相同的名字在路径中已经存在则将被覆盖，如果不能创建文件，安装程序将带错误消息中断。

/NOCANCEL

通过禁用“取消”按钮和忽略在“关闭”按钮的单击动作，来防止用户在安装进行时取消操作，与 /SILENT 或 /VERYSILENT 一起使用有用。

/NORESTART

告诉安装程序即使需要重新启动，也不重新启动。

/RESTARTEXITCODE=exit code

指定当需要重新启动时安装程序返回的自定义退出代码。通常跟随“/NORESTART”。同时请查阅安装退出代码。

/LOADINF="filename"

告诉安装程序在选中命令行后从指定的文件载入设置。这个文件可以是使用下面说明的“/SAVEINF=”命令提供的。

如果文件名包含空格，不要忘记使用引号。

/SAVEINF="filename"

告诉安装程序保存安装设置到指定的文件。

如果文件名包含空格，不要忘记使用引号。

/LANG=language

指定要使用的语言。指定的语言是 [Languages] 段条目中指定的语言内部名字。

当使用了一个有效的 /LANG 参数时，选择语言对话框将被禁用。

/DIR="x:\dirname"

不考虑在目标位置向导页中显示的默认目录名。必须指定一个完整的路径。

/GROUP="文件夹名"

不考虑在选择开始菜单文件夹向导而中显示的默认文件夹名。如果 [Setup] 段指示 DisableProgramGroupPage 设为 yes，这条命令行参数将被忽略。

/NOICONS

告诉安装程序在初始时选中选择开始菜单文件夹向导页中的“不创建任何图标”选择框。

/COMPONENTS="用逗号分隔的组件名列表"

不考虑默认组件设置。使用这个命令行参数使安装程序自动选择一个自定义安装类型。

/PASSWORD=密码

指定要使用的密码。如果 [Setup] 段指示 Password 未设置，这条命令行参数被忽略。

当指定无效的密码时，这个命令行参数也被忽略。

6、安装退出代码

从 Inno Setup 3.0.3 开始，安装程序可以返回下列退出代码中的一个:

0 安装程序运行完成。

1 安装程序初始化失败。

2 用户在实际安装开始前单击向导页中的“取消”，或在“这将安装...”消息框中选择了“否”。

3 当准备移到下一个安装页 (例如，从显示的准备安装向导页到实际安装进程) 遇到一个致命错误。这应该不会发生，除非在意外的环境中，就象内存溢出或 Windows 资源溢出。

4　在实际安装进程中发生一个致命错误。

注意: 导致错误时会显示一个“中断、重试、忽略”框的不是致命错误。如果在这种消息框中选择“中断”，将返回退出代码 5。

5　在实际安装进程时单击“取消”，或在“中断、重试、忽略”消息框中选择“中断”。

6　安装进程被调试器强制终止 (运行 | 在 IDE 使用终止)。

在返回 1、3 或 4 退出代码前，将显示一个说明问题的错误消息。

以后的 Inno Setup 版本可能会返回另外的退出代码，应用程序检查退出代码的用意是能直观地处理意外的退出代码。任何非零值退出代码表示安装程序未完成。

7、卸载命令行参数

卸载程序 (unins???.exe) 接受可选的命令行参数。这些对于系统管理员是非常有用的，可以由其它程序调用卸载程序。

/SILENT, /VERYSILENT

当指定时，卸载程序将不询问用户启动确认或显示任何消息就开始卸载，直到完成。不再使用的共享的文件将不提示自动删除。但致命错误的消息仍将在屏幕中显示。当指定“/VERYSILENT”时，卸载进度窗口不显示。

如果需要重新启动，并且“/NORESTART”命令未使用 (看下面)，以及“/VERYSILENT”指定，卸载程序将不询问重新启动。

/SUPPRESSMSGBOXES

命令卸载程序禁止消息框。只在用“/SILENT”和“/VERYSILENT”编译时有效。

/LOG

使卸载程序在用户的临时目录创建一个记录安装程序卸载期间和 [UninstallRun] 详细动作的日志文件。这对于安装调试有帮助。

日志文件若要当前日期的唯一名字创建 (它不覆盖或添加到现有的文件中)，当前它不能自定义文件名。

日志文件中包含的信息是用自然语言，非常容易看懂，对于用户来说不会难以理解。也不用进行机器分析；文件的格式是不预先通知服从改变。

/NORESTART

告诉卸载程序不重新启动，即使需要这样做。

8、卸载退出代码

从 Inno Setup 4.0.8 开始，如果用户取消或遇到一个致命错误，卸载程序将返回一个非零值退出代码。程序检查退出代码以发现失败，而不是检查特殊的非零值；所有非零值退出代码说明卸载程序运行未完成。

请注意，从卸载程序获取到一个退出代码时，一些与卸载程序有关的代码可能仍在运行，因为 Windows 不允许程序删除它们自己的 EXE，卸载程序在临时目录创建并复制它自己的副本，由该“克隆”的程序执行实际卸载，结束后，终止原始的卸载程序 EXE (在这里你可以获取返回的退出代码)，并删除它，然后显示“卸载完成”消息框 (如果没有使用 /SILENT 或 /VERYSILENT 参数的话)。

9、不安全文件

对于不熟悉哪个文件应该或不应该发布的用户有用。如果尝试使用 [Files] 段安装某些“不安全”文件，Inno Setup 编译器将显示一个错误消息。下面是“不安全”文件列表。
(注意: 这可以在 [Files] 段条目中使用某些标记禁用错误消息，但不推荐。)
任何来自我自己的 Windows 系统目录的 DLL 文件

你不应该配置你的 Windows 系统目录中的 DLL，因为大多数只适合你自己指定的 Windows 版本，安装到其它版本后不会工作。如果你安装了一个来自其它 Windows 版本的 DLL，经常会导致你的系统不能正常启动。当你在电脑中安装了程序，DLL 可能被替换为其它不兼容的版本，并且你没有注意到以及采取措施，它在你构建新的安装程序时也会导致用户系统问题。所有不要这样做。
作为配置来自你的 Windows 系统目录中的 DLL 替代动作，你应该应该查明“redistributable”的版本。Redistributable DLL 一般能在多个 Windows 环境中工作。要查明 Visual Basic 和 Visual C++ run-time DLL 的 redistributable 版本，查阅 Setup 常见问题解答。

如果你有一个绝对有把握进行 redistributable 的 Windows 系统目录中的 DLL，将它复制到你的脚本来源目录并从那里配置它。
ADVAPI32.DLL, COMDLG32.DLL, GDI32.DLL, KERNEL32.DLL, RICHED32.DLL, SHELL32.DLL, USER32.DLL, UXTHEME.DLL

这些都是 Windows 的核心组件，必须不要用安装程序配置它。用户只能通过安装一个新版本的 Windows 或服务包或 Windows 修正程序来获取这些 DLL 的新版本。

(特殊场合) COMCAT.DLL, MSVBVM50.DLL, MSVBVM60.DLL, OLEAUT32.DLL, OLEPRO32.DLL, STDOLE2.TLB

如果 DestDir 设为不同于 {sys} 的位置，以及使用了 regserver 或 regtypelib 标记，那么上述文件将被考虑为“不安全”。这些文件不要在不是 {sys} 的目录中进行配置和注册，因为这样做有可能会导致系统中的所有程序不能使用在 {sys} 中的这些文件。如果你的文件副本老于 {sys} 中的文件，同样，如果你的文件副本被删除，其它应用程序将崩溃。

COMCAT.DLL 版本 5.0

版本 5.0 的 COMCAT.DLL 不能重新分配，因为它不能工作于 Windows 95 或 NT 4.0。如果你需要安装 COMCAT.DLL，用版本 4.71 替换。

参考: http://support.microsoft.com/support/kb/articles/Q201/3/64.ASP
COMCTL32.DLL

Microsoft 不允许个别地进行 COMCTL32.DLL 重新分配 (较好的理由 - 不同平台之间的文件不同)，因此，你不应该在脚本的 [Files] 段放置 COMCTL32.DLL。可是你可以指引用户从 Microsoft 下载 COMCTL32 升级版本，或将 COMCTL32 升级版本连同你的程序一起发布。
参考: http://www.microsoft.com/permission/copyrgt/cop-soft.htm#COM
参考: http://www.microsoft.com/downloads/details.aspx?FamilyID=cb2cf3a2-8025-4e8f-8511-9b476a8d35d2&DisplayLang=en
CTL3D32.DLL, Windows NT 专用版本

以前，在“安装 Visual Basic 5.0 和 6.0 应用程序”中指引页中的 zip 文件中包含一个

CTL3D32.DLL，在那时包含它，是因为这不知道它只与 Windows NT 兼容。现在你想安装专用版本的 CTL3D32.DLL，你必须使用 MinVersion 设置来限制它只安装到 Windows NT 平台。(总之，在 Windows 95/98/Me 中你不需要安装 CTL3D32.DLL，以后的所有版本均已有 3D 外观。)
SHDOCVW.DLL, SHLWAPI.DLL, URLMON.DLL, WININET.DLL
这些是 Internet Explorer 的核心组件，同时也用于 Windows 资源管理器。替换它们可能会导致上述软件的问题。如果你的应用程序基于这些 DLL，或它们的最新版本，那么你的用户需要安装最新版本的 Internet Explorer 获取它们。

10、感谢

下面是提供 Inno Setup 方案重要代码和其它特殊贡献的人员列表:
Jean-loup Gailly 和 Mark Adler: Inno Setup 使用的 zlib 压缩库作者。
Julian Seward: Inno Setup 使用的 bzlib 压缩库作者。
Igor Pavlov: Inno Setup 使用的 7-Zip LZMA 压缩库作者。
?: 磁盘延伸代码 (1.09) 的大多数。(抱歉，不知何故居然将你的名字忘记!)
Vince Valenti: [Setup] 段“Window”指示的多数代码 (1.12.4)。
Joe White: [Setup] 段“ChangesAssociations”指示的多数代码 (1.2.?)。
Jason Olsen: 添加到现有的卸载日志 多数代码 (1.3.0)。
Martijn Laan: Rich Edit 2.0 和 URL 侦测支持代码 (1.3.13)；后台卸载 (1.3.25)，驱动器和目录列表中系统映像列表支持 (1.3.25)；后台安装 (2.0.0)；[Types]，[Components] 和 [Tasks] 段 (2.0.0)； postinstall 标记 (2.0.0)；[Code] 段 (4.0.0)；子组件和子任务支持 (4.0.0)；各种其它 4.0.0+ 功能。
Alex Yackimoff: TNewCheckListBox (4.0.0) 部分。
Carlo Kok: Innerfuse Pascal 脚本 (4.0.0)。
SynEdit 作者: 用于编译器 (2.0.0) 的语法加亮编辑器。
glyFX: Inno Setup 标识，编译器图标和文档图标， Inno Setup 安装向导图标 IDE 工具栏图标。
如果我有任何遗漏，请不要不好意思告诉我。

11、与我联系

我写的最新版本的 Inno Setup 和其它软件可以在我的网站中下载:
http://www.jrsoftware.org/
要获取联系作者的更多信息和获取 Inno Setup 技术支持，转到这个页面:
http://www.jrsoftware.org/contact.php
获取最新版本的汉化信息，请访问汉化新世纪:
http://www.hanzify.org/